LOADING CASSETTE PROGRAMS

Machine Language programs:

Cassette tapes use the Netronics format. To load, use these monitor commands: 04 ab cd ef gh, where ab cd is the starting address and ef gh the ending address of the program. After a satisfactory load you may wish to make your own tape using another cassette. To run, use 00 ij kl, where ij kl is the running address (not necessarily the same as the starting address) for the program. See table below:

program	start addr. (abcd)	end addr. (efgh)	run addr. (ijkl)
Star War	0100	0672	0100
Breakout	0000	07ff	0330
Chess Board	0000	0236	0100

Tiny Basic programs, either display, Netronics Full Basic programs:

Of course the Netronics format is used here. Get your Basic up and running. For programs with a USR routine in machine language the USR is loaded first. Branch or USR to the monitor. use 04 ab cd ef gh as for machine language programs. See table below. Then restart Basic (either cold or warm, it doesn't matter) and use LOAD to insert the main program. If there is no USR just LOAD the program.

program	USR start addr. (abcd)	USR end addr. (efgh)
Roundup Bugger (no USR)		
Carrier (no USR)		
Wages (no USR)		
Super Lander (no USR)		
Nomonitor	1b02	1b1d
Klaybor Shuttle (no USR)		
Battleship (no USR)		
USCF Ratings (no USR)		
Hex Dump	e004	e056

USR's can be relocated if necessary to anywhere in addressable memory not used by the BASIC interpreter, the main user program, or where other conflicts might arise. If you relocate, you will have to change: the USR,POKE,PEEK references in the main user program, and machine code register initialization in the USR itself. If you need help doing this, write me where you want to put things and I'll help as much as I can.

Cassette Chess Board

1. Using the program. The purpose of this program is to display a chess board with pieces and allow movement of the chess pieces to play a game or follow the progress of a game. The board and pieces are 'chunky' looking but you should be able to distinguish the light and dark squares, each of the pieces, and the black and white pieces. The board is displayed in the standard format at the start of a game, that is, with the white pieces at the bottom and the black pieces at the top of the chess board with a light square at the bottom right. The pieces are moved using "full " algebraic notation. This means that each of the squares is defined by a letter from a to h and a number from 1 to 8; the letter is always first. To move, type in the letter and number of the square of origin and then the letter and number of the destination square. See the board layout and move examples below: Captures are automatic if appropriate.

A8 0408	B8 0409	C8 040a	D8 040b	E8 040c	F8 040d	G8 040e	H8 040f
A7 0488	B7 0489	C7 048a	D7 048b	E7 048c	F7 048d	G7 048e	H7 048f
A6 0508	B6 0509	C6 050a	D6 050b	E6 050c	F6 050d	G6 050e	H6 050f
A5 0588	B5 0589	C5 058a	D5 058b	E5 058c	F5 058d	G5 058e	H5 058f
A4 0608	B4 0609	C4 060a	D4 060b	E4 060c	F4 060d	G4 060e	H4 060f
A3 0688	B3 0689	C3 068a	D3 068b	E3 068c	F3 068d	G3 068e	H3 068f
A2 0708	B2 0709	C2 070a	D2 070b	E2 070c	F2 070d	G2 070e	H2 070f
A1 0788	B1 0789	C1 078a	D1 078b	E1 078c	F1 078d	G1 078e	H1 078f

(all for white)

Move	algebraic	hex
P-K4	E2 - E4	070c 060c
N-KB3	G1 - F3	078e 068d
P-Q4	D2 - D4	070b 060b
P-QR3	A2 - A3	0708 0688

(all for black)

Move	algebraic	hex
P-K4	E7 - E5	048c 058c
N-KB3	G8 - F6	040e 050d
P-Q4	D7 - D5	048b 058b
P-QR3	A7 - A6	0488 0508

2. Limitations. Castling, promotion of pawns, and capturing "en passant" are not recognized. (I'm still working on this and when implemented it will be available for only copying and postage costs.)

3. Hardware. The program takes input from the #7 N-line parallel input port on the Netronics Giant Board. Any parallel port can be utilized by changing the appropriate instruction (see below). If no such port is available, you can use the hex keypad to enter moves using the addresses of the square of origin and square of destination. (see layout above)

4. Important memory locations:

0121 - 1861 turn on instruction.

0200, 0206, 021a, 0220 - INP7 instructions, EF3 is status ready flag, ASCII.

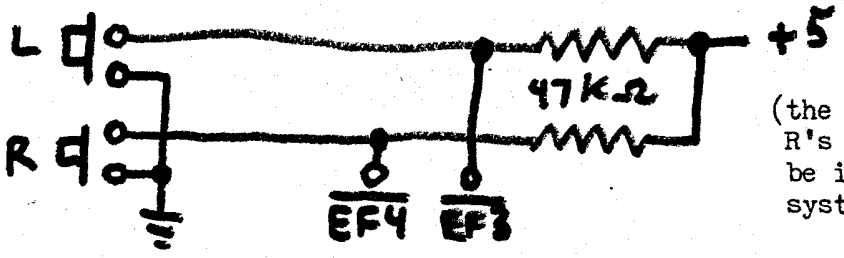
0177, 017f, 0187, 0190 - INP4 instructions for entry using hex pad, hit input after each nibble of addresses, EF4 is status ready flag.

0029 to 00b8 - silhouettes of pieces, one every 6 bytes, in following order: black pawn, black rook, black knight, black bishop, black queen, black king, white pawn, white rook, white knight, white bishop, white queen, white king.

HAVE FUN !

BREAKOUT
c, 1981
O.R. SHROYER

1. Hardware modifications. Add this circuit to move the game paddle:



(the 47 k R's may already be in your system)

2. Playing the game. To start the game, push the R switch. A ball will appear. Use the paddle to bounce the ball against the walls. R moves the paddle right, L left. The object of the game is to destroy as much of the walls using five allotted balls as possible. For each section of wall eliminated, 03 hex is added to your score and displayed on the hex readout. The Q light comes on when you are on your last ball. The direction of the ball bounce can be controled using "english" off the paddle: Using the left half of the paddle bounces the ball left and the right half of the paddle bounces the ball right. It is possible to increase game speed once you'breakthrough; that is, the ball strikes the rear of the court (see below). You can also choose whether the paddle is as fast or twice as fast as the ball.(see below).

3. Important memory locations:

04a3 - OUT to hex display.
033d, 04be - ball speed/paddle speed index: 00 = different speeds, 01 = speeds equal.
034c - breakthrough game speed: 00 = speed up, 01 = no change.
04aa,046b, 04e7*- game speed: data proportional to speed.
0458 - number of balls per game: data = number allowed.
045d - when Q light on (Q high): data = ball number when Q on.
034f - 1861 chip turn on instruction.

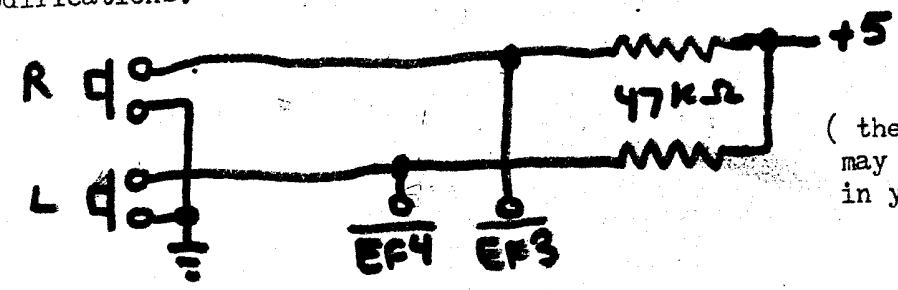
*04e7 is relevant when it is desired to speed up the game after breakthrough. The effect is to add the 04e7 data to the 04aa/046b data. Don't get carried away though because there is no carry.

Any of these locations can be changed to suit your taste, especially game speed since as written it is fairly fast.

HAVE FUN !

Cassette STAR WAR
c. 1979, Donald Richard Shroyer

1. Harware modifications. Add this circuit used to fire the "phasors":



(the 47k R's may already be in your system)

2. Playing the game. The weapons of each ship are fired by grounding the appropriate EF, EF3 for the right ship, EF4 for the left ship. Ships are moved by means of the hex keypad, the left half of the pad for the left ship, the right half for the right ship according to the following: 0 - left ship down fast, 4 - left ship down slow, 8 - left ship up slow, C - left ship up fast, 1,5,9, or D freeze left ship. 3 - right ship down fast, 7 - right ship down slow, B - right ship up slow, F - right ship up fast, 2,6,A, or E freeze right ship. After executing the program, press the input switch to start the game. Nothing can happen until the right ship makes some move (the bad guys have to start the fight). During the game the hex display shows the last two moves made. At the end of the game the hex display shows the score, left nibble is the left ship's score, right nibble is the right's score. To play again, reexecute, push input, and then move each ship across the screen to erase the "ghosts" of the last game.

3. Important memory locations:

0149 - 1861 chip turn on instruction.
01ac to 01b1 - silhouette of the left ship.
01b6 to 01bb - silhouette of the right ship.
0367, 0368 - weapons delay counter (both must be the same).
03ea - score to win, high nibble is left score to win, low nibble is right score to win. you can give odds by changing this data but a score to win must be a perfect square of two.
065d, 066b - game speed counter (both must be the same).

Any of these locations can be changed to suit your taste eg. by creating new ship silhouettes, slowing down or speeding up the game, giving odds, etc.

HAVE FUN !