A C64 Cartridge Without EPROMs

John Bush and Noel Nyman
Seattle, Washington

you won't need any expensive programming devices to make your own cartridges for a C64 or C128 with this special technique

Cartridges are convenient and easy to use. Programs on cartridge Load instantly. You can make a cartridge using EPROMs (Erasable Programmable Read-Only Memories) for about $25, if you shop carefully.

But, the EPROMs must be programmed or "burned" using an EPROM burner, which costs about $125. If you make any mistakes, or want to change the programs, you'll need an EPROM eraser, another $40.

The inexpensive EPROM cartridge requires close to $200 in start-up costs.

An alternative is to use RAM (Random Access Memory) in place of EPROMs. RAM can be programmed by the computer itself, and the information can be changed at any time. No additional special equipment is required.

The problem with RAM is that it loses everything in memory when the power is turned off, not exactly what we have in mind for a cartridge. But, by using special CMOS (Complementary Metal Oxide Semiconductor) RAMs that have low stand-by current requirements, we can use a small battery to hold the information in the RAM. The memory is retained even with the computer turned off or when the cartridge is removed. The 4464-15s, made by NEC Corp, used in this project have a typical stand-by current drain of 0.1 micro-amperes. A battery the size of a quarter can power them for several years.

Building The RAM Cartridge

We used a Vector 3795-1 "perf" board for our prototype. It has 44 circuit traces (22 on each side) at the proper spacing to line up with the C64 expansion socket. If you have the equipment to etch your own circuit boards, that may be a less expensive alternative. You may be able to adapt an old cartridge board, or purchase one intended for use in a C64. Be sure that address lines A13 through A15 (pins F, H, and J) are available on the board you use. They aren't needed by EPROM cartridges and may not appear on circuit boards designed for that purpose.

Although we used wire-wrap to build the circuit, any wiring method will work. Sockets are recommended for the integrated circuits, but are not mandatory. Be sure to observe proper precautions when working with the CMOS RAM's. They can be permanently damaged by improper handling.

Figure #1 shows the schematic for an 8K RAM cartridge. Figure #2 has the additional circuitry required to add another 8K. Switch S1 controls the power to the CMOS RAMs. With the switch closed, power comes from the C64. With either S1 open or the computer turned off, the battery takes over and retains the data in memory. S2 controls the READ/WRITE signals to the RAMs. With this switch closed, the computer can change the data. Opening S2 makes the RAMs look like ROM to the C64.

S3 and S4 allow the RAM cartridge to emulate the three types of cartridge used with the C64, which we'll look at shortly. S5 is used only with the 16K version. It allows us to "move" the upper 8K of RAM to an area where it can be programmed. The diodes electrically remove the battery from the circuit when the computer is supplying power and prevents the battery from trying to run the entire C64. The various resistors establish default values for the signal lines and switch the RAMs to their low current stand-by state when S1 is opened.

The 74LS42 is a decoder that monitors the three highest address lines (A13 - A15), and produces a discrete output for each combination of these addresses. There are eight outputs, so we can select eight 8K banks of memory with this chip. Capacitors C1 and C2 are used to remove any noise from the power line. C1 should be placed close to the edge of the board that plugs into the computer. C2 should be mounted as close as possible to the 74LS42.

You may find other 8 x 8K RAMs with similar stand-by current characteristics. If they have 150ns (nano-second) access time or less, they should work for this application. Be sure to get data sheets for them. The pin-outs may be different from those shown on these schematics. See the end of this article for a source for the NEC 4464-15s we used, or check your yellow pages under "Electronic Equipment" for a local NEC distributor.

Parts List

B1	- 3 Volt Circuit Battery (see text)
C1, C2	- 0.05 mfd 12VDC Ceramic Disk Capacitor
D1-D4	- 1N4148 or Similar Small Signal Diode
R1,R3,R4,R5,R7	- 2K 1/4 Watt Resistor
R2,R6	- 22K 1/4 Watt Resistor
S1-S4	- SPST Switches, DIP Arrays Work Well
S5	- SPDT Miniature Switch
74LS42	- 1 of 10 BCD Decoder
4464	- Low Stand-By Current CMOS Static RAM (see text)

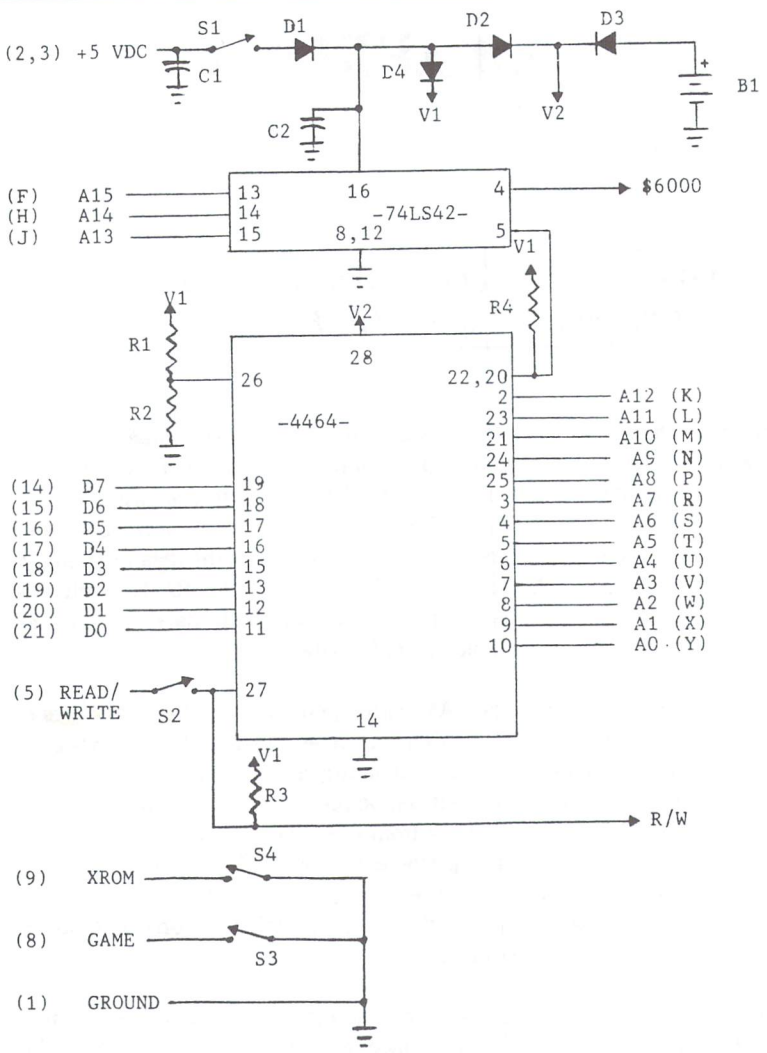

Figure 1: All references in parentheses are pin numbers for the C64 expansion port, see pg.396 of the C64 Programmers Reference Guide.

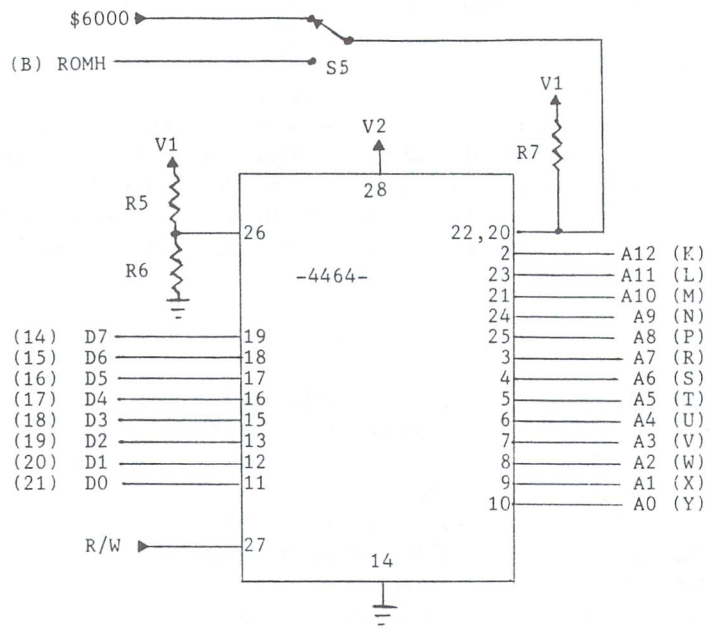

Figure 2: Additional parts required for a 16K cartridge.

How Cartridges Work

The C64 uses a PLA (Programmed Logic Array) to control the access of RAM, ROMs, and cartridges to the address and data buses. For an excellent discussion of how the PLA works, see "Commodore 64 Memory Configurations" by William Levak (Transactor 6-05). Cartridges can have three configurations. The PLA identifies the cartridge by two control lines. These are called "GAME" (pin 8) and "XROM" (pin 9). The RAM cartridge uses switches S3 and S4 to activate the control lines.

An 8K cartridge always appears at address range $8000 – $9FFF. It has an internal jumper that pulls the XROM line low. Closing S4 simulates that configuration. A 16K cartridge also has 8K at $8000 – $9FFF. The upper 8K can reside in one of two other areas. If only the GAME line is low (S3 closed, S4 open), the upper 8K appears at $E000 – $FFFF. If both GAME and XROM are low (S3 and S4 closed), all 16K is contiguous from $8000 – $BFFF.

An 8K cartridge normally contains either a self contained program, or one that uses the BASIC and Kernal ROM routines built into the C64. A 16K cartridge in the $8000 – $BFFF range replaces the BASIC ROM. The upper 8K may contain a modified BASIC, and the lower 8K may have BASIC extensions. The third configuration was intended for games only. Levak's article shows that in this mode, the VIC chip will look for the character set at the upper portion of the $E000 – $FFFF memory. This makes for easier low resolution graphics for games, but is unsuitable as a Kernal replacement. The programs in these cartridges must stand entirely on their own.

All memory chips, RAM or ROM, are switched onto the address and data buses with "chip select" lines. In the C64, the PLA controls these lines, and so decides whether RAM, or one of the system ROMs, or the cartridge is selected. If the PLA senses that a cartridge is in place (through the GAME and XROM lines), and a "READ" command is issued by the microprocessor, the cartridge memory will be selected. The PLA controls this selection through the "ROML" (pin 11) and "ROMH" (pin B) lines. If a "WRITE" command is issued, the PLA switches off the cartridge memory and selects RAM at those addresses instead.

Commodore never intended that cartridges would contain RAM. So the PLA will not write data into our RAM cartridge. To accomplish that, we by-pass the PLA and do our own decoding. Some is done automatically by the 74LS42 chip, and some we control manually with switch S5.

Programming The RAM Cartridge

When the C64 is turned on, reset with an external reset switch, or the "RESTORE" key is pressed, routines in the Kernal ROM look for a cartridge. All cartridges will have 8K starting at location $8000. The Kernal looks for the code "CBM80" starting at address $8004. The high bit of

each letter must be set. If the code is there, the normal initialization routines are bypassed, and control is passed to the program in the cartridge. On power-up or hardware reset, the address stored in low-high order at $8000/$8001 is used for an indirect jump. If "RESTORE" has been pressed, the address stored at $8002/$8003 is used instead.

To create an auto-starting program in cartridge, you'll need to install the code phrase and the proper addresses. You may also need to call some of the bypassed initializing routines. You can store machine code in the RAM cartridge without the auto-start phrase and SYS to the code from BASIC or direct mode instead of auto-starting.

If you want to use the RAM cartridge to store a favourite BASIC program, use the program in Listing #1. RUNning the program creates a file called "RAMCART" on disk device #8. You can change those defaults in line 100. The source code of the file is shown in PAL format in Listing #2.

To use the program, install the RAM cartridge, and close S1 and S2. Be sure S3 and S4 are both open. Then turn on the computer. The cartridge RAM is now "in parallel" with system RAM. The two are examined together by the C64, and the same data is stored in each at the corresponding addresses. This step is important. If the two RAMs contained different data, they would fight each other on the data bus.

LOAD the "RAMCART" program with ",8,1". This places the code at the start of RAM cartridge memory. Now LOAD the BASIC program you want to store. Do not RUN it. Type

SYS 32882

The machine code stored by "RAMCART" will copy the BASIC program into the cartridge RAM. If the program is too big, over 31 disk blocks, you'll get an error message instead. When the "READY" prompt appears, open S2. This disconnects the cartridge from the READ/WRITE line, and the data cannot be changed by the computer.

Turn off the C64. The battery will retain the program in the cartridge RAM. Close S4 to tell the PLA that this is an 8K cartridge, and turn the computer back on. The auto-start code in the RAM cartridge will cause the system to initialize BASIC normally. Then it copies your program back to the BASIC memory area. The "RUN" command is placed in the keyboard buffer and the computer executes it, starting your program.

The RUN-STOP/RESTORE combination will bring you out of your BASIC program and display the "READY" message. To re-RUN the program in the cartridge, use a hardware reset switch or type

SYS 64738

A different technique is required to program the upper 8K of RAM in a 16K cartridge. We need to use the ROMH line from the PLA to select the cartridge memory, since the PLA will switch system ROM in otherwise. But the PLA will not let us write data to the memory selected by ROMH. S5 switches the upper 8K RAM select line between the ROMH output from the PLA and the $6000 - $7FFF output from the 74LS42. With S5 in the $6000 position, you can change the upper 8K of data by writing to the RAM at this lower location. Moving S5 back to the ROMH side causes the PLA to switch in the RAM at either $A000 or $E000, depending on the settings of S3 and S4.

For example, to change BASIC, place a 16K ram cartridge in the computer. Close S1 and S2, open S3 and S4, and move S5 to the $6000 position. Turn on the computer. LOAD a machine language monitor that resides below $6000 or above $C000, and use it to copy the BASIC ROM to the RAM at $6000. Use the memory examine mode to look at the nine bytes starting at $6378. This is the text "READY." followed by a "RETURN" ($0D), a line feed ($0A), and a terminating zero byte ($00). Use the monitor to change the text.

Now open S2 to lock the changes in RAM, and turn off the computer. Move S5 to the ROMH position. Close S3 and S4. This tells the PLA to place the 8K of RAM with the modified BASIC in the address area normally used by the BASIC ROM. Turn on the computer and you'll see your modified "READY" prompt. You'll also see only 30,719 BASIC bytes free, because the lower 8K of ram cartridge is also switched in by the PLA. You can use the lower 8K to hold BASIC programs, or extensions in addition to any modifications you make to the BASIC operating system.

The switch settings for programming and using the cartridge are summarized in Figure 3.

Figure 3

	S1	S2	S3	S4	S5
Reading From Cartridge:					
8K Cartridge	ON	OFF	OFF	ON	X
16K Cart., Upper 8K At $A000	ON	OFF	ON	ON	ROMH
16K Cart., Upper 8K At $E000	ON	OFF	ON	OFF	ROMH
Writing To Cartridge:					
8K Cartridge	ON	ON	OFF	OFF	X
16K Cartridge	ON	ON	OFF	OFF	$6000

The ram cartridge is fully compatible with expansion cards which allow several cartridges to be plugged in at the same time. Be sure to turn S1 off when you select a different cartridge so the RAM at $8000 will be removed from the buses. You can use the ram cartridge on a C128 also. The GAME and XROM lines aren't used in C128 mode. The MMU (Memory Management Unit) looks for a different code instead. You'll have to write a C128 auto-boot routine, but use the procedure above from C64 mode to install it.

We think you'll find the ram cartridge an inexpensive alternative to purchasing an EPROM burner and eraser to make your own cartridges. Even if you already have EPROM programming equipment, the ease and speed of making changes to your cartridge software may be an asset.

Although Geoduck Developmental is not in the retail component sales business, we will make 4464-15 RAMs and battery/socket kits available at cost for Transactor readers. Please send $15 (Canadian) for each RAM and $5 for each battery and socket. For orders outside Canada or the USA, add $5 for postage. Send orders or any questions or comments on the ram cartridge to:

Geoduck Developmental Services
PO Box 58587
Seattle WA 98188
USA

Listing 1: Basic Loader To Create RAMCART Module On Disk

```
FO 1000 rem save " 0:ramcart.ldr " ,8
AH 1010 rem ** by: john bush and noel nyman - seattle, wa
IK 1020 rem ** auto-start support prg
KF 1030 rem ** for c64 ram cartridge
EI 1040 :
CI 1050 rem ** this program will create
JB 1060 rem ** a load " ,8,1 " module on
HO 1070 rem ** disk called 'ramcart'
MK 1080 :
NC 1090 open 15,8,15: open 8,8,1, " 0:ramcart "
BN 1100 input#15,e,e$,b,c: if e then close 15: print e;e$;b;c:
        stop
FH 1110 for j=32768 to 32999: read x: print#8,chr$(x);:
        ch=ch+x: next: close8
ED 1120 if ch<>28345 then print " checksum error! " : stop
LC 1130 print " ** module created ** " : end
IO 1140 :
NL 1150 data   0, 128,   9, 128,  94, 254, 195, 194
PI 1160 data 205,  56,  48, 162,   5, 142,  22, 208
LH 1170 data  32, 163, 253,  32,  80, 253,  32,  21
AM 1180 data 253,  32,  91, 255,  88,  32,  83, 228
FO 1190 data  32, 191, 227, 162, 251, 154, 172, 224
KO 1200 data 128, 174, 225, 128, 132,  43, 134,  44
PM 1210 data 172, 228, 128, 174, 229, 128, 132,  95
OD 1220 data 134,  96, 172, 226, 128, 174, 227, 128
KC 1230 data 132,  88, 134,  89, 136, 192, 255, 208
AN 1240 data   1, 202, 132,  45, 134,  46, 169, 160
AB 1250 data 133,  91, 169,   0, 133,  90,  32, 191
AG 1260 data 163, 169,  82, 141, 119,   2, 169,  85
GL 1270 data 141, 120,   2, 169,  78, 141, 121,   2
CA 1280 data 169,  13, 141, 122,   2, 169,   4, 133
NG 1290 data 198, 108,   2,   3,  56, 165,  46, 229
PL 1300 data  44, 170, 165,  45, 229,  43, 168, 224
NE 1310 data  31, 176,  67, 140, 228, 128, 142, 229
GL 1320 data 128,  56, 169, 159, 237, 229, 128, 141
DG 1330 data 229, 128, 169, 255, 237, 228, 128, 141
GF 1340 data 228, 128, 165,  43, 141, 224, 128, 133
CO 1350 data  95, 165,  44, 141, 225, 128, 133,  96
EI 1360 data 164,  45, 166,  46, 200, 208,   1, 232
OG 1370 data 140, 226, 128, 132,  90, 142, 227, 128
KN 1380 data 134,  91, 169, 160, 133,  89, 169,   0
DA 1390 data 133,  88,  32, 191, 163,  96, 169, 204
CH 1400 data 160, 128,  32,  30, 171,  96,  80,  82
FA 1410 data  79,  71,  82,  65,  77,  32,  84,  79
MO 1420 data  79,  32,  76,  65,  82,  71,  69,  10
HP 1430 data  13,   0,   0,   0,   0,   0,   0,   0
```

Listing 2: PAL Source for support program

```
MM 1000 rem save " 0:ramcart.pal " ,8
AH 1010 rem ** by: john bush and noel nyman - seattle, wa
IL 1020 rem ** auto-start support prg for c64 ram cartridge
KH 1030 :
JP 1040 open 8,8,1, " 0:ramcart "
LO 1050 sys 700
HE 1060 .opt o8
EB 1070 *       =   $8000
OK 1080 ;
FP 1090 ;*** equates ***
CM 1100 ;
KM 1110 txttab  =   $2b     ;start of basic text
HL 1120 vartab  =   $2d     ;end of basic text
BL 1130 source  =   $5f     ;start of source to copy
KI 1140 end     =   $5a     ;end + 1 of source to copy
MC 1150 dest    =   $58     ;end + 1 of destination
NC 1160 ndx     =   $c6     ;no of characters in keyboard
                            buffer
BC 1170 keyd    =   $0277   ;start of keyboard buffer
IK 1180 warm    =   $0302   ;basic warm start vector
HA 1190 copy    =   $a3bf   ;copy memory
LK 1200 strout  =   $ab1e   ;print string
LG 1210 vicctrl =   $d016   ;vic control register
DN 1220 vectors =   $e453   ;copy basic vectors to ram
DF 1230 init    =   $e3bf   ; initialize basic interpreter
LL 1240 ioinit  =   $fda3   ;initialize i/o
HA 1250 ramtas  =   $fd50   ;initialize memory pointers
HM 1260 restor  =   $fd15   ;restore i/o vectors
EA 1270 cint    =   $ff5b   ;init screen and keyboard
NP 1280 nmicont=    $fe5e   ;continue with nmi routine
AI 1290 ;
GE 1300 ;*** auto-start basic program ***
EJ 1310 ;
BG 1320 ;place start of code in cartridge vectors
PM 1330 .byte   <start,>start
AE 1340 .byte   <nmicont,>nmicont
KI 1350 ; 'cbm' with bit 7 set
FH 1360 .byte   $c3,$c2,$cd
OI 1370 .asc    "80"
KN 1380 ;
LF 1390 ;'start' calls most of the routines
GK 1400 ;which would be executed if a cartridge
ID 1410 ;had not been detected. system vectors
AD 1420 ;and basic are initialized.
MA 1430 ;
BH 1440 start ldx   #5
FE 1450       stx   vicctrl
EH 1460       jsr   ioinit
FI 1470       jsr   ramtas
EF 1480       jsr   restor
EF 1490       jsr   cint
KG 1500       cli
MO 1510       jsr   vectors
FN 1520       jsr   init
DA 1530       ldx   #$fb
KA 1540       txs               ;initialize stack pointer
EI 1550 ;
PL 1560 ;copy the basic program from
JH 1570 ;the area under $a000 to the start-of-basic
IP 1580 ;and set up the basic text and variables
DM 1590 ;vectors. place 'run' in the keyboard buffer and
OP 1600 ;enter basic through the warm start vector.
AM 1610 ;
PI 1620       ldy   txtt        ;store start of basic
IJ 1630       ldx   txtt+1      ;saved with program
```

```
OK 1640         sty  txttab      ;at op system vector
LE 1650         stx  txttab+1
PM 1660         ldy  stsour      ;store start of source
LJ 1670         ldx  stsour+1    ;at vector for copy routine
LJ 1680         sty  source
DG 1690         stx  source+1
GA 1700         ldy  vart        ;store end of destination
                                 (+1)
FA 1710         ldx  vart+1      ;at copy routine vector
FN 1720         sty  dest
HO 1730         stx  dest+1
PJ 1740         dey              ;subtract one from low byte
FA 1750         cpy  #$ff
MP 1760         bne  cont
NH 1770         dex              ;subtract borrow
ND 1780 cont    sty  vartab      ;store op system vector
FK 1790         stx  vartab+1
NN 1800         lda  #$a0        ;end of source (+1) =
                                 $a000
PB 1810         sta  end+1
HA 1820         lda  #0
ME 1830         sta  end
KD 1840         jsr  copy
HI 1850         lda  #"r"
KN 1860         sta  keyd
EK 1870         lda  #"u"
KP 1880         sta  keyd+1
DK 1890         lda  #"n"
AB 1900         sta  keyd+2
IP 1910         lda  #$0d        ;<return>
GC 1920         sta  keyd+3
GB 1930         lda  #4          ;number of characters
IB 1940         sta  ndx
JN 1950         jmp  (warm)
OB 1960 ;
DO 1970 ;*** store basic program to cartridge ***
JC 1980 ;calculate the size of the basic text, and
NF 1990 ;print an error message if too large to fit
OB 2000 ;in the cartridge. if okay, subtract the size
MM 2010 ;from $9fff to get the location of the start
DA 2020 ;of the copy to be saved to cartridge. save
PA 2030 ;that vector, and the start and end of basic
ND 2040 ;text for future use. set-up vectors for
JE 2050 ;copy routine and copy program to cartridge.
CI 2060 ;
GL 2070 store    sec
NC 2080          lda  vartab+1
DM 2090          sbc  txttab+1   ;find size of basic program
PP 2100          tax
JI 2110          lda  vartab
FN 2120          sbc  txttab
BC 2130          tay
NO 2140          cpx  #$1f       ;max size allowed
CI 2150          bcs  error      ;print error message and
                                 quit
DP 2160          sty  stsour     ;store size temporarily
HG 2170          stx  stsour+1
FA 2180          sec
CP 2190          lda  #$9f       ;subtract size from $9fff to
                                 find
NP 2200          sbc  stsour+1   ;start of program in car-
                                 tridge memory
DD 2210          sta  stsour+1
JG 2220          lda  #$ff
HF 2230          sbc  stsour
PI 2240          sta  stsour
GD 2250          lda  txttab     ;store start of basic for
                                 cartridge
EJ 2260          sta  txtt       ;use and in vector for copy
                                 routine
JI 2270          sta  source
HC 2280          lda  txttab+1
EP 2290          sta  txtt+1
JG 2300          sta  source+1
PF 2310          ldy  vartab     ;store end of basic (+1) for
                                 cartridge
LK 2320          ldx  vartab+1   ;use and vector for copy
                                 routine
KP 2330          iny
CK 2340          bne  cont1
KA 2350          inx
BF 2360 cont1    sty  vart
IM 2370          sty  end
IH 2380          stx  vart+1
PL 2390          stx  end+1
OE 2400          lda  #$a0       ;store $a000 (end of car-
                                 tridge memory +1)
HO 2410          sta  dest+1     ;in vector for read routine
PF 2420          lda  #0
LD 2430          sta  dest
CJ 2440          jsr  copy
OH 2450          rts
CB 2460 ;
LJ 2470 ;*** print error message ***
GC 2480 ;
LD 2490 error    lda  #<message
JC 2500          ldy  #>message
NM 2510          jsr  strout
EM 2520          rts
IF 2530 ;
JG 2540 message  =*
AE 2550 .asc     "program too large"
NE 2560 .byte    $0a,$0d,$00
AI 2570 ;
AK 2580 ;*** system vector storage ***
EJ 2590 ;
AD 2600 txtt     .word 0         ;start of program in ram
JI 2610 vart     .word 0         ;end of program in ram
IC 2620 stsour.word 0            ;start of source in cartridge
ML 2630 ;
MC 2640 .end
```