JASON-RANHEIM COMPANY

1805 Industrial Drive
(916) 823-3284

Auburn, California 95603
(916) 823-3285

PTM-2 CARTRIDGE FOR THE COMMODORE 128 COMPUTER

The PTM-2 cartridge provides a means whereby 128 mode programs for the C128 computer can be put into cartridge form. It is intended for use with 2764, 27128, 27256 or 27512 EPROMs.

Two EPROMs can be accommodated. These are referred to on the cartridge PC board as the "L" and "H" EPROMs. The "L" (low) EPROM is accessed by the computer in external ROM bank 8 at addresses from $8000 to $BFFF. The "H" (high) EPROM is accessed in bank 8 at addresses from $C000 through $FFFF.

The C128 looks at both $8000 and $C000 for an auto-start cartridge. Either location or both can be used. A cartridge program located at $8000 will be started first and therefore has priority over one located at $C000. These areas are not completely equivalent, however because of the "hole" at $FF00-$FF04 where MMU registers are always found. In addition, I/O normally overlays addresses from $D000-$DFFF and this complicates access to the "H" EPROM. So if the user doesn't require both, he should use the "L" EPROM for his applications.

Bank Switching on the PTM-2

The PTM-2 offers a limited bank switching capability making it possible to use 27256 (32k bytes) or even 27512 (64k byte) EPROMs. The PTM-2 takes advantage of the C128's ability to use the expansion port GAME and EXROM lines as outputs.

These lines are controlled by the MMU. They appear as bits 4 and 5 of the MODE CONFIGURATION REGISTER at $D505. Normally, these bits are high (ones), but they can be set low by software if desired. Bit 5 controls the EXROM line. On the PTM-2, this line is connected to Pin 27 of the EPROM. This is address bit A_{14} on the 27256 and 27512.

If the modification for the 27512 is made, described in FIG. 1 below, then the GAME line will be connected to PIN 1 of the EPROM. This is address bit A_{15} on the 27512. Clearing bit 4 of the MCR will set this pin low.

It becomes apparent then that 16k byte blocks of these larger EPROM's can be "banked" in and out of the EXTERNAL computer bank by clearing or setting bits 4 & 5 of the MCR.

This bank switching ability is "free" in the hardware sense. But from the software standpoint it can be complicated.

The auto-boot routine called "PTM2" establishes the convention that these larger EPROMs will be accessed in 16k blocks <u>from the top down</u>. For example: on a 27256 (32k), the auto-boot routine itself will be located at the beginning of the upper half of the EPROM starting at EPROM address 16384.

EPROM addresses from 16640 through 32767 will contain the first 16128 bytes of the program. The remainder of the program will be found in the bottom half of the EPROM beginning at address 0.

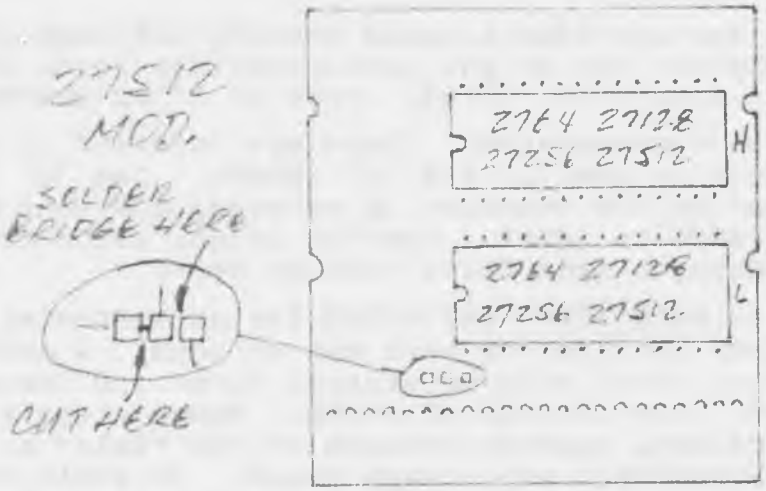

FIGURE 1

Making an Auto-Start Cartridge

We will first illustrate the procedure for a 27128 EPROM where our program is less than 16128 bytes in length.

STEP 1- Plug in your PROMENADE C1. Turn on your computer and load the 128 mode program you wish to put on cartridge:

LOAD"OBJECTPROGRAM",8 <CR>

STEP 2- Get into 64 mode by:

GO64 <CR>
(boldly answering 'Y' to "Are You Sure?")

STEP 3- LOAD"PROMOS 1.1 C64",8 <CR>
POKE56,208 <CR>
RUN <CR>

PROMOS is now installed just below $D000.

STEP 4- LOAD"PTM2",8,1 <CR>

The PTM2 auto-boot is loaded into the computer at $C000. Don't forget the ",1" secondary address in entering the LOAD command.

STEP 5- POKE49190,PEEK(4624) <CR>
POKE49191,PEEK(4625) <CR>

Store the location of the end of your program in the auto-boot routine.

STEP 6- ME=PEEK(4624)+256*PEEK(4625)-1 <CR>

Calculates the address of the last byte of your object program and stores the result as the BASIC variable "ME".

STEP 7- π49152,49407,0,5,7 <CR>

Programs the auto-boot routine into the first 256 bytes of the 27128.

STEP 8- π7169,ME,256,5,7 <CR>

Programs the object program into the 27128.

STEP 9- Plug your EPROM into the "L" socket of the PTM-2, orienting the notch as shown in FIGURE 1.

With the power off, plug your PTM-2 into the expansion port. Turn on your computer and your object program will run automatically.

Next, we illustrate using a 12.5 volt 27256 to contain a longer program.

Do steps 1 through 6 exactly as above.

STEP 7- π49152,49407,16384,230,7 <CR>

Program the auto-boot into the beginning of the upper half of the 27256.

STEP 8- π7169,23296,16640,230,7 <CR>

Fill the remainder of the upper half with 16128 basic bytes.

STEP 9- π23297,ME,0,230,7 <CR>

Finish the job by programming the remainder of the program into the bottom half of the EPROM. Now plug the 27256 into the "L" socket of the PTM-2 and verify that your program runs correctly.

```
                        1     *       PTM-2 DOWNLOADER        *
                        2     ********************************
                        3               ORG   $C000
                        4     ********************************
 C000: 78               5               SEI                   ;disable interupts
C001: B8                6               CLV
C002: 50 06             7               BVC   START
C004: FF                8               HEX   FF              ;two bytes set aside for
C005: FF                9               HEX   FF              ;user ID
C006: 02                10              HEX   02              ;cart ID byte
C007: 43 42 4D          11              HEX   43424D          ;CBM
C00A: 20 AE 02          12    START     JSR   $02AE           ;find out where we are--
C00D: BA                13              TSX                   ;$8000 or $C000
C00E: BD 00 01          14              LDA   $0100,X
C011: 85 A4             15              STA   $A4
C013: A9 00             16              LDA   #$00
C015: 85 A3             17              STA   $A3
C017: A0 26             18              LDY   #$26
C019: B1 A3             19    PUTSTK    LDA   ($A3),Y         ;get downloader from ROM
C01B: 99 12 01          20              STA   $0112,Y         ;and move to stack
C01E: C8                21              INY
C01F: D0 F8             22              BNE   PUTSTK
C021: 84 05             23              STY   $05
C023: 4C 3A 01          24              JMP   $013A           ;jump to next instruction
                        25                                    ;now on stack
C026: FF                26              HEX   FF              ;low byte end of program
C027: FF                27              HEX   FF              ;high byte
C028: A5 A4             28              LDA   $A4
C02A: AA                29              TAX
C02B: 18                30    SETPTS    CLC
C02C: 8D 49 01          31              STA   $0149           ;set masked LDX
C02F: 69 40             32              ADC   #$40
C031: 8D 68 01          33              STA   $0168           ;set end of ROM
C034: E8                34              INX
C035: 2C A2 80          35    MSKLD     BIT   $80A2           ;masked LDX #$80 or #$C0
C038: A0 00             36              LDY   #$00
C03A: 86 A4             37              STX   $A4
C03C: A2 2B             38              LDX   #$2B            ;set config to ext. ROM from
C03E: 8E 00 FF          39              STX   $FF00           ;$8000 up
C041: B1 A3             40    MVDATA    LDA   ($A3),Y         ;get cart data
C043: 91 2D             41              STA   ($2D),Y         ;store data in RAM bank 0
C045: C8                42              INY
C046: D0 F9             43              BNE   MVDATA
C048: A5 2E             44              LDA   $2E
C04A: CD 39 01          45              CMP   $0139           ;are we finished
C04D: B0 18             46              BCS   END             ;yes--go start the program
C04F: E6 2E             47              INC   $2E
C051: E6 A4             48              INC   $A4
C053: A5 A4             49              LDA   $A4
C055: C9 C0             50              CMP   #$C0            ;out ROM in current bank?
C057: D0 E8             51              BNE   MVDATA          ;no--go get some more
                        52    ********************************
C059: A9 3A             53              LDA   #$3A
C05B: 8D 00 FF          54              STA   $FF00           ;switch in IO
C05E: AD 05 D5          55              LDA   $D505
C061: 85 A6             56              STA   $A6
C063: B8                57    BANKS     CLV                   ;to check and switch banks
C064: A5 A6             58              LDA   $A6             ;get current bank
C066: AA                59              TAX                   ;save it to X register
```

```
C067: 29 20     60              AND   #$20       ;these two instr. to isolate
C069: 49 20     61              EOR   #$20       ;and check bit five
C06B: F0 0F     62              BEQ   CLR5       ;if bit 5 set go clear it
C06D: 8A        63              TXA
C06E: 29 10     64              AND   #$10       ;same as above,
C070: 49 10     65              EOR   #$10       ;for bit 4
C072: D0 10     66              BNE   NEWROM     ;if 4 & 5 clear--get next socket
C074: 8A        67              TXA
C075: 49 30     68              EOR   #$30       ;toggle bits 4 and 5
C077: 8D 05 D5  69              STA   $D505      ;and set up exrom and game
C07A: 50 14     70              BVC   CHKDTA     ;go to check--same data?
C07C: 8A        71    CLR5      TXA
C07D: 29 DF     72              AND   #$DF       ;clear bit 5
C07F: 8D 05 D5  73              STA   $D505
C082: 50 0C     74              BVC   CHKDTA
C084: 8A        75    NEWROM    TXA              ;switch to $C000 ROM
C085: 09 30     76              ORA   #$30       ;set bits 4 and 5
C087: 8D 05 D5  77              STA   $D505
C08A: A9 C0     78              LDA   #$C0       ;set up A and X registers
C08C: AA        79              TAX              ;for next ROM socket
C08D: CA        80              DEX
C08E: D0 9B     81              BNE   SETPTS     ;and go get more data
                82    ********************************
C090: A0 03     83    CHKDTA    LDY   #$03       ;to check 3 bytes
C092: 85 A5     84              STA   $A5
C094: A6 A6     85    CHK       LDX   $A6
C096: 8E 05 D5  86              STX   $D505      ;switch in former bank
C099: B9 00 80  87              LDA   $8000,Y    ;get byte
C09C: A6 A5     88              LDX   $A5
C09E: 8E 05 D5  89              STX   $D505      ;switch to current bank
C0A1: D9 00 80  90              CMP   $8000,Y    ;compare data
C0A4: D0 90     91              BNE   MSKLD+1    ;if different, go get next block
C0A6: 88        92              DEY
C0A7: D0 EB     93              BNE   CHK        ;check three bytes
C0A9: F0 B8     94              BEQ   BANKS      ;banks are same--get another bank
                95    ********************************
C0AB: A5 2D     96    END       LDA   $2D
C0AD: 8D 10 12  97              STA   $1210      ;set end of BASIC program
C0B0: A5 2E     98              LDA   $2E
C0B2: 8D 11 12  99              STA   $1211      ;text pointer (in bank zero)
C0B5: A9 1C     100             LDA   #$1C       ;restore start of BASIC program
C0B7: 85 2E     101             STA   $2E        ;text pointer (in bank zero)
C0B9: A9 63     102             LDA   #$63       ;Disable RUN-STOP/
C0BB: 8D 28 03  103             STA   $0328      ;RESTORE
C0BE: A9 EF     104             LDA   #$EF       ; ^
C0C0: 8D 29 03  105             STA   $0329      ; ^
C0C3: A9 00     106             LDA   #$00       ;switch in BASIC and
C0C5: 8D 00 FF  107             STA   $FF00      ;KERNAL ROMs
C0C8: 58        108             CLI              ;restore interupts
C0C9: 4C A6 5A  109             JMP   $5AA6      ;start BASIC program
```