# Chapter 2 The History of Programmable Switching Systems

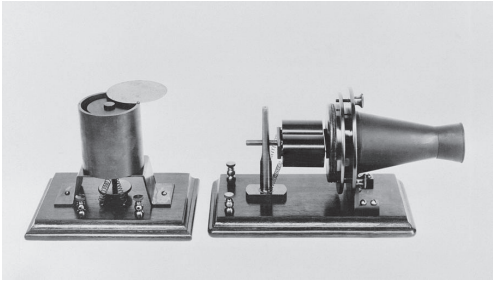
It's hard to say exactly what the first switching system was. It was probably an irrigation system of some type, which allowed the distribution of water to different locations using dikes being dug open or built shut. It would have only been considered programmable in the context of some person of authority ordering minions to close the dike by the fig tree and open the dike by the rock that looks like the Pharaoh.

Switching systems were basically used as flow devices such as this for a long time. Systems of gates were used to keep track of cattle in stockyards, and mechanical switches were used to redirect cars in train yards. But, once again, these were not really automated since they almost always relied on a person making a decision and taking an action or pushing a button. In order for a switching system to be considered truly programmable, it would have to be operated automatically by some non-human event or device.

One of the sad aspects of the switching system business is that the system is almost always designed to replace a human being, as any decent robotic device usually does. This would cause me to lose much sleep, if it wasn't for the fact that the switching system is usually taking a job that a human being has begged to be relieved from. Anyone who doesn't believe this should find a computer or audio device and unplug all the cables and then plug them all back in. Now repeat this procedure for two hours and imagine how you would feel about doing it for eight hours a day, five days a week, for endless months. A professor I once studied under argued that infinity was subjective by pointing out that if he asked you how far it was from Denver to New York City you would look this up on a map and give a reasonable estimate in miles. But if he gave you a six inch ruler and told you to measure how far it was, you would probably come back in less than 10 miles and say it was infinity. With that in mind, it isn't hard to imagine that after doing the job of plugging and unplugging cables for a couple of months you would probably describe the boredom level as infinite. There is the other argument, on top of the mind numbing tedium, which is that the switch eventually becomes a necessity due to the physical limitations of how fast people can perform such tasks and at what cost. This brings us to the first major application of switching systems.

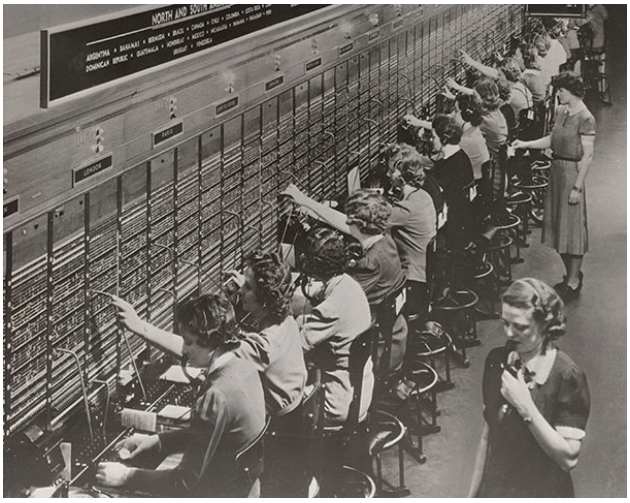
## Telephone switch boards

To gain an appreciation of the speed at which technological advances now move, you need only look at the early days of the telephone business. The first telephone like devices were invented in the mid 1870's. The first semi-automated telephone exchange switches were introduced in 1891. The first coast-to-coast US call was placed in 1915.



The true automation of telephone exchanges did not start until the 1920s and local human operators continued to fill this need in rural areas into the 1960s. The total number of telephone subscribers did not equal 50% of the population until 1971, which means it took nearly 100 years. Compare this to the internet which had its first commercial Internet Service Provider go on-line in 1989 and had reached 50% of the US population by 2001. By the time this book was started, it was difficult to imagine life without the internet. And now, more and more bandwidth is available at lower and lower prices. People walk around with hand-held phones that have faster internet access than was justifiably affordable to most businesses just 10 years earlier.

There were a variety of reasons it took so long for the entire country to get telephones: service was expensive and fairly monopolized, huge amounts of physical infrastructure (copper telephone lines) had to be installed and maintained, and the business of connecting anyone to anyone was done mainly by humans. When you picked up the phone and jiggled the hook up, a human being would answer and ask who you wanted to talk to. As the number of phones increased, people were given telephone numbers and area names so the operator could identify them out of the thousands of available telephones. When you called long distance, a series of operators would make the connection between the different local phone exchanges. This was all done through a series of switches and patch panels or cross bar switches that would be controlled by the "switch board operator". The work was considered menial and tedious and men of that day were unwilling to do it for the wages offered. Young boys were hired, but were easily distracted and customers complained that they were rude. The telephone company needed a source of polite, subservient employees that could be worked like dogs for low wages. Since this person had to actually communicate verbally and interface directly with important customers, they could not tap the normal source of low wage help, which was immigrants and minorities, and decided they would have to settle for women. The horrors of what this says about society in the early 1900s is a much more important and interesting social subject than the job itself, but that would be a subject for an entirely different book. Women excelled at the profession, and given their few choices in the job market at the time they were not necessarily unhappy to have the work. Only unmarried women between a certain age were hired, which was not an uncommon practice in the labor market then.



As the number of telephones increased, so did the number of operators needed. While it was easy for a single person to act as the operator in a fairly small town, the exchanges in large cities were fighting to keep up with demand. Some estimates put the number of operators working in New England alone in 1914 at 8000. Two things happened to change this: one was that the phone companies began to realize that they were quickly running out of the type of women for the job that fit their hiring criteria, and women began to realize the power they held and started to strike, demanding better conditions and pay.

This combination of growth and loss of control motivated phone companies to reconsider automation. As stated earlier, automated switches had actually been around since the 1890s, but they were complicated mechanical devices that had reliability issues. This technology also required that the person making the phone call have a more advanced telephone, which could send the signals necessary to tell the switch what to do. With advances in technology such as the "rotary dial" phone and better "cross bar" switches, the 1920s saw a huge increase in automation. The introduction of the crossbar switch, which was basically an XY grid of signals with relays able to make connections between the X and Y grids, greatly simplified the switch and increased reliability. The "telephone number" was used as a programming code for the switches, which would automatically direct local calls eliminating the need for operators.

The number of local operators decreased and by the late 1960s they were an extreme oddity. Operators continued to function as directory assistance on long distance calls and switch board operators ( receptionists) acted as the control point for directing calls at business numbers. But even these positions are rapidly being replaced by automated systems, voice recognition software and cellular service -- whether this is good or bad is left to the reader. While some would argue that these advances have decreased cost and enhanced service, there was something reassuring about a fellow human (if reasonably intelligent) answering your call and connecting you to the person you needed to talk to. This element continues to be extremely difficult to automate.

The growth of communication systems increased at a phenomenal rate. Other methods of communication such as radio and satellite competed in military, space, and private communications networks for businesses or municipalities. And switches were important for all of these. When it was realized that digital networks could be interconnected to rapidly share data instead of just voice ( a nod to Paul Baran, who thought up packet switching and just passed away as I write this) the obscure network switch became another household item and business necessity.

## Automating Traffic Flow


The other major use for switching systems that started in the mid 1800's was directing the flow of traffic. The growth of trains exploded after the transcontinental rail was finished. Every town had a train station and freight yards and the only way to let trains share tracks was by using switches to route them on and off the main lines and shuffle them through stations and freight yards. The early systems all depended on humans to manually change the switches into the right positions as needed. But as rail yards increased in size and complexity, manual switches were replaced by electric switches controlled by a single person or group of people at a control center. As large people moving systems such as subways sprang up, much of this switching became automated with manual overrides in the event of emergencies. One of the interesting things about switching trains is that due to their linear nature and large physical size and the need to have many trains sharing a small number of main lines, there were a number of interesting methods devised for switching them between tracks. Tree switch topologies were used at busy freight yards and stations and rotary switches were used at repair facilities and engine houses. Many of these switch configurations would have applications in the world of electronics, which will be addressed later in this book.

The growth of automobiles in the early 1900's saw another need for automated switching. When cars first appeared, they were an oddity that scared horses and children. They were considered so frightening and dangerous, that some towns made a person walk ahead of them with a flag to warn pedestrian and equestrian traffic of their approach. But the convenience of these robotic horses quickly became apparent. If you don't believe this, buy a real horse. No matter how romantic your notions of riding a horse may be, owning one is work. If you had to feed and water your car three times a day while it crapped all over your garage floor and ran off whenever you left the door open, you would quickly gain an appreciation for the "horseless carriage". As cars become faster, more powerful and larger, and traffic increased in urban areas, the real dangers became apparent necessitating flow control at intersections. Traffic lights were introduced as early as 1912 for automobile traffic. The early systems were manually controlled units, which had previously been used for trolley and horse traffic. The first automatic systems were introduced in the 1920s and typically relied on timers. Later the timers would be enhanced by devices that detected vehicles, which could over-ride the timers.

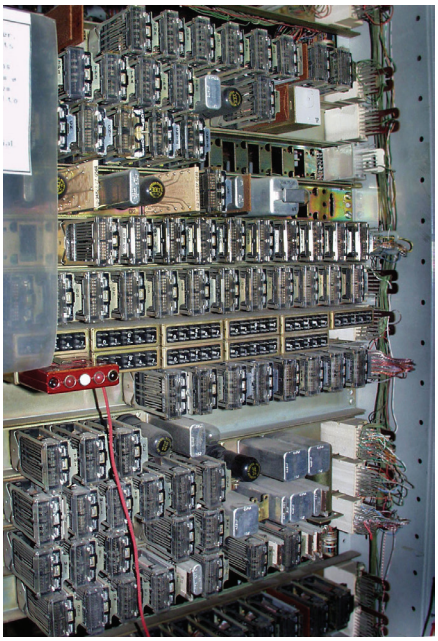
Many cities now have systems in place that can automatically regulate the timing of lights to compensate for traffic flow problems. What the traffic signals cannot compensate for is the human element of driving a car however the hell you feel like. It is a well known fact that traffic flow through cities could be greatly enhanced if you removed the human control of the car and let certain controls over speed and acceleration between lights be better regulated. There would be enormous benefits to this as the car became just another method of public transportation -- fuel savings would be enormous, safety could be greatly improved, and people could retain some of the privacy and independence they still cherish in their cars. In all likelihood, we will eventually have systems in place that are interconnected to all the other cars on the road and the entire driving process will become semi-automatic, thus, freeing us up to text, work, put on our make-up and eat breakfast during our morning commute. In other words, it will be just like it is now, but we won't kill each other doing it. The human controlled car will be relegated to the scrap heap like washboards, phone booths and horse drawn carriages. It will become a museum curiosity, recreational oddity or ritually religious holdover everywhere except in the third world. Future generations will wonder in amazement at how we were willing to sacrifice 40,000 lives per year for the "benefit" of driving prior to the application of these safety measures. And I, as someone who loves to drive, can only hope I'll be dead by then. Probably due to a car accident.

While traffic flow is an obvious application surrounding us on a daily basis, there are hundreds of less obvious applications in use. Most switching systems used in flow control have a fairly simplistic control device -- the timer. Production lines will used timed switching systems to move assemblies along to the next station and deliver parts to work areas where they are needed. Irrigation systems will use simplistic timers and switches to water lawns and crops. There are countless industrial controls and switching systems being used in manufacturing plants across the planet to control and automate the production of everything from food to airplanes. As technology has progressed, the number of humans required to work assembly lines has decreased, but the number of humans required to create and maintain the automation has increased. This is not usually a bad thing in the long run despite the temporary disruptions to jobs.

The future will see endless new applications for automated switching technology and robotics. Everything from robotic distribution warehouses and automated pharmacies that will dispense your prescription without human assistance, to automated cow milking stations. As the automation trend increases, there will no longer be a need to have large manufacturing plants located in the areas of the cheapest labor, and things will simply be manufactured by robotics and 3D printers on a much more local and "on demand" basis.

## Automated test and data acquisition

As the proliferation of electronics exploded in the 1960s, so did the need to test these devices. In order to ensure quality control and safety, every electronic device had to have at least some functional test before being shipped to consumers. Early automated test systems often relied on exotic mechanical timing systems, clunky relays, and usually still required a lot of human intervention.

The two things that changed this drastically were reed relays and the microprocessor.

Reed relays were small, inexpensive to build, capable of switching a large variety of electrical signals, and had life expectancies typically 100 to 1000 times greater than clunky armature relays. These cheaper, more reliable switching systems were much easier to justify on a tight test equipment budget and proved their worth quickly by greatly decreasing the cost of labor, and also making much better use of very expensive test equipment. Rather than buy 25 signal generators and decade boxes, you could simply buy one set and hook up a switch that would automatically switch between devices and test them while the operators changed devices in fixtures, recorded data and sorted pass from fail. Having automated testing often meant that engineers could run long-term tests overnight or in the background, so they could spend their time more wisely.

Microprocessors provided the ability to program the switches for automated operation. With the push of a button, a complete test routine that took 20 minutes could now be run in 20 seconds. Engineers began to learn programming and established ways for test instruments and equipment to communicate with each other to automate data acquisition and record keeping.

The explosion in use of microprocessors and ICs generated an entirely new field of automated test applications, since all of these devices needed to be tested and were produced at a rate that would be impossible for humans to keep up with. Devices such as “Bed of Nails” test beds were developed and some required massive amounts of switching .

## Data Acquisition

As the fields of scientific research advanced, data acquisition became an important step in proving concepts in the lab and gathering data in the field. This saved both time, money and human lives. We could now simulate real world situations in a monitored environment and make predictions that improved almost every product, process or experiment. Whether you're measuring stress, monitoring temperature and vibration, developing medical equipment and cell phones, or building space ships and submarines, nearly everything is subjected to data acquisition before it becomes a real product. Every great idea begins with the accumulation of experimental data.

In the early days, data was usually recorded on to paper using a variety of electrical or mechanical devices that simply plotted recorded data to a rotating or rolling piece of paper. The movement of the paper was accurately timed and the paper was usually printed to reflect the timing so the data could be tracked. This method worked great for measuring relatively slow events over relatively long time periods. This technique is still used for things such as seismographs and polygraphs, but even those are now recorded into digital formats.


**Volt-Ohm-Amp Meter with built in data recorder**

Paper was replaced by "Magnetic Recorders" that used tape to record the information, which could then be fed into computers. While most people think of reel-to-reel and cassette tapes as audio sources, they were widely used as data recorders, often before becoming affordable enough for the average home audio user.
For many data acquisition applications, there will be a very large number of sensors or transducers, however they do not all need to be monitored constantly. Switching systems greatly simplify the process by allowing a small number of data recorders, instruments and sources to scan through the different test points over time, accumulating a large sample of data from many test points in a much shorter period of time. As data recorders improved, it allowed much more accurate collection of data pertaining to much faster events.

With the advance of computers came floppy drives, hard drives and solid state memory with almost unlimited storage capacities. When you consider how much data we can cheaply store compared to only 20 or 30 years ago, it is utterly ridiculous.

The 8" floppy shown here stored approximately 256 Kb of data which was the equivalent of 2000 IBM punch cards. A new 8" floppy drive cost around $700.00 in 1980. The 2 Tb USB hard drive sitting on the floppies costs $90.00 in 2015. So the cost of storing data has gone from $2,800.00 per megabit to only .005 cents per megabit in 35 years.

Dryden Flight Research Center ECN-28683 Photographed 1984
Strain-gage instrumentation installed on the interior
of the AFTI/F-16 wing. (NASA photo)

The state of modern data acquisition is amazing. When a new missile or fighter jet is test flown, thousands of sensors record everything that is happening on board a million times per second and broadcast it to receiving stations.

As the test flight moves across the sky, automated software detects the level of the transmitted data received and automatically switches the information gathering from one receiving station to the next. All of this information is sent through fiber optics and gathered for analysis in enormous data banks to be studied later.

From a scientific point of view, data acquisition switching system projects tend to be the most interesting. While automated test applications can be complicated and play an important roll in manufacturing and product development, the data acquisition projects tend to be more fun, since they often involve doing things that have never been done before.

Some of the more interesting data acquisition projects I've been involved in:

- Monitoring a glider built to fly on Mars by testing it in a wind tunnel built inside an evacuation chamber. Because of the difficulty of building a wind tunnel inside an evacuation chamber, the switching system had to operate inside the chamber in an atmosphere equivalent to 125 thousand feet on earth. Aside from the power supply getting warmer than usual due to a complete lack of ambient cooling, the system operated fine.

- Teasing bats in a geodesic dome with audio transducers in order to investigate their ability to see objects through solid walls using only audio signals. The uses for this kind of technology would be endless.

- Developing new ultrasound and imaging techniques to better detect breast cancer without painful and inaccurate mammograms.

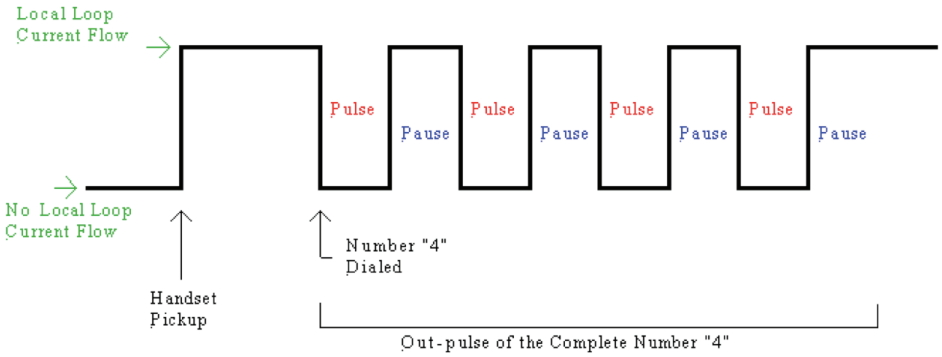
- Monitoring large paper-mill equipment to predict bearing failures and eliminating unscheduled down time.

- Research covering a variety of wireless charging techniques.

As the speed and quantity of data being acquired increases, so do the challenges of switching systems needed to help gather it all. Using FPGA controllers and ultra fast solid state switches, we are able to keep up with these challenges and look forward to new ones.

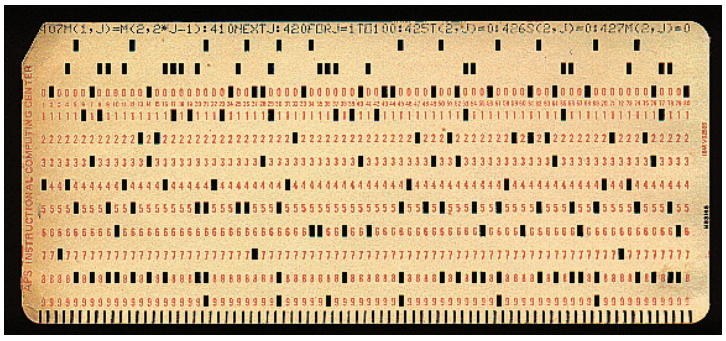
# Switching Control

In the early days of switching systems, the programmatic control was based on the same system as the rotary dial telephone. The system would receive electronic pulses that were translated into a relay location and that relay was told to open or close, which is pretty much exactly what happens today. Nothing has changed as far as what happens inside the switching system. What has changed is how the system receives those pulses.


With the invention of transistors and Integrated Circuits (ICs) the control of systems became primarily parallel TTL logic circuits. Computers would be programmed using punch cards, which could run the same switching routine endlessly at the push of a button. Punch-cards gave way to tapes, tapes gave way to floppy drives, and the "PC" became so mainstream that proprietary TTL interfaces were no longer considered convenient.

As the fields of automated test and data acquisition began to explode in size and money, what was needed was a communication and control method that could be used with all equipment from different vendors without having to learn a different TTL interface for each device.



## RS232 Serial Data Control

RS232 was developed in the late 50s and early 60s as a means to communicate to teletype machines. It was introduced as an EIA standard in 1962 and revision C in 1969 introduced the commonly accepted form of the bus. RS232 became very popular because it could operate over long distances over existing telephone wires and operated at what were considered very high speeds at the time. Improvements to electronic components lead to increased baud rates and because it was invented to work over phone lines and modems, it eventually became the default for the first internet connections. In March of 1968, ASCII was officially designated as the means for computers to store and communicate the normal English alphabet and keyboard input. The combination of these things meant you could now communicate between computing devices using text instead of binary code, greatly simplifying human-machine interactions.

DTE Device (Computer) **DB25** — **DTE to DCE Connections** — DCE Device (Modem) **DB25**

| Pin# | DB25 RS-232 Signal Names | | Signal Direction | Pin# | DB25 RS-232 Signal Names | |
|---|---|---|---|---|---|---|
| #1 | Shield to Frame Ground | FGND | ⟷ | #1 | Shield to Frame Ground | FGND |
| #2 | Transmit Data (Tx) | TD | → | #2 | Transmit Data (Tx) | TD |
| #3 | Receive Data (Rx) | RD | ← | #3 | Receive Data (Rx) | RD |
| #4 | Request to Send | RTS | → | #4 | Request to Send | RTS |
| #5 | Clear to Send | CTS | ← | #5 | Clear to Send | CTS |
| #6 | Data Set Ready | DSR | ← | #6 | Data Set Ready | DSR |
| #7 | Signal Ground/Common (SG) | GND | ⟷ | #7 | Signal Ground/Common (SG) | GND |
| #8 | Carrier Detector (DCD) | CD | ← | #8 | Carrier Detector (DCD) | CD |
| #20 | Data Terminal Ready | DTR | → | #20 | Data Terminal Ready | DTR |
| #22 | Ring Indicator | RI | ← | #22 | Ring Indicator | RI |

By the early 80s, the RS232 port was more commonly referred to as simply a "serial" or "COM" port and was found on most computers and peripherals. Terminal emulation software allowed you to make a connection to any device and simply type commands to communicate.

RS232 was cheap, standardized to a high degree and worked. At one point, almost every electronic device with a microprocessor that had to interface with a human came with a COM port. But it was not without drawbacks.

Communicating via RS232 required an understanding of the bus that would never fly today in the world of "Plug and Play". You had to understand the difference between a DTE and DCE device. There were numerous settings for things like Baud rate, Start and Stop bits, Parity, and Handshaking modes. To complicate things further, many devices came with hardware jumpers allowing you to change things like DTE/DCE and Handshaking modes. Different computers and operating systems all had different "Default" settings.

Some devices worked simply by connecting the TX and RX lines, while others required every pin on the connector. The early devices all had D25 connectors, but these were replaced in the 90's by D9 connectors which created entire industry just to make adapters. Some people used male connectors for DTE devices and Female connectors for DCE. Other companies went the other way, leading to numerous adapters and gender changing devices. Some vendors didn't even comply with the standard pin-outs or voltage levels opting instead to provide adapter cables or declare that the device was "RS232 like".


D25 Female to
D9 Female Adapter.
Pray that the "Null Modem"
sticker never falls off!

Added to this was the invention of "Null Modem" cables and adapters. These devices were a simplistic way of connecting a DCE device to another DCE device, or two DTE devices directly, without having to change jumpers or hardware settings. They basically just criss-crossed the wires within the cable or adapter. Often these things were not permanently marked and the only way you knew what it was (once you threw out the bag it came in) was to verify the pin-out with an ohmmeter.

At one point in the late 90's, I probably spent half of my tech support time walking people through RS232 connection problems. We had a list of about 20 questions based on the most common issues and each question would sometimes require 15 minutes of explanation. With the explosion of Modem use in the 90's for internet access, the issues became worse, leading to the real problem with RS232.

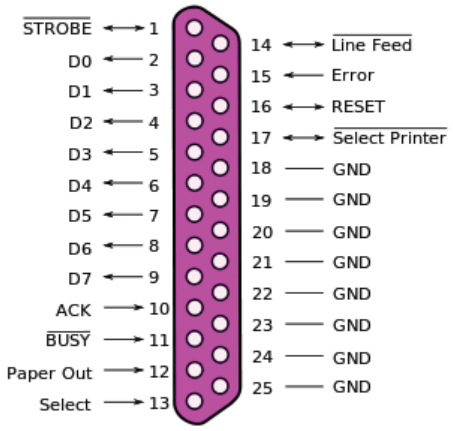
RS232's biggest drawback was that it was a 1-to-1 connection. That meant you had to have a separate COM port for every device you wanted to talk to. As more and more things like printers and modems came into common use, the computer manufactures would add more com ports, but typically they were limited to three or four from the factory. Many automatic test systems had at least four devices and often they had up to 10 or 12. Special computer accessory boards could be purchased that could give you anywhere from 4 to 24 extra com ports, but dealing with this number of interrupt driven COM ports was difficult, to say the least, especially when you consider the complexities listed above. In reality, the RS232 bus was usually used in situations when there was a limited number of devices.

Despite the drawbacks and difficulties of RS232, COM ports were still being provided on new computers well into the first decade of the 2000s. Fully functional USB ports were introduced around 1998 but it took ten years for manufactures to finally stop providing computers with COM ports ( see section on USB ). The only reason they were finally able to do that, was the availability of USB to COM port adapters. RS232 was so completely entrenched in the computer industry that it is still requested as a common control option on automated test equipment, despite the lack of COM ports on computers.

## Parallel Printer Port Control

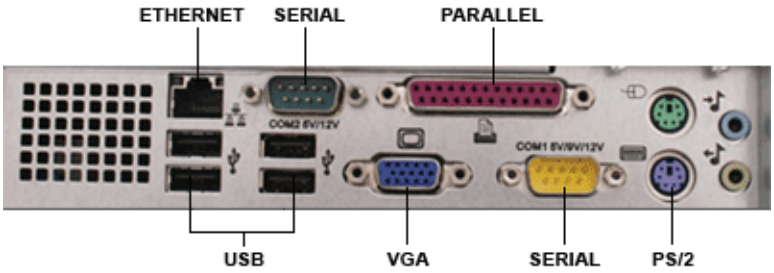
One of the common parallel TTL control options that briefly became popular was the Parallel Printer Port, or LPT ( Line Printer ) port. It was used on early Centronic printers and became so popular it was eventually accepted as the standard 1284 by the IEEE.

All of the early IBM PCs in the 1980s came with one of these ports, which made it handy to use as a parallel TTL bus for controlling other devices. Since the bus was designed to transmit words to a printer, it was easy to use as a communication bus for other devices that understood ASCII test strings. Many of the early programming languages incorporated commands for the parallel printer port. You could send a command to your device and then pulse the strobe bit to make the device run the command. Similar to RS232, it was a 1-to-1 bus, and since computers usually only had one parallel port, it was very limited in applications. It was also fairly slow ( 150 Kbps ) and the cables were bulky and expensive.




The LPT port died off pretty quickly for anything other than printers, and by the 90's it was increasingly being replaced with RS232. By the 2000's it disappeared completely as USB and Ethernet greatly increased the speed and ease of communication to modern printers.

Modern laptop computers typically come very uncluttered, with wireless ethernet, and a few USB ports, but it wasn't long ago that nearly all computers came with a large variety and mixture of ports, in an attempt to support both new and legacy equipment. I'm looking forward to the day when everything is just able to talk to everything else wirelessly on a common standard such as ethernet. I have boxes of cables that I'll probably never use again, but can't bring myself to throw out, because of a need to support some old product that might show up one day to be repaired.




Typical computer rear ports, circa 2002

# HPIB - GPIB - IEEE488


The original HPIB ( Hewlett Packard Interface Bus ) was developed in the late 60s. Hewlett Packard was the leader in the test equipment business and recognized the need for a better communication bus that could talk to multiple pieces of test equipment at a high rate of speed.

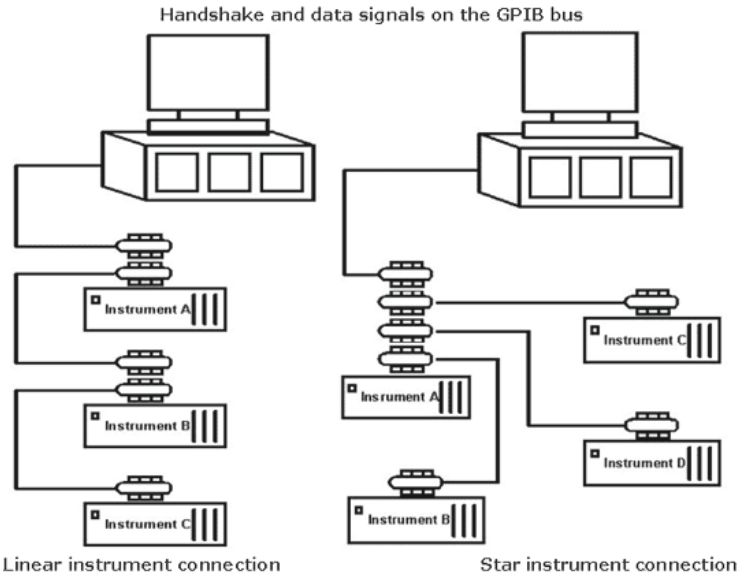
The original bus was a simplistic 8 bit parallel data bus, but it could be daisy-chained across multiple pieces of equipment using "stackable cables". This meant the beginning of one cable could be stacked onto the end of the last cable and you could do this up to 16 times with a maximum total cable length of 20 meters. Each device on the bus had a 5 bit address that you set with jumpers or dip switches, which allowed you to talk to each device as a separate "device #". The bus would run at speeds up to 1 Mbps.


The bus worked so well that HP began licensing it to other equipment vendors. At one point it was a standard port on the popular Commodore PET computers.

As other vendors began to license and use it, the bus name was more commonly referred to as GPIB (General Purpose Interface Bus) and was eventually standardized as the IEEE488 Bus in 1975. It was revised quite frequently for standardization purposes and eventually common programming commands known as SCPI (Standard Commands for Programmable Instruments ) were adopted to standardize control.

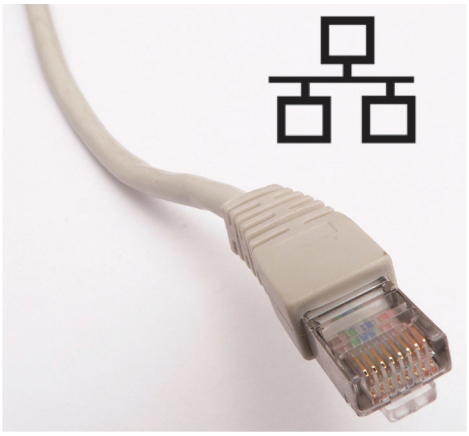
The GPIB bus basically allowed each device to operate as a "Talker" or a "Listener" dependent on whether it was sending data or receiving data.

In the early days, the "talker" was the computer port and many of the devices on the bus were "Listen Only" devices, which would simply wait for commands and perform them as instructed without much feedback to the computer. There was an SRQ bit that could be set to 1 or 0 to give simplistic status feedback. This allowed these devices to be operated very quickly and a "Listen only" GPIB port was one of the fastest ways to control an instrument well into the late 90's.




During the 80s, as the need and desire to get data back from the devices on the bus became increasingly important, and as computer speeds increased to allow processing this information, most GPIB devices became "Talk / Listen" devices and began using ASCII text commands instead of binary control bits. While the GPIB connector was rarely seen as a standard connector on new PCs after the 80s, the bus was so entrenched within the automated test and instrument industry that it was standard practice to just order a GPIB controller board for almost every computer being used in any department that had anything to do with engineering. The introduction of high level software in the 90s made controlling the instruments even easier and as of 2015, the GPIB bus is still one of the more popular ways to control instrumentation.

## Ethernet LAN control



In the 1980s, MSDOS and CPM were the dominate operating systems for IBM and clone PC's, everything was done on floppy drives, and Netware by Novell was the dominate network operating system using the IPX/SPX standard to communicate over network cables. Competing LAN technologies such as token ring, 10Base5 and 10Base2 ethernet connected each computer to a central server through coax cables. Novell's Netware dominated nearly 90% of the office network server market and the software was able to communicate over various network hardware.

By the early 90s, 10BaseT four wire ethernet had taken over as the preferred physical layer network standard and Microsoft had set their sights on attacking the server/network market. At the time, this seemed ridiculous as Microsoft's new "Windows" OS seemed too unstable for serious network server consideration and Netware was so entrenched that they didn't seem to take the challenge seriously.

Around 1995, Novell was preparing to introduce its latest platform and had to make a very serious decision. Should they stick with their IPX / SPX network protocol? Or consider changing over to the TCP/IP protocol which Microsoft was making use of in their NT server software? Previously, the TCP/IP protocol had only been used for communicating between devices on the "Internet" which was an obscure network developed for military and university use. Novell decided to stick with IPX/SPX.

I have no idea what motivated Novell to make that decision. Maybe they didn't want to look like they were bowing to Microsoft in any way? Maybe they looked at the internet at the time and decided it was an interesting way to distribute research and pornography, but had no real commercial use? Maybe the challenge of making the switch to a different protocol was just too daunting?

Or maybe it was just arrogance. I used to visit Network Interop shows back then and Novell was the 800 lb gorilla. They had the biggest booth and the flashiest presentations. When everyone else was giving out pens and cup holders, Novell was giving away nice golf umbrellas and inviting everyone to open bar buffets. I think they found it inconceivable that they were on the cusp of extinction.

In 1994, you could log onto bulletin boards with a modem and download a fairly small text file that listed all of the available "World Wide Web" domains. It was the first attempt at listing web sites that the public could access and would become known as the YAHOO list, and then yahoo.com -- the first web directory. By 1998, any business not making use of the web was in serious danger of being left behind. And for Novell, it was too late.

The combination of full TCP/IP support and a familiar GUI allowed Microsoft to take over networking in a matter of years, leaving many of us to look back at our last stack of 150 Netware Installation floppies as an interesting old person conversation piece. "Yes!" We would exclaim to our children as they rolled their eyes, "You had to load all 150 disks one by one and if it crashed on disk 145 you had to start all over again!".

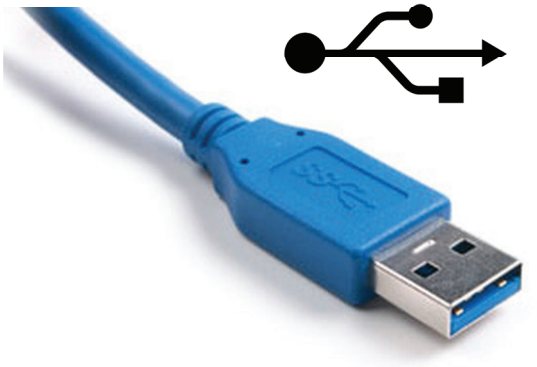

Big box of 162, 3.5" Netware 3.0 disks waiting for full day of installation.

Around the time that 10BaseT Ethernet became the industry standard, it became obvious that this format could be very useful in the automated test world. Every computer came with an Ethernet port and most buildings were being wired for the network. The number of ports available were limited only by the number of IP addresses, which were infinite for all practical purposes and you could add 24 or 48 ports to any location simply by buying another, relatively cheap, hub. With no real length limitation, you could access your test equipment from the room, your office, or pretty much anywhere in the world with the right network connection.

To make matters easier, devices were developed that allowed you to connect legacy RS232 serial ports to the network by converting the TCP/IP protocol to RS232. One of my customers started a very successful company based on setting up network monitoring equipment around large companies and LAN controlled physical layer switches, which gave network professionals the ability to monitor, troubleshoot and diagnose network issues anywhere in the building from their office or home, saving the company hundreds of thousands of dollars in equipment, man hours and downtime.

As the networks have become faster and cheaper, and as everyone has become more familiar with the technology, it has only escalated the desire to do everything from a network connection. You can now buy fully functional network computers that are only two square inches and cost less than $50.00, complete with cables. Products like Raspberry Pi, Arduino and Rabbit Core have given people the ability to connect anything to the internet. I can now carry a demo switching system unit around with a cheap wireless router and control my equipment at any site with my cell phone. At the rate it is going in 2016, I would assume that most of the other control interfaces will disappear completely within 10 years except for legacy, highly secured and specialty situations. I for one will not be sorry to see most of them go.

### USB -- The Universal Serial Bus



I suppose I'll start this section by noting that I'm not a fan of the USB as a test equipment or instrumentation bus. It's a wonderful peripheral bus. It just lacks the qualities highly prized by the superficial and lazy test engineer.

As the use of home computers exploded with the rise of the internet, the problems of connecting peripherals became obvious. You had a different connector and cable for your mouse, keyboard, printer, scanner, camera, PDA, and any other device that you wanted to plug in. If the device didn't come with the computer, it often had an oddball cable. You had to load special drivers that were often dependent on the type of computer and OS you had. All of this lead to entire departments that did nothing but walk customers through connecting devices to a computer, installing the drivers and getting it up and running.

What the industry needed was a "universal" connection that would automatically recognize the device and install the needed drivers and software that would be distributed with the OS. So, a large group of computer related companies came together and developed the "USB". Note, that I'm completely ignoring Apple users who, by now, are screaming: "Yeah, we already had that!". I've ignored Apple throughout this book for a reason. I have nothing against Apple, which has made some great hardware over the years. Microsoft basically copied most of what they did, and Steve Jobs was without doubt one of the greatest marketing geniuses that ever lived, but the problem with Apple was always the legal department. By making everything they did proprietary, they locked out developers and engineers who all flocked toward DOS and IBM PC/AT/XT clones where they could develop hardware and software in peace. This resulted in hundreds of specialty engineering apps being available on cheaper and easier-to-modify computers, effectively relegating Apples to marketing departments.

**USB continued....**

I'm not one of those people who bash any other computer or operating system being used by someone else. I find the whole Microsoft vs Apple vs Linux arguments to be stupid. Computers are simply tools that allow us to be more productive. Arguing over competing computer platforms is like arguing over competing hammer brands -- as long as it allows you to get the job done and you know how to use it, I don't care what kind of hammer you prefer. But I've strayed off the topic of USB.

The USB was developed in the late 90's and was going to provide everyone with a way to enjoy "Plug and Play" peripherals. The hardware was available long before Microsoft finally rolled it into an OS release. Many people, including myself, not only anxiously awaited it's adoption, but actually considered using it as a communication bus for our equipment. The USB roll out was bumpy to say the least, and Microsoft's first big public presentation resulted in an embarrassing blue screen of death. This led to early versions jokingly being referred to as "Plug and Pray."

However, it quickly found success and, before long, delivered what it promised. You could buy peripherals, plug them into any available USB port, and it would recognize the device, install the driver, and away you went. It quickly replaced RS232 COM ports as the way to talk to printers, scanners, modems and everything else you plugged into the PC. It was inexpensive and could provide enough power to a device to eliminate the need for powercords. You could buy small USB hubs and easily attach 20 devices to the 3 or 4 USB ports that came with the computer. So, why don't I like it as a communication bus for automated test?

- *Length limitations.* USB was basically invented to connect devices that were normally right next to your computer. They never really envisioned people using it to connect 19" racks filled with equipment. So, a single USB cable could only be 5 meters max and 3 meters to more realistically operate at various speeds. You could add hubs down the chain, but that in itself became tedious, since many hubs limited power and speed once again. Compare this to GPIB with 20 meter cable lengths, RS232 with possible lengths of hundreds of feet, or Ethernet with single cable 100 meter limitation and unlimited length with hubs.

- *No standard communication software.* Unlike RS232 or Ethernet, there is no standard software that just lets you talk to your device from a command line or terminal. Having a problem with your RS232 or LAN device? Fire up Hyperterm, puTTy, Telnet, or any other standard com programs and see whats going on. Having a problem with a USB device? Hope you have software from the vendor because there is no standard communication language for USB devices. So, now you don't know if it's the hardware or software or firmware that's giving you the problem, and there are no great ways to figure it out.

- *Plug and Play.* This was a wonderful thing for companies making thousands or millions of peripheral devices like keyboards, mice and printers. But, in order to be Plug and Play, you had to join the USB Alliance, write your compliant drivers, and submit them for inclusion with the OS distribution. Now you will need to write some software that allows the end user to interface directly to the device, since it isn't a normal device like a keyboard, webcam, or thumbdrive supported by default drivers. The annual cost of doing that is no big deal when spread over thousands of products every year. But, if you're in an obscure or niche market, and only 30 customers per year are requesting USB, the added overhead and support can be a real pain.

So, USB has mainly been regulated to small systems or single device test equipment applications usually targeting the lower price end of the market. That may change in the future as better versions of USB come out and the standards adapt to other uses. But as of 2016, there simply are a lot better choices.

## The Future of Control

Currently the future points to TCP/IP over Ethernet LAN becoming the standard. IEEE488 is seeing its last days and to my knowledge, there is only one company left making the communication ICs. When they decide to stop, you won't see anything but legacy support. USB works fine for talking to legacy equipment with an RS232 port through an adapter cable, but the knowledge required seems to be rapidly disappearing. And RS232 is already being forgotten by the last generation which dealt with it on a regular basis, and is not being considered by the next generation.

Who knows what is around the corner? Maybe a secure, real-time wireless standard, fine tuned for Instrumentation and Automated test? A better version of USB? It's hard to say. Current trends point to an eventual switch to some form of wireless.

I for one will not attempt to be the oracle on the subject. In 32 years of dealing with this kind of equipment, I've seen a big idea every couple of years that claimed it was going to revolutionize the industry. Some with great success, some forgotten before they reached production.

All of this brings me to the subject of stand alone bus technologies. The history of switching systems and the entire industry of automated test, data acquisition and communications would not be complete without it.

"I've seen things you people wouldn't believe. 32 bit standard bus modules, FDDI controllers, SCSI bus interfaces, ZIP drives, and wireless bluetooth instrumentation specs. All those technologies will be lost in time, like tears...in...rain. Time to die..."

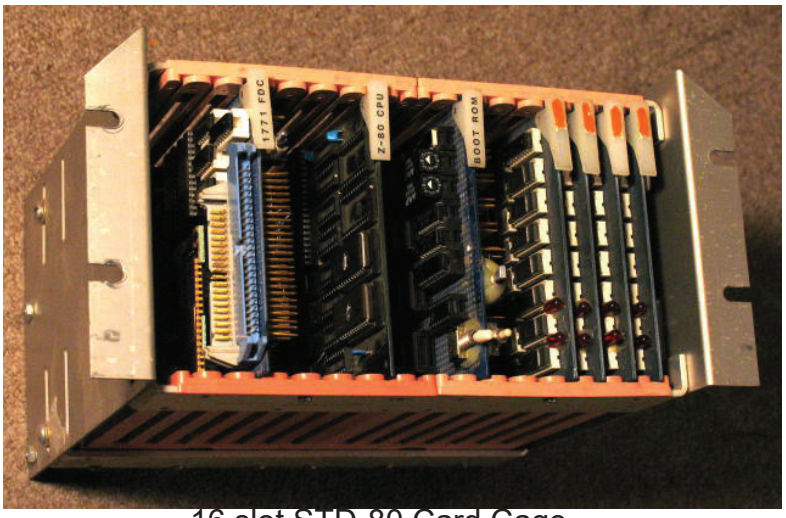
## Stand Alone Bus Technologies

The idea behind all stand alone bus technologies is a good one. The desktop, laptop, or even the modern tablet computers are designed the way they are because they interact with a human. For many automated test, data acquisition and communication applications, the human interaction with the system is minimal. Like any good robot, they are basically designed to do a certain task without much human interaction unless there is a problem. So naturally, at some point, someone asked why we were building these systems out of complicated rack mount boxes with separate power supplies, separate and different interface connections, and a nightmare of different software layers and cables. What if we just designed plug in modules that all spoke the same language and plugged into a single control bus with a well defined control protocol?

### The Standard Bus or STD Bus or STD-80 Bus ( possibly the "Simple To Design" bus? )

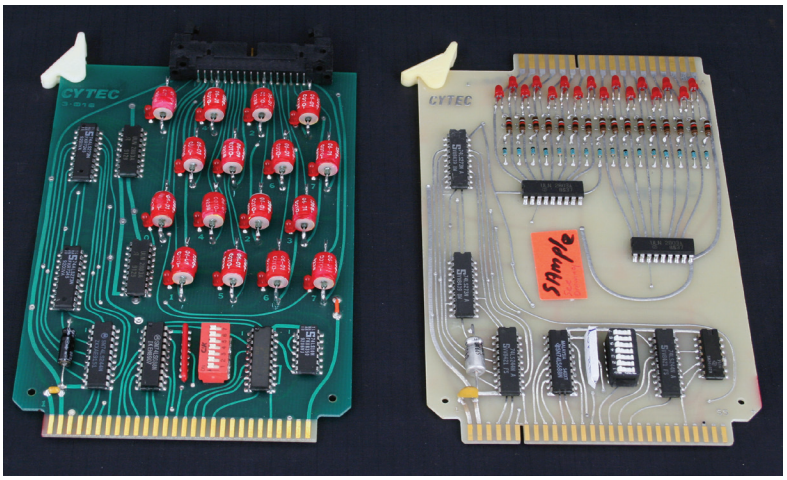
The original 8-bit STD bus was created by Prolog. It used a Zilog Z80 processor on the main control board, so it was often referred to as the STD-80. It was upgraded to a 16-bit bus and then Ziatech upgraded it even further to the STD-32 when it moved up to a 32-bit bus in 1989. Ziatech would eventually create a new bus technology, which we will get to later.

The STD Bus was basically a card edge motherboard with screw terminals at each end for power supply connections. Sometimes they were mounted in a card cage as shown, and sometimes they would just sit on a bench. You would install a computer board into one of the card slots, and put whatever you needed in the other slots. The bus gained traction as an industrial control platform, but in may ways it was the Raspberry Pi or Arduino of its day. A lot of amateur electronic enthusiasts thought up creative things to do with the available boards, or bought prototyping boards and made their own circuits.



16 slot STD-80 Card Cage

We sold a lot of relay modules and relay or solenoid driver modules for this bus. They were fairly simple and easy to maintain and didn't require complicated programming to control. One of the industrial control applications was for a system that was installed in robotic underwater submarines used to repair oil drilling platforms. These submarines were in use for many years after the STD bus was no longer popular and we seriviced the boards until the last of those subs were taken off-line.



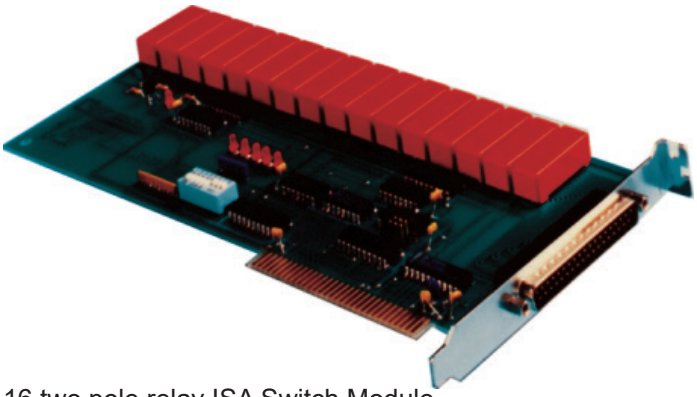
A 16x1 relay module and 16 channel driver module with LED indicators.

Ziatech tried to keep the bus alive by going to the 32-bit version, but by then the AT/XT ISA bus in the PC had taken over much of the market.

When computers moved to the newer PCI bus, Ziatech resurrected the idea of a stand alone bus with Compact PCI. Soon after, they were bought by Intel.

## Desktop Computer Buses, The ISA and PCI Bus

Almost as soon as the IBM PC and all the clones hit the market, we began getting requests for modular cards that could be inserted into the PC/XT peripheral bus. These cards were simple to make and relatively easy to talk to. When the ISA bus was introduced, it was mostly backwards compatible ( it truly was for our products ), so the same modules could be used for many generations of computers. Later, when the computers went to the EISA bus and then the VESA bus, they still retained a section that would accept the old ISA peripherals. So, the same bus modules were basically in use from the introduction of the IBM PC in 1981, all the way up to the late 90s when the PCI bus finally took over completely (at least on mainstream PC's).



16 two pole relay ISA Switch Module

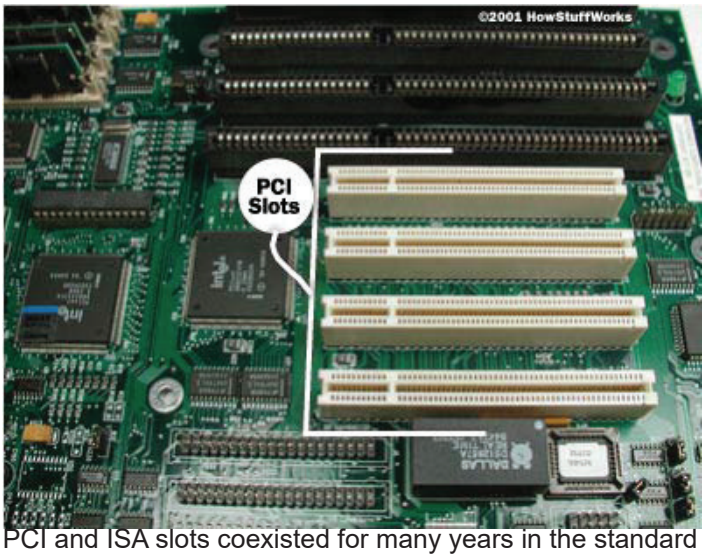
These little modules were a handy addition to the list of PC peripherals. You weren't going to build a test system around them, but if all you needed was a small multiplexer for a specific test, then these fit the bill and were pretty cheap.

Most of the programming languages at the time included functions such as "inport" and "outport", which simply let you poke bits on the bus to control things.

## Conventional PCI Bus Modules

Beginning in the 90's, the ISA bus and its variants were replaced with the PCI bus (Peripheral Component Interconnect bus). It was much faster and could be up to 64-bits. Plug and play standards were enacted, so that after you installed a peripheral and rebooted your computer, it could find and install the drivers for you.

Once again, a small market opened up for switching products as add-on peripherals for this bus. Things such as digital I/O modules, GPIB interface cards, and SCSI controllers were popular products in desktop computers, and some people started building small test sets based on the availability and cheap price's.




PCI and ISA slots coexisted for many years in the standard desktop computer as shown above.

While the PCI bus was a wonderful thing, the limited number of slots in a desktop computer, and the difficulties of opening up your computer to get to anything made it a bit of a pain. As home computer users began to dominate the market, the desktop PC kept shrinking in size. Things like extra PCI slots just added cost. By the mid 2000's, it was increasingly rare to find a new desktop PC that was designed for expansion and adding peripherals.

But it didn't matter. The company that came up with the standard bus had come up with a new format called Compact PCI, which would solve all these problems for the people who wanted to use the bus. We'll cover that a little later.

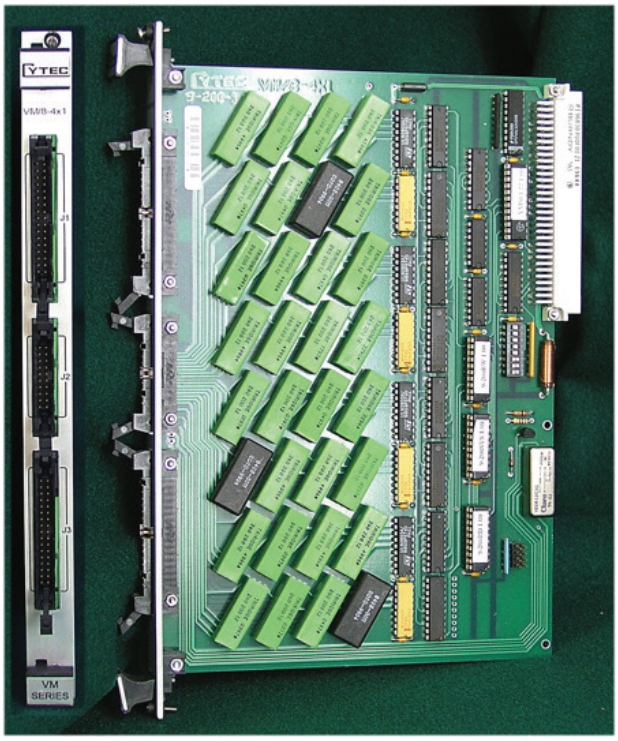
## The VME bus (Versa Module Europa bus)

The standard bus was rather quickly outdated. Users wanted faster processors, a more robust and durable card cage, a faster backplane with more pins, and additional real estate to do increasingly complex tasks. The VME bus was introduced in 1981 and completely standardized in 1987 as a major innovation. The original VME bus was built around the Motorola 68000 microprocessor and cards were available in two sizes:

Size A: 3U 160mm x 100mm, which took up half the height ( 5.25" ) and was 6.3" deep.
These modules had a single 96-pin backplane connector.

Size B: 6U 160mm x 233mm, which was 10.5" high and 6.3" deep. It had two 96-pin connectors.

The VME bus was available in 16- and 32-bit versions and eventually 64-bit versions. It had a rugged card cage, a lot of backplane pins, and early versions ran at 40 Mbps.




The VME bus found a large market in industrial controls, military and aerospace applications, medical equipment and telecommunications. Because boards all had to operate the same, it was a wonderful bus that allowed people to simply make use of the best fit boards available at the best price, and you could build systems to fit a very specific need. All of the modules were register based and very fast, so they tended to work and play well together.

We began building switch modules for VME in the mid 80s and had about ten different switch modules to choose from by the early 90s.

But, people were still unhappy about the limitations of VME. There were certain kinds of products that wouldn't fit on the board space. In about 1986, Colorado Data Systems began selling VME "C" sized modules in earnest. The new modules were the same height as "B" size modules but more than twice as deep, at 13".


What followed was the biggest upheaval in the automated test equipment and data acquisition industries yet to be seen. The "C" sized VME modules would become "VXI" modules and change the industry drastically. How and why is a complicated story.

**A rather exotic VME Bus card cage by Weiner with built in power supplies, cooling, monitor and control features and rack mount kit. Many card cages were much simpler than this.**

## VXI

In July of 1987, the VMEbus Extensions for Instrumentation (VXI) standard-effort was launched by Colorado Data Systems, Hewlett-Packard, Racal Dana Instruments, Tektronix and Wavetek. People who were already buying VME systems wanted instruments such as oscilloscopes, power sources, network analyzers, signal generators, and an entire host of equipment which would normally be a bench-top or rack mount device that was large, heavy and independent.

It made sense that once you had the VMEbus computer platform in place, you could add some instruments and have a much more versatile test station in a much smaller space. And, you had already paid for the slots, so why wouldn't you want to make better use of them?

One of things that had made this possible was the availability of new test software and graphical programming languages. Microsoft has introduced Visual C and Visual Basic, and National Instruments introduced two graphical programming languages called LabView and LabWindows. LabWindows was based on C and could be used to write code which was easily ported into LabView. LabView was a purely graphical programming language and old school C programmers often reacted badly to it. The beauty of LabView, was if you had instrument drivers already built, you could put together test system software very quickly by simply tying the drivers together and modifying small amounts of code. This meant that if the equipment manufacturers, who were very familiar with controlling their own equipment, were willing to make drivers for LabView available at little or no charge, the end user was much more likely to consider your products. Before long, almost every single test equipment vendor was offering free LabView, LabWindows, HP VEE, and Matlab drivers, as well as examples in Visual Basic and other platforms.

HP 1410 C size VXI Multi-Meter module

The next logical argument was the combination of the VXI hardware platform, and graphical programming languages, eliminated the need for bulky instruments with separate cathode ray tube screens and physical knobs and buttons. With graphical programming, you could simulate the front panel of the o-scope on the computer screen, and adjust it with the mouse while you watched the waveform and automatically stored the data. To use the marketing vernacular of the era: The software *is* the instrument.

This quickly led to the VISA ( Virtual Instrument Software Architecture ) standard, and the VXI plug and play alliance. The companies involved in these groups hammered out structured standards for both hardware and software, so that everyones equipment worked and played together well.

Soon, the industry was saying it was the end of the bench top and rack mount instrument. Test equipment and switching systems would all be reduced to plug and play modules with a software interface. Some companies completely discontinued their standard product lines and put all of their development, marketing and research into VXI. National Instruments, who had basically been a GPIB hardware company, was suddenly a huge software company and then immediately moved into the VXI hardware business. And the deep pocketed, large companies controlled standards and alliances, so you either played along or faced being swept aside.

So, what happened? Anyone reading this who wasn't involved in the business at the time is probably questioning why their test department isn't filled with VXI modules. It all made sense, from a big picture point of view and VXI systems filled a perfect niche for large scale test, where every bench would have a complete VXI test station with an exotic software interface. The reasons it didn't succeed long-term were varied and complex. The reasons it didn't succeed with switching systems were a bit simpler.

In general you can point to a few major reasons VXI never succeeded in dominating the market.

1) Overall cost -- From a single module point of view, it seemed like a good deal. The problem was you had to start with a very expensive VXI chassis ( $3,000 to $25,000 ), and then pick a "Slot 0" Control Module ($2,000 to $6,000 ). This meant that every slot in the chassis was probably worth close to $2,000.00 before you put anything in it. And since very few people would actually make use of every slot, there was often physical space and slot $ wasted. Now factor in the cost of graphical programming software with annual renewals and upgrades.

2) Plug and Play or Plug and Pray? -- The industry talked a lot about the wonders of the VISA standard and the plug and play alliance, but like all large competitive groups in a free market system, they all wanted an edge. The problems encountered when trying to make that many products, from that many different vendors, all work well together were fairly insurmountable. The big players wanted everyone to conform to their complicated software structures, such as IVI, while the smaller companies wanted to use their standard syntax and methods. Some modules were entirely register based and others took advantage of more user friendly "Message based" commands. Some tried to do both. Some companies made use of reserved pins on one of the available connectors to provide special features, which were only usable if you bought their slot 0 control module. End users devoted a lot of time making everything work well together and never saw the benefits claimed in the marketing.

3) Performance Limitations -- While some modules did a fantastic job of performing functions comparable to bench top equipment in a smaller size and lower cost, there were a lot of things, such as network analyzers and spectrum analyzers, which couldn't be duplicated on a module of that physical size. VXI chassis needed a ton of power, often leading to the use of very noisy power supplies that had previously been regulated to purely digital devices that would tolerate them. Low level measurements were difficult due to all the noise from bus activity and other devices. Connector options were limited due to the lack of physical space on the front panels. Many of these things meant you ended up with bench top equipment sitting right next to your VXI chassis, anyway.

4) Lack of Independence, knobs and buttons -- While VXI worked well in situations where a test "set" was needed, a lot of test departments didn't work that way. Many test departments and R&D labs require specific pieces of equipment for one test and another for the next test; they are simply moved around the room as needed. They are not needed at every test station every day, so it is much more cost effective to share an expensive piece of equipment than it is to keep one at every bench. And in those kinds of labs the test parameters change on the fly. Despite what the software people would claim, it turned out that humans still feel better about tactile controls, such as knobs or buttons in those situations. Having to change software on the fly because the person that wrote the interface didn't nail the function you need on the interface is infuriating, and often the people writing the software were not the people using the software, adding a layer of miscommunication.

5) Unsustainable Alliances -- The whole idea of building a system using modular boards from a wide variety of vendors was flawed from the start. Especially when you consider how fast companies change, merge, and go out of business. If your stand alone scope vendor goes under or stops supporting your model, it has no effect on your stand alone network analyzer, and a different scope is readily available. But because VXI never succeeded in taking over, it became increasingly difficult to replace modules from vendors that failed or bailed on the bus. This had a cascading effect as more and more VXI vendors discontinued support and customers became disillusioned.

Some of the Cytec VXI Bus modules available in the late 90's.

As with the other buses, we built switch modules for VXI, while the trend lasted. We still support them today. VXI carved out a niche and still exists as a viable alternative for some applications. It ended up seeing large scale use in military test sets and avionics.

Looking back on the original founders of the VXI platform, it is rather indicative of what has happened throughout the test and measurement business. Hewlett Packard spun off the test and measurement business as Agilent ( now Keysight ) and unloaded the VXI division to VXI Technologies ( now known as VTI ). Racal Dana developed a large military and avionic VXI market and eventually was bought by Astronics. Colorado Data Systems pretty much disappeared and little info is available online. Wavetek eventually became part of Aeroflex, and Tektronix was gobbled up by Danaher; both have discontinued VXI products to my knowledge. Why I don't like stand alone buses for switching applications will be discussed at the end of this chapter.

## Compact PCI and PXI

Sometime in the mid 90's, Ziatech, the same company that made the Standard Bus popular, created the "Cpci bus" or Compact PCI. The bus combined the euro style card cage and modularity of the VME bus, with the low cost components of the PCI computer bus. As the number of home computer users exploded with the rise of the internet, the cost of parts for the PCI bus had dropped drastically. Ziatech saw this as an opportunity to replace the STD bus, which was slowly fading into extinction.

As a small company that marketed to the lower cost end of the switching system industry, we immediately became interested in the technology. Having already developed VME Modules, and conventional PCI cards for the internal computer bus, it was an easy leap for us to make. We introduced a line of Compact PCI modules in 1999. VME had morphed into VXI and it was increasingly hard to compete with the big players in the limited market, so Cpci seemed like a logical step.

But, we were not the only ones to notice. Around the same time, National Instruments, who had grown enormously through their sales of test development software and their line of VXI hardware, had already begun to form the PXI Alliance. This group of companies did to Compact PCI what they had done to VME with the VXI alliance. Once again, to get any traction, you were required to join the alliance, conforming to their standards and compete on their field.

To be honest, at this point I was pretty much done chasing the latest bus technologies. By now, the larger companies were all manufacturing overseas in volume and cutting margins to the bone. For a company that specialized in engineered switching solutions and large, one-off projects, it didn't make a lot of sense.
PXI has done well with a small number of companies dominating the market for test equipment applications. It filled the niche for test set applications much better than VXI, and at a lower cost and smaller size. The only threats to its existence will be LAN and the discontinuation of the PCI bus, which looks pretty stable in 2015.

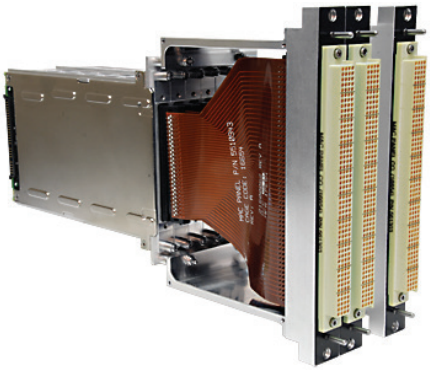
## The problems with switching systems on a stand alone bus

I probably sound like a grumpy old man whining about all these "gosh darn bus technologies." If so, I apologize. It isn't out of any fear of new technology or wanting to do things the way I always have. It is more an issue with the specific application of switching systems on these technologies. Since that is what I've done most of my life, I tend to get stuck in that grove. There is definitely a need and fairly large market for switch modules within the realm that bus technologies are a good fit. But, just like solid state switches, they tend to be application specific. So I'll just list the things to consider if you are debating going down that route.

1) **Performance overkill --** One of the main motivations to move instrumentation onto a bus was speed. Many instruments have to operate at very high speeds and stream a lot of information back and forth. This isn't the case for switching systems. A lot of the bus switch modules are still using electro-mechanical relays, which take 1 to 20 ms to open or close, and the commands to do that are minimal and rarely require many bits. Even the fastest solid state relays tend to operate in the micro-second range and often require settling times before a decent reading can be taken. So, building a large switching system based on a 400 MHz, 64 bit bus is a bit like buying a Ferrari to tow your trailer -- yes, you can do it, it just might not be the best option.

Just cause it can be done, that doesn't mean it should be done.

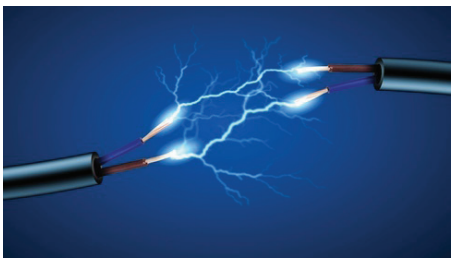
**2) Physical space --** Great gains have been made to pack a lot of relays into a little tiny space on bus modules. There are single PXI modules available with 256 relays. But, it all comes with limitations on voltage and current. While they still have a wider range than most solid state switches, they are not as broad a fit as larger relays on standard footprints, which may be available with different options. It also typically involves extremely high density connectors which may be both proprietary and tricky to work with: Need coax connectors? RJ45's? Individual D9's or Screw terminals? You're suddenly limited to small configurations per slot due to the connector size.



**3) Cost per Slot --** Since the bus systems usually have a fairly high cost per slot before you even plug a module in ($1,000 to $2,000 per slot), things like microwave relays tend to take a lot of slot $'s in panel space, with limitations on what microwave relays you can use. Need to do a lot of switching that may require 8 slots? You're suddenly looking at doubling the cost of the switching system. Already have a nice PXI chassis set-up with three spare slots open, and just need a small amount of switching? Perfect fit!

It takes 2 slots to drive a single SP6T microwave relay that's already expensive

**4) Signal Limits --** Need to switch a high-voltage or very low-level signal? Might not be as easy as you think. I once saw a VXI module advertised to take femto amp current readings. A customer was replacing it with one of our rack mount systems on a semi conductor test set. It took the VXI module 30 seconds per reading to settle and still couldn't hit the levels we did in 50 ms at 1/3rd the price. I once had a customer want to put a 5000 volt hi-pot test switch into a VXI system already worth over $75K. Was I wrong to talk them out of it? Doubt it!



## The Future of Switching Systems

When the company I worked for was started in the early 80s, the founder predicted that we might need to find something else to do in the next ten years, since some solid state relay solution would come along and wipe out the need for engineered solutions. It never did. If anything, it has become more complicated with so many different ways to approach the problem. Technology advances have simply created the need for new and better switching systems.

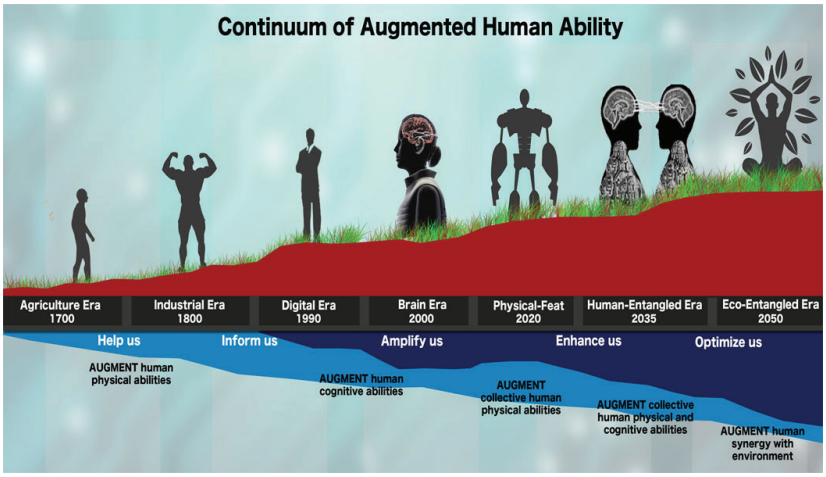
In the 80s, signals above 500 MHz were mostly seen in military and aerospace applications and were a small segment of the business. But, with the rise of the internet, cell phones, and commercial satellite technology, it is now one of the largest markets. 10 Mbps RS422 differential signals were considered a fast communication standard 25 years ago, and now CML (Current Mode Logic ) can easily run over 4 Gbps.

Voltages have gone higher in research applications, currents have gone smaller on semi-conductor testing, and fast switches are now less than a micro second, instead of 1 or 2 milliseconds. Basically, as Moore's law has continued to hold true, the technologies needed to maintain that pace have constantly provided new opportunities and challenges.

There are arguments to be made that changes in manufacturing and purchasing habits have decreased the need for automated test switching systems in the electronics industry. When cell phones are being spit out in the 10's of millions, using the most high tech assembly equipment, and only have a 2 to 3 year life expectancy, it is easy to understand why they would rather throw away bad phones than waste time carefully testing each one. Improvements in processes and quality control have eliminated the need for appropriate levels of testing, and built-in test functions eliminate problems before they ever get to assembly lines.

But, the markets have expanded in so many area's that I don't believe it's a need that will be going away soon. I won't attempt to guess what the long term future will be. There are control trends such as PXI and LAN control, which are obvious, but as previously shown, will most likely be replaced by the next widely accepted standard. I'm sure that when every device on earth is seamlessly interconnected through some 60 Gbps wireless standard piped directly into our cerebral cortex, they will look back and laugh at our cables, competing communication standards, graphical user interfaces and people that wanted to drive cars with their hands and feet, the same way we laugh at 8 track audio cassettes, smoking on airplanes and pay-phone booths.

You can look at life as a series of problems, or a series of opportunities and challenges, and that's what I love about the business I'm in. It's nice to have a front row seat in a technology field, and even if it's not the kind of work that you're ever going to hear about on the nightly news. It is the kind of work that helps you enjoy getting up in the morning to solve problems. So, I'll look forward to whatever comes next.




Is obtaining "eco-nirvana" the last step? Hard to say. It will probably be part of it, unless the robots overlords are the evolutionary end game.