G A M E P A C 1

USER'S MANUAL

PROCESSOR TECHNOLOGY CORPORATION

7100 Johnson Industrial Drive
Pleasanton, CA 94566

 No. 727007

I M P O R T A N T N O T I C E

This copyrighted software product is distributed on an individual sale basis for the personal use of the original purchaser only. No license is granted herein to copy, duplicate, sell or otherwise distribute to any other person, firm or entity. This software product is copyrighted and all rights are reserved; all forms of the program are copyrighted by Processor Technology Corporation.

T H R E E M O N T H L I M I T E D W A R R A N T Y

Processor Technology Corporation warrants this software product to be free from defects in material and workmanship for a period of three months from the date originally purchased.

This warranty is made in lieu of any other warranty expressed or implied and is limited to repair or replacement, at the option of Processor Technology Corporation, transportation and handling charges excluded.

To obtain service under the terms of this warranty, the defective part must be returned, along with a copy of the original bill of sale, to Processor Technology Corporation within the warranty period.

The warranty herein extends only to the original purchaser and is not assignable or transferable and shall not apply to any software product which has been repaired by anyone other than Processor Technology Corporation or which may have been subject to alterations, misuse, negligence, or accident, or any unit which may have had the name altered, defaced or removed.

G A M E P A C 1

Table of Contents

I. INTRODUCTION

GAMEPAC 1 is a collection of four games designed to run on a Sol or other 8080 based computer with a Processor Technology VDM-1 Video Display Module. GAMEPAC 1 is distributed on cassette requiring a hardware cassette interface such as the Processor Technology CUTS circuit board to read the programs into memory for execution. Although these programs are designed to interface with either SOLOS/CUTER, CONSOL or other user written surrogate, standard input routines are also provided.

All input to the games is via either the SOLOS/CUTER jump table (refer to the interface specification in the appendix) or the standard input routines provided. All output from the games is to the screen--either the Sol display or the VDM-1.

A. Hardware Requirements for All of the Games

1. All of the games require no more than 4K of RAM from location zero through 0FFFH.

2. All games are entered or re-entered at location zero.

3. All games require either the Sol display circuitry or a Processor Technology VDM-1 circuit board. The display scroll port must be either 0FEH or 0C8H.

4. The character generator chip number 6574 is suggested.

5. The video display switches should be set as follows:

	1	2	3	4	5	6
Sol	off	off	off	on	off	on
VDM-1	off	on	on	on	on	off

B. CUTS Cassette Tape Information

The games of GAMEPAC 1 are recorded using the Processor Technology CUTS/Kansas City standard recording format. The tape is loaded using a Sol with SOLOS or CONSOL personality module or a computer running under CUTER and a CUTS (Computer Users Tape Standard) audio cassette board. The SOLOS/CUTER interface specifications in the appendix describe the format of the tape so that a user written routine may be used to read the games into memory from tape.

To load a game from the GAMEPAC 1 tape, rewind the tape, set the tape counter to zero and advance the tape to just ahead of the counter indication number for the game to be loaded. Make sure the tone and volume are adjusted correctly and the necessary cables are connected.

The following examples show the SOLOS/CUTER commands used to load and execute the games.

XEQ (name)cr

Where:	XEQ	is a SOLOS/CUTER command which causes the next (or named) program to be read from tape into memory.
		(name) is the name of the program to be loaded. (name) is optional, and the next program on the tape will be loaded if it is not used.
	cr	This is the carriage return key.

The game, which is a program, will be loaded into memory and run at location zero. It will then display any necessary instructions on the screen.

For example:

XEQ TARG	to play TARGET
XEQ LIFE	to play LIFE
XEQ PTRN	to play PATTERN
XEQ ZING	to play ZING

"GET (name)cr" may also he used to load the tape.

After the program loads and the prompt character reappears on the screen, type "EX 0cr" to execute the program.

If you have any trouble loading the tape, refer to the cassette operating procedures in the Sol manual or the CUTS manual.

II. GAMEPAC 1 INPUT ROUTINES

All of the games use the same standard input routines from location 0-26.

The first time a game is executed, this input routine will be initialized. A description of this input routine and the initialization procedures follow. If the games are used with either SOLOS/CUTER, CONSOL or a compatible surrogate, no modification will be necessary.

A standard input routine will be selected automatically in the event that none of the above routines are used.

An assembly listing of the standard input and initialization routines is on the following pages.

```
0001 *
0002 ************************************************
0003 *                                              *
0004 *          < GAMEPAC 1 INPUT ROUTINES >         *
0005 *                                              *
0006 ************************************************
0007 *
0008 *
0009 *   ALL OF THE GAMEPAC 1 PROGRAMS USE THE
0010 *   SOLOS/CUTER/STANDARD INPUT ROUTINE.
0011 *
0012 *   THE ROUTINE SOURCE CODE/ASSEMBLY IS SHOWN
0013 *   BELOW AS IT APPEARS IN ALL GAMEPAC 1 GAMES.
0014 *
0015 *   THE 'START' VALUE IN EACH PROGRAM WILL BE
0016 *   THE STARTING ADDRESS OF THE ACTUAL GAME,
0017 *   AND WILL BE UNDEFINED IN THE LISTING BELOW.
0018 *
0019 *
0020 *
0021 ************************************************
0022 *                                              *
0023 *  < SOLOS/CUTER AND STANDARD INPUT ROUTINES >  *
0024 *                                              *
0025 *  VERSION 2.4   APRIL 1,1977   S. DOMPIER     *
0026 *                                              *
0027 ************************************************
0028 *
0029 *
0030 *  THIS PROGRAM MAY USE ONE OF THREE POSSIBLE
0031 *  INPUT ROUTINES.  ON ENTRY TO THIS INITIALIZING
0032 *  ROUTINE, THE FIRST TWO BYTES POINTED TO BY
0033 *  REGISTERS HL ARE CHECKED TO DETERMINE IF
0034 *  THE EXECUTING PROGRAM IS 'SOLOS' OR 'CUTER'.
0035 *
0036 *  WHEN A PROGRAM IS CALLED BY THE 'XEQ' OR 'EXEC'
0037 *  COMMAND FROM SOLOS/CUTER, REG HL IS SET TO THE
0038 *  FIRST ADDRESS OF SOLOS/CUTER.  THE FIRST TWO
0039 *  BYTES OF SOLOS = 00 C3; THE FIRST TWO BYTES
0040 *  OF CUTER = 7F C3.  IF THE DATA IN THE FIRST TWO
0041 *  BYTES POINTED TO BE REG HL MATCHES, AND THE LDA
0042 *  INSTRUCTION (3AH) AT SOLOS/CUTER ADDRESS xx1FH
0041 *  ALSO MATCHES THE INPUT ROUTINE ADDRESS OF
0044 *  SOLOS/CUTER IS INSERTED AT 'INADD' WHICH IS THE
0045 *  INPUT ROUTINE CALL ADDRESS.
0046 *
0047 *
0048 *  IF NO MATCH IS MADE, A STANDARD INPUT ROUTINE
0049 *  IS USED WITH THE FOLLOWING VALUES:
0050 *
0051 *
0052 *  STATUS PORT = 0
0055 *  DATA PORT   = 1
0054 *  DAV MASK    = 40H  DATA AVAILABLE
0055 *
0056 *
0057 *
0058 *  THERE IS ROOM TO ADD A 'CMA' INSTRUCTION
0059 *  TO COMPLEMENT THE INPUT STATUS WORD FOR
0060 *  ACTIVE LOW STATUS.
0061 *  (SEE THE STANDARD INPUT ROUTINE BELOW FOR
0062 *  ACTUAL VALUE ADDRESS INFORMATION.)
0063 *
0064 *  TYPING 'ESC' (escape) WILL EXIT THE MAIN
0065 *  PROGRAM.  IF SOLOS/CUTER HAS CALLED, A
0066 *  JUMP BACK TO SOLOS/CUTER 'RETRN' (xx04H)
0067 *  WILL BE MADE.  THIS JUMP RETURNS TO THE
```

```
                    0068 *  SOLOS/CUTER MONITOR AND ISSUES A PROMPT.
                    0069 *  OTHERWISE THE DEFAULT RETURN JUMP ADDRESS
                    0070 *  WILL BE TAKEN.  IT IS INITIALLY SET TO
                    0071 *  RETURN TO THE 'ALS8', ADDRESS 0E060H
                    0072 *
                    0073 *  IF YOU ARE USING THE STANDARD INPUT ROUTINE
                    0074 *  AND WISH TO INSERT YOUR OWN EXIT ADDRESS,
                    0075 *  DO SO AT 'EXADD' ADDRESS 001AH-001BH.
                    0076 *  LSB - MSB
                    0077 *  (SEE EXIT ROUTINE BELOW.)
                    0078 *
                    0079 *
                    0080 *  TYPING 'DEL' WILL RESTART THE MAIN PROGRAM
                    0081 *  AT ITS STARTING ADDRESS
                    0082 *
                    0083 *  A JUMP ADDRESS TO THE ACTUAL PROGRAM
                    0084 *  IS INSERTED AT 'IJMP' AT THE COMPLETION
                    0085 *  OF THIS INPUT INITILIAZION.
                    0086 *  THIS ALLOWS RESTARTING THE GAME FROM
                    0087 *  LOCATION ZERO (00).
                    0088 *
                    0089 *
                    0090 *
                    0091 *
                    0092 ************************************************
                    0093 *
                    0094 *
                    0095 *
                    0096 *  < INPUT ROUTINE EQUATES >
                    0097 *
     007E           0098 DEL    EQU    7FH    DELETE KEY CODE
     001B           0099 ESC    EQU    1BH    ESCAPE KEY CODE
     00C3           0100 JMP    EQU    0C3H   JUMP INSTRUCTION CODE
     00CD           0101 CALL   EQU    0CDH   CALL INSTRUCTION CODE
     0004           0102 RETRN  EQU    4      SOLOS/CUTER RETURN LOW ADDRESS
     0000           0103 NOP    EQU    0      SOLOS FIRST BYTE
     007F           0104 MOVAA  EQU    7FH    CUTER FIRST BYTE
     003A           0105 LDA    EQU    03AH   SOLOS/CUTER INPUT FIRST BYTE
     001F           0106 LOWIN  EQU    1FH    SOLOS/CUTER INPUT LOW ADDRESS
                    0107 *
                    0108 *
                    0109 *
                    0110        XEQ    0      PROGRAM EXECUTE ADDRESS
0000                0111        ORG    0      ASSEMBLER ORIGINATE ADDRESS
                    0112 *
                    0113 *
                    0114 *
                    0115 *  < INITIALIZE INPUT ADDRESS >
                    0116 *
                    0117 *   IJMP = INIT2 - INITIALIZE INPUT ON PASS 1.
                    0118 *
                    0119 *   AFTER THE INITIALIZATION PASS, IJMP = THE
                    0120 *   STARTING ADDRESS OF THE PROGRAM TO BE RUN.
                    0121 *
                    0122 *
0000 C3             0123 INIT   DB     JMP    START JUMP
0001 27 00          0124 IJMP   DW     INIT2  PASS 1= INIT2  PASS 2= START
                    0125 *
                    0126 *
                    0127 *
                    0128 *  < WAIT FOR KEYBOARD INPUT >
                    0129 *
0003 CD OA 00       0130 INWAIT CALL   INCHR  CHECK IF INPUT
0006 CA 03 00       0131        JZ     INWAIT  FOR INPUT
0009 C9             0132        RET    .      WITH CHARACTER IN REG A
                    0133 *
                    0134 *
                    0135 *
                    0136 *  < INPUT ROUTINE CALL >
                    0137 *
```

```
000A CD             0138 INCHR  DB     CALL   FIRST BYTE OF CALL INSTRUCTION
000B 1C 00          0139 INADD  DW     INPUT  INPUT ROUTINE ADDRESS
0OOD C8             0140        RZ     .      NO INPUT
                    0141 *
                    0142 *
                    0143 *
                    0144 *  < EXIT/RESTART CODE CHECK >
                    0145 *
000E FE 1B          0146 RTEST  CPI    ESC    ESCAPE KEY?
0010 CA 19 00       0147        JZ     EXIT
0013 FE 7F          0148        CPI    DEL    DELETE KEY?
0015 CA 00 00    U  0149        JZ     START  START PROGRAM OVER
0018 C9             0150        RET    .      CHARACTER IN REG A
                    0151 *
                    0152 *
                    0153 *
                    0154 *  < PROGRAM EXIT JUMP >
                    0155 *
0019 C3             0156 EXIT   DB     JMP
001A 60 FO          0157 EXADD  DW     FORMS  PROGRAM EXIT ADDRESS
                    0158 *
                    0159 *
                    0160 *
                    0161 *  < STANDARD INPUT ROUTINE >
                    0162 *
                    0163 *  THIS ROUTINE IS USED IF THE CALLING PROGRAM
                    0164 *  IS NOT SOLOS OR CUTER.  THE DAV MASK AND PORTS
                    0165 *  MAY BF CHANGED AS REQUIRED FOR ANY INPUT VALUES.
                    0166 *  IF INPUT STATUS IS ACTIVE LOW, INSERT THE 'CMA'
                    0167 *  (2FH) INSTRUCTION AT THE 'NOP' ADDRESS 001EH BELOW
                    0168 *
                    0169 *
                    0170 *  ZERO FLAG IS SET IF NO INPUT RCV'D. (Z)
                    0171 *  ZERO FLAG IS RESET IF INPUT IS RCV'D. (NZ)
                    0172 *  CHARACTER IS RETURNED IN REG A.
                    0173 *
                    0174 *  REGISTERS MODIFIED: A
                    0175 *
0000                0176 STAT   EQU    0      STATUS PORT
     0001           0177 DATA   EQU    1      DATA PORT
     0040           0178 DAV    EQU    40H    DATA AVAILABLE MASK- ACTIVE HIGH
     007E           0179 PARITY EQU    7FH    PARITY MASK
     E060           0180 FORMS  EQU    0E060H  EXIT ADDRESS
                    0181 *
                    0182 *
                    0183 *
001C DB 00          0184 INPUT  IN     STAT   STATUS PORT = O
                    0185 *
001E 00             0186        NOP    .      INSERT 'CMA' (2FH) HERE
                    0187 *             FOR ACTIVE LOW STATUS
                    0188 *
001E E6 40          0189        ANI    DAV    DATA AVAILABLE MASK = 40H
0021 C8             0190        RZ     .      NO INPUT
0022 DB 01          0191 INDATA IN     DATA   DATA PORT = 1
0024 E6 7F          0192        ANI    PARITY  STRIP PARITY
0026 C9             0193        RET    .      WITH CHARACTER IN REG A
                    0194 *
                    0195 *
                    0196 *
                    0197 ************************************************
                    0198 *
                    0199 *
                    0200 *
                    0201 *  < INITIALIZE INPUT ADDRESS (SOLOS/CUTER) >
                    0202 *
                    0203 *  THIS CODE CAN GO ANYWHERE; IT IS USED
                    0204 *  ONLY ONCE AND MAY BE OVERLAYED AFTER
                    0205 *  THE INPUT IS INITIALIZED.
                    0206 *
                    0207 *  NOTE: TO FORCE STANDARD INPUT ROUTINE,
```

```
                    0208 *  EXEC 'IDONE'
                    0209 *
                    0210 *
0027 23             0211 INIT2  INX    H       CHECK SECOND BYTE (JMP)
0028 7E             0212        MOV    A,M
0029 FE C3          0213        CPI    JMP     JMP INSTRUCTION?
002B C2 4A 00       0214        JNZ    IDONE   NO MATCH, USE STANDARD INPUT
002E 2B             0215        DCX    H
002E 7E             0216        MOV    A,M     CHECK FIRST BYTE
0030 FE 00          0217        CPI    NOP     CHECK IF SOLOS: =ZERO
0032 CA 3A 00       0218        JZ     SETUP   YES
                    0219 *
0035 FE 7F          0220        CPI    MOVAA   CHECK IF CUTER: =7FH
0037 C2 4A 00       0221        JNZ    IDONE   NOPE, USE STANDARD
                    0222 *
                    0223 *
003A 2E 1F          0224 SETUP  MVI    L,LOWIN  SET SOLOS/CUTER INPUT >ADDR
003C 7E             0225        MOV    A,M     CHECK FOR 3AH
003D FE 3A          0226        CPI    LDA     SOLOS/CUTER INPUT FIRST BYTE
003E C2 4A 00       0227        JNZ    IDONE   NOPE, USE STANDARD
                    0228 *
0042 22 0B 00       0229        SHLD   INADD   SET INPUT ROUTINE ADDRESS
                    0230 *
                    0231 *
                    0232 *   SET SOLOS/CUTER RETURN ADDRESS 023'
                    0233 *
0045 2E 04          0234        MVI    L,RETRN  SOLOS/CUTER RETURN (xx04)
0047 22 1A 00       0235        SHLD   EXADD   INSERT INTO EXIT ROUTINE
                    0236 *
                    0237 *
                    0238 *  SET PROGRAM ADDRESS AT IJMP.
                    0239 *
004A 21 00 00   U   0239 IDONE  LXI    H,START  INSERT PROGRAM START ADDRESS
004D 22 01 00       0241        SHLD   IJMP    SET UP PROGRAM RESTART AT ZERO
0050 E9             0242        PCHL   .       GOTO PROGRAM
                    0243 *
                    0244 *
                    0245 *  < END OF INPUT INITIALIAZION ROUTINE >
                    0246 ************************************************
                    0247 *

CALL  00CD 0138
DATA  0001 0191
DAV   0040 0189
DEL   007F 0148
EORMS E060 0157
ESC   001B 0146
EXADD 001A 0235
EXIT  0019 0147
IDONE 004A 0214 0221 0227
IJMP  0001 0241
INADD 000B 0229
INCHR 000A 0130
INDAT 0022
INIT  0000
INIT2 0027 0124
INPUT 001C 0139
INWAI 0003 0131
JMP   00C3 0123 0156 0213
LDA   003A 0226
LOWIN 001F 0224
MOVAA 007E 0220
NOP   0000 0217
PARIT 007F 0192
RETRN 0004 0234
RTEST 000E
SETUP 003A 0218
STAT  0000 0184
```

III. TARGET (TARG)

(Version 2.4 January 7, 1977 S. Dompier)

TARGET is an animated Sol-VDM game with sound.

A. Mission

A movable photon missile is aimed and fired in an attempt to stop unmanned runaway robot spaceships.

There are several types of spaceships containing dangerous cargoes of pesticides, DNA experiments, artificial flavorings, TV commercials and so on. They should be stopped before they reach a civilized area of the universe and endanger the populace.

Remote control of the missiles in flight is achieved by aiming the launching tube. The ships (and their contents) are generated by random logic and follow no pattern.

If two ships should collide, the flight log as well as the most dangerous cargo on board are jettisoned as a mass-seeking ion parachute which must be considered the most dangerous hazard of all.

[Author's note: The game player may relate to the ships and missiles of TARGET as objects personally imagined by him. The above scenario is provided for those with an aversion to the destructive type games who may otherwise mistake the robot spaceships as earthly in origin. Aggression, still being a common human trait in 1977, is better exercised with a zero-sum game than spent on the physical real world. Besides--it's fun.]

B. Scoring

HITS:

BIG CARGO SHIPS	100 points
SMALL (& fast) SCOUT SHIPS	200 points
PARACHUTES	600 points (if you can hit them!)

Chain reaction multiple hits score extra!

MISSES:

ANY SHIP ESCAPING OFF-SCREEN	-20 points
MISSILE MISSES (or wasted by hitting explosion)	-30 points

Occasionally, an explosion will blow out the engines or destroy part of another nearby ship or parachute, leaving parts of it floating in space. This space debris will remain until it is hit by a missile or by another ship, the crash resulting in the generation of a parachute. A missile hitting the debris will score.

C. Game Start and Action Speed

After the instructions are displayed on the screen, the game is started by typing one of the numeric keys (1 - 9). This also determines the speed of play. The number keys may be used at any time to change the action speed with 1 designating the fastest action.

D. Aiming and Flight Direction

Missile aiming and in-flight direction are controlled by pressing the "," key to aim left and the "." key to aim right. (The "," keytop has a "<" and the "." keytop has a ">".)

There are five aiming positions: left, left-center, center, right-center and right. The missiles are launched by typing any letter key or depressing the space bar. This keyboard arrangement is the easiest to use.

As soon as the first missile has left the launching tube, another missile may be launched. The directions of missiles already launched will be altered by the aiming position of the launching tube.

The left and right aiming command keys may be changed if your keyboard layout makes the standard keys undesirable. Place the ASCI I code (7 bits, MSB parity should be 0) for the keys to be used as follows:

Left key:	Address	0600H	Currently 2CH (,)
		060DH	
Right key:	Address	0607H	Currently 2EH (.)
		061CH	

E. Demonstration (DEMO) Mode

A demonstration self-run mode may be initiated by typing "D" at the start of the game. The "D" takes the place of a speed key (1 -9) to start the game. The game will run itself until stopped by typing the "DEL" key. All aiming, launching and speed controls are enabled during the demo mode, allowing for manual operation as the system "helps" the operator along!

F. Sound

TARGET is equipped with sound-effects. Place any AM radio near the computer and run the demo mode. Adjust the radio dial and the radio itself in relation to the computer until a good sound is found. Small ships, big ships, parachutes,

and especially explosions should all be distinctive. For the best sound from a Sol, place the radio next to the center of the left side.

G. Game Time

During play, "Time" will flash and a countdown will appear at the top of the screen when eight seconds of play time remain. If the score is 4000 or greater, "extra time" goes into effect, and 20 extra seconds of play time are provided.

When the game is over, (TIME = 0), the instructions will be displayed on the screen and the score information will remain until a new game is started by typing one of the speed keys (1 - 9). If the current score is greater than the previous high score, it will become the "NEW HIGH SCORE". The high score may be cleared by typing "R" before starting a new game.

H. Extra Time

The thousands digit in the score is used to determine whether "extra time" is to be initiated. This value may be altered to any digit (1 - 9) by placing the ASCII value of the desired digit at location D02H in memory. For example, if 2000 is to be used as the minimum score to earn extra time, place 32Hex at location D02H in memory.

I. Other Commands

If the "DEL" (delete - 7FH) is typed at any time, the game will restart.

There are two commands which are not displayed on the screen. One is a super slow speed activated by typing "%" (shift-5). To resume normal speeds, type any numeric key.

The other command is a continuous run mode which is activated by typing Control-C (03H) either before or during the game action. The game will then run continuously until stopped by typing the "DEL" key.

J. Exit from TARGET program

An exit from TARGET is provided by typing "ESC" (1BH), ("ALT" on some keyboards). See the standard SOLOS/CUTER input routine for complete information.

IV. LIFE (LIFE)

(version 2.3 January 7, 1977 S. Dompier)

The game of LIFE was originally described in SCIENTIFIC AMERICAN magazine, October, 1970, in an article by Martin Gardner. The game was originated by John Conway of Cambridge University, England.

The computerized version of LIFE can be found on many computer systems--in many cases with Teletype print routines. This version, using the Sol computer's video display capabilities or the VDM-1 Video Display Module with other computers, allows initial patterns to be composed directly on-screen and instant visualization of each generation as it is created. In addition, patterns may be stored and recalled from seven memory pattern registers. The generation speed may be controlled from the console.

A. Genetic Rules

Cells (organisms, ducks, people, plants, etc.) reproduce, exist or die according to certain genetic laws. Conway derived the genetic law of the game of LIFE from the following criteria:

1. There should be no initial patterns for which there is a simple proof that the population can grow without limit.

2. There should be initial patterns that apparently do grow without limit.

3. There should be simple initial patterns that grow and change for a considerable period of time before coming to an end in one of three possible ways:

    a. fading away completely (no life)

    b. becoming stable (no change in pattern or population)

    c. a pattern oscillates in an endless cycle of two or more periods.

Think of each cell as being a square of a checkerboard. A cell may be either empty (shown as a space [no *] on the screen and in the following examples) or living (shown as an * both on the screen and in the examples). In the following examples, a '+' indicates an empty cell which is becoming a living cell.

1	2	3
4	*	5
6	7	8

IV. LIFE (cont.)

SURVIVALS Each live cell with TWO or THREE live neighbors will survive for the next generation.

```
   *              *      These cells
 *  *          *  *  *   all survive.
```

DEATHS Each cell with FOUR or MORE live neighbors will die from over-population.

```
   *
 *  *  - This cell dies (4 neighbors)
 *  *
```

Each cell with ONE or NO live neighbors will die from isolation.

```
 *     Both of these cells die from isolation,
 *     each having only one neighbor.
```

BIRTHS Each empty cell with EXACTLY THREE live neighbors is a birth cell and a new live cell will appear at the next generation.

```
    *
 *  +  *   = birth - three neighbors
```

Note: Births and deaths occur simultaneously.

Don't count a new cell until next generation.

```
Generation 1                Generation 2

                    +               *     (+ = birth - had three neighbors)

  *  *  *    =     x   *   x       =   *   (x = death - only one neighbor)

                    +               *
```

B. Operating Instructions

In this version of LIFE you have a choice of either a flat world or a round world display.

FLAT WORLD - Cells on the edge of the display do not have neighbors past the edge, and any births that occur there immediately fall off and are not counted. In computing the count of neighbors, cells past the edge are considered empty. This is similar to a petri dish.

ROUND WORLD - Presented here as a flat surface projection, cells on the edge of the display have neighbors at the opposite edge of the display (top-bottom; left-right). If a pattern is moving off the edge of the display, it will continue at the opposite side.

The round world representation is more representative of our Earth, and it usually yields more interesting pattern activity.

C. Pattern Storage

A pattern may be stored and recalled from seven memory pattern registers. When the question is asked, type the appropriate register number (1 - 7) to recall a previous pattern. The pattern stored in that register will be copied to the screen. The pattern may then be 'activated' by typing a speed key (1 - 9 and 0) or modified by the edit functions and then run.

When the question is asked, type the register number (1 - 7) in which to save an initial or modified pattern. The pattern will be saved after the numeric (speed) key is typed. (Note: If you type 'DEL' to start over before the pattern has been run, no register storage will occur.)
There are seven preset patterns in register storage. When the LIFE program is first loaded from tape, get and run these patterns in both the round and flat world modes. This provides familiarity as well as examples of some of the possible LIFE activities.

D. Generation Speed

The time between each generation may be controlled by typing a speed key (1 - 9 and 0). 1 is the fastest and 9 is the slowest. Typing '0' (zero) will stop the generation activity to allow extended study of a pattern.

The pattern may then be SINGLE STEPPED by typing the space bar. Typing a speed key will resume automatic generations.

Typing the 'DEL' key will restart the LIFE program.

Typing the 'ESC' key will exit from the program.

V. PATTERN (PTRN)

PATTERN is a pattern generating program for use with the Sol computer or a computer with a VDM-1 Video Display Module. The patterns are generated in a kaleidoscopic format using a horizontal and a vertical value as the initial input data.

The pattern may be selected from a possible list of 256 different patterns. Each combination of vertical and horizontal values will produce a unique pattern. There is also an automatic pattern mode which generates a sequence of some of the more interesting patterns.

The initial speed of PATTERN change is selected by typing any key. This key also starts the program. The binary value of the key used to start the pattern is used as a timer; the lower the value the faster the rate of change. The ASCII bias (30H) is removed from the speed key used, and the resulting value is decremented by 1. Therefore, the fastest speed would be selected by typing the number "1". The space bar produces a very slow rate of change. The number keys (1 - 9) produce a good range of speeds, with the number "9" being quite slow.

The program may be restarted by typing the "DEL" key, or by restarting the program at location zero. Typing the "ESC" key will exit from the PATTERN program.

A. Loading PATTERN from CUTS Tape

Set the GAMEPAC 1 tape so that PTRN is the next program on the tape, and read the tape using the XEQ command, i.e., XEQ PTRN or just XEQ.

PATTERN will then load and run, printing instructions on the screen. The hexadecimal value of the numbers typed for the pattern data is represented as an eight bit word on the screen and is a good way to become familiar with the hex numbering system:
(0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

VI. ZING (ZING)

Written by: Terry L. Todd May 1976
Revised by: Steven Dompier June 3, 1976
Revised for SOLOS/CUTER/STANDARD July 5, 1977

ZING is a ping-pong type game played using a Processor Technology VDM-1 Video Display Module or a Sol computer.

When using ZING with a Sol, a switch bank must be constructed in order to play the game. Details and a schematic are provided in the following pages.

If the computer is other than a Sol, a switch bank may be constructed incorporating a parallel port; however, the normal mode of play will use the front panel sense switches.

A. ZING Operation

Two paddles move up and down the screen sides and return any ball that hits them. Balls are generated at random from the center of the screen, and up to five balls may be in play at any one time. If all five balls have been returned, the balls will move faster. The balls will gain momentum each time all five balls are returned until a maximum speed is attained. As soon as any ball is missed, the initial slower speed is restored.

B. Paddle Operation

The left four sense switches control the left paddle, and the right four switches control the right paddle. The paddles are positioned up or down according to the binary setting of each player's four switches.

	LEFT PADDLE					RIGHT PADDLE			
DATA LINE:	D7	D6	D5	D4	:	D3	D2	D1	D0
SWITCH:	A15	A14	A13	A12	:	A11	A10	A9	A8
ROW:	8	4	2	1	:	8	4	2	1

Row (above) specifies the row at which the Paddle is located. The row is selected by the binary value of all four switches. The top row (0) is accessed by turning all four switches off. Row 1 would require a switch setting of 0001. Row 3 would have both the one switch and the two switch on, giving a total value of three (0011), and so on, counting in binary, until the last row (15) has all four switches turned on: 1111 or F hexadecimal).

VI. ZING (cont.)

C. Game Start

The switch settings at the start of the game are used to determine two modes of play. If the left player's "8" switch is on when the game starts, hexadecimal row numbering will be displayed at the sides of the screen. If the right player's "1" switch is on at the beginning, the game will run continuously until stopped by pushing the 'DEL' key to restart the game. If the right player's "1" switch is off at the start, the game will declare a winner when either player scores 21 points. The game may then be restarted by pushing the 'DEL' key.

SWITCH A15 up	=	Display hex numbering on screen side
SWITCH A15 down	=	No numbers
SWITCH A8 up	=	Continuous game - No stop at 21
SWITCH A8 down	=	Winner at 21

The momentum of the game increases each time both players have returned all five balls with no misses!

Run and restart game at location zero.

Type 'DEL' to restart game anytime.

Type 'ESC' to exit from the program.

D. Sol Parallel Port Switches

A bank of eight switches (or two bank's of four switches--one bank for each player) are required to play ZING.

These switches are connected to the parallel port at the rear of the Sol (connector J2M. Use SPDT type switches without a middle "off" position. Connect the switches using a DB-25 connector to mate with the parallel port connector J2 on the Sol. Provide enough connecting wire so that the switches may be positioned conveniently for play.

Refer to the Sol manual for the parallel port pin-out information, and see the schematic (Figure 1) for the correct hook-up of the switches.

If you wish to use external switches with a computer other than a Sol, construct the switch bank(s) as shown and connect to a parallel port using the lines indicated.

On parallel ports other than the Sol, +5 volts must be obtained to power the switches. Notice that this is obtained from pin 3 on the Sol J2 connector.

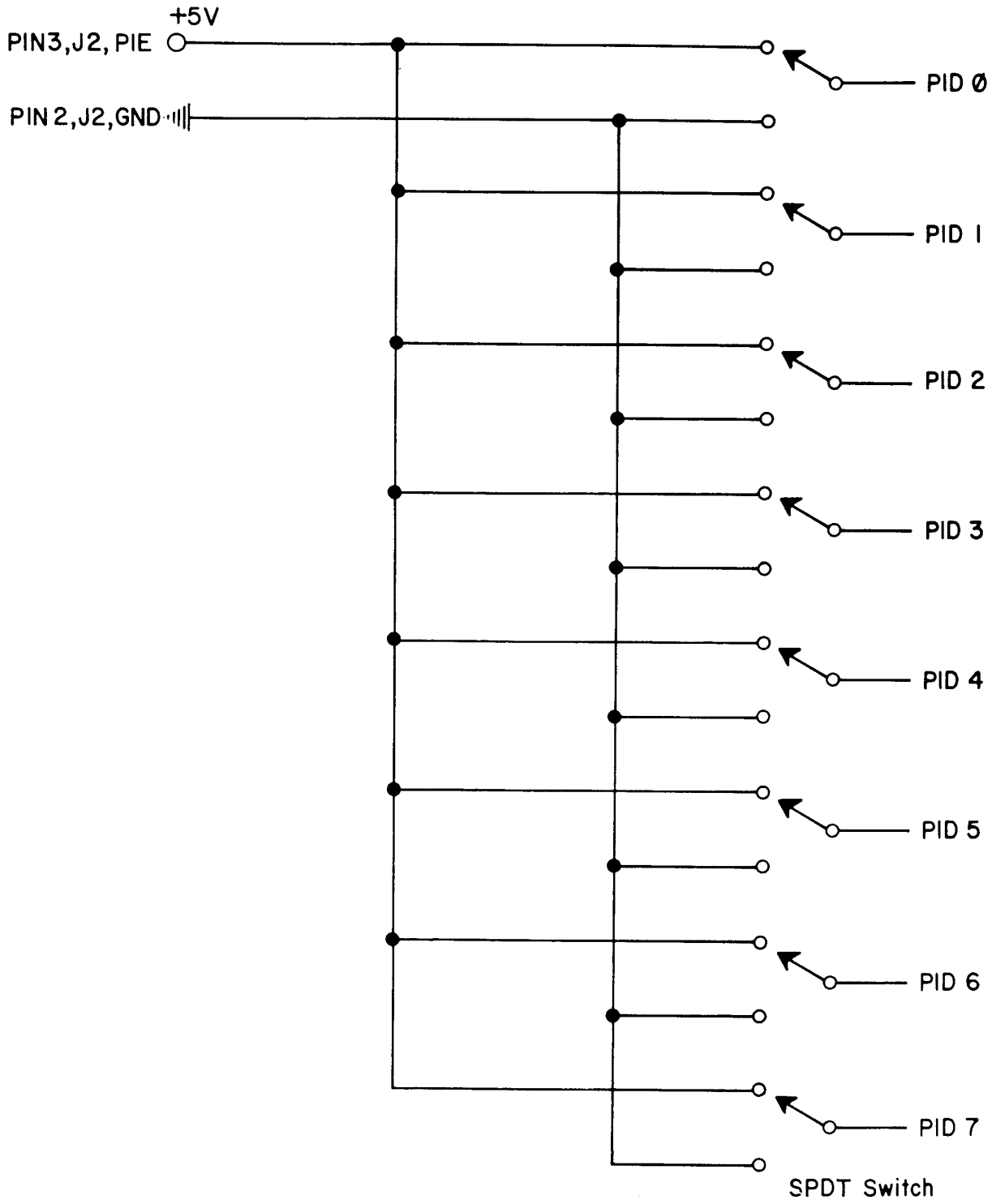


Figure 1

VI. ZING (cont.)

E. Patches

The GAMEPAC 1 programs use the SOLOS/CUTER or standard input routine and determine if the computer running the program is a Sol using SOLOS, some other computer using CUTER, or something else. The information below will allow you to modify the program for different combinations, such as using external sense switches with the Sol running SOLOS or some other computer running CUTER and using an external switch bank.

If the computer is a Sol, the parallel port (FD) is used for the switch bank input. If the computer is not a Sol running SOLOS, the sense switch port (FF) is used. To change this input port, first run the program to initialize the input routines, then stop the computer and make the patches needed, then restart ZING at location zero. The new port address wiII then be used.

The switch bank input port used is loaded from address 0027H. Change this byte if necessary.

APPENDIX A

SOLOS/CUTER Interface Specifications

The SOLOS/CUTER interface is based on:

1. A predefined set of 'pseudo' I/O ports allowing software compatibility and providing an easy means of supporting any I/O device.

2. A well defined set of register usage conventions.

3. A system jump table of entry points.

4. A defined tape format including headers and CRC characters.

Both SOLOS and CUTER observe and support these specifications such that any program written using this interface will function (except for specific device dependencies) under the control of either SOLOS or CUTER. A part of the interface specifications also allows a user written SOLOS/CUTER surrogate. Such a surrogate, when properly written, will allow a program written for SOLOS/CUTER to function with the surrogate.

The first aspect of the interface is that of the pseudo ports. The basic SOLOS/CUTER interface allows the support of four 'pseudo' I/O ports (0 - 3). These pseudo ports are logical ports providing a reference for the program only. System input (keyboard) and output (display) are directed via these pseudo ports. The STANDARD definition for pseudo ports is:

Pseudo Port	Input	Output
0	Keyboard	VDM Display
1	Serial input	Serial output
2	Parallel input	Parallel output
3	User defined input	User defined output

These pseudo ports allow device independent I/O. Provided that device dependent character sequences are not used, an I/O request to pseudo port 0 appears to the requesting program to be the same as a request to pseudo port 1, 2 or 3. What this means is that, although four pseudo ports are defined in the interface specifications, a user written surrogate would not need to decode pseudo ports.

The second aspect of the SOLOS/CUTER interface is the defined register usage. Each of the system entry points has specific register requirements which will be discussed later.

Whenever a program is executed via SOLOS/COTTER the stack pointer, the stack, and registers HL are defined as follows:

1. The Stack Pointer (register SP) is valid and offers a useable stack. The size of this stack is not specified but should be adequate for at least a few calls. The executed program is expected to establish its own stack; however, some stack should be available.

2. The stack itself should be established such that:

   (a) A "REV instruction can be used as an exit by the executing program.

   (b) The locations at Stack Pointer -1 and -2 in memory contain the address of the executed program itself. This information can be accessed by machine code similar to:

```
LXI  H,-1    A constant minus one.
DAD  SP      HL=SP-1 now.
MOV  A,M     A=our own high address.
```

   Code such as this can be used to allow a routine to be made self-relocating to a 256 byte boundary.

3. Registers HL contain the address of the SOLOS/CUTER jump table. Because this jump table may be located at any 256 byte boundary in memory, register L will be zero. Register H can then be used to alter the executing program accordingly. As noted later, the jump table also provides an indication whether the program is executing on a Sol or other computer.

The third aspect of the SOLOS/CUTER interface is the jump table. By making all system requests via this jump table, an executed program can be made compatible between SOLOS, CUTER or other properly written surrogate. The jump table is described on the following page. A more complete description is contained in the SOLOS/CUTER User's Manual.

SOLOS/CUTER JUMP TABLE

Address	Label	Length	Brief Description
xx00	START	1	This byte allows power-on reset for SOLOS. It is 00 hex on a Sol; 7F hex on other than a Sol.
xx01	INIT	3	This is a "JMP" to the power-on reset.
xx04	RETRN	3	Enter at this point to return control from an executing program.
xx07	FOPEN	3	Byte access file open.
xx0A	FCLOS	3	Byte access file close.
xx0D	RDBYT	3	Byte access read one byte.
xx10	WRBYT	3	Byte access write one byte.
xx13	RDBLK	3	Read an entire file into memory.
xx16	WRBLK	3	Write an entire file from memory.
xx19	SOUT	3	Standard character output routine. This must be an "LDA" pointing to the byte containing the current system output pseudo port value.
xx1C	AOUT	3	Character output to pseudo port specified in register "A".
xx1F	SINP	3	Standard character input routine. This must be an "LDA" pointing to the byte containing the current system input pseudo port value.
xx22	AINP	3	Character input to pseudo port specified in register "A".

The most often used routines are: RETRN, SOUT and SINP. Other entry points may or may not be used.

JUMP TABLE INPUT ENTRY POINTS

SINP address xx1F

This entry point will set register "A" to the current system input pseudo port. This must be an "LDA" instruction. After loading register "A", this entry point proceeds by executing "AINP" described below.

AINP address xx22

This entry point is used to input one character or status information from any pseudo port. On entry register "A" indicates the desired pseudo port. Because this entry point is a combination status/get-character routine, it is the user's responsibility to interpret return flags properly. When a character is not available, the zero flag will be set. When a character is available, the zero flag will be reset and the character will be returned in the "A" register. As an example, the following code will wait for a character to be entered:

```
LOOP CALL   SINP      get status or the character
     JZ     LOOP      status says character not
                      available yet
     ...    ...       character is in register "A"
```

JUMP TABLE OUTPUT ENTRY POINTS

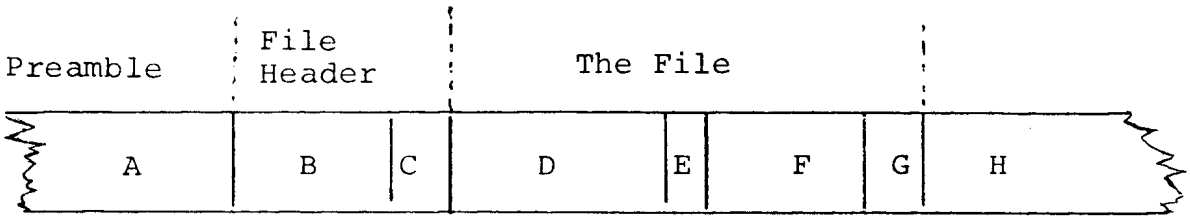
SOUT address xx19

This entry point will set register "A" to the current system output pseudo port. This must be an "LDA" instruction. After loading register "A", this entry point proceeds by executing "AOUT" described below.

AOUT address xx1C

This entry point is used to output the character in the "B" register to the pseudo port specified by the value in the "A" register. On return, the PSW and register "A" are undefined. All other registers are as they were on entry. A user written output routine (AOUT surrogate) may buffer or delay the output as required for the supported device.

The fourth aspect of the SOLOS/CUTER interface is the format in which the data is recorded on tape. When data is written to tape it is referred to logically as a "file". Each file has its own header which describes the file. On cassette tape, each header is followed by the file itself. The file itself is written to tape in segments of 1 to 256 bytes. Each segment is immediately followed by a Cyclic Redundancy Check character (the CRC). The following is the general format of one file on cassette tape:



Where:

A. Preamble

Preceding every file header is a special preamble. This is a series of at least ten nulls (zeroes) followed by a one (01 hex). This special sequence, and only this sequence, indicates a probable file header follows.

B. File Header

This is the 16 byte file header. The layout of a file header is:

```
NAME  ASC  'ABCDE'  A 5 character file name.
      DB   0        Should always be zero.
TYPE  DB   'B'+80H  File type character.  If bit
                     7=1, this is a non-executable
                     data file.
SIZE  DW   LENGTH   Number of bytes in file.
ADDR  DW   FROM     Address file is to be read into
                    or written from.
XEQ   DW   EXEC     Execution beginning address.
      DS   3        Space not currently used.
```

C. File Header CRC

This is the CRC character for the file header. If, when reading a file header, the CRC character is not correct, then the file header is to be ignored. A search would then be made for a new preamble (A above).

D. File Segment First

This is the first segment of the file itself. A segment is from 1 to 256 bytes. In this example, this segment is 256 bytes.

E. File Segment One CRC

This is the CRC character for the preceding segment-- in this example, the preceding 256 bytes.

F. File Segment Last

This is the last segment of the file. In this example, this is 44 bytes. Therefore, the length of this file is 256+44=300 bytes.

G. File Segment Last CRC

This is the CRC character for the preceding segment--in this example, the preceding 44 bytes.

H. Interfile GAP

This is a gap between files and is typically a clear carrier for about five seconds.

CRC Computation

The CRC character is computed for each segment or header. The following code performs the CRC computation assuming: Register "A" is the character just written to tape, and Register "C" is the final CRC. Register C should be set to zero prior to writing the first character of a segment. After writing the last character of a segment and executing this code, Register "C" is the CRC character for this segment.

An 8080 Subroutine to do CRC Computation

```
DOCRC EQU   $     A=NEXT character and C=CRC
      SUB   C
      MOV   C,A
      XRA   C
      CMA
      SUB   C
      MOV   C,A
      RET
```