PICO-8 中文手册

v0.2.6c_rev1

译者: hp7hao

PICO-8 像素炸弹！爱好者交流群：143554779

PICO-8 由以下工具或类库构建:

SDL2 http://www.libsdl.org
  Lua 5.2 http://www.lua.org // 查看 license.txt
  ws281x 作者 jgarff // 查看 license.txt
  GIFLIB http://giflib.sourceforge.net/
  WiringPi http://wiringpi.com/
  libb64 作者 Chris Venter
  miniz 作者 Rich Geldreich
  z8lua 作者 Sam Hocevar https://github.com/samhocevar/z8lua

可以从下面的链接访问最新版本的手册（HTML和TXT格式）以及一些其他资源:

https://www.lexaloffle.com/pico-8.php?page=resources

▨ 欢迎来到 PICO-8!

PICO-8 是一个用于制作、分享和玩小型游戏和其他计算机程序的幻想游戏主机。当你打开它时，机器会用一个用于输入 Lua 程序的控制台迎接你，并提供一些简单的内置工具来创建精灵、地图和声音。

PICO-8 对于规格的严格限制是经过精心选择的，旨在使与之交互的过程是有趣的，且鼓励小型但富有表现力的设计，并希望给 PICO-8 卡带一种独特的外观和感觉。

▨ 规格

显示: 128x128, 固定 16 色调色板
输入: 6 按钮手柄
卡带: 32k 数据量，被编码为 png 文件
声音: 4 通道, 64 个可定义芯片声音
代码: P8 Lua (最多 8192 个代码标记)
CPU: 4M 虚拟机指令/秒
精灵: 一个可容纳 128 个 8x8 精灵的精灵表 (+128 个共享)
地图: 128 x 32 瓦片地图 (+ 128 x 32 共享)

1 开始使用

1.1 键盘快捷键

ALT+ENTER: 切换全屏
ALT+F4: 快速退出 (Windows)
CTRL-Q: 快速退出 (Mac, Linux)
CTRL-R: 重新加载 / 运行 / 重启卡带
CTRL-S: 快速保存工作中的卡带
CTRL-M: 静音 / 取消静音声音
ENTER / P: 暂停菜单 (运行卡带时)

玩家1默认按键: 方向键 + ZX / NM / CV
玩家2默认按键: SDFE + tab, Q / shift A

要更改默认按键，请使用 PICO-8 内部的 KEYCONFIG 工具:

```
> KEYCONFIG
```

1.2 Hello World

启动 PICO-8 后，尝试输入以下命令并按回车：

```
> PRINT("HELLO WORLD")
> RECTFILL(80,80,120,100,12)
> CIRCFILL(70,90,20,14)
> FOR I=1,4 DO PRINT(I) END
```

（注意：PICO-8 仅显示大写字母 -- 无需使用大写锁定键正常输入即可！）

您可以通过在代码编辑模式下使用这些命令以及两个特殊回调函数 **_UPDATE** 和 **_DRAW** 来构建交互式程序。例如，以下程序允许您使用方向键移动一个圆圈。按 Esc 键切换到代码编辑器并输入或复制粘贴以下代码：

```
X = 64  Y = 64
FUNCTION \_UPDATE()
  IF (BTN(0)) THEN X=X-1 END
  IF (BTN(1)) THEN X=X+1 END
  IF (BTN(2)) THEN Y=Y-1 END
  IF (BTN(3)) THEN Y=Y+1 END
END

FUNCTION \_DRAW()
  CLS(5)
  CIRCFILL(X,Y,7,14)
END
```

现在按 Esc 键返回到控制台并输入 RUN（或按 CTRL-R）查看效果。请参阅演示卡带以获取更复杂的程序（输入 INSTALL_DEMOS）。

如果您想保存程序以供以后使用，请使用 SAVE 命令：

```
> SAVE PINKCIRC
```

要再次加载它：

```
> LOAD PINKCIRC
```

1.3 示例卡带

这些卡带包含在 PICO-8 中，可以通过以下命令安装：

```
> INSTALL_DEMOS
> CD DEMOS
> LS
```

```
HELLO       来自 PICO-8 的问候
API         演示大多数 PICO-8 函数
JELPI       支持双人的平台游戏演示
CAST        2.5D 射线投射演示
DRIPPY      绘制一条滴答的曲线
WANDER      简单的步行模拟器
COLLIDE     墙壁和角色碰撞示例
```

要运行一个卡带，请打开 PICO-8 并输入：

```
> LOAD JELPI
> RUN
```

按 escape 键停止程序，再次按以进入编辑模式。

还可以通过以下命令将一些 BBS 卡带安装到 /GAMES：

```
> INSTALL_GAMES
```

1.4 文件系统

这些命令可以用来管理文件和目录（文件夹）：

```
LS          列出当前目录
CD BLAH     更改目录
CD ..       返回上一级目录
CD /        返回到 PICO-8 虚拟磁盘的顶级目录
MKDIR BLAH  创建一个目录
FOLDER      在主机操作系统的文件浏览器中打开当前目录

LOAD BLAH   从当前目录加载一个 cart
SAVE BLAH   将 cart 保存到当前目录
```

如果你想移动文件、复制文件或删除文件，请使用 FOLDER 命令并在主机操作系统中进行操作。

PICO-8 驱动器的默认位置为：

```
Windows: C:/Users/Yourname/AppData/Roaming/pico-8/carts
OSX:     /Users/Yourname/Library/Application Support/pico-8/carts
Linux:   ~/.lexaloffle/pico-8/carts
```

你可以在 pico-8/config.txt 中更改此和其他设置。

提示：驱动器目录可以映射到云驱动器（由 Dropbox、Google Drive 或类似服务提供），以便在不同的主机机器上创建一个共享的磁盘。

1.5 加载和保存

使用 LOAD 和 SAVE 时，可以省略 .P8 扩展名，它会自动添加。

```
> SAVE FOO
SAVED FOO.P8
```

也可以将卡带文件拖放到 PICO-8 的窗口中进行加载。

使用 .p8.png 文件名扩展名将卡带以特殊图像格式写入，该格式看起来像一个卡带。使用 .p8.rom 写入原始 32k 二进制格式。

使用文件名为 "@clip" 可以加载或保存到剪贴板。

使用文件名为 "@url" 可以将内容保存到剪贴板作为 pico-8-edu.com 的 URL，前提是内容可以编码在 2040 个字符以内（仅限代码和图形）。

一旦卡带已加载或保存，也可以使用 CTRL-S 快速保存。

■ 使用文本标签和预览图像保存 .p8.png 卡带

要生成与卡带一起保存的标签图像，请先运行程序，然后按 CTRL-7 抓取屏幕上的内容。程序开头以 '--' 开始的前两行也会绘制到卡带的标签上。

```
-- OCEAN DIVER LEGENDS
-- BY LOOPY
```

■ .p8.png / .p8.rom 格式的代码大小限制

在保存为 .png 或 .rom 格式时，代码的压缩大小必须小于 15360 字节，以便总数据量 <= 32k。

要查看当前代码的大小，请使用 INFO 命令。保存为 .p8 格式时，不会强制执行压缩大小限制。

1.6 使用外部文本编辑器

使用单独的文本编辑器直接编辑 .p8 文件是可能的。使用 CTRL-R 运行卡带时，如果满足以下条件，文件会自动重新加载：

1. PICO-8 编辑器中没有未保存的更改，AND
2. 文件的内容与上次加载的版本不同。

如果磁盘上的卡带和编辑器中都有更改，会显示通知（不要重新加载，存在未保存的更改）：

```
DIDN'T RELOAD; UNSAVED CHANGES
```

另外，可以在单独的编辑器中修改 .lua 文本文件，然后每次运行卡带时使用 **#INCLUDE**（在所需代码位置）包含到卡带的代码中：

```
#INCLUDE YOURFILE.LUA
```

1.7 备份

当不保存更改而退出，或覆盖现有文件时，会将卡带的备份保存到 {appdata}/pico-8/backup。还可以通过输入 BACKUP 命令将当前卡带的额外副本保存到同一文件夹中。

要使用主机操作系统中的文件浏览器打开备份文件夹，请使用：

```
> FOLDER BACKUPS
```

然后可以将文件拖放到 PICO-8 窗口中以加载它们。

从 0.2.4c 版本开始，在编辑器中非空闲状态时每 20 分钟会自动保存一次定期备份，这意味着备份文件夹大约每 5 小时会增长约 1MB。可以在 config.txt 中禁用或调整此功能。

1.8 配置

PICO-8 在每次会话开始时从 config.txt 文件读取配置设置，并在退出时保存（因此，当 PICO-8 没有运行时应编辑 config.txt）。

config.txt 文件的位置取决于主机操作系统：

```
Windows: C:/Users/Yourname/AppData/Roaming/pico-8/config.txt
OSX: /Users/Yourname/Library/Application Support/pico-8/config.txt
Linux: ~/.lexaloffle/pico-8/config.txt
```

使用 -home 开关（见下文）以使用不同的路径来存储 config.txt 和其他数据。

某些设置可以在运行 PICO-8 时通过输入 CONFIG SETTING VALUE 来更改。（单独输入 CONFIG 可以查看设置列表）

■ 命令行参数

```
// 注意: 这些参数会覆盖 config.txt 中的设置

pico8 [开关] [filename.p8]

-width n             设置窗口宽度
-height n            设置窗口高度
-windowed n          设置窗口模式为关闭 (0) 或开启 (1)
-volume n            设置音频音量 0..256
-joystick n          手柄控制从玩家 n 开始 (0..7)
-pixel_perfect n     1 表示在整数缩放时使用无过滤屏幕拉伸（默认开启）
-preblit_scale n     在绘制到屏幕前按 n 缩放显示（与 -pixel_perfect 0 结合使用时有用）
-draw_rect x,y,w,h   绘制 pico-8 屏幕的绝对窗口坐标和大小
-run filename        加载并运行一个卡带
-x filename          以无头模式执行一个 PICO-8 卡带然后退出（实验性功能！）
-export param_str    以无头模式运行 EXPORT 命令然后退出（参见 export 下的说明）
-p param_str         向指定卡带传递一个参数字符串
-splore              以 splore 模式启动
-home path           设置存储 config.txt 和其他用户数据文件的路径
-root_path path      设置存储卡带文件的路径
-desktop path        设置保存截图和 GIF 的位置
-screenshot_scale n 截图的缩放比例。默认: 3 (368x368 像素)
-gif_scale n         GIF 捕获的缩放比例。默认: 2 (256x256 像素)
-gif_len n           设置 GIF 最大长度（秒）。范围: 1..120
-gui_theme n         使用 1 表示使用高对比度的编辑器颜色方案
-timeout n           下载超时等待时间（秒）。默认: 30
-software_blit n     使用软件绘制模式关闭 (0) 或开启 (1)
-foreground_sleep_ms n   每帧之间休眠的毫秒数。
-background_sleep_ms n   在后台运行时每帧之间休眠的毫秒数
-accept_future n     1 表示允许加载使用未来版本的 PICO-8 创建的卡带
-global_api n        1 表示将 API 函数保留在全局作用域（调试时有用）
```

▨ 控制器设置

PICO-8 使用 SDL2 控制器配置方案。它会在启动时检测常见的控制器，并且还会查找与 config.txt 相同目录下的自定义映射文件 sdl_controllers.txt。sdl_controllers.txt 中每行一个映射。

要为你的控制器生成自定义映射字符串，可以使用随 SDL2 提供的 controllermap 程序，或者尝试 http://www.generalarcade.com/gamepadtool/

要查找 SDL2 检测到的控制器 ID，请在运行 PICO-8 后在 log.txt 中搜索 "joysticks" 或 "Mapping"。此 ID 可能在不同的操作系统下有所不同。参见: https://www.lexaloffle.com/bbs/?tid=32130

要设置哪些键盘按键触发手柄按钮按下，请使用 KEYCONFIG。

1.9 截图和 GIF

在运行卡带时使用以下快捷键：

```
CTRL-6 保存截图到桌面
CTRL-7 捕获卡带标签图像
CTRL-8 开始录制视频
CTRL-9 保存 GIF 视频到桌面（默认 8 秒）
```

您可以在任何时间保存视频（它始终在录制）；CTRL-8 只是重置视频的起始点。要录制超过 8 秒，请使用 CONFIG 命令（最大：120 秒）

```
CONFIG GIF_LEN 60
```

如果您希望每次录制都重置（以创建非重叠序列），请使用：

```
CONFIG GIF_RESET_MODE 1
```

GIF 格式无法精确匹配 30fps，因此 PICO-8 使用最接近的匹配：33.3fps。

如果您在保存到桌面时遇到问题，请尝试在 config.txt 中配置替代的桌面路径。

1.10 分享卡带

在 PICO-8 中有三种分享卡带的方式：

1. 直接将 .p8 或 .p8.png 文件分享给其他 PICO-8 用户

输入 FOLDER 打开当前文件夹在主机操作系统中的位置。

2. 将卡带发布到 Lexaloffle BBS 以获取一个可网页播放的版本

http://www.lexaloffle.com/pico-8.php?page=submit

3. 将卡带导出为独立的 html/js 或原生二进制播放器（详情请参见导出部分）

1.11 SPLORE

SPLORE 是一个内置工具，用于浏览和组织本地和 BBS（在线）卡带。输入 SPLORE [回车] 启动它，或者使用 -splore 启动 PICO-8。

可以通过手柄完全控制 SPLORE：

```
- 左右键导航卡带列表
- 上下键选择每个列表中的项目
- X、O 或 MENU 启动卡带
```

在卡带内部，按 MENU 可以将卡带设为收藏，或退出到 SPLORE。如果使用键盘，也可以在卡带列表视图中选择项目后按 F 将其设为收藏。

查看 BBS 卡带列表时，使用顶部列表项重新下载卡带列表。如果处于离线状态，将显示上次下载的列表，仍然可以播放已下载的任何卡带。

如果在没有互联网访问的机器上安装了 PICO-8，也可以使用 INSTALL_GAMES 将少量预安装的 BBS 卡带添加到 /games

2 编辑工具

按 ESC 在控制台和编辑器之间切换。
点击右上角的编辑模式选项卡以切换，或按 ALT+LEFT/RIGHT。

2.1 代码编辑器

```
按住 Shift 键可以选择（或使用鼠标点击并拖动）
CTRL-X, C, V 剪切、复制或粘贴选中的内容
CTRL-Z, Y 撤销、重做
CTRL-F 在当前标签页中搜索文本
CTRL-G 重复上一次搜索
CTRL-L 跳转到行号
CTRL-UP, DOWN 跳转到文件的开头或结尾
ALT-UP, DOWN 导航到上一个、下一个函数
CTRL-LEFT, RIGHT 逐词跳转
CTRL-W, E 跳转到当前行的开头或结尾
CTRL-D 复制当前行
TAB 缩进选中的内容（Shift 取消缩进）
CTRL-B 注释或取消注释选中的块
CTRL-U 获取光标下关键字的帮助
```

要输入代表按钮的特殊字符（以及其他符号），使用 SHIFT-L, R, U, D, O, X。有 3 种额外的字体输入模式可以切换：

```
CTRL-J 日文平假名  // 输入罗马音等价字符 (ka, ki, ku..)
CTRL-K 日文片假名  // + shift-0..9 输入额外符号
```

ⓘ 默认情况下，小字体字符在复制/粘贴时会被编码为 Unicode 替换字符，而大小写 ASCII 字符会被粘贴为常规的 PICO-8 字符。要将小字体字符作为大写 ASCII 复制/粘贴，请确保小字体模式（CTRL-P）已打开。

■ 代码标签

点击顶部的 [+] 按钮添加新标签。通过左键点击或使用 CTRL-TAB, SHIFT-CTRL-TAB 导航标签。要删除最后一个标签，请删除其中的所有内容，然后移开光标（CTRL-A, DEL, CTRL-TAB）

运行一个卡带时，通过按顺序连接所有标签生成一个单一的程序。

■ 代码限制

代码标记的数量显示在右下角。一个程序最多可以有 8192 个标记。每个标记是一个单词（例如变量名）或运算符。成对的括号和字符串各计为 1 个标记。逗号、句号、LOCAL、分号、END 和注释不计入。

右键点击切换其他统计信息（字符计数、压缩大小）。如果达到限制，警告灯会闪烁。可以通过右键点击禁用此功能。

2.2 精灵编辑器

精灵编辑器既可以用于精灵级别的编辑，也可以用于自由形式的像素级别编辑。屏幕底部的精灵导航器提供了对精灵表的 8x8 精灵级别的视图，但在处理较大或形状不规则的区域时，可以使用自由形式工具（平移、选择）。

■ 绘制工具

点击并拖动精灵以绘制像素，或使用右键选择光标下的颜色。

所有操作仅适用于可见区域，或如果有部分，则适用于该部分。

按住 CTRL 可以搜索并替换颜色。

■ 图章工具

点击以在复制缓冲区中盖章。按住 CTRL 可以将颜色 0（黑色）视为透明。

■ 选择工具（快捷键：SHIFT 或 S）

点击并拖动以创建矩形选择。要移除选择，按 ENTER 或点击任何地方。

如果没有像素级选择，许多操作将应用于精灵级选择或可见视图。要选择精灵，请在精灵导航器中按住 SHIFT 并拖动。要选择精灵表，请按 CTRL-A（重复以切换关闭与地图数据共享的下半部分）

■ 平移工具（快捷键：SPACE）

点击并拖动以移动精灵表。

■ 填充工具

用当前颜色填充。这仅适用于当前选择，如果没有选择，则适用于可见区域。

■ 形状工具

点击工具按钮以循环选择：椭圆、矩形、线条选项。

按住 CTRL 可以获得填充的椭圆或矩形。
按住 SHIFT 可以对齐为圆形、正方形或低整数比的线条。

■ 额外快捷键

```
CTRL-Z:    撤销
CTRL-C/X:  复制/剪切选中的区域或选中的精灵
CTRL-V:    粘贴到当前精灵位置
Q/A,W/Z:   切换到前一个/下一个精灵
1,2:       切换到前一个/下一个颜色
TAB:       切换全屏视图
鼠标滚轮或 < 和 > 来缩放（全屏时居中）
CTRL-H 切换十六进制视图（显示精灵索引的十六进制）
CTRL-G 切换放大时的黑色网格线
```

■ 对选中的区域或选中的精灵的操作

```
F：水平翻转精灵
V：垂直翻转精灵
R：旋转（需要方形选择）
方向键平移（如果精灵选择则循环）
DEL/BACKSPACE 清除选中的区域
```

■ 精灵标志

8 个彩色圆圈是当前精灵的精灵标志。这些标志没有特定含义，但可以使用 **FGET**() / **FSET**() 函数访问。它们从左开始索引为 0。

有关更多信息，请参见 **FSET**()。

■ 将 .png 文件加载到精灵表

要将任何大小的 .png 文件加载到精灵表中，首先选择应作为左上角目标的精灵，然后键入 "IMPORT IMAGE_FILE.PNG" 或将图像文件拖放到 PICO-8 窗口中。在这两种情况下，图像都会根据当前显示调色板进行颜色匹配。

2.3 地图编辑器

PICO-8 地图是一个 128x32（或使用共享空间时为 128x64）的 8 位值块。每个值在编辑器中显示为对精灵的引用（0..255），但你可以使用这些数据来表示任何你想要的内容。

ⓘ **警告**：精灵表的第二部分（银行 2 和 3）和地图的下半部分共享相同的卡带空间。数据的使用方式由你决定，但请注意，在精灵表的第二部分绘图可能会覆盖地图的数据，反之亦然。

工具与精灵编辑模式中使用的工具类似。选择一个精灵并点击拖动以在地图上绘制值。

要绘制多个精灵，使用 Shift+拖动从精灵导航器中选择。要复制一个值块，使用选择工具，然后使用图章工具粘贴。要平移地图，使用平移工具或按住空格键。Q/W 切换到前一个/下一个精灵。鼠标滚轮或 < 和 > 来缩放（全屏时居中）。Ctrl-H 切换十六进制视图（显示瓦片值和精灵索引的十六进制）。

在不破坏地图中引用的情况下移动精灵表中的精灵有点棘手，但可以实现：

1. 按 ENTER 清除任何地图选择
2. 选择你想要移动的精灵（仍在地图视图中），然后按 Ctrl-X

3. 选择你想要更改的地图区域（默认为地图的上半部分） // 按两次 Ctrl-A 以选择包括共享内存在内的整个地图
4. 选择目标精灵（仍在地图视图中），然后按 Ctrl-V

// 注意：此操作会修改地图和精灵编辑器的撤销历史记录，但 // PICO-8 会尽可能保持它们同步。否则，通过手动在精灵编辑器中撤销更改， // 可以撤销移动地图精灵引起的更改。

2.4 SFX 编辑器

PICO-8 卡带中有 64 个 SFX（声音效果），用于声音和音乐。

每个 SFX 有 32 个音符，每个音符包含：　　频率　(C0..C5)
　乐器　(0..7)
　音量　　(0..7)
　效果　　(0..7)

每个 SFX 还有以下属性：

播放速度 (SPD) : 每个音符播放的 'tick' 数。
　// 这意味着 1 是最快，3 是 3 倍慢，等等。

循环开始和结束 : 这是循环返回的音符索引
　// 当开始索引 >= 结束索引时，循环关闭

当只使用两个数字中的第一个（第二个为 0）时，它表示要播放的音符数量。这通常不需要用于声音效果（可以留空剩余的音符），但对于控制音乐播放很有用。

有 2 种模式用于编辑/查看 SFX：音高模式（更适合声音效果）和跟踪模式（更适合音乐）。可以通过左上角的按钮更改模式，或使用 TAB 切换。

▨ 音高模式

在音高区域点击并拖动以设置每个音符的频率，使用当前选定的乐器（由颜色指示）。

按住 shift 只应用选定的乐器。
按住 CTRL 将输入的音符对齐到 C 小调五声音阶。
右键点击抓取该音符的乐器。

▨ 跟踪模式

每个音符显示：频率 八度 乐器 音量 效果
要输入一个音符，使用 q2w3er5t6y7ui zsxdcvgbhnjm（钢琴布局）

输入音符时按住 shift 转移 -1 八度 .. +1 八度
新音符将获得选定的乐器/效果值
要删除一个音符，使用 backspace 或将音量设置为 0

点击并然后 shift-点击以选择一个范围，可以复制（CTRL-C）和粘贴（CTRL-V）。请注意，只有选定的属性会被复制。双击选择单个音符的所有属性。

导航:
PAGEUP/DOWN 或 CTRL-UP/DOWN 跳过 4 个音符
HOME/END 跳转到第一个或最后一个音符
CTRL-LEFT/RIGHT 跳过列

▨ 两种模式的控制

- + 导航当前 SFX
< > 更改速度。SPACE 播放/停止
SHIFT-SPACE 从当前 SFX 四分之一（8 个音符）播放
A 释放循环采样
左键或右键点击增加 / 减少 SPD 或 LOOP 值
// 按住 shift 点击增加 / 减少 4
// 或者点击并拖动左右或上下
Shift-点击乐器、效果或音量以应用到所有音符。

▨ 效果

0 无
1 滑音 // 滑到下一个音符和音量
2 颤音 // 在一个四分之一音内快速变化音高
3 下降 // 快速降低频率到很低的值
4 渐入 // 从 0 升音量
5 渐出 // 将音量降到 0
6 快琶音 // 以速度 4 迭代 4 个音符组
7 慢琶音 // 以速度 8 迭代 4 个音符组

如果 SFX 速度 <= 8，琶音速度减半为 2, 4

▨ 滤波器

每个 SFX 在跟踪模式下有 5 个滤波器开关:

NOIZ: 生成纯白噪声（仅适用于乐器 6）
BUZZ: 各种波形变化，使声音更像蜂鸣声
DETUNE-1: 调音第二个声音以创建类似颤音的效果

DETUNE-2: 各种第二个声音调音，大多上下八度
REVERB:　应用延迟为 2 或 4 ticks 的回声
DAMPEN:　两级低通滤波器

当使用 BUZZ 和乐器 6，且 NOIZ 关闭时，生成纯棕色噪声。

2.5 音乐编辑器

PICO-8 中的音乐由一系列“模式”控制。每个模式是一个包含 4 个数字的列表，指示该通道将播放哪个 SFX。

▨ 流程控制

播放流程可以通过右上角的 3 个按钮进行控制。

当一个模式播放完毕时，会播放下一个模式，除非：

- 没有数据可供播放（音乐停止）
- 该模式设置了 STOP 命令（第三个按钮）
- 设置了 LOOP BACK 命令（第二个按钮），在这种情况下，音乐播放器会回溯查找设置了 LOOP START 命令的模式（第一个按钮），如果没有找到，则返回到模式 0。

当一个模式包含不同速度的 SFX 时，模式会在最左边的非循环通道播放完毕时结束播放。这可以用来设置双倍时间的鼓点或不寻常的复节奏。

对于 3/4 这样的时间签名，如果应该在跳转到下一个模式之前播放少于 32 行，可以通过仅调整第一个循环位置并保持第二个位置为零来设置 SFX 的长度。这将在 SFX 编辑器中显示为 "LEN"（表示“长度”），而不是 "LOOP"。

▨ 复制和粘贴音乐

要选择一系列模式：单击模式导航器中的第一个模式，然后按住 Shift 单击最后一个模式。可以选择的模式可以使用 CTRL-C 和 CTRL-V 进行复制和粘贴。粘贴到另一个卡带时，如果 SFX 不存在，每个模式指向的 SFX 也会被粘贴（可能具有不同的索引）。

▨ SFX 乐器

除了 8 个内置乐器外，还可以使用前 8 个 SFX 定义自定义乐器。使用乐器右侧的切换按钮选择一个索引，该索引将在乐器通道中显示为绿色而不是粉色。

当播放 SFX 乐器音符时，它实际上会触发该 SFX，但会改变音符的属性：

- 音高相对于 C2 添加
- 音量乘以
- 效果会叠加到 SFX 乐器的效果上
- SFX 乐器中启用的任何滤波器也会为该音符启用

例如，可以通过在 SFX 0 中定义一个快速在音量 5 和 2 之间交替的简单颤动效果来实现。使用该乐器播放音符时，音量可以像平常一样进一步改变（通过音量通道或使用淡入/淡出效果）。这样，SFX 乐器可以用来控制音量、音高和纹理的详细变化组合。

SFX 乐器仅在音高变化或前一个音符的音量为零时重新触发。这对于随时间变化较慢的乐器很有用。例如：逐渐淡出的铃声。要反转此行为，可以在触发音符时使用效果 3（通常是“drop”）。触发 SFX 乐器时，其他所有效果值具有其通常的含义。

▨ 波形乐器

波形乐器的工作方式与 SFX 乐器相同，但由一个自定义的 64 字节循环波形组成。在 SFX 编辑器中点击波形切换按钮以使用 SFX 0..7 作为波形乐器。在这种模式下，可以使用鼠标绘制样本。

▨ 音阶捕捉

在音高模式下绘制音符时，按住 CTRL 可以捕捉到当前定义的音阶。默认音阶为 C 小调五声音阶，但可以使用音阶编辑模式进行自定义。右下角有一个小键盘图标可以切换此功能。有 2 个移调按钮、1 个反转按钮和 3 个音阶预设按钮：

```
Dim   减七度音阶 // 反转以获得全半音阶
Maj   大调音阶          // 反转以获得五声音阶
Who   全音阶     // 反转以获得.. 另一个全音阶
```

更改音阶不会改变当前的 SFX，只有在按住 CTRL 绘制新音符时才会应用音阶。

3 导出器 / 导入器

EXPORT 命令可以用于生成 png、wav 文件以及独立的 html 和原生二进制 cartridges 应用程序。输出格式根据文件扩展名推断（例如 .png）。

您可以自由分发和使用导出的 cartridges 和数据，前提是您获得了 cartridge 作者和贡献者的许可。

精灵表 / 标签 (.png)

```
> IMPORT BLAH.PNG    -- 期望 128X128 PNG 并且颜色适合 PICO-8 调色板
> EXPORT BLAH.PNG    -- 使用 "FOLDER" 命令定位导出的 PNG
```

导入时，可以使用 -x 和 -y 开关指定目标位置（以像素为单位）：-s 可以用于缩小图像（3 表示从 384x384 缩放到 128x128）

```
> IMPORT BLAH.PNG -X 16 -Y 16 -S 3
```

使用 -l 开关与 IMPORT 和 EXPORT 一起使用，可以从 cartridge 的标签读取和写入：

```
> IMPORT -L BLAH.PNG
```

ⓘ 导入精灵表或标签时，调色板会根据当前绘图状态的调色板进行颜色适配。

SFX 和音乐 (.wav)

要从当前模式（当编辑器模式为音乐时）或当前 SFX 导出音乐：

```
> EXPORT FOO.WAV
```

要导出所有 SFX 为 foo0.wav, foo1.wav .. foo63.wav：

```
> EXPORT FOO%D.WAV
```

地图和代码

cartridge 的地图或源代码可以导出为名为 .map.png 或 .lua.png 的单个图像：

```
> EXPORT FOO.MAP.PNG
> EXPORT FOO.LUA.PNG
```

地图图像为 1024x512（128x32 8x8 精灵）。Lua 图像大小适合，但每行固定（并裁剪）为 192 像素宽。

cartridges (.p8, .p8.png, .p8.rom)

使用 EXPORT 保存 cartridge 与使用 SAVE 相同，但不会更改当前的工作 cartridge。这在例如要保存一个 .p8.png 格式的副本用于分发而不意外继续更改该文件而不是原始 .p8 文件时很有用。

EXPORT 还可以用于从命令行执行 cartridge 文件格式转换。例如，在 Linux shell 中：

```
> pico8 foo.p8 -export foo.p8.png
```

3.1 Web 应用程序 (.html)

可以生成一个独立的 HTML 播放器（mygame.html, mygame.js）：

```
> EXPORT MYGAME.HTML
```

或者只生成 .js 文件：

```
> EXPORT MYGAME.JS
```

使用 -f 选项将文件写入名为 mygame_html 的文件夹，并使用 index.html 而不是 mygame.html

```
> EXPORT -F MYGAME.HTML
```

可以使用 -p 选项提供自定义 HTML 模板：

```
> EXPORT MYGAME.HTML -P ONE_BUTTON
```

这将使用文件 {application data}/pico-8/plates/one_button.html 作为 HTML 外壳，用 .js 文件名替换特殊的字符串 "##js_file##"（不带引号），并可选地用卡带的标签图像替换字符串 "##label_file##" 作为数据 URL。

使用 -w 选项导出为 .wasm + .js：

```
> EXPORT -W MYGAME.HTML
```

ⓘ 当导出为 .wasm 时，页面需要通过 Web 服务器提供，而不是直接从本地文件系统在浏览器中打开。对于大多数用途，默认的 .js 导出已经足够，但 .wasm 稍小且更快。

3.2 二进制应用程序 (.bin)

要生成适用于 Windows、Linux (64 位)、Mac 和 Raspberry Pi 的独立可执行文件：

EXPORT MYGAME.BIN

默认情况下，卡带标签将用作图标且没有透明度。要指定一个来自精灵表的图标，请使用 -i，并可选地使用 -s 和/或 -c 来控制大小和透明度。

```
-I N  使用默认透明色为 0（黑色）的图标索引 N。
-S N  大小为 NxN 的精灵。大小 3 将生成一个 24x24 的图标。
-C N  将颜色 N 视为透明。使用 16 表示无透明。
```

例如，要使用精灵表中索引 32 开始的 2x2 精灵，并将颜色 12 作为透明色：

```
> EXPORT -I 32 -S 2 -C 12 MYGAME.BIN
```

要将一个额外的文件包含在输出文件夹和归档文件中，请使用 -E 开关：

```
> EXPORT -E README.TXT MYGAME.BIN
```

ⓘ Windows 文件系统不支持创建 Linux 或 Mac 可执行文件所需的文件元数据。PICO-8 通过以一种保留文件属性的方式导出 zip 文件来解决这个问题。因此，建议您按原样分发输出的 zip 文件，以确保其他操作系统上的用户可以运行它们。否则，Linux 用户在下载二进制文件后可能需要对文件执行 "chmod +x mygame" 以运行它，而 Mac 用户则需要对 "chmod +x mygame.app/Contents/MacOS/mygame" 执行相同操作。

3.3 上传到 itch.io

如果你希望将导出的卡带到 itch.io 作为可玩的 HTML 上传：

1. 在 PICO-8 内部: EXPORT -F MYGAME.HTML
2. 在你的 itch 仪表板中创建一个新项目。
3. 将文件夹压缩并上传（设置“This file will be played in the browser”）
4. 嵌入页面，大小为 750px x 680px。

5. 设置“Mobile Friendly”为开启（默认方向）和“Automatically start on page load”为开启。
  // 不需要全屏按钮，因为默认的 PICO-8 模板已经自带。
6. 将背景（BG2）设置为深色（例如 #232323），并将文本设置为浅色（#cccccc）

3.4 导出多个卡带

最多可以将 32 个卡带打包在一起，通过将它们传递给 EXPORT，当生成独立的 HTML 或原生二进制播放器时。

```
> EXPORT MYGAME.HTML DAT1.P8 DAT2.P8 GAME2.P8
```

在运行时，额外的卡带可以像本地文件一样访问：

```
RELOAD(0,0,0X2000, "DAT1.P8") -- 从 DAT1.P8 加载精灵表
LOAD("GAME2.P8")              -- 加载并运行另一个卡带
```

ⓘ 导出的卡带无法加载和运行 BBS 卡带，例如通过 LOAD("#FOO")

3.5 从主机操作系统运行 EXPORT

使用启动 PICO-8 时的 -export 开关以无头模式运行导出器。文件路径相对于当前目录而不是 PICO-8 文件系统。

导出命令的参数作为单个（小写）字符串传递：

```
pico8 mygame.p8 -export "-i 32 -s 2 -c 12 mygame.bin dat0.p8 dat1.p8"
```

4 Lua 语法入门

PICO-8 程序使用 Lua 语法编写，但不使用标准的 Lua 库。以下是 Lua 语法的基本摘要。

更多详细信息或了解标准 Lua，请访问 www.lua.org。

注释

```
-- 使用两个短横线写注释
--[[ 多行
注释 ]]
```

类型和赋值

Lua 中的类型包括数字、字符串、布尔值和表：

```
NUM = 12/100
S = "这是一个字符串"
B = FALSE
T = {1,2,3}
```

PICO-8 中的数字都是 16:16 固定点。范围从 -32768.0 到 32767.99999

可以使用带有可选小数部分的十六进制表示法：

```
?0x11        -- 17
?0x11.4000   -- 17.25
```

以十进制书写的数字会被四舍五入到最近的固定点值。要查看 32 位十六进制表示法，请使用 PRINT(TOSTR(VAL,TRUE))：

```
?TOSTR(-32768,TRUE)      -- 0x8000.0000
?TOSTR(32767.99999,TRUE) -- 0X7FFF.FFFF
```

除以零的结果为 0x7fff.ffff（如果为正），或 -0x7fff.ffff（如果为负）。

条件语句

```
IF NOT B THEN
  PRINT("B 为 FALSE")
ELSE
  PRINT("B 不是 FALSE")
END
```

-- 使用 ELSEIF

```
IF X == 0 THEN
  PRINT("X 为 0")
ELSEIF X < 0 THEN
  PRINT("X 为负数")
ELSE
  PRINT("X 为正数")
END
```

```
IF (4 == 4) THEN PRINT("相等") END
IF (4 ~= 3) THEN PRINT("不相等") END
IF (4 <= 4) THEN PRINT("小于或等于") END
IF (4 > 3) THEN PRINT("大于") END
```

循环

循环范围是包含的：

```
FOR X=1,5 DO
  PRINT(X)
END
-- 打印 1,2,3,4,5
```

```
X = 1
WHILE(X <= 5) DO
  PRINT(X)
  X = X + 1
END
```

```
FOR X=1,10,3 DO PRINT(X) END   -- 1,4,7,10
```

```
FOR X=5,1,-2 DO PRINT(X) END  -- 5,3,1
```

函数和局部变量

使用 LOCAL 声明的变量作用域仅限于其包含的代码块（例如，函数、FOR 循环或 IF THEN END 语句）。

```
Y=0
FUNCTION PLUSONE(X)
  LOCAL Y = X+1
  RETURN Y
END
PRINT(PLUSONE(2)) -- 3
PRINT(Y)          -- 0
```

表

在 Lua 中，表是键值对的集合，其中键和值的类型都可以混合。可以通过整数索引使用表作为数组。

```
A={} -- 创建一个空表
A\[1\] = "BLAH"
A\[2\] = 42
A\["FOO"\] = {1,2,3}
```

数组默认使用 1 基索引：

```
> A = {11,12,13,14}
> PRINT(A\[2\]) -- 12
```

但如果希望使用 0 基数组，只需写入零槽位：

```
> A = {\[0\]=10,11,12,13,14}
```

具有 1 基整数索引的表是特殊的。可以使用 # 运算符找到此类数组的长度，并且 PICO-8 使用此类数组来实现 ADD、DEL、DELI、ALL 和 FOREACH 函数。

```
> PRINT(#A)   -- 4
> ADD(A, 15)
> PRINT(#A)   -- 5
```

字符串索引可以使用点表示法编写

```
PLAYER = {}
PLAYER.X = 2 -- 等价于 PLAYER\["X"\]
PLAYER.Y = 3
```

有关更多详细信息，请参见 **表函数** 部分。

PICO-8 简写

PICO-8 还允许使用几种非标准的简写方式来编写常见模式。

1. IF THEN END 语句和 WHILE THEN END 可以写在一行中，如下所示：

```
IF (NOT B) I=1 J=2
```

等价于：

```
IF NOT B THEN I=1 J=2 END
```

注意，简写条件周围的括号是必需的。

2. 赋值运算符

如果整个语句在一行中，也可以使用简写赋值运算符。它们可以通过在任何二进制运算符后面附加 '=' 来构造，包括算术运算符（+=, -= ..），按位运算符（&=, |= ..）或字符串连接运算符（..=）

```
A += 2   -- 等价于: A = A + 2
```

3. != 运算符

不是简写，但 pico-8 也接受 != 代替 ~= 表示“不等于”

```
PRINT(1 != 2) -- TRUE
PRINT("FOO" == "FOO") -- TRUE (字符串是内部化的)
```

PICO-8 程序结构

当 PICO-8 程序运行时，所有标签中的代码会从左到右连接并执行。可以手动提供自己的主循环，但通常 PICO-8 程序使用 3 个特殊函数，如果作者定义了这些函数，它们会在程序执行期间被调用：

_UPDATE() -- 每更新一次调用一次，帧率为 30fps。

_DRAW() -- 每个可见帧调用一次

_INIT() -- 程序启动时调用一次。

一个使用所有三个函数的简单程序可能如下所示：

```
FUNCTION _INIT()
  -- 总是开始于白色
  COL = 7
END

FUNCTION _UPDATE()
  -- 按下 X 键以随机颜色
  IF (BTNP(5)) COL = 8 + RND(8)
END

FUNCTION _DRAW()
  CLS(1)
  CIRCFILL(64,64,32,COL)
END
```

_DRAW() 通常以 30fps 调用，但如果无法及时完成，PICO-8 将尝试以 15fps 运行并每帧调用 _UPDATE() 两次以补偿。

▨ 以 60fps 运行 PICO-8

当定义 _UPDATE60() 而不是 _UPDATE() 时，PICO-8 将以 60fps 模式运行：

- _UPDATE60() 和 _DRAW() 都以 60fps 调用
- 每帧可用的 PICO-8 CPU 占比为一半，然后降至 30fps

请注意，并非所有主机机器都支持以 60fps 运行。较旧的机器或 Web 版本可能会请求 PICO-8 以 30 fps（或 15 fps）运行，即使 PICO-8 CPU 不过载。在这种情况下，每个 _DRAW 调用都会进行多个 _UPDATE60 调用，方式与之前相同。

▨ #INCLUDE

源代码可以在卡带启动时注入（但不能在运行时），使用 "#INCLUDE FILENAME"，其中 FILENAME 可以是纯文本文件（包含 Lua 代码）、另一个卡带的标签，或另一个卡带的所有标签：

```
#INCLUDE SOMECODE.LUA
#INCLUDE ONETAB.P8:1
#INCLUDE ALLTABS.P8
```

当运行卡带时，每个包含文件的内容被视为已粘贴到编辑器中该行的位置。

- 文件名相对于当前卡带（因此，需要先保存）
- 不会递归执行包含操作。
- 正常的字符计数和令牌限制适用。

当卡带以 .P8.PNG 保存或导出为二进制时，任何包含的文件都会被展平并保存与卡带一起，因此没有外部依赖。

#INCLUDE 可用于：

- 在卡带之间共享代码（库或通用多卡带代码）
- 使用外部代码编辑器而无需直接编辑 .p8 文件。
- 将卡带视为数据文件，加载 PICO-8 编辑工具进行修改。
- 加载和存储由外部（非 PICO-8）工具生成的数据。

▨ PICO-8 的一些特性

常见的陷阱要注意：

- 精灵表的下半部分和地图的下半部分占用相同的内存。 // 如果不确定如何工作，最好只使用其中一个。
- PICO-8 数字具有有限的精度和范围；数字之间的最小步长约为 0.00002 (0x0.0001)，范围为 -32768 (-0x8000) 到大约 32767.99999 (0x7fff.ffff) // 如果每帧向计数器加 1，大约 18 分钟后会溢出！
- Lua 数组默认是 1 基的，而不是 0 基的。FOREACH 从 TBL[1] 开始，而不是 TBL[0]。
- **COS**() 和 **SIN**() 接受 0..1 而不是 0..PI*2，且 **SIN**() 是反向的。
- **SGN**(0) 返回 1。

▨ CPU

尽管 PICO-8 没有明确定义的 CPU，但有一个虚拟 CPU 速度为 8MHz，其中每个 Lua VM 指令大约消耗 2 个周期。内置操作（如绘制精灵）也有 CPU 开销。这意味着在具有强大 CPU 的主机机器上创建的 PICO-8 卡带仍然可以在较慢的机器上很好地运行，并且不会在手机上或在 Web 上运行时消耗过多电池。

要查看运行中的卡带的 CPU 负载，按 CTRL-P 切换 CPU 计量器，或在每帧末尾打印 **STAT**(1)。

6 应用程序接口参考

PICO-8 基于 Lua 编程语言，但不包括 Lua 标准库。相反，提供了一个小型 API，符合 PICO-8 的极简设计和有限的屏幕空间。有关使用大多数 API 函数的示例程序，请参见 / DEMOS/API.P8

函数在此处写为：

FUNCTION_NAME(PARAMETER, [OPTIONAL_PARAMETER])

ⓘ 请注意，PICO-8 不区分大小写 -- 如果直接编辑 .p8 或 .lua 文件，函数名应全部为小写。

6.1 系统

通过命令行调用的系统函数可以省略通常的括号和字符串引号。例如，而不是 LOAD("BLAH.P8")，可以写为：

```
LOAD BLAH.P8
```

LOAD(FILENAME, [BREADCRUMB], [PARAM_STR])

加载或保存卡带

当从正在运行的卡带加载时，加载的卡带会立即运行，并带有参数字符串 PARAM_STR（可通过 STAT(6) 访问），并且会插入一个菜单项，名称为 BREADCRUMB，返回到之前的卡带。

以 '#' 开头的文件名被视为 BBS 卡带 ID，会立即下载并运行：

```
> LOAD("#MYGAME_LEVEL2", "BACK TO MAP", "LIVES="..LIVES)
```

如果 ID 是卡带的父帖子，或未指定修订号，则会获取最新版本。BBS 卡带可以从其他 BBS 卡带或本地卡带加载，但不能从导出的卡带加载。

SAVE(FILENAME)

加载或保存卡带

FOLDER

在主机操作系统中打开卡带文件夹。

LS([DIRECTORY])

列出给定目录（文件夹）中的 .p8 和 .p8.png 文件，相对于当前目录。目录项以斜杠结尾（例如 "foo/"）。

当从正在运行的卡带调用时，LS 只能本地使用，并返回结果表。当从 BBS 卡带调用时，LS 返回 nil。

目录只能解析到 PICO-8 的虚拟驱动器内部；从根目录调用 LS("..") 将解析到根目录。

RUN([PARAM_STR])

从程序的开头运行。

RUN() 可以在正在运行的程序中调用以重置。

当提供 PARAM_STR 时，可以在运行时通过 STAT(6) 访问它。

STOP([MESSAGE])

停止卡带并可选地打印消息。

RESUME

恢复程序。使用 R 作为快捷方式。

使用单个 "." 从命令行进入单帧模式。这会进入帧级模式，可以通过 stat(110) 读取。当帧级模式处于活动状态时，输入空命令（通过按回车键）会前进一帧。

ASSERT(CONDITION, [MESSAGE])

如果 CONDITION 为假，则停止程序并打印 MESSAGE（如果提供）。这对于调试卡带很有用，通过 ASSERT() 确保预期为真的事情确实是真实的。

```
ASSERT(ADDR >= 0 AND ADDR <= 0x7FFF, "OUT OF RANGE")
POKE(ADDR, 42) -- THE MEMORY ADDRESS IS OK, FOR SURE!
```

REBOOT

重启机器。适用于启动新项目

RESET()

将 RAM 中 0x5f00..0x5f7f 的值重置为其默认值。这包括调色板、相机位置、裁剪和填充模式。如果在命令提示符处迷路，因为绘图状态使得查看文本变得不可能，可以尝试输入 RESET! 也可以在正在运行的程序中调用。

INFO()

打印有关卡带的一些信息：代码大小、标记、压缩大小

还会显示：

UNSAVED CHANGES 当内存中的卡带与磁盘上的卡带不同时
EXTERNAL CHANGES 当磁盘上的卡带自加载以来已更改
（例如，通过使用单独的文本编辑器编辑程序）

FLIP()

翻转后缓冲区到屏幕并等待下一帧。当定义了 **_DRAW** 或 **_UPDATE** 回调时，不需要此调用，因为会自动进行翻转。但在使用自定义主循环时，通常需要调用 FLIP：

```
::_::
CLS()
FOR I=1,100 DO
  A=I/50 - T()
  X=64+COS(A)\*I
  Y=64+SIN(A)\*I
  CIRCFILL(X,Y,1,8+(I/4)%8)
END
FLIP()GOTO _
```

如果程序在帧结束前没有调用 FLIP，并且没有 **_DRAW** 回调正在进行中，则会将后缓冲区的当前内容复制到屏幕。

PRINTH(STR, [FILENAME], [OVERWRITE], [SAVE_TO_DESKTOP])

将字符串打印到主机操作系统的控制台进行调试。

如果设置了 filename，则将字符串追加到主机操作系统中的文件（默认为当前目录 -- 使用 FOLDER 查看）。

设置 OVERWRITE 为 true 会导致该文件被覆盖而不是追加。

设置 SAVE_TO_DESKTOP 为 true 会将内容保存到桌面而不是当前路径。

使用 "@clip" 作为文件名可以写入主机的剪贴板。

ⓘ 使用 stat(4) 读取剪贴板内容，但剪贴板内容仅在运行时按下 CTRL-V 后可用（出于安全考虑）。

TIME()

T()

返回自卡带运行以来的秒数。

这不是实际世界的时间，而是通过计算

_UPDATE 或 @_UPDATE60 被调用的次数来计算的。

STAT(X)

获取系统状态，其中 X 为：

```
0   内存使用情况 (0..2048)
1   自上次翻转以来使用的 CPU (1.0 == 100% CPU)
4   剪贴板内容（用户按下 CTRL-V 后）
6   参数字符串
7   当前帧率

46..49  当前在通道 0..3 上播放的 SFX 的索引
50..53  通道 0..3 上的音符编号 (0..31)
54      当前播放的模式索引
55      播放的模式总数
56      当前模式播放的节拍数
57      (Boolean) 当前是否正在播放音乐

80..85  UTC 时间：年、月、日、小时、分钟、秒
90..95  本地时间

100     当前面包屑标签，或 nil
110     当处于帧级模式时返回 true
```

ⓘ 音频值 16..26 是音频状态查询 46..56 的旧版本。它们仅报告音频混音器的当前状态，该状态每秒变化约 20 次（取决于主机声音驱动程序和其他因素）。46..56 而是存储了每个节拍的混音器状态，以提供当前可听状态的更高分辨率估计。

EXTCMD(CMD_STR, [P1, P2])

特殊系统命令，其中 CMD_STR 为字符串：

```
"pause"          请求打开暂停菜单
"reset"          请求卡带重置
"go_back"        如果存在则返回到之前的卡带
"label"          设置卡带标签
"screen"         保存屏幕截图
"rec"            设置视频开始点
"rec_frames"     设置视频开始点为帧模式
"video"          保存 .gif 到桌面
"audio_rec"      开始录制音频
"audio_end"      将录制的音频保存到桌面（Web 不支持）
"shutdown"       退出卡带（从导出的二进制文件）
"folder"         在主机操作系统中打开当前工作文件夹
"set_filename"   设置屏幕截图 / gif / 音频录制的文件名
"set_title"      设置主机窗口标题
```

某些命令具有可选的数字参数：

"video" 和 "screen": P1: 一个整数缩放因子，覆盖系统设置。P2: 当 > 0 时，保存到当前文件夹而不是桌面

"audio_end" P1: 当 > 0 时，保存到当前文件夹而不是桌面

▨ 录制 GIF

EXTCMD("REC"), EXTCMD("VIDEO") 与使用 ctrl-8, ctrl-9 相同，并使用当前的 GIF_SCALE 设置保存桌面中的 gif（使用 CONFIG GIF_SCALE 更改）。

可以使用两个附加参数来覆盖这些默认值：

```
EXTCMD("VIDEO", 4)    -- SCALE \*4 (512 X 512)
EXTCMD("VIDEO", 0, 1) -- 默认缩放，保存到用户数据文件夹
```

用户数据文件夹可以通过 EXTCMD("FOLDER") 打开，默认为与卡带相同的路径，或 {pico-8 appdata}/appdata/appname 对于导出的二进制文件。

由于 gif 格式的性质，所有 gif 都以 33.3fps 录制，并且 PICO-8 生成的帧会根据用户看到的内容进行跳帧或重复。要记录每次调用 **FLIP** 时恰好一帧，无论运行时帧率或生成帧所需的时间，使用：

```
EXTCMD("REC_FRAMES")
```

gif（以及截图、音频）的默认文件名是 foo_%d，其中 foo 是卡带的名称，%d 是从 0 开始的数字，自动递增，直到不存在同名文件。使用 EXTCMD("SET_FILENAME","FOO") 可以覆盖默认值。如果自定义文件名包含 "%d"，则使用自动递增数字行为，否则即使存在同名文件也会写入文件。

6.2 图形

PICO-8 拥有 128 个 8x8 像素的精灵固定容量，再加上另外 128 个与地图数据下半部分重叠的精灵（“共享数据”）。这 256 个精灵统称为精灵表，可以看作是一个 128x128 像素的图像。

PICO-8 的所有绘图操作都受当前绘图状态的影响。绘图状态包括相机位置（用于向所有坐标添加偏移）、调色板映射（用于重新着色精灵）、裁剪矩形、绘图颜色和填充模式。

每次程序运行时，绘图状态都会重置，或者通过调用 **RESET**() 来重置。

颜色索引：

```
  0  黑色    1  深蓝色    2  深紫色    3  深绿色
  4  棕色    5  深灰色    6  浅灰色    7  白色
  8  红色    9  橙色     10  黄色     11  绿色
 12  蓝色   13  靛蓝色    14  粉色     15  桃色
```

CLIP(X, Y, W, H, [CLIP_PREVIOUS])

设置像素单位的裁剪矩形。所有绘图操作都会被裁剪到 x, y 处宽度为 w、高度为 h 的矩形。

CLIP() 重置裁剪区域。

当 CLIP_PREVIOUS 为真时，新的裁剪区域会被旧的裁剪区域裁剪。

PSET(X, Y, [COL])

将 x, y 处的像素设置为颜色索引 COL (0..15)。

当 COL 未指定时，使用当前绘图颜色。

```
FOR Y=0,127 DO
  FOR X=0,127 DO
    PSET(X, Y, X\*Y/8)
  END
END
```

PGET(X, Y)

返回屏幕上 (X, Y) 处像素的颜色。

```
WHILE (TRUE) DO
  X, Y = RND(128), RND(128)
  DX, DY = RND(4)-2, RND(4)-2
  PSET(X, Y, PGET(DX+X, DY+Y))
END
```

当 X 和 Y 超出范围时，PGET 返回 0。可以通过以下方式指定自定义返回值：

```
POKE(0x5f36, 0x10)
POKE(0x5f5B, NEWVAL)
```

SGET(X, Y)

SSET(X, Y, [COL])

获取或设置精灵表像素的颜色 (COL)。

当 X 和 Y 超出范围时，SGET 返回 0。可以通过以下方式指定自定义值：

```
POKE(0x5f36, 0x10)
POKE(0x5f59, NEWVAL)
```

FGET(N, [F])

FSET(N, [F], VAL)

获取或设置精灵 N 的标志 F 的值 (VAL)。

F 是标志索引 0..7。

VAL 是 TRUE 或 FALSE。

标志 0..7 的初始状态可以在精灵编辑器中设置，因此可以用来创建自定义精灵属性。还可以通过在 **MAP**() 中提供掩码来绘制地图瓦片的子集。

当 F 未指定时，所有标志将作为一个位字段被获取/设置。

```
FSET(2, 1 | 2 | 8)   -- 设置位 0,1 和 3
FSET(2, 4, TRUE)     -- 设置位 4
PRINT(FGET(2))       -- 27 (1 | 2 | 8 | 16)
```

PRINT(STR, X, Y, [COL])

PRINT(STR, [COL])

打印字符串 STR 并可选地设置绘图颜色为 COL。

快捷方式：单行书写时，可以使用 ? 调用 print 而无需括号：

```
?"HI"
```

当 X, Y 未指定时，会自动追加换行符。可以通过在字符串末尾添加显式的终止控制字符来省略换行：

```
?"THE QUICK BROWN FOX\\0"
```

此外，当 X, Y 未指定时，打印的文本低于 122 时会导致控制台滚动。可以通过以下方式在运行时禁用：

```
POKE(0x5f36,0x40)
```

PRINT 返回打印时最右边的 x 位置。可以通过打印到屏幕外的方式找到某些文本的宽度：

```
W = PRINT("HOGE", 0, -20) -- 返回 16
```

有关控制代码和自定义字体的信息，请参阅 **附录 A** (P8SCII)。

CURSOR(X, Y, [COL])

设置光标位置。

如果指定了 COL，还将设置当前颜色。

COLOR([COL])

设置绘图函数使用的当前颜色。

如果未指定 COL，则当前颜色设置为 6

CLS([COL])

清除屏幕并重置裁剪矩形。

COL 默认为 0 (黑色)

CAMERA([X, Y])

为所有绘图操作设置 -x, -y 的屏幕偏移

CAMERA() 重置偏移

CIRC(X, Y, R, [COL])

CIRCFILL(X, Y, R, [COL])

在 x,y 处绘制一个半径为 r 的圆或填充圆

如果 r 为负数，则不绘制圆。

当 COL 的位 0x1800.0000 被设置，并且 **0x5F34** & 2 == 2 时，圆将被反转绘制。

OVAL(X0, Y0, X1, Y1, [COL])

OVALFILL(X0, Y0, X1, Y1, [COL])

绘制一个关于 x 和 y 对称（椭圆）的椭圆，给定边界矩形。

LINE(X0, Y0, [X1, Y1, [COL]])

从 (X0, Y0) 绘制一条线到 (X1, Y1)

如果未给出 (X1, Y1)，则使用上次绘制线的终点。

LINE() 不带参数意味着下一次调用 LINE(X1, Y1) 将仅设置终点而不会绘制。

```
CLS()
LINE()
FOR I=0,6 DO
  LINE(64+COS(I/6)\*20, 64+SIN(I/6)\*20, 8+I)
END
```

RECT(X0, Y0, X1, Y1, [COL])

RECTFILL(X0, Y0, X1, Y1, [COL])

绘制一个矩形或填充矩形，角点位于 (X0, Y0), (X1, Y1)。

PAL(C0, C1, [P])

PAL() 交换颜色 c0 和 c1 对于三个调色板重映射中的一个（p 默认为 0）：

0: 绘图调色板

绘图调色板在绘制时重映射颜色。例如，可以通过将第 9 个调色板值设置为 8 来将一个橙色花精灵绘制为红色：

```
PAL(9,8)      -- 将后续橙色（颜色 9）像素绘制为红色（颜色 8）
SPR(1,70,60) -- 该精灵中的任何橙色像素都将用红色绘制，而不是橙色
```

更改绘图调色板不会影响已经绘制到屏幕上的任何内容。

1: 显示调色板

显示调色板在每帧结束时显示整个屏幕时重映射。例如，如果启动 PICO-8 并输入 PAL(6,14,1)，可以看到所有灰色（颜色 6）文本立即变为粉色（颜色 14），即使它已经绘制完成。这对于屏幕范围的效果（如淡入淡出）非常有用。

2: 辅助调色板

由 **FILLP**() 绘制精灵时使用。这提供了从单个 4 位颜色索引到两个 4 位颜色索引的映射。

PAL() 重置所有调色板到系统默认值（包括透明度值）
PAL(P) 重置特定调色板（0..2）到系统默认值

PAL(TBL, [P])

当 pal 的第一个参数为表时，颜色将为每个条目分配。例如，要将颜色 12 和 14 重新映射为红色：

```
PAL({\[12\]=9, \[14\]=8})
```

或者将整个屏幕重新着色为灰色阴影（包括已经绘制的所有内容）：

```
PAL({1,1,5,5,5,6,7,13,6,7,7,6,13,6,7,1}, 1)
```

因为表索引从 1 开始，所以颜色 0 在这种情况下在末尾给出。

PALT(C, [T])

设置颜色索引的透明度为 T (布尔值) 透明度被 **SPR**(), **SSPR**(), **MAP**() 和 **TLINE**() 观察

PALT(8, TRUE) -- 在后续的精灵/TLINE 绘制调用中不会绘制红色像素

PALT() 重置为默认：所有颜色不透明，除了颜色 0

当 C 为唯一参数时，它被视为用于设置所有 16 个值的位字段。例如：要将颜色 0 和 1 设置为透明：

```
PALT(0B1100000000000000)
```

SPR(N, X, Y, [W, H], [FLIP_X], [FLIP_Y])

在位置 X,Y 绘制精灵 N (0..255)

W (宽度) 和 H (高度) 默认为 1, 1 并指定要绘制的精灵数量。

默认情况下颜色 0 透明（参见 **PALT**()）

当 FLIP_X 为真时，水平翻转。

当 FLIP_Y 为真时，垂直翻转。

SSPR(SX, SY, SW, SH, DX, DY, [DW, DH], [FLIP_X], [FLIP_Y])

将精灵表中的一个矩形区域 (sx, sy, sw, sh) 拉伸到屏幕上的一个目标矩形 (dx, dy, dw, dh)。在这两种情况下，x 和 y 值是矩形左上角的坐标，宽度为 w，高度为 h。

默认情况下颜色 0 透明（参见 **PALT**()）

dw, dh 默认为 sw, sh

当 FLIP_X 为真时，水平翻转。

当 FLIP_Y 为真时，垂直翻转。

FILLP(P)

PICO-8 填充模式是一个 4x4 的 2 色平铺模式，被 **CIRC**() **CIRCFILL**() **RECT**() **RECTFILL**() **OVAL**() **OVALFILL**() **PSET**() **LINE**() 观察

P 是一个从最高位开始按顺序读取的位字段。要计算所需的模式的 P 值，将所需的位值相加：

```
.-----------------------.
|32768|16384| 8192| 4096|
|-----|-----|-----|-----|
| 2048| 1024| 512 | 256 |
|-----|-----|-----|-----|
| 128 |  64 |  32 |  16 |
|-----|-----|-----|-----|
|  8  |  4  |  2  |  1  |
'-----------------------'
```

例如，FILLP(4+8+64+128+ 256+512+4096+8192) 将创建一个棋盘图案。

这可以用二进制更简洁地表示：FILLP(0b0011001111001100)。

默认填充模式为 0，这意味着绘制单色。

要指定模式的第二种颜色，使用任何颜色参数的高位：

```
FILLP(0b0011010101101000)
CIRCFILL(64,64,20, 0x4E) -- 棕色和粉色
```

其他设置在位 0b0.111 中给出：

0b0.100 透明

当此位被设置时，第二种颜色不会被绘制

```
-- 棋盘带透明方块
FILLP(0b0011001111001100.1)
```

0b0.010 应用于精灵

当设置时，填充模式应用于精灵（spr, sspr, map, tline），使用辅助调色板提供的颜色映射。

精灵中的每个像素值（在应用绘图调色板后）被视为辅助调色板的索引。辅助调色板中的每个条目包含用于渲染填充模式的两种颜色。例如，要为精灵中的蓝色像素（颜色 12）绘制白色和红色（7 和 8）的棋盘图案：

```
FOR I=0,15 DO PAL(I, I+I\*16, 2) END  --  其他所有颜色映射到自身
  PAL(12, 0x87, 2)                     --  在辅助调色板中重新映射颜色 12

  FILLP(0b0011001111001100.01)          --  棋盘调色板，应用于精灵
SPR(1, 64,64)                          --  绘制精灵
```

0b0.001 全局应用辅助调色板

当设置时，辅助调色板映射也被所有尊重填充模式的绘图函数（circfill, line 等）应用。这在与精灵绘制函数结合使用时很有用，因为每个精灵像素的颜色索引与提供给绘图函数的颜色索引含义相同。

```
FILLP(0b0011001111001100.001)
PAL(12, 0x87, 2)
CIRCFILL(64,64,20,12)                  -- 红色和白色棋盘圆
```

辅助调色板映射在常规绘图调色板映射之后应用。因此，以下也会绘制一个红色和白色棋盘圆：

```
PAL(3,12)
CIRCFILL(64,64,20,3)
```

填充模式也可以通过在任何颜色参数中设置位来设置（例如，提供给 **COLOR**() 的参数，或提供给 **LINE**(), **RECT**() 等的最后一个参数）。

```
POKE(0x5F34, 0x3) -- 0x1 启用高位中的填充模式  0x2 启用反转模式
CIRCFILL(64,64,20, 0x114E.ABCD) -- 设置填充模式为 ABCD
```

当使用颜色参数来设置填充模式时，使用以下位：

位 0x1000.0000 需要设置：这意味着“观察位 0xf00.ffff”
位 0x0100.0000 透明
位 0x0200.0000 应用于精灵
位 0x0400.0000 全局应用辅助调色板
位 0x0800.0000 反转绘制操作（circfill/ovalfill/rectfill）
位 0x00FF.0000 是常规颜色位
位 0x0000.FFFF 被解释为填充模式

6.3 表函数

⚠ 除了 PAIRS() 之外，以下函数和 # 运算符仅适用于从索引 1 开始且不包含 NIL 条目的表。所有其他形式的表可以被视为哈希映射或集合，而不是具有长度的数组。

ADD(TBL, VAL, [INDEX])

将值 VAL 添加到表 TBL 的末尾。等效于：

```
TBL[#TBL + 1] = VAL
```

如果提供了索引，则元素将插入到该位置：

```
FOO={}         -- 创建空表
ADD(FOO, 11)
ADD(FOO, 22)
PRINT(FOO[2]) -- 22
```

DEL(TBL, VAL)

删除表 TBL 中第一个值为 VAL 的实例。剩余的条目将左移一个索引以避免空洞。

请注意，VAL 是要删除的项的值，而不是表中的索引。（要删除特定索引处的项，请使用 DELI）。DEL 返回被删除的项，如果没有删除任何内容，则不返回任何值。

```
A={1,10,2,11,3,12}
FOR ITEM IN ALL(A) DO
  IF (ITEM < 10) THEN DEL(A, ITEM) END
END
FOREACH(A, PRINT) -- 10,11,12
PRINT(A[3])       -- 12
```

DELI(TBL, [I])

类似于 **DEL**，但从表 TBL 中删除索引 I 处的项。当未提供 I 时，将删除并返回表的最后一个元素。

COUNT(TBL, [VAL])

返回表 t 的长度（与 #TBL 相同）。如果提供了 VAL，则返回该表中 VAL 的实例数量。

ALL(TBL)

用于 FOR 循环以按添加顺序迭代表中的所有项（具有 1 基整数索引）。

```
T = {11,12,13}
ADD(T,14)
ADD(T,"HI")
FOR V IN ALL(T) DO PRINT(V) END -- 11 12 13 14 HI
PRINT(#T) -- 5
```

FOREACH(TBL, FUNC)

对于表 TBL 中的每一项，使用该项作为单个参数调用函数 FUNC。

```
> FOREACH({1,2,3}, PRINT)
```

PAIRS(TBL)

用于 FOR 循环以迭代表 TBL，提供每个项的键和值。与 **ALL**() 不同，PAIRS() 会迭代表中的每个项，而不考虑索引方案。顺序不保证。

```
T = {["HELLO"]=3, [10]="BLAH"}
T.BLUE = 5;
FOR K,V IN PAIRS(T) DO
  PRINT("K: "..K.."  V:"..V)
END
```

输出:

```
K: 10  v:BLAH
K: HELLO  v:3
K: BLUE  v:5
```

6.4 输入

BTN([B], [PL])

获取玩家 PL 的按钮 B 状态（默认 0）

B: 0..5: 左 右 上 下 按钮_o 按钮_x
PL: 玩家索引 0..7

除了使用数字表示 B，还可以使用按钮符号。（在编码编辑器中，使用 Shift-L R U D O X）

如果不提供任何参数，则返回玩家 0 和 1 的所有 12 个按钮状态的位字段 // P0: 位 0..5 P1: 位 8..13

默认键盘映射到玩家按钮：

```
玩家 0: [DPAD]: 方向键, [O]: Z C N   [X]: X V M
玩家 1: [DPAD]: SFED,  [O]: LSHIFT  [X]: TAB W  Q A
```

⚠ 尽管 PICO-8 接受所有按钮组合，但请注意，在物理游戏控制器上同时按下 LEFT 和 RIGHT 通常是不可能的。在某些控制器上，UP + LEFT/RIGHT 也很难按下，如果可以使用 [X] 或 [O] 代替 UP（例如跳跃 / 加速）。

BTNP(B, [PL])

BTNP 是 "Button Pressed" 的缩写；与按钮被按住时返回 true 不同，BTNP 在按钮被按下且上一帧未按下时返回 true。它还会在 15 帧后重复，之后每 4 帧返回一次（在 30fps 下 -- 在 60fps 下是两倍）。这可以用于菜单导航或网格状玩家移动等场景。

BTNP 读取的状态在每次调用 **_UPDATE** 或 **_UPDATE60** 开始时重置，因此最好在这些函数内部使用 BTNP。

可以通过写入以下内存地址来设置自定义延迟（以帧数 30fps 计算）：

```
POKE(0X5F5C, DELAY) -- 设置重复前的初始延迟。255 表示永不重复。
POKE(0X5F5D, DELAY) -- 设置重复延迟。
```

在这两种情况下，0 可用于默认行为（延迟 15 和 4）

6.5 音频

SFX(N, [CHANNEL], [OFFSET], [LENGTH])

播放音效 N (0..63) 在通道 CHANNEL (0..3) 从音符 OFFSET (0..31 的音符) 播放 LENGTH 个音符。

使用负的 CHANNEL 值具有特殊含义：

CHANNEL -1: （默认）自动选择一个未使用的通道
CHANNEL -2: 停止给定音效在任何通道上的播放

N 可以是给定 CHANNEL 的命令（或当 CHANNEL < 0 时对所有通道）：

N -1: 停止该通道上的声音
N -2: 释放该通道上的音效循环

```
SFX(3)    --  播放音效 3
SFX(3,2)  --  在通道 2 上播放音效 3
SFX(3,-2) --  停止音效 3 在任何通道上的播放
SFX(-1,2) --  停止通道 2 上播放的任何内容
SFX(-2,2) --  释放通道 2 上的循环
SFX(-1)   --  停止所有通道上的所有声音
SFX(-2)   --  释放所有通道上的循环
```

MUSIC(N, [FADE_LEN], [CHANNEL_MASK])

从模式 N (0..63) 开始播放音乐
N -1 停止音乐

FADE_LEN 以毫秒为单位（默认：0）。因此，要在 1 秒内淡入模式 0：

```
MUSIC(0, 1000)
```

CHANNEL_MASK 指定仅用于音乐的通道。例如，仅在通道 0..2 上播放：

```
MUSIC(0, NIL, 7) -- 1 | 2 | 4
```

保留的通道仍然可以用于播放音效，但仅当通过 **SFX**() 显式请求该通道索引时。

6.6 地图

PICO-8 地图是一个 128x32 的 8 位值网格，或在使用共享内存时为 128x64。在使用地图编辑器时，每个值被视为精灵表中的一个索引（0..255）。但是，它也可以用作一般的数据块。

MGET(X, Y)

MSET(X, Y, VAL)

获取或设置 X,Y 处的地图值 (VAL)

当 X 和 Y 超出范围时，MGET 返回 0，或者可以通过以下方式指定自定义返回值：

```
POKE(0x5f36, 0x10)
POKE(0x5f5a, NEWVAL)
```

MAP(TILE_X, TILE_Y, [SX, SY], [TILE_W, TILE_H], [LAYERS])

从 TILE_X, TILE_Y 开始绘制地图的一部分，在屏幕位置 SX, SY (像素) 处。

要从地图的 0,0 开始绘制 4x2 块的瓦片到屏幕的 20,20 处：

```
MAP(0, 0, 20, 20, 4, 2)
```

TILE_W 和 TILE_H 默认为整个地图（包括适用的共享空间）。

MAP() 通常与 CAMERA() 结合使用。要使玩家对象（以像素为单位位于 PL.X, PL.Y）居中：

```
CAMERA(PL.X - 64, PL.Y - 64)
MAP()
```

LAYERS 是一个位字段。当提供时，只有具有匹配精灵标志的精灵会被绘制。例如，当 LAYERS 为 0x5 时，只有具有标志 0 和 2 的精灵会被绘制。

精灵 0 被视为“空”且不被绘制。要禁用此行为，使用：POKE(0x5F36, 0x8)

TLINE(X0, Y0, X1, Y1, MX, MY, [MDX, MDY], [LAYERS])

从 (X0,Y0) 到 (X1,Y1) 绘制一条纹理线，从地图采样颜色值。当指定 LAYERS 时，只有具有匹配标志的精灵会被绘制（类似于 MAP()）

MX, MY 是地图坐标，以瓦片为单位进行采样。颜色值从每个地图瓦片上的 8x8 精灵中采样。例如：

2.0, 1.0 表示地图上位置 2,1 处的精灵的左上角
2.5, 1.5 表示相同精灵的像素 (4,4)

MDX, MDY 是在绘制每个像素后添加到 mx, my 的增量。（默认为 0.125, 0）

地图坐标 (MX, MY) 通过从地址 0x5F38 和 0x5F39 的值减去 0x0.0001 进行掩码。简单来说，这意味着可以通过 pocking 想要循环的宽度和高度来循环地图的一部分，只要它们是 2 的幂（2,4,8,16..）

例如，要每 8 个瓦片水平循环，每 4 个瓦片垂直循环：

```
POKE(0x5F38, 8)
POKE(0x5F39, 4)
TLINE(...)
```

默认值 (0,0) 提供一个掩码 0xff.ffff，这意味着样本每 256 个瓦片循环一次。

也可以在地址 0x5f3a, 0x5f3b 指定一个采样偏移（以瓦片为单位）：

```
POKE(0x5F3A, OFFSET_X)
POKE(0x5F3B, OFFSET_Y)
```

精灵 0 被视为“空”且不被绘制。要禁用此行为，使用：POKE(0x5F36, 0x8)

■ 设置 TLINE 精度

默认情况下，tline 坐标 (mx,my,mdx,mdy) 以瓦片为单位表示。这意味着 1 个像素是 0.125，只有 13 位用于小数部分。如果需要更高的精度，可以通过调用 TLINE 并传递一个参数来调整坐标空间，以允许更多位用于小数部分。这对于纹理墙等场景很有用，因为 mdx,mdy 的累积误差在近距离查看时可能会变得明显。

每个像素的小数部分使用的位数存储在一个特殊寄存器中，可以通过调用 TLINE 并传递一个参数来调整：

TLINE(16) -- MX,MY,MDX,MDY 以像素为单位表示

6.7 内存

PICO-8 有 3 种类型的内存：

1. 基础 RAM (64k): 请参见下方布局。使用 PEEK() POKE() MEMCPY() MEMSET() 访问
2. 卡带 ROM (32k): 与基础 RAM 布局相同，直到 0x4300
3. Lua RAM (2MB): 编译后的程序 + 变量

ⓘ 技术说明: 在使用编辑器时，正在修改的数据位于卡带 ROM 中，但 API 函数如 **SPR**() 和 **SFX**() 仅操作基础 RAM。PICO-8 在以下 3 种情况下自动将卡带 ROM 复制到基础 RAM（即调用 **RELOAD**()）:
1. 当加载一个卡带时
2. 当运行一个卡带时
3. 当退出任何编辑模式时 // 可以通过 poke(0x5f37,1) 关闭

[

■ 基础 RAM 内存布局

](https://www.lexaloffle.com/dl/docs/pico-8_manual.html#Base_RAM_Memory_Layout)

```
0X0    GFX
0X1000 GFX2/MAP2 (共享)
0X2000 MAP
0X3000 GFX 标志
0X3100 SONG
0X3200 SFX
0X4300 用户数据
0X5600 自定义字体（如果有定义）
0X5E00 持久化卡带数据（256 字节）
0X5F00 绘制状态
0X5F40 硬件状态
0X5F80 GPIO 引脚（128 字节）
0X6000 屏幕（8K）
0x8000 用户数据
```

用户数据没有特定含义，可以通过 **MEMCPY**(), **PEEK**() & **POKE**() 使用。持久化卡带数据映射到 0x5e00..0x5eff，但仅在调用 **CARTDATA**() 后存储。颜色格式（gfx/screen）是每字节 2 像素：低位编码每对像素的左像素。地图格式是一个字节每个瓦片，其中每个字节通常编码一个精灵索引。

■ 重新映射图形和地图数据

GFX, MAP 和 SCREEN 内存区域可以通过在以下地址设置值进行重新映射:

```
0X5F54 GFX:    可以是 0x00（默认）或 0x60（使用屏幕内存作为精灵表）
0X5F55 SCREEN: 可以是 0x60（默认）或 0x00（使用精灵表作为屏幕内存）
0X5F56 MAP:    可以是 0x20（默认）或 0x10..0x2f，或 0x80 及以上。
0X5F57 MAP SIZE: 地图宽度。0 表示 256。默认为 128。
```

地址可以以 256 字节的增量表示。因此，0x20 表示 0x2000，0x21 表示 0x2100 等。地图地址 0x30..0x3f 被视为 0x10..0x1f（共享内存区域）。地图数据只能包含在内存区域 0x1000..0x2fff, 0x8000..0xffff 内，地图高度由给定区域中可以容纳的最大尺寸决定。

GFX 和 SCREEN 地址还可以映射到更高的内存位置 0x80, 0xA0, 0xC0, 0xE0，但约束条件是 MAP 不能与这些地址重叠（在这种情况下，冲突的 GFX 和/或 SCREEN 映射将恢复到默认映射）。

ⓘ GFX 和 SCREEN 内存映射发生在低级别，也影响内存访问函数（peek, poke, memcpy）。从 0x0 和 0x6000 开始的 8k 内存块可以被视为指向单独视频 RAM 的指针，设置 0X5F54 和 0X5F56 的值会改变这些指针。

PEEK(ADDR, [N])

从基础 RAM 的地址读取一个字节。如果指定了 N，则 PEEK() 返回指定数量的结果（最大: 8192）。例如，要读取视频内存的前 2 个字节：

```
A, B = PEEK(0x6000, 2)
```

POKE(ADDR, VAL1, VAL2, ...)

将一个或多个字节写入基础 RAM 的地址。如果提供了多个参数，则它们将按顺序写入（最大: 8192）。

PEEK2(ADDR)

POKE2(ADDR, VAL)

PEEK4(ADDR)

POKE4(ADDR, VAL)

16 位和 32 位版本的 PEEK 和 POKE。读取和写入一个数字（VAL），采用小端格式：

```
16 位: 0xffff.0000
32 位: 0xffff.ffff
```

ADDR 不需要对齐到 2 或 4 字节边界。

还可以使用以下运算符进行 peek（但不能 poke），并且速度稍快：

```
@ADDR  -- PEEK(ADDR)
%ADDR  -- PEEK2(ADDR)
$ADDR  -- PEEK4(ADDR)
```

MEMCPY(DEST_ADDR, SOURCE_ADDR, LEN)

从源复制 LEN 字节的基础 RAM 到目标。段可以重叠

RELOAD(DEST_ADDR, SOURCE_ADDR LEN, [FILENAME])

与 MEMCPY 相同，但从卡带 ROM 复制。

代码段（>= 0x4300）受到保护，无法读取。

如果指定了文件名，则从单独的卡带加载数据。在这种情况下，卡带必须是本地的（BBS 卡带不能以这种方式读取）。

CSTORE(DEST_ADDR, SOURCE_ADDR, LEN, [FILENAME])

与 memcpy 相同，但从基础 RAM 复制到卡带 ROM。

CSTORE() 等同于 CSTORE(0, 0, 0x4300)

代码段（>= 0x4300）受到保护，无法写入。

如果指定了 FILENAME，则数据直接写入磁盘上的该卡带。一次会话最多可以写入 64 个卡带。有关更多信息，请参见 **卡带数据**。

MEMSET(DEST_ADDR, VAL, LEN)

将 8 位值 VAL 写入从 DEST_ADDR 开始的内存，长度为 LEN 字节。

例如，要将视频内存的一半填充为 0xC8：

```
\> MEMSET(0x6000, 0xC8, 0x1000)
```

6.8 数学

MAX(X, Y)

MIN(X, Y)

MID(X, Y, Z)

返回参数的最大值、最小值或中间值

```
> ?MID(7,5,10) -- 7
```

FLR(X)

```
> ?FLR ( 4.1) -->  4
> ?FLR (-2.3) --> -3
```

CEIL(X)

返回大于或等于 x 的最接近的整数

```
> ?CEIL( 4.1) -->  5
> ?CEIL(-2.3) --> -2
```

COS(X)

SIN(X)

返回 x 的余弦或正弦值，其中 1.0 表示一个完整的旋转。例如，要制作一个每秒旋转一圈的指针：

```
FUNCTION _DRAW()
  CLS()
  CIRC(64, 64, 20, 7)
  X = 64 + COS(T()) \* 20
  Y = 64 + SIN(T()) \* 20
  LINE(64, 64, X, Y)
END
```

PICO-8 的 SIN() 返回一个反转的结果以适应屏幕空间（其中 Y 表示“向下”，而数学图中 Y 通常表示“向上”）。

```
> SIN(0.25) -- 返回 -1
```

为了获得基于弧度的常规三角函数（不带 Y 轴反转），请将以下代码片段粘贴到程序的开头附近：

```
P8COS = COS FUNCTION COS(ANGLE) RETURN P8COS(ANGLE/(3.1415\*2)) END
P8SIN = SIN FUNCTION SIN(ANGLE) RETURN -P8SIN(ANGLE/(3.1415\*2)) END
```

ATAN2(DX, DY)

将 DX, DY 转换为 0..1 之间的角度

与 cos/sin 类似，角度在屏幕空间中逆时针运行。例如：

```
> ?ATAN(0, -1) -- 返回 0.25
```

ATAN2 可以用于查找两个点之间的方向：

```
X=20 Y=30
FUNCTION _UPDATE()
  IF (BTN(0)) X-=2
  IF (BTN(1)) X+=2
  IF (BTN(2)) Y-=2
  IF (BTN(3)) Y+=2
END

  FUNCTION _DRAW()
  CLS()
  CIRCFILL(X,Y,2,14)
  CIRCFILL(64,64,2,7)

      A=ATAN2(X-64, Y-64)
  PRINT("角度: "..A)
  LINE(64,64,
    64+COS(A)\*10,
    64+SIN(A)\*10,7)
END
```

SQRT(X)

返回 x 的平方根

ABS(X)

返回 x 的绝对（正值）值

RND(X)

返回一个随机数 n，其中 0 <= n < x

如果需要整数，使用 flr(rnd(x))。如果 x 是一个数组样式的表，返回一个介于 table[1] 和 table[#table] 之间的随机元素。

SRAND(X)

设置随机数种子。种子在卡带启动时会自动随机化。

```
FUNCTION _DRAW()
  CLS()
  SRAND(33)
  FOR I=1,100 DO
    PSET(RND(128),RND(128),7)
  END
END
```

■ 位运算

位运算类似于逻辑表达式，不同之处在于它们在位级别上工作。

假设你有两个数字（使用二进制前缀 "0b" 写出）：

```
X = 0b1010
Y = 0b0110
```

一个按位 AND 操作会在 X 和 Y 的相应位都设置时给出结果

```
> PRINT(BAND(X,Y)) -- 结果:0B0010 (2 以十进制表示)
```

PICO-8 中有 9 个按位函数可用：

```
BAND(X, Y) -- 两个位都设置
BOR(X, Y)  -- 任一位设置
BXOR(X, Y) -- 任一位设置，但不是两者都设置
BNOT(X)    -- 每个位未设置
SHL(X, N)  -- 左移 N 位（右侧填零）
SHR(X, N)  -- 算术右移（左侧的位状态被复制）
LSHR(X, N) -- 逻辑右移（左侧填零）
ROTL(X, N) -- 将 X 中的所有位向左旋转 N 位
ROTR(X, N) -- 将 X 中的所有位向右旋转 N 位
```

还提供了运算符版本：& | ^^ ~ << >> >>> <<> >><

例如：PRINT(67 & 63) -- 结果:3 等价于 BAND(67,63)

运算符比它们对应的函数稍快。它们的行为完全相同，除了如果任何操作数不是数字，结果将是一个运行时错误（函数版本则默认为 0）。

■ 整数除法

整数除法可以使用 \

```
> PRINT(9\\2) -- 结果:4  等价于 FLR(9/2)
```

6.9 自定义菜单项

MENUITEM(INDEX, [LABEL], [CALLBACK])

添加或更新暂停菜单中的一个项目。

INDEX 应该是 1..5 并决定每个菜单项显示的顺序。

LABEL 应该是一个最多 16 个字符的字符串

CALLBACK 是用户选择该项时调用的函数。如果回调返回 true，暂停菜单将保持打开状态。

当没有标签或函数被提供时，菜单项将被移除。

```
MENUITEM(1, "RESTART PUZZLE",
  FUNCTION() RESET\_PUZZLE() SFX(10) END
)
```

回调接收一个参数，该参数是 L,R,X 按钮按下的位域。

```
MENUITEM(1, "FOO",
  FUNCTION(B) IF (B&1 > 0) THEN PRINTH("LEFT WAS PRESSED") END END
)
```

为了过滤可以触发回调的按钮按压，可以在 INDEX 的位 0xff00 中提供一个掩码。例如，要为特定菜单项禁用 L, R，可以在索引中设置位 0x300：

```
MENUITEM(2 | 0x300, "RESET PROGRESS",
  FUNCTION() DSET(0,0) END
)
```

菜单项可以在回调内更新、添加或移除：

```
MENUITEM(3, "SCREENSHAKE: OFF",
  FUNCTION()
    SCREENSHAKE = NOT SCREENSHAKE
    MENUITEM(NIL, "SCREENSHAKE: "..(SCREENSHAKE AND "ON" OR "OFF"))
    RETURN TRUE -- 不要关闭
  END
)
```

6.10 字符串和类型转换

Lua 中的字符串可以用单引号或双引号或匹配的 [[ ]] 方括号表示：

```
S = "THE QUICK"
S = 'BROWN FOX';
S = \[\[
  JUMPS OVER
  MULTIPLE LINES
\]\]
```

可以使用 # 运算符获取字符串的长度（字符数）：

```
>PRINT(#S)
```

可以使用 .. 运算符连接字符串。连接数字会将其转换为字符串。

```
>PRINT("THREE "..4) --> "THREE 4"
```

当作为算术表达式的一部分使用时，字符串值会被转换为数字：

```
>PRINT(2+"3")   --> 5
```

TOSTR(VAL, [FORMAT_FLAGS])

将 VAL 转换为字符串。

FORMAT_FLAGS 是一个位域：

```
  0x1: 以原始十六进制值写入数字、函数或表。
  0x2: 通过将其左移 16 位，将 VAL 写作一个有符号的 32 位整数。
```

TOSTR(NIL) 返回 "[nil]"

TOSTR() 返回 ""

```
TOSTR(17)       -- "17"
TOSTR(17,0x1)   -- "0x0011.0000"
TOSTR(17,0x3)   -- "0x00110000"
TOSTR(17,0x2)   -- "1114112"
```

TONUM(VAL, [FORMAT_FLAGS])

将 VAL 转换为数字。

```
TONUM("17.5")  -- 17.5
TONUM(17.5)    -- 17.5
TONUM("HOGE")  -- 没有返回值
```

FORMAT_FLAGS 是一个位域：

```
0x1: 将字符串作为十六进制（无符号、整数）读取，不带 "0x" 前缀
   非十六进制字符被视为 '0'。
0x2: 将字符串作为有符号的 32 位整数读取，并右移 16 位。
0x4: 当 VAL 无法转换为数字时，返回 0
```

```
TONUM("FF",       0x1)  -- 255
TONUM("1114112",  0x2)  -- 17
TONUM("1234abcd", 0x3)  -- 0x1234.abcd
```

CHR(VAL0, VAL1, ...)

将一个或多个字符代码转换为字符串。

```
CHR(64)                    -- "@"
CHR(104,101,108,108,111)   -- "hello"
```

ORD(STR, [INDEX], [NUM_RESULTS])

将字符串 STR 中的一个或多个字符转换为其序数值（0..255）。

使用 INDEX 参数指定要使用的字符串中的字符。当 INDEX 超出范围或 str 不是字符串时，ORD 返回 nil。

当提供了 NUM_RESULTS 时，ORD 从 INDEX 开始返回多个值。

```
ORD("@")         -- 64
ORD("123",2)     -- 50 (字符串中的第二个字符: "2")
ORD("123",2,3)   -- 50,51,52
```

SUB(STR, POS0, [POS1])

从字符串 str 中提取子字符串，从 pos0 开始到 pos1 结束。当未指定 POS1 时，返回从 pos0 开始的剩余字符串。当指定了 POS1，但不是数字时，返回 pos0 处的单个字符。

```
S = "THE QUICK BROWN FOX"
PRINT(SUB(S,5,9))    --> "QUICK"
PRINT(SUB(S,5))      --> "QUICK BROWN FOX"
PRINT(SUB(S,5,TRUE)) --> "Q"
```

SPLIT(STR, [SEPARATOR], [CONVERT_NUMBERS])

根据给定的分隔符（默认为 ","）将字符串拆分为元素表。当分隔符是一个数字 n 时，字符串将被拆分为 n 个字符的组。当 convert_numbers 为真时，数值标记将被存储为数字（默认为真）。空元素将被存储为空字符串。

```
SPLIT("1,2,3")               -- {1,2,3}
SPLIT("ONE:TWO:3",":",FALSE) -- {"ONE","TWO","3"}
SPLIT("1,,2,")               -- {1,"",2,""}
```

TYPE(VAL)

返回 val 的类型作为字符串。

```
> PRINT(TYPE(3))
NUMBER
> PRINT(TYPE("3"))
STRING
```

6.11 卡带数据

使用 **CARTDATA**(), **DSET**() 和 **DGET**()，可以在用户的 PICO-8 上存储 64 个数字（256 字节）的持久化数据，这些数据在卡带卸载或 PICO-8 关闭后仍然保留。这可以作为一种轻量级的方式存储高分或其他玩家进度。它还可以用于在不同的卡带或卡带版本之间共享数据。

如果需要超过 256 字节的数据，也可以直接使用 **CSTORE**() 写入卡带。缺点是数据将绑定到该特定版本的卡带。例如，如果游戏更新，玩家将丢失他们的存档。此外，还需要在卡带的数据部分留出一些空间作为存储。

另一种方法是通过向 **CSTORE**() 指定第四个参数，将数据写入第二个卡带。这需要进行卡带交换（实际上只是用户需要观看一个旋转的卡带动画 1 秒）。

```
CSTORE(0,0,0X2000, "SPRITE SHEET.P8")
-- 后续，恢复保存的数据:
RELOAD(0,0,0X2000, "SPRITE SHEET.P8")
```

CARTDATA(ID)

打开一个由 ID 索引的永久数据存储槽，可以使用 **DSET**() 和 **DGET**() 存储和检索最多 256 字节（64 个数字）的数据。

```
CARTDATA("ZEP_DARK_FOREST")
DSET(0, SCORE)
```

ID 是一个最多 64 个字符的字符串，应该足够独特，以避免其他卡带意外使用相同的 ID。合法字符包括 a..z, 0..9 和下划线 (_)

如果成功加载数据则返回 true，否则返回 false。

CARTDATA 每个卡带执行只能调用一次，因此只能使用一个数据槽。

一旦设置了 cartdata ID，内存区域 0X5E00..0X5EFF 将映射到永久存储，可以通过直接访问或使用 **DGET**()/**DSET**() 访问。

无需刷新写入的数据 -- 即使通过直接 **POKE**() 修改 0X5E00..0X5EFF，数据也会自动保存到永久存储。

DGET(INDEX)

获取 INDEX (0..63) 处存储的数字

使用此函数之前必须先调用 **CARTDATA**()

DSET(INDEX, VALUE)

设置 INDEX (0..63) 处存储的数字

使用此函数之前必须先调用 **CARTDATA**()

6.12 GPIO

GPIO 代表“通用输入输出”，允许机器之间进行通信。PICO-8 将 0x5f80..0x5fff 范围内的字节映射到可以使用 POKE() 写入（例如点亮 LED）或 PEEK() 读取（例如读取开关状态）的 GPIO 引脚。

GPIO 对于不同的主机平台具有不同的含义：

```
CHIP:        0x5f80..0x5f87 映射到 xio-p0..xio-p7
Pocket CHIP:  0x5f82..0x5f87 映射到 GPIO1..GPIO6
  // xio-p0 & p1 在外壳内部的原型区域暴露。
Raspberry Pi: 0x5f80..0x5f9f 映射到 wiringPi 引脚 0..31
  // 请参见 http://wiringpi.com/pins/ 以获取不同型号的映射。
  // 另外：注意 BCM 和 WiringPi GPIO 编号的区别！
```

CHIP 和 Raspberry Pi 的值都是数字：0 (LOW) 和 255 (HIGH)

一个程序用于闪烁连接的任何 LED 开关：

```
T = 0
FUNCTION _DRAW()
 CLS(5)
 FOR I=0,7 DO
  VAL = 0
  IF (T % 2 < 1) VAL = 255
  POKE(0X5F80 + I, VAL)
  CIRCFILL(20+I*12,64,4,VAL/11)
 END
 T += 0.1
END
```

串口

对于更精确的定时，可以使用 **SERIAL**() 命令。GPIO 写入是缓冲的，并在每帧结束时分派，允许以比手动使用 **POKE**() 调用更高的速度或更规则的时钟循环。

SERIAL(CHANNEL, ADDRESS, LENGTH)

CHANNEL:
 0x000..0x0fe　对应 GPIO 引脚编号；发送 0x00 表示 LOW 或 0xFF 表示 HIGH
 0x0ff　　延迟；长度表示“持续时间”（微秒，不包括开销）
 0x400..0x401　ws281x LED 字符串（实验性）

ADDRESS: PICO-8 内存位置读取/写入。

LENGTH: 要发送的字节数。允许 1/8 发送部分位字符串。

例如，要逐位发送一个字节到典型的 APA102 LED 字符串：

```
VAL = 42          -- 要发送的值
DAT = 16 CLK = 15 -- 数据和时钟引脚取决于设备
POKE(0X4300,0)    -- 要发送的数据（单个字节：0 或 0XFF）
POKE(0X4301,0XFF)
FOR B=0,7 DO
  -- 发送位（高位在前）
  SERIAL(DAT, BAND(VAL, SHL(1,7-B))>0 AND 0X4301 OR 0X4300, 1)
  -- 循环时钟
  SERIAL(CLK, 0X4301)
  SERIAL(0XFF, 5) -- 延迟 5
  SERIAL(CLK, 0X4300)
  SERIAL(0XFF, 5) -- 延迟 5
END
```

额外的通道可用于与主机操作系统进行字节流通信。这些通道主要用于在开发工具链时的类 UNIX 环境中最有用，而在运行 BBS 或导出的卡带时不可用 [1]。所有情况下的最大传输速率为 64k/sec（阻塞 CPU）。

0x800 拖放的文件 // stat(120) 返回 TRUE 时数据可用
0x802 拖放的图像 // stat(121) 返回 TRUE 时数据可用
0x804 stdin
0x805 stdout
0x806 文件指定为: pico8 -i filename
0x807 文件指定为: pico8 -o filename

拖放到 PICO-8 的图像文件以字节流形式出现在通道 0x802 上，具有特殊格式：前 4 字节是图像的宽度和高度（每个 2 字节小端，类似于 PEEK2），然后是按读取顺序排列的图像，每个像素一个字节，颜色适合于文件拖放时的显示调色板。

[1] 通道 0x800 和 0x802 在导出的二进制文件中可用，但最大文件大小为 256k，或 128x128 对于图像。

HTML

导出为 HTML / .js 的卡带使用整数数组（pico8_gpio）表示 GPIO 引脚。外壳 HTML 应该定义该数组：

```
var pico8_gpio = Array(128);
```

6.13 鼠标和键盘输入

// 实验性 -- 但在大多数平台上都能正常工作

通过启用 devkit 输入模式可以实现鼠标和键盘输入：

POKE(0x5F2D, 标志) -- 其中标志为：

0x1 启用
0x2 鼠标按钮触发 btn(4)..btn(6)
0x4 指针锁定（使用 stat 38..39 读取移动）

请注意，并非每个 PICO-8 都会连接键盘或鼠标，因此在发布到 Lexaloffle BBS 时，建议使键盘和/或鼠标控制可选且默认关闭（如果可能）。当启用 devkit 输入模式时，会向 BBS 用户显示一条消息，警告程序可能期望超出标准 6 按钮控制器的输入。

鼠标和键盘的状态可以通过 stat(x) 查找：

STAT(30) -- (布尔值) 当有按键可用时为真
STAT(31) -- (字符串) 键盘返回的字符
STAT(32) -- 鼠标 X
STAT(33) -- 鼠标 Y
STAT(34) -- 鼠标按钮（位字段）
STAT(36) -- 鼠标滚轮事件
STAT(38) -- 相对 x 移动（以主机桌面像素为单位） -- 需要标志 0x4
STAT(39) -- 相对 y 移动（以主机桌面像素为单位） -- 需要标志 0x4

6.14 附加的 Lua 功能

PICO-8 还暴露了 Lua 的两个功能供高级用户使用：元表和协程。

有关更多信息，请参阅 Lua 5.2 手册。

▨ 元表

元表可以用于定义对象在特定操作下的行为。例如，要使用表来表示可以相加的二维向量，可以通过为元表定义一个 "__add" 函数来重新定义 "+" 运算符：

```
VEC2D={
  __ADD=FUNCTION(A,B)
    RETURN {X=(A.X+B.X), Y=(A.Y+B.Y)}
  END
}

V1={X=2,Y=9} SETMETATABLE(V1, VEC2D)
V2={X=1,Y=5} SETMETATABLE(V2, VEC2D)
V3 = V1+V2
PRINT(V3.X..","..V3.Y) -- 3,14
```

▨ SETMETATABLE(TBL, M)

将表 TBL 的元表设置为 M

▨ GETMETATABLE(TBL)

返回表 t 的当前元表，如果没有设置元表则返回 nil

▨ RAWSET(TBL, KEY, VALUE)

▨ RAWGET(TBL, KEY)

▨ RAWEQUAL(TBL1,TBL2)

▨ RAWLEN(TBL)

绕过元方法直接访问表

▨ 函数参数

函数参数列表可以使用 ...

```
FUNCTION PREPRINT(PRE, S, ...)
  LOCAL S2 = PRE..TOSTR(S)
  PRINT(S2, ...) -- 将剩余的参数传递给 PRINT()
END
```

要接受可变数量的参数，可以将它们定义为一个表，或者使用 Lua 的 select() 函数。select(index, ...) 返回索引之后的所有参数。

```
FUNCTION FOO(...)
  LOCAL ARGS={...} -- 成为参数表
  FOREACH(ARGS, PRINT)
  ?SELECT("#",...)    -- 另一种方式来计算参数数量
  FOO2(SELECT(3,...)) -- 将第 3 个参数及之后的参数传递给 FOO2()
END
```

▨ 协程

协程提供了一种以某种并发方式运行程序的不同部分的方法，类似于线程。函数可以作为协程调用，使用

▨ YIELD() 任意次数暂停，然后在相同点恢复。

```
FUNCTION HEY()
  PRINT("DOING SOMETHING")
  YIELD()
  PRINT("DOING THE NEXT THING")
  YIELD()
  PRINT("FINISHED")
END

C = COCREATE(HEY)
FOR I=1,3 DO CORESUME(C) END
```

▨ COCREATE(F)

为函数 f 创建一个协程。

▨ CORESUME(C, [P0, P1 ..])

运行或继续协程 c。参数 p0, p1.. 传递给协程的函数。

如果协程没有错误完成，则返回 true；如果发生错误，则返回 false, error_message。

在协程内部发生的运行时错误不会导致程序停止运行。建议将 CORESUME() 包装在 ASSERT() 中。如果断言失败，它将打印由 coresume 生成的错误消息。

▨ ASSERT(CORESUME(C))

▨ COSTATUS(C)

返回协程 C 的状态字符串： "running" "suspended" "dead"

▨ YIELD

暂停执行并返回到调用者。

7.01 附录 A: P8SCII 控制码

使用 PRINT() 打印时，某些字符具有特殊含义，可以用来改变光标位置和文本渲染样式。PICO-8 中的控制字符是 CHR(0)..CHR(15)，可以写成转义序列（例如 "\n" 表示换行等）。

下面的一些控制码需要参数，这些参数使用一种超集于十六进制格式的方案书写。也就是说，'0'..'f' 也表示 0..15。但是 'f' 之后的字符也被接受：'g' 表示 16 等等。这样的参数在下面表示为 P0, P1。

例如，要以蓝色背景 ("\#c") 和深灰色前景 ("\f5") 打印：

```
PRINT("\\#C\\F5 蓝色 ")
```

PRINT() 对绘图状态的唯一影响是改变光标位置和前景色；所有其他属性在每次调用 **PRINT**() 时都会重置。

■ 控制码

0 "\0" 终止打印
1 "*" 重复下一个字符 P0 次。例如："*3a" --> aaa
2 "\#" 用颜色 P0 绘制实心背景
3 "\-" 水平移动光标 P0-16 像素
4 "\|" 垂直移动光标 P0-16 像素
5 "\+" 移动光标 P0-16, P1-16 像素
6 "\^" 特殊命令（见下文）
7 "\a" 音频（见下文）
8 "\b" 退格
9 "\t" 制表符
a "\n" 换行
b "\v" 装饰前一个字符（见下文）
c "\f" 设置前景色
d "\r" 回车
e "\014" 切换到 0x5600 处定义的字体
f "\015" 切换到默认字体

■ 特殊命令

这些命令都以 "\^" 开头，并且最多接受两个参数（P0, P1）。例如，要将屏幕清除为深蓝色：PRINT("\^c1")

1..9 跳过 1,2,4,8,16,32..256 帧
c 清屏为颜色 P0，将光标设置为 0,0
d 设置每打印一个字符的延迟为 P0 帧
g 将光标位置设置为起始位置
h 将起始位置设置为当前光标位置
j 跳转到绝对位置 P0*4, P1*4（以屏幕像素为单位）

r 设置右侧字符换行边界为 P0*4
s 设置制表符宽度为 P0 像素（用于 "\t"）
x 设置字符宽度（默认：4）
y 设置字符高度（默认：6）

■ 渲染模式选项

// 在这些前面加上 "-" 以禁用：例如 ?"\^i on \^-i off "

w 宽模式：按 2x1 缩放
t 高模式：按 1x2 缩放
\= 条纹模式：当使用宽模式或高模式时，只绘制偶数像素
p 弹球模式：相当于同时设置宽模式、高模式和条纹模式
i 反转
b 边框：在左侧和顶部添加 1 像素的填充（默认开启）
实心背景 // 默认关闭，但当使用 "\#" 时会自动开启

■ 原始内存写入

下面两个命令接受 4 个字符的十六进制参数：

@addrnnnn[binstr] 将 nnnn 字节写入地址 addr
!addr[binstr]　将所有剩余字符写入地址 addr

例如，要将 4 字节写入屏幕内存的中间位置：

```
>?"\\^@70000004xxxxhello"
```

■ 单次字符

可以使用 \^. 后跟 8 字节的原始二进制数据，或 \^: 后跟 8 个 2 位十六进制值来指定和打印内联字符数据。数据格式与自定义字体相同；每个字节指定一行 1 位像素值，最低位在左侧。

\^.[8 字节原始二进制数据]
\^:[16 字节十六进制]

例如，打印一个猫：

```
> ?"\\^:447cb67c3e7f0106"
```

■ 音频

? ?"\A" -- 单音蜂鸣 ?"\A12" -- 播放现有数据在 SFX 12

如果没有指定 SFX 索引，则会自动选择一个未使用的 SFX 在 60..63 之间。为了在播放前填充 SFX 数据，可以附加以下命令。

1. （可选）SFX 属性必须在开始时出现一次，因为它们适用于整个声音：

s P0　设置 SFX 速度
l P0 P1　设置 SFX 循环开始和结束点

2. 音符数据：

音符写成 a..g，可选后跟升号 # 或降号 - 和八度号。

```
PRINT "\\ACE-G" -- 小三和弦
```

空音符可以写成点：

```
PRINT "\\AC..E-..G" -- 断奏小三和弦
```

音符属性命令适用于随后的音符：

i P0　设置乐器（默认：5）
v P0　设置音量　（默认：5）
x P0　设置效果　（默认：0）
< > 增加或减少音量 1

例如，要以快速（速度 4）、断奏（效果 5）的琶音从 C1 开始播放：

```
PRINT "\\AS4X5C1EGC2EGC3EGC4"
```

■ 装饰字符

控制字符 \v 可以用来在不管理光标位置的情况下，使用给定偏移量装饰最后打印的字符。打印装饰字符后，会恢复之前的光标位置。

格式为 \v P0 char，其中 P0 是一个数字，给出所需的偏移量，char 是要打印的字符（相对于之前打印的字符）。

偏移量将 x 打包到最低 2 位，并从阅读顺序的 (-2,-8) 开始。因此 3 表示 (+1, -8)，4 表示 (-2, -7) 等等。

例如，要写 "cafÃ©!"，使用逗号来绘制尖音符号：

```
PRINT"\\NCAFE\\VB,!"
```

在这种情况下 P0 是 'b'，读作数字 11。因此逗号绘制在：

```
x = (11%4)-2 = 1
y = (11\\4)-8 = -6
```

■ 自定义字体

自定义字体可以在 0x5600 处定义，由每个字符 8 字节 * 256 字符 = 2048 字节组成。每个字符是一个 8x8 位图（1 位/像素），从顶部开始，每一行是一个字节，从 0x1 开始在左侧。

前 128 字节（字符 0~15 从不绘制）描述字体的属性：

0x5600 字符宽度（像素）（可以大于 8，但只绘制 8 像素）
0x5601 字符宽度（字符 128 及以上）
0x5602 字符高度（像素）
0x5603 绘制偏移 x
0x5604 绘制偏移 y
0x5605 标志：0x1 应用_size_adjustments 0x2: 相对于光标起始位置应用制表符
0x5606 制表符宽度（像素）（仅在使用替代字体时使用）
0x5607 未使用

接下来的 120 字节用于调整字符 16..255 的宽度和垂直偏移。每个半字节（低位半字节优先）描述一个字符的调整：

位 0x7: 调整字符宽度为 0,1,2,3,-4,-3,-2,-1
位 0x8: 当设置时，将字符向上绘制一个像素（适用于拉丁字符的重音符号）

■ 默认属性

虽然属性在每次调用 **PRINT**() 时都会重置，但可以通过写入内存地址 0x5f58..0x5f5b 来设置它们的默认值。

0x5f58 // 位域
  0x1 当设置为 0x1 时，位 1..7 被观察：
  0x2 填充
  0x4 宽
  0x8 高
  0x10 实心背景
  0x20 反转
  0x40 条纹（当宽或高时）
  0x80 使用自定义字体

// 例如：poke(0x5f58, 0x1 | 0x2 | 0x4 | 0x8 | 0x20 | 0x40) -- 处处弹球

0x5f59 char_w （低半字节), char_h （高)
0x5f5a char_w2（低半字节), tab_w （高)
0x5f5b offset_x (低半字节), offset_y (高)

// 任何半字节等于 0 都被忽略
// tab_w（全局制表符宽度）值映射到 4..60