

1. Introduction
2. How it works
3. Auto start and program Function Keys
4. Menus
5. Loaders for different sized programs
6. Loaders for C64 programs
7. Appendix 1 – Source Code listings
8. Appendix 2 – Parts Lists and Schematics

INTRODUCTION

I started playing with the Internal ROM socket in the C128 back in the late 1980's. At first I used 32K (27256) EPROMs, but ran out of room quickly. I decided to try to find something that had more room for programs.

Intel was making an EPROM (27513) that they called a Paged EPROM. It had 4 – 16K pages of ROM, equivalent to a 64K (27512) EPROM. I used the I/O at the Expansion Port to select a page on the EPROM (1,2,3 or 4). Put the page number on the Data Bus and write to the I/O address (\$DE00). Whatever page you select, 16K of ROM will show up at \$8000 – \$9FFF and again at \$A000 – \$BFFF. It was OK for awhile, but I still did not have enough room for the programs I used most of the time.

Then Intel came out with an 8 paged EPROM (27011) 8 – 16K pages (1 MEG) of ROM. I was able to put most of my programs and a couple of large programs that I use, Merlin 128 and Word Writer, in the ROM. Intel talked about making a 2 MEG Paged EPROM, but it never showed up. Their next 2 MEG EPROM was the 27020, 32-pin package. In order to use the 32-pin package, I would have to make an adapter with controlling circuitry. I didn't want to fool with it at the time, I thought 1 MEG of ROM would be enough.

The 27513 and the 27011 EPROMs are 28-pin packages and were easy to install. The page select pin is pin 27. After the ROM was programmed, bend pin 27 up enough so the pin would not go into the socket when installed in the C128, tack solder a lead to pin 27 with a clip on the other end and clip it to the I/O on the Expansion Port. That's it, no adapter and easy to install. Later, I started using a socket with pin 27 removed. It was easier to erase and reprogram the EPROM.

I programmed several 27010 EPROMs for friends in our computer club and I gave a demonstration at the Kansas City, MO computer club. After awhile I moved away from Commodore and concentrated on my job and preparing for retirement. I retired 5 years ago and started playing with the C128 again.

I decided to play around with the large capacity EPROMs, 27010 (1 MEG), 27020 (2 MEG), 27040 (4 MEG) and 27080 (8 MEG).

First I made a cartridge board for the C64 that would accommodate any one of the four high capacity EPROMs.

The C64 will only see 8k or 16k of ROM, so I set it up to use 16k, from \$8000 to \$BFFF. So with the 1 meg EPROM you will have 8 – 16k sections of PROM available. From this point forward the word section(s) will be referred to Page(s). With a 2 MEG EPROM there will be 16 – 16k Pages of PROM, with a 4 MEG EPROM there will be 32 – 16k Pages and with an 8 MEG EPROM there will be 64 pages of PROM. With an 8 meg EPROM you could put 63 – 16k cartridge games in one cartridge (the first page, Page 0, is where you would put the menu).

I did not want to restrict the cartridge to just 16k cartridge games, so I modified the board so that it would be able to shut itself off and go to C64 basic, or load a bigger game, and even multipart games. But we can get into that later when I cover the C64 section.

I needed more than 1 meg of ROM space for the C128, so I made an adapter for the Internal ROM socket. With the adapter it will allow the use of a 32-pin EPROM in a 28-pin socket. Like the C64 Cartridge, it will accommodate any one of the high capacity EPROMs.

I stopped using the I/O at the Expansion Port because it interfered with some cartridge programs. The C128 has two unused I/Os, pin 12 and pin 14 of U3 (74LS138). I chose pin 12 that is available at \$D700 in the I/O section. With the I/O in C128 memory active you can READ or WRITE to address \$D700 and the voltage at pin 12 of U3 will transition from high to low. This I/O is used to select one of the possible pages of the EPROM (0-32). All you have to do is put the page number (in HEX format) on the data buss and write to \$D700 and the EPROM will switch to the page you selected. For example:

```
LDA #$1B          ; PAGE 27
NOP
NOP
STA $D700         ; TRANSITION PIN 12
```

The NOPs are to allow enough time for the page number to appear on the data buss before you transition pin 12. Another example (this is the way I do it):

```
LDA #$1B          ; PAGE 27
STA $D700
STA $D700
```

HOW IT WORKS

DIP PIN CONFIGURATIONS

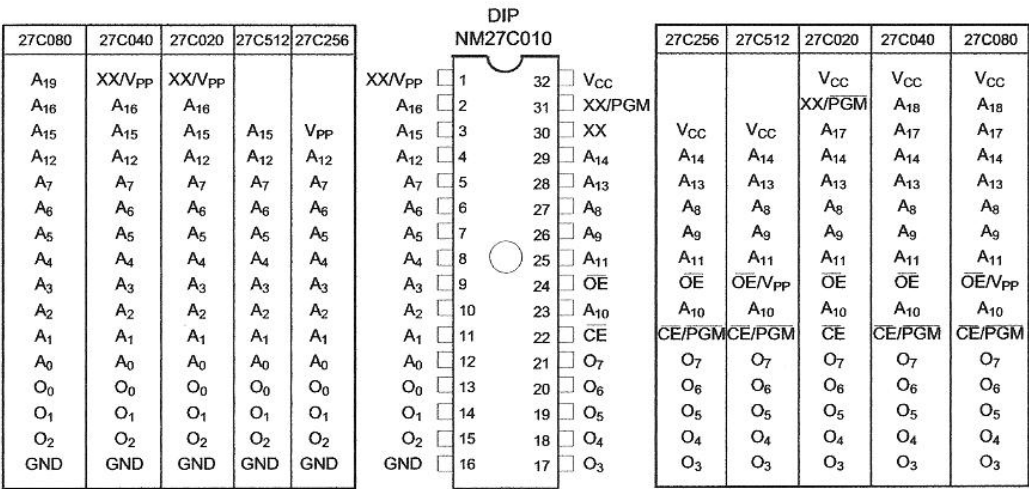


Figure 1

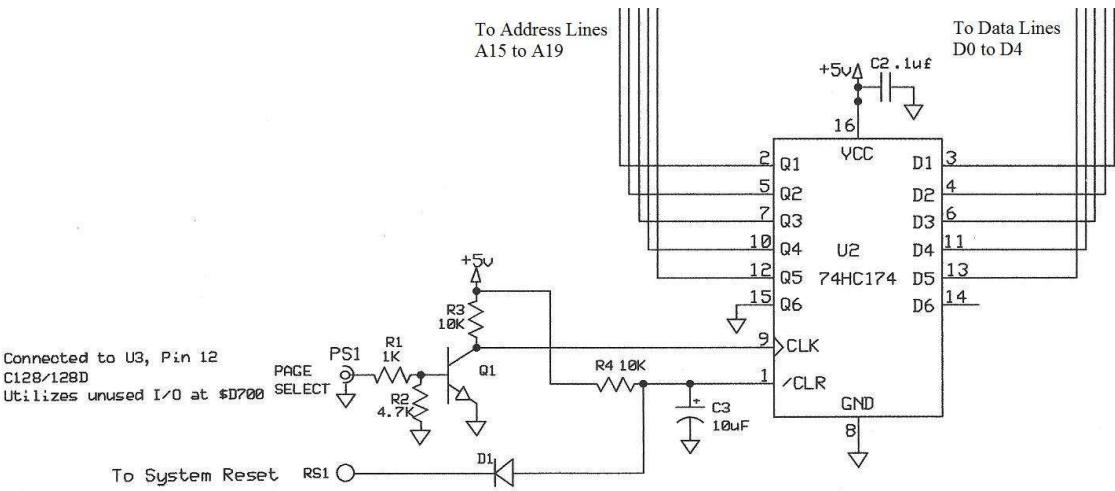


Figure 2

The Internal ROM area will use only 32k of memory space (\$8000 - \$FFFF). If you look at a schematic of the C128, the Internal ROM area (U36) has only 15 address lines (A0-A14). That is the same as a 32k EPROM (27256). Below is the total number of address lines for the high capacity EPROMs. Refer to Figure 1.

27010	17 adrs lines	A0-A16	1 meg
27020	18 adrs lines	A0-A17	2 meg
27040	19 adrs lines	A0-A18	4 meg
27080	20 adrs lines	A0-A19	8 meg

To use the other areas of the larger EPROMs, we have to control the remaining address lines. I have done that by using a D-type Flip-Flop for each extra address line. Refer to Figure 2. The page number is placed on the data buss and the binary value will appear at the D inputs of the HEX D Flip-Flop (27HC174). When the clock input is transitioned from a low to a high the data at the D inputs will be transferred to the Q outputs and the address lines of the EPROM.

The transistor in figure 2 is used as an inverter because the signal from U3 pin-12 transitions from high to low, the clock input on the HEX D flip-flop requires a low to high transition.

Resistor R4 and C3 are used to hold the Flip-Flops in reset mode during power up. That way the ROM will always come up on Page 0 at power up. (Optional) If you connect RS1 to system reset, the ROM will be reset to Page 0 every time you push the reset button.

Here is how it looks with a 27010 (1 MEG EPROM):

<u>Pg HEX</u>	<u>A16</u>	<u>A15</u>	
Page 00	0	0	32K x 8
Page 01	0	1	32K x 8
Page 02	1	0	32K x 8
Page 03	1	1	32K x 8
Total			128K x 8

Here is how it looks with a 27020 (2 MEG EPROM):

<u>Pg HEX</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	
Page 00	0	0	0	32K x 8
Page 01	0	0	1	32K x 8
Page 02	0	1	0	32K x 8
Page 03	0	1	1	32K x 8
Page 04	1	0	0	32K x 8
Page 05	1	0	1	32K x 8
Page 06	1	1	0	32K x 8
Page 07	1	1	1	32K x 8
Total				256K x 8

Here is how it looks with a 27040 (4 MEG EPROM):

<u>Pg HEX</u>	<u>A18</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	
Page 00	0	0	0	0	32K x 8
Page 01	0	0	0	1	32K x 8
Page 02	0	0	1	0	32K x 8
Page 03	0	0	1	1	32K x 8
Page 04	0	1	0	0	32K x 8
Page 05	0	1	0	1	32K x 8
Page 06	0	1	1	0	32K x 8
Page 07	0	1	1	1	32K x 8
Page 08	1	0	0	0	32K x 8
Page 09	1	0	0	1	32K x 8
Page 0A	1	0	1	0	32K x 8
Page 0B	1	0	1	1	32K x 8
Page 0C	1	1	0	0	32K x 8
Page 0D	1	1	0	1	32K x 8
Page 0E	1	1	1	0	32K x 8
Page 0F	1	1	1	1	32K x 8
Total					512K x 8

Here is how it looks with a 27080 (8 MEG EPROM):

<u>Pg HEX</u>	<u>A19</u>	<u>A18</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	
Page 00	0	0	0	0	0	32K x 8
Page 01	0	0	0	0	1	32K x 8
Page 02	0	0	0	1	0	32K x 8
Page 03	0	0	0	1	1	32K x 8
Page 04	0	0	1	0	0	32K x 8
Page 05	0	0	1	0	1	32K x 8
Page 06	0	0	1	1	0	32K x 8
Page 07	0	0	1	1	1	32K x 8
Page 08	0	1	0	0	0	32K x 8
Page 09	0	1	0	0	1	32K x 8
Page 0A	0	1	0	1	0	32K x 8
Page 0B	0	1	0	1	1	32K x 8
Page 0C	0	1	1	0	0	32K x 8
Page 0D	0	1	1	0	1	32K x 8
Page 0E	0	1	1	1	0	32K x 8
Page 0F	0	1	1	1	1	32K x 8
Page 10	1	0	0	0	0	32K x 8
Page 11	1	0	0	0	1	32K x 8
Page 12	1	0	0	1	0	32K x 8
Page 13	1	0	0	1	1	32K x 8
Page 14	1	0	1	0	0	32K x 8
Page 15	1	0	1	0	1	32K x 8
Page 16	1	0	1	1	0	32K x 8
Page 17	1	0	1	1	1	32K x 8
Page 18	1	1	0	0	0	32K x 8
Page 19	1	1	0	0	1	32K x 8
Page 1A	1	1	0	1	0	32K x 8
Page 1B	1	1	0	1	1	32K x 8
Page 1C	1	1	1	0	0	32K x 8
Page 1D	1	1	1	0	1	32K x 8
Page 1E	1	1	1	1	0	32K x 8
Page 1F	1	1	1	1	1	32K x 8
Total						1024K x 8

AUTO START AND PROGRAM FUNCTION KEYS

The first thing you have to do is make the programs easy to access. The C128 has an auto boot routine during power up and reset for cartridge, Internal ROM and disk (device 8). We will just cover the Internal ROM.

During the power up or reset routine your computer will look for CBM at the beginning of Internal ROM \$8000 or \$C000. Here is what it will look like:

<u>Bytes</u>	<u>Description</u>
x000-02	Cold start entry
x003-05	Warm start entry
x006	ID byte, \$00 for due nothing, anything else for auto-start
x007-09	"CBM" string

Note: x = \$8 (middle) or \$C (high)

\$8000 EA EA EA 4C 0A 80 FF 43 42 4D

\$8000	NOP	;COLD START
\$8001	NOP	
\$8002	NOP	
\$8003	JMP \$800A	;WARM START
\$8006	FF	;ID, \$00 OR \$01
\$8007	'CBM'	;CBM STRING
\$800A		;START OF PGM

It will make note in lower RAM that there is an auto-start in Internal ROM and return to the ROM during the PHENIOX routine and run the program at \$800A.

The program at \$800A will reprogram one of the Function Keys that will jump to the MENU of all the programs in your Internal ROM. Here is the Merlin listing for the program, you can find the source code on disk:

```

1
2 *****
3 * re-program a function key *
4 * x reg = f-key no. *
5 * y reg = length of string *
6 * acu = pointer to location of string $24/$25 *
7 * store bank no. of string loc in $26 *
8 *****
9
10 STLOC = $24 ;string location
11 BANK = $26 ;bank loc. of string
12 PGSW = $D700 ;switch page of int. ROM
13 PFKEY = $FF65 ;routine to reprogram
14 ; f-keys
15 CONFIG = $FF00 ;change memory
16 ; configuration
17
18 ORG $8000
19
8000: EA 20 NOP
8001: EA 21 NOP
8002: EA 22 NOP
8003: 4C 0A 80 23 JMP PGKEY
8006: FF 24 HEX FF
8007: 43 42 4D 25 TXT 'cbm'
26
800A: AD 00 FF 27 PGKEY LDA CONFIG ;get configuration
800D: 48 28 PHA ;put on stack
800E: A9 06 29 LDA #$06 ;int ROM & KERNAL
8010: 8D 00 FF 30 STA CONFIG ;set config
8013: 85 26 31 STA BANK ;store bank no. in $26
8015: A9 36 32 LDA #<STR ;get lo byte of string loc
8017: 85 24 33 STA STLOC ;store in $24
8019: A9 80 34 LDA #>STR ;get hi byte of string loc
801B: 85 25 35 STA STLOC+1 ;store in $25
801D: A2 0A 36 LDX #$0A ;Fkey number to prg
37 ; #$0A is help key
801F: A0 08 38 LDY #$08 ;no. of bytes for string
8021: A9 24 39 LDA #STLOC ;loc. of string
8023: 20 65 FF 40 JSR PFKEY ;program f-key
8026: A2 0F 41 LDX #$0F ;no. of bytes to relocate
8028: BD 3E 80 42 LOOP LDA JPMENU,X ;get byte
802B: 9D A0 10 43 STA $10A0,X ;store byte
802E: CA 44 DEX ;decrease X by 1
802F: 10 F7 45 BPL LOOP ;loop until finished
8031: 68 46 PLA ;get org. config
8032: 8D 00 FF 47 STA CONFIG ;set config
8035: 60 48 RTS ;return
49

```

```

50 *****
51 * new f-key definition, 8 bytes *
52 *****
53
8036: 53 59 53 54 STR      TXT      'sys4256',0d  ;this will system to
8039: 34 32 35 36 0D
55                                     ; jmpmenu at loc. $10A0
56
57 *****
58 * ml routine that will jmp to menu *
59 *****
60
803E: A9 06      61 JPMENU   LDA      #$06      ;internal ROM & kernal
8040: 8D 00 FF  62          STA      CONFIG    ;set configuration
8043: A9 00      63          LDA      #$00      ;page 0
8045: 8D 00 D7  64          STA      PGSW      ;switch page
8048: 8D 00 D7  65          STA      PGSW
804B: 4C 4E 80  66          JMP      MENU      ;jmp to internal ROM menu
67
804E: EA        68 MENU     NOP              ;append menu here
69

```

Place the object code for the above routine at the beginning of a 27256 EPROM, plug it into the Internal ROM socket and turn the computer on. It will reprogram the HELP key, when you push the HELP key, the computer will attempt to run the menu if you have one in ROM. The line marked in red is what determines which function key is programmed. Hex numbers \$01 - \$08 cover the function keys, \$09 will reprogram SHIFT RUN/STOP and \$0A will reprogram the HELP key

Below is what the MENU in the Megabit C128 Internal ROM looks like. You can find the Merlin source listing on the disk.

```

1
2 JPLDR   =      $10E0      ;jump to loader
3 PRESET  =      $10F0      ;program reset
4 WRMSTRT =      $4003      ;warm basic start
5 PRINT   =      $C00C      ;print to screen
6 PGSW    =      $D700      ;switch page of Int ROM
7 CONFIG  =      $FF00      ;set memory configuration
8 STOP    =      $FFE1      ;check stop key
9 GETIN   =      $FFE4      ;get a char in acu
10 RESET  =      $FFFC      ;hard reset
11
12          ORG      $1300
13
1300: A2 FF      14          LDX      #$FF
1302: 9A        15          TXS
1303: A2 16      16          LDX      #$16      ;internal rom with i/o
1305: 8E 01 D5  17          STX      $D501      ; at $ff01
1308: E8        18          INX              ;internal ROM w/o i/o
1309: 8E 02 D5  19          STX      $D502      ; at $ff02

```

130C:	58	20		CLI	
130D:	D8	21		CLD	
130E:	A5 D7	22		LDA	\$D7 ;ck for 40 or 80 col scrn
1310:	F0 0D	23		BEQ	SCN40 ;40 col scrn
1312:	AD 11 D0	24		LDA	\$D011 ;80 col scrn
1315:	29 6F	25		AND	#\$6F ;switch to fast mode
1317:	8D 11 D0	26		STA	\$D011 ; and blank 40-col
131A:	A9 01	27		LDA	#\$01
131C:	8D 30 D0	28		STA	\$D030
131F:	A9 00	29	SCN40	LDA	#\$00
1321:	85 C6	30		STA	\$C6
1323:	85 C7	31		STA	\$C7
1325:	85 FD	32		STA	\$FD ;clr menu counter
1327:	A2 0A	33		LDX	#\$0A
1329:	BD 29 15	34	LOOP1	LDA	LDRSTG,X
132C:	9D E0 10	35		STA	JMPLDR,X
132F:	CA	36		DEX	
1330:	10 F7	37		BPL	LOOP1
1332:	20 C3 14	38	NEWSC	JSR	PRTSCN
1335:	20 E1 FF	39	GETK	JSR	STOP
1338:	F0 2D	40		BEQ	QUIT
133A:	20 E4 FF	41		JSR	GETIN
133D:	F0 F6	42		BEQ	GETK
133F:	C9 20	43		CMP	#\$20
1341:	F0 16	44		BEQ	NXMN ;next menu
1343:	C9 41	45		CMP	#\$41 ;is it below #\$41?
1345:	90 EE	46		BCC	GETK ;yes, then go back
1347:	C9 60	47		CMP	#\$60 ;is it above #\$60?
1349:	B0 EA	48		BCS	GETK ;yes, then go back
134B:	38	49		SEC	
134C:	E9 41	50		SBC	#\$41 ; subtract #\$41
134E:	0A	51		ASL	;multi by 2
134F:	AA	52		TAX	;trans acu to X
1350:	BD 83 13	53		LDA	CMD+1,X ;get adrs loc hi
1353:	48	54		PHA	;put on stack
1354:	BD 82 13	55		LDA	CMD,X ;get adrs loc lo
1357:	48	56		PHA	;put on stack
1358:	60	57		RTS	;CMD entry
		58			
1359:	E6 FD	59	NXMN	INC	\$FD ;incr menu ctr
135B:	A5 FD	60		LDA	\$FD
135D:	C9 03	61		CMP	#\$03 ;is it #\$03?
135F:	D0 D1	62		BNE	NEWSC ;no, then prt nxt menu
1361:	A9 00	63	CLR	LDA	#\$00
1363:	85 FD	64		STA	\$FD ;clr menu ctr
1365:	F0 CB	65		BEQ	NEWSC ;print menu 1
		66			
1367:	A9 93	67	QUIT	LDA	#\$93 ;clr screen
1369:	20 0C C0	68		JSR	PRINT
136C:	A2 07	69		LDX	#\$07 ;trans basic jump
136E:	BD 7A 13	70	LOOP2	LDA	TOBAS,X ;to program reset
1371:	9D F0 10	71		STA	PRESET,X
1374:	CA	72		DEX	
1375:	10 F7	73		BPL	LOOP2
1377:	4C F0 10	74		JMP	PRESET
		75			
137A:	A9 00	76	TOBAS	LDA	#\$00

```

137C: 8D 00 FF 77          STA  CONFIG
137F: 4C 03 40 78          JMP  WRMSTRT
79
1382: F6 13      80  CMD    DA    MER-1      ;merlin 128
1384: FF 13      81          DA    PRO-1      ;promos 2.0
1386: 08 14      82          DA    DE-1       ;my disk editor
1388: 0B 14      83          DA    FK-1       ;function key display
138A: 0E 14      84          DA    VW-1       ;vizawrite 128
138C: 17 14      85          DA    VS-1       ;vizastar 128
138E: 20 14      86          DA    SQR-1      ;seq file reader
1390: 23 14      87          DA    BGEN-1     ;begin & end adrs
1392: 26 14      88          DA    FTFC-1     ;fastrac file copier
1394: 2F 14      89          DA    DIRE-1     ;directory editor
1396: 38 14      90          DA    C080-1     ;color 80col
1398: 3B 14      91          DA    DM-1       ;basic data maker
139A: 3E 14      92          DA    MON64-1    ;monitor 64
139C: 47 14      93          DA    ZED-1      ;zed 128
139E: 50 14      94          DA    MERGP-1    ;basic merge +
13A0: 53 14      95          DA    MAVFC-1    ;maverick file copy
13A2: 5C 14      96          DA    MAVTE-1    ;maverick track editor
13A4: 65 14      97          DA    SGL41-1    ;single 41 data copy
13A6: 6E 14      98          DA    DUL41-1    ;dual 41 data copy
13A8: 77 14      99          DA    SGLNY-1    ;single nybbler
13AA: 80 14     100          DA    DULNY-1    ;dual nybbler
13AC: 89 14     101          DA    SGL81-1    ;single 81 data copy
13AE: 92 14     102          DA    MAVFT-1    ;maverick file tracer
13B0: 9B 14     103          DA    MAVTS-1    ;maverick track & sector ed
13B2: A4 14     104          DA    VDC-1      ;64k vdc ram test
13B4: AD 14     105          DA    REU-1      ;reu test
13B6: 34 13     106          DA    GETK-1     ;fill
13B8: BF 13     107          DA    BAS8-1     ;basic 8, \ key
13BA: 34 13     108          DA    GETK-1     ;fill
13BC: C7 13     109          DA    SERV-1     ;servant, ^ key
13BE: CF 13     110          DA    KEYD-1     ;keydos, _ key
111
112 ***** loaders for 32k programs *****
113
13C0: 20 D8 13 114  BAS8    JSR    TROM
13C3: A9 0D      115          LDA    #$0D      ;ld page no
13C5: 4C F0 10 116          JMP    PRESET
117
13C8: 20 D8 13 118  SERV    JSR    TROM
13CB: A9 0E      119          LDA    #$0E      ;ld page no
13CD: 4C F0 10 120          JMP    PRESET
121
13D0: 20 D8 13 122  KEYD    JSR    TROM
13D3: A9 0F      123          LDA    #$0F      ;ld page no
13D5: 4C F0 10 124          JMP    PRESET
125
13D8: A9 93      126  TROM    LDA    #$93      ;clr the scrn
13DA: 20 0C C0 127          JSR    PRINT
13DD: A2 0D      128          LDX    #$0D
13DF: BD E9 13 129  LOOP3   LDA    TOROM,X
13E2: 9D F0 10 130          STA    PRESET,X
13E5: CA          131          DEX
13E6: 10 F7      132          BPL    LOOP3
13E8: 60          133          RTS

```

```

134
13E9: 8D 00 D7 135 TOROM STA PGSW
13EC: 8D 00 D7 136 STA PGSW
13EF: A9 00 137 LDA #$00
13F1: 8D 00 FF 138 STA CONFIG
13F4: 6C FC FF 139 JMP (RESET)
140
141 ***** to loaders of smaller programs *****
142
13F7: A9 01 143 MER LDA #$01 ;ld page no
13F9: A2 00 144 LDX #$00 ;ld low byte
13FB: A0 80 145 LDY #$80 ;ld high byte
13FD: 4C B7 14 146 JMP GO ;jmp to merlin
147
1400: A9 03 148 PRO LDA #$03 ;ld page no
1402: A2 00 149 LDX #$00 ;ld low byte
1404: A0 90 150 LDY #$90 ;ld high byte
1406: 4C B7 14 151 JMP GO ;jmp to promos
152
1409: 4C 00 B0 153 DE JMP $B000 ;jmp to disk edit
154
140C: 4C 00 97 155 FK JMP $9700 ;jmp to fkeys
156
140F: A9 0A 157 VW LDA #$0A ;ld page no
1411: A2 00 158 LDX #$00 ;ld low byte
1413: A0 85 159 LDY #$85 ;ld high byte
1415: 4C B7 14 160 JMP GO ;jmp to vizawrite
161
1418: A9 0C 162 VS LDA #$0C ;ld page no
141A: A2 00 163 LDX #$00 ;ld low byte
141C: A0 90 164 LDY #$90 ;ld high byte
141E: 4C B7 14 165 JMP GO ;jmp to vizastar
166
1421: 4C 00 9D 167 SQR JMP $9D00 ;jmp to seq reader
168
1424: 4C 00 A4 169 BGEN JMP $A400 ;jmp to begin & end
170
1427: A9 03 171 FTFC LDA #$03 ;ld page no
1429: A2 00 172 LDX #$00 ;ld low byte
142B: A0 91 173 LDY #$91 ;ld high byte
142D: 4C B7 14 174 JMP GO ;jmp to ft file copy
175
1430: A9 02 176 DIRE LDA #$02 ;ld page no
1432: A2 00 177 LDX #$00 ;ld low byte
1434: A0 81 178 LDY #$81 ;ld high byte
1436: 4C B7 14 179 JMP GO ;jmp to dir edit
180
1439: 4C 00 9A 181 C080 JMP $9A00 ;jmp to color80
182
143C: 4C 90 A4 183 DM JMP $A490 ;jmp to data maker
184
143F: A9 02 185 MON64 LDA #$02 ;ld page no
1441: A2 00 186 LDX #$00 ;ld low byte
1443: A0 80 187 LDY #$80 ;ld high byte
1445: 4C B7 14 188 JMP GO ;jmp to monitor64
189
1448: A9 02 190 ZED LDA #$02 ;ld page no

```

144A:	A2 00	191		LDX	#\$00	;ld low byte
144C:	A0 82	192		LDY	#\$82	;ld high byte
144E:	4C B7 14	193		JMP	GO	;jmp to zed 128
		194				
1451:	4C 0E 93	195	MERGP	JMP	\$930E	;jmp to merge +
		196				
1454:	A9 08	197	MAVFC	LDA	#\$08	;ld page no
1456:	A2 00	198		LDX	#\$00	;ld low byte
1458:	A0 80	199		LDY	#\$80	;ld high byte
145A:	4C B7 14	200		JMP	GO	;jmp to mav file copy
		201				
145D:	A9 04	202	MAVTE	LDA	#\$04	;ld page no
145F:	A2 00	203		LDX	#\$00	;ld low byte
1461:	A0 80	204		LDY	#\$80	;ld high byte
1463:	4C B7 14	205		JMP	GO	;jmp to mav trk edit
		206				
1466:	A9 07	207	SGL41	LDA	#\$07	;ld page no
1468:	A2 00	208		LDX	#\$00	;ld low byte
146A:	A0 90	209		LDY	#\$90	;ld high byte
146C:	4C B7 14	210		JMP	GO	;jmp to 41 data copy
		211				
146F:	A9 06	212	DUL41	LDA	#\$06	;ld page no
1471:	A2 00	213		LDX	#\$00	;ld low byte
1473:	A0 90	214		LDY	#\$90	;ld high byte
1475:	4C B7 14	215		JMP	GO	;jmp to dual data copy
		216				
1478:	A9 05	217	SGLNY	LDA	#\$05	;ld page no
147A:	A2 00	218		LDX	#\$00	;ld low byte
147C:	A0 90	219		LDY	#\$90	;ld high byte
147E:	4C B7 14	220		JMP	GO	;jmp to sgl nybbler
		221				
1481:	A9 05	222	DULNY	LDA	#\$05	;ld page no
1483:	A2 00	223		LDX	#\$00	;ld low byte
1485:	A0 91	224		LDY	#\$91	;ld high byte
1487:	4C B7 14	225		JMP	GO	;jmp to dual nybbler
		226				
148A:	A9 04	227	SGL81	LDA	#\$04	;ld page no
148C:	A2 00	228		LDX	#\$00	;ld low byte
148E:	A0 81	229		LDY	#\$81	;ld high byte
1490:	4C B7 14	230		JMP	GO	;jmp to 81 data copy
		231				
1493:	A9 06	232	MAVFT	LDA	#\$06	;ld page no
1495:	A2 00	233		LDX	#\$00	;ld low byte
1497:	A0 91	234		LDY	#\$91	;ld high byte
1499:	4C B7 14	235		JMP	GO	;jmp to file tracer
		236				
149C:	A9 07	237	MAVTS	LDA	#\$07	;ld page no
149E:	A2 00	238		LDX	#\$00	;ld low byte
14A0:	A0 91	239		LDY	#\$91	;ld high byte
14A2:	4C B7 14	240		JMP	GO	;jmp to trk & sec edit
		241				
14A5:	A9 08	242	VDC	LDA	#\$08	;ld page no
14A7:	A2 00	243		LDX	#\$00	;ld low byte
14A9:	A0 81	244		LDY	#\$81	;ld high byte
14AB:	4C B7 14	245		JMP	GO	;jmp to vdc ram test
		246				
14AE:	A9 08	247	REU	LDA	#\$08	;ld page no

14B0:	A2 00	248		LDX	#\$00	;ld low byte
14B2:	A0 82	249		LDY	#\$82	;ld high byte
14B4:	4C B7 14	250		JMP	G0	;jmp to reu test
		251				
14B7:	8D E1 10	252	G0	STA	\$10E1	
14BA:	8E E9 10	253		STX	\$10E9	
14BD:	8C EA 10	254		STY	\$10EA	
14C0:	4C E0 10	255		JMP	\$10E0	;jmp to loader
		256				
14C3:	A5 FD	257	PRTSCN	LDA	\$FD	
14C5:	C9 00	258		CMP	#\$00	
14C7:	F0 0B	259		BEQ	MU1	;menu 1
14C9:	C9 01	260		CMP	#\$01	
14CB:	F0 12	261		BEQ	MU2	;menu 2
14CD:	C9 02	262		CMP	#\$02	
14CF:	F0 19	263		BEQ	MU3	;menu 3
14D1:	4C 61 13	264		JMP	CLR	
		265				
14D4:	A9 34	266	MU1	LDA	#<MENU1	;get lo byte of menu1
14D6:	85 FA	267		STA	\$FA	;store in \$fa
14D8:	A9 15	268		LDA	#>MENU1	;get hi byte of menu1
14DA:	85 FB	269		STA	\$FB	;store in \$fb
14DC:	4C F2 14	270		JMP	PRT	;prt menu1
		271				
14DF:	A9 CC	272	MU2	LDA	#<MENU2	
14E1:	85 FA	273		STA	\$FA	
14E3:	A9 16	274		LDA	#>MENU2	
14E5:	85 FB	275		STA	\$FB	
14E7:	4C F2 14	276		JMP	PRT	
		277				
14EA:	A9 64	278	MU3	LDA	#<MENU3	
14EC:	85 FA	279		STA	\$FA	
14EE:	A9 18	280		LDA	#>MENU3	
14F0:	85 FB	281		STA	\$FB	
14F2:	A0 00	282	PRT	LDY	#\$00	
14F4:	B1 FA	283	LOOP4	LDA	(\$FA), Y	
14F6:	F0 10	284		BEQ	SKIP2	
14F8:	C8	285		INY		
14F9:	C0 00	286		CPY	#\$00	
14FB:	D0 02	287		BNE	SKIP	
14FD:	E6 FB	288		INC	\$FB	
14FF:	C9 FF	289	SKIP	CMP	#\$FF	
1501:	F0 06	290		BEQ	SKIP1	
1503:	20 0C C0	291		JSR	PRINT	
1506:	D0 EC	292		BNE	LOOP4	
1508:	60	293	SKIP2	RTS		
		294				
1509:	84 FC	295	SKIP1	STY	\$FC	
150B:	A5 D7	296		LDA	\$D7	
150D:	D0 01	297		BNE	SKIP5	
150F:	C8	298		INY		
1510:	B1 FA	299	SKIP5	LDA	(\$FA), Y	
1512:	AA	300		TAX		
1513:	E0 00	301		CPX	#\$00	
1515:	F0 08	302		BEQ	NOSPC	
1517:	A9 20	303		LDA	#\$20	
1519:	20 0C C0	304	NXSP	JSR	PRINT	

```

151C: CA      305      DEX
151D: 10 FA    306      BPL    NXSP
151F: E6 FC    307  NOSPC    INC    $FC
1521: E6 FC    308      INC    $FC
1523: A4 FC    309      LDY    $FC
1525: D0 CD    310      BNE    LOOP4
1527: F0 CB    311      BEQ    LOOP4
312
313 ***** routine to loaders *****
314
1529: A9 00    315  LDRSTG    LDA    #$00
152B: 8D 00 D7 316      STA    PGSW
152E: 8D 00 D7 317      STA    PGSW
1531: 4C 00 00 318      JMP    $0000
319
320 *****
321 * ff = end of sentence, next is 80-col spaces, *
322 * next is 40-col spaces, 12 = reverse print *
323 * 00 before ff = end of menu *
324 *****
325
1534: 93 8E 0D 326  MENU1    HEX    93,8E,0D,0D,FF,1E,0A,12
153C: 20 49 4E 327      TXT    ' internal rom menu '0d,0d,ff,20,0c
1554: 42 59 20 328      TXT    'by d.c. newbury'0d,0d,ff,22,0e
1568: 4D 45 4E 329      TXT    'menu 1 of 3'0d,0d,ff,14,00
330
1578: 41 29 20 331      TXT    'a) merlin c128      '
158C: 42 29 20 332      TXT    'b) promos 2.0      '0d,ff,14,00
333
15A4: 43 29 20 334      TXT    'c) my disk editor      '
15B8: 44 29 20 335      TXT    'd) function keys      '0d,ff,14,00
336
15D0: 45 29 20 337      TXT    'e) viza write 128      '
15E4: 46 29 20 338      TXT    'f) viza star 128      '0d,ff,14,00
339
15FC: 47 29 20 340      TXT    'g) seq reader 128      '
1610: 48 29 20 341      TXT    'h) begin & end adrs '0d,ff,14,00
342
1628: 49 29 20 343      TXT    'i) fastrac filecopy      '
163C: 4A 29 20 344      TXT    'j) directory editor '0d,ff,14,00
345
1654: 4B 29 20 346      TXT    'k) color 80 col      '
1668: 4C 29 20 347      TXT    'l) basic data maker '0d,ff,14,00
348
1680: 4D 29 20 349      TXT    'm) monitor 64      '0d,0d,ff,1e,0a,12
350
169A: 20 50 52 351      TXT    ' press stop to exit '0d,ff,1c,08,12
16B3: 20 50 52 352      TXT    ' press space next menu '0d,00
353
354
16CC: 93 8E 0D 355  MENU2    HEX    93,8E,0D,0D,FF,1E,0A,12
16D4: 20 49 4E 356      TXT    ' internal rom menu '0d,0d,ff,20,0c
16EC: 42 59 20 357      TXT    'by d.c. newbury'0d,0d,ff,22,0e
1700: 4D 45 4E 358      TXT    'menu 2 of 3'0d,0d,ff,14,00
359
1710: 4E 29 20 360      TXT    'n) zed 128      '
1724: 4F 29 20 361      TXT    'o) basic merge +      '0d,ff,14,00

```

		362			
173C:	50 29 20	363	TXT	'p) mav file copy	'
1750:	51 29 20	364	TXT	'q) mav track editor	'0d,ff,14,00
		365			
1768:	52 29 20	366	TXT	'r) sgl 41 data copy	'
177C:	53 29 20	367	TXT	's) dul 41 data copy	'0d,ff,14,00
		368			
1794:	54 29 20	369	TXT	't) single nybbler	'
17A8:	55 29 20	370	TXT	'u) dual nybbler	'0d,ff,14,00
		371			
17C0:	56 29 20	372	TXT	'v) sgl 81 data copy	'
17D4:	57 29 20	373	TXT	'w) mav file tracer	'0d,ff,14,00
		374			
17EC:	58 29 20	375	TXT	'x) mav trk&sec edit	'
1800:	59 29 20	376	TXT	'y) 64k vdc ram test	'0d,ff,14,00
		377			
1818:	5A 29 20	378	TXT	'z) reu test	'0d,0d,ff,1e,0a,12
		379			
1832:	20 50 52	380	TXT	' press stop to exit	'0d,ff,1c,08,12
184B:	20 50 52	381	TXT	' press space next menu	'0d,00
		382			
		383			
1864:	93 8E 0D	384	MENU3	HEX	93,8E,0D,0D,FF,1E,0A,12
186C:	20 49 4E	385		TXT	' internal rom menu
1884:	42 59 20	386		TXT	'by d.c. newbury'0d,0d,ff,22,0e
1898:	4D 45 4E	387		TXT	'menu 3 of 3'0d,0d,ff,14,00
		388			
18A8:	5C 29 20	389		TXT	'£) basic 8
18BC:	5E 29 20	390		TXT	'↑) servant
		391			'0d,ff,14,00
18D4:	5F 29 20	392		TXT	'←) key dos
18E8:	20 20 20	393		TXT	'
		394			'0d,ff,14,00
1900:	20 20 20	395		TXT	'
1914:	20 20 20	396		TXT	'
		397			'0d,ff,14,00
192C:	20 20 20	398		TXT	'
1940:	20 20 20	399		TXT	'
		400			'0d,ff,14,00
1958:	20 20 20	401		TXT	'
196C:	20 20 20	402		TXT	'
		403			'0d,ff,14,00
1984:	20 20 20	404		TXT	'
1998:	20 20 20	405		TXT	'
		406			'0d,ff,14,00
19B0:	20 20 20	407		TXT	'
		408			'0d,0d,ff,1e,0a,12
19CA:	20 50 52	409		TXT	' press stop to exit
19E3:	20 50 52	410		TXT	' press space next menu
		411			'0d,00

You can load and run C128 programs or C64 programs from the Internal ROM area. But when you quit the C64 programs you will have to reset the computer to get back into C128 mode.

You can run BASIC or ML programs from Internal ROM, C128 or C64. We will look at loaders for the different types of programs. The C64 programs will have to be transferred to RAM first, then switched to C64 mode. You cannot get access to Internal ROM from C64 mode.

There is something else that I need to mention about using programs in C64 mode. If the program is a large program that extends past the start of BASIC (\$A000) or loads into the beginning of that area, your program will be corrupted when you switch to C64 mode.

When the C64 is initializing it will check RAM to find the end of basic RAM. When it bumps into ROM at \$A000, \$9FFF is the end of BASIC RAM. It checks RAM by saving a byte at each location, stores a byte \$55 and then checks to see if it is \$55 and then puts the original byte back into the RAM location. When it gets to the edge of ROM (\$A000), it tries to store byte \$55 at that location and it will, only in RAM. When it reads that location it will read byte \$94 instead of byte \$55. It will stop checking RAM and move on without restoring the original byte at RAM location \$A000. If there is important data at that location, your program will crash when it gets there. So keep that in mind when loading and starting C64 programs.

The next program is a loader used to transfer ZED 128 to RAM from Internal ROM and start the program with a basic one liner. The program is in two parts. The first part will transfer the loader to RAM at \$0C00, jump to the RAM loader and continue on.

At the end of the listing (line 92) is where the data is that tells the loader where the program is in ROM (\$85BF - \$C814) and where in RAM (\$1C01) it's to be loaded.

When finished transferring the program (line 74), it will set the computer up to run the program. Starting at line 75 it will switch memory to RAM 0 and switch in all system ROMs. Then switch Internal ROM to page 0, if you don't have the reset option of the adapter hooked up then you will have to switch in page 0 at the end of each loader before you start the program. That way when you push the reset button the Internal ROM will re-program the function key.

Next ZED 128 requires that the pre-configuration registers be restored to default values, then sets device 8 as the default disk drive. At line 90 you will see the basic one liner (sys7200 and return). To run it you will have to put the low byte of the address location - 1 in the X register and the high byte of the address location in the Y register and jump to \$AFA5 (execute a line in basic). See the source listing at line 86, X = \$7D and Y = \$0C that would mean that the one liner is at \$0C7E.

```

2
3 *****
4 *   LOADER FOR ZED 128   *
5 *****
6 *
7             ORG     $8200
8 *
9 FROM      =      $60      ;loc at start of ROM
10 TO       =      $62      ;load to start of RAM
11 END      =      $64      ;loc at end of ROM
12 DRV      =      $BA      ;disk drive number
13 STBAS     =      $AFA5    ;execute a line
14 PRTSCN    =      $C00C    ;print char to screen
15 PCRA      =      $D501    ;mem pre-config A reg
16 PCRB      =      $D502    ;mem pre-config B reg
17 PGSW      =      $D700    ;int ROM page switch
18 CONFIG    =      $FF00    ;mem configuration reg
19 LCRA      =      $FF01    ;mem load-config A reg
20 LCRB      =      $FF02    ;mem load-config B reg
21
8200: A9 92    22             LDA     #RAMLD+LAST;trans loader routine
8202: 85 FA    23             STA     $FA      ; to RAM at $0C00
8204: A2 00    24             LDX     #$00
8206: BD 14 82 25 NXTRAN    LDA     BEGIN+3,X
8209: 9D 00 0C 26             STA     RAMLD,X
820C: E8       27             INX
820D: E4 FA    28             CPX     $FA
820F: D0 F5    29             BNE     NXTRAN
8211: 4C 00 0C 30 BEGIN     JMP     RAMLD      ;jump to loader routine
31
32             ORG     $0C00
33
0C00: A9 93    34 RAMLD     LDA     #$93      ;clear screen
0C02: 20 0C C0 35             JSR     PRTSCN
0C05: 78       36             SEI              ;set the interrupt
0C06: A2 00    37             LDX     #0       ;clear information indexer
0C08: 8D 01 FF 38 NXPG      STA     LCRA      ;RAM 0, Int ROM, I/O
0C0B: BD 90 0C 39             LDA     PGNO,X    ;get first rom page no.
0C0E: 8D 00 D7 40             STA     PGSW      ;select page at $d700
0C11: 8D 00 D7 41             STA     PGSW
0C14: BD 8A 0C 42             LDA     FROMLO,X  ;from rom low byte
0C17: 85 60    43             STA     FROM
0C19: BD 8B 0C 44             LDA     FROMHI,X  ;from rom high byte
0C1C: 85 61    45             STA     FROM+1
0C1E: BD 8C 0C 46             LDA     TOLO,X    ;to ram low byte
0C21: 85 62    47             STA     TO
0C23: BD 8D 0C 48             LDA     TOHI,X    ;to ram high byte
0C26: 85 63    49             STA     TO+1
0C28: BD 8E 0C 50             LDA     ENDLO,X   ;end adrs rom low byte
0C2B: 85 64    51             STA     END
0C2D: BD 8F 0C 52             LDA     ENDHI,X   ;end adrs rom high byte
0C30: 85 65    53             STA     END+1
0C32: A0 00    54             LDY     #$00
0C34: 8D 02 FF 55             STA     LCRB      ;RAM 0, Int ROM
0C37: B1 60    56 LOOP1     LDA     (FROM),Y    ;get byte from rom
0C39: 91 62    57             STA     (TO),Y     ;store byte in ram
0C3B: A5 61    58             LDA     FROM+1    ;ck for end lo byte

```

```

0C3D: C5 65      59      CMP      END+1
0C3F: D0 06      60      BNE      SKIP1      ;not end, then cont.
0C41: A5 60      61      LDA      FROM      ;ck for end hi byte
0C43: C5 64      62      CMP      END
0C45: F0 0E      63      BEQ      SKIP3      ;if end, ck for nx page
0C47: E6 60      64      SKIP1    INC      FROM      ;inc from lo
0C49: D0 02      65      BNE      SKIP2
0C4B: E6 61      66      INC      FROM+1    ;inc from hi
0C4D: E6 62      67      SKIP2    INC      TO      ;inc to lo
0C4F: D0 E6      68      BNE      LOOP1
0C51: E6 63      69      INC      TO+1      ;inc to hi
0C53: D0 E2      70      BNE      LOOP1    ;get and store nx byte
0C55: E8          71      SKIP3    INX
0C56: BD 90 0C    72      LDA      PGNO,X    ;get next page
0C59: C9 FF      73      CMP      #$FF      ;ck for end of transfer
0C5B: D0 AB      74      BNE      NXPG      ;if not $ff, jmp to nx page
0C5D: A9 00      75      LDA      #0      ;switch to
0C5F: 8D 00 FF    76      STA      CONFIG    ; RAM with all sys ROMs
0C62: 8D 00 D7    77      STA      PGSW      ; page 0 Int ROM
0C65: 8D 00 D7    78      STA      PGSW
0C68: A9 3F      79      LDA      #$3F      ;restore pre-conf regs
0C6A: 8D 01 D5    80      STA      PCRA
0C6D: A9 7F      81      LDA      #$7F
0C6F: 8D 02 D5    82      STA      PCRB
0C72: A9 08      83      LDA      #$08      ;set device 8 as
0C74: 85 BA      84      STA      DRV      ; default drive
0C76: 58          85      CLI      ;clr the interrupt
0C77: A2 7D      86      LDX      #<BST-1    ;lo adrs -1 of start of bas
0C79: A0 0C      87      LDY      #>BST      ;hi adrs of start of bas
0C7B: 4C A5 AF    88      JMP      STBAS      ;jump execute a line ($AFA5)
                        89
0C7E: 53 59 53 90  BST    TXT      'sys7200',00,0d,00,00,00
0C81: 37 32 30 30 00 0D 00 00
0C89: 00

                        91
                        92      *** TRANSFER INFORMATION FOR ZED ***
                        93      *** $FF INDICATES END OF TRANSFER ***
                        94
0C8A: BF          95      FROMLO    HEX      BF
0C8B: 85          96      FROMHI    HEX      85
0C8C: 01          97      TOLO      HEX      01
0C8D: 1C          98      TOHI      HEX      1C
0C8E: 14          99      ENDLO     HEX      14
0C8F: C8         100     ENDHI     HEX      C8
0C90: 02 FF      101     PGNO      HEX      02, FF
                        102     LAST

```

The next loader is basically the same as the previous loader, except it does not use BASIC start. Beginning at line 70 the program will switch in all system ROMs and switch to Page 0 of the Internal ROM. Then clear the interrupt and jump to the beginning of the program at \$2000.

```

2
3
4      *      *      *
5      *      *      *
6      *
7      *      ORG      $8000
8      *
9      FROM      =      $60
10     TO        =      $62
11     END        =      $64
12     PRTSCN     =      $C00C
13     PGSW       =      $D700
14     CONFIG     =      $FF00
15     LCRA       =      $FF01
16     LCRB       =      $FF02
17
8000: A9 6E      18             LDA      #RAMLD+LAST
8002: 85 FA      19             STA      $FA
8004: A2 00      20             LDX      #$00
8006: BD 14 80   21     NXTRAN  LDA      BEGIN+3,X
8009: 9D 00 0C   22             STA      RAMLD,X
800C: E8         23             INX
800D: E4 FA      24             CPX      $FA
800F: D0 F5      25             BNE      NXTRAN
8011: 4C 00 0C   26     BEGIN  JMP      RAMLD
27
28             ORG      $0C00
29
0C00: A9 93      30     RAMLD  LDA      #$93          ;clear screen
0C02: 20 0C C0   31             JSR      PRTSCN
0C05: 78         32             SEI
0C06: A2 00      33             LDX      #0          ;clear information indexer
0C08: BD 6C 0C   34             LDA      PGNO,X        ;get first rom page no.
0C0B: 8D 01 FF   35     NXPG   STA      LCRA          ;internal rom w/ io
0C0E: 8D 00 D7   36             STA      PGSW         ;select page at $d700
0C11: BD 66 0C   37             LDA      FROMLO,X       ;from rom low byte
0C14: 85 60      38             STA      FROM
0C16: BD 67 0C   39             LDA      FROMHI,X        ;from rom high byte
0C19: 85 61      40             STA      FROM+1
0C1B: BD 68 0C   41             LDA      TOLO,X          ;to ram low byte
0C1E: 85 62      42             STA      TO
0C20: BD 69 0C   43             LDA      TOHI,X          ;to ram high byte
0C23: 85 63      44             STA      TO+1
0C25: BD 6A 0C   45             LDA      ENDLO,X        ;end adrs rom low byte
0C28: 85 64      46             STA      END
0C2A: BD 6B 0C   47             LDA      ENDHI,X        ;end adrs rom high byte
0C2D: 85 65      48             STA      END+1
0C2F: A0 00      49             LDY      #$00
0C31: 8D 02 FF   50             STA      LCRB          ;all int rom/ram 0
0C34: B1 60      51     LOOP1  LDA      (FROM),Y      ;get byte from rom
0C36: 91 62      52             STA      (TO),Y        ;store byte in ram
0C38: A5 61      53             LDA      FROM+1       ;ck for end lo byte

```

```

0C3A: C5 65      54          CMP    END+1
0C3C: D0 06      55          BNE    SKIP1      ;not end, then cont.
0C3E: A5 60      56          LDA    FROM      ;ck for end hi byte
0C40: C5 64      57          CMP    END
0C42: F0 0E      58          BEQ    SKIP3      ;if end, then ck for nx page
0C44: E6 60      59      SKIP1  INC    FROM      ;inc from lo
0C46: D0 02      60          BNE    SKIP2
0C48: E6 61      61          INC    FROM+1    ;inc from hi
0C4A: E6 62      62      SKIP2  INC    TO      ;inc to lo
0C4C: D0 E6      63          BNE    LOOP1
0C4E: E6 63      64          INC    TO+1      ;inc to hi
0C50: D0 E2      65          BNE    LOOP1      ;get and store nx byte
0C52: E8         66      SKIP3  INX
0C53: BD 6C 0C   67          LDA    PGNO,X    ;get next page
0C56: C9 FF      68          CMP    #$FF      ;ck for end of transfer
0C58: D0 B1      69          BNE    NXPG      ;if not $ff, jmp to nx page
0C5A: A9 00      70          LDA    #0        ;switch to sys ROMs
0C5C: 8D 00 FF   71          STA    CONFIG
0C5F: 8D 00 D7   72          STA    PGSW      ;internal ROM pg to 0
0C62: 58         73          CLI
0C63: 4C 00 1C   74          JMP    $2000      ;start Merlin 128
                                75
                                76      *** TRANSFER INFORMATION FOR MERLIN ***
                                77      *** $FF INDICATES END OF TRANSFER ***
                                78
0C66: 00         79      FROMLO  HEX    00
0C67: 81         80      FROMHI  HEX    81
0C68: 40         81      TOLO    HEX    40
0C69: 1A         82      TOHI    HEX    1A
0C6A: 49         83      ENDLO   HEX    49
0C6B: F0         84      ENDHI   HEX    F0
0C6C: 01 FF      85      PGNO    HEX    01, FF
                                86      LAST

```

The next loader is for VizaWrite Word Processor, you'll notice it has two loaders. The first one is for the main program which is quite large. It loads from \$0B00 to \$FEDF and it takes two 32K ROM pages to hold it all. The next loader will load a small amount of code that goes from \$0400 to \$089A.

VizaWrite and VizaStar both have a small amount of ROM code that is plugged into the cartridge port. It works like a dongle, if it's not there, then the program will crash. I looked through the code of the main program and found where the program checks for the cartridge ROM, changed it to look at Internal ROM instead. So, the last thing the second loader will do before starting the program (\$0B1A) is to switch in the page (line 59) that has the ROM code at \$8000 (Page 3).

```

2
3 *****
4 *  VizaWrite LOADER FOR PAGED EPROM  *
5 *****
6 *
7             ORG      $8500
8 *
9  FROM      =      $60
10 TO        =      $62
11 END        =      $64
12 PGSW       =      $D700      ;page select
13 CONFIG     =      $FF00
14 LCRA       =      $FF01
15 LCRB       =      $FF02
16 SWAPPER    =      $FF5F      ;40,80-col screen swapper
17 PRIMM      =      $FF7D      ;print text to screen routine
18 SECLDR     =      $8700      ;adrs for $0400 ldr
19
8500: A9 06    20             LDA      #$06      ;RAM 0 & Int ROM,I/O,kernal
8502: 8D 00 FF 21             STA      CONFIG
8505: A5 D7    22             LDA      $D7      ;ck for 80-col
8507: D0 03    23             BNE      N040
8509: 20 5F FF 24             JSR      SWAPPER    ;swap from 40 to 80-col
850C: 20 7D FF 25  N040      JSR      PRIMM      ;print copyright notice
850F: 93 0D 0D 26             HEX      93,0D,0D,0D,0D,0D
8512: 0D 0D 0D
8515: 09 09 09 27             HEX      09,09,09,09
8518: 09
8519: 20 20 56 28             TXT      ' viza'0d,0d,09,09,09,09
851C: 49 5A 41 0D 0D 09 09 09
8524: 09
8525: 53 4F 46 29             TXT      'software'0d,0d,09,09,09,09
8528: 54 57 41 52 45 0D 0D 09
8530: 09 09 09
8533: 20 20 31 30             TXT      ' 128'0d,0d,0d,09,09,09,09
8536: 32 38 0D 0D 0D 09 09 09
853E: 09
853F: 56 49 5A 31             TXT      'vizawrite'0d,0d,09,09,09,09
8542: 41 57 52 49 54 45 0D 0D
854A: 09 09 09
854D: 20 20 20 32             TXT      '          by kelvin lacy'0d,0d,0d,09,09
8550: 20 20 42 59 20 4B 45 4C
8558: 56 49 4E 20 4C 41 43 59
8560: 0D 0D 0D 09 09
8565: 20 20 20 33             TXT      '          copyright 1985 viza software
ltd.'0d,00
8568: 20 43 4F 50 59 52 49 47
8570: 48 54 20 31 39 38 35 20
8578: 56 49 5A 41 20 53 4F 46
8580: 54 57 41 52 45 20 4C 54
8588: 44 2E 0D 00

```

```

858C: A2 00      35      LDX    #$00      ;transfer loader to $0400
858E: BD 9A 85   36      NXTRAN LDA    BEGIN+3,X
8591: 9D 00 04   37      STA    $0400,X
8594: E8         38      INX
8595: D0 F7      39      BNE    NXTRAN
8597: 4C 00 04   40      BEGIN JMP    RAMLD      ;jmp to loader
                        41
                        42
                        43
0400: 78         44      RAMLD  SEI          ;set the interrupt
0401: A2 00      45      LDX    #$00      ;clear information indexer
0403: BD 6F 04   46      LDA    PGNO,X    ;get first rom page no.
0406: 8D 01 FF   47      NXPG   STA    LCRA
0409: 8D 00 D7   48      STA    PGSW      ;select page at $D700
040C: BD 63 04   49      LDA    FROMLO,X   ;from rom low byte
040F: 85 60      50      STA    FROM
0411: BD 65 04   51      LDA    FROMHI,X   ;from rom high byte
0414: 85 61      52      STA    FROM+1
0416: BD 67 04   53      LDA    TOLO,X    ;to ram low byte
0419: 85 62      54      STA    TO
041B: BD 69 04   55      LDA    TOHI,X    ;to ram high byte
041E: 85 63      56      STA    TO+1
0420: BD 6B 04   57      LDA    ENDLO,X   ;end adrs rom low byte
0423: 85 64      58      STA    END
0425: BD 6D 04   59      LDA    ENDHI,X   ;end adrs rom high byte
0428: 85 65      60      STA    END+1
042A: 8D 02 FF   61      STA    LCRB
042D: A0 00      62      LDY    #$00
042F: B1 60      63      LOOP1  LDA    (FROM),Y   ;get byte from rom
0431: 91 62      64      STA    (TO),Y   ;store byte in ram
0433: A5 61      65      LDA    FROM+1   ;ck for end lo byte
0435: C5 65      66      CMP    END+1
0437: D0 06      67      BNE    SKIP1      ;not end, then cont.
0439: A5 60      68      LDA    FROM      ;ck for end hi byte
043B: C5 64      69      CMP    END
043D: F0 0E      70      BEQ    SKIP3      ;if end, then ck for nx page
043F: E6 60      71      SKIP1  INC    FROM      ;inc from lo
0441: D0 02      72      BNE    SKIP2
0443: E6 61      73      INC    FROM+1   ;inc from hi
0445: E6 62      74      SKIP2  INC    TO      ;inc to lo
0447: D0 E6      75      BNE    LOOP1
0449: E6 63      76      INC    TO+1      ;inc to hi
044B: D0 E2      77      BNE    LOOP1      ;get and store nx byte
044D: E8         78      SKIP3  INX
044E: BD 6F 04   79      LDA    PGNO,X    ;get next page
0451: C9 FF      80      CMP    #$FF      ;ck for end of transfer
0453: D0 B1      81      BNE    NXPG      ;if not $ff, then jmp to nx
page
0455: A9 06      82      LDA    #$06      ;switch cr to RAM 0 & Int ROM
0457: 8D 00 FF   83      STA    CONFIG
045A: 58         84      CLI
045B: A9 0A      85      LDA    #$0A      ;switch to page for $0400 ldr
045D: 8D 00 D7   86      STA    PGSW
0460: 4C 00 87   87      JMP    SECLDR      ;jmp to $0400 loader
                        88

```

```

89 *** TRANSFER INFORMATION FOR VizaWrite ***
90 *** $FF INDICATES END OF TRANSFER ***
91
0463: 00 1B 92 FROMLO HEX 00,1B
0465: 80 8A 93 FROMHI HEX 80,8A
0467: 00 FE 94 TOLO HEX 00,FE
0469: 0B 89 95 TOHI HEX 0B,89
046B: FD FD 96 ENDLO HEX FD,FD
046D: FE FE 97 ENDHI HEX FE,FE
046F: 09 0A FF 98 PGNO HEX 09,0A,FF
99

```

```

2
3 *****
4 * VizaWrite LOADER FOR PAGED EPROM *
5 *****
6 *
7 ORG $8700
8 *
9 FROM = $60
10 TO = $62
11 END = $64
12 PGSW = $D700
13
8700: A2 00 14 LDX #$00
8702: BD 0E 87 15 NXTRAN LDA BEGIN+3,X
8705: 9D 00 09 16 STA $0900,X
8708: E8 17 INX
8709: D0 F7 18 BNE NXTRAN
870B: 4C 00 09 19 BEGIN JMP RAMLD
20
21 ORG $0900
22
0900: A2 00 23 RAMLD LDX #$00 ;clear information indexer
0902: BD 63 09 24 LDA PGNO,X ;get first rom page no.
0905: 8D 00 D7 25 NXPG STA PGSW ;select page at $d700
0908: 8D 00 D7 26 STA PGSW
090B: BD 5D 09 27 LDA FROMLO,X ;from rom low byte
090E: 85 60 28 STA FROM
0910: BD 5E 09 29 LDA FROMHI,X ;from rom high byte
0913: 85 61 30 STA FROM+1
0915: BD 5F 09 31 LDA TOLO,X ;to ram low byte
0918: 85 62 32 STA TO
091A: BD 60 09 33 LDA TOHI,X ;to ram high byte
091D: 85 63 34 STA TO+1
091F: BD 61 09 35 LDA ENDLO,X ;end adrs rom low byte
0922: 85 64 36 STA END
0924: BD 62 09 37 LDA ENDHI,X ;end adrs rom high byte
0927: 85 65 38 STA END+1
0929: A0 00 39 LDY #$00
092B: B1 60 40 LOOP1 LDA (FROM),Y ;get byte from rom
092D: 91 62 41 STA (TO),Y ;store byte in ram
092F: A5 61 42 LDA FROM+1 ;ck for end lo byte
0931: C5 65 43 CMP END+1
0933: D0 06 44 BNE SKIP1 ;not end, then cont.

```

```

0935: A5 60      45      LDA    FROM      ;ck for end hi byte
0937: C5 64      46      CMP     END
0939: F0 0E      47      BEQ     SKIP3      ;if end, then ck for nx page
093B: E6 60      48      SKIP1  INC     FROM      ;inc from lo
093D: D0 02      49      BNE     SKIP2
093F: E6 61      50      INC     FROM+1    ;inc from hi
0941: E6 62      51      SKIP2  INC     TO      ;inc to lo
0943: D0 E6      52      BNE     LOOP1
0945: E6 63      53      INC     TO+1    ;inc to hi
0947: D0 E2      54      BNE     LOOP1    ;get and store nx byte
0949: E8          55      SKIP3  INX
094A: BD 63 09    56      LDA     PGNO,X    ;get next page
094D: C9 FF      57      CMP     #$FF      ;ck for end of transfer
094F: D0 B4      58      BNE     NXPG      ;if not $ff, jmp to nx page
0951: A9 03      59      LDA     #$03      ;switch to page for int-cart
0953: 8D 00 D7    60      STA     PGSW
0956: 8D 00 D7    61      STA     PGSW
0959: 78          62      SEI
095A: 4C 1A 0B    63      JMP     $0B1A    ;jmp to start pgm
                                64
                                65      *** TRANSFER INFORMATION FOR VizaWrite ***
                                66      *** $FF INDICATES END OF TRANSFER ***
                                67
095D: 00          68      FROMLO  HEX     00
095E: 80          69      FROMHI  HEX     80
095F: 00          70      TOLO    HEX     00
0960: 04          71      TOHI    HEX     04
0961: 9A          72      ENDLO   HEX     9A
0962: 84          73      ENDHI   HEX     84
0963: 0A FF      74      PGNO    HEX     0A, FF
                                75

```

The next loader is for a program that runs in C64 mode, Monitor 64. In the first part of the loader, the code for the monitor is transferred to RAM at \$C000. Then the auto-run routine is transferred to \$8000. When both programs are in place, a small routine will be placed at \$10F0 and then the loader will jump to the routine. All system ROMs are set, page 0 of the Internal ROM is switched in and then the program jumps to Kernal routine GO64.

When the C128 goes to C64 mode it will start at the reset vector (\$FCE2). The first thing it will do is Set the Interrupt, clear the Stack, Clear the Decimal and then check for "CBM80" at \$8004 - \$8008. If it finds CBM80, then it will jump to the vector at \$8000. At \$8009 the code will setup C64 mode and stop short of printing the opening screen and printing the READY prompt. At the very end it will jump to the start of Monitor 64 (\$C000).

```

1
3
4     MON64     =     $C000     ;monitor entry
5     PRINT     =     $C00C     ;print a char to scrn
6     PGSW      =     $D700     ;switch page in Int ROM
7     CONFIG    =     $FF00     ;configuration register
8     G064      =     $FF4D     ;128 jmp to 64 mode
9     INIT      =     $E3BF     ;initialize basic
10    BASVEC     =     $E453     ;copies bas vectors
11    RESTOR     =     $FD15     ;restores I/O vectors
12    RAMTAS     =     $FD50     ;RAM test
13    IOINIT     =     $FDA3     ;initialize CIA
14    CINT       =     $FF5B     ;initialize scrn editor
15
16            ORG     $8000
17
8000: A9 93      18            LDA     #$93
8002: 20 0C C0   19            JSR     PRINT     ;clr the screen
8005: A2 05      20            LDX     #$05
8007: BD 57 80   21    LOOP     LDA     TRAN,X     ;transfer to & from
800A: 95 60      22            STA     $60,X     ; info to variables
800C: CA        23            DEX
800D: 10 F8      24            BPL     LOOP
800F: 78        25            SEI
8010: A0 00      26            LDY     #$00
8012: B1 60      27    LOOP1     LDA     ($60),Y
8014: 91 62      28            STA     ($62),Y
8016: A5 61      29            LDA     $61
8018: C5 65      30            CMP     $65
801A: D0 06      31            BNE     SKIP1
801C: A5 60      32            LDA     $60
801E: C5 64      33            CMP     $64
8020: F0 0E      34            BEQ     SKIP3
8022: E6 60      35    SKIP1     INC     $60
8024: D0 02      36            BNE     SKIP2
8026: E6 61      37            INC     $61
8028: E6 62      38    SKIP2     INC     $62
802A: D0 E6      39            BNE     LOOP1
802C: E6 63      40            INC     $63
802E: D0 E2      41            BNE     LOOP1
8030: A2 27      42    SKIP3     LDX     #AUST+END
8032: BD 5D 80   43    LOOP2     LDA     TRAN+6,X
8035: 9D 00 80   44            STA     $8000,X
8038: CA        45            DEX
8039: 10 F7      46            BPL     LOOP2
803B: A2 0D      47            LDX     #$0D
803D: BD 49 80   48    LOOP3     LDA     RAMST,X
8040: 9D F0 10   49            STA     $10F0,X
8043: CA        50            DEX
8044: 10 F7      51            BPL     LOOP3
8046: 4C F0 10   52            JMP     $10F0
53

```

```

8049: A9 00      54  RAMST   LDA    #0
804B: 8D 00 FF  55          STA   CONFIG
804E: 8D 00 D7  56          STA   PGSW
8051: 8D 00 D7  57          STA   PGSW
8054: 4C 4D FF  58          JMP   G064
      59
8057: 3E F0      60  TRAN    DA    $F03E      ;from in ROM
8059: 00 C0      61          DA    MON64      ;to in RAM
805B: FD FE      62          DA    $FEFD      ;end in ROM
      63
      64  *****
      65  * this is the auto-start routine *
      66  * that will start C64 monitor   *
      67  *****
      68
      69          ORG    $8000
      70
8000: 09 80      71  AUST    DA    $8009
8002: 09 80      72          DA    $8009
      73
8004: C3 C2 CD  74          TXT    "cbm"          ;bit 7 is set on cbm
8007: 38 30      75          TXT    '80'
      76
8009: A2 05      77          LDX    #$05
800B: 8E 08 80  78          STX    $8008      ;disable auto-start
800E: 8E 16 D0  79          STX    $D016
8011: 20 A3 FD  80          JSR    IOINIT
8014: 20 50 FD  81          JSR    RAMTAS
8017: 20 15 FD  82          JSR    RESTOR
801A: 20 5B FF  83          JSR    CINT
801D: 58        84          CLI
801E: 20 53 E4  85          JSR    BASVEC
8021: 20 BF E3  86          JSR    INIT
8024: 4C 00 C0  87          JMP    MON64
      88  END

```

The next loader program is used for all the Maverick programs. The C64 auto-start routine for the C64 is at the end of the program code.

```

2
3  *****
4  *   LOADER FOR MAVERICK PGMS   *
5  *****
6  *
7          ORG    $9100
8  *
9  FROM    =    $60
10 TO      =    $62

```

		11	END	=	\$64	
		12	PRTSCN	=	\$C00C	
		13	PGSW	=	\$D700	
		14	CONFIG	=	\$FF00	
		15	G064	=	\$FF4D	
		16				
9100:	A9 71	17		LDA	#RAMLD+LAST	
9102:	85 FA	18		STA	\$FA	
9104:	A2 00	19		LDX	#\$00	
9106:	BD 14 91	20	NXTRAN	LDA	BEGIN+3,X	
9109:	9D 00 0C	21		STA	RAMLD,X	
910C:	E8	22		INX		
910D:	E4 FA	23		CPX	\$FA	
910F:	D0 F5	24		BNE	NXTRAN	
9111:	4C 00 0C	25	BEGIN	JMP	RAMLD	
		26				
		27		ORG	\$0C00	
		28				
0C00:	A9 93	29	RAMLD	LDA	#\$93	;clear screen
0C02:	20 0C C0	30		JSR	PRTSCN	
0C05:	78	31		SEI		
0C06:	A2 00	32		LDX	#0	;clear information indexer
0C08:	BD 6F 0C	33		LDA	PGNO,X	;get first rom page no.
0C0B:	8D 00 D7	34	NXPG	STA	PGSW	;select page at \$d700
0C0E:	8D 00 D7	35		STA	PGSW	
0C11:	BD 69 0C	36		LDA	FROMLO,X	;from rom low byte
0C14:	85 60	37		STA	FROM	
0C16:	BD 6A 0C	38		LDA	FROMHI,X	;from rom high byte
0C19:	85 61	39		STA	FROM+1	
0C1B:	BD 6B 0C	40		LDA	TOLO,X	;to ram low byte
0C1E:	85 62	41		STA	T0	
0C20:	BD 6C 0C	42		LDA	TOHI,X	;to ram high byte
0C23:	85 63	43		STA	T0+1	
0C25:	BD 6D 0C	44		LDA	ENDLO,X	;end adrs rom low byte
0C28:	85 64	45		STA	END	
0C2A:	BD 6E 0C	46		LDA	ENDHI,X	;end adrs rom high byte
0C2D:	85 65	47		STA	END+1	
0C2F:	8D 02 FF	48		STA	\$FF02	
0C32:	A0 00	49		LDY	#\$00	
0C34:	B1 60	50	H0C24	LDA	(FROM),Y	;get byte from rom
0C36:	91 62	51		STA	(T0),Y	;store byte in ram
0C38:	A5 61	52		LDA	FROM+1	;ck for end lo byte
0C3A:	C5 65	53		CMP	END+1	
0C3C:	D0 06	54		BNE	H0C3B	;not end, then cont.
0C3E:	A5 60	55		LDA	FROM	;ck for end hi byte
0C40:	C5 64	56		CMP	END	
0C42:	F0 0E	57		BEQ	H0C49	;if end, then ck for nx page
0C44:	E6 60	58	H0C3B	INC	FROM	;inc from lo
0C46:	D0 02	59		BNE	H0C41	
0C48:	E6 61	60		INC	FROM+1	;inc from hi
0C4A:	E6 62	61	H0C41	INC	T0	;inc to lo
0C4C:	D0 E6	62		BNE	H0C24	
0C4E:	E6 63	63		INC	T0+1	;inc to hi
0C50:	D0 E2	64		BNE	H0C24	;get and store nx byte

```

0C52: E8      65   H0C49   INX
0C53: BD 6F 0C 66           LDA   PGNO,X      ;get next page
0C56: C9 FF      67           CMP   #$FF      ;ck for end of transfer
0C58: D0 B1      68           BNE   NXPG      ;if not $ff, jmp to nx page
0C5A: A9 00      69           LDA   #0        ;switch to page 0
0C5C: 8D 00 FF 70           STA   CONFIG
0C5F: 8D 00 D7 71           STA   PGSW
0C62: 8D 00 D7 72           STA   PGSW
0C65: 58        73           CLI
0C66: 4C 4D FF 74           JMP   G064
                                75
                                76   *** TRANSFER INFORMATION FOR MAVERICK ***
                                77   *** $FF INDICATES END OF TRANSFER ***
                                78
0C69: EC        79   FROMLO   HEX   EC
0C6A: 9C        80   FROMHI   HEX   9C
0C6B: 00        81   TOLO     HEX   00
0C6C: 4F        82   TOHI     HEX   4F
0C6D: F4        83   ENDLO    HEX   F4
0C6E: CD        84   ENDHI    HEX   CD
0C6F: 05 FF     85   PGNO     HEX   05,FF
                                86   LAST
                                87

```

The Maverick program is transferred to memory at \$4F00 (see line 81 and 82 above), the end of the program is at \$7EFF. The following program will setup C64 mode and transfer the program to its proper location and start it.

The end of all the Maverick programs in Internal ROM is at \$8008. The following at \$8000 will auto-start the C64:

```
$8000 00 7F 00 7F C3 C2 CD 38 30
```

```

$7F00
$7F00
CBM80

```

When the beginning of C64 code sees the CBM80 it will jump to \$7F00 and run the following code. It will move the program from \$4F00-\$7EFF to \$0800-\$37FF and start the program at \$0800.

```

1
2   *****
3   * maverick loader and start *
4   *****
5
6   MAVST   =   $0800      ;maverick entry
7   INIT    =   $E3BF      ;initialize basic

```

```

8      BASVEC  =      $E453      ;copies basic vectors
9      RESTOR  =      $FD15      ;restores I/O vectors
10     RAMTAS  =      $FD50      ;RAM test
11     IOINIT  =      $FDA3      ;initialize CIA
12     CINT    =      $FF5B      ;initialize scrn editor
13
14             ORG      $7F00
15
16     *****
17     * setup C64 mode *
18     *****
19
7F00: A2 05    20             LDX      #$05
7F02: 8E 08 80 21             STX      $8008      ;disable auto-start
7F05: 8E 16 D0 22             STX      $D016
7F08: 20 A3 FD 23             JSR      IOINIT
7F0B: 20 50 FD 24             JSR      RAMTAS
7F0E: 20 15 FD 25             JSR      RESTOR
7F11: 20 5B FF 26             JSR      CINT
7F14: 58       27             CLI
7F15: 20 53 E4 28             JSR      BASVEC
7F18: 20 BF E3 29             JSR      INIT
30
31     *****
32     * transfer maverick pgm to *
33     * proper location & start *
34     * from $4F00-$7EFF to $0800 *
35     *****
36
7F1B: A9 00    37             LDA      #$00
7F1D: 85 FB    38             STA      $FB
7F1F: 85 FD    39             STA      $FD
7F21: AA       40             TAX
7F22: A9 54    41             LDA      #$4F
7F24: 85 FC    42             STA      $FC
7F26: A9 08    43             LDA      #$08
7F28: 85 FE    44             STA      $FE
7F2A: A0 00    45             LDY      #$00
7F2C: B1 FB    46     LOOP    LDA      ($FB),Y
7F2E: 91 FD    47             STA      ($FD),Y
7F30: 8A       48             TXA      ;clean up as you go
7F31: 91 FB    49             STA      ($FB),Y
7F33: C8       50             INY
7F34: D0 F6    51             BNE      LOOP
7F36: E6 FC    52             INC      $FC
7F38: E6 FE    53             INC      $FE
7F3A: A5 FC    54             LDA      $FC
7F3C: C9 7F    55             CMP      #$7F
7F3E: D0 EC    56             BNE      LOOP
7F40: A9 08    57             LDA      #$08      ;set drv 8 as default
7F42: 85 BA    58             STA      $BA
7F44: 4C 00 08 59             JMP      MAVST      ;jmp to pgm start
60

```

THE C64 CARTRIDGE

The C64 Cartridge will have the same memory space available as the C128 Internal ROM Adapter, but you can only access 16K bytes at a time (\$8000 to \$BFFF). Then you will have to switch to another page for more data or you can start your program.

The cartridge can be used in several different ways, 8K ROM cartridge programs, 16K ROM cartridge programs, any size programs that run in RAM. Also you can switch the cartridge off and go to Basic, all under software control.

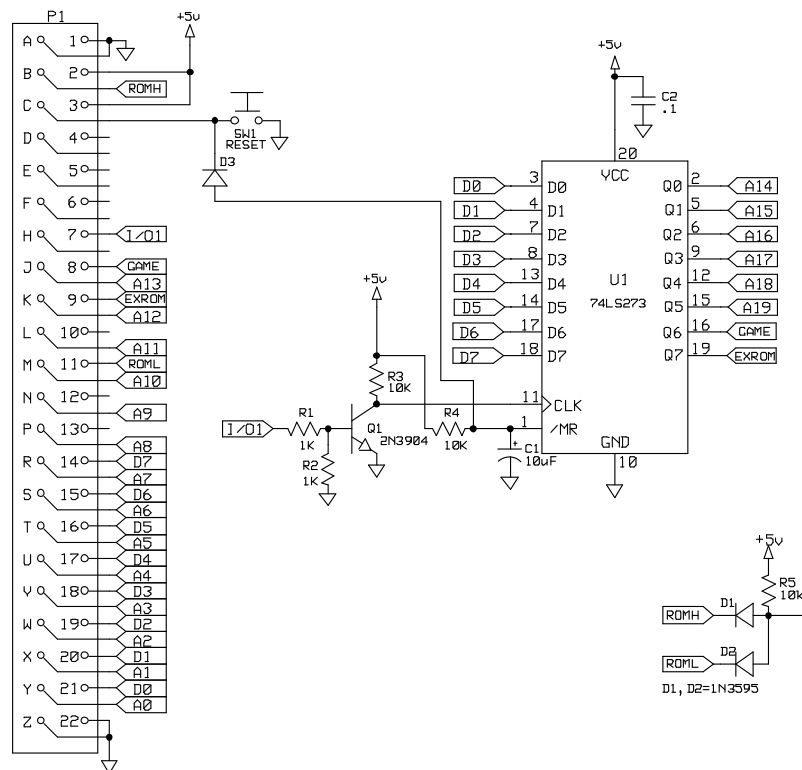


Figure 3

In Figure 3 above you'll notice a 74LS273 (Octal D Flip-Flop) is used to control the extra address lines of the high capacity EPROMs. The 74LS273 is a high-speed 8-Bit Register. The register consists of eight D-Type Flip-Flops with a Common Clock and an asynchronous active LOW Master Reset. It works the same as the C128 Internal ROM Adapter, except it has two extra control lines that are used to control the GAME and EXROM lines on the C64. Bit 6 will control GAME and Bit 7 will control EXROM.

Here is how it looks with a 27010 (2 MEG EPROM):

<u>Pg HEX</u>	<u>A16</u>	<u>A15</u>	<u>A14</u>	
Page 00	0	0	0	16K x 8
Page 01	0	0	1	16K x 8
Page 02	0	1	0	16K x 8
Page 03	0	1	1	16K x 8
Page 04	1	0	0	16K x 8
Page 05	1	0	1	16K x 8
Page 06	1	1	0	16K x 8
Page 07	1	1	1	16K x 8

Total 128K x 8

Here is how it looks with a 27020 (4 MEG EPROM):

<u>Pg HEX</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	<u>A14</u>	
Page 00	0	0	0	0	16K x 8
Page 01	0	0	0	1	16K x 8
Page 02	0	0	1	0	16K x 8
Page 03	0	0	1	1	16K x 8
Page 04	0	1	0	0	16K x 8
Page 05	0	1	0	1	16K x 8
Page 06	0	1	1	0	16K x 8
Page 07	0	1	1	1	16K x 8
Page 08	1	0	0	0	16K x 8
Page 09	1	0	0	1	16K x 8
Page 0A	1	0	1	0	16K x 8
Page 0B	1	0	1	1	16K x 8
Page 0C	1	1	0	0	16K x 8
Page 0D	1	1	0	1	16K x 8
Page 0E	1	1	1	0	16K x 8
Page 0F	1	1	1	1	16K x 8

Total 256K x 8

Here is how it looks with a 27040 (8 MEG EPROM):

<u>Pg HEX</u>	<u>A18</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	<u>A14</u>	
Page 00	0	0	0	0	0	16K x 8
Page 01	0	0	0	0	1	16K x 8
Page 02	0	0	0	1	0	16K x 8
Page 03	0	0	0	1	1	16K x 8
Page 04	0	0	1	0	0	16K x 8
Page 05	0	0	1	0	1	16K x 8
Page 06	0	0	1	1	0	16K x 8
Page 07	0	0	1	1	1	16K x 8
Page 08	0	1	0	0	0	16K x 8
Page 09	0	1	0	0	1	16K x 8
Page 0A	0	1	0	1	0	16K x 8
Page 0B	0	1	0	1	1	16K x 8
Page 0C	0	1	1	0	0	16K x 8
Page 0D	0	1	1	0	1	16K x 8
Page 0E	0	1	1	1	0	16K x 8
Page 0F	0	1	1	1	1	16K x 8
Page 10	1	0	0	0	0	16K x 8
Page 11	1	0	0	0	1	16K x 8
Page 12	1	0	0	1	0	16K x 8
Page 13	1	0	0	1	1	16K x 8
Page 14	1	0	1	0	0	16K x 8
Page 15	1	0	1	0	1	16K x 8
Page 16	1	0	1	1	0	16K x 8
Page 17	1	0	1	1	1	16K x 8
Page 18	1	1	0	0	0	16K x 8
Page 19	1	1	0	0	1	16K x 8
Page 1A	1	1	0	1	0	16K x 8
Page 1B	1	1	0	1	1	16K x 8
Page 1C	1	1	1	0	0	16K x 8
Page 1D	1	1	1	0	1	16K x 8
Page 1E	1	1	1	1	0	16K x 8
Page 1F	1	1	1	1	1	16K x 8
Total						512K x 8

Here is how it looks with a 27080 (8 MEG EPROM):

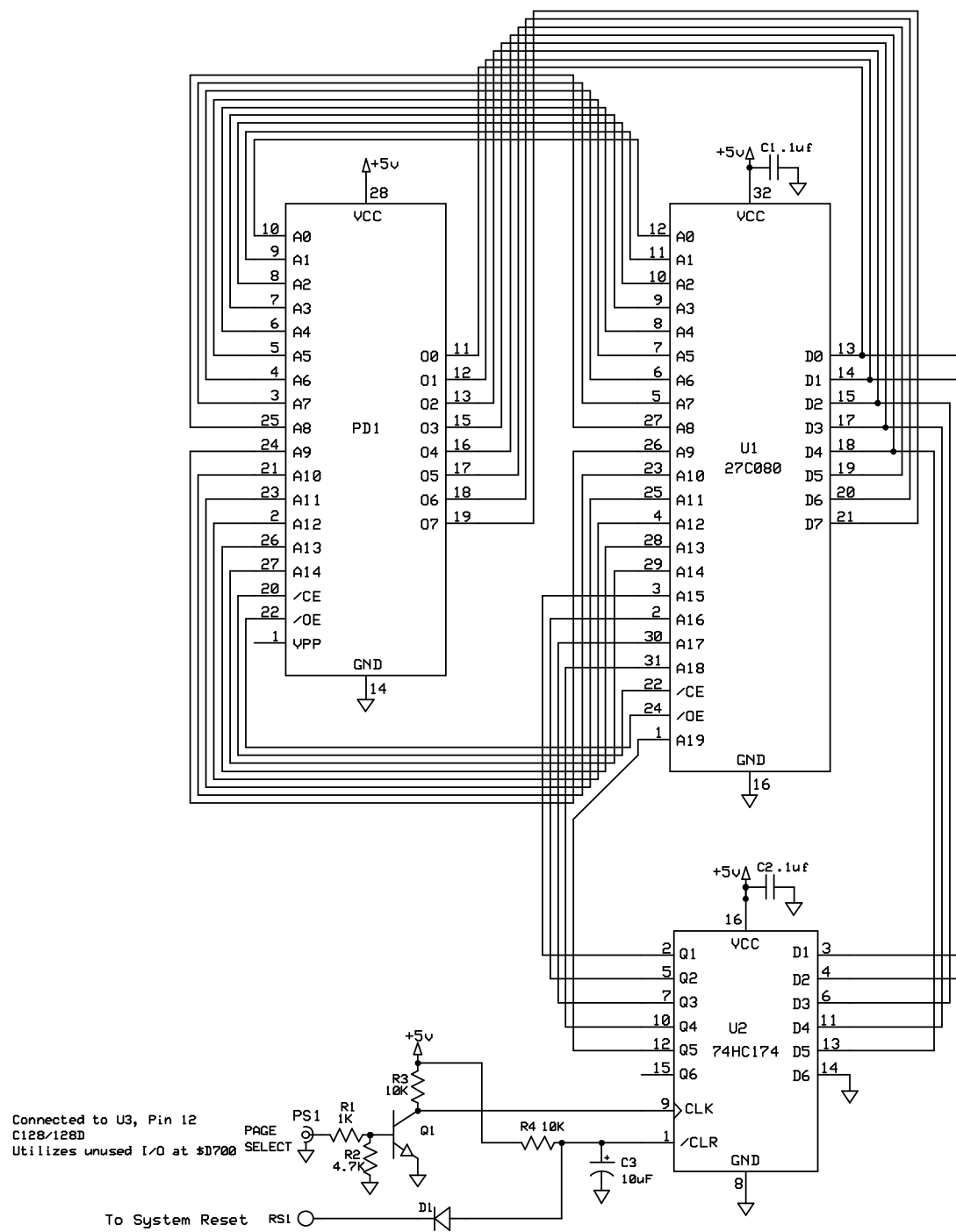
<u>Pg HEX</u>	<u>A19</u>	<u>A18</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	<u>A14</u>	
Page 00	0	0	0	0	0	0	16K x 8
Page 01	0	0	0	0	0	1	16K x 8
Page 02	0	0	0	0	1	0	16K x 8
Page 03	0	0	0	0	1	1	16K x 8
Page 04	0	0	0	1	0	0	16K x 8
Page 05	0	0	0	1	0	1	16K x 8
Page 06	0	0	0	1	1	0	16K x 8
Page 07	0	0	0	1	1	1	16K x 8
Page 08	0	0	1	0	0	0	16K x 8
Page 09	0	0	1	0	0	1	16K x 8
Page 0A	0	0	1	0	1	0	16K x 8
Page 0B	0	0	1	0	1	1	16K x 8
Page 0C	0	0	1	1	0	0	16K x 8
Page 0D	0	0	1	1	0	1	16K x 8
Page 0E	0	0	1	1	1	0	16K x 8
Page 0F	0	0	1	1	1	1	16K x 8
Page 10	0	1	0	0	0	0	16K x 8
Page 11	0	1	0	0	0	1	16K x 8
Page 12	0	1	0	0	1	0	16K x 8
Page 13	0	1	0	0	1	1	16K x 8
Page 14	0	1	0	1	0	0	16K x 8
Page 15	0	1	0	1	0	1	16K x 8
Page 16	0	1	0	1	1	0	16K x 8
Page 17	0	1	0	1	1	1	16K x 8
Page 18	0	1	1	0	0	0	16K x 8
Page 19	0	1	1	0	0	1	16K x 8
Page 1A	0	1	1	0	1	0	16K x 8
Page 1B	0	1	1	0	1	1	16K x 8
Page 1C	0	1	1	1	0	0	16K x 8
Page 1D	0	1	1	1	0	1	16K x 8
Page 1E	0	1	1	1	1	0	16K x 8
Page 1F	0	1	1	1	1	1	16K x 8
Page 20	1	0	0	0	0	0	16K x 8
Page 21	1	0	0	0	0	1	16K x 8
Page 22	1	0	0	0	1	0	16K x 8
Page 23	1	0	0	0	1	1	16K x 8
Page 24	1	0	0	1	0	0	16K x 8
Page 25	1	0	0	1	0	1	16K x 8
Page 26	1	0	0	1	1	0	16K x 8
Page 27	1	0	0	1	1	1	16K x 8

<u>Pg HEX</u>	<u>A19</u>	<u>A18</u>	<u>A17</u>	<u>A16</u>	<u>A15</u>	<u>A14</u>	
Page 28	1	0	1	0	0	0	16K x 8
Page 29	1	0	1	0	0	1	16K x 8
Page 2A	1	0	1	0	1	0	16K x 8
Page 2B	1	0	1	0	1	1	16K x 8
Page 2C	1	0	1	1	0	0	16K x 8
Page 2D	1	0	1	1	0	1	16K x 8
Page 2E	1	0	1	1	1	0	16K x 8
Page 2F	1	0	1	1	1	1	16K x 8
Page 30	1	1	0	0	0	0	16K x 8
Page 31	1	1	0	0	0	1	16K x 8
Page 32	1	1	0	0	1	0	16K x 8
Page 33	1	1	0	0	1	1	16K x 8
Page 34	1	1	0	1	0	0	16K x 8
Page 35	1	1	0	1	0	1	16K x 8
Page 36	1	1	0	1	1	0	16K x 8
Page 37	1	1	0	1	1	1	16K x 8
Page 38	1	1	1	0	0	0	16K x 8
Page 39	1	1	1	0	0	1	16K x 8
Page 3A	1	1	1	0	1	0	16K x 8
Page 3B	1	1	1	0	1	1	16K x 8
Page 3C	1	1	1	1	0	0	16K x 8
Page 3D	1	1	1	1	0	1	16K x 8
Page 3E	1	1	1	1	1	0	16K x 8
Page 3F	1	1	1	1	1	1	16K x 8

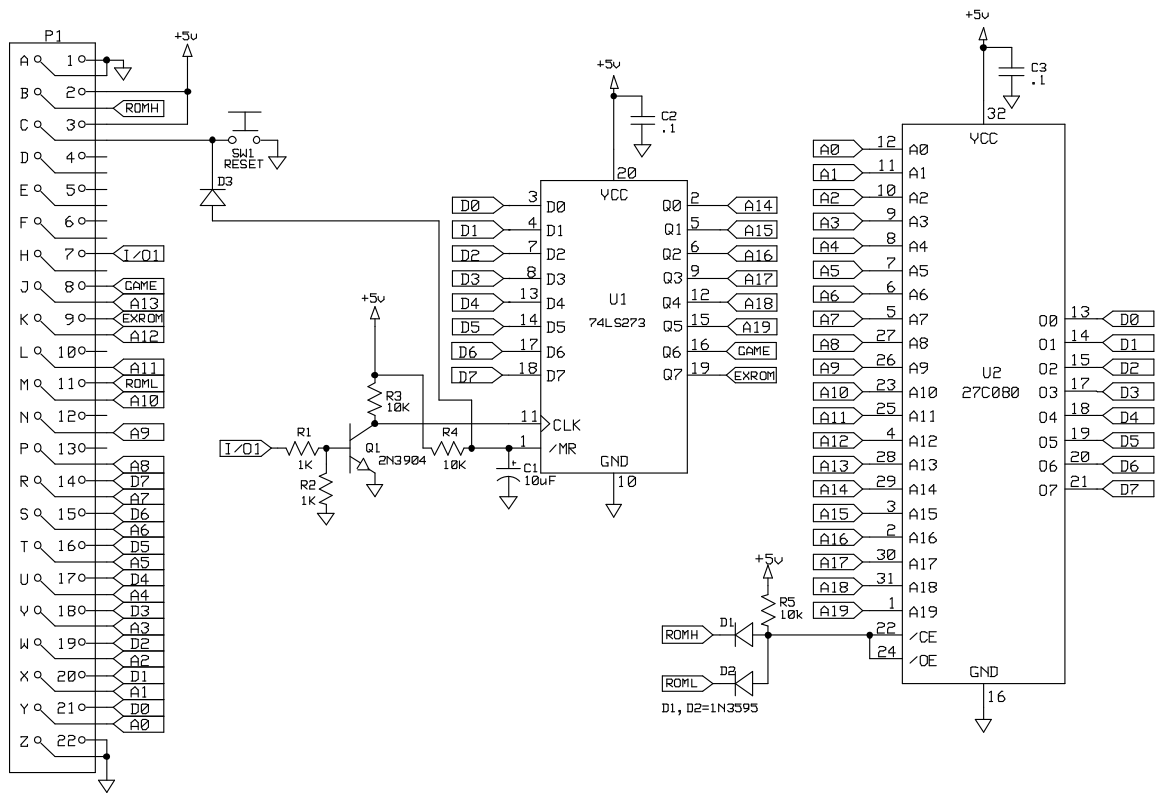
Total 1024K x 8

<u>Page</u>	<u>Bit 6</u>	<u>Bit 7</u>	
00	0	0	GAME & EXROM pulled low (access pg #0 to pg #63)
40	0	1	GAME is high & EXROM is low
80	1	0	GAME is low & EXROM is high
C0	1	1	GAME is high & EXROM is high (turn off cartridge)

When Bits 6 and 7 are high, then the cartridge is turned off. Selecting page number \$C0 will turn off the cartridge. In the Menu the STOP key will turn off the cartridge and send you to BASIC. Also when you load and run a program that runs in RAM, you will have to turn the cartridge off before the program is started.



MEGABIT 128 INTERNAL ADAPTER



MEGABIT C64 CARTRIDGE