Carl Helmers:

What is BYTE?

"It could not have been long before some wizard of verbal magic figured out that a group of little bits must constitute a mouth watering BYTE."

"... the term byte has become part of the lexicon."

For the hardware person, "the fun is in the building."

This is the first issue of a new publication – BYTE – a monthly compendium of information for the owners and users of the new microcomputer systems becoming widely available at moderate cost. To formal and informal students of computer science, the choice of the name BYTE is quite appropriate. For a large number of applications of this new technology of inexpensive computers, character string and text, data (basic unit, one byte) is an important consideration. Bytes are the units of data manipulated by many of the small computer systems designed by readers – or assembled using one of a number of kit products now on the market.

The most common definition of a byte is that of a unit of information containing 8 bits. This unit of information can at any time represent one of $2^8 = 256$ possible things – for instance, one of the ASCII or EBCDIC character codes, one of the integers from 0 to 255, a signed integer from -128 to +127, etc. The origin of the term "byte" lies in IBM's documentation and terminology for the extremely successful System 360 series. The folk tale has it that IBM needed a more "personalized" (i.e. unique) term for the old standby of earlier generation computers, the "character". The term had to be less tied to a specific type of data such as character codes – and had to take on a generic meaning as "unit of storage". With that functional specification for the required term, it could not have been long before some wizard of verbal magic figured out that a group of little bits must constitute a mouth watering byte.

With the term's widespread use in the computer field due to IBM's benign influence, the term byte has become part of the lexicon. The fundamental significance of a byte as a unit of information makes BYTE an appropriate name for the publication. BYTE is your unit of information on the state of the art of small computer systems for individual persons, clubs and classroom groups. Each month you will find information ranging from computer club announcements to manufacturers' advertisements, from technical details of hardware and software to humorous articles and editorial opinions.

The Home Brew Computing Trilogy

The story of computing is a story composed of several elements. A good way to look at the story is as a trilogy of interrelated themes . . .

HARDWARE

SOFTWARE

APPLICATIONS . . .

You need the hardware before you can progress through the first gate of a system. A virgin computer is useless so you add some software to fill it out. And the whole point of the exercise – in many but not all cases – is to come up with some interesting and exotic applications.

The technical content of BYTE is roughly divided into the trilogy of hardware, software and applications. Each component of the trilogy is like a facet of a brilliant gem – the home brew computer applied to personal uses. The trilogy is not confined to home brew computers alone of course.

In the personal computing field as in any endeavor there are people who will have foremost in their minds any one of these three topics to the exclusion of the other two. For instance some of the people I know are interested in software – and pretty exclusively software. They'll tend to concentrate on software as much as possible and try to get a minimal amount of hardware sufficient to experiment with software. A person with a primary interest in software will oftentimes be the person who purchases a kit computer because the kit minimizes the amount of hardware knowledge the person is required to have.

An example of another kind of person – in terms of isolated characteristics – is the hardware kind of person. Here the emphasis of the work with home brew computing is on putting together the hardware, designing the hardware, making things that quote "work" unquote. A "hardware person" in many cases may not do much else – but he or she certainly will accomplish the design goal. This is the person who builds

— (the first) editorial

up a computer system to the state where it might even be able to do a bootstrap off tape — then drops the system and decides to build a better one. The fun here is in the building, not in the using and programming.

Then in this description of possible ways of approaching the home brew hobby there is the applications person. This person's attitude is somewhat a synthesis of the other two types. The applications person is typically interested in getting a particular program up and running. So a Space War freak would spend a good portion of available time getting the hardware and software needed to play space war. A LIFE addict would spend a fair amount of time getting the hardware and software for the game of LIFE — and fooling around with LIFE patterns. And a person who enjoys other computer games — using the game as a goal — spends much time assembling a hardware/software system for the game. A person interested in toy robots would have a combined hardware/software problem of coordinating and controlling movement — this home roboteer must design the mechanical details, design a control algorithm — and if sophisticated fun is required, must design a pattern recognition input device and algorithm for interpreting scenes. A model railroader requiring a computer controlled layout again has this applications model — computer controlled yard and main line switches — and faces a choice of possible hardware and software components needed to make the application.

Now, aspects of the trilogy exist in any particular person who experiments with the computer systems. A common combination is for the application to drive the hardware and software choices. Then there is the person who builds the hardware first — getting caught up in the "neatness" of a logical construction in the same way that a mathematician goes off the deep end with a neat theorem. If you're starting as such a hardware hacker, you may come to the point where you say "Hmmm — I've built the hardware, so now what do I do with it?" Here the applications are following the design of the computer. It's the same way with many kinds of programming . . . the software is an exploration of the possibilities of the hardware. As a software hacker you might turn to the pure logic of programming — writing and trying out routines for things ranging from augmentations of the instruction set to file managers, to games simple and sophisticated. Or you may just fool around with programming with no specific end in mind in terms of applications, for the sole purpose of seeing what you can do with the machine.

As a home brew computer software experimenter, you'll find an emotional kinship with the people who take part in the automotive hobbies. What does a "performance" automobile buff do with the machine — once all the optional improvements and features have been added underneath the personalized paint job? The auto nut takes his car out to the local drag strip or other test track and opens up the throttle to see what the engine and drive train will do in terms of speed and acceleration (proper spelling: exhilaration). Well, for computer experimenters there is a logical drag strip in every computer — an instruction set waiting to be explored. You exercise this logical drag strip by seeing what you can do in the invention of neat little (and not so little) programs to do useless (equivalent of real drag strip) performance tests in artificial circumstances — or really useful tasks (equivalent to normal transportation functions of autos). I haven't yet figured out what the computer equivalent of an air scoop hood is — or the equivalent of the weird mechanical contrivances I often see on the derriere of "muscle" cars.

You might even be the type of person who wants to do a certain programming technique just for the sake of programming — you say to yourself: "OK — I want to write a BLURPTRAN language compiler and code generator, so what hardware do I need to do it?" As such a person, you would then choose a computer system in packaged and/or self-designed form such that it would fit the compiler writing goal.

In truth many will find it best to seek a sort of

"A virgin computer is useless so you add some software to fill it out . . ."

Sophisticated fun requires sophisticated thought and hard work . . .

"Hmmm — I've built the hardware, so now what do I do with it?"

"Well, for computer experimenters there is a logical drag strip in every computer — an instruction set waiting to be explored . . ."

interactive – balanced – relationship between the three aspects of the computer trilogy. At any given time, almost anyone has some aspects of all three combined within his own philosophy of home brew computing.

BYTE – the magazine – addresses this mixture that occurs in various people by providing articles permuting and combining these areas.

In this first issue *hardware* articles include "Deciphering Mystery Keyboards," an article on recycling used ICs, and Don Lancaster's article on serial output interfaces. Articles on *software* include a description of the assembler concept by Dan Fylstra. *Applications* are found in the first segment of LIFE Line. This application includes information on programming techniques as well as suggestions regarding required hardware.

So here in the first issue you find an example of the mixtures of these factors which go into home brew computing. This mixture of aspects is a guiding theme of the current and following issues of BYTE – one of the key editorial goals is to cover a complete range of ideas spanning this triumvirate of concepts.

The Impossible Dream

or, "Wouldn't it be neat to have a computer all one's own without being as rich as Croesus?"

The art of home brew computing has come a long way in the past few years. To paraphrase the science fiction author Robert Heinlein, "When it's time to do home brew computing, people do home brew computing." That time has come today, with the advances in memory and processor technology inherent in large scale integration. The present devices are not as good as the "Thorsen Memory Tubes" in Heinlein's *Door Into Summer* – but it's getting almost to the point where a basement tinkerer can put together a manufacturable robotic device and plant the economic acorn which will grow into an industrial oak tree.

My own first exposure to the idea of home brew computing was about eight years ago when I was attending high school in rural New Jersey. A ham (radio amateur variety) friend of mine at that time was attempting to get a surplus RCA computer card rack into operation as his own conception of a home processor. I didn't know enough at the time even to ask an intelligent question about its design. The thing was a monstrous 3-level card rack with a heavy wire wrap back plane and transistor logic with integration to the level of modular cards. I don't think this friend of mine ever got his processor working to any significant extent – but the impression was made: "Wouldn't it be neat to have a computer all one's own without being as rich as Croesus?" I filed away the thought of a home brew computer as an "impossible" dream at that time – how could I afford a computer if I could barely afford a beat up old Hallicrafters SX-99 receiver and flea power ham transmitter? That did not stop me from having fun with computers – it merely caused a redirection of attention to the use of computers financed by agencies other than myself . . . for a while.

The while lasted several years as I bootstrapped myself through college with FORTRAN, COBOL, PL/1, BAL and a bit of financial aid from a private foundation. Along about 1972 when I started reading about the LSI computers being designed by Intel – the 8008 and 4004 – I began to revive that old dream of "having a personal computer." Here was a single IC chip – the 8008 – which would give me a real stored program machine at reasonable ("reasonable" = under $1000) cost. After attending an Intel seminar in 1972, I resolved that I would actually build an 8008 computer.

The resolution was a long time being turned into reality – I did not actually begin design and construction until January 1974. I took my time for numerous reasons . . . among them being the fact that I had to learn something about the way hardware works, had to equip a laboratory of sorts, got this bug about self-publishing the results along the way*, and so on . . . I finally got an 8008 computer which would execute instructions in the middle of the summer of 1974 – and to quote one of the subscribers to my self-published series of articles, "I learned a lot . . .," just as he did. So much for the personal involvement – I've built a kluge of sorts – the software hacker's first attempt at hardware – and have learned quite a bit as a result. You – the reader of BYTE – can go that route or use a much easier route – there are several manufacturers of kit products advertising in this magazine. And you'll find a magazine full of helpful information which I didn't have.

. . . CARL

*I got the idea of self-publishing from a pamphlet put out by SOL III Publications, Farmington, Maine – which has since been turned into a book entitled *The Shoestring Publisher's Guide* – full of useful materials on publishing by individuals and small organizations.