## DCC pod.box 310 menlo park, ca. 94025

Deliver to $\Rightarrow$


## This year's miracle

Not too long ago it became possible for large scale chips to be made. Intel rolled the ball with the 8008, then rolled it some more with the 8080 . This year's real winner looks like being the PACE chip, from National Semiconductor. These things are not easy to make. The difficulty of making them is enormously increased as they get bigger, and a 16 bit CPU is awful big.

The story starts with a few enthusiasts, a blackboard and a whole crew who doubted it could be made at all. Moby chip, as it was known, was not a project for the faint hearted.


Now the magic begins. It starts with a single crystal of pure silicon. PURE silicon. The chip is made by making parts of of it very slightly impure - if there are impurities already in the silicon, nothing will work. (Repeatedly chalk sodium on the sidewalk outside a semi-conductor house and you are likely to cause people to have nervous br sakdowns.)

The silicon is sliced into wafers, the wafers are ground optically flat, cleansed of scars and cracks, and then processed. By etching, cooking, defusing and oxidising. Bear in mind that we are discussing an item so delicate that a human hair looks like a tree trunk beside it. Yet, this thing gets cooked in an oven. Repeatedly. This really is the time for the good fairy to come along and wave her wand, though black magic describes the process better than science. Science there is, for instance, each mask step requires alignment. Precise alignment, for any error will plary things where they shor id not be, and the things to be olaced are very, very small. And there are many masks.


Magic there is too. Sometimes the process works without trouble. Sometimes the process is completely lost after working perfectly for months, and if it can't be found again, the product is dropped. This happened to one of the larger, and better makers. They were making a device which, for this article, we will call a 2 N 4249 . They then made an improved version, the 2N4250. This was fine in the lab. Fine in pre-production. But they forgot how to make it, struggled and struggled to remember, couldn't and quit making them!

Sometimes a particular person can make things when others can't. There is the tale of the big manufacturer which was licensing a company in another country to second source its product. They set up a production line and taught the other company to run it. Products were excellent but somehow the deal went sour and the men from the big company went home. And the production stopped - nobody in the other company could manage to make them any more.

Sometimes it is literally a single person. Some years back a company was trying to make about 500 transistors. They had to handle 1,000 volts, so it wasn't easy. Out of each batch, they were lucky to get two which worked and they were losing there
and they were losing their shirt until they noticed that one employee, on the night shift, was making far more than all the others combined. But when they watched her, she did no better than anyone else. The mystery was not cleared up until someone who she knew was watching her work and chatting, very late one night. She was loading a batch into the oven and, rather tentatively, remarked that she usually smoked a cigarette about that time. She was told to go ahead, just do everything the way she usually did when when she was alone. Semiconductors are processed in clean clean rooms, so that was like a surgeon smoking over his patient, but the batch turned out fine. Later experiments showed that tobacco smoke upped the yield for other employees, but not by very much and they never figured out why, so she eventually made the entire 500 units.

If the process could be really defined, the devices could be reliably produced, but it is far too complex. In reality it is much akin to cooking-one ounce of this, a teaspoon of that, a pinch of the other, then cook until it is just ready for something else. When is it ready? Well, a good cook just knows . ... The analogy goes further - if you want a good meal, go to a restaurant with a good reputation. And if you like to eat at the little place on the corner which has those super exotic dishes, better hope they keep their kitchen clean.....


After surviving all this brutal treatment, the wafer is broken into pieces. Even Moby chip can be made many times on a single wafer. Now comes the job of mounting the final chip, the survivors that is; obvious bad ones being culled. The chip is placed gently and carefully into its final position, and then the connecting wires are welded on. Yes - welded - the final brutality.

The girls who do this work are careful but a few more chips bite the dust here. Leaving the survivors to go to testing.


## NATIONAL SEMICONDUCTOR

PACE was not easy to make. Moby chip has come of age though, and is beginning to pour out from National. Quite a few are pouring onto Bill Godbout's desk - the beginnings of a torrent, to judge from what we have seen of his kit.


Another complication comes about because the testing does not separate chips into good chips and bad chips. This cannot be done for the same reason that one cannot separate pies into good pies and bad pies. The best pies are superb. The worst inedible. All the cook can do is to draw an imaginary line, throw out all those on one side and sell the others. Knowing of course, that his reputation as a cook depends on the flavor of the pies.

If he is a good cook, he doesn't throw many pies away and prospers If not he goes under. The word for this is yield, and a $70 \%$ yield, less than one third thrown away, is remarkably good. A $5 \%$ yield will close out a product. The cost of making a batch is identical whether it is good or bad, and the cost must be recouped from the sale price.

Bill, by the way, recently bought a lot of 4 K EROMs He sells them cheap, programs them for the buyer and has an 8 K Altair board to suit them. A lot of them were ordered by a company which was going to sell them with an assembler for the Altair, they cancelled and so he has around 200 of his 8 K boards, with 4 K of EROM containing the assembler to sell.

and so on. With 4 candles, we can go up to 15 years old, with 5 candles, up to 31 years old and with 6 candles, up to 63 .


But if you are 58, get people to look at the cake from the back instead of the front . . . in fact, if you don't put something on the cake to indicate the front, your age will be ambiguous (unless, of course, it is a binary palindrome!).


A PALINDROME!

How many candles would Methuselah need?
How many candles would Gandalf need?
How many candles would a Dragon need?

And, for all you computer people out there -


Here is a cake with an extra BYTE

Tho following article in adapted from sections of the narration in filmmaker／domebuilder Jerry Brown＇s DOME FIL．M． Jerry is also vica－president of Dymax and cosuthor of BASIC the programmad instruction text published by Witoy \＆Sons．
Geodesic domes have enjoyed a sort of fadish popularity in the last decade．Because it can be shown mathematically that domes enclose a maxi－ mum of volume with a minimum of surface area， everybody thought they would be economical of building materials and therefore ecological．And they might be，if you＇re working on the scale of doming over whole cities．But on a people－size scale，such as cabins，houses community gathering places，or workshops and the like，well，the theoreti－ cal benefits tend to be offset by real life considerations． For example，if you＇re using lumber in your construction， the way fumber is mifted and shaped makes it hard to use economically in the dome part of a dome house， where right（ 90 degree）angles don＇t exist．＂Every－ thing customized＂is a lament of domebuilders，and if you aren＇t into hand－crafted houses，domes aren＇t for you．
Domes do have some things going for them．They can be made in modules for portable shelter，such as dome tents，even circus or carnival big tops．You can prefabricate the framework and other parts of a dome，which is efficient if it＇s done accurately． But mostly domes are popular for aesthetic reasons， and for the subjective good feelings of being in a round， high－ceiling，non－claustrophobic space．Geodesic domes are developed from the same set of very basic geometric（Platonic）solids that includes the pyramid， and for many people domes have the same sort of fairly mystical good energy vibes that pyramids are supposed to have．

A lot of experiments in domebuilding were done at Pacific High School in the Santa Cruz Mountains of California in the late＇ 60 ＇s，and this is where the classic DOMEBOOK II was put together．When Pacific decided to go residential／communal，a crash program of dome construction started．The adults and kids at PHS did about 15 domes，trying out various designs，materials and construction techniques．Those domes and the ones in the Dome Film are spinoffs of Buckminster Fuller＇s work in geodesic geometry． You could say that Fuller opened the mathematical door，then he patented and publicized some of the possibilities，and Bucky could be called the person most responsible for wide－spread experiments in dome building．

About Jerry Brown＇s DOME FILM：The Sales Pitch
For rental or purchase information，contact J．Brown， 1682 Indian Valley Road，Novato，CA． 94947.

Jerry Brown＇s DOME FILM is a new 16 mm color sound film derigned to give a practical overview of hand－crafted dome－ homes of geodesic derivation．The film is an honest and objective assessment of the possibilities and uses of the dome，as well as the limitations and problems of this unusual structure．The DOME FILM is valuable in learning carpentry， construction and design，and is of special interest to those considering building their own dome，whether for a cabin， a home，a workshop or studio，a greenhouse，an auditorium， a community meeting place，a church，or to house a business （all popular uses of domes）．

There is a problem with most owner－built domes： they leak．In fact l＇ve never met a dome that didn＇t leak－at least a little at the beginning of a rainy season．There are three reasons for this：poor design， poor construction，and poor maintenance．A dome is all roof and little or no wall．In a regular rectilinear （squarish）building you have eaves overhanging the walls and vertical windows helping block the rain，but in a dome all the windows are like skylights，and they have to be installed and sealed like skylights．Also the fact that window installation comes somewhat late in the construction process，when people start to get blown out，doesn＇t help the problem．And after all the drying，warping，expansion and contraction of the summer months，seams and windows get loose if you haven＇t figured on that in your design and you haven＇

You can build your own dome．If you are inexperienced in construction，start modest，on greenhouses or sheds for practice．With patience，foresight and planning you can build a sturdy，tight dome that will keep you warm and dry，a handcrafted nest in the woods．
There were two big personal benefits to me from working on this project．One is the feeling in my hands and heart that I can build my own good，sturdy house，dome or otherwise，and that is a good feeling．The second is the ability to see and understand how almost any structure is built，with some judgment as to its condi－ tion and sturdiness．

Watertight seams that depend on caulk，hydroseal or any chemical sealer will just have to have some patching every year，you can count on it．At the first rains，I＇d go around with grease pencil and chalk and mark where the drips seemed to be coming from．The next dry spell， most leak places identified by these marks were visually obvious on the outside．When I lived in the Sun Dome at Pacific High one fall and winter，I got flooded out on the first rain．That dome has pre－fab triangles of aluminum sheets with wood－strut edges， with adjacent edges of triangles bolted together，with a neoprene rubber＂gasket＂between the wood．Nobody had tightened these bolts since the dome had been put together several years earfier．There were hundreds of bolts；I felt like Chaplin in＂Modern Times＂．But there were only two minor leaks to deal with after the next rain．
with vertical walls，on which sits the second story，a $30^{\prime} 4$－frequency alternate breakdown dome hemisphere，There is also a sleeping loft inside the dome．


The DOME FIL．M shows Joe，age 22，and Robin，19，constructing a well－built two－ stary dome houso，using $\$ 5000$ in mat－ erials．Note the two 8 －sided radial plat－ forms，giving them a lower story room
回 $\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$
$\square$


 $\square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square \square$
$\square$
回回圖圖圖圖回回回回回回


The lower story is not visible in the last photo，being obscured by the sun deck or porch that extends from the edge of the porch that extends from the edge of the dome the second story）out to the
side on which the dome was built．


The intraduction provides an overview of ownerbuitt domes，a nontechnical presentation of dome vocabulary， and the basic geometric ideas from which domes were developed by Bucky Fuller and others．A dozen inexpensive domes of varied design and materials are reviewed．The introductory section concludes with an examination of wivaterproofing，as reducing the risks of leaks is a major building consideration
Part two of the DOME FILM，on the staps in dame con－ struction，follows two projects in some detall from foundations to finish（see the stills from the DOME FILM below）

## 

The introductory section of the DOME FILM includes an outstanding nontechnical presentation of the basic geometry from which geodesic domes are derived，using a combination of computer animation，models and real structures． The computer graphics were filmed directly from a CRT display on Tri－X Reversal film at 12 fps ，using a PDP－10 implemented program in LISP by Bruce Baumgart．

As a model of what an instnuctional documentary film should be，the DOME FILM is earthy，humorous，honest and highly informative．It features some outstanding colo cinematography，an excellent original music sound track mixed from quad and 8 －track studio masters，and an honest， no nonsense narration by domebuilders Joe，Robin，Jemy and Peter．They explain the problems encountered before， during and after constnuction，as well as the solutions they found．More than a＂how－to－do－it＂film on building domes， the DOME FILM also captures the energy，hoper and vaiues of young people living in touch with the land and literally building alternatives for themselves

# K1月GDOM 

This is the story of a computer game called KINGDOM, which like many other computer games, is based on the premise that there are a lot of things the dverage person would like to do, but, for some reason or another, cannot. To satisfy these urges, there are games through which it is possible to land a spaceship on the moon, quarterback a football team, beat the house at the Las Vegas blackjack table, or indulge in any number of similar activities. The game of KINGDOM adds one more item to this list which seemingly appeals to everyone: it lets you rule the world.

Just why everybody seems to have this compulsion to rule the world is still one of nature's great mysteries, but when the KINGDOM game was created during the latter part of 1972, it soon became apparent that everyone who got to play the game turned out to have a little Alexander the Great (or, in some cases, Ivan the Terrible) hiding inside them somewhere. Allowing people to play a game which gave them a chance at ruling the whole world has a truly amazing effect on some of them; but this is getting a bit ahead of the story. Better to start with the game itself.

What the KINGDOM game is, in effect, is a computerized simulation of a very simple economic system, in the form of a game. The overall premise of the KINGDOM game is that the player, with the help of the computer, is put in charge of a small, mythical, and somewhat medieval kingdom. As the ruler of this kingdom and its inhabitants, the player is expected to make certain decisions which will affect the welfare of the kingdom. If the player is good at it, the kingdom will survive and prosper; if the player is very good, the kingdom may eventually expand into a worldwide empire.

The reason for choosing a medieval kingdom as opposed to the United States of Industrialized Russia, for example, is to keep the rules of the game simple enough to be understandable (and if you have looked at the state of the US economy lately, you may realize how terribly complex the 'rules' can get, and the kind of terrible mess you can get into by not totally understanding them!). The ruler of KINGDOM does not have to worry about Urban Development, Defense Spending, Prohibition, Internal Revenue or the Garbage Strike; he has only to keep the subjects of the kindgom fed enough to prevent mass starvation and, hopefully, turn a profit while doing so.

The prime means of support for the populace in KINGDOM is agriculture; the people plant grain at the beginning of each year, harvest it at the end, and survive by consuming it throughout. At harvest time, the harvested grain is delivered to the Royal Granary, where it is put into storage. The court Scribe (in this case, the computer) is responsible for keeping track of such things, and immediately after the harvest, presents the ruler with a complete report on the resources of the kingdom, including the size of the harvest, the amount currently in storage, the acreage of land in the kingdom, and a complete census of the population: how many subjects are present, how many were added during the last year, how many were lost.



On the other hand, the fleet may come in with a shipment of grain for the kingdom from the outer provinces, just as supplies are running low, or the exploration of neighboring lands may result in a border expansion, increasing the kingdom's acreage and population accordingly.

The odds of any of these occurances happening in a given year are known, but the ruler has no direct control over them; the distribution is relatively random. This makes for an interesting (and sometimes frustrating) game.

There are, however, other factors which are at least somewhat under the ruler's control. If the population is underfed, there will be starvation in the land; if enough people get hungry, there may be a food riot, which is a form of social p:otest in which the people break into the Royal Granary and make off with whatever is to be found (not to mention calling the ruler a few dirty names in passing). Conversely, if the ruler overfeeds the populace, it tends to increase by an amount proportional to the surplus (the exact scientific explanation for this effect has yet to be found, but it is generally referred to as the 'Adam and Eve Law'). There are additional fine points to the game which can go without specific mention here; suffice it to say that there are enough to make for an interesting game.
Now that you have a general idea of what KINGDOM is all about, it is time to get into what is perhaps the most interesting characteristic of the game: its effect on the people who play it.


The true history of the game of KINGDOM began in the fall of 1972, when it appeared on a Honeywell 6060 system at the Milwaukee Technical College, implemented by one of the authors of this article, who, at that time, was an employee-student of the Physics department there. After a few weeks of working out the fine points of the game, it became 'available for public consumption' (which means that anyone who could get their hands on a copy of the program could play it). This immediately resulted in a number of problems, among them the fact that timesharing usage on the Honeywell went up by a noticeable amount. It was then that the strange effects of the game on people first became noticeable as well. Armed with rule sheets, portable calculators, and various touches of greed, a veritable army of students descended on the system, each with his or her own 'infallible system' for winning. As it turned out, none of these were truly infallible, but they did their best, to the point that a number of departments in the school banned the playing of KINGDOM during regular class hours. Sometimes it was just as much fun to watch someone else play the game; this was great sport, especially some of the Physics department staff, who would sit at the terminal mumbling 'greed! greed!' as they typed in various transactions, and cursed loudly as the Huns attacked and made off with hardearned wealth.

# Int lee sthuturu - toud burus 



Having pretty well saturated one school, a copy of the program was slipped into the Marquette University Computer Center in Milwaukee just before Christmas of ' 72 (arriving at the back door in a plain brown envelope in the dead of night), and, lo and behold, suddenly appeared on their timesharing system (which consisted of one antique IBM 360/30). Timesharing usage on the system went up a few hundred percent; mostly due to engineering students who, after a hard day slaving over a hot slide rule (remember them?), would relax by trying to rule the world a few times. The staff members at this comp center did not ban the game, nor did they complain about excessive computer usage - they were too busy playing the game themselves.

Out of all this playing came an interesting development: a computer was programmed to play the KINGDOM game by itself. Though this may seem like a silly thing for a big computer to be doing, there was a definite reason.

It seems that with so many people playing and trying to win the game (and sometimes succeeding), there was constant discussion and disagreement on what was the 'best' strategy for success. Some players believed that playing conservative was the answer: they often had reigns which lasted over 100 'years', carefuilly maintaining a stable population and depending on the natural course of events to eventually allow for a win. Others preferred the reckless style of play; gambling on long odds with every thing they had, neglecting the populace in the interest of a quick profit, keeping nothing in reserve for hard times. The latter style of play (which became known as KAMIKAZE KINGDOM) resulted in an occasional win in record time, but with a majority of games ending in spectacular wipeouts due to one nautral disaster or another.
The former camp believed that consistent winning was the object of a successful strategy; the latter insisted that the quick win was the thing. Neither group could come up with a definite strategy which achieved either goal, and, with the randomness involved in each game, a meaningful comparison of two strategies could not be made unless a terrifically large number of games could be played using each strategy, which nobody (ourselves included) was willing to do.

Enter the computer. To resolve the argument, the original authors of KINGDOM (namely us) spent most of the summer of ' 73 writing and rewriting a monsterous program called RAMSES. This program, which was run on Marquette University's Engineering Department's old (and now defunct) IBM 7040 computer system, played the game of KINGDOM by itself, blinking its console lights furiously as it planted, harvested, bought and sold. Running at a rate of one game every $5-10$ seconds, RAMSES played the game in groups of 100 games each (called Dynasties), looking over its won/lost record at the end of each Dynasty and modifying its strategy accordingly to try to improve it. By changing the parameters of the RAMSES program, the computer could be instructed to play either conservative or liberal; 'success' could be defined as either consistent winning or fast winning of games. A sub-program of RAMSES called PLAYER simulated the human player, and another sub-program called MASTER kept track of the game records and statistics, and based on these, juggled the parameters automatically in search of improving the play.

Ten Dynasties made an Era, and at the end of each Era, the computer spouted forth masses of printout describing how the games were going, what strategy was being used, and the factors used for making decisions, the shortest successful game during each Dynasty and the longest, and various other information, which after an average day's run, amounted to a pile of printout about one foot high.

Occasionally, programming bugs in RAMSES would result in some interesting results; like the time the random number generator got stuck and a game played for 15000 years (even RAMSES himself couldn't do that), or the time the population was inadvertently set at zero, resulting in the computer running through each game trying to get rid of all of the kingdom's population, then spending great amounts of time at the end of each Dynasty trying to figure out why it wasn't winning any games.

By the end of the summer, various versions of RAMSES had played, as close as we are able to compute, about 1.5 million games of KINGDOM. We never did find any 'sure win' strategies, though the computer did develop a mode of play which generally resulted in a better track record than the average human player: For a numberof months, RAMSES held the school record for both the shortest and longest winning games. The computer also found a number of interesting tactics for playing, most of which had been previously discovered by human players (such as the trick of converting all your grain into land when the population gets too large, enabling you to starve the populace and suffer no loss of grain due to food riots, as the land may be sold and re-converted to grain the next year).


Our experiences with the RAMSES program provided us with three things: (1) an interesting experience in the construction of a self-learning, self optimizing program, (2) the satisfaction of knowing that we [and the computer] had played more games of KINGDOM than anyone in the place, and (3) a lifetime supply of scratch paper to be written on the back of.

We discussed our project with a number of people during that time, among them the local representative of DEC, who thought it might be an interesting item to write up for the DEC educational computer magazine, EDU. Since one of the few things we enjoyed more than playing KINGDOM at that time was talking about playing KINGDOM, a short article got written up and sent off to the DEC office for them to look over, and eventually appeared in EDU No. 12 (Summer, 1974). At the end of the article, we offhandedly mentioned that if anyone wanted a copy of the program listing for KINGDOM, they could drop us a line requesting KINGDOM along with a stamped, self-addressed envelope and we would send them one. After all, why not share the game with other unfortunate computer freaks who had never experienced KINGDOM before?

Our mistake was in underestimating the number of computer freaks in the country. For the next four months after the article came out, we spent our evenings folding up listings and mailing them to New York, Virginia, Illinois, Colorado, California, Arizona, Texas, and half the provinces of Canada. We even sent a copy to a Navy Lieutenant in the Pentagon, where he put it onto an HP2100C minicomputer system and wrote back telling us about all the Army majors and colonels who gathered at the computer center every lunch hour and practiced ruling the world (yes folks, that's our defense department at work): the only complaint he received was that the Army brass got annoyed when the Huns attacked and they had no retaliatory strike power (as one player remarked, "the least the computer could do is let us bomb the bastards!").


At the present time, we can report that KINGDOM is certainly alive and well in a number of computer centers around the country, satisfying that basic rule-the-world urge for amateur Napoleon and Attilas and Marie Antoinettes everywhere.

The last time we saw the game being played at Marquette, it was running as the second most popular game in the center, right behind the Star Trek game *SPACEWAR* (perhaps this is because SPACEWAR lets the player rule the whole galaxy, whereas the KINGDOM player must be content with just the earth). The University of Wisconsin - Milwaukee has a number of versions running on its DEC TSS/8 system, including a recently added version written in Spanish for the benefit of students and visitors who are more comfortable with that language; other foreign language versions are in the planning.

Some versions of KINGDOM on the larger computer systems have been provided with the option of 'save' the status of a game in case the player has to leave in the middle, such that he may return and pick up where he left off an hour later or the next day. A number of modifications have been made in the game itself by various indivduals, but there would be insufficient space to list all the variations here.

One final note, though: since KINGDOM was first introduced several years ago, we have been accumulating notes on possible modifications, new options, and various modifications which could be made in the game to make it even more interesting and/or challenging. If time ever permits, we hope to sit down someday and write a game of 'super-KINGDOM' (tentatively named EMPIRE), which would include such items as agricultural development, mining and manufacturing, trading and/or warfare with neighboring kingdoms, a monetary system, military and police forces, civil service, and perhaps even a multi-player option where two or more players compete (through the computer) for control of the resources. In the meantime, though, there are other things to be taken care of, programming to be done, bills to be paid, and so forth: and by the way, we see the lunch hour is approaching - time to go and rule the world again!

Lee L. Schneider
Robotics and Artificial
Intelligence Laboratory
University of Wisconsin - Milwaukee
Todd L. Voros
A. O. Smith Corp.

Data Systems Division Milwaukee

10 REM＂KINGDOM＂FOR TSS／8 BASIC
11 KEM
12 KEM
14 KEM
15 KEM
16 REM
17 REM
VERSION 1 MOD 2 12／28／73
$\begin{array}{ll}\text { hriITEEN } & \text { BY：} \\ \text { I．} & \text { VOROS }\end{array}$
REM I．VOKOS
8 REM
L．SCHNEIDER A．O．SMITH CORP．
29 RANDOMIZE
25 DEF FNR（21）＝INT $(1 N T(21)$ QRND $(8))$
30 DEF FNL $(Z 2)=F N R(108)-Z 2$
35 REM
98 REM＊＊＊SET INITIAL PARAMETERS＊＊＊
99 REM
190 LET $Y=0$
105 LET L $\theta=1016$
118 LET LI＝3
115 LET $\mathrm{NJ}=\mathrm{FNR}(75)+75$
120 LET L2＝FNR $(250)+250$
125 LET NI＝FNR $(3006)+2000$
130 LET L3＝8
135 LET N2＝0
136 PRINT
137 KEM
138 REM＊＊＊PKINT YEAKLY REYORT＊＊＊
134 KEM

145 PRINT
IS日 PNINT MREPORT FOR YEAR＂IY
155 PRINT
160 PRINT＂POPULATION IS＂JNG
165 PRINT＂ACNES OF LAND OWNED＂HLZ
170 PRINT＂BUSHELS IN STORAGE＂INI
175 PRINT＂PRICE OF LAND IS＂ュLIJ＂BUSHELS PER ACRE＂
188 PRINT
198 REM
191 REM＊＊READ AND VERIFY LAND TRANSACTIONS＊＊＊
192 REM
215 PRINT＂HOW MANY ACRES TO BUY？＂
228 INPUT B
225 LET $B=1 N T(B)$
230 IF B＜0 GOTO 2026
235 IF $\mathrm{B}=0$ GOTO 270
248 LET $A=N I-B+1$
245 IF $A>=O$ OOTO
245 IF $A>=0$ G010 260
250 PRO YOUR STORAGE IS ONLY＂JNII＂BUSHELSI＂
255 GOTO 215
260 LET NI $=A$
265 LET L2 2 L2＋B
278 PRINT＂TO SELL？＂
275 INPUT C
288 LET $C=I N T(C)$
285 IF C＜i GOTO 2420
298 IF $\mathrm{C}=\mathrm{A} G \mathrm{GOTO} 325$
295 LET $A=L 2-C$
309 1F $A>=1 才$ GOTO 315
385 PRINT＂YOU ONLY OWN＂IL2；＂ACRESI＂
310 GOTO 270
315 LET L2＝A
329 LET N1＝NI＋C＊LI
325 IF $(B+C)=6$ GOTO 335
330 PRINT＂TRANSACTION RESULTS：LAND＂JL2：＂ACRESS GRAIN＂INII＂BUSHELS＊
332 REM
333 REM＊＊＊TEST FOR WIN＊＊＊
334 REM
335 IF $(L 2+N I-L \theta)>=0$ GOTO 2010
340 IF $\mathrm{NJ}=\mathrm{H}$ GOTO 490
348 REM
342 REM
343 REM＊e\＃NEAD AND VERIFY FOOD AND SEED ALLOTMENT＊＊＊
344 REM
345 PRINT
358 PRINI＂HOW MANY BUSHELS FOR FOOD？＂
355 INPUT N2
$360 \mathrm{~N} 2=1 \mathrm{NT}\left(\mathrm{N}_{2}\right)$
365 IF N2 20 © 60102020
3715 LET $A=N 1-N 2$
375 1F $A>=0$ GOTO 390
380 PRINT＂YOUZ SIOKAGE IS ONL＇Y＂INIJ＂BUSHELSI＂
385 GOTO 35
390 LET NI＝
395 IF NI $=0$ GOTO 490
4W＠PRINT＂HOW MANY ACRES TO BE PLANTED？＂
405 INPUT L3
410 LET L3＝INT（L3）
415 IF L3＜8 GOTO 242
420 IF $13=0$ GOTO 590
420 IF L3
425 IF（L2－L3）$>=i 0$ 6010 440
43才 PRINT＂YOU ONLY OWN＂sLZנ＂ACRES！＂
435 GOTO 480
440 If（2＊N0－L3）$>=0$ G0TO 455
445 PRINT＂YOUR POPULATION IS ONLY＂INE $"$ PEOPLEI＂
450 GOTO 48v
455 LET $A=N 1-34$ L 3
460 IF $A>=0$ GOTO 475
A65 PNINT＂YOUR STORAGE IS ONLY＂JNII＂BUSHELSI＂
4715 GOTO 49E
475 LET N1＝A
488 IF N1＞8 GOTO 500
485 PRINT
499 PRINT＂STORAGE IS NOW EMPTY．．．GOOD LUCKI＂
491 REM
495 REM＊＊＊COMPUTE EFFECTS OF FOOD ALLOTMENT＊＊＊
497 NEM


The game of KINGDOM is based upon the premise that the player is to act as the ruler of a mythical，medieval country．The ruler must make decisions which maintain the kingdom and advance the player toward the goal．

The goal of the player is to rule the world．This is accomplished by collecting a total wealth equal to，or exceeding，one million units．Wealth is measured in terms of two quantities：acres of land owned，and bushels of grain in storage．

At the start of the game，the player is assigned an initial population，a parcel of land，and a storage house of grain．At the beginning of each year，the player must make decisions which will concern the transaction of land and grain，allot food to the populace，and specify the amount of land to be planted．A yearly report is printed at the beginning of each year，giving the total resources of the kingdom at that time．

If the player makes a decision which exceeds the limits of the total resources at that time，the player is informed of the error and the question is repeated．

The player wins at any time when the total wealth （acres plus bushels）equals or exceeds，one million units． The player loses if the total population reaches a value of zero．

Game constants are of three basic types．The first are absolute constants，which are fixed at the same value through－ out the play of the game．The second type are factors which are always present，but change on a fairly random basis each year．The third type is usually referred to as random occurrence， which governs a number of happenings which may or may not occur in a given year，based upon a standard fixed percentage．

All user input is in the form of numeric values．The user may terminate the play at any time by entering a negative value in response to any question．

```
s0g LET Y=Y+1
S&5 LET N3=FNR (N0/3*3)
510 LET NA=FNR(NO/A+2)
S15 LET NS=NH=IN1 (N2/10.
520 IF NS>0 GOTO 566
52S IF NS=0 GOTO 59%
52S IF NS=0 GOT A=FNR(3-N5/2)
S35 LET A=
540 PRINT "***FOOD SURPLUS*** POPULATION INCREASE "IA
SA! PRINT NS=#*
550 LET N3=N3 *A
55 GOTO 59%
560 IF (FNL(S*(NS-2)))>0 GOTO 598
S65 LET A=FNR((N5*N1)/(2*N0))
57% PRINT
578 PRINT "***FOOD RIOTS*** "3AI" BUSHELS LOST"
586 LET NI=NI-A
590 1F FNL(10) >8 GOTO 620
595 LET A=INT (N|/3) +FNN'(N|/2+2)
597 REM
598 KEM ***COMPUTE RANDOM OCCURANCES***
599 REM
600 PKINT
605 PRINI "e**RLLAGUE*** "BAJ" DIED"
605 PRIN NA=NA+A
610 LET NA:NA+A
620 IF FNL(18) >0 GOTO 670
625 LET A=FNR(N|/S)+1NT (N|/S)
630 LET B=FNK (N1/20) +INT (N1/20)
635 LET C=FNR(L2/50)
6 4 0 ~ P R I N T ~
6AS PRINT "***HUNS ATTACK***"
650 PKINT AJ" PEOPLE KILLED "נB," BUSHELS TAKEN "IC3" ACRES DESTROYED"
6 5 5 ~ L E T ~ N A = N A + A ~
660 LET NI=NI-B
665 LET L2=L2-C
6 7 8 ~ I F ~ F N L ( 1 5 ) > 0 ~ G O T O ~ 7 8 5 ~
675 LET A=FNR(2b) +16
68| LET B=FNR(N| +258) +56
6 8 5 ~ P R I N T ~
690 PRINT "***BORDER EXPANSION*** YOU GAINED "BAS" PEOPLES "IBS" ACRES"
695 LET N3=N3+A
708 LET L2=L2+B
705 LET NG=NB +N3-NA-NS
71! IF ND <=|\ GOTO 2015
715 PRINT N3," ARRIVED**
725 PRINT NS{" DIED OF STARVATION"*
73B PKINT NA'" DIED NATUNAL CAUSES"
736 PKINT NAI", DIED NATURAL CAUSES
735 1F FNL(10)>(8 G010 760 
74S PRINT
O5 PKINT "**THEFT*** "BAZ" BUSHELS STOLEN**
755 LET NI=NI -A
760 IF FNL(18) >8 GOTO 785
765 LET A=FNR (L2/10)+INT (L2/20)
7 7 0 \text { PRINT}
27S PRINT "**********EARTHQUAKE********** "IAI" ACRES DESTROYED"
780 LET L2=L2-A
785 IF FNL(IS)>8 GOTO B18
785 IF FNL(IS) >8 G0T0 810
790 LET A
795 PRINT
80日 PRINT "***GRAIN SHIPMENT ARRIVES*** "IAJ" BUSHELS"
8 0 5 ~ L E T ~ N 1 ~ = N I + A ~
810 IF FNL(15)>0 GOTO 835
815 LET LI =FNK(2)+1
82g PKINT
825 PRINT "***DKOUGHT***"
```



```
835 IF FNL(15) >0 GOTO 852
840 LET LI*FNK(3)+7
8AS PRINT
850 PRINT "***RAIN*&&"
8 5 1 ~ G O T O ~ 8 6 0 ~
B52 LET LI=FNR (A) +3
855 REM
855 REM
856 REM ***COMPUTE HARVEST ***
857 REM
860 LET A=LI *L3
865 LET B=FNR(A/2)
876 LET C=A-B
8 7 5 \text { PRINT}
88O PRINT
88S IF A=g GOTO 910
894 PRINT "HANVEST WAS "JLI|" BUSHELS PER ACRE FOR A TOTAL "JAS" BUSHELS"
895 PRINT "LOSS TO KATS "sBr" BUSHELSS NET HARVEST WAS "JC," BUSHELS"
90% LET NI=NI +C
903 GOTO 130
910 PRINT "HARVEST WAS "HLI&" BUSHELS PER ACRE FOR A TOTAL "JAJ" BUSHELS"
915 GOTO 920
29.0 REM
2001 REM ***SPECIAL MESSAGES FOK ENDINGS****
2002 REM
2010 PKINT
2011 PRINT "***CONGRATULATIONS*** YOU NOW RULE THE WORLDI"
2012 GOTO 2020
20.15 PRINT
2016 PRINT "***DIASTER*** - THERE ARE NO MORE PEOPLE LEFTI"
2026 PRINT
gW2! PRINT "YOUR REIGN LASTED "JY," YEANS"
20&1 ERN
```

The following are purely random factors: (the probability of the event'soccurrence in any givenyear is given in parenthesis): PLAGUE - ( $10 \%$ ) - may reduce population up to $80 \%$.
THEFT - ( $10 \%$ ) - may reduce grain storage by up to $10 \%$.
EARTHQUAKE - $(10 \%)$ - may reduce acreage by up to $15 \%$.
ATTACK BY HUNS - $(10 \%)$ - may reduce population up to $40 \%$, storage up to $10 \%$ and land up to $2 \%$.
RAIN - ( $15 \%$ ) - increases harvest to 7,8 or 9 bushels/acre. DROUGHT - ( $15 \%$ ) - decreases harvest to 1 or 2 bushels/acre.
GRAIN SHIPMENT - $(15 \%)$ - increases grain storage - depends on the size of the kingdom (acres of land).
BORDER EXPANSION - $(15 \%)$ - increases population and land holdings - depends on present population of kingdom.

The following are absolute game constants:
FOOD - Each subject of the kingdom requires a minimum of 10 bushels of grain per year for use as food.
SEED - Each acre of land planted requires 3 bushels of grain per year for use as seed.
LABOR - Each subject may plant a maximum of 2 acres per year.
The following are variable factors:
LAND PRICE - (expressed as bushels per acre) - normally varies between 3 and 6 bushels per acre, unless affected by the occurrence of rain or drought. The harvest per acre for a given year becomes the trading value for the following year.
POPULATION - normally varies by 20 to $40 \%$ due to natural causes, unless affected by other natural occurrences.
GRAIN LOSS - from 0 to $50 \%$ of the harvest of a given year may be lost to rats. Only the harvest (not existing grain in storage) is susceptible to attack.
OVERFEEDING - a food surplus occurs when more grain is alloted for food than the minimal requirement. There is about a $50 \%$ probability of gaining one person for each 20 bushels over the minimum.
UNDERFEEDING - the ruler may starve off 2 persons per year without ill effect. Each starvation over 2 increases the probability of food riots by $5 \%$ each. A food riot may result in a loss of up to $50 \%$ of the grain in storage.


## CHOMP for 8008

## CHOMP for BAsic

The game of CHOMP was introduced in Martin Gardner's Mathematical Games Department of Scientific American for January, 1973. It was invented by David Gale at UC Berkeley.
This version is our first pass. We changed the rules somewhat from the published version. Not only that, but this version doesn't play the game at all - it just keeps track of the moves and informs the loser (who already knows he lost, anyway).
Scientific American has described some winning strategies for boards of given sizes. In our version the players INPUT the dimensions - makes it harder to figure out the trivial wins. The rest of the way our program works should be cleat from the directions. We're still finding out how kids solve the puzzle of deciphering printed rules to games (this is a game in itself sometimes). The program runs on HP 2000 series, and on 8K PDP8/L with Edu 20.55.

Peter Lynn Sessions
Published in PCC. Volume 1, Number 3.

Any number of players can play CHOMP. It is played (in this version) on an 8 by 8 board of +'s and "'s in the upper right hand corner. Players take turns chomping until one player is forced to take the poison *. To take a Chomp, you type in the letter and number of one of the +'s. That + , and all of the +'s to the left and below disappear. No fari chomping empty space.
This version of CHOMP uses the Mark-8 with the TV Typewriter I and an ASCII keyboard. The TVT is on output port 1, and the keyboard is on input port 0 . Strobe lines are appropriately connected (as shown in the diagram). First load the "Load Comments" program then interrupt with a Restart 005. Now set the switch register to 300 (NOP) and type in the comments using returns and line feeds as shown. The output 0 LED's will count characters as they are loaded. LED's should read 365 when done.
Now load the program from 0000 to 0164 . Interrupt with 005 , then set switch register to 300 , home TVT and press any key. When the game is over clear the screen and hit any key to play again. Have fun!

May 20, 1975

Phil Mork
12 Woodland Road
Weston, MA. 02193

Published in Micro 8 Computer User Group Cabrillo Computer Center
4350 Constellation Rd., Lompoc, Ca. 93436


* Remove jumper!
** Remove cursor C16

Load Coneents
1000066 LLI
$\begin{array}{lll}000 & 066 & \text { LLI } \\ 001 & 000 \\ 002 & 058 & \text { LHI }\end{array}$
$\begin{array}{ll}002 & 058 \\ 003 & 001\end{array}$
$\begin{array}{ll}003 & 001 \\ 004 & 306\end{array}$
$\begin{array}{lll}004 & 306 \\ 005 & 121 & \text { LAL } \\ 005 \\ 005 & 000 & \end{array}$
$\begin{array}{lll}005 & 121 & \text { NUT } \\ 005 & 000 & \text { HLT }\end{array}$
006
007
101
INP 000
$\begin{array}{llll}007 & 101 \\ 010 & 123 & \text { OUP } & 000 \\ 011 & 370 & \text { IMA }\end{array}$
011370 LMA
$\begin{array}{lll}012 & 060 & \text { IML } \\ 013 & 104 & \text { JMP }\end{array}$
013104
014004
014004
015000



THIS GAME USES A MASKING TECHNIQUE AND SEVERAL RANDOMIZING SEQUENCES. HAS LOTS OF ACTION TO KEEP THE PLAYER INTERESTED. MACHINES WITH BETTER STRING FACILITY COULD PROBABLY REDUCE THE PROGRAM A BIT.

1 REM "WITCHY THE NAME WITCH" BY THE OLD SOLDIER (WALTER E WALLIS) 2 REM RIGHTS SHARED WITH P.C.C. TO BENEFIT COMMUNITY COMPUTER CENTER

4 REM SET UP FOR PDP-8L, EDU20-C
5 PRITPRITPRI

## $\ll$

8 RANDOM
$9 \mathrm{Z}=1$
10 PRI"HELLO THERE, DEARIE, I AM WITCHY THE NAME WITCH!*
15 PRI"I HAVE CAPTURED FOUR OF YOUR FRIENDS AND AM HOLDING THEM" $2 \emptyset$ PRI"FOR RANSOM. IF YOU WANT ME TO RELEASE THEM, YOU MUST" 20 PRI"FOR RANSOM. IF YOU WANT
25 PRI"TELL
30 GOSUB 108
$\begin{array}{lll}30 & \text { GOSUB } 108 \\ 34 & \text { IF } Z>1 \text { THEN } 100\end{array}$
35 PRI"THANK YOU DEARIE, "ES", CACKLE, CACKLE"\FORI=1TO2O\PRICHRS(7)
36 NEXT I
40 PRI"YOU FOOL, IT WASN'T YOUR FRIENDS I WANTED," 3
41 PRI"IT WAS THEIR NAMESII!"
45 PRI"AND YOU GAVE THEM TO ME"CHRS(7)" NOW UNLESS YOU CAN GUESS" 50 PRI"WHERE I HAVE HIDDEN THEM, YOUR FRIENDS WILL HAVE TO GO THROUGH"

55 PRI"LIFE WITHOUT NAMES. HE HA HO HU HEE HAAI!!!! !"
56 FOR I=1 TO 10 \PRI CHRS (7)3 \NEXT I
60 PRI
65 PRI"I HAVE HIDDEN EACH NAME BEHIND A FENCE THAT LOOKS LIKE THIS: "
67 PRI \PRITAB ( 30 )"XXXXXX"CHRS ( 13 )TAB (30) "000000"\PRI
69 PRI"THERE ARE 6 BOARDS IN EACH FENCE. IF A BOARD FALLS DOWN"
71 PRI"(AND ONE ALWAYS DOES) YOU CAN SEE ONE LETTER OF THE NAME" ${ }^{\prime \prime}$ 72 PRI"BEHIND" 3
73 PRI "THE FENCE. (I CUT OFF ALL LETTERS OVER 6 AND ATE THEM)"
74 PRI"IF YOU GUESS WHICH NAME IS BEHIND A FENCE, I LET THAT NAME GO.
76 PRI"IF YOU GUESS WRONG, I EAT THAT NAME.
78 PRI"IF YOU PICK MY FENCE BY MISTAKE I EAT YOUR NAME AND THE GAME" 80 PRI"IS OVER, YOU LOSE."
$8 \%$ PRI"IF YOU GUESS MY FENCE ON PURPOSE, I EAT MY NAME AND YOU WIN." 84 PRI "NAMES MUST BE SPELLED EXACTLY THE SAME, OR THEY DON "T COUNT." 86 GO TO 130
100 PRI"WHAT IS YOUR FIRST FRIEND'S NAME" ${ }^{\prime \prime}$ IINPUT AS 105 PRI"WHAT IS YOUR SECOND FRIEND'S NAME"I INPUTBS 10 PRI"WHAT IS YOUR THIRD FRIEND'S NAME" IIENPUT CS 15 PRI"WHAT IS YOUR FOURTH-FRIEND 'S NAME" I IINPUT DS 120 PRI"AND WHAT IS YOUR NAME, DEARIE"MINPUT ES


The article referred to by Mr. Ivanhoe requires some explanation on my part as well as my own kind of apology. Perceptive readers may have noticed that the title and graphic treatment did not quite fit the contents of the article, which was a highly favorable review of the Altair 8800 hardware.

I had written a relatively unfavorable review of the Altair on the basis of inspection of an early model shipped to PCC. Based on what I and others saw in that unit, there would be noise problems. An early phone conversation with Ed Roberts of MITS led me to believe that he had given up trying to suppress the noise and was placing his hopes on the observation that the noise reduced somewhat as more modules were plugged in.

Noise is an elusive thing, and the design considerations involved in its suppression are of the sort that are more a matter of style than of rigorously defined technique. The only real way to determine the noise sensitivity of a design is to observe it in operation. The 8800 which we had been sent was incapable of doing anything except Dompier's music program, which works by using the pickup of the noise on a radio.

PCC was about to go to press when a call came in from Ed Roberts on another matter Since I consider myself a journalist as well as an engineer, I decided to read the article to Ed over the phone.

His answer was, essentially, that "the problem has been fixed and everyone has been told about it but you". I would be making a fool of myself if it went out as it was. With no way to check out Ed's statements before the deadline, I decided to rewrite the article to reflect Ed's statements.

I should have witheld the article until some confirmation could be obtained. I traded off my integrity for both my own convenience and for the presumed convenience of PCC. As a journalist I must apologize for this,

As an engineer, I feel that my apology should be of a more useful nature. Accordingly, I have been in vestigating the problems with the Altair 8800 and can make some statements and suggestions with confidence.

First - the Altair 8800 has no noise problem inherent in the bus, memory or processor that would preclude operation with reasonable reliability.

Second - the bulk of the problems with the 8800 can be traced to three sources:
(a) Inadequate power supply bypassing on the front panel circuit board.
(b) Interactions between dynamic memory refreshing and the processor during critical transient conditions such as direct memory access.
(c) Processor chips carrying the suffix " -8 ", which appear to have reduced margins, In several cases problems observed with these chips were resolved by substitution by others of the same type or of standard types.

Mr. Lee Felsenstein
People's Computer Company
Dear Mr. Felsenstein,
I read your article on page 22 in "P.C.C" vol, 3, issue S, and for a moment I thought you were writing not about MITS but another company. I am not aware of what is the source of your information but if it is the maker of ALTAIR, which I suspect, you are showing only one side of the story, now here is another side.

I read the advertisement of MITS in "Radio Electronics" February 1975 issue and took all their claims at face value and wrote to them iommediately on the 28th February, 1975 enclosing a bank draft for $\$ 665$ plus $\$ 15$ airfreight to cover the cost on one of their Altair 8800 computers and at the same time requesting information on interfacing their machine to a Facit input/output typewriter informing them that our electricity was 240 volts 50 cycles. On the 6th March I wrote again asking information on using Altair 8800 computer for numerical control on two and three axis machines with stepping motors On receiving no reply I wrote again on the 19th and 24th March and 16th May to find out what was happening about my computer. Eventually on the 10th June I received my computer by air, paid $\$ 60$ airfreight charges and noticed that no communication whatever accompanied this and no mention of the refund of the \$15 already paid for freight. When I opened the computer I found that the power supply was for 100 volts and three or four capacitors in the power supply had become undone from the wire leads As there was no physical damage to the parcel nor the computer this could onty indicate poor quality of the components I wrote to them explaining what had happened on the 10th June and again on the 11th August but to this date I have not received a reply.

It is quite apparent there is no agreement between your story and mine as far as service and quality of capacitors are concerned. As to the quality of the computer itself, so far I cannot say a thing as for obvious reasons I am not in a position to use it.

I feel that you have two alternatives: Either show this letter to somebody responsible at MITS in the hope that after you giving them such a beautiful and free write-up, they might feel morally obliged to prove your story correct, or, else publish my letter in order to keep faith with your readers, thus showing two sides of the story.

I will be obliged if you let me hear personally what action, if any, you choose to take.
yours faithfully,
Mark Ivanhoe
67 Blackshaw Avenue Mortdale, NSW 2223 AUSTRALIA

Third - I recommend the installation of 0.1 uf disc ceramic capacitors between the +5 volt and ground pins of IC's $\mathrm{U}, \mathrm{A}, \mathrm{X}, \mathrm{T}, \mathrm{B}$, and D of the front panel circuit board.

If dynamic memory is used, I recommend that the "reset" switch be conditioned so that the signal to the processor is free of bounce. One way to do this involves paralleling the switch with a 0.1 uf capacitor and adding components to the processor board to create a Schmitt trigger circuit using pins 1,2 , and 3 of IC R. (see schematic).

On the memory boards (dynamic only), I recommend connecting a jumper between pins 11 and 16 of IC 1 ( $74123 /$ AM26L123), as well as lifting pin 10 of IC T and jumpering it to pin 11.

For reliable operation of both dynamic and static memory in the same machine, I recommend using a capacitor no greater than 4700 pf for C 7 on the front panel board.
( 1 am making available a kit of parts and complete instructions for carrying out the above modifications. It is being marketed by Processor Technology Inc, 2465 Fourth St., Berkeley CA 94710 . Price is $\$ 3.00$ ).

Hal Singer's Micro 8 Newsletter contains another suggestion by Dave Tritsch, who relays a modification done on his Altair by Don Tarbell of the Southern California Computer Society. It involves breaking the connection on the front panel board from pin 1 of IC G to ground and jumpering pin 1 to pin 13. This is said to fix intermittent operation of the deposit switch

Letters to PCC have indicated various levels of responsiveness on the part of MITS to problems of purchasers. Missing parts seem to arrive much faster than responses to inquiries about shipments, and applications and interfacing information seems to have the lowest priority at MITS. This last observation seems reasonable, since "customer engineering" must pretty obviously be traded off in exchange for the low purchase price.

Readers should note that I am making no statements regarding MITS' software, nor am I imputing any motive to anyone involved in that corporation. PCC and I welcome any response that MITS would care to make to this article or the accompanying letter.

Lee Felsenstein
LGC Engineering
LGC Engineering
1807 Delaware St.
Berkeley CA 94703


## SIGNALSA.

## NEW THINGS no. 1.

A new 6800 -based microcomputer system was demonstrated at the Sept. 3 meeting of the Bay Area Homebrew Computer club. The manu-facturer-to-be wants to avoid being buried under a mountain of inquiries until things are a little further developed. Still, we can describe some of the features here.

Software features the standard 6800 features, including two accumulators, 72 basic instructions, 3 hardware and 1 software interrupt, as well as a memory-to-tape dump and loader program.

In hardware, the system will be based on plug-in cards and a bus structure, each memory card will contain 8 K and the power supply will be a 12 amp OEM monster. A serial TTY I/O port is included in the basic system, as are seven-segment front panel displays and a real time, 24 -hour clock (the front panel displays time when it isn't doing anything else.)

The front panel connects to input and output ports, and memory can be loaded from it. Binary and hexidecimal readout of data and address are provided, as are provisions for cyclestealing DMA ,

Got all that? Availability will be approximately December of this year, and price is yet to be announced.

## GLITCH COUNTER GLITCH

One problem with the Altair has two possible fixes, which could cancel each other out if done in the same machine.

George Morrow pointed out to me that the 8800 has two tri-state drivers trying to drive the PRDY line, pin 72 on the bus. One is located in the front panel card and the other is the one in whatever memory card is being addressed. If one is trying to pull up when the other is trying to pull down, trouble can result.

The two driver chips will sweat until one of them wears out under the strain (yes, IC's DO wear out when they get into impossible situations).

Two fixes are possible; either rewire the line from the front panel to pin 3 of the bus (XRDY - same function as PRDY but not used), or rewire the lines originally hooked to pin 72 on each memory card over to pin 3.

Processor Technology has taken the latter route, so that their boards will fit in existing Altairs. But the first fix is more desireable for the user because only one change has to be made.
If BOTH fixes are done, you're right back with the original problem: two drivers fighting each other on bus pin 3.

If you have or get an Altair 8800, you may want to put a sticker inside the case showing which fix has been done. Then make sure that any new cards plugged in fit with the fix.

To summarize: the wire from the front panel which originally went to pin 72 should be moved to pin 3, OR the lines on each card (except CPU) which originally went to pin 72 should be moved to pin 3 BUT NOT BOTH.

## by Lee Felsenstein

## INTERRUPT!!

Don Alexander of Ohio Spectronics writes to tell us that ha will have a vectored interrupt card available for the Altair 8800 around November 1.

The card will accept inputs from the Vectored Interrupt bus (VI) 0 through 71 and will set latches Which send the restart information onto the bus when the CPU asks for it. The latches have a priority structure
to allow servicing interrupts from fast devices more frequently than from slow ones.
$\$ 90$ for a Don quotes $\$ 110$ for an assembled card and sonnectore kit. AII IC's will have sockets, and the edge connectors will be gold-plated. If the response is good, says Don, he wilt make available a board containing four or eight programmable interrupt ports. This will enable software control of the relative interrupt priorities

It searns as if we're all going to have to start to learn about interrupts. No more excuses.

Don can be reached at Ohio Spectronics, 300 Colonial Avenue, Worthington OH 43085 .


## DRIVE A DRAGON

Bob Mullen is working on a plug-in card for the Altair 8800 which will provide 8 bits of input and output through relays and optical isolators.

The board is intended for use operating machinery such as mechanical dragons (the dragon driver?) and other equipment using high voltages or currents.

Bob will also be marketing an extender board for the Altair, useful for testing and troubleshooting other boards.

Bob can be reached through PO Box 6214, Hayward CA 94545.

## YES IT IS.....NO IT ISN'T

In the current issue of Computer Notes, David Bunnell of MITS makes a comparison of the power drawn by an unnamed "widely advertised" 4 K RAM board and finds that it uses 4 times the power of the Altair equivalent.

Gary Ingram of Processor Technology has sent an open letter to MITS challenging them to a detailed comparison of specifications and performance. Gary says that if it is the PT product that Bunnell is referring to, the power drawn is no more than $25 \%$ above that of the MITS dynamic board.


Here's a photo of the newest hobbyist modem, which will sell as a kit for under $\$ 100$. It's so new that we can't tell you who will be making it, but we can mention some of its capabilities:

- It was designed for hobbyist and experimenter use, and has very few adjustments. Adjustment requires an audio signal generator, frequency counter, and an oscilloscope or VOM.
- It has all the necessary acoustic hardware and case included (the transparent case shown was for display only).
- It will record to and play back from audio tape without critical speed sensitivity. This capability was designed in, not just added on.
- It will operate at up to 300 Baud with very good reliability. It has a switch-selectable "high band" transmit capability which makes possible unit-to-unit communication.
- It will be called the Pennywhistle 103.

Keep your eyes on this space and the major magazines for more information.

## NEW THINGS no. 2

Processor Technology has two new additions to their line of Altair 8800 plug-ins and add-ons ready to be announced.

One is the MB-1 Mother Board with bus One is the MB-1 Mother Board with bus
terminator. It replaces the four small mother boards with one huge one - $1 / 8$ inch thick and carrying power and ground traces inches wide.

The MB-1 includes terminating resistors which effectively damp out any little hiccup on the bus signals and produce a waveform of textbook quality.

The other is the VDM-1, or Video Display Module. This is basically a TV Typewriter (without keyboard) on a single card which plugs into the bus. The memory for the screen is contained on the card and is addressable for read and write by the processor.

The VDM has full upper-lower case display, 64 characters per line, 16 lines per screen, solid video inversion cursor (black characters on white background I which can be set for any and every character on the screen, and a unique scrolling capability which will let the display "crawl" without tying up the processor.

Contrary to the first published descriptions, the VDM does not use direct memory access (DMA) which ties up the bus when it is displaying. The processor is delayed 42 microseconds only when it tries to access the VDM's internal memory at an inopportune time.

More information is available from Processor Technology Inc., 2465 4th St., Berkeley CA 94710.


Academic in tone, well written, lengthy articles. The place to look for in depth treatment.

## Computer Hobbyist

P.O. Box 295

Cary, N.C. 27511
\$6/yr

Hard dense info for hard but not dense hobbyists. Is the premier source of 8 bit technical information, particularly hardware, and almost the sole source for 8008 goodies!

Micro-8 Newsletter
Cabrillo Computer Center
4350 Constellation
Lompoc, Ca. 93436
\$6/yr

The Digital Group appears to be becoming just another commercial kit source, but there is nevertheless much of interest in their publication for the experienced kit-builder.

The Digital Group Clearinghouse
Box 6528
Denver, Co. 80209
\$6/yr
Homebrew Computer Club Newsletter
P.O. Box 626

Mountain View, Ca. 94040
no set price/donation
Interface
11557 Sunshine Terrace
Studio City, Ca. 91604
$\$ 10 / \mathrm{yr}$.
BITS \& PIECES
San Diego Computing Society
Gary Mitchell
Box 35
Chula Vista, Ca. 92012
$\$ 2.50 / \mathrm{yr}$.

The home computer world can be loosely divided into three parts - those who process information, those who move information (eg. Hams) and those who store and retrieve it. This is the magazine for those who see the importance of a library in its widest sense.


This is the magazine of the Altair users group, but it has articles deserving of a much wider audience. If you can't get a copy from MITS - find someone with an Altair and borrowit.

Computer Notes<br>MITS<br>P.O. Box 8636<br>Albuquerque, N.M. 87108<br>comes with an Altair

# Disw by Dennis allison, 選, \& friends 

## SOME MOTIVATIONS

A lot of people have just gotten into having their own computer. Often they don't know too much about software and particularly systems software, but would like to be able to program in something other than machine language. The TINY BASIC project is aimed at you if you are one of these people. Our goals are very limited-to provide a minimal BASIC-like language for writing simple programs. Later we may make it more complicated, but now the name of the game is keep it simple. That translates to a limited language (no floating point, no sines and cosines, no arrays, etc.) and even this is a pretty difficult undertaking.

Originally we had planned to limit ourselves to the 8080 , but with a variety of new machines appearing at very low prices, we have decided to try to make a portable TINY 3ASIC system even at the cost of some efficiency. Most of the language processor will be written in a pseudo language which is good for writing interpreters like TINY BASIC. This pseudo language (which interprets TINY BASIC) will then itself be implemented interpretively. To implement TINY BASIC on a new machine, one simply writes a simple interpreter for this pseudo language and not a whole interpreter for TINY BASIC.

We'd like this to be a participatory design project. This sequence of design notes follows the project which we are doing here at PCC. There may well be errors in content and concept. If you're making a BASIC along with us, we'd appreciate your help and your corrections.

Incidentaliy, were we building a production interpreter or compiler, we would probably structure the whole system quite differently. We chose this scheme because it is easy for people to change without access to specialized tools like parser generator programs.

## THE TINY BASIC LANGUAGE

There isn't much to it. TINY BASIC looks like BASIC but all variables are integers There are no functions yet (we plan to add RND, TAB, and some others later). Statement numbers must be between 1 and 255 so we can store them in a single byte. LIST only works on the whole program. There is no FOR-NEXT statement. We've tried to simplify the language to the point where it will fit into a very small memory so impecunious. tyros can use the system.

The language design was done in consultation with the PCC Dragon. We asked him what he had to have to write a useful program, then we took some of it away. Some other things (computed GOTO labels, for example) come free with our proposed implementation and might be useful.

The boxes shown define the language. The guide gives a quick reference to what we will include. The formal grammar defines exactly what is a legal TINY BASIC statement. The grammar is important because our interpreter design will be based upon it.

## A SIMPLE TINY BASIC PROGRAM


$1 \emptyset \emptyset$ PRINT "POWERS"
NPUT N
$13 \emptyset$ IF $N<>\emptyset$ THEN GOTO $11 \phi$ $14 \varnothing$ END

IT'S ALL DONE WITH MIRRORS-......
OR HOW TINY BASIC WORKS
All the variables in TINY BASIC: the control information as to which statement is presently being executed and how the next statement is to be found, the returnaddresses of active GOSUBS-all this information constitutes the state of the TINY BASIC interpreter.

There are several procedures which act upon this state. One procedure knows how to execute any TINY BASIC statement. Given the starting point in memory of a TINY BASIC statement, it will execute it changing the state of the machine as required. For example,

100 LET $S=A+6$ C
would change the value of $S$ to the sum of the contents of the variable A and the interger 6 , and sets the next line counter to whatever line follows 100 , if the line exists.

A second procedure really controls the interpretation process by telling the line interpreter what to do. When TINY BASIC is loaded, this control routine performs some initialization, and then attempts to read a line of information from the console. The characters typed in are saved in a buffer LBUF. It first checks to see if there is a leading line number. If there is, it incorporates the line into the program by first deleting the line with the same line number (if it is present) then inserting the new line if it is of nonzero length. If there is no line number present, it attempts to execute the line directly. With this strategy, all possible commands, even LIST and CLEAR and RUN are possible inside programs. Suicidal' programs are also certainly possible.


## IMPLEMENTATION STRATGIES AND ONIONS

When you write a program in TINY BASIC there is an abstract machine which is necessary to execute it. If you had a compiler it would make in the machine language of your computer a program which emulates that abstract machine for your program. An interpreter implements the abstract machine for the entire language and rather than translating the program once to machine code it translates it dynamically as needed. Interpreters are programs and as such have their's as abstract machines. One can find a better instruction set than that of any general purpose computer for writing a particular interpreter. Then one can write an interpreter to interpret the instructions of the interpreter which is interpreting the TINY BASIC program. And if your machine is microprogrammed (like PACE), the machine which is interpreting the interpreter interpreting the interpreter interpreting BASIC is in fact interpreted.

## THY BASIC GRAMMAB

The things in boid face stand for themselves. The names in lower case represent classes of things. $\because=$ is read is defined as. The asterisk denote zero or more occurances of the object to its immediate left. Parenthesis group objects. $\epsilon$ is the empty set. I denotes the alternative (the
exclusive-or).
line::= number statement (C) | statement (C)
statement::= PRINT expr-list
IF expression relop expression THEN statement
GOTO expression
INPUT var-list
LET var = expression
GOSUB expression
RETURN
CLEAR
LIST
RUN
END
expr-list::= (string | expression) ( , (string | expression) ")
var-list::= var (, var)

- expression:: $=(+1-\mid \varepsilon)$ term $(1+1-)$ term $)^{*}$
term::= factor ( $(* 1 / / \text { factor })^{*}$
factor::= var | number | (expression)
var::=A|B|C ... |Y|Z
number::= digit digit*
digit:: $=0|1| 2|\ldots| 8 \mid 9$
relop:: $=<(>|=| \varepsilon)|>|<|=| \varepsilon)|=$
A BREAK from the console will interrupt execution of the program.


## OUCER RHFRBMEE GUDER POR TNWY BASIC

## LINE FORMAT AND EDITING

- Lines without numbers executed immediately
- Lines with numbers appended to program
- Line numbers must be 1 to 255
- Line number alone (empty line) deletes line
- Blanks are not significant, but key words must contain no unneeded blanks
- ' - ' deletes last character
- $\mathbf{X}^{\mathrm{C}}$ deletes the entire line


## EXECUTION CONTROL

CLEAR delete all lines and data
RUN run program
LIST list program
EXPRESSIONS

## Operators



## INPUT / OUTPUT

PRINT $X, Y, Z$
PRINT 'A STRING'
PRINT 'THE ANSWER IS'
INPUT $X$
INPUT $X, Y, Z$

## ASSIGNMENT STATEMENTS

LET $X=3$
LET $X=-3+5 . Y$
CONTROL STATEMENTS
GOTO $X+10$
GOTO 35
GOSUB $x+35$
GOSUB 50
RETURN
IF $X>Y$ THEN GOTO 30


This multilayered, onion-like approach gains two things: the interpreter for the interpreter is smaller and simpler to write than an interpreter for all of TINY BASIC, so the resultant system is fairly portable. Secondly, since the major part of the TINY BASIC is programmed in a highly memory efficient, tailored instruction set, the interpreted TINY BASIC will be smaller than direct coding would allow. The cost is in execution speed, but there is not such a thing as a free lunch.


## LINE STORAGE

The TINY BASIC program is stored, except for line numbers, just as it is entered from the console. In some BASIC interpreters, the program is translated into an intermediate form which speeds execution and saves space. In the TINY BASIC environment, the code necessary to provide the transformation would easily exceed the space saved.

When a line is read in from the console device, it is saved in a 72 -byte array called LBUF (Line BUFfer). At the same time, a pointer, CP, is maintained to indicate the next available space in LBUF. Indexing is, of course, from zero.

Delete the leading blanks. If the string matches the BASIC line, advance the cursor over the matched string and execute the next IL instruction. If the match fails, continue at the IL instruction labeled ld.

The TINY BASIC program is stored as an array called PGM in order of increasing line numbers. A pointer, PGP, indicates the first free place in the array. $\mathrm{PGP}=0$ indicates an empty program; PGP must be less than the dimension of the array PGM. The PGM array must be reorganized when new lines are added, lines replaced, or lines are deleted.

Insertion and deletion are carried on simultaneously. When a new line is to be entered, the PGM array searches for a lirie with a line number greater than or equal to that of the new line. Notice that lines begin at PGM (0) and at PGM ( $\mathrm{j}+1$ ) for every j such that PGM $(\mathrm{j})=$ [carriage return]. If the line numbers are equal, then the length of the existing line is computed. A space equal to the length of the new line is created by moving all lines with line numbers greater than that of the line being inserted up or down as appropriate. The empty line is handled as a special case in that no insertion is made.
byte in memory treated as an integer
byte treated as a character
Iop PRINTU"POWER S" (3) "\% I NPUTUN (3) ${ }^{12} P R 1$ $N T \sim N * N, N * N * N$ (3) ${ }^{1 / 3}$ I $F$ N<> ЯUTHENшGOTON1


ERRORS AND ERROR RECOVERY
There are two places that errors can occur. If they occur in the TINY BASIC system, they must be captured and action taken to preserve the system. If the error occurs in the TINY BASIC program entered by the user, the system should report the error and allow the user to fix his problem. An error in TINY BASIC can result from a badly formed statement, an illegal action (attempt to divide by zero, for example), or the exhaustion of some resource such as memory space. In any case, the desired response is some kind of error message. We plan to provide a message of the form: 1 mmm AT nnn
where mmm is the error number and nnn is the line number at which it occurs. For direct statements, the form will be:

1 mmm
since there is no line number.

## Some error indications we know we will need are:

## 1 Syntax error

2 Missing line
3 Line number too large
4 Too many GOSUBs
5 RETURN without GOSUB
6 Expression too complex
7 Too many lines
8 Division by zero


ONE POTENTIAL IL ENCODING


The execution routine is written in the interpretive language, IL. It consists of a sequence of instructions which may call subroutines written in IL, or invoke special instructions which are really subroutines written in'machine language. Two different things are going on at the same time. The routines must determine if the TINY BASIC line is a legal one and determine its form according to the grammar; secondly, it must call appropriate action routines to execute the line. Consider the TINY BASIC statement: GOTO 100
At the start of the line, the interpreter looks for BASIC key words (LET, GO, IF, RETURN, etc.) In this case, it finds GO, and then finds TO. By this time it knows that it has found a GOTO statement. It then calls the routine EXPR to obtain the destination line number of the GOTO. The expression routine calls a whole bunch of other routines, eventually leaving the number 100 (the value of the expression) in a special place, the top of the arithmetic expression stack. Since everything is legal, the XFER operator is invoked to arrange for the execution of line 100 (if it exists) as the next line to be executed.

Each TINY BASIC statement is handled similarly. Some procedural section of an IL program corresponds to tests for the statement structure and acts to execute the statement.

## ENCODING

There are a number of different considerations in the TINY BASIC design which fall in this genera category. The problem is to make efficient use of the bits available to store information without loosing out by requiring a too complex decodinq scheme.

In a number of places we have to indicate the end of a string of characters (or else we have to provide for its length somewhere). Commonly, one uses a special character (NUL $=00 \mathrm{H}$ for example) to indicate the end. This costs one byte per string but is easy to check. A better way depends upon the fact that ASCII code does not use the high order bit; normally it is used for parity on transmission. We can use it to indicate the end (that is, last character) of a string. When we process the characters we must AND the character with 07 FH to scrub off the flag bit.

The interpreter opcodes can be encoded into a single byte. Operations fall into two distinct classes--those which call machine language subroutines, and those which either call or transfer within the IL. language itself. The diagram indicates one encoding scheme. The CALL operations have been subsumed into the IL instruction set. Addressing is shown to be relative to PC for IL perations. Given the current IL program size, this seems adequate. If it is not, the address could be used to index an array with the ML class instructions.

## TINY BASIC INTERPRETIVE OPERATIONS

TST lbl,'st
CALL lbl

RTN Return to the IL location specified by the top of the control stack.

DONE

INSRT

XINIT
delete leading blanks
If string matches the BASIC line, advance cursor over the matched string and execute the next IL instruction. If a match fails, execute the IL instruction at the labeled Ibl.

CALL Ibl

Report a syntax error if after deletion leading blanks the cursor is not positioned to read a carriage return.

Continue execution of IL at the label specified.
Print characters from the BASIC text up to but not including the closing quote mark. If a cr is found in the program text, report an error. Move the cursor to the point following the closing quote.
Print number obtained by popping the top of the expression stack.

Insert spaces to move the print head to next zone.
Output CRLF to Printer.
If the present mode is direct (line number zero), then return to line collection. Otherwise, select the next sequential line and begin interpretation.
Test valugat the top of the AE stack to be within range. If not, report an error. If so, attempt to position cursor at that line. If it exists, begin interpretation there; if not report an error.

Place present line number on SBRSTK. Report overflow as error.

Replace current line number with value on SBRSTK. If stack is empty, report error.
Compare AESTK(SP), the top of the stack, with AESTK (SP-2) as per the relation indicated by AESTK(SP-1). Delete all from stack. If condition specified did not match, then perform NXT action.

Read a number from the terminal and push its value onto the AESTK.

Return to the line collect routine.
Report syntax error and return to line collect routine. Replace top two elements of AESTK by their sum.
Replace top two elements of AESTK by their difference.
Replace top of AESTK with its negative.
Replace top two elements of AESTK by their product.
Replace top two elements of AESTK by their quotient.
Place the value at the top of the AESTK into the variable designated by the index specified by the value immediately below it. Delete both from the stack.

Test for variable (i.e. letter) if present.. Place its index value onto the AESTK and continue execution at next suggested location. Otherwise, continue at lbl.
Test for number. If present, place its value onto the AESTK and continue execution at next suggested location. Otherwise, continue at lbl.

Replace top of stack by variable value if indexes.
list the contents of the program area.
Performs global initialization
Clears program area, emptys GOSUB stack, etc.
Input a line to LBUF.
After editing leading blanks, look for a line number. Report error if invalid; transfer to lbl if not present.

Insert lin
number.
Perform initialization for each stated execution. Empties AEXP stack.

## A STATEMENT EXECUTOR WRITTEN IN IL

This program in IL will execute a TINY BASIC statement. The operators TST, TSTV, TSTN, and PRS all use a cursor to find characteristics of the TINY BASIC line. Other operations (NXT, XPER) move the cursor so it points to another TINY BASIC line.

## THE IL CONTROL SECTION

START: INIT

| co: | NLINE GET LINE |  | ; WRITE CR/LF |
| :---: | :---: | :---: | :---: |
|  |  |  |  |
|  |  |  |  |
|  | TSTL | XEC | ;TEST FOR LINE NUMBER : INSERT IT (MAY BE DELETE) |
|  | JMP | CO |  |
| STMT: | XINIT |  | NITIALIZE F |

STATEMENT EXECUTOR
STMT: TIT SI;'LET ;ISSTATEMENTALET?
;YES': PLACE VAR ADDRESS ON AESTK.
; PLACE EXPR VALUE ON AESTK.
; REPORT ERROR IF er NOT NEXT.
; STORE RESULT:
; AND SEQUENCE TO NEXT.
;GOTO OR GOSUB?
; YES ...TO OR ...SUB.
; GET LABEL.
; ERROR IF cr NOT NEXT.
; SET UP AND JUMP.
; ERROR IF NO MATCH.
; GET DESTINATION.
; ERROR IF cr NOT NEXT.
; SAVE RETURN LINE.
; AND JUMP.
; PRINT.
:TEST FOR QUOTE.
;PRINT STRING.
; IS THERE MORE?
;SPACE TO NEXT ZONE.
; YES, JUMP BACK.
; NO, ERROR IF NO cr:
: GET EXPR VALUE.
; PRINT IT.
; IS THERE MORE?
; IF STATEMENT.
: GET EXPRESSION.
; DETERMINE OPR AND PUT ON STK.
;GET EXPRESSION.
: PERFORM COMPARISON-PERFORMS NEXT IF FALSE.
; GET NEXT STATEMENT.
; INPUT STATEMENT.
;GET VAR ADDRESS.
;MOVE NUMBER FROM TTY TO AESTK.'
; STORE IT.
;IS THERE MORE?
; YES.
;MUST BE cr.
SEQUENCE TO NEXT.
: RETURN STATEMENT.
;MUST BE cr.
:MUST BE cr.
;SEQUENCE TO NEXT STATEMENT.
; LIST COMMAND.
; RUN COMMAND.
;CLEAR COMMAND.
;SYNTAX ERROR.
:TEST FOR UNARY -.
; GET VALUE.
; NEGATE IT.
; LOOK FOR MORE.
:TEST FOR UNARY +
:LEADING TERM.
;SUM TERM.
; ANY MORE?
;DIFFERENCE TERM.
; ANY MORE?


## WE'RE NOT DONE YET

This isn't everything one needs to make a TINY BASIC. It's just a beginning and nothing is tested yet. We'll have more next issue. In the meantime, we welcome your comments, ideas, letters, and corrections.


## 

About your Do-lt-Yourself BASIC: I have about 100 pages of BASIC block diagrams a friend of mine got from Kiewit Computing Center at Dartmouth. If you can use them, I can either get you a copy or send my copy to you for you to copy. The flowcharts are designed for a single address computer with Accumulator and Quotient registers, but with no indexing or indirect addressing, so it might be just what is needed for home-brew types.

Mel Pullen
64 Muswell Rd.
London N10, England educationalists make - underestimating kids. As soon as kid who is manipulating a machine in BASIC divides one by anything and gets zero they'll be so pissed off they'Il lose interest. How do you expect a kid to learn if the machine can't show them what happens. They don't need much accuracy in their figures but at least they should know what happens when they do things differently.

I'd like to help in a sort of ephemeral way; I arrived at computing from teaching - I love teaching kids - the smaller the better and I'll go back to it sometime. But first I want to sort out the computer because it can help. Your ideas for a hard wired BASIC is fantastic but for the problem of $1 / 0$ which might not be standardized. Perhaps dummy status and buffer locations as in DEC 11's would be an idea - anyway it'll happen because it'll be perfect for school and l'd like to get it happening here in England. Perhaps someone could send me details of the Intel chip and the instruction set. Perhaps I could play with a mini floating point package or fixed point.

All these are perhaps because l've only just discovered PCC and there doesn't appear to be much like it in England.

Anyway, l'll try writing (using your tiny integer only) BASIC programs and send them.

Thank god you included GOSUB - kids love building bricks.

- e -

Stephen Ketcham
Academic Computing Center
University of Vermont Burlington, VT. 05401

Dear Mr. Allison;
A friend of mine gave me a copy of PCC magazine; I am now in the process of entering my subscription. In the magazine, I noticed your advertisement for "Building Your Own Basic," by Dennis Allison and others. I am really excited about it. Did you get started yet on the series? If so, I would like to try to obtain all issues. Have you written any books on the subject? Are you discussing the strategies of using a lexical analyzer, translator, or interpreter?

The BASIC language should have the following functions:

Modularity - ability to change building blocks Special features package
Function s package
Floating point package
Device independence - supervisor might be too big Device drivers - ability to add (or delete) driver Error handling

> resident fault dictionary, or
> many, many error numbers, or
combination of above.
I have had experience with many assembly languages, various BASICs and FORTRAN. I think DEC's BASIC PLUS is the best by far. I am anxious to get started on this project.

I hope to hear from you shortly.

## 華

Thank you for your two letters; I didn't expect such fast answers. Dennis wrote me a long and very informative letter in regard to Do It Yourself BASIC.

I found a copy of the book you told me about, Donald Spencer's Game Playing With Computers, at the Ohio State University technical library. It has a lot of good ideas, but I wish Mr. Spencer had more of these games programmed. Perhaps he is coming out with a new textbook - I'm going to write to him. I'm ambitious, but not that ambitious! It takes a lot of time to get the more complicated games perfected.

Where are all the FORTRAN users? I can't believe they don't have any games. You printed Eric Haines' letter in Vol. 3, January 1975, of PCC asking for FORTRAN games. He received only two letters in response. I would like to see game in any language. In the future I would like to see some information on heuristic approaches for games.

I guess I had better get busy and submit a game or two. Also, I have some pictures with overprinting. Does the PCC Bookstore plan to offer source ASCII papertapes for programs and pictures? "Creative Computing" prints pictures (posters) in their magazine. One can order a copy of these posters but one can't get the source. I would like PCC to offer such a service.

Will we ever be able to get some of the old Vol. 1 and 2 issues that are sold out?

Did you ever consider printing your PCC on $81 / 2 \times 11$ paper? It makes filing much easier.

## James E. Rathsack

North Electric Company
P. O. Box 20345

Columbus, Ohio 43220

I would like to add my vote to all the others for the good series of articles on "Building Your Own BASIC", "Tiny BASIC" etc. I have had a Mark-8 system up and running since last fall and have not begun to fully utilize the 8008. Now with Altair and a flurry of BASIC's for 8080 based machines it looks like the only way I'll ever get a BASIC running on my 8008 is by writing it myself with your guidance.

## BY: KEITH BRITTON ROBERT MULLEN

MITS does a lot of expensive advertising and their machine, while it has proved basically sound, has a number of problems requiring fixes - mostly evidence of hurried development. This has led to adverse comment and skepticism regarding their BASIC. MITS has done a lot wrong regarding the 8800 - but they have done a great deal that is right. too.... and received little credit for it. The Altair has proved fundamentally sound. That is to say the faults, though irritating, have cheap fixes that are effective. (People have designed and sold computers that did not work and could not be fixed.) But not only was it what was needed at the time, it was designed to be what was needed for the future. This shows commendable foresight on the part of Ed Roberts \& crew, that is obvious. Less obvious, and something for which they deserve credit which they have not received, is the fact that they were willing to pay a considerable price to provide the hobbyist with things which he did not yet know he needed. The bus, for instance, has 100 lines. This cost money directly, more connections and more expensive sockets, and indirectly in that design freedom encourages competitors. But this philosophy has provided the hobbyist with something open ended. Their BASIC is similar but the features which render it open ended are sufficiently powerful that they amount to an advance in the development of the language.

BASIC (which is a registered trade mark of Dartmouth College) is a comparatively new language. It was developed because no existing language was suited to its application - an easy to learn language for instructional use by beginnerson time shared computers. All languages are compromises and this one gave up a great deal for what it gained. Most professionals sneered at it because of its limitations but through the efforts of a small number of enthusiasts, (among them Bob Albrecht, who is still a member of the BASIC Standards Committee) the language became widely known. The more it was known, the more apparent became its unique merits. These were sufficiently important that BASIC may already have become the most widely known computer language, and is clearly destined to dominate the computer world over the next decade. Naturally. the race is on to incorporate into BASIC the things which were given up in the original compromise, resulting in a series of dialects of the language. Where does Altair BASIC fit in? At the moment, right out there in front of the field, but to explain why, let me back up a bit.

The first machines used machine code - which was fine for machines but miserable for humans. So that humans could generate machine code without losing their minds, a language was produced which allowed us to use phrases or mnemonics to assemble machine code, called, appropriately. ASSEMBLY LANGUAGE. This allowed the programmer to use every feature built into the machine and write programs which were the smallest and/or fastest possible. But assembly language was tedious, since every tittle step had to be de fined, and since each new computer used different machine code, programs assembled for one machine would not run on another and the programmer had to learn a new language for each machine. To get over this new languages were developed which took much more general instructions, written in a form familiar to humans, and from them compiled the machine code necessary to carry them out. Thus were born FORTRAN and ALGOL, both math oriented, which accepted arithmetic/ algebra like instructions.

Note the 'like'. While they were a great advance, both languages were still more adapted to the machine than to the human. Thus, they produced comparatively efficient code, but were difficult to use and very difficult to learn. Programs. however, had become portable to the point that they often needed only minor adaption for use on a different machine. Space requirements had risen sharply. The program to compile code was appreciably larger than that needed to assemble it, though the machine code was not much bigger. COBOL soon fixed that. COBOL was produced in an unsuccessfull attempt to make a language suitable for humans, principally by the expedient of writing everything out in English. This conclusively demonstrated that the information content of 'plus' and ' + ' are the same; that a human is perfectly at ease with either; that such a compiler requires a horrendous amount of memory to produce even inefficient code - but contributed little else and can, for our purposes, be forgotten.

Now these latter languages were all compiler languages. A program was written in source language compiled into machine code and then run. This made it difficult to run more than one program at once, so queues developed and people got bad tempered - particularly when they had an error and had to start again from the back. This wasn't too bad in industry or business where there weren't many programmers per computer and they had sufficient experience not to make many mistakes, but in education where fifty students might be trying to get their program to run.. Educators also met another problem - programming languages were so difficult to learn that only those who were going to specialize In computer science had the time necessary.

Kemeny and Kurtz, brilliantly, solved this problem at Dartmouth College by inventing a new kind of language. Beginners. All Purpose Symbolic Instruction Code was interpretive, not a compiler or an assembler. This meant that the BASIC source language program instead of being converted to machine code, was only slightly altered, being turned instead into a funny stuff called pseudo-code. When the time came to actually run the program, the BASIC interpreter looked at the first pseudo-code statement. It pulled it apart, parsing it, decided what operations were asked for, did them and then looked at the next pseudo-code statement (unless a jump was called for, of course), keeping going until it came to a statement which said 'end'. This had lots of disadvantages. It was deathly slow because the machine had to keep on working out what the statements meant. It was very limited in the things that it would allow a programmer to do and it needed a lot of memory because, when the program was run, both the program and the inter: preter had to be in the machine. Despite this, it lived and prospered. The speed didn't matter too much because, even used inefficiently, computers are fast. The limitations were more of a problem but not critical because the language was used for education, and so did not need the sophistication of a business or scientific language. The memory requirement was no problem at all, because now lots of people could share the same space. Pseudo code was easy to shift into and out of a machine and, so long as the machine remembered which statement it was supposed to tackle next, it could simply continue after a pause. And the pauses came about because the machine would now time share, sharing processor time among many users by working on one program, moving it out of the machine onto a disk memory, moving in another program and working on that, etc., etc. Alternatively, several programs could be in the same machine each in its own memory partition, all sharing the same interpreter and processor.

All this was very nice, but the true reason for the sucess of BASIC was that it suited people. They could quickly and easily learn to use it, and it was powerful enough to do interesting and useful things. But man, unfortunately, is a chronically disatisfied animal. So then he wanted the speed he had given up and a lot of the features of the other languages to be built into BASIC - and there was no way he was going to give up the things he liked.

Well, the only direct way to speed in BASIC is to compile it instead of interpreting it. This can be done but then one loses one of the nicest traits of BASIC, the ease with which one can change the program. This is only possible because of pseudo-code. Since BASIC interprets one statement at a time, it doesn't mind a bit if you slip in an extra one or delete a few, but the slightest change in a compiled program requires that the whole thing be re-compiled. But the MITS software crew have an indirect way to blinding speed (or, more accurately, all the speed of which the processor is capable), they allow BASIC to jump back and forth to machine code subroutines. These can be written as such, assembled or compiled. So the Altair user can simply keep a library of such routines and use them as needed.

This has been done before, as has another of Altair BASIC's major features - the ability to talk out to user-supplied $1 / 0$ devices, but not as simply and not as cheaply. A BASIC with the power of this one is a major acheivement, there's a monstrous amount of work in it, and they have sold in the low kilobuck range for each copy. Some of the other features, particularly string and numeric matrix handling are certainly unusual and may actually be unique, but the user may notice them less than the general attention to detail evident in the programmers' approach. They, Bill Gates-Paul Allen-Monte Davidoff, must have spent literally hundreds of man-hours mulling over comparatively minor details - but details which sum to a degree of convenience for the user which will endear the dialect to those who are lucky enough to own an Altair. And, hopefully, the fact that MITS have done their BASIC so well will spur their competitors to improve the language yet more. Or, if they can't take a step foward, copy MITS's innovations and at least not take a step back.

All major programs must be expected to have bugs and the extent of the testing required to identify the more subtle ones is so great that it is more reasonable to test for the obvious ones andthen wait for the users to find the others, updating software with fixes at intervals. Altair BASIC version 2.0 had a serious problem in that a jump out of a FOR., NEXT loop left garbage on the stack. Do this too often and the stack would grow relentlessly down from high memory until it ate the program. This has been fixed in version 3.0, according to Paul Allen, but we have not yet received a copy to pull apart. We also hear that there have been a number of other improvements and fixes. Presumably, someone punished it a lot harder than we did, since we found little else to criticize, but we did find one problem and it has unfortunately been left in version 3.0. If you print a value less than 0.1 , it appears in scientific notation. So, if you are printing out dollars and cents, and you have less than 104, you suddenly see some thing like $-3.00000 \mathrm{E}-02$.

MITS has frozen development at the 3.0 version, except for bug fixing, and are close to releasing the first versions of their extended BASIC. And THAT has things that will make even more difference to the way BASIC will grow over the next few years. The worst single item BASIC gave up, relative to the other languages, was flexibility in precision, adopting 6 digit floating point for all variables. With floating point even incrementing by one is slow, and business use requires more precision. Altair Extended Basic is to have integer, 6 digit and extended precision variables . . . plus a lot more.

There have been two major streams of development in BASIC, which can be broadly described as HP BASIC and DEC BASIC, the most obvious difference being multiple statements per line in the latter. MITS has followed the DEC stream and is very similar to PDP11 BASIC. Few will find much difficulty in translating listings to the new code, however, and almost all programs will become smaller.

A lot of space will be saved by the use of multiple statements per line, but the separator between statements is a colon instead of the backslash used by DEC. Just perversity? Not at all-you don't have to shift for a colon. This rendering of JOTTO, courtesy of Kent Cross, shows how much compression can be acheived. He is now scrunching Star Trek, but don't hold your breath - it's a lot of work!

This saves space principally by reducing the number of line numbers which have to be stored. Another simple improvement saves a line and greatly improved clarity can be seen in lines 40,80 and 270. Mostly, when one asks for input, one also prints a line to say what input is wanted. This seems to have escaped most people but MITS noticed and allowed a print string after INPUT.

## NEXT

The MITS people also noticed that space was wasted in FOR...NEXT loops. If you don't have a nested loop, you don't need to specify the variable and if you do, often you could avoid a second NEXT if you could specify more than one. It's easier on the programmer too, fewer keystrokes. (See lines 50\&150.)

## STRINGS

Programmers with small machines become space conscious. And to help them is a function, FRE which returns the amount of space left, or with a string argument the free string space. The latter is in a state of constant flux in the Altair. Unlike most BASICS, with this only the total string space is defined. Any string can be up to 255 characters long, but only the space in the strings which are filled with characters count. And that, of course changes dynamically.

This isn't the only unusual feature of Altair strings. For one real mind blower, they can be defined as subscripted variables. Of course a $10 \times 10$ matrix of 255 character strings would need memory........

Another nice string trick - if you need strings in DATA statements, so long as you don't need leading blanks or commas, you can type them without quotes, see lines $240 \% 250$. Think how many keystrokes that can save. One complaint though, the DEC BASIC convention is followed regarding substrings. LEFT\$ \& RIGHT\$ can be convenient, but when one wants towork in the middle, MID\$ is clumsy compared to HP BASIC's A\$ $(5,7)$.

Not a complaint but PRINT USING would have been nice to have. Still, with VAL\$ and STR\$ one can convert from string to numeric and back, which gives a lot of control, and CHR\$ allows one to specify an ASCII numbered character.

## MATs

Altair BASIC has matrices, naturally, but theirs can have up to 255 dimensions.... Use three for 3D, and for time travel games...... One warning though. The

MATs start from D not from 1. Here again is a great division (The BASIC standards committee voted 14 for 0 , 14 for 1) and for the author's money MITS did this right too. HP matrices will work on the Altair, with some waste space but the reverse won't be true. Matrices, like the strings, are dynamically dimensioned, so you can work with one larger than the memory of the machine - so long as you don't ever fill it at one timel

## END

As a space saver, the END statement can be left out, but this produces curious consequences. Since the machine has not been told to stop, it doesn't, and the effect is as though RUN had been typed instead.

[^0]0
K. Britton and B. Mullen at work


## 0410 FOR THE AL FAIR BY KENT CROSS

7 HKINT:FKINI
RESTORE:T1=も
$\Gamma 1=11+1: I F \quad \mid 1=15 \quad$ IHEN 234
$T=\emptyset: \angle=1$

FOR $x=1$ TO 5

NEXT END

## VARIABLES

Partly as a space saver, partly as a people saver, the RUN command initialises all variables to zero, so the first time a variable is used it will be initialized. One does wonder though, whether this will train people to bad habits. There is one useful consequence though. Few will feel the need to set a mess of variables to zero and be caught by trying $\mathrm{A}=\mathrm{B}=\mathrm{C}=\mathrm{D}=0$ or similar. Doesn't work because one of the $=$ behaves as an assignment operator, the other becomes relational.

Yet another good idea, though unfortunately not fully implemented, is the use of datanames. In most high level languages, and assemblers for that matter, one can call a spade a spade. If you are using a variable to store a total, you can call it TOTAL. This helps the programmer to remember what goes where. In this BASIC one can call a variable FRAN if it helps but one must proceed with considerable caution thereafter. Only the first two letters are checked so any later dataname using the same two will be confused. And you had best not use TOTAL at all. It contains the reserved word TO and any embedded reserved word will cause a syntax error. They are not always easy to spot and MITS BASIC only checks for this at run time, a serious weakness in any BASIC, and you could find that you have to change every occurrence of a bad dataname. So if you are starting on a long program, try the proposed datanames first!

10 JOHN = 15: FRUMP = 13 : GEKTRUDE $=14$ 2Ø FRINT JOHN;FRUMP; GERTKUDE
30 PRINT JOE;FKANNY; GENE
44 END
OK
RUN
15
15
13

OK


40 LNPUI "DO YOU WANT INSTRUCTIONS (Y/N)";Aक:IF AE="N" THEN 47
41 HKINI "IHIS IS THE GAME OF JOTTO. I AM IHINKING OF A"
42 HRINI "FIVE LETIER WORD. YOU TKY AND GUESS THE WOKD BY"
43 HKINT "HKINIING S-LETIEK WORDS AND I'LL IELL YOU HOK"
44 PRINT "MANY LEIIERS IN YOUK WORD ARE IN IHE SAME ROSITION"
5 HRINI "AS IN MY WORD. YOU GET 15 GUESSES.":rRINT
FOR $x=1$ TO INT (KND ( $(\checkmark) * 2 \Delta)+1$ :KEAD CS:NEXI

INPUT "YOUK WORD";AS:PRINI: $1 F$ LEN $(A S)=5$ THEN 110
IF LEN $(A S)>5$ THEN PKINT "ONLY S LETTEKS LONG, RLEASE.":GUTO BO
5 PNINT "NOT":LEN(AS):"LETTERS LONG, FIVE LETTERS LONG!":GOTO 80
RRINT "YOU KNOW DAMN WELL THAT WOKD DOESN'T EXIST!!":GO10 75
IF $M \operatorname{IDS}(A S, X, 1)=M \operatorname{IDS}(A S, X+1,1)$ THEN $\angle=\angle+1: 1 F \angle=3$ THEN 108 IF M1OS $(A S, X, 1)=\operatorname{MIDS}(C s, X, 1)$ THEN $T=T+1:$ IF $T=5$ THEN 190

IF I=1 THEN PRINT"YOU MAVE 1 LETTER IN THE RIGHI POSIIION.":GOTO 7 IV HKINT "YOU HAVE"; I;"LETIERS IN THE KIGHT ROSITION.": GOIO 7 7 PKINT "YOU GUESSED IT!! AND IN ONLY";11:"TRIES!":GOTO 274 KKINI "SOKRY, IJO MANY GUESSES. THE WOKD WAS ";CS
 DATA DIGIT, ARAKT, DKAMA, START, OPERA, RESET, BASIC,MANCH, COUNT, FAUST INRUT "DO YOU WANI 10 TKY AGAIN (Y/N)"BAS:IF AS="Y" IHEN 47

Since we haven＇t yet gotten version 3．0，we have not bottered to punish the random number generator and can＇t give an opinion on its random－ ness．But we can，certainly，offer an opinion of the control over it given to the user．This control will make RND useable for games，simulations and for encryption．One can give RND（ ）a positive，negative or zero arguement，The effects：

```
10 FOR X=1 TO 3
20 PKINT RND(O)
30 NEXT
40 END
RUN
    .811635
    .811635
    .811635
20 PKINT RND(-.3)
RUN
    .400015
    .400015
    .400015
20 PRINT RND(-.4)
RUN
    Different seed, different value.
    .792188
    .792188
    .792188
2\emptyset PRINT RND(1)
RUN
```

.946891
1．7И491E－nこ
3．＇2り927E－ウえ
positive number
This give a new random number each time －and the value of the arguement does not matter．This is the one to use for games．
$5 X=R N D(-.3)$
RUN
.996024
2．1321E－02
.919327

For simulation you will need random numbers－But often the same sequence every time you run．So add something like this to set the random number generator to the same value each time you run．

```
NEGATIVE NUMBER
This changes the number in the random number generator，using the value as a seed．Same seed，same value．
Different seed，different value．
． 792188
－792188
－792188
Ø PRINT RND（1）
```


## AND NOW ENCRYPTION

Suppose you want to protect a file or make it difficult for someone to read a tape which you are going to send to a friend who has an Altair． Choose a routine like this．（Takes a character
A\＄and converts－set up 1000－1060 then
gosub to 1100 for each character．）

The two most unusual features of Altair BASIC are the ability to use machine code subroutines，and the ability to talk to data ports．The latter uses the verb OUT， which，with a decimal argument between 0 and 255 ， imposes
which，with two arguments between 0 and 255 decimal imposes an 8 bit pattern on a selected port．The function INP retrieves the status of any port in similar fashion．

This may seem trivial，but any machine that can be controlled by switches，or by use of a $D$ to $A$ converter， can now be run from an Altair using a BASIC program． With 256 ports and 8 bits per port，this represents some 2，000 switches．Enough for an organ？Washing mach－ ine？How about the cat door，using a pressure trans－ ducer，A to D converter and INP to weigh the cat and see if it is really yours？

[^1]Mostly the big people in the semi－conductor world just don＇t want to know an individual with problems or needs，though there are good guys who are likely to help with technical data．This is one of the reasons why kits for the hobbyist have emanated from small outfits．Some of the prices have been steep but the difference between the big company prices and the little guy＇s have really shown where to go for value．No big company has shown any interest in selling in a market where a microcomputer starter kit can be had for around $\$ 250$ ．

## Until now．

One of the biggest，Motorola，is offering its development package for nearly half that price！For $\$ 149$ you can buy the entire thing．Match this for an offer you little guys．．．．

We have been using one of Bob Mullen＇s prototype boards at PCC．（He will be selling a relay／opto－isolator kit very shortly．）The fascination of game playing is nothing to the thrill of making a machine do something physical！
 time

## WAIT

How do you like the thought of leaving an Altair at home watching for smoke or intruders？There is another command，WAIT，which sets the machine to watch for a bit to change on a selected input port to help your programming．


## poke



When you require a program which for some reason (speed, or special bit handling) must be written in assembly language, it becomes a task to supply the I/O and other features which are taken for granted in the high level languages. It would be great if there were a way to use the nice features of a high level language and the utility of assembly language.
Altair 8K BASIC allows the user to link his own functions into BASIC. The main steps in using an assembly language function with BASIC are: saving space in memory, putting the program in memory, and linking the program with BASIC.
When Altair 8K BASIC is loaded, it asks the question "MEMORY SIZE?". If you respond with a numeric value it is used as the top of memory for BASIC. In the example below, 7935 ( 17377 octal) is used. This leaves 7936 and above available for the user.
To store the program in memory, it could be loaded before or after BASIC is loaded, using a loader or using BASIC as the loader. Altair 8K BASIC has a function "POKE" which provides the ability to modify memory. Again, in the example, the user function is placed in memory in statements 210 and 230. In the same manner, data is furnished in statements 270 thru 290.
Linkage to the program is formed by putting the starting address in 117 and 120 octal; this is done in statement 160.
The example program handles octal numbers by carrying them in decimal and converting them before use, this is done by the subroutine in statements 330 thru 350.


OK
101 REM PROGRAM TO DEMONSTRATE LINKAGE TO ANOTHER PROGRAM USING BASIC
110 KEM THE PROGRAM IS A VERY SLIGHTLY MODIFIED VERSION OF SIEVEN
120 REM DOMPIER'S MUSIC ROUTINE. SEE PCC VOL. 3 NO. 5 PAGE 9.
140 REM PUT THE PROGRAM START ING ADDRESS IN USRLOC. SEE APPENDIX I
150 REM ALTAIR BASIC REFERENCE MANUAL.
$168 \mathrm{~A}=7936: \mathrm{I}=\mathrm{INT}(\mathrm{A} / 256): \mathrm{J}=\mathrm{A}-\mathrm{I} * 256$ :POKE 79, J:POKE 80, I
170 REM 7936 DECIMAL $=17400$ OCTAL. 79 DECIMAL $=117$ OCTAL
190 REM GET PKOGKAM FROM DATA, CONVEKT TO TKUE OCTAL, AND
200 KEM STORE IN MEMOKY STARTING AT 174 O O OCTAL.
210 FOR $N=0$ TO 25:READ K:GOSUB 330
$230 \mathrm{~A}=7936+\mathrm{N}: \mathrm{POKE} A$, D:NEXT $N$
240 KEM INPUT NOTES AND/OK TERMINATE SIGNAL FROM KEYBOARD, CONVERT
250 REM INTO TRUE OCTAL AND STOKE IN MEMOKY JUST ABOVE THE PROGRAM.
$260 \quad A=7968$
270 INHUI K:GOSUB 330
280 IF $D<0$ THEN 270
285 POKE $A, D: A=A+1$ : IF $D<>255$ THEN 270
290 GOTO 450
3Ø0 REM SUBROUTINE TO CONVERT WHAT LOOKS LIKE AN OCTAL NUMBEK
310 KEM INTO THE CORRECT VALUE. ALSO CHECKS FOR NUMBERS WHICH
320 REM CANNOT BE OCTAL, WHICH ARE GREATER THAN 377, OR ARE MINUS.
$330 X=I N T(K / 1 \emptyset \emptyset): Y=I N T((K-(X * 1 \emptyset \emptyset)) / 10): \angle=K-(X * 1 \emptyset \emptyset)-(Y * 1 \emptyset)$
340 IF $K>=\emptyset$ AND $X<=3$ AND $Y<=7$ AND $Z<=7$ THEN $D=64 * X+8 * Y+\angle: R E T U R N$
$35 \% \mathrm{D}=-1$ : PRINT "EKROK":RETUKN
370 KEM PKOGRAM STORED IN FAKE OCTAL IN A DATA STATMENT
38 DATA $41,40,37,176,376,377,310,26,40,5,302,16,37,106,15,302,11$
390 DATA $37,25,302,11,37,54,303,3,37$
400 REM ENTER HERE TO CHANGE IEMPO
410 INPUT "TEMPO" K: GOSUB 336
420 IF $D<\theta$ THEN 410
430 POKE 7944, D
440 REM KUN THE HROGKAM BY CALLING USER FUNCTION
450 W=USK(1)
460 STOP
999 END


## Assembly Language Programming

## BY: BERNARD GREENING

PCC has an Intelec 8/MOD 80. This system came with a monitor, a text-editor, and an assembler. It is Intel's system based on an 8080 chip for programming in assembly languague. NUMBERS is a BASIC program which appeared in the first issue of PCC (Vol1, no.1). By writing the same program in assembly language, the two languages can be compared.

The first step is to see what the BASIC program does, Then write down assembly language statements that do the same thing. This may take several hours or days depending on how complex the task is. This is called the source, which is then typed on the teletype using the text-editor. The text editor will then make a "computer readable" source.
The next step is to assemble the program. This is a process of converting the assembly language instructions into machine language instructions. Assembly language , has labels which identify memory locations You can branch to a labeled instruction like GOTO in Basic. You also have to set up labeled memory locations in which to store data. This is done auto matically in BASIC. If you misspell anything, or if you do not follow the proper assembly language format, the assembler will indicate an error. You then have to fix your source with the text-editor and assemble it again.

Soon all your errors will be corrected. The assembler will make an object program. This consists of the machine language code which the computer can run.

The Intelec 8 monitor can load the object code and run it. If the program does not work, you can set traps called breakpoints and get a better idea of what is wrong by looking at the contents of the registers and critical memory locations.

As an example of how NUMBERS was converted from BASIC to assembly language, two lines of the program get a guess, compare it with a previously generated randon number and branch if they are equal.

430 INPUI G
440) IF $G=X$ THEN 500

## MONITOR:

On the Intelec $8 /$ MOD 80 , the monitor is put on 8PROM's. It loads object programs, examines the contents of the registers and memory, puts breakpoints in programs, and branches to start programs

ASSEMBLER:
The assembler is a program that reads a source program and creates a paper tape of the object program. It also creates a listing showing each memory location and the machine instructions that occupy them. It shows errors in your program too.

## TEXT-EDITOR:

The text-editor is a program that creates and makes corrections in source programs or other text. It can insert and delete lines and make changes in lines

The assembler converts the four assembly statements to the following machine language instructions.

$$
\begin{array}{lll}
315 & 270 & 001 \\
041 & 152 & 000 \\
312 & 117 & 000
\end{array}
$$

(octal)

Each op-code like CALL, LXI, CMP, and JZ is converted to its machine language code. Labels like DNUM, CNUM, and WON are converted to the addresses that they correspond to. If changes are made in the program, the assembler will recompute the addresses as necessary. When the machine instructions are loaded in the computer, it can read through its memory, interpret the instructions, and perform the functions in NUMBERS. It generates random numbers, does input and output to the teletype, compares guesses and random numbers, and tells the user if the guess is correct.

The same thing in assembly language takes four statements.

CALL DNUM
; (A) GETS A NUMBER FROM THE TELETYPE
3
3 DOES THE RANDOM NUMBER AND THE GUESS AGREE?
3
NCHK: LXI H,CNUM $3(H, L)$ POINTS 10 THE RANDOM NUMBER CMiP $N_{1}$ COMPARE (A) AND ( $\left.(H, L)\right)$ $\begin{array}{lll}\text { CMI } & N_{1} & \text { WON COMPARE (A) AND } \\ J L & \text { EEUUAL, A WINNER }\end{array}$

The advantage of assembly language is that it makes an object program which can be loaded and run in an ALTAIR with 1 K of memory. NUMBERS takes less than 800 words in assembly language. BASIC needs 4096 or 4 K words.

One way to load the object program into memory is to do it from the console keys. This requires setting about $800^{\circ} 8$ or 6400 console keys correctly. A better method is to load a program of about 40 words which can read in NUMBERS or any other object program from paper tape and run it.

Currently, PCC has no fast, accurate method of making paper tapes. Are there enough people who are interested in loading programs from paper tape to justify our working on this? You have to have an ASR-33 or other method of reading paper tape.

We know of no "approved standard" for binary paper tape with check digits that is short and can be read into memory with a short, simple loader program. We have made one up, but would like to see if there are others. Does anyone out there know of one?

Another method of getting an object is to use a cross assembler. Cross assemblers are available on some large computers. A person with access to one can use the facilities of the larger computer to create object programs for his home computer.


## **************

More about Assembly Language and Numbers next time. Send some ideas!



## on the Altair

## by: Adolph P. Stumpf

In October, 1970, SCIENTIFIC AMERICAN first published details about John Conway's game of LIFE. published details about John Conway's game of LIFE
People all over the world have been discovering new and People all over the world have been discovering new and
interesting LIFE patterns. They've been given names like interesting LIFE patterns They've been given names like
GLIDR, SPACESHIT, BARBER POLE, CHESHIRE CAT, GLIDER, SPACESHIP, BARBER POLE, CHE
BLINKER, HARVESTER, FUSE, and BOX.

In PCC News \& New Stuff you asked for programs. Well, here is my version of John Conway's game Life.
The game was described in a number of articles by
Martin Gardner in the Scientific American about
three years ago. Haven't come across it anywhere
else.
The program was written to run on an Altair 8800 with 1 K of memory and a SWTP CT-1024 terminal. With minor changes it should work with other systems using an 8080 or 8008 CPU .
At present the program is somewhat limited primarily because of small display, 16 lines of 32 characters, and lack of memory system. Computation is relatively fast, less than one half second. In comparison the display update seems agonizingly slow.
However, it's still fun to watch the display of fantastic combinations which this simple game can generate.
I'm working on some improvements which will include

1. Input of initial population and complete program control from the keyboard.
2. A 64 by 64 (or larger) character display with page selection from the keyboard.
I'll send the new version (hopefully written in BASIC) when it's finished and when I have the hardware to run it on.

The program plays John Conway's game Life. Display is 30 by 15 characters on TV terminal.
Data (a generation of the initial configuration) is stored in locations 002000 through 003 377. The first line on the display is 002000 through 002337. the second 002340 through 002 377, etc. and the last line is 003340 through 003377.

The present generation is represented in the most significant bit; a ONE for an occupied cell and a ZERO for an unoccupied cell. The next generation is computed and stored in the next most sig nificant bit using the same format.

## TO RUN PROGRAM

1. Run program starting at location 000315 to clear memory.
2. When HLTA light comes on, actuate STOP and RESET switches.
3. Load initial population with front panel toggle switches.

Format is (octal) 3 XX , where XX are "don't care" bits.
4. Initiate HOME UP and ERASE FRAME functions on terminal. Turn off cursor for a more pleasing display.
5. Run program starting at location 000000 .
6. To terminate, actuate STOP switch.


OUTPUT RCUTINE, DISFLAYS 512 CHAKACTERS ON TV FRCM MEMORY LCCATICNS 002000 THRCUGH 003377

```
LXI D INITIALIZE START ADDR
LDAX D GET DATA WORD
RLC
STAX D STORE UPLATED DATA WCRD FUK NEXT FASS
RLC DATA TC CAFRY
JC CARRY = CELL OCCUPIED
D
MVI A CELL UNOCCUPIED, LOAD SPACE
JMP
E
MVI A CELL CCCUFIED, LCAD *
RST OUTPUT C.IARACTER
INX D INCR DATA WORD ADDR
DAD D CHECK FCR END
JNC CARRY = DONE
F
```


## CONTRCL RCURINE

|  | 030 | 006 | MVI B |
| :--- | :--- | :--- | :--- |
| C | 031 | 006 |  |
| B | 032 | 016 | MVI C |
| 033 | 377 |  |  |
| 034 | 015 | DCF C |  |
| 035 | 302 | JNZ |  |
| 036 | 034 | B |  |
| 037 | 000 |  |  |
| 040 | 005 | DCR B |  |
|  | 041 | 302 | JN2 |
|  | 042 | 032 | C |
| 043 | 000 |  |  |
|  | 044 | 311 | RET |
|  | (000 045 through 000 077 not used) |  |  |



| 000154 | 041 | LXI H |  |
| :---: | :---: | :---: | :---: |
| 155 | 374 |  |  |
| 156 | 001 |  |  |
| 157 | 136 | MCV E, N . | GET CULRENT DATA WLCD ADDR |
| 160 | 043 | INX H |  |
| 161 | 126 | MCV D, N. |  |
| 162 | 033 | DCX D | DECREMUNT |
| 163 | 162 | MCV M, D |  |
| 164 | 053 | DCX H |  |
| 165 | 163 | MOV M, E | STCRE UPDATED DATA WURD ADDR |
| 166 | 043 | INX H |  |
| 167 | 043 | INX F |  |
| 170 | 065 | DCR M | LECR CHARACTAK CCUNTER |
| 171 | 302 | JNZ | COUNTER $=0$, DONE WITH IINE |
| 172 | 146 | H |  |
| 173 | 000 |  |  |
| 174 | 033 | DCX D | DECR DATA WCRD ADDR TWICE FCR NEW LINS |
| 175 | 033 | DCX D |  |
| 176 | 066 | NVI M | reset Chait ccunter for new IINe |
| 177 | 036 |  |  |
| 200 | 053 | DCX H |  |
| 201 | 162 | MOV M, D | STCRE DATA WORD ADDR FCK NEW LINE |
| 202 | 053 | DCX H |  |
| 203 | 163 | NOOV M, E |  |
| 204 | 053 | DCX H |  |
| 205 | 065 | DCR M | DECR LINE CCUNTER |
| 206 | 302 | JNZ | LINE CCUNTEK $=0,16$ LINES DCNE |
| 207 | 146 | H |  |
| 210 | 000 |  |  |
| 211 | 303 | JMP | GC BACK AND DC ANCTHER FRAME |
| 212 | 100 | A |  |
| 213 | 000 |  |  |


| 000 | 232 | 021 |
| :--- | :--- | :--- |
| 233 | 076 | LXI D |
| 234 | 000 |  |
| 235 | 031 | DAD D |
| 236 | 317 | RST |
| 237 | 043 | INX H |
| 240 | 317 | KST |
| 241 | 043 | INX H |
| 242 | 317 | KST |
| 243 | 052 | LHLD |

CCUNT NETGHBCK AT BCTTCM LEFT
INX H
KST
INX H
LHLD
CCUNT NEIGHDCE AT BCTTCM
CCUNT NEIGHDCR AT BCTTCK RIGHT

NEXT GENERATION RCUTINE, DETERNINES BIRTHS, SURVIVALS, AND DEATHS


I
DCX H
RST
INX H
INx H RST
DCX H
NOV A,N GET DATA WCRD
RAL DATA TC CARRY
JC CARRY = CELL OCCUFIED
I

NCV A,B NUMBER CF NEIGHBCRS
CPI ISIT 3 ?
JL YES, EIRTH
J
RRA A RETURA, DEATH
MCV A,B NUMEER CF NEIGHBCRS
CFI ISIT 2?
JZ YES, SUKVIVAL
K
CFI ISIT 3 ?
JL YES, SURVIVAL
K

MVI A
RET FETURN, DEATH
MVI A

ReT
RETURN, SURVIVAI
MVI A

KET KETURN, BIRTH
CLEAR MEMORY RCUTINE, WRITES ZERUS IN LOCATICNS 002000 THECUGH $00337 ?$
$\vdots .$.
$\vdots$
$\vdots \ldots$
$\vdots . . .$.
$\vdots$

The CHESHIRE CAT
slowly disappears,

HERE'S HOW TO PLAY LIFE
Births and deaths happen at the same time.
Mark the cells that are to die. Next, mark
those that are born - you can count those
that are dying. Now you can remove the
dead ones and put a mark in the new cells.
SURVIVALS. Each alive cell with 2 or 3
live neighbors will survive
another generation.

|  | 000 | 315 | 021 |
| :--- | :--- | :--- | :--- |
|  |  | LXI D |  |
|  | 316 | 000 |  |
|  | 317 | 002 |  |
|  | 320 | 041 | LXI H |
|  | 321 | 000 |  |
|  | 322 | 374 |  |
|  | 323 | 257 | XRA A |
|  | 324 | 022 | STAX D |
|  | 325 | 023 | INX D |
|  | 326 | 031 | DAD D |
|  | 327 | 322 | JNC |
|  | 330 | 320 | L |
|  | 331 | 000 |  |
|  | 332 | 166 | HLT |

BIRTHS.
If it has less than 2 neighbors, it dies from isolation.

Each empty cell with 3 neighbors, no more and no less, is a birth cell. A mark will appear in it in the next generation.

REMEMBER : Births and Deaths take place simultaneously. So don't count a new cell for ovorpopulation.

## The Dragon Welcomes

## the Chimera

Here is the information which you asked for.
This year is the first year that we had timesharing, and languages other than FORTRAN' as well as batch facilities because of the installation of a Burroughs B5500 just before the classes began in September. I was first exposed to timesharing at the Second Conference on Computers in the Undergraduate Curriculum at Dartmouth College. Since then, the Dartmouth style has been one of my personal paradigms. (The PCC is another, Project Solo a third ....) Borrowing from the Dartmouth model, one faculty member offered to underwrite a prize in computing, and money obtained by selling unsolicited desk copies to a wandering used book buyer paid for two more prizes. With the special subscription rates to the PCC, we knew what we could give. The three prizes of a year's subscription to the PCC were for
(1) excellence in programming (this was won by Paul Kinney, a first year student),
(2) greatest contribution to a non-computer-science class (this was won by Craig Bock, a third year student, with his program for a course in cryptology. The program simulated the "Purple Machine" used by the Japanese in World War II), and
(3) enthusiastic support of academic computing (this was won by Debbie Campbell, a first year student).
No one knew about the contest last year, and these prizes will be formally announced on 4 Deptember when next year's contest, with a few rules this time, will begin.

If the PCC is symbolized by the Dragon, perhaps our symbol should be the Chimera (look that up in your Funk and Wagnell's.).


## Resource Dine

PCC'ers; this is a long blurb on a San Francisco computer organization, a short blurb on Resource One is reportedty avalitable in People's Yellow Pages. I was given approval by Resource One for PCC to publish the information below, to aid the paper, its readers, and to publicize Resource One.

Resource One is a non-profit community computer center providing the sophisticated technology of a computer to other non-profit organizations working for social change. Our current projects include the following: Computerized bookkeeping and mailing list services, information retrieval, forms processing, collecting a computerized 'library' of information on the City (e.g., property ownership, land use, census information, etc.), publishing a directory of social services available in San Francisco, and general programming and consulting to meet specific needs of user organizations.

Counting house, Resource One's bookkeeping service, is a system designed to help non-profit organizations keep their finances straight. The raw data is entered into the computer directly from the checkbooks, deposit stubs, and other original documents via a typewriter like terminal. Income statements and balance sheets can then be produced on demand. Also, lists of transactions can be provided by account, date, fund, or any combination. Currently the system handles cash disbursements, cash receipts, accounts receivable, accounts payable, payroll and general journals. The system can be used without prior knowledge of computers or bookkeeping.

NIMS, our mailing list system, is another easy-touse system which allows someone from the user organization to type into the computer a mailing list which can then be printed out on gummed or cheshire labels. Its main advantages are that various subsets of the list can be printed, allowing an organization to target mailings to a specific audience, and that corrections to the list are easy to make. The mailing list can be sorted by either name or zip code.

ROGIRS, the Resource One Generalized Information Retrieval System, is designed to manage large amount of information through cross-indexing by 'keywords'. The text of the item is entered with keywords which describe it. The item can be pulled out of the computer by simply saying 'find' and a list of keywords, One could say 'find this or that' and II of the items with the keyword 'this' or the keyword 'that' would be found. This system is currently being used by Amnesty International to maintain a list of contacts and by the North American Congress on Latin America to research CIA involvement in Latin America.

We have done processing of client intake forms for the San Francisco neighborhood legal assistance. This involved typing into the computer the responses directly from the forms. The results were reports containing cross-tabulations of legal problems, income, ethnic grouping, etc. This program has been modified to handle other forms and could be expanded to produce correlations or other statistical reports.

The Urban Data Base Project, a collection of computerized information about the City, can be a useful tool for learning about the community and as an aid in organizing to respond to community needs, The Urban Data Base has been used by La Raza En Accion Local and the Mission Planning Council to successfully down-zone properties bordering Mission Street in San Francisco from commercial to its residential use. It has also been used by students from University California at Berkeley who are wroking with the Central City Coalition to produce an alternative 'south of Market plan' and by the Tenant's Action Group to organize housing in San Francisco.

This is a brief summay of what Resource One is doing and of the potential for what can be done. If you would like more detailed information about these projects or if we can be of assistance to your organization please do not hesitate to call. We are in Project One at 1380 Howard St., Our phone number is 'UNIT ONE' (864-8663).

## Dean Kahn

224 Robin Way
Menlo Park, Ca. 94025

## Intergalactic Question Man

Does Dartmouth give away/sell/trade/distribute the programs that live there? What about Lawrence Hall of Science? Anybody got any programs that they want to give away or trade? Especially games ...

What legalities govern the using/trading/printing/ copying of programs not your own? I am interested in setting up an exchange for BASIC programs, especially games. Would also like to see a book in the style of "The Whole Earth Catalog" filled with games. It would hopefully retail for \$3 or \$4. Something like "101 Basic Games" but with more games per dollar, and hopefully better games. I hope to eventually get your games book, but my supply of money is currently depleted.

Anybody at Datapoint want to befriend me?? I know BASIC some assembler, and have even read a booktet on Databus.

I am also looking for Grady Hicks and Jim Korp. They wrote a really neat Star Trek game that lives on the University of Texas system under the name of SPCWAR. The listing I have was listed on July 1973. Can I legally trade this program without their permission? There is also an advanced modification of this program on the United Computer Services system, Though I have not played it, I have seen a copy of the instructions, and it looks neat. Can anybody find me the address of UCS so I can try to beg a listing off them?

I am also working on version II of Spacer. I finally got the first version running, but it was too boring and serious, so I chunked it. I am in the designing and planning stage on Spacer II. It is not a standard Star Trek program, it has stuff like enemy warships that move between Quadrants, Death Rays time travel, teleporters, starmakers, smoke screens, lifeboats, a full range of experimental weapons, starships (enemy) that attack your starbases, starbases that move, blasters that allow you to shoot at all the enemy ships in your Quadrant or at only one of them, missile spreads, gravitational analomies, black holes, supernovas fand a neat way of explaining how a starship that is parsecs away can be destroyed by them), hyperspace, varying degrees of complexity selected by you, and a plot. I have tried to present a unified concept of the universe and man's place in it. Basically it is a rescue mission through enemy held territory. I have an elementary look-ahead routine which helps the enemy warships to prepare ambushes. There are also enemy scout ships in each Quadrant whose mission it is to destroy your starbases and to hinder your movement. I have about 20 commands planned so far. I would like everybodys ideas on what to put in it. What would you like it to do? Please tell me . . . it will be written in BASIC, probably HP, but I am trying to write it in as simple a BASIC as possible, and am keeping the use of strings to a minimum. It should take some time to finish, but should be finished around November. Hick's and Korp's Star Trek is written in Dartmouth BASIC I would also like to write a graphics Spacewar on a graphics terminal that lets you draw lines. What do you think that the best system would be, letting the SR scan stay on the screen and printing the other stuff off to the side, displaying the SR scan only when it is asked for, or something else??? I would also like to write a simplified version on a CRT terminal which allows the cursor to be repositioned. I realize that most of the freakies (including myself) will be playing it on the lowly (and slowly) TTY, so I will write that version first. Can anybody tell me more about look ahead functions? Is there any Spacewar game on one of the larger systems where you play against another person (or dragon or other assorted creatures, whatever the case may be)?? Expecially one that lets you each command a fleet of ships? What are some good sources on designing and making conflict and war games? Are there any non-conflict games around? How do I join SHAFT? Have you gotten HP to loan you one of their new graphics terminals to you yet? Their publicity blurb looked good, but I would like to hear a user say something about it. Graphic CRT terminals are the wave of the future, hopefully the near future. TTYs are too slow, noisy, and eat trees. You still need some kind of hard copy device, but for running a program a CRT is hard to beat. Hey Bob, the world wants to hear more about how PCC is organized and run.

May you never run into a star,
John A. McClenny
5819 Brenda
San Antonio, TX 78240


## Star Trader

Many moons ago you sent me a listing of "Star Trader" which I promised to translate into FORTRAN IV. Well, during school I was pretty busy and was only able to make the initialization program, so during the summer, I worked on it continuously, and, at last, it's donel!! (I think) Anyway, it is enclosed.

## Notes -

(1) I have added two functions besides 'save' They are 'MAP' which gives a MAP and 'HOLD' which give what is in your ship's hold.
(2) I also added to the format statement in banking to tell you how much you have on the ship.
(3) Some warnings (as in banking, selling etc.) have been removed for lack of space.
If I had the space, money would be in real numbers, as would produced goods, etc.

## Eric Haines

212 N. Riding Drive
Moorestown, NJ 08057


## Wanted

We would like to buy equipment that will permit operation of Wang No. 732A, flat-bed, $31^{\prime \prime} \times 41^{\prime \prime}$ plotter, from Altair 8800.

> Liss Engineering
> 1366 H Logan Avenue
> Costa Mesa, Ca. 92626

## Aquarius

Here's our request:
AQUARIUS ELECTRONICS would like to trade software, of which we have little, and biofeedback hardware, of which we have lots, for software and computer hardware. We especially are looking for a copy of the Intel 8008 assembler, the one which goes with the Intellec 8 .

You asked for a book list. I'd say read, in this order "Alpha Brain Waves" or "Biofeedback," in our catalog, followed by Barbara Brown's "New Mine, New Body" and then perhaps Ornstein's "Psychology of Mind Consciousness." Tart's "Altered States of Consciousness" is heavier going, but good.

Robert M. Coppock
Customer Service
Aquarius Electronics
P. O. Bos 96

Albion, CA 95410

It would be nice to sit down at a terminal and type you a note then send it to a central computer in this city that would use shortwave to route the note to you. I just got the first part of my Altair kit and have the what do I do until it all gets here blues. I've been making a TVT terminal as per an article in Radio TV Electronics. Unfortunately the PC boards were discontinued by the time I got around to ordering them. Made them myself and have modified the termina to use 2102's instead of the recirculating shift registers.

Got to wondering how to hook the TVT up to the Altair without an I/O board. Well it looks like two 4 to 16 line demultiplexors would form a 16 by 16 matrix that could feed 256 gates separated by diodes and then an 8 bit latch should do the job of getting everything in and out. 256 in 256 out. So what happens if you output and the output is hooked to another board of the same type. Assuming that each output connected to a device that would use the 1st byte to address 256 more bytes and then transmit the second byte well that's over 500 K of output. This all leads to using the thing as a central exchange for a club. If there are two such clubs in neighboring cities several I/O's could be used as an inter city communications link. MA BELL lookout!

Enclosed is an old program that got me wasted for wasting computer time on games. Heuristic Tic-Tac-Toe. It learns from mistakes. Unfortunately there is a minor bug that bombs to monitor after 10 games. I never got to debug it.

One good program for say BASIC looks fike this when ran.

## RUN

I DON'T WANT TO
RUNNH
WHAT'S THE MATTER DON'T YOU KNOW HOW TO SPELL RUN

A good version will include all the possible commands plus a control-c intterupt so you are stuck until you type the name of the program to reach the END instruction.

The MARK 8 plans show a digital to analog converter hooked onto the output part. Programs to produce waveforms are given. Better would be to latch the output into a $D$ to $A$ converter and then control a voltage controlled amplifier or oscillator or filter. The Altair's 256 outputs would make a jim dandy synthe sizer.

Has anyone got a program to correct spelling errors? And on that subject I almost took a job at the local newspaper repairing their electronics. Far out all typewriters in the place have the same type. Time to format the paper and they feed everything into a reader that puts it all on mag tape. The IBM then uses format programs to put everything into place and then zips it off on computer controlled typesetting machines.

XANAXAN
WIZARD OF NICE NINE
Joe W. McCarty


# Canada 8 <br>  

Do-It-Yourself

You asked that people who have acquired Altairs write, so here I am writing. I just completed the basic unit of the 8800 (CPU and 256 words memory) and am very impressed. The quality of the kit is as good or better than Heathkits, which is a compliment. My only criticism is the uneven presentation in the operation theory manual. On the one hand they go into great detail about the binary system (elementary for nearly all computer nuts), but then gloss over the programming aspects of the Intel 8080. Wish they could include a few more expertly written sample programs and routines. For example, what is the best way to add two numbers in scientific notation form with 10 digit mantissas and two digit exponents?

The big news is that the school where I teach (the Harvard School in North Hollywood - independent school for boys grades 7-12) has taken my suggestion and ordered the Altair DOS/III system, in kit form! Yes folks, the kids and I are actually going to BUILD our own computer this summer, all for under \$6K. We'll keep you posted about this project, because I think it's something many schools could and should do. Why spend many megabucks on hardware and endless maintenance agreements (we almost bought a DEC system for $\$ 70 \mathrm{~K}$, which means about $\$ 7 \mathrm{~K}$ per year maintenance) when you can have the unparalled learning experience of building and caring for your own.

At the National Computer Conference, 1 asked the Altair people about a timesharing operating system, since we would like to have multiple simultaneous users on our school system. Their response was "Why use timesharing when each person can have his own processor for under \$600?" Indeed! So now there is a conflict in my mind, which I would appreciate having resolved through discussion/communication among computer builders/hobbyists:

THESIS: Timesharing allows multiple users, sharing a common memory (stored programs). Dis advantage is complicated operating system, and fact that a serious malfunction shuts everybody down.

ANTITHESIS: Everyone gets his own Altair 8800 with say 4 K of memory and simple BASIC. One Altiar has 16 K with disk for the big programs. Advantages: faster running times, and malfunctions won't shut everyone down. Disadvantage (big): inefficient use of memory fredundant operating systems), and no commonly shared data base.

POSSIBLE SOLUTION: Is there some way to multiplex several processors connected to a single large memory? Coutd we have several 8800 's running on say 1 K of internal memory but interfaced with say 20K of "common" memory? This may be an elementary computer design problem (I don't know; it seems easy enough). Is someone willing to construct such a system with the necessary software? I think it would be tremendously valuable to schools and other users.

I have enjoyed reading the newsletter very much. and would appreciate hearing from anyone who would like to communicate. As l've indicated, I'll be heavily into designing this school system, with BASIC and machine language operating systems, getting all the good games running, etc. We already have been running timeshared BASIC for five years using TTY's, acoustic couplers, and commercially purchased computer time. so we know what wére aiming for at the minimum.

Thanks again for the good work.

## Gene Murrow

Harvard School
3700 Coldwater Canyon
N. Hollywood, CA 91604


WEASK
${ }^{c} 10^{c}$
LBBERTYANDSAFETY
Fiscqulemminl:İmas?

## Foresight

I have been under the unfortunate impression since last year that the PCC was dead, and only through a chance encounter with the latest Datamation did I conclude that you're alive and well and living in the Bay Area.

I (or rather we) are Foresight, and while we are not as far along as you (one month formally) and not as pure (we are not non-profit) we are interested in the same kinds of things as you. Of course, since we are profit motivated we don't anticipate living off of the same kinds of things as PCC: we aim to provide business software and in the future systems for business and education. But at our (or at least my) heart, PCC has the right way. Way back, maybe in 1973, I had some thoughts about the PCC: even remember reading something by Bob in Saturday Review of Education and saying "precisely". The part of Foresight which will eventually do that kind of thing is KIDS: Kehoe Interactive Data Systems. In the Fall, we hope to have a small mini (maybe a PDP8 or even CLASSIC) to carry around in our van to demo at schools and teach intro BASIC within existing math classes. I'II keep in touch.

We are just getting into the programming business, and hence are still working at miscellaneous jobs in computers, etc. All except Macy, our salesman, who is off to Nova Scotia on his bike (I should say bicycle). When we get Macy back to town in August, maybe we'll get out there to visit. Anyway, keep up the good work.

## Miles Kehoe

Macy Teetor, III
5524 Willow Street
New Orleans, LA 70115


## Personal Experience

For any hobbyists out there, I can recommend the following from personal experience:
Machine Language Programming for the 8008 from Scelbi Computer Consulting, 1322 Rear, Boston Post Road, Milford CT 06460 . Price is $\$ 19.95$
James Electronics, P. O. Box 822, Belmont, Ca. 94002. For parts - fastest service I have experienced. Good prices on prime integrated circuits. Reputed to honor their guarantee instantly.
Poly Paks, P. O. Box 942E, Lynnfield, Ma 01940 also good, but a little slower. Still faster than industrial parts houses!
The Computer Hobbyist, Box 295, Cary, NC 27511.
Hal Singer's MICRO-8 User's Group Newsletter. Excellent!l! Start here! Address: Cabrillo HS, 4350 Constellation, Lompoc, CA. 93436.
The Digital Group's stuff. Also excellent! Calculator chip interface, cassette interface, beautiful TVT, etc. P. O. Box 6528, Denver, CO 80206

Robert Cook for CREED (Baudot Code).TTYs. Excellent man. Unbelievably helpful. Give him your business. Write to him at Wilcox Enterprises, 25 W. 178 - 39th St. Naperville, IL. 60540.

Edward DeGraaf
8516 - 47th St. No. 13
Lyons, IL 60534

## Hooked on Star Trek

I've got an Altair up and running. I should have a TV Typewriter hooked to it before long. Next I hope to get either MITS' BASIC or yours or somebody's. I noticed you were starting a series on a BASIC interpreter. Is that going to come off; I'm so hooked on Star Trek and other computer games I can't wait.

I really liked the article by Steve Dompier on "Music of A Sort". I had my Altair spewing forth Cat Stevens, Beatles and even a little Rolling Stones in no time. It liked "Daisy" best for some strange reason.

I'll keep you informed in how things are coming along and would be glad to hear from other people on how their projects are going.

From the land of Budweiser
(Anchor Steam costs too much down here)
Bob Beard
721 30th St
721 30th St
Hermosa Beach, Ca. 90254

## Computers for the Performing Arts

The person with the names and addresses of - those interested in joining a 8800 Altair Users Group (San Diego) is:

## Frank MacLachlan <br> Gamma Scientific, Inc. <br> 3777 Ruffin Rd. <br> San Diego, Ca. 92123

As I told you, I'm getting into low-cost computers because I'm starting a nonprofit organization to promote the performing arts (classical music, theater) and our production system (IBM Exec., + headline machine) proved too slow to get out the kind of publication and other services we wish to offer. Instead of starting an empire of "slaves" and "gofers" to accomplish the work, I'd rather go into a higher fevel technology.

So we would go to keyboard and CRT display for input, with printer output (giving "hard" copy proof, handling correspondence and address labels) and another output to a phototypesetter (probably an old Photon).

Instead of paper tape, the Photon would get modified to read Philips cassette. Looks like mag tape machine might be Phi-Deck (completely electronic control, can search a 30 -minute cassette in $20-25$ secs, is dual track - from the Economy Corp., Box 25308, Oklahoma City, OK 73125).

I'd like to pursue multiple answers to some of our needs. Both TVT2 and the Suding CRT interface look promising. In the case of TCT2, I'd like to modify it to use the Motorola clc 128-character chip. As to the Suding, would want to figure out how to hook it to a DEC LSI-11.

As to computers, I want to see about getting quickly and easily (laugh here) into a 8008 of some sort, with Suding CRT interface. At same time, would like to take our Altair 8800 budget and put it into Digital Equipment Corp LSI-11 (completely compatible PDP-11, 16 -bit machine with incredibly versatile instruction set, 4 K of RAM, selling for $\$ 990$ for one, $\$ 653$ for $50-99$, backed by DECUS users' organization with 175 programs, free membership and reasonable program acquisition cost.

Our computer would handle all word processing for the organization, as well as some for San Diego Symphony, Opera, Old Globe, etc. Also would maintain subscriber addresses. Eventually might handle accounts receivable. Word Processing problems follow:

LINE LENGTH - On phototypesetting input, would look up character width for each character typed and would decide when to end each line. Would figure word spacing needed to justify right margin. Would be hyphenless justification until dictionary project is online, then would automatically look up word splits.

DICTIONARY - Would put about 35,000 word spelling dictionary, with word splits indicated, onto floppy disc. As writer completes words on keyboard, computer will look them up in dictionary and beep if unfind. That indicates either misspelling or unusual word. Freudian slips will be immune from this treatment, since the typo completes a word, although the wrong word.

ADDRESS MANAGEMENT - Probably on floppy disc. Since nearly all addresses are in San Diego and San Diego County, city, state and complete zip code can be coded with just a three-digit number (county zips a zero and two numbers, San Diego zips a one and two numbers). This should be a time saver for address maintenance (approximately $1 / 3$ of time), as well as conserving storage. There are approximately 50 county zips ( 45 cities). File would be by zip. Alphabet readout not necessary if we keep a card file, so must decide whether to have electronic or card alphabet file.

Another problem we want to pursue in parallel is that of keyboards. Want to try a number of low-cost keyboards to see if they can comfortably be used by someone working all day on them, as typist would. Have seen one keyboard only a hunt-and-peck specialist would tolerate. This would flunk our specifications.

How are we going to pay for this program. Have talked to a newspaper group and they are unfamiliar with all of this stuff. Am suggesting they pay for it all, and get report on our effort, have a lab with it all running available to them. In addition, they will have aided our nonprofit cultural project. I think this is the kind of stuff a newspaper could use quite effectively (for instance, 15 Suding CRT displays and keyboards interfacing with $\$ 1,000$ DEC LSI-11 and floppy dise dictionary, the CRTs, tape units etc., average $\$ 600$ per single unit, very cheap when compared to approximate - $\$ 4,000$ per unit for current Selectric/ OCR system).

I'm going to send copies this letter to area computer experimenters. Also some non-area.

Bill McLaughlin, Editor


Program Guide
Box 158
San Luis Rey, Ca. 92068


## Harciware Stuff

## From Duter Space

1 just received volumes $\mathbf{2}$ and $\mathbf{3}$ or your newspaper. Miscellaneous comments:
(1) What's all this about dragons??? (Dragons would stay away from computers. Their fiery breath would burn out the ICs.)
(2) Why does the "Iow cost home/school microprocessor system" use COSMAC? It is the most expensive microprocessor I know of.
(3) More suggestions for INCHWORM: Add a conditional branch capability, and perhaps a WAIT provision (stay somewhere for a while).
(4) I doubt if BASIC is the best language for small homemade systems where memory is limited. A Reverse Polish notation language similar to the system used on HP calculators would use less memory, be simpler to develop, and interpret faster. I hope to play around with such a language and maybe even implement it if and when I get around to building a microcomputer. (My interests are primarily theoretical - I don't usually actually do things, just figure out how.)
(5) The Do It Yourself PDP-8 or Microprocessors Strike Again! (Edward Epp \& Sorosh Ahmed, take note.) a. get rid of those TTL books, they won't be needed. b. Write and ask Intersil, 10900 N. Tantau Ave., Cupertino, Ca. 95014, for information on their IM6100 microprocessor and support circuitry ( 6508 RAM, etc.). The IM6100 is a 12 bit microprocessor with the PDP-8 instruction set. It is a fairly expensive microprocessor but has the additional advantages of an on chip clock, and, being CMOS, operates over an extended tempera ture range, has fully static internal circuitry, doesn't need regulation of the single 5 to 10 volt power supply, and only uses .01 watt of power, as well as having excellent noise immunity.
c. Design the CPU, using CMOS support circuits (see later section).
d. Decide on the memory to be used. If you settle for 2102 s , the whole system with 4 K bytes of memory would cost between $\$ 500$ and $\$ 700$. If, however, you're willing to pay around $\$ 300$ per kilobyte of memory, you could make an all CMOS system with several major advantages:

1. It would operate at 10 volts and would therefore be 40\% faster.
2. It would run on batteries (rechargable) and could easily be built into a briefcase for a portable system ( $\$ 40$ would add a solar battery chargerl).
3. Continuously applied power would make the memory nonvolatile.
e. Build it.
(6) All Star Trek programs I've seen (and some I haven't) are based on a space battle - but the show isn't. I've invented a Star Trek game (G20) that isn't battle based. It is played on a huge 3-D board and the primary objective is to explore planets. If anyone has several months of computer time and would like to try programming it let me know and I'll supply a more detailed explanation. Warning: It's complicated. Definitely not for BASIC.
(7) More Star Trek. My former high school had a HP9100B programmable calculator and I programmed it to draw the Enterprise and Klingon ships. I'm including a copy. The Klingon program is probably one of the most sophisticated programs ever written for the 9100B. After two revisions, every possible step saving technique and some impossible ones are used to cram the program in the 392 step capability of the machine. Anyone want the programs?

(8) The Warp Factor (no, this doesn't have anything to do with Star Trek) - my high school also had a timeshare terminal for an IBM 360. I discovered that outputting an integer * 2 variable between 256 and 511 inhibited output and had all sorts of exotic effects. (In FORTRAN, of course.) This number, named the Warp Factor (also how tilted our pool table was!), came in handy quite a few times. Anyone with an IBM 360 terminal might want to experiment with it. I have no idea if it works on other systems.
(9) Back to hardware - most of this has been sent to the MICRO-8 Group

Use CMOS for all new circuits. The long predicted CMOS price drop has rendered all TTL based 8008 and nearly all TTL 8080 circuitry obsolete. See PDP-8 section for the advantages of CMOS.

Stay away from dynamic RAMs. They're designed for big systems and just aren't economical for small systems. For a 4K RAM try the Electronic Memories \& Magnetic 4401 \& 4402. They are lightning fast, static, and probably cost a fortune, If "char je pump" 4 K RAMs are out they would be an alternative. (semistatic and even faster.)

Don't buy the Altair 8800 . It's TTL and now vastiy overpriced. Build your own and save $50 \%$.

If even that is too expensive, keep a very close eye on MOS Technology's 6501. Altho it won't be available until late this year, the 100 quantity price is only $\$ 201$ and it's NMOS with 8080 capabilities.

As soon as I find out something about the ICs, I hope to design improved 8008 and 8080 microcomputers. I'tl keep you up to date on my efforts.

Incidentally, I'll be taking some minicomputer courses at Johns Hopkins University this fall. It should be interesting. Well, that's all for now.

Bobby Baum
6607 Pyle Rd.
Bethesda, MD 20034

Note on HP 9100B plots - I'm nearly out of Enterprises so I couldn't send a full sheet of paper. Arithmegig is the mathematical division of GIPPGIG Consolidated Enterprises, my one man company that does nearly everything.

## 1/O Bus-Altair

I've done something that might be of interest to you and your readers. I've designed and built a parallel I/O BUS for my Altair. Back in March, my $\$ 397$ Altair with 256 words of memory was running - with no hope of getting any more goodies from MITS for a good long while - I decided to design my own I/O interface with as little impact as possible on the basic 8080 I/O configuration.

I liked the RGS Electronics I/O BUS philosophy - 256 I/O devices multi-dropped on a single I/O BUS and providing input, output, and control to each device - so that's what I did.

My circuit uses four Altair I/O port addresses to provide the parallel I/O capability. Actually I've only taken three ports as my CONTROL strobe coresides on prot 255 with the Altair sense switches. The RGS Electronics parallel 1/O board (or kit) can be used to interface an $1 / 0$ device to the parrallel I/O BUS. Therefore, any RGS Electronics I/O device can be run on the Altair - or your own device or you can construct your own I/O device interface. Anyway . . . it really works great.

The paraliel I/O interface board recognizes addresses $252,253,254$ and 255 , and according to the actual address decoded - plus the command tself
(I/P or $\mathrm{O} / \mathrm{P}$ ) provides a strobe signal (select device, input, output, or control out) down the I/O BUS. Data from the 8080 AC accompanies the strobe to the previously selected device interface. If the strobe is a select, all I/O device interfaces on the BUS take the data as a device address. If an address match occurs, that device is then the selected device and will remain selected. Selecting a non-existant device will de-select all devices on the I/O BUS.

I built my original paralliel I/O BUS interface on a breadboard (it's still therel), but as soon as time permits, I'm going to build it on an Altair 88 PPCB. Hopefully, by the time you get this note, it'll be on the 88PPCB. Of course - it can also be built on a SSM General Purpose board.

I'm putting together a full set of documentation I have some friends who want to build it - and when I have it completed (hopefully within a week) I'll send you a copy.

Jim Brick
820 Sweetbay Drive
Sunnyvale, Ca. 94086

## Music \& Games \&i Ching

1 just finished reading Vol. 3 Nos 1-4, and 1 am really excited. As soon as school starts again, I'll be writing some programs of course. A few of the things that I'm going to be doing -
I CHING a complete printout, hexagram, decision, commentary, everything. Using initialization of RND to generate oach line.
MUSIC PROGRAMS something to compose it (software) and print a staff something to play it directly from the TTY without completely rewiring the TTY (because its the schools TTY).
SOMETHING to generate words, which is the reason this is on graph paper, I used my last piece of regular paper to write Byron Caloz (PCC, Mar, ${ }^{\prime} 75$ page 10) about his SUMWRD. What I have is a formula for monosyllabic possible English words. See your local copy of "Essays on Language and Usage" for more, it's one of the essays. I have a rough program (it only works half the time) and I do get some words off it.

- Another thing that I am thinking about is using a mini to control some aspects of a synthesizer, such as maybe envelope or hows about sticking a minicomposer in there and then jam along with it! As soon as I get some dough together I am building a synth, so I'd appreciate any info you guys have on all aspects of synths, hard and soft wise.

Another thing that I want to do is to program some really complex simulations, the socio-econom-politico polemic kind, which is every kind. My school has two TTY's, so what l'd like to do is to inter-connect them and get the computer to act as an umpire and bookkeeper using LSI (that's limited strategic intelligence). Maybe you could play against the computer in a game such as USN or Diplomacy - any thoughts on that? Along this line, do you have a run of ST AR TRADER in any issue? If not, l'd really like to see it.

Maybe you'd like someone to write an article on this. If you don't mind, I'd like to help. Also, if you know of anyone with similar interests, I'd appreciate it if you connected us.


Tony Audas
5530 Pebbleshire Rd.
Birmingham, MI 48010

## THE ENERGY PRIMER

Portola Institute
If all of the recent books and articles on non-depletable energy resources spawned by the "Energy Crisis" were to be judged on the basis of the contribution they can make to the average reader's energy picture, most would serve their highest as barbecue lighters. After showing you how to harness the tides of the Bay of Fundy for only $\$ 60,000,000$ or heat your house with solar energy for $\$ 14,000$ or pave the Southwestern desert with solar cells, most authors have shot their wad.
Not so with the Portola Institute's (The Whole Earth people) "Energy Primer". While the Primer does not totally ignore the more grandiose schemes that are the stuff that Sunday supplements are made of, by far the larger part of the book is made up of information usable to the individual or small community.
The section on wind power presents an almost total picture of the state of the art, of generating electricity, pumping water or running a mill with wind. The sources, prices and application information are enough to permit an economic analysis just from the information contained in the book. Just as in the sections on solar, water and biofuel, you are first given an overview of the theoretical factors involved. This is both comprehensive enough to be understandable to any layman possessed of the practical skills needed to start out on a program of energy independence. Next comes a listing of additional reference sources, and comprehensive data on most of the wind machines available on the market today.

The configuration of the other sections follows the same pattern of introduction; explanation, references and sources. One of the characteristics that I as an Engineer appreciate is the fact that none of the authors make it sound easy. There is none of this " $\$ 200$ makes you free of PG \& E (or Con Ed) forever." The difference between being tied to the infinite capacity of a utility and having to match your demands to your own resources is clearly brought out. In fact, some of the energy discipline this book suggests would be a good idea for us all to practice even when we do have utility connections.

If any section of the book could be said to be my favorite, it would be the section on Architecture. Here, one of the most important lessons, how to configure your residence to require less total energy input, is given thorough treatment. Even the apartment dweller can find valuable advice to cut down energy use without degredation of comfort. If the principles expounded here were to become general practice, the Arabs would be out of their Cadillacs and back on camels in no time. Closer to home, more of the paycheck would be left for the finer things in life.

This is one of the few books about which it can be said, there is something in it for everyone.



## TITLES

BASIC, Albrecht, Finkel \& Brown, 1973, p. 323 - \$3.95 Basic BASIC, James Coan, 1970, p. 256 - \$7.45

BASIC PROGRAMMING, Kemeny \& Kurtz, 1967, p. 145-\$6.95
COMPUTERS \& COMPUTATION, Scientific American, p. 280 - $\$ 6.00$ COMPUTER LIB \& DREAM MACHINES, Theodore H. Nelson, 1974, p. 186 - $\$ 7.00$ DRAGON SHIRTS, Nancy Hertert, 1974 - \$3.50
GAMES, TRICKS \& PUZZLES, Wallace Judd, 1974, p. 100-\$3.95 GIMME SOMETHING TO FEEL, Jane Wood, 1973, p. 125 - \$2.95

MATH,WRITING \& GAMES, Herbert Kohl, 1974, p. 252-\$2.45
MY COMPUTER LIKES ME, Bob Albrecht, 1972, p. $64-\$ 2.00$
101 BASIC GAMES, Ed. David Ahl, 1974, p. 250-\$7.50
PROBLEMS FOR COMPUTER SOLUTION, Gruenberger \& Jaffray, 1965 - $\$ 7.25$
PROFESSOR GOOGOL, Sam Valenza, Jr., 1973, p. 144 - $\$ 3.25$ PROBABILITY, D. J. Koosis, 1973, p. 163 - $\$ 2.95$

PCC GAMES, Program Listings - \$2.00
PRACTICAL, LOW-COST HOME/SCHOOL MICROPROCESSOR SYSTEM, 1974 - $\$ 1.00$ SELLING WHAT YOU MAKE, Jane Wood, 1973, p. 111 - \$2.25 STATISTICS, D. J. Koosis, 1972, p. 282 - \$3.95
THE ENERGY PRIMER, Portola Institute, 1974, p. $200-\$ 4.50$ TTL COOKBOOK, Don Lancaster, 1974, p. 328 - \$7.95

II CYBERNETIC FRONTIERS, Stewart Brand, 1974, p. $96-\$ 2.00$
WHOLE EARTH EPILOG, Stewart Brand, Editor, 1974, p. 318-\$4.00 WHAT TO DO AFTER YOU HIT RETURN, PCC, 1975 , p. $157-\$ 6.95$ DIGITAL LOGIC CIRCUITS, Sol Libes, 1975, p. 184 - $\$ 5.98$


## 31

TO ORDER -
name
address
zip

| Quantity | Item | Price Each | Total |
| :---: | :---: | :---: | :---: |
|  |  |  |  |
|  |  |  |  |
| * Please add $\$ 0.50$ for orders under $\$ 10.00$ and $\$ 1.00$ for orders $\$ 1.00$ and over. |  |  |  |
|  |  | Sales Tax Postage* |  |
| pce P. P. Box 310 Menlo Park, Ca. 94025 |  | Total |  |


edfitor bob albrecht production mary jo albrecht circulation lois baitton contributors

## COMTENTS

1 Subscription Information
2 Chiptalk
4 Dragonsmoke
5 Domefilm
6 Kingdom
10 CHOMP
11 WITCHY
12 Criticism \& Self Criticism
13 Signals \& Glitches
14 BYTE
15 Design Notes for Tiny BASIC
18 Tiny BASIC Letters
19 Altair BASIC
23 Assembly Language Programming
24 LIFE on the Altair
26 Letters
30 Energy Primer
31 PCC Bookstore

Oback issues


41
O vol. III
$\$$

## PEOPLES COMPUTER COMPANY

## Robin Allison

Dennis Allison Bernard Greening Lee Felsenstein Kent Cross Dean Kahn Carter dePaul Bob Mullen Mac Mullen Old Soldier Jerald Brown

## R.O. Box 370 Menlo Park, Ca. 94025

$\qquad$
Oregular ${ }^{1 \text { y yer }} \mathrm{s5} \mathrm{O} 2$ vears $s 9 \mathrm{O}$
Ogroup (how many?) sip

06 is a not-for-profit corporation. The newspaper is about recreational and educational uses of computers - computers for everyone. PCC is published 6 times during the year.

## SMBANi)NB are $\$ 5.00$ for 6 issues. ( $\$ 6.00$

 outside the U.S.A. - surface mail; $\$ 12.00$ air mail.) Subscriptions begin with the July issue.Nert Issue
Unusual and exotic devices for your home, school or personal computer. Beginner's info on computer arithmetic and machine language programming. More BUILD YOUR OWN BASIC. And of course, games!

Pef 6 glies get them while they last at the following
low low prices:

$$
\begin{array}{ll}
\text { Vol. I, Nos. 1-3t } & \$ 1.00 \\
\text { Vol. II, Nos 5 \& } 6 \dagger & \$ 1.00 \\
\text { Vol. III, Nos. 1-5 } & \$ 4.00
\end{array}
$$

$\dagger$ Sorry, we are out of Vol. 1, Nos, $4 \& 5$ and Vol II, Nos. 1-4.
Or mix up individual issues -

|  | $\$ 0.80$ each |
| :--- | :--- | :--- |
|  | $\$ 0.70$ each |
| $\$ 0.60$ each |  |


| $2-9$ | $\$ 4.00$ each |
| :---: | :---: |
| $10-99$ | $\$ 3.50$ each |
| 100 or more | $\$ 3.00$ each |

## Donations

People's Computer Company is a tax-exempt, non-profit corporation. Donations to People's Computer Company may be deducted. Retaining subscriptions to the People's Computer Company are $\$ 25$ for one year ( $\$ 20$ tax deductible). Sustaining subscriptions are $\$ 100+$ per year ( $\$ 95+$ tax deductible). Names of retaining and sustaining subscribers will be in the newspaper.


[^0]:    
    

[^1]:    1000 INPUT＂ 10 ASSWORD＂ $3 B \$:$ REM GET HASSWOKD
    1010 FOK $X=1$ IO LEN(BS):REM CONVEKT TO ASCII. BUILD SEED
    $1020 S=S / 10+A S C(M I D S(B S, X)): N E X T$
    1030 IF $S>1$ THEN $S=S / 10:$ GOTO 1030 :KEM SET $S<1$
    $1040 \quad S=$ KiND $(-S)$ :REM SEED KND GENERATOR
    1050 INPUT"IN OK OUT" 3 B $\$:$ REM SET PROGRAM SWITCH
    $1060 \quad S=-1$ :IF LEFT $\$(B \$, 1)=" O "$ THEN $S=1$
    $1100 \quad \mathrm{X}=\mathrm{ASC}(\mathrm{AS})+\operatorname{SGN}(S) * I N T(128 * R N D(.5))$ REM CONVERT CHR AND CHANGE
    1110 IF $x>127$ THEN $x=x-128:$ REM CONVERT TO STRING AND QUIT
    112 IF $x<0$ THEN $x=x+128$
    $1130 \mathrm{~A} \$=\operatorname{CHR}(X):$ RETURN
    OK

