Winning the Micro Game

by Don Lancaster

Dilettantism won't do when it comes to learning about microcomputers.

Hands on is everything. The only way to ever learn anything about computers is to jump in with both hands and feet, get on line and do some computing. Until you actually do and see just what the micro world is all about, you have accomplished nothing. For you must do things yourself, on your very own terms, in front of a working, real computer, alone.

It is both funny and sort of sad to hear any student say he just took a DP course but couldn't get any CPU time. He got taken, not the course.

You can become computer literate by applying computers, not by having someone tell you about them or by reading about them.

Understand a timing loop by writing one and watching it work. Make an interface by taking a trlac, an optocoupler and a 100-Watt light bulb and shining light on the real world. Find out what an interrupt is by interrupting a computer. Do it all by yourself.

You have to make misteaks. If you are learning micros or developing any new product, half your experiments should fail. A canned set of exercises on a micro trainer ends up next to worthless if everything falls into place and works perfectly the first time.

In the micro world, you make mistakes to learn and to move forward. Thus, you should expect mistakes. Prepare for them and welcome them. Aggressively seek them out.

Of course, it makes sense to **never make the same mistake twice.** Build on what you have. To expand your microcomputer universe, try new things that may fall. Find out why they fall, and use this as a newer and bigger base to work from.

Usually, you'll never be anywhere near where you think you are in solving any hardware or software problem. Unexpected surprises and plain old stupidity are always between you and reality. If you think you may have something working perfectly, you probably do not even understand the prob›lem.

You must mix hardware and software. Some heads-in-the-clouds "pure" software people out there still believe that hardware is some mundane inconvenience standing between them and real "computing." And there are technician types who do everything with bushel baskets crammed full of integrated circuits.

Neither approach is good. Sometimes a simple and inexpensive hardware circuit might replace scads of software. Other times and other places, a few lines of elegant software might eliminate the need for custom circuits or a special device.

Winning computer products will combine both hardware and software, using the best features of each to give you the simplest system along with the lowest possible cost.

This means that if you are a hardware person, you should learn programming and learn it fast. If you have a software background, start soldering or breadboarding with a vengeance.

Synergy says that 1 + 1 = 4. This is definitely the case any time you get an opimum mix of

hardware and software interacting with each other. Neither can stand alone. Not any longer.

The real world is fuzzy. Some textbooks and lab experiments do work every time. Everything is nice and clean and neatly tied up. You do exactly what you need to do the job, no more, no less.

Unfortunately, reality does not work that way. Firstoff, you must deal with people, and that will always mess things up. Key items will be missing or late. The magic chip may just be a figment of an ad writer's dreams. Or a problem may have a simple and inexpensive technical fix that ends up politically or socially unacceptable. Egos conflict.

Expect and accept fuzziness. As you get into a new computer area, things will usually start out completely confusing. Then they might become fuzzy. Then they should become, for a glorious instant, crystal clear. Then, of course, they will get fuzzy again as you become more involved.

As you go to the bigger picture, expect more fuzziness. Also recognize that there really isn't very much in the way of real world beginnings or endings. Rather, things sort of dribble off into the great whatever.

Micros might—just might—be the missing link between ordinary people and intelligent life in the universe.

Hit the basics hard. Any 6502 micro freak can sit down and immediately "prove" that the 6502 is ten times better than most any other micro in the world. The trouble is that you can do the same with any other micro family as well.

For most micro uses, it makes no difference which micro from which family you use. Even if there temporarily was a "best" micro, other factors such as your own skills and attitude, the available software, the elegance of all of your competitor's programs and so on will reduce any advantage of the "best" micro to zilch.

If you don't happen to like the "best" micro, just wait a month or two, and it will get shot out of the saddle by something far more promising. This all means that **the micro you learn is not the micro you wil use.** Later on, there will be much better ones to work with, and they are sure to have completely different tech details.

To beat this, hit the basics hard. All the micros have address space and addressing modes. All do have interrupts, use subroutines, clocks, ports, memory, and I/O. Use any micro you like to add tech details to the fundamentals. But be sure to get the essentials down solid.

Reach out and put the touch on some one. All of the nickels in the micro world are newly to get made in places where people are not yet using micros. Find those places and get involved with these areas and people on their own terms.

You can learn a lot more about micros watching fourth graders zap Klingons than you ever will in any university COBOL course.

Put all the new micros to work feeding cattle, treating sewage, gambling against Wall Street, designing looms and mixing cement, baking calzones, milking goats, hulling pecans, questing tinajas, animating video, co-oping groceries or hybridizing sinsemilla, improving wood stoves, redesigning bicycles, or restoring steam calliopes. Or improving stream gauges, selling paper clips, cutting dress patterns, or teaching trumpets.

When you do reach out, always work with the other person's terms and language, bending the micro info to fit as best you can. If they are smart enough to learn micros, they won't need you for anything. Find places where they don't yet know that micros can help. Then jump in.

Don't reinvent the wheel. Instead, find and steal the plans. Much of your needed and obvious micro-related information has already been done and is readily available for your use. For instance, if you want to drive a teletype or another printer, use some else's driver routine. Don't stop what you are doing and invent your own. Unless you

truly want to know how a drive program works. Or how to go about modifing one.

Scads of Morse code trainer programs are out there. Why write yet another? The same goes for sorting and word justification subroutines. And you'll find more versions of Lunar Lander than there are moons in the solar system. How many Hangman or Nim games have you seen?

Now, if you want to explore these programs, that's fine. But if your goal is using something rather than creating something, find out what has already been done and go with it. Or improve it and then go with it.

Refer to monitor listings, software exchanges, micro mags, ap notes, club newsletters, program books, and micro information exchanges for scads of stuff to use.

Better wrong now than right later. In anything you do in the micro world, your first attempt will be wrong and will need reworking. So, quickly kludge up your first attempt and let your mistakes show you the way to go. Often, you don't even understand what the real problem is until you are inside a program or some hardware looking at it.

Try a simple, quick, and dirty tactic that least sends you in roughly the direction you want to go. Make some guesses. Take a stab at it.

In your early attempts, if it works, use it. Start your project flying more or less right side up. Later on, you might go back and add structure, elegance, convenience, and simplicity. Add the final spit and polish on the way out the door. But never early in the game.

Write it down! And not on the back of an old envelope, either. Documentation is the password to avoid self-destruct modes. You do not record only final programs and schematics. Instead, keep track of what you did and why you did it. Neatly, and in some semblance of order. Keep accurate records of where you have gone and where you are heading. Put together files of everything that is related to your projects.

Software will be worthless if you can't show someone else how to use it. Hardware has no way to operate if there is no way to connect or fix it. You must be able to go back and reuse or modify what you did a week or a year ago.

Documentation is not just a hex dump; it is a micro way of life. You cannot survive without it.

Back it up! Continuously save ongoing copies of your work. Never erase or overwrite any older copy. For when (not if) something goes terribly wrong, you need to be able to easily restore. And not lose all of your previous work.

Don't separate work from play. Which of these is more important: Designing some efficient sort algorithm for a business general leger program or figuring out how to use Adventure's oily slime?

In the long run, that oily slime is vastly more important because that is what is stimulating interest in micros and what is now making people computer literate.

Any program run on a computer is a game! However, stuffy institutions, banks, bureaucrats, and other so-called "serious" computer users have rules that say you are not supposed to smile while you are playing their games.

Simon says don't smile. It is still a game.

You will never get enough No matter how far you have gone in microcomputing and no matter how much of what kind of hardware and software you have on hand, you will always "need" more of something.

More memory? Start with a 1k trainer, then 4K, then a 16K micro. The overfill the 16 megawords of an extended micro space. Hard copy? Start off with Excedrin headache number ASR-33, then on to thermal, a Selectric, and then a daisywheel.

From Plain Jane video, step up to graphics, color, hi-res, and then super resolution color with full gray scale. From cassettes, it is on to floppy, dual floppies, quad density, and on to bunches of new stuff not yet even thunk of.

There never is, nor will there ever be, a time when you have "enough" of anything. What looks like a light at the end of a tunnel is just a train that is speeding towards you.

You will find only one way out of the "more" syndrome. Always go with what you have. Make it work. Live with it as long as you can. Force it to pay. Make it do. Use it up. Wear it out.

If it is old line, stomp on it. Some pre-micro people and institutions are still kicking around the lunatic fringe of the micro world. Who persist with enormous, bureaucratic, centralized, insanely priced, and unavailable megacomputers run by an elite priesthood singing the incantations of some arcane language. They utterly fail to recognize the power of the micro as a highly personal, one-on-one, decentralized, inexpensive, interactive, and individual convivial tool.

You can learn far more about micros in twenty minutes watching a pair of fourth graders zap Klingons that you ever will in a university course.

Old line stuff includes IBM, batch processing, COBOL, decollators, Honeywell, key-to-disk, data encryption, FORTRAN, klutzy keypunches, and centralized billing. They are totally without any redeeming social values. They had their chance and blew it. We tried it their way and it didn't work. Old line fails to see the problem.

Always ask, "Why are you telling me this?" The useful products and ideas in the micro world are not heavily advertised. In fact, anything genuinely useful takes a lot of trouble to nail down.

If a micro is widely or extensively advertised, it more than likely tells us that a something vastly better is now available elsewhere. If someone is radically trying to convert you to his micro or his way of doing things, the chances are he has drifted into right field and become snookered into a bad scene. He is looking for converts to ease the pain when they are shot out of the saddle.

When anyone tries to tell you about micros, always ask, "What is the real reason you are telling me this?" Find out the motives involved. Then get a second opinion, check out another choice or find some different viewpoint before you plunge ahead. Or get sucked in.

Nail down all resources. It is easy to assume that formal courses and rather expensive hardbound textbooks are the only way to "learn" micros. In fact, these are two of the worst possible ways to become computer literate. Most of these learning aids are stillborn, hopelessly obsolete and badly misdirected.

Anything you can relate to that involves micros is a resource. Your first and foremost resource is yourself through hands on experience.

Other resources do include micro magazines, clubs, game playing or Dungeons and Dragons, micro trainers, computer stores, used hardware boards, tech journals, funky books, reader-service cards, benchmarks, students, teachers, surplus stores, trade shows, computer fairs, rap sessions and swap meets.

But most important of all...

Go on your own vibes. There is no right or wrong direction in the micro world. In fact, 99.9 percent of the micro world remains unknown, unexplored and uncharted. So, if "they" insist on something, most often they do not have the faintest clue what they are talking about.

If you are interested in something and want to go in that direction, fine. Do it! Your surest bet for long-term winning is to roll with your own vibes. Explore what you want to. Ignore the herd thundering the other way. Get off the beaten path. Make yourself your own best customer.

Satisfy your own needs and your own curiosity. Put as much psychic energy and personal value as you can in the routes that you pick, and you are certain to win the micro game.

You are, by definition, the center and the most important part of the micro universe.

Don't ever forget it.