300 BAUD

A PERIODICAL JOURNAL OF RETROCOMPUTING

No. 2 - MAY 2010

300 BAUD

A PERIODICAL JOURNAL OF RETROCOMPUTING

ISSUE NO. 2 – MAY 2010

Once upon a time, there were three little girls who went to the police academy. And they were each assigned very... no wait, that's not right!

Once upon a time, there were two guys who were chatting away on the RetroChallenge BBS, who then came up with an idea... that idea was 300 Baud Magazine. And after much huffing and puffing the first issue made it out the door. It wasn't as easy or as smooth as it should have been, but we made it through.

Thank you to everybody who gambled and bought the first issue; a double-thank-you, to the people who told their friends about it and who then bought an issue; and a triple-thank-you to those who got in touch to say 'well done' – it was truly appreciated – it seemed like we were yelling into the internet darkness for a bit; and a quadruple-thank-you to the people who, after reading the first issue, offered their services to help grow this issue; to make it bigger and better; and a quintuple-thank-you to you. Yes, you, who are holding the current issue in your hands – especially if you came back!

Thank you.

This issue? Well it has more pages and from that a better spread of material than before – all good reading of course, but also some ideas for people who may wish to get their hands dirty. And... and there is also the competition!

Every home needs a robot – I'm pretty sure somebody said that once... it might even have been me. Anyway, this issue you get a chance to win a genuine replica of what is believed to be the second robot toy ever made. Produced a couple of years after the end of the Second World War, this [replica] tin robot practically screams "have a happy, happy post-apocalypse". What self-respecting geek doesn't want their own robot?

In this issue you will find a flyer. Fill in your details and send it to the editor's address in Scotland. Yes, send it in the post – just like all those Saturday mornings kids' TV programs. All entries received by the 23rd of June, 2010 (my birthday as it happens) will be placed in a hat, or bucket, or some other opaque receptacle and a single entry will be randomly drawn, and that entry shall win the robot.

Onwards!

CONTRIBUTIONS WANTED!

*This is a retro-zine written by the retro-scene to be read by the retro-keen.**

If you have a passion, a project or something that would be of interest then do get in touch. We certainly can't promise to run with every idea that comes our way, but the more people who write for this, the more frequent the issues are. Contact me at the email below even if you just want to chat about an article idea. Oh, and if I am trawling the web you might even get an email from me asking for help. It's a small retro world after all.

**I offer no apologies!*

William Dale Goodfellow
Editor-in-training
editor@300baudmagazine.com

ONLINE DESTINATIONS
IN THE ERA OF 300 BAUD

BY MICHEAL H. MCCABE

300 Baud: The title of this publication implies something more than simple hobbyist retro-computing. It implies telecommunication – something that we're all familiar with now that we have the Internet in all its glory. It begs the question though, with whom were we communicating in the days of 8-bit systems, 300 baud modems, and usurious telephone rates? I hope this article can shed some light on the nearly forgotten online destinations of the 1970s and early 1980s.

DESTINATION 1: TIMESHARING SERVICES

If we travel back in time a few years, we find that the first real "online destination" for many computer hobbyists was a mainframe or minicomputer timesharing system. In my case, it was the local college and its UNIVAC 90/60. The casual web-surfer might wonder WHY, but the reason has little to do with hacking or mischief and much to do with homework. My homework required a FORTRAN compiler and a statistical package called MINITAB II; neither of which was available on the tiny Commodore VIC-20 that I scrimped for, saved for, and finally extorted my mother into purchasing.

The usual means of communication with the UNIVAC was a Teletype – most were using the sleek and sophisticated Model 43 by 1981; others were still using the mechanical beast known as the Model 33. Both were large, heavy, expensive, and slow. A microcomputer like the VIC-20 or the Apple II (the one I REALLY wanted) was smaller, lighter, cheaper, and faster. They also allowed you to minimize your connect charges by composing code "off line" using a text editor and uploading the completed work at the lighting-fast speed of 30 characters per second.

One reason that speed was an issue was that, in those days, Edinboro State College was a long distance call for which the phone company charged a hefty premium. Although only located 12 miles from my home, it was actually (much) cheaper to drive over there than to make a telephone call. Sadly, I didn't have my own car and the "family car" was generally busy doing something else. Public transportation? Non-existent. (This is the rural United States, after all!) Another cost associated with use of the UNIVAC was the actual charge for using the computer – six cents per CPU second and some fraction of that for overall connect time. They also charged for disk data storage, aggregate I/O operations, and core-sets used by your program. Computer access was a valuable commodity in those days and they charged accordingly.

These were not unique circumstances I found myself in. It turns out that the problem

of making that connection to a remote mainframe was the reason that the Internet was ultimately invented. Yes, a form of the Internet existed in 1981 but you didn't have legitimate access to it unless you were doing business directly with the government. There weren't many systems online yet, but the number was growing.

A more immediate solution for businesses in that period was Tymnet. This was a private data network consisting of minicomputer nodes in various U.S. cities connected by "high speed" (generally 56kbit/sec) leased lines to form a private wide-area network. Tymnet offered a "local" (as opposed to long-distance) dial-up modem number for most cities in the United States. They first began offering online access to time-sharing services in 1964. No, that's not a misprint!

Tymnet users connected to remote systems using a simple command-line interface. Users would dial into Tymnet and then interact with "the supervisor" to establish a connection with a remote computer. Once connected, data was passed to and from the user as if connected directly to the distant system. The advantage here was the lack of any significant long-distance charges. The communications cost was generally borne by the owner of the computer you were connecting to.

Tymnet was extensively used by large companies to provide dial-up services for their employees who were "on the road", as well as businesses with far-flung offices like travel agencies, realtors, and florists. If David Lightman (the hacker protagonist in WarGames) had dialed into the local Tymnet node instead of using his sequential dialer, he would have found his final destination much more quickly!

A hacker exploring Tymnet in 1981 would have found all sorts of interesting systems – many without even the slightest bit of security beyond a username / password combination. A typing error I made while connecting to a nearby university (beyond Edinboro) once connected me to the load dispatch system for a trucking company. Intrigued by this, I was quickly able to locate a related system that could have been used to easily re-route a railroad car (or cars) to a siding located 100 feet from my house. Grand theft train anybody? I was actually afraid that this might have been a "honey pot" intended to trap hackers intent on mischief. In retrospect, I'm pretty sure that it was just an unsophisticated system that was depending on "security through obscurity." Needless to say though, I did not pursue the matter or re-route any trains!

The legitimate connections available through Tymnet included early online service providers like The Source, CompuServe, Genie, Delphi, MCI Mail, Prodigy and America Online. The availability of broadband Internet connections ultimately made the service provided by Tymnet redundant. The last Tymnet nodes were shut down for good in March 2004 after forty years of service.

DESTINATION 2: ONLINE SERVICE PROVIDERS

While university-based timesharing computers were typically used by the faculty, staff, and students of that university, the model of a "data processing utility" was something that appealed to many organizations that lacked the money, expertise, and space for a mainframe computer of their own. Early online services were provided by computer service bureaus that offered remote access to a computer and appropriate software for a particular business genre.

Perhaps the original example of this is ADP (Automatic Data Processing, LLC) founded in 1949 by Henry Taub of Patterson, New Jersey. ADP began providing outsourced payroll services using tab-card machinery and moved into the era of modern electronic stored-program digital computers in 1958. By 1962, ADP was providing online access (using Teletype equipment) for high-volume brokerage services in the financial sector. In 1972, ADP began providing services to automobile dealerships. In 1975, with the purchase of Cyphernetics Corporation in Ann Arbor, Michigan ADP moved into the generalized computer-service business.

A natural evolution of the computer service bureau was a shift in focus from specific industries and tasks to a more general information-services platform. CompuServe Information Services (CIS), began in 1969 as the data-processing subsidiary of Golden United Life Insurance. By 1978, it was offering its services to users of the Radio Shack TRS-80 family of microcomputer systems as "MicroNET" and in 1979 began providing online services to the general public under its own name. Derided by its own management as "schlock timesharing", the service was nonetheless very popular, albeit expensive. Although not nearly as pervasive as the Internet would later become, CompuServe could attribute half its operating revenue to its consumer network operations by 1987. Like Tymnet, CompuServe would eventually be killed by the widespread availability of broadband Internet service. "Classic" CompuServe was taken offline on June 30, 2009.

Also commencing operations of its consumer network in 1979, The Source, was a similar online service provider that offered news, weather, stock quotations, a shopping service, electronic mail, various databases, online text of magazines, and airline schedules. Like CompuServe, The Source, was an expensive luxury item – charging upwards of $20 per hour during daylight hours. The initial registration fee to get online was $100 for the software, manuals, and a username. The hourly rate at night (after 6:00 PM) was $7.75 with a $5.00 per hour surcharge if you used a "fast" (1200 baud) modem. Putting these costs in context, I was earning $3.50 per hour in 1981 and was pleased as punch that it was 15 cents more than the $3.35 minimum wage offered by most employers in the area. Clearly, the expense of an online service like The Source, was beyond the means of the average computer hobbyist.

DESTINATION 3: BULLETIN BOARD SYSTEMS

The early computer bulletin board systems are somewhat hard to describe – particularly to a generation that takes streaming video, MP3 downloads, and broadband access for granted. The bulletin board systems, and the culture they represented, were as diverse as the system operators that constructed and operated them.

First of all, unlike the commercial online services or the university mainframes, the overall tone and content of a bulletin board system was entirely dependant upon the system operators own personality and technical prowess. They ranged in hardware complexity from tiny Commodore 64 systems running off a single floppy disk and maintained by a 12 year old computer prot√©g√© to hardcore mainframe computers operated by MIS professionals. The amateur BBS systems were generally hardware specific – Apple II machines went to Apple II BBS Systems, Commodore machines went to Commodore BBS Systems, CP/M machines went to CP/M BBS Systems. Many of the cross-platform boards were run by professionals; the Byte Information eXchange (BIX) was cross-platform, but had the backing of a major technical publication. Here, the content was strictly regulated by the publishers.

The first BBS I can recall hearing about was the Community Memory Bulletin Board System. This was established in San Francisco in the early 1970s and consisted of an SDS 940 mainframe computer connected to a public teletype located in a Berkeley record store. Although originally conceived as a public information resource for the Berkeley counter-

culture, it soon became the prototype for many online user paradigms – email, newsgroups, a virtual "flea market" that presaged eBay, computerized personal ads that were remarkably similar to modern day online dating, and even SPAM created by purveyors of illicit narcotics and the sex trade.

The original Community Memory was shut down in January of 1975, but inspired a number of follow-on incarnations that helped establish our online culture.

The first microcomputer BBS I'm familiar with was the Computer Bulletin Board System (CBBS) created by Ward Christensen and Randy Suess in early 1978. Operating out of Chicago, Illinois, the system went online on February 16, 1978 running on a homebrew S-100 microcomputer with homebrew software. The pair documented their efforts for an article that appeared in the November 1978 issue of Byte magazine.

The purposes and content of a computer bulletin board system were as varied as their owners; some were strictly above-board and dedicated to a particular topic – I visited boards that were operated by "Youth for Christ", amateur radio aficionados , model train buffs, and hardcore computer science types. Others were darker and mostly concerned with software piracy, PHREAKING, and other "black hat" activities. There were BBS systems dedicated to erotic literature, general anarchy, and the drug culture. One memorable BBS opened with an image of a nude woman committing an unspeakable act. The image was rendered in glorious green-on-black ASCII art; sized for the 40-column UPPERCASE ONLY display of an Apple II Plus. Guess which boards got the most traffic from the adolescent male demographic?

Bulletin board systems reached their apex in the mid 1990s when many began offering access to the nascent commercial Internet. Although some BBS systems were the first to provide this service to the public in many areas, the commercial ISPs eventually dominated the dialup market. Widespread availability of broadband Internet access in the early 21st century killed off most of the remaining dialup BBS systems. There are a few lonely survivors, but the BBS culture has largely disappeared. If you missed the BBS era, much of the flavor of early BBS culture has been preserved at the website http://www.textfiles.com/.
Final Thoughts

The online destinations mentioned in this article are ones that I personally found useful, or at least intriguing, during the first few years of my online experience. I know that other people did strange and wonderful things with their modem that I could never fathom; like sending faxes to each other and checking their bank balance. I hope that they write it down, put it out on the web, or publish it in a magazine somewhere. The online culture that exists today is one that was created by us – if we are to claim our place in history, we must first write the history we claim.

A BRIEF LOOK AT GRAPHICS IN BASIC

BY JEFFREY ARMSTRONG

INTRODUCTION

Many people first learned the art of programming via BASIC, the Beginner's All-Purpose Symbolic Instruction Code. This normally interpreted language was designed initially in the mid 1960s for teaching the concepts of programming to people who were not coming to computers from the world of computer science. While many versions of BASIC show similarities, BASIC can be more safely considered a syntax. Little differences exist in conditional statements (IF...THEN...), loop structure (FOR...NEXT...), and jumping (GOTO). The various hardware manufacturers tended to also add variations to support their unique hardware platforms, creating command sets that differed from machine to machine.

A selection of these BASIC dialects allowed access to the computers' integrated graphics hardware. The command sets for access to graphics could vary wildly, as personal computers' capabilities could likewise vary wildly. This article explores a small subset of BASIC dialects to better show graphics capabilities accessible from BASIC interpreters.

THE DIFFICULT BUNCH

Due to limitations in 8-bit consumer machines, BASIC interpreters often made graphics access difficult or impossible. The most popular personal computer, the Commodore 64, is a notable member of the "difficult" category. The C64 featured the Microsoft BASIC interpreter stored in ROM and adapted to the C64's unique hardware. However, no explicit graphics commands are included with this interpreter.

The C64 did include the powerful PEEK/POKE commands present in many interpreters. In simple terms, these commands allow the programmer to read and write arbitrary memory locations. Special combinations of data and addresses combined via the POKE command could gain access to the C64's graphics hardware. However, specialized POKEs do not exactly constitute a pure-BASIC graphics command set, the true purpose of this article.

APPLE II

Apple delivered a selection of BASIC interpreters for its machines, and two popular interpreters, Integer BASIC and Applesoft BASIC, are discussed here. Integer BASIC was developed by Apple itself and featured only integer math, while Applesoft BASIC was a joint product of Apple and Microsoft now providing floating point math.

Integer BASIC is the eldest of the two and while the mathematics of the language were limited, Integer BASIC did include a rudimentary set of graphics commands. Integer BASIC could directly access a low-resolution graphics mode on the Apple II hardware, allowing the user access to a 40-by-40 pixel mode featuring sixteen colors. Graphics could be enabled via the GR command. The following table outlines the graphics commands available:

Command	Description
GR	Enable graphics
COLOR= c	Define the current drawing color
PLOT x,y	Plot a single "pixel"
HLIN x1,x1 AT y	Create a horizontal line
VLIN y1,y2 AT x	Create a vertical line

The oddest feature of Integer BASIC's graphics language are the line commands; functions only exist for drawing horizontal and vertical lines. Considering the resolution, though, the lack of more general line drawing may be forgivable.

The Applesoft interpreter added significant features beyond Integer BASIC, but the additional graphics commands are of particular interest. Specifically, Applesoft allows access to a higher-resolution graphics mode via the HGR and HGR2 commands. In high resolution mode, the screen measures 280-by-160 pixels, and sixteen colors are available from BASIC. The command set is simpler than the corresponding low-resolution set:

Command	Description
HGR	Enable graphics with four lines of text (for interactive programming)
HGR2	Enable full-screen graphics
HCOLOR= c	Define the current drawing color
HPLOT ...	A multi-use drawing command

The HPLOT command is surprisingly versatile for drawing under high-resolution graphics. Nominally, it may be used similarly to the low-resolution PLOT command, specifying a single pixel to plot. However, the HPLOT command may also be extended using "TO" to provide far more functionality. For example, to plot a line from one arbitrary point to another, the HPLOT command can be used as follows:

```
HPLOT 0,0 TO 20,20
```

Furthermore, the initial point can be excluded, meaning a line will be plotted from the last plotted point to the specified point as follows:

```
HPLOT TO 20,30
```

Many BASIC dialects offer merely one syntax or the other. Later Microsoft dialects notably implement the two-point solution via the LINE command, whereas Atari BASIC, for example, is more similar to the point-by-point specification.

The Applesoft command set is still somewhat limited, with no fill commands or built-in shape primitives. However, this interpreter was relatively early, and its functionality exceeded many of its peers. Furthermore, the HPLOT command appears to set the standard for line drawing in later Microsoft interpreters.

ATARI 8-BITS

Atari BASIC is a powerful interpreter for Atari's 8-bit home computer. Unlike many of its competitors at the time, Atari BASIC is not based on Microsoft's interpreter. Because Atari BASIC is somewhat unique, several differences exist between its syntax and other interpreters. In particular, string handling deviates from the quasi-standard created by Microsoft interpreters. The graphics command set, however, is the true focus of this article.

The Atari 8-bit line of machines, specifically the 400/800, the XL series, and the XE series, featured a large number of graphics modes relative to many other early personal computers; eleven distinct modes are accessible via BASIC on the XL and later machines, for example. The highest resolution available from BASIC via the GRAPHICS command is 320 pixels wide by 192 pixels tall, slightly higher than the Apple II series (although the Atari mode is monochrome). Most graphics modes by default on the Atari contain a four line text window, similar to Apple's HGR mode. Using trickery, the four lines can be removed by adding sixteen to the graphics mode number. For this reason, many example programs begin with:

```
GRAPHICS 3+16
```

The example above enables graphics mode three, which features 40x20 pixels, but allows the use of an additional four rows of pixels by eliminating the text window.

The color specification command, SETCOLOR, uses an odd syntax to define colors on the Atari. This command accepts three integers. The first argument specifies a color register, of which there are five. Although the term "color register" may seem to imply only five colors are available, they actually refer to components in different graphics and text modes. For example, in graphics mode six, the registers are defined as follows:

Register	Description
0	Foreground color
1	Color of text in the text window
2	Background color of the text window
3	Not used
4	Background and border color

While the table seems simple, the meanings of each register changes based on the graphics or text mode in use. The second and third parameters in the SETCOLOR command specify the color (hue) and brightness (luminance). The term "hue" can be misleading as it is not exactly equivalent to a sliding hue scale one might assume; rather, each value, from zero to sixteen, of "hue" simply indexes a predefined color. The brightness, or luminance, term behaves exactly as one expects.

The sheer number of graphics modes available and the odd color definition syntax preclude their discussion in this article. Programming graphics on the Atari almost requires reference materials as the graphics modes and SETCOLOR command options are not exactly obvious to the casual programmer. The Atari graphics command set, aside from the assortment of graphics modes and color options, is similar to the Applesoft equivalent, featuring setting colors (COLOR), plotting a single pixel (PLOT), and line drawing to a specified point (DRAWTO).

The COLOR command is delightfully misleading in Atari BASIC. Notably, the single parameter accepted does not refer to a register index as the SETCOLOR command's first parameter does. Instead, the COLOR command specifies a color index, some of which are configurable via the SETCOLOR command. For example, the command "COLOR 1" in graphics mode five refers to the color stored in the zeroth register. The color index definitions also change meaning with different Atari graphic modes.

The PLOT and DRAWTO commands are fairly obvious. Both are somewhat equivalent to the functionality rolled into Apple's single HPLOT command. The Atari command set, like the Apple equivalents, does not implement any graphics primitives. Some filling can be

achieved using Atari's low-level XIO command, but its use is far beyond the scope of this description.

IBM PC AND EARLY CLONES

Many people often overlook the fact that IBM's original personal computers shipped with a BASIC interpreter in ROM. The first revision is referred to as Cassette BASIC since the interpreter only had access to the PC's cassette port for data storage. This ROM-based interpreter was normally enhanced using either Disk BASIC or Advanced BASIC (BASICA) after booting to DOS. The PCjr, IBM's odd and arguably unnecessary attempt at a home computer, shipped with Cartridge BASIC. All these interpreters, developed by Microsoft, were able to utilize the Color Graphics Adapter (CGA) if available.

While the IBM PC is certainly remarkable, many people were introduced to 16-bit computing on the plethora of clones and pseudo-clones that followed the IBM's introduction. Clones could not include a BASIC interpreter in ROM, lest they incur the wrath of IBM's legal department. Instead, Microsoft developed GW-BASIC, which became the default BASIC interpreter on MS-DOS machines for years.

GW-BASIC's graphics command set drastically exceeds the others discussed here. The relatively large amounts of memory available and the lack of ROM size limitations allowed Microsoft to implement far more graphics commands than were present on their 8-bit offerings. Early IBM-compatibles featured two graphics modes on computers with CGA cards, namely 320-by-200 pixels in four colors as well as 640-by-200 pixels in monochrome. Because GW-BASIC was disk-based, Microsoft was able to add additional modes as hardware advanced.

BASIC on the IBM PC introduced the LINE statement. The syntax is reminiscent of HPLOT in Applesoft, but less flexible:

```
LINE (1,10)-(10,10)
```

GW-BASIC and its predecessors did not include the equivalent of HPLOT TO from Applesoft or DRAWTO from Atari BASIC. This exclusion is forgivable due to Microsoft's inclusion of the CIRCLE command, fill commands, and, surprisingly, the DRAW statement.

The DRAW statement is a shocking inclusion in GW-BASIC as it nominally implements a simple turtle graphics language. People familiar with Logo should feel comfortable drawing figures under GW-BASIC using DRAW. For example, to draw a simple 100-by-100 pixel rectangle, the following command will work:

```
DRAW "U100; R100; D100; L100"
```

GW-BASIC also supports a primitive animation via the powerful GET/PUT statements. As one might guess, GET reads graphics from the screen, while PUT draws data (normally from a GET) back to the screen. While these commands may seem only marginally useful, the PUT command's assortment of drawing modes bring flexibility. Specifically, the PUT command supports XOR operation, meaning that PUT-ing the same data twice actually erases the image, returning the background to its original state. Thanks to the architecture of CGA (and later) graphics, memory operations of this type are relatively fast even on the early IBM offerings and their clones.

TANDY MODEL 100/102/200

The versatile line of Tandy laptops are interesting machines in many respects. The laptops feature Microsoft-designed software suite, an 8-bit processor, and a BASIC interpreter that almost resembles GW-BASIC. Obviously the interpreter has been stripped down to live in the limited ROM space of the Tandy laptops.

The most charming feature of graphics within BASIC on these machines is that, rather than having to switch to graphics mode, graphic commands can be used in the default mode. Furthermore, the fact that the display is monochrome simplifies the language, eliminating color-related commands. The interpreter implements three commands, namely PSET, PRESET, and LINE. The PSET command turns a pixel black while PRESET unsurprisingly does the opposite. The LINE command acts entirely the same as the equivalent LINE command in GW-BASIC. The simplicity of this tiny command set is reminiscent of Integer BASIC.

CONCLUSION

The graphical command sets under BASIC on four different platforms vary significantly considering all the interpreters are implementing the "same" language. When people state they have written a graphical game in BASIC, the hardware platform is always an interesting aspect of the design, as often such development efforts can't simply be shifted to another platform. If a platform features a graphics command set in BASIC, the code related to graphics is usually understandable even in unfamiliar BASIC dialects. But we should all admire those who have attempted graphical BASIC programs on the Commodore 64.

300 BAUD

THINKING ABOUT INTERFACING

-OR-

STICKING SOMETHING INTO THE HOLE AT THE BACK

BY DALE GOODFELLOW

How does it work? What exactly goes on at that edge connector? and can I make it do anything? Mostly I have been kinda scared of the edge connector. I stuck things on it – games cartridges and RAM packs – but that was it. I knew that it did something but I had no idea how it did it. Address or Data Lines? MREQ? HALT? WAIT? Ummm.....!

I don't program, you see. I can, if pushed, knock up a ten or twenty line BASIC program that wont do very much, and if you hold a gun to my head then I may be able to hack someone else's 20 line program, but that's about it.

I like old computers and I like to play. That means running someone else's software or playing with the hardware. Now I am new at this and I have precious few achievements. I have made a data loading thingy by stripping a Commodore Datassette to its bare parts, so I could load audio files direct to my PET; I have made a really crap drawing arm for my Dragon 32; I even hacked up an old PowerBook 160 to make a telnet terminal that would show info sent from the StarTTY telnet service – I have done 'stuff'. I just want to do more, deeper down, lower down into the hardware. I mean, just what the hell do those little black rectangles actually do?

A wee while back I decided to see if I could figure out a way to get my ZX81 to take an input through the EAR port and create an output through the MIC port. I wanted to hack, but as I said I had a bit of a fear about that edge connector. I was using some plans from a magazine, changing some code along the way, and after much head banging and failing to grasp Z80 assembly I called it quits. Really though, I was tackling things in a pretty arse-about sort of way, and it's no surprise I failed. The manufacturers give you a dedicated IO port on these machines for a reason after all.

Time moved on...

One day I found online some scanned pages from the "PET User Port Cook Book" and I thought I would give this 'interfacing' a second go. It seemed to make more sense this time, so I thought would share the knowledge.

Now, I should say that everything from this point, I would guess, is real baby step stuff to the true hackers out there, but baby needs to learn to walk and this might help a little, if anything its really just to show that virgin newbies like myself can mess around at this stuff too – just less impressively than he hardcore hacker types.*

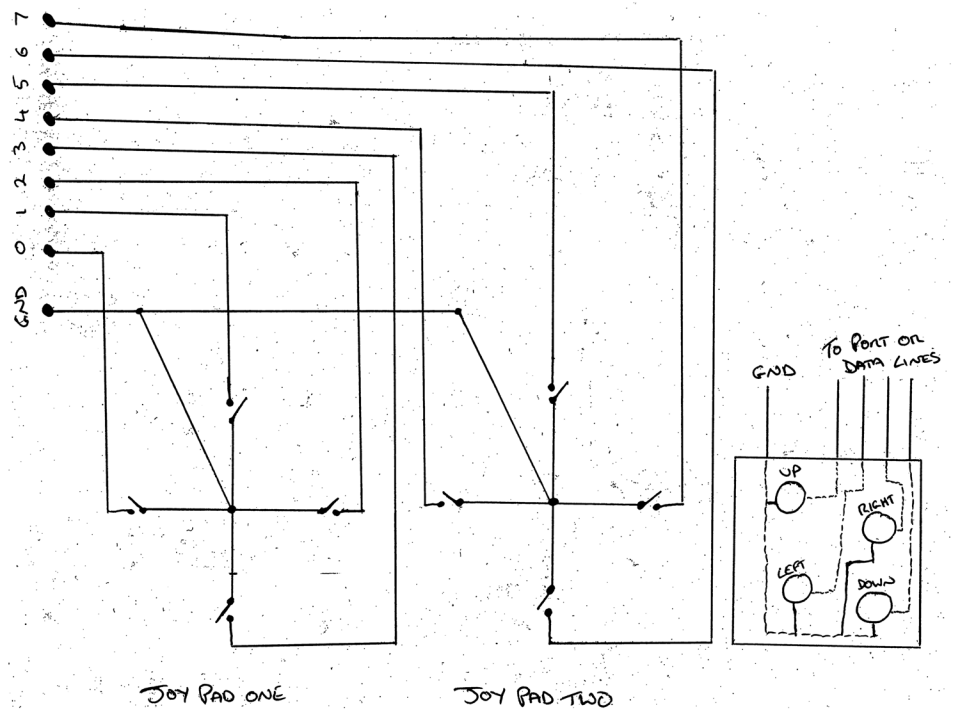
**(a virgin figuratively, of course; as you know how the ladies all love an over weight, pasty-white geek)*

Let's start with some really simple concepts (for that's all I can do!)... The edge connector has little gold bits on it – teeth perhaps; anyway, these little gold teeth are the ways into and out of the computer. Information goes in and information comes out – but what information?

Well, we are talking computers here, aren't we? And computers are digital. And digital devices, when you get to the bare bones of it, operate with this binary stuff – you know, ones and zeroes – on or off.

Now this used to confuse me – kinda still does; but we know that (because we are told) you can represent anything through a series of ones and zeroes, what I always found hard to visualize, and hence move forward from, was how this could be actually made to do anything. I knew it could, I just didn't understand what happened and how.

Well the answer to this is a tricky thing called a 'program'. Say it with me 'P-R-O-G-R-A-M'. Yes that's right, a program. No magic, just a program. Sometimes you write the program, other times it's written already and sits on the chips in the computer and sometimes it's both, in that you write a program and it also includes bits of the programs on the little black multi-legged rectangle thingies inside.



THIS SIMPLE DIAGRAM SHOULD CLEAR UP ANY CONFUSION...

Let's say I want to output the letter Z, well the first question is output to what? That's the question. The computer will happily output the required ones and zeroes, but if the output 'doo-hicky' doesn't understand the code, you are not getting a Z anytime soon. Similarly going in. I am going to have to develop a code for the letter Z, perhaps "01011010" (this is the binary for Z – notice that it's made from eight characters – more later) and the program in the computer will need to understand this code. Now lots of this code has been done for us way back when. We have a binary representation for each letter, both upper and lower case and, importantly for human ease, each of these has been further codified into the ASCII codes with which you are no doubt familiar – i.e. PRINT CHR$(90) will print the

upper case Z, in most flavours of BASIC. So these basic principles are everywhere, and are intrinsic to the little box you play with, we now just need to harness it a little – just need to decide what we want to put out or bring in and how we are going to deal with it; deal with the ones and zeroes; deal with the 'code'.

So these ones and zeroes are ONs and OFFs. Or as is often said, HIGHs and LOWs. But on and off or high and low what?

Voltages!

Yes nice and simple voltages. Get a multi-meter – go buy one, they are dirt-cheap and they really help in visualizing this stuff if you are more practical that math minded.

With regard to this topic then, in a general sense it looks like this: We are likely to have a machine that when you get inside, shoves things around based on a start point of around +5v. If we have, for example, a voltage in the range of say +2.5 volts and above it would be classed as "high" and +2.4v and below would be classed as "low". This high and low creates our ones and zeroes.

If you have a multi-meter you can for example measure each of the gold teeth on your edge connector to see if it is 'high' or 'low' or changing between both states. Do be careful you don't short anything when doing this – know your ground and keep a steady hand – it's only risky if you are dumb (I have personally destroyed my PETs 6522 VIA IC doing such a thing). You will also notice that you have more than one gold tooth on your edge connector, many of these will do specific things that the computer needs to drive printers and external disks and some of these teeth may also directly connect to the CPU or other particular ICs, whatever the actual configuration (your manual will tell you), some things, regardless of the machine, will hold true.

There is likely to be a +5v tooth or line (handy for powering things as long as its light-weight – a few LEDs or a peizo buzzer for example; a corresponding ground -GND- or 0v line (you cant have a circuit without + and – after all) and importantly for us, eight lines to allow us to get information in or out. These tend to be numbered from 0 to 7 and tend to be labeled in some way as DATA lines; each of these is a 'bit'. There are other things on the edge connector, but at the level we are discussing here we can mostly ignore them. Saying that, depending on your machine you may also need to become familiar with eight additional ADDRESS lines – that however is your problem, and one I am currently trying to tackle with my ZX81... and have yet to solve.

So you can input or output eight bits at a time, which means we have an 8-bit word or byte, which is just the right size for your 8-bit machine; yes it's all linked!

*We have 8 lines and each line can go high and low. So that means that I could have 00000000 as 'off' or white screen or no sound or, or, or... and I could have 11111111 as 'on', a pixel or boom! It also means that there are a fair few patterns between both of these values, such as 00001111 or 10101010 or 00100000 or... In fact you will have 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2, in other words, 2 to the power of 8, or 256 variations – 0 to 255. Meaning that we have a fair degree of flexibility. Imagine you were inventing a language; in that language you had the ability to invent 256 words, give it a pause and you realize that there is a hell of a lot you can do with 256 *something's*. Its only dependent on your imagination, programming skills and focus of the task.*

I suppose all that's left now is a measure of control, how do we control this, how do we get the bits to turn on and off? There are two simple ways, at least as far as we are concerned here.

As far as output is concerned then, it is all about POKEing values to the data lines – your 0 to 7 gold teeth on the edge connector (again, your manual will tell you which teeth correspond with which line from 0 to 7 and also what the POKEs are to switch them on and off). Google, of course, is also your friend.

Depending on what you have attached to the line will depend on what you are going to make it do. At it's very simplest you could make a light or beeper go on or off. The chances are the +5v you will find on your edge connector will be able to power this, although it may be an idea to power it separately – especially if you plan to build and play further. From the skilful program you have managed to hack together you switch all the lines or a single line on (or high) and then off (or low) – Flash / Beep! Easy. It obviously gets complicated when you consider that with 256 variations you can do more than just switch a light on or off, you could make a pretty good light show with sufficient know-how.

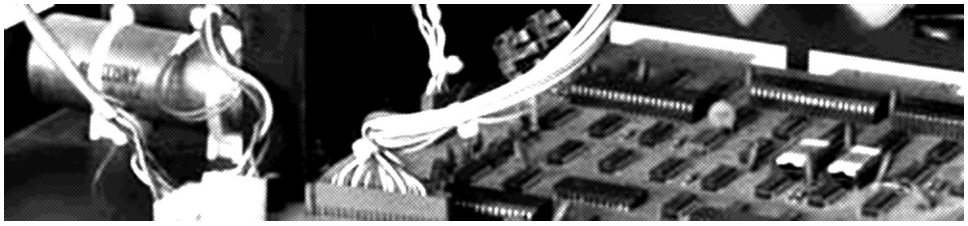
As far as going into the machine, it gets a little more complicated in that you have to remember that the computer is going to be expecting something in the form of an 8-bit word, while the human world tends to be a little bit analogue, so you will have to either start at that point or find a way to use an analogue to digital converter. Their use is well out of the scope of this article, perhaps next time, but single ADC ICs can be bought for just such a purpose and would allow you for example to translate an analogue room temperature (just an idea!) into a form the computer would understand. What you can do now, however, is turn each line on or off – staying digital!

I managed to make a very clunky and amateurish 8 key piano this way – although only seven of the keys actually worked on completion – all that I was doing was sending each line high by moving it from ground, in that I moved a physical switch and the line was then disconnected from the GND line sending it high. This alters the value of the 8 bit word. I then PEEKed at the value – and gave that value a 'sound'.

You could similarly make a joystick or joy-pad to control a cursor in the same way, but would, of course, only need four lines – one each for up, down, left and right; plus a fifth for 'fire', I suppose, oh and a sixth for warp and the smart bomb could be seventh line, and the eighth? Well I can't do all the thinking for you!

To close I will say, that as poorly executed as my 7 key piano was, prior to this edge connector playing around, never would I have thought that I had the skills to make it happen. This sort of hardware stuff was for true hackers, and indeed making a Bluetooth module for an Apple 2 still is, but for the rest of us, attaching simple circuits and projects to our retro machines just takes a bit of reading, a few tools and the desire to do it. If I can manage to interface some switches to my PET and then use them to make sounds, I assure you, anyone can.

Happy hacking!



300 BAUD

HACK THE MAC

AN INTRODUCTION TO MACINTOSH PROGRAMMING TOOLS

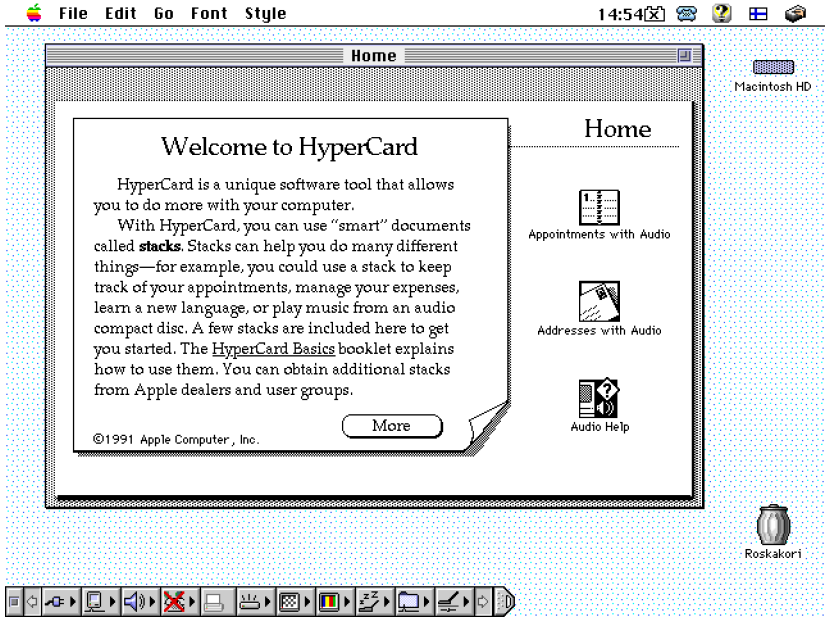
BY PEKKA SARKIMO

There are many different different programming languages and environments for 'Classic' Macintosh computers. Classic Macintosh means here, machines with 680x0 processor, some may have other definitions, but this is mine. Some of the tools discussed here will also work well on the PowerPC processor should you be so inclined.

This is not a complete list but there should be something for everyone; for beginners and advanced programmers and on another axis something for the lowest end machines and something for the higher end machines.

HYPERCARD

Data stack and script based Hypercard was a program produced by Apple for creating object oriented interactive applications. It was released in 80s for free with the original compact Macs. Later there was also a commercial version with some more advanced technologies included.



Hypercard programs are actually documents called stacks, which are played by the HyperCard application itself. The editor allows the user to create graphics and interfaces with common graphics and objects such as text fields and buttons. Objects can be scripted using the very english-like and beginner friendly HyperTalk scripting language.

The original Hypercard requires only 700 kilobytes of ram and 400 kilobytes of disk space. Because of its age it does not always run properly on the newer color Macs but its a great and pretty powerful toy for Macintosh Plus.

MICROSOFT BASIC

The Basic language has been a traditionally good option for people starting programming. Microsoft Basic for Mac is about as old as the Macintosh itself. It offers a text editor for writing and editing programs and a runtime environment, all in one application. It is not able to compile stand alone applications however, instead, as is the usual way the basic programs are loaded and run from within the editor.

Microsoft Basic provides access to many of the features of the early Macintoshes such as serial communication and both sound and graphic routines. Advanced basic hackers can even develop methods to produce color graphics.

Version (v2) works well on most low end macintoshes and has a 24 bit memory addressing requirement. As a high level programming language Basic is not as fast as C or Pascal when running, but still provides a fun way to explore Macintosh programming.

THINK'S LIGHTSPEED PASCAL

In the beginning Pascal was intended to be the native programming language for the macintosh. Lightspeed Pascal is one of the highest quality Pascal compilers available for the Mac.

The user interface is attractive and initiative. The text editor supports some syntax highlighting and in compiler options there are settings for different kinds of project types such as desk accessories and device drivers.

Lightspeed Pascal requires one megabyte of ram, 850 kilobytes of disk and a 24 bit memory addressing mode. There are newer versions of Think Pascal for the newer 32 bit Machines. The version described here works best on a memory upgraded Mac Plus and similar spec'd machine.

SYMANTEC THINK C

Think C is similar to Think Pascal but for the C programming language. Later versions also included the C++ language.

The project manager is very similar to Think Pascal but for some reason the editor, options and debugger are bit more sparse compared to the Pascal version. It compiles fast but not always providing the most optimal machine code. Its possible to write inline assembly where performance is critical.

Think C includes a couple of example projects, the software toolbox reference and the standard C and Macintosh toolbox libraries.

Think C takes a megabyte of ram and 5 megabytes of hard disk. It works well on any 68k machine meeting those requirements.

METROWERKS CODEWARRIOR

CodeWarrior is one of the best development environments for the 68k Macs. It has been

released over the years in several different versions. The earliest versions offered C and Pascal compilers for Macintosh programming. Later versions included Java and later additional platforms such as Windows and BeOS were provided for.

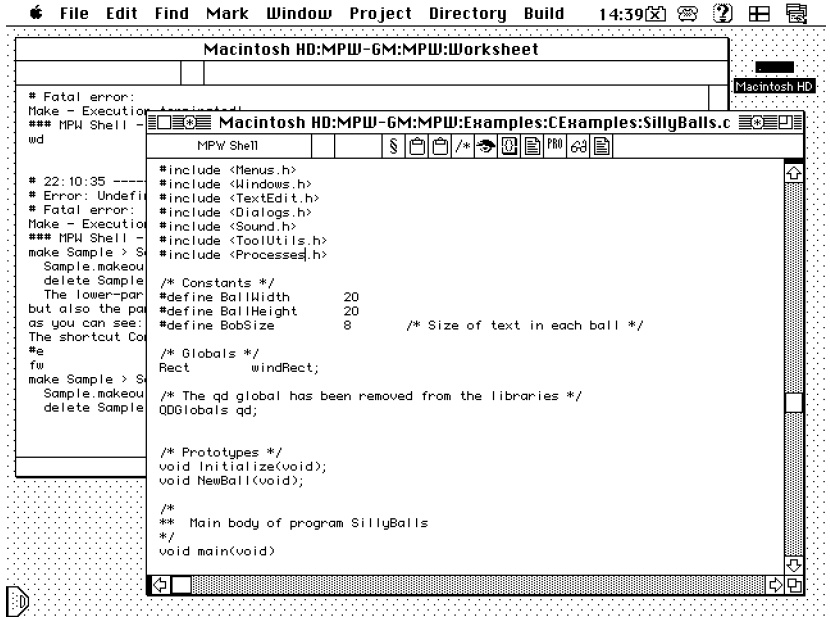
The complier is pretty fast and produces reasonably well optimized machine code. The later versions even support Apple's universal programming SDK which allowed the addition new technologies when they were published by Apple.

The editor supports syntax highlighting, has great search functionality and the compiler has a rich set of options.
After few days of working with CodeWarrior its almost misery return to the modern tools!

Machine requirements depends on the version. The latest version (Pro 5) with 68k support requires 8mb of ram and good amount of hard disk. With a generic Quadra or similar it should run well.

APPLE MPW

Macintosh Programming WorkShop (MPW) is the programming environment create by Apple. The latest version supports assembly and C programming for 68k and PowerPC platforms.



MPW is not the easiest tool to learn because it is similar to a command line shell with a set of tools, that to use are like the Unix and Linux make/gcc toolchain. However it does have a graphical user interface for creating project files so that a brave programmer can get some examples compiled in under one hour inspectioning.

Apple does offer detailed online documentation for the MPW shell and for building tasks. Additionally the MPW installation does includes a decent set of examples for learning.
The latest version takes about 70mb of hard disk space and 2mb of ram. The MPW is freely available at Apple's ftp server.

LIGHTSOFT FANTASM LIDE

Assembly is the only realistic choice for real hairy chested programmers. Fantasm LIDE is a very advanced assembly programming environment for the macintosh. Additionally it supports the C language if it also has access to the MPW complier tools.

The editor is close to CodeWarrior in quality and contains a rich set of tools including even a MOD music player. Fantasm is not the easiest tool to learn but there are examples and documentation to help in the initial steep learning curve.

Fantasm requires 4mb and around 55mb of hard disk space. Disk requirement can be decreased by removing unwanted libraries and tools. An average color macintosh should run this without trouble.

Fantasm LIDE is available for free at LightSoft.

WHAT ELSE?

In addition to the programming environment the Macintosh has some practical tools for creating applications.

MacBugs is a low level debugger which activates automatically when the program crashes to the bomb-dialog. It is often able to tell what went wrong and why and crashing program can be terminated without restarting the computer. Its not the most user friendliest of programs but it does save a lot of time at its best

Typical applications store their data in the file's resource fork where it can be easily accessed with toolbox calls. ResEdit is an editor by Apple for editing the resources and it is almost invaluable in any serious development.

Where to get the tools?

Most of the discussed programming applications and tools are nowadays obsolete and abandoned. Some of them are still freely availably from their creators. Some others may be found on ebay or in other second hand stores. Determined explorers may also find some of them listed on abandoned software sites or from secretive hotline servers.
However you get hold of them, happy coding to you all.

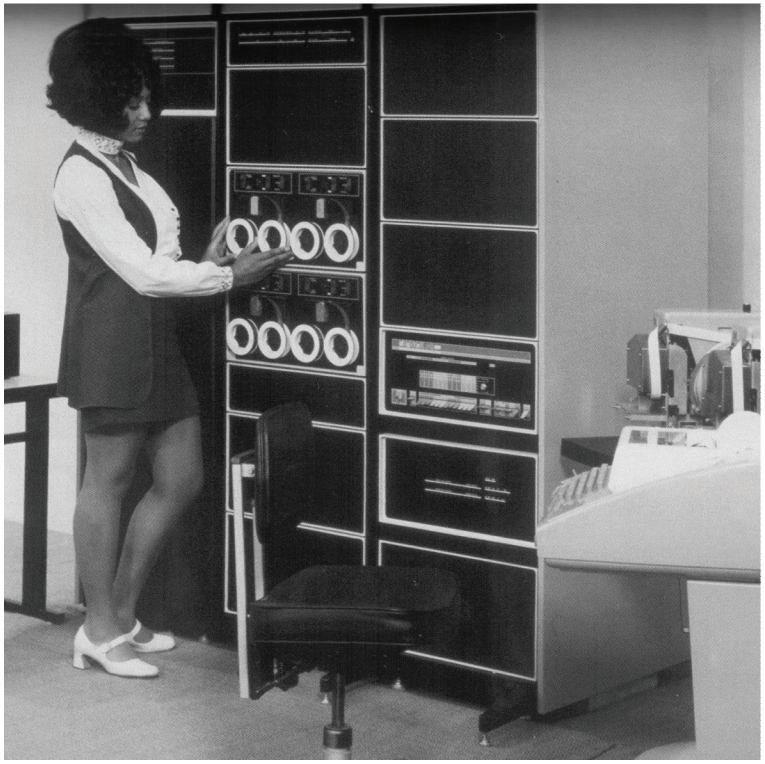
COMPARISON TABLE

	Languages	Requirements	Difficulty
Think C	C	***	****
Think Pascal	Pascal	**	***
CodeWarrior	C / C++ Pascal	****	****
MS Basic	Basic	*	**
HyperCard	HyperTalk	*	*
MWP	Asm/C	*****	*****
Fantasm LIDE	Assembly	****	*****

retro Bits

SIZE MATTERS!

Might as well get used to the notion that these big-iron beauties are just not interested in your micro!



300 BAUD

A SHORT HISTORY OF PEN COMPUTING

OR

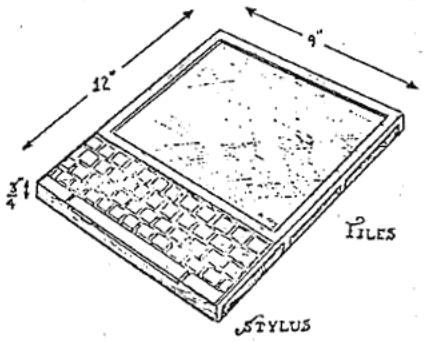
THE MORE THINGS CHANGE, THE MORE THEY STAY THE SAME!

BY DAVID GREELISH

Have you heard about Apple's new iPad? C'mon, you must have by now. It's new, new, new, and it's going to change the world! I'm being a little sarcastic, but it will change the way many use a computer, at least in some specific tasks. The iPad will be the first serious contender in a mode of computing that has been struggling to mature for almost thirty years. It will cause a paradigm shift that is already taking place in the smartphone market (more on that later). The iPad is commonly known as a "tablet" computer, but it belongs in the category that has arguably existed since shortly after the personal computer became mainstream – the handheld. The handheld falls in between the two other main categories of personal computing, the desktop and the laptop. The handheld can break down further, with many different sizes, styles and interfaces. The most common ones are the palmtop (with QWERTY keyboard), PDA (personal digital assistant) and the tablet.

ATARI PORTFOLIO

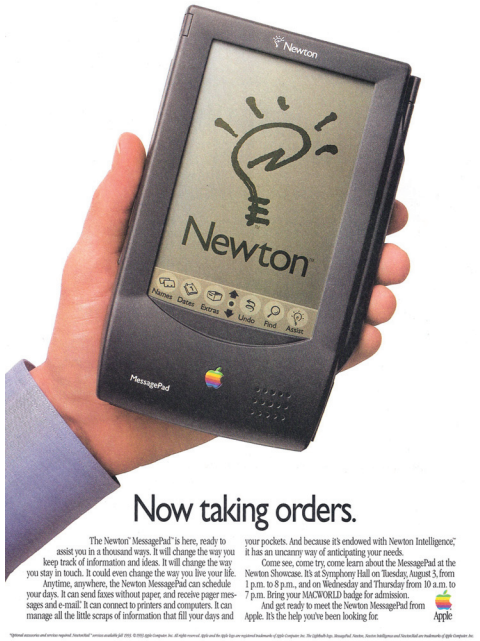
Ever since the laptop started taking off around the mid 1980s, it's life has seemed ultimately doomed, to be replaced by just a screen. A screen that could do everything a regular laptop could do, and more. In the late 1980s, there arose a small buzz in the portable computer industry, and it grew louder and louder. The buzz was about the so-called "pen computers." These systems generated a lot of excitement at the time, and it seemed inevitable that they would eventually replace conventional laptop computers, perhaps even traditional desktops. After all, they all used keyboards. It just made sense that a singular input / output device would be the way to go – the screen. Using a "pen" was the natural choice for writing without a keyboard. The pen, as a stylus, also seemed the natural choice for the interface too, particularly when graphical user interfaces were first starting to appear.



DYNABOOK CONCEPT

About twenty years earlier, before there were personal computers of any kind, Alan Kay created the concept for the "Dynabook" in 1968. This was two years before the founding of Xerox PARC, where he would go on to work. Many of his ideas led to the development of the Xerox Alto prototype, which was originally called "the interim Dynabook." This Dynabook concept paved the way for what is now known as a laptop computer, or a tablet computer in some of its other incarnations. As personal computers got faster and faster, they also became smaller, which then made the "luggable" computer possible, then the laptop. The central theme of Alan Kay's idea however, was of a book-sized device, with the power of a super computer. The technology to make his vision real had yet to be developed, but his vision was shared by others.

Keyboards originated in the 19th century with mechanical typewriters. The pen seemed the logical evolution in how people would work on computers in the future. The pen was also seen early on by many as critical to the handheld concept, especially when coupled with handwriting recognition. It was this mindset that became the "albatross" to the pen computing paradigm. In the early 1980s, handwriting recognition was predicted as the most important future technology in computing. Unfortunately, the technical reality was unable to live up to expectations, at least initially. The most notable example of this was embarrassingly demonstrated with the arrival of Apple's Newton. Or at least, that was the perception. The handwriting system on the original MessagePad was designed to recognize entire words rather than single letters as Palm eventually developed with their Graffiti system in 1994. This was because focus group data had told Apple's engineers that people wanted to write in cursive. When the system made a mistake, it would often pick a word that was different than the one intended, which lead to much ridicule. This ridicule and criticism was summed up regularly in the newspaper comic strip Doonesbury by Gary Trudeau and was even lampooned in The Simpsons. Apple corrected these problems with the Newton OS 2.0 (almost three years later), but the damage had been done. Ironically, Newton's handwriting recognition after 2.0, has been considered by many users, testers and reviewers as the best in the industry, even over a decade later.



THE ORIGINAL IPAD?

Pen computing hype reached it's peak between 1992 and 1994, as a number of companies introduced software and hardware. GO released it's PenPoint OS, which was truly pen-centric, but had a steep learning curve. Microsoft released Windows for Pen Computing, which used a layer on top of Windows 3.1, but ran all existing Windows applications. Momenta created it's own hardware and interface on top of DOS, and Lexicus Longhand handwriting recognition was released. Other hardware manufacturers were AT&T, NCR, Samsung, Dauphin, Fujitsu, TelePad, Compaq, Toshiba, and IBM.

Some of these were laptops, and others were more palmtop and tablet-like. But it was really

Apple's Newton and the PDA concept that brought pen computing and the awareness of touch interfaces into the public consciousness. The hype surrounding the Newton was incredible, as Apple did a pretty good job of PR, even way back in 1993. The Newton was quickly emulated by the likes of Amstrad, Casio / Tandy, AT&T and a few others. The hype surrounding the Newton had set high expectations, but the device did not live up to it. The "cloners" did an even poorer job in meeting the expectations of this computing future.

MOMENTA HYPE BEFORE THE MOMENTUM RAN OUT!

The PDA market collapsed as fast as it had false-started. Pen computer companies began failing, many non PDA and hybrid manufacturers included. Momenta closed, AST purchased pen pioneer GRiD for its manufacturing capacity, then stopped all pen projects. Dauphin went bankrupt, owing IBM over $40 million. GO was taken over by AT&T, and AT&T closed the company in August 1994 (this was immediately after the interesting "You will . . ." - faxing on the beach, etc. TV commercials). Compaq, IBM, NEC, and Toshiba had all stopped making consumer pen products by 1994. By this time, the very term "pen computing" seemed like a joke. The market mocked and abandoned the whole concept, especially that of handwriting recognition. Pen computing as a new interface to computing, was seen as a failed technology. For many years after, the pen, as stylus, was mostly used in the more lowly organizer category by Sharp and a few others. It wasn't until late 1997, when the PalmPilot became well known and it's sales began to grow, that this category was taken seriously again. Slowly, renewed interest and confidence in this input method grew, but in a more specific and more humbled way.

In the last decade or more, the term "pen computer" has mostly gone away, and replaced with the concept of the tablet or tablet / hybrid laptop. During this time, tablet concepts have been almost completely dominated with pen based Windows operating systems. In 2000, Microsoft showed off a prototype and said the first Tablet PCs would come to market in 2002. In 2001, Microsoft announced the "Tablet PC" standard, which was defined by Microsoft to be a pen-enabled computer conforming to hardware specifications devised by them and running a licensed copy of Windows XP Tablet PC Edition. At Comdex in the same year, Microsoft chief software architect Bill Gates said in his keynote: "The Tablet is a PC that is virtually without limits, and within five years I predict it will be the most popular form of PC sold in America." Microsoft has predicted the rise of the tablet PC on numerous occasions since, yet it has not materialized. In the last five years or so, there has been established a small market for expensive hybrid laptop tablets, where the screen turns around to lie flat and thus allow the computer to be used as a tablet.

So, what's the future for pen computing? Well, it's thriving, or it's dead, it depends on how you look at it. The idea of pen computing was like many, ahead of it's time, especially in the PDA concept. The real irony with the PDA was that it turned into the organizer, a far more limited computer, but far more successful. After reaching a refinement by the year 2000 or so, and gaining more computer-like features, it was upgraded into a phone! Thus was born the smartphone, the Handspring (Palm) Treo 180 in 2002. This development is what is

SUPPLY YOUR OWN CAPTION

driving the current push towards true handheld computing, and, in a way "pen computing." Pen computing exists now, and has a very promising future, just without . . . well, without pens! Yes, there are still those Windows based hybrids, and there are even some tablets and rugged PDA-like devices that use pens, but that isn't the mainstream now, or the future. In 2007, Apple introduced a new smartphone, and it was so incredibly ahead of everything else, that it changed the standard, in a big way. All smartphones are now judged against the iPhone. The iPhone gave way to the iPod Touch, and that has now given us the iPod Touch XL – the iPad! The finger is the new pen, and all of these devices have virtual keyboards, so apparently, the keyboard is not dead, yet. The iPhone finally has serious competition now in Android phones, and we are seeing the first announcements of Android based tablets. So this is what "pen computing" is now – multi-touch based, web enabled tablets. Not to sound like too much of an apple fanboy, but it appears that Apple truly drives the standard on all FOUR of our current computing platforms – desktop, laptop, handheld / tablet and smartphone.

There is an interesting video out on the web in which Microsoft has created a new, unique computing device. It brings three components back together – the pen / stylus, Windows (or at least a Microsoft OS) and handwriting recognition. The device is called the Microsoft Courier. That is the codename for a rumored upcoming Booklet PC. It will contain two multi-touch screens that face each other, in a book form factor, and will utilize a stylus and finger touch for input. The device is setup like a real book, hinged, with seven-inch screens. The user is able to input, drag, copy, etc. across the "pages." It's an impressive video, do a

David Greelish is a long time computer history enthusiast and collector. He started the Historical Computer Society in 1992, and began creating a newsletter about old computers in 1993, "Historically Brewed." He now runs classiccomputing.com and produces a video podcast.

THIS AD IS VERY OLD – WE DO NOT RECOMMEND ATTEMPTING TO ORDER THESE PRODUCTS!

GETTING DOWN WITH GOPHER

BY ROB SAYERS

Pop quiz: You have many documents and files you want to share with the world, what do you do?

Today most people would use their website, some sort of file hosting site or Peer-2-Peer, such as Bit Torrent or Limewire. Many years ago this task would not have been so easy. So let's take a trip back to the early 1990s and consider things from then.
Back then sharing files usually meant FTP. Anyone who has ever tried sharing files over ftp knows it can be a pain it can be for the administrator, the user, or both. Just sharing public files still required a user to login and for the admin to setup permissions for the user and even sometimes for the individual files themselves, what a pain!

Faced with this dilemma, computer scientists at the University of Minnesota worked on creating a more user-friendly way to present content to users over the Internet. In 1991 they released the Gopher protocol to the world. It's name comes from the University's mascot, The Golden Gopher. In a time when the World Wide Web existed only in a laboratory in Europe, Gopher gave users a way to easily locate and retrieve content no matter where it was stored.

For a few years Gopher was the de facto method of distributing information on the Internet. Gopher was simple in that it presented all its content in a hierarchical directory structure similar to a directory tree on a computer. Unlike the web, Gopher is not concerned with the presentation of the content at all, and leaves those details up to the client. The protocol was designed to be simple, easy to implement, and light-weight.

So what happened to the protocol?
Why is it that most people today have never heard of it?
In short, the web happened.

In the mid 1990s, the web was growing in popularity and offered many of the same features that Gopher offered and more. Like Gopher, the web made it easy to present documents to users as well as link to resources on other servers. Unlike Gopher, the web also had inline hyperlinking within documents and with html, a webmaster could control how their information appeared to the user. Another big advantage was the ability to embed media such as images, and later audio and video directly into a document. Unfortunately at around the same time, the University of Minnesota announced that it would begin charging for the use of their Gopher server implementation. As many worried that the university might try to extend it's copyright reach to other server implementations; server administrators had even more reason to flock to the web.

Within a few short years, Gopher had largely vanished as people moved their content from Gopher to the web. When I first started using the Internet in 1995, I had heard of Gopher and remember using it a time or two. But even then it was largely gone.

To paraphrase Monty Python, it's not quite dead yet, it is merely sleeping. In fact, over the past few years the number of servers in Gopherspace, the unofficial name given to the collection of Gopher servers on the Internet, has increased.

Part of the reason Gopher remains popular is its simplicity. Unlike many network protocols, it's concise and one can hack out a client or server in very little time.
Lets see how it works.

When a client connects to a Gopher server (almost always on port 70), it immediately requests a resource, the server responds with the requested information. No headers, no negotiation, or any other overhead. The protocol is so simple in fact that one can use gopher without a special client at all just using telnet:

```
rsayers@rsayers-desktop:~$ telnet gopher.robsayers.com 70
Trying 67.18.186.67...
Connected to gopher.robsayers.com.
Escape character is '^]'.
/serverinfo.txt
About this server
=================

This machine is not a machine at all actually, It's a virtual server hosted
at
...
```

The first line in the session was my request for serverinfo.txt in that particular servers root directory. When viewing a menu, it can be a little more complicated, as that is sent in a special, but still plain text, format. Menus in gopher present as a list of what is available to view, they can either be an automatically generated directory listing or made by the server admin to list only certain items.

In a menu, the items are listed on a single line with a few elements:
A single character which tells the client what type of resources you are accessing, the title of the file, it's path on the server, what server it is on, and what port should be used.

Going back to the file from the previous example, its menu item would be:

```
0About this server    /serverinfo.txt        gopher.robsayers.com    70
```

The 0 indicates that it is a plain text file, this single character saves some of the headache of handling mime type correctly, but there are a limited number of these officially defined, and many are for older filetypes not used often today. The currently defined list as listed in Wikipedia[1]:

* *0 plain text file*
* *1 directory menu listing*
* *2 CSO search query*
* *3 error message*
* *4 BinHex encoded text file*
* *5 binary archive file*
* *6 UUEncoded text file*
* *7 search engine query*
* *8 telnet session pointer*
* *9 binary file*
* *g GIF image file*
* *h HTML file*
* *i informational message*
* *I Image file of unspecified format. Client decides how to display.*
* *s Audio file format, primarily a WAV file*
* *T tn3270 session pointer*

With such a concise protocol, Gopher is very popular in Retro-computing as it allows older machines with little power an easy way to access a lot of content that is still out there in Gopherspace. Clients and servers have been written for platforms such as the TRS-80 Model 100 and the C=64 among others. As the web continues to advance, older machines and other limited devices are more and more often left without a way to get information from the internet. For this reason many people consider Gopher not as an outdated precursor to the web, but a still useful alternative.

So how to Gopher today? If you are using Firefox, you already have everything you need as it has retained Gopher support from its roots in Netscape Navigator. Simply go to any gopher site such as Floodgap at gopher://gopher.floodgap.com which is an excellent place to start.

Unfortunately Firefox has a rather basic implementation which lacks certain features such as the ability to access Gopher on any port other than 70. To remedy this, we use Cameron Kaiser's excellent Overbite extension for Firefox[2]. This makes browsing Gopher a much more pleasant experience.

While most other modern browsers have dropped support for Gopher, Lynx continues support and for that remains a popular choice. This text-mode web browser handles Gopher quite well. Since Gopher is almost all plain text anyhow, Lynx is a natural fit. If you do find yourself in a situation where you don't have access to any sort of client, Floodgap also offers a proxy[3] which allows you to browser Gopherspace from any browser. This is especially good option for web enabled phones.

And finally, if you just can't stand to sit on the sidelines and want to run your own server, there are several options there as well. The original University of Minnesota server is still out there but I recommend against using it. Most people go with either Pygopherd[4], a Python implementation, or Bucktooth[5], a Perl implementation.

Gopher usage is a fraction of what it was at its peak and has nearly been completely replaced by the web. However, as the web becomes more and more complex, many are (re)discovering that Gopher is a great alternative thanks to simplicity, ease of use, and maybe just a little bit of nostalgia.

WEBLINKS:

[1] Wikipedia - Gopher (Protocol)
http://en.wikipedia.org/wiki/Gopher_(protocol)
[2] The Overbite Project
http://gopher.floodgap.com/overbite/
[3] Public Gopher Proxy
http://gopher.floodgap.com/gopher/
[4] Pygopherd
gopher://gopher.quux.org/1/devel/gopher/pygopherd
[5] Bucktooth
gopher://gopher.floodgap.com/1/buck

MICROWARE'S OS-9

THE MULTITASKING 8-BIT OPERATING SYSTEM

BY AARON WOLFE

As a kid growing up in the '80s, the monthly arrival of the computer magazine was a Big Deal. This much anticipated event always resulted in a few days of renewed obsession with my Tandy Color Computer. Hours would be spent carefully typing in BASIC program listings, reading and dreaming about all the interesting upgrades and peripherals, and generally having a great time. Within a week's time I would read the issue cover to cover, often more than once.

Towards the back of the magazine I'd typically find a section of mysterious articles that didn't make much sense to me. They were full of strange terms, commands that did not work on my CoCo, and had grown-up sounding titles. I eventually gleaned that there was something called "OS-9", and that I didn't have it, and turned back to the articles about BASIC adventure games.

Twenty some years later I found myself interested in the Color Computer again. What began as a stroll down memory lane soon grew into a proper hobby. After playing around with some emulators, I dug out my old CoCo 3 on a trip to my parent's home. I found that it still worked perfectly and in short order I was once again a card carrying CoCoNut. Upon joining the Color Computer mailing list, I was delighted to find a great group of people still interested in these old machines.

Once again I found "OS-9" a frequent topic of discussion, but the terms and concepts made a lot more sense than they did when I was eleven. Multitasking, interprocess communication, pipes, reentrant modules... on a Color Computer, with 64K of RAM? I had no idea such things were possible, and became fascinated with this new found side of my old friend.

I discovered that OS-9 is a powerful and efficient operating system. It provides many of the same core features found in a modern operating system using a fraction of the resources. The term "Unix-like" is common in descriptions of the system. However, unlike Unix, OS-9 is quite small and the source is easy to understand. There aren't many systems that "fit in your head" as a programmer yet provide so much power. Having always wanted to learn more about the internals of an operating system, I soon found myself pouring through the excellent documentation and well commented code that can be found online.

As I studied, I tried to piece together the story of how this interesting operating system came to exist. I learned that OS-9 was created by a unique partnership between engineers at Motorola and programmers at Microware Systems Corporation. Although probably known best for its role on the Color Computer and Dragon systems, OS-9 was actually created before these systems existed and continues to be supported and developed today.

In 1978, Motorola was finishing work on their 6809 processor. They believed correctly that in the future most software for microcomputers would be written in high level languages rather than assembler. They also thought that applications and software libraries would be distributed via Read Only Memories (ROMs) rather than by tape or disk. (Although

OS-9 itself was initially distributed on ROMs, very little additional software actually used this medium). Motorola designed the 6809 specifically so that high level languages like Pascal and C could be implemented more easily than on previous 8 bit microprocessors. To support the use of ROMs as a software delivery mechanism, they added strong support for position independent code. These capabilities became the foundation that OS-9 was built upon.

Motorola wanted to provide a new high level programming environment that demonstrated the power of the 6809. Although I was not able to find documentation of the exact events that lead to their partnership, Microware had previously developed both a sophisticated real time operating system and a BASIC compiler for Motorola's older 6800 CPU. Given this experience, they would have been a natural choice for the task.

The Microware programmers worked directly with the designers of the 6809 CPU. This collaboration resulted in both a powerful new programming language and an efficient operating system that made full use of the new chip's power. At a time when CP/M was the dominate operating system on "serious" microcomputers and MS-DOS was yet to exist, OS-9 supported multiple users, preemptive multitasking, a unified I/O system, and many other UNIX-like capabilities in just 64K of RAM. The new language, BASIC09, combined the best parts of BASIC and Pascal and generated tight code that ran up to 10 times faster than existing BASICs. Compared to other microcomputer environments of the period, this software was in a class of its own.

OS-9 was able to provide advanced features using such limited

resources because it took full advantage of the 6809's new tricks. Although we don't give it a second thought when using modern computers, the ability to run multiple programs at the same time was uncommon in the microcomputer world of the late 70s. Most microcomputer code used absolute memory addresses and assumed it was the only program running.

This made it quite difficult to have more than one program in memory. Early multitasking systems required that each program be adjusted to use its own address space and left the task of making sure each space was unique up to the user. This made distribution of software on ROM nearly impossible, since the end user would not be able to adjust the addresses used by a program stored on a read only medium.

The 6809's new relative addressing modes were specifically designed to make it easy for programmers to write code that could run from any given location. Rather than "hard wiring" the address of each subroutine, the assembler could define these addresses in a number of dynamic ways. For example, instead of writing an instruction to jump to address #400, one could write an instruction to jump to a position 200 bytes in front of the CPU's current location. Other instructions provided easy methods for reading and writing data in relative positions. Although the initial motivation for these features may have been to allow use of software on ROM, OS-9 used this same feature to allow multiple programs to run at the same time.

Each OS-9 program and device driver resides in a self contained module. These modules can be loaded anywhere in the computer's memory, whether in RAM or ROM. An OS-9 module is typically also re-entrant, meaning that multiple instances do not require multiple copies of the entire module. The module definition indicates which portion of the module's memory needs to be unique for each instance, and only this small amount of RAM is needed for each additional copy that executes. This meant that several copies of a program would use only slightly more RAM than a single instance, allowing multiuser systems on very modest hardware.

OS-9 seemed to be a superior operating system for the microcomputers of the time, however its popularity was limited because the 6809 CPU did not gain widespread acceptance. Microware decided to focus on specialized markets such as real time traffic control systems and embedded devices used in manufacturing. By the late 1980's they had largely abandoned the personal computing world. OS-9 was ported to other platforms and remains a supported product today, although it is found primarily in dedicated control systems.

When I decided to put together a system that could run OS-9, I soon found that obtaining a working set of disk drives for a Color Computer is difficult. Fortunately, thanks to a free program called DriveWire, this was not an obstacle. DriveWire uses the serial port built into every Color Computer combined with a server that runs on a modern PC to provide virtual disk drives that actually work faster than real drives do. With DriveWire, I was able to boot OS-9 on my CoCo for the first time, over 20 years after I first read about it as a kid.

I soon found one frustrating aspect of DriveWire. The ability to support multiple users on serial terminals was really interesting, but DriveWire claimed the single built in serial port. I thought maybe I could do something about this, and so began a project that continues to consume my free time to this day. I started working on a new version of the DriveWire server that could provide a virtual serial port in addition to the virtual disks and other features.

I quickly found myself in well over my head. Thanks to the assistance of the kind, patient and knowledgeable folks on the Color Computer mailing list, a version of DriveWire that supported a serial terminal eventually came together. Soon I realized that while one serial port was good, 8 or 16 might be better. Why not connect these virtual ports to TCP/IP sockets? And why not create a TCP/IP API that OS-9 programs could use to access the internet? How about virtual modems? "Feature creep" doesn't even begin to describe it.

As I pondered these new capabilities publicly on the mailing list, other programmers became interested and stepped up to help. Boisy Pitre, author of the original DriveWire, helped to improve my "less than awesome" 6809 assembly routines and redesign many portions to be more efficient. Jim Hathaway began working on a web interface to make configuring the server more user friendly. Wayne Campbell used the new DriveWire API to create the CoCo's first IRC client, and I wrote a simple web server in BASIC09 that we continue today to use to blog about the project's progress.

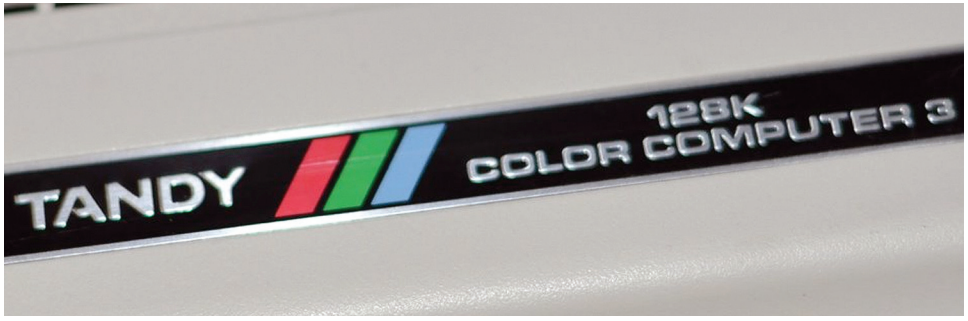
OS-9 remains a unique and interesting system for many reasons, but the most impressive to me is the community of users that still enjoy working with OS-9 today. If you want to learn more about the inner workings of a capable operating system without clawing your way through layers of bloated code, take a look at OS-9. For me, it was worth the wait.

FOR MORE INFORMATION...

The NitrOS-9 project is an open source version of OS-9 that runs on 6809 based systems such as the Tandy Color Computer (1, 2, and 3), the Dragon 64 and Alpha, and their emulators.
http://www.nitros9.org/

DriveWire is free software that provides virtual disk drives, serial ports, TCP/IP networking and more using the serial port built into a Tandy Color Computer.
http://sites.google.com/site/drivewire4/

The Color Computer mailing list is a group of knowledgeable enthusiasts that welcomes new members.
http://five.pairlist.net/mailman/listinfo/coco



300 BAUD

OUR BBS TOWN

BY GREG GOODWIN

When I heard about 300 Baud magazine being published the first thing that I thought was, "Wow, we need to get a BBS list in there". For those that were lucky enough to use computers in the '70s, '80s, and '90s you probably know of the computer BBS (Bulletin Board System) quite well. I'm banking on the fact that if you are reading this article, you are one of three people:

1) Someone who used to call BBS's, and thought they were gone.
2) Someone who has never called BBS's, and wish you could have.
3) Someone who still calls BBS's.

People are often surprised when I tell them that BBS's still exist, but why wouldn't they? All it takes to run a BBS is a computer and a modem, or during the late nineties ('90s) and into the naughties ('00s) you started seeing more and more BBS's over telnet and through the Internet.

Aside from the initial surprise they ask me "if I can get on the Internet, why would I need to get on a BBS?". I think this is a matter of scale and community.

First off, technical scale, if you are on a retro computer of your choice, you could probably easily put together a modem and a term program with at least VT-100 emulation, preferably ANSI emulation. Those two emulation standards are just methods of being able to read the character text and even color graphics as they should be seen to take in the full BBS experience. The nice thing is, most older computers (like my Atari 800) can generally hit 9600 baud, or even 19.2K easily with little modification, and while that would be mind numbingly slow for the Internet and intense graphics, it is fine for the simple text of a BBS.

Secondly, Community would consist of the numbers of people visiting a BBS. Many times in the past I have considered a BBS like a little town, complete with voting areas, places you can talk (either by message or internal chat), gaming areas where you can play against other users like going to a arcade, and you could almost see the download area as a store while the text file area is a library.

For the more active BBS's, they do seem like a town, though at first if you get a BBS that only has it's SysOp (system operator) replying to messages, it is like going to someone's virtual house, playing their games, hanging out and talking to your friend. Eventually other friends come over, hang out, and eventually you get a nice private party going, a little family, and in time can truly turn into a good sized community or 'town like' feel when things really get going.

Really all this needs is someone with a computer, a modem (or way to access telnet), a good term program, and well, you. :)

If you are using a more classic computer, this a great way to get more out of your system as you use it to communicate with other computers, and depending on the computer type, there are even BBS's out there run on the computer you are using. I know of many Commodore and Atari run BBS's out there as I write this. As a fan of retro computers, I am a fan of computer variety. A few weeks ago I met a man who ran the 2nd BBS ever hosted on a IMSAI. I hope that in time he might get that going again.

While I have been a caller of BBS's throughout the eighties, nineties, and naughties, using both dialup and telnet, I did enjoy the dialup aspect. Being truly old school I enjoyed the high pitched tone after my modem would dial, and then the returning tone of the modem coming from the BBS, and then the fun hearing chaos as those two tones crashed together. Those that used dialup for the Internet have heard this sound too. There is a great set of podcasts that call this the "Carrier Whistle" which ironically is the name of their podcast[1]

Here in the Dallas area we have shrunk down from hundreds of dialup BBS's to three brave souls that kept their BBS's (their virtual homes) open for people to come and visit, and those being Eclectic (which has been running since the '80s), the Prison Board BBS, and the Marsden BBS. So imagine the '80s joy I felt when I found a list online[2] that gave all the BBS's in the US and Canada, and the realization that with a cheap free long distance calling plan, I could call them all like local calls, and once again have way too many BBS's to keep up with.

I've called most on that list, and while i am still working to settle in on most of those BBS's, here are summaries of some of the BBS's I've already clicked quite well with. Mind you, I like a good variety of online games, but also really look for an active local BBS community in that BBS, or at least a SysOp that seems to care.

BBS Name	Modem	telnet address
BandMaster	604-266-5271 604-266-1977	bandmaster.tzo.com

This BBS has a caring sysop that converses with his users on an active message base.

> *"I haven't got many dialup users, so it's rather nice to see one once in a while!! Don't worry about tying up the lines -- there are still 2 phone lines and I haven't seen both in use for years!!!"*

BBS Name	Modem	telnet address
Empire of the Dragon	303-679-0161	bbs.eotd.com

While Dallas Hinton (SysOp) says he does not get many dialup callers, many do call through the telnet address (provided next to the number).
IceDragon (SysOp) wrote me back on what he wants to do with the BBS in 2010...

> *"I want to get a sort of old school museum sort of file library for old phreaking stuff --- Not for the criminal aspect! But a lot of us here are old Bell System fans. I'd like to put some of our AT&T LongLines files up here, too. Eventually I want to get a power supply for the Smartmodem 300 and run the Retro Line too :)*
> *I know, sounds weird, but I think it's important to know your roots."*

I'm sure to many 300 Baud magazine readers with a retro computer that does not sound like a weird idea at all.

BBS Name	Modem	telnet address
Prism BBS	607-655-5652	filegate.net

Out of all the BBS's I've called recently, I was most impressed with the local message community of this BBS. Just floored me how talkative they were. (Option R to read, J to join, then scroll down to "Main Board", enter message number)

BBS Name	Modem	telnet address
Prison Board	972-329-0781	rdfig.net

My home BBS. While the log in wall has been active, and people have a blast playing the door games against each other, we have been working on getting message activity going on local board #78, (at main, D for the games. For messages select M, then J for Join, enter 78. After that R to read, and you can select the message number, or N for new, and C for Current.)

BBS Name	Modem	telnet address
Vertrauen	951-549-9994	vert.synchro.net

A good stream of local messages. Must be running on an IP line because the connection was unstable even at 19.2K baud. A good selection of online games (option X at main menu). BBS has good ANSI character graphics.

This article has featured the (re) introduction to you the reader of the BBS world, the network of computers that is run and connected not by corporations and government but (as my local PBS station would say) by people like you. Sometimes BBS's piggy back off the Internet, but if the Internet were gone tomorrow, BBS's could still function as long as we had phones or HAM radio.

Internet communication to BBS's through telnet and SSH is readily possible, which is why telnet addresses were featured above if you want to quickly check out some BBS's. Get to the terminal on your computer; type "telnet", then "open", then as an example "rdfig.net" and hopefully you'll see the opening BBS screen.

Then when you want to go old school? I was able to get a long distance service plan here for not much more and with it I can call all numbers in the US, Canada, and US territories for free.

At one time (before that setup), I had a cell phone with free calling after 9 pm. I would call my home number (with computer and modem attached), and then with special timing, would call the BBS over long distance and set up a three way conference on the cell phone. I think I got 19.2K baud to work well, but i proved that it worked.

And while I had not tried this myself, here is one submitted to me by a user on the Prison Board BBS that you could try:

> *"Is it cheating to use Google voice to call a BBS? (it works). In short (YourGVoiceNum),,*,(pass code),2972-329-0781# from the terminal but I bet you could just ATA it when the phone rings." - Andrew T*

New ways of calling and old, and mix of the two are possible. And it really can be a joy typing away on your retro computer connecting with new friends in a little online town like environment that a BBS can be.

Try it out, and if you have any questions, stop by the Prison Board BBS and write me a message (or drclu@swbell.net if you get really stuck).

WEBLINKS

[1] http://www.amigaz.org/category/carrier-whistle/feed/
[2] http://www.telnetbbsguide.com/dialbbs/dialbbs.htm

300 Baud Magazine
1 Yester Place, Haddington
East Lothian, Scotland, UK, EH41 3BE
www.300baudmagazine.com
Editor *William Dale Goodfellow* • **Art Director** *Simon D. Williams*
Production by LD8 Media Design • www.ld8.ca • Printed in Canada

PERSONAL COMPUTER

50p U.S.$1.50

PERSONAL POWER IS HERE

THE

Nascom Construction Manual - Issue 1 February 1978

Point 12.
Do not attempt to work more than 2 or 3 hours at a stretch. A reasonable time for construction would be 20 hours (say a week of evenings) although it may take from under 10 to 40 hours or more depending on previous experience. These times are in addition to the initial "reading, considering and inwardly digesting" time for which at least another 10 hours should be allowed. We repeat that we do not advise construction to begin at all until a few days after receipt of the kit and full reading of the documentation.

Point 13.
GOOD LUCK!

— PERFECTLY AT HOME

WIN
A NASCOM 1
TWO PROGRAMMABLES
BOOKS
•
Details Inside

THIS YEAR, GET YOUR OWN COMPUTER

The Nascom 1 was the UK's first widely available home micro computer. The UK in the late '70s was a significantly poorer place financially to be launching or buying home computers and for that reason the Apple II, TRS-80 and the Commodore PET never made the same sort of impact as they did in the US market. It wasn't until Sinclair and Acorn arrived that the UK scene really took off. The Nascom 1, was the first proper computer that was a realistic proposition for many in the UK - costing less than £200 in 1978. It may have only been a single board, but it was still substantially more powerful than the so called 'trainer boards' such as Sinclair's MK-14 or Commodore's KIM-1. Generally, the Nascom 1 (it was called the Nascom 1 even before there was a Nascom 2) arrived in kit form, but for an additional charge it could be assembled for you. To construct, the builder would have had to have embarked upon the soldering of around 3000 joints and connections. Nascom eventually went on to produced a more refined version in the Nascom 2 and finally a fully complete system including a case in the Nascom 3.