To the Graduate Council:

I am submitting herewith a thesis written by Robert C. Fusarini entitled "A Construction and Utilization of a Basic Environmental Simulation Model." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Randall E Cline

Randall E. Cline, Major Professor

We have read this thesis
and recommend its acceptance:

R. P. Leinius

Asa C. Bishop, Jr.

Accepted for the Council:

Hilton A. Smith

Vice Chancellor
Graduate Studies and Research

A CONSTRUCTION AND UTILIZATION OF A BASIC ENVIRONMENTAL SIMULATION MODEL

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee

Robert C. Fusarini

December 1974

ACKNOWLEDGMENTS

I wish to take this opportunity to thank several people who made this thesis and my graduation possible. Dr. Asa O. Bishop and Dr. Ronald P. Leinius gave me encouragement, assistance and instruction while attending The University of Tennessee.

Clarence A. Thompson, my previous supervisor, made it possible for me to receive a leave of absence to enter graduate school and Dr. Randall E. Cline's excellent guidance made it possible for me to graduate. He forced me to dig a little bit deeper into my own abilities. These men demonstrate that excellence in any profession includes a sincere concern for others, even though each states he is "just doing my job."

To Mary L. Noblitt, who did an excellent job typing this thesis, I express a special thanks for doing those errands that I could not do while out-of-state.

1176051

ABSTRACT

This thesis extends Conway's game of Life into a general purpose simulation model, called the Basic Environmental Model, that can be used especially to examine abstractions of phenomena occurring in the environment. This model uses a two-dimensional checkerboard-type cellular space composed of automata which act within a set of time-dependent rules defining the effect of a cell upon its neighbors.

Because the evaluation of the deterministic and probabilistic relationships within this space can be a tedious manual process, the Basic Environmental Model was implemented on a computer with an interactive capability to monitor and modify the relationships during the simulation.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	BACKGROUND	3
III.	FORMULATION	10
IV.	IMPLEMENTATION	17
V.	SUMMARY AND CONCLUSIONS	23
BIBLIOGRAPHY		24
APPENDICES		27
A.	COMMAND FORMATS AND DATA STRUCTURES	28
B.	BEM EXAMPLES	34
C.	FLOWCHARTS AND PROGRAM	51
VITA		102

LIST OF FIGURES

FIGURE		PAGE
1.	Illustration of a Pattern Evolution in the Game of "Life" .	4
2.	Characteristic Life Pattern Behaviors	6
3.	Illustration of a Pattern Evolution in the Gard-Curtis Forest Fire Simulation	8
4.	The Extended Neighborhood of a Cell	11
5.	Illustrations of Life Factor Duplication	13
6.	An Example of the Relationship Between the Rule and Life Matrices	15
7.	Illustration of Life Factor Display	18
8.	Example of a Single Instruction Upon a Single Cell	22
9.	Miscellaneous Dialog Command Formats	29
10.	Dialog Format of the Cell Instruction Command	30
11.	Cell Instruction Structure	32
12.	Data Structures of Data Items	33

CHAPTER I

INTRODUCTION

Computer simulation and cellular automata can be combined to produce abstractions of the real world applicable to many diverse topics. As illustrated by the variety of papers, journals, and books (1, 2, 5, 6, 8, 11, 12, 17, and 18), computer simulation has been used extensively in the study and evaluation of biological, economic, physical, military, and social systems. A major advantage in the use of models is that it is possible to study the various aspects of the system without altering, destroying, or having to build the actual system. The system is abstracted into a model which includes only the major components but hopefully still adequately represents the system. Although somewhat artificial, frequently lengthy, and sometimes inflexible, computer simulation models have become widely accepted because they are completely reproducible methods of collecting and processing data about systems.

Since its introduction by von Neumann (23), cellular automata theory has been applied to such diverse topics as signal propagation (7), graph theory (10, 16) heuristic processes (17), biological cells (17, 24) and molecules (4). Special purpose computer simulation models have been constructed for some of these systems. In this thesis a general cellular type model is constructed that can be applied to a variety of problems. Although the formulation of such a model can always be

subject to question, the particular structure assumed herein appears to be sufficiently general to include a variety of applications. As indicated in the discussion in subsequent sections, the game of "Life" (12, 13, 14) and a forest fire simulation model (11) have been used as the starting point for the present development. It is obvious, however, that many types of physical phenomena may be abstracted to fit the general framework of this model. The orientation of this thesis therefore was to construct a model with sufficient generality and interactive capability that a user could study the abstraction to gain insight into a particular system.

CHAPTER II

BACKGROUND

In 1936, Turing (21) proposed a class of simple computing machines which could be used to determine the computability of a function in a finite time. He later (22) extended this concept into a Universal Turing Machine that could simulate any of his previous machines. Subsequently, von Neuman (23) proved the existence of a Universal Construction Machine that could reproduce any given machine including itself. It required more than 200,000 cells just for the basic machine plus additional cells to contain the reproduction, and each cell was required to be able to assume one of 29 different states. In contrast to the complexity of the von Neuman machine, Conway (12) proposed the simple game of "Life" that could be "played" within an array of predetermined size (either finite or infinite) in which a cell could be only alive or dead and all state transitions occur simultaneously. If both the number of live cells in an instant and the total number of time intervals, which Feller (9) called epochs, remain small, Life may be played with checkerboard and checkers or with pen and paper. However, as the number of cells and epochs increase, the computer becomes indispensible. Because this thesis extends the basic game of Life into a more general game, stated below are Conway's rules of life and death in which a neighbor is any of the eight adjacent cells and the game is illustrated in Figure 1:

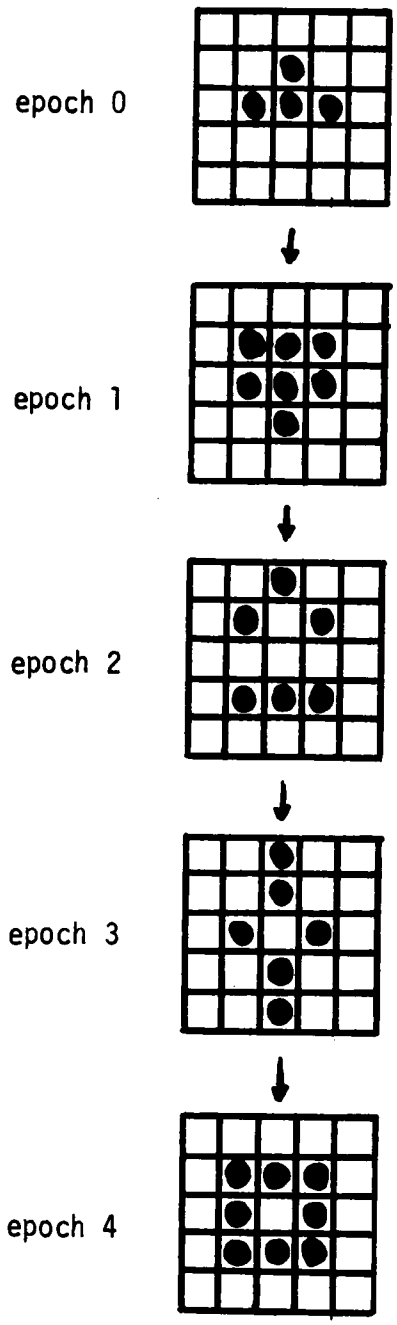


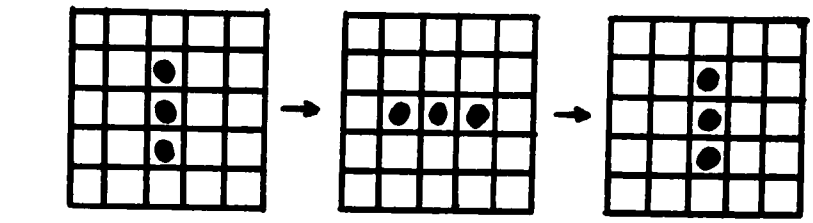
Figure 1. Illustration of a Pattern Evolution in the Game of "Life."

1. Births -- any dead cell adjacent to exactly three live neighbors will be born on the next epoch.
2. Survivals -- any live cell with two or three live neighbors will remain alive for the next epoch.
3. Deaths -- any live cell with none, one, or more than three live neighbors will die in the next epoch.

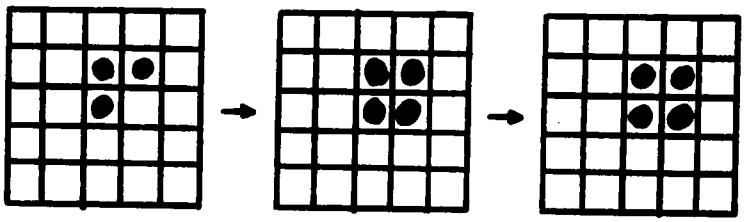
Different initial patterns will give rise to various patterns, but a set of characteristic behaviors has been summarized by Gardner (12) for patterns that have evolved into a systematic growth. In general, patterns may exhibit oscillitory action, stability, or disappearance due to either under or overpopulation as shown in Figure 2.

It is easily seen that the game of Life can be used as an abstraction of physical pehnomena if, for example, a cell represented a small geographical area and the state of the cell indicated that the area was or was not populated by live organisms, Life could be used as a model of population growth and migration. However, this basic model would frequently be inadequate because an area is usually considered to be more than just populated or devoid of life. For example, a population density number could be used to indicate if the population is sparse, dense, or overcrowded. Extending the game of Life to include some measure of the level of activity within a cell may provide a more realistic simulation applicable to a variety of physical problems.

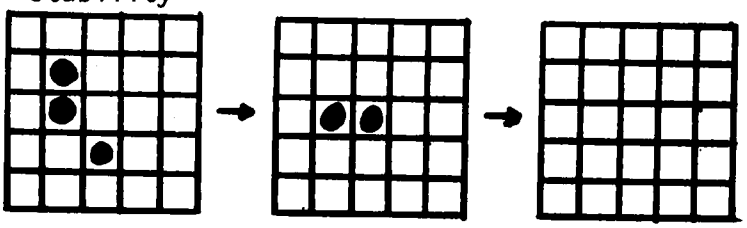
Many specific problems have been simulated with special purpose models which cannot easily be adapted to other problems of a similar nature. An example of special purpose simulations is the Gard-Curtis



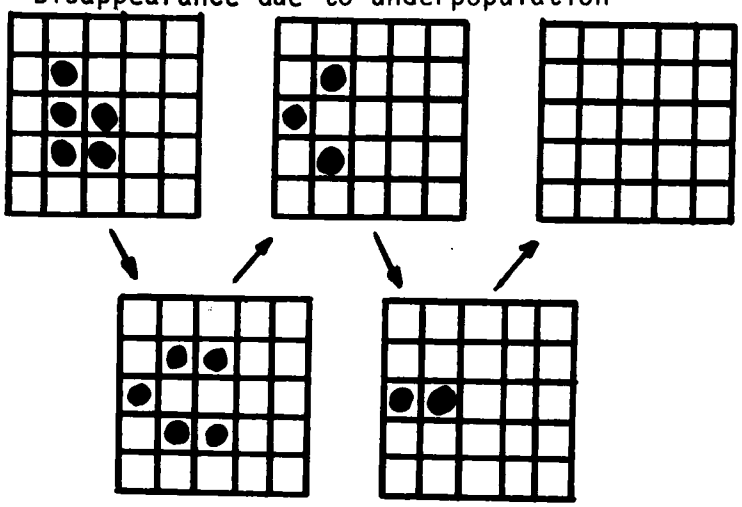
(a) Oscillation



(b) Stability



(c) Disappearance due to underpopulation



(d) Disappearance due to overpopulation

Figure 2. Characteristic Life Pattern Behaviors.

Forest Fire Model which uses a cellular space to represent a woodland area and a set of probabilities to determine if a fire would spread from one area to its orthogonally adjacent neighbor. The situation illustrated in Figure 3 assumes the initial conditions within a densely wooded area to be a twenty mile-per-hour wind blowing toward the South when the humidity is high. The three cell states permitted in this model are indicated in Figure 3 with a "B" to mark cells in the "burning" state, an "O" for cells "burned-out," and "unburned" cells are left blank.

The Gard-Curtis model and the game of Life appear to have little in common except the cellular space. However, a fire spread model of three states can be simulated on an extended game of Life which permits a cell to have two categories of states. The first category could have two states indicating that the cell is or is not burning, and the second category could indicate that the fuel or burnable material does or does not exist in the cell. Thus an unburned area could be represented by a cell with fuel but not burning, then the cell could start burning and remain burning until there was no fuel available at which time the area would become burned-out.

The realism of this fire simulation is improved if the fire is represented as an activity level which, for example, permits the absence of a fire to be indicated by activity level "0," a smoldering fire by level "1," a blaze by level "2," and a raging inferno could be indicated by level "3." Also it is easily seen that if this fuel was not just two states, but a fuel factor of a number of units of burnable

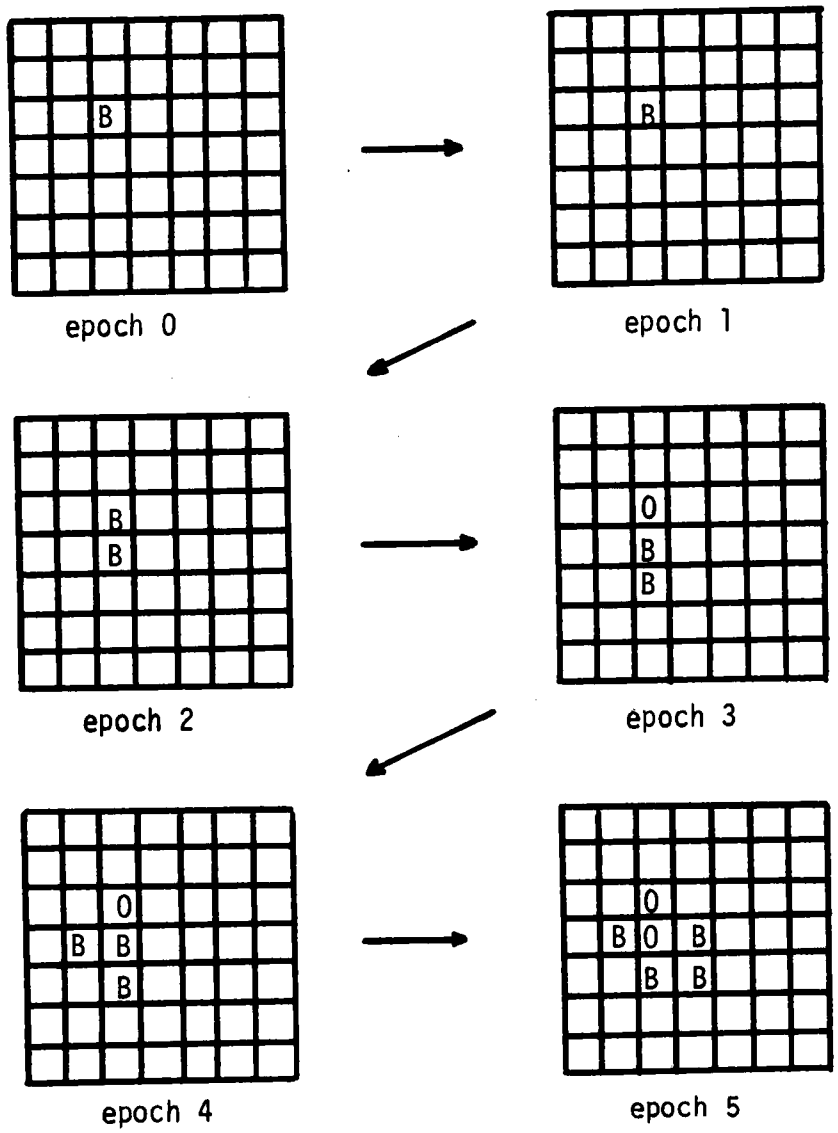


Figure 3. Illustration of a Pattern Evolution in the Gard-Curtis Forest Fire Simulation.

material, then the life span of the fire could be controlled by a set of instructions that would consume the fuel factor at a rate dependent on the activity level of the cell. Thus this "extended" game of Life with instructions, a fuel factor, and an activity level can simulate a fire model.

If this game is yet further extended to permit a cell to have a different influence upon each of its neighbors that could be varied in a random manner, then such a model should be able to simulate other environmental problems. The next section presents the formulation of Basic Environmental Model which meets this criteria and permits simulation of a time-dependent problem which can be abstracted into a two-dimensional space of cells acting uniformly. The actions of these cells are described by a set of rules that define the activity level of a cell in terms of the activity of its neighboring cells in the previous epoch. This model was implemented on a computer using an interactive graphic display technique which permits the monitoring and modification of the spacial relationships.

CHAPTER III

FORMULATION

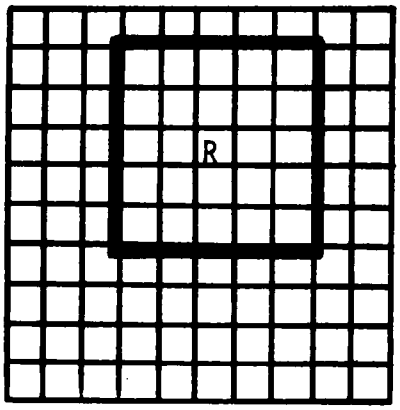
This section contains the definitions of the terms employed in the construction of the Basic Environmental Model (BEM) and a summarization of the concepts applied to determine how long a cell with a given amount of fuel can remain at a certain life level, what life level a cell can attain, what rules are used to indicate how a cell can affect its neighbors, and which cells are to be considered as the neighbors.

A cell in the Gard-Curtis model or Life had only four or eight neighbors, but the BEM extends the neighborhood of a given cell to the square 25-cell area with the given cell as the central element as illustrated in Figure 4(a) by a cell R located in an array M at M(4, 6). References to cells within this extended neighborhood will be made from the central element with

$$R(i, j) \text{ where } -2 \leq i \leq 2 \text{ and } -2 \leq j \leq 2$$

as illustrated in Figure 4(b). (It should be noted that any cell may be within the neighborhood of up to 25 cells as illustrated by the two overlapping neighbors in Figure 4(c).)

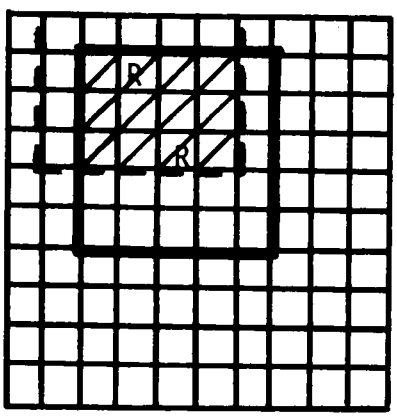
The BEM rates the level of activity of a cell on a life scale which is defined as a measure of the effect of a cell, at time t and located in array M at M(I, J), upon its neighbors in the next epoch t+1 and shall be denoted by



(a) The extended neighborhood of a single cell "R."

2,-2	2,-1	2,0	2,1	2,2
1,-2	1,-1	1,0	1,1	1,2
0,-2	0,-1	0,0	0,1	0,2
-1,-2	-1,-1	-1,0	-1,1	-1,2
-2,-2	-2,-1	-2,0	-2,1	-2,2

(b) The numbering of the neighbors in reference to the central element (0, 0).



(c) The overlapping extended neighborhoods of two cells.

Figure 4. The Extended Neighborhood of a Cell.

$$M_t(I, J) \quad \text{where } 0 \leq M_t(I, J) \leq 1.$$

If it is assumed that the life factor of a cell is duplicated into every neighbor in the next epoch, then the calculated life factor of a cell is denoted by the relationship

$$M'_{t+1}(I, J) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} M_t(I+i, J+j)$$

where $1 \leq I+i \leq S$, $1 \leq J+j \leq S$ and S is the maximum size of the array M. Figure 5(a) illustrates one epoch for a single initial cell with a non-zero life factor, and Figure 5(b) illustrates one epoch for two initially non-zero neighboring cells.

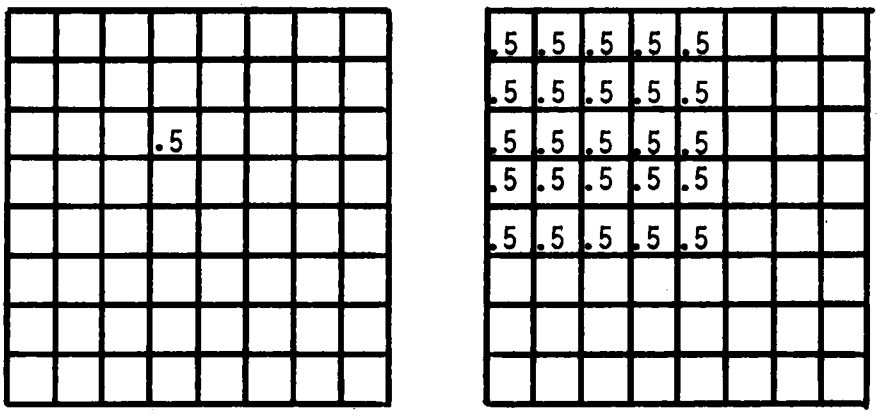
The BEM permits a cell to affect each of its neighbors differently during each epoch as defined by a rule matrix R_t which is a 5 x 5 matrix in which each element is denoted by

$$R_t(i, j) \quad \text{where} \quad 0 \leq R_t(i, j) \leq 1$$

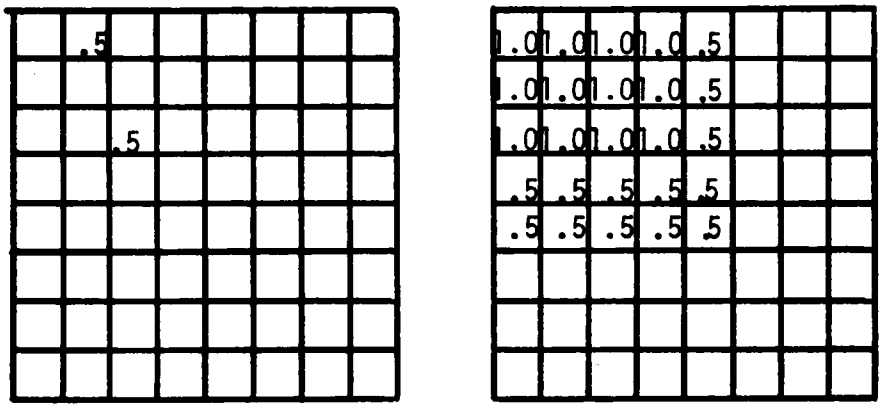
and indicates the portion of the life factor of a given cell, acting as the central element in its neighborhood, which will be added to the calculated life factor of a neighbor in the next epoch. The rule and life matrices are assumed to satisfy the relationship

$$M'_{t+1}(I, J) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} M_t(I+i, J+j) \cdot R_t(i, j).$$

That is to say, the calculated life factor of a cell M(I, J) at time



(a) A single initial cell with a non-zero life factor.



(b) Two initial cells with non-zero life factors.

Figure 5. Illustrations of Life Factor Duplication.

t+1 is assumed to be a linear combination of the effects of the life in all of its neighbors' life factors at time t, where the weights are the $R_t(i, j)$ as defined by the rule matrix. A simple example of this relationship is illustrated in Figure 6. Note that the result in Figure 6(c) resembles the result in Figure 2(a). Moreover, it is to be shown in Section V that Life can be simulated on the BEM using the rule shown in Figure 6(b).

The model created thus far is strictly deterministic, but the BEM can simulate a phenomena of a probabilistic nature such as the Gard-Curtis model by assuming that the rule is subject to a maximum known error. If the user realizes that any $R_t(i, j)$ has a known worst error at time t denoted by

$$E_t, \text{ where } 0 \leq E_t \leq 1,$$

and $E_t=0$ indicates a rule matrix with no error. Then a probabilistic BEM is created by generating a random number

$$E_t[(I, J), (i, j)] \text{ such that } |E_t[(I, J), (i, j)]| \leq E_t$$

and extending the relationship between the rule and life matrices to

$$M'_{t+1}(I, J) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} M_t(I+i, J+j) \cdot R_t(i, j) \cdot A$$

where the accuracy of the rule matrix is denoted by

$$A = 1+E_t[(I, J), (i, j)]$$

(a) The initial life matrix M_t.

		1.0	1.0	1.0		

(b) The rule matrix R_t.

	.33	.33	.33	
	.33		.33	
	.33	.33	.33	

(c) The resulting life matrix M'_{t+1}

	.33	.66	.99	.66	.33	
	.33	.33	.99	.33	.33	
	.33	.66	.99	.66	.33	

Figure 6. An Example of the Relationship Between the Rule and Life Matrices.

The probabilistic nature of the above will be illustrated in Section V by a fire simulation.

An additional condition to be associated with each cell is a fuel factor denoted by

$$m_t(I, J) \text{ where } 0 \leq m_t(I, J) \leq 1.$$

This factor may be used in a fire simulation, for example, to control the life span of the fire within the cell. In addition, each cell has an associated condition called a flag that can be either set or clear, which may be used to indicate that a condition either does or did exist within the cell.

The flexibility of the BEM is improved if an operator is able to direct the model to perform a set of instructions which can test and change a cell's flag, life factor and fuel factor between epochs. Throughout this section it has been assumed that the life factor of a cell in the t+1 epoch would be $M'_{t+1}(I, J)$. However, the instructions are included to permit the operator to direct the model to select this calculated value or another specified value to become $M_{t+1}(I, J)$, the life factor of the cell in the next epoch. These instructions are explained in more detail in the next section, and examples in Section V illustrate the use of the instructions to control the flag, life factor, fuel factor. All variables were defined throughout this section in the range 0 through 1 for simplicity, however, the capability of the BEM is improved if this range is extended for some of the variables. Such extensions are indicated in the next section.

CHAPTER IV

IMPLEMENTATION

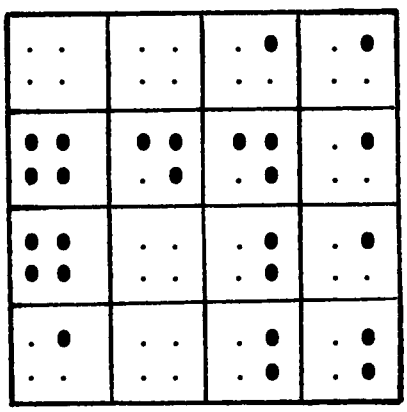
The BEM was implemented on a Digital Equipment Corporation PDP-11/20 computer with a Tennecomp TP5000 cathode ray tube graphic display device and teletype because the interrupt and priority feature of the computer, combined with the direct memory access display modes and light pen features of the display unit, permitted a relatively simple interactive implementation. (A different display unit or mode may be used if the display generation, light pen interrupt, and size routines are modified accordingly.)

As illustrated in Figure 7, the life factor of each cell is represented by four dots which are sequentially illuminated as the life factor achieves specific life factor threshold levels were .25, .50, .75, and 1.00. This display technique permits the operator to see the approximate life factor of every cell within the array. A new display is generated at the completion of each epoch and ceases at the beginning of the calculation for the next epoch.

The program starts in a dialog phase in which a BEM operator enters commands to initialize the program parameters and to start the simulation run phase. He may monitor the life of all cells during the run by viewing the display and can interrupt the run to modify any parameter or receive a list of all cell data. The following example in which

.00	.10	.25	.30
1.00	.80	.75	.35
1.15	-.12	.70	.48
.25	-.05	.60	.51

(a) Life Factors in an Example Area



(b) Generated Display for Example

Figure 7. Illustration of Life Factor Display.

commands are explained and command key words are capitalized and underscored illustrates the normal sequence of commands used by an operator to:

1. ZERO the life and fuel factors, clear every flag;
2. define the SIZE of the square array space to be considered;
3. use the light pen to indicate which cells are to have the FLAG, LIFE, and FUEL values reset. The computer will respond by typing the row, column flag, life and fuel values before resetting the instructed field;
4. define the RULE matrix;
5. set the random WALK limit to indicate the error of the rule;
6. write the CELL instructions which determine the flag, life, and fuel values for each cell in the next epoch;
7. ΦRUN the program for a specified number of epochs before returning to the dialog (where Φ is a blank);
8. interrupt the normal run by entering a ! character to return to the dialog (This character is also used to mark the end of a string of repeated input entries); and
9. DUMP the flag, life, and fuel for every cell in the considered space.

The formats for these commands are shown in Appendix A.

All dialog between the operator and the model uses integers rather than the decimal fractions as indicated in the previous Section III. For example, the WALK command uses the integer W to define the maximum error of the rule matrix where

$$E_t = 0.01 \cdot W \quad \text{with } 0 \leq W \leq 100,$$

and the FUEL command uses the integer F to define the fuel of a cell as

$$m_t(I, J) = 0.01 \cdot F \quad \text{with } 0 \leq F \leq 100.$$

Note that this range was selected for its simplicity although it could be extended.

As noted in the previous section, the value range of some variables are extended beyond the formulated range, thus increasing the BEM capabilities. This was done without increasing the program complexity. For example, the value range of a RULE element is permitted to be

$$R_t(i, j) = 0.01 \cdot R \text{ where R, is an integer } -99 \leq R \leq 199,$$

and the LIFE is permitted to be

$$M_t(I, J) = 0.01 \cdot L \text{ where L, is an integer } -12 \leq L \leq 115.$$

This permits, for example, an overpopulated cell to be indicated by $L<100$, and permits a cell which is destroying life in its neighborhood to be indicated by $L<0$.

As indicated in the previous section, the CELL command instructions provide greater flexibility for the BEM because they provide

the capability for the operator to direct the selective replacement of the flag, life, and fuel factors for each cell. The instructions are performed between epochs when the flag, life and fuel from the current epoch and the calculated life for the next epoch are available for testing. Each instruction operates on an "if-then-else" procedure -- "if" the conditions exist as indicated, "then" the specified replacements will be performed and the specified instruction will be executed next, "else" the next sequentially numbered instruction will be executed. It is possible to test "if" the flag is set or clear, and "if" the current life factor, calculated life factor, or fuel factor is between specified limits. Any or all of these items may be tested. It is possible to reset the flag, current life factor, or fuel factor. It should be noted, however, that once the data has been replaced, the old value cannot be tested later. When there are no more sequential instructions or the "then" portion indicates that execution of the instructions for that cell should terminate, the values for the flag, life, and fuel factors are set for the next epoch (if the values have not been reset, they will remain unchanged). Figure 8 illustrates the effect of a single instruction upon a single cell.

(a) Cell conditions in epoch t.

Flag . . . . . Set

Current life factor 50

Calculated life factor 30

Fuel factor 10

(b) Instruction meaning.

If: the flag is set,
there is no current life factor test,
$25 \leq$ calculated life factor ≤ 50, and
$0 \leq$ fuel factor ≤ 50,

Then: clear the flag,
set the current life factor = 0,
set the fuel factor = 100, and
terminate.

(c) Effect upon cell conditions appearing in the t+1 epoch.

Flag . . . . . Clear

Current life factor 0

Fuel factor 100

Figure 8. Example of a Single Instruction Upon a Single Cell.

CHAPTER V

SUMMARY AND CONCLUSIONS

The capabilities of the BEM are exhibited in Appendix B with two simulations. The first is a Life simulation reproducing the patterns in Figures 2 and 3, pages 6 and 8, in order to demonstrate the use of the basic commands. The second simulates a forest fire with and without an error in the expected fire spread pattern to illustrate the use of the fuel factor and WALK command.

This BEM program does not require the support of an operating system or other core resident programs, and it may be executed within 14,000 bytes. This small size causes the CRT display to flicker when the array is larger than SIZE 32, however, the modular programming of this BEM simplifies program modifications. A user may allocate more core to the display to eliminate the flicker or increase the number of CELL instructions, or he may reduce the program core requirements to execute the program in a smaller machine.

The BEM permits a user to prepare a model without writing a special purpose simulator and should encourage more simulation of environmental problems by users without extensive computer programming experience.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Academic Games Associates. Simulation and Games.

2. American Society for Cybernetics. Journal of Cybernetics.

3. Arbib, Michael A. Theories of Abstract Automata. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1969.

4. Bastin, E. W. Abstract (published in ref 22) of a paper "Concerning a Self-Organizing Machine (CASPAR) having a Hierarchial Structure of Alegbraic Levels." King's College, Cambridge, England, 1960.

5. Brown, Albert and Ronald F. Kirby. "Measuring Urban Performance." Journal of Cybernetics, 2, 4 (Oct.-Dec., 1971), pp. 32-54.

6. Cline, Randall E. A Survey and Summary of Mathematical and Simulation Models as Applied to Weapon System Evaluation. ASD Technical Report 61-276, Operations Research Department, Institute of Science and Technology, University of Michigan, Ann Arbor, Michigan, October, 1961.

7. Codd, E. F. Cellular Automata. New York: Academic Press, 1968.

8. Evans, George W., II, Graham F. Wallace, and Georgia Sutherland. Simulation Using Digital Computers. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1967.

9. Feller, William. An Introduction to Probability Theory and Its Application. New York: Wiley, 1967.

10. Frederick, Terry J. and Eldo C. Koenig. "Analysis for an Effective Operation of a General Automation: Recursive Methods Applied to a Graph Model." Journal of Cybernetics, 1, No. 1 (Jan.-Mar., 1971), pp. 53-70 and No. 2 (Apr.-June, 1971), pp. 49-66.

11. Gard, Thomas and James Curtis. "Forest Fire Simulation." Unpublished research report at The University of Tennessee, 1973.

12. Gardner, Martin. "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life'." Scientific American, 223, 4 (Oct., 1970), 120-123.

13. Gardner, Martin. "Mathematical Games: On Cellular Automata, Self-Reproduction, the Garden of Eden and the Game of 'Life.'" Scientific American, 224, 2 (Feb. 1971), 112-117.

14. Gardner, Martin. "Mathematical Games: How to Triumph at Nimly playing Life" and John Houston Conway's game "Hackenbush." Scientific American, 226, 1 (Jan., 1972), 107.

15. Ledley, Robert S. Recursive Functions and Switching-Circuit Theory. Ed. Julius T. Tou. New York: Academic Press, 1968.

16. Levitt, K. N., and W. H. Kautz. "Cellular Arrays for the Solution of Graph Problems." Communications of the Association of Computing Machinery, 15, No. 19 (September, 1972), 789-801.

17. Nagano, Takashi. "A New Model of Learning." Journal of Cybernetics, 1, No. 4 (Oct.-Dec., 1971), 77-90.

18. Simulation Councils Inc., Simulation.

19. Tennessee Systems Inc., TP5000 Systems Manual. Oak Ridge, Tennessee: 1972.

20. Tou, Julius T. "Stochastic Automata and Discrete Systems Theory." Applied Automata Theory. Ed. Julius T. Tou. New York: Academic Press, 1968.

21. Turing, A. M. "On Computable Numbers with an Application to the Entscheidungs-Problem." Proceedings of London Mathematical Society Series 2-42 (1936-7), 230-265.

22. Turing, A. M. "On Computable Numbers with an Application to the Entscheidungs-Problem." Proceedings of London Mathematical Society Series 2-43 (1936-7), 544-546.

23. von Neumann, John. Theory of Self-Reproducing Automata. Ed. Arthur W. Burns. University of Illinois Press, Urbana, Illinois, 1966.

24. Yovits, Marshall C. and Scott Cameron. Self-Organizing Systems. New York: Pergamon Press, 1960.

APPENDICES

APPENDIX A

COMMAND FORMATS AND DATA STRUCTURES

This appendix illustrates the correct formats of the BEM commands for operating usage and indicates the data storage structures for use by persons desiring to extend or modify the program.

Figures 9 and 10 describe the commands using the following special characters and symbols:

{ }	braces - a user choice of input;
( )	parentheses - a comment, not input;
cr	carriage return character;
d	a delimiter such as a comma, equals sign, carriage return, or semicolon;
b	blank character, treated as a zero if found in a number; and
xxx	a three-digit number.

The computer generated question mark (?) indicates that a command key word is expected. Incorrect field entries that are noted by the computer require resubmission. The rubout character may be used to cancel any character except a delimiter, which initiates command analysis of all entries proceeding from right to left (therefore, excess characters to the left will be disregarded).

```
!           (Cease simulation run and start command dialog.  Computer)
            (responds with "STOP" and follows with "?" to indicate   )
            (that a command key word is desired.                     )

ZEROd

SIZEdxxxd

FLAGd      { S  (Set)   }  d        (User marks cells with light pen.)
           { C  (Clear) }

    (Computer responds with "@xxx,xxx = C xxx,xxxcr" each mark.  )
    (for example:                                                )
    (@004,001 = S100,050 means before the change, the cell (4,1))
    (had the flag set, life = 100, and fuel = 50.                )
    (Flag dialog continues until halted by a "!".                )

    d { S } d   (Computer)   d { S } d   (...)  !
      { C }     (response)     { C }

LIFEdxxxd          (Computer responds as above) dxxxd (...) !

FUELdxxxd          (Computer responds as above) dxxxd (...) !

RULEdxxxdxxxdxxxdxxxdxxxd          (row  2)
     xxxdxxxdxxxdxxxdxxxd          (row  1)
     xxxdxxxdxxxdxxxdxxxd          (row  0)
     xxxdxxxdxxxdxxxdxxxd          (row -1)
     xxxdxxxdxxxdxxxdxxxd          (row -2)

WALK { dSxxxd } { E }
     {   d    } { M }   xxxd    ('S' sets the seed for the random number    )
               { R }             (      generates                           )
               { b }             (The random number will be changed         )
                                 (each epoch if 'E' is entered, each cell in )
                                 (the main array if 'M' is entered every Rule)
                                 (calculation if 'R' blank                  )
                                 (or no entry.                              )

bRUN       { cr   }  (Assume run one epoch only)
           { dxxxd }  (Run xxx number of epochs )

DUMPd                (Computer responds with data for all cells. )
```

Figure 9. Miscellaneous Dialog Command Formats.

(a) Sample Instructions.

```
CELL,
Abb1,N,bb0,100,N,b99,101=S,C,R100,Gb20;
Dbb5;
Cbb2,S,bb9,bb9,N,N=C,R100,0000,T;
E;
```

(b) Form and Options of the Instruction.

key word — CELLd

instruction number — { Axxxd (Add); Cxxxd (Change); Dxxxd (Delete) (If delete, stop here.); E (End of instructions) }

test flag — { C (Clear); S (Set); N (No test) } d

test current life factor — { N (No test); xxxdxxx (test if between these two inclusive values) } d

test calculated life factor — { N; xxxdxxx (as above) } d

test fuel — { N; xxxdxxx (as above) } d

reset flag to — { C (Clear); S (Set); N (No change) } d

reset current life factor to — { C (Calculated value); N (No change); Rxxx (Reset to xxx) } d

reset fuel to — { Cxxx (Consume xxx units); N (No change); Pxxx (Percentage xxx of old fuel left); Rxxx (Reset to xxx units) } d

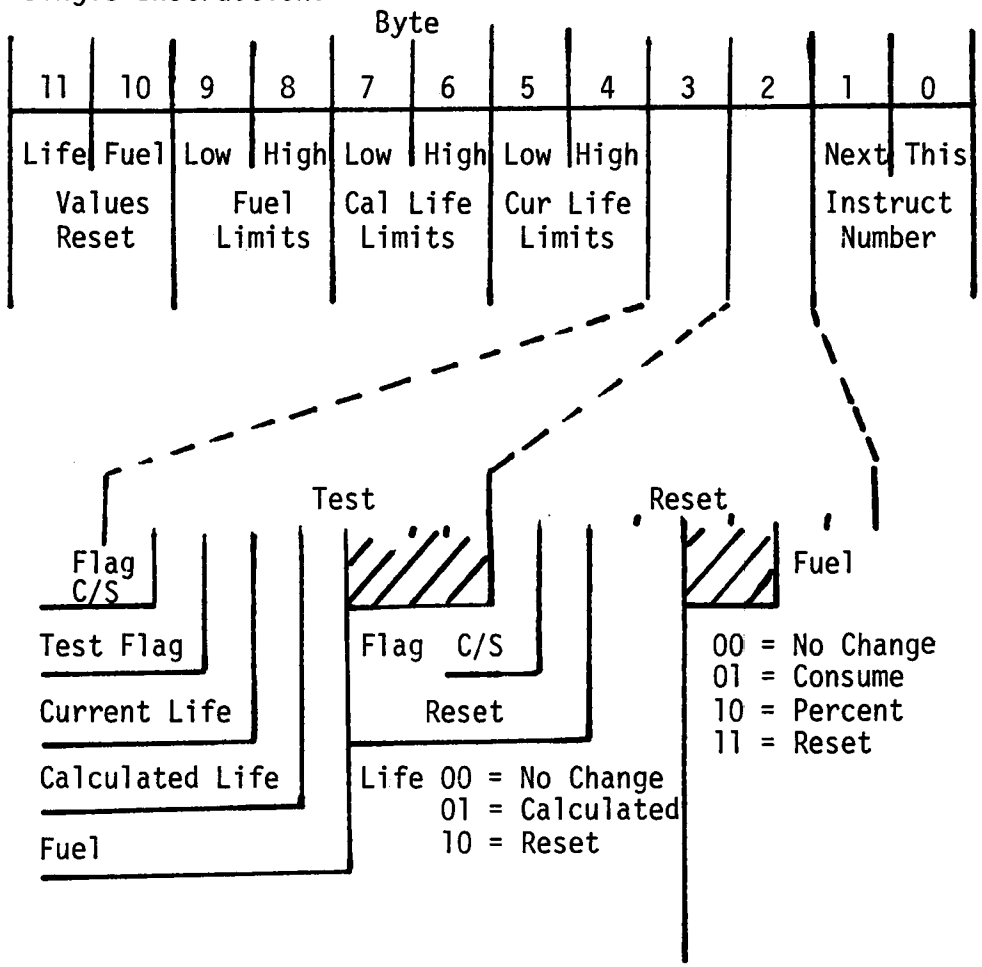
then go to — { Gxxx (Go to xxx); N (Next instruction); T (Terminate this cell) } d

Figure 10. Dialog Format of the CELL Instruction Command.

Data structures are designed to use the PDP-11 capability to address either an eight-bit byte or a sixteen-bit word. To eliminate the inherent error present in the binary representation of a decimal fraction, data fields are normally the integers as indicated in the Implementation Section. However, the life L of a cell is represented internally by the integer L' where L'=L+12; this eliminates the need for a sign bit. The bit codes used in other fields are explained on each figure.

The direct memory access display mode 1 is a scan which sweeps across the display from left to right after displaying each vertical display column. Because the data sequence was dictated by the choice of the mode, all data is stored in a sequence that simplifies the preparation for that display. (Note that in Figures 11 and 12 that the cells are stored in the sequence indicated, but the locations are only representative.)

(a) Single Instruction.

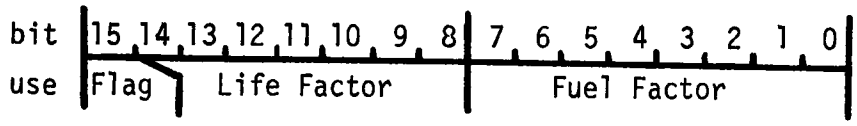


(b) Sequence of Instructions in Core.

Location	196	184	172	160	148	136	124	112	100
Instruction Number	0	0	190	20	10	6	5	4	1

Figure 11. CELL Instruction Structure.

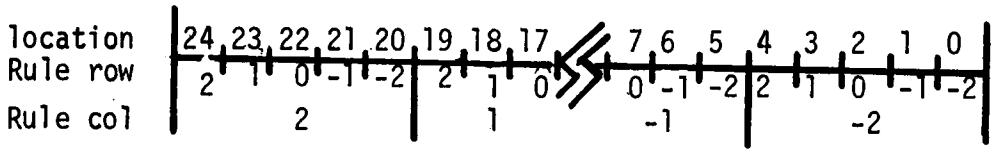
(a) A cell in the M matrix.



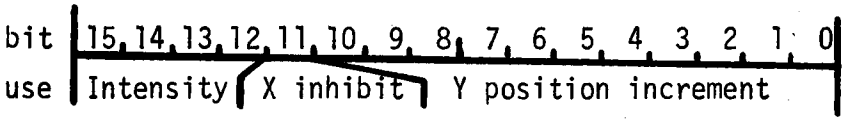
(b) An example of the storage sequence of cells using SIZE=3.

location	16	14	12	10	8	6	4	2	0
cell	(3,1)	(3,2)	(3,3)	(2,1)	(2,2)	(2,1)	(1,1)	(1,2)	(1,3)

(c) The storage of the rule matrix where each element is a byte.



(d) Word required for the display unit to display one point.



(e) An example of a section of the display work area for SIZE=3.

location	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
cell row	2	2	2	2	s	1	1	1	1	1	1	1	1	1	1	1	1
cell col	2	2	3	3	k	1	1	2	2	3	3	1	1	2	2	3	3
cell dot	3	4	3	4	i p	1	2	1	2	1	2	3	4	3	4	3	4

Figure 12. Data Structures of Data Items.

APPENDIX B

BEM EXAMPLES

This appendix contains two example BEM simulations -- Life and a forest fire.

```
?THIS APPENDIX DEMONSTRATES THE BEM/
 (A SLASH IS USED TO END A COMMENT LINE)/
/
/
THE FIRST EXAMPLE SIMULATES THE GAME OF LIFE/
USING AN ARRAY OF       SIZE=5;
?CLEAR ALL CELLS        ZERO,
?BECAUSE THERE IS NO RANDOM NATURE    WALK=0,
?NEXT DEFINE THE EFFECT OF A CELL UPON ITS NEIGHBORS/
 RULE=000,000,000,000,000,
      000,033,033,033,000,
      000,033,000,033,000,
      000,033,033,033,000,
      000,000,000,000,000,
?NEXT CREATE THE INSTRUCTIONS WHICH DEFINE /
CONDITIONS IN THE NEXT EPOCH/
CELL=
ADD INSTRUCTION              A001
TO DEFINE WHEN A CEEL SHOULD BE BORN/
A DEAD CELL (FLAG CLEAR) ADJACENT TO /
EXACTLY 3 LIFE NEIGHBORS WILL BE BORN/
IN THE NEXT EPOCH/
IF/
THE CELL IS DEAD (FLAG CLEAR)      C
DO NOT TEST CURRENT LIFE           N
AND CALCULATED LIFE EXACTLY   99,99
DO NOT TEST THE FUEL               N
THEN   RESET/
CELL ALIVE (FLAG SET)              S
RESET LIFE                      R100
NO CHANGE TO FUEL                  N
TERMINATE INSTRUCTION EXECUTION    T

/
/
SURVIVALS ARE DEFINED BY/
A002;S;N;66,99;N=S;R100;N;T;
DEATHS DEFINED BY/
A003;N;N;N;N=C;R000;N;T;
BECAUSE THAT IS ONLY THING LEFT/
END INSTRUCTIONS                   E;
?/
```

```
THE DEMONSTATION WILL CONSIST OF PATTERNS/
SEEN      IN FIGURES 1 AND 2/
/
          STABILITY/
INITIALIZE THE ARRAY USING ONLY THE KEYBOARD/
DATA,S100,000;2,3;3,3;3,4;;
?WHICH RESULT  IN THE ARRAY
GOOF
?(THIS INDICATES OPERATOR ERROR)/
DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=003 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN FOR ONE EPOCH        RUN=1;STOP EPOCH=001
?AND SHOW RESULTS       DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=003 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

THE PATTERN TRANSITION CAN BE SEEN/
ONCE MORE      RUN
STOP EPOCH=002
?D\MP,GOOF
?DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=003 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN,1,STOP EPOCH=003
?DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=003 C000,000 C000,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?
```

```
          OSCILLATION/
CLEAR THE ARRAY    ZERO,
?INSERT CORRECT DATA=S100,000=2,3;3,3;4,3;;
?WHCH IS    DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=003 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=004 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=001
?DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN
STOP EPOCH=002
?DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=003 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=004 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=2;STOP EPOCH=005
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?
```

```
        UNDERPOPULATION/
ZERO,
?DATA=$100,000,2,213,214,31;
?WHICH IS    DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=001
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=002
?DUMP,
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?
```

```
      OVER POPULATION/
ZERO,
?DATA=S100,000=2,2;3,2;3,3;4,2;4,3;;
?WHICH IS DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 C000,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 C000,000 C000,000
ROW=004 C000,000 S100,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN
STOP EPOCH=001
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 S100,000 C000,000 C000,000
ROW=003 S100,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 S100,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN
STOP EPOCH=002
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 C000,000 C000,000 C000,000
ROW=003 S100,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 S100,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 S100,000 S100,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=004
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?
```

```
THE PATTERN SHOWN IN FIGURE 1/
ZERO,
?DATA=$100,000=2,3;3,2;3,3;3,4;
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=003 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=001
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=003 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=004 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=002
?DUMP;
ROW=001 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=002 C000,000 S100,000 C000,000 S100,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?RUN=1;STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=002 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=003 C000,000 S100,000 C000,000 S100,000 C000,000
ROW=004 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=005 C000,000 C000,000 S100,000 C000,000 C000,000

?RUN=1;STOP EPOCH=004
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=003 C000,000 S100,000 C000,000 S100,000 C000,000
ROW=004 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=005 C000,000 C000,000 C000,000 C000,000 C000,000

?
```

```
RUN
STOP EPOCH=005
?DUMP;
ROW=001 C000,000 C000,000 S100,000 C000,000 C000,000
ROW=002 C000,000 S100,000 C000,000 S100,000 C000,000
ROW=003 S100,000 C000,000 C000,000 C000,000 S100,000
ROW=004 C000,000 S100,000 C000,000 S100,000 C000,000
ROW=005 C000,000 C000,000 S100,000 C000,000 C000,000

?NOW JUMPING AHEAD   RUN=5;STOP EPOCH=010
?DUMP;
ROW=001 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=002 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=003 S100,000 S100,000 C000,000 S100,000 S100,000
ROW=004 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=005 C000,000 S100,000 S100,000 S100,000 C000,000

RUN=10;STOP EPOCH=020
?DUMP;
ROW=001 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=002 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=003 S100,000 S100,000 C000,000 S100,000 S100,000
ROW=004 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=005 C000,000 S100,000 S100,000 S100,000 C000,000

RN     RUN
STOP EPOCH=021
?DUMP;
ROW=001 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=002 S100,000 C000,000 C000,000 C000,000 S100,000
ROW=003 S100,000 C000,000 C000,000 C000,000 S100,000
ROW=004 S100,000 C000,000 C000,000 C000,000 S100,000
ROW=005 C000,000 S100,000 S100,000 S100,000 C000,000

?RUN
STOP EPOCH=022
?DUMP;
ROW=001 C000,000 S100,000 S100,000 S100,000 C000,000
ROW=002 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=003 S100,000 S100,000 C000,000 S100,000 S100,000
ROW=004 S100,000 C000,000 S100,000 C000,000 S100,000
ROW=005 C000,000 S100,000 S100,000 S100,000 C000,000

?ZERO;
 ?
```

```
      FOREST FIRE SIMULATION/
ASSUME THAT THE FOREST LOOKS LIKE/
  W  H  M  M  L      W IS WATER/
  H  M  M  M  M      B IS BARREN/
  L  H  W  H  W       L IS LIGHT VEGETATION/
  B  M  H  W  H      M IS MEDIUM GROWTH/
  M  H  W  L  M      H IS HEAVY GROWTH/
WHICH IS INSERTED INTO THE ARRAY BY/
LIGHT DATA=C000,050=1,5; 3,1; 5,4;;
?MED DATA,C000,075=1,3;1,4;2,2;2,3;2,4;2,5;4,2
     5,1;5,5;;
?AND HEAVY GROWTH BY DATA=C000,100=1,2
     2,1;3,2;3,4;4,3;4,5;5,2;;
?ASSUME AS A FIRE SPREAD PATTERN THAT A WIND/
IS BLOWING FORM THE SOUTH-WEST/
REPLACE THE OLD RULE BY ENTERING A NEW ONE/
 RULE=000,000,000,010020\\\,020,
      000,030,050,075,030,
      000,020,100,060,000,
      000,010,020,030,000,
      000,000,000,000,000,
?DEFINE THE INSTRUCTIONS NEXT  BY   CELL;
FIRST DELETE THE OLD SET/
D001,
D002,
D003,
TO LET A CELL CATCH ON FIRE          A001,/
IF NOT ALREADY BURNING (FLAG CLEAR)    C
NO OLD FIRE SO NO TEST                   N
THE CALCULATED FIRE              060,115
THE CELL MUST HAVE FUEL          001,100
THEN START BURNING                     S
WITH CALCULATED INTENSITY              C
FUEL CONSUMED NEXT EPOCH               N
END OF INSTRUCTION FOR THIS CELL        T;
```

```
TO STOP A CELL THAT IS BURNINGBECAUSE OF NO FUEL  /
A010,S/
ON FIRE                                          S
INTENSITY DOES NOT MATTER                        N
NOR DOES THE CALCULATED INTENSITY                N
IS THERE FUEL AVAILABLE                  000,009
(INSUFFICIENT FUEL TO CONTINUE)/
STOP BURNING                                     C
RESET FIRE INTENSITY TO ZERO                R000
LEAVE FUEL ALONE                                 N
AND TERMINATE                                    T,
/
CONTINUE BURNING UNDER THE FOLLOWING CONDITIONS/
A020S
GOOF
A020,S
BURNING STRONGLY          075,115
DOES NOT MATTER                  N
NO FUEL TEST                      N
CONTINUE BURNING                 N
BUT AT THE CALCULATED RATE        C
CONSUMINGFUEL               C020
AND THEN GO TO              G050

/
BURNING MODERATELY/
A030;S;  050,074; N; N=N;C;C010;G050;
/
AND RESET THE FUEL  IN INSTRUCTION 050/
A050;N;N;N;-12,GOOF 000,000,C;R000,R000,T;
BECAUSE WITHOUT FUEL THE FIRE MUST STOP/
/
AND END THE INSTRUCTIONS      E;
?/
STARTING A FIRE IN ONE CELL/
DATA=S100,075;4,2;;
?AS SHOWN BY DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S100,075 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075
```

```
?     RUN=1,STOP EPOCH=001
?DUMP,
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S100,055 S060,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

TWO CELLS ARE NOW BURNNING/
RUN=1;STOP EPOCH=002
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 S068,100 C000,000 S075,100 C000,000
ROW=004 C000,000 S112,035 S115,090 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

THE FIRE IS SPREADING AND FUEL IS BEING CONSUMED/
RUN
STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 S084,075 S091,075 S079,075
ROW=003 C000,050 S115,090 C000,000 S115,080 C000,000
ROW=004 C000,000 S115,015 S115,070 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

RUN
STOP EPOCH=004
?DUMP;
ROW=001 C000,000 C000,100 S080,075 S115,075 S115,050
ROW=002 C000,100 S074,075 S115,055 S115,055 S115,055
ROW=003 C000,050 S115,070 C000,000 S115,060 C000,000
ROW=004 C000,000 C000,000 S115,050 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

THE ORIGINAL CELL HAS BURNED OUT/
RUN
STOP EPOCH=005
?DUMP;
ROW=001 C000,000 S087,100 S115,055 S115,055 S115,030
ROW=002 C000,100 S115,065 S115,035 S115,035 S115,035
ROW=003 C000,050 S115,050 C000,000 S115,040 C000,000
ROW=004 C000,000 C000,000 S115,030 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075
```

```
RUN
STOP EPOCH=006
?DUMP;
ROW=001 C000,000 S115,080 S115,035 S115,035 S115,010
ROW=002 S066,100 S115,045 S115,015 S115,015 S115,015
ROW=003 C000,050 S115,030 C000,000 S115,020 C000,000
ROW=004 C000,000 C000,000 S115,010 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=007
?DUMP;
ROW=001 C000,000 S115,060 S115,015 S115,015 C000,000
ROW=002 S115,090 S115,025 C000,000 C000,000 C000,000
ROW=003 C000,050 S115,010 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=008
?DUMP;
ROW=001 C000,000 S115,040 C000,000 C000,000 C000,000
ROW=002 S115,070 S115,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=009
?DUMP;
ROW=001 C000,000 S115,020 C000,000 C000,000 C000,000
ROW=002 S115,050 C000,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=010
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 S115,030 S092,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075
```

```
IT CAN BE SEEN THAT SOME AREAS ARE STILL BURNING/
BUT SOME HAVE NOT BEEN TOUCHED BY THE FIRE/
RUN
STOP EPOCH=011
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 $115,010 C000,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=012
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 $069,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
STOP EPOCH=013
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

    THE FIRE HAS BURNED OUT  NOW/
AND LOOKING AT IT LATER    RUN=10
STOP EPOCH=023
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,005 C000,000 C000,000 C000,000
ROW=003 C000,050 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 C000,000 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

THE SITUATUION REMAINS CONSTANT/
/
/
```

```
NOW ASSUME THAT THE FIRE SPREAD PATTERN WAS/
IN ERROROR THAT A GUSTY WIND WAS BLOWING/
FIRST RETURN THE AREA TO THE INITAL CONDITIONS/
ZERO,
?DATA=C000,050=    1,5;    3,1;    5,4;;
?DATA=C000,075=    1,3;    1,4;    2,2;    2,3
      2,4;   2,5;    4,2;    5,1;    5,5;;
?DATA,C000,0\100,    1,2;    2,1;    3,2;    3,4
      4,3;    4,5;    5,2;;
?INSERT THE FIRE DATA=S100,075,4,2,,
?ALL THE RULES REMAIN THE SAME /
AND THE CELL INSTRUCTIONS REMAIN THE SAME/
BUT NOW THE RANDOM ERROR AFFECTS THE RULE/
WALK=    SET THE SEED (IF DESIRED)  S011,/
        NOW SET THE MAXIMUM EXPECTED ERROR/
        (CHANGED ONCE EACH EPOCH)  E050

?RUN
STOP EPOCH=001
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S000,055 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

THE FIRE HAS NOT SPREAD THIS TH\IME/
ONCE MORE    RUN
STOP EPOCH=002
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S096,035 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

STILL NOT SPREAD     RUN
STOP EPOCH=003
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S077,015 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075
```

```
AGAIN IT HAS NOT SPREAD BUT THE FUEL IS BURNING/
RUN
STOP EPOCH=004
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 C000,000 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

THE FIRE HAS BURNED OUT WITHOUT SPREADING TO /
NEIGHBORING CELLS/
/
/
REPEATING THE SIMULATION WITH A DIFFERENT ERROR/
DATA=S100,075,4,2,,
?WALK,S111,GOOF S011,/
   THIS TIME WITH A 100% ERROR POSSIBLE FOR /
    EACH RULE CALCULATION     R100;
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 C000,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S100,075 C000,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
RANDOM FIXSTOP EPOCH=005
?THE ABOVE ERROR MESSAGE MEANS THAT THE RANDOM/
NUMBER GENERATOR WAS UNABLE TO GENERATE A /
VALID NUMBER WT\ITHIN 10 TRIES...IT FIXED/
  ITSELF AND CONTINUED/
BECAUSE THE ARRAY WAS NOT ZEROED THE EPOCH/
COUNT INDICATE THAT THIS IS EPOCH NUMBER 5/
BUT ZEROING THE  ARRAY WOULD HAVE REQUIRED /
RE-ENTERING ALL DATA AGAIN/
     DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 C000,075 C000,075 C000,075
ROW=003 C000,050 S088,100 C000,000 C000,100 C000,000
ROW=004 C000,000 S115,055 S100,100 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075
```

```
THE  FIRE IS NOW SPREADING TO ADJACENT AREAS/
RUN
RANDOM FIXSTOP EPOCH=006
?DUMP;
ROW=001 C000,000 C000,100 C000,075 C000,075 C000,050
ROW=002 C000,100 C000,075 S115,075 C000,075 C000,075
ROW=003 S056,050 S115,080 C000,000 C000,100 C000,000
ROW=004 C000,000 S-12,035 S115,080 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

IT IS NOW APPARENT THAT THE CELL INSTRUCTIONS/
WERE INCOMPLETE BECAUSE THE FIRE INTENSITY/
IN ONE CELL IS AT A NEGATIVE LEVEL BUT IT /
CEK\LL INSTRUCTIONS WILL NOT BE CHANGED FOR /
THIS EXAMPLE/
HOWEVER THE ERROR WILL CHANGE/
WALK=(NOT ENTERING SEED) CHANGE EACH MAIN CELL/
     M050,
?RUN;1;STOP EPOCH=007
?DUMP;
ROW=001 C000,000 C000,100 S115,075 S115,075 C000,050
ROW=002 S112,100 S115,075 S115,055 S115,075 C000,075
ROW=003 S115,030 S115,060 C000,000 S109,100 C000,000
ROW=004 C000,000 S-12,035 S115,060 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

     WALK=M010,
?RN\UN
RANDOM FIXSTOP EPOCH=008
?DUMP;
ROW=001 C000,000 S115,100 S115,055 S115,055 S115,050
ROW=002 S115,080 S115,055 S115,035 S115,055 S115,075
ROW=003 S115,010 S115,040 C000,000 S115,080 C000,000
ROW=004 C000,000 S-12,035 S115,040 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

RUN=1;STOP EPOCH=009
?DUMP;
ROW=001 C000,000 S115,080 S115,035 S115,035 S115,030
ROW=002 S115,060 S115,035 S115,015 S115,035 S115,055
ROW=003 C000,000 S115,020 C000,000 S115,060 C000,000
ROW=004 C000,000 S-12,035 S115,020 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

 ?
```

```
RUN
RANDOM FIXRANDOM FIXSTOP EPOCH=010
?DUMP;
ROW=001 C000,000 S115,060 S115,015 S115,015 S115,010
ROW=002 S115,040 S115,015 C000,000 S115,015 S115,035
ROW=003 C000,000 C000,000 C000,000 S115,040 C000,000
ROW=004 C000,000 S-12,035 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

?RUN
RANDOM FIXSTOP EPOCH=011
?DUMP;
ROW=001 C000,000 S115,040 C000,000 C000,000 C000,000
ROW=002 S115,020 C000,000 C000,000 C000,000 S115,015
ROW=003 C000,000 C000,000 C000,000 S115,020 C000,000
ROW=004 C000,000 S-12,035 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

RUN
RANDOM FIXSTOP EPOCH=012
?DUMP;
ROW=001 C000,000 S115,020 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 S-12,035 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

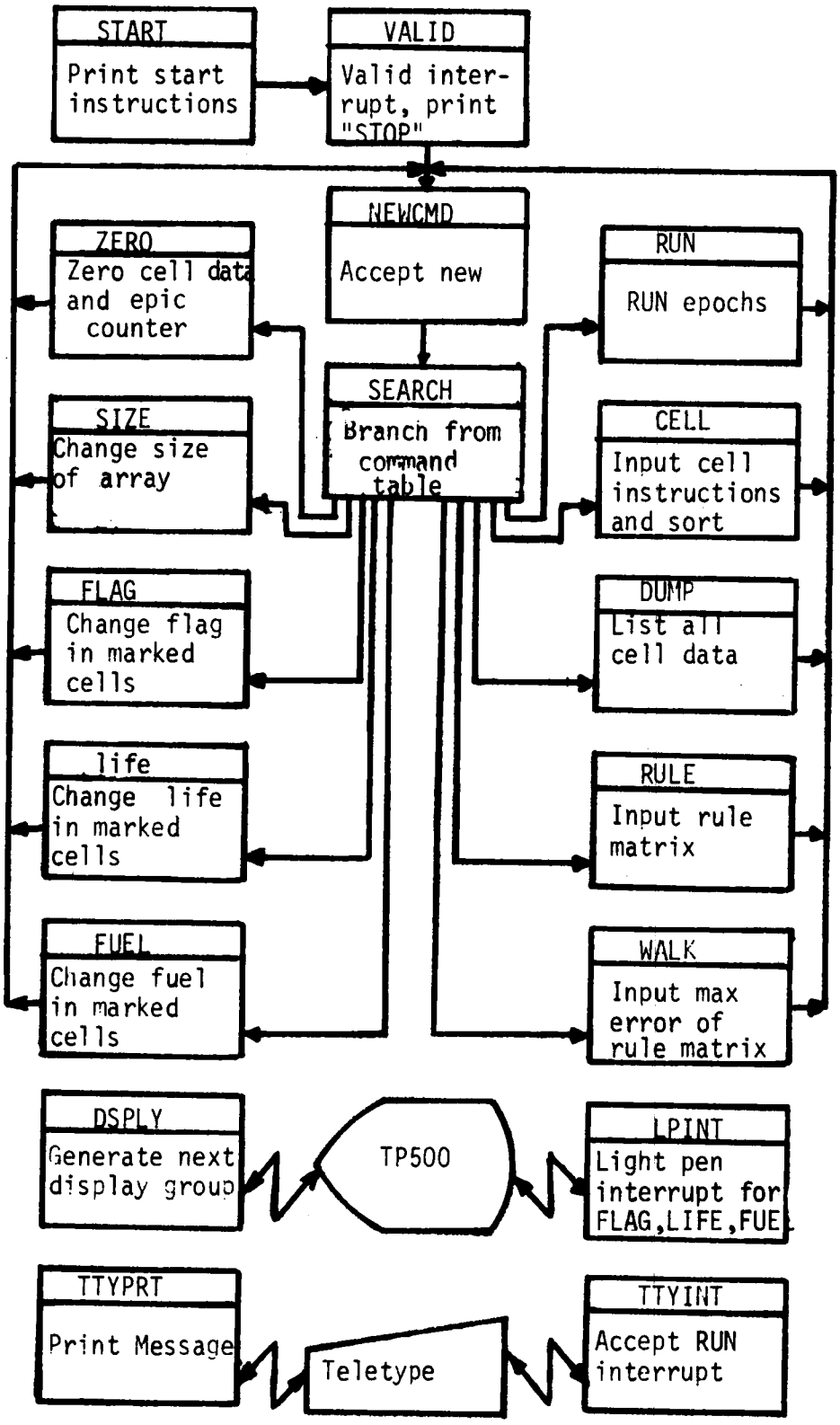
RUN;1;STOP EPOCH=013
?DUMP;
ROW=001 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=002 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=003 C000,000 C000,000 C000,000 C000,000 C000,000
ROW=004 C000,000 S-12,035 C000,000 C000,000 C000,100
ROW=005 C000,075 C000,100 C000,000 C000,050 C000,075

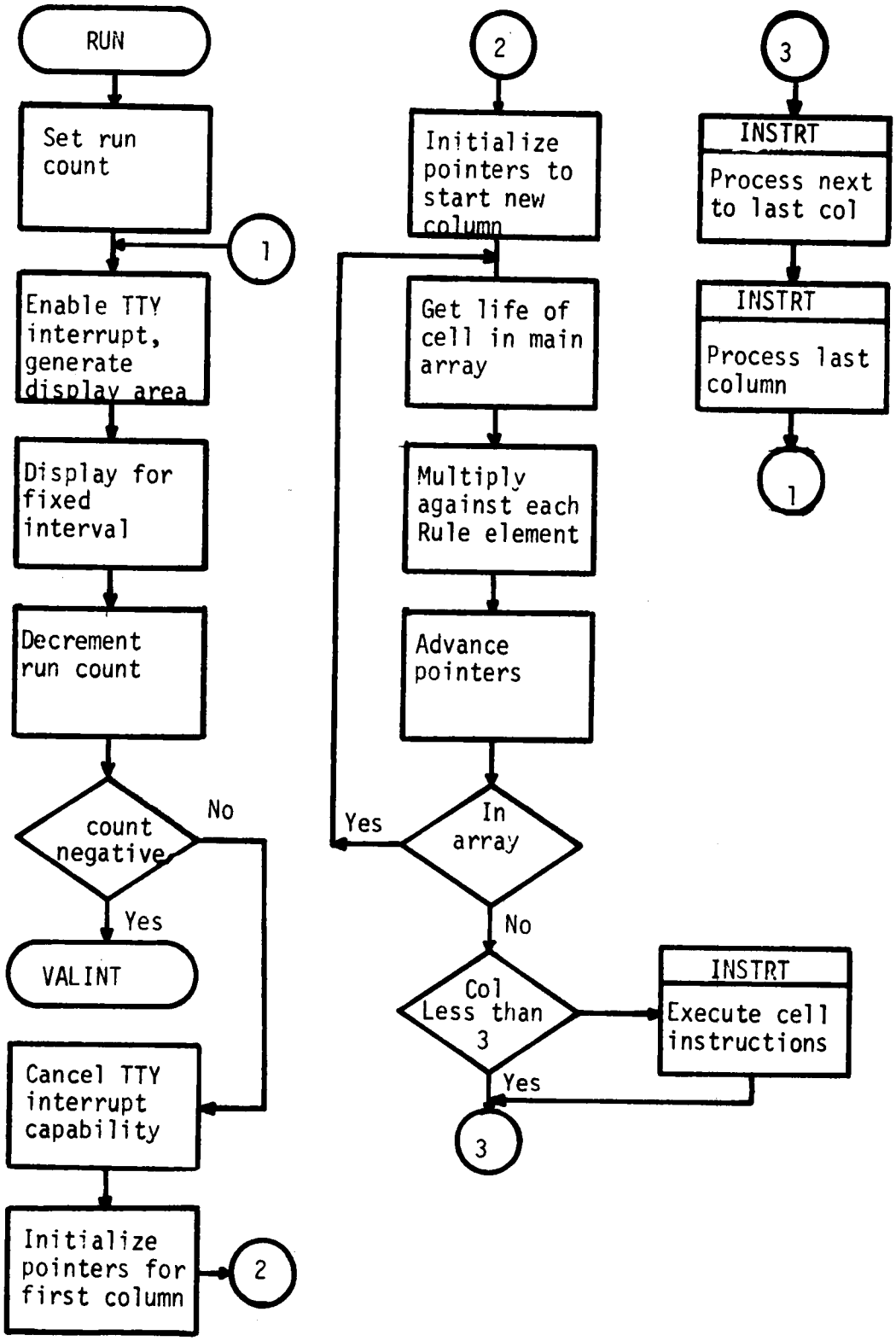
THIS COMPLETES ONLY TWO SIMULATIONS USING  THE BEM/
```

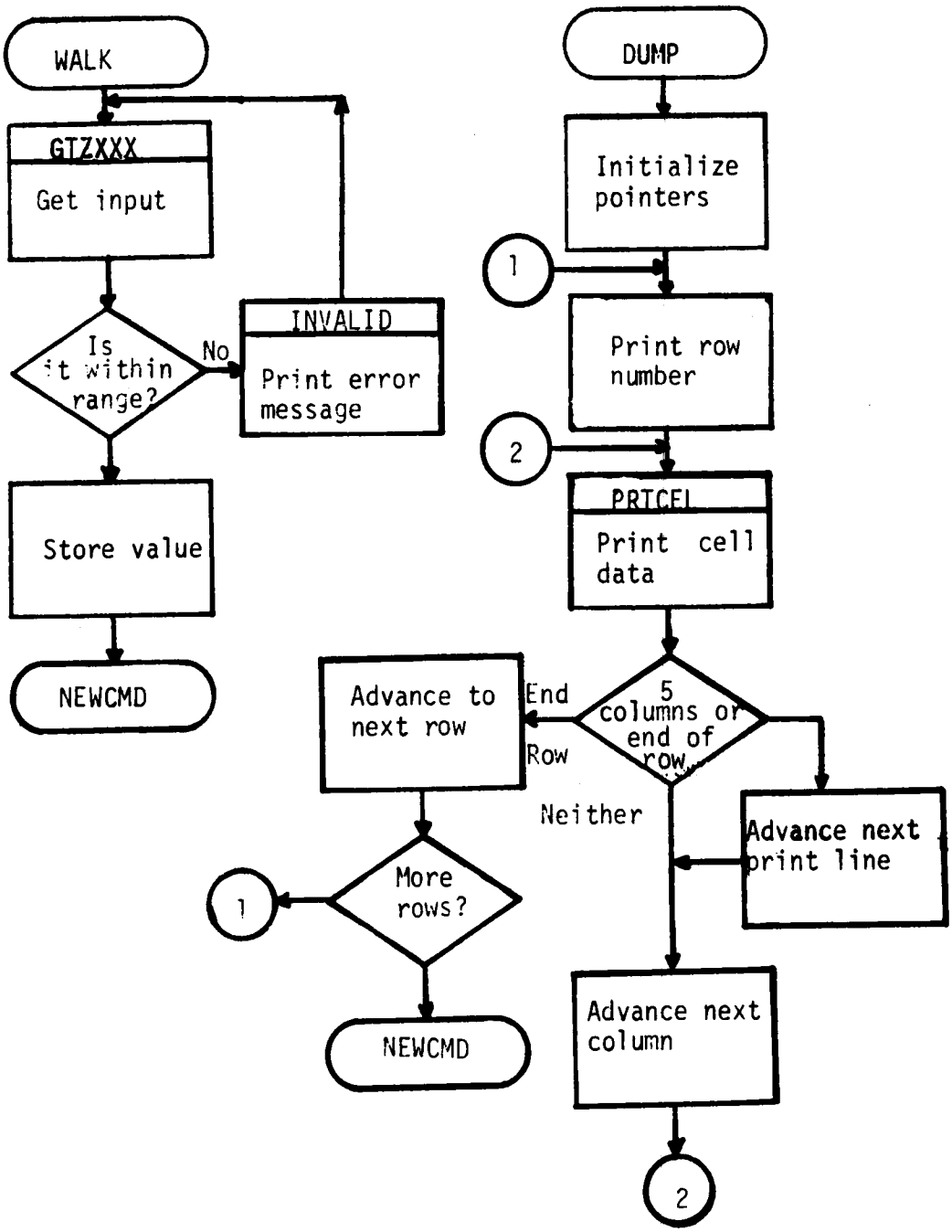
APPENDIX C

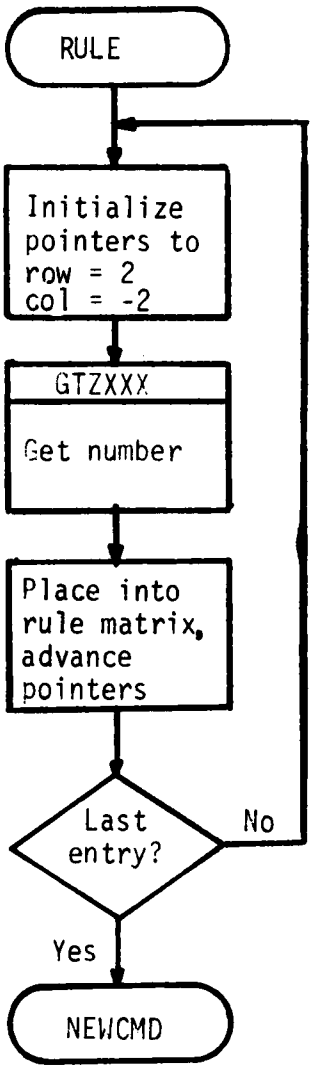
FLOWCHARTS AND PROGRAM

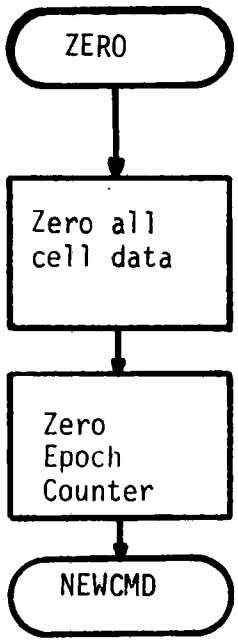
This appendix contains the program flowcharts and actual program text.

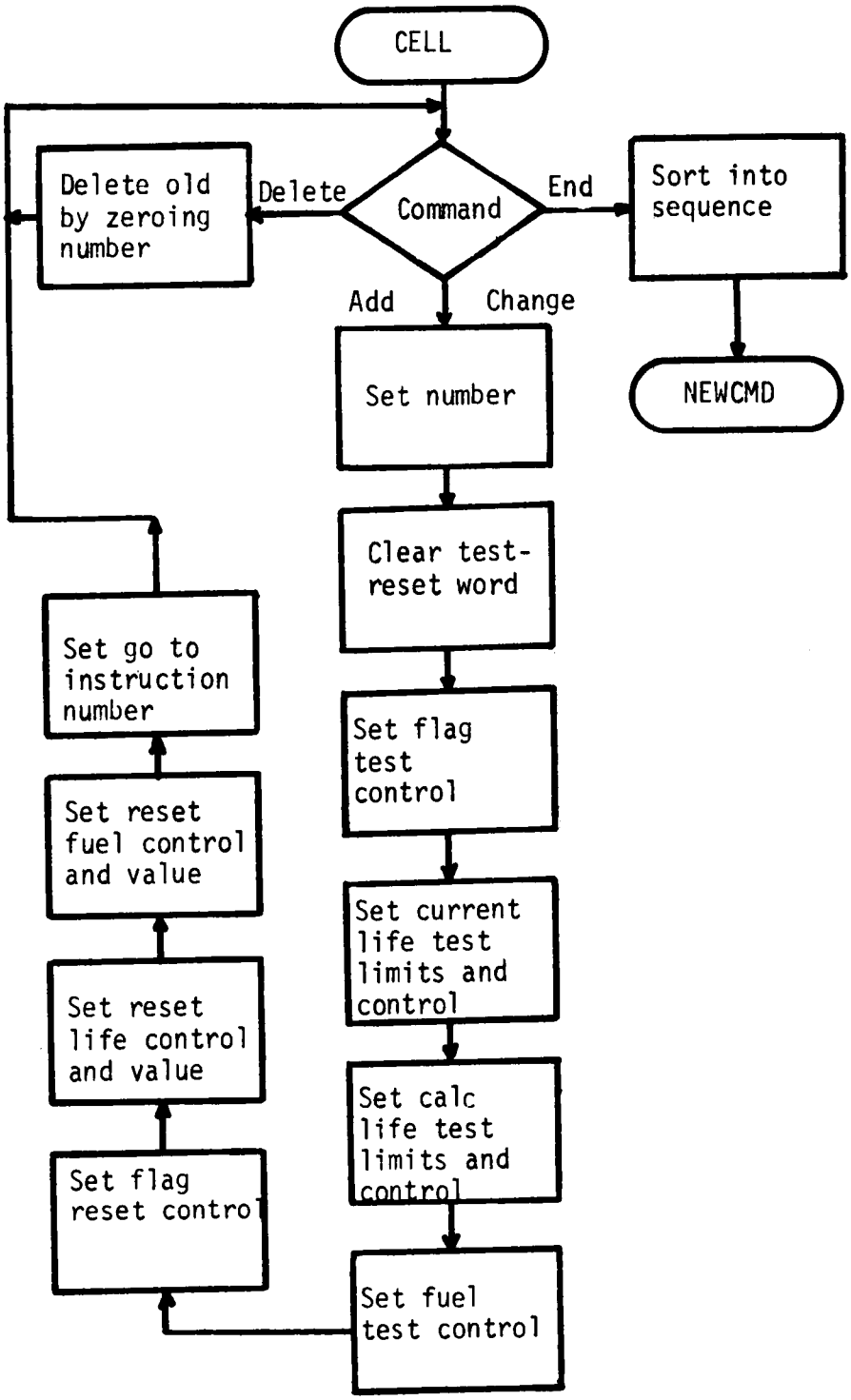

START
Print start instructions
VALID
Valid interrupt, print "STOP"
NEWCMD
Accept new
ZERO
Zero cell data and epic counter
SEARCH
Branch from command table
RUN
RUN epochs
SIZE
Change size of array
CELL
Input cell instructions and sort
FLAG
Change flag in marked cells
DUMP
List all cell data
life
Change life in marked cells
RULE
Input rule matrix
FUEL
Change fuel in marked cells
WALK
Input max error of rule matrix
DSPLY
Generate next display group
TP500
LPINT
Light pen interrupt for FLAG,LIFE,FUE
TTYPRT
Print Message
Teletype
TTYINT
Accept RUN interrupt

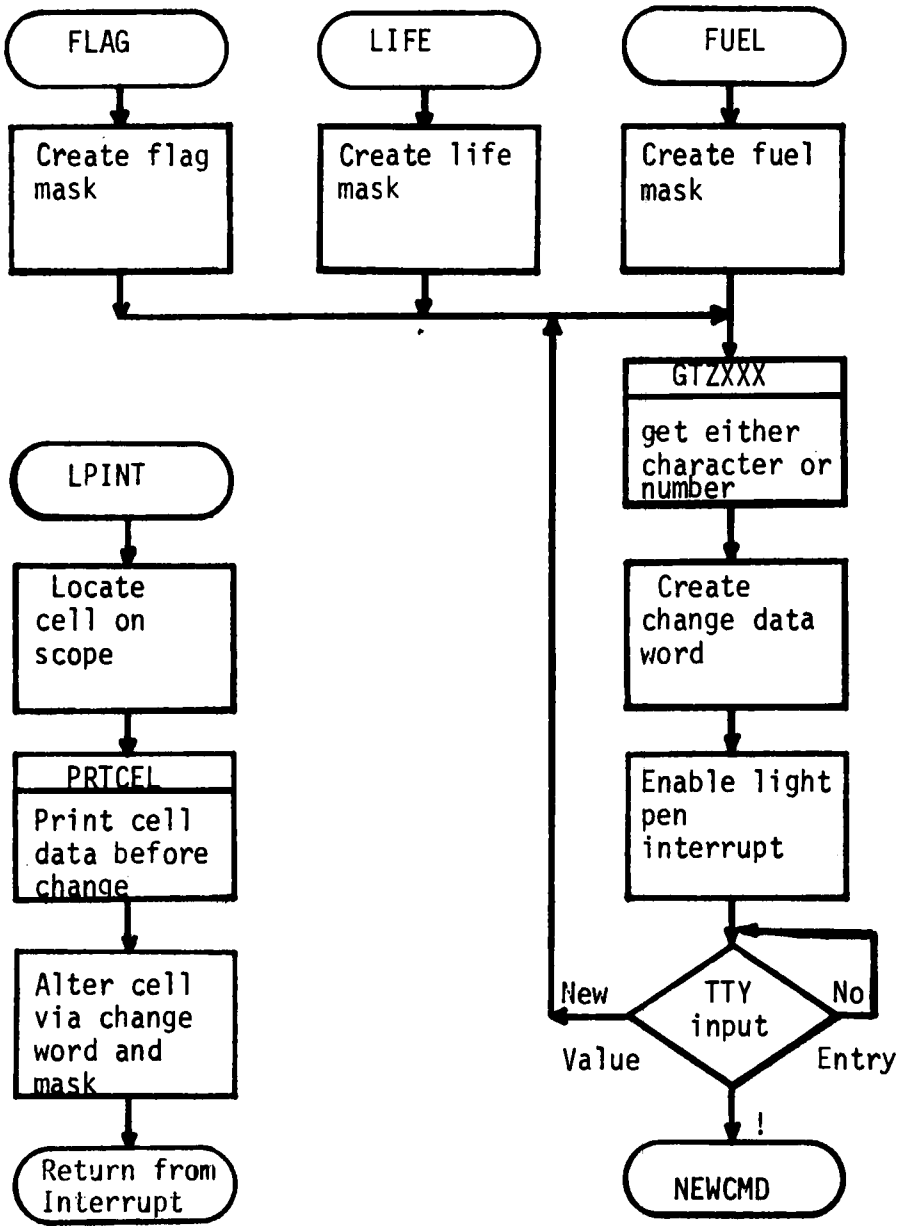

RUN
Set run count
1
Enable TTY interrupt, generate display area
Display for fixed interval
Decrement run count
count negative
No
Yes
VALINT
Cancel TTY interrupt capability
Initialize pointers for first column
2
2
Initialize pointers to start new column
Get life of cell in main array
Multiply against each Rule element
Advance pointers
In array
Yes
No
Col Less than 3
INSTRT
Execute cell instructions
Yes
3
3
INSTRT
Process next to last col
INSTRT
Process last column
1

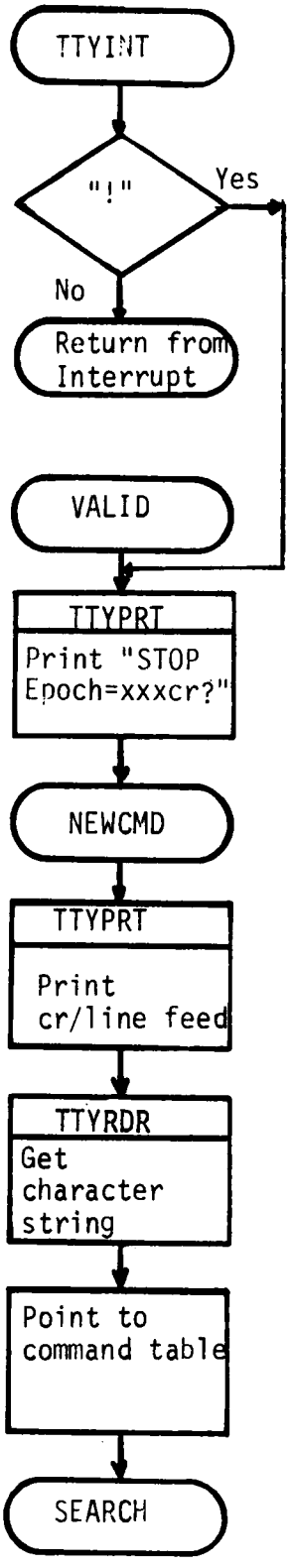

WALK
GTZXXX
Get input
Is it within range?
No
INVALID
Print error message
Store value
NEWCMD
DUMP
Initialize pointers
1
Print row number
2
PRTCEL
Print cell data
5 columns or end of row
End Row
Neither
Advance to next row
Advance next print line
More rows?
1
NEWCMD
Advance next column
2

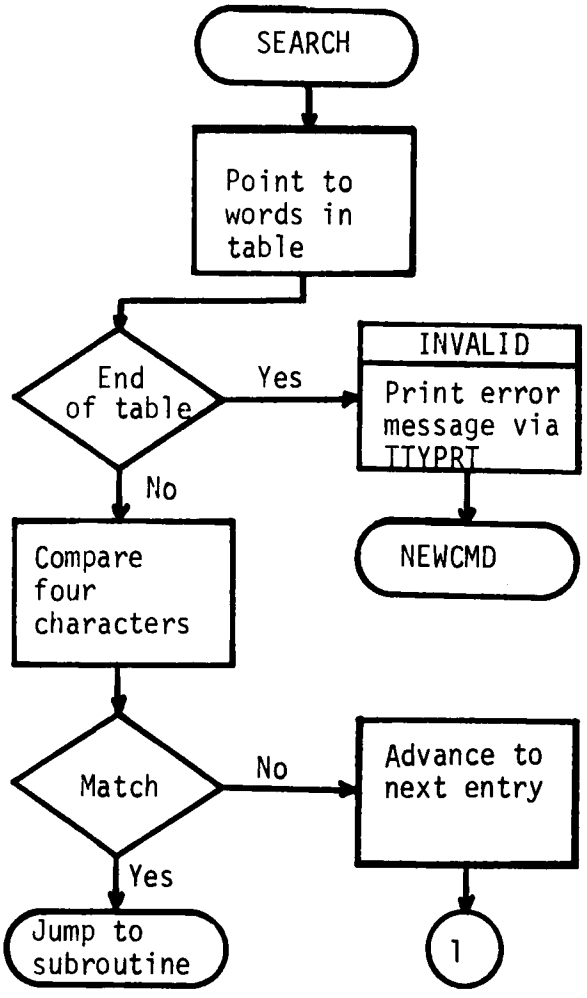

RULE
Initialize pointers to row = 2 col = -2
GTZXXX
Get number
Place into rule matrix, advance pointers
Last entry?
No
Yes
NEWCMD

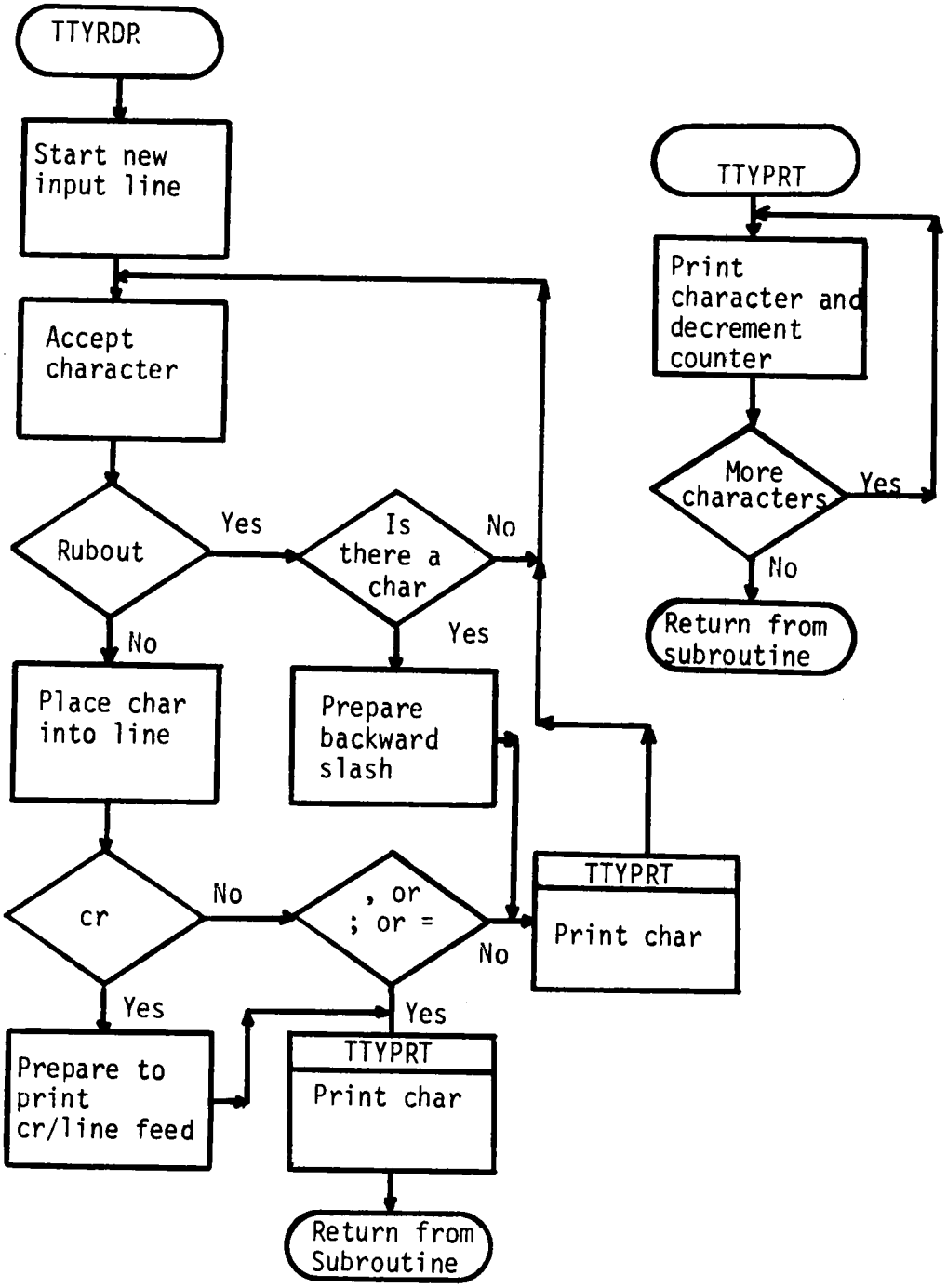

ZERO
Zero all cell data
Zero Epoch Counter
NEWCMD

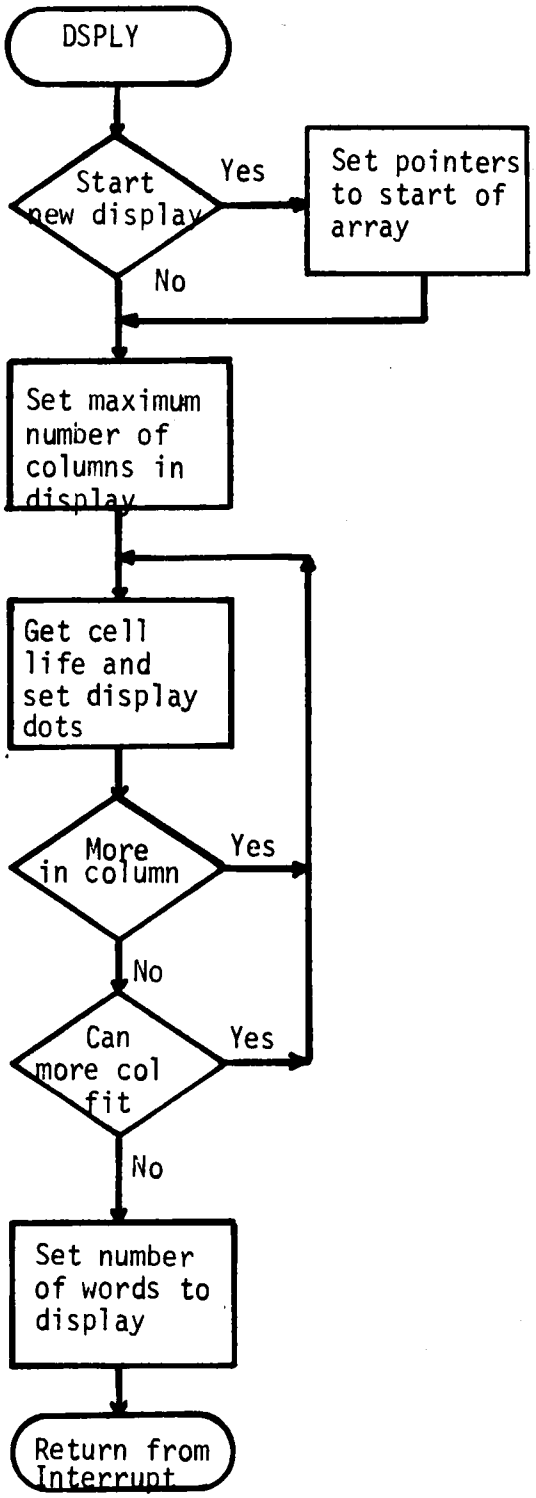

CELL
Command
Delete
End
Delete old by zeroing number
Sort into sequence
Add
Change
Set number
NEWCMD
Clear test-reset word
Set flag test control
Set current life test limits and control
Set calc life test limits and control
Set fuel test control
Set flag reset contro
Set reset life control and value
Set reset fuel control and value
Set go to instruction number

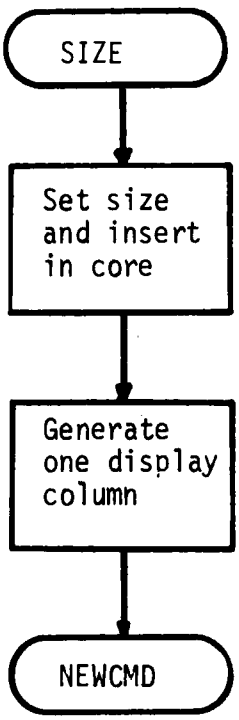

FLAG
LIFE
FUEL
Create flag mask
Create life mask
Create fuel mask
GTZXXX
get either character or number
LPINT
Locate cell on scope
Create change data word
PRTCEL
Print cell data before change
Enable light pen interrupt
Alter cell via change word and mask
New Value
TTY input
No Entry
!
Return from Interrupt
NEWCMD


TTYINT
"!"
Yes
No
Return from Interrupt
VALID
TTYPRT
Print "STOP Epoch=xxxcr?"
NEWCMD
TTYPRT
Print cr/line feed
TTYRDR
Get character string
Point to command table
SEARCH


SEARCH
Point to words in table
End of table
Yes
INVALID
Print error message via TTYPRT
NEWCMD
No
Compare four characters
Match
No
Advance to next entry
Yes
Jump to subroutine
1


TTYRDR
Start new input line
Accept character
Rubout
Yes
Is there a char
No
No
Yes
Place char into line
Prepare backward slash
cr
No
, or ; or =
No
Yes
Yes
Prepare to print cr/line feed
TTYPRT
Print char
Return from Subroutine
TTYPRT
Print char
TTYPRT
Print character and decrement counter
More characters
Yes
No
Return from subroutine


DSPLY
Start new display
Yes
Set pointers to start of array
No
Set maximum number of columns in display
Get cell life and set display dots
More in column
Yes
No
Can more col fit
Yes
No
Set number of words to display
Return from Interrupt


SIZE
Set size and insert in core
Generate one display column
NEWCMD

```
.TITLE  BEM
;R. C. FUSARINI     BASIC ENVIRONMENTAL MODEL SIMULATOR
;
;INSTRUCTIONS THAT ARE REQUIRED ONLY FOR THE
;CRT DISPLAY ARE MARKED AS COMMENTS
;WITH THE CHARACTERS......;$$$$......
;IF A USER SHOULD DESIRE TO RUN WITHOUT THE CRT
;THESE INSTRUCTIONS AND THE SUBROUTINES....DSPLY
;GENER AND LPINT SHOULD BE
;REMOVED FROM THE PROGRAM.
;THESE 5 CHARACTERS MUST BE REMOVED
;IF THE CRT IS USED
.ASECT
.ENABL  AMA
. = 40000   ;START
;SYSTEM DEFINITIONS
R0 = %0
R1 = %1
R2 = %2
R3 = %3
R4 = %4
R5 = %5
R6 = %6         ;ALSO USED AS SP
SP = %6
PC = %7
TSR = 177560    ;TTY STATUS REG
TKB = 177562    ;TTY INPUT CHAR BUFFER
TPS = 177564    ;TTY PRINTER STATUS BUFFER
TPB = 177566    ;TTY PRINTER BUFFER
LPS = 177514    ;LINE PRINTER STATUS REG
LPB = 177516    ;LINE PRINTER BUFFER
SWR = 177570
;$$$$DSPCSR = 164060;DISPLAY CONTOL & STATUS REG
;$$$$XPOS = 164062
;$$$$YPOS = 164064
;$$$$STEP = 164066
;$$$$XSTEP = 164066
;$$$$YSTEP = 164067
;$$$$CMA1 = 164070  ;ADDRESS OF SOURCE DATA
;$$$$CMA2 = 164072  ;ADDR NOT USED THIS DISPLAY TECH
;$$$$WDCNT = 164074 ;WORD COUNT (LOAD W/ NEG OF NUMBER)
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       DSPLY   ENTER VIA INTERRUPTF FROM DISPLAY
;AT INTERRUPT, CREATE DISPLAY POINTS FROM CELLS INFO
;IF COL LAST DONE = 0, THEN MUST RESTART AT BEGIN OF MAIN
DSPLY:   MOV   R0,-(SP)
```

```
        MOV   R1,-(SP)
        MOV   R2,-(SP)
        MOV   R3,-(SP)
        MOV   R4,-(SP)
        MOV   R5,-(SP) ;SET R5 TO HOLD INTENSITY
        MOV   #040000,R2     ;HALF TONE
        MOV   #170000,R5     ;SET OFF/ON BITS IN DISPLAY
        MOV   DSPGRP,DSHOLD ;NUMBER OF GROUPS THAT FIT IN
        TST   DCOL      ;IF ZERO NEED TO RESTART MAIN
        BGT   DSNXCL    ;JUST CONTINUE WITH NEXT COL
        MOV   #MAIN,PNTM     ;RESTART...RESET POINTER
        DEC   DSPCNT    ;ABOUT TO DO ONE MORE
        CLR   XPOS
        CLR   YPOS      ;RESTART
        MOV   HSIZE,DCOL     ;ALL COL MUST BE DONE
DSNXCL: MOV   #DSPTAB,R3     ;ADDR OF DSP START
        MOV   PNTM,R1   ;WORK WITH PNTM IN A REG FOR SPEED
DSCOL:  MOV   HSIZE,R4 ;POINT TO SECOND DISPLAY COL
        MOV   R4,DROW   ;RESET NUMBER OF CELLS IN ROWS
        ASL   R4        ;R4 = R3 + 4*SIZE
        ASL   R4
        ADD   R3,R4     ;SECOND SET OF DISPLAY DOTS
DSCEL:  MOV   (R1)+,R0 ;GET CELL INFO
;*
;TST   TYPE- LATER MAY PERMIT DISPLAY OF FLAG,LIFE,FUEL
;FOR NOW ONLY SHOW LIFE
;*
        BIC   #100377,R0 ;LEAVE JUST THE LIFE FACTOR
        SWAB  R0  ;MOVE TO LOW BYTE, DO NOT CONVERT (-12)
        BIC   R5,(R3)+ ;CLEAR FOR INTENSITY
        BIC   R5,(R3)+ ;OF LIFE OR HALF TONE
        BIC   R5,(R4)+
        BIC   R5,(R4)+
DOT4:   CMP   R0,#112. ;GREATER THAN 100., SHOW ALL 4 DOTS
        BGE   SDOT4     ;>= 100.
        BIS   R2,-4(R3);HALF TONE
DOT3:   CMP   R0,#87.  ;SHOW 3 DOTS >=75
        BGE   SDOT3
        BIS   R2,-2(R3)
DOT2:   CMP   R0,#62.  ;>= 50.
        BGE   SDOT2
        BIS   R2,-4(R4)
DOT1:   CMP   R0,#37.
        BGE   SDOT1     ;>=25.
        BIS   R2,-2(R4)
        BR    DSSET     ;DISPLAY POINTS FOR CELL SET
SDOT4:  BIS   R5,-4(R3)
SDOT3:  BIS   R5,-2(R3)
SDOT2:  BIS   R5,-4(R4)
```

```
SDOT1:  BIS   R5,-2(R4)
DSSET:  DEC   DROW      ;HANDLED ONE CELL
        BGT   DSCEL     ;GET NEXT CELL
        TST   (R4)+     ;COMPLETED COL
        MOV   R4,R3     ;OVER SKIP WORD, RESET R3
        DEC   DCOL      ;HANDLED ONE MORE COL
        BLE   DSSTRT    ;NO MORE CELLS IN MAIN, START DSP
        DEC   DSHOLD    ;CAN DSPTAB HOLD ANY MORE GROUPS?
        BGT   DSCOL     ;YES, GENERATE NEXT GROUP
DSSTRT: SUB   #DSPTAB,R4  ;NUMBER OF WORDS TO TRANSFER
        ASR   R4        ;THERE ARE 2 BYTES PER WORD
        INC   R4        ;FOR THE SKIP WORD BLOCK END
        NEG   R4        ;WDCNT HAS NEG OF COUNT
        MOV   R4,WDCNT  ;SET UP FOR DMA ON DSP
        MOV   R1,PNTM   ;SAVE LOCATION IN MAIN
        MOV   (SP)+,R5
        MOV   (SP)+,R4
        MOV   (SP)+,R3
        MOV   (SP)+,R2
        MOV   (SP)+,R1
        MOV   (SP)+,R0
        MOV   #DSPTAB,CMA1   ;LOAD START ADDR
        INC   DSPCSR    ;START DISPLAY...SET GO BIT
        RTI
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       GENER   VIA JSR
;FILL DISPLAY TABLE AREA WITH DISPLAY DATA WORDS TO STEP Y
;INTENSITY DATA INSERTED IN DSPLY
GENER:  MOV   R2,-(SP)
        MOV   R3,-(SP)
        MOV   R4,-(SP)
        MOV   #DSPTAB,R2;ADDR OF START OF DSPTAB
        MOV   DSPSEC,R3 ;ADDR OF SECOND SECTION OF DSPTAB
        MOV   NDSPTS,R4 ;NUMBER OF WORDS TO COPY
CGEN:   MOV   (R2)+,(R3)+
        DEC   R4
        BGT   CGEN
        MOV   (SP)+,R4
        MOV   (SP)+,R3
        MOV   (SP)+,R2
        RTS   PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
```

```
;
;         LIGHT PEN INTERRUPT
;PRINT  '@RRR,CCC=XLLL,FFF'   RRR=ROW, CCC=COL,
;X = S/C SET OR CLEAR
;LLL= LIFE FACTOR, FFF= FUEL FACTOR
;HANDLES REPLACEMENT OF ALL DATA FIELDS IN CELL
;USES R5...MASK WORD INDICATES BITS
LPINT:  MOV    R0,-(SP)
        MOV    R1,-(SP)
        MOV    R3,-(SP)
        MOV    R4,-(SP)
        MOV    R2,-(SP)
        MOV    R5,-(SP)
        MOV    WDCNT,R3 ;GET WDCOUNT
;ROW IS INDICATED BY THE NUMBER OF ENTRIES THAT
;HAVE BEEN DISPLAYED...THE COL CAN BE DETERMINED
;FROM THE XPOS, BUT THE ROW MUST BE DETERMINED BY
;COUNTING FROM THE TOP DOWN...IT IS FIRST NEEDED
;TO HANDLE THE GROUP SKIP WORD (BACK OUT 1)....
;THEN ROUTINELY TAKE OUT FOR COL SKIP
;SUB SIZE*2...IF NEG GOT IT...ELSE
;RESTART AND CONT UNTIL NEG
LPRAGN: ADD    #1,R3        ;SKIP WORD
        ADD    HSIZE,R3
        ADD    HSIZE,R3
        BPL    LPRW         ;GOT IT
        ADD    HSIZE,R3
        ADD    HSIZE,R3
        BPL    LPRW
        BR     LPRAGN       ;TRY AGAIN
LPRW:   ASR    R3           ;PAIRED DOTS
        MOV    R3,LROW      ;DOT POSITION FROM BOTTOM UP
        MOV    HSIZE,R2
        SUB    R3,R2        ;GET ROW FOR PRINTING
        MOV    #INLINE,R1   ;PREP FOR PRINT
        MOVB   #100,(R1)+   ;'@' SYMBOL
        JSR    PC,OASTRT    ;CONVERT  INTO PRINT LINE
        MOVB   #54,(R1)+    ;',
        MOV    #5,R0        ;PRINT 5 CHAR  @RRR,
        MOV    #INLINE,R1
        JSR    PC,TTYPRT
        MOV    XPOS,R2      ;CALC COL LOCATION
        BIC    #170000,R2   ;XPOS IS ONLY BITS 0-11
        MOV    DISZ,R3      ;XPOS/(STEPSIZE*3)+1
        ASL    R3
        ADD    DISZ,R3
        JSR    PC,DIVIDE
        MOV    R3,R2
        INC    R2           ;COL COUNT FROM LEFT
```

```
        MOV   #INLINE,R1
        JSR   PC,OASTRT
        MOVB  #75,(R1)+   ; '=
        MOV   #4,R0       ;PRINT 4 CHAR  'CCC='
        MOV   #INLINE,R1
        JSR   PC,TTYPRT
        MOV   R3,R2       ;LOCATE CELL IN CORE
        MOV   HSIZE,R3    ;((COL-1 *SIZE)+RMW)*2+START MAIN
        JSR   PC,MULT     ;ALREADY COL=1
        ADD   LROW,R2
        ASL   R2          ;2 BYTES/CELL
        ADD   #MAIN,R2    ;START OF MAIN IN CORE
        MOV   R2,R4
        MOV   (R4),R0     ;PREP FOR NEXT SUBROUTINE
        JSR   PC,PRTCEL   ;PRINT CELL DATA
        MOV   #CRLF,R1
        MOV   #2,R0
        JSR   PC,TTYPRT
        MOV   (SP)+,R5    ;REPLACE INDICATOR
        MOV   (SP)+,R2    ;NEW DATA WORD ON STACK
        BIC   R5,(R4)     ;CLEAR BITS TO BE SET
LPBITS: BIS   R2,(R4)     ;SET TO NEW VALUE
        MOV   (SP)+,R4
        MOV   (SP)+,R3
        MOV   (SP)+,R1
        MOV   (SP)+,R0
        INC   DSPCSR      ;RESTART DISPLAY
        RTI
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       RUN          ENTER VIA COMMAND TABLE
RUN:;$$$$     CLR   DSPCSR      ;STOP DISPLY
        CMPB  #015,(R2)   ;IF= CR ASSUME 1 EPOCH ONLY
        BNE   RUNGNM
        MOV   #1,R2
        BR    RUNCOK
RUNGNM: JSR   PC,TTYRDR   ;GET NUMBER
        JSR   PC,ATBIN    ;CONVERT TO BINARY
        TST   R2          ;0<=XXX<=100
        BMI   RUNCBD
        CMP   R2,#100.
        BLE   RUNCOK
RUNCBD: JSR   PC,INVALID  ;PRINT GOOF
        BR    RUNGNM      ;GET NUMBER
RUNCOK: MOV   R2,RUNCNT   ;PLACE COUNTER
        MOV   #100,TSR    ;ENABLE TTY INTERRUPT DURING RUN
```

```
RUNNEW:  JSR    PC,GENER
;START DISPLAY        SWITCH REG USED TO CONTROL
;LENGTH OF DISPLAY TIME....DECREMENTED IN DISPLAY ROUTINE
;$$$$           MOV    SWR,DSPCNT
          CLR    DCOL   ;INDICATE TO RESTART DISPLAY
;$$$$           MOV    #-1,WDCNT
;$$$$           MOV    #001525,DSPCSR ; 0,000,001,101,010,101
CNTDSP:;$$$$   TST    DSPCNT
;$$$$           BGT    CNTDSP
          DEC    RUNCNT      ;IF NEG GO FOR NEW COMMAND
          BPL    STARCN      ;NEGATIVE, COMPLETE
          JMP    VALINT
;CALCULATE NEW LIFE FACTORS FOR NEW ARRAY NEXT EPOCH
;ARRAY STORAGE BY COL FROM COL 1 THRU COL SIZE
;ROWS ARE STORED FROM ROW = SIZE TO ROW = 1
;THIS IS DONE TO FACILITATE DISPLAY TECHNIQUE
;    MAIN   IS THE MAIN MATRIX TO BE HANDLED
;    WK     IS THE POINTER TO WORKING MATRIX.
;IT IS 5*SIZE ARRAY...CALC DONE HERE COL BY COL..
;WHEN A COL IS PROCESSED THE LOW COL
;IS TRANSFERED BACK INTO THE MAIN ARRAY (IF APPLICABLE)
;    MRULE IS THE 5*5 RULE ARRAY STORED SAME TECHNIQUE
;    PNTM  POINTER TO THE MAIN ARRAY FROM WHICH ALL CALC
;IS DONE (THIS IS THE CENTER CELL IN RULE ARRAY)
;    PNTMW POINTER TO MAIN STEPS THROUGH WITH THE RULES
;    PNTR  POINTER INTO RULE INDICATING CELL EFFECT CAL
;    EXAMPLE                    ASSUME SIZE = 16
;IF PNTM => COL 6, ROW 5 AND THE FIRST CELL OF THE RULE
;IS BEING PROCESSED...THEN PNTR => COL 1,ROW 5
;PNTMW=> COL 4, ROW 7
;PTWK  POINTS  TO CELL EFFECT BEING CONSIRERED  WK ARRAY
;IN ORDER TO PROCESS THE ENDS OF THE ROW AND COL WITHOUT
;GOING OFF THE ARRAY THE FOLLOWING COUNTERS ARE USED
;IF THE SIZE OF THE RULE MATRIX IS CHANGED FROM 5*5 THESE
;MUST BE MODIFIED
;    CM IS COL OF PNTM          RM ROW OF PNTM
;    CMW COL OF PNTMW           RMW ROW OF PNTMW
;    CR COL OF PNTR             RR ROW OF PNTR
;    CLEAR WORK AREA FOR 5* SIZE SPACE
STARCN:  MOV    #5,R3     ;START CALC FOR NEW ARRAY
;$$$$           BIC    #100,DSPCSR   ;CUT OFF DISPLAY
          MOV    HSIZE,R2
          INC    EPOCH     ;INCREMENT THE EPOCH COUNTER
          TST    CWALK     ;CHANGE RANDOM NUMBER EACH EPOCH??
          BNE    NORANE    ;NO
          JSR    PC,RANDOM;GET NEW RANDOM NUMBER
NORANE:  JSR    PC,MULT   ;R2=SIZE*5=SIZE OF WORKAREA
          MOV    WK,R3     ;SETUP POINTER
;ZERO THE WORKING ARRAY
```

```
CNCLWK: CLR    (R3)+     ;CLEAR AND ADVANCE
        DEC    R2        ;DECREMENT COUNTER
        BGT    CNCLWK    ;MORE TO GO???
;STEP THRU  COLUMNS ...LEFT TO RIGHT (COL 1= SIZE)
        MOV    #1,CM     ;START WITH TO COL 1
        MOV    #MAIN,PNTM    ;POINT TO FIRST CELL
NXCELC: MOV    HSIZE,RM  ;WORKING ON BOTTOM ROW
        MOV    WK,PTWK   ;THIS IS LOWER LEFT EDGE OF WK
;
;START LOOP #1
;
NXCELR: SUB    #4,PTWK   ;FAKE LOC AFFECTED (COL=1,ROW=4)
;GET THE LIFE FACTOR OF THE BASE CELL FROM THE MAIN
        MOV    @PNTM,R0  ;USING POINTER IN CORE
        BIC    #100377,R0    ;ONLY BITS 14-8 ARE LF
        SWAB   R0        ;MOVE TO LOW BITS (7)
        SUB    #12.,R0   ;MUST USE NORMAL NUMBER FOR CALC
        BEQ    NXEND     ;SKIP (EVEN W/WALK) NO AFFECT
;GOT LIFE FOR GIVEN CELL NOW CALC AFFECT ON NEIGHBORS
;START AFFECTING NEIGHBOR CELL -2 COL - 2ROWS
;FROM PNTM POSITION...R0=LIFE OF MAIN CELL
        TST    CWALK     ;GET RANDOM NUMBER EACH MAIN CELL?
        BLE    NORANM    ;NO
        JSR    PC,RANDOM
NORANM: MOV    #SRULE,PNTR    ;START RULE PROCESSING
        MOV    CM,CMW    ;SET COL & ROW INDICATORS
        SUB    #2,CMW    ;GO LEFT 2 COL
;CHECK  THE COL TO INSURE THAT IT IS LEGAL (NOT OFF ARRAY)
;
;START LOOP #2
;
        MOV    #5,R4     ;5 COL TO PROCESS
NXTCOL: MOV    #5,R5     ;5 ROW MEMBER IN COL
        MOV    RM,RMW
        ADD    #2,RMW    ;DOWN 2 ROWS
        TST    CMW       ;TOO FAR TO LEFT???
        BGT    NXTROW    ;PROCESS THE MEMBERS OF THIS COL
        ASL    R5        ;ADVANCE 2*SIZE (2 BYTES / CELL)
        ADD    R5,PNTR   ;BAD COL...ADVANCE RULE POINTER
        ADD    R5,PTWK   ;MOVE POINTER IN WK
        BR     NXCADV    ;ACT AS IF THIS COL WAS PROCESSED
;
;START LOOP #3
;
NXTROW: CMP    RMW,HSIZE;HAS IT GONE TOO FAR DOWN
        BGT    NXRADV    ;ACT AS IF PROCESSED (SKIP FAKE)
;PROCESS  LIFE FACTOR OF CELL IN MAIN ARRAY  AT PNTM (RM,RM)
;LF(PNTM) * RULE(PNTR...(RR,CR),..)+LF(PTWK)  PLACE IN PTWK
;R0=LF(PNTM)...R3=RULE(RR,CR)
```

```
          MOV    @PNTR,R3
          MOV    R0,R2
          JSR    PC,MULT
          TST    CWALK      ;CHANGE RANDOM NUMBER EACH RULE?
          BPL    NORANR
          JSR    PC,RANDOM
NORANR:   TST    HWALK      ;IF NO RANDOM NUMBER SKIPFOLLOWING
          BEQ    NOMULT     ;DO NOT SHIFT RESULT
;IF THERE IS NO RANDOM NUMBER, ONE WORD CAN HOLD THE
;FULL RESULT AND LATER BE DIVIDED BY 100.
;HOWEVER...IF THERE IS A RANDOM NUMBER,
;THERE MUST BE ONE MORE MULTIPLICATION AND THE RESULT
;WILL BE MORE THAN ONE WORD....THE LAST
;6 BITS OF THE PRESENT RESULT MAY BE SIFTED OUT
;BECAUSE IT IS LESS THAN 1% OF THE RESULT
          ASR    R2
          ASR    R2
          ASR    R2
          ASR    R2
          ASR    R2
          ASR    R2
          ADC    R2         ;ADD LAST BIT FOR ROUND OFF
          MOV    WLKMUL,R3
          JSR    PC,MULT
NOMULT:   ADD    R2,@PTWK
          BVC    NXRADV     ;OVERFLOW???
          NEG    @PTWK      ;ERROR MADE...RESET TO MAX POS
          BIS    #077777,@PTWK   ;OR NEG AS APPROPRIATE
NXRADV:   ADD    #2,PNTR    ;ALL CELL IN RULE IN SEQ
          ADD    #2,PTWK    ;MOVE TO NEXT MEMBER IN ROW
          DEC    R5         ;IF ZERO END OF COL
          BLE    NXCADV     ;FINISHED THIS COL GO TO NEXT
          DEC    RMW
          BLE    NXRADV     ;<= 0 NO PROCESS, BUT MOVE POINTER
          BR     NXTROW     ;PROCESSED ALL MEMBERS LAST COL
;ADV TO NEXT-IF CMW>SIZE, SKIP OUT
;
;LOOP #3...END
;
NXCADV:   DEC    R4         ;REDUCE COUNTER OF COL
          BLE    NXEND      ;NO MORE RULE, END ROUTINE
          INC    CMW        ;GO TO RIGHT 1 COL
          CMP    CMW,HSIZE;GONE TOO FAR TO RIGHT   ????
          BGT    NXEND      ;QUIT IF YES    MUST NOW ADVANCE
;2*SIZE FROM PLACE OF FIRST ROW MEMBER LAST COL
          SUB    #10.,PTWK  ;ALREADY DID 5 MEMBERS LAST COL
          ADD    HSIZE,PTWK ;ADVANCE NOW ONE COL IN WORK
          ADD    HSIZE,PTWK ;2 BYTES / ENTRY
          BR     NXTCOL     ;WORK ON NEXT COL
```

```
;
;LOOP#2...END
;
;PROCESSED ALL CELLS IN NEIGHBORHOOD  OF CELL(RM,CM)
NXEND:   ADD    #2,PNTM   ;ADVANCE ONE CELL IN MAIN
         DEC    RM        ;UP ONE ROW
         BLE    NXCLM     ;FINISHED THAT COL IF ROW = 0
;EXAMPLE...PNTM WAS (4,3) BUT NOW IS ( 4,2)
;THE WORKING POINTER MUST POINT TO THE FIRST AFFECTED CELL
;IN WORK AREA...COL=1, ROW= RM-2(DONE AT BRANCH BACK)
         MOV    HSIZE,R3  ;RESET WORKING POINTER
         SUB    RM,R3     ;WK + SIZE - RM
         ASL    R3        ;OR 2*(SIZE-RM) + WK
         ADD    WK,R3
         MOV    R3,PTWK   ;RESET
         BR     NXCELR    ;WORK NEXT MAIN CELL
;
;LOOP#1...END
;
NXCLM:   CMP    CM,#3     ;IF <3 COL 1 & 2 NOT FINISHED CALC
         BLT    NXCLM2    ;SHIFT AND CONT FOR <3
;FINISHED COL THAT GOES BACK INTO MAIN (2 COL BACK)
;(IF JUST FINISHED COL3, PNTM POINTS TO COL 4 BOTTOM ROW
;AND COL 1 IS NOW READY TO BE REPLACED IN MAIN FROM THE
;FIRST COL OF THE WORK MATRIX)...PNTM=SIZE*3*2
         MOV    HSIZE,R2  ;USE PNTM TO GET REPLACE ADDR
         MOV    R2,R1
         ASL    R1
         ADD    R2,R1
         ASL    R1
         NEG    R1
         ADD    PNTM,R1   ;USE R1 FOR POINTER FOR "TO" ADDR
         MOV    WK,R0     ;"FROM" START OF WK
;SUBROUTINE HANDLES ONE COL...
;EXECUTES INSTRUCTIONS AND REPLACES ALL MEMBERS
         JSR    PC,CNDVRP
NXCLM2:  CMP    CM,HSIZE  ;LAST COL PROCESSED???
         BGE    NXLAST    ;LAST...MOVE LAST 2 COL
         JSR    PC,SHIFT  ;CONT...MOVE TOP 4 COL DOWN 1 COL
         INC    CM        ;NEXT COL
         JMP    NXCELC
NXLAST:  JSR    PC,CNDVRP;R0,R1 WERE ADVANCED LAST PASS
         JSR    PC,CNDVRP;GET LAST 2 COL
         JMP    RUNNEW    ;GO BACK TO BEGINNING FOR NEXT PASS
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
```

```
;         INSTRT     VIA JSR
;EXECUTES INSTRUCTIONS FOR ONE CELL
;INPUT CELL DATA BROKEN DOWN INTO FLGFUL (FLAG,FUEL)
;CALLIF (CALCULATED LIFE),CURLIF (CURRENT LIFE)
;OUTPUT SAME FIELDS  -  LIFE DATA IS IN  +12 BASE
INSTRT: MOV    R0,-(SP)
        MOV    R1,-(SP)
        MOV    R2,-(SP)
        CLR    R2          ;GET FIRST INST
INSTNX: JSR    PC,FIND     ;GET NEXT INST
        TST    R2          ;IF + NO INST, IF 0 END
        BEQ    ITERM       ;STANDARD TERMINATION
        TST    R1          ;IF ZERO NO INST
        BNE    IX
        MOV    BADINT+17,R1
        JSR    PC,OASTRT
        MOV    #21.,R0
        MOV    BADINT,R1
        JSR    PC,TTYPRT;PRINT ERROR
        HALT
ITERM:  MOV    (SP)+,R2
        MOV    (SP)+,R1
        MOV    (SP)+,R0
        RTS    PC
;;TEST FOR CONDITIONS
IX:     BIT    #40000,2(R1)    ;TEST FLAG?
        BEQ    IXTOL       ;NO TEST
        TST    2(R1)       ;CLEAR/SET???
        BMI    IXTFGS      ;TEST SET
        TST    FLGFUL      ;IS IT CLEAR???
        BPL    IXTOL       ;YES
        JMP    IXEND       ;NO IS NOT,,,GO TO NEXT INST(SEQ)
IXTFGS: TST    FLGFUL      ;SET???
        BMI    IXTOL
        JMP    IXEND       ;NO, CLEAR
;
;         WILM              VIA  JSR
;TEST WITHIN LIMITS....USED BY NEXT THREE SECTIONS
;INPUT   R0 LIMITS(LOW,HIGH) R2= VALUE TO TEST
;OUTPUT  R0=    += YES    -= NO
WILM:   CMPB   R2,R0       ;CHECK HIGH FIRST
        BGT    WILMNO      ;NOT WITHIN LIMITS
        SWAB   R0          ;GET LOW OVR
        CMPB   R2,R0       ;CHECK LOW
        BLT    WILMNO
        MOV    #1,R0       ;OK...WITHIN LIMITS
        BR     WILMRT      ;RETURN
WILMNO: MOV    #-1,R0
WILMRT: RTS    PC
```

```
;
IXTOL:  BIT   #20000,2(R1)  ;TEST CURRENT LIFE  ??????
        BEQ   IXTNL     ;NO TEST
        MOV   4(R1),R0 ;GET LIMITS
        MOV   CURLIF,R2
        JSR   PC,WILM  ;TEST WITHIN LIMITS
        TST   R0
        BMI   IXEND     ;NO
IXTNL:  BIT   #10000,2(R1)  ;TEST CALC LIFE
        BEQ   IXTFL
        MOV   6(R1),R0
        MOV   CALLIF,R2
        JSR   PC,WILM
        TST   R0
        BMI   IXEND
IXTFL:  BIT   #4000,2(R1)   ;TEST FUEL
        BEQ   IXRFG     ;NO START RESETING
        MOV   8.(R1),R0
        MOV   FLGFUL,R1
        JSR   PC,WILM
        TST   R0
        BMI   IXEND
;ALL TESTS SATISFIED...START RESETTING AS DIRECTED
IXRFG:  BIT   #100,2(R1)    ;RESET FLAG???
        BEQ   IXRLF     ;NO
        BIT   #200,2(R1)
        BEQ   IXRFC     ;CLEAR THE FLAG
        BIS   #100000,FLGFUL
        BR    IXRLF
IXRFC:  BIC   #100000,FLGFUL
;LIFE RANGE IS NOT CHECKED (DONE BEFORE)
IXRLF:  BIT   #60,2(R1);RESET LIFE???
        BEQ   IXRFU     ;NO ...TRY FUEL
        BIT   #20,2(R1);CALC VALUE
        BEQ   IXRLFR
        MOV   CALLIF,CURLIF
        BR    IXRFU
IXRLFR: BIT   #40,2(R1);RESET TO VALUE XXX
        BEQ   IXRFU
        CLR   CURLIF
        MOVB  11.(R1),CURLIF
IXRFU:  BIT   #3,2(R1) ;RESET FUEL
        BEQ   IXRIN
        MOV   2(R1),R0
        BIC   #177774,R0   ;LEAVE JUST BITS 0-2
        CMP   #3,R0     ;RESET???
        BNE   IXRFU2
        MOVB  10.(R1),FLGFUL
        BR    IXRIN
```

```
IXRFU2: CMP    #2,R0     ;PERCENT CONSUME
        BNE    IXRFU1
        MOVB   10.(R1),R2     ;PREP FOR MULT
        MOVB   FLGFUL,R3      ;FUEL
        JSR    PC,MULT
        BR     IXRFUC    ;SUBTRACT FROM FUEL THERE
IXRFU1: MOVB   10.(R1),R2
IXRFUC: MOVB   FLGFUL,R3
        SUB    R2,R3
        BPL    IXRFUP    ;RESULT MUST BE 0=<XXX=<100
        CLR    R3
        BR     IXRFUZ
IXRFUP: CMP    R3,#100.
        BLE    IXRFUZ
        MOV    #100.,R3
IXRFUZ: MOVB   R3,FLGFUL
IXRIN:  MOVB   1(R1),R2 ;NEXT INST NUMBER
        BEQ    IXRINX    ;IF ZERO END OF INST
        JMP    INSTNX    ;EXECUTE NEXT INST
IXRINX: JMP    ITERM     ;TERMINATE
IXEND:  MOV    #-1,R2;   ;GET NEXT SEQ INST NUMBER
        JMP    INSTNX
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;CONVERT, DIVIDE, REPLACE LIFE FACTORS IN MAIN
;FROM R0 INTO R1(MAIN) ENTIRE COL
CNDVRP: MOV    HSIZE,R4      ;COUNTER
;IF HWALK=0...USE 100.AS DIVISOR OF CALC LIFE
;ELSE  USE 100.*100.AS THE DIVISOR (SHIFTED RIGHT 6 BITS)
        MOV    #100.,R5 ;ASSUME 100.
        TST    HWALK
        BEQ    CNDVAG
        MOV    #234,R5  ;10000.=23420 (OCTAL)SHIFT IT
CNDVAG: MOV    (R0)+,R2 ;GET LF FULL WORD
        MOV    R2,R3     ;PREP FOR R2 = 0
        BEQ    CNDVMN    ;SKIP DIVISION IF NEW =0
        MOV    R5,R3     ;PREP FOR DIV
        JSR    PC,DIVIDE;CONVERT TO PERCENTAGE
        CMP    R3,#115.  ;MAX ALLOWABLE IN STORAGE TECH
        BLE    CNDVMN    ;OKAY CHECK FOR MIN
        MOV    #115.,R3 ;RESET TO MAX
CNDVMN: ADD    #12.,R3  ;CONVERT TO STORE TECH
        BPL    CNDVOK    ;SMALLEST NORMAL VALUE = -12
        CLR    R3        ;CLEAR TO REPRESENT -12
CNDVOK: MOV    R3,CALLIF;CALC LIFE
        MOV    (R1),R2  ;GET CURRENT DATA
```

```
        MOV   R2,R3
        BIC   #100377,R2     ;LEAVE THE LIFE
        BIC   #077400,R3     ;LEAVE FLAG/FUEL
        MOV   R3,FLGFUL
        SWAB  R2
        MOV   R2,CURLIF      ;GET CURRENT LIFE
        JSR   PC,INSTRT      ;EXECUTE INSTRUCTIONS
        CLR   (R1)
        SWAB  CURLIF         ;PUT IN HIGH END
        BIS   FLGFUL,(R1)    ;INSERT FLAG/FUEL
        BIS   CURLIF,(R1)+   ;INSERT LIFE
        DEC   R4             ;DECR COUNTER
        BGT   CNDVAG         ;GET NEXT CELL
        RTS   PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       SHIFT              ENTER VIA JSR
;MOVE TOP 4 COL ONE ARRAY COLUMN IN WK,CLEAR TOP COL
SHIFT:  MOV   HSIZE,R2   ;R1=START OF 2 COL WORK AREA
        ASL   R2         ;OR =2*SIZE +START ADDR WORK AREA
        MOV   WK,R0      ;R0=START OF WORK AREA
        MOV   R0,R1
        ADD   R2,R1
        ASL   R2         ;4*SIZE = # CELLS TO MOVE
SHFTNX: MOV   (R1)+,(R0)+    ;MOVE ONE CELL
        DEC   R2
        BGT   SHFTNX
        MOV   HSIZE,R2
SHFTCL: CLR   (R0)+      ;CLEAR TOP COL IN WK
        DEC   R2
        BGT   SHFTCL
        RTS   PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       PRINT CELL DATA           VIA JSR
;R0= CELL DATA...DOES NOT ALTER CELL
; ' XLLL,FFF'   X = S/C  LLL= LIFE FACTOR, FFF= FUEL
PRTCEL: MOV   R1,-(SP)
        MOV   R2,-(SP)
        MOV   R3,-(SP)
        MOV   #INLINE,R1     ;START OF OUTPUT LINE
        MOVB  #040,(R1)+     ;INSERT BLANK
        MOVB  #123,(R1)+     ;ASSUME SET'S'
```

```
        TST     R0
        BMI     PTCA            ;IF MINUS THEN CELL SET
        MOVB    #103,-1(R1)     ;IT IS CLEAR
PTCA:   MOVB    R0,R3           ;GET FUEL
        MOV     R0,R2
        BIC     #100377,R2      ;LEAVE JUST LIFE
        SWAB    R2              ;MOV TO LOW END
        SUB     #12.,R2         ;CONVERT TO REG NOTATION
        JSR     PC,OASTRT       ;CONVERT & MOV TO OUTPUT LINE
        MOVB    #054,(R1)+      ; ',',
        MOV     R3,R2
        JSR     PC,OASTRT       ;CONVERT & OUTPUT
        MOV     R0,-(SP)
        MOV     #INLINE,R1      ;START ADDR
        MOV     #9.,R0          ;PRINT 8 CHAR XLLL,FFF
        JSR     PC,TTYPRT
        MOV     (SP)+,R0
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        RTS     PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       GTZXXX          JSR FROM CELL LIFE/FUEL
;R2 = BINARY NUMBER ...R0 = CHAR READ
GTZXXX: JSR     PC,TTYRDR
        MOV     R2,R0
        CMPB    -2(R2),#060     ;IF NUMBER THEN CONVERT
        BLT     GTZ             ;MUST BE JUST CHAR
        CMPB    -2(R2),#071
        BGT     GTZ
        JSR     PC,ATBIN        ;CONVERT TO BINARY
        SUB     #4,R0           ;BACK UP TO CHAR
        BR      GTZZ
GTZ:    DEC     R0              ;BACK UP OVER DELIM
        CLR     R2              ;NO NUMBER READ
GTZZ:   MOVB    -(R0),R0        ;GET CHAR...IF NO CHAR GIVEN
        RTS     PC  ;CALLING PROG RESPONSIBLE FOR CHECKING
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       RANDOM  JSR
;GENERATOR PROPOSED BY D. H. LEHMER IN EVANS,WALLACE
;AND SUTHERLAND - 'SIMULATION USING DIGITAL COMPUTERS'
```

```
;    R(I+1) = <<(10**K)*(R(I))> + R(I) + C>
;WHERE <X> DENOTES FRACTIONAL PART OF NUMBER 0<R<1
;    P  DIGITS IN RANDOM NUMBER    2=< K < P
;    C = A * 10**(-P) WHERE 0<A<10**P AND  A NOT = N*2
;           OR A NOT = N*5
;    R(0) = R*10**(-P) &  0<R<10**P
;ALTERED TO BASE 2 FOR THE PROG
;    R(I+1) = <<(2**K)*(R(I))>+R(I)+C>
;    WHERE    2=< K < P ...DECIDED BY NONE ZERO BIT
;    LET C BE DECREMENTED FROM THE MAX ERROR-1 UNTIL = 0
;DECREASED FOR EACH RANDOM NUMBER GENERATED
;    <X>  DENOTES W*(-1) <=R(I) =<W
;    R(0) = W...LIMITED BY TRUNCATION (IF TOO LARGE)
;    SIGN OF RESULT IS BIT 2      R(0) = W
;    LIMIT FUNCTION PERFORMED BY TRUNCATION THEN CHECK
;    AGAINST  W .    IF TOO GREAT GENERATES AGAIN.
RANDOM: TST   HWALK     ;IF ZERO DISREGARD ALL OF THIS
        BEQ   RANQUT    ;DO NOT NEED RANDOM NUMBER
        MOV   R3,-(SP)
        MOV   R2,-(SP)
        MOV   R0,-(SP)
        MOV   WLKRAN,R3
        MOV   R3,R2     ;USE FOR SHIFT TESTING
RANAGN: MOV   #10.,R0   ;MAX TIMES TO TRY LOOP
        DEC   CON       ;DECREMENT ON EACH PASS
        BGT   RANDM1    ;UNLESS C = 0...
        MOV   HWALK,CON;THEN RESET TO MAX ERROR-1
        DEC   CON
RANDM1: MOV   R3,R2     ;USE FOR SHIFT DECISION
        ASR   R2        ;SHIFT ONE LESS THAN MAX DIGITS
        ASR   R2        ;SHIFT AT LEAST ONCE
        ASL   R3
        ASR   R2        ;SHIFT AT LEAST TWICE
RANSFT: ASL   R3
        ASR   R2        ;SHIFT UNTIL ZERO BIT
        BCS   RANSFT
        BIC   MWALK,R3 ;CLEAR EXCESS BITS VIA TRUNCATION
        CMP   R3,HWALK ;IF LE THEN WITHIN RANGE
        BLE   RANDM2    ;OK
        SUB   HWALK,R3 ;REDUCE AND TRY AGAIN
        DEC   R0        ;DO NOT GET INTO ENDLESS LOOP
        BGT   RANDM1
RANDM2: ADD   WLKRAN,R3
        NEG   RANSGN    ;ALTERNATE SIGN EACH PASS
        ADD   CON,R3
        BIC   MWALK,R3
        CMP   R3,HWALK ;IF LE THEN WITHIN RANGE?
        BLE   RANEND
        SUB   HWALK,R3
```

```
          DEC    R0        ;DO NOT LOOP TOO MANY TIMES
          BGT    RANDM2
          MOV    #10.,R0   ;PRINT ERROR MESS
          MOV    #WLAKFL,R1
          JSR    PC,TTYPRT
          BR     RANAGN    ;TRY AGAIN
RANEND:   MOV    (SP)+,R0
          MOV    R3,WLKRAN;SAVE FOR NEXT PASS
          TST    RANSGN
          BPL    RANEP     ;RANDOM NUMBER IS POS
          NEG    R3        ;MAKE NEG
RANEP:
;******TEST INSERT HERE FOR RANDOM NUMBER PRINTOUT
;******    JSR    PC,PRTRAN
          ADD    #100.,R3 ;100%+RANDOM ERROR (+ OR -)
          MOV    R3,WLKMUL;USED FOR MULT IN MAIN ROUTINE
          MOV    (SP)+,R2
          MOV    (SP)+,R3
RANOUT:   RTS    PC
;
;
;******TEST INSERT PRINT RANDOM NUMBERS
.NLIST BEX
RANPRT:.ASCII /                              /
       .ASCII /                              /
       .BYTE  15,12
.EVEN
.LIST BEX
PRTPNT:  RANPRT + 3
RANPCT:  0   ;RANDOM PRINT COUNTER
          .EVEN
PRTRAN:  MOV    R2,-(SP)
         MOV    R1,-(SP)
         MOV    R3,R2
         ADD    #5,PRTPNT
         MOV    PRTPNT,R1
         JSR    PC,OASTRT
         INC    RANPCT
         CMP    #10.,RANPCT
         BGE    PRTCON
         MOV    R0,-(SP)
         MOV    #RANPRT,R1
         MOV    #52.,R0
         MOV    LPCTRL,LPCSAV
         MOV    #100000,LPCTRL      ;USE JUST LP
         JSR    PC,TTYPRT
         MOV    LPCSAV,LPCTRL
         MOV    #200,LPCSAV
         MOV    (SP)+,R0
```

```
        MOV   #RANPRT-3,PRTPNT
PRTCON: CLR   RANPCT
        MOV   (SP)+,R1
        MOV   (SP)+,R2
        RTS   PC
;******TEST
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       CONVERT ASCII TO BINARY     ENTRY VIA JSR
;SOURCE ADDR IN R2+2(FOR DELIM)...PLACE RESULT IN R2
;IF NUMBER INVALID IN ANY WAY ...SET RETURN + 77777
;THIS ROUTINE WILL HANDLE -99 TO (THRU) 199
ATBIN:  MOV   R5,-(SP)
        MOV   R4,-(SP)
        MOV   R1,-(SP)
        MOV   R0,-(SP)
        MOV   #2,R5     ;CONVERT 3 DIGITS(CONTROL WITH 2)
        CLR   R4
        CLR   R0        ;TEMP RESULT HOLD ARE
;WORK RIGHT TO LEFT ...LOOP 3 TIMES CONTROLLED BY R5
        DEC   R2        ;SKIP BACK TO DELIM
ATCYL:  MOVB  -(R2),R1
        DEC   R5        ;IF ZERO POSSIBLE SIGN
        BPL   ATLOW     ;WORK WITH 2 RIGHT HAND DIGITS
;WORKING ON POSSIBLE SIGN '-', '+', ' ', '0', '1'
        CMP   #55,R1    ;'-' SIGN?
        BNE   ATPLUS
        NEG   R0        ;TAKE 2'S COMPLEMENT
        BR    ATEND
ATPLUS: CMP   #53,R1    ;IS IT A PLUS SIGN?
        BEQ   ATEND     ;DISREGARD IT THEN
        CMP   #60,R1    ;ZERO
        BEQ   ATEND
        CMP   #40,R1    ;BLANK
        BEQ   ATEND
        CMP   #61,R1    ;'1'
        BNE   ATER      ;ERROR
        ADD   #144,R0
        BR    ATEND     ;END UP
ATLOW:  CMP   #40,R1    ;BLANK ??? NONEXISTENT
        BEQ   ATCYL     ;GET NEXT
        SUB   #60,R1    ;LOOK FOR NUMBER
        BEQ   ATCYL     ;IF ZERO, FORGET IT
        BLO   ATER      ;ERROR NOT A NUMBER
        CMP   #11,R1    ;COMPARE TO ASCII '9'
        BLT   ATER      ;ERROR
```

```
          TST    R5          ;* THEN RIGHTMOST DIGIT(1'S)
          BGT    ATAD        ;JUST ADD TO RESULT
          ASL    R1          ;MIDDLE DIGIT * 12 (BASE 8)
          ADD    R1,R0       ;1010 BINARY
          ASL    R1
          ASL    R1
ATAD:     ADD    R1,R0       ;ADD UNITS OR LAST 10'S DIGIT
          BR     ATCYL       ;GET NEXT DIGIT
ATER:     MOV    #77777,R0   ;TO MARK AS INVALID
ATEND:    MOV    R0,R2
          MOV    (SP)+,R0
          MOV    (SP)+,R1
          MOV    (SP)+,R4
          MOV    (SP)+,R5
          RTS    PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         OCTAL (BINARY) TO ASCII     VIS JSR
;R2 = N THE BINARY NUMBER TO CONVERT
;MOVE TO POSITION INDICATED BY R1...NORMALLY INLINE
;USE WORKING REG     R3,R4
OASTRT:   MOV    R3,-(SP)
          MOV    R4,-(SP) ;USED AS COUNTER & FOR DIVID
          TST    R2
          BGE    OAPLS       ;ANSWER IS NOT NEG
          MOVB   #055,(R1)+      ;PREP TO PRINT '-' SIGN
          NEG    R2          ;REVERSE...IF NEG LOWEST =12
          BR     OA10        ;WORK ON SECOND DIGIT
OAPLS:    MOV    #100.,R4 ;PREP TO DIV BY 100.
OADIV:    MOV    R4,R3       ;SET UP FOR DIV SUBROUTINE
          JSR    PC,DIVIDE;MUST USE R3,R4...R3 DESTROYED
OALST:    ADD    #60,R3      ;CONVERT TO ASCII CHAR
          MOVB   R3,(R1)+ ;INSERT INTO LINE
          CMP    #10.,R4  ;DECIDE WHICH POSITION WORKING
          BEQ    OA1         ;NEXT IS LAST DIGIT
OA10:     MOV    #10.,R4  ;SECOND DIGIT
          BR     OADIV
OA1:      ADD    #60,R2   ;LAST DIGIT IS REMAINDER LAST DIV
          MOVB   R2,(R1)+ ;ASSUMES NUMBER -99<X<199
          MOV    (SP)+,R4
          MOV    (SP)+,R3
          RTS    PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
```

```
;
;         MULTIPLY SUBROUTINE       VIA JSR
;MAXIMUM HANDLED 15 BIT RESULT W/ SIGN
;NO CHECK MADE ON RESULT BECAUSE GIVEN ONLY -12 THRU +115
;R2 * R3...RESULT IN R2 AT END.
MULT:   MOV   R0,-(SP)
        CLR   R0
        TST   R2          ;IF = 0 NO MULT NEEDED
        BEQ   MPYEND
MPYCNT: TST   R3
        BGT   MPYSFT      ;STILL MORE LEFT
        BEQ   MPYEND      ;END OF MULT (FIRST PASS R3=0,R0=0)
        NEG   R2          ;ONLY POSSIBLE AT START OF MULT
        NEG   R3          ;MAKE R3   & R2 REVERSED
MPYSFT: ASR   R3          ;GET LOW BIT
        BCC   MPYNO       ;NO MULT BIT = 0
        ADD   R2,R0       ;INCREMENT RESULT
;CHECK FOR INVALID RESULT WOULD GO HERE IF NEEDED
MPYNO:  ASL   R2          ;SHIFT FOR NEXT BIT MULT
        BR    MPYCNT      ;GET NEXT BIT
MPYEND: MOV   R0,R2       ;RESULT IN R2
        MOV   (SP)+,R0
        RTS   PC          ;RESULT IN R2...R3 LOST
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         DIVIDE SUBROUTINE         VIA JSR
;HANDLES 2-15 BIT NUMBERS    USES INTEGER ARITHMETIC
;INPUT R2 / R3
;OUTPUT R2 = REMAINDER...R3 = DIVIDEND
;R0= WORKING DIVIDEND...R4= SHIFT COUNTER..
DIVIDE: MOV   R0,-(SP) ;R5= SIGN OF RESULT
        CLR   R0          ;SET TEMP RESULT = 0
        TST   R2          ;IF = 0 RESULT = 0
        BEQ   DIVE        ;QUIT
        BGT   DIVR3T      ;IF R2 <0, MAKE R3 HAVE RESULT SIGN
        NEG   R2
        NEG   R3          ;COMPLEMENT R2 AND R3
DIVR3T: MOV   R4,-(SP)    ;SAVE IT
        MOV   R5,-(SP)
        CLR   R4          ;ZERO STEP COUNTER
        CLR   R5          ;ASSUME SIGN POSITIVE NOW
        TST   R3          ;IS IT PLUS
        BEQ   DIVER       ;ILLEGAL,HALT ..........
        BGT   DIVSFT      ;START ACTUAL DIVIDING
        NEG   R3          ;MAKE IT POSITIVE
        DEC   R5          ;RESULT & REMAINDER NEG
DIVSFT: CMP   R2,R3       ;SHIFT PREP UNTIL R2 < R3
        BLE   DIVOK       ;START AT LEFT END FOR DIVISION
        INC   R4          ;COUNT NUMBER OF SHIFTS
        ASL   R3          ;SHIFT DIVIDEND
```

```
        BVC     DIVSFT      ;DO NOT SHIFT INTO OVERFLOW
        ROR     R3          ;ELIM NEG RESULT OF SHIFTING
        DEC     R4          ;TOO MANY SHIFTS
DIVOK:  ASL     R0          ;SHIFT RESULT TO LEFT
        SUB     R3,R2       ;DECR R2 BY R3 SHIFT R3
        BPL     DIVYES      ;+ OR 0
        ADD     R3,R2       ;SHOULD NOT HAVE SUB
        BR      DIVNXT
DIVYES: INC     R0          ;INCR RESULT
DIVNXT: ASR     R3
        DEC     R4          ;DID ONE MORE SHIFT
        BPL     DIVOK       ;BUT MORE TO GO
DIVEND: TST     R5          ;SIGN OF RESULT & REMAINDER
        BPL     DIVRPL      ;RESULT POSITIVE
        NEG     R0          ;MAKE RESULT NEG
        NEG     R2          ;MAKE REMAINDER NEG
DIVRPL: MOV     (SP)+,R5
        MOV     (SP)+,R4
DIVE:   MOV     R0,R3       ;RESULT IN R3...R2 REMAINDER
        MOV     (SP)+,R0
        RTS     PC
DIVER:  HALT                ;COULD USE MESSAGE LATER
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       TTY CONTROL ROUTINE
;THE INTERRUPT REQUEST CHARACTER IS THE EXCLAMATION POINT
;ALL OTHERS DURING THE RUNNING ARE REJECTED...
;EXCLAMATION POINT STOPS THE RUNNING AT START NEXT EPOCH
;TTY PRINTS EXCLAMATION POINT TO NOTE REQUEST...CHANGES
;RUNCNT # 0 TO CAUSE STOP...'STOP' WILL BE PRINTED
;TTYINT HAS CONTROL DURING TTY INTERRUPT...
;COMMANDS ARE LOOKED UP IN TABLE OF BRANCH ADDR
;DELIM (CR,',','*',';') CAUSES RETURN TO PROGRAM
;LOOK BACKWARD FROM THE DELIM FOR COMMAND/NUMBER
;(EXAMPLE IF 'ABCD RUN*' IS INPUT COMMAND IS ' RUN')
;IF A NUMBER IS EXPECTED IT IS ONLY 3 DIGITS
;(IF '123456789' IS INPUT, THEN ONLY  '789' IS THE NUMBER)
TTYINT: MOV     R0,-(SP)
        MOV     TKB,R0
        BIC     #200,R0
        CMPB    R0,#041     ;CHECK IF VALID & CLR BUFFER
        BNE     TTYRTI      ;INVALID INTERUPT CHARACTER
        CLR     RUNCNT      ;STOP AT NEXT CHANCE
        MOV     #041,TPB    ;WRITE EXCLAMATION POINT
TTYRTI: MOV     (SP)+,R0
        RTI
VALINT: CLR     TSR         ;CANCEL INTERRUPT FROM TTY
        MOV     #STOP+13,R1 ;LOAD ADDR OF STOP EPOCH COUNT
        MOV     EPOCH,R2    ;COUNT OF EPOCH SINCE ZERO
        JSR     PC,OASTRT   ;INSERT COUNT
```

```
          MOV    #STOP,R1
          MOV    #14.,R0   ;MESSAGE LENGTH
          JSR    PC,TTYPRT;PRINT
NEWCMD:   MOV    #CRLF,R1  ;'CR/LF?'
          MOV    #3,R0     ;LENGTH
          JSR    PC,TTYPRT;PRINT
          JSR    PC,TTYRDR;GET COMMAND
          MOV    #COMTAB,R0;SEARCH COMMAND TABLE
;BR     SEARCH     LATER USE ROUTINE AS SEARCH IN A DUMP TABLE
;         SEARCH BRANCH TABLES
;TABLE HAS 'WORD' FOLLOWED BY ADDR OF THAT ROUTINE
;R0 POINTS TO TABLE TO SEARCH...
;(SO THAT LATER MODIFICATION MAY PERMIT DIFFERENT DUMPS)
;R1 IS USED  TEMP (SAVE & RESTORE)
;R2 POINTS TO 2 CHAR BEYOND COMMAND WORD
;IF NOT FOUND JUST GO BACK AFTER A NEW COMMMAND
SEARCH:   MOV    #4,R1     ;COUNTER FOR 4 LETTER COMPARISON
          SUB    R1,R2     ;SET POINTER TO FIRST CHAR
          DEC    R2        ;BACK UP FOR DELIM
          TST    (R0)      ;AT END OF TABLE
;END OF TABLE IS MARKED BY HAVING THE BYTE AT END = 0
          BEQ    SERBAD    ;NOT FOUND IN TABLE
SERCMP:   CMPB   (R0)+,(R2)+   ;SAME?...GO TO NEXT CHAR
          BEQ    SEROK     ;TRY NEXT CHAR THESE MATCHED
          ADD    R1,R2     ;ADVANCE IN INPUT
          ADD    R1,R0     ;SET TABLE POINTER TO NEXT ENTRY
          INC    R0        ;TABLE ENTRY IS 6 CHAR LONG
;2 + NUMBER;ADVANCE TO NEXT ENTRY
          BR     SEARCH    ;KEEP SEARCHING
SEROK:    DEC    R1        ;MATCHED ...MORE TO GO???
          BGT    SERCMP    ;STILL GOT MORE TO GO
          JMP    @(R0)+    ;BRANCH ADDR IN TABLE
SERBAD:   JSR    PC,INVALI;PRINT THAT COMMAND BAD
          BR     NEWCMD    ;GET NEW COMMAND
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;
;         TTY READER ROUTINE        VIA JSR
;R0 = LENGTH OF MESSAGE FOR OUTPUT
;R1 = START ADDR OF OUTPUT MESSAGE
;R2 = NEXT INPUT CHARACTER LOCATION
;
;TTY READ INPUT CHAR INTO (R2) ADDR START AT INLINE
TTYRDR:   MOV    R5,-(SP)
          MOV    R1,-(SP)
TTYRD1:   MOV    #INLINE,R2    ;SETUP INPUT BUFFER
```

```
TTYR1:  TSTB   TSR             ;WAIT UNTIL INPUT A CHAR
        BPL    TTYR1           ;NOT YET
        MOV    TKB,R5          ;MOVE INTO WORKING REG
        BIC    #177600,R5      ;CLEAR TOP 9 BITS
        MOV    #1,R0           ;EXPECT 1 CHAR FOR OUTPUT
        MOV    R2,R1
        CMP    #177,R5         ;IS IT A RUBOUT CHAR
        BNE    NRUB
        CMP    R2,#INLINE      ;DO NOT RUBOUT TOO MUCH
        BLOS   TTYRD1          ;RESTART LINE
        DEC    R2              ;BACKUP
        MOV    #RUBOUT,R1      ;USE A BACKWARD SLASH
        BR     TPRT            ;PRINT
NRUB:   MOVB   R5,(R2)+        ;INSERT CHAR
        CMP    #015,R5         ;IS IT CR,..END OF LINE
        BNE    NCR             ;NOT CARRIAGE RETURN
        INC    R0              ;2 CHAR,..PRINT CR/LF
        MOV    #CRLF,R1        ;ISSUE CR/LF
        BR     RCOM
;PRINT CHAR AS READ IF NOT TOO MANY CHAR IN LINE
NCR:    CMP    R2,#INLINE+70;DO NOT GO TOO FAR
        BLOS   TACPT           ;ACCEPT
        DEC    R2              ;FORGET THIS CHAR
        BR     TTYR1           ;TRY FOR CR OR RUBOUT
;IS IT A DELIM,..IF IS RETURN TO CALL ROUTINE
TACPT:  CMP    R5,#075         ;'=' ?
        BEQ    RCOM
        CMP    R5,#054         ;',' ?
        BEQ    RCOM
        CMP    R5,#073         ;';'?
        BEQ    RCOM
;IF CHAR IS A SLASH, TREAT ENTIRE LINE AS COMMENT
;AND ADVANCE TO NEXT LINE
        CMP    R5,#'/          ;SLASH
        BEQ    TPER
TPRT:   JSR    PC,TTYPRT       ;PRINT CHAR
        BR     TTYR1           ;GET NEXT CHAR
TPER:   MOV    #3,R0           ;PRINT '/' 'CR/LF'
        MOVB   #15,(R2)+       ;INSERT CR
        MOVB   #12,(R2)+       ;INSERT LF
        SUB    #3,R2
        MOV    R2,R1
        JSR    PC,TTYPRT
        BR     TTYRD1
RCOM:   JSR    PC,TTYPRT       ;PRINT
        MOV    (SP)+,R1
        MOV    (SP)+,R5
        RTS    PC
;
```

```
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         INVALID   VIA JSR
;ENTRY FOR COMMAND OR NUMBER OR (WHATEVER) WAS WRONG
;'GOOF ' INDICATES IT WAS REJECTED...TRY AGAIN
INVALID: MOV    R1,-(SP)
         MOV    R0,-(SP)
         MOV    #GOOF,R1
         MOV    #5,R0
         JSR    PC,TTYPRT
         MOV    (SP)+,R0
         MOV    (SP)+,R1
         RTS    PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         PRINT ROUTINE              VIA JSR
;PRINT R0 NUMBER OF CHARACTERS....STARTING AT R1 ADDRESS
;LPCTRL...10000000,10000000...HI IND LP,LO IND TTY
TTYPRT:  TSTB   LPCTRL    ;TTY TO BE USED?
         BPL    TTYLPT    ;DO NOT USE TTY
TTYTTY:  TSTB   TPS       ;TEST IF TTY READY
         BPL    TTYTTY    ;GOT TO WAIT
         MOVB   (R1),TPB  ;MOVE TO OUTPUT
TTYLPT:  TST    LPCTRL    ;USE LINE PRINTER
         BPL    TTYREC
TTYLP:   TST    LPS       ;ERROR???
         BGE    TTYLPR
         HALT              ;FIX IT
         BR     TTYLP     ;TRY AGAIN
TTYLPR:  TSTB   LPS       ;READY???
         BPL    TTYLPR
         MOVB   (R1),LPB  ;LOAD BUFFER
TTYREC:  INC    R1        ;ADVANCE TO NEXT BYTE
         DEC    R0        ;REDUCE COUNTER
         BGT    TTYPRT    ;MORE CHAR TO PRINT
         RTS    PC        ;RETURN
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         CELL  JSR VIA COMMAND TABLE
;A/C/D/E   ADD/CHANGE/DELETE/END  CELL INST
CELL:    CLR    R2        ;START TABLE OF INST
```

```
        MOV   #2,R0     ;CR/LF ON TTY
        MOV   #CRLF,R1  ;CR/LF
        JSR   PC,TTYPRT
        JSR   PC,GTZXXX;GET ZXXX (Z= ALPHA, XXX= NUMB
        MOV   #INSTAB,R1      ;START OF INSTRUCTION TABLE
        CMP   R0,#105   ;'E' END OF INST
        BNE   CNSORT    ;NEW COMMAND AFTER SORT INST IN SEQ
        JMP   CSORT
CNSORT: TST   R2        ;R2 = XXX (BINARY...R0 = Z (CHAR)
        BLE   CIGOOF
        CMP   R2,#199.  ;0<R2<=199.
        BLE   CIOK      ;CELL INST # OK
CIGOOF: JSR   PC,INVALID     ;GOOF
        BR    CELL      ;TRY AGAIN
CIOK:   CMP   R0,#101   ;IF 'A' GO TO ADD ROUTINE
        BEQ   CIADD
        CMP   R0,#103   ;'C'
        BEQ   CICD
        CMP   R0,#104   ;'D'
        BNE   CIGOOF    ;NOT LEGAL
CICD:   JSR   PC,FIND
        TST   R1        ;FOUND IT???
        BNE   CICDFD
        MOV   #NFOUND,R1
        MOV   #9.,R0
        JSR   PC,TTYPRT
        BR    CELL
CICDFD: CMP   R0,#104   ;DELETE STOPS HERE
        BNE   CCCD
        CLR   (R1)      ;ELIM THIS INST #
        BR    CELL
CIADD:  MOV   R2,-(SP) ;SAVE
        JSR   PC,FIND  ;DOES IT ALREAY EXIST???
        TST   R1        ;IF = 0 NOT THERE PROCEED
        BEQ   CIADNF    ;ELSE WRONG
        MOV   #ALLGOT,R1    ;ERROR ALREADY GOT THAT INST
        MOV   #14.,R0
        JSR   PC,TTYPRT
        TST   (SP)+
        BR    CELL
CIADNF: MOV   #-2,R2    ;TO REQUEST FREE SPACE
        JSR   PC,FIND
        MOV   (SP)+,R2 ;GET INST NUMBER
        TST   R1        ;IF =0 NO SPACE AVAIL
        BNE   CCCD
        MOV   #NOSPAC,R1
        MOV   #8.,R0
        JSR   PC,TTYPRT
        BR    CELL
```

```
CCCD:    MOV   R2,(R1)+ ;SET INST #,CLR NEXT#
;INITIALIZED WITH NO TEST/NO RESET (ALL BITS CLEAR)
         CLR   (R1)     ;CLEAR CODE FOR TEST/RESET
;SET FLAG TEST CHARS
CFLAG:   JSR   PC,TTYRDR;GET CHAR
         MOVB  -2(R2),R2;GET CHAR INPUT
         CMP   R2,#123  ;TEST SET???
         BNE   CFLGD
         BIS   #140000,(R1)+ ;TEST IF SET
         BR    OLTEST
CFLGD:   CMP   R2,#103  ;TEST IF CLEAR? ? ?
         BNE   CFLGN
         BIS   #040000,(R1)+
         BR    OLTEST
CFLGN:   CMP   R2,#116  ;NO TEST
         BNE   CFLGGF   ;GOOF
         TST   (R1)+    ;ADVANCE R1
         BR    OLTEST
CFLGGF:  JSR   PC,INVALID;GOOF
         BR    CFLAG    ;TRY AGAIN
;R1 = OLD LIFE LOCATION
;LOAD STACK (0,0),(L,H),(PC,PC)
;RETURN STK (LT,HT),(0,Z)
;GIVE LIMITS OF TEST LOW & HIGH IN NEXT STACK ENTRY.
;-12,=-014 =  -(00 001 100) = 11 110 100
;115,= 163=                              01 110 011
;WORD =   172163 =(-12,115)=  1 111 010 001 110 011
OLTEST:  CLR   -(SP)       ;PLACE TO RETURN LOW & HIGH
         MOV   #172163,-(SP)
;LIMITS ON STACK...RETURN CHAR IN THIS STACK POSTION
         JSR   PC,LIMIT    ;GET ZXXX ,XXX  (LIMITED)
         CMP   (SP)+,#116  ;'N' ONLY POSSIBLE HERE
         BEQ   OLTSTS      ;NO CHANGE
         BIS   #020000,-2(R1)   ;TEST OLD LIFE SET
         JSR   PC,CHGBAS   ;CHANGE TO +12 BASE
OLTSTS:  MOV   (SP)+,(R1)+ ;LOAD LIFE TEST & ADVANCE SP/R1
;NEW LIFE TEST
NLTEST:  CLR   -(SP)       ;SET UP STACK
         MOV   #172163,-(SP)    ;LIMITS (-12,115)
         JSR   PC,LIMIT    ;GET (Z)XXX,XXX
         CMP   (SP)+,#116  ;'N' ONLY
         BEQ   NLTSTS
         BIS   #010000,-4(R1)   ;SET NEW LIFE TEST BIT
         JSR   PC,CHGBAS   ;CHANGE TO +12 BASE
NLTSTS:  MOV   (SP)+,(R1)+ ;SET NEW LIFE TEST
;FUEL TESTS
FTEST:   CLR   -(SP)       ;STACK SET UP
         MOV   #000144,-(SP)  ;(0,100)
         JSR   PC,LIMIT
```

```
        CMP    (SP)+,#116  ;'N'???
        BEQ    FTSTS
        BIS    #004000,-6(R1);SET TEST FUEL BIT
FTSTS:  MOV    (SP)+,(R1)+   ;MOV FUEL TEST
;'IF' CONDITIONS NOW SET... GET 'THEN' CONDITIONS
SETFGG: JSR    PC,TTYRDR
        MOVB   -2(R2),R2   ;CHAR READ
        CMP    R2,#123     ;RESET 'SET'
        BNE    SETFGD
        BIS    #000300,-8.(R1)
        BR     SETFLG
SETFGD: CMP    R2,#103     ;RESET TO CLEAR
        BNE    SETFGN
        BIS    #100,-8.(R1);RESET TO 'CLEAR'
        BR     SETFLG
SETFGN: CMP    R2,#116     ;NO CHANGE
        BEQ    SETFLG
        JSR    PC,INVALID  ;GOOF
        BR     SETFGG      ;TRY AGAIN
SETFLG: CLR    (R1)        ;(CLEAR LIFE & FUEL SET )
;GET THEN SET LIFE AND FUEL SETS
SETLIF: JSR    PC,GTZXXX
        BIC    #70,-8.(R1) ;CLEAR POSSIBLE ERROR INST
        CMP    R0,#116     ;NO CHANGE  'N'
        BEQ    SETFUL
        CMP    R0,#103     ;RESET TO CALC VALUE  'C'
        BNE    SETLFR
        BIS    #020,-8.(R1);SET BITS TO USE CALC VALUE
        BR     SETLFK
SETLFR: CMP    R0,#122     ;RESET TO FIXED VALUE XXX  'R'
        BNE    SETLFE
        BIS    #40,-8.(R1) ;MARK FOR RESET
        BR     SETLR
SETLFE: JSR    PC,INVALID  ;GOOF
        BR     SETLIF
SETLR:  CMP    R2,#-12.
        BLT    SETLFE      ;TOO SMALL
        CMP    R2,#115.
        BGT    SETLFE
        ADD    #12.,R2
        SWAB   R2          ;GET INTO LEFT BYTE
SETLFK: BIS    R2,(R1)     ;SET LIFE VALUE
;SET FUEL

SETFUL: JSR    PC,GTZXXX
        BIC    #3,-8.(R1)  ;CLEAR MARKS
        CMP    R0,#116     ;'N' NO CHANGE
        BEQ    SNINST
        TST    R2          ;0<=R2<=100
```

```
          BLT   SETFER
          CMP   R2,#100.
          BLE   SETFCT
SETFER:   JSR   PC,INVALID
          BR    SETFUL
SETFCT:   CMP   R0,#103       ;CONSUME FIXED AMOUNT
          BNE   SETFPT
          BIS   #001,-8.(R1)
          BR    SETFOK
SETFPT:   CMP   R0,#120       ;PERCENT CONSUME
          BNE   SETFRT
          BIS   #002,-8.(R1)
          BR    SETFOK
SETFRT:   CMP   R0,#122       ;RESET TO XXX
          BNE   SETFER        ;ERROR
          BIS   #003,-8.(R1)
SETFOK:   BIS   R2,(R1)       ;SET NEW FUEL
SNINST:   JSR   PC,GTZXXX     ;GET INST NUMBER
          CMP   R0,#107       ;'G'...GOTO XXX
          BNE   SETIQT
          TST   R2
          BLE   SNIER         ;0< R2 <=199
          CMP   R2,#199.
          BLE   SETGIN
SNIER:    JSR   PC,INVALID
          BR    SNINST
SETIQT:   CMP   R0,#124       ;TERM???
          BNE   SETINT
          CLR   R2
          BR    SETGIN
SETINT:   CMP   R0,#116       ;NEXT INST
          BNE   SNIER         ;ERROR
          MOV   #377,R2       ;PREP FOR SWAB(-1 TOP BITS)
SETGIN:   SWAB  R2            ;MOVE TO HIGH END
          BIS   R2,-10.(R1)   ;NEXT INST # SET
          JMP   CELL          ;GET NEXT CELL INST
;
;
;CSORT...SORT CELL INST...PLACE ZERO INST NUMBER AT TOP
CSORT:    MOV   #INSTAB,R0 ;START OF TABLE
CSORTX:   MOV   R0,R1       ;R0 LOWER ...R1 IS UPPER
          ADD   #12.,R1     ;ADVANCE ONE INST
CSORTN:   TSTB  (R1)         ;IF R1 INST # = 0 NO NEED TO SWAP
          BLE   CSORTC      ;NO SWAP
CSORTA:   TSTB  (R0)        ;IF 0 MOVE UP
          BEQ   CSWAPG
          CMPB  (R0),(R1)   ;COMP INST #'S
          BLE   CSORTC      ;NO SWAP...CONT SORTING
CSWAPG:   MOV   #6.,R2      ;SWAP 6 WORDS IN EACH INST
```

```
CSWAPC: MOV    (R0),R3      ;MOVE WORD TO HOLD
        MOV    (R1),(R0)+   ;MOVE AND ADVANCE R0
        MOV    R3,(R1)+
        DEC    R2           ;DEC WORD COUNTER
        BGT    CSWAPC       ;CONT SWAPPING
        SUB    #12.,R0      ;BACK TO OLD INST # POSTION
        BR     CTSTR1       ;CHECK R1
CSORTC: ADD    #12.,R1      ;ADVANCE NON-SWAPPED
CTSTR1: CMP    R1,#EITAB    ;AT END OF INST TABLE
        BLT    CSORTN       ;SORT NEXT MEMBERS
        ADD    #12.,R0      ;ADVANCE TO NEXT LOWER MEMBER
        CMP    R0,#EITAB-12.     ;QUIT SORTING???
        BLT    CSORTX       ;MORE TO GO
        JMP    NEWCMD       ;GOT NEW COMMAND
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       CHGBAS   CHANGE TO +12 BASE
;USED IN THE CELL ROUTINE...CONVERT THE NUMBER IN THE
;STACK (IN BYTES) INTO LIFE STORAGE METHOD
CHGBAS: MOVB   2(SP),R0     ;GET THE LOW LIMIT
        ADD    #12.,R0      ;CONVERT WITH +12
        MOVB   R0,2(SP)     ;BACK ON STACK
        MOVB   3(SP),R0
        ADD    #12.,R0
        MOVB   R0,3(SP)
        RTS    PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       LIMIT            JSR FOR LIFE & FUEL
;STACK  SP+5    SP+4     SP+3     SP+2      SP+1
;INPUT  0        0       LOW TEST HIGH TEST PC
;OUTPUT LOW      HIGH    0        CHAR      X
;IF EITHER ENTERED VALUE IS BAD,BOTH ARE REJECTED
LIMIT:  JSR    PC,GTZXXX
        MOV    R0,-(SP)     ;CHAR TO STACK TEMP
        CMP    R0,#116      ;'N' ???
        BEQ    LIMEND       ;NO MORE TO DO
        CMPB   R2,5(SP)     ;TOO LOW???
        BLT    LMLOGF
        CMPB   R2,4(SP)     ;TOO HIGH
        BLE    LMGTHI       ;GET HIGH VALUE
LMLOGF: JSR    PC,INVALID
        TST    (SP)+        ;POP CHAR FROM STACK
```

```
        BR      LIMIT
LMGTHI: MOVB    R2,7(SP)        ;INSERT LOW
        JSR     PC,GTZXXX       ;GET HIGH
        CMPB    R2,5(SP)        ;TOO LOW
        BLT     LMLOGF          ;GOOF SAME LINE
        CMPB    R2,4(SP)
        BGT     LMLOGF          ;BAD
        MOVB    R2,6(SP)
LIMEND: MOV     (SP)+,2(SP)     ;MOVE CHAR TO STACK
        RTS     PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       FIND            JSR FROM CELL/RUN
;INPUT  R1=LOC OF POINTER    R2=INST#,0=GET FIRST INST
; -1=GET NEXT INST IN SEQ, -2= GET FREE SPACE
;OUTPUT...R1=LOC POINTER(0= NOT FOUND/NO SPACE)
; R2=INST# (+OR 0)
FIND:   TST     R2              ;WHAT TYPE CALL???
        BGT     FINDPL          ;GET SPECIFIC INST #
        BMI     FINDMI
        MOV     #INSTAB,R1      ;START TABLE INST #
        BR      FINLIN          ;LOAD INST #/LEAVE
FINDMI: INC     R2
        BMI     FINDM2          ;=2???
        ADD     #12.,R1         ;GET NEXT INST
        CMP     #EITAB,R1       ;LEGAL???
        BHI     FINLIN          ;LOAD#/LEAVE
        CLR     R2              ;NO INST #
        CLR     R1              ;ADDR
        BR      FINEND          ;LEAVE
FINDM2: CLR     R2              ;GET FREE SPACE(#0)
        MOV     #INSTAB,R1      ;RESTART IN TABLE
FINDPL: CMPB    (R1),R2
        BEQ     FINEND
        ADD     #12.,R1
        CMP     #EITAB,R1       ;END OF TABLE???
        BHI     FINDPL          ;TRY AGAIN
        CLR     R1              ;NOT FOUND INST
        BR      FINEND
FINLIN: MOVB    (R1),R2         ;GET INST FROM TABLE
FINEND: RTS     PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
```

```
;         WALK  JMP VIA COMMAND TABLE
;'WALK≠<SXXX,><E OR M OR R OR (BLANK)>XXX;'
;'S' FOR SEED TO START RANDOM GENERATOR
;OPTIONAL RANDOM  NUMBER CHANGE...EACH
;R= RULE...M= MAIN CELL...E (OR BLANK)= EPOCH
WALK:     JSR    PC,GTZXXX   ;GET ZXXX
          TST    R2          ;0<=XXX<=100
          BMI    WALKGF
          CMP    R2,#100.
          BLE    WALKOK
WALKGF:   JSR    PC,INVALID
          BR     WALK
WALKOK:   MOV    R2,HWALK
WALKS:    CMP    R0,#123     ;'S' SET EED?
          BNE    WALKM       ;NO SKIP
          MOV    R2,WLKRAN
          BR     WALK        ;GET MAX ERROR
WALKM:    CLR    CWALK       ;DEFAULT...0= CHANGE EVERY EPOCH
          CMP    R0,#115     ;'M' EACH CELL IN MAIN
          BNE    WALKR
          INC    CWALK       ;+1= M
          BR     WALKE
WALKR:    CMP    R0,#122     ;'R'EACH RULE CALC
          BNE    WALKE
          DEC    CWALK       ;-1=R
WALKE:    MOV    #1,WLKMUL   ;ASSUME WALK=0
;WLKMUL IS THE NUMBER ACTUALLY MULT TIMES THE CELLS
;USING 1 WILL CAUSE QUICK RETURN FROM MULT IF
;WALK=0...THIS WILL CAUSE NO CHANGE IN THE VALUE
;THE DIVISION LATER WILL USE ONLY 100. RATHER THAN 10000.
;MUST NOW CREATE THE TRUNCATION MASK
;IF WALK=5=...000101   THEN MASK =1...111000
          MOV    #-1,R3      ;SET ALL BITS ON
WALK3:    BIT    R3,R2       ;SHIFT TIL NO BIT COMMON W/ R2
          BEQ    WALKEN
          ASL    R3          ;SHIFT LEFT
          BR     WALK3
WALKEN:   MOV    R3,MWALK    ;MASK FOR TRUNCATION
          CLR    CON         ;SETUP CONSTANT START
          JMP    NEWCMD
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         DUMP  VIA COMMAND TABLE
;PRINTS....ROWXXX=SXXX,XXX CXXX,XXX,...
DUMP:     TST    LPCTRL   ;PRT JUST TO LP IF POSSIBLE
          BPL    DUMNLP
```

```
          MOV   LPCTRL,LPCSAV
          MOV   #100000,LPCTRL      ;USE JUST LP
DUMNLP:   MOV   #1,RM      ;ROW BEING PRT
          MOV   HSIZE,RMW   ;ROW COUNTER ONLY
DUMNRW:   MOV   #MAIN-2,R5  ;LOC OF FIRST CELL BACK1
          ADD   RMW,R5      ;(MAIN-2)+2(ROW)
          ADD   RMW,R5
          MOV   HSIZE,CM ;COL COUNTER
          MOV   #WROW+6,R1
          MOV   RM,R2       ;PRINT 'CR/LFROW=XXX'
          JSR   PC,OASTRT
          MOV   #9.,R0
          MOV   #WROW,R1
          JSR   PC,TTYPRT
          MOV   #5,CMW
DUMPRT:   MOV   (R5),R0   ;PRT 'BZLLL,FFF'
          JSR   PC,PRTCEL
          DEC   CM          ;FINISHED THAT COL
          BLE   DUMNXR
          DEC   CMW         ;DONE 5 COL???
          BGT   DUMNXC
          MOV   #CRLF7B,R1    ;NEXT LINE
          MOV   #9.,R0
          JSR   PC,TTYPRT
          MOV   #5,CMW
DUMNXC:   ADD   HSIZE,R5 ;ADV 1 COL
          ADD   HSIZE,R5 ;2 BYTES/CELL
          BR    DUMPRT
DUMNXR:   INC   RM
          DEC   RMW
          BGT   DUMNRW
          MOV   #CRLF,R1 ;LEAVE A BLANK LINE
          MOV   #2,R0
          JSR   PC,TTYPRT
          MOV   LPCSAV,LPCTRL
          MOV   #200,LPCSAV
          JMP   NEWCMD
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         RULE  JMP VIA COMAND TABLE
;NORMAL                    SPECIAL
;RULE=X,X,X,X,X,          RULE=R002=X,X,X,X,X,
;      X,X,X,X,X,               X,X,X,X,X,   ETC
;      X,X,X,X,X,          OR...
;      X,X,X,X,X,          RULE=R002,C001=X,X, ETC
;      X,X,X,X,X,
```

```
;IN BOTH SITUATIONS, THE DATA ENTRY INTO THE ARRAY
;STARTS AT THE INDICATED POSITION(BY DEFAULT
;THE TOP LEFT EDGE OF THE ARRAY, OR THE LEFT OF
;EDGE OF A ROW) AND CONTINUES ACROSS AND DOWN
;UNTIL THE LAST LOWER RIGHT ELEMENT IS ENTERED, OR THE
;OPERATOR ENTERS A 'Q'.
;ROW...=2<+2          COL...=2<+2
;IF Z=R...CHANGE ROW NUMBER...IF C CHANGE COL
RULE:    MOV    #2.,R4     ;SET TO START AT ROW +2
RULE1:   MOV    #-2,R5     ;SET COL
RULE2:   JSR    PC,GTZXXX
         CMP    R0,#040    ;MUST BE NORMAL ELEMENT
         BEQ    RULES
         CMP    R2,#-2     ;CHANGE
         BLT    RULEG
         CMP    R2,#2
         BGT    RULEG
         CMP    R0,#122    ;SET NEW ROW
         BNE    RULEC
         MOV    #-2,R5     ;START AT THE LEFT EDGE(COL=-2)
         MOV    R2,R4
         BR     RULE2
RULEC:   CMP    R0,#103    ;SET IN COL
         BNE    RULEG
         MOV    R2,R5
         BR     RULE2
RULEQ:   CMP    R0,#121    ;QUIT????
         BEQ    RULEND
RULEG:   JSR    PC,INVALID
         BR     RULE2
RULES:   CMP    R2,#77777;OUT OF RANGE
         BEQ    RULEG
         MOV    R5,R0      ;SRULE+2*(12*ROW+(COL*5))
         ASL    R0         ;SRULE+24+2*(ROW+COL*5)
         ASL    R0
         ADD    R5,R0
         ADD    R4,R0
         ASL    R0
         ADD    #SRULE+24.,R0    ;+24 IS THE CENTER(0,0)
         MOV    R2,(R0)
         CMP    R5,#2      ;AT END OF ROW???
         BGE    RULER      ;YES
         INC    R5
         BR     RULE2
RULER:   CMP    R4,#-2     ;AT END OF RULE MATRIX???
         BLE    RULEND     ;YES
RULEX:   DEC    R4         ;ADVANCE TO NEXT ROW
         MOV    #CRLF7B,R1   ;INDIC BY GO TO NEXT PRT LINE
         MOV    #8.,R0     ;JUST USING 5 BLANKS
```

```
          JSR    PC,TTYPRT
          BR     RULE1
RULEND:   JMP    NEWCMD
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         ZERO  JMP VIA COMMAND TABLE
;CLEAR FLAG...ZERO LIFE & FUEL (LIFE= +12)
ZERO:     MOV    #4096.,R0 ;NUMBER WORDS IN TABLE
;12.*14(8)*00,001,100,; 00,000,000,*00600
          MOV    #006000,R1
          MOV    #MAIN,R2
ZERONX:   MOV    R1,(R2)+
          DEC    R0
          BGT    ZERONX
          CLR    EPOCH
          JMP    NEWCMD
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;         SIZE=XXX...     ENTER VIA COMMAND TABLE
;CHANGE SIZE OF WORKING ARE TO DISPLAY
;WHERE XXX IS THE NUMBER OF CELLS ON ONE SIDE
;SIZE SHOULD ONLY BE USED TO START A NEW RUN.....
;DATA IS STORED IN COL ORDER FROM BOTTOM UP...
SIZE:     JSR    PC,TTYRDR    ;GET XXX.
          JSR    PC,ATBIN     ;CONVERT TO BINARY
          CMP    R2,#4.       ;NO SENSE IF SMALLER
          BLT    SZBD
          CMP    R2,#64.      ;MUST BE XXX<= 64.
          BLE    SZOK         ;OKAY
SZBD:     JSR    PC,INVALID   ;'GOOF'
          BR     SIZE         ;GET NEW SIZE
SZOK:     MOV    R2,HSIZE     ;HOLD SIZE
;EACH CELL IN ARRAY HAS 4 DISPLAY POINTS TOTAL....
;THERE ARE 2 DISPLAY COLUMNS REQUIRED FOR EACH CELL
;         .........................
;         .             .         .
;         . DOT 3       .  DOT 1  .
;         .             .         .
;         .........................
;         .             .         .
;         . DOT 4       .  DOT 2  .
;         .             .         .
;         .........................
```

```
;AFTER TWO DISPLAY COLUMNS, A SINGLE WORD ENTRY
;SKIPS THE NEXT X INCREMENT IN THE PRESENTATION
;VERTICAL HAS 1024 POSITIONS BUT APPEARS TO BE 4096
;3 SPOTS PER CELL PER COL (2 DISPLAY, 1 BLANK)
;SETS UP THE DISPLAY TABLE FOR ONE CELL COLUMN ONLY
;WHEN THE DISPLAY IS ABOUT TO START, THE REMAINDER
;MUST BE PREPARED, THIS SAVES SPACE.  ROUTINE ALSO
;CREATES INFO NEEDED LATER (START OF SECOND CELL COL,
;COL IN DISPLAY, NUMBER OF WORDS TO TRANSFER VIA DMA)
        MOV   R2,R3
        MOV   R2,R1         ;SAVE SIZE IN REG FOR LATER
        MOV   #340.,R2      ;1024/3/SIZE=341/SIZE
        JSR   PC,DIVIDE     ;SCALE THE VERTICAL
        CMP   R3,#40        ;DO NOT SPREAD TOO MUCH
;IF EQUAL 200...THEN IT IS NEGATIVE
        BLT   SZDOK         ;KEEP IT SMALL
        MOV   #37,R3        ;CLOSE IT UP
SZDOK:  MOV   R3,DISZ       ;GET X-STEP SIZE
        ASL   DISZ
        ASL   DISZ
;SSSS        MOVB  DISZ,XSTEP
;SET UP FOR DMA MODE 1...(INTENSITY & Y INCREMENT)
;SSSS        CLRB  YSTEP
;SSSS        CLR   XPOS          ;INITIALIZE
;SSSS        CLR   YPOS
        MOV #004000,R2      ;SETUP STEP & X INHIB
        MOV   #DSPTAB,R4    ;START DISPLAY TABLE ADDR
        MOV   R2,(R4)+      ;SET UP DOT 4, TO ZERO
;ADVANCE THROUGH DISPLAY BY DISZ, AND
        ADD   R3,R2         ;1 STEP DOT 3
        MOV   R2,(R4)+      ;SETUP 1ST CELL DOT3
        MOV   R1,R5         ;SAVE SIZE
;SET UP ALL OTHER DISPLAY POINTS IN THE FIRST DISPLAY COL
SZCLST: DEC   R1            ;AT END????
        BEQ   SZLTCL        ;QUIT ALL PROCESSES
        ADD   R3,R2
        ADD   R3,R2
        MOV   R2,(R4)+      ;ADVANCE 2*Y STEP
        ADD   R3,R2
        MOV   R2,(R4)+      ;ADVANCE YSTEP ONLY
        BR    SZCLST        ;CONT CELL SETUP
;ADVANCE X STEP IN LAST POINT(CANCEL INHIB)
SZLTCL: BIC   #04000,-2(R4)   ;ADVANCE X STEP NEXT
;ONE DISPLAY COL HAS BEEN CREATED....DUPLICATE
;TO COMPLETE ONE CELL COLUMN FOR DISPLAY...
        MOV   R5,R1
        MOV   #DSPTAB,R3    ;START OF DSPTAB
SZAGIN: MOV   (R3)+,(R4)+ ;MOVE POINT 4,3,4,3
```

```
          MOV    (R3)+,(R4)+ ;MOVE POINT 2,1,2,1...
          DEC    R1
          BGT    SZAGIN      ;MORE LEFT
          CLR    (R4)+       ;NO DISPLAY,JUST SKIP COL
          MOV    R4,DSPSEC   ;START SECOND DISPLAY SET
;THIS WILL BE DUPLICATED AS MANY TIMES AS POSSIBLE
;NEED WHERE THE SECOND DISPLAY COLUMN STARTS
;(ONLY THE FIRST CELL, DISPLAY COL IS GENERATED NOW, THE
;REMAINDER WILL BE CREATED LATER)
;1024 BYTES IS ENOUGH FOR 2=64 CELL DISPLAY COLUMNS
;(OR ENOUGH ROOM FOR A COMPLETE 11 X 11 DISPLAY)...
;CALC NUMBER OF DISPLAY COL THAT WILL FIT IN 512 WORDS
;EXTRA 32 BYTES PERMITS THE SKIP COL WORD TO BE
;IN THE DATA AREA
          CLR    (R4)+       ;LEAVE SPACE FOR GROUP SEPARATOR
          MOV    R4,WK       ;SET UP POINTER TO WORK SPACE
          SUB    #2,R4       ;CORRECT FOR GROUP SEPARATOR
          MOV    #128.,R2    ;(128/SIZE)*(SIZE*4+1) OR
          MOV    R5,R3       ;(128/SIZE)*(NUMBER WORDS
;REQUIRED TO DISPLAY ONE CELL COLUMN)
          JSR    PC,DIVIDE   ;NUMBER POSSIBLE REPETITIONS
;HOWEVER THERE SHOULD NOT BE MORE REPETITIONS
;THAN THERE ARE REQUIRED...IF SIZE=64 THEN (128./SIZE)*
;2 REP.....BUT IF SIZE =4. THEN REPETITON MUCH MORE THAN
;ACTUALLY NEEDED...
;THERFORE SET REP MAX AT SIZE
          CMP    R3,R5       ;REP : SIZE
          BLE    SZRPOK
          MOV    R5,R3
SZRPOK:   MOV    R3,DSPGRP   ;# OF REPETITIONS
          DEC    R3          ;ALREADY HAVE GROUP
          SUB    #DSPTAB,R4  ;NUMBER BYTES IN DISPLAY COL
          ASR    R4          ;CONVERT TO NUMBER OF WORDS
          MOV    R4,R2
          JSR    PC,MULT
          MOV    R2,NDSPTS   ;NUMBER WORDS TO COPY
          JMP    NEWCMD      ;GET NEW COMMAND
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
; FLAG/LIFE/FUEL   ENTER VIA COMMAND TABLE JUMP
;SET UP LIFE FACTORS IN CELLS VIA LIGHT PEN
;FLAG=(S/C),LIFE=XXX,FUEL=XXX
;THEN USE THE LIGHT PEN TO MARK THE CELLS
;ROUTINE WILL  PRINT OUT THE CELL DATA
;BEFORE THE FACTOR WAS CHANGED (IF ERROR MADE)
LIFE:     MOV    #077400,R5
```

```
        JSR     PC,GTZXXX       ;GET ENTRY
        CMP     R2,#-12.        ;MUST BE -12<= R2 <=115
        BLT     LBAD
        CMP     R2,#115.
        BLE     LFOK
LBAD:   JSR     PC,INVALID      ;GOOF, TRY AGAIN
        BR      LIFE
LFOK:   ADD     #12.,R2         ;SET UP FOR STORAGE AS LF
        SWAB    R2              ;MOVE TO TOP END
        JSR     PC,CHNGCL
        BR      LIFE
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
CHNGCL: MOV     #18.,R0         ;LENGTH OF MESSAGE
        MOV     #ROWPRT,R1      ;MESSAGE INDICATING WHAT WILL
        JSR     PC,TTYPRT       ;BE PRINTED LATER
        BIS     #40,DSPCSR      ;ENABLE LIGHT PEN INTERRUPT
CHWAIT: TSTB    TSR
;WAIT FOR EXCLAMATION POINT OR
;ANY CHARACTER WILL CAUSE ROUTINE TO PERMIT NEW ENTRY
;EXCLAMATION POINT PERMIT NEW COMMAND
        BPL     CHWAIT          ;LPINT CHANGES CELL
;SSSS          BIC   #40,DSPCSR  ;CUT OFF LIGHT PEN INTERRUPT
        MOV     TKB,R0
        BIC     #177600,R0
        CMP     #041,R0         ;EXCLAMATION POINT
        BNE     NEWENT          ;GET NEW ENTRY FROM TTY
        TST     (SP)+
        JMP     NEWCMD          ;GET NEW COMMAND
NEWENT: RTS     PC
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       FLAG        VIA COMMAND TABLE
;ONLY ACCEPT 'C' OR 'S'
FLAG:   MOV     #100000,R5      ;MASK
        JSR     PC,GTZXXX       ;GET CHAR
        CLR     R2
        CMP     R0,#103         ;"C"
        BEQ     FLGOK
        CMP     R0,#123         ;"S"
        BEQ     FLGOKS
        JSR     PC,INVALID
        BR      FLAG            ;TRY AGAIN
```

```
FLGOKS: MOV   R5,R2        ;MARK TO SET FLAG
FLGOK:  JSR   PC,CHNGCL    ;WAIT FOR LIGHT PEN
        BR    FLAG
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       FUEL          VIA COMMMAND TABLE
;'FUEL=XXX'
FUEL:   MOV   #377,R5
        JSR   PC,GTZXXX
        TST   R2            ;VALID NUMBER
        BMI   FULGOF        ;BAD CANNOT BE NEG
        CMP   R2,#100.
        BLE   FULOK
FULGOF: JSR   PC,INVALID
        BR    FUEL
FULOK:  JSR   PC,CHNGCL
        BR    FUEL
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;        DATA   JUMP VIA JUMP TABLE
;USED FOR TESTING PURPOSES WITHOUT CRT DISPLAY DEVICE
;PERMIT INSERT FLAG,LIFE,FUEL INTO SPECIFIED ROW,COL
;NO CHECK IS PERFORMED ..FOR TEST USAGE ONLY
;ENTER ZXXX,XXX=XXX,XXX;XXX,XXX;.....;0
;FLAG(C/S),LIFE,FUEL=ROW,COL;...
;IF ROW=0 GO FOR NEW COMMAND
DATA:   JSR   PC,GTZXXX ;GET FLAG,LIFE
        MOV   R2,R4     ;HOLD
        ADD   #12.,R4   ;CONVERT TO INTERNAL METHOD
        SWAB  R4
        BIC   #100377,R4;LEAVE SPACE FOR FUEL
        CMP   #123,R0   ;'S'
        BNE   DATSET
        BIS   #100000,R4
DATSET: JSR   PC,GTZXXX
        BIC   #177400,R2
        BIS   R2,R4
DATROW: JSR   PC,GTZXXX ;GET ROW
        TST   R2
        BEQ   DATEND    ;ROW=0  GET NEW COMMAND
        MOV   HSIZE,R1
        SUB   R2,R1
        ASL   R1
```

```
        ADD    #MAIN,R1
        JSR    PC,GTZXXX
        SUB    #1,R2       ;CORRECT FOR START IN COL1
        MOV    HSIZE,R3
        ASL    R3
        JSR    PC,MULT
        ADD    R2,R1
        MOV    R4,(R1)     ;SET FLAG,LIFE,FUEL INTO CELL
        BR     DATROW
DATEND: JMP    NEWCMD
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
;       COMMAND TABLE
.NLIST BEX
;4BYTE COMMAND KEY WORD FOLLOWED BY ADDR OF COMMAND
.EVEN
COMTAB: .ASCII /CELL/
        .WORD   CELL
        .ASCII /FLAG/
        .WORD   FLAG
        .ASCII /LIFE/
        .WORD   LIFE
        .ASCII /FUEL/
        .WORD   FUEL
        .ASCII / RUN/
        .WORD    RUN
        .ASCII /DUMP/
        .WORD   DUMP
        .ASCII /WALK/
        .WORD   WALK
        .ASCII /RULE/
        .WORD   RULE
WSIZE:  .ASCII /SIZE/
        .WORD   SIZE
        .ASCII /ZERO/
        .WORD   ZERO
        .ASCII /DATA/
        .WORD   DATA
        .WORD 0             ;END OF TABLE
;DUMPTB: .ASCII  /TAPE/     ;EXAMPLE FOR LATER USE
;       .WORD 0,0
;
;       LITERALS FOR INPUT & OUTPUT
NOSPAC: .ASCII /NO SPACE/
ALLGOT: .ASCII /ALREADY EXISTS/
NFOUND: .ASCII /NOT FOUND/
```

```
STOP:     .ASCII /STOP EPOCH=000/
CRLF:     .BYTE  15,12      ;CR,LF
QUES:     .ASCII /?/
RUBOUT:   .BYTE  134        ;BACKWARD SLASH
GOOF:     .ASCII /GOOF /
WROW:     .BYTE  15,12
          .ASCII /ROW=    /
WLAKFL:   .ASCII /RANDOM FIX/
          .BYTE  15,12
BADINT:   .ASCII /NO INSTRUCTION #XXX/
          .BYTE  15,12
ROWPRT:   .ASCII /INTO@ROW,COL=WAS/  ;PRINT BEFORE CHANGE
          .BYTE  15,12      ;CR/LF
.EVEN
CRLF7B:   .BYTE  15,12
.ASCII /          /;ALSO USED FOR INLINE CUSHION
INLINE:   .       ;WORKLINE FOR INPUT
. = . + 70.
.LIST BEX
;         DATA LOCATIONS
.EVEN
LPCTRL:   200     ;LINE PRINTER CONTROL INFO  L,T(200,200)
LPCSAV:   200     ;SAVE LOCATION WILL IN DUMP
EPOCH:    0       ;NUMBER OF EPOCHS SINCE LAST ZERO
DSPCNT:   0       ;NUMBER OF TIMES TODISPLAY BETWEEN EPOCHS
RUNCNT:   0       ; NUMBER EPOCH STEPS TO BE DONE BEFORE STOP
YMAX:     0       ;MAX DISP Y AXIS
DSHOLD:   0       ;WORKAREA FOR DISPLAY GROUPS
DISZ:     228.    ;DISPLAY STEP SIZE IN Y DIRECTION
;START WITH ASSUMED SIZE 4
HSIZE:     4.     ;SIZE OF ARRAY ...ONE SIDE
NDSPTS:   51.     ;COUNT OF TIMES TO REPEAT COPY WORDS
DSPSEC:   34.+DSPTAB ;START SECOND SECTION DSPTAB
DSPGRP:   4       ;NUMBER OF DISPLAY GROUPS
PNTM:     MAIN    ;POINTER TO CENTRAL ELEMENT  BEING WORKED ON
PNTMW:    MAIN    ;STEPPED THRU MAIN
PNTR:     SRULE   ;STEPPED THRU SRULE
PTWK:     WK      ;STEPPED WK
DROW:     0       ;USED BY DISPLAY
DCOL:     0
LROW:     0       ;USED BY LIGHT PEN INTERRUPT
CM:       0       ;USED BY RUN & RULE
RM:       0
CMW:      0       ;USED BY RUN
RMW:      0
WK:       0       ;LOC OF WORK SPACE FOR CALC SHARED W/ DISPLA
;MAX (5COL*64ROWS)*(2/CELL)=640BYTE
.EVEN
SRULE = .         ;START OF 25 WORDS OF RULES
```

```
. = . +50.
CON:    0       ;CONSTANT ADDED EACH PASS THRU GENERATOR
MWALK:  0       ;TRUNCATION MASK TOC LEAR TOP BITS IN RANDOM
RANSGN: 1       ;SIGN OF LAST NUMBER
WLKRAN: 0       ;LAST RANDOM NUMBER GENERATED
WLKMUL: 1       ;NUMBER TO MULTIPLY IN EQUATIONS (1=WLKRAN)
CWALK:  0       ;CHANGE RANDOM NUMBER
;  0=EVERY EPOCH...+1=EVERY MAIN CELL...-1=EVERY RULE CALC
HWALK:  0       ;HOLD AREA FOR WALK LIMIT
CURLIF: 0       ;HOLD CURRENT LIFE FOR INSTRT
CALLIF: 0       ;HOLD CALCULATED LIFE
FLGFUL: 0       ;HOLD FLAG/FUEL
INSTAB = .      ;10 CELL INSTRUCTIONS ONLY
. = . + 120.
EITAB= .        ;MARKS END OF INST TABLE
DSPTAB = .      ;MAX (2 COL * 64 ROW)
;THE DISPLAY AREA IS USED FOR TWO THINGS
;DISPLAY AND TO START PROGRAM
;THE PROGRAM START IS OVERLAID AS
;DISPLAY IS GENERATED
;
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
START:;$$$$   CLR   R1
;$$$$         MOV   #2,R2           ;GOTO ADDR IN VECTOR TABLE
CLRVEC:;$$$$  MOV   R2,(R1)+
;$$$$         CLR   (R1)+
;$$$$         CMP   (R2)+,(R2)+     ;ADD 4 TO R2
;$$$$         CMP   R1,#500.
;$$$$         BLT   CLRVEC
          CLR   TSR             ;CANCEL TTY INTERRUPT
          MOV   #100,@#177776 ;SET PS
          MOV   #1000.,SP
          MOV   #TTYINT,@#60  ;TTY INT
          MOV   #200,@#62       ;SET PRIORITY =4
          MOV   #DSPLY,@#320
          MOV   #240,@#322      ;PRIORITY =5
          MOV   #LPINT,@#324
          MOV   #340,@#326
          MOV   #INSTAB,R1   ;ELIM OLD INST
ZERINT:   CLR   (R1)          ;ELIM INST NUM
          ADD   #12.,R1
          CMP   R1,#EITAB
          BLO   ZERINT
          MOV   #MESS,R1        ;PRINT START INST
          MOV   #335.,R0        ;LENGTH
          JSR   PC,TTYPRT
```

```
          JSR    PC,TTYRDR      ;GET L OR (CR)
          SUB    #2,R2          ;BACK UP TO CHAR
          MOVB   (R2),R2
          CMP    #114,R2  ;'L'...LINE PRINTER ATTACHED
          BNE    STEND
          BIS    #100000,LPCTRL
STEND:    JMP    NEWCMD   ;START ACCEPTING COMMANDS
.NLIST BEX
MESS:     .BYTE 15,12,12
.ASCII /A SLASH IS USED TO END A COMMENT LINE./
.BYTE  15,12
.ASCII /IF PRINTER AVAILABLE, TYPE 'L'./
.BYTE  15,12
.ASCII /SET SWITCH REGISTER TO THE NUMBER /
.BYTE 15,12
.ASCII /OF DISPLAYS SHOWN BETWEEN EPOCHS(TRY 100(8))./
.BYTE  15,12
.ASCII /NEXT TYPE 'ZERO', 'SIZE=XXX', 'WALK=XXX',/
.BYTE  15,12
.ASCII /'RULE=XXX....', 'CELL=....', ' RUN=000'/
.BYTE 15,12
.ASCII  /TO START DISPLAY./
.ASCII  / THEN TURN UP THE INTENSITY/
.BYTE 15,12
.ASCII /ON THE DISPLAY TO /
.ASCII  /SHOW ALL 4 DOTS PER CELL./
.BYTE 15,12,12
.EVEN
.= DSPTAB + 1056.  ;(4 PTS/CELL* 2 /PT=1024 BYTE
;INCREASED BY 32 FOR SKIP WORD
MAIN =  .; MAX 64*64*2=8192
. = . + 8192.  ;SOME MACHINES MAY BE TOO SMALL
.END   START
```

VITA

Robert Carl Fusarini was born in Washington, D. C. on July 26, 1944. He attended elementary school in Pittsburgh, Pennsylvania and graduated from Madison High School, Madison, Tennessee in 1962. In August 1966, he received a Bachelor of Science degree in Mathematics from Tennessee Technological University and was commissioned a second lieutenant in the United States Army Reserves and served two years in the Air Defense Artillery.

He has been employed by E. I. duPont de Nemours & Company since February 1969, as a computer programmer and investment analyst. He was granted a leave of absence to enter the Graduate School at The University of Tennessee in September 1972, and served as an instructor in the Computer Science Department. He received the Master of Science degree in Computer Science in December 1974.

He is single, a member of the Association of Computing Machinery, and a captain in the Army Reserves.