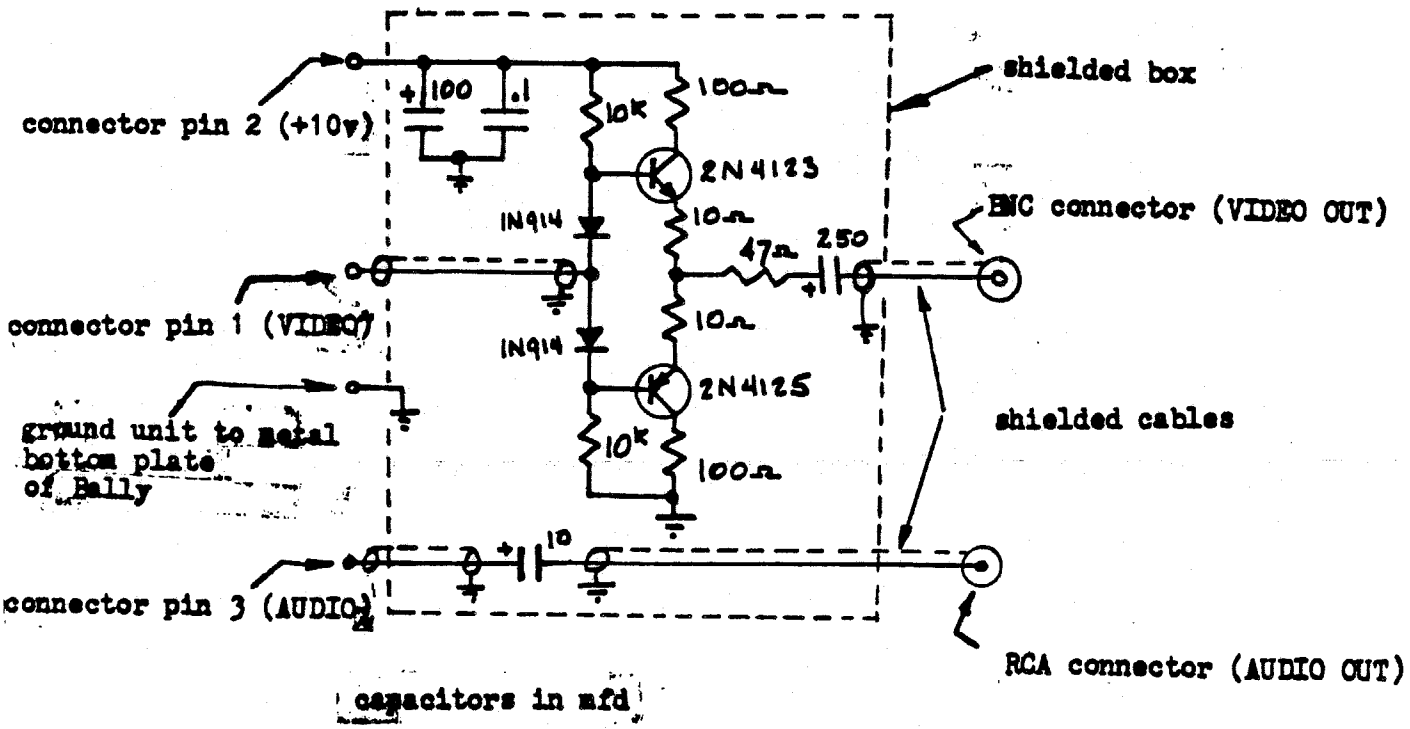
Arcadian - Important (or Neat) Articles

Table of Contents

DIRECT VIDEO CIRCUIT DIAGRAM (monochrome) is included below. Designed and built by Dan Sandin, it was submitted by Phil Morton. I built one and tried it out on my b/w monitor. (There is no color capability in this circuit). My breadboard layout works, but needs proper shielding for best results. Items to keep in mind:

- Use shielded coax - RG 174/U recommended.
- Remove Bally plastic upper cover and pull off (to the side) the RF modulator (tin box, 2"x3-3/4") on left. Pin numbers are 1 to 8, with 1 towards the front.
- BNC connector is standard for video. Audio could be either BNC or RCA
- Don't ask me how to hook it up to your TV.
- If there is enough interest, we could make up an inexpensive kit (say $15 tops)




transistor substitutes:
2N4123 = RCA SK3444 = Motorola HEP S0015
2N4125 = RCA SK3466 = Motorola HEP S0029

POWER SUPPLY The special transformer used by Bally (Fig. 1) can be replaced by a combination of three separate transformaers as shown in Fig.2 for a home-built supply. Assure that the secondaries are in phase - that is, the voltage across pins 1 and 3 should be 20, and across 1 and 4 it should be 32. If not, reverse one set of connections and retest.

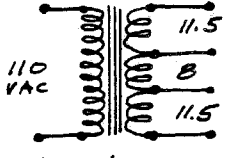



FIG 1

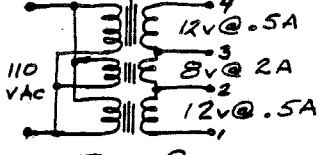
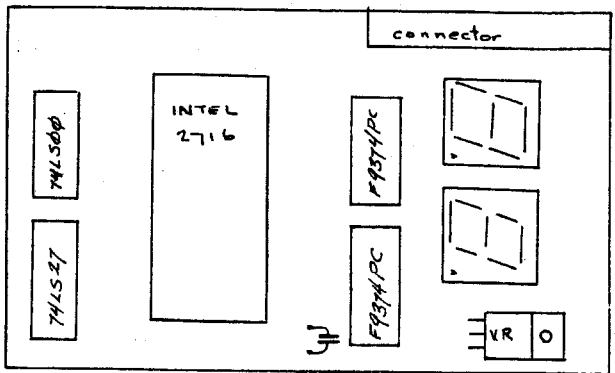



FIG 2

The above data from Al Rathmell

BALCHECK For those of you technical types that obtained the material offerred in the last issue, here is a parts layout of the pc board as was produced by Bally:



BLUE RAM PROGRAM. New capabilities of the Bally Arcade with the Blue Ram add-on was the program topic at a recent meeting of the Tidewater Computer Club at Norfolk, Virginia. John Perkins of Perkins Engineering presented practical home uses of the Bally and the following accessories developed by Perkins Engineering: Blue Ram, Keyboard, Modem interface, BSR controller interface, and Music compiler. Inputs were made easily with the keyboard, complete with the fancy features of N-KEY ROLLOVER, REPEAT, CONTROL CHARACTERS, etc.

The BSR, a popular home device controller, was operated via the controller interface by the Bally executing a demonstration program. A lamp was controlled by the BSR, and, while the club watched, the lamp flashed on and off and slowly dimmed and brightened as the Bally processed a time table set of instructions. We discovered how convienent and perhaps economical it would be to control automatically the hours certain home devices are turned on and off.

Next, we listened with pleasure to several Bally compiled musical selections from Bach and Handel in three part harmony. It was explained that the compiler is used to translate a person's own composition or regular sheet music into a form that is played by the Bally.

Then, a quick phone call connected the Bally (with modem) to the Source, using the modem interface. The Source is a nationwide communication/information sharing network. Previous club programs demo'ed the Source, but never was such realtime excitement generated. First of all, everyone was able to read inputs/responses in large print on the T.V. Secondly, as we sampled the smorgasbord of Source information, a New Hampshire computer group broke in, wanting to "chat" with us! Well, when they discovered we were communicating via a Bally Arcade, we read across the screen, "YOU'VE GOT TO BE KIDDING."

Yes, Bally Arcade and Perkins Engineering brought us (and New Hampshire) some surprises that night!

P.S. Bally users are in a majority at club meetings, and new Bally users are invited to attend bimonthly at ECPI, Stanwick Building.

Karen P. Cravedi, VEEP

MOTHERBOARD MODIFICATIONS The following changes are recommended if your machine has any of the listed symptoms. The author, Barry Ellerson, has sent us some "inside information", and can provide a small, built-up addition, ready for installation. Check his ad on p.80

If your unit has these symptoms: Screen Tearing, Loss of Horizontal Sync. on warm-up, Unit goes Dead - or keeps Resetting after warm-up, then the following modifications will correct them. If your unit went completely dead following these symptoms, these modifications will probably repair it.

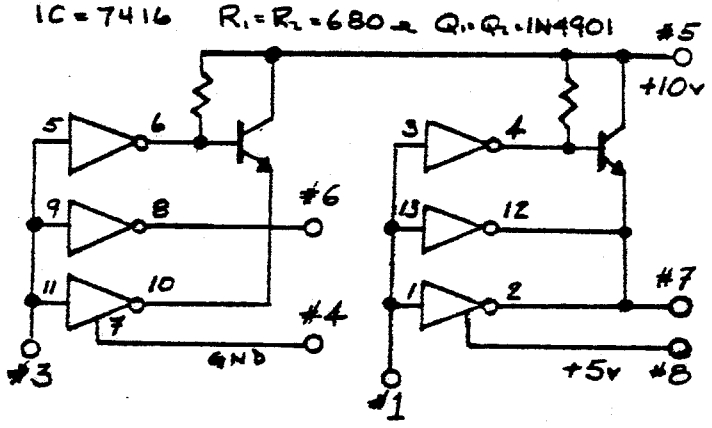
## NECESSARY MODIFICATIONS

1. Replace 74LS74 (U-16, Clock) with 74S74.
2. Replace Driver 75361 (U-32, 8-pin,clock) with Kludge board assembly. NOTE: Be sure to heat pins enough to cause solder to flow around double sided foils, as there is no way to see if there is a good connection once the board is in place. Remove crystal - extend leads - cover w/spaghetti or heat shrink tubing - place suitable insulating material over board to prevent crystal from shorting points on top of the board - Lay crystal flat back over top of Kludge board. CAUTION! Use extreme care when removing 75361 driver. If plated through holes are pulled out during removal, repair by pushing thin wire through bottom of board, bend over, and solder to top foil. Then install and solder Kludge assembly. Cut off excess wire on bottom.
3. Remove resistor and capacitor (see diagram), and place jumper where capacitor was. (These units may not exist on your board, or this may have been done by the factory.)
4. Jump 27 ohm resistor R-1 (10v supply, 1w) with a 47 ohm resistor, 1/4 watt or larger.
5. If you have a grey colored data chip (under the keypad), this old style unit which can cause further problems with the clock, DM81LS95, and/or memory should be replaced with a new verson (black color) and properly heat sunk, after cutting a hole in the top shield.
6. Replace 82 ohm resistors in clock (R-12,13) with 47 ohm resistors.

## OTHER MODIFICATIONS USEFUL BUT NOT ABSOLUTELY NECESSARY

1. Put in jumper wire from cathode of CR-3 to "+" end of C-6.
2. If C-19 (I/O) is glass, orange and black, replace.
3. Check front edge of key pad and file off any protruding leads.
4. If line filter (in metal box 1"x2") is high resistance type (no tape, or not toroid) replace with new style, low resistance.
5. Put hot melt wax on base of key pad.
6. Replace CR-3 to -6 with schottky 1N5817 or equivalent.

The wiring schematic for the Kludge board follows.



PRECAUTIONS

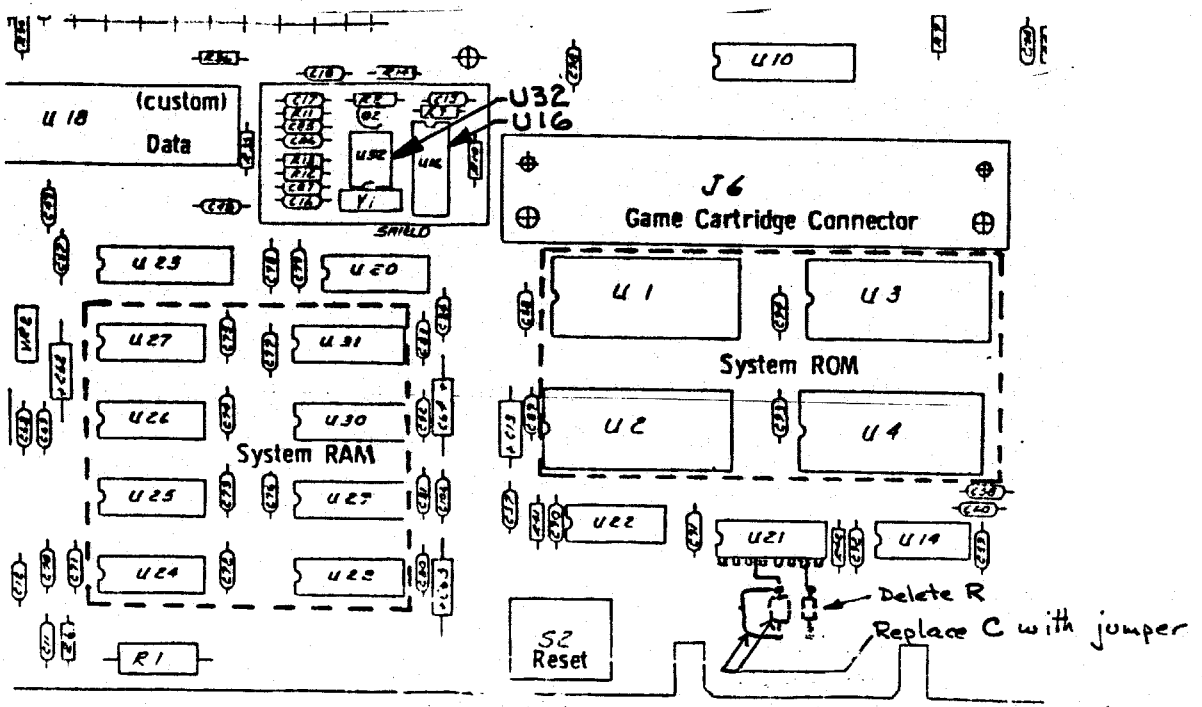
1. Do not wear nylon clothes. Work in a static-free environment, preferably grounded.

2. If the unit is operated outside of its case, short across C-6 before further handling. Inadvertent shorting to other points on board could blow components.

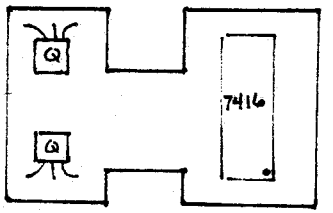
3. Check to be sure metal bushings in bottom shield pan do not short across any foils.

4. Check on-off switch for center lead that extends beyond board edge as it could short to the shield pan.

5. Check 5v. heat sink for good mechanical contact and check clearance of spring clips and board foils.




Full size pc board of Kludge:



PRE-TUTORIAL

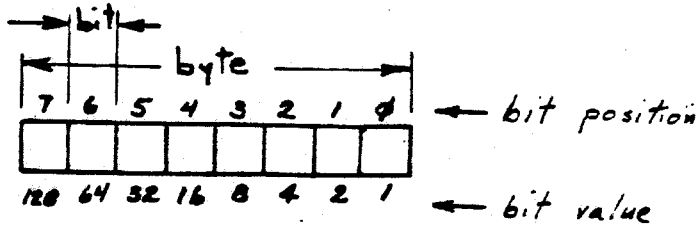
The "pixel" is the smallest object that can be shown on the screen. Its actual size is determined by two things, the physical size of the TV screen itself, and the "resolution" that the computer is capable of. The Arcade will divide every screen into 16320 pixels, disposed at 102 high by 160 wide. Obviously, the pixel is therefore bigger on a bigger screen. These 16320 pixels form the "low resolution mode", and by adding more memory (plus a few other items), the resolution can be increased to twice as many (medium resolution) or twice again to 49280 pixels (high resolution). The Perkins mod described on page 63 can provide any of these resolutions. Well, with four times as many pixels on the same screen, each one has to be one-fourth the size of the original, so detail of objects, characters, etc., is greatly enhanced.

Each pixel is turned "on" or "off" by the TV's electron gun each time it makes a sweep of the screen, so you have to have enough memory on board to remember the status of each pixel. The screen is therefore "refreshed" at every sweep.

The computer stores such common things as characters, letters, etc., in little subroutines and "calls" each one as requested by the program, without having to generate the character every time. The program in Rich's tutorial below indicates a simplified way for you to generate your own character (for a game, etc.,) and control its location by setting up a subroutine that your big program would call as needed. To do this, he sets up a 16 pixel by 10 pixel workspace, and determines which of he pixels in that space has to be "on" to generate the character. This is done outside the program, but the answers are entered into the program for permanence.

BITS, BYTES, and PIXELS

There are eight "bits" in a "byte". The computer uses bytes in most of its computations, and each one has a number assigned to it, from 0 to 256. This value is calculated by a unique system, as shown:




Each bit can be either a "1" or a "0". For each "1" that is shown, one adds together all the corresponding "bit values". This sum is the "byte value". Every number from 0 to 256 can be developed this way, and there is only one combination to do it. As as example, for a byte value of 84, bits number 6, 4, and 2 must be set equal to "1", for their bit values are 64, 16, and 4 respectively, their sum is 84. Mathematicians will see that the bit values represent the powers of two raised to the bit position.

Now to pixels. Each bit is equivalent to a pixel insofar as the screen picture is concerned. At every "1", the screen will be "on", so we can easily generate a picture by selecting the bits that we want to be "on", determining the byte value, and telling the computer that information - as Rich does in the following:

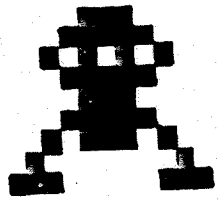
TUTORIAL  Creating Special Graphics, by Rich Tietjens

Most of us who program in Bally BASIC have often wished for a faster means of creating complex characters on the screen. The most obvious method uses a machine-language subroutine, but many hackers have no desire to "get involved" in the complexities of this mysterious and arcane art. However, if you are willing to do some doodling on paper and a little basic arithmetic, you can create your own special characters without headaches.

First, you must write (or modify) your program to employ the subroutine. This requires at least two variables to position the character; I chose H (for Horizontal and V (for Vertical).

Now, wherever you want to draw your character, you must set the H and V values; the position 0,0 is in the upper left corner of your screen. Position H,V will define the upper left corner of a character block 16 pixels wide by 10 pixels high. H may range from 0 to 159; V from 0 to 99 (which is off the bottom of the screen). Then POKE 20203,Vx256+H (( that is, program the statement to read - - %(20203)=Vx256+H ))
Now, CALL 20200  Voila! There is your character! But wait, we can't RUN your program without the subroutine in place; so make your changes; then go ahead with the rest of this story.

The following routine will draw, move, and limit to the screen area a special character.



When your main BASIC program is modified to your satisfaction, save it to tape; verify a good recording and set it aside.

Now let's define your characters; each is drawn on a 16x10 matrix, thusly:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | X | X | X | | | | | | | | | |
| 2 | | | | X | X | X | X | X | | | | | | | | |
| 3 | | | X | | X | | X | | X | | | | | | | |
| 4 | | | | X | X | X | X | X | | | | | | | | |
| 5 | | | | | X | X | X | | | | | | | | | |
| 6 | | | | X | X | X | X | X | | | | | | | | |
| 7 | | | X | | X | X | X | | X | | | | | | | |
| 8 | | X | | | | | | | | X | | | | | | |
| 9 | X | X | X | | | | | | X | X | X | | | | | |
| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Note that this positions the figure to the left of the available 16 pixel block. As a drill, see if you can determine the entries that will place the figure at the right side within the block.

RICH TIETJENS
501 S. ARCHER #5
SAN ANGELO, TEXAS 76903

Vertical wrap doesn't work

```
10 . POSITION AND DRAW
20 %(20203) = 256 b V + H; CALL 20200
30 . GET NEW POSITION
40 X = H + JX(1); Y = V - JY(1)
50 . WRAPAROUND
60 IF X > 159  X = 0
70 IF X < 0  X = 159
80 IF Y > 99  Y = 0
90 IF Y < 0  Y = 99
100 . ERASE
110 CALL 20200
120 . CHANGE POSITION AND GO DO IT AGAIN
130 V = Y; H = X; GOTO 20
```

In this example, representing an Apollo Lunar Module, the X's show the pixels which will be turned on when the subroutine is called. Since no pixels are turned on in row 0, both bytes equal zero. In row 1, bits 3, 2, and 1 of the left byte are turned on, equalling 8 plus 4 plus 2, or 14; the right byte is still zero.

When you have all the values figured out, load PROGRAM A; RUN it and enter the values computed in the byte value table. If you have two characters, enter any number except zero when asked "two figures?"; enter 0 if you only have one graphic character.

Now, the program will ask "READY?". Press any key (except HALT) to have your character displayed (both will display if you have two.)

If it (they) is/are ok, put the main program tape back in the recorder, with the tape positioned just past the trailing end of the BASIC listing. Start the recorder and press "GO". PROGRAM A will assemble your subroutine and put it on tape. When it is done, you can RESET the memory, load the tape just created, and there is your pet graphic character! If you wish to switch to the second graphic character. just POKE 20213,129 (that is, %(20213)=129 ). POKE 20213, 128 to switch back.

If instructions are included on the tape, using the REM (.) statement, the machine language subroutine should be recorded after the instructions; otherwise the remarks will override the subroutine. And that's all there is to creating and using your own graphic characters!

(Note for Blue Ram owners: you can do the same thing, but have a lot more room to play around in. Some addresses within PROGRAM A must be changed, along with the CALL address.)

Special thanks to Tom Wood for the effort expended on the On-Board ROM subroutines.

PROGRAM A

```
1001 . GRAPHIC CHARACTER MAKER
1002 . BY RICH TIETJENS
1010 GOTO 1030
1020 %(Y) = V;  Y = Y + W; RETURN
1030 CLEAR ; NT = 1;X = 0; Y = 20200; R = Y; W = 2; Z = 1020
1040 FOR U = X TO  X + 19 STEP 2
1050 CY = 32; PRINT #2, "ROW", U c 2, ":
1060 INPUT "LEFT BYTE?" @(U)
1070 INPUT "RIGHT BYTE?" @(U +1)
1080 NEXT U
1090 IF U < 21 INPUT "2 FIGURES?" T; IF  T  K = 20; GOTO 1040
1100 V = -43; GOSUB Z; V = 6965; GOSUB Z
1110 V = 10240; GOSUB Z; V = 20210; GOSUB Z
1120 V = -13871; GOSUB Z; V = -1936; GOSUB Z
1130 V = -32690; GOSUB Z; V = 12288; GOSUB Z
1140 V = 2432; GOSUB Z; V = 527; GOSUB Z
1150 V = -247; GOSUB Z; V = 78; GOSUB Z
1160 W = 1; Y = Y - W; FOR S = 0 TO U
1170 V = @(S); GOSUB Z; NEXT S
1180 PRINT "READY"; K = KP; CLEAR
1190 %(20203) = 0; CALL R
1200 IF X > 21 %(20203) = 75; %(20213) = 129; CALL R
1210 CY = 0; PRINT "OK?"; IF KP # 13 GOTO 1210
1220 :PRINT ; NT = 2; PRINT ; PRINT "CLEAR"; CY = -32
1230 FOR X = R TO Y STEP Z
1240 PRINT #5, "%(", X, ")=", #6, %(X)
1250 NEXT X; PRINT ":RETURN; RUN
1260 :RETURN; .END
```

don't forget,
b means x
c means ÷

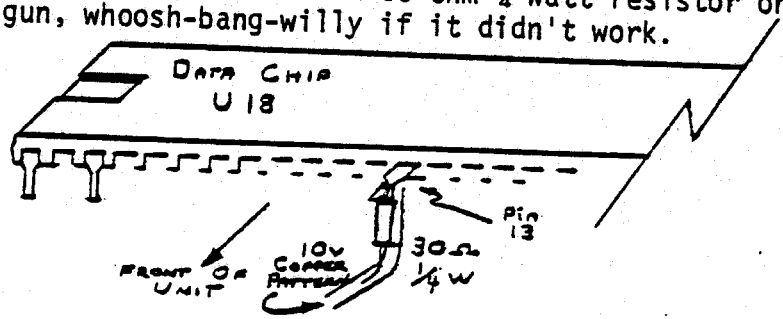
One of the modifications that have been incorporated into the AstroVision Arcade motherboard is the reduction in the power supply voltage. Dan Koppen is doing roughly the same thing to his Bally board...

## Krazy Koppen's Heat Sink

Does your machine poop out after an hour of intense game playing? Do football players get mysteriously tackled by strange dots on the screen? Or does your game just stop as you are shooting your last space invader out of the sky? Well Krazy Koppen just might have the cure for you!!

The cause of all your woes is the data chip (U18). This little babe gets so hot you can fry eggs on it and what's worse, it sits right under the keyboard which makes ventilation nearly impossible. After trying everything short of installing an attic fan, a friend of mine (who shall remain nameless (after all why should I share the credit)) claimed this chip could run on much less then the 10 volts supplied to it.

With trembling fingers I wired a 30 ohm ¼ watt resistor on pin 13, and son-of-a-gun, whoosh-bang-willy if it didn't work.




The above drawing shows exactly how I did it, with one word of caution. Don't over heat pin 13 when soldering on the resistor. You may damage the data chip and have to replace it.

I have had great luck with this fix and now can leave my machine on indefinately. Hope you have the same results!

Dan Koppen
(804)-484-1907
Suffolk, Va.

BALCHECK is the name of a program that is used to check out the operation of the motherboard at various times during the manufacture of the Bally Arcade. In the original advertising literature, a Videocade with this self-checking ability was to have been made available, but it was never done as far as I have been able to determine. But by George, one looks to be upon the very near horiz on. The program (available at $6.50) has been successfully entered into a PROM, and the plan is to make up a BALCHECK device for sale at a reasonable cost. The unit will connect to the 50-pin connector at the rear, and it is also very possible to package it into a cartridge-sized box so that it will slip into the game slot. This device will then analyze your machine, looking at a number of various items, such as keypad operation, chip functions, color display (it puts up a beautiful rainbow effect with every one of the 256 colors on the screen), and has the capability to accept machine code. If it finds some discrepancy, it uses a pair of alpha-numeric LEDs to indicate, through a code, where the problem is located. I expect Dick Belton, 4906 Willshire Ave., Baltimore, MD 21206 to have an ad in the next issue.

BALCHEK A program was developed by the Bally software engineers which 'looked at' the operation of the printed circuit board and determined if all was well, or it would identify the problem area. The program was entered into a 2716 chip, and a couple of 7-segment LED drivers and LEDs added to make up a package. All boards were inspected by this machine prior to insertion into the box. Tom Wood ran this through his disassembler and provided us with the listing, while I have added the sparse instructions. 60 pages, $7.00 (NOTE - Dick Belton 301-488-2806 can provide either the ROM cartridge or complete unit.)

TUTORIAL MAGIC REGISTER by Brett Bilbrey

The Bally Arcade has a very powerful graphics register that not very many people know about, or know how to use. Let's find out what it is, where it is, what it does, and mostly, how we can use it. But first its name: MAGIC REGISTER

The Magic Register is a hardware register (storage space) that exists in the custom chips. It is port addressable, which means it is accessed by the &(X)=Y construct from Basic, or the OUT command from machine language. It is classified as an output port which means that values may only be sent to it, never received from it. It has been assigned the value of 12decimal, or OChex.

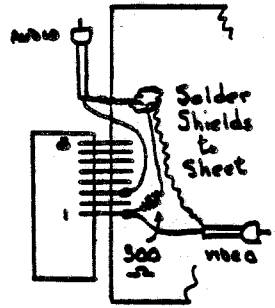
I hope to show in the following text and examples how you can make use of it. But first lets start with its principle of operation. The Magic system is enabled (set into operation) when data is written to a memory location (X) between 0 and 16383dec. or 0 to 3FFFhex. Since the first half of this area is the ROM operating system and the second half is reserved for the ROM game cartridges, we cannot write data to this area. If we try to write to this area, the Magic system knows to add 16384 to our location (X), and instead write to the new location (X+16384) the data modified by the contents of the Magic Register. The type of modification done is determined by the bits that are set in the Magic Register. The bits are assigned as follows:

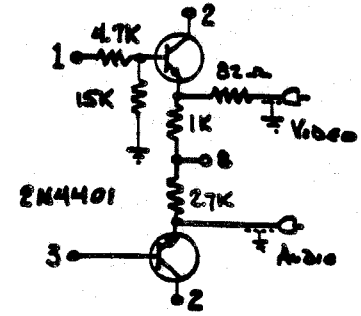
Bit 0 Least significant bit of shift amount
1 Most significant bit of shift amount
2 Rotate (not supported by our Arcade)
3 Expand
4 OR
5 XOR
6 Flop
7 Not used

The order in which Magic functions are performed is as follows: Expansion is done first, then Rotation or Shifting, then Flopping, and lastly OR or XOR. As many as four functions can be used at any one time, and any function can be bypassed. Rotate and Shift as well as OR and XOR can not be done at the same time.

Next I will give a quick explanation of the Magic functions. For a more detailed explanation I refer you to the Operating System Manual.** When using the shifter, the data will be shifted 1,2, or 3 pixels to the right, depending on the value of bits 0 and 1 of the Magic Register. Rotate can only be performed by the commercial version of the Arcade. Expand takes the 8 bits of data to be written and expands it to 16 bits by making a 0 bit into a pixel defined by bits 0 and 1 of the expand register, and a 1 bit into a pixel defined by bits 2 and 3 of the expand register. When using the expand option, it is necessary to prepare the expand register (output port 19hex or 25dec) to determine in what way a bit will be expanded into a pixel. This is useful for storing a two-color pattern in one-half the normally required memory space. The OR and XOR functions will first read the memory location (X+16384), then OR or XOR the data that we want to write there with the data that was already there, and then store the result in the location (X+16384). Flop exchanges pixel 0 with pixel 3, and pixel 1 with pixel 2.

COLOR MONITOR CIRCUIT - $0.10: We provided a circuit in Volume One which allowed the Arcade to be connected directly to a TV monitor (bypassing the rf link). It was only good for black and white. We thought... With some clues from John Perkins and a little experimentation with 100 ohm resistors, the following simple circuit developed. The Video signal is taken from the video output line and ground, and a 300 ohm resistor shunted across as well. The rf modulator must remain connected, as the signal appears, after conversion, on the input side as well as the output side. This circuit works very well with my NEC monitor (purchased for my Apple, but now...) YOUR monitor may require a bufferring circuit, and an effective one is shown:



Michael Prosise
48-G Ridge Road
Greenbelt, Maryland
20770

(301) 474-5973

# OVERCOMING LOADING PROBLEMS

**By Michael Prosise**

After having communi cated with several people across the country, it has become apparent that a fair number of BASIC users are experiencing some degree of difficulty in loading recorded programs from tape into the ASTROCADE.

I have spent considerable time researching this problem, trying to pinpoint where the problem lies and how it may be solved; I believe I have uncovered some good concrete information that will solve loading problems for most ASTROCADE owners.

According to Tom Kraus, chief repair technition at the ASTROCADE repair facility in Iowa, this has been the most frequent complaint about the new BASIC (#6004) cartridge. He reports that approximately 1 out of every 20, or 5%, of all the BASIC cartridges in the field have a problem in the input/output section. A problem cartridge will load a program off a high quality tape, such as TDK, but not from a less expensive tape or some of the computer data tapes.

Tom points out that if you are experiencing a loading problem, do not assume you have a bad BASIC. First, use a VOM (Volt-Ohm-Meter) to check the resistance of the phone cable that connects the BASIC to the cassete recorder. He states, "It must be less than 1 ohm", and a short cable is best.

He goes on to report that the quality of the cassete recorder plays a role also, a fact confirmed by ASTROCADE spokesperson Ali Pearce. When the new BASIC first came out, Ms. Pearce says she received 20 or more phone calls a day from people having problems. Originally it was thought that any recorder would work well, but as it turns out, that is not so, reports Tom.

Does ASTROCADE recommend a particular recorder? No, not presently. However, Ali Pearce says that she would like to eventually compile a list of recommended blank tapes and cassette recorders for ASTROCADE owners, but at the present time has too many other projects to work on.

After questioning several programers as to the cassette recorders they use, I gathered a few makes and models and tried them out. I had 100% success with the Radio Shack CTR-37 portable cassette recorder. It will literally load anything! Even with the "cheapo-cheapo" tapes, a volume setting of 8 (1 to 10 scale) was sufficient. Loading of the C-10 data cassettes, such as those advertised in the SOURCEBOOK, required only a volume setting of 7. All the tapes I have from several software manufactuers load without problem. I highly recommend this particular recorder. It features a record LED, (like the BASIC cartridge has), which will glow bright and steady during program recording, making recorder-to-recorder program recording very simple. It also has a counter, which makes it easy to locate programs on multi-program tapes. The CTR-37 lists for $49.95, and is on sale for $31.88 until August 23.

One other recorder reported to perform well is the Panasonic Slimline. This is the one that the folks at ASTROCADE use. I could not personally test this particular model, but several people have stated that it performs well. It lists for $39.95.

If you are experiencing some loading problems, here are a few tips and suggestions solicited from Tom Kraus, ASTROCADE, and some experienced programers:

* Keep the Arcade, recorder, and phone cable away from the T.V.'s power supply area. RF interference is possible;
* Start the recorder on PLAY, then push GO;
* Start with a volume setting of 7, and progress upwards by a half (7, 7½, 8, 8½, etc.) until you find the best setting;
* A volume setting too high will also cause a program not to load;
* Try TONE control at zero, halfway, and full. The TONE control can really help when using a poorer quality cassette. Likewise, if you are using a high quality tape, too much TONE can cause trouble too;
* Keep tape recorder heads clean and demagnetized;
* Do not use a battery operated tape recorder. Use a battery eliminator. Even good new batteries can cause wow and flutter;
* Do not record programs close to the beginning of a tape. Start after 20 seconds of blank tape, since tape stretches sometimes after repeated rewinding.

If none of the above helps, try the Radio Shack CTR-37. If that does not help, you very likely have a bad BASIC cartridge. Send it to the ASTROCADE repair facility.

I hope this report proves helpful. If you own a recorder that you "swear" by, let Bob F. know, and perhaps we will publish a list of brands with the most votes. If I can be of any assistance, feel free to contact me.

***"CHRDIS"***
BY MIKE SKALA

I'VE SEEN QUITE A BIT OF SOFTWARE LATELY UTILIZING THE "GRAPHIC CHARACTER MAKER", A MACHINE CODE ROUTINE THAT "ARCADIAN" HAS PUBLISHED IN THE PAST YEAR. THIS ALLOWED US TO USE A DISPLAY ROUTINE FROM THE ON-BOARD ROM AND PUT COMPLEX GRAPHICS ON THE SCREEN INSTANTLY, RATHER THAN A SLOW SERIES OF BOX AND LINE COMMANDS. THE MAJOR DRAWBACK HERE WAS WHEN MOVING THE GRAPHICS, ERASING AND REDRAWING LEFT US WITH CONSIDERABLE FLASHING OR BLINKING. IF YOU HAVE BEEN WITH US FOR A WHILE, YOU KNOW THAT WE ARE CONTINUALLY EVOLVING AND IMPROVING. THE FOLLOWING TUTORIAL IS OUR NEW GENERATION OF SCREEN ANIMATION FOR THE ASTROCADE!!!

ON-BOARD SUBROUTINE #51,"CHRDIS",IS A SINGLE CHARACTER DISPLAY ROUTINE MUCH AKIN TO THE ROUTINE USED IN THE GRAPHIC CHARACTER MAKER. WE CALL THE ROUTINE AS FOLLOWS:

```
 DEC  HEX
 255   FF   SYSSUK
  51   33   CHRDIS
   N    N   E-HOR POS. (0-159 DEC.)
   N    N   D-VER POS. (0-99 DEC.)
   N    N   C-CHAR DISPLAY PARAMETER
   N    N   A-CHAR TO CALL
```

SYSSUK/CHRDIS--ANY ON-BOARD SUBROUTINE CAN BE CALLED IN ONE OF TWO WAYS:"SYSTEM" ASSUMES ALL NECCESARY INFO IS ALREADY IN THE REGISTERS. (MEMORY CELLS WITHIN THE C.P.U.) "SYSSUK" MEANS THE INFO WILL BE DIRECTLY FOLLOWING THE CALL, AND BE "SUCKED" IN.

E--THIS REGISTER IS THE HORIZON TAL COORDINATE FOR YOUR GRAPHIC. ZERO IS THE LEFT SIDE OF THE SCREEN, AND 159 IS THE RIGHT SIDE. IF YOU GO PAST 159, THE GRAPHIC WILL REAPPEAR ON THE LEFT SIDE, ONE PIXEL LOWER. THIS IS FINE UNTIL YOU REACH 256, ABOUT HALF WAY ACROSS THE SCREEN, WHERE IT WILL AGAIN DROP ONE PIXEL DOWN AND BACK OVER TO THE LEFT SIDE OF THE SCREEN. FOR THIS REASON, IT IS BEST TO LIMIT-CHECK THE GRAPHIC TO BETWEEN 0 AND 159.

D--THIS REGISTER IS THE VERTICAL COORDINATE. ZERO IS THE TOP OF THE SCREEN, 99 THE BOTTOM. BE CAREFUL NOT TO RUN YOUR GRAPH DOWN INTO THE SCRATCHPAD AREA HIDDEN AT THE BOTTOM OF THE SCREEN. LIMIT-CHECK YOUR GRAPHIC AGAIN NOT TO RUN OFF THE BOTTOM, OR THE WHOLE PROGRAM MIGHT CRASH!!
HERE'S SOMETHING INTERESTING: THERE IS A WHOLE OTHER SCREEN ABOVE THE ONE WE NORMALLY SEE. YOU'LL NEVER SEE IT, BUT YOU CAN MOVE THINGS AROUND UP THERE WITH NEGATIVE VALUES IN THIS REGISTER. YOU COULD, FOR EXAMPLE, START A GRAPHIC UP THERE AND HAVE IT FALL DOWN INTO THE VISIBLE SCREEN.

C--THIS CONTROLS A LOT OF INTERESTING THINGS, AND WE'LL GO INTO DETAIL IN A FUTURE ARTICLE. FOR NOW USE 40 (28 HEX), WHICH GIVES US AN XOR WRITE.

A--THIS IS WHICH CHARACTER WE ARE GOING TO DISPLAY. IT RESPONDS TO THE STANDARD ASCII CODE TO DISPLAY ALL RESIDENT CHARACTERS, 0 THROUGH 127, OR OUR OWN THAT WE CAN CREATE.

NOW THAT YOU UNDERSTAND ALL THAT STUFF, LET'S ASSEMBLE A SMALL MACHINE CODE PROGRAM AT THE BACK END OF OUR LINE INPUT BUFFER. THE FIRST THING YOU MUST DO WHEN GOING INTO A MACHINE CODE PROGRAM FROM BASIC IS TO SAVE THE DE REGISTER. THIS IS A LITTLE MEMORY CELL THAT REMEMBERS WHERE YOU WERE BEFORE YOU LEFT. NOW WE GO INTO OUR CHRDIS ROUTINE. NOW COMES THE NEW TWIST: WE GO RIGHT INTO ANOTHER CHRDIS, BRING BACK OUR DE REGISTER, AND RETURN TO BASIC. THERFORE, IF WE ALREADY HAVE OUR GRAPHIC ON THE SCREEN, THIS WILL ERASE THE OLD AND REDRAW THE NEW IN ONE CALL, YIELDING MINIMUM "OFF" TIME.

ENTER THE FOLLOWING DIRECT COMMAND (NO LINE NUMBER), AND THEN INPUT THE DECIMAL VALUES LISTED BELOW:

```
NT=1:FOR A=20241TO 20257:PRINT A;:INPUT
"   ";B:%(A)=B:PRINT :NEXT A
          DEC  HEX
%(20241)=213   D5  PUSH DE
%(20242)=255   FF  SYSSUK
%(20243)= 51   33  CHRDIS
%(20244)=  0    0  E (HOR.)
%(20245)=  0    0  D (VER.)
%(20246)= 40   28  C
%(20247)=  0    0  A (CHAR#)
%(20248)=  0    0  NOP (NO OPERATION)
%(20249)=255   FF  SYSSUK
%(20250)= 51   33  CHRDIS
%(20251)=  0    0  E (HOR.)
%(20252)=  0    0  D (VER.)
%(20253)= 40   28  C
%(20254)=  0    0  A (CHAR#)
%(20255)=  0    0  NOP
%(20256)=209   D1  POP DE
%(20257)=201   C9  RET
```

NOW WE NEED A LITTLE BASIC PROGRAM TO MANIPULATE ALL THIS STUFF. ENTER THE FOLLOWING:

```
10 %(20244)=-9999:V=0:H=0
20 V=V-JY(1):H=H+JX(1)
30 IF V<0V=0
40 IF V>75V=75
50 IF H<0H=0
60 IF H>159H=159
70 %(20251)=Vx256+H
80 C=&(28)÷65+94:%(20254)=C
90 CALL20241:%(20244)=%(20251):%(20247)=C:
GOTO 20
```

LINE 10:WE SET THE COORDINATES FOR THE FIRST CHRDIS UPSTAIRS SOMEWHERE OUT OF SIGHT, AND THEN THE SECOND CHRDIS WILL DRAW THE INITIAL GRAPHIC. THIS LINE IS ONLY RUN ONCE, AFTER WHICH THE PROGRAM WILL USE THE FIRST CHRDIS TO ERASE THE OLD GRAPHIC. LETTER VARIABLES V AND H ARE ALSO ZEROED OUT HERE.

LINE 20:PROGRAM READS THE JOYSTICKS TO UPDATE VARIABLES USED FOR VERTICAL AND HORIZONITAL COORDINATES.

LINES 30-60:LIMIT CHECKS THE GRAPHIC TO KEEP IT ON THE SCREEN.

LINE 70:HERE IS WHERE THE NEW COORDINATES ARE PLUGGED INTO OUR SECOND CHRDIS. SINCE ANY POKE WORKS ON TWO MEMORY LOCATIONS, WE MUST USE THE FORMAT Vb256+H.

LINE 80:FN(I) IS READ, AND VARIABLE C WILL BE SET TO 94,95,96 OR 97. THESE ARE THE ASCII VALUES FOR THE FOUR ARROWS FOUND ON OUR KEYPAD. THIS VALUE IS THEN POKED INTO THE CHAR# POSITION OF OUR SECOND CHRDIS. IN CASE YOU'VE WONDERED WHY A "NOP" FOLLOWED THE A REGISTER IN BOTH CHRDIS'S, IT'S AGAIN BECAUSE OF OUR POKE SITUATION. A NOP IS A MACHINE CODE COMMAND THAT DOES ABSOLUTELY NOTHING EXCEPT WASTE A BYTE. WERE IT NOT THERE IN THIS PROGRAM, WE SIMPLY COULD NOT POKE IN OUR ASCII VALUE. WE WOULD INSTEAD END UP WITH SOME LARGE NEGATIVE NUMBER TO POKE IN, AND THE FIRST AND SECOND CHRDIS'S WOULD REQUIRE DIFFERENT FUDGE FACTORS.

LINE 90:THE MACHINE CODE PROGRAM IS NOW CALLED, THE OLD COORDINATES ARE SET EQUAL TO THE NEW ONES, THE OLD CHAR# IS SET EQUAL TO THE NEW ONE, AND THE PROGRAM LOOPS BACK TO LINE 20.

I WOULD SUGGEST THAT YOU DUMP THE PROGRAM TO TAPE BEFORE YOU RUN IT, BECAUSE ONE LITTLE MISTAKE WITH MACHINE CODE CAN CAUSE BIG PROBLEMS. TO DUMP, USE THE FOLLOWING COMMAND:

```
:PRINT ::PRINT %(20241),17
```

TO LOAD IT BACK FROM TAPE, USE:

```
:INPUT::INPUT %(20241)
```

RUN THE PROGRAM AND FIDDLE WITH THE JOYSTICK AND KNOB. I'LL ADMIT IT ISN'T A GAME OR LOTS OF FUN, BUT IT DEMONSTRATES A FAIRLY SIMPLE MEANS OF SMOOTH ANIMATION. THIS WILL WORK WITH HOMEMADE CHARACTERS AS WE DID WITH THE GRAPHIC CHARACTER MAKER, AND THIS WILL BE COVERED IN THE NEXT TUTORIAL.

ONE MAJOR PRECAUTION MUST BE OBSERVED WITH THIS ROUTINE. WHEN USING THE NEW ASTRO BASIC, THE PROGRAM ACTIVELY USES THE 104 BYTE LINE INPUT BUFFER. SINCE WE HAVE STORED OUR MACHINE CODE IN THE LAST 17 BYTES OF THE BUFFER, NO LINE IN OUR BASIC PROGRAM CAN BE MORE THAN 97 BYTES LONG!!!!!

YOU WILL NOTICE A SLIGHT FLICKER OF OUR GRAPHIC. THIS HAS SOMETHING TO DO WITH TIMING OR SCREEN INTERRUPTS OR SOMETHING. I DON'T KNOW I'M SURE SOMEONE OUT THERE KNOWS THE CURE. SO PLEASE WRITE IN!!

VARIATIONS:

```
"WRAPAROUND"
30 IF V<0V=75
40 IF V>75V=0
50 IF H<0H=159
60 IF H>159H=0

"BIG TIME"
5 %(20246)=163:%(20253)=163
40 IF V>65V=65

"LETTER DROP"
80 C=&(28)c3+33:%(20254)=C
90 CALL20241:%(20244)=%(20251):%(20247)=C: IF
TR(1)GOTO 10
100 GOTO 20
```

MIKE SKALA
544 E. OVERLOOK
EASTLAKE, OH
44094

# CHRDIS II

## HOW TO USE HOME MADE GRAPHICS

BY MIKE SKALA

BUILDING YOUR OWN GRAPHICS FOR USE WITH OUR CHRDIS ROUTINE IS A FAIRLY SIMPLE TASK. THE "GRAPHIC CHARACTER MAKER" PUBLISHED IN THE ARCADIAN (VOL.3 PP.82-84) COULD BE MODIFIED READILY IF YOU KNOW WHAT YOU ARE DOING. IF YOU DON'T, THEN READ ON...

THE "CHRDIS" WILL LOOK AT YOUR CHARACTER IN BLOCKS ONE PIXEL HIGH BY EIGHT PIXELS WIDE, WITH EACH PIXEL BEING EITHER "OFF" (BC) OR "ON" (FC). YOU MUST FIGURE OUT THE VALUE OF EACH BLOCK BY TOTALLING THE "PIXEL VALUES"

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

FIG. 1

(REFER TO FIG.1). IF A PIXEL IS "ON", IT'S VALUE IS ADDED TO THE TOTAL. FOR EXAMPLE, ALL EIGHT PIXELS "ON" WOULD HAVE A BLOCK VALUE OF 255, ALL "OFF", A VALUE OF ZERO, OR JUST THE FOUR ON THE RIGHT HAND SIDE "ON" WOULD EQUAL 15.(8+4+2+1=15) LET'S CREATE A SMALL GRAPHIC TO ILLUSTRATE. LOOK AT FIG.#2 TO SEE HOW WE GOT OUR BLOCK VALUES . YOU CAN GO EITHER HIGHER, WIDER, OR BOTH, AND BLOCK VALUES WILL BE READ FROM LEFT TO RIGHT,(IF MORE THAN ONE BLOCK WIDE), AND TOP TO BOTTOM.

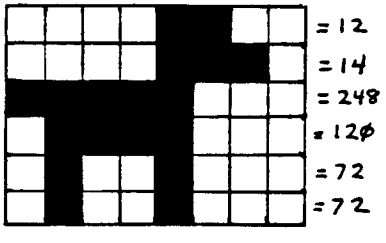



FIG. 2

SO NOW THAT YOU HAVE SOME BLOCK VALUES, WHERE DO THEY GO? WELL, WITHOUT EXTERNAL MEMORY, (E.G. A BLUE RAM, ETC.), YOU HAVE TWO CHOICES. EITHER WE STORE THEM IN THE LINE INPUT BUFFER WITH THE REST OF OUR MACHINE CODE, OR AT THE VERY BOTTOM OF OUR SCREEN. THE FORMER GETS RATHER CROWDED IN A HURRY, THE LATTER CAN GET WIPED OUT BY CLEARING THE SCREEN OR RUNNING GRAPHICS INTO THE BOTTOM. DIRECTLY FOLLOWING THE LINE INPUT BUFFER IS A MEMORY AREA CALLED THE "STACK". THIS IS SORT OF A "PARKING LOT" FOR BASIC TO STORE AND RETRIEVE DATA. SELDOM DOES THIS AREA GET FILLED UP, SO WE CAN GENERALLY RUN A FEW DOZEN BYTES INTO THIS AREA WITHOUT PROBLEMS. THE BEST APPROACH HERE IS TO PUT OUR GRAPHIC INFO IN THE DEEP END, AND OUR MACHINE CODE ROUTINE UP IN THE SAFE END. THIS WAY, IF THE STACK RUNS OVER OUR GRAPHICS, WE GET FUNNY LOOKING CHARACTERS, WHEREAS RUNNING OVER OUR MACHINE CODE ROUTINE WOULD CAUSE OUR PROGRAM TO BOMB.

WHAT WE ARE DOING HERE IS CREATING AN ALTERNATE CHARACTER FONT WITH THE MUTT BEING OUR FIRST AND ONLY CHARACTER. WE START OUR LIST OF CHARACTERS WHERE ASCII CODES END, SO HE WILL BE CHARACTER NUMBER #128. TO USE THIS FONT, WE NOW HAVE A NEW RESPONSIBILITY. WE MUST CONSTRUCT A TABLE IN MACHINE CODE THAT TELLS OUR COMPUTER ALL ABOUT THIS NEW FONT, AND LET IT KNOW WHERE WE HID THIS TABLE. LOAD THE DECIMAL VALUES BELOW WITH THIS DIRECT COMMAND:

```
FOR A=20237TO 20270;CY=0;PRINT A,;
INPUT "    ",%(A);BOX 0,0,160,20,2;
NEXT A
```

```
          DEC  HEX
%(20237)=221   DD   LOAD IX WITH ADDR
%(20238)= 33   21   LD IX,nn  OF OUR ALTERNATE
%(20239)= 34   22   n         FONT TABLE.
%(20240)= 79   4F   n         (20258)
%(20241)=213   D5   PUSH DE
%(20242)=255   FF   SYSSUK
%(20243)= 51   33   CHRDIS
%(20244)= 00   00   E (HOR)
%(20245)= 00   00   D (VER)
%(20246)= 40   28   C              SAME AS
%(20247)= 00   00   A (CHAR#)      "CHRDIS"
%(20248)= 00   00   NOP
%(20249)=255   FF   SYSSUK
%(20250)= 51   33   CHRDIS
%(20251)= 00   00   E (HOR)
%(20252)= 00   00   D (VER)
%(20253)= 40   28   C
%(20254)= 00   00   A (CHAR#)
%(20255)= 00   00   NOP
%(20256)=209   D1   POP DE
%(20257)=201   C9   RET
%(20258)=128   80 - # OF OUR ALT. CHARACTER
%(20259)= 00   00   UPDATE VALUES
%(20260)= 00   00   (WE DON'T USE 'EM)
%(20261)=  1   01 - CHR SIZE WIDTH (x8 PIXELS-1 BLOCK)
%(20262)=  6   06 - CHR SIZE HEIGHT
%(20263)= 41   29   STARTING ADDR. OF
%(20264)= 79   4F   BLOCK VALUES (20265)
%(20265)= 12   0C
%(20266)= 14   0E
%(20267)=248   F8   (BLOCK VALUES
%(20268)=120   78   FOR OUR PUP.)
%(20269)= 72   48
%(20270)= 72   48
```

NOW ENTER THE FOLLOWING BASIC PROGRAM:

```
>10 %(20244)=-9999;V=0;H=0;C=128;%(20247)=C;
%(20254)=C
>20 V=V-JY(1);H=H+JX(1)
>30 IF V<0V=0
>40 IF V>82V=82
>50 IF H<0H=0
>60 IF H>152H=152
>70 %(20251)=V*256+H
>80 CALL20237
>90 %(20244)=%(20251)
>100 GOTO 20
```

THE FOLLOWING PARAGRAPH & MODIFICATIONS WERE ACCIDENTALLY LEFT OUT OF "CHRDIS II" IN LAST MONTH'S ISSUE:

NOTICE THAT OUR CALL IS MADE NOW TO %(20237) SO AS TO LOAD THE IX REGISTER WITH OUR ALT. FONT TABLE ADDRESS. SINCE YOU ARE GOING TO WANT BIGGER AND/OR MORE CHARACTERS, YOU MAY WANT TO GO DEEPER INTO THE STACK AREA, OR EXPERIMENT WITH STORING STUFF AT THE BOTTOM OF THE SCREEN. JUST BE SURE TO LOAD THE ALT. FONT TABLE WITH THE PROPER CHR HEIGHT & WIDTH AND THE CORRECT ADDRESS OF YOUR BLOCK VALUES.

MAKE YOUR HOUND HAPPY WITH THE FOLLOWING MODIFICATIONS:

```
%(20271)=12    ADD LINES: >25 C=C+1;IF C>129
%(20272)=142                  C=128
%(20273)=120              >26 %(20254)=C
%(20274)=120              >95 %(20247)=
%(20275)=72                   %(20254)
%(20276)=72
```

* * *

# CHRDIS III

BY MIKE SKALA

CHARACTER DISPLAY PARAMETERS

THIS IS THE THIRD AND FINAL SEGMENT IN MY THREE PART SERIES EXPLAINING THE ON-BOARD SUBROUTINE "CHRDIS". WE WILL FINALLY LOOK AT THE "C" BYTE WITHIN "CHRDIS", WHICH IS AN OPTION BYTE USED TO CONTROL THREE SEPERATE FUNCTIONS IN ALL SYSTEM ALPHANUMERIC DISPLAY ROUTINES: SIZE, TYPE OF SCREEN WRITE, AND COLOR.

FIRST WE BETTER BE SURE WE UNDERSTAND HOW IMAGES AND OUR BASIC TEXT ARE USING OUR RAM TO ENSURE WE DON'T LET THEM CONFLICT. OUR TEXT IS STORED IN THE EVEN NUMBER BITS OF RAM, THE GRAPHICS CAN THEN USE ONLY THE ODD BITS. THAT MEANS WE CAN SET AN ODD BIT TO ONE, OR TURN IT ON (FC), OR WE CAN RESET THE BIT TO ZERO, OR TURN IT OFF (BC). SINCE THESE ONBOARD SUBROUTINES WERE DESIGNED FOR FOUR COLOR IMAGES, THEY ARE ALWAYS WORKING WITH TWO BITS AT A TIME, I.E., THEY ARE GOING TO LOOK AT THE GRAPHICS THAT WERE BUILT IN A ONE BIT PER PIXEL FORMAT AND EXPAND THEM INTO TWO BITS PER PIXEL. FIG.#1 SHOWS A TYPICAL BYTE OF RAM, AND FIG.#2 SHOWS THE FOUR POSSIBLE COMBINATIONS WE CAN EXPAND INTO. WE HAVE THREE DIFFERENT WAYS WE CAN DO A WRITE: 'PLOP', 'OR', AND 'XOR'. 'PLOP' (FIG.#3) WILL SIMPLY REPLACE OUR RAM VALUE WITH OUR IMAGE VALUE. YOU CAN SEE THAT THIS WILL DESTROY OUR TEXT BITS, SO FORGET ABOUT 'PLOPPING'. AN 'OR'(FIG.#4) WILL FIRST LOOK AT THE RAM, AND THE RESULT WILL BE 'ON' IF EITHER OR BOTH THE IMAGE BIT AND RAM BIT WAS ON. IF BOTH WERE OFF, THE RESULT WILL BE OFF. AN 'XOR' (FIG.#5 A&B) IS AN EXCLUSIVE 'OR'. THIS WILL SET THE RESULT IF THE IMAGE BIT OR THE RAM BIT WAS ON, BUT RESET IF BOTH WERE ON. THIS IS WHAT WE'VE BEEN USING IN THE PREVIOUS TUTORIALS, WHERE YOU 'XOR' AN IMAGE TO A BLANK AREA AND THE GRAPHIC APPEARS, THEN 'XOR' IT TO THE SAME PLACE AND IT IS ERASED. I HOPE YOU CAN SEE BY NOW THAT IF WE EXPAND INTO 00 OR 10 AND USE 'OR' & 'XOR' WE CAN STILL WRITE TO THE SCREEN WITHOUT DISTURBING OUR TEXT. IF THIS IS STILL CONFUSING, JUST REMEMBER NOT TO EXPAND INTO 01 OR 11 AND DON'T 'PLOP'. YOU MAY ALSO WISH TO LEARN MORE ABOUT BINARY LOGIC (OR, XOR,ETC.) AS IT IS USED QUITE A BIT IN MACHINE CODE.

FIG. #1

| BIT# | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | G | T | G | T | G | T | G | T |

FIG #2

| | |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

PLOP
FIG. 3

| | | |
|---|---|---|
| IMAGE | 1 | 0 |
| RAM | 0 | 1 |
| RESULT | 1 | 0 |

OR
FIG.4

| | | |
|---|---|---|
| I | 1 | 0 |
| R | 0 | 1 |
| RES. | 1 | 1 |

XOR
FIG.5

A

| | |
|---|---|
| 1 | 0 |
| 0 | 1 |
| 1 | 1 |

B

| | | |
|---|---|---|
| I | 1 | 0 |
| R | 1 | 1 |
| RES. | 0 | 1 |

LET'S PROCEED. FIG.#6 SHOWS HOW THE 'C' BYTE IS CONSTRUCTED. STARTING BACKWARDS (AS ALWAYS) BITS 1 & 0 ARE WHAT AN OFF BIT IN OUR GRAPHIC WILL BE EXPANDED INTO (00 HERE WILL SET IT EQUAL TO BC). BITS 3 & 2 ARE WHAT AN ON BIT IS EXPANDED INTO (10 WILL EQUAL FC). YOU COULD SET BITS 3 & 2 TO 00, AND BITS 1 & 0 TO 10 AND GET A "REVERSE" IMAGE.

FIG. 6

| ENLARGE FACTOR | XOR | OR | ON COLOR | OFF COLOR |
|---|---|---|---|---|
| 7 6 | 5 | 4 | 3 2 | 1 0 |

BITS 5 & 4 CONTROL THE TYPE OF SCREEN WRITE. DON'T SET EITHER AND YOU'LL GET A 'PLOP' (A NO-NO), SET BIT 5 TO GET AN 'XOR', SET BIT 4 TO GET AN 'OR'. I THINK WE'VE KILLED THIS SUBJECT ALREADY.

BITS 7 & 6 CONTROL A CUTE TRICK WE'VE SEEN AS FAR BACK AS VOL. #1 OF THE ARCADIAN, AND IN NEARLY EVERY GAME CARTRIDGE. THIS ALLOWS US TO DISPLAY ANY CHARACTER (ASCII OR HOMEMADE) IN NORMAL SIZE OR IN AN ENLARGED FASHION. (SEE FIG.#7). TWO THINGS TO REMEMBER HERE; THE ADDRESS OR LOCATION OF THE CHARACTER ALWAYS REFERS TO THE UPPER LEFT HAND CORNER, NOT THE CENTER, AND KEEP YOUR IMAGE OUT OF THE SCRATCHPAD AREA AT THE BOTTOM OF THE SCREEN.

FIG. 7

| BIT# 7 | BIT# 6 | FINAL SIZE |
|---|---|---|
| 0 | 0 | NORMAL |
| 0 | 1 | 2X |
| 1 | 0 | 4X |
| 1 | 1 | 8X |

SO JUST BUILD THIS "C" BYTE IN BINARY AND CONVERT TO DECIMAL, LIKE WE CONVERTED OUR BLOCK VALUES FOR OUR PUPPY IN THE LAST ARCADIAN.BE SURE TO PUT THE PROPER "C" BYTE IN BOTH THE "CHRDIS" IN YOUR MACHINE CODE ROUTINE

WELL, THIS SHOULD GIVE YOU HACKERS SOME HANDY TOOLS TO GENERATE SOME PRETTY CLASSY PROGRAMMING. IF YOU HAVE ANY FURTHER QUESTIONS YOU CAN CONTACT ME DIRECT BY MAIL OR PHONE (EVENINGS). ALSO, DON'T BE AFRAID TO SHOW APPRECIATION BY SENDING A FEW OF YOUR PROGRAMS. I'D LOVE TO SEE THEM!!!

MIKE SKALA 544 E. OVERLOOK
EASTLAKE, OHIO 44094 (216) 951-2564

SCROLLING II: We issued a challenge last year, to develop programs that would scroll the screen in different directions. In Volume 4, p.98 and 105, we presented some schemes to shift the screen sideways. Now we have a general method to scroll the screen in any of the four directions using four subroutines. (We will provide the other three routines in future issues.) The method provides a routine to be loaded in any unused area of the system RAM or add-on memory. In this issue we will scroll the screen downward, using the bottom of the AstroBasic stack, as an example. load as follows:

```
FOR A=20258 TO 20297; INPUT %(A); NEXT A
```

Then input the decimal values in the second column.

To scroll once, type in CALL 20258

To scroll n pixels, type in

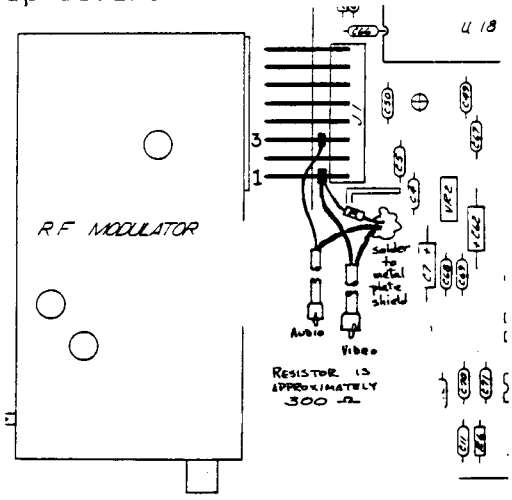
```
  FOR A= 1 TO n; CALL 20258; NEXT A
```

SUBROUTINE TO SCROLL SCREEN DATA 1 PIXEL DOWN

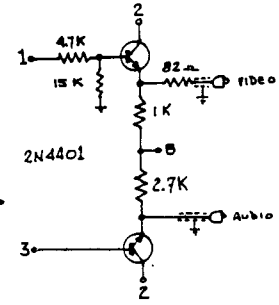
| BYTE # | DECIMAL | HEX | MNEMONICS | LABELS | COMMENTS |
|---|---|---|---|---|---|
| 0 | 213 | D5 | PUSH DE | START | Save BASIC pointer. |
| 1 | 175 | AF | XOR A | | Clear the A register. |
| 2 | 33 | 21 | LD HL, nn | | Set pointer to bottom |
| 3 | 15 | 0F | | | of screen RAM. |
| 4 | 78 | 4E | DW $19983_{10}$ | | |
| 5 | 17 | 11 | LD DE, nn | | Address of next byte up. |
| 6 | 231 | E7 | | | |
| 7 | 77 | 4D | DW $19943_{10}$ | | |
| 8 | 14 | 0E | LD C, n | | Set # lines to scroll. |
| 9 | 89 | 59 | DB $89_{10}$ | | |
| 10 | 6 | 06 | LD B, n | LOOP 2 | Set # bytes per line |
| 11 | 40 | 28 | DB $40_{10}$ | | to scroll. |
| 12 | 126 | 7E | LD A, (HL) | LOOP 1 | Get a byte. |
| 13 | 230 | E6 | AND n | | Mask out screen data. |
| 14 | 85 | 55 | DB $85_{10}$ | | |
| 15 | 119 | 77 | LD (HL), A | | Save program data. |
| 16 | 26 | 1A | LD A ,(DE) | | Get next byte. |
| 17 | 230 | E6 | AND n | | Mask out program data. |
| 18 | 170 | AA | DB $170_{10}$ | | |
| 19 | 174 | AE | XOR (HL) | | Move down one line and mix |
| 20 | 119 | 77 | LD (HL), A | | with program there. |
| 21 | 27 | 1B | DEC DE | | Next pair of |
| 22 | 43 | 2B | DEC HL | | bytes' addresses. |
| 23 | 16 | 10 | DJNZ, dis | | Decrement B; if not 0, |
| 24 | 243 | F3 | DB $-13_{10}$ | | jump to LOOP 1; else |
| 25 | 13 | 0D | DEC C | | Decrement C; |
| 26 | 32 | 20 | JRNZ, dis | | if C#0, jump to |
| 27 | 238 | EE | DB $-18_{10}$ | | LOOP 2; else |
| 28 | 209 | D1 | POP DE | | Restore BASIC pointer. |
| 29 | 201 | C9 | RET | | Return to BASIC. |

MONITOR CONNECTION The latest TV's now on the market are including a set of 'MONITOR' input jacks (TV and Audio) for direct attachment of a VCR and such. You can connect your Astrocade here and take advantage of the clearer graphics that result. The changes in the Arcade unit are minimal. (This addition was originally printed in Vol. 4, p.62)

Remove the screws on the bottom so that the upper plastic cover can be removed. At the lover left you will see a large metal box with a number of heavy pins at its upper right. Temporarily remove the box by pulling up at the left side until the box is clear of its mount and then pull the box to the left, off the big pins. You have to make three solder connections to two of the pins and the flat metal shield, as shown in the diagram. When complete, push the box back onto the pins until it can be pressed down into the lower cover. Lead the new wires out the back of the unit. Replace the top cover.




I experimented to find the value of the 300 ohm resistor. This scheme is working with my NEC, but you might want to try the buffer circuit shown



# CHARACTER DISPLAY IN EXTENDED BASIC A TUTORIAL BY GEORGE MOSES

RECENTLY, I TRIED TO CALL THE GAME OVER ROUTINE USING THE BLUE RAM EXTENDED BASIC AFTER ALL, THE ROUTINE EXISTS BURIED IN THE ASTROCADE'S 8-K OF ON-BOARD ROM JUST WAITING FOR ME TO INVOKE IT. INCREDIBLE WIZARD THAT I AM. SO OFF I GO, CONJURING UP THE BIG CALL FEVERISHLY TYPING HEX CODES TO DISPLAY CHARACTERS ENLARGED 8X. BUT I GOT STRIPED BOXES WHERE THE LETTERS "GAME OVER" WERE SUPPOSED TO BE. BUT THE ROUTINE WORKED PERFECTLY WHEN I DISPLAYED THE CHARACTERS IN THEIR 1-X SIZE. THE SOLUTION? I ASKED JAY FENTON WHEN HE WAS IN TOWN FOR OUR ANNUAL FISHING TRIP.

JAY SAYS "IF ENLARGED CHARACTERS ARE TO BE DISPLAYED THE STACK POINTER MUST BE RELOCATED TO SCREEN RAM TO ALLOW MAGIC RAM TO OPERATE THE EXPANDER. THIS ROUTINE WILL DO THE JOB."

## THE ROUTINE:

```
8000 F5      ;PUSH AF (SAVE REGISTERS)
8001 C5      ;PUSH BC
8002 D5      ;PUSH DE
8003 E5      ;PUSH HL
8004 210000  ;LD HL,0000  CLEAR HL
8007 39H     ;ADD HL,SP  SAVE OLD STACK
8008 31FF4F  ;LD SP,4FFF MOVE SP TO SCREEN
800B E5      ;PUSH HL SAVE HL ON STACK
800C FF      ;RST 38H PREPARE FOR SUB CALL
800D 33H     ;SUBROUTINE 32+1 (SYSSUCK)
800E 3C      ;REGISTER E HORIZ. POSITION
800F 14H     ;REGISTER D VERTICAL POSITION
8010 DC      ;REGISTER C DISPLAY OPTION BYTE
             ;ENLARGE 8X, OR IT TO THE SCREEN
             ;TO ENLARGE 2X USE 5C; FOR 4X
             ;USE 9C; TO XOR IMAGE USE:
             ;8X=EC, 4X=AC, 2X=6C
8011 41H     ;HEX-ASCII CODE FOR LETTER "A"
             ;SUCKED INTO REGISTER A
8012 E1      ;POP HL  RETRIEVE OLD STACK ADDR
8013 F9      ;LD SP,HL  REESTABLISH STACK
8014 E1      ;POP HL  RESTORE ALL REGISTERS
8015 D1      ;POP DE
8016 C1      ;POP BC
8017 F1      ;POP AF
8018 C9      ;RET     GO BACK TO BASIC
```

I TYPED THIS PROGRAM IN AT 8000H USING THE NEW BLUE RAM UTILITY 2.3 JUST RELEASED BY PERKINS ENGINEERING. THE UTILITY IS AN 8K MACHINE LANGUAGE PROGRAM THAT RESIDES FROM 6000H TO 7FFF. IT HAS, AMONG OTHER FEATURES, A FULL SCREEN EDITOR OF OPCODES AND A NICE DISSASSEMBLER. SO, AFTER TYPING IN THE ABOVE HEX CODES USING THE SCREEN EDITOR, ALL I HAD TO DO WAS GIVE THE COMMAND "DASM" 8000H AND THE DISSASSEMBLER PRODUCED THE PRINTOUT YOU SEE ABOVE. IT WILL ALSO DISSASSEMBLE A CARTRIDGE OR ANY OTHER MACHINE LANGUAGE PROGRAM THE SAME WAY!

THE ROUTINE TO MOVE THE STACK POINTER SHOWN ABOVE IS RELOCATABLE TO ANY FREE RAM YOU WISH TO LOAD IT INTO. ALL YOU HAVE TO DO TO USE IT IS TO CALL IT FROM BASIC. IN THE CASE OF THE ABOVE ROUTINE STASHED AT 8000H YOU WOULD USE CALL!8000

BLUE RAM BASIC CORNER...

# A FEW WELL AIMED POKES

BY CLYDE PERKINS

THERE HAS BEEN SOME CONFUSION AMONG OWNERS OF 16K- AND 32K-BLUE RAMS CONCERNING THE USE OF PROGRAMMING AREAS IN THE EXPANDED MEMORY. THE ORIGINAL UNITS (1980-81) CONTAINED 4K OF NEW MEMORY AND, OF COURSE, THE IO PORTS. THE EXTENDED BASIC CARTRIDGE PROVIDED 2000 BAUD TAPING, AND ALSO RESERVED 996 BYTES OF RAM FOR STACK, VARIABLES AND A FEW REGISTERS. WHEN THE BLUE RAM WAS REDESIGNED WITH 16K OR 32K OF MEMORY, THE DEDICATED AREA (!6C1D TO !6FFF) WAS MAINTAINED FOR THE SAKE OF COMPATABILITY WITH B.R. PROGRAMS, AND IT REQUIRES A FEW TRICKS TO "DUCK" AROUND THE AREA WHEN WRITING PROGRAMS LARGER THAN 3100 BYTES (!6000 TO !6C1C). THE EASIEST WAY TO DO THAT IS TO START YOUR PROGRAM AT !7000 OR !8000. NOTE: THE"!" MEANS HEXADECIMAL NOTATION (BASE 16).

INSTRUCTIONS SUPPLIED WITH THE NEW RAMS INCLUDED THE POKES NECESSARY FOR STARTING AT !7000, BUT MOST OF US CAREFULLY POKED THEM IN AND WERE HAPPY IF THEY WORKED. HOWEVER, IF YOUR PROGRAM CONTAINS A LARGE FOR-NEXT LOOP YOU MAY RUN INTO TROUBLE AS IT APPROACHES !8000. AS WITH ALL COMPUTERS, YOUR ARCADE HAS A VERY STUBBORN STREAK. IT REQUIRES PRECISE PROTOCOL AND ALSO TRIES TO IMPOSE LIMITATIONS SUCH AS MAXIMUM ADDRESS OF !7FFF OR 32767 DECIMAL. !8000 IS CONSIDERED A NEGATIVE NUMBER BY THE ARCADE(-32768), !8001=-32767, !8002=-32766, ETC. THE BALLY AND ASTRO-BASIC CARTRIDGES AVOIDED THIS TROUBLE BY IMPOSING THE SAME LIMIT. (+32767). BLUE RAM EXTENDED BASIC APPLIES NO LIMIT ON ADDRESSES, BUT THE ARCADE, IF COUNTING THRU A LOOP, REFUSES TO CROSS THE "SNAKE-PIT" AT !8000, AND REVERSE DIRECTION IN MID-STREAM. THE OBVIOUS SOLUTION IS TO START YOUR PROGRAM AT !8000, LEAVING THE FIRST 7196 BYTES FOR SNAPS OR POKED STRINGS.

HERE, WE MIGHT POKE IN THE MODIFIED DATA AS WE DID WITH !7000, BUT IF WE UNDERSTAND WHAT WE ARE DOING WE CAN START AT ANY ADDRESS AND EVEN INCLUDE A ROUTINE WITHIN THE PROGRAM FOR JUMPING TO ANY UNUSED AREA UTILIZING EVERY LAST BYTE! THE PROCEDURE USES THREE LINES, AS FOLLOWS:

```
>1  :INPUT %(!6FFC);:INPUT %(!8000);
    :INPUT %(!6C20);:INPUT %(!6CAA);RUN

THEN: %(!6FFC)=!8000;%(!6FFE)=!A07E;
      %(!6C82)=!8002;%(!8000)=-256
```

LINE 25000 IS FOR DUMPING TO TAPE:

```
>25000 :PRINT %(!6000),20;:PRINT %(!6FFC),2;
       :PRINT %(!8000),(8322-SZ)c2;:PRINT
       %(!6C20),65;:PRINT %(!6CAA),47
```

LINE #1 LOADS THE PROGRAM AND DATA FROM TAPE THE DATA AS WHATEVER WAS RECORDED BY LINE #25000. THE SECOND LINE, SINCE IT HAS NO LINE NUMBER, IS NOT PART OF THE PROGRAM, BUT STUFFS VALUES INTO THE PROPER BYTES.
"%(!6FFC)=!8000" PUTS OUR STARTING ADDRESS IN A REGISTER THAT IS ALWAYS REFERENCED BY THE SYSTEM.

!6FFE POINTS TO OUR "END OF PROGRAM" AREA, USUALLY THE HIGHEST ADDRESS AVAILABLE (!A07E OR !E07E).

!6C82 POINTS TO AN ADDRESS 2 BYTES BEYOND THE LAST BYTE IN THE CURRENT PROGRAM. AT THIS POINT IT IS !8002 BECAUSE WE HAVEN'T YET WRITTEN ANY PROGRAM. THIS VALUE INCREASES AS WE WRITE.

FINALLY, AT !8000, WE PUT AN ILLEGAL LINE NUMBER (-256). THIS "FLAG" WILL MOVE UP FROM !8000 AS WE WRITE AND WILL ALWAYS BE THE LAST BYTE IN OUR REGULAR PROGRAM, SIGNALLING TH SYSTEM TO STOP.

LINE #25000, (OR ANY LINE NUMBER HIGH ENOUGH TO NOT INTERFERE WITH THE MAIN PROGRAM) DUMPS EVERYTHING TO TAPE IN SEGMENTS:
":PRINT %(!6000),20" DUMPS 20 WORDS (40 BYTES), STARTING AT !6000, WHICH SAVES LINE #1.
":PRINT %(!6FFC),2" SAVES ONLY THE POINTERS. !8000 IS, OF COURSE, THE START OF YOUR PROGRAM, AND "(8322-SZ)c2" IS IT'S SIZE IN WORDS.
"%(!6C20),65" AND "%(!6CAA),47" SAVE 228 BYTES OF STACK AND VARIABLES.

DON'T BE AFRAID TO PLAY WITH THESE VALUES, ESPECIALLY AFTER THE POKES ARE ON TAPE. IF YOU DUMP ONLY THE POKES ON TAPE, PUT A STOP AT 24990 TO PREVENT A LOAD FROM TRYING TO EXECUTE LINE #25000.

# BASICALLY RAMBLING ON!

BY KEN LILL

---

A SERIES OF ARTICLES DESIGNED AS AN AID TO "BLUE RAM" EXPANSION UNIT OWNERS. ALTHOUGH MOST PORTIONS OF THIS COLUMN CAN BE APPLIED TO "VIPERSOFT BASIC", SOME MAY HAVE INFORMATION PERTAINING ONLY TO THE "BLUE RAM EXTENDED BASIC"

---

## USING THE 'OP' COMMAND

FINALLY, WITH THE HELP OF THE 'GREAT' GUYS AT R & L, DALE SMITH AND RUSTY BLOMMAERT, I HAVE LEARNED THE SECRET BEHIND JOHN PERKINS' ELUSIVE 'OP' COMMAND!

TO DESCRIBE HOW ALL OF US PROGRAMMERS WERE INFORMED OF IT'S USAGE, I'LL TELL YOU. "IT IS A 'USER DEFINED' OPTION MADE TO ADD EXTRA COMMANDS TO THE BASIC. YOU MUST USE MACHINE LANGUAGE FOR THIS COMMAND!" BELEIVE ME, THIS SURE DIDN'T MAKE THINGS VERY EASY FOR US TO USE IT!

THIS IS WHAT YOU HAVE TO DO:

1. SET THE ADDRESS %(!6DCC) [28108 FOR YOU DECIMAL TYPES OUT THERE] TO !C3 [195]
2. SET %(!6DCD) AND %(!6DCE) [28109 & 28110] TO THE ADDRESS THAT IS THE VERY FIRST BYTE OF YOUR MACHINE LANGUAGE PROGRAM (REMEMBER TO INVERT ANY NUMBERS THAT ARE NOT OPERATIONAL CODES!!!)
3. ENTER YOUR MACHINE LANGUAGE PROGRAM AT THE ADDRESS YOU HAVE SELECTED
4. FINISH OFF THE PROGRAM WITH A 'JUMP TO !2460 [9312]

THIS COMPLETES THE 'INSERTION' OF THE NECESSARY MACHINE LANGUAGE. NOW ALL YOU HAVE TO DO TO GET YOUR ROUTINE TO WORK IS TO TYPE IN THE 2 LETTER COMMAND 'OP'. YOU DON'T NEED TO FOLLOW IT WITH A PERIOD OR A SPACE.

NOW TO EXPLAIN JUST WHAT YOU DID. THE FIRST COMMAND (!C3) IS JUST A JUMP, FOLLOWED BY THE ADDRESS TO JUMP TO. AT THE AND OF THE PROGRAM, YOU NEED TO JUMP TO THE 'ON-BOARD' SUBROUTINE THAT EVALUATES THE NEXT BYTE IN YOUR 'BASIC' PROGRAM TO FIND OUT IF IT A SEMICOLON OR A 'GO'. IF IT IS NEITHER, IT WILL GIVE YOU AN ERROR MESSAGE.

********** REMEMBER **********

YOU MUST PUSH THE 'DE' REGISTER PAIR AS YOUR FIRST COMMAND TO SAVE THE ADDRESS THAT YOU ARE GOING TO RETURN TO IN YOUR 'BASIC' PROGRAM. THE LAST COMMAND PRIOR TO YOUR 'JP 2460 MUST BE A 'POP DE' COMMAND, SO THAT THE !2460 ROUTINE DOESN'T GET 'LOST'.

ANOTHER LITTLE HINT, 'REGISTER PAIRS' AF, BC AND HL MEAN 'NOTHING' TO YOUR 'BASIC PROGRAM, SO THERE IS NO REAL NEED TO 'PUSH' THEM ONTO THE STACK IF YOU PLAN TO GO BACK INTO YOUR 'BASIC' PROGRAM. IF YOU THINK THAT IS PRETTY NIFTY, LET'S EXPAND THE USE OF THIS COMMAND BY 'LABELING' IT TO DO MORE THAN ONE MACHINE LANGUAGE ROUTINE. THE WAY YOUR 'BASIC' COMMANDS MAY LOOK WILL BE ENTIRELY UP TO YOU AND YOUR PROGRAM. THE THING TO REMEMBER IS THAT EACH 'LABEL' MUST BE A 'VALID' NUMBER OR ARITHMATIC EQUATION. ANY NUMBER OR EQUATION THAT CAN BE PLACED BETWEEN THE COMMAS IN A BOX ,LINE ,POINT , CIRCLE OR DATA COMMAND, PROVIDING THE ANSWER IS NOT LONGER THAN !FF [255], WILL WORK AS A LABEL, LIKE OP A+B-C. IF YOU USE A NUMBER FIRST, YOU DON'T EVEN NEED THE SPACE [OP1]. LET'S SET UP A SAMPLE SET OF MACHINE CODE PROGRAMS, AND THEN WE'LL LABEL EACH OF THEM.

| ADDRESS | HEX | DESCRIPTION |
|---|---|---|
| !7006 | 00 | NOP-DO NOTHING |
| !7007 | FF | SYSGEN |
| !7008 | 48 | PIZBRK |
| !7009 | D1 | POP DE |
| !700A | C3 | JP NN |
| !700B | 60 | |
| !700C | 24 | 2460H |
| !700D | 00 | NOP |
| !700E | 06 | LD B,N |
| !700F | F0 | F0H |
| !7010 | FF | SYSGEN |
| !7011 | 50 | PAWS |
| !7012 | D1 | POP DE |
| !7013 | C3 | JP NN |
| !7014 | 60 | |
| !7015 | 24 | 2460H |
| !7016 | 00 | NOP |
| !7017 | 06 | LD B,N |
| !7018 | 28 | 28H |
| !7019 | B5 | PUSH BC |
| !701A | FF | SYSGEN |
| !701B | 50 | PAWS |
| !701C | B1 | POP BC |
| !701D | 10 | DJNZ |
| !701E | F9 | F9H |
| !701F | D1 | POP DE |
| !7020 | C3 | JP,NN |
| !7021 | 60 | |
| !7022 | 24 | 2460H |
| !7023 | 00 | NOP |
| !7024 | CD | CALL NN |
| !7025 | CE | |
| !7026 | 3C | 3CCEH |
| !7027 | D5 | PUSH DE |
| !7028 | 7D | LD A,L |
| !7029 | 87 | ADD A,A |
| !702A | 21 | LD HL,NN |
| !702B | 00 | |
| !702C | 70 | 7000H |
| !702D | 6F | LD L,A |
| !702E | 5E | LD E,(HL) |
| !702F | 23 | INC HL |
| !7030 | 56 | LD D,(HL) |
| !7031 | EB | EX DE,HL |
| !7032 | E9 | JP (HL) |

NOW WE HAVE TO TELL THE COMPUTER WHERE EACH OP IS LOCATED. TYPE IN THIS INFORMATION.

NOW TYPE IN WITHOUT A LINE #:

```
FOR A=0TO 6STEP 2:PRINT #0," %(",A+!7000,:
INPUT ")="%(A+!7000):NEXT A
```

WHEN ASKED, TYPE IN THESE NUMBERS:
!7007
!700E
!7017
THIS IS WHAT IS HAPPENING:

LET'S ASSUME THAT YOU HAVE SELECTED 'OP 2'. THE FIRST THING IS THAT THE 'BASIC' ENCOUNTERS IT AND THEN IT DOES THE INSTRUCTION LOCATED AT THE ADDRESSES !6DCC-!6DCE. THAT IS TELLING IT TO JUMP TO [GOTO] %(!7023). THEN IT CALLS UP THE ROUTINE LOCATED AT %(!3CCE) INSIDE OF THE BLUE RAM BASIC. THIS ROUTINE 'EVALUATES THE EXPRESSION' THAT FOLLOWS THE COMMAND. THIS ROUTINE IS USED BY BOX ,LINE ,ETC. TO FIND OUT THE EXACT NUMBER OF THE EXPRESSIONS LIKE: A+B , A*B+100-RND(10) ETC. THIS WILL SKIP AUTOMATICALLY OVER ANY SPACES THAT YOU MIGHT HAVE PUT INTO THE EXPRESSION. IF YOU JUST HAVE 1 NUMBER, THAT IS YOUR ANSWER. THE 3 THINGS THAT WILL END AN EXPRESSION ARE (1) A 'GO', (2) A SEMICOLON AND (3) A COMMA. WHATEVER YOU DO, DON'T USE A COMMA IN THESE OPS! IF YOU DO, WHEN THE PROGRAM HITS IT, IT WILL BE EXPECTING ANOTHER MACHINE LANGUAGE PROGRAM THAT IS MEANT TO 'EVALUATE' THE COMMA AND IT'S COMMAND. THIS IS NOT IN OUR MACHINE LANGUAGE PROGRAMS GIVEN HERE. AFTER THE EXPRESSION EVALUATION, I PUSH THE DE REGISTER PAIR. THE REASON BEING THAT THE DE PAIR NOW CONTAINS THE ADDRESS OF THE LAST BYTE IN OUR EXPRESSION. WE WANT TO SAVE THIS SO THAT WE CAN GO BACK TO THAT EXACT SPOT WHEN WE RETURN TO OUR 'BASIC' PROGRAM. THE HL PAIR CONTAINS THE ANSWER OF THE MATH. NOW WE WANT THE LOWER BYTE OF THE ANSWER, BECAUSE WE ARE ONLY USING NUMBERS IN THIS PROGRAM THAT ARE LOWER THAN !FF AS 'LABELS'. WE THEN LOAD THAT BYTE INTO 'A' OF THE Z80. NEXT WE DOUBLE IT BECAUSE IT TAKES 2 BYTES TO STORE OUR ADDRESS VECTORS! NOW WE PUT OUR LOWEST ADDRESS OF OUR 3 VECTORS INTO HL. THEN WE CAN DIRECTLY LOAD L WITH A TO 'ADD' OUR ANSWER. IF YOU START WITH AN ADDRESS THAT DOESN'T HAVE 2 0'S AT THE END, YOU'LL HAVE TO 'ADD A,L' HERE AND MOVE EVERYTHING DOWN ONE BYTE. DOING THIS WILL NOT EFFECT ANY OF THE JUMPS IN THESE PROGRAMS. NOW WE LOAD E WITH THE NUMBER IN THE ADDRESS POINTED TO BY HL. WE ADD 1 TO HL AND THEN WE TAKE AND LOAD D WITH THE NEXT BYTE OF THE ADDRESS. NOW WE 'SWAP' OR EXCHANGE DE WITH HL, SO THAT WE CAN JUMP TO THAT ADDRESS. IN OUR CASE, WE ARE NOW AT %(!7007). IN THIS ROUTINE, I CALL UP PIZBRK, WHICH IS A ROUTINE THAT CLEARS THE SCREEN, WAITS FOR A SWITCH, AND THEN RETURNS TO THE POINT WE CAME FROM. IT THEN EXECUTES THE NEXT COMMAND. THIS IS TO POP DE. THEN WE GO BACK TO 'BASIC'.

OP 1 GIVES US A PAUSE OF 140/60THS OF A SECOND. THIS IS 2 1/3 SECONDS. WE THEN POP DE AND RETURN TO BASIC. IN OP 2, PAWS IS CALLED UP AGAIN, BUT THIS TIME IT'S ONLY FOR 40/60THS. BECAUSE BC IS PUSHED, WE CAN NOW POP IT, DO A DJNZ [DISPLACEMENT JUMP, NOT ZERO]. WHAT THIS DOES IS DECREMENT BC AND CHECK TO SEE IF IT IS ZERO. IF NOT, JUMP TO THE POINT IN THE PROGRAM DESIGNATED BY THE NEXT BYTE. BECAUSE THIS IS A DISPLACEMENT JUMP, THE JUMP WILL START FROM THE BYTE THAT GIVES YOU THE NUMBER OF BYTES, AND NOT FROM THE JUMP BYTE. WE THEN JUMP BACK TO THE PUSH BC COMMAND AND DO THIS ALL OVER AGAIN UNTIL BC=0. AFTER THAT, WE POP DE AND GO BACK TO OUR BASIC PROGRAM.

I HOPE YOU HAVE THE MAIN IDEA ON HOW TO USE THE 'OP' COMMAND WITH AND WITHOUT USING LABELS. UNTIL NEXT TIME;

**HAPPY PROGRAMMING!!**

KEN LILL
6608 S. CAMPBELL
CHICAGO, ILLINOIS 60629

---