this sequential file into a BASIC-format (SYMASS compatible) program using Chris Zamara's STP program from Volume 5, Issue 06; or the C-64 BASIC STP found in the bits and pieces column in the same issue as the unassembler. Use STP to convert the file to BASIC, then save the resulting source. This file is entirely compatible with SYMASS 3.13, and can be assembled immediately after loading. Once you have changed the unassembler to its new format, the conversions take no time at all.

Using the DOS Wedge With Two Drives — **Joel Pickett, Levelland, Texas**

I use the DOS support program that comes with the 1541 disk drive. I have two drives, but the DOS program only works on one. I modified the DOS loader so it will run on the drive it is loaded from. To do this, line 5 (below) was added -- it peeks location 186, which holds the number of the last device used. Also, the 'dv' in line 10 replaces the '8'.

```
5 dv = peek(186): rem  location 186 is current device #
10 if a = 0 then a = 1: load " dos 5.1 " ,dv,1
20 if a = 1 then sys 12*4096 + 12*256
30 new
```

The DOS support program (at $CC00) gets the current device number from location 186 and stores it internally at $CC77 (52343). Whenever you want to use a DOS command on another drive, simply POKE 52343,(device number).

Should you disable the DOS with a warm start (sys 64738), you can often run it again this way:

```
poke 186,8: sys 52224: return
```

Fast File — **Rick Nash, Millersburg, Ohio**

Here is a short utility that can speed up programs that read from disk files. It works with any kind of file, but it especially handy for direct access (reading a given sector), since the INPUT command is not always reliable under these circumstances. The INPUT command stops reading data whenever it sees a delimeter character (carriage return, colon or comma), so to read unpredictable data the GET command must be used to read the bytes one at a time. This is far too slow for most applications. The program below, Fast File, will read a given number of bytes from a disk file into a string variable, and only stop reading when the given number of characters have been read, or end of file occurs. It reads the data as fast as the disk drive can supply it, since the program is in machine language.

The syntax for using Fast File is:

```
sys 49152,#f,n,v$
```

where 'f' is the file number (the # must be present), 'n' is the number of characters to read, and 'v$' is the name of a string variable that will receive the data.

For example, to read a sequential file:

```
1000 open 1,8,2, " file "
1010 sys 49152,#1,255,a$
1020 print a$;
1030 if st = 0 then 1010
1040 close 1
```

To read 128 bytes of track 18, sector 0 (you can't read all 256 bytes of a sector, since a string can only hold 255 bytes):

```
1000 open 15,8,15
1010 open 2,8,2, " # "
1020 print#15, " u1: " ;2;0;18;0
1030 sys 49152, #2, 128, a$
1040 print a$
1050 close 15
```

The program is fully relocatable; just change the assignment in line 30 of the BASIC loader below. Using Fast File instead of GETs will give you typical speed increases of nine to eleven times!

```
NK  10 rem** fast file **
NE  20 rem read from a file into a variable
PG  30 a = 49152: rem  program is relocatable
AA  40 print " usage: sys " ;a; " ,#<file#>,<# bytes>,
        <string var$> "
BK  50 for i = a to a + 85: read d: c = c + d: poke i,d: next i
HC  60 if c<>11661 then print " !data error! " : stop
KL  70 :
HC  100 data  32, 253, 174, 169,  35,  32, 255, 174
GM  110 data  32, 158, 183, 134, 251,  32, 253, 174
EN  120 data  32, 158, 183, 134, 252,  32, 253, 174
HN  130 data  32, 139, 176, 133,  73, 132,  74,  36
IN  140 data  13,  48,   3,  76, 153, 173, 165, 252
EP  150 data  32, 125, 180, 166, 251,  32, 198, 255
AO  160 data 176,  15, 165, 252, 240,  26, 160,   0
OM  170 data 165, 144, 208,   8,  32,  19, 238, 144
MA  180 data   8,  76, 249, 224, 132,  97,  76,  80
DP  190 data 192, 145,  53, 200, 196, 252, 144, 232
OA  200 data  32, 204, 255,  76, 100, 170
```

Modifying The Epyx Fast Load Cartridge — **James Craig, Waco, TX**

When using the Epyx Fast-load cartridge with the C-128, you have to shut off the machine and install the cartridge in order to switch from C-128 to 64 mode. Besides being a nuisance, this can quickly wear out the cartridge port.

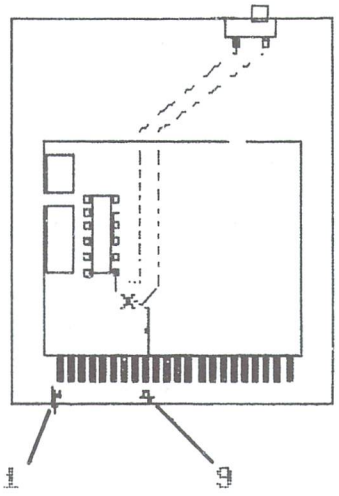
I decided something had to be done. I took the Fast Load cartridge apart and found that my troubles were little ones. I installed a switch in the "EXROM" line to take the ground off the circuit when using C-128 mode. By throwing the switch to connect the ground and hitting the reset button, I was immediately in C-64 mode with the Fast Load cartridge enabled! To go back to C-128, just throw the switch to disconnect the ground, then hit reset again.

To open the cartridge, feel around the top surface for the indentation of the screw that holds the unit together. Just cut away enough to remove the screw. Cut around the box at the seam. then using a

knife blade, pry up all around the box and lift straight up to avoid damaging the interlocking catches.

Install a SPST slide or toggle switch at any convenient location. This could even be outside the case someplace. Cut the printed circuit lead from the #9 male prong about where it makes a bend going to the EXROM connector. Solder a wire on each side and run to each terminal of the switch -- it doesn't make any difference which wire goes where on the switch. Reassemble the case and you're in business. Enjoy your C-64 again!

FAST LOAD



1541 Disk Swap Checker **John Chong, Syracuse, NY**

The following program waits until the current disk in the drive is removed, and another disk (or the same one) re-inserted. It does this by checking the write-protect status of the drive to see if a disk is there or not. It only works if the disks being inserted are NOT write-protected, and even then it can be fooled if you partially remove and then re-insert the disk. Although not bullet-proof, the program shows the technique of checking the write-protect status, and the subroutine at 3000 that actually does the checking may come in handy in one of your programs.

```
2000 print " please change disks. "
2010 open 15,8,15
2020 gosub 3000: if a<>0  then 2020
     :rem  wait for disk to be removed
2030 gosub 3000: if a<>16 then 2030
     :rem  wait for no disk in drive
2040 gosub 3000: if a<>0  then 2040
     :rem  wait for disk to be inserted
2050 for d = 1 to 1500: next: close 15
2060 print " ok, thanks! "
2070 end
2080 :
3000 print#15, " m-r " ;chr$(0)chr$(28)chr$(1)
     :get#15,a$:a = asc(a$)and16:return
```

Easy Retrieval of Last Filename Used **Dave Newberry, Duluth, Minnesota**

In the Bits & Pieces section of Volume 6, Issue 06, Jeffrey Coons wrote in with a one-liner that allowed you to find the name of the last file used (Finding the missing file page 5). Though the line works well, there is an easier way to achieve the same result. A single SYS call is all it takes to get the name of the last file accessed. The magic number is 62913. A **SYS 62913** will print the filename on the screen for all to see.

Chromatic Scale Register Values **Arne Storjohann, Scotland, Ont.**

The following routine generates the SID chip register values which correspond to eight octaves of chromatic scale. The values are separated into high and low byte format and stuffed into two ninety-six element integer arrays to allow for maximum speed of use later in your BASIC program. Due to the ninth place constant D, the values generated are exceedingly precise, limited in resolution only by the 1 through 65535 range imposed by the SID chip. The usual approach is to use data statements and read the 192 values into an array, but with a running time of less than three seconds, this routine is much more compact, efficient, and above all, a more elegant solution.

Anyone who has ever tried to program music on the 64 will appreciate this algorithm!

```
LI  110 rem** routine to generate chromatic
MP  120 rem** scale register values (hi/lo)
AO  130 rem** by arne storjohann - 86,05,04
AA  140 :
EH  150 dim lo%(95),hi%(95): g=2↑(1/12)
DK  160 f=3520*g*g: d=0.06095948: b=256
FF  170 for i = 95 to 0 step -1: n=f/d: hi%(i)=n/b
DP  180 lo%(i)=n-hi%(i)*b: f=f/g: next
CD  190 :
LH  200 rem ** demo **
GE  210 :
MM  220 s=54272: for i = s to s+15: poke i,0: next
BB  230 poke s+5,96: poke s+6,251: poke s+4,33
OA  240 poke s+24,15: for i=-72 to 72
FJ  250 x=71-abs(i)+16: poke s,lo%(x)
EF  260 poke s+1,hi%(x):for j=1to200: next
EF  270 next: poke s+4,32: end
```

C-64 Underlined Characters **D. Munro, Port Elizabeth, South Africa**

This program is based on the C-64 italics program in Bits & Pieces, Volume 7 Issue 01. Instead of giving italics in place of reverse characters however, it gives underlined characters. Both of the 64's built-in character sets are altered, so that underlined letters are available from either upper/lowercase or graphics modes. The new character set is located from 8192 (hex $2000) to 12287 ($2FFF). Consequently, the start of BASIC is moved to $3001.