VIPER INTERNAL DUAL SWITCHPAKS FOR MEMORY ADDRESSING AND BUS CONTROLS

Switchpak #2

#1 and #2 switches are the extended memory addressing control lines. Switches #1 and #2 should never be on at the same time. Unless you are using the Syste: 5 with its extended memory addressing capabilities switch, 1 should always be off and switch 2 should always be on.

Switch #3 is used to modify the addressing values of Switchpak #1 by 4K increments.

Switch #4 is a signal called System-Enable. The Viper must disable this signal in order to use the Bally bus to communicate with the Z-80 microprocessor inside.

Switch #5 is called Buzzoff. This signal must also be used to control the Viper to Bally bus signals.

Switch #6 is the Casette enable signal. This is used to enable or disable the cassette cartridge slot when using the Viper 1.

Switch #7 is the 7 meg. clock signal from the Bally bus. This clock is used to control the Viper Ram operation timing.

Switch #8 is used for changing the clock to the Ram from external (the Bally clock) to the internal Crystal Y1. Y1 is optional and is not included with the system.

THE NEW EXTENDED BASIC 1.0 COMMANDS LIST

To execute the following, type in the command, fill in the variables and hit "go".

"NEW" This command will erase all memory and reprint extended basic 1.0. This command is the equivalent to pressing the reset button.

"DEFAULT" The default command will reset all the graphics and character window variables plus the music prosessor.

"ZERO" Clears all letter variables A to Z.

"DATA" Initializes variables. Example - DATA A,5,10,15=a=5; b=10; c=15.

Edit

NEW GRAFIX COMMANDS

"CIRCLE" X,Y,R,M. To draw a circle, you must enter the X and Y coordinates for the center, then the radious R, and then the mode. Example - Circle 0,0,50,1 will draw a circle at the center of the screen 50 pixels wide in color 1.

"SCROLL" X, Y, X size, Y size ± #lines.

SCROLL 0, 0, 50, 50, - 100. This will scroll an area 50 pixels wide, by 50 pixels high, down 100 pixels. This will scroll upwards if # positive, downwards if negative.

"SNAP" X, Y, XS, YS @ (#). This command takes a snapshot of the image on the screen, starting with XY as the center and XS, YS are the variables used for the size of the area you want to save. The image is then stored in the @ (#). Example - 0,0,20,20@(100). This will snap the image starting at the center of the screen 0,0.

An area 20 pixels wide and 20 pixels high will then be stored in string @ (100).

"SHOW" X, Y, SHOWMODE, @ (#). This command will display the snapped image with XY as the center. The string # (@(#)) then must be the same as the snap @(#). Special showmodes are:

0 PLOP

1 OR

2 XOR

NEW MODES !

The following grafix commands use the new mode table below:

CIRCLE,
POINT,
LINE,
BOX.

MODE:

0-nothing	4-PLOP 0
1- XOR 1	5-PLOP 1
2- XOR 2	6-PLOP 2
3- XOR 3	7-PLOP 3

The XOR mode used is used to mix the color with whatever color is already on the screen. This means that if you were to use a box in mode 1 the color would be equal to the FA value and when it passed over an area of a different color the two would mixwith each other rather than erase one with the other.

The Plop mode will place the box on the screen in the color you select anderase anything that was there before.

This quick program will help to familiarize you with this feature and also verify the proper operation of the Viper System.

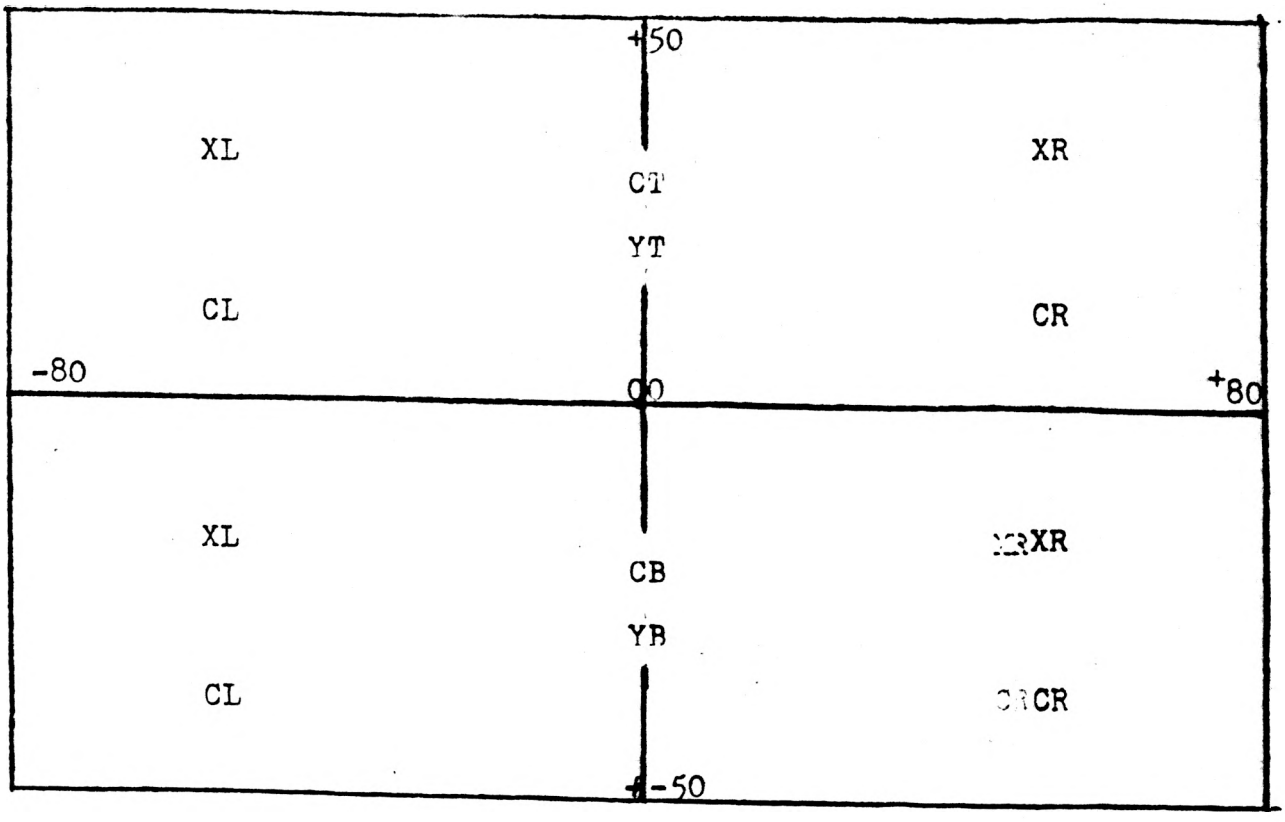
LOADING EXTENDED BASIC FROM TAPE:

The auto write protect switch position is primarily designed to implement the use of Jay Fenton's excellent Extended Basic, which is included free with every Viper 1. The language occupies the first 8K of Ram Space. This will leave you approximately 8K of programming Ram minus approximately 1K of Ram for the basic to use as stock and scratch pad. Since the language is loaded from tape, it is possible that not every tape recorded will load properly due to poor tape head alignment on the low end priced recorders. It is strongly recommended that a quality tape recorder be used when loading the language to insure proper operation. A check sum program is included to verify if your load was 100% correct or not. Once you have verified a perfect load, you then must make a copy of it onto your regular tape recorder so you will have a compatible copy for your regular recorder.

To use:

Set the 8K/24K switch to 24K, then with your regular Bally Basic, load the tape as if it were any program : input/go.

The tape will tape approximately 7-8 minutes to load, at the end of the load it will print out (load done, throw write switch to Auto Write Protect Position), then press the go key to jump to the extended basic. If the load was at least partially successful, the screen should clear, and then print in the new small character font "Extended Basic 1.0".



GRAFIX WINDOW SCREEN LOCATIONS, XL, XR, YT, YB.
CHARACTER WINDOW SCREEN LOCATIONS AT, CL, CR, CT, CB.

ADDRESS OPERATOR

Putting a ← in front of a variable, will give its address. for example:

PRINT ← A will print -17238, which is the memory location where A's value would be stored.

PROGRAMMABLE WRITE PROTECT/WRITE ENABLE CIRCUIT:

The Programmable Write Protect Circuit enables the user to protect the entire 16K of memory from being accidently erased if your program causes the computer to crash. This saves time in not having to reload the extended memory because it will not have been erased. (Power failure or holding the reset button down for too long will cause a loss of memory.) With this feature, you can remove the Bally Basic cartridge, play some games, then stick Bally Basic back in and continue programming where you left off, without having to reload extended Basic because the data will remain intact. To use, set the Auto Write/Programmable Write switch in the Programmable position, then enter &(192)=0, this command will allow you to enter data to the Ram. To protect it (convert Ram to Rom) simply enter &(64)=0. Now the Ram is protected from being accidently or otherwise changed or lost.

Example: set the 8K/24K switch in 24K position.

Program	Comments
&(192)=0	Write enable.
%(24576)=32767	Poke 32767 into first extended memory address.
Print %(24576)	Test to see if it has been written to our 32767.

Now that the number is in our first extended memory address, turn off the Write Enable.

&(64)=0	Write Protect the Ram.
%(24576)=0	Try to change the value in the memory to zero.
Print %(24576)	Verify that it did not change to 0 but is still 32767.

AUTO WRITE SWITCH:

The reason you must use the Auto Write Switch position with the Extended Basic running is because 1/2 of the Ram is being used to store the new basic language and the other half is being used by you to write programs in. Because of this, you must be able to write to the upper 8K of memory, but still protect the lower 8K. If the lower 8K was ever written into, you'd destroy the memory containing the Extended Basic. The Auto Write Circuit is designed to check to see what area of memory you are in, and then automatically turn on and off the Write Protect/Write Enable Circuit. Whenever the memory address is using Extended Basic, it will protect it from accidently being written over. Then when the address is above the limits of the language, it allows you to write it to memory so you can write programs in Extended Basic.

COPY CARTRIDGE PROGRAM

```
10    : input ; for I = 24576 to 28672; %(I) = KP; Next I
50    : return ; print "1-switch to Auto Write,"
52    Print "2-remove basic cartridge,"
54    Print "3-throw switch to 8K"
58    Print "press reset to play!."
60    STOP
100   Clear ; : print; list; print "Run"; For I = 1 to 2000;
      next I; C+=-46; N+=0
110   For I = 8192 to 12288; TV=%(I); Next I : Return
150   : Return; Default; clear; Print "Done"
```

TOKENS

The following control characters type the corresponding tokens.

A	RND (AMBIGUITY)
B	BOX
C	CLEAR
D	DATA
E	(edit key for line editor)
F	FOR
G	GOSUB
H	RUBOUT or ERASE
I	INPUT
J	GOTO (jump)
K	IF (near I)
L	LIST
M	GO
N	NEXT
O	CIRCLE (there's a circle on the key)
P	PRINT
Q	SNAP (near SHOW on keyboard)
R	RETURN
S	STEP
T	TO
U	POINT (UPDATE A POINT)
V	DEFAULT (VARIABLE DEFAULT)
W	SHOW (WRITE TO SCREEN)
X	RUN (X FOR EXECUTE)
Y	SCROLL (WHO KNOWS)
Z	ZERO

EXAMPLE: To abbreviate PRINT type PR. PRI. or just P.

NEW VARIABLES

GRAFIX VARIABLES

XL
XR
YT
YB — left, right, top and bottom limits of the graphics window

CHARACTER VARIABLES

CL
CR
CT
CB — left, right, top and bottom limits of character window

CC — character color

FA — foreground color 1
FB — foreground color 2
FC — foreground color 3
BC — background color

LC — last character displayed on screen

NB — number base - ie. changing to 16 will print Hex, 2 binary and so on

BYTE (VARIABLE, byte #) = EXPR/SETS indicated byte (0 - low order, 1 high order) = to EXPR

C = BYTE (EXPR, byte #) returns indicated tyte of EXPR

CHANGING CHARACTER FONTS

To change from the default of SMALL 3x5 to LARGE 5x7, type the following:

CF = LARGE

To reset to small characters, tye:

CF = SMALL

HEX # INPUT

By putting a ! in front of a hexidecimal number, it becomes decimal. for example:

PRINT !2FF will type: 767, the decimal equivalent.