33-1/3 to Light Speed and Back Again

My personal experience with technology From the early years of computers in the 1970's to today in 2008

By Rich Love May 7, 2008

I was born January 3, 1948, three years before the first computer (UNIVAC) existed in the United States. It was the year that the 33-1/3 RPM record was invented. This new record format allowed 30 minutes of music on each side of the record. Typically, there were six songs per side for a total of 12 songs. This was a vast improvement over the previous 45 and 78 RPM records, which only had one song per side for a total of two songs.

The transistor was invented just a few months before I was born. Today in 2010, an iPod digitally stores thousands of songs using electrons traveling at the speed of light.

My grandfather was born before the flashlight was invented and before electricity was being distributed widely as AC current in the United States. He lived in the desert with no electricity and no running water. He drove to town once a week in his Model A Ford to get water and supplies. During the hot summer months, he slept in a mineshaft because it was cooler underground. Nobody had air conditioners.

My grandfather has long since departed this Earth. I am now 62 years old. I am a witness to an extraordinary time in human history. The invention of electricity and the computer radically changed the human condition and the knowledge about human origins. The computer has enabled us to map the human genome and trace our history by examining the DNA of our ancestors. Through geology, astronomy, genealogy, physics, chemistry and all the other sciences, we are learning who we are and where we came from. The computer has rapidly advanced the speed at which all of these sciences are making new discoveries.

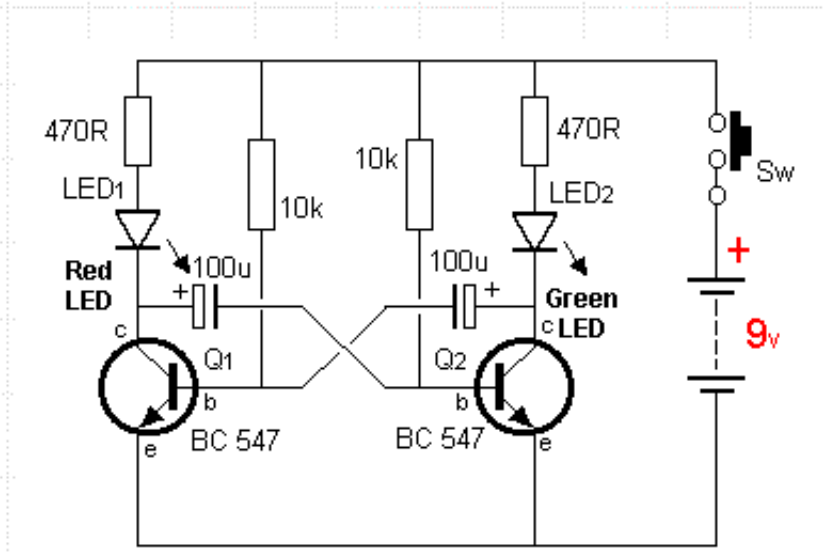
Science now traces modern human gnomes to fossils found in Africa 150,000 years ago. Older human fossils have been found millions of years old. It is amazing to think that humans lived on this earth for so long without electricity. No electric lights, no electronic communication, no appliances. It has only been barely over 100 years since electricity started to be distributed on Earth. The computer has only been available to the masses for 30 years.

NASA has sent spacecraft to Mars and determined that there was water at one time. They have also photographed Europa, a Moon of Jupiter, with a flyby and determined that it has liquid below a frozen surface. They plan to send a submarine there in the future to look for life in the ocean. The human genome has been mapped using super computers, paving the way for totally understanding the evolutionary design of the human body. None of this would be possible without the computer.

I don't think the average person appreciates this moment in time. It really is a privilege to be part of this generation.

My History, Experience and Observations about Computer Hardware and Software Evolution over the past 37 years.

I attended the University of Alaska at Fairbanks from 1971 though 1973 and graduated with a degree in Electronics Technology. The classes I took the first year, taught basic electronics and electronics math. The following years, I learned the theory behind radio, television, radar and computers. In our labs, we built a simple AM radio. We also learned how computers were designed. We built a flip-flop (the most basic way to make a computer gate flip from 0 to 1 and back again). The first computers were built using these flip-flop circuits. Two transistors, and a few resistors, capacitors and diodes were used to create one bit of computer information. (One bit, on or off). Note: It takes 8 bits to create one text character. Therefore, it would take 8 of these circuits to generate the letter A.



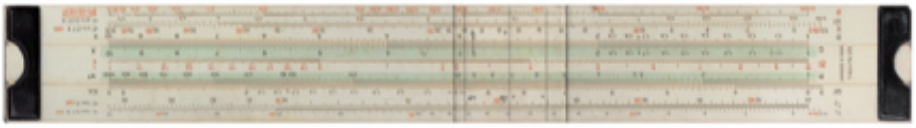
THE FLIP FLOP CIRCUIT

The Flip-Flop installed on a circuit board.

[Reference: How to build a flip-flop]

As a side note: When I entered college, I had no experience with electronics at all. My only exposure was ignition systems in cars since I was an avid car enthusiast and mechanic. I had two years of auto mechanics in high school and always maintained my own cars. It was difficult for me to learn electronics because I had previously lived in a mechanical world. Electronics seemed like an imaginary world to me because you had to imagine electrons moving through a wire. You had to imagine what the electrons did when they met a diode in their path or a capacitor or resistor. I remember having dreams at night where I was an electron traveling through a wire, blocked by a diode.

Electronic calculators were just starting to emerge as an affordable alternative to the slide rule in 1971. We were not allowed to use a calculator in class. That was considered cheating. So I became very good at using the slide rule for my electronics math calculations for designing circuits.



Slide rule I used in College

In 1972, Hewlett Packard released the HP-35 Scientific calculator. That changed everything. Now it was possible to do calculations quickly and accurately. It was too expensive for me to own one, but it paved the way for less expensive calculators that followed. I remember the absolute giddiness of the electronics and scientific community over this calculator. Before, I left college in 1973, the calculator was now acceptable to use in class and had become commonplace.

HP-35 Calculator

This time in history was during the transition from tubes to transistors. Integrated circuits were emerging but were so new that our electronics instructor could only talk about them in theory. Due to equipment funding, colleges are always a bit behind the technology curve anyway.

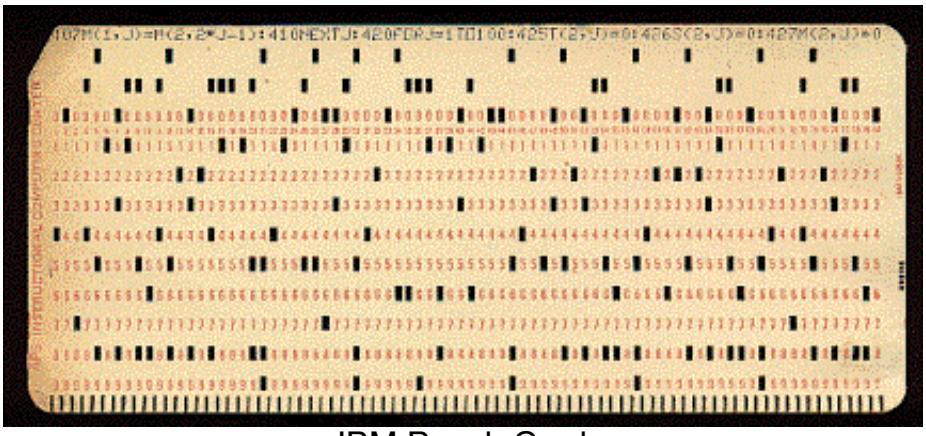
Learning electronics was an important stepping-stone toward programming. Understanding electronics and programming are very similar because you live in a virtual world. Your thought

processes for design are very slow, step-by-step actions and the end result is something that happens instantly. It takes an incredible amount of man-hours to design a complex circuit or program. The average person cannot begin to comprehend the thought process that goes into designs that they take for granted.

I learned to write my first program on an old Athena computer that was donated to the college. It filled a huge room and ran on many modules in sealed metal cans. These rectangular cans were about 4 inches by 2 inches and had a handle on the outside and plugged into a socket. That way you could replace any defective modules. The modules contained transistors, resistors, capacitors etc. (the same stuff we were soldering together in class to build our flip-flops, shift registers etc). The computer had no display or printer. The only way to program it was to push buttons on a very large console. The console was about five-feet wide. You would enter your program as binary numbers one line at a time and then push the run button and watch the lights blink to see if you programmed it correctly.

Rich at the console of an Athena computer at the University of Alaska, Fairbanks.

The next computer I programmed was hidden in a computer room at the college. I don't think I ever saw it. I used IBM punch cards to type binary instructions on each card. Then the card stack was submitted to the computer administrator to be run during the night. The next day I could see whether my program worked by the report that was generated by the computer operator.



IBM Punch Card

During my college years, I worked nights at the USO in Fairbanks. My first useful electronic project was at the USO. I designed and installed a public address system with a row of switches at the front desk and speakers in every room of the USO. I had a microphone at the desk and could flip a switch to make announcements in any room I wanted.

After graduating in 1973, my first job was at a radar site, 27 miles north of Fairbanks, working for Stanford Research Institute. My job was to maintain the computer there, a Scientific Data Systems 930. This computer was used to record data from the radar to analyze the aurora borealis. There were a couple of electronics engineers at the radar site and a couple of physicists.

Physicists would also visit the site from all over the world to perform experiments using the radar. As I remember, the computer was about eight-feet wide and five-feet high. It did not have a hard drive. It recorded data onto reel-to-reel tape drives. There was a Teletype printer attached as the console device. There was no CRT screen. You used the keyboard on the Teletype and its printed output to operate the computer.

You could not program the computer from the console. You used a punch card machine instead. The punch card machine had a keyboard for data entry. You would program one line of computer code on each card. Then when you had all of your cards punched, you would take the stack and drop it into the card reader. Then on the console, you would tell the program to read the stack of cards and then compile and run your program. Of course, it would not work and you would have to figure out which cards had errors on them. You would fish though the cards and find the cards with the errors and re-type those cards. Take the stack back to the card reader and try again.

I wrote my first important (actually useful) program using this method and the FORTRAN programming language. A visiting physicist, Vince Wickwire, gave me the project. He asked me to write a program to monitor the radar output and detect when certain changes occurred. The radar was taking samples from the ionosphere about 300 miles above earth. Recording to tape could go for hours. The physicists needed a way to index the tapes to look for data. The program I wrote tracked changes in the radar azimuth and elevation. It also looked for significant changes in electron densities and ion temperatures. This was a great learning experience for me. I did not know the first thing about FORTRAN, but Vince taught me enough

to get started. I figured out the rest on my own.

The second electronic project I designed was a flash adaptor for the Polaroid SX-70 camera. I think this was in 1972. Polaroid cameras were popular at the time but used flash bulbs instead of an electronic flash. You would buy a flash bar that had five flash bulbs. After five pictures, you would buy another flash bar. That was expensive. I designed a circuit that would plug into the flash bar socket. It took the output from the Polaroid and connected it to a standard electronic flash socket using a diode, capacitor and a transistor. I can't seem to find the schematic I drew for this design, but I still have the flash adaptor. I used the flash bar body but removed all of the flash bulbs and installed my circuitry inside. Then mounted the electronic flash shoe on top of the flash bar. I used it for years and it worked perfectly. I tried to sell this device to Kalt (they sell photographic accessories). I shipped a sample to them and they were very interested but said they could not get it manufactured cheaply enough to sell them.

SX-70 Polaroid Camera

Rich's Polaroid SX-70 Flash Adaptor Prototype

In 1975, I packed up the family and we moved to Washington State. My first job was with a company that designed keyboards and monitors. I wire-wrapped circuit boards for the engineers. This was a great learning experience for me. I read the engineer's schematics and wired up the board, connecting all of the electronic components together. Then I would troubleshoot the circuit with the engineer to make it work. There could be several things wrong – wired incorrectly or wires shorting, bad electronic components or incorrect design by the engineer. We used an oscilloscope to troubleshoot the circuit and get it working.

1976, my next job was with CX, a film processing company. They made equipment for the photo

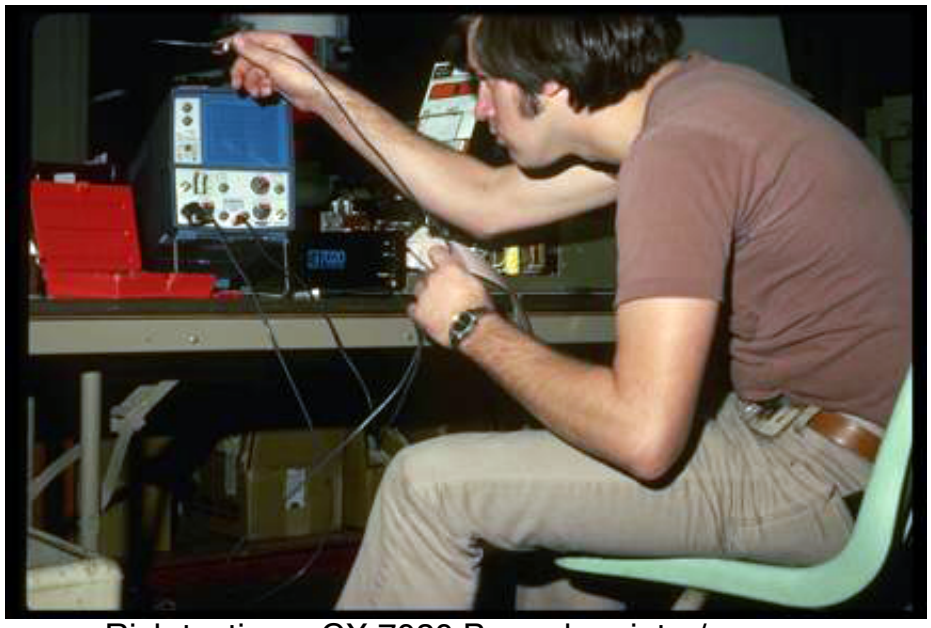
processing industry (barcode readers, printers and terminals). I worked in the final test department for the first four years there. My experience at my previous job really helped me fit into this job. They sold a complete system that used a DEC PDP-8 minicomputer along with their own peripheral equipment.

This was before digital cameras existed. Film was still a big business. Film processing plants would track orders by using a film bag with a barcode at the bottom of the bag. The unprocessed roll of film would be placed in the bag at the retail store when the customer dropped off the film to be processed. When the bag arrived at the film processing plant, an operator would insert the bag into a CX-7020 printer/scanner. The barcode was read and a price was printed on the bag. The transaction was recorded in the computer. A terminal with a single line display would display some information about the order for the operator.

At the CX manufacturing plant, I would test the entire system before it was shipped to the processing plant. The PDP-8 (and later PDP-11) computer was purchased from DEC. I would test the computer with all of the peripheral equipment connected to it before shipping it to the processing plant. As I mentioned earlier, the peripheral equipment included the CX-7020 printer/scanner and a terminal. The printer and terminal were manufactured at CX by low-wage assemblers and delivered to my final test department. The circuit boards were all assembled and soldered by hand. It was absolutely guaranteed that the printer and terminal would not work. There were just too many possibilities for errors. The circuit boards would have solder bridges between pins, IC chips and other components would be bad, wires would have bad connectors or be wired incorrectly. I used an oscilloscope and schematics to troubleshoot and repair the printers and terminals. I became so good at this that I later wrote the service manuals for all of the equipment that I repaired during my four years in the test department. I wrote nine manuals total.

The field service technicians and the final test department used these manuals. I think it was 1981 when CX got their first circuit board testing equipment. It was a programmable machine that you could program to test circuit boards. You would lay the board down on a flexible bed that had a vacuum to lower the board onto a bed of nails. Each one of the nails in the bed would contact the correct spot on the underside of the circuit board to test it.

This created a whole new department of technicians. Their job was to take the printout from the testing machine and replace the suggested component and/or repair solder bridges. Now a reliable circuit board could be installed in the printer or terminal before it was delivered to final test, making their job much easier. This was the beginning of the end for electronic technicians with my skills. Now computers were starting to analyze computers to let the technicians know what to repair or replace.

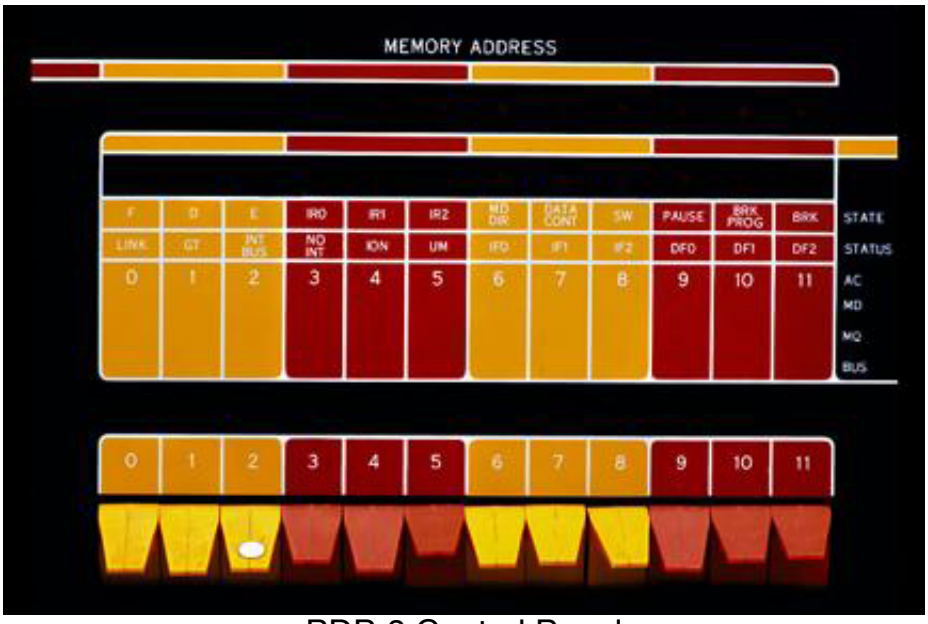


Rich testing a CX 7020 Barcode printer/scanner

While at CX, I learned to program the PDP-8 computer in Pal-8 assembly language. I started by learning how the existing test programs worked and then modified them to add more features. Then I wrote a few of my own test programs. PDP-8's were manufactured from 1965 through 1974.

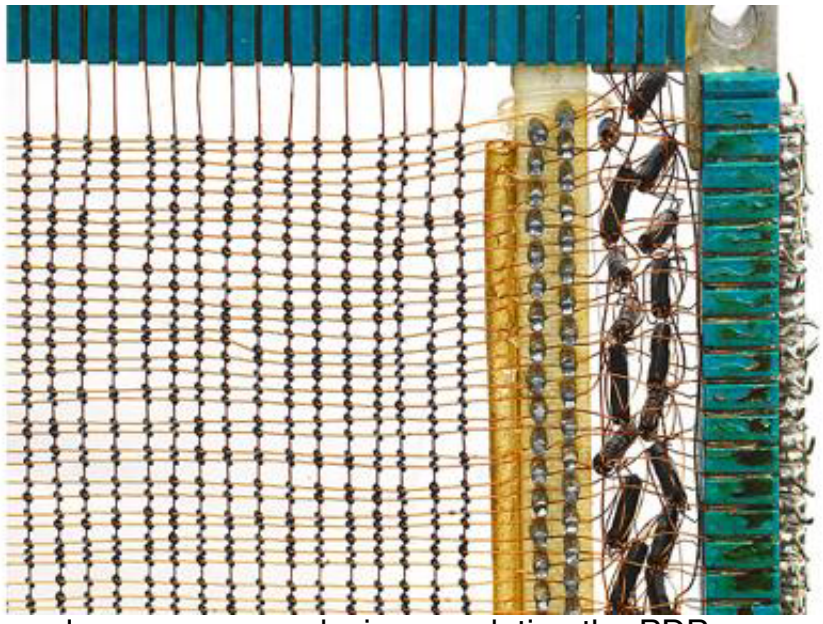
These computers had a panel of switches that you used to boot the computer. You would flip the switches to set the correct address and store or read the data in that address position. You could actually write an entire program and run it from those switches. However, it was more efficient to use punched paper tape to enter your program. You could also type in the program using a Teletype. It was not until the late 1970s that CRT terminals were in widespread use with the release of the VT50 CRT terminal.

The computer did not have a CPU chip. It used a set of three logic boards to create the CPU. The CPU could only address 4K of memory at a time. The CPU handled fields of 4K memory. A program would typically take 3K of memory, leaving 1K for user space. But you could have more than one program running (one in each 4K memory field) up to eight fields. This was the beginning of a multi-user system because you could assign one user to each field.



PDP-8 Control Panel

The PDP-8 used core memory. This was very expensive memory consisting of tiny ferrite iron cores that looked like little doughnuts. Tiny wires passed through the core to magnetize it in a positive or negative direction. Then the memory was read by a sense wire that would detect the polarity of the charge and determine if the value was a one or a zero. This was one bit of information. The circuit board containing these cores was 10.5 inches by 9 inches. Each core represented one bit of data. The cores were installed on stacks. Each stack had an array of 64 x 64 cores for a total of 4096 cores. But to get a 12-bit word, you needed 12 stacks of cores. These stacks were sandwiched on top of each other on one circuit board to produce a total of 49,152 cores. Divide 49,152 by 12 and you get 4096 12-bit words (or 4K). The whole 4K board could store 4,096 characters . Each character required 12 bits of data or 12 cores. Eight bits were used for the ASCII code of data and the remaining four bits were used for error checking and other functions. You could daisy-chain up to eight boards for a maximum of 32K. Today, it is very hard to imagine a computer that could run on 32K of memory. But these computers cost $18,000 with 4K of memory and no peripherals.



Very early core memory design pre-dating the PDP computers.

To read a character, the computer would activate the sense lines going though 12 ferrite cores and detect the zero or one value. Using the binary numbering system, you could store and read characters. For instance, a character is represented by reading the first eight core values (the last four cores were not used for the character, but were used for operation codes). DEC used octal math to describe characters. These were in groups of three. However, I have always thought of them in groups of four, which is BCD or binary coded decimal, which I mentally convert to decimal.
Here is the letter 'A':
0110 0101
That is a decimal 65, which is the ASCII code for the letter 'A'. The letter 'B' is 66 and the letter 'C' is 67 etc. ASCII codes from 01 through 31 are control characters that are invisible.

Using binary arithmetic, you can decode the zeros and ones in the binary number. It's really not hard to understand (well, maybe). The numbers are in groups of four. The first group represents the six and the second group is the five (65 = the letter A). Look at the first group, which is 0110. The decimal value of the last character on the right is one, the second from the right is two, the third from the right is four and the last one is 16. All you do is look at the value of each position. If it has a one value, then you assign the decimal value of that position and then add all the numbers together. So, our first group has a 0110. You add 2 + 4 to get 6 and that is the value of the group. The second group has a value of 1 + 4 to get 5. Hence, the decimal value of the two groups is 65, which is the letter A.

Computers still use this same logic to read and write characters. The word "stack" is still used to describe sections of memory today.

Core memory boards were originally used in the ENIAC computer in 1949. They saw their last use in the late 1970s. This is an amazing technology to me. These memory boards were originally handmade using a stereo-microscope. The ferrite cores were set in place and tiny wires were threaded through each core.

Since memory was so expensive, the programmers had to be very clever about memory usage. Virtual memory was used on the disk drives to have enough room for running software. It is hard to imagine a computer today running on 4K of memory. But in the 1970s, that's what they did. The early PDP-8 computers used punched paper tape to read a program. Or you could type in a program using a 110-baud Teletype. There were no CRTs connected to computers yet.

While working at CX, I taught myself the BASIC programming language and used it to program a manufacturing job control system from scratch. I also wrote a Bulletin Board System in my spare time. I wanted it to be an in-house system for the employees to use, but never got approval for its use. Bulletin Board Software was used at that time (before the invention of the internet). You would use a dumb terminal and a modem to dial into BBS systems and leave messages and respond to others.

In 1977, I purchased a used terminal and set it up in my kitchen at home. I built my own 300-baud modem using schematics I found somewhere. At first, the only thing you could use it for was to dial into BBS systems. But later, Compuserve came into existence and that became more popular than BBS systems. America Online was written for Macintosh in 1989 and I was one of the first members and still am, just for old time sake.

In 1978 I purchased my first computer. It was an Apple II. Personal computers were just starting to emerge. The Apple II was first released in 1977. The Apple I was released the year before, but was considered a hobbyist computer. The Apple II was the first successful personal computer to be widely distributed to the masses. For the first time, you could afford to buy a computer that you could use to write letters, store and retrieve a database, calculate numbers with a spreadsheet, balance your checkbook and play games.

The Apple II had 4K of random access memory expandable to 48K. The CPU clock speed was one megahertz. Disk storage was a 5-¼ floppy drive that held 112K of data. Typically you would have two of these drives. The first drive was used to boot and run one program. The second drive was used to store your data for that program.

First Apple I Computer Designed by Steve Wozniac

Apple II

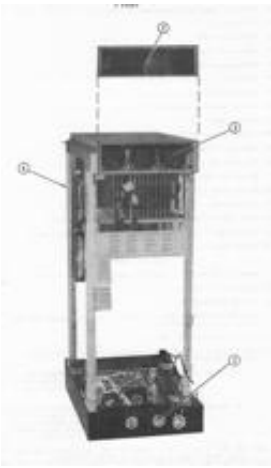
In 1982, I went to work for Microdata as a Field Service Engineer. I drove all around the Seattle area repairing mini-computers and peripheral equipment at customer sites. Companies were still paying over $100,000 for a computer in those days. The personal computer was just starting to emerge but was not in wide use in business yet. Most businesses had a mini-computer with dumb terminals connected to it via RS-232 serial cable. The main printer was attached to the computer. It was quite often a Printronix 300-line printer. This was a 300-lines-per-minute dot matrix printer. Each user had a terminal and sometimes had a small slave printer attached to the terminal. The terminals were called dumb terminals because they were not a

computer (they do have a program running in firmware but are not programmable and have no storage). They had a keyboard and a CRT display. When you typed on the keyboard, you were actually sending each keystroke to the mini-computer via the serial cable. The mini-computer would echo the character you typed back to your CRT screen.

The mini-computer would also send escape and control characters to the CRT that were invisible to the operator. Those sequences would tell the cursor where to go on the screen, change screen colors and character colors, erase characters or whole blocks of characters on the screen. It could designate areas of the screen that were protected and then do an entire screen erase leaving the protected characters on the screen. This was purely a text-based system with no graphics. The closest thing to graphics was that it would place special drawing characters on the screen to form boxes around text. These systems actually worked quite well and there are many businesses using dumb terminals today in 2008.

The mini-computers stood about five-feet tall and were about two-feet wide and three-feet deep. This was a rack system. With the covers on, you could not tell that there was a rack inside. But when you took the covers off, you could see that the computer only took up about one foot in height of the five-foot tall system. It slid into the rack on rails. You could pull the whole computer assembly outward to get at the circuit boards.

Each board was approximately eight-inches wide. There was a CPU board, memory board, serial interface board, tape interface board and disk interface board. Reel-to-reel tapes were still used as the main boot device. When building a system from scratch, you would always start with a boot tape, which would load the software onto the hard drive. Backups were always done on tape. So, it was common to have two reel-to-reel tape drive units. These were large tape units that were about three-feet high and were mounted in their own separate racks, one tape unit per rack.



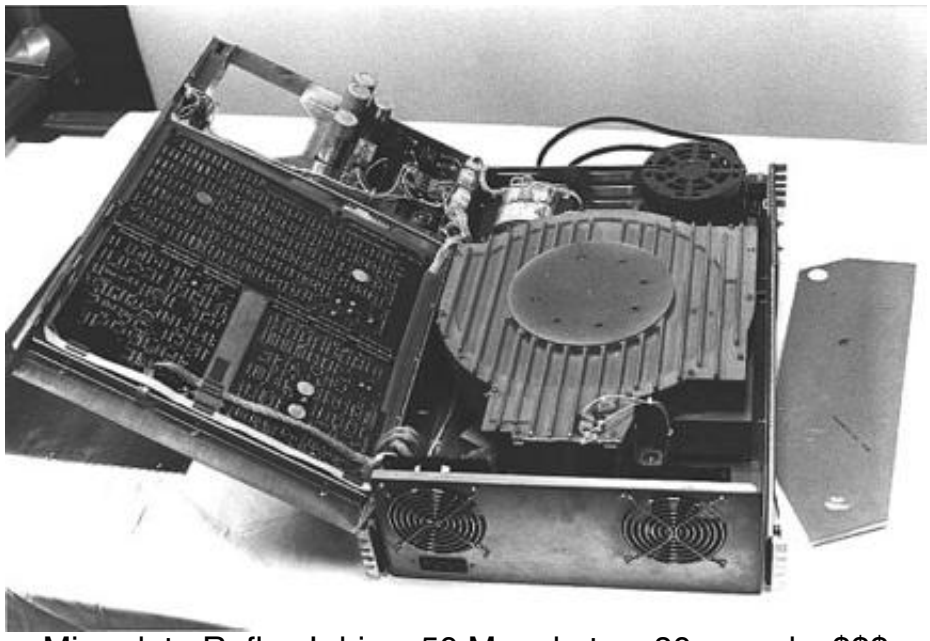
Mini-Computer Rack
Actual size, five-feet high

The tape drives needed constant maintenance. You had to clean the read-head with an alcohol pad at least once a week. The stepper motors used to drive the reels would require electronic adjustment using an oscilloscope every few months. The alignment of the tape to the head would also need adjustments. When the drive went bad, I would figure out which part was bad

and replace it. Then re-calibrate the drive with the oscilloscope again. There were many replaceable parts: read/write head, reel motors, electronic circuit boards, servo motors, tension arms, rollers etc.

Microdata tape drive

The disk drives were huge in those days. Internal disk drives that could mount inside of the main rack were about two-feet wide and 2.5-feet deep and six-inches high. They weighed about 80 pounds and consisted of large multiple platters that were about one foot in diameter. There were several circuit boards in the unit to control the drive. A typical drive like that would hold 10 MB of data. During the next few years, the drives stayed the same size but the capacity increased to 50 MB of data. Compare that to drives in 2008 that hold eight GigaBytes of data on a USB thumb drive that you put in your pocket or on your keychain and cost $50. That's 160 times more data than the 80-pound drive of the 1970s costing $10,000.



Microdata Reflex I drive. 50 Megabytes, 80 pounds, $$$

Those disk drives were very expensive and were expensive to maintain. When the drive crashed, due to the floating head coming in contact with the platter, my job would be to open the drive up and replace a platter and the head. Then use an oscilloscope to re-align the head to the spinning platter.

Larger external drives were affectionately called washtub drives. They stood about four-feet tall and were about two-feet square. They had a removable cartridge full of platters. These cartridges typically had a capacity of 50 MB. A large computer room could typically be 50-feet square with washtub drives and tape drive racks all over the room.

Today in 2008, you can get a solid-state USB flash drive that is one inch long, holds 8000 MB of data and is portable (put it in your pocket). Hard drives are five-inches long and hold 750 GB of data. When a drive goes bad, you throw it away and buy a new one.

As a field service engineer, the knowledge and skill required to repair computers in the 1970s and 1980s was at a much higher level than today's computer technicians. You were actually a combination of technician and engineer. The equipment was so expensive, it was worth the time and money to repair. Customers would typically pay $5000 per month for a maintenance contract that covered their computer, terminals, printers, tape drives and disk drives.

While working for McDonnell Douglas as a Field Engineer in the Seattle area, I was sent to southern California many times to learn how to maintain and repair new equipment as it was released. Whenever a new disk drive hit the market, I would be back in California for several weeks to learn it. The classes were very in-depth. The instructor would go over each schematic for the circuit boards and explain the complete logic of the circuits. We would use oscilloscopes to troubleshoot the boards and make adjustments. There was a lot of lab time where the instructor would plant a bad part in the drive or misadjust something to break the drive. Then I would figure out what he did and fix it by replacing the bad board I found or making an adjustment. The disk platters and heads were replaceable items also. The drives usually cost

about $10,000. I also went to the same intense training sessions for tape drives, entire computer systems and printers.

The most common business printer was the Printronix 300-line printer. It was a dot matrix printer that used 132 hammers with little points at the end of each hammer. When the hammer would strike the paper, it would make a dot. This hammer assembly shuttled back and forth from left to right as it printed. They also called this a shuttle printer.

There were several circuit boards that controlled the shuttle, hammers and paper feeding. The hammers would get weak or break. They were all individually replaceable. The shuttle was necessary to print the width of a character. If a character was six dots wide, the shuttle would move over by the width of six dots and then move back again. During this horizontal movement, the hammers would fire in the correct location to form a character. A common problem with these printers was misaligned characters. They would get wavy looking when it was not aligned properly. Using an oscilloscope, I would adjust potentiometers on the circuit boards to do the alignment.

Printronix Printer with Sequel mini-computer and Terminal

Programming

As I mentioned earlier, the first programming language I learned was FORTRAN while at Stanford Research Institute in Alaska during the early 1970s, using IBM punch cards. I wrote the LIST-A-CHON program for cataloging tapes recorded by the radar system.

In the late 1970s, I was working for CX Corporation as a test technician for photo processing equipment. During that time, I taught myself assembly language on a DEC PDP-8 mini-computer. Programs that tested the functionality of the photo processing equipment were written is assembly language. I modified these programs to improve the tests. When equipment design changes were implemented, I changed the programming to work with the new designs.

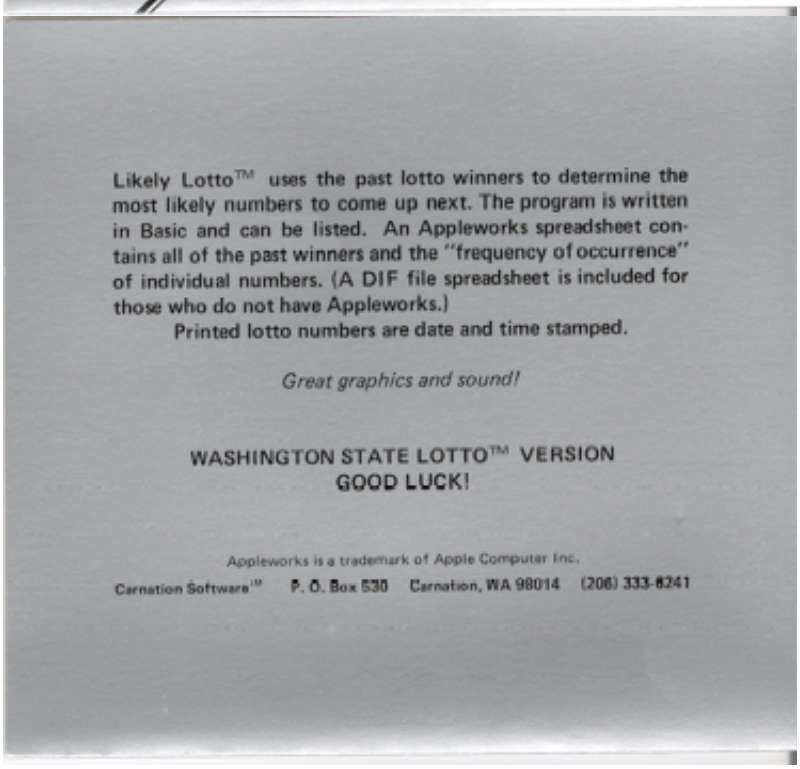
I also taught myself BASIC, using the PDP-8 computer. I programmed a Bulletin Board System in BASIC on a Data General computer at CX and wrote a biorhythm program in my spare time.

In 1978, I bought my first personal computer, an Apple II. I converted my biorhythm program to Apple II BASIC.

My first commercial program was a lotto-number-picking program for the Apple II called Likely Lotto. I sold it at Empire Electronics in Burien, WA. I loved that store. In the early 80s, it was the

only Apple Store in the Seattle area. It was such a thrill to see new products come on the market for the Apple II. It was a newly emerging market and there was not that much available yet. It made every product release special (mice, joysticks, disk drives, monitors, software). I also sold the lotto program at Programs Plus in South Seattle.

My Likely Lotto software was on a 5-¼ floppy disk. My wife, Karen, designed the package cover and I made the disk copies myself. I later converted it to run on Apple IIGS on a 3-½ floppy disk.



3.5-inch Floppy Disk

Terminal Emulation Software

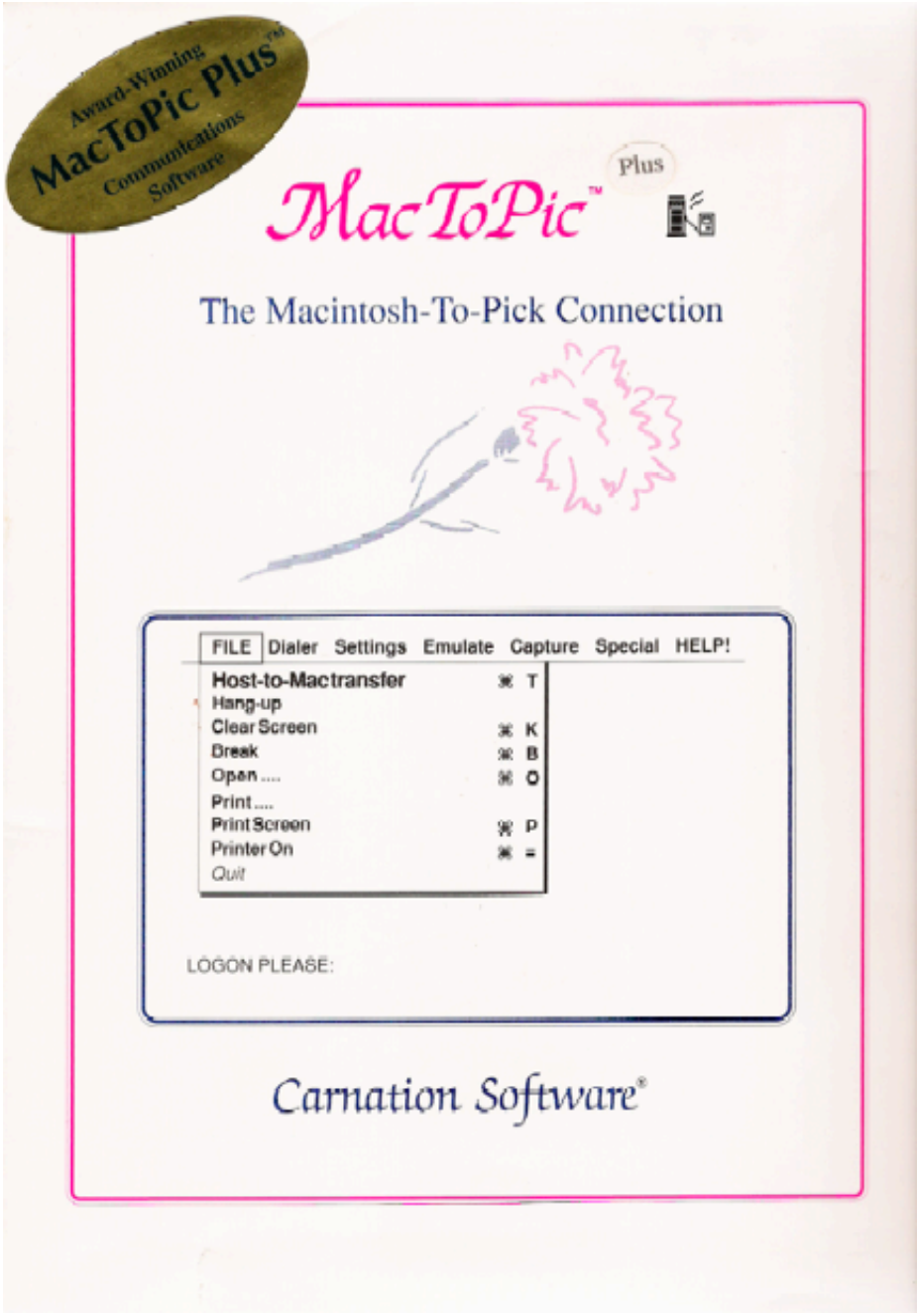
In 1986, I purchased my first Macintosh and learned to use Microsoft QuickBASIC. A few years later, I was working for Microdata as a Field Service Engineer. I drove all over the Seattle area repairing mini-computers and peripherals. A couple of my co-workers were working on a Prism terminal emulation program for their Atari. I wrote my own version for Macintosh and called it MacPrism. I soon discovered that someone else was using the name MacPrism and changed the name to MacToPic. The Microdata systems ran on PICK software, but Microdata had their own version called Reality. MacToPic was a play on words. I was afraid to use the word 'PICK' in the name because Dick Pick was known for suing anyone who did. The PICK system was a relational database system and quite popular with businesses at the time.

In 1990, a consultant for Bank of America heard that I had written MacToPic and asked me if I would modify the program to transfer data to and from the bank system that was running on a Microdata computer. They also needed Prism terminal emulation. Bank of America banks in the Seattle area were using Mac SE computers with a little nine-inch screen. MacToPic had to work on those small screens as well as the Mac II 12-inch screens.

I wrote the data transfer routines to convert the PICK data format to a Macintosh format that you could save as a Microsoft Excel spreadsheet. Bank of America was my first large customer and purchased a multiple-user site license for MacToPic.

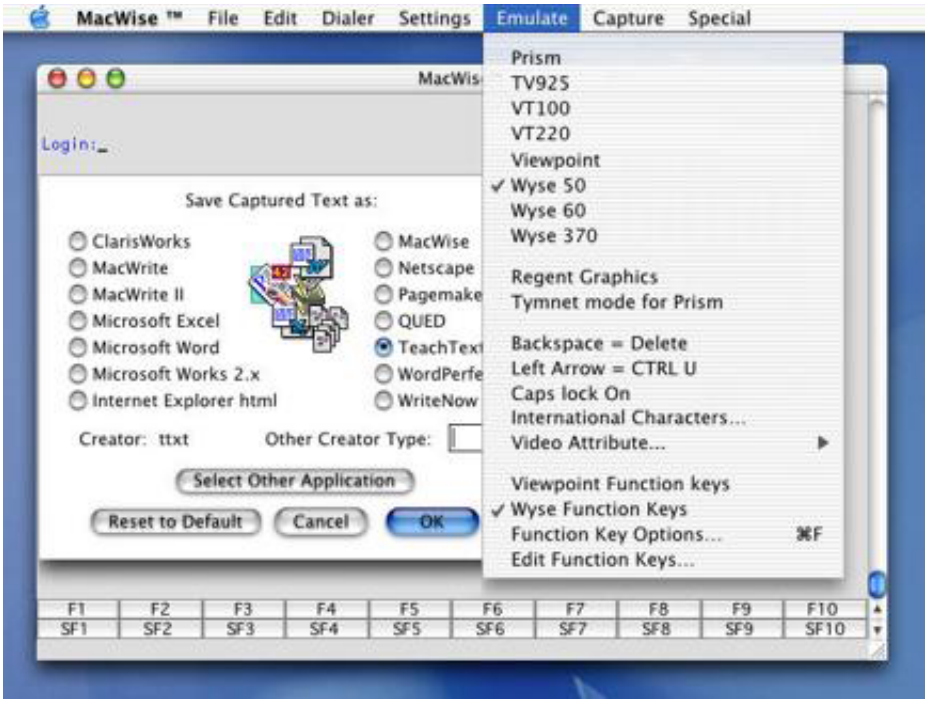
After that success, I realized that there was a market for my software in the PICK business community. I obtained a mailing list and started promoting MacToPic. Mc Donnell Douglas purchased Microdata in 1989. My software was such a large success that I quit my job at McDonnell Douglas and started my own company selling MacToPic at $295 per user. I was able to maintain that price for several years until the dot-com bust in the mid 1990s. During those

years, I added new features to MacToPic (more terminal emulation types, AppleScript and other features).



In 1995, companies were not willing to pay high prices for software anymore. I decided to come out with a stripped down version of MacToPic and call it MacWise; I priced it at $95. I also had to convert from QuickBASIC to FutureBASIC. That was about a six-month intense programming job. Microsoft forced me into this situation because they stopped selling and supporting QuickBasic for Mac.

A few years later, my MacToPic sales were so low that I decided to include most of the features in MacWise and retain the $95 price.



Today, in 2008, I am still selling MacWise (18 years of terminal emulation programming and sales) and sales are increasing due to the increasing market share of Macintosh computers.

I have gone though some major conversion programming during that time. First, there was the conversion from QuickBasic to FutureBASIC in the early 1990s. Then Apple released the CommToolBox for serial and telnet communications. I spent another six months working on that. Then there was the conversion to OS X, which took me almost a year. Most recently, I have converted MacWise from Basic to C, making MacWise a universal application that will run natively on Intel and PowerPC Macs. That took me seven months of intense programming and another month of debugging after the release.

All of this work has been worth the effort. I have many thousands of customers who rely on MacWise continuing to work as new operating system technologies are released.
www.macwise.com

Back to 33-1/3 Records

It turns out that there is something very appealing about vinyl records. I recently took my old record collection out of the closet and set up my old record player. It is a real joy to play the old 60s and 70s albums. Bob Dylan – Nashville Skyline, Beatles – Abby Road, Gordon Lightfoot. There is something very special about that old analog sound. Even with the clicks and pops you get from vinyl, the sound has a richness and spaciousness you just don't get from digital recordings.

My Hope for the Future

Computers have been very useful, enabling scientists to discover who we are, what we are made of, where we came from and where we fit into the cosmic universe and beyond. Not all of those questions have been answered yet.

It is my hope that technology will be used with intelligence and compasion to make the earth a better place to live for everyone.

Use technology to educate ourselves and ensure the survival of the earth and the human race.

Use technology to find new sources for energy and stop the danger of wars and financial crises caused by our limited oil resources.

Use technology to make the earth a greener, safer, a more enjoyable place to live.

Use technology to prevent disasters that can extinguish life on Earth – asteroids, comets, earthquakes, water shortages and food shortages.

Find the cure for cancer, heart disease and other life-threatening illnesses.

I have the greatest respect for scientists who devote their lives to searching for the truth, regardless of political and religious influences. Making life more enjoyable for the short time we have to spend on Earth and ensure the survival of our offspring – that is the best use of technology.

I know that this is naive, wishful thinking, but it is something we should strive for nonetheless.

Rich Love
Carnation Software
www.carnationsoftware.com
richlove@carnationsoftware.com