THE COMPUTER JOURNAL®

For Those Who Interface, Build, and Apply Micros

Vol. II. No. 3

\$2.50 US

Heuristic Search in Hi-Q page 2

Build a High-Resolution S-100 Graphics Board

Part Two: Theory of Operation Page 12

Multi-user:

Etherseries Page 17

System Integration

Part Two: Disk Controllers and CP/M 2.2 System Generation Page 21

New Products page 28

The Computerist's Calendar Page 20

Editor's Page

The End of the "Programmer Prima Donna"

The computer software market is changing rapidly now that more computers are being used in offices and homes by non-computerists. Until recently most of the microcomputers were used by people interested in computers. These knowledgeable users were willing to make do with awkward programs or rewrite them, but the market is changing and has now reached a point where programs will have to be designed for the end user in order to be successful.

When microcomputers first became popular, only programmers knew enough about these marvelous new devices to foresee what they could do. These pioneering individuals wrote business programs, and we were amazed at the power available in word processors, data bases, and other useful programs. These early business programs were the primary reason for the rapid expansion of micro sales.

Unfortunately, the programmers (who were the only ones to understand the micro) did not understand the requirements of a business system. The result was that hundreds of programs were developed which were difficult to use, and which *almost* did the job. The programmers worked in isolation and produced elaborate programs which sold because they were so much better than working without a computer.

The programmers were "high priests" who decided what the customers should use, and the customers did not have enough experience with computers to intelligently evaluate the offerings. The programmers also lost sight of who their users were, and were not really interested in understanding their needs. The software industry talks about alpha and beta testing of programs, but this testing was (and perhaps still is) done by experienced computerists. The testing should really be done by the lowest level of people expected to use the program. If you are selling a program which will be purchased and used by other programmers it should be tested by programmers. If the program will be used by people on the street, it should be tested by people on the street. It took a long time for other industries to realize that they had to identify their market, really visualize the individuals expected to lay their cash on the counter and put the product to use, and then spend enough time in the users' environment to understand what they did in a typical work day. If you want to test a program for real estate offices, you should load the program and a computer into your car, and drive around until you find an office whose personnel represent the least knowledgeable segment of your market. Offer them whatever cash it takes to have them try your program using the computer you supply, give them the documentation, and then sit in the corner and watch what happens. If they have to ask you a question, or you have to point out something they are doing wrong, the experiment has failed and you had better go back and make revisions. You may also find that while the program is easy to understand and put to use, it just may not perform the required functions.

The problem of non-performing software became very evident to us while we were searching for a data base for use on our new CP/M machine. We had been using *The General Manager* by Sierra On-Line on our Apple[®], and were quite satisfied with it, but we are changing to a S-100 system and need a simple data base. So far the offerings we have seen for CP/M are expensive, hard to use, and are incapable of handling our mail list needs. We also reviewed some mail list programs, but they failed to handle our needs for 3rd *continued on page 25*

| Editor/Publisher | Art Carlson |
|----------------------|----------------|
| Art Director | Joan Thompson |
| Technical Editor | Lance Rose |
| Technical Editor | Phil Wells |
| Production Assistant | Judie Overbeek |
| Contributing Editor | Ernie Brooner |

The Computer Journal[®] is published 12 times a year. Annual subscription is \$24 in the U.S., \$30 in Canada, and \$39 in other countries.

Entire contents copyright © 1984 by The Computer Journal.

Postmaster: Send address changes to: The Computer Journal, P.O. Box 1697, Kalispell, MT 59903-1697.

Address all editorial, advertising and subscription inquires to: The Computer Journal, P.O. Box 1697, Kalispell, MT 59903-1697.

HEURISTIC SEARCH IN HI-Q

by Henry W. Davis

Computer Science Department Wright State University, Dayton, Ohio

C omputer games challenge humans to be smart. Have you ever wanted to reverse the role? This article is about how to write a program which makes the computer appear as smart as a human, or smarter! It also demonstrates a technique called "heuristic search," an important component in many artificial intelligence systems.

Our medium is "hi-Q," a popular puzzle played in a certain chain of restaurants by customers who are waiting for their food. When I first encountered hi-Q, I was so intrigued that I decided to share it with my students in an artificial intelligence course at Wright State University. The course includes heuristic search algorithms. On two occasions, I asked students to try these algorithms on hi-Q. They used a wide variety of languages and computers, from BASIC, FORTRAN, and PASCAL on home computers to SIMSCRIPT on a CYBER. They obtained a number of fascinating results showing that heuristic programs can do quite well at hi-Q, considerably better than most humans.

In this article, I will describe the basic algorithm used along with specifics of the hi-Q environment. Then I would like to share with you some aspects of a particularly nice program written by Ed Dudzinski, a former masters degree student in computer science at Wright State. Ed currently works on software optimization for array processors at the Wright-Patterson Air Force Base in Dayton, Ohio. His program, written in FORTRAN, is interesting because it performs well while demanding little memory—less than 40,000 bytes in a home computing environment. Nevertheless it has analyzed game situations involving as many as 190,000 different board configurations.

Our results are only a beginning. It is still not clear what the best computer hi-Q strategy is, even for the simplest hi-Q version described below.

The Rules of Hi-Q

A common version of hi-Q has 21 holes drilled into a piece of wood. Figure 1 shows the pattern: there is assumed to be one hole in the center of each of the numbered squares. The puzzle begins with pegs in 20 of the holes. The goal is to "jump and remove" 19 of these pegs, ending with only one peg on the board. The rules are as follows:

1) A peg can be moved only by jumping and thereby removing a single adjacent peg.

2) A peg can jump either horizontally (eg., from hole 5 to hole 7), vertically (eg., hole 17 to 7), or diagonally (eg., hole 7 to hole 15).

3) The jump can be made only if the destination is empty (eg., hole 7 in a 5-to-7 jump) and the jumped hole (hole 6) is full; the jumped peg is removed.

Figure 2 shows the first three moves of a typical game. This

| l | , | 2 | 2 | |
|----|----|----|----|----|
| | | 2 | 3 | |
| 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 |
| | 19 | 20 | 21 | |

Figure 1: Hi-Q is played on a board with positions patterned as shown above. In the center of each numbered square is a hole for a single peg. Initially 20 pegs are placed randomly into the holes. The pegs may jump one another horizontally. vertically or diagonally as long as they land in a vacant location. Jumped pegs are removed from the board and the goal is to remove all but one peg.

algorithm is a variation of Nils Nilsson's "ordered search algorithm" which may be found in Chapter 3 of his 1971 book **Problem solving Methods in Artificial Intelligence.** This is also the "graphsearch procedure in Chapter 2 of his 1981 book **Principles of Artificial Intelligence.** It is shown in Figure 4 and explained below.

The many possible configurations of the hi-Q board are called *states*. The states are connected by arrows, or directed arcs, which represent legal moves transforming one state into another. The result is a directed graph, called the *state space*. Figure 3 shows a very small portion of the state space for the hi-Q puzzle whose initial state is Figure 2a.

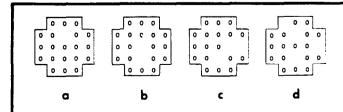


Figure 2: A typical initial state for hi-Q is shown in Figure 2a. Any square could be left empty, but in this case it's square number 6 (using the numbering scheme of Figure 1). Figures 2b. 2c, and 2d show the results of typical first, second and third moves. For example, in the third move, the peg at 1 jumps the peg at 6 and lands in location 12.

game was initialized by arbitrarily choosing hole 6 to be empty. The game is easily played and studied by simply drawing a big version of Figure 1 (without the numbers) and using pennies instead of pegs. In fact, for this reason it is sometimes called the "20-penny puzzle."

し湯

い鍋

1.8

白糖

ាត្រ

- 4

14**6**

8**-%**

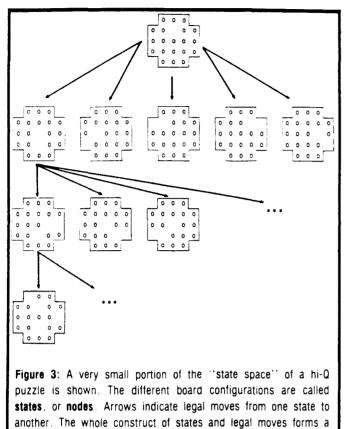
: 4

៍រាំ

The version just described will stump most people for quite a while. Try it! For the very ambitious there are harder versions coming up.

The Basic Algorithm

Algorithm The basic hi-Q search on's "ordered search



directed graph, called the state space.

The algorithm "knows" the legal moves and takes as input an initial state. It performs all legal moves on the initial state, generating new states. The legal moves are then performed on some or all of these new states obtaining still more states, etc. Intuitively, this process has the program "wandering around the state space" looking for a goal state—much like a rat in a maze. Obviously if this is to work, the program needs some bookkeeping devices to keep track of where it has been, how it got there, and where it might still go. It also needs a decision mechanism to determine where to go next. The key ingredients for doing this are the open list, the closed list, the search tree, and the evaluation function, each described below.

1) OPEN is a list of states, or nodes, in the state space which the program knows about but to which the legal moves have not yet been applied. Initially the beginning puzzle configuration is placed on OPEN. In a typical cycle of the search procedure a node is removed from OPEN and all legal moves are applied to it obtaining a set of successor nodes. One says that the node removed from OPEN was expanded and that the successor nodes were generated from it.

2) CLOSED is a list of nodes which have been removed from OPEN and expanded. Why bother to remember a node we have already expanded and hence seem to be through with? The reason is so that whenever we generate a new node we can tell whether or not it has been previously generated. Namely, we compare the new node with the entries on OPEN and CLOSED. If we find a copy of it there, then we will behave differently than we would had the node not been

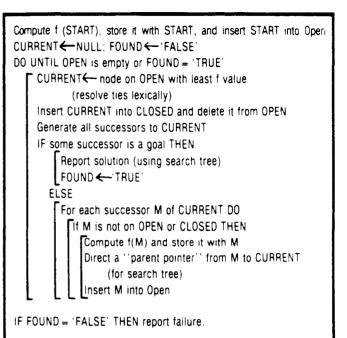


Figure 4: The basic hi-Q search algorithm starts by putting the initial puzzle configuration on the OPEN list. In the main iteration a node is removed from OPEN and its successors are generated by applying all legal moves. If a goal (only one peg left) is not found, then those successors not previously seen are added to OPEN. If a goal is found, then search tree pointers enable the program to report the route it discovered.

previously discovered.

3) When a new node is generated, the program will place in its record of the node a pointer to the node's parent. The resulting structure of nodes and pointers forms a tree, called the search tree. The initial problem state is the tree's root. The sole purpose of the search tree is to enable the program to find its way back to the start once it has found the goal. By following the parent pointers back to the root, a listing of the legal moves from start to goal is obtained.

4) We come now to the crucial question: In what pattern shall the program "move through the state space?" This is equivalent to asking "in what order shall it expand the nodes which are on OPEN?" This decision is made by an evaluation function, f. When f is evaluated on a puzzle state it returns a real number: the smaller that number is, the more likely it is felt that the given state is close to a goal state. The search program expands that node on OPEN whose f value is lowest. Most of the apparent "smartness" of our program is due to f. Without a good evaluation function, all is lost.

Let us examine the algorithm of Figure 4 more closely. In the outside iteration, the program removes from OPEN the "most promising node" (lowest f value), puts this node on CLOSED and then expands it, obtaining all legal successors. If there is no solution, then those nodes not seen before are placed on OPEN, and another iteration is made. CURRENT is a location which references the node now being expanded. If several nodes on OPEN are tied with the lowest f value, then the "lexically" lowest is chosen. This simply means that the program views the board description as a string of

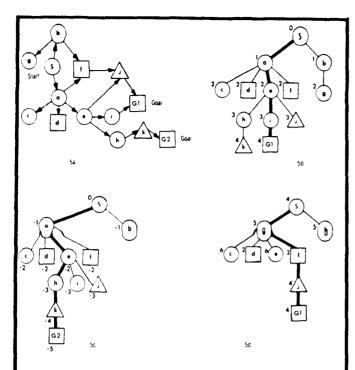


Figure 5: The state space for a fictitious puzzle is shown in Figure 5a. Figures 5b, 5c, and 5d show the search tree and solution (darkened path) found by three evaluation functions. In 5b, f rewards being close to START, yielding a breadth first search. In 5c, f rewards being far away from START, yielding a depth first search. In 5d, the f of 5b has added to it a heuristic component which punishes nodes for being circular, less for being triangular and not at all for being square. The numbers beside each node are the f values (see article for computation details). It is assumed that ties are broken lexically (eg., in figure 5b node a is expanded before node b).

characters and chooses the smallest string relative to normal sorting of character strings. Alternatively, such ties could be broken "arbitrarily." But it turns out that the success of many evaluation functions on a particular initial puzzle varies tremendously (sometimes by a factor of 1000) with how such ties are broken. Rather than leaving the tiebreaking up to an accident of coding, we choose to make it an explicit part of the algorithm. For one thing, our results are more easily duplicated by others. We return to this problem later because it raises the question of how we can properly judge the effectiveness of an evaluation function.

Evaluation Functions: The Main Source of Intelligence

It is important to understand how different evaluation functions can alter the search pattern. Figure 5 illustrates this. In Figure 5a the state space for a fictitious puzzle is shown. Some of the states are shaped as circles and others as triangles or squares. Each state is given a name (eg., S,a,b, or G2) and there are two goal states. For each node N in Figure 5a let g(N) be the least distance from S to N that the program has seen at a given instance, where we assign to each directed arc a distance of 1. One common practice is to set f(N) = g(N). Figure 5b shows the search tree that results when the algorithm of Figure 4 is run on this example. The darkened path shows the particular solution which is discovered. The numbers beside each node give its f value. As the algorithm iterates, the search tree is built up level-by-level, left-to-right. A completed level in the search tree corresponds to having investigated all nodes within a fixed distance of START in the state space. Due to this systematic expansion from START in all directions, the algorithm is called "breadth first." It is an example of a "blind search" because no heuristics are used.

Another type of blind search is called "depth first" because it always favors expanding those nodes which are furthest away from START. To achieve it in the puzzle of Figure 5a, set f(START) = 0; whenever a node Q is generated from a node P, we set f(Q) = f(P) - 1. The resulting search tree and solution obtained is shown in Figure 5c. A deeper and more costly goal is found than in the breadth first case; however, fewer nodes are generated. For hi-Q it is very important to go deep quickly. More about this shortly.

Now suppose we would like to have a more-or-less breadth first search tempered by some heuristic information. For example, suppose a hunch tells us that whenever the program generates a square node (eg., like d or f in Figure 5a) then it is probably close to a solution and it should keep moving in that direction. A triangular node (like j in Figure 5a) is similar but not as good. Then we might define the "heuristic function" h by:

> h(N) = 0 if N is square h(N) = 1 if N is triangular

h(N) = 4 if N is round

The heuristic function punishes nodes for being round or triangular, but the latter less. Define f = g + h. The effect of g is to "add breadth to the search." If the program gets carried away following square and triangular nodes deeply into the state space, then sooner or later g will grow and force it back to shallower nodes which it ignored earlier. In Figure 5d this f is applied to the puzzle of Figure 5a. It happens to work very well: the least cost goal is found while fewer nodes are generated or expanded than with the other evaluation functions. Setting f = h and dropping the g out works just as well in this case. In general, however, conventional wisdom suggests leaving the g component in for the reasons mentioned above: h might sometimes lead one astray and g helps cover for that event. (It's easy to alter Figure 5a so that this happens.)

The Hi-Q Landscape

Since the state space for hi-Q is finite, even a blind search, like breadth first, will eventually find a goal. The solution it reports will require only 19 moves because all solutions require exactly 19 moves. On what basis, then, can we say that one computer search heuristic is better than another? There are several criteria which students in my classes have used:

a) The number of nodes generated: An algorithm which consistently generated fewer nodes then others would seem to be working less hard.

b) The number of nodes expanded: When a node is expanded the algorithm is saying, "I think the goal is in this

19

16**9)**

彩色

ı.

· 41

12**9**

فنرد

- i Ladi
- 1.3**8** 13**8**
- : أهبر
- 24.**9**

沁得

14

direction." If only 19 nodes are expanded, then the program went directly to the solution. I've never seen a human do that. An algorithm which consistantly scores in the low twenties in this area would have to be considered good.

c) CPU time: A low value is good while a high value offsets good performance in the first two areas.

d) Consistency: A good algorithm performs well for all initial states.

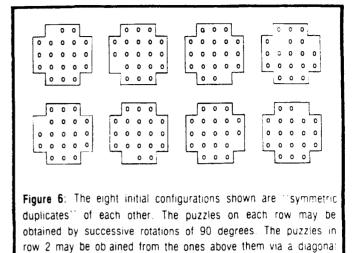
As we noted earlier, the performance of an evaluation function varies tremendously on a given initial puzzle when the f ties on OPEN are broken differently. A related phenomenen is that the same algorithm will often get wildly different answers when working on "symmetric duplicates." One configuration is a symmetric duplicate (SD) of another if it may be mapped into the other by a combination of 90° rotations and reflections on the diagonals. The initial configurations of Figure 6 are all SDs. In row two of Figure 3, states three and four are SDs. My students and I were initially surprised to see the same algorithm generate 100 nodes to solve one puzzle and then 100,000 nodes to solve one of its SDs. This will happen even if ties are broken lexically. The reason is very simple: most people code their programs so that symmetric holes are assigned different numbers and these numbers affect the order in which successors are generated. Thus the successors of two SDs will usually be put on OPEN in a different order. The effect is that ties are broken differently when the program works on two unequal SDs. Lexical breaking of ties does not help because the fact that one state is lexically smaller than another is no guarantee that an SD of the one state will be lexically smaller than an SD of the other.

There are several possible approaches to this problem. One is to collect statistics in categories (a), (b), (c), and (d), above, for all 21 initial puzzle configurations. This method was used to evaluate the Dudzinski program described below. A very elegant approach was taken by Joel W. Arnold: the order in which successors to a node are generated depends only on the class of SDs to which the node belongs and not on which particular SD it is. SD nodes generate SD successors in a corresponding order. His program behaves identically with respect to (a), (b), and (c) when input states are SDs. (Arnold's SDs are relative to rotation, but not to reflection.)

It is common to include a breadth component g (discussed above) in the evaluation function of puzzles. My students have universally found that in hi-Q this is harmful. On the contrary, a depth component is required. Without this, the search tree becomes too wide and programs run out of memory, or time. The evaluation functions we use have the form f = d + h, where h is the heuristic component and d forces depth, much the same way that h forces breadth. In hi-Q a natural form for d is d(N) = number of pegs left in the configuration N.

Heuristics and Results

Five search techniques are compared in Ed Dudzinski's program. All use evaluation functions of the form f = d + h,



where d is the depth component discussed above (number of pegs on board) and h is a heuristic function. All searches are depth first. This is achieved by requiring that each h be less than one in absolute value and never change sign. Thus the dominant rule in choosing nodes from OPEN is "take the deepest;" only if there are depth ties is h relevant. Further ties are resolved lexically. The five heuristic functions compared are as follows:

reflection. Puzzles with more than one peg missing can also be

SD's. Treating SD's "equally" requires careful coding.

1) Blind depth search; h is zero. This gives a basis for comparison of the other heuristic functions.

2) Avoid filled corners; h is (number of filled corners)/10. The corner holes are numbered 1, 4, 14, 19, 21, 18, 8, and 3 in Figure 1. This heuristic punishes states with filled corners. The 1/10 factor keeps h less than unity assuring a depth first search.

3) Favor a filled center; h is - (number of filled holes)/10. Nodes are rewarded to the extent that their nine centermost holes are filled.

4) Favor filled pivotal spots; h is -(number of filled pivotal spots)/5. In Figure 1 locations 6, 10, 16, and 12 are called pivotal because more jumps can be made from and over them than any other hole.

5) Favor a high degree of freedom; h is - (number of possible next moves)/30. This heuristic says to move in such a way as to keep the number of options at a maximum. The maximum number of successors a node can have is about 18.

Ed ran the program on all 21 initial configurations using each of the five evaluation functions. Table 1 gives a summary of his results. The five algorithms are compared with respect to average number of nodes generated, average number of nodes expanded, and average CPU time required. To measure the consistency, or stability, of the algorithms, the standard deviation is also given. Table 2 gives the actual results for each algorithm and each possible start state. Ed evaluates the situation as follows:

The exhaustive depth-first search algorithm with lexical discrimination gives surprisingly good results.

but this seems somewhat accidental. As algorithm 2 shows, and as other runs not included here have borne out, there are dead end branches that can easily trap an uninformed depth-first search into generating many thousands of nodes.

Results for algorithm 2 (avoid filled corners) are deceptive in the average. For all but three cases, this approach gives very good results with a small investment in CPU time. Unfortunately, one run caused the expansion of almost 190,000 nodes, greatly swelling the mean figures. Hence, stability was poor, to say the least.

The last three algorithms show more stability. Approach 3, favoring pegs in center holes, significantly improves on exhaustive depth-first search in all categories. Algorithm 4 (pivotal holes) improves on algorithm 3.

By far the most stable was the 5th algorithm (degree of freedom). Obviously, this approach will result in the generation of a lot of nodes, since it chooses the path of most successors: the average number of nodes generated by this approach exceeds the values for algorithms 3 and 4. However, the performance of algorithm 5 in terms of nodes expanded, or actual moves, is remarkable. Six times, out of the 21 starting configurations, the solution is found in the minimum 19 moves! In fact, for only two of the starting states does the program exceed 22 moves. The price for this facility of decision is paid in CPU time. Calculations of the number of possible next moves is considerably more time-consuming than methods used by the other algorithms.

The ideal heuristic function, if it exists, will combine the predictive power of algorithm 5 with the speed and simplicity of algorithm 4.

The Dudzunski Program

Many people who write successful hi-Q search programs do so on home computers. The Dudzinski program was

| | | of nodes trated | | of nodes anded | CPU time (sec) | | |
|------------------------------|---------|--------------------|---------|-----------------------|-------------------|-----------------------|--|
| | average | standard deviation | average | standard deviation | average | standard deviation | |
| 1. Blind | 372.4 | 489.8 | 280.9 | 498.0 | .240 | .188 | |
| 2. Avoid Corners | 9506.2 | 40204.8 | 9384.7 | 40208.7 | 3.893 | 16.068 | |
| 3. Favor Center Holes | 205.4 | 170.5 | 95.6 | 176.7 | .174 | .073 | |
| 4. Favor Pivotal Holes | 160.4 | 168.3 | 93.5 | 164.4 | .155 | .061 | |
| 5. Maximize Freedom | 214.5 | 44.1 | 26.2 | 23.9 | .310 | .147 | |

Table 1: Five search algorithms are compared on all 21 initial hi-Q states. The first is a blind depth first search to give some basis of comparison for the others. The others are informed depth first searches. The average performance with respect to nodes generated, nodes expanded, and CPU time is given. In each case the standard deviation is given to measure the consistency of the algorithm. Algorithm 5 performs best in terms of number of nodes expanded and is very consistent. Unfortunately its CPU time is high.

written in CDC extended FORTRAN and run on the CYBER 750. The CYBER is a big machine but the program requires very little memory. For example, it allows space φ**i** for only three hundred nodes to exist at a single time. ٠.đ Because of judicious space management, this has proven adequate for solving puzzles which generate over 189,000 nodes. The program has 220 executable FORTRAN statements and uses 9000 storage locations for arrays. In the CYBER the total lead module was 15,249 60-bit words. Upon converting this to a home computing environment, I 細囊 estimate 39,000 bytes as a generous upper bound on space أهرد needs. This will be considerably reduced if the hi-Q board is represented more efficiently.

:::

12

6 **%**

1

1.18

£1

i-A

The terminology of the program is in terms of pennies instead of pegs. Each node has six fields:

a) 5×5 matrix to hold the board configuration (1 = penny,

- 0 = open, 9 = corner).
- b) Open pointer.
- c) Parent pointer.
- d) Closed pointer.
- e) F value, where F is the evaluation function.
- f) Number of pennies on the board.

"Pointer" means the node number of another node. Three hundred nodes are created at load time via six arrays 29**8** corresponding to each of a,...f, above. The FORTRAN (5,5,300), BOARD declaration creates OPENL(300), PARENTL(300), CLOSEDL(300). F(300). and NR 1.4 PENNY(300). Any particular node "straddles" these arrays. 1.1 For example, node 23 has its F value stored in F(23), the number of pennies on its board in NR PENNY(23) and the 80**8** actual board configuration stored in the locations associated with BOARD(, ,23). If node 18 generates node 23, then the value in PARENTL(23) is 18.

> OPEN and CLOSED are maintained 1.**Q** as linked lists; i.e., as nodes chained 1-18 together via the pointer values in fields b and d, above. Removing a 1:18 node from OPEN and putting it on CLOSED means simply changing a 1-3**1** few values in fields b and d. The nodes in OPEN are kept sorted on F 12. value, and lexically where F values 1.1 are tied. Thus the main routine always chooses the first node on 1-1 OPEN for expansion.

To save time, puzzles with more than seven pennies removed are not checked for duplicates on OPEN or CLOSED. This is the only deviation from the algorithm of Figure 4. The program checks for symmetric as well as perfect duplicates for the first three moves and perfect duplicates for the next four.

To conserve memory, the following management device is used. Suppose the previous node expanded has five pennies on the board and the current one has eight. Since the search is depth first, the only way this could happen is for the previous node to have been a dead-end, i.e., no successors. The program is effectively backing up to a node three levels higher. There are no nodes on OPEN with five, six, or seven pennies. Therefore we can "free" the previous node and three generations of its ancestors without danger of eliminating a node involved in an eventual solution. As long as such freedom candidates are below the first seven levels, they are removed from CLOSED and placed back on a list of available nodes. This is how Dudzinski's program is able to solve puzzles requiring the generation of 189,000 nodes while there is only space for three hundred at a time.

Figure 7 shows the subprogram structure. The main routine (listing 1) first initializes the board and various lists by calling INIT. INIT reads the input puzzle and puts the first node on OPEN (see listing 2). PENNY loops through the open list calling PRUNE if the current node has no fewer pennies than the previous one (indicating a dead-end). Nodes are removed from OPEN, then placed on CLOSED if they are no more than seven deep. MOVE is called to generate successor nodes, check for a solution and insert some or all of the successors onto OPEN.

GETNODE and FREE (listing 3) manage the list of available nodes. This list is chained together via the "open pointer field" mentioned earlier. GETNODE provides the caller with the node number of an available node and removes that node from the available list. FREE returns a node to the available list.

EVAL is passed a pointer to a newly created node and returns its F value. The five evaluation functions used are shown in listing 4. PRUNE (listing 5) replenishes the list of available nodes whenever a dead-end is encountered. This

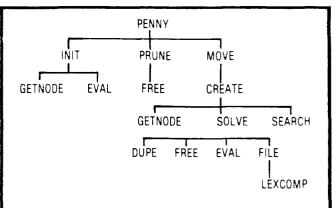


Figure 7: The calling structure of subroutines in the Dudzinski program is shown. PENNY initializes the board (via INIT) then removes nodes from OPEN, placing them on CLOSED as appropriate. MOVE is called periodically to generate successors, check for a solution and insert appropriate nodes onto OPEN. PRUNE frees nodes when dead-ends are encountered. EVAL evaluates f. The other module functions are described in the article.

| | 1. B i | ind search | | 2. Av | 2. Avoid corners | | 3. Fav | or cent | 8 7 | 4. Fa | vor pivots | 1 | 5. Favo | r treedo | m |
|----------|---------------|------------|------|----------|------------------|--------|--------|---------|------------|-------|------------|------|---------|----------|------|
| Initial | nodes | nodes | CPU | nodes | nodes | CPU | nodes | nodes | CPU | nodes | nodes | CPU | nodes | nodes | CPU |
| Open | gen. | exp. | time | gen. | exp. | time | gen. | exp. | time | gen. | exp. | time | gen. | exp. | time |
| Hole | • | | | - | | | - | | | - | | | • | | |
| 1 | 558 | 468 | .321 | 147 | 28 | .150 | 232 | 129 | .192 | 109 | 27 | .128 | 215 | 20 | .279 |
| 2 | 130 | 42 | .139 | 171 | 34 | .160 | 154 | 52 | .157 | 250 | 179 | .172 | 213 | 22 | .288 |
| 3 | 135 | 39 | .145 | 143 | 22 | .140 | 456 | 385 | .282 | 110 | 33 | .132 | 215 | 20 | .287 |
| 4 | 125 | 21 | .141 | 152 | 32 | .161 | 155 | 19 | .154 | 224 | 154 | .185 | 215 | 20 | .277 |
| 5 | 1182 | 1098 | .547 | 189,257 | 189,153 | 75.718 | 107 | 22 | .124 | 247 | 168 | .199 | 201 | 34 | .322 |
| 6 | 148 | 60 | .154 | 877 | 780 | .441 | 165 | 27 | .160 | 86 | 20 | .141 | 214 | 19 | .279 |
| 7 | 125 | 21 | .146 | 162 | 22 | .163 | 108 | 22 | .122 | 79 | 20 | .135 | 185 | 22 | .249 |
| 8 | 1182 | 1098 | .542 | 173 | 33 | .164 | 170 | 22 | .166 | 80 | 19 | .126 | 196 | 21 | .263 |
| 9 | 143 | 60 | .148 | 112 | 19 | .137 | 107 | 19 | .127 | 146 | 73 | .136 | 214 | 19 | .284 |
| 10 | 245 | 149 | .196 | 119 | 23 | .149 | 119 | 28 | .137 | 84 | 20 | .136 | 211 | 22 | 323 |
| 11 | 154 | 63 | .153 | 153 | 29 | .139 | 184 | 30 | 156 | 82 | 20 | .122 | 199 | 19 | .256 |
| 12 | 161 | 58 | .157 | 3,407 | 3,277 | 1.496 | 157 | 19 | .156 | 78 | 20 | .128 | 174 | 19 | .243 |
| 13 | 117 | 19 | .139 | 136 | 22 | .137 | 109 | 19 | .129 | 184 | 104 | .154 | 405 | 132 | .961 |
| 14 | 151 | 63 | .151 | 150 | 29 | .150 | 181 | 30 | .154 | 79 | 20 | .132 | 215 | 20 | .278 |
| 15 | 557 | 468 | .310 | 217 | 91 | .170 | 124 | 28 | .139 | 114 | 52 | .130 | 188 | 22 | .260 |
| 16 | 161 | 58 | .172 | 3,407 | 3,277 | 1.508 | 157 | 19 | .161 | 77 | 20 | .126 | 202 | 19 | .275 |
| 17 | 118 | 19 | .137 | 137 | 22 | .133 | 110 | 19 | .132 | 243 | 174 | .179 | 207 | 20 | .283 |
| 18 | 131 | 32 | .149 | 184 | 31 | .171 | 180 | 109 | .167 | 80 | 19 | .131 | 215 | 20 | .271 |
| 19 | 132 | 62 | .148 | 131 | 20 | .150 | 147 | 19 | .145 | 79 | 19 | .128 | 196 | 21 | .271 |
| 20 | 2042 | 1981 | .884 | 134 | 19 | .131 | 881 | 786 | .459 | 860 | 782 | .411 | 210 | 19 | .272 |
| 21 | 124 | 19 | .159 | 261 | 115 | .189 | 311 | 205 | .227 | 77 | 20 | .124 | 215 | 20 | .282 |
| max | 2042 | 1981 | .884 | 189,257 | 189,153 | 75.718 | 881 | 786 | .459 | 860 | 782 | .411 | 405 | 132 | .961 |
| Min | 117 | 19 | .137 | 112 | 19 | .131 | 107 | 19 | .122 | 77 | 19 | .122 | 174 | 19 | .243 |
| Average | 372.4 | 280.9 | .240 | 9,506.2 | 9,384.7 | 3.893 | 205.4 | 95.6 | .174 | 160.4 | 93.5 | .155 | 214.5 | 26.2 | .310 |
| st. dev. | 489.8 | 498.0 | .188 | 40,204.8 | 40,208.7 | 16.068 | 170.5 | 176.7 | .073 | 168.3 | 164.4 | .061 | 44.1 | 23.9 | .147 |

 Table 2: This shows the performance of all five algorithms on all 21 initial states with respect to number of nodes generated, nodes expanded and CPU time. Minimum, maximum, average and standard deviation figures are also given. The hole numbers in column 1 refer to Figure 1.

 We see, for example, that algorithm 5 usually scores in the low 20's for nodes expanded, going almost directly to the solution. Algorithm 2 expanded 189,153 nodes for initial state 5 but only 22 for its symmetric duplicate, state 17.

was described earlier.

MOVE (listing 6) makes all possible next moves and for each one calls CREATE (listing 7) to generate the successor, check it for goal status and put it on OPEN, if appropriate. When a goal is found, SOLVED (listing 8) is called to print out the solution along with the number of nodes generated and expanded to reach it. The solution is a display of twenty successive board configurations from start to goal.

Newly created non-goal nodes are passed by CREATE to SEARCH (listing 9). If a node is no more than seven deep, SEARCH calls DUPE to compare it with nodes already on OPEN and CLOSED. DUPE checks for symetric duplication on the first three levels and perfect duplication thereafter (see listing 10). If a duplication is found, then SEARCH frees the node. Otherwise it is inserted into OPEN by FILE (listing 11), lowest F values first and ties broken lexically. LEXCOMP (listing 12) compares game states and tells which is lexically smallest.

The names and uses of all global variables are tabulated in listing 1.

Conclusions

Ed's results suggest how humans can play a better hi-Q game. I can't do the computations for algorithm 5 in my head very fast, but I do find that algorithm 4 improves my game. Ed's results also indicate that the best computer search strategy for hi-Q has yet to be found. Among the ideas that have not been thoroughly explored are these: a) weight the board squares in a graduated fashion that is more subtle than any of algorithms 2, 3, or 4; b) reward and punish certain clustering patterns like three colinear pegs or density around center of gravity, that are independent of particular board squares; c) change the strategy according to puzzle depth; d) combine strategies in a weighted fashion and then look for the best weights.

To what extent is it cost-effective to check for duplicates on OPEN and CLOSED? I don't know. Perhaps a good heuristic goes so directly to the solution that there is little need to worry about duplicates.

There is another unanswered question: What is the size of the hi-Q space?

There are several harder versions of the puzzle that people often prefer. Consider using pennies instead of pegs. Replace one of the pennies with a nickel and demand that the nickel be the last coin on the board. Harder still, demand that the nickel be left in the center. Another variation: use all pennies and demand that the last penny be in the center. Joel Arnold wrote a very nice search program to solve this problem. He worked backwards, depth first, from the goal and found the following heuristics effective: a) down to an intermediate depth, concentrate on filling the perimeter of the puzzle; b) in later stages give orthogonal moves precedence over diagonal moves; c) up to the last move, filling the goal spot is rewarded and emptying it is penalized. (The last move must empty it.) Another ruse: he sought any state symmetric to the desired goal. If the found state was not the goal, transformations were made on the sequence of moves, yielding a true solution. I'm not sure that this is "fair." It does improve average CPU time, and,

unsurprisingly, the number of nodes generated.

The last point raises the question of what is a fair solution. In some sense we want generality because this is what a human who plays the puzzle uses. Any program which has 21 different procedures for each of the 21 start states would have to be rejected. There are at least two ways one could enforce generality and toughen the problem a bit. 1) Demand that the program perform well when given a partially solved puzzle; thus instead of 21 start states, there are thousands. 2) Don't tell the program until input time what the exact board shape is. It might be told upper bounds on size, but no more.

I hope some of you have as much fun with computer hi-Q as have my students and I.

13

白機

1.4

ist

1.1

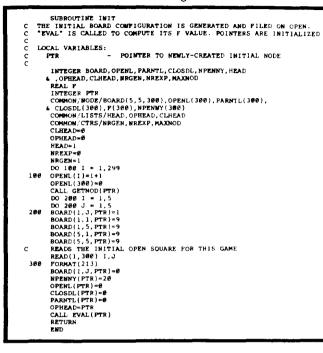
Editor's Note: The following program listings in FORTRAN are slight modifications of those supplied by the author A few changes were necessary in order to be compatible with plain vanilla FORTRAN. These changes involved substituting size-character carable names for some that were seven characters long, removing some extended assignment statements (eg. A = B = C = 0), substituting single quotes for double quotes surrounding literal constants and removing message strings from STOP statements. Even so, the changes required were far fewer than converting one dialect of BASIC to another, which illustrates the advantage of programming in a reasinably standardized language.

We ran the programs listed here with Microsoft FORTRAN-80 under (P:M and obtained the same results listed by the author in Table (x), the only difference being that the CPU times were about two orders of magnitude greater. Lance Rose Technical Editor.

Listing 1

| | DOCCLAM DOC | rv | |
|---|---|--|-----------------------|
| C TH | PROGRAM PENN | IY IE INITIALIZES VARIABLES AND LOOPS THROUGH | \$24 |
| | | WHICH IS ORDERED BY P-VALUE. SUBROUTINE | |
| | | TO GENERATE SUCCESSORS TO THE CURRENT NODE. | |
| c | | | |
| | OBAL VARIABLE | | ¥ C |
| С | COMMON NODE | | |
| с | BOARD | - THE 5 X 5 PLAYING BOARD (CORNERS ARE UNUSED) | |
| c | OPENL | - LINKS NODES ON THE OPEN LIST, WHICH CONTAINS ALL UNCHECKED GAME STATES: THIS POINTER IS ALSO | 450 |
| C C | | USED TO LINK THE LIST OF AVAILABLE NODES | |
| Ċ | PARNTL | - LINKS A NODE TO ITS PARENT, WHICH IS A BOARD | |
| è | THUR DE | CONFIGURATION ONE MOVE PREVIOUS TO ITSELF | 8. |
| c | CLOSDL | - LINKS NODES ON THE CLOSEC LIST FOR DUPLICATE | |
| с | | CHECKING; THE BOARD CONFIGURATIONS IN THE CLOSED | |
| с | | LIST HAVE ALREADY BEEN EXPANDED | |
| С | F | - THIS IS THE VALUE OF THE EVALUATION FUNCTION WHICH | 49 |
| C | | ATTEMPTS TO PREDICT THE PROBABILITY OF A SOLUTION | |
| С | | IN A NODE'S SUCCESSORS | |
| C C | NPENNY | - THE NUMBER OF PENNIES REMAINING ON THE BOARD - HEADS OF THE THREE LISTS OF NODES | ş. |
| | COMMON LISTS | - HEAD OF THE LIST OF AVAILABLE MODES | |
| | OPHEAD | - HEAD OF THE LIST OF AVAILABLE BODES | |
| | CLHEAD | - HEAD OF THE LIST OF CLOSED NODES | i: |
| | COMMON CTRS | - COUNTERS USED TO TRACK THE EFFECTIVENESS OF THE | |
| с | | PROGRAM HEURISTICS AD MECHANICS | |
| | NRGEN | - THE NUMBER OF MODES GENERATED | |
| | NREXP | - THE NUMBER OF NODES EXPANDED | 4 |
| | MAXNOD | - THE MAXIMUM NUMBER OF NODE DATA AREAS IN USE AT | |
| C LC | | ANY TIME | |
| | CAL VARIABLES CURENT | - POINTER TO THE NODE BEING EXAMINED FOR POSSIBLE | ŀ |
| c | CORNEL | EXPANSION | |
| | HLDPTR | - POINTER OT THE NODE EXPANDED PREVIOUS TO THE | |
| c | | CURRENT NODE | , |
| с | PASSI | - LOGICAL SWITCH TO SKIP PRUNING TEST ON FIRST PASS | |
| с | | | |
| | | D, OPENL, PARNTL, CLOSDL, NPENNY, HEAD | |
| | | EAD, NRGEN, NREXP, MAXIOD | 1 |
| | REAL P INTEGER CURR | NT HLDPTR | |
| | LOGICAL PASS | | |
| | COMMON / NODE / | BOARD(5,5,300), OPENL(300), PARNTL(300), | |
| | | | , |
| | E CLOSDL(JOF/ | , F(SOD), RFLAMI(SOD) | , |
| | COMMON/LISTS | , P (300), NPENNY (300) :/Head, Ophead, Clinead | , |
| | COMMON/LISTS COMMON. CTRS/ | :/HEAD, OPHEAD, CLHEAD NRGEN, NREXP, MAXNOD | |
| | COMMON/LISTS COMMON.CTRS/ DATA PASS1/ | :/HEAD, OPHEAD, CLHEAD NRGEN, NREXP, MAXNOD | |
| | COMMON/LISTS COMMON/CTRS/ DATA PASS1/- CALL INIT | /HEAD, OPBEAD, CLNEAD NRGEN, NREXP, MAXNOD TRUE. / | |
| | COMMON/LISTS COMMON/CTRS/ DATA PASS1/- CALL INIT HLDPTR-OPHEA | /HEAD,OPBEAD,CLHEAD NRGEN,NREXP,MAXNOD TRUE./ | 1 |
| 100 | COMMON/LISTS COMMON.CTRS/ DATA PASS1/. CALL INIT HLDPTR-OPHEA IF(OPHEAD.EC | (HEAD, OPBEAD, CLAEAD NRGEN, NREXP, MAXNOD TRUE. / .0) GO TO 986 | 1 |
| | COMMON/LISTS COMMON.CTRS/ DATA PASS1/- CALL INIT HLDPTR-OPHEA IF(OPHEAD.EC CURRNT-OPHEA | (HEAD, OPBEAD, CLAEAD NRGEN, NREXP, MAXNOD TRUE. / .0 .0 GO TO 966 .0 | 1 |
| 160 | COMMON/LISTS COMMON.CTRS/ DATA PASS1/- CALL INIT HLDPTR=OPHEA IF(OPHEAD.EC CURRNT=OPHEA OPHEAD=OPENL IF(PASS1) GC | (HEAD, OPBEAD, CLATEAD NRGEN, NREXP, MAXNOD TRUE. / .0 .0 GO TO 966 .0 (CURNAT) .70 286 | 1 |
| 1 00 c 1P | COMMON/LISTS COMMON.CTRS/ DATA PASS1/. CALL INIT HLDPTR-OPHEA IP(OPHEAD-EC CURRNT=OPHEA OPHEAD=OPENL IP(PASS1) GO THE SEARCH I | (HEAD, OPBEAD, CLAEAD NRGEN, NREXP, MAXNOD TRUE./ D (CURRAT) TO 200 NO LONGER GOING DOWN INTO THE STATE SPACE, |) |
| 160 C 1P C TH | COMMON/LISTS COMMON.CTRS/ DATA PASSI/- CALL INIT HLDPTR=OPHEA IF(OPHEAD=CO CURRNT=OPHEA OPHEAD=OPENL IF(PASSI) GO THE SEARCH I EN WE HAVE RE | (HEAD, OPBEAD, CLHEAD NRGEN, NREXP, MAXNOD TRUE./ D .0) GO TO 986 (URRHT) TO 286 5 NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS | |
| C IP C TH C OP | COMMON/LISTS COMMON.CTRS/ DATA PASSI/. CALL INIT HLDPTR-OPHEAD.EC CURRT-OPHEAD.EC CURRT-OPHEAD-OPENL IF(PASSI) GC THE SEARCH I EN WE HAVE RE DEPTH-FIRST) | (HEAD, OPBEAD, CLATEAD NRGEN, NREXP, MAXNOD TRUE./ .0 .0 .0 .0 .0 .0 .0 .0 .0 .0 |) |
| C IP C TH C OP | COMMON/LISTS COMMON.CTRS/ DATA PASS1/- CALL INIT HLDPTR-OPHEA CURRNT-OPHEA OPHEAD-DEC URRNT-OPHEA OPHEAD-OPENL IF(PASS1) GO THE SEARCH I EN WE HAVE RE DEPTH-FIRST) UMED FROM THE | (HEAD, OPBEAD, CLIEAD NRGEN, NREXP, MAXNOD TRUE./ D .00 .00 .00 .00 .00 .00 .00 .00 .00 | |
| C IP C TH C OP | COMMON/LISTS COMMON.CTRS/ DATA PASS// CALL INIT HLDPTR-OPHEAD.EC CURRTT-OPHEAD.EC CURRTT-OPHEAD.EC CURRTT-OPHEAD.EC CURRTT-OPHEAD.EC CURRTT-OPHEAD.EC CURTT-DIAL IP(NPENDY(CU IP(NPENDY(CU | (HEAD, OPBEAD, CLATEAD NRGEN, NREXP, MAXNOD TRUE./ .0 .0 .0 .0 .0 .0 .0 .0 .0 .0 | |
| C IP C TH C OP C PR | COMMON/LISTS COMMON.CTRS/ DATA PASS1/. CALL INIT HLDPTR-OPHEAD.EC CURRNT-OPHEAD-OPENL OPHEAD-OPENL OPHEAD-OPENL IP(PASS1) GC THE SEARCH I EN WE HAVE RE DEPTH-FIRST) UNED FROM THE IP(NPENNY(CU 6 CALL PRUME | (HEAD, OPBEAD, CLIEAD NRGEN, NEXP, MAXNOD TRUE./ D .0) GO TO 986 D (CURAUT) TO 286 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITEMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH TREE TO PREE STORAGE. RNMT).GE.WPENNY(CHAPTH)) (HLDPTR, NUPENNY(CHAPTH)) | |
| C IP C TH C OP C PR C SI | COMMON/LISTS COMMON.CTRS/ DATA PASSI/- CALL INIT HLDPTR-OPHEAD -EC CURRTH-OPHEAD -EC CURRTH-OPHEAD -OPHEAD IP(PASSI) GC THE SEARCH I EN WE HAVE RE DEPTH-FIRST) UNED FROM THE DEPTH-FIRST) UNED FROM THE CALL PRUME MCC ONLY BOAR | (HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ D .0) GO TO 900 (CURNET) TO 200 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PENNIES ARE SEARCH TREE TO FREE STORAGE. RINT).GC. MPENNY(RLDPTR)) (HLDPTR.NPENNY(CURNET)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED | |
| C IP C TH C OP C PR C SI C FO | COMMON/LISTS COMMON.CTRS/ DATA PASSI/- CALL INIT HLDFTR-OFERL DFHEAD-DEFAL IF(OPHEAD-DEFAL IF(OPHEAD-OFERL IF(OPHEAD-OFERL IF(OPHEAD-COFENC IF(OPHEAD-COFENC) DEFTH-FIRST) USED FROM THE IF(UPENFY(CC) CALL PRUME MCC ONLY BOAR MC DUPLICATES, | (HEAD, OPBEAD, CLIEAD NRGEN, NEXP, MAXNOD TRUE./ D .0) CO 0 986 D (CURRINT) 10 286 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITENS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH TREE TO PREE STORAGE. RINT).CE.WPENNY(CLIPTR)) (HLOPTR. NPENNY(CLIPTR)) (HLOPTR.NPENNY(CLIPTR)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE MOT SAVED | |
| C IP C TH C OP C PR C SI C FO | COMMON/LISTS COMMON.CTRS/ DATA PASSI/. CALL INIT HLDPTR-OFBEA CURRNT-OFBEA OFMEAD-OFBEA CURRNT-OFBEA OFMEAD-OFBEA IF(MFEAD-OFBEA IF(MFEAD-OFBEA) BEN ME HAVE RE DEPTH-FIRST) IF(MFEAD-OFBEA IF(MFEAD-OFBEA) IF(MFEAD-OFBEAD-OFBEAD IF(MFEAD-OFBEAD-OFBEAD IF(M | (HEAD, OPBEAD, CLIEAD NRGEN, NEXP, MAXNOD TRUE./ D .0) CO 0 986 D (CURRINT) 10 286 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITENS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH TREE TO PREE STORAGE. RINT).CE.WPENNY(CLIPTR)) (HLOPTR. NPENNY(CLIPTR)) (HLOPTR.NPENNY(CLIPTR)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE MOT SAVED | |
| C IP C TH C OP C PR C SI C PO C ON | COMMON/LISTS COMMON.CTRS/ DATA PASSI/. CALL INIT HLDPTR-OFBEA CURRNT-OFBEA CURRNT-OFBEA OFMEAD-OFBEA IF(AFSI) GC THE SEARCH I IF(MFENDY(CU & CALL PRUBE NEC ONLY BOAR R DUPLICATES, MEC ONLY BOAR | (HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXONOD TRUE./ D (CURNET) TO 286 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITEMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PENNIES ARE SEARCH THEE TO FREE STORAGE. RANT J.GE. NPENNY (CLURNTI) (HLDFTR, NPENNY (CURNTI)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED MODES LOWER IN THE STATE SPACE ARE NOT SAVED IST. RNT J.T. 13) GO TO 580 | 1 |
| 180 C 1P C TH C 0P C PR C SI C PO C 00 200 | COMMON/LISTS COMMON.CTRS/ DATA PASSI/ CALL INIT HLDPTA-OFHEA UP(DPHEAD-EC UURNT-OFHEAD-OFEL IP(DPSSI) CO THE SEARCH I P(DPSSI) CO THE SEARCH I P(DPENT/CU DEPTH-FIRST) UNED FROM THE MCC ONLY BOAR R DOPLICATES, THE CLOSED L IP(WPENT(CU CLOSED L(CURNT) CLERAD-CURNA | (HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ .0 .0 .0 .0 .0 .0 .0 .0 .0 .0 | 1 |
| C IP C TH C OP C PR C SI C PO C ON | COMMON/LISTS COMMON.CTRS/ DATA PASSI/. CALL INIT HLDFTR-OFHEA OFHEAD-OFELN OFHEAD-OFELN IF(DASSI) GC THE SEARCH I IF(DASSI) GC THE SEARCH I IF(DASSI) GC THE SEARCH I IF(DASSI) GC THE SEARCH I IF(DASSI) GC ONLY BOAM R DUPLICATES, THE CLOSED L IF(UPEDWR(CC CLOSED (CURAN CLAELAD-CURAN CLAELAD-CURAN | <pre>(HEAD, OPBEAD, CLIEAD NRGEN, NEXP, MAXNOD TRUE./ D .0) GO TO 986 (CURRANT) TO 286 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH TREE TO PREE STORAGE. RNNT).GE.WPENNY(CURDANT)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED HODES LOWER IN THE STATE SPACE ARE NOT SAVED 1ST. RNNT).LT.13) GO TO 586 T T OCLIEAD T RNNT)</pre> | |
| 180 C 1P C TH C 0P C PR C SI C PO C 00 200 | COMMON/LISTS COMMON.CTRS/ DATA PASSI/ CALL INIT HLDPTR-OPHEAN IF(OPHEAD-GPEL) CURRNT-OPHEAD-OPELI IF(PASSI) GC THE SEARCH I IF(PASSI) GC THE SEARCH I IF(NPENNY(CU & CALL PRUBE MCC ONLY BOAR R DUPLICATES, MCC ONLY BOAR R DUPLICATES, IF(COSED L) IF(NPENNY(CU CLOSED L) CLOSED L) CLOSED L) CHEAD-CURRN CALL NOVE(CU | <pre>(HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ D 0) GO TO 900 (CURNET) TO 200 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH THEE TO FREE STORGE. RINT).G. UPBENNY(RLDPTR)) (HLDFT, NPENNY(CURNET)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE NOT SAVED 1ST. RENT).LT.13) GO TO 500 T) CLIMEAD T RENT)</pre> | |
| 180 C 1P C TH C 0P C PR C SI C PO C 00 200 | COMMON/LISTS COMMON.CTRS/ DATA PASSIA/ CALL INIT HLDFTR-OPHERD IF(OPHERD-EEC UURNT-OPHERD-OPHERD IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASI) CC UNED FROM THE IF(PENTY(CC CLOSDL(CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM | <pre>(HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ D 0) GO TO 900 (CURNET) TO 200 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH THEE TO FREE STORGE. RINT).G. UPBENNY(RLDPTR)) (HLDFT, NPENNY(CURNET)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE NOT SAVED 1ST. RENT).LT.13) GO TO 500 T) CLIMEAD T RENT)</pre> | |
| 100 C 1P C TH C PR C PR C PR C SI C PO C ON 200 500 | COMMON/LISTS COMMON.CTRS/ DATA PASSI/. CALL 1NIT HLDPTR-OPHEAD CURRNT-OPHEAD DPHEAD-OPELN CURRNT-OPHEAD-OPELN IF(PASSI) GC THE SEARCH I F(PASSI) GC THE SEARCH I F(PASSI) GE NE HAVE RE DEPTH-FIRST) UNED FROM THE ELOSED L IF(NPENNY(CU CLOEDI.CURRN CLEEND-CURRN CALL PNUE CLOEDI.CURRN CLEEND-CURRN CALL NOVE(CU HLDPTR-CURN PASSI-FIALS | <pre>(HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ D 0) GO TO 900 (CURNET) TO 200 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH THEE TO FREE STORGE. RINT).G. UPBENNY(RLDPTR)) (HLDFT, NPENNY(CURNET)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE NOT SAVED 1ST. RENT).LT.13) GO TO 500 T) CLIMEAD T RENT)</pre> | , , , , , |
| 180 C 1P C TH C 0P C PR C SI C PO C 00 200 | COMMON/LISTS COMMON.CTRS/ DATA PASSIA/ CALL INIT HLDFTR-OPHERD IF(OPHERD-EEC UURNT-OPHERD-OPHERD IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASSI) CC THE SEARCH I IF(PASI) CC UNED FROM THE IF(PENTY(CC CLOSDL(CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM CLASERD-CURRM | <pre>(HEAD, OPBEAD, CLAEAD NRGEN, NEEXP, MAXNOD TRUE./ D 0) GO TO 900 (CURNET) TO 200 S NO LONGER GOING DOWN INTO THE STATE SPACE, ACHED A DEADEND (ALL ALGORITHMS ARE VARIATIONS AND NODES WITH FEWER THAN 13 PERMIES ARE SEARCH THEE TO FREE STORGE. RINT).G. UPBENNY(RLDPTR)) (HLDFT, NPENNY(CURNET)) DS GENERATED BY THE FIRST 7 MOVES ARE CHECKED NODES LOWER IN THE STATE SPACE ARE NOT SAVED 1ST. RENT).LT.13) GO TO 500 T) CLIMEAD T RENT)</pre> | • |

Listing 2

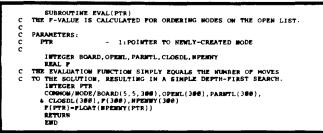


Listing 3

| | SUBROUTINE FREE(PTR) |
|---|---|
| с | DATA AREAS NO LONGER NEEDED ARE RETURNED TO THE LIST OF AVAILABLE |
| č | NODES. |
| č | |
| | PARAMETERS : |
| c | PTR - 1/0:POINTER TO USED NODE BEING RETURNED TO THE |
| č | LIST OF AVAILABLE NODES |
| č | |
| C | INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY, HEAD |
| | . OPHEAD, CLHEAD |
| | REAL F |
| | INTEGER PTR |
| | COMMON/NODE/BOARD(5, 5, 300), OPENL(300), PARNTL(300), |
| | 6 CLOSDL (300), P (300), NPENNY (300) |
| | COMMON/LISTS/HEAD, OPHEAD, CLHEAD |
| | PARNTL (PTR)=0 |
| | CLOSDL(PTR)=0 |
| | NPENNY (PTR)=0 |
| | F(PTR)=0. |
| | OPENL (PTR)=HEAD |
| | HEADEPTR |
| | PTR=0 |
| | RETURN |
| | END |
| | |
| | |
| | SUBROUTINE GETNOD(PTR) |
| с | AVAILABLE DATA AREAS FOR NEW NODES ARE RETURNED |
| с | TO THE CALLING ROUTINES. |
| с | |
| С | PARAMETERS: |
| с | PTR - G: POINTER TO AVAILABLE BODE DATA AREA |
| с | |
| | INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY, HEAD |
| | 6 , OPHEAD, CLHEAD, NRGEN, NREXP, MAXMOD |
| | REAL F |
| | INTEGER PTR |
| | COMMON/NODE/BOARD(5,5,366), OPENL(366), PARNTL(366), |
| | 5 CLOSDL (300), P (300), NPENNY (300) |
| | COMMON/LISTS/HEAD, OPHEAD, CLHEAD |
| | COMMON/CTRS/HRGEN, HREXP, MAXNOD |
| | DATA MAXNOD/0/ IP(HEAD.EO.300) STOP 2 |
| | PTR=HEAD |
| | |
| | IP(PTR.GT.HAXHOD) MAXHOD=PTR |
| | HEAD=OPENL (PTR) |
| | return End |
| _ | |
| | |

Listing 4: The five elvaluation functions sampled are shown.

Listing 4a



Listing 4h

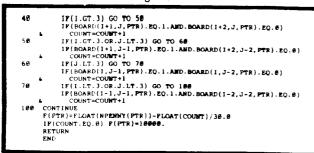
| Listing 4b |
|--|
| SUBROUTINE EVAL(PTR) C THE F-VALUE IS CALCULATED FOR ORDERING NODES ON THE GPEN LIST. C C PARAMETERS: C PTR - I:POINTER TO NEWLY-CREATED NODE C LOCAL VARIABLES: |
| C CORNER - THE NUMBER OF CORNER PENNIES C INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY |
| REAL F C BOARD COMPIGURATIONS WITH PEWEST CORNER PEMNIES ARE FAVORED. INTEGEP FTR_CORNER COMMON/NODE/BOARD(5,5,300),OPENL(300),PARNTL(300), & CLOSDL(300),P(300),NPENNY(300) CORNER=0 |
| IP(BOARD(1,2,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(1,4,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(2,1,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(2,5,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(4,5,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(5,2,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(5,2,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(5,2,PTR).EO.1) CORMER=CORMER+1 IP(BOARD(5,4,PTR).EO.1) CORMER=CORMER+1 IP(PTR)=PLOAT(MPENNY(PTR))+PLOAT(CORMER)/10. RETURM END |
| Listing 4c |
| SUBROUTINE EVAL(PTR) C THE P-VALUE IS CALCULATED FOR ORDERING NODES ON THE OPEN LIST. |
| C C PARAMETERS: - I:POINTER TO NEWLY-CREATED NODE C LOCAL VARIABLES: C COUNT - THE NUMBER OF PENNIES IN THE 9 CENTRAL LOCATIONS |
| C INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY |
| REAL F C BOARD CONFIGURATIONS WITH PENNIES IN 9 CENTRAL LOCATIONS C ARE FAVOREL. INTEGER PTR,COUNT COMMON/MODE/BOARD(5,5,300),OPENL(300),PARNTL(300), & CLOSDL(300),F(300),NPENNY(300) |
| COUNT=0 DO 106 J = 2,4 DO 106 J = 2,4 106 J (0ARD(1,J,PTR).EQ.1) COUNT=COUNT+1 P(PTR)=FLOAT(NPENNY(PTR))=FLOAT(COUNT)/10.0 RETURN END |
| Listing 4d |
| SUBROUTINE EVAL(PTR) C THE F-VALUE IS CALCULATED FOR ORDERING NODES ON THE OPEN LIST. C DEALWERERS |
| C PARAMETERS: C PTR - I:POINTER TO NEWLY-CREATED WODE C LOCAL VARIABLES: |
| C COUNT - THE NUMBER OF PENNIES IN THE 4 PIVOTAL LOCATIONS C INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY |
| REAL P C BOARD CONFIGURATIONS WITH PERMILES IN 4 PIVOTAL LOCATIONS C ARE PAVORED. |
| CONNEL PRODECT INTECER PTR.COUNT CONNEW/NODE/BOARD(5,5,300),OPENL(300),PARNTL(300), L CLOSDL(300),P(300),NPENNY(300) COUNT-0 |
| IF(BOARD(2, 3, PTR).E0.1) COUNT-COUNT+1 IF(BOARD(3,2, PTR).E0.1) COUNT-COUNT+1 IF(BOARD(3,4, PTR).E0.1) COUNT-COUNT+1 IF(BOARD(4,3, PTR).E0.1) COUNT-COUNT+1 F(PTR)-FLOAT(WPENNY(PTR))-FLOAT(COUNT)/5.0 RETURM END |
| Listing 4e |
| SUBROUTINE EVAL(PTR) |
| C THE F-VALUE IS CALCULATED FOR ORDERING NODES ON THE OPEN LIST. C C parameters: |
| C PTR - I:POINTER TO NEWLY-CREATED MODE C Local Variables: C Court - The Number of Successors of The Passed Mode |
| C Integer Board, Openil, Parifil, Closdl, Npenny |
| REAL P C THIS EVALUATION FUNCTION REDUCES THE ACTUAL P VALUE (HOVES TO C SOLUTION) BY A FRACTION PROPORTIONAL TO THE SUMBER OF CHILDREN C OF THE MODE, THUS FAVORING THOSE BOARD CONFIGURATIONS WITH THE C GREATEST NUMBER OF SUCCESSON MODES. BY CONVERTING THE SUMBER C OF CHILDREN INTO A FRACTION LESS TRAN ONE (THE MAXIMUM SUMBER OF |

c c

CF THE NODE, THUS FAVORING THOSE BOARD COMFIGURATIONS WITH THE GREATEST NUMBER OF SUCCESSON MODES. BY CONVERTING THE UNBER OF CHILDREN INTO A FRACTION LESS THAN ONE (THE MAXIMUM HUMBER OF SUCCESSON NODES IS < 30), WE RETAIN A DEPTH-FIRST SHARCH. INTEGER PTR, COMPT COMPAND (5,5,506), OFERL(300), PARMITL(366), 4 CLOSDL(306), P(300), MPERMY(360) COUNT-6 DO 186 J = 1,5 IF(BOARD(1,J,FTR).ED.1) GO TO 196 IF(1.LT.3) GO TO 18 IF(1.LT.3) GO TO 18 IF(1.LT.3) GO TO 18 IF(1.LT.3) GO TO 18 IF(1.LT.3) GO TO 28 IF(1.LT.3) GO TO 38 IF(1.LT.3) GO TO 38 IF(1.LT.3) GO TO 38 IF(1.GC.3) GO TO 38 IF(1.GC.3) GO TO 38 IF(1.GC.3) GO TO 38 IF(1.GC.3) GO TO 48 IF(1.GC.3) GO,J.GC.3) GO TO 48 IF(1.GC.3) COUNT+1 COUNT-COUNT+1 COUNT-CO

- 18
- 20
- 38 continued

Listing 4e, continued



Listing 5

| SUBROUTINE PRUNE(PTR.HILEV) |
|---|
| C NODES WITH FEWER THAN 13 PENNIES BEGINNING WITH THE PASSED NODE |
| C AND BACK THROUGH ITS PARENT NODES TO THE PARENT WITH "HILEY" |
| C NUMBER OF PENNIES ARE RETURNED TO THE LIST OF AVAILABLE NODES. |
| C NODES WITH 13 OF MORE PENNIES REMAIN ON CLOSED FOR DUPLICATE COM- |
| C PARISONS. "HILEV" IS THE NUMBER OF PENNIES ON THE NODE CURRENTLY |
| C BEING EXPANDED, WHICH IS WHERE OUR SEARCH HAS BACKED UP TO APTER |
| C REACHING A DEAD END. BY PRUNING ONLY TO THE LEVEL OF THE CURRENT |
| C NODE, WE DON'T CHANCE ELIMINATING PARENT NODES OF AN EVENTUAL |
| C SOLUTION, BUT WE WILL ELIMINATE ALL DEADEND BRANCHES BELOW THE |
| C 13 PENNY LEVEL. |
| |
| PARAMETERS: |
| C PTR - 1/0: POINTER TO NODE EXAMINED FOR POSSIBLE EXPAN- |
| C SION JUST BEFORE THE NODE BEING CURRENTLY EXAM- |
| C INED; THIS NODE COULD NOT BE EXPANDED; IF THIS |
| C NODE IS PRUNED, PTR IS SET TO ZERO |
| C HILEV - 1: THE NUMBER OF PENNIES ON THE GAME BOARD OF THE |
| C NODE CURRENTLY BEING EXAMINED FOR POSSIBLE |
| C EXPANSION |
| C LOCAL VARIABLES: |
| C PARENT - POINTER TO PARENT NODE |
| C HLDPTR - POINTER USED TO TRACE FROM THE PASSED NODE BACK |
| C THROUGH ITS PARENTS |
| c |
| INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY |
| REAL P |
| INTEGER PTR, HILEV, PARENT, HLDPTK |
| COMMON/NODE/BOARD(5,5,302), OPENL(300), PARNTL(300), |
| L CLOSDL(300), F(300), NPENNY(300) |
| IF (NPENNY (PTR). LT. HILEV) GO TO 100 |
| IF(WPENNY(PTR).GE.13) RETURN |
| CALL PREE(PTR) |
| RETURN |
| 100 HLDPTR=PTR |
| PTR=0 |
| 200 PARENT=PARNTL(HLDPTR) |
| IF(NPENNY(HLDPTR).GE.13) RETURN |
| CALL FREE(HLDPTR) |
| IP(NPENNY(PARENT).CT.HILEV) RETURN |
| HLDPTR=PARENT |
| GO TO 200 |
| END |

Listing 6

| | SUBROUTINE MOVE (PTR) |
|------|--|
| | LL POSSIBLE NEXT MOVES ARE TAKEN, GENERATING EVERY SUCCESSOR |
| | D THE CURRENT MODE. |
| с | |
| | NRAMETERS: |
| 2 | PTR - I: POINTER TO NODE BEING EXPANDED |
| с | |
| | INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY, NRGEN |
| | L , BREXP, MAXNOD |
| | REAL P |
| | INTEGER PTR |
| | CONMON/MODE/BOARD(5,5,300),OPENL(350),PARNTL(360), |
| | & CLOSDL(368), F(386), HPENNY(368) |
| | CONNON/CTRS/BRGEN, BREXP, MAXINOD |
| | NREXP=NREXP+1 |
| | DO 188 I = 1.5 |
| | DO 168 J = 1,5 |
| | IF(BOARD(1, J, PTR).WE.1) GO TO 188 |
| | IF(1.LT.3) GO TO 18 |
| | IF (BOARD (I-1, J, PTR). EQ. 1. AND. BOARD (I-2, J, PTR). EQ. 6) |
| | E CALL CREATE(PTR. 1. J. I-1. J. I-2. J) |
| 10 | IF(1.LT.3.OR.J.GT.3) GO TO 28 |
| | IF(BOARD(I-1, J+1, PTR).E0.1.AND.BOARD(I-2, J+2, PTR).E0.8) |
| | 4 CALL CREATE(PTR. 1, J, 1-1, J+1, 1-2, J+2) |
| 20 | IF(J.GT.3) GO TO 38 |
| •• | IF (BOARD(1, J+1, PTR).EQ.1.AND.BOARD(1, J+2, PTR).EQ.0) |
| | 4 CALL CREATE (PTB. 1, J, 1, J+1, 1, J+2) |
| 30 | IP(I.GT. 3.OR.J.GT. 3) GO TO 48 |
| | IF (BOARD (1+1, J+1, PTR), EO. 1, AND, BOARD (1+2, J+2, PTR), EO. 8) |
| | CALL CREATE(PTR.1.J.1+1.J+1.1+2.J+2) |
| 40 | IP(1.GT.3) GO TO 50 |
| | IF (BOARD (1+1. J. PTR), EQ. 1. AND. BOARD (1+2. J. PTR). EQ. 6) |
| | 6 CALL CREATE (PTR, I, J, I+1, J, 1+2, J) |
| 50 | IF(1.GT.3.OR.J.LT.3) GO TO 68 |
| | IF (BOARD (1+1, J-1, PTR), EO, 1, AND, BOARD (1+2, J-2, PTR), EO, 0) |
| | 6 CALL CREATE(PTR. I.J. I+1. J-1. I+2. J-2) |
| 68 | IF(J.LT.3) GO TO 70 |
| 96 | $IP(J,LT,J) \in U \cap U$ IP(BOARD(1, J-1, PTR), EQ, 1, AND, BOARD(1, J-2, PTR), EQ, 0) |
| | $ = \frac{1}{2} \left(\frac{1}{2} - \frac{1}{2}, \frac{1}$ |
| 78 | IF(1.LT.3.OR.J.LT.3) GO TO 188 |
| 14 | IF(1.LT.3.OR.3.LT.3) GO TO 100 IF(BOARD(1-1, J-1, PTR).EQ.1.AMD.BOARD(1-2, J-2, PTR).EQ.6) |
| | E CALL CREATE(PTR. I. J. I-1. J-1. I-2. J-2) |
| 1.00 | |
| 190 | RETURN |
| | |
| | |

Listing 7

1 🧌

1#

主播

1-18

丰貴

.....

科制

1.10

-84

1. đ

18**8**) e.#

- 24 4.3**Ú**

杨确 ,-*i*i

に種 1.4

1-46 أو رو

13

1.00

.è-∰ 1.1

标囊 1.1

нŋ 1.1

16

.....

14

فغرر

---wał

1:8

18

continued

| | SUBROUTINE CREATE(PTR, L, M, N, O, P, Q) |
|-----|--|
| с т | HE PARENT NODE IS COPIED INTO A NEW DATA AREA, THE IN ATED |
| C M | OVE IS MADE, AND POINTERS ARE SET. IF THE GENERATED ARD |
| C 1 | S A SOLUTION, IT IS PRINTED OUT AND EXECUTION STOPS. |
| c | |
| | NRAMETERS : |
| С | PTR - It POINTER TO NODE BEING EXPANDED |
| | L,M - I:COORDINATES OF "JUMP PROM" POSITION |
| | N,O - I:COORDINATES OF "JUMP-OVER" POSITION |
| | P.Q - I:COORDINATES OF JUNP TO POSITION |
| | OCAL VARIABLES: |
| C | NEWPTR - POINTER TO NEWLY-CREATED SUCCESSOR NODE |
| с | |
| | INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY, NRGEN |
| | L , NREXP, MAXNOD |
| | REAL F |
| | INTEGER PTR, NEWPTR, 1, J, L, M, N, O, P, Q |
| | COMMON/NODE/BOARD(5,5,300), OPENL(300), PARMTL(300), |
| | E CLOSDL(300), P(300), NPENNY(300) |
| | COMMON / CTRS / BRGEN , BREXP , MAXBOD |
| | CALL GETNOD(NEWPTR) |
| | MRGEN=NRGEN+1 |
| | OPENL(NEWPTR)=0 |
| | NPENBY (NEWPTR) = NPENBY (PTR) - 1 |
| | PARNTL(NEWPTR)=PTR |
| | DO 189 $I = 1.5$ |
| | DO 100 J = 1,5 |
| 160 | BOARD(I, J, HEWPTR)=BOARD(I, J, PTR) |
| | BOARD(L, M, NEWPTR) =0 |
| | BOARD (N, O, NEWPTR)=0 |
| | BOARD(P, O, NEWPTR)=1 |
| | IF (#PENNY (NEWPTR). HE.1) GO TO 150 |
| | CALL SOLVED(NEWPTR) |
| | STOP |
| 150 | CONTINUE |
| ••• | CALL SEARCH (NEWPTR) |
| | RETURN |
| | END |
| | |

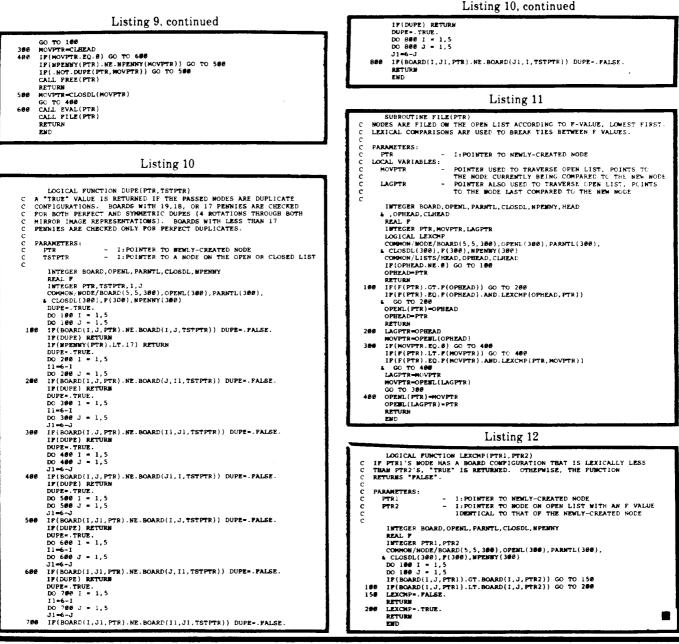
Listing 8

| SUBROUTINE SOLVED(PTR) |
|---|
| C THE SOLUTION IS PRINTED OUT, ALONG WITH THE NUMBER OF NODES C GENERATED, THE NUMBER EXPANDED, AND THE MAXIMUM NUMBER OF WODE |
| C DATA AREAS IN USE AT ONE TIME IN REACHING THIS SOLUTION. |
| C |
| C PARAMETERS: |
| C PTR - I: POINTER TO SOLUTION NODE |
| C LOCAL VARIABLES: |
| C PRTPTR - ARRAY OP POINTERS TRACING PATH FROM ROOT NODE |
| C TO SOLUTION |
| C INTEGER BOARD, OPENL, PARNTL, CLOSDL, NPENNY, NRGEN |
| 4 , NREXP, MAXHOD |
| REAL P |
| INTEGER PTR, PRTPTR(20) |
| COMMON/NODE/BOARD(5,5,366), OPENL(366), PARNTL(366), |
| LOSDL(300), F(300), NPENRY(300) Common/CTRS/NRGEN, NREXP, MAXNOD |
| WRITE(1,1000) NRGEN, NREXP |
| PRTPTR(1)=PTR |
| DO 50 1 = 2.20 |
| I1=PRTPTR(I-1) |
| 50 PRTPTR(1)=PARNTL(11) |
| DO 2566 K = 1,17,4 |
| 14=21-K |
| 11=PRTPTR(14) |
| 14=20-K 12=PRTPTR(14) |
| 12=PR1PTR(14) 14=19~K |
| 13 ± 2^{-1} |
| 14=18-K |
| I4=PRTPTR(I4) |
| WRITE(1,2000) |
| DO 200 I = 1,5 |
| 200 WRITE(1,3000) (BOARD(1,J,11),J=1,5), |
| <pre>4 (BOARD(I,J,I2),J=1,5),(BOARD(I,J,I3),</pre> |
| 6 J=1,5), (BOARD(I,J,I4), J=1,5) |
| WRITE(1,4000) MAXHOD 1000 PORMAT(1H1,19,' NODES GENERATED',5X,18,' MODES EXPANDED') |
| 2666 PORMAT(1H, 1B, HODES GEMERATED, 3X, 18, HODES EXPREDED) |
| 3888 FORMAT(1H , 4(513,2X)) |
| 4888 PORMAT(1H8, 'MAXIMUM NODES IN USE =',16) |
| RETURN |
| 2mD |
| |

Listing 9

SUBROUTINE SEARCH(PTR) BOARDS WITH 13 OR MORE PENNIES ARE CHECKED TO SEE 1F DUPLICATES EXIST ON THE OPEN OR CLOSED LIST AND FREED IF TEAT IS THE CASE. THOSE THAT ARE UNIQUE, AND ALL BOARDS WITH FEWER THAN 15 PENNIES ARE FILED ON THE OPEN LIST IN INCREASING ORDER OF F VALUE. 00000000000 PARAMETERS: PTR - I/0:POINTER TO NEWLY-CREATED NODE LOCAL VARIABLES: MOVPTR - POINTER USED TO TRAVERSE THE OPEN AND CLOSED LISTS NOVPTR - POINTER USED TO TRAVERSE THE UPEN / INTEGER BOARD, OPENL, PARNTL, CLOGDL, NPENNY, READ & , OPERAD, CLEERD REAL F INTEGER PTR, NOVPTR LOGICAL DUPE COMMON/NODE/BOARD(5, 5, 368), OPENL(368), PARNTL(388), COMMON/LISTS/HEAD, OPENAD, CLEEAD NOVPTR-OPENAD IF(SPENNY(FTR).LT.13) GO TO 688 IF(NOTR.EQ.8) GO TO 368 IF(NOTR.EQ.8) GO TO 368 IF(INTER.EQ.8) GO TO 368 IF(INTER.EQ.8) GO TO 268 IF(INTER.EQ.8) GO TO 268 CALL PRES(FTR) RETURN 188 200 NOVPTR-OPENL (MOVPTR)

Listing 10. continued



Did You Miss Any of These Issues?

To order back issues, send \$5.25 (includes postage) to The Computer Journal, PO Box 1697, Naiispell, MT 59905, Allow 3 to 4 weeks for delivery

Volume 1, Number 1:

- The RS-232 C Serial Interface, Part One.
 Telecomon management of the Apple I: Transferring Binary Files.
 Beginner's Column. Part One: Anyone for a Little "KISS" Electronics?
- Build an "Epram."

.

Volume 1. Number 2:

- · File Transfer Programs for CP/M.
- The RS-232-C Serial Interface, Part Two.
- · Build a Hardware Print Spooler, Part One.
- A Review of Floppy Dick Formats.
- Sending Morse Code With an AppleI.
- Beginner's Column, Part Two: Anyone for a Little "KISS" Electronics?

Volume 1, Number 3:

- Add an 8087 Math Chip to Your Dual Processor Board.
- Build an A/D Converter for the Apple].
- ASCII Reference Chart.
- Modems for Micros
- The CP/M Operating System.
- · Build a Hardware Print Spooler, Part Two.

• Multi-user.

True RMS Measurements

· Making the CP/M User Function More Useful.

• Build a Hardware Print Spooler, Part Three.

Gemini-10X: Modifications to allow both serial and parallel operation

· Beginner's Column, Part Three: Power Supply, Anyone for a Little "KISS" Electronics?

Volume 2, Number 2:

Volume 2, Number 1:

Volume 1, Number 4:

• Optoelectronics. • Multi-user.

Optoelectronics.

- · Build a High Resolution S-100 Graphics Board
- System Integration
- Optoelectronics, Part Three: Fiber Optics
- · Controlling DC Motors • Multi-User
- DC Motor Applications

Build a High-Resolution S-100 Graphics Board

Part Two: Theory of Operation

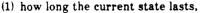
by Lance Rose, Technical Editor

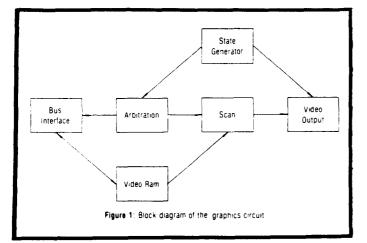
In the first part of this series we saw how the video monitor uses its sweep circuits to create a raster scan with which text or graphics information can be displayed. In this installment I'll explain how the video board operates in order to send the desired information to the video display device.

Figure 1 contains a block diagram of the graphics circuit. Let's look at each block and its function in turn. The state generator contains the state information for each of the possible logic states that the board can have (more about "states" in a moment). It is essentially the brain of the board and all the other parts center around it. The scanning block continuously scans the video RAM and extracts information a byte at a time for display. The video output circuit combines the scanned information with timing information provided by the state generator and outputs a composite video waveform of the proper amplitude and impedance. The arbitration circuit controls when the video RAM may be accessed by the system for updating or reading information stored in it. And finally, the bus interface circuit contains the necessary buffers and control lines to interface the graphics board to the S-100 bus.

Before looking at the actual schematic, we need to understand a bit about what a "state" is. If this is redundant to you advanced builders, please bear with me for a moment.

Intuitively one might think that a state is a set of conditions for a circuit which has all signal levels and timings specified, and in fact that is pretty much it. What we are using here is called a "state machine" which is, in simple terms, just a ROM or set of ROMS where each memory address of the ROM(s) causes the pertinent information for that state to be output on the data lines. This pertinent information is:





(2) the necessary output signals for this state in order to control the rest of the circuit, and

- 🦓

1

5-**8**

5 co**li**

科機

i di

い着

1. **Ú**

ы**н**

. .

口角

1.1

13

5 . **Ú**

-

ia)

4.98

(3) what the next state will be.

Complicated state machines can perform things like looping and branching to other states (microprocessors are examples of complicated state machines) but here the circuit is simplified by assuming the next state to always be one address higher in the ROM than the previous state. This does away with looping and branching (not needed here anyway) as well as additional ROM space to store the address of the next state. This method is analogous to the program counter in a microprocessor which simply fetches its next instruction from the next higher memory address unless a branch occurs in the program.

When the highest address of the ROMs is reached, the address counters simply "wrap around" to the first state again. Since we're dealing with a full interlaced video frame, this wraparound occurs every 1/30 of a second.

Let's go through the circuit in Figure 2 and look at how the various parts function.

Two of the gates in U1 are arranged in a conventional 84 crystal oscillator with a third gate serving as a buffer. This oscillator is known as the "dot clock" since a new dot on the horizontal line is displayed with each clock cycle. The 16MHz value is chosen as a frequency which will be compatible with most video monitors, i.e. will not cause the 1. dit display to go off the edge of the screen but will give a full screen display. If insufficient adjustment is available in the monitor to view the entire horizontal extent of the display, a higher frequency crystal can be used and the values in the state ROMs changed. Conversely, a slower clock will widen the horizontal display. Another choice is to program the ROMs to display less than 640 horizontal dots. One nice thing about a state machine is that it can be reprogrammed to tailor the board to different systems. It is possible to be compatible with some foreign TV standards by doing this. I will discuss this in more detail later on.

Since the timing is critical in the scanning portion of the circuit, gate delays have to be taken into account. This is done by providing a second dot clock signal which is nominally 180° out of phase with the first. This allows us to use either of two clock signals which differ by about 31nsec. This figure is just about right to compensate for the additional delays incurred by the state length counters U3-U5 so that video timing information latched out of the state ROMs will synchronize with video data retrieved from the video RAM.

The actual state generator is made up of U3-U5, 2732 EPROMs U6 and U7, ROM address counters U8 and U9, inverter U1e and hex D flip-flop U10. At each state change, the time of the next state is latched into counters U3-U5. This time is measured in units of the dot clock period. At the same time the outputs for this state are latched into U10. These signals are, in order, (D0) blanking for the composite video, (D1) sync for same, (D2) fast count enable for the RAM address counters, (D3) and (D4) control signals for bus access arbitration, and (D5) reset for the RAM address counters.

The dot clock is divided by eight with counter U2 which generates a byte clock signal. This signal, after inversion with U1f is used to load a new byte of information from the video RAM into shift register U11 for subsequent output in the composite video. This sequential process only occurs when the display is not blanked. During blanking, the blanking signal is fed back (after a little delay through U13a) to U12a forcing the parallel load inputs of both the byte clock and the LS165 shift register low. This effectively prevents output from the byte clock from interfering with the dot clock's advance of the RAM address pointers via U14a and U12b. This fast advance of 80 counts occurs between lines so that the next 80 bytes of information fetched from the video RAM will correspond to the scan line two lines from the previous one (full interlace display). We go to all this trouble so that the screen memory addresses are contiguous and relate one-to-one with the visible lines of the display in spite of the fact that in each field only every other line is actually being displayed.

After a byte is loaded into U11 by the byte clock, it is shifted out at the dot clock rate during the display portion of the line. Open collector hex inverter gates U15a, U15b and U15c combine the output from the video signal, blanking and sync to create a composite video signal at the base of transistor Q1. This signal is amplified by Q1 operating in an emitter follower configuration. Resistors R5 and R6 provide approximately a 75 ohm output for the video signal. Since the risetimes of parts of the signal are quite short, this may cause ringing in the video amplifier portion of some monitors. This is seen as an extra bright band near the left side of larger images and a higher brightness for individual dots. Strangely enough, this effect is actually useful in some types of display as it causes thin line images made up of individual dots to merge together more smoothly than if it were not present. Capacitor C2 bypasses the video output to reduce the risetime and eliminate this effect if desired. A lot depends on the individual monitor used so it should be tried both ways to find the better setting.

Counters U16 and U17 provide the local RAM address during periods of time when data is being scanned and output to the video generator. Each time a new byte of data is loaded from the video RAM into the shift register, the address counters are incremented by 1 through U12b. At the end of each line, pin 6 of U12b is held low by the blanking signal and the output from U12b is controlled by the input on pin 5. This input is the fast count signal previously discussed.

Tri-state buffers U18 and U19 buffer the local RAM address counters and drive the RAM address lines during

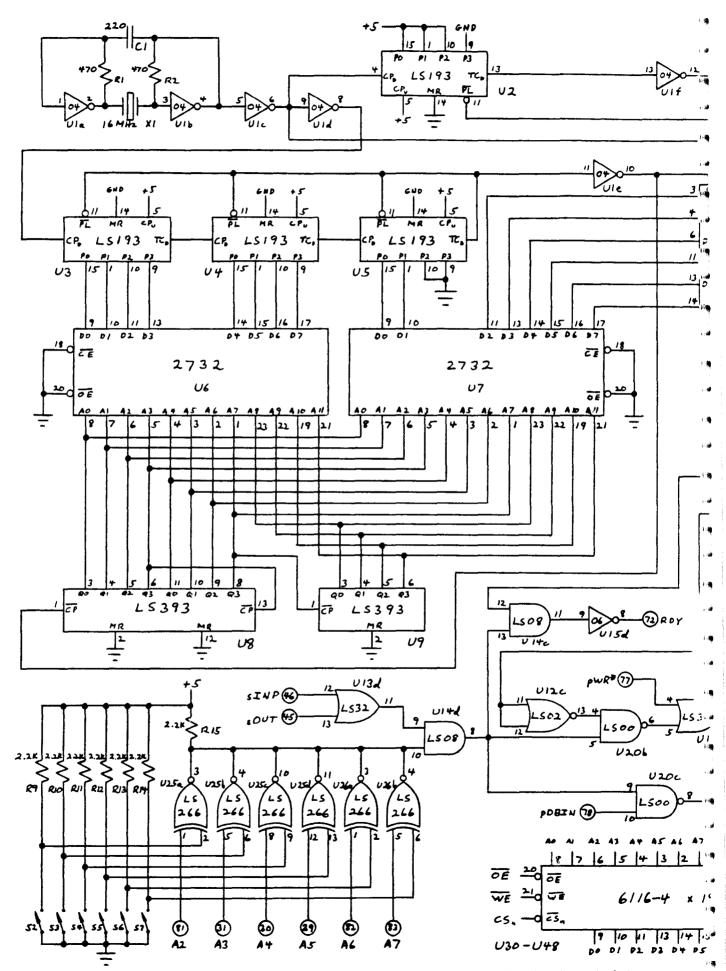
local access. Since the video output circuit doesn't need access to the RAM during horizontal and vertical retrace, these are the times we use to allow the processor to have access to the video RAM from the S-100 bus. When the outputs from the state ROMs cause both pin 1 and pin 2 of U20a to be high, U18 and U19 are enabled and the RAM address lines are driven by the local RAM address counters. Pin 6 of U14b is just the opposite and the bus RAM address latches U21 and U22 are turned off. When either one of the inputs to U20a goes low, the outputs switch and the video RAM address lines are driven by the latches holding an address written into them from the S-100 bus. In this state the video RAM address read from or written to will be determined by the S-100 system.

Since with two bits of information we can select one of four states, you might wonder what the additional states are. To explain this, let's assume the following: say that the RAM is being accessed from the bus and a write cycle has just begun. At this moment, a state change occurs and either pin 10 or pin 12 of U11 goes high. Acting through U13b this will generate a wait state on the S-100 RDY line since pin 13 of U14c will also be high, indicating that the board is currently selected. If the processor enters a wait state before finishing the write cycle it will leave pWR* active during the entire wait state. If we allow pWR* to remain active at the RAM chips and switch to the local address counter for the scan line, a spurious write will occur on all RAM chips selected during that scan line. If we go to another state with pin 10 low and pin 12 high on U11, we maintain the wait state but in addition disconnect pWR* from the RAM chips via U12c, U20b and U13c. Once pWR* is disconnected from the RAM we can go ahead and let the local address counters select the RAM address.

At the end of this scan line we perform the inverse operation. First we switch back over to the bus RAM address latches, and only then do we re-enable pWR^* . remove the wait state and allow the processor to continue its write cycle. This arbitration scheme effectively shares the video RAM access time between the video scanning circuitry and the bus access circuitry and prevents any hash in the video display due to bus access during a scan line.

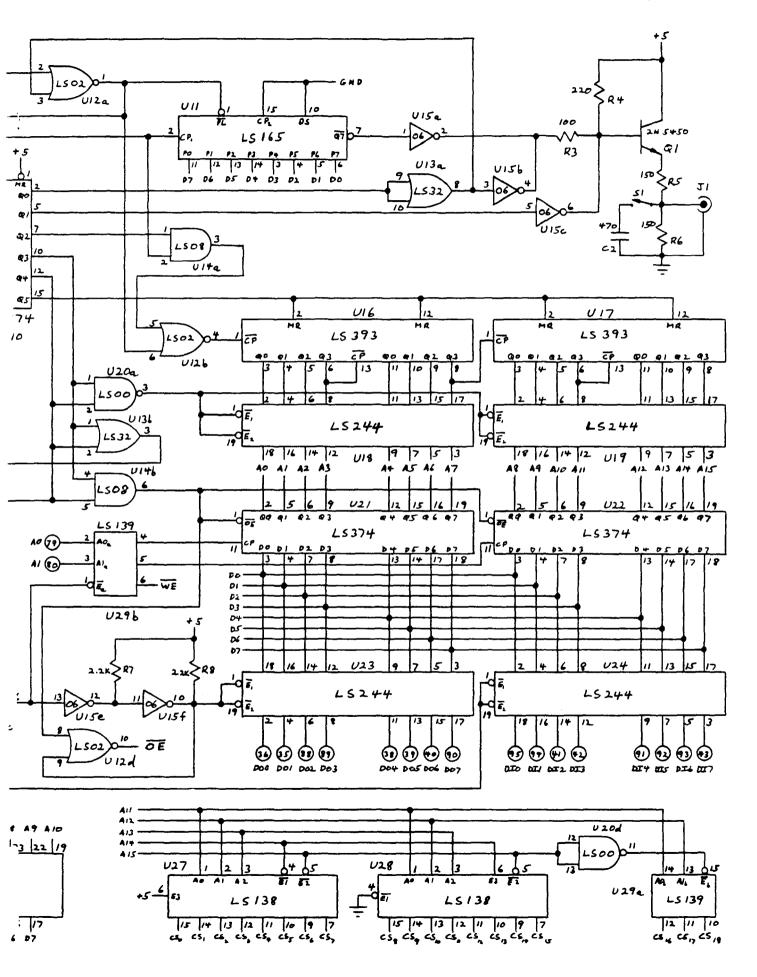
We don't have this problem during a processor read cycle because it doesn't matter if the S-100 data-in lines contain data from the RAM address currently being scanned and not the address we will ultimatley read from. The actual read operation will not complete until the desired RAM address is back on the RAM address lines and the proper data is on the data-in lines.

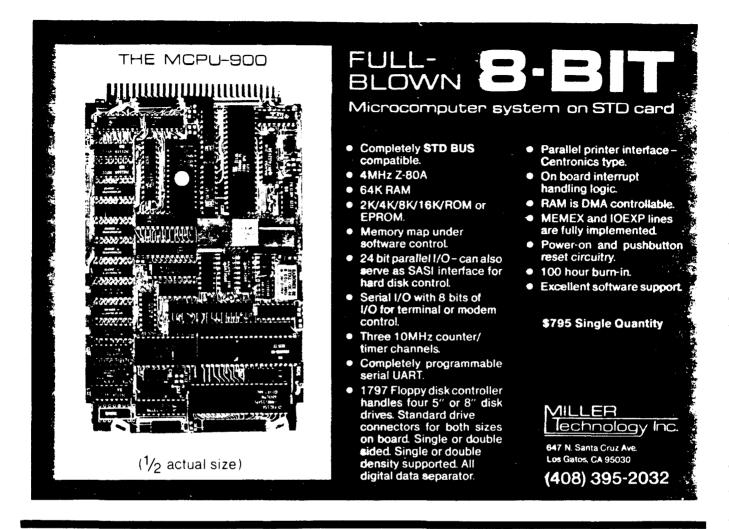
The bus access is performed in a pretty straightforward manner. Exclusive NOR gates U25 and U26a and U26b act as a comparator between the address on A2-A7 of the S-100 bus and the values preset by switches S2-S7. If the address matches and an I/O operation is selected by either sINP or sOUT high, pin 8 of U14d will go high indicating board select. If a scan line is currently in progress, a wait state will occur until pin 12 of U14c goes low. In addition pWR* will not be activated as discussed above. If the bus cycle is a read operation, pDBIN will be active and pin 8 of U20c will



NOTE: The above schematic is published for our readers' personal use. The author retains all resale rights to the design.

1



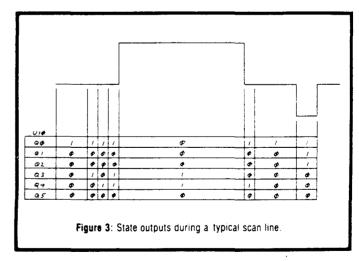


turn on the data input buffer U24. If the bus cycle is a write and bus access is allowed, the board select will combine with pWR^* through U13c, U15e and U15f to enable the data output buffer U23. When this occurs, OE* to the RAM chips is turned off to prevent conflicts between a RAM chip and the data out buffer both driving the local data bus at the same time. U15e and U15f provide some delay so that the bus data out buffer holds its data valid until after WE* goes false to the 6116's. In order to select the proper device to write to, U29b is set up as an address decoder for A0-A1 of the S-100 bus. Depending on which port number of the four is selected, data will be written to the low byte bus address latch, the high byte latch or the video RAM itself at a RAM location determined by the most recent values written into the bus address latches.

Decoders U27, U28 and U29a select one of the 19 6116 RAM chips for reading or writing on both scan lines and bus accesses. U20d is used as an inverter so that U29a will select RAM address values of 8000H and up.

Though details of the arbitration may sound complicated, it is actually a pretty straightforward scheme. Figure 3 shows the state ROM outputs at the various places in a typical scan line.

Something to keep in mind here is that you need not totally understand the circuit in order to successfully build



it and use it. Some experience with wire-wrapping prototypes is helpful, along with some patience since there are quite a few connections to be made, particularly in the video RAM portion of the board. 行機

14

.

In the next part of this series I will describe how to build and check out the board, and will also provide a program for generating the state ROMs if you have access to an EPROM programmer. We can also supply preprogrammed ROMs at a nominal cost for those without such facilities.

Multi-user

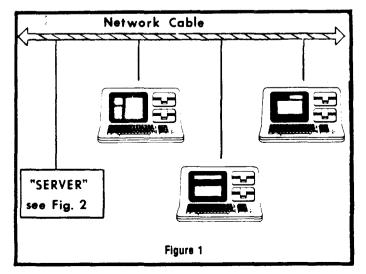
A Column by E.G. Brooner

In previous columns we tried to avoid technicalities as we outlined the general kinds of milti-user systems. With all of that behind us, it's time to get a bit technical as we describe a particular kind of network (Ethernet) and a particular implementation known as "Etherseries." Very briefly. Ethernet is one of the earliest and most widespread types of network, originally meant for use with sophisticated minicomputers, and Etherseries is the particular version designed for use with the very popular IBM PC.

In contrast to many advertised net systems, this one is actually available as an off-the-shelf product and this reviewer has seen it working. Bob Metcalfe, one of Ethernet's inventors, founded the 3COM corporation to build and market network components. In his California plant we observed 50 PCs all working away at the tasks typically performed by desktop computers, all sharing one large hard disk storage system and a few strategically located printers. Figure 1 illustrates, in a simple way, how several computers are connected to form a typical network

What's Different About Ethernet?

Designers attempt to define networks within the framework of a seven-level protocol system; this sometimes leads to more confusion than clarification. Briefly, a protocol is a set of specifications for performing a certain function. If you connect a peripheral (printer etc.) to your computer, you use a certain standard method – RS232 for example. This is a "level-one" protocol, which is a definition of how two pieces of equipment communicate with one another. The way the data is bunched together for transmission to the printer constitutes a "level-two" protocol, of which there are many kinds. These two protocols, whatever they may be,



constitute a complete data exchange system.

So much for communicating between two devices. Now, if we want to connect a larger number of devices, and selectively communicate with one or the other at various times, we need a third protocol to route and control the communication process. Three protocols define a network. The third protocol (again, whatever set of rules is used for the purpose) is what distinguishes a network from simpler systems.

Ethernet is really a definition of the two lower protocols. It defines them so clearly that any equipment connected via an Ethernet can (theoretically, at least) communicate. However, the third, or network control layer will differ as we move from one machine to another. And the level one and two protocols, which define Ethernet, are more-or-less lumped together. For example, we cannot say that it uses some particular standard or otherwise defined protocol at each level. Ethernet is a communication system that performs, in an integrated way, the functions that would otherwise require two separate protocols.

There are a lot of Ethernets (the generic term) for use with various kinds of computers. Etherseries as presently defined is only for the IBM PC. (3COM'S literature also calls the PC version Etherlink). A very similar Etherseries package would suffice for Apples or some other micros, but it would not be identical at the third level. The third level, then, is usually unique to a particular machine or operating system; it is a special, customized interface, usually combining hardware and software components. Etherseries for the PC consists of a single plug-in board and a 5⁻ diskette and, of course, the interconnecting cable. An important point to remember is that to the user, this network appears simply as an enhancement to PCDOS, in the form of a few new commands.

How It Works.

All Ethernets have in common the CSMA/CD (carrier, sensing, multiple access, collision detection) scheme for "directing traffic" on the single coaxial cable that links all of the users. CSMA/CD is one of the two major schemes that are in use for this purpose (the other major method and the many lesser ideas for accomplishing the same thing will be discussed in future issues). When a user attempts to communicate with another device, his network software composes a "message" which includes both the data and the routing instructions. The message may be so long that it has to be broken into several smaller "packets" that will be reassembled at the receiving end of the circuit. All of this is done automatically; the user only enters simple commands, such as would be used with one of his own peripherals.

Assume that a message is ready to go. The net communication system is always "listening" to the cable. If it has not heard any signals (data) being passed for the last nine microseconds, it goes ahead and transmits the message or packet. While transmitting (remember, at something like 10 megabits per second), it continues to listen. If what it hears does not exactly match what it is sending, it assumes that either: a) an error has been made in transmission, or b) someone else has tried to transmit at the same time. In any event, it has detected a "collision" on the cable. By means of a rather complicated formula, known as the backoff algorithm, it calculates a brief waiting period and tries again. The back-off algorithm operates in such a way, and the transmissions are so rapid, that if two messages do collide on the first try, their next attempts will probably not coincide, and both will succeed.

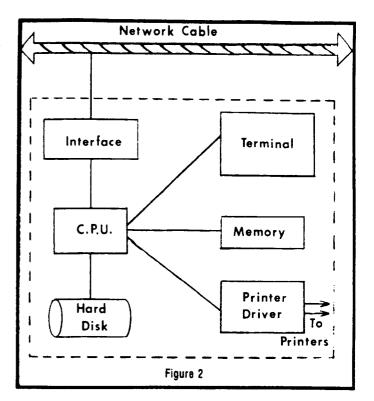
Every station on the cable "hears" every transmission, but ignores those not addressed to it. This is typical of most networks; the undesired messages simply go on by without interrupting normal operation of the other users. In fact, a user has no way of knowing whether messages are flitting along the cable or not, unless one is addressed directly to his equipment. Even if it is, the interruption is very brief; network communication transactions typically take only fractions of a second.

As It Is Really Used

As many as 1000 PCs could conceivably be connected together, exchanging data and messages and sharing peripherals; but let's be reasonable and stick to the 50 that 3COM was using when I visited their headquarters in Mountain View, CA. First of all, each of the 50 PCs could, and usually did, operate just as if it were the only one in the building. Each had one or two floppys installed and some had printers attached; some were doing engineering work, some were for bookkeeping, and so on. At this point there was no obvious difference between this installation and any other large group of personal computers, except for the coaxial cable running unobtrusively past the rear of each machine.

But somewhere in the system there was a "disk server"-Ethershare is the trademarked name for 3COM's; almost every network has something similar. Ethershare is itself a computer complete with a keyboard, screen, gobs of memory, and a 40 megabyte hard disk. The hard disk is accessible in "chunks," each containing about the same amount of storage as a double density floppy. Figure 2 is a schematic representation of a typical server.

These subdivisions of the disk are known as "volumes." Some are accessible to one user, some to another, and some are available to anyone wishing to read them. The public volumes are the key to many of the net's features. One or more may be a common data base, or contain commonly used software, available to everyone. One or more may be used as a repository for "electronic mail" which is really a set of electronic in-baskets and out-baskets where memos can be exchanged. The others may be assigned to individual users and can be password-protected.



1.38

1.1

1

-

us.

÷:đ

Say that a user has his own floppys, designated "A" and "B." He can also have two volumes on the server, designated "C" and "D." To access one of the extra storage units, he enters a simple LINK command that names the volume and equates it with "C." From that point on, until he "UNLINKs" it, he is accessing that volume, via the network, just as if it were a third drive on his own PC.

Ethershare also controls a pair of printers. Everything sent to one of the printers, from any location, is first "spooled" and then printed when the printer is free. Of course, this also frees the user to do other work after sending a print command. (See the description of print spooling which appeared in recent issues of *The Computer Journal*)

When we reviewed the operation of this particular installation it was new and the users were still in the process of adapting themselves to it. The electronic mail feature was the one which seemed to most impress those to whom it was available. The mail program included a fairly competent word-processing capability. With it a user can compose a note or letter, edit it as with any other word processor, then deposit it in one of the mail files which is accessible to the addressee. Such "mail" can be sent to any station or individual within the bounds of the network, or to a group of addressees. The mail can be accessed on-screen by the addressee anytime after it has been "sent." After that, it can be printed and/or destroyed. All messages are date and time stamped and listed on a directory. This system should be more practical than hand written memos, and it is certainly more convenient to use.

Evaluation

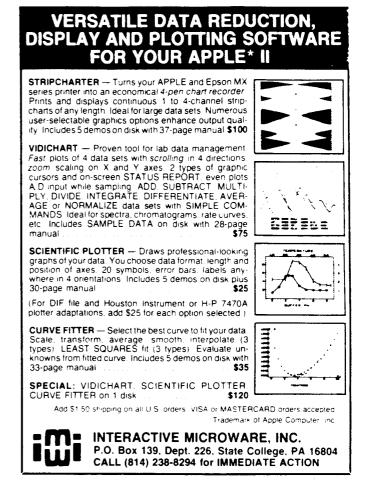
Etherseries for the IBM PC is an available product and not just someone's hope for the future. It is typical in many ways of any other Ethernet, and bears some resemblance to almost any of its many competitors. As far as price is concerned, it is neither the lowest nor the highest priced system available; at \$950 per station, along with the relatively high initial cost of the PC, (and the server) it is probably out of the price range of all but the most serious micro users.

Its performance is superior to many competing products; the mail feature in particular is slicker than some others we reviewed. 3COM is a leader in the Ethernet business and a 3COM product, associated with an IBM product is sure to be dependable and supportable. If it is within your price range, you could do much worse than tie a bunch of IBM PCs together with this network.

For further information, see your IBM PC dealer, or contact \$COM Corporation, 1390 Shorebird Way, Mountian View. CA, 94043.

Acknowledgement

An acknowledgement which should have been included in Volume II, Number 2 of *The Computer Journal* was inadvertently omitted. The article "Controlling DC Motors with a Microcomputer" by N. Bungard was in part made possible by research funded by National Science Foundation grant MCS-8210194.



Computer Nostalgia:

Who Invented the Personal Computer?

One of the present manufacturers likes to advertise that they invented the personal computer; I think 1978 is the year this supposedly happened. True? Absolutely not. In 1978 Apple was just beginning to be heard of; the Radio Shack model 1 was selling well, and the Commodore Pet, though less agressively marketed, had been around longer than either and was (in many opinions) a better machine.

Before any of those "personal computers" were even a gleam in their designer's eye, there were a large number of what were then called "hobby computers." The hobby computers were usually available optionally as kits or assembled products. The best known of these were the Imsai and Altair. And before the commercially available hobby computers, there were the real pioneers, individuals and groups building essentially the same thing strictly from scratch. These were the people who really "invented the personal computer."

The leader of that movement was Dr. Johnathon Titus of Blacksburg, VA. The 8008 (the first 8-bit microprocessor chip) came out in 1972. It was designed basically for control applications, but Dr. Titus, who was then doing scientific work with minicomputers, thought it had potential as the basis of a homemade personal machine.

In 1972 he managed to obtain some 8008 chips (then \$120 each) and designed and built the first so-called "Mark-8" microcomputer. According to Titus, this first micro had 750 bytes (that's less than 1K) of memory! A refined, printed circuit version was written up in Radio Electronics magazine in 1974, and this marked the beginning of the movement toward hobby (or personal) computers. User groups all over the country, but primarily in southern California, built and used Mark-8 microcomputers. And in what was later to be known as "Silicon Valley" one of the earliest computer clubs, known as the Homebrew Club, saw hobbyists and professional engineers pooling their knowledge to develop even more "hobby computers." As someone said at the time, "These guys build computers in their garages and become millionaires."

By late 1974 the 8080 chip, successor to the cruder 8008, had dropped to \$200 each. MITS (a now extinct company) built a kit computer around the 8080 and what is now known as the S-100 or IEEE-696 bus. They were swamped with orders before 1975 got under way. MITS' Altair was closely followed by improved look-alikes such as the (now also extinct) IMSAI and several others.

Who invented the personal computer? It certainly wasn't Apple, or even Tandy or Commodore and it happened long before 1978. by E.G. Brooner

Ine Computerist's Caleriua

.

-

7

.

MAY 07C0

| SUN | MON | TUES | WED | THURS | FRI | SAT |
|-----------|-----|------|-----|-------|-----|-----------|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | A | В | С |
| D | E | F | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 1A |
| 1B | 1C | 1D | 1E | 1F | | |

SYSTEM INTEGRATION

Part Two: Disk Controllers and CP/M 2.2 System Generation

by Bill Kibler

Disc Controllers

In part one of this series I pointed out that a system could be built with used parts for less than \$1000. With this in mind, I will try to show why buying a new disk controller instead of a used one might be advisable. If you go to swap meets, you will find many of the same type of disk controllers for sale-mainly the CCS 2422. I have considerable experience with this card, and I feel that it is not a good buy. The documentation is good, but the unit's ability to interface (hardware) with other types of CPUs and memory is rather poor. The CCS 2422 is not IEEE-696 (S-100) compatible and was designed for their own cards only. These cards abound at swap meets because their owners have been unable to make them work satisfactorily.

Most CPUs, memory, and I/O ports, will work with each other, and if not compatible can usually be altered. Timing problems with I/O ports are rare, and memory is cheaper to replace than to modify. Disk controllers are another matter, however since their timing is quite critical. The VLS controllers used in the newer units are either NEC upd765 or WD 179x. Both of these have a lot of special features, some good and some bad. Common to both is the need to read the data as fast as the chip provides it. The CPU's memory fetch and write cycles must be faster than the controller's read or write times in order to clear the controller's data buffer and be ready for the next byte of data. When the timing is not correct, the unit will error and stop the data transfer. Software has advanced somewhat since the units were first made and buying a used card with no hope of upgrading is not desirable. CP/M 3.0 uses multiple banks, and consequently disk cards that are memory mapped may not work under banked operation. Some manufacturers are supplying new BIOSs and possible upgrades; these are always preferred but seldom seen at discount prices. The cost of a system without a controller is \$300 to \$400, and adding a new controller for \$400 (with software) will still keep the system price under \$1000. This new unit will be IEEE-696 compatible and therefore will work without hardware modifications. The software will be current, possibly even CP/M plus, and the utilities will aid in bringing it up. These are the premises under which I bought the the SDSystem's Versa Floppy II/696 and CP/M 3.0. However, much of what I will review is also applicable to buying used controllers or upgrading existing systems.

System Configuration

My system consists of the Computime CPU, SD's VFII, and SD's Econoram II (a 256 K banked memory). Although this is not an ideal system, it will help to illustrate the ease and the problems of system integration. The first step in any integration is understanding the individual components, their memory map, and any special problems to be handled.

The CPU is a Z80 with a serial and parallel port. The boot jump PROM or monitor will be on this card along with the serial and parallel ports. The SD memory is 256K, with one bank of 64K and four banks of 48K. One port is used to switch the banks on the memory card, port FF hex. The disk controller uses a WD 1795 and six port addresses; those switchable ports are 60 through 67 hex. The software purchased with the unit is CP/M 3.0 for their SBC-300. The sale price, including CP/M + was just under \$400 (CP/M for \$90!) and thus falls into our under \$1000 system cost. Keeping in mind the important points covered in part one, let's see how these units rate.

a) Documentation: Computime is better than SDSystems.

b) Hardware: generally good design (both IEEE/696).

c) Software: SD's needs more and better documentation.

d) Adaptability: hardware ok, software needs help.

e) Support: both ok, SD needs better follow-through.

f) Repairability: no PROMS or PALS.

From the above quick guide, you can see that some plusses and minuses do exist. Let's review three of these points in more detail.

Documentation

Computime produces a rather complete manual on their CPU, and SDSystems could learn something by looking at it. Although I would not consider the CPU manual to be the best, it does cover the topics completely enough to make bringing up the unit fairly easy and straightforward. SD's manuals are quite brief and do not include a theory of operations. There is a section called "Functional Description" which replaces theory with an overview that hides all important facts from the user. This manual should be read carefully both for errors and for oversimplification of explanations. A case in point is the section called "Port Usage Data"; not mentioned is the fact that some entries must be complimented first (drive select). The drive select data has one line's data already complimented (bit 6, the single/double density flag), showing how confusing inadequate explanations can be even for the writers of the manual.

The listing provided for the controller is only part of the monitor and will not work as printed. SD has their own operating system (COSMOS) and the DDBIOS listing has routines for it. The disk select routines will fail if used as written. This error was discovered when I mistyped the program and the routines worked. I later discovered the typo, corrected it and found the monitor would not work. The next major problem is the lack of information on how the system was intended to interface with other components, both theirs and others. Normally, a theory of operation would describe in detail the various handshake operations, the this-before-that stuff, and would help the software hacker to write his own programs. SD must consider all information to be trade secrets, as they provide little insight into the inner workings of their cards.

Software

SDSystem's implementation of CP/M 3.0 does not have much competition to compare it with, and therefore it is hard to know how good or bad it really is. It appears that some work has gone into setting it up, but a considerable number of programs were not included when I first started on this project. SD was not supplying the full CP/M BDOSs; you got either the banked or nonbanked version, but not both. After three months of fighting, and a talk with the VP of Sales, they should now be supplying all the original Digital Research files. For the integrator this leaves only the how and why of their disk controller's timing in question. A new DDBIOS is available (they are aware of the many errors in the printed DDBIOS) but I have yet to receive mine. This condition is to be expected, and a lot of playing around will be needed to find the right software handshakes for your system.

Support and Hardware

The Versa Floppy II is the only card that I have been able to bring up without a lot of cutting and hacking. I feel that in terms of hardware, the product is quite sound and should give a lot of trouble-free service. The design is rather straightforward and, except for not having a PROM on board, I have not been able to find anything to complain about. The factory support has been about as expected, with one surprise; the support person is still there after six months. The company appears to be serious in wanting to produce a good product and improve their image, but as yet they haven't achieved that goal. They welcome constructive comments and will change their policy if a strong enough case is presented.

Making It Work

Now that we have some idea of the product and the support, let's look at what it takes to make it work. In considering how to set up the system, I toiled long over the final memory map, much as you should before starting. To make this work for most people, the system will have to come up in stages, first a monitor, then CP/M 2.2, CP/M 3.0 non-banked, and lastly CP/M 3.0 banked. Another consideration is the use of existing 2.2 systems, both SD and others. Most systems are port addressed disk controllers with some form of boot/monitor PROM (either mapped or phantom). The SD system was designed with a monitor at E800hex, and disk PROM at F000hex. Through banked switching, the loss of the memory space was minimized, but for most users this will not be an acceptable solution. The entry points to both the monitor and disk functions. however, were the F000hex entries. In my design, a fixed 2K EPROM resides at F000hex that provides both monitor and disk functions. I feel that this is fairly close to what most users will have. This design also leaves the memory open above the PROM for non-banked buffers. disk byte storage tables, SCBs, RAMdrives, or whatever. The BIOS also becomes quite short by making calls to the PROM for disk I/O, CON I/O, and initialization. My personal preference is to boot from a monitor after I know the basic system is running rather than to wonder why the disk keeps on running without anything happening. Checking the memory map listing will indicate options and give comparisons to other systems.

部制

日調

e site

eq.

词

8**q**

13

1.4

镧

u d

. .

153

24

أوب

14

11*4*

. á

÷ N

£6**4**

の費

1.4

13**9**

CP/M 2.2

Assuming that you are bringing your system up from scratch, you will have to burn a PROM for F000hex. This PROM should contain a monitor and the disk routines found in DDBIOS. Listing 1 contains the needed changes to make it run. Those routines dealing with the FORMAT portion will not fit if you include a monitor. They can be included in a separate new format program or in the PROM if an autoboot operation is intended. I have made a separate format program because of the monitor and because I have changed the disk parameters, the number of drives and the layout. As the user of the system, you must at this point make some decisions as to the final operation of the system. Now is the time to determine the number of drives, the number of formats to be used, the types of I/O, and any other special functions you may desire. To make it easier to get the 2.2 software up and running, I have chosen to include only information for 8" single density disk drives. For transferring information and working between other systems, this is the preferred format. Larger densities require disk buffers and deblocking algorithms to combine CP/M's 128 byte sectors with the disk's actual physical sector sizes. The use of the non-standard 128 byte double density sector saves this blocking problem and is what SD used at first (users could still implement this format if needed in 2.2; 3.0 does the deblocking internally). The DDBIOS listing is somewhat mixed up when it comes to double density and will not work correctly as listed in the manual. For an easy way out, SIG/M has a ready BIOS on their disk #26, complete with a format program. The software listings provided here show how short a BIOS can be when using PROM based functions, and what is needed if only the listings from a manual are used. I recommend The Programmer's CP/M Handbook by Andy Johnson-Laird for more in-depth discussions on CP/M 2.2's inner workings.

For people bringing up other systems, the source code needed for the monitor is usually found in the disk controller manual, either in a monitor or as a separate BIOS program. The major advantage of bringing the unit up under CP/M 2.2 is the readily available support currently at hand. I obtained the original 2.2 BIOS from a fellow club member. This BIOS, with some modifications, was up and running in 30 minutes. This made me sure my system worked and gave me a system with which to write the BIOS for CP/M 3.0. Let's now look more closely at the necessary steps in bringing up

CP/M 2.2.

Step 1: List all ports and memory locations that will be used by the various cards. In CP/M 3.0 these are listed in PORTS. LIB. Starting a list now will make things easier later. I chose to put my disk buffers and storage locations above the PROM so that they would be protected during bank moves. When listing ports or memory locations, it helps to show what happens when accessing these entries. Note what bits contain information and how it is used. Remember that these ports/memory locations must be the same as those used in the monitor (the listings addresses are the same as DDBIOS and not above the PROM as I had suggested).

Step 2: Photocopy the listings needed for each port function. Typically these are the CON IN/OUT routines that are found in the manuals. Past experience has shown that it is best to just steal the routines word for word from the manuals. It is not unusual that the mention of timing problems which require special software never find their way to the books. So steal them all. Using the copies will be faster than trying to find them again in one of the many manuals, and also allows you to make lots of notes on the listings.

Step 3: Create the monitor program. composed of the new (stolen) I/O routines, system initialization routines (also stolen), disk functions (yes-stolen), and monitor functions (try stealing them from SIG/M disk #26). My monitor routines were originally from a CCS Z80 monitor that I converted to 8080 nemonics. Using Wordstar to block delete the old disk routines and add the new ones will speed up the operation (if the monitor is for another system). It is strongly recommended that you make small files of the routines for block adds later as these can then be turned into macros for CP/M 3.0.

Step 4: Assemble and burn the new monitor PROM. Test it to see that all functions and routines work, as CP/M will be calling these later. What you should have is a PROM at F000hex to F7FFhex with a jump table at F000hex similar to CP/M's BIOS entry table. This will make it easier to call routines, especially if you later add or delete a byte or two (the tables entry points become fixed at this time).

Step 5: Compile the new BIOS for CP/M 2.2. What will be needed are the Disk Parameter tables, any routines not in the monitor, or those that may change often. My BIOS has several of the routines which are a simple call to the PROM entry table, and a return out of it. About 600hex should be more than adequate for this type of BIOS. The CP/M systems or interface guide will list the needed routines. See the sample BIOS for more insight.

Step 6: Assemble your new BIOS, correct errors found in the assembly, and add it to a CPM60K.com file. This file is generated by "MOVCPM 60K**" or SYSGEN and doing a "SAVE 36 CPM60K.COM," when prompted to write to ? drive or reboot. This is covered in more detail in the SYSGEN manual. Use DDT to add the BIOS at 1F80hex (I like to fill 1F80 to 2500 with 00 so that dumping the memory will show if the addition is correct or not). Use "IBIOS.HEX,"cr,"R3580,"cr, then "D1F80,"cr to see if the jump table is where it is supposed to be. The "R3580" is the offset needed to load a hex file at 1F80 when it was intended to go at EA00hex.

Step 7: Save the file by "GO," then "SAVE 36 NEWCPM.COM,"cr. Use SYSGEN next and skip the "load from drive?" by hitting the return key. Write the new system to a spare disk, and then reset the system and try it. If, like myself, you are upgrading, then this testing will involve removing the old disk controller card, doing some jumper changes, a PROM change, installing the new controller, and then trying the new disk. If a second system can be borrowed for this initial start up, a lot of frustration can be saved. Do not be surprised if it does not work the first or second time. There are normally a number of typos that will need to be corrected first.

Step 8: After booting the system successfully, make lots of backups and then test it fully in all modes and ways to check for more errors. Assemble the format program and generate new disks.

For installing the new BIOS without an existing system. there are some alternatives. It is possible to generate a running system from a monitor (assuming a complete CP/M already existed for the system with only incorrect I/O codes) if the PROM can be programmed elsewhere. Some systems have the PROM/monitor on the controller along with a serial port, thus allowing them to be brought up initially from disk (the Micromation Doubler is just such a controller). Most companies have their PROMs and monitors set for their own I/O. These will need to be changed for mixed systems. To make these changes, a running monitor is needed that can do memory changes, dumps, and disk reads/writes. It may be necessary to buy such a PROM from a local dealer or fellow club member, but the cost will be low in comparison to buying all matching equipment. To change the I/O, just read in the system with the disk read function, change the I/O port addresses (must be the same length or shorter, and will have to be converted to machine language) and then rewrite to a new disk. Change the disk and try booting the system. Normally it will not work the first time, but keep trying; it will work if you have stolen all the right code.

This multiple-step method of system generation should get you up and running. Keep in mind that you will encounter plenty of obstacles, but knowing that they will appear and can be overcome should keep the frustration level low. Hopefully I have shed some light on what is needed.

Review

In this installment, I have provided some insight into the SDSystems Versa Floppy II controller, listed some things to watch for, and reviewed CPM 2.2 system generation. In the next article I will list the changes from 2.2 to 3.0 and help you generate a new non-banked BIOS.

The listings for this article are found on pages 24 and 25.

| *FFFF *FFFF *FFFF *FFFF *FFFF *FFFF *FFFF BOS bIOS monitor RAM-DRVS *F800 *F800 *F800 BOS BDOS BDOS monitor monitor monitor monitor BOS BDOS BDOS BDOS monitor monitor monitor *Ex00 *F400 *F200 *F000 *F000 *F000 *F000 *Ex00 *Ex00 *F200 *F000 *F000 *F000 *F000 *DF00 *DC00 *F200 *F000 *F200 *F200 *F200 *DF00 *DC00 *F200 *F800 BDOS *FA00 BDOS *FA00 *DF00 *DC00 *DC00 *Ex00 BDOS *Ex00 *Ex00 *Ex00 *Ex00 *Ex00 *Ex00 *Ex00 | Normal extended prom buffers SCE SCE BIOS bIOS monitor RAH-DRVS FF00 buff FR400 FF200 FF200 FF200 FF200 FF200 FF200 BIOS BDOS BIOS monitor monitor monitor monitor F400 FF200 FF200 FF200 FF200 FF200 FF200 BIOS BIOS BIOS FE400 FF200 FF200 FF200 CCP CCP BDOS BIOS FF200 FF200 FF200 TPA CCP BDOS FE400 FF200 FF200 FF200 TPA CCP BDOS FE400 TFA FFA TPA CCP BDOS FE400 TFA BIOS BIOS FE400 TFA Secon CP/M CP/M CP/M CP/M PS0 Pg0 Pg0 Pg0 Pg0 Secon | Dormal extended prom buffers SCB's | STANDA | RD | PROM | | SDSystems C | |
|---|---|--|--------------|------------------|---|-----------------------------------|--|--------------|
| Dormal extended prom buffers SCB* SCB* BIOS bIOS monitor MAHDRNS PF00 buffers FA00 F200 FF00 buffers SCB* SCB* SCB* BIOS BIOS BIOS BIOS PF00 FF00 FF00 FE00 F200 FF00 FF00 FF00 FF00 FF00 FE00 F400 FF00 FF00 FF00 FF00 FF00 CCP CCP BDOS FE00 FE00 FF00 FF00 FDA TPA CCP BDOS FE00 FF00 FF00 FDA TPA CCP BDOS FF00 FF00 FF00 FFA CCP BDOS FF00 FF00 FF00 FF00 FFA FFA CCP BDOS FF00 FF00 FF00 FFA FFA CCP BDOS FF00 FF00 FF0 FFA< | Dormal extended prom buffers SCF SCF B105 b103 monitor RRH-DRVS FF00 buff FR400 FF200 FF200 FF200 FF200 FF200 FF200 B005 BD05 B105 monitor monitor monitor monitor FE00 FF200 FF200 FF200 FF200 FF200 FF200 B005 B105 FF200 FF200 FF200 FF200 FF200 CCP CCP BD05 B105 FF200 BT0 FF200 DF200 FC200 FC200 FC200 FC200 FC200 FC200 TPA TPA CCP BD05 BT05 FFA0 DF200 FC200 FC200 FC200 FC200 FC200 DF200 FC200 FC200 FC200 FC200 FC200 DF200 FC200 FC200 FC200 FC200 FC200 FFA00 B105 <th>normal extended prom buffers SCS* SCS* 9105 9105 9105 9105 9105 9105 9005 9105 9105 9105 9105 9105 9005 9105 9105 9105 9105 9105 0005 9200 9200 9105 9105 9105 0005 9005 9105 9105 9105 9105 0005 9005 9105 9005 9105 9005 7PA 7PA CCP 9005 9105 7000 7PA 7PA CCP 9005 9105 7000 7PA 7PA CCP 9005 9105 9100 900 930 930 930 930 930 930 9010 9010 900 930 930 930 930 9010 900 930 930 930 930 930 9010 900<th>non-prom</th><th>system</th><th>besed syst</th><th></th><th></th><th>non-banke</th></th> | normal extended prom buffers SCS* SCS* 9105 9105 9105 9105 9105 9105 9005 9105 9105 9105 9105 9105 9005 9105 9105 9105 9105 9105 0005 9200 9200 9105 9105 9105 0005 9005 9105 9105 9105 9105 0005 9005 9105 9005 9105 9005 7PA 7PA CCP 9005 9105 7000 7PA 7PA CCP 9005 9105 7000 7PA 7PA CCP 9005 9105 9100 900 930 930 930 930 930 930 9010 9010 900 930 930 930 930 9010 900 930 930 930 930 930 9010 900 <th>non-prom</th> <th>system</th> <th>besed syst</th> <th></th> <th></th> <th>non-banke</th> | non-prom | system | besed syst | | | non-banke |
| BIOS bIOS monitor NAHORVS FF00 buff F400 *F200 *F800 *F800 *F800 *F800 BOOS BDOS BIOS monitor monitor monitor monitor TE600 *E400 *F200 *F000 *F000 *F000 *F000 CCP CCP BDOS BIOS resident 8105 *FP00 *DC00 *Fe1dent *C800 BDOS *DC00 *DC00 *E400 TPA *C800 *DC00 *DC00 *E400 TPA *C800 *DC00 *DC00 *E400 TPA *C800 *DC00 *DC00 *C000 C000 BNK BNK *0100 *DI30 *0130 *0130 *0100 *0100 *0100 *DI30 *0130 *0130 *0130 *0100 *0100 *DI30 *0130 *0130 *0130 *0100 *0100 *DI30 *0130 *0130 *0130 *0130 *0100 *DI30 | BIOS bUS wonstor RAH-DRYS *FR00 bus *F400 *F200 *F800 *F800 *F800 *F800 BDOS BDOS BIOS monitor monitor monitor *F400 *F200 *F800 *F800 *F800 *F800 *F800 CCP CCP BDOS BIOS resident BIOS *F800 *DF00 *CCP ACCP BDOS *F800 BDOS *F800 *DF00 *CCP BDOS resident *C60 BDOS TPA *DF00 *CCP BDOS *E400 *E400 TPA TPA *DF00 *CCP BDOS TPA TPA TPA *E400 *E400 *DC00 *CC00 CCP BDOS TPA *E400 *E400 *E400 *DFA TPA TPA TPA *E400 *E400 *E400 *E400 *CC00 COV CC00 BDOS *DIOS *DIOS *E400 *E400 *E400 *E400 *E400 | BIOS bIOS monitor RAM-DRYS FF00 bord F400 F200 FF00 FF00 FF00 FF00 BDOS BDOS BIOS monitor monitor monitor F000 F000 F000 F000 F000 F000 CCP CCP BDOS BIOS Faso Faso F000 F000 F000 Faso Faso Faso F000 F000 Faso Faso Faso Faso FPA FCC0 FDC00 Feaso Faso Faso Faso FPA FCC0 FDC00 Feaso Faso Faso <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> | | | | | | |
| *F400 *F200 *F200 *F200 *F200 *B005 BD05 B105 monitor monitor monitor *E800 *E400 *F200 *F000 *F000 *F000 *CCP BD05 BI05 resident BI05 *DF00 *DC00 *F400 *EA00 BD05 *TPA TPA CCP BD05 *EA00 BD05 *TPA CCP BD05 *EA00 BD05 *DC00 *DC00 *Ea00 TPA CCP BD05 *D00 *DC00 *Ea00 TPA CP/M CCP BD05 *0100 *0130 *0130 *0130 *0130 *0130 *0100 *0100 *0130 *0130 *0130 *0130 *0100 *0130 *0100 *0100 *0130 *0130 *0130 *0130 *0130 *0100 *0100 *0130 *0130 *0130 *0130 *0100 *0130 *0100 *0130 *0130 *0130 *0130 *0130 | *F400 *F200 *F800 *F800 <td< td=""><td>************************************</td><td>8105</td><td>bIO²</td><td>monitor</td><td>BAN+DRVS</td><td>PEEDO</td><td></td></td<> | ************************************ | 8105 | bIO ² | monitor | BAN+DRVS | PEEDO | |
| BODS BDDS BIDS COD +F200 *F000 *F000 <t< td=""><td>#DOS BDOS BIOS #DOS #FA00 *FA00 *FA</td><td>#DOS #DOS #IGS #DOS #DOS #DOS #TOSO *TOSO *TA *TPA *COSO *TA< *TA <t< td=""><td>*F400</td><td>•F200</td><td>*F 800</td><td>*F800</td><td>*F800</td><td></td></t<></td></t<> | #DOS BDOS BIOS #DOS #FA00 *FA00 *FA | #DOS #DOS #IGS #DOS #DOS #DOS #TOSO *TOSO *TA *TPA *COSO *TA< *TA *TA <t< td=""><td>*F400</td><td>•F200</td><td>*F 800</td><td>*F800</td><td>*F800</td><td></td></t<> | *F400 | •F200 | *F 800 | *F800 | *F800 | |
| *1600 *E400 *F200 *F000 *F000 *F000 CCP CCP BDOS BIOS resident BIOS *DF00 *D200 *F400 *DC00 *BOS *EA00 BDOS *DC00 *DC00 *DC00 resident *C800 TPA TPA CCP BDOS *EA00 BDOS *DC00 *DC00 *DC00 resident *C800 BNK BNK 0 1 bnk TPA *C000 C000 BNK BNK 0 1 bnk TPA BIOS 800 *010C *D100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M PG0 PG0 PG0 PG0 PG0 PG0 PG0 *010C *D100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M PG0 PG0 PG0 PG0 PG0 PG0 PG0 *6.5K 5AX 5AK 5AK 5AK 5AK 5AY 5AK * note the TPA of a banked CP/M 3.0 using standard BIOS fnon-row 1s typically A0K. *Disk controller ports SFLFCT: FU' 0ADA : status read port SFLFCT: FU' 0ADA : starer f0 SFLFCT: FU' 0ADA : starer f0 SFLFCT: FU' 0ADA : starer f0 SFLFCT: FU' 0ADA : starer control port SFLFCT: FU' 0ADA : starer f0 SFLFCT: FU' 0ADA : starer f | *E600 *F600 *F200 *F000 *F000 *F000 CCP CCP BDOS *E400 *BIOS *FA00 BDOS TPA TPA CCP BDOS *FA00 BDOS *FA00 TPA TPA CCP BDOS TFA CCP BDOS TFA *C000 *DC00 *DC00 *E400 TFA CCP BDOS TFA *C000 C000 E400 TFA TFA CCP BDOS TFA *C000 C000 E400 TFA TFA CCP BDOS TFA *C000 C000 C000 E400 TFA TFA CCP BDOS TFA *0100 *0100 *0100 *0100 *0100 *0100 TFA CFP CFP CFP CFP CFP CFP STA | *1630 *1400 *7200 *7000 *7000 *7000 *700 CCP CCP BDOS BIOS resident BIOS *0700 *0200 *1000 *0000 BDOS *0000 *0000 *0000 resident *0000 *0000 *0000 *0000 resident *0000 *0000 *0000 *0000 *0000 *000 BNK BNK 0 1 000 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M P00 P00 P00 P00 P00 P00 P00 P00 *000 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M P00 P00 P00 P00 P00 P00 P00 P00 *000 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M P00 P00 P00 P00 P00 P00 P00 P00 *000 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M CP/M P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 P00 *000 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M CP/M P00 TABUT P00 TABUT P00 TABUT P00 | | | 8105 | 800 1 F 0 F | | monit |
| CCF CCF BIOS FEADO BIOS FEADO BIOS FEADO BIOS TPA TPA CCP BDOS *EADO BDOS TPA TPA CCP BDOS *EADO BDOS TPA *DC00 *DC00 resident *C800 BDOS TPA CCP BDOS TPA *C000 C000 BHK BHK 0 1 bhK BHK 0 1 bhK BHK 0 100 *0100 *0100 CP/M CP/M CP/M CP/M Pg3 Pg4 Pg3 pg3 PG4 CFA SAK SAK STLFCT: GU SAK SAK | CCF CCF EUGS Flob Flob BLOS FEIGENT BLOS FEIGENT BLOS FEIGENT BLOS FEIGENT BLOS FEIGENT FEIGE | CCF CCP BUDS BIDS readent BIDS TPA TPA CCP BDOS *RA00 BOSS TPA TPA TPA CCP BDOS reaident *C800 TPA TPA CCP BDOS reaident *C800 TPA TPA TPA TPA *C800 BOSS TPA TPA TPA TPA *C800 BOSS *0400 *E400 TPA TPA *C800 BOSS TPA *0100 *0130 *0130 *0130 *0130 *0100 *0130 *0100 *0100 *0130 *0130 *0130 *0100 *0130 *0100 *0100 *0130 *0130 *0130 *0100 *0130 *0100 *0100 *0130 *0130 *0130 *0130 *0100 *0100 *0130 *0130 *0130 *0100 *0130 *0100 *0130 *01 | | | •F200 | • 7 0 0 0 | | *F000 |
| TPA TPA CCP BDOS *EA00 BDOS *DC00 *DC00 relationt *C800 TPA *DC00 F000 relationt *C800 TPA *D400 *E400 TPA *CC0 BDOS *D400 *E400 TPA *C000 C000 BNK BNK BNK 0 1 bnk TPA BDOS *C000 *0100 *0100 *C000 *0100 *0100 *0100 *0100 *0100 *C000 CP/M CP/M CP/M CP/M CP/M CP/M PG0 Pg0 Pg0 Pg0 Pg0 Pg0 Pg0 PG3 Pg2 Pg0 Pg0 Pg0 Pg0 Pg0 *D44 CP/M CP/M SL Cary Control *D54 Sable TPA Cary Control Control Control SG00 Pg0 Pg0 Pg0 Pg0 Pg0 Pg0 SG010 TAL Statis Cary <td< td=""><td>TPA TPA CCP BDOS *EAO BDOS *DC00 *DC00 resident *CC0 Francient *CC0 Francient *CC0 Francient *CC0 Francient *CC00 TPA *CC00 TPA *CC00 CD00 BNK Francient *CC00 CD00 BNK BNK</td><td>TPA TPA CCP BDOS *Exoo BDOS *DC00 *DC00 resident *CR00 resident *CR00 TPA CCP BDOS resident *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *DIS *DIS *DIS *DIS *DIS *0100 *DIS <td< td=""><td></td><td></td><td>s DUS</td><td>8105</td><td></td><td>BIOS</td></td<></td></td<> | TPA TPA CCP BDOS *EAO BDOS *DC00 *DC00 resident *CC0 Francient *CC0 Francient *CC0 Francient *CC0 Francient *CC00 TPA *CC00 TPA *CC00 CD00 BNK Francient *CC00 CD00 BNK | TPA TPA CCP BDOS *Exoo BDOS *DC00 *DC00 resident *CR00 resident *CR00 TPA CCP BDOS resident *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *TPA TPA TPA *CR00 TPA *D000 *DIS *DIS *DIS *DIS *DIS *0100 *DIS <td< td=""><td></td><td></td><td>s DUS</td><td>8105</td><td></td><td>BIOS</td></td<> | | | s DUS | 8105 | | BIOS |
| *DC00 *DC00 readent *C000 TPA CC00 readent *C000 TPA CC00 readent *C000 TPA TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA BIOS and BDOS *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M P30 P90 P90 P90 P90 P90 P90 *6.4K 54K 54K 54K 54K 54K 54K 54K 54K 54K 5 | *DC00 *DC00 resident *C00 TPA CCP Bn05 TPA *C000 C000 BNK BNK *C000 C000 BNK BNK 0 1 bnk TPA BIOS and BNOS *0100 *0100 *0100 *0100 *0100 *0100 *0100 *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M P30 PgC Pg0 pg0 pg0 pg0 pg0 pg0 Pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 *Sec the TPA of a banked CP/M 3.0 Using standard BIOS (non-ro is typically 60x. *Dist that/F bank switch port on SD Econoram : Disk controller ports SFT. FQC 0AD ::controller ports SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port STATS: F.' 0AD ::controller port SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port STATS: F.' 0AD ::controller port SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port SFT. FQC 0AD ::controller port STATS: F.' 0AD ::control port TD:: FQC 0AD ::timer 40 TD:: FQC 0AD ::timer 41 TD:: FQC 0AD :timer 41 TD:: FQC 02D ::parrallel in/out port CONTA:: FQC 02FF ::serial data port CONTA:: FQC 02FF ::serial control port FUCOL: EQU 02FF ::serial control port FUCOL: EQU 02FF ::serial control port SFT. FQC 02FF ::serial control port FUCOL: EQU 02FF ::serial control port | *DC00 *DC00 Feedo Feedo TPA TPA CC0 Feedo TPA *C000 C000 TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA BLOS BODS *D100 *0130 *0130 *0130 *0130 CP/M CP/M CP/M CP/M CP/M CP/M CP/M Pg3 pg0 pg0 pg0 pg0 pg0 pg0 pg0 CC/M CP/M CP/M CP/M CP/M CP/M CP/M * note the TPA of a banked CP/M 3.0 using standard BLOS (non-rom is typically A0X. *Controller prese address STATUS: F,0 0AD: ctrack port STATUS: F,0 0AD: ctrack port ST | | | | | | |
| TPA CCP BROS TPA *0400 *E400 TPA TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA BIOS and BNOS *0100 *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H CP/H Pg0 pg0 pg0 pg0 pg0 pg0 pg0 resolution to the | TPA CCP BDOS TPA *0000 *1400 TPA TPA *0000 F2400 TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA 8NK BIOS *0100 *0100 *0100 *0100 *0100 *0100 *0100 *0100 CP/H CP/M CP/M CP/M CP/M Pg3 Pg4 Pg4 Cg4 Cg4 * Dote the TPA of a banked CP/M 3.0 using standard BIOS (non-rois typically 60k. *0000 *0000 (non-rois typically 60k. * Dote the TPA of a banked CP/M 3.0 using standard BIOS (non-rois typically 60k. *0000 (non-rois typically 60k. *0000 (non-rois typically 60k. * Disk controller port of the track forman : Disk controller ports *0000 (non-rois typically 60k. * Disk controller reset address SFETCT: FU* 0A6h :sector port SFETCTS SFETCR: FU* 0A6h :sector port SFETCR: FU* 0A6h :sector command SFETCR: FU* 0A6h :sector command :secon-rois address com-rois < | TPA CCP BDOS TPA *0400 *E400 TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA *0100 *0100 *0100 *0100 CP/M CP/M SA CP/M Dohn chan Dohn <tdchan< td=""> <</tdchan<> | | | | BDOS | | |
| TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA BIOS #dd BDOS #dd BDOS *0100 *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H CP/H pg0 pg0 pg0 pg0 pg0 pg0 resource the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 40K. POLT TABLE BANK. FL OFFn ::bank switch port on SD Econoram POLT TABLE BANK. FL OFFn ::bank switch port on SD Econoram STLFCT: FU' 0 A0h ::controller ports PSFT. FU' 0 A0h ::controller ports STLFCT: FU' 0 A0h ::status read port STLFCT: FU' 0 A0h ::status read port CMD: FU' 0 A0h ::status read port CMD: FU' 0 A0h ::status read port TO: FU' 0 A0h ::status command MCCMD: FU' 0 CAPh ::write status command MCCMD: FU' 0 CAPh ::write | TPA TPA *C000 C000 BNK BNK 0 1 bnk TPA 8NK BNK 0 1 bnk TPA BIOS BIOS *010C *0130 *0130 CP/H CP/H CP/H STATIS FL OFFn : Disk controller ports SOS STATIS: FL'OFFn : Disk controller ports STATIS: FL'O AAn : statis read port STATIS: FL'O OAAn : statis read port STATIS: FL'O AAN : statis read port STATIS: FL'O AAN : statis read port STATIS: FL'O AAN : statis noraa | TPA TPA "COOD COOD BNK BNK 0 1 bnk TPA BIOS *0100 *0130 *0130 *0100 *0100 CP/H CP/M CP/M CP/M CP/M CP/M CP/M CP/M Pp3 pg6 pg0 pg0 pg0 pg0 *0.4K 4k | | | | CCP | | |
| *C000 C000 BNK BNK 0 1 bnk TPA BIOS #01 60 60 80 80 80 80 80 80 80 80 80 8 | *C000 C000 BNK BK 0 1 bnk TPA BIOS end BDOS *0100 *0100 *0100 *0100 *0100 CP/H CP/H CP/M CP/M CP/M CP/M CP/M CP/M p30 p30 p30 p30 p30 p30 p30 p30 p30 r46,4k CAk C4k C4k C4k C4F | <pre>*C000 C000 BNX BNX 0 1 bnh TPA BIOS and bNOS *0100 *0100 *0100 *0100 *0100 CP/H CP/M CP/M CP/M CP/M CP/M P90 p90 p90 p90 p90 p90 r54.6% CAK CAK CAK CAY CP/M *000 the TPA of a banked CP/M 3.0 using standard BIOS (non-row is typically 40%.</pre> | | | | | | |
| BNK BNK O 1 0 1 bnk TPA BIOS 80d BIOS *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H Pg0 pg0 pg0 pg0 pg0 Pg3 pg0 pg0 pg0 pg0 * note the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically KOK. | BNK BNK 0 1 bnk TPA BIOS and BDOS and BDOS and PG0 PG0 PG0 PG100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H PG2 PG0 PG0 PG0 PG3 PG0 PG0 PG0 CASK SAK SK SAK * note the TPA of a banked CP/M 3.0 using standard BIOS (non-rois typically 40k. touring standard BIOS (non-rois typically 40k. ************************************ | BAX BAX BAX BAX BAX BAX C 1 bank TPA BIOS BIOS BIOS BIOS *0100 *0130 *0130 *0130 *0130 *0130 *0100 CP/H | | | | TPA | TPA | |
| 0 1 bnk TPA BIOS BIOS *0100 *0100 *0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M CP/M CP/M P30 P30 P30 P31 P30 P30 CP/M CP/M CP/M CP/A CAA Sapher EA Statis Compation is a statis Compation is a statis PSFT FU OAA Statis STATTS: FU OAA Statis STATS: FU OAA Statis STAT FU | 0 1 bnk TPA BIOS and BDOS 0100 *0100 *0100 CP/M CP/M CP/M Statis PST T_O BARX F_U STATTS: F_U STATTS: F_U STATS: F_U STATS: F_U STAT | 0 1 bnk TPA BIOS *0100 *0130 *0130 *0100 *0130 *0130 *0100 *0130 *0130 *0100 CP/M CP/M CP/M CP/M CP/M PS3 Pg6 Pg0 Pg3 Pg6 Pg0 rate the TPA of a banked CP/M 3.0 using standard BIOS (non-row is typically KOK. PST. FU 0.5h PST. FU 0.5h PST. FU 0.5h STATTS: FU 0.5h STATS: FU 0.5h </td <td></td> <td></td> <td></td> <td></td> <td>•000 0000</td> <td>)</td> | | | | | •000 0000 |) |
| bak TPA BIOS and BHOS POIDC *3130 *0130 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H pg3 pg6 pg0 pg3 pg0 pg3 *64.4% Cak Cak Cak Cay Cok * note the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 40K. * note the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 40K. * Disk controller ports FSFT. FQ: 0.43h :status read port STAT'S: FQ: 0.45h :status command MACMOD EQ! CAPh :status for c | bnk TPA BIOS and BOS *0100 *0100 *0100 *0100 CP/H CP/M CP/M CP/M CP/M PB3 Pg0 Pg0 Pg0 Pg0 Pg0 Psable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA saable TPA saable TPA saable TPA saable TPA Saable TPA Saable TPA saable TPA saable TPA saable TPA Saable TPA Saable TPA Saable TPA saable TPA saable TPA Saable TPA Saable TPA Saable TPA saable TPA saable TPA | bak TPA BIOS BIOS BIOS CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM CPUM PB3 pg6 pg0 pg3 pg0 pg3 rectance the TPA of a banked CPUM 3.0 Using standard BIOS (non-row is typically 60k. | | | | | | |
| BIOS and BDOS BDOS CP/H CP/H CP/H Dale TPA CP/H CP/H CP/H Dale TPA CP/H CP/H Polot table TPA Polot table CP/T CP/H CP/T CP/H CP/T CP/T SPST. | BIOS and BDOS *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H P93 P90 P90 P90 P90 P90 P90 Factor Jaable TPA CP/H | BIOS Bind BROS *0100 *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H pg0 pg0 pg0 pg0 pg0 pg0 *00 pg0 pg0 r g0 pg0 pg0 pg0 *00 pg0 pg0 r g0 pg0 pg0 sg0 pg0 pg0 r g0 pg0 r g0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 pg0 r g0 * note the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 40K. ************************************ | | | | | bok TPA | |
| and BDOS *0100 *0130 *0100 *0100 *0100 CPVH CPVH CPVH CPVH CPVH PG0 Pg0 Pg0 Pg0 Pg0 Pg1 Sample TPA Sample TPA Sample TPA * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. * Dote the TPA of a banked CPVH 3.0 using standard BIOS (non-row is typically 40K. PDFT: FU SPETT: FU DATA: FU DATA: FU DATA: FU DATA: FU BOMI: FU DATA: FU DATA: FU DATA: FU DATA: FU DATA: FU DATA: FU DATA:< | and BDDS *0100 *0100 *0100 *0100 *0100 CPVH | and BOOS *0100 *0100 *0100 *0100 *0100 *0100 CP/H CP/H CP/H CP/H CP/H CP/H CP/H P93 P95 P90 P93 P90 P93 *0.000 the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 60K. * note the TPA of a banked CP/H 3.0 using standard BIOS (non-row is typically 60K. * Disk controller ports POLT TABLE BASK. F1 OFFN : bank switch port on SD Feoneram : Disk controller ports SFETT: FU: 0.00h :controller reset address SFETT: FU: 0.00h :status read port TACA:: F1: 0.00h :status read port TO: F1: 0.00h :status read port TO: F1: 0.00h :status read port TO: F1: 0.00h :status read port CONTA: F1: 0.00h :status read control port F1: 0.00h :status read control port F1: 0.00h :status read control port F1: 0.00h :status read port CONTA: F1: 0.00h :status read : TACA: NUT : AND A:001H :status read : TACA: NUT : TACA: NUT : TACA: NUT | | | | | | |
| BDOS 0100 0130 0000 0000 0000 0000 CP/M CP/M CP/M CP/M CP/M CP/M CP/M pg3 pg0 pg0 pg0 pg0 pg0 pg0 66.6K CAK CAK CAK CAK CAK CAK * note the TPA of a banked CP/M 3.0 using standard BIOS (non-row is typically KON. * 0100 * 0100 Controller ports PGAT TABLF * 0100 * 0100 Controller ports * 0100 PGAT TABLF * 0100 * 01000 * 01000 * 00000 SPST. FQ10000 controller ports * 010000 * 010000 SPST. FQ10000 controller ports * 0100000000000000000000000000000000000 | BDOS 0100 0130 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 001000 001000 0010000 0010000 | BDOS •0100 •0100 •0100 •0100 •0100 CP/M CP/M CP/M CP/M CP/M CP/M CP/M pg0 pg0 pg0 pg0 *4.5% Sek Sek Sek Sek Sek Sek Sey Sok * note the TPA of a banked CP/M 3.0 using standard BIOS (non-row is typically 60K. ************************************ | | | | | | |
| CPUN | CF/M | <pre>CPUN CUM CUM CUM CUM CUM CUM CUM CUM CUM STATES Jaable TPA</pre> | | | | | BBOE | |
| CPUN | CF/M | <pre>CPUN CUM CUM CUM CUM CUM CUM CUM CUM CUM STATES Jaable TPA</pre> | | • 01 00 | • 01 30 | •0100 | • 01 0 0 | •0100 |
| <pre>S4.5% Cax 54% S4% S4% Cay 60% S4% Cay 50% S4% S4% S4% S4% S4% S4% S4% S4% S4% S4</pre> | SALE SAK SAK SAK SAK SAY SAY SAY SAY SAY SAY SAY SAY SAY SAY | <pre>44.5% Sax Sqk Sqk Sqk Sqk Sqk Sqk Sqk Sqk Sqk Sqk</pre> | C P/M | CP/M | CP/M | CP/M | (+/ - | 1 CF/R |
| DOLE the TPA of a banked CP/M 3.0 using standard BIOS (non-rominative specially 408. POLT TABLE POLT TABLE DANK. FU OFFN : bank switch port on SD Econoram DISK controller ports PSET. FU 0 A0A : controller cest address STLFCT: FU 0 A0A : status read port TAKCA: FU 0 A0A : status read port STATTS: FU 0 A0A : status read port STATTS: FU 0 A0A : status read port DATA: FU 0 AAA : status read port SECTOR. FU 0 AAA : status read port CADA : command write port CADA : FU 0 AAA : command write port CADA : FU 0 AAA : command write port CADA : command write port CADA : write status command RDCM: FU 0 CAAA : write status command MACMOD EU' 0 CAAA : write status command COMPULATE CP! ports TO: FU 0 CAAA : timer 40 TO: FU 0 CAAA : timer 41 TO: FU 0 CAAA : timer 41 TO: FU 0 CAAA : timer 40 TO: FU 0 CAAA : timer 40 TO: FU 0 CAAA : timer 40 CONTAL: FU' 0 CAAA : stare control port | note the TPA of a banked CP/M 3.0 using standard BIOS (non-rolis typically 60k. PULT TABLE PULT TABLE Disk controller ports PSFT. FUL 0FFn : Dank switch port on SD Econoram Disk controller ports PSFT. FUL 0A0h : controller reset address STLFTT: FUL 0A0h : status read port STATTS: FUL 0A0h : status read port SFCTOR. FUL 0A0h : status read port SFCTOR. FUL 0A0h : status read port SFCTOR. FUL 0A0h : status read port CONTAL FUL 0A0h : status command RDCM: FUL 0A0h : status track command RDCM: FUL 0A0h : stimer 40 TOIL FUL 020h : timer 41 TOIL FUL 020h : timer 43 TOIL FUL 020h : timer 43 TOIL FUL 020h : stimer 100t port CONTAL FUL 020h : stimer 100t port | <pre>* note the TPA of a banked CP/M 3.0 using standard BIOS (non-rom</pre> | pgJ | pgC | pg0 | P9.) | pg 0 | Pg 0 |
| note the TPA of a banked CP/M 3.0 using standard BIOS (non-rominative splically 408.) POLT TABLE POLT TABLE DANK. FU OFFn : bank switch port on SD Econoram STATTS: FU 0.40h :controller ports PSFT. FU 0.40h :controller test address STATTS: FU 0.40h :status read port TAKCA: FU 0.40h :status read port STATTS: FU 0.40h :status read port CADA : FU 0.40h :status read port CADA : FU 0.40h :status read port CADA : command write port MECMU: FU 0.40h :write status command MECMU: FU 0.40h :timer 41 TOTL FU 0.40h :timer 43 TOTL FU 0.40h :status control port CONTAL :FU 0.40h :timer 43 CONTAL :FU 0.40h :timer 43 CONTAL :FU 0.20h :timer 40 CONTAL :FU 0.20h :stare control port | note the TPA of a banked CP/M 3.0 using standard BIOS (non-rolis typically 40%. PL-T TAULF PARK: F1 0FFh : bank switch port on SD Feonoram : Disk controller ports PSFT. F0 0+0h : controller reset address STLFGT: F0 0+0h : status read port TAKCA: F1 0+0h : status read port STATS: F1 0+4h : status read port STATS: F1 0+4h : status read port CMD: F1 0+4h : command write port CMD: F1 0+4h : read address command RDCM: F1 0+4h : read address command RDCM: F1 0+4h : command write port CMD: F1 0+4h : read address command RDCM: F1 0+4h : read sector command RDCM: F1 0+4h : write sector command RTCM: F1 0+4h : write sector command RTCM: F1 0+4h : trans the track command RTCM: F2 024h : timer 41 T2: F20 024h : timer 43 TCTL F1 024h : trans control port NUTT: F1 024h : sectal control port CONTAL FU 024h : sectal control port | <pre>* note the TPA of a banked CP/M 3.0 using standard BIOS (non-rom</pre> | 56 5V | | siv | TPA | | |
| PORT TABLE PORT TABLE BANK. FL OFFN :Dank switch port on SD Feonoram : Disk controller ports PSFT. FQ! OADH :controller ports STUFCT: FQ! OADH :drive select port STATTS: FQ! OADH :status read port TRACN: FQ! OADH :status read port TRACN: FQ! OADH :status read port DATA: FQ! OADH :status read port DATA: FQ! OADH :status read port DATA: FQ! OADH :status read address command ROCM: FQ! OADH :read address command ROCM: FQ! OADH :read address command RACMD: FQ! OIDH :read a | PLAT TABLE PLAT TABLE bANK. FL OFFN : bank switch port or SD Econoram : Disk controller ports PSFT. FQL 0+0: controller ports STUFCT: FQL 0+0: status read port TRACK: FL 0+4: status read port TRACK: FL 0+4: status read port DATA: FL 0+4: command write port DATA: FL 0+7: clata in/out port CHD: FQL 0+4: command write port BCMC: FQL 0+4: command write score non- BCMC: FQL 0+4: cread address command MRCMD: CRCMD: cread address command MRCMD: CRCMD: cread address command MRCMD: cread address com | <pre>ls typically 40k. PC-T TABLF bARK. FL OFFN :Dask Switch port on SD Fconoram : Disk controller ports SFLFCT: FU! 0ADA : controller reset address SFLFCT: FU! 0AAA : status read port TRACK: FL! 0AAA : status read port TRACK: FL! 0AAA : status read port DATA: FL! 0AAA : command write port DATA: FL! 0AAA : command write port BCCML: FU! 0CCA : read address command MRCMD: FL! 0AAA : command write sector command MRCMD: FL! 0AAA : command write sector command MRCMD: FL! 0AAA : read sector command MRCMD: FL! 0AAA : stimer #0 TO: FL! 0AAA : timer #0 TO: FL! 0AAA : timer #1 TC: FL! 0CAA : timer #1 TC: FL! 0CAA : timer #1 TC: FL! 0CAA : sectal control port NUMT: FL! 0CAA : sectal control port SCONTUR: FL! 0CAA : sectal control port FL! 0CA : sectal control port FL! 0CAA : sectal control port FL! 0CAAA : sectal control port FL! 0CAAA : sectal control port FL! 0CAAA : sectal control port FL! 0CAAAAA FL! 0IA : sectal control port FL! 0CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</pre> | • note t | he TPA of | A DARKED CP/ | M 3.0 Jac | no standard E | 105 (non-rom |
| PLAT TABLE bARK. FL 0FFH : bank switch port on SD Fconoram : Disk controller ports SPST. FU: 0AD ::controller reset address SFLFCT: FU: 0AD ::stave select port STATES: FU: 0AD ::status read port TACA: FU: 0AD ::status read port SFCTOR. FU: 0AD ::status read address command ROCM: FU: 0AD ::read address command ROCM: FU: 0AD ::read sector command ROCM: FU: 0AD ::timer \$3 ROCM: FU: 0AD ::read sector command ROCM: FU: 0AD ::read sector command R | $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | PORT TABLE BARK. FL OFFN ::Dank switch port on SD Fconoram PSFT. FU! 0x0n ::Controller perts PSFT. FU! 0x3n ::drive select port STATTS: FU! 0x3n ::drive select port STATTS: FU! 0x4n ::status read port TACA: FU! 0x4n ::status read port STATTS: FU! 0x4n ::status read port STATTS: FU! 0x4n ::status read port CMD: FU! 0x4n ::status command MACMD: FU! 0x4n ::status read command MACMD: FU! 0x4n ::status read command MACMD: FU! 0x4n ::status read command MACMD: FU! 0x4n ::status command TD: FU! 0x4n ::timer 0 T1: FU! 0x4n :timer 0 T1: FU! 0x4n :timer 0 T1: FU! 0x4n :status read CONTTA: FU! 0x4n :status read CONTTA: FU! 0x4n :status read FU! 0x5n ::serial data port CONTTA: FU! 0x4n :serial data port CONTTA: FU! 0x1n :serial control port FU! 0x1n :serial cont out status mage FU! 0x1n :serial control port FU! 0x1n :serial control port FU! 0x1n :serial control port FU! 0x1n :serial cont out status mage FU! 0x1n :serial control port FU! 0x1n :serial | 18 | typically | 60K. | | ing scendera a | |
| PG-T TABLE BARK. FL 0FFh :bank switch port on SD Fconoram : Disk controller ports :Disk controller ports PSFT. FU: 0+3h :controller reset address STLFCT: FU: 0+3h :dive select port STAT'S: FU: 0+3h :status read port TACA: FU: 0+3h :status read port DATA: FU: 0+3h :status read port DATA: FU: 0+3h :status read port CMD: FU: 0+3h :status read port DATA: FU: 0+3h :status read port CMD: FU: 0+4h :status read port CMD: FU: 0+4h :status read RDCM: FU: 0+4h :write stack command RTCMD: FU: 0+4h :write stack command T1: FU: 0+2h | $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | PD-T TABLE bANK. F. OFFn ::Dank switch port on SD Fconoram : Disk controller ports PSFT. F.O 040n ::Controller reset address STATTS: F.O 040n ::Controller reset address STATTS: F.O 040n ::Status read port TACA: F.O 040n ::Status read port STATTS: F.O 040n ::Status read port DATA: F.O 040n ::Status read port CMD: F.O 040n ::Status read port TO: F.O 020n ::Imer 01 TO: F.O 020n ::Imer 01 TO: F.O 020n ::Imer 01 TO: F.O 020n ::Imer 01 TO: F.O 020n ::Imer 02 TO: F.O 020n ::Status read CONCTU: F.O 020n ::Status read CONCTU: F.O 020n ::Status read F.O 01n :Status read F.O 01n :Status read F.O 01n :Status read . Imer 10 :Status read . Imer 10 :Status read . Imer 10 :Status read | | | | | | |
| PSFT. FUI 0+0h :controller ports SFLFCT: FUI 0+0h :controller reset address SFLFCT: FUI 0+3h :citive select port STATIS: FUI 0+4h :status read port TARCA: FUI 0+4h :status read port SFCTOR. FUI 0+4h :status read port SFCTOR. FUI 0+4h :status read port DATA: FUI 0+4h :status read port DATA: FUI 0+4h :status read address command NCMD: FUI 0+4h :command write port ROCMD: FUI 0+4h :read address command NCCMD: FUI 0+4h :read address command CONDTA: F | PSFT. FUI 0ADH : Disk controller ports SFLFCT: FUI 0ADH : controller reset address SFLFCT: FUI 0ADH : status read port TARCA: FUI 0ADH : status read port TARCA: FUI 0ADH : status read port TARCA: FUI 0ADH : status read port SFCTOR. FUI 0ADH : status read port CHD: FUI 0ADH : status read port CHD: FUI 0ADH : status read address command RCM: FUI 0ADH : treef 40 TO: FUI 0ADH : timer 41 TO: FUI 0ADH : timer 43 TOIL FUI 0ADH : timer control port CONCTA: FUI 0ADH : serial control port CONCTA: FUI 0AFH : serial control port CONCTA: FUI 0AFH : serial control port FUI 0AFH : serial control port | PSFT. FUI 0x01 :controller ports SFLFCT: FUI 0x01 :controller reset address SFLFCT: FUI 0x31 :status read port TARCA: FUI 0x41 :status read port TARCA: FUI 0x41 :status read port SFCTOR. FUI 0x41 :status read port DATA: FUI 0x71 :data in/out port CMD: FUI 0x41 :scator port DATA: FUI 0x41 :scator command MECMD: FUI 0x41 :scator command MECMD: FUI 0x41 :status score command MECMD: FUI 0x41 :status read scotor command MECMD: FUI 0x41 :status read scotor command MECMD: FUI 0x41 :status read command MECMD: FUI 0x41 :status read command MECMD: FUI 0x41 :status read command TO: FUI 0x41 :status command CONSTA: FUI 0x41 :status command CONSTA: FUI 0x41 :status control port TU: FUI 0x41 :status control port FUI 0x41 :sectal con | | PCHT TA | 8LF | | | ••••• |
| Disk controller ports PSFT. EQ: 0.00 controller reset address STLTTS: EQ: 0.40 controller reset address STLTTS: EQ: 0.44 status read port TRACA: EQ: 0.46 status read port SECTOR. EQ: 0.46 status read port SECTOR. EQ: 0.46 command write port CHO: EQ: 0.46 command write port ROCM: EQ: 0.46 command write sector command ROCM: EQ: 0.46 command write sector command ROCM: EQ: 0.46 command write sector command ROCM: EQ: 0.46 compatible controller compatible controller compatible controller compatible controller controller <licontroller< li=""> controller controller</licontroller<> | Disk controller ports PSFT. FUI 0ADN : controller ports STEFCT: FUI 0ADN : drive select port STATTS: FUI 0ADN : status read port TRACA: FUI 0ADN : status read port SECTOR. FUI 0ADN : status read port SECTOR. FUI 0ADN : status read port ROCH: FUI 0ADN : status read adress command ROCH: FUI 0ADN : read adress command ROCH: FUI 0ADN : read adress command ROCH: FUI 0ADN : status read adress command ROCH: FUI 0ADN : status read adress command ROCH: FUI 0ADN : state status read ROCH: FUI 0ADN : state status command ROCH: FUI 0ADN : state control port CONCTA: FUI 02FN : serial control port FUI 0ADN : serial control port FUI 0ADN : serial control port | <pre>PSFT. EV: 0.Non ::Controller ports PSFT. EV: 0.Non ::controller ports PSFT. EV: 0.Non ::drave select port STLTES: EV: 0.Non ::track port TRACA: EV: 0.Non ::track port DATA: FV: 0.Non ::data in/out port DATA: FV: 0.Non ::data in/out port CND: FV: 0.Non ::read address command ROCML: FV: 0.Non ::read address command ROCML: FV: 0.Non ::read sector command WRCMD: FV: 0.Non ::read sector command WRCMD: FV: 0.Non ::track sector command WRCMD: FV: 0.Non ::track command ROCML: FV: 0.2No ::trace f0 TO: FV: 0.2No ::trace f0 TO: FV: 0.2No ::trace f1 TO: FV: 0.2No ::trace information FV: 0.2No ::trace f1 TO: FV: 0.2No ::trace f2 FV: 0.2No ::trace f3 TO: FV: 0.2No ::trace</pre> | BANK . | F_ 0 F | | switch p | ort on SD Fee | noram |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | 3FLFCT: FU! 0.4h; status read port TRACk: FU! 0.4h; status read port TRACk: FU! 0.4h; status read port SFCTOR: FU! 0.4h; status read port DATA: FU! 0.4h; status read port DATA: FU! 0.4h; status read port CMD: FU! 0.4h; status read port CMD: FU! 0.4h; status read port ROCM: FU' 0.4h; status read WRTOND: FU' 0.4h; status read ROCM: FU' 0.4h; status read TO: FU' 0.2h; stimer fl TO: FU' 0.2h; stimer fl TO: FU' 0.2h; stimer lanyout CONCTA: FU' <td>SPLET: Full 0.83h :drive select port STATUS: Full 0.84h :status read port TRACK: Full 0.84h :status read port SPETCE: Full 0.84h :status read port DATA: Full 0.84h :status read port ROCM: Full 0.84h :status read port ROCM: Full 0.84h :status read port TO: Full 0.84h :status read port TO: Full 0.24h :write sector command HRTCND: Full 0.24h :timer 40 TO: Full 0.24h :timer 41 TO: Full 0.24h :timer 41 TO: Full 0.24h :timer 41 TO: Full 0.24h :stare control port TO: Full 0.25h :stare CONTTA: Full 0.25h :stare Full 0.27h :stare :stare <</td> <td></td> <td></td> <td>: D19</td> <td>k control</td> <td>ler ports</td> <td></td> | SPLET: Full 0.83h :drive select port STATUS: Full 0.84h :status read port TRACK: Full 0.84h :status read port SPETCE: Full 0.84h :status read port DATA: Full 0.84h :status read port ROCM: Full 0.84h :status read port ROCM: Full 0.84h :status read port TO: Full 0.84h :status read port TO: Full 0.24h :write sector command HRTCND: Full 0.24h :timer 40 TO: Full 0.24h :timer 41 TO: Full 0.24h :timer 41 TO: Full 0.24h :timer 41 TO: Full 0.24h :stare control port TO: Full 0.25h :stare CONTTA: Full 0.25h :stare Full 0.27h :stare :stare < | | | : D19 | k control | ler ports | |
| STATUS: EV 0445 :status read port TARCA: EV 0465 :status read port SFCTOR. EV 0465 :status/outport DATA: FV 0475 :command write port ROCM: FV 0255 :read address command ROCM: FV 0255 :read address command ROCM: FV 0475 :read address command TO: FV 0475 :read address command ROCM: FV 0475 :read address command CONTA: FV 0475 :read address common CONTA: FV 0475 :read address common CONTA: FV 0475 :read address common CONTA: FV 0475 :read address common FV 0475 :read address common CONTA: FV 0475 :read address common CONTA: FV 0475 :read address common CONTA: FV 0475 :read address common FV 0475 :read address common CONTA: FV 0475 :read address common FV 0475 :rea | STATUS: E. 0.4n :status read port TACA: E. 0.6n :track port SFCTOR. E. 0.6n :track port DATA: F. 0.7n :data infold port CMD: F. 0.4n :command write port RDCM: F. 0.4n :cread address command RDCM: F. 0.4n :read address command RDCM: F. 0.4n :read address command RDCM: F. 0.4n :vrite track command RDCM: F. 0.4n :vrite track command RTCM: F. 0.4n :vrite track command RTCM: F. 0.4n :vrite track command RTCM: F. 0.4n :vrite track command TD: F. 0.4n :track command TD: F. 0.4n :timer 40 TD: F. 0.2n :timer 41 TC: F. 0.4n :timer 43 TCTL F. 0.4n :timer 43 TCTL F. 0.4n :serial data port CONCTA: F. 0.4n :serial control port F. 0.4n :serial control port | STATES: F. 0.44h : status read port TACA: F. 0.04h : status read port TACA: F. 0.04h : status read port SFCTOR. F. 0.44h : status port CMD: F. 0.44h : command write port RDCMI. F. 0.44h : read sector command RCCMI. F. 0.44h : write status command MACMOD F. 10.62h : write status command MACMOD F. 10.62h : write status command MACMOD F. 10.62h : timer 41 TO: F. 0.23h : timer 41 TCL. F. 0.12h : timer 43 TCL. F. 0.12h : starer 61 TCL. F. 0.12h : starer 63 TCL. F. 0.12h : starer 64 TCL. F. 0.12h : starer 64 TCL. F. 0.12h : starer 64 F. 0.10h : stares will new channin. 14h F. 0.10h : stares will stares will new channin. 14h F. 0.10h : stares will | | Fý On | on ;cont | roller re | set address | |
| TRACK: Full 0.5h :track port SFCTOR. Full 0.5h :sector port DATA: Full 0.5h :sector port DATA: Full 0.5h :sector port DATA: Full 0.5h :sector port RDCML: Full 0.5h :sector command RDCML: Full 0.5h :sector command WRCMD: Full 0.5h :sector command WRCMD: Full 0.5h :sector command WRCMD: Full 0.5h :sector command WRCMD: Full 0.2h :sector command T3: Full 0.2h :stimer 40 T3: Full 0.2h :stimer 40 TCTL: Full 0. | TRACK: Fit Onth: strack port SFCTOR: Fit Onth: sector port DATA: Fit Onth: sector port CMD: Fit Onth: scaladiness command RDCMI: Fit Onth: scaladiness command RDCMI: Fit Onth: scaladiness command RDCMI: Fit Onth: scaladiness command WRCMD: Fit Onth: scaladiness command WRCMD: Fit Onth: scaladiness command WRCMD: Fit Onth: scaladiness command WRCMD: Fit Onth: scaladiness command The fit Onth: scaladiness command The fit Onth: scaladiness command The fit Onth: scaladiness command The fit Onth: scaladiness command TOTL: Fit Olden stimer fit TCTL: Fit Olden stimer fit TC | <pre>TRACK: F_t 0.5h :treek port SFCTOR. F_t 0.5h :deta in/out port DATA: F_t' 0.5h :deta in/out port CHO: F_t' 0.4c :command write port RDCMI: F_t' 0.4c :command write port RDCMI: F_t' 0.4c :command write sector command WRCMU: F_t' 0.4c :tread address command WRCMU: F_t' 0.4c :computate CPU ports T0: F_t' 0.4c :computate CPU ports T1: F_t' 0.2h :timer #0 T2: F_t' 0.2h :timer #1 T2: F_t' 0.2h :stimer for T0: F_t' 0.2c :persile in/out port CONTA: F_t' 0.2F :secial data port CONTA: F_t' 0.2F :secial control port F_t' 0.2F :secial con in status mask F_t'' 0.1n :secial con out status mask F_t'' 0.1n :secial con secient files already i on f' ouse with deferent files already i on f' ouse with openentative.</pre> | STATUS: | F 1 0 4 | an jariv | | | |
| $\begin{array}{rcl} SFCTOR. & FQ^{(r)} & 0.46h & :sector port \\ DATA: & FQ^{(r)} & 0.4ch & :command write port \\ CHO: & FQ^{(r)} & 0.4ch & :command write port \\ RDCM(: & FQ^{(r)} & 0.4ch & :read address command \\ RDCM(: & FQ^{(r)} & 0.4ch & :read sector command \\ RDCM(: & FQ^{(r)} & 0.4ch & :write sector command \\ RTCM(: & FQ^{(r)} & 0.4ch & :write track command \\ RTCM(: & FQ^{(r)} & 0.4ch & :write track command \\ RTCM(: & FQ^{(r)} & 0.4ch & :write track command \\ RTCM(: & FQ^{(r)} & 0.4ch & :write track command \\ RTCM(: & FQ^{(r)} & 0.4ch & :timer 40 \\ T1: & FQ^{(r)} & 0.4ch & :timer 40 \\ TC1: & TQ1: & TQ1: \\ TC1: & FQ^{(r)} & 0.4ch & :timer 40 \\ TC1: & TQ1: & TQ1: \\ TC1: & TQ1: & TQ1: \\ TC1: & TQ1: & TQ1: \\ TC1: & T$ | SFCTOR. $F_{Q}^{(r)}$ 0.45h :sector port DATA: $F_{1}^{(r)}$ 0.45h :command write port CMD: $F_{2}^{(r)}$ 0.4cr :command write port RDCMI: $F_{2}^{(r)}$ 0.4cr :read address command RDCMI: $F_{2}^{(r)}$ 0.4ch :read sector command WRCMD: $F_{2}^{(r)}$ 0.4ch :write track command WRCMD: $F_{2}^{(r)}$ 0.4ch :write track command TO: $F_{2}^{(r)}$ 0.4ch :write track command TO: $F_{2}^{(r)}$ 0.2ch :timer 40 T1: $F_{2}^{(r)}$ 0.2ch :timer 41 TCIL $F_{2}^{(r)}$ 0.2ch :timer 43 TCTL $F_{2}^{(r)}$ 0.2ch :timer control port NUTT: $F_{1}^{(r)}$ 0.2ch :parallel in/out wort CONCTA: $F_{2}^{(r)}$ 0.2ch :secial data port CONCTA: $E_{2}^{(r)}$ 0.2ch :secial control port $F_{2}^{(r)}$ 0.2ch :secial control port | SFCTOR. FQF 044h :sector port DATA: F,F 0.5h :data in/out port CMD: F,F 0.5h :data in/out port RDCMI. F,F 0.5h :read address command RDCMI. F,F 0.5h :read sector command WRCMD: F,F 0.5h :write sector command WRCMD: F,F 0.5h :write sector command MRCMD: F,F 0.5h :write sector command TD: FQF 0.2h :timer 41 TD: FQF 0.2h :timer 41 TD: FQF 0.2h :timer 41 TD: FQF 0.2h :timer 41 TD: FQF 0.2h :timer 41 TC: FQF 0.2h :timer 41 CONCTL: FQF 0.2h :sectal data port CONCTL: FQF 0.2h :sectal data port CONCTL: FQF 0.2h :sectal control port FQF 0.2h :sectal control starts mage FQF 0.1h :sectal control starts mage | TRACK: | F 0 0 | in jatat | Te tear h | 510 | |
| DATA: F ⁽¹⁾ 0+7H (data in/out port CMD: F ⁽¹⁾ 0+4H (command write port RDCMI: F ⁽¹⁾ 04H (command write sector command RDCMI: F ⁽¹⁾ 04H (command write sector command WRCMD: F ⁽¹⁾ 074H (command write sector command WRCMD: F ⁽¹⁾ 074H (command write sector command T0: F ⁽¹⁾ 074H (command write sector command T1: F ⁽¹⁾ 02H (command write sector command T1: F ⁽¹⁾ 02H (command sector command sector command T1: F ⁽¹⁾ 02H (command sector command sector | DATA: F." 0ATh ::data in/out port CMD: F.' 0AA: :command write port RDCMI: F.' 0C3r ::read address command RDCMI: F.' 0'44: :read address command RDCMI: F.' 0'44: :read address command WRCMD: F.' 0FAn :write sector command WRCMD: F.' 0FAn :write sector command T3: F.' 0FAn :write sector command T3: F.' 024h :timer \$0 T1: F.' 024h :timer \$1 TC1: F.' 024h :timer \$1 CONCTA: F.' 024F :serial control port CONCTA: F.' 024F :serial control port F.' 027 : serial control port | DATA: F.P. 0+74 :data infoit port CHO: F.P. 0+4: :command write port RDCMT: F.P. 0+4: :read address command RDCMT: F.P. 0+4: :read address command RCMD: F.P. 0+4: :read sector command WRCMD: F.P. 0+4: :read sector command WRCMD: F.P. 0+4: :read sector command T. F.P. 0+4: :write sector command T. F.P. 0+4: :read sector command T. F.P. 0+5: :sectal data port CONTTA: F.P. 0+7: :sectal data port F.P. 0+7: :sectal con in status mask F.P. 0+7: :sectal con in status mask F.P. 0+7: :sectal con out status | | F.C. 0.6 | th react | or port | | |
| CMD: FU' DA4r :command write port BCPKL: FU' DC5r :read address command RDCML: FU' DC4r, :read address command RDCML: FU' DC4r, :read address command WRCMD: FU' DC4r, :write sector command WRCMD: FU' DC4r, :write sector command TD: FU' DC4r, :write sector command TD: FU' DC4r, :write sector command TD: FU' DC4r, :stimer 40 TC1L: FU' DC4r, :timer 40 TC1L: FU' DC4r, :timer 40 TC1L: FU' DC4r, :timer 40 TC1L: FU' DC4r, :stimer control port TC1L: FU' DC4r, :sectal control port CONCTU: FU' DC4r, :sectal control port FU' DC4r, :sectal control port | CMD: FJ' 044r ;command write port BOCML: FJ' 024r ;read address command ROCML: FJ' 044r ;read address command ROCML: FJ' 044r ;read address command WRCMD: FJ' 044r ;write sector command WRCMD: FJ' 074n ;write sector command TO: FJ' 074n ;write sector command TO: FJ' 074n ;timer 40 TC: FJ' 029h ;timer 41 TC: FJ' 020h ;timer 43 TCTL FJ' 020h ;timer 63 TCTL FJ' 020h ;timer 63 TCTL FJ' 020h ;timer 64 TCTL FJ' 020h ;timer 64 TCTL FJ' 020h ;timer 63 TCTL FJ' 020h ;timer 64 TCTL FJ' 020h ;timer 60h ;timer 64 TCTL FJ' 020h ;timer 64 TCTL | CHO: FQ: 0.44: :command write port ROCML: FQ: 0.55: :read address command ROCML: FQ: 0.44: :read address command ROCML: FQ: 0.44: :read address command WRCMD: FQ: 0.44: :read sector command : Compatime CPU ports TD: FQ: 0.29: :timer 40 TD: FQ: 0.29: :timer 41 TC: FQ: 0.29: :timer 43 TCTL: FQ: 0.27: :sectal data port CONTTAL FQ: 0.27: :sectal control port FQ: 0.27: :sectal control port FQ: 0.27: :sectal control port FQ: 0.27: :sectal control mask FU: 01: :sectal control mask FU: 01: :sectal con out statis mask FU: 01: :sectal con out statis mask FU: 01: :sectal con out statis mask FU: 01: :sectal control port : For the SNSysters "Oblu" supplied in the : Table two actas will need Chantin. 1: is : suggested that SL: M disk fL: := utaine: as : it will give you the different files already : OF PT disk wit: docimentation. : ID CHOTYP: clange to below code : LDA DAI: :SKIF :: CLFA bIT 7 : ANA Ay071: :CLFA bIT 7 : ANA Ay071: :CLFA bIT 7 : TAT AVAD | | F. P. 0 M | 'ni :data | in/out a | ort | |
| RDCM1: Full CC3F :read address command RDCM1: Full CAPh :read sector command WRCMD1: Full CAPh :write sector command WRCMD1: Full CAPh :write sector command WRCMD1: Full CAPh :write sector command T0: Full CAPh :write frack command T0: Full C2Ph :timer #0 T1: FUll C2Ph :timer #1 TC1: Full C2Ph :timer control port TCT1: Full C2Ph :timer control port TCT1: Full C2Ph :timer d2Ph CONTA: Full C2Ph :serial data port CONTA: Full C2Ph :serial control port Full C2Ph :serial control n status | RDCM1: Fu' 0.24t; :read sector command RDCM1: Fu' 0.4t; :read sector command WRCMD1: Fu' 0.4t; :read sector command WRCMD1: Fu' 0.4t; :read sector command WRCMD1: Fu' 0.4t; :read sector command TCTU1: Fu' 0.2t; :compating CP' ports T0: Fu' 0.2t; :timer #0 T1: Fu' 0.2t; :timer #1 TC1: Fu' 0.2t; :pirraile1 in/dat port T0: Fu'' 0.2t; :pirraile1 in/dat port CONTTA: Fu'' 0.2t; :serial control port Fu'' 0.2t; :serial control port :pirraile0 in/dat port | <pre>RDCM1: FU CC3F :read address command RDCM1: FJ 044: :read sector command WRCMD1 FJF 0F4n :write sector command WRCMD1 FJF 0F4n :write track command : Compatime CPF ports T3: FU 024h :timer 40 T3: FJF 024h :timer 41 T3: FJF 024h :timer 41 T3: FJF 024h :timer 43 TCTL: FJF 024h :parallel in/out port CONCTU: FJF 024F :sectal data port CONCTU: FJF 024F :sectal control port FJF 02F :sectal control port FJF 02F :sectal control port FJF 02F :sectal control supplies in the FJF 01h :sectal control supplies in the T4: suppested that SIF dust supplies in the T4: suppested that SIF dust status mask FJF 01h :sectal control supplies in the T4: suppested that SIF dust status mask FJF 01h :sectal control supplies in the T4: suppested that SIF dust status mask : it will give you the different files sized : it will give you the different files sized : if AF 05T : AF 05T : clarge to below code : if AF 05T : AF 4,05T : CLFA bit 7 : AF 4,05T : C</pre> | | | t: ;comm | and write | port | |
| <pre>whichu: /u: OP4n :white track commann : Computame CPU ports T0: EUC 029h :timer #0 T1: F0U 029h :timer #1 T2: EUL 024h :timer #1 TCTL: FUU 025h :timer control port TCTL: FUU 025h :timer control port EUC 02Fh :serial data port CONCTU: EUL 02Fh :serial control port FUU 02Fh :serial control port FUU 02Fh :serial control meak</pre> | <pre>Mileu: J OFAD</pre> | <pre>Milcu: /u of 4: :witte track comman:</pre> | RDCM: . | FG1 60. | in iread | address | commana | |
| <pre>whichu: /u: OP4n :white track commann : Computame CPU ports T0: EUC 029h :timer #0 T1: F0U 029h :timer #1 T2: EUL 024h :timer #1 TCTL: EUL 024h :timer control port TCTL: EUC 025h :timer control port EUC 025h :serial data port CONCTU: EUC 025h :serial control port EUC 025h :serial control port EUC 025h :serial control meak</pre> | <pre>Mileu: J OFAD</pre> | <pre>Milcu: /u of 4: :witte track comman:</pre> | RDCMC | F.' 0' | n :read | sector o | ommand | |
| : Complaime CPU ports : Complaime CPU ports T1: EQU 023h :timer 40 T2: EQU 024h :timer 41 T2: EQU 024h :timer control port INUTT: EQU 024h :timer control port CONTA: EQU 025h :serial data port CONTA: EQU 025h :serial data port EQU 025h :serial control port EQU 027h :serial control port | : Compliane CPP ports T3: EQU 029h :timer 40 T1: EQU 029h :timer 41 T2: EQU 02Ah :timer 43 TCTL: EQU 02Ah :timer control port NUMT: EQU 02Ah :timer control port CONTA: EQU 02Ah :serial data port CONTA: EQU 02Ah :serial control port EQU 02Ah :serial control port EQU 02Ah :serial control port | <pre>: Complaime CPP' ports : Complaime CPP' ports T1: FU' 029h :timer #1 T2: FU' 029h :timer #1 T2: FU' 029h :timer #1 T2: FU' 029h :timer control port NUTT: FU' 029h :timer control port CONCTA: FU' 029h :serial data port CONCTU: EU 029h :serial control port FU' 021 :serial control port FU' 01h :serial control supplies in the</pre> | | EQU CA | h ywrit | | | |
| T0: EUK 023h :timer #0 T1: F0H 029h :timer #1 T2: F0H 029h :timer #3 T0TL: FUH 026h :timer control port T0HT: FUH 026h :prrallelin/sutport CONTA: F0H 02Fh :serial data port CONTU: EUH 02Fh :serial control port FUH 02Fh :serial control port FUH 02Fh :serial control meak | T0: EuC 022h ;timer #0 T1: FCC 029h ;timer #0 T2: FCC 029h ;timer #3 TCTL FLC 02h ;timer #3 TCTL FLC 02h ;timer Control port TCTL FLC 02Fh ;serial control port FCC 02Fh ;serial control port FLC 02- ;serial cont n statis mask | <pre>To: EQC 0.28h :timer #0 To: FQC 0.28h :timer #0 To: FQC 0.28h :timer #1 To: FQC 0.28h :timer #1 To: FQC 0.26h :parrallel in/Dut port EQUATA: FQC 0.26h :serial data port CONCTA: FQC 0.26h :serial con in status mask FQC 0.1 iserial con out status FQC 0.1 iserial</pre> | WRICHDE | 8121 OF | in jwrit | e track o | om man : | |
| T1: F0 ^H 02Ph ;timer #1 T2: E0H 01Ah ;timer #3 TCTL F0H 02Ah ;timer control port INUHT: F0H 02Ch ;parrallel in/cut port CONTA: F0H 02Fh ;serial data port CONCTU: E0H 02Fh ;serial control port F0H 02Ch ;serial control port | T1: F0 ¹⁰ 0.29h ;timer \$1 T2: EUC 0.1Ah ;timer \$3 TCTL FUC 0.1Ah ;timer control port INUTT: EUC 0.20h ;parrallel in/out port CONTTA: FUC 0.2Fh ;serial data port CONTTU: EUC 0.2Fh ;serial control port FUC 0.2Fh ;serial control port FUC 0.2Fh ;serial control mask | <pre>T1: F0* 029h :timer #1 T2: EU 02h :timer #1 T2: EU 02h :timer #3 T0: F0* 02h :parallel in/out port CONTA: EU 02Fh :secial data port CONTA: EU 02Fh :secial control port F1* 02r :secial control port F1* 02r :secial con out stars mask F1* 01h :secial con out</pre> | * 1 · | F/ 11 0.2 | : LOM | putime Le | · porce | |
| T2: EUL OCAH :timer 83 TCTL: Full Olah :timer control port TUUTT: Full Olah :parrallel in/cut port CONTTA: FULL 02FH :serial data port CONCTU: EUL 02FH :serial control port Full 02F :serial control port | T2: EU 01Ah :timer 43 TCTL: Ful 02Ah :timer control port TUUTT: Ful 02Ah :parrailei in/out wort CONTMA: FUL 02Fh :serial data port CONCTU: EU 02Fh :serial control port Ful 02- :serial con in statis mask | <pre>T2: EJL 01Ah :timer 43 TCL: EJL 01Ah :timer control port INUMT: EJL 02Ah :parrallel in/dit port CONTTA: EJL 02Fh :serial data port CONTTA: EJL 02Fh :serial control port EJL 02Fh :serial control port EJL 01h :serial con out status mask EJL 01h :serial con out status mask : EJL 01h :serial con out status mask : EJ</pre> | ту. ту. | | | | | |
| INUMT: FUT GUCH (permalle) in/out port CONTA: FUT GUFH (serial data port CONTU: EUH GUFH (serial control port FUT G2T (serial control statis mask | INUMT: E.º 01Ch ;parrallel in/dit wort CONTTA: FQU 02Fh ;serial data port CONTTU: EQU 02Fh ;serial control port FQU 02F ;serial con in status mask | <pre>14UHT: EUT 010h ::parrallel in/out port CONTA: EUT 012h ::serial data port CONTA: EUT 02h ::serial control port FUT 02h ::serial control port FUT 01h :serial control status mask FUT 01h :serial control status ; For the SDSysters "Child supplied in the r manual two status vill need offerent files already ; on PT disk with docimentativ. ; In CHOTYP: clange to below code ; EDA FWIT ; ADA FWIT ; ADA AUTH :CLEAN BIT 7 ; ADA AUTH :CLEAN BIT 7 ; ATH MUT</pre> | T 2 : | FUL 0.1 | 1 | | | |
| INUMT: FUT GUCH (permalle) in/out port CONTA: FUT GUFH (serial data port CONTU: EUH GUFH (serial control port FUT G2T (serial control statis mask | INUMT: E.º 01Ch ;parrallel in/dit wort CONTTA: FQU 02Fh ;serial data port CONTTU: EQU 02Fh ;serial control port FQU 02F ;serial con in status mask | <pre>14UHT: EUT 010h ::parrallel in/out port CONTA: EUT 012h ::serial data port CONTA: EUT 02h ::serial control port FUT 02h ::serial control port FUT 01h :serial control status mask FUT 01h :serial control status ; For the SDSysters "Child supplied in the r manual two status vill need offerent files already ; on PT disk with docimentativ. ; In CHOTYP: clange to below code ; EDA FWIT ; ADA FWIT ; ADA AUTH :CLEAN BIT 7 ; ADA AUTH :CLEAN BIT 7 ; ATH MUT</pre> | teti. | FUL 02 | n ;time | r control | port | |
| CONTA: FUT 02Fh :serial data port CONCTU: EUE 02Fh :serial control port FUT 02h :serial con in statis mask | CONTA: FUT DIFN :serial data port CONCTU: EUT DIFN :serial control port FUT DIF :serial con in statis mask | CONTA: EQU 02Fh :serial data port CONTU: EQU 02Fh :serial control port FUT 02Fh :serial con in status mask FUT 01h :serial con out status mask : For the SDSysters NDEDU supplied in the manual two areas will need contained as : It will give you the different files already : On PF disk with docimentative : In CHUTYF: crange to below cous : ATC AVD40; :mANF 01 OFF : ANF AVD10 :mANF 01 OFF | INCHT: | E [11 6 2) | ih ;parr | allel in/ | out port | |
| F1F 02F (serial con in statis mask | Fill 02- (serial con in statis mask | Fill 02- (Serial Con in Status mask Full Oln (Serial Con out status mask Full Oln (Serial Con out status mask) For the SDSysters NOELD supplied in the manual two areas will need channed. If is supposed that SL-M disk full the otrained as if the will give you the different files already (On H* disk with docimentation) In CHOTYP: clange to below code (Difference) (Status) In CHOTYP: clange to below code (Difference) (Status) ANA Avoit (SCEFAX ELT 7) (CTA Media) | | FQ1 0.2 | r :seri | al data p | ort | |
| FUE 025 (serial con in status mask | F10 02m (serial con in status mask | Fill 02- :#erial con in status maak Fill 01n :#erial con out status maak : For the SDSysters NOBIGE supplies in the manual two areas will need chaning. If is supposed that SLOW disk \$1* cm distance as : it will give you the different files already : on F cuss with documentative. : : In CHOTYP: crange to below code : LTA DAIT : ACT AJ040; :MAIF H OFF : ANA AJ01; :CLFAK bIT 7 : OTA MAIT | CONCTLE | EQ1 0.21 | 'n ;seri | | | |
| °u" Oln :serial con out stafig mage | fu‼ Oln :serial con out status ma⊛k | : For the SDSysters "RETOI supplies in the manual two acess will need channing. If is supdested that SIGM disk \$1" or distance as it will give you the different files diready i on "F disk with documentative." . In CHOTYF: crange to below code . LTA EXIT . ALL AURANCE (CFA SIT 7 . ANA AURIT | | F_1 02 | ;seri | al con in | status mask | |
| | | <pre>c manual two areas will need changin. If is j suggested that SLAW disk all the different files already i of P[*] disk with anoimentation In CHUTYP: drange to below doug</pre> | | F _ ! 01: | ser1 | al con ou | t status mesr | |
| | | * Tanial two areas will new changin. It is a substor that SLIVE disk for the ortainer as of the substored of the substored of the substored of on P ² disk with and mentation. in CHGTYP: change to below onus in the substored of the substored of the interval of of the interval of the interval of | | | | | | |
| : For the SDSysters Applies in the | | 1 LDA HAIT 1 AJD436 (MARENG UFF 1 ANA AJOTH (CLFAN 617 7 1 JTA HAIT | | ; 1 | uquesteu fhaf t will give y n Af gisk wif | SILEM C: Ou the C: T COCIME | lek ≹lt c# drt Lfferent film: htatissi | alter as |
| I manual two areas will need channing of a providented that SLOW dime for orthannes as only will give your the different files already | ; suggested that SICH disk \$1* th otherhow as ; it will give you the different files already | : AFC A,0403 (MALES) UFA : ANA A,078m (CLEAK 617.7 : UTA "NIT | | 1 | 1 6 8 11 11 | • • | | |
| " Manual two areas will need changin. If is a superstead that SL. M disk \$1: th distained as of the will give you the different files already on #" disk with and mentation. " In CHUTYP: crange to below chue the distance with the optime." | ; suudested that SlovM disk \$10 re ustained as ; it will glow you the different files already ; on P° disk with documentation; ; in CHOTYE: crange to below code | 3TA - HN1T | | 1 | AFT A, | 0437 | MANE OF CER | |
| " Marial two areas will need changes. It is a supested that SLOM disk \$10 m ortainer as rit will give you the different files already ron P [*] disk with one mentation. In CHUTYP: crange to below code r | : diagested that SlovM disk tit remoration; as : it will give you the different files already : On P [*] disk with anoimentation; In CHOTYP: crange to below code : LDA PNIT : AJD40; : MAKEON CP4 | | | - | | | CLIAN BIT 7 | |
| <pre>^ Marial ten areas will need changin. If is suprested that Sin W disk \$1. ten trians, as it will give you the different files already on P* disk with docimentation In CHUTYE: change to below chue</pre> | : diguested that Slove disk fit or distance as : it will give you the different files already : on P ² disk with anotmentation. : in CHOTYP: crange to below code : IDA PNIT : ADA PNIT : ADI ALODA : ADA ALOT : CATAN ET 7 | | | | | 12.1 | | |
| <pre>1 Marcial two areas will need changin. If in 1 Suprested road SLAW disk \$2 road the different files already 2 of #* disk wath and mentation. 2 in CHUTYF: crange to below chue 3</pre> | 2 Buddested that SILM disk bit in strained as 1 it will give you the different files already 1 on P* disk with and mentation. 1 in CHGTYF: crange to below chue 2 IDA PNIT 1 ALDA PNIT 2 ALDA 2014 (CLFAK bIT 7 1 ANA ALDIM (CLFAK bIT 7 1 ANA ALDIM (CLFAK bIT 7) 1 ANA ALDIM (CLFAK bIT 7) 1 ANA ALDIM (CLFAK bIT 7) | | | | A 7 1 | | | |

| | RET |
|--------|--|
| | |
| | Under the BOOT: heading remove the code |
| | : sections bT41; BT51; JOMON1; and HALT1 |
| | ; Make BT2: the below code |
| | : CALL FRITSL |
| | : JNZ MUNITUH (GO TU MUNITIR |
| | : DISK FEROR |
| | : BOOTLD: SAME AS LISTED |
| | I IN MANUAL AND |
| | NOW IS NEXT |
| | AFTEN BT. |
| | · · · · · · · · · · · · · · · · · · · |
| | : CP/M BASIC INPUT/OUTPUT OPERATING SYSTEM (BIGL) |
| | |
| | : This version boots in single density |
| | calls A.B single density. |
| | sytem equates and definitions start first |
| | . Bytem equates and definitions start first |
| F000 - | STPROM: FOP OFJOON :LOCATION OF SOPROM |
| 0030 = | MSIZE: FEP 60 :MEMORY SIZE IN REVIES. |
| | |
| 0028 - | CONDIA: FOU 268 (NEEDE: TU CLEAR AT SUCT |
| | |
| | |
| | : sdsytem disk storage information |
| | • |
| 0042 · | UNIT EQU 42H SPNIT BYTH FOR DISK SPLFCT |
| | ; These and other values in the 040 to 080 nex. |
| |) range could be changed to DFD4/nex range. Some |
| | ; systems start their equates with a base aduress |
| | ; and then add to it for their byte locations. |
| | |
| | · cpm equates |
| | |
| X0C0 - | 2BASE FOR (MS12F+20)*1024 (BIAF FUR CPM |
| D400 - | :LARGEN THAN 17K. CPME FOR CHASE+3400H START OF CPM |
| D400 - | |
| EA00 - | БОСБ ГСР СРМВ+206Н ;START ОГ ВОО. ВІСБ ГО№ СРМБ+1600Н |
| 002C - | |
| | NSECTS FOR 44 INFMBER OF SECT TO LOAD |
| | |

start of 2.2 bios

continued

| EADO | | : •n | | BIOS ocations for | START | OF BIOS. |
|---|---|---|---|--|--|--|
| | | | | d 2.2 entry | | |
| | C 37C FA | | JHP | 8001 | FROM C | OLD START LOADER. |
| | C 3 D 3 E A C 3 O 5 E B | WBOOTE: | JMP JMP | WBOOT Const | | ARN BOOT. Console KB Status. |
| EA09 | C309EB | | JMP | CONIN | READ C | ONSOLF CHARACTER. |
| | C30DEB C35CEb | | JMP JMP | CONOT | | CONSOLE CHARACTER. |
| EA12 | C362EB | | JMP | PUNCH | WRITE | LISTING CHAR. Punch Char. |
| | C363EB C3CEEA | | 4ML JHP | READER | ; READ R | EADER CHAR. |
| | C386EA | | JHP | HOMF1 TDSKSL | SELECT | ISK TO TRACE SERC. DISK DRIVE. |
| | C31EF0 C321F0 | SETTRK: | | | SEEX T | O TRACK IN REG A. |
| | C324F0 | SETSEC : | JMP | | | CTOR MUMBER. Sk Starting Adr. |
| EA27 | C327F0 | | JN P | SDPRON+27H | ; READ \$ | ELECTED SECTCR. |
| | C32AF0 C360EB | | JMP JMP | SDPROM+2AH PRSTAT | WRITE | SELECTED SECTOR. TATUS CHECK |
| | C364EB | | JNP | SECTRAN | SECTOR | TRANSLATE ROUTINE |
| | | ; | | DPH | | |
| | | : d1 | sk pa | | ers for | the disk drives |
| EA33 | - 62EA0000 | DPBASE: DPEG: | EQU DW | S ; BA X1T0,000 | | SK PARAMETER BLOCKS ;TRANSLATE TABLF |
| EA37 | 000000000 | | DW | 00000,00 | 0 0 H | SCRATCE AREA |
| EA3B | 6CEB5 3EA01 | DW | DW | DIRBUT,D CSV0,ALV0 | PB0 | DIR BUPF, PARM BLOCK |
| | 62FA0000 | DPE1: | DW | XLT0,000 | | TRANSLATE TABLE |
| | 00000000 6CF653EA | | 0 W | 0000H,00 | NOOH | SCRATCH AREA |
| | SCFB-JEA JAFC18FC | | DW DW | DIRBUF, D CSV1, ALV | | DIR BUFF, PARH BLOCK CHECK, ALLOC VECTORS |
| | | : | | | | |
| FA53 EA53 | 1400 | DPB0: | EQ1' DW | \$ 26 | | SD DISK PARM BLOCK SEC PER TRACK |
| EASS | 03 | | DB | 3 | | BLOCK SHIFT |
| FA56 FA57 | | | DB DB | 7 | | :BLOCK MASK ;EXTNT MASK |
| EASR | F200 | | DW | 242 | | DISK SIZE-1 |
| | 3F00 | | DW | 63 | | DIRECTORY MAX |
| EASD | 00 | | DS DB | 192 | | ;ALLOCO ;ALLOCI |
| FASE | 1 3 8 8 | | DW | 16 | | CHECK SIZE |
| FA60 EA62 | 0200 | XLTO: | D₩ EO'' | 2 5 | | :OFFSET ;TRANSLATE TABLE |
| FA62 | 010700131 | • | Die | 1,7,13,1 | | 1,17,23 |
| | 03090F150: 060C121804 | | D B. D B. | | 1,2,8,14 | |
| | | ; | | -,, | | |
| | | ; ; 800T | | | | |
| | | | 18 t) | e first entr | y entere | d from monitor |
| | | : after | | t and after | system 1 | s loaded from |
| | | ; syste ; | - T.F. | ick (tk 0 and | 11 | |
| EA7C | 318000 | BOOT : | LXI | SP,80H | SET STA | CK POINTER. |
| | | : 11 | .11 | initlization | 19 not | done in monitor |
| | | : •n | ter o | ode here *** | ••• | |
| EA7F | | · | XRA | * | | |
| FA80 EA83 | 320400 320300 | | STA STA | 4 | COISK DISK | |
| EAPS | 322800 | | STA | 4 2 | | |
| FA89 | | | CALL | | | |
| | CD96EA | | | | SET UP | |
| EAPE | CD9BEA DB2E 211CEL | | 1N LXI | H, OMSG | CLEAR C | JUMPS. ONSOLF STATUS. PENING MESSAGE. |
| EV01 | CD96FA DB2F 211CEE CD11FB | GOCPH | IN LXI CALI | H, OHSG PHSG | CLEAR C | ONSOLF STATUS. Pening MFSSAgf. |
| EA91 FA94 FA97 | CD96FA D82F 211CEE CD11FB 3A0400 4F | GOC PM : | IN LXI CALI LDA NOV | H, OMSG PMSG 4 C, A | CLFAR C PRINT O GFT DIS PASS TO | ONSOLF STATUS. Pening Message. K Number to CCP in C. |
| EA91 FA94 FA97 | CD96FA D82F 211CEE CD11FB 3A0400 | | IN LXI CALI LDA | H, OMSG PMSG 4 | CLEAR C PRINT O GFT DIS | ONSOLF STATUS. Pening Message. K Number to CCP in C. |
| EA91 FA94 FA97 | CD96FA D82F 211CEE CD11FB 3A0400 4F | : | IN LXI CALI LDA MOV JMP | H, OMSG PMSG 4 C, A | CLFAR C PRINT O GFT DIS PASS TO | ONSOLF STATUS. Pening Message. K Number to CCP in C. |
| EA91 FA94 FA97 EA98 | CD96FA DB2F 211CEL CD11FB 3A0400 4F C300D4 | : ; SFT (' | IN LXI CALI LDA MOV JMP P JUP | H, ONSG . PMSG 4 C, A CPMB IPS TO CP/M | CLFAR C PRINT O GFT DIS PASS TO JUMP TO | ONSOLF STATUS, PERING MESSAGE, K NUMBER TO CCP IN C. CCP. |
| EA91 FA94 FA97 EA98 EA98 EA98 | CD95EA DB2F 211CEL CD11FB 3A0400 4F C300D4 3EC3 320000 | : ; SFT (* | IN LXI CALI LDA MOV JMP P JUP MVI STA | H, OMSG , PMSG 4 C, A CPMB IPS TO CP/M A, 0C3H 0 | CLFAR C PRINT O GFT DIS PASS TO JUMP TO | ONSOLF STATUS, Pening Mpssagt, K Number to CCP in C. CCP. TC WBOOT |
| EA91 FA94 FA97 EA98 EA98 EA98 EA98 | C D96FA D82F 211CFb CD11FB 3A0400 4F C300D4 3FC3 320000 2103EA | : ; SFT (' | IN LXI CALI LDA MOV JMP P JUP MVI STA LXI | H, OMSG , PMSG 4 C, A CPMB IPS TO CP/M A, OC3H 0 H, WBOOTE | CLFAR C PRINT O GFT DIS PASS TO JUMP TO | ONSOLF STATUS, Pening Mpssagt, K Number to CCP in C. CCP. TC WBOOT |
| EA91 FA94 FA97 EA98 EA98 EA98 EA90 EA40 EAA0 EAA3 EAA6 | CD98FA DB2F 211CFL CD11FB 3A0400 4F C300D4 3ECJ 320000 2103EA 220100 320500 | : ; SFT (' ; SETUP: | IN LXI CALI LDA MOV JMP P JUP MVI STA | H, ONSG , PMSG 4 C, A CPMB NPS TO CP/M A, 0C3H 0 H, NBOOTE 5 | CLFAR C PRINT O GFT DIS PASS TO JUMP TO | ONSOLF STATUS, Pening Mpssagt, K Number to CCP in C. CCP. TC WBOOT |
| EA91 FA94 FA97 EA98 EA98 EA98 EA90 EA40 EAA0 EAA3 EAA6 | CD98FA DB2F 211CFL CD11FB 3A0400 4F C300D4 3ECJ 320000 2103EA 220100 320500 | : ; SFT (' ; SETUP: | IN LXI CALI LDA MOV JMP P JUP MVI STA LXI SHLI STA LXI | H, OMSG 4 C, A CPMB IPS TO CP/M A, 0C3H 0 H, NBOOTE 1 5 H, BDOS | CLFAR C PRINT O GFT DIS PASS TO JUMP TO PUT JMP ADR AT PUT JUM | ONSOLF STATTS, PERING HESSAGF, K NUMBER TO CCP IN C. CCP. TC WBOOT IERO. |
| EA91 FA94 FA97 EA98 EA98 EA98 EA98 EA90 EAA0 EAA0 EAA0 | CD98FA DB2F 211CFL CD11FB 3A0400 4F C300D4 3ECJ 320000 2103EA 220100 320500 | : ; SFT (' ; SETUP: | IN LXI CALI LDA MOV JMP P JUP MVI STA LXI SHLI STA | H, OMSG 4 4 C, A CPMB IPS TO CP/M A, OC3H 0 H, MBOOTE 1 5 H, BDOS 6 | CLFAR C PRINT O GFT DIS PASS TO JUMP TO PUT JMP ADR AT PUT JUM AT ADR | ONSOLF STATTS, PERING HESSAGF, K NUMBER TO CCP IN C. CCP. TC WBOOT IERO. |
| EA91 FA97 EA98 EA98 EA98 EA498 EAA0 EAA0 EAA0 EAA1 EAAF EAAF | CD9BFA DB2F 211CFL CD11FB 3A0400 4F C300D4 320000 2103EA 220100 320500 2106DC 220600 21000C240 | : ; SFT (' ; SETUP: | IN LXI CALI LDA MOV JMP P JUP MVI STA LXI SHLI SHLI LXI 40H | H, OMSG 4 4 C, A CPMB IPS TO CP/M A, OC3H 0 H, MBOOTE 1 4 H, BDOS 6 | CLFAR C PRINT O FFT DIS PASS TO JUMP TO PUT JUMP ADR AT PUT JUM SAT ADR SET DEF | ONSOLF STATUS, PERING MESSAGE, K NUMBER TO CCP IN C. CCP. TC WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. |
| EA91 FA94 FA97 EA98 EA98 EA98 EA90 EA40 EAA0 EAA3 EAA6 | CD9BFA DB2F 211CFL CD11FB 3A0400 4F C300D4 320000 2103EA 220100 320500 2106DC 220600 21000C240 | : ; SFT U ; SETUP: | IN LXI CALI LDA MOV JMP P JUP MVI STA LXI SHLI LXI | H, OMSG 4 4 C, A CPMB IPS TO CP/M A, OC3H 0 H, MBOOTE 1 4 H, BDOS 6 | CLFAR C PRINT O FFT DIS PASS TO JUMP TO PUT JUMP ADR AT PUT JUM SAT ADR SET DEF | ONSOLF STAT'S, PENING MESSAGE, K NUMBER TO CCP IN C. CCP. TC WBOOT ZERO. P TO BDOS 5.6.7. |
| EA91 FA94 FA97 EA98 EA98 EA98 EA40 EAA0 EAA3 EAA6 EAA6 EAA6 EAAF | CD9BFA DB2F 211CFL CD11FB 3A0400 4F C300D4 320000 2103EA 220100 320500 2106DC 220600 21000C240 | : ; SFT U ; SETUP: SHLD ; : The f | IN LXI CALI LDA JMP P JUP MVI STA LXI SHLI LXI 40H RET | H.ONSG PMSG C.A CPMB IPS TO CP/M A.OC3H 0 M.MBOOTT 1 4 M.BOOS 6 6 0 M.BOS 6 4 M.BOS 6 4 4 4 4 4 4 4 4 4 4 4 4 4 | CLFAR C PRINT O SFT DIS SFASS TO JUMP TO PUT JMP TO PUT JMP ADR AT SET DEF SET DEF SET DEF SET URN | ONSOLF STAT'S, PENING MESSAGE, K NUMBER TO CCP IN C. CCP. TO WBOOT IERO. P TO BDOS K/6,7. AULT DHA ADR. FROM SETUP. Code for the drive |
| EA91 FA94 FA97 EA98 EA98 EA98 EA40 EAA0 EAA3 EAA6 EAA6 EAA6 EAAF | CD9BFA DB2F 211CFL CD11FB 3A0400 4F C300D4 320000 2103EA 220100 320500 2106DC 220600 21000C240 | : ; SFT U ; SETUP; SHLD ; ; The f ; if m th ; if m th | IN LXI CALA MOV JMP P JUP MVI STA LXI STA LXI SHLI LXI RET Ollow Fe di | H.ONSG .PMSG .PMSG C.A C.PMB IPS TO CP/M A.OC3H 0 M.MBOOTT 1 4 M.BOOTT 1 4 M.BOOS 6 6 60 10 10 10 10 10 10 10 10 10 1 | CLFAR C PRINT O GFT DIS GFT DIS FASS TO JUMP TO PUT JUMP TO PUT JUMP TO PUT JUMP TO SET DEF RETURN the SD ation. A Change C | ONSOLF STATUS, PERING MESSAGE, K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7, AULT DHA ADR. FROM SETUP. |
| EA91 FA94 FA97 EA98 EA98 EA98 EA40 EAA0 EAA3 EAA6 EAA6 EAA6 EAAF | CD9BFA DB2F 211CFL CD11FB 3A0400 4F C300D4 320000 2103EA 220100 320500 2106DC 220600 21000C240 | : ; SFT U ; SETUP; SHLD ; ; The f ; if m th ; if m th | IN LXI CALA MOV JMP P JUP MVI STA LXI STA LXI SHLI LXI RET Ollow Fe di | H, OHSG - PMSG 4 C,A CPMB IPS TO CP/M A, 0C3H 0 H, WBOOTF 5 4 H, BDOS 6 H, 80H | CLFAR C PRINT O GFT DIS GFT DIS FASS TO JUMP TO PUT JUMP TO PUT JUMP TO PUT JUMP TO SET DEF RETURN the SD ation. A Change C | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7, ALLT DHA ADR. FROM SFTUP. Code for the drive (58 are single density |
| Е ГА4 Г ГА4 Г ГА4 Г ГА4 Г ГА4 В В В В В В В В В В В В В В В В В В В | CD9BFA DB2F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 320500 21060C 21060C C9 210000 | : ; SFT U ; SETUP; SHLD ; ; The f ; if m th ; if m th | IN LXII CALLI LDA MOV JMP P JUP STA SHLI STA LXI SHLI SHLI LXI 40H RET DDPH' | H.ONSG PMSG C.A CCPMB IPS TO CP/M A.OC3H 0 M.MBOOTT 1 4 M.BOOS 6 6 M.BOOS 6 4 M.BOOS 6 6 1 1 4 M.BOOS 6 6 1 1 1 1 1 1 1 1 1 1 1 1 1 | CLFAR C PRINT O GFT DIS GFT DIS FASS TO JUMP TO PUT JUMP TO PUT JUMP TO PUT JUMP TO SET DEF RETURN the SD ation. A Change C | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7, ALLT DHA ADR. FROM SFTUP. Code for the drive (58 are single density |
| ЕГРА 993 ЕГРА 993 ВС 2003 6 900 2003 6 900 2003 6 900 2003 6 900 2003 6 900 2003 6 900 2003 6 900 2003 8 900 2004 7 8 900 2004 7 8 900 2004 7 8 900 2004 7 8 900 2004 7 8 900 2004 7 8 900 2004 7 900 2004 8 900 2000 2000 2000 2000 2000 2000 200 | CD95FA D52F 211CFL CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 2103EA 220500 2180002240 C9 2100000 79 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXII CALLI LDA NOV JMP P JUP P JUP STA LXI STA LXI STA LXI LXI LXI LXI A0H RET CIION C LXI NOV | H.0HSG PMSG C.A CCPMB IPS TO CP/M A.0C3H 0 N.WBOOTT 1 4 N.BOOS 6 H.80N Ing Storeges 1. weenry loc Inves needon de Inves needon de H.0 A.C H.0 A.C H.0 A.C H.0 A.C H.0 H.0 H.0 H.0 H.0 H.0 H.0 H.0 | CLFAR C PRINT O SFT DIS SFT DIS SFT DIS STASS TO STUT JUMP TO PUT JUMP TO PUT JUMP TO PUT JUM AT ADR AT SET DEF SRETURN the SD ation. A Change C Ive. | ONSOLF STATUS, PERING MESSAGE, K NUMBER TO CCP IN C. CCP. TO WBOOT JERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SFTUP. Code for the drive (18 are single density code here and add |
| ЕГЕХ БГЕХ БЕРЕ БЕРЕ БЕРЕ БЕРЕ БЕРЕ БЕ БЕ БЕ БЕ БЕ БЕ БЕ БЕ БЕ Б | CD9BFA DB2F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 320500 21060C 21060C 21060C C9 210000 79 FF02 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXII CALLI LDA MOV JMP P JUP STA LXI SHLI LXI SHLI LXI 40H RET 001000 e uni re du DPH (LXI | H.0HSG PMSG 4. C,A CPMS IPS TO CP/M A.0C3H 0. H.WBOOTT 1. H.BOOS 6. H.BOS 6. H.80N H.BOS 1. | CLEAR C PRINT O PRINT O PRINT O PUT JMP ADR AT PUT JMP AT ADR STT JUM AT ADR STT JUM Che SD Ston. A Change C Ive. FIND IF | ONSOLF STATTS, PERING HESSAGE, CCP IN C. CCP IN C. CCP. TC WBOOT IERO. P TO BDOS 5/6/7. AULT DHA ADR. FROM SFTUP. Code for the drawe is8 are single density oode here and add |
| EA91 FA94 FA94 EA98 EA98 EA98 EA49 EA49 EA49 EA49 EA49 EA49 EA49 EA49 | CD9BEA DB2F 211CFb CD11FB 320400 4F C300D4 3EC3 320000 2103EA 220100 320500 210600 210600 210600 210000 C9 2100000 79 FF02 D0 E6FD | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA NGV JMP P JUP MVI STA LXI SHAL STA LXI SHAL STA LXI SHAL LXI SHAL LXI SHAL LXI SHAL LXI SHAL CXI AN P JUP P JUP P JUP P JUP STA SHAL SHAL SHAL SHAL SHAL SHAL SHAL SHA | H.0HSG PMSG 4. C,A CPMB IPS TO CP/M A.0C3H 0 H.WBOOTT 1 4. H.BDOS 6 H.BOS 6 H.BOS 6 H.BOS 6 H.BOS 7 4. H.BOS 6 4. H.BOS 6 4. 4. 4. 4. 5. 5. 6 1. 5. 6 1. 5. 7. 7. 7. 7. 7. 7. 7. 7. 7. 7 | CLEAR C PRINT O PRINT O PRINT O PUT JMP ADR AT PUT JMP AT ADR SET DEF SELON AT ADR SET DEF SELON A Change C Ver FIND IF SELECTE SEROVE | ONSOLF STATUS, PERING HESSAGE, CCP IN C. CCP. TC WBOOT ZERO. P TC BDOS 5/6/7. AULT DHA ADR. FROM SFTUP. Code for the drive 658 are single density ode here and add MORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS |
| EA91 FA94 FE EA98 EA94 FE EA98 EAA98 EAAA9 EAAA EAAA EAAA EAAA EAA | CD9BEA DB2F 211CFL CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 2103EA 220500 210000 210000 2100000 2100000 C9 2100000 79 FF02 D0 E6FD 324200 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA MOV JMP P JUP MVI STA SHLI STA LXI SHLI LXI SHLI LXI SHLI LXI MOV CPI RET RET ROP STA | H.0HSG PMSG C.A. CPMB IPS TO CP/M A.0C3H 0 H.WBOOTT 1 4 H.BODS 6 H.80N Ing Storeges 1. wenory loc 1. we per dr H.0 A.C 2 OTDH UNIT | CLFAR C PRINT O PRINT O SPUT JIMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO SUT JMP AT ADR AT ADR SET JMP SELECTE SET TME SET | ONSOLF STATUS, PERING MESSAGE, K NUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add MORF THAN TWO DISKS DIF SO RETURN THEN CPH BITS EDISK UNIT REMORY LOCAT |
| EA91 FA94 FA94 FA98 EA98 EA98 EA400 | CD9BFA DB2F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 320500 21060C 220600 21060C 21060C C9 2100000 79 2100000 79 2100000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 79 2200000 2200000 2200000 2100000 79 200000 2200000 2200000 2100000 2100000 210000 2100000 2100000 2100000 2100000 210000 20000 20000 210000 200000 2000000 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA MOV P JUP P JUP MVI STA SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI SHI LXI SHI SHI SHI SHI SHI SHI SHI SHI SHI SH | H. ONSG PMSG 4 C.A CPMB IPS TO CP/M A. 0C3H A. 0C3H H. WBOOTT 1 S. 1005 6 H. BOOS 6 H. BOOS 6 H. 800 H. 800 H. 800 H. 9 H. 9 H. 10 A.C. 2 OFDN UNIT L.C N.0 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FRA94 FRA97 EA98 EA98 EA98 EA89 EA80 EA80 EA80 EA80 EA80 EA80 EA80 EA80 | CD95EA D52F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 320500 2106DC 220600 2180002240 C9 210000 79 FF02 D0 E6FD 324200 69 2600 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA MOV JMP P JUP P JUP P JUP STA LXI STA LXI STA LXI STA A UP I CPI CPI RNC CPI RNC ANI STA MOV LXI LXI LXI LXI LXI LXI STA LXI LXI STA LXI LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA STA STA STA STA STA STA STA STA STA | H.0HSG .PMSG .PMSG C.A C.A CPMB IPS TO CP/M A.0C3H 0 M.WBOOTT 1 4 .WBOOTT 1 5 .WBOOS 6 6 .WBOOS 6 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATTS, PERING HESSAGE, CCP IN C. CCP IN C. CCP. TC WBOOT IERO. P TO BDOS 5/6/7. AULT DHA ADR. FROM SFTUP. Code for the drive is8 are single density ode here and add MORF THAN TWO DISKS ID IF SO RETURN THEN CPM BITS IDISK UNIT HEMORY LOCAN |
| EA91 FA94 FA94 FA98 EA98 BD BD BAAA9 EAAA0 EAAA0 EAAA0 EAAA0 EAAA0 EAAA0 EAAA0 EAAA0 EAAA0 EAABF EAABF EAABF EAABF EAA22 FAAC3 | CD9BEA DB2F 211CFb CD11FB 320400 4F C300D4 2103EA 220100 320500 2103EA 220100 320500 21060C 21060C 21060C 210000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 220000 210000 79 220000 210000 79 220000 210000 79 220000 210000 79 220000 210000 79 20000 210000 79 20000 210000 20000 210000 210000 20000 210000 20000 210000 20000 210000 2000000 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA MOV P JUP P JUP MVI STA SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI LXI SHI SHI LXI SHI SHI SHI SHI SHI SHI SHI SHI SHI SH | H, ONSG A A C, A C, A CPMB IPS TO CP/M A, 0C3H 0 H, NBOOTT 1 4 H, BOOS 6 H, BON Lt memory loc 1 ves needed - one per dr H, 0 A,C OTDH UNIT L,C N,0 PBASE H | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 EA98 EA98 EA98 EA403 EA403 EA403 EA403 EAA4 EA84 EA84 EA84 EA84 EA84 EA84 EA84 | CD9BEA DB2F 211CFE CD11FB 320400 4F C300D4 2103EA 220100 320500 2103EA 220100 320500 210600 220600 210000 210000 C9 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 2110000 79 2100000 79 2100000 2110000 79 2100000 2110000 79 200000 2110000 79 200000 2100000 210000 220000 210000 2200000 22000000 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXIILDA MOV JMP P JUP P JUP NVI STA LXILSIL STA LXILSIL STA LXIL STA LXI LXI LXI LXI LXI LXI LXI LXI LXI LXI | H, ONSG PMSG 4, C,A CPMB IPS TO CP/M A, 0C3H 0 H, WBOOTT 1 4, H, BDOS 6 6 H, 80N Ling storeges t memory loc lives needed - one per dr H, 0 A,C 2 OTDH UNIT L,C H, 0 D, 0 PBST H H H | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA97 EA98 EA98 EA98 EA88 EA88 EA88 EA88 EA88 | CD9BEA DB2F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 2103EA 220100 2106DC 220600 210600 2100002240 C9 2100000 79 FF02 D0 E7 24000 1133EA 29 24 29 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXI CALL LDA MOV JJMP P JUP MVI STA LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI SHLL LXI DDP MOV STA SHLL DAD DAD DAD DAD DAD | H.0HSG .PMSG .PMSG C.A CPMB IPS TO CP/M A.0C3H 0 M.WBOOTT 1 4 M.BOOS 0 6 H.80N A.C 4 H.80N A.C 2 0 0 0 0 0 0 0 0 0 0 0 0 0 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA97 EA98 EA98 EA98 EA88 EA88 EA88 EA88 EA88 | CD9BEA DB2F 211CFE CD11FB 320400 4F C300D4 2103EA 220100 320500 2103EA 220100 320500 210600 220600 210600 220600 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 210000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 20000 200000 2000000 | : ; SFT U ; ; SETUP; ; ; ; SHLD ; ; In th ; if mo ; ; ore; ; | IN LXIILDA MOV JMP P JUP P JUP NVI STA LXILSIL STA LXILSIL STA LXIL STA LXI LXI LXI LXI LXI LXI LXI LXI LXI LXI | H.0HSG .PMSG .PMSG C.A CPMB IPS TO CP/M A.0C3H 0 M.WBOOTT 1 4 M.BOOS 0 6 H.80N A.C 4 H.80N A.C 2 0 0 0 0 0 0 0 0 0 0 0 0 0 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA97 EA98 EA98 EA98 EA40 EA40 EA40 EA40 EA40 EA40 EA40 EA40 | CD9BEA DB2F 211CFE CD11FB 320400 4F C300D4 2103EA 220100 320500 2103EA 220100 320500 210600 220600 210600 220600 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 79 2100000 210000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 2000 20000 20000 200000 2000000 | : SFT U SETUP: SHLD: The f i The f i The f TDSKSL: | IN LXIILDA MOV JMP P JUP MVIISTA LXIILSTA SHLI LXIILSTA SHLI LXII SHLI LXII SHLI LXII SHLI LXII SHLI LXII SHLI LXII SHLI LXII DAD DAD DAD DAD DAD DAD | H.0HSG .PMSG .PMSG C.A CPMB IPS TO CP/M A.0C3H 0 M.WBOOTT 1 4 M.BOOS 0 6 H.80N A.C 4 H.80N A.C 2 0 0 0 0 0 0 0 0 0 0 0 0 0 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA94 FA94 FA98 FA98 FA47 FA47 FA47 FA47 FA47 FA47 FA47 FA47 | CD95EA D52F 211CFb CD11FB 3A0400 4F C300D4 32C000 2103EA 220100 2103EA 220100 210000 21060C 21060C 21060C 210000 79 FF02 D0 FF02 D0 E6FD 324200 69 2600 1133EA 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20 | : ; SFT U ; ; SETUP; ; SHLD ; ; In th ; if mo ; mote ; | IN LXI LALI LALI LALI LALI JMP P JUP NVI STAL SHLL LXI SHLL LXI SHLL LXI SHLL LXI LXI LXI LXI LXI LXI LXI LXI LXI L | H.0HSG .PMSG .PMSG C.A CPMB IPS TO CP/M A.0C3H 0 H.WBOOTT 1 4 H.BOOS 6 6 4 H.BOOS 6 6 4 H.BOOS 6 6 4 H.BOOS 6 6 6 1 1 4 H.BOOS 6 6 6 1 1 1 4 H.BOOS 6 6 6 1 1 1 1 1 1 1 1 1 1 1 1 1 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA94 FA94 FA98 FA98 FA47 FA47 FA47 FA47 FA47 FA47 FA47 FA47 | CD9BEA DB2F 211CFb CD11FB 3A0400 4F C300D4 3EC3 320000 2103EA 220100 2103EA 220100 21060C 220600 2140002240 C9 2100000 79 FF02 D0 E6FD 324200 49 2600 1133EA 29 24 20 5 | : SFT U SETUP: SHLD: The f i The f i The f TDSKSL: | IN LXII LAI LAI LAI LAI SAI SHLI SHLI SHLI SHLI SHLI SHLI LXII SHLI LXII SHLI LXII SHLI LXII SHLI LXII SHLI SHLI LXII SHLI LXII SHLI SHLI SHLI SHLI SHLI SHLI SHLI SH | H, OHSG , PMSG , ACSA C,A CPMB IPS TO CP/M A, 0C3H 0 H, MBOTT 1 5 H, BOOS 6 H, 80H A, 0C3H 0 0 4 H, 80H A, 0C3H 0 0 1 4 H, 80OT 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 5 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA94 FA94 FA98 FA98 FA47 FA47 FA47 FA47 FA47 FA47 FA47 FA47 | CD95EA D52F 211CFb CD11FB 3A0400 4F C300D4 32C000 2103EA 220100 2103EA 220100 210000 21060C 21060C 21060C 210000 79 FF02 D0 FF02 D0 E6FD 324200 69 2600 1133EA 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20 | : SFT U SETUP: SHLD: The f i The f i The f TDSKSL: | IN LXI LALI LALI LALI LALI JMP P JUP NVI STAL SHLL LXI SHLL LXI SHLL LXI SHLL LXI LXI LXI LXI LXI LXI LXI LXI LXI L | H.0HSG .PMSG .PMSG C.A CPMB IPS TO CP/M A.0C3H 0 H.WBOOTT 1 4 H.BOOS 6 6 4 H.BOOS 6 6 4 H.BOOS 6 6 4 H.BOOS 6 6 6 1 1 4 H.BOOS 6 6 6 1 1 1 4 H.BOOS 6 6 6 1 1 1 1 1 1 1 1 1 1 1 1 1 | CLFAR C PRINT O PRINT O STAFT DIS STAT PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TADR AT SET DEF STATA STIND IF SELECTI SET ME SET ME S | ONSOLF STATUS, PERING MESSAGE. K HUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. AULT OHA ADR. FROM SFTUP. Code for the drive (68 are single density rode here and add HORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UND F THE NL REGS FOR NO OF THE PARAMETERS |
| EA91 FA94 FA94 FA94 FA94 FA94 FA94 FA44 FA94 FA44 FA4 | CD95EA D52F 211CFb CD11FB 3A0400 4F C300D4 32C000 2103EA 220100 2103EA 220100 210000 21060C 21060C 21060C 210000 79 FF02 D0 FF02 D0 E6FD 324200 69 2600 1133EA 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20 | : SFT U : SFT U : SETUP: SHLD : The f : In th : If mo : TDSKSL: ; MOMF1: : WARH-1 | IN LXII LXI CALLA MOVE J MP JUP F JUP F JUP F JUP STA LXII STAL LXII STAL LXII STAL LXII STAL LXII STAL LXII STAL LXII STAL LXII LXII STAL LXII LXII STAL LXII STAL LXII LXII STAL LXII STAL LXII STAL LXII STAL LXII STAL LXII STAL STAL STA STA STA STA STA STA STA STA STA STA | H, ONSG PMSG 4, C,A CPMB IPS TO CP/M A, 0C3H 0 H, WBOOTT 1 4, WBOOTT 1 4, WBOOT 1 4, C 1 4, C 1 4 | CLEAR C PRINT O PRINT O PRINT O PUT JMP ADR AT PUT JMP ADR AT PUT JMP ADR AT PUT JMP CAR AT CAR AT | ONSOLF STATUS, PERING HESSAGE, K NUMBER TO CCP IN C. CCP. TC WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SPTUP. Code for the drive 658 are single density ode here and add MORF THAN TWO DISKS D IF SO RETURN THEN CPM BITS DISK UNIT HEMORY LOCAN NOT THE PARAMETERS SELFCTED DRIVE |
| EA,94 FA,94 FA,94 FA,94 FA,94 FA,94 FE,444 FA,44 FE,445 FA,44 FE,445 FA,44 FE,445 FA,44 FE,445 FA,445 FE,445 FA,445 FE,445 FA,445 FE,445 FA,445 FE,445 FA,455 FE,445 FA,455 FE,455 FA,45 | CD95EA DB2F 211CFE CD11FB 3A0400 4F C3000D4 2103EA 220100 2103EA 220100 2106DC 210600 210600 210600 210600 210000 79 FF02 D0 C9 2100000 79 FF02 D0 24000 2133EA 29 24 20 20 20 20 20 20 20 20 20 20 20 20 20 | : SFT U SETUP: SHLD: The f in th if mo if | IN LXI LXI CALLA MOV P JUP P JUP P JUP STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA LXI STA STA STA STA STA STA STA STA STA STA | H, ONSG PMSG 4, C,A CPMS IPS TO CP/M A, 0C3H 0 H, NBOOTF 1 4 H, BOOS 6 H, BOOS 6 C, 0 SETTAK BFAD ALL 0 S, THFN JVMP | CLEAR C PRINT O PRINT O PRINT O PUT JMP ADR AT PUT JMP ADR AT PUT JMP AT ADR SET DEF RETURN the SD ST THE SELECT SELECT POR THE POR THE F CPH BA TO CCF. | ONSOLF STATUS. PERING MESSAGE. K NUMBER TO CCCP IN C. CCCP. TC WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SETUP. Code for the drive 68 are single density fode here and add MORF THAN TWO DISKS 10 IF SO RETURN THEN CPM BITS DISK UNIT HEMONY LOCAT UP THE ML REGS FOR NOT THE PARAMETRY SELFCTED DRIVE |
| EA91 FA94 FA94 EA98 EA98 EA49 EA49 EA49 EA49 EA49 EA49 EAA9 EAA9 | CD95EA D52F 211CF5 CD11F5 3A0400 4F C300D4 JEC3 320000 2103EA 220100 2103EA 220100 210600 2140002240 C9 2100000 79 FF02 D0 E6FD 324200 49 2600 1133EA 29 29 29 29 0F00 C31EEA | : SFT U : SFT U : SETUP: SHLD : The f : In th : If mo : TDSKSL: ; MOMF1: : WARH-1 | IN LXI LAI LAI LAI LAI LAI SCALL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL LAI STAL STAL LAI STAL STAL LAI STAL STAL LAI STAL STAL STAL STAL STAL STAL STAL STAL | H.0HSG . PMSG . PMSG C.A CPMB IPS TO CP/M A.0C3H 0 | CLEAR C PRINT O PRINT O PRINT O STASS TO STASS TO PUT JMP TO PUT JMP TO TO PUT JMP TO SET JMP SET JMP SET JMP SET JMP SET JMP SET SET F CPM BA TO CCP SET STA | ONSOLF STATUS. PERING MESSAGE. K MUMBER TO CCP IN C. CCP. TO WBOOT ZERO. P TO BDOS 5,6,7. ALLT OHA ADR. FROM SETUP. Code for the drive (68 are single density rode here and add "MORF THAN TWO DISKS ID IF SO RETURN THEN CPM BITS DISK UNIT MEMORY LOCAT "UP THE NL REGS FOR NO OF THE PARAMETERS "BELFOTED DRIVE CK IN CK IN |
| EA014 FA044 FA044 FA044 EA044 EA044 </td <td>CD95EA DB2F 211CFE CD11FB 3A0400 4F C3000D4 2103EA 220100 2103EA 220100 2106DC 210600 210600 210600 210600 210000 79 FF02 D0 C9 2100000 79 FF02 D0 24000 2133EA 29 24 20 20 20 20 20 20 20 20 20 20 20 20 20</td> <td>: SFT U SETUP: SHLD: The f in th if mo if mo if</td> <td>IN LXI LXI CALLA MOV P JUP P JUP P JUP STAL STAL LXI STAL STAL LXI STAL STAL LXI STAL LXI STAL LXI STAL STAL STAL STAL STAL STAL STAL STAL</td> <td>H.0HSG . PMSG . PMSG C.A CPMB IPS TO CP/M A.0C3H 0 </td> <td>CLEAR C PRINT O PRINT O PRINT O STASS TO STASS TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO SET JMP SET JMP SET DEF SELECTI SELECTI SELECTI F CPN BA TO CCP. SET STA</td> <td>ONSOLF STATUS. PERING MESSAGE. K NUMBER TO CCCP IN C. CCCP. TC WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SETUP. Code for the drive 68 are single density fode here and add MORF THAN TWO DISKS 10 IF SO RETURN THEN CPM BITS DISK UNIT HEMONY LOCAT UP THE ML REGS FOR NOT THE PARAMETRY SELFCTED DRIVE</td> | CD95EA DB2F 211CFE CD11FB 3A0400 4F C3000D4 2103EA 220100 2103EA 220100 2106DC 210600 210600 210600 210600 210000 79 FF02 D0 C9 2100000 79 FF02 D0 24000 2133EA 29 24 20 20 20 20 20 20 20 20 20 20 20 20 20 | : SFT U SETUP: SHLD: The f in th if mo if | IN LXI LXI CALLA MOV P JUP P JUP P JUP STAL STAL LXI STAL STAL LXI STAL STAL LXI STAL LXI STAL LXI STAL STAL STAL STAL STAL STAL STAL STAL | H.0HSG . PMSG . PMSG C.A CPMB IPS TO CP/M A.0C3H 0 | CLEAR C PRINT O PRINT O PRINT O STASS TO STASS TO PUT JMP TO PUT JMP TO PUT JMP TO PUT JMP TO SET JMP SET JMP SET DEF SELECTI SELECTI SELECTI F CPN BA TO CCP. SET STA | ONSOLF STATUS. PERING MESSAGE. K NUMBER TO CCCP IN C. CCCP. TC WBOOT ZERO. P TO BDOS 5,6,7. AULT DHA ADR. FROM SETUP. Code for the drive 68 are single density fode here and add MORF THAN TWO DISKS 10 IF SO RETURN THEN CPM BITS DISK UNIT HEMONY LOCAT UP THE ML REGS FOR NOT THE PARAMETRY SELFCTED DRIVE |

2.28 1-84

ı di

1:49 .

19

> 13

19

13 11 11

19 _____

F8

1.18

| EADC 3E00 | MVI A,0 ;0 for single and 40 for |
|--|---|
| EADE 324200 | MVI A-0 :0 for single and 40 for STA 4224 :DDUBLF MVI C-0 CALL STTEM |
| FAEL DEDO | AVI C.O |
| FAE3 CDIEFA | MVI C,0 Call Settrk |
| | CADD DETTAR |
| EAF6 3E2C | MVI A ,NSECTS ;GET & SECTORS FOR CPM READ. |
| FAE8 324500 | STA 45H |
| EAEB OF02 | NVI C,2 |
| EAED CD21FA | CALL SETSEC |
| EAFO, 210004 | LXI H,CPMB (GET STARTING ADDRESS. |
| EAF3 224000 | SHID ADV |
| FAF5 CD20F0 | CALL SDPROM+20H |
| EAF9 3A6BEB | LDA TEMP |
| FAFC 324200 | |
| EAFC 324200 | STA 42H |
| FAFF CD9BEA | CALL SETUP ;SET "P J"MPS. |
| EB02 C394FA | JMP GOCPM ; GO BACK TO OPM. |
| | |
| | |
| | : CHFCK CONSOLF INPUT STATUS. |
| | |
| | |
| FROS COOFFOCS | CONST: CALL SDPROM+06H (READ CONSOLE STATUS, |
| | RET (RETURN FROM CONST. |
| | REF REFRONCONSI. |
| | |
| | ; |
| | ; READ A CHARACTER FROM CONSOLE. |
| | ; |
| EBO9 CD09F0 | CONIN: CALL SDPROM+09H |
| EB0C C9 | RET |
| | , |
| | WRITE A CHARACTER TO THE CONSULE DEVICE. |
| | |
| | |
| EBOD CDOCFO | CONOT: CALL SDPROM+OCH |
| EB10 C9 | RET |
| | |
| | |
| | |
| | : PRINT THE MESSAGE AT HEL UNTIL & ZERO. |
| | PRINT THE HESSAUF AT HEL UNITE A LERUT |
| EB11 7F | |
| E B11 / C | PMSG: MOV A,M ;GFT A CHARACTER. |
| EB12 B7 | ORA A : IF IT'S ZFRO, |
| EB13 C8 | R2 JRETURN. |
| EB14 4F | MOV C,A ;OTHERWISE, |
| EB15 CD0DEB | CALL CONOT (PRINT IT. |
| FB13 23 | INX H ; INCREMENT HEL, |
| F819 C311Eb | JMP PMSG ;AND GET ANOTHER. |
| | THE ENDO (AND OCT ANOTHER. |
| | CBIOS MESSAGES |
| | |
| | |
| | : |
| | : |
| | |
| EB1C 00044350 | |
| EB1C 0D044350 EB3A 0D043822 | |
| EB3A 00043822 E645 000A3630 | ZFOMSG: DB ODH,OAH, CP/M 2.2 SD SYSTEMS-KEBLFR SJ DB ODH,OAH, PSD ALB D 0D,OAH, MSIZE/10+'0',MSIZF MOD 10 + '0' |
| EB1C 0D044350 EB3A 00043922 E645 00043430 E649 48205631 | 2FOMSG: DB 0DH,9AH, CP/M 2.2 SD SYSTEMS-K4BLFR * *3 DB 0DH,9AH, #9 SD A4B * DB 0DH,3AH, MS12FL30+*0,MS12F MOD 10 + *0* |
| EB3A 00043822 E645 000A3630 | 2FOMSC: DB 0DH,0Ah, CP/M 2.2 SD SYSTEMS-K1BLFR 3 DB 0DH,0AH, 9°SD A48 DB 0DH,0AH, 9°SD A48 2 DB 0DH,0AH, 9512F104+0°, NS12F MOD 10 + 10° 2 F DB K V1.0 of 2/01/84 ,0 |
| EB3A 00043822 E645 000A3630 | 2FOMSG: DB 0DH, 3AH, 'CP/M 2.2 SD SYSTEMS-KİBLFR ' 53 DB 0DH, 3AH, 'B°SD A&B ' DB 0DH, 3AH, 'MSIZE/13+'0', MSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. |
| EB3A 00043822 E645 00043830 E649 46205631 | 2FOMSG: DB ODH.OAH., CP/M 2.2 SD SYSTEMS-K1BLFR 5.3 DB ODH.OAH., B'SD A48 DB ODH.OAH., B'SD A48 DB ODH.OAH., MSIZE/10+'0', HSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF |
| EB3A 000A3822 E645 000A3630 E649 48205631 F65C CD12F0 | ZFOMSG: DB ODH, JAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' 5] DB ODH, JAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZFIJO+'O', MSIZF MOD 10 + 'O' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH12H |
| EB3A 00043822 E645 00043630 E649 46205631 | 2FOMSG: DB ODH.OAH., CP/M 2.2 SD SYSTEMS-K1BLFR 5.3 DB ODH.OAH., B'SD A48 DB ODH.OAH., B'SD A48 DB ODH.OAH., MSIZE/10+'0', HSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF |
| EB3A 300A3822 EB45 003A3A30 EB49 4B205A31 FB5C CD12F0 EB5F C9 | ZFOMSG: DB ODH, JAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' 5] DB ODH, JAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZFIJO+'O', MSIZF MOD 10 + 'O' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH12H |
| EB3A 300A3922 EB45 000A3630 EB49 4B205631 FB5C CD12F0 EB5F C9 EB60 AF | 2FOMSG: DB ODH.OAH./CP/M 2.2 SD SYSTEMS-KIBLFR ' 5) DB ODH.OAH./B*SD A48 ' DB ODH.OAH./B*SD A48 ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRF LIST: CALL SDPROM+12H RET : PRSTAT XRA A |
| EB3A 300A3822 EB45 003A3630 EB49 4B205631 FB5C CD12F0 EB5F C9 | 2FOMSG: DB ODH, OAH, 'CP/H 2.2 SD SYSTEMS-KIBLFR ' SJ DB ODH, OAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZF10+'0', MSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : : INSERT YOUR ROWTINF HFRF LIST: CALL SDPROM+12H RFT : PRSTAT XRA A RET : RET''RN ALWAYS NOT READY |
| EB3A 300A3922 EB45 000A3630 EB49 4B205631 FB5C CD12F0 EB5F C9 EB60 AF | 2FOMSG: DB ODH.OAH./CP/M 2.2 SD SYSTEMS-KIBLFR ' 5) DB ODH.OAH./B*SD A48 ' DB ODH.OAH./B*SD A48 ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRF LIST: CALL SDPROM+12H RET : PRSTAT XRA A |
| EB3A 300A3922 EB45 000A3630 EB49 4B205631 FB5C CD12F0 EB5F C9 EB60 AF | 2FOMSG: DB ODH, OAH, 'CP/H 2.2 SD SYSTEMS-KIBLFR ' SJ DB ODH, OAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZF10+'0', MSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : : INSERT YOUR ROWTINF HFRF LIST: CALL SDPROM+12H RFT : PRSTAT XRA A RET : RET''RN ALWAYS NOT READY |
| EB3A 300A3922 EB45 000A3A30 EB49 4B205A31 FB5C CD12F0 EB5F C9 EB60 AF | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A4B ' DB ODH, GAH, 'B'SD A4B ' PB OCH, GAH, 'B'SD A4B ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR RO'TINF HFPR LIST: CALL SDPROM+12H RFT : PRSTAT XRA A RFT : RET''RN ALWAYS NOT READY : PUNCH PAPER TAPE. |
| EB3A 30043922 F644 00434510 E649 48205431 F85C CD12F0 E85F C9 E856 AF F861 C9 | 2FOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' SJ DB ODH, OAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROM-12H RET : PRSTAT XRA A RET : : PUNCH PAPER TAPE. : PUNCH : |
| EB3A 300A3922 EB45 000A3630 EB49 4B205631 FB5C CD12F0 EB5F C9 EB60 AF | 2FOMSG: DB ODH, OAH, CP/M 2.2 SD SYSTEMS-KIBLFR ADB ODH, OAH, 'B'SD A48 ' DB ODH, OAH, 'B'SD A48 ' DB OT, OAH, 'B'SD A48 ' K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH-12H RET : PRSTAT XRA A RET : RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET : RETURN FROM PUNCH. |
| EB3A 30043922 Fb44 00439430 EB49 4B205631 FB5C CD12F0 EB5F C9 EB50 AF FB61 C9 | 2FOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-K{BLFR ' SJ DB ODH, OAH, 'B'SD A4B ' DB ODH, JAH, 'MSIZF 10+'0', MSIZF MOD 10 + '0' 2F DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINE HFRF LIST: CALL SDPROM-12H RET : PRSTAT XRA A RET : : PUNCH PAPER TAPE. : PUNCH: RET : RETURN PROM PUNCH. |
| EB3A 30043922 Fb44 00439430 EB49 4B205631 FB5C CD12F0 EB5F C9 EB50 AF FB61 C9 | 2FOMSG: DB ODH, OAH, CP/M 2.2 SD SYSTEMS-KIBLFR ADB ODH, OAH, 'B'SD A48 ' DB ODH, OAH, 'B'SD A48 ' DB OT, OAH, 'B'SD A48 ' K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH-12H RET : PRSTAT XRA A RET : RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET : RETURN FROM PUNCH. |
| EB3A 30043922 Fb44 00439430 EB49 4B205631 FB5C CD12F0 EB5F C9 EB50 AF FB61 C9 | 2FOMSG: DB ODH, OAH, 'CP/H 2.2 SD SYSTEMS-KIBLFR ' DB ODH, OAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZF MOD 10 + 'O' 2F DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROM+12H RET : PRSTAT XRA A RET : PUNCH PAPER TAPE. : UNCH ALLY 'SED TO READ PAPER TAPE. |
| EB3A 3D0A3922 Fb4< 00A3A30 EB49 4B205631 Fb5C CD12F0 EB5F C9 EB56 AF Fb61 C9 EB62 C9 | 2FOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, OAH, 'B'SD A48 ' DB ODH, OAH, 'B'SD A48 ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH-12H RET : RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET : RETURN PROM PUNCH. : NORMALLY "SED TO READ PAPER TAPE. : READER: |
| EB3A 30043922 Fb44 00439430 EB49 4B205631 FB5C CD12F0 EB5F C9 EB50 AF FB61 C9 | 2FOMSG: DB ODH, OAH, 'CP/H 2.2 SD SYSTEMS-KIBLFR ' DB ODH, OAH, 'B'SD ALB ' DB ODH, JAH, 'MSIZF MOD 10 + 'O' 2F DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROM+12H RET : PRSTAT XRA A RET : PUNCH PAPER TAPE. : UNCH ALLY 'SED TO READ PAPER TAPE. |
| EB3A 3D0A3922 Fb4< 00A3A30 EB49 4B205631 Fb5C CD12F0 EB5F C9 EB56 AF Fb61 C9 EB62 C9 | ZFOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-K1BLFR ' G3 DB ODH, OAH, 'B'SD A48 ' B ODR, OAH, 'B'SD A48 ' I DB CF, OAH, 'B'SD A48 ' I NSET YOUR ROUTINF HFFF LIST: CALL SDPROM-12H RET : RFTURN ALWAYS NOT READY : PUNCH PAPER TAPE. : NORMALLY "SED TO READ PAPER TAPE. : RET : NORMALLY "SED TO READ PAPER TAPE. : RET : SET : RET |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB4C CD12F0 Eb5F C9 EB60 AF FB61 C9 EB62 C9 EB63 C9 | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLER 'J DB ODH, GAH, 'B'SD ALB 'DB ODH, GAH, 'B'SD ALB ', NSIZF 'ZF DB 'K VI.0 of 2/01/84 ', O ': INSERT YOUR RO'TINF HFPR IST CALL SDPROM+12H ': INSERT YOUR RO'TINF HFPR NOT READY P PRSTAT XRA A RET : ': PUNCH |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB5C CD12F0 EB5F C9 EB56 AF FB51 C9 EB52 C9 EB53 C9 EB54 EB | 2FOMSG: DB ODH.GAH./CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH.GAH./BrSD A45 ' DB ODH.GAH./BrSD A45 ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : PRSTAT XRA A RET : RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET : RETURN FROM PUNCH. : NORMALLY "SED TO READ PAPER TAPE. ; NORMALLY "SED TO READ PAPER TAPE. ; READER: RET : RETURN FROM READER. ;SECTAR TRANSLATION ROUTINE FOLLOWS |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB4C CD12F0 Eb5F C9 EB60 AF FB61 C9 EB62 C9 EB63 C9 | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A4B ' DB ODH, GAH, 'B'SD A4B ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH+12H RFT : PRSTAT XRA A RET ; RETURN FROM PUNCH. : PUNCH PAPER TAPE. : PUNCH : RET ; RETURN FROM PUNCH. : NORMALLY 'SED TG READ PAPER TAPE. ; READER: RET ; RETURN FROM READER. ; SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DAD B |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB5C CD12F0 EB5F C9 EB56 AF FB51 C9 EB52 C9 EB53 C9 EB54 EB | 2FOMSG: DB ODH.GAH./CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH.GAH./BrSD A45 ' DB ODH.GAH./BrSD A45 ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : PRSTAT XRA A RET : RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET : RETURN FROM PUNCH. : NORMALLY "SED TO READ PAPER TAPE. ; NORMALLY "SED TO READ PAPER TAPE. ; READER: RET : RETURN FROM READER. ;SECTAR TRANSLATION ROUTINE FOLLOWS |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB4C CD12F0 Eb5F C9 EB60 AF FB61 C9 EB62 C9 EB64 EB EB65 096E EB67 2600 | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A4B ' DB ODH, GAH, 'B'SD A4B ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFF LIST: CALL SDPROH+12H RFT : PRSTAT XRA A RET ; RETURN FROM PUNCH. : PUNCH PAPER TAPE. : PUNCH : RET ; RETURN FROM PUNCH. : NORMALLY 'SED TG READ PAPER TAPE. ; READER: RET ; RETURN FROM READER. ; SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DAD B |
| EB3A 3D0A3922 EB44 020A3A30 EB49 4B205A31 FB5C CD12F0 EB5F C9 EB60 AF FB61 C9 EB62 C9 EB64 C9 EB64 EB EB65 096E | 2FOMSG: DB ODH, GAH, 'CP/H 2.2 SD SYSTEMS-KIBLER ' DB ODH, GAH, 'B'SD ALB ' DB ODH, GAH, 'B'SD ALB ' PB OCH, SAH, 'MSIZF MOD 10 + 'O' 2F DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROWINS HERE LIST: CALL SDROM+12H RET : RETURN ALWAYS NOT READY PVNCH PAPER TAPE. : PUNCH PAPER TAPE. : PUNCH: RET : RETURN FROM PUNCH. : NORMALLY "SED TO READ PAPER TAPE. : RET : RETURN FROM READER. SECTOR TRANSLATION ROWINE FOLLOWS SECTRAN XCHG DAD B HOV L/H |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB4C CD12F0 Eb5F C9 EB60 AF FB61 C9 EB62 C9 EB64 EB EB65 096E EB67 2600 | 2FOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, OAH, 'B'SD A4B ' DB ODH, OAH, 'B'SD A4B ' DB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFFE LIST: CALL SDPROH-12H RET :RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH PAPER TAPE. : PUNCH : RET :RETURN FROM PUNCH. : NORMALLY 'SED TO READ PAPER TAPE. : READER: RET :RETURN FROM READER. :SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DA B MOV L/M MVI H,0 |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 FB4C CD12F0 Eb5F C9 EB60 AF FB61 C9 EB62 C9 EB64 EB EB65 096E EB67 2600 | ZFOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' 'J DB ODR, OAH, 'B'SD A48 ' 'D ODR, OAH, 'B'SD A48 ' O' 'ZF DB 'K V1.0 of 2/01/84 ',0 :WRITE A CHARACTER ON LIST DEVICE. :INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET :RETURN ALWAYS NOT READY :PUNCH PAPER TAPE. : :NORMALLY 'SED TG READ PAPER TAPE. : NORMALLY 'SED TG READ PAPER TAPE. : RET :READER: RET RET :RETURN FROM READER. :SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG B MOV L,H MVI H,0 RET RET |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 P55C CD12F0 EB5F C9 EB60 AF Fb61 C9 EB62 C9 EB64 C9 EB64 EB EB65 096E FB67 2600 EB69 C9 | ZFOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' 'J DB ODR, OAH, 'B'SD A48 ' 'D ODR, OAH, 'B'SD A48 ' O' 'ZF DB 'K V1.0 of 2/01/84 ',0 :WRITE A CHARACTER ON LIST DEVICE. :INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET :RETURN ALWAYS NOT READY :PUNCH PAPER TAPE. : :NORMALLY 'SED TG READ PAPER TAPE. : NORMALLY 'SED TG READ PAPER TAPE. : RET :READER: RET RET :RETURN FROM READER. :SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG B MOV L,H MVI H,0 RET RET |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205431 P55C CD12F0 EB5F C9 EB60 AF Fb61 C9 EB62 C9 EB64 C9 EB64 EB EB65 096E FB67 2600 EB69 C9 | ZFOMSG: DB ODH, OAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' R3 DB ODR, OAH, 'B'SD A48 ' 'DB OK, OAH, 'B'SD A48 ' O' 'INSERT YOUR ROUTINF HFRE STATUST ROUTINF HFRE LIST: CALL SDPROH-12H RET :RETURN ALWAYS NOT READY 'PUNCH PAPER TAPE. 'S RET :RETURN FROM PUNCH. : NORMALLY 'SED TO READ PAPER TAPE. : NORMALLY 'SED TO READ PAPER TAPE. : SECTOR TRANSLATION ROUTINE FOLLOWS SECTOR TRANSLATION SOLTINE FOLLOWS SECTOR TRANSLATION ROUTINE FOLLOWS SECTOR TRANSLATION SOLTINE FOLLOWS |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 EB60 AF Fb61 C9 Eb62 C9 Eb64 C9 Eb64 C9 Eb65 096E Eb65 096E Eb65 C9 Eb64 C9 Eb65 C9 Eb64 C9 | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A4B ' DB ODH, GAH, 'B'SD A4B ' DB OTH, GAH, 'B'SD A4B ' I ODB 'K VI.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFPE LIST: CALL SDPROM+12H RFT : PRSTAT XRA A RET :RETURN ALWAYS NOT READY : PUNCH PAPER TAPE. : PUNCH: RET :RETURN FROM PUNCH. : READER: RET :RETURN FROM READER. :SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DAD B MOV L,M MUI H,0 RET : DISK DATA STORAGE AREA DONOT CHANGE |
| EB3A 3D0A3922 EB44 020A3A30 EB49 4B205A31 EB49 4B205A31 EB5F C9 EB56 C9 EB56 C9 EB56 C9 EB56 C9 EB56 C9 EB56 C9 EB56 C9 EB56 - | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-K1BLFR ' G3 DB ODK, GAH, 'B'SD A48 ' '' DB OCK, GAH, 'B'SD A48 ' '' DB 'K V1.0 of 2/G1/84 ',0 '' WRITE A CHARACTER ON LIST DEVICE. 'INSERT YOUR ROUTINF HFRE LIST: CALL SDPROM-12H RET 'RETURN ALWAYS NOT READY '' PUNCH PAPER TAPE. '' '' '' NORMALLY ''SED TO READ PAPER TAPE. '' '' '' RET '' '' '' '' '' '' '' '' '' ODE B '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' '' < |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 Eb5F C9 Eb60 AF Fb61 C9 Eb62 C9 Eb64 C9 Eb65 C9 Eb65 096E Eb65 096E Eb64 EB Eb65 09 Eb64 EB | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' 'J DB ODR, GAH, 'B'SD A4B ' 'DB ODR, GAH, 'B'SD A4B ' ' 'DB CR, GAH, 'B'SD A4B ' ' 'DB STATUAH, 'B'SD A4B ' ' 'DB 'K V1.0 of 2/01/84 ',0 ' :WRITE A CHARACTER ON LIST DEVICE. :INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH+12H RET :RET ':RETURN FROM PUNCH. : NORMALLY 'SED TG READ PAPER TAPE. :NORMALLY 'SED TG READ PAPER TAPE. ' : NORMALLY 'SED TG READ PAPER TAPE. : SECTOR TRANSLATION ROUTINE FOOLLOWS SECTOR TRANSLATION ROUTINE POLLOWS SECTAN XCHG DAD MOV L,M MU H,0 RET : : DISK DATA STORAGE AREA DONOT CHANGE BEGDAT EQU \$ DIRBUF: DS : DISK DATA STORAGE AREA DONOT CHANGE |
| EB3A 3D0A3820 EB44 020A3840 EB44 4B205631 FB5C CD12F0 EB5F C9 EB60 AF FB61 C9 EB62 C9 EB64 C9 EB64 C9 EB64 C9 EB65 096E EB65 096E EB65 C9 EB66 C9 EB66 EB EB65 E9 EB66 EB | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' GD ODK, GAH, 'B'SD A48 ' GD ODK, GAH, 'B'SD A48 ' IDB OK, GAH, 'B'SD A48 ' ' DB ' DB ' DB ' NTTF A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : ' PNSTAT XRA A RET : ' PUNCH PAPER TAPE. ' PUNCH PAPER TAPE. ' : ' NORMALLY ''SED TO READ PAPER TAPE. : NORMALLY ''SED TO READ PAPER TAPE. : : NORMALLY ''SED TO READ PAPER TAPE. : : : : NORMALLY ''SED TO READ PAPER TAPE. : : : : : : : : : : : : : : : : : : |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 EB60 AF FB61 C9 EB64 C9 EB64 EB EB65 096E Eb65 096E Eb64 EB Eb65 09 EB64 EB Eb65 09 EB64 EB Eb65 09 | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A4B ' DB ODH, GAH, 'B'SD A4B ' DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFPE LIST: CALL SDPROH+12H RFT : PRSTAT XRA A RET : RETURN FROM PUNCH. : PUNCH PAPER TAPE. : PUNCH SET : RETURN FROM PUNCH. : READER: RET : RETURN FROM READER. :SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DAD B HOV L,M MVI H,0 RET : DISK DATA STORAGE AREA DONOT CHANGE BEGDAT EQU S DIRBUFY DS 128 :DIRECTORY ACCESS BUFFER ALVO: DS 31 CSVG: DS 16 |
| EB3A 3D0A3922 EB44 00A3A50 EB49 4B205A31 EB49 4B205A31 EB5F C9 EB60 AF FB61 C9 EB62 C9 EB64 C9 EB64 C9 EB64 C9 EB64 C9 EB64 C9 EB65 096E EB66 09 EB66 0 EB66 0 EB66 0 EB68 | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' GD ODR, GAH, 'B'SD A48 ' DB OER, GAH, 'B'SD A48 ' '' DB 'K V1.0 of 2/01/84 ',0 '' WRITE A CHARACTER ON LIST DEVICE. ! INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET :RET '' PUNCH PAPER TAPE. '' PUNCH PAPER TAPE. '' PUNCH: '' RET '' NORMALLY "SED TO READ PAPER TAPE. '' PUNCH: '' RET '' RET '' NORMALLY "SED TO READ PAPER TAPE. '' NORMALLY "SED TO READ PAPER TAPE. '' RET '' RET '' RET '' NORMALLY "SED TO READ PAPER TAPE. '' '' '' NORMALLY "SED TO READ PAPER TAPE. '' '' '' '' '' '' '' '' '' '' '' '' |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 Eb5F C9 Eb60 AF Fb61 C9 Eb62 C9 Eb64 EB Eb65 095E Eb64 EB Eb65 095E Eb64 Eb65 e9 Eb66 = Eb66 = Eb66 = Eb66 = Eb66 = Ec0A EC39 | 2FOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR ' DB ODH, GAH, 'B'SD A48 ' DB ODH, GAH, 'B'SD A48 ' DB OT, GAH, 'B'SD A48 ' '' WRITE A CHARACTER ON LIST DEVICE. ' WRITE A CHARACTER ON LIST DEVICE. ' INSERT YOUR ROUTINF HFFE LIST: CALL SDPROH-12H RET 'RETURN FROM PUNCH. '' PRSTAT XRA A RET 'RETURN FROM PUNCH. '' PUNCH PAPER TAPE. '' PUNCH FAPER TAPE. '' PUNCH SED TG READ PAPER TAPE. '' READER: RET 'RETURN FROM PUNCH. '' READER: RET 'RETURN FROM READER. 'SECTOR TRANSLATION ROUTINE FOLLOWS SECTAAN XCHG DAD B MOV L/M MVI H,0 RET TENP: DS 1 '' ISK DATA STORAGE AREA DONOT CHANGE BEGDAT EQU'S DIRBUF; DS 129 :DIRECTORY ACCESS BUFFER ALVO: DS 31 CSV0: DS 16 ALVI: DS 31 CSV1: DS 16 |
| EB3A 3D0A3922 EB44 00A3A50 EB49 4B205A31 EB49 4B205A31 EB5F C9 EB60 AF FB61 C9 EB62 C9 EB64 C9 EB64 C9 EB64 C9 EB64 C9 EB64 C9 EB65 096E EB66 09 EB66 0 EB66 0 EB66 0 EB68 | ZFOMSG: DB ODH.GAH./GPSD 2.2 SD SYSTEMS-KIBLFR ' DB ODH.GAH./GPSD 2.45 ' DB DB ODH.GAH./SIZE/13+'O',HS12F HOD 10 + '0' ZF DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : ? PUNCH PAPER TAPE. : INSERT XRA A RET : ? PUNCH PAPER TAPE. : NORMALLY "SED TO READ PAPER TAPE. : SECTAR TRANSLATION ROUTINE FOLLOWS SECTAR TRANSLATION ROUTINE FOLLOWS SECTAR TAN STORAGE AREA DONOT CHANGE BEGDAT EQU S SIRECTORY ACCESS BUFFER I, DISK DATA STORAGE AREA DONOT CHANGE BEGDAT EQU S SI DIRBUF: DS 128 SIRECTO |
| EB3A 3D0A3922 Eb44 00A3A30 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 Eb5F C9 Eb60 AF Fb61 C9 Eb62 C9 Eb64 EB Eb65 095E Eb64 EB Eb65 095E Eb64 Eb65 e9 Eb66 = Eb66 = Eb66 = Eb66 = Eb66 = Ec0A EC39 | ZFOMSG: DB ODH.GAH./GPSD 2.2 SD SYSTEMS-KIBLFR DB ODH.GAH./GPSD A45 DB CK.GAH./GPSD A45 DB CK.GAH./GPSD A45 CF DB SCOMP.GAH./MSIZE/13+'0',HSIZF HOD 10 + '0' INSERT YOUR ROUTINF HERE LIST: CALL SDPROH-12H RET : PRSTAT XRA A RET : PUNCH: RET : PUNCH PAPER TAPE. : PUNCH: : RET : PUNCH: : RET : NORMALLY "SED TO READ PAPER TAPE. : SECTAR NCHG DAD B MOV L/M MU H.0 RET : |
| EB3A 3D0A3922 Eb44 00A3A50 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 Eb5F C9 Eb60 AF Fb61 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb65 - Eb65 - | ZFOMSG: DB ODH.GAH./GPSD 2.2 SD SYSTEMS-KIBLFR ' DB ODH.GAH./GPSD 2.45 ' DB DB ODH.GAH./SIZE/13+'O',HS12F HOD 10 + '0' ZF DB 'K V1.0 of 2/01/84 ',0 : WRITE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : ? PUNCH PAPER TAPE. : INSERT XRA A RET : ? PUNCH PAPER TAPE. : NORMALLY "SED TO READ PAPER TAPE. : SECTAR TRANSLATION ROUTINE FOLLOWS SECTAR TRANSLATION ROUTINE FOLLOWS SECTAR TAN STORAGE AREA DONOT CHANGE BEGDAT EQU S SIRECTORY ACCESS BUFFER I, DISK DATA STORAGE AREA DONOT CHANGE BEGDAT EQU S SI DIRBUF: DS 128 SIRECTO |
| EB3A 3D0A3922 Eb44 00A3A50 Eb49 4B205A31 Eb49 4B205A31 Eb5F C9 Eb5F C9 Eb60 AF Fb61 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb64 C9 Eb65 - Eb65 - | ZFOMSG: DB ODH, GAH, 'CP/M 2.2 SD SYSTEMS-KIBLFR SJ DB ODR, GAH, 'B'SD A4B 'DB ODR, GAH, 'B'SD A4B 'DB OK, GAH, 'B'SD A4B 'DB SYSTEMS-KIBLFR 'DB OK, GAH, 'B'SD A4B 'RTTE A CHARACTER ON LIST DEVICE. : INSERT YOUR ROUTINF HFPE LIST: CALL SDPROH+12H RET : RETT 'PNCH PAPER TAPE. 'PUNCH: RET 'RET : RETURN FROM PUNCH. ' NORMALLY 'SED TO READ PAPER TAPE. 'PUNCH: RET 'READER: RET RET : RETURN FROM PUNCH. ' NORMALLY 'SED TO READ PAPER TAPE. 'SECTOR TRANSLATION ROUTINE FOLLOWS SECTRAN XCHG DAD DAD B MOV L,M MUT H.0 RET TENP: 'S 128 JIBBUF: DS 128 JIBUF: DS 129 JIBUF: DS 16 ALV0: DS JIS 16 ALV1: DS 31 </td |
| EB3A 3D0A3922 EB44 00A3A30 EB49 4B205A31 EB49 4B205A31 EB5F C9 EB60 AF FB61 C9 EB64 C9 EB64 C9 EB64 C9 EB64 C9 EB65 096E EB65 096E EB65 C9 EB66 - EB66 - E66B - E66B - E66B - E66B - E66B - E66A - E63A - E64A - E64 | ZFOMSG: DB ODH.GAH./GPCM 2.2 SD SYSTEMS-KIBLFR ' DB ODH.GAH./BPCM 2.6 SD SYSTEMS-KIBLFR ' DB YENGALS INSERT YOUR ROUTINF HFRE LIST: CALL SDPROH-12H RET : PNSTAT XRA A RET : PUNCH: RET : PUNCH: : RET : NORMALLY ''SED TO READ PAPER TAPE. : SECTAR TRANSLATION ROUTINE FOLLOWS SECTAR TANS COMAGE AREA DONOT CHANGE BEGDAT FOU S 10 : DIS DATA STORAGE AREA DONOT CHANGE BEGDAT FOU S 11 : DIS DATA STORAGE AREA DONOT CH |

BOOT LOADER . :this is a quick bootstrap that loads at track 0 :sector 1 it will be put into memory at 0000 by the :Sdprom bootstrap disk read then it will be :secuted and read in the rest of the first two tracks 003C = F000 = EA00 = 0000 313000 0003 210200 0006 224300 SOPRON FOU 0F000H 5A00H+ (MSIZE=24) *1024 SP,80H H,2H :SET TO 43H :SECTOR BIOS EOU LXI ;SET TO 2ND SECTOR ;SECTOR TO READ ;START OF CPM'S CCP ;DMA LOCATION LXT SHLD LXI SHLD 0006 224300 0009 2100D4 000C 224000 000F 3E33 0011 324500 0014 CD2DF0 0017 C300EA 43H H,BIOS-1600H 40H A,51 45h MVI Sta NUMER OF SECTORS TO READ CALL SDPROM+2DH READ MULTIPLE SECTORS JMP 8105 0014 • N D

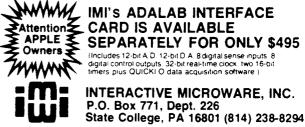


 Interactive Microware's general-purpose ADALAB^{**} data acquisition and control system interfaces with virtually any lab instrument using a recorder or meter, including GC and HPLC systems, spectrophotometers, pH meters, process control apparatus, thermocouples, etc.

 Lab Data Manager[™] software facilitates single or multichannel acquisition, storage, display and chart recorder style output of lab instrument data. IMI QUICKI/O software operates within easy-to-use BASIC!

 Thousands of scientists currently use IMI software and or ADALAB products worldwide!

Price includes 48K APPLE II+ CPU, disk drive with controller, 12¹¹ monitor, dot matrix printer with interface, IMI ADALAB¹¹⁴ interface card. Trademark of Apple Computer inc.



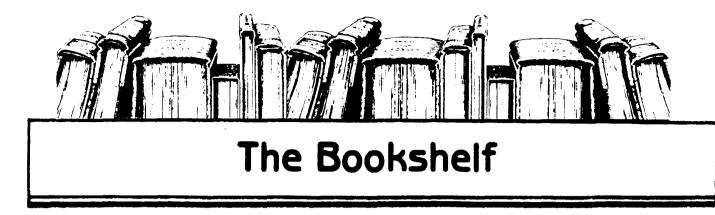
Editor's Page, continued from page 1

class bulk mailing.

It is obvious from looking at these programs that they were designed by a programmer who had no idea of what goes on in the real world. There probably *are* some useful programs available, but the ones we looked at did not fit our needs at all. Fortunately Ernie Brooner, who is writing our multiuser column, offered a data base which he wrote for his own use. It is not exactly what we need, but Ernie said "That's simple—I wrote it, so I'll just change a few lines."

The difference between a program written by a user and one written by a programmer who does not actually use the software himself is obvious to anyone who tries to use the program in a day-to-day basis.

It is time for the software industry to get out in the field and spend some time with their customers to learn what it is that people really do with computer software.



TTL Cookbook

SCRs and Related Thyristor Devices

Instrumentation: Transducers, Experimentation, and Applications

The Programmer's CP/M Handbook

Interfacing to S-100 (IEEE 696) Microcomputers

Microprocessors for Measurement and Control

Osborne CP/M[®] User Guide (Second Edition)

A new revised edition which includes expanded sections on CP/M⁶ 86 and CP/M⁶ 80, as well as CP/M⁶ 's relationship to assembly language programming, MP/M⁶ and CP/NET⁶ operating environments. By Thom Hogan, 292 pages, $6^{1/8}x9^{1/4}$, softbound.........\$15.95.

Discover FORTH

68000 Assembly Language Programming

Z8000^g Assembly Language Programming

This book is filled with real-world programming examples. sample problems. and troubleshooting hints that will guide the reader to mastery of this powerful new 16-bit "super chip". The entire Z8000⁴ instruction set is described in detail. By Lance A Leventhal, Adam Osborne, and Chuck Collins. 928 pages, 6³/₄ x9³/₄, softbound......\$19.99

The 8086 Book

Anyone using, designing, or simply interested in an 8086-based system will be delighted by this book's scope and authority. As the 16-bit microprocessor gains wider inclusion in small computers, this book becomes invaluable as a reference tool which covers the timing, architecture and design of the 8086, as well as optimal programming techniques, interfacing, special features, and more. By Russell Rector and George Alexy, 624 pages, $6^{1/2}x9^{1/4}$, softbound. \$16.99.

11

118

記載

1.1

15**10**

词

網

Z80[®] Assembly Language Programming

Programming examples illustrate software development concepts and actual assembly language usage. More than 80 sample programming problems with solutions and a complete Z80^e instruction set reference table. By Lance A. Leventhal, 640 pages, 6¹/₂x9¹/₄, softbound. \$18.95

8080A/8085 Assembly Language Programming

Microprocessor Circuits, Volume 2: I/O Interfacing & Programmable Controllers

IC Timer Cookbook (2nd edition)

Microprocessor-Based Robotics

Introduces you to robotics - a dynamic new field of science that uses your computing and electronic talents as well as your mechanical and electrical knowledge. First, you'll learn the mechanics of robot hands, arms, and legs; then, tactile sensing, motion and attitude sensing, and vision systems. After that, you learn controlling with microprocessors and BASIC programs, and finally, you learn to control the entire robot system with voice commands! Fascinating and not machine specific. By Mark J. Robillard. 224 pages ±¹/₂x11, softbound. \$16.95

TV Typewritter Cookbook

Microcomputer Math

A step-by-step introduction to binary, octal, and hexidecimal numbers, and arithmetic

operations on all types of microcomputers. Excellent for serious BASIC beginners as well as assembly language programmers. Treats addition and subtraction of binary, multipleprecision and floating-point operations, fractions and scaling, flag bits, and more. Many practical examples and self-tests. By William Barden, 160 pages, 5¹/₂×8^{1/2}, softbound\$11.95

Understanding Digital Logic Circuits

A working handbook for service technicians and others who need to know more about digital electronics in radio, television, audio, or related areas of electronic troubleshooting and repair. You're given an overview of the anatomy of digital-logic diagrams and introduced to the many commercial IC packages on the market. By Robert G. Middleton. 392 pages, 5½ x8½, softbound. \$18.95.

CMOS Cookbook

SCRs and Related Thyristor Devices

A comprehensive guidebook to the operational theory and practical applications for silicon controlled rectifiers, triacs, diacs, unijunction transistors, and other members of the thyristor family. Also contains a microprocessor mini-course to help you in interfacing thyristors with digital control circuits. If you're involved with design, installation, or maintenance of electronic power-control equipment, this is the book for you. By Clay Laster, 136 pages, $8/\pi 11$, softbound.

Real Time Programming: Neglected Topics

This book presents an original approach to the terms. skills, and standard hardware devices needed to connect a computer to numerous peripheral devices. It distills technical knowledge used by hobbyists and computer scientists alike to useable, comprehensible methods. It explains such computer and electronics concepts as simple and hierarchical interrupts, ports, PIAs, timers, converters, the sampling theorem, digital filters, closed loop control systems, multiplexing, buses, communication, and distributed computer systems. By Caxton C. Foster, 190 pages, $6^{1}x5^{94}$, softbound.

Interfacing Microcomputers to the Real World

Here is a complete guide for using a microcomputer to computerize the home, office, or laboratory. It shows how to design and build the interfaces necessary to connect a microcomputer to real-world devices. With this book, microcomputers can be programmed to provide fast, accurate monitoring and control of virtually all electronic functions – from controlling houselights, thermostats, sensors, and switches, to operating motors, keyboards, and displays. This book is based on both the hardware and software principles of the Z80 microprocessor (found in several minicomputers, Tandy Corporation's famous TRS-80, and others). By Murray Sargent III and Richard Shomaker, 288 pages, 6¹⁴ x9¹⁴, \$15.55

IC Timer Cookbook

CP/M Primer

Soul of CP/M: Using and Modifying CP/M's Internal Features

The S-100 and Other Micro Buses (2nd Edition)

Examines microcomputer bus systems in general and 21 of the most popular systems in particular, including the S-100. Helps you expand your computer system through a better understanding of what each bus includes and how you can interface one bus with another. By Elmer C. Poe and James C. Goodwin, II. 208 pages. $5^1 + x5^1 + soft = -1981$995$

Interfacing & Scientific Data Communications Experiments

Active-Filter Cookbook

A practical discussion of the many active-filter types and uses, written by one of Sams most popular authors. Teaches you how to construct filters of all types, including high-pass, low-pass, and bandpass having Bessel. Chebyshev, or Butterworth response. Easy to understand – no advanced math or obscure theory. Can also be used as a reference book for analysis and synthesis techniques for active-filter specialists. By Don Lancaster: 240 pages, $5^{10}x8^{10}x$, 514.95

IC Converter Cookbook

Discusses and explains data conversion fundamentals, hardware, and peripherals A valuable guide to help you understand and use d/a and a/d converter applications. Includes manufacturers' data sheets. By Walter G. Jung. 576 pages, 5½x8½x, soft. 1978....\$14.95

IC Op-Amp Cookbook

An informal, easy-to-read guide covering basic op-amp theory in detail, with 200 practical, illustrated circuit applications to reflect the most recent technology. JFET and MOSFET units are shown in both single and multiple formats. Includes manufacturers' data sheets, and lists addresses of the companies whose products are featured. By Walter G Jung. 480 pages, $5^{1/2}x8^{1/2}$, soft. ©1980......\$15.95

Regulated Power Supplies (3rd Edition)

The Computer Journal

PO Box 1697 Kalispell, MT 59903

| Order [| Date: | | | |
|-----------|------------|---------------------------------------|--------|--|
| Print Nam | ne | · | ······ | |
| Address_ | | · · · · · · · · · · · · · · · · · · · | | |
| City | <u></u> | State | Zip | |
| | Check | Mastercard | □Visa | |
| Card No. | | | pires | |
| Signature | for Charge | | | |

| Qty | Title | | Price | Total |
|---|-------|------------------------|-------|-------|
| | | | | |
| | | | | |
| | | 1 | | |
| | | | | |
| | | | | |
| Shipping charges are: \$1.00 for the first book, and \$.50 for all subsequent books. Please allow 4 weeks for delivery. | | Book Total Shipping | | |
| | | | | |
| | | | TOTAL | |

New Products

SYBEX Releases "Mastering CP/M."

Mastering CP/M, an advanced guide to using, altering, and adding features to the CP/M microcomputer operating system, has just been released by SYBEX. CP/M users and systems programmers will better understand the organization and operation of CP/M with this book. The BIOS (Basic Input/Output System) and the BDOS (Basic Disk Operating System), are described in detail, illuminating for the reader the subtleties of the useful CP/M system. Macros instructions, powerful tools that enable programmers to design more efficient assembly language programs, are introduced, and a valuable library of macros is developed.

This well-written and fascinating book takes the reader on a step-by-step journey of discovery, leading to a more thorough understanding of the organization and operation of CP/M. An important set of appendices is included, making this a comprehensive reference for CP/M users and programmers. The book is priced at \$17.95. Add \$1.50 for postage when ordering directly from SYBEX, 2344 Sixth Street, Berkeley, CA, 94710.

Free Thermistor Catalog

Thermometrics, Inc. of Edison, New Jersey announces the publication of it's 52 page Thermistor Catalog number 181-D. The new catalog will prove to be of great value to anyone who has to design, specify or use thermistors and thermistor networks. Some of the useful features are as follows.

• A four page foldout "Thermistor Selection Guide" which provides comparison of all the styles and sizes of thermistors at Thermometrics, and includes physical, thermal and electrical properties for each type.

• A review of the extensive calibration and test facilities and services available at Thermometrics.

• A technical applications and data section which includes definitions of thermistor terminology, the various equations which describe the thermistor R-vs-T characteristics, a discussion of curve tolerances and two design examples on linearized voltage and resistance networks including output "S" curves for different material systems.

• A product section for each of the standard thermistor types available detailing all dimensions, R-vs-T characteristics, thermal properties, options and ordering information.

In addition to the new catalog there are some very useful application notes available which deal with thermistor theory, measurement, design techniques, stability and theory of self heated thermistors (including their use in flow measurement.) This information is available free of charge to interested readers from Thermometrics, 808 US Highway 1, Edison, New Jersey, 08817, Tel. 201-287-2870.

FORTH Tutorial at Half Price

MicroMotion announces the availability of the FORTH-79 Tutorial & Reference Manual at half price (\$10.00). This professionally written manual was the first complete FORTH tutorial to teach the FORTH computer language, including FORTH-79 and FIG-FORTH. It has been replaced with their new publication, FORTH Tools (\$20.00), which teaches the new 1983 International Standard. For further information contact MicroMotion, 12077 Wilshire Blvd. #506, Los Angeles, CA, 90025, Tel. 213-821-4340.

(a)

84**8**

1.1

눼

آهي.

أهرر

. á

- 4

Interface Breadboard Package from Group Technology

The Color Computer Expansion Connector Breadboard, Model CC-100, for the TRS-80 Color Computer 1 or 2 makes it possible to connect external devices to the expansion connector signals of the computer. Combined with a solderless breadboard and the book TRS-80 Color Computer Interfacing, With Experiments (book no. 21893), it forms the CoCo-100 package providing basic interfacing instructions for any version of this versatile computer. In addition, the CC-100 Experiment Component Package contains the parts necessary to do the experiments in the book.

With the CoCo-100, the user can learn in step-by step fashion how to access the signals available in the parallel expansion connector of the TRS-80 Color Computer and how to construct and use a peripheral interface adapter (PIA). The experiments demonstrate how to enter and retrieve binary data and how analog-to-digital and digital-to-analog conversion is performed both within the computer and using external devices. With the fundamental understanding and hands-on experience developed through the interface package, users are well-equiped to extend their interfacing capabilities to a variety of applications.

Readers and reviewers alike have praised Andy Staugaard's book for its clarity and thoroughness. The aspiring experimenter needs only a working knowledge of Color BASIC programming and the binary number system (reviewed in the Appendix) to embark on a delightful journey toward proficiency in interfacing. The reader is shown how to construct input/output (I/O) ports and to use them to connect the computer to the mostly analog world that lies outside.

Model CoCo-100, Interface Breadboard Package, is priced at \$51.25, a 10% reduction from the cost of the individual components, plus \$2.50 shipping. Virginia residents add 4% sales tax. VISA and Master Cards accepted. For purchase or further information, contact Group Technology, Ltd., PO Box 87, Check,VA, 24072, Tel. 703-651-3153.