来自 ba733bf60e8fe4f6e7a7267e2cc78009d8b2d21b 2001 年 9 月 17 日星期一 00:00:00
来自：mooglyguy <therealmogminer@gmail.com>
日期：2018 年 11 月 3 日星期六 17:31:44 +0100
主题：MASKROM/MASK ROM -> mask ROM, nw

---
 src/mame/drivers/aleck64.cpp | src/mame/drivers/aleck64.cpp | 10+-
 src/mame/drivers/alpha68k.cpp | src/mame/drivers/alpha68k.cpp | 2+-
 src/mame/drivers/asuka.cpp | src/mame/drivers/asuka.cpp | 20+--
 src/mame/drivers/cat.cpp | src/mame/drivers/cat.cpp | 42 +++---
 src/mame/drivers/cave.cpp | src/mame/drivers/cave.cpp | 16+-
 src/mame/drivers/cischeat.cpp | src/mame/drivers/cischeat.cpp | 20+--
 src/mame/drivers/cps1.cpp | src/mame/drivers/cps1.cpp | 18 +--
 src/mame/drivers/cps2.cpp | src/mame/drivers/cps2.cpp | 6+-
 src/mame/drivers/ddragon3.cpp | src/mame/drivers/ddragon3.cpp | 2+-
 src/mame/drivers/dec8.cpp | src/mame/drivers/dec8.cpp | 2+-
 src/mame/drivers/deco_mlc.cpp | src/mame/drivers/deco_mlc.cpp | 2+-
 src/mame/drivers/djboy.cpp | src/mame/drivers/djboy.cpp | 20+--
 src/mame/drivers/dynax.cpp | src/mame/drivers/dynax.cpp | 4+-
 src/mame/drivers/electron.cpp | src/mame/drivers/electron.cpp | 2+-
 src/mame/drivers/esd16.cpp | src/mame/drivers/esd16.cpp | 8+-
 src/mame/drivers/fcrash.cpp | src/mame/drivers/fcrash.cpp | 36++---
 src/mame/drivers/gcpinbal.cpp | src/mame/drivers/gcpinbal.cpp | 18 +--
 src/mame/drivers/hikaru.cpp | src/mame/drivers/hikaru.cpp | 20+--
 src/mame/drivers/hp_ipc.cpp | src/mame/drivers/hp_ipc.cpp | 4+-
 src/mame/drivers/itech32.cpp | src/mame/drivers/itech32.cpp | 8+-
 src/mame/drivers/jchan.cpp | src/mame/drivers/jchan.cpp | 6+-
 src/mame/drivers/kaneko16.cpp | src/mame/drivers/kaneko16.cpp | 2+-
 src/mame/drivers/konamigv.cpp | src/mame/drivers/konamigv.cpp | 32++--
 src/mame/drivers/konamim2.cpp | src/mame/drivers/konamim2.cpp | 2+-
 src/mame/drivers/lastfght.cpp | src/mame/drivers/lastfght.cpp | 6+-
 src/mame/drivers/m107.cpp | src/mame/drivers/m107.cpp | 10+-
 src/mame/drivers/macrossp.cpp | src/mame/drivers/macrossp.cpp | 2+-
 src/mame/drivers/midzeus.cpp | src/mame/drivers/midzeus.cpp | 58 +++----
 src/mame/drivers/model1.cpp | src/mame/drivers/model1.cpp | 12+-
 src/mame/drivers/model3.cpp | src/mame/drivers/model3.cpp | 46 +++---
 src/mame/drivers/namcofl.cpp | src/mame/drivers/namcofl.cpp | 26++--
 src/mame/drivers/namconb1.cpp | src/mame/drivers/namconb1.cpp | 44 +++---
 src/mame/drivers/namcos10.cpp | src/mame/drivers/namcos10.cpp | 8+-
 src/mame/drivers/namcos11.cpp | src/mame/drivers/namcos11.cpp | 14+-
 src/mame/drivers/namcos12.cpp | src/mame/drivers/namcos12.cpp | 62 ++++----
 src/mame/drivers/namcos2.cpp | src/mame/drivers/namcos2.cpp | 54 +++---
 src/mame/drivers/namcos22.cpp | src/mame/drivers/namcos22.cpp | 26++--
 src/mame/drivers/namcos23.cpp | src/mame/drivers/namcos23.cpp | 102 ++++++++--------
 src/mame/drivers/naomi.cpp | src/mame/drivers/naomi.cpp | 16+-
 src/mame/drivers/nitedrvr.cpp | src/mame/drivers/nitedrvr.cpp | 4+-

```
 src/mame/drivers/nmk16.cpp | src/mame/drivers/nmk16.cpp | 76 +++++-----
 src/mame/drivers/pgm.cpp | src/mame/drivers/pgm.cpp | 50+++----
 src/mame/drivers/powerins.cpp | src/mame/drivers/powerins.cpp | 14+-
 src/mame/drivers/psikyo.cpp | src/mame/drivers/psikyo.cpp | 6+-
 src/mame/drivers/psikyo4.cpp | src/mame/drivers/psikyo4.cpp | 4+-
 src/mame/drivers/raiden2.cpp | src/mame/drivers/raiden2.cpp | 340 +++++++++++++++++++++++++---------------------
 src/mame/drivers/segahang.cpp | src/mame/drivers/segahang.cpp | 10+-
 src/mame/drivers/seta2.cpp | src/mame/drivers/seta2.cpp | 4+-
 src/mame/drivers/shangha3.cpp | src/mame/drivers/shangha3.cpp | 12+-
 src/mame/drivers/speglsht.cpp | src/mame/drivers/speglsht.cpp | 6+-
 src/mame/drivers/ssv.cpp | src/mame/drivers/ssv.cpp | 30++--
 src/mame/drivers/studio2.cpp | src/mame/drivers/studio2.cpp | 6+-
 src/mame/drivers/subsino2.cpp | src/mame/drivers/subsino2.cpp | 4+-
 src/mame/drivers/suprnova.cpp | src/mame/drivers/suprnova.cpp | 20+--
 src/mame/drivers/taito_f3.cpp | src/mame/drivers/taito_f3.cpp | 4+-
 src/mame/drivers/tnzs.cpp | src/mame/drivers/tnzs.cpp | 64 ++++----
 src/mame/drivers/tsispch.cpp | src/mame/drivers/tsispch.cpp | 36++---
 src/mame/drivers/vamphalf.cpp | src/mame/drivers/vamphalf.cpp | 36++---
 src/mame/drivers/vendetta.cpp | src/mame/drivers/vendetta.cpp | 94 ++++++------
 src/mame/drivers/vt100.cpp | src/mame/drivers/vt100.cpp | 252 +++++++++++++++++-----------------------
 src/mame/drivers/zn.cpp | src/mame/drivers/zn.cpp | 44 +++---
 src/mame/drivers/zr107.cpp | src/mame/drivers/zr107.cpp | 4+-
 更改了 62 个文件，插入了 954 个（+），删除了 954 个（-）

diff --git a/src/mame/drivers/aleck64.cpp b/src/mame/drivers/aleck64.cpp
索引 01ae969d44c..ee4dfe272db 100644
---a/src/mame/drivers/aleck64.cpp
+++ b/src/mame/drivers/aleck64.cpp
@@ -80,7 +80,7 @@ 注意：
 购物车包含：
                    CIC-NUS-5101：开机保护芯片
                    BK4D-NUS ：类似于 N64 控制台推车中使用的保存芯片
- NUS-ZHAJ.U3：64Mbit 28针DIP串行MASKROM
+ NUS-ZHAJ.U3 : 64Mbit 28 针 DIP 串行掩码 ROM

       - RCA 音频插头输出立体声。输出常规单声道声音
         也可通过标准 JAMMA 连接器。
@@ -150,7 +150,7 @@ PCB 布局
 笔记：
       水平同步：15.73kHz
       垂直同步：60Hz
- * : 8M - 32M 42 针 DIP MASKROM 的未安装插座
+ * : 8M - 32M 42 针 DIP 掩码 ROM 的未安装插座
       O：按钮复位开关
       X：特殊（Aleck64？）数字操纵杆的连接器
       CPU-NUS A：PCB 上标记为“VR4300”
@@ -158,8 +158,8 @@ 注意：
```

```
        只读存储器
        ----
- TET-01M.U5：8Mbit 42 针 MASKROM
- NUS-CZAJ.U4：128Mbit 28针DIP串行MASKROM
+ TET-01M.U5：8Mbit 42 引脚掩模 ROM
+ NUS-CZAJ.U4：128Mbit 28 针 DIP 串行掩码 ROM
        AT24C01.U34：128字节x 8位串行EEPROM

        - RCA 音频插头输出立体声。输出常规单声道声音
@@ -169,7 +169,7 @@ 注意：
          大多数人不需要关心的逻辑设备:-)

        - Seta/N64 Aleck64 硬件也类似，但不是高容量
- 串行 MASKROM 位于主板上，位于插入插槽的推车中。
+ 串行掩码 ROM 位于主板上，位于插入插槽的推车中。

 */

diff --git a/src/mame/drivers/alpha68k.cpp b/src/mame/drivers/alpha68k.cpp
索引 1723cf11bd3..7e57fe0eff0 100644
---a/src/mame/drivers/alpha68k.cpp
+++ b/src/mame/drivers/alpha68k.cpp
@@ -3155,7 +3155,7 @@ ROM_START( sbasebalj )
         ROM_LOAD（“kcb-chr0.h16”，0x1e0000，0x80000，CRC（b8a1a088）SHA1（cb21a04387431b1810130abd86a2ebf78cf09a3b））
 ROM_END

-ROM_START( tnextspc ) /* gfx 的 MASKROM */
+ROM_START( tnextspc ) /* gfx 的掩模 ROM */
         ROM_REGION( 0x40000, "主CPU", 0 )
         ROM_LOAD16_BYTE（“ns_4.bin”，0x00000，0x20000，CRC（4617cba3）SHA1（615a1e67fc1c76d2be004b19a965f423b8daaf5c））
         ROM_LOAD16_BYTE（“ns_3.bin”，0x00001，0x20000，CRC（a6c47fef）SHA1（b7e4a0fffd5c44ed0b138c1ad04c3b6644ec463b））
diff --git a/src/mame/drivers/asuka.cpp b/src/mame/drivers/asuka.cpp
索引 323ff465456..699427314b9 100644
---a/src/mame/drivers/asuka.cpp
+++ b/src/mame/drivers/asuka.cpp
@@ -198,13 +198,13 @@ Earthjkr：屏幕尺寸错误？绿色蓝图的左边缘
 吸引看起来像是错误地离开屏幕。

 Cadash：用于双街机设置的挂钩：将涉及模拟额外的
-微控制器，07 ROM可能是它的程序。卡达什背景
+微控制器，07 ROM可能就是它的程序。卡达什背景
 保存状态时颜色无法正确重新初始化。

 Galmedes：测试模式中 select1/2 卡在打开状态。

 Eto：$76d0 可能是保护支票？它读取和写入
```

```
- 程序ROM。不过似乎不会造成问题。
+程序ROM。不过似乎不会造成问题。

 DIP 位置已验证：
     - bonzeadv（手册）
@@ -1337,7 +1337,7 @@ ROM_START( asuka ) /* Taito PCB: ASKA&ASKA - K1100388A / J1100169A */
         ROM_LOAD（“b68-05.ic43”，0x00000，0x104，CRC（d6524ccc）SHA1（f3b56253692aebb63278d47832fc27b8b212b59c））
 ROM_END

-ROM_START( asukaj ) /* 已知存在但未转储：带有 B68 08-1 和 B68 09-1 程序 ROM 的修订版 1 */
+ROM_START( asukaj ) /* 已知存在但未转储：带有 B68 08-1 和 B68 09-1 程序 ROM 的修订版 1 */
         ROM_REGION( 0x100000, "maincpu", 0 ) /* 68000 代码为 1024k */
         ROM_LOAD16_BYTE（“b68-09.ic23”，0x00000，0x20000，CRC（1eaa1bbb）SHA1（01ca6a5f3c47dab49654b84601119714eb329cc5））
         ROM_LOAD16_BYTE（“b68-08.ic8”，0x00001，0x20000，CRC（8cc96e60）SHA1（dc94f3fd48c0407ec72e8330bc688e9e16d39213））
@@ -1421,7 +1421,7 @@ ROM_START( cadashp )
         ROM_REGION( 0x08000, "subcpu", 0 ) /* HD64180RP8 代码 (链接) */
         ROM_LOAD（“com.ic57”，0x00000，0x08000，CRC（bae1a92f）SHA1（dbe10a02a294dfa7d6052a692c3a49aad85d6ffd））

- // 所有其他 rom 都在某种环氧树脂下，假设是相同的..
+ // 所有其他 ROM 都在某种环氧树脂下，假设是相同的..
         ROM_REGION( 0x80000, "gfx1", 0 )
         ROM_LOAD( "c21-02.9", 0x00000, 0x80000, CRC(205883b9) SHA1(5aafee8cab3f949a7db91bcc26912f331041b51e) ) /* SCR 块 (8 x 8) */

@@ -1678,10 +1678,10 @@ ROM_START( Earthjkr ) /* Taito PCB: K1100388A / J1100169A */
         ROM_LOAD16_WORD( "ej_30e.ic30", 0x80000, 0x80000, CRC(49d1f77f) SHA1(f6c9b2fc88b77cc9baa5be48da5c3eb72310e471) ) /* 修复 ROM */

         ROM_REGION( 0x80000, "gfx1", 0 )
- ROM_LOAD( "ej_chr-0.ic3", 0x00000, 0x80000, CRC(ac675297) SHA1(2a34e1eae3a4be84dbf709053f5e8a781b1073fc) ) /* SCR 块 (8 x 8) - 掩码 ROM */
+ ROM_LOAD( "ej_chr-0.ic3", 0x00000, 0x80000, CRC(ac675297) SHA1(2a34e1eae3a4be84dbf709053f5e8a781b1073fc) ) /* SCR 块 (8 x 8) - 掩模 ROM */

         ROM_REGION( 0xa0000, "gfx2", 0 )
- ROM_LOAD ( "ej_obj-0.ic6", 0x00000, 0x80000, CRC(5f21ac47) SHA1(45c94ffb53ee9b822b0676f6fb151fed4ce6d967) ) /* 精灵 (16 x 16) - MASK ROM */
+ ROM_LOAD ( "ej_obj-0.ic6", 0x00000, 0x80000, CRC(5f21ac47) SHA1(45c94ffb53ee9b822b0676f6fb151fed4ce6d967) ) /* 精灵 (16 x 16) - 掩模 ROM */
         ROM_LOAD16_BYTE（“ej_1.ic5”，0x80000，0x10000，CRC（cb4891db）SHA1（af1112608cdd897ef6028ef617f5ca69d7964861））
         ROM_LOAD16_BYTE（“ej_0.ic4”，0x80001，0x10000，CRC（b612086f）SHA1（625748fcb698ec57b7b3ce46019cf85de99aaaa1））

@@ -1698,9 +1698,9 @@ ROM_END

 // 已知存在（未转储）带有 ROM 3 和 4 的日文版本，也带有与上述相同或不同版本的“A”标记？
 // 还已知存在（未转储）美国版的《地球小丑》，标题屏幕显示“DISTRIBUTED BY ROMSTAR, INC.” ROM 已编号
-// 从 0 到 4，IC30 处的修复 ROM 标记为 1，即使 IC5 也标记为 1，类似于以下设置：
+// 从 0 到 4，IC30 处的修复 ROM 标记为 1，即使 IC5 也标记为 1，类似于以下设置：

-ROM_START( Earthjkrp ) // 生产 PCB 是否带有 MASK rom，可能只是早期修订版，而不是原型
+ROM_START( Earthjkrp ) // 是带有掩模 ROM 的生产 PCB，可能只是早期修订版，而不是原型
         ROM_REGION( 0x100000, "maincpu", 0 ) /* 68000 代码为 1024k */
         ROM_LOAD16_BYTE（“3.ic23”，0x00001，0x20000，CRC（26c33225）SHA1（b039c47d0776c90813ab52c867e95989cab2c567））
```

```
        ROM_LOAD16_BYTE（“4.ic8”，0x00000，0x20000，CRC（e9b1ef0c）SHA1（5e104146d37922a8c7e93696c2c156223653025b））
@@ -1708,10 +1708,10 @@ ROM_START( Earthjkrp ) // 是带有 MASK rom 的生产 PCB，可以
        ROM_LOAD16_WORD( "5.ic30", 0x80000, 0x80000, CRC(bf760b2d) SHA1(4aff36623e5a31ab86c77461fa93e40e77f08edd) ) /* 修复 ROM */

        ROM_REGION( 0x80000, "gfx1", 0 )
- ROM_LOAD( "ej_chr-0.ic3", 0x00000, 0x80000, CRC(ac675297) SHA1(2a34e1eae3a4be84dbf709053f5e8a781b1073fc) ) /* SCR 块 (8 x 8) - 掩码 ROM */
+ ROM_LOAD( "ej_chr-0.ic3", 0x00000, 0x80000, CRC(ac675297) SHA1(2a34e1eae3a4be84dbf709053f5e8a781b1073fc) ) /* SCR 块 (8 x 8) - 掩模 ROM */

        ROM_REGION( 0xa0000, "gfx2", 0 )
- ROM_LOAD ( "ej_obj-0.ic6", 0x00000, 0x80000, CRC(5f21ac47) SHA1(45c94ffb53ee9b822b0676f6fb151fed4ce6d967) ) /* 精灵 (16 x 16) - MASK ROM */
+ ROM_LOAD ( "ej_obj-0.ic6", 0x00000, 0x80000, CRC(5f21ac47) SHA1(45c94ffb53ee9b822b0676f6fb151fed4ce6d967) ) /* 精灵 (16 x 16) - 掩模 ROM */
        ROM_LOAD16_BYTE（“1.ic5”，0x80000，0x10000，CRC（cb4891db）SHA1（af1112608cdd897ef6028ef617f5ca69d7964861））
        ROM_LOAD16_BYTE（“0.ic4”，0x80001，0x10000，CRC（b612086f）SHA1（625748fcb698ec57b7b3ce46019cf85de99aaaa1））

diff --git a/src/mame/drivers/cat.cpp b/src/mame/drivers/cat.cpp
索引 a2bc4f38a6c..f56118d5591 100644
---a/src/mame/drivers/cat.cpp
+++ b/src/mame/drivers/cat.cpp
@@ -120,7 +120,7 @@ Canon Cat 版本：
 事实上，猫只有一个版本被广泛发行，那就是美国版本。
 * 可能会发布极少数英国/欧洲单位作为测试。
   如果是这样，它们的键盘键帽和不同的键帽将略有不同
- 系统和拼写检查ROM。
+ 系统和拼写检查 ROM。

 至于原型/开发猫机器，存在一些小的变体：
 * 原型猫主板使用 16k*4bit DRAM，而不是 64k*4bit 作为
@@ -174,11 +174,11 @@ 待办事项：
   原始编译，以及 Dwight 从发布的源代码重新编译），
   2.42（需要转储）
   可能会生产一些 UK 1.74 或 2.40 原型车；代码
- 这些的 rom 会有所不同（它们包含不同的拼写检查“核心”代码），因为
- 以及拼写检查 ROM、键盘 ID 和键帽。
-- 已知的拼写检查 ROM：NH7-0684（美国，已弃用）；NH7-0724（英国，需要转储）；
+ 这些的 ROM 会有所不同（它们包含不同的拼写检查“核心”代码），因为
+ 以及拼写检查 ROM、键盘 ID 和键帽。
+- 已知的拼写检查 ROM：NH7-0684（美国，已弃用）；NH7-0724（英国，需要转储）；
   NH7-0813/0814（魁北克/法国，需要转储）；NH7-1019/1020/1021（德国，需要转储）
- 非美国版 ROM 可能从未正式发布。
+ 非美国版 ROM 可能从未正式发布。
   词汇表来源：美国遗产（美国和英国）、拉鲁斯图书馆（法国）、
   朗根沙伊特 (德国)
 -（真的很好但完全不必要的功能）：由于开放巴士，
@@ -186,7 +186,7 @@ 待办事项：
   分别在真实的机器上（因此出现反转/“失败”状态）。
   这还需要子周期精确的 68k 开放总线仿真才能实现
   模拟 UDS/LDS“未连接”这一事实 （不清楚，因为这
```

```
 - 发生在 SVROMS（或 svram 或代码 ROM，例如
 + 发生在 SVROMS（或 svram 或代码 ROM，例如
    那很重要！）
  - 将电池低电压输入挂接到拨码开关。
  - 挂钩无法连接到拨码开关。
@@ -721,7 +721,7 @@ a23 a22 a21 a20 a19 a18 a17 a16 a15 a14 a13 a12 a11 a10 a9 a8 a7 a6 a5 a4
 0 0 0 xx 1 1 0 xxxxxxxxxxxxxxx 0 0 开放总线（读为 0x2e）[可以通过 GA2 /RAMCS 控制？]
 0 0 0 xx 1 xxxxxxxxxxxxxxxxx 1 0 开放总线（读为 0x80）[可以通过 GA2 /RAMCS 控制？]
 0 0 1 xx 0 * * * * * * * * * * * * * * * * * * * * * * * * * 0 R SVROM 2 ic7（售出的 Cat 上不存在，开放总线读取为 0x2e）[通过 GA2 /SVCS0 控制]
-0 0 1 xx 0 * * * * * * * * * * * * * * * * * * * * * * * * * 1 R SVROM 0 ic6 (MASK ROM tc531000) [通过 GA2 /SVCS0 控制]
+0 0 1 xx 0 * * * * * * * * * * * * * * * * * * * * * * * * * 1 R SVROM 0 ic6 (掩码 ROM tc531000) [通过 GA2 /SVCS0 控制]
 0 0 1 xx 1 * * * * * * * * * * * * * * * * * * * * * * * * * 0 0 开放总线（读为 0x2e）[通过 GA2 /SVCS1 控制] *见下文*
 0 0 1 xx 1 * * * * * * * * * * * * * * * * * * * * * * * * * 1 R SVROM 1 ic8（售出的 Cat 上不存在，开放总线读取为 0x80）[通过 GA2 /SVCS1 控制] *SEE以下*
                                                                                                                                *注：在 Dwight E 的用户制作的开发单元上，映射
了两个 128K SRAM 来代替
@@ -751,7 +751,7 @@ a23 a22 a21 a20 a19 a18 a17 a16 a15 a14 a13 a12 a11 a10 a9 a8 a7 a6 a5 a4
 1 0 0 xx 1 1 0 xxxxxxxxxxxxxxx * R?W {'tcb'} 测试控制位（读为 0x0000）
 1 0 0 xx 1 1 1 xxxxxxxxxxxxxxx * ? 未知（读作 0x2e80）

-1 0 1 xxxxxxxxxxxxxxxxxxxx 0 OPEN BUS (读作 0x2e80) [68k DTACK 在访问该区域时由门阵列 1 断言，用于测试？] 在真正的 IAI Shadow ROM 板上，至少有
0x40000 的 ram 驻留在此处。
+1 0 1 xxxxxxxxxxxxxxxxxxxx 0 OPEN BUS (读作 0x2e80) [68k DTACK 在访问该区域时由门阵列 1 断言，用于测试？] 在真正的 IAI 影子 ROM 板上，至少有
0x40000 的 ram 驻留在此处。
 1 1 xxxxxxxxxxxxxxxxxxxx x 0 OPEN BUS（读为 0x2e80） [访问该区域时，68k VPA 由门阵列 1 置位，用于测试？]
 */

@@ -780,7 +780,7 @@ void cat_state::cat_mem(address_map &map)
        映射(0x850000, 0x850001).r(FUNC(cat_state::cat_wdt_r)).mirror(0x18FFFE); // 看门狗和电源故障状态读取
        映射(0x860000, 0x860001).rw(FUNC(cat_state::cat_0000_r), FUNC(cat_state::cat_tcb_w)).mirror(0x18FFFE); // 测试模式
        映射(0x870000, 0x870001).r(FUNC(cat_state::cat_2e80_r)).mirror(0x18FFFE); // 开巴士吗？
- 地图(0xA00000, 0xA00001).r(FUNC(cat_state::cat_2e80_r)).mirror(0x1FFFFE); // 打开总线/dtack？0xA00000-0xA3ffff 区域是用于 cat 开发人员机器上的影
子 ROM 存储的 ram，它可以存储在普通 rom 之上，也可以跳转到普通 rom。
+ 地图(0xA00000, 0xA00001).r(FUNC(cat_state::cat_2e80_r)).mirror(0x1FFFFE); // 打开总线/dtack？0xA00000-0xA3ffff 区域是用于 cat 开发人员机器上的影
子 ROM 存储的 ram，它可以存储在普通 ROM 之上，也可以跳转到普通 ROM 之上
        映射(0xC00000, 0xC00001).r(FUNC(cat_state::cat_2e80_r)).mirror(0x3FFFFE); // 打开总线/vme？
 }

@@ -1122,8 +1122,8 @@ ROM_START( 猫 )
        ROMX_LOAD（“r240h1.ic5”，0x20000，0x10000，CRC（898dd9f6）SHA1（93e791dd4ed7e4afa47a04df6fdde359e41c2075），ROM_SKIP（1）| ROM_BIOS
（1））
        /* 此 v1.74 代码（可能）来自首次运行的“主要美国版本”
         * 佳能猫，并从机器序列号 R12014979 中转储
- * Canon cat v1.74 rom 标记为 r74；他们只添加了主要号码
- * v2.0之后的rom标签？
+ * Canon cat v1.74 ROM 标记为 r74；他们只添加了主要号码
+ * v2.0之后的ROM标签？
         */
```

```
        ROM_SYSTEM_BIOS( 2, "r174", "佳能 Cat V1.74 US 固件")
        ROMX_LOAD( "r74__01__c18c.blue.ic2", 0x00001, 0x10000, CRC(b19aa0c8) SHA1(85b3e549cfb91bd3dd32335e02eaaf9350e80900), ROM_SKIP(1) |
ROM_BIOS(2))
@@ -1135,12 +1135,12 @@ ROM_START( 猫 )

        ROM_REGION（0x80000，“svrom”，ROMREGION_ERASE00）
        // 拼写验证 ROM (SVROM)
- /* 这里的 Romspace 有点奇怪：板上有 3 个 rom 插槽：
+ /* 这里的 Romspace 有点奇怪：板上有 3 个 ROM 插槽：
         * svrom-0 映射到 200000-21ffff 每个 ODD 字节 (d8-d0)
         * svrom-1 映射到 200000-21ffff 每个偶数字节 (d15-d7)
- *（因为套接字中没有 ROM；它读取为开放总线，有时为 0x2E）
+ *（因为套接字中没有 ROM；它读取为开放总线，有时为 0x2E）
         * svrom-2 映射到每个 ODD 字节 (d8-d0) 240000-25ffff
- *（因为套接字中没有 ROM；它读取为开放总线，有时为 0x80）
+ *（因为套接字中没有 ROM；它读取为开放总线，有时为 0x80）
         * 没有 svrom-3 插座；240000-25ffff EVEN 始终读取为 0x2E
         * 由于 ROM_FILL16BE(0x0, 0x80000, 0x2e80) 不存在，
         * svrom 空间的偶数字节和后面的块需要填充
@@ -1148,13 +1148,13 @@ ROM_START( 猫 )
         * 一旦 mame/mess 核心支持“开放总线”。
         * 注意：至少还有 6 个以上存在的 SVROM（可能在
         * 有限形式），并且不被倾销：
- * 英国（1 ROM，NH7-0724）
- * 法语/魁北克（2 间客房，NH7-0813/0814）
- * 德语（3 ROM，NH7-1019/1020/1021）
- * 其中每一个也将有自己的代码 romset。
+ * 英国（1 ROM，NH7-0724）
+ * 法语/魁北克语（2 个 ROM，NH7-0813/0814）
+ * 德语（3 个 ROM，NH7-1019/1020/1021）
+ * 其中每一个也将有自己的代码 ROMset。
         */
        // NH7-0684（美国，已弃用）：
- ROMX_LOAD( "uv1__nh7-0684__hn62301apc11__7h1.ic6", 0x00000, 0x20000, CRC(229ca210) SHA1(564b57647a34acdd82159993a3990a412233da14), ROM_SKIP(1))
// 这是28pin tc531000 mask rom，长128KB；“美国”SVROM
+ ROMX_LOAD( "uv1__nh7-0684__hn62301apc11__7h1.ic6", 0x00000, 0x20000, CRC(229ca210) SHA1(564b57647a34acdd82159993a3990a412233da14), ROM_SKIP(1))
// 这是一个28pin tc531000掩膜ROM，长128KB；“美国”SVROM

        /* IC9 处有一个未填充的 PAL16L8，其原始用途（基于
         * 在原理图上）可能会导致 68k 总线错误
@@ -1163,11 +1163,11 @@ ROM_START( 猫 )
         * 其连接为（其中 Ix = 引脚 x 上的输入，Ox = 引脚 x 上的输出）：
         * I1 = A23、I2 = A22、I3 = A2、I4 = R/W、I5 = A5、I6 = FC2、I7 = 接地、
         * I8 = A1，I9 = 接地，I11 = 接地，O16 = /BERR，
- * I14 = REMAP（连接到模拟器“shadow rom”板或未使用时连接到 gnd）
+ * I14 = REMAP（连接到仿真器“shadow ROM”板或未使用时连接到 gnd）
         * 基于这个朋友的输入和输出，几乎（如果不是全部）
```

```
          * 可以创建cat地址空间的开放总线和镜像区域
          * 导致总线错误。REMAP 可能被用来“打开”A00000-A7ffff
- * 影子 rom/ram 区域并使其可写而不会出错。
+ * 影子 ROM/RAM 区域并使其可写而不会出错。
          */
 ROM_END

diff --git a/src/mame/drivers/cave.cpp b/src/mame/drivers/cave.cpp
索引 01567df1cf5..5a4bdc9fe19 100644
---a/src/mame/drivers/cave.cpp
+++ b/src/mame/drivers/cave.cpp
@@ -399,7 +399,7 @@ WRITE16_MEMBER(cave_state::gaia_coin_lsb_w)
 }

 /* - 没有硬币锁定
- - 写入 0xcf00 不应向 eeprom 发送 1 位 */
+ - 写入 0xcf00 不应向 EEPROM 发送 1 位 */
 WRITE16_MEMBER(cave_state::metmqstr_eeprom_msb_w)
 {
        如果（数据&~0xff00）
@@ -4518,7 +4518,7 @@ 注意：
             U82：27C040 EPROM
             PR12*：27C040 EPROM
             程序*：27C040 EPROM
- 所有其他 ROM 均焊接在 16M 42 引脚 MASKROM（读作 27C160）
+ 所有其他 ROM 均焊接在 16M 42 引脚掩模 ROM（读作 27C160）
 */

 ROM_START（传说）
@@ -5059,13 +5059,13 @@ ROM_START( uopoko )
        ROM_LOAD16_BYTE（“u25.int”，0x000001，0x080000，CRC（a1258482）SHA1（7f4adc4a6d069032aaf3d93eb60fde16b59483f8））

        ROM_REGION( 0x400000 * 2, "sprites0", 0 ) /* 精灵：* 2 */
- ROM_LOAD( "cave_cv-02_u33.u33", 0x000000, 0x400000, CRC(5d142ad2) SHA1(f26abcf7a625a322b83df44fbd6e852bfb03663c) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u33.u33", 0x000000, 0x400000, CRC(5d142ad2) SHA1(f26abcf7a625a322b83df44fbd6e852bfb03663c) ) /* 掩码 ROM */

        ROM_REGION( 0x400000, "layer0", 0 ) /* 第 0 层 */
- ROM_LOAD( "cave_cv-02_u49.u49", 0x000000, 0x400000, CRC(12fb11bb) SHA1(953df1b16b5c9a6c3eb2fdebec4669a879270e73) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u49.u49", 0x000000, 0x400000, CRC(12fb11bb) SHA1(953df1b16b5c9a6c3eb2fdebec4669a879270e73) ) /* 掩码 ROM */

        ROM_REGION( 0x200000, "ymz", 0 ) /* 样本 */
- ROM_LOAD( "cave_cv-02_u4.u4", 0x000000, 0x200000, CRC(a2d0d755) SHA1(f8493ef7f367f3dc2a229ba785ac67bc5c2c54c0) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u4.u4", 0x000000, 0x200000, CRC(a2d0d755) SHA1(f8493ef7f367f3dc2a229ba785ac67bc5c2c54c0) ) /* 掩码 ROM */

        ROM_REGION16_BE( 0x80, "eeprom", 0 )
        ROM_LOAD16_WORD（“eeprom-uopoko.bin”，0x0000，0x0080，CRC（f4a24b95）SHA1（4043f0ffed24e38b4f7dbe1a5a4a9e79bdde7dfd））
@@ -5077,13 +5077,13 @@ ROM_START( uopokoj )
```

```
        ROM_LOAD16_BYTE（“u25.bin”，0x000001，0x080000，CRC（68cb6211）SHA1（a6db0bc2e3e54b6992a44b7d52395917e66db49b））

        ROM_REGION( 0x400000 * 2, "sprites0", 0 ) /* 精灵: * 2 */
- ROM_LOAD( "cave_cv-02_u33.u33", 0x000000, 0x400000, CRC(5d142ad2) SHA1(f26abcf7a625a322b83df44fbd6e852bfb03663c) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u33.u33", 0x000000, 0x400000, CRC(5d142ad2) SHA1(f26abcf7a625a322b83df44fbd6e852bfb03663c) ) /* 掩码 ROM */

        ROM_REGION( 0x400000, "layer0", 0 ) /* 第 0 层 */
- ROM_LOAD( "cave_cv-02_u49.u49", 0x000000, 0x400000, CRC(12fb11bb) SHA1(953df1b16b5c9a6c3eb2fdebec4669a879270e73) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u49.u49", 0x000000, 0x400000, CRC(12fb11bb) SHA1(953df1b16b5c9a6c3eb2fdebec4669a879270e73) ) /* 掩码 ROM */

        ROM_REGION( 0x200000, "ymz", 0 ) /* 样本 */
- ROM_LOAD( "cave_cv-02_u4.u4", 0x000000, 0x200000, CRC(a2d0d755) SHA1(f8493ef7f367f3dc2a229ba785ac67bc5c2c54c0) ) /* 掩码ROM */
+ ROM_LOAD( "cave_cv-02_u4.u4", 0x000000, 0x200000, CRC(a2d0d755) SHA1(f8493ef7f367f3dc2a229ba785ac67bc5c2c54c0) ) /* 掩码 ROM */

        ROM_REGION16_BE( 0x80, "eeprom", 0 )
        ROM_LOAD16_WORD（“eeprom-uopoko.bin”，0x0000，0x0080，CRC（f4a24b95）SHA1（4043f0ffed24e38b4f7dbe1a5a4a9e79bdde7dfd））
diff --git a/src/mame/drivers/cischeat.cpp b/src/mame/drivers/cischeat.cpp
索引 3f5a98cc7c5..f4369ee09a8 100644
---a/src/mame/drivers/cischeat.cpp
+++ b/src/mame/drivers/cischeat.cpp
@@ -3424,10 +3424,10 @@ 注：
       ROM -
             PR88004Q_8.IC102 - 82S147 可编程只读存储器
             AC91106_VER1.2_7.IC99 - 27C010 EPROM
- MR91042007-R66_6.IC95 - 4M掩膜ROM
+ MR91042007-R66_6.IC95 - 4M掩膜ROM
             AC91106_VER1.7_4.IC63 - 27C010 EPROM
             AC91106_VER1.7_3.IC62 - 27C010 EPROM
- MR91042-08_2.IC57 - 4M掩膜ROM
+ MR91042-08_2.IC57 - 4M掩膜ROM
             AC91106_VER1.2_1.IC56 - 27C4001 EPROM
             PR91042_5.IC91 - 82S129 可编程只读存储器

@@ -3483,14 +3483,14 @@ 注：
             CH9072-5_11.IC33 - Atmel AT27HC642R
             CH9072-6_12.IC35 - Atmel AT27HC642R
             CH9072-8_15.IC59 - Atmel AT27HC642R
- MR90015-35-W33_14.IC54- 4M掩膜\
- MR90015-35-W33_17.IC67- 4M MASKROM / 包含相同数据
- MR91042-01-R60_1.IC1 - 4M 掩模ROM
- MR91042-02-R61_2.IC2 - 4M掩膜ROM
- MR91042-03-R62_3.IC5 - 4M掩膜ROM
- MR91042-04-R63_4.IC6 - 4M 掩模ROM
- MR91042-05-R64_5.IC11 - 4M掩膜ROM
- MR91042-06-R65_6.IC12 - 4M掩膜ROM
+ MR90015-35-W33_14.IC54- 4M掩膜ROM \
+ MR90015-35-W33_17.IC67- 4M掩膜ROM / 包含相同数据
```

```
 + MR91042-01-R60_1.IC1 - 4M掩膜ROM
 + MR91042-02-R61_2.IC2 - 4M掩膜ROM
 + MR91042-03-R62_3.IC5 - 4M掩膜ROM
 + MR91042-04-R63_4.IC6 - 4M掩膜ROM
 + MR91042-05-R64_5.IC11 - 4M掩膜ROM
 + MR91042-06-R65_6.IC12 - 4M掩膜ROM

  ********************************************************** **************************/

 diff --git a/src/mame/drivers/cps1.cpp b/src/mame/drivers/cps1.cpp
 索引 f3eb46dbb08..aef5530818c 100644
 ---a/src/mame/drivers/cps1.cpp
 +++ b/src/mame/drivers/cps1.cpp
 @@ -5053,7 +5053,7 @@ ROM_START( ffightu )
  ROM_END

  /* B板89624B-3 */
 -/* 请注意，该组与 ffightu 等效，但位于 @ 8H 的 4Mbit MASK ROM FF-32M 被 4 个 1Mbit EPROM 取代。*/
 +/* 请注意，该组相当于 ffightu，但位于 @ 8H 的 4Mbit 掩码 ROM FF-32M 被 4 个 1Mbit EPROM 取代。*/
  ROM_START（fightu1）
         ROM_REGION( CODE_SIZE, "maincpu", 0 ) /* 68000 代码 */
         ROM_LOAD16_BYTE( "ff_36.11f", 0x00000, 0x20000, CRC(f9a5ce83) SHA1(0756ae576a1f6d5b8b22f8630dca40ef38567ea6) ) // 在“30”套接字中
 @@ -7582,7 +7582,7 @@ ROM_START( sf2ebbl )
         ROM_LOAD16_BYTE（“10.bin”，0x80001，0x40000，CRC（0c83844d）SHA1（4c25ba4a50d62c62789d026e3d304ed1dfb3c248））

         ROM_REGION( 0x600000, "gfx", 0 )
 - /* 该 PCB 上的 12 个 MASK ROM 与原始 ROM 完全匹配 */
 + /* 该 PCB 上的 12 个掩模 ROM 与原始 ROM 完全匹配 */
         ROMX_LOAD( "1b_yf082.bin", 0x000000, 0x80000, CRC(22c9cc8e) SHA1(b9194fb337b30502c1c9501cd6c64ae4035544d4) , ROM_GROUPWORD | ROM_SKIP(6) )
         ROMX_LOAD( "1d_yf028.bin", 0x000002, 0x80000, CRC(57213be8) SHA1(3759b851ac0904ec79cbb67a2264d384b6f2f9f9) , ROM_GROUPWORD | ROM_SKIP(6) )
         ROMX_LOAD( "1a_yf087.bin", 0x000004, 0x80000, CRC(ba529b4f) SHA1(520840d727161cf09ca784919fa37bc9b54cc3ce) , ROM_GROUPWORD | ROM_SKIP(6) )
 @@ -7595,7 +7595,7 @@ ROM_START( sf2ebbl )
         ROMX_LOAD( "1l_ye040.bin", 0x400002, 0x80000, CRC(3e66ad9d) SHA1(9af9df0826988872662753e9717c48d46f2974b0) , ROM_GROUPWORD | ROM_SKIP(6) )
         ROMX_LOAD( "1i_yf038.bin", 0x400004, 0x80000, CRC(c1befaa8) SHA1(a6a7f4725e52678cbd8d557285c01cdccb2c2602) , ROM_GROUPWORD | ROM_SKIP(6) )
         ROMX_LOAD( "1k_ye039.bin", 0x400006, 0x80000, CRC(0627c831) SHA1(f9a92d614e8877d648449de2612fc8b43c85e4c2) , ROM_GROUPWORD | ROM_SKIP(6) )
 - /* 这些映射在这个盗版上的 MASK ROM 上，以摆脱 CAPCOM 徽标（浪费，但正确） */
 + /* 这些映射在这个 bootleg 上的掩码 ROM 上，以消除 CAPCOM 徽标（浪费，但正确） */
         ROMX_LOAD( "05.bin", 0x400000, 0x10000, CRC(a505621e) SHA1(8ffa8cedad54948870bbd8f629d927332dc9fcf6) , ROM_SKIP(7) )
         ROM_继续（0x400004，0x10000）
         ROMX_LOAD( "07.bin", 0x400002, 0x10000, CRC(de6271fb) SHA1(574ec5d9992941a405fd00abe52da41aba4b29a7) , ROM_SKIP(7) )
 @@ -7625,7 +7625,7 @@ ROM_START( sf2ebbl2 )
         ROM_LOAD16_BYTE（“27c020.4”，0x80001，0x40000，CRC（0c83844d）SHA1（4c25ba4a50d62c62789d026e3d304ed1dfb3c248））

         ROM_REGION( 0x600000, "gfx", 0 )
 - /* 该 PCB 上的 5 个 MASK ROM 与原始 ROM 完全匹配，*/
 + /* 该 PCB 上的 5 个掩模 ROM 与原始 ROM 完全匹配，*/
         ROMX_LOAD( "a-se235.bin", 0x000000, 0x80000, CRC(a258de13) SHA1(2e477948c4c8a2fb7cfdc4a739766bc4a4e01c49) , ROM_GROUPWORD | ROM_SKIP(6) )
```

```
        ROM_继续（0x000004，0x80000）
        ROMX_LOAD( "c-se005.bin", 0x000002, 0x80000, CRC(c781bf87) SHA1(034baa9807c2ce8dc800200963a38cd9262b21fb) , ROM_GROUPWORD | ROM_SKIP(6) )
@@ -7639,7 +7639,7 @@ ROM_START( sf2ebb12 )
        ROMX_LOAD（“f-sf001.bin”，0x400002，0x80000，CRC（5b585071）SHA1（ad3371b1ba0441c67d9fcbb23b09464710e4e28a），ROM_GROUPWORD | ROM_SKIP
（6））
        ROM_继续（0x400006，0x80000）

- // 这些映射在这个盗版上的 MASK ROM 上，为什么？这不是浪费eprom吗？
+ // 这些映射在这个盗版上的掩模ROM上，为什么？这不是浪费eprom吗？
        ROMX_LOAD( "27c1024.10", 0x400000, 0x20000, CRC(84427d1b) SHA1(f988a2b53c8cc46eeb8032084f24966a539b3734) , ROM_GROUPWORD | ROM_SKIP(6)
)//e-sf004.垃圾箱 [1/8] 相同
        ROMX_LOAD( "27c1024.12", 0x400002, 0x20000, CRC(55bc790c) SHA1(a1114b89f6fa4487210477676984c77ad94b5ef8) , ROM_GROUPWORD | ROM_SKIP(6)
)//f-sf001.bin [ 1/8]相同
        ROMX_LOAD( "27c1024.9", 0x400004, 0x20000, CRC(f8725add) SHA1(fa3fcf6637ee4dd7667bd89766074b3c6ba4f166) , ROM_GROUPWORD | ROM_SKIP(6) )//e-
sf004.bin [5/8]完全相同
@@ -7684,7 +7684,7 @@ ROM_START( sf2ebb13 )
        ROMX_LOAD( "1-i-yf224.03", 0x400004, 0x80000, CRC(c1befaa8) SHA1(a6a7f4725e52678cbd8d557285c01cdccb2c2602) , ROM_GROUPWORD | ROM_SKIP(6) )
        ROMX_LOAD( "1-k-yf036.06", 0x400006, 0x80000, CRC(0627c831) SHA1(f9a92d614e8877d648449de2612fc8b43c85e4c2) , ROM_GROUPWORD | ROM_SKIP(6) )

- /* 这些映射在这个盗版上的 MASK ROM 上，以摆脱 CAPCOM 徽标（浪费，但正确） */
+ /* 这些映射在这个 bootleg 上的掩码 ROM 上，以消除 CAPCOM 徽标（浪费，但正确） */
        ROMX_LOAD( "27010.09hi", 0x400000, 0x10000, CRC(a505621e) SHA1(8ffa8cedad54948870bbd8f629d927332dc9fcf6) , ROM_SKIP(7) )
        ROM_继续（0x400004，0x10000）
        ROMX_LOAD（“27010.11”，0x400002，0x10000，CRC（de6271fb）SHA1（574ec5d9992941a405fd00abe52da41aba4b29a7），ROM_SKIP（7））
@@ -12478,7 +12478,7 @@ ROM_START( pang3b )
 ROM_END

 /* B板91635B-2 */
-/* 请注意，这套美国套装似乎是唯一将 GFX 存储到 EPROM 中而不是通常的 MASK ROM 中的套装。*/
+/* 请注意，这套美国套装似乎是唯一将 GFX 存储到 EPROM 中而不是通常的掩模 ROM 中的套装。*/
 ROM_START（洛克人）
        ROM_REGION( CODE_SIZE, "maincpu", 0 ) /* 68000 代码 */
        ROM_LOAD16_WORD_SWAP（“rcmu_23b.8f”，0x000000，0x80000，CRC（1cd33c7a）SHA1（687fb3b6d660d7350447193f1911c47972e7a020））
@@ -12969,7 +12969,7 @@ READ16_MEMBER(cps_state::sf2dongb_prot_r)

 无效 cps_state::init_sf2dongb()
 {
- // 5f 插槽中有一个被破解的 Altera EP910PC-30 DIP，而不是第 4 个 eprom
+ // 5f 插槽中有一个被破解的 Altera EP910PC-30 DIP，而不是第 4 个 EPROM
        m_maincpu->space(AS_PROGRAM).install_read_handler(0x180000, 0x1fffff, read16_delegate(FUNC(cps_state::sf2dongb_prot_r),this));

        init_cps1();
@@ -13155,7 +13155,7 @@ 游戏( 1988, daimakai, 食尸鬼, cps1_10MHz, daimakai, cps_state, init_cps1,
 GAME( 1988, daimakair, ghouls, cps1_12MHz, daimakai, cps_state, init_cps1, ROT0, "Capcom", "Daimakaimura (Japan Resale Ver.)",
MACHINE_SUPPORTS_SAVE ) // Wed.26.10.1988 // ROM 中 // 12MHz 已验证
 游戏（1989，strider，0，cps1_10MHz，strider，cps_state，init_cps1，ROT0，“Capcom”，“Strider（美国，B-Board 89624B-2）”，
MACHINE_SUPPORTS_SAVE）
```

```
 游戏（1989，striderua，strider，cps1_10MHz，stridrua，cps_state，init_cps1，ROT0，“Capcom”，“Strider（美国，B-Board 89624B-3）”，
MACHINE_SUPPORTS_SAVE）
-GAME( 1989, strideruc, strider, cps1_10MHz, stridrua, cps_state, init_cps1, ROT0, "bootleg (Capcom)", "Strider (USA, B-Board 90629B-3, buggy
Street Fighter II conversion)", MACHINE_SUPPORTS_SAVE ) //甚至 PCB 上也存在各种错误，请参阅 ROM 加载
+GAME( 1989, strideruc, strider, cps1_10MHz, stridrua, cps_state, init_cps1, ROT0, "bootleg (Capcom)", "Strider (USA, B-Board 90629B-3, buggy
Street Fighter II conversion)", MACHINE_SUPPORTS_SAVE ) //即使在 PCB 上也存在各种错误，请参阅 ROM 负载
 游戏（1989，striderj，strider，cps1_10MHz，strider，cps_state，init_cps1，ROT0，“卡普空”，“Strider Hiryu（日本）”，MACHINE_SUPPORTS_SAVE）
 GAME( 1989, striderjr, strider, cps1_12MHz, strider, cps_state, init_cps1, ROT0, "Capcom", "Strider Hiryu (日本转售版)", MACHINE_SUPPORTS_SAVE )
// 12MHz 已验证
 GAME( 1989, dynwar, 0, cps1_10MHz, dynwar, cps_state, init_cps1, ROT0, "Capcom", "Dynasty Wars (USA, B-Board 89624B-?)", MACHINE_SUPPORTS_SAVE )
// (c) Capcom USA
diff --git a/src/mame/drivers/cps2.cpp b/src/mame/drivers/cps2.cpp
索引 277b6d6e4db..27000713ec1 100644
---a/src/mame/drivers/cps2.cpp
+++ b/src/mame/drivers/cps2.cpp
@@ -283,8 +283,8 @@ 注意：

       ROM -
             请注意，上面布局中显示的 ROM 名称是通用的。每个游戏的每个 EPROM 都有
- 附有独特的贴纸。所有的 MASKROM 都标有唯一的名称
- 每场比赛。每个游戏使用的 EPROM/MASKROM 数量也不同，具体取决于
+ 附有独特的贴纸。所有掩膜 ROM 都标有唯一的名称
+ 每场比赛。每个游戏使用的 EPROM/掩模 ROM 数量也不同，具体取决于
             要求。默认情况下，PCB 针对某些尺寸的 ROM 进行了接线，但通过跳线它们
             可以重新配置以允许接受其他尺寸的设备。

@@ -449,7 +449,7 @@ 注意：
                         到目前为止，SIMM 上保存的数据等于或小于 ROM 容量
                         B 板上的插槽，因此 SIMM 的使用是一个谜。
                         一些可能的解释是它们的使用是一种削减成本的措施，或者它们是
- 比使用旧的 42 引脚 MASKROM 更容易从供应商处采购。
+ 比使用旧的 42 引脚掩模 ROM 更容易从供应商处采购。
                         另一种可能性是它们是从剩余的 CPS3 板中重新使用的，因为
                         它们是相同的并且很容易重新编程。相比之下，GFX SIMM
                         与 CPS3 板中用于主程序和 QSound 的类型相同
diff --git a/src/mame/drivers/ddragon3.cpp b/src/mame/drivers/ddragon3.cpp
索引 c7b91052aae..b0997b273fb 100644
---a/src/mame/drivers/ddragon3.cpp
+++ b/src/mame/drivers/ddragon3.cpp
@@ -20,7 +20,7 @@ 双龙3 PCB布局
 ---------------------------

 TA-0030-P1-03（带有 EPROM 的早期版本）
-TA-0030-P1-04（带有 MASKROM 的更高版本）
+TA-0030-P1-04（带有掩模ROM的更高版本）
 |-------------------------------------------------------|
 |VOL M51516 YM3012 YM2151 3.579545MHz IC15 |
```

```
  | MB3615 1.056MHz Z80 IC14|-|
 diff --git a/src/mame/drivers/dec8.cpp b/src/mame/drivers/dec8.cpp
 索引 710e5bbe002..4bfe058850a 100644
 ---a/src/mame/drivers/dec8.cpp
 +++ b/src/mame/drivers/dec8.cpp
 @@ -3173,7 +3173,7 @@ CPU PCB 下有一个小背驮，里面装满了 74Sxx

   该板看起来像是 Data East 的合法 PCB，即使没有 Data East 徽标。

 -所有记忆都是面具!
 +所有记忆都是面具ROM!

   */
   ROM_START( 明久哈 )
 diff --git a/src/mame/drivers/deco_mlc.cpp b/src/mame/drivers/deco_mlc.cpp
 索引 ac76f7d9094..0a40189fa08 100644
 ---a/src/mame/drivers/deco_mlc.cpp
 +++ b/src/mame/drivers/deco_mlc.cpp
 @@ -603,7 +603,7 @@ 定制DE156加密CPU。
   笔记：
         - SH2 (QFP144) 时钟：21.000MHz (42 / 2)
         - 所有 ROM SD* 均为 4M x 16 位 EPROMS (27C4096)
 - - 所有 MCG* ROM 均为表面贴装 16M MASK ROM
 + - 所有 MCG* ROM 均为表面贴装 16M 掩膜 ROM
         -（mcg-01.1d 以 8 位模式读取，因为该 ROM 在以 16 位模式读取时具有固定位
           模式下，以 8 位读取，读取效果良好。其他以16位模式读取）
   */
 diff --git a/src/mame/drivers/djboy.cpp b/src/mame/drivers/djboy.cpp
 索引 ab85f21956f..8a49c17479d 100644
 ---a/src/mame/drivers/djboy.cpp
 +++ b/src/mame/drivers/djboy.cpp
 @@ -74,16 +74,16 @@ 注意：
                  BS15.6Y 27C512 EPROM (DIP28) \ 有一个替代品。用于这些带有“S”的 ROM 的标签集
                  BS07.1B 27C512 EPROM（DIP28）| 添加到名称中（即“BS15S”），但实际的 ROM 内容是相同的
                  BS19.4B 27C1001 EPROM (DIP32) / 常规设置（两组均已转储/验证）
 - BS-000.1H 4M 掩模ROM (DIP32) {精灵}
 - BS-001.1F 4M MASKROM (DIP32) {精灵}
 - BS-002.1D 4M MASKROM (DIP32) {精灵}
 - BS-003.1K 4M MASKROM (DIP32) {精灵}
 - BS-004.1S 4M MASKROM (DIP32) {瓷砖}
 - BS-005.1U 4M MASKROM (DIP32) {瓷砖}
 - BS-100.4D 1M 掩模ROM (DIP28) {z80}
 - BS-101.6W 1M MASKROM (DIP28) {z80 数据}
 - BS-200.8C 1M 掩模只读存储器 (DIP28) {z80}
 - BS-203.5J 2M MASKROM (DIP32) {oki-m6295 样品}
 + BS-000.1H 4M 掩模 ROM (DIP32) {精灵}
 + BS-001.1F 4M 掩模 ROM (DIP32) {精灵}
```

```
+ BS-002.1D 4M 掩模 ROM (DIP32) {精灵}
+ BS-003.1K 4M 掩模 ROM (DIP32) {精灵}
+ BS-004.1S 4M 掩模 ROM (DIP32) {瓷砖}
+ BS-005.1U 4M 掩模 ROM (DIP32) {瓷砖}
+ BS-100.4D 1M 掩模 ROM (DIP28) {Z80}
+ BS-101.6W 1M 掩模 ROM (DIP28) {Z80 数据}
+ BS-200.8C 1M 掩模 ROM (DIP28) {Z80}
+ BS-203.5J 2M 掩模 ROM (DIP32) {OKI-M6295 样品}

       DIP - SW1
                |-----------------------------------------------------|
diff --git a/src/mame/drivers/dynax.cpp b/src/mame/drivers/dynax.cpp
索引 f1e40db1b14..6ddfa35fc5b 100644
---a/src/mame/drivers/dynax.cpp
+++ b/src/mame/drivers/dynax.cpp
@@ -6797,8 +6797,8 @@ 注：
       ROM - 文件名设备
                      ---------------------------------------------- --
                      TAICOM-00.2C - ST M27C2001 256K x8 EPROM (DIP32)
- TAICOM-01.15B - 4MBit MASKROM (DIP32)
- TAICOM-02.11B - 4MBit MASKROM (DIP32)
+ TAICOM-01.15B - 4MBit 掩模 ROM (DIP32)
+ TAICOM-02.11B - 4MBit 掩模 ROM (DIP32)
                      TAICOM-03.13B - AMD AM27C040 512K x8 EPROM (DIP32)
                      TMP91P640N-10.5B - MCU 的内部 16K ROM

diff --git a/src/mame/drivers/electron.cpp b/src/mame/drivers/electron.cpp
索引 be07cc1b85e..41e13784c7b 100644
---a/src/mame/drivers/electron.cpp
+++ b/src/mame/drivers/electron.cpp
@@ -36,7 +36,7 @@ 注：（显示所有 IC。仅使用 16 个 IC）
             早期的 PCB 修订版在插座中使用了 PLCC68 芯片。后来的修订使用了
             PGA68芯片直接焊接到主板上
      4164 - 4164 64k x4 位 DRAM（4 个芯片，总共 32kbytes）
- ROM - Hitachi HN613256 32k x8 位 MASK ROM 包含操作系统和 BASIC
+ ROM - Hitachi HN613256 32k x8 位掩码 ROM，包含操作系统和 BASIC
     LM324 - 德州仪器 LM324 运算放大器
       MOD - UHF 电视调制器 UM1233-E36
      CVBS——复合彩色视频输出插座
diff --git a/src/mame/drivers/esd16.cpp b/src/mame/drivers/esd16.cpp
索引 9e5764d5dcf..056b535dbb7 100644
---a/src/mame/drivers/esd16.cpp
+++ b/src/mame/drivers/esd16.cpp
@@ -1573,10 +1573,10 @@ 注：
               文件名类型用途
               ---------------------------------------------- ------------------------
               68K_PRG.BIN 日立 HN27C4096 256K x16 EPROM 68000 程序
```

```
 - Z80_PRG.BIN Atmel AT27C020 256K x8 OTP MASKROM Z80 程序
 - SAMPLES.BIN Atmel AT27C020 256K x8 OTP MASKROM Oki M6295 样品
 - BG0/1.BIN Macronix 29F8100MC 1M x8 SOP44 FlashROM 背景图形
 - SP0/1.BIN Macronix 29F8100MC 1M x8 SOP44 FlashROM Sprite 图形
 + Z80_PRG.BIN Atmel AT27C020 256K x8 OTP 掩模 ROM Z80 程序
 + SAMPLES.BIN Atmel AT27C020 256K x8 OTP 掩模 ROM Oki M6295 样品
 + BG0/1.BIN Macronix 29F8100MC 1M x8 SOP44 闪存 ROM 背景图形
 + SP0/1.BIN Macronix 29F8100MC 1M x8 SOP44 闪存 ROM Sprite 图形

                请注意，PCB 上没有 IC 位置，因此 ROM 的扩展名只是“BIN”

 diff --git a/src/mame/drivers/fcrash.cpp b/src/mame/drivers/fcrash.cpp
 索引 673c11806ef..814f246fc2f 100644
 ---a/src/mame/drivers/fcrash.cpp
 +++ b/src/mame/drivers/fcrash.cpp
 @@ -25,10 +25,10 @@ 最终崩溃（最终战斗的盗版）
  1x 振荡器 10mhz
  1x 振荡器 24MHz

 -EPROM:
 -1.bin声音eprom
 -从2.bin到9.bin程序eprom
 -10.bin 至 25.bin gfx eprom
 +EPROM:
 +1.bin声音EPROM
 +从 2.bin 到 9.bin 程序 EPROM
 +10.bin 至 25.bin 图形 EPROM

  ---

 @@ -64,7 +64,7 @@ cawingb2，cawingbl：好的

  迪诺皮克：没有声音

 -dinopic2：没有声音，一个坏的 gfx rom。从 dinopic 复制 8.bin 可以修复该问题。
 +dinopic2：没有声音，图形 ROM 损坏。从 dinopic 复制 8.bin 可以修复该问题。

  fcrash，kodb：旧的精灵显示在下一个屏幕上。使用了补丁。

 @@ -2048,7 +2048,7 @@ ROM
  1x AM27C020 (2)（声音）
  2x AM27C040 (3,4)（主）
  1x Am27C040 (bp)(gfx)
 -7x maskrom (ai,bi,ci,di,ap,cp,dp)(gfx)
 +7x 掩模 ROM (ai,bi,ci,di,ap,cp,dp)(gfx)
  1x GAL20V8A（未倾倒）
  3x GAL16V8A（未倾倒）
```

```
 1x PALCE20V8H（未倾倒）
@@ -2126,7 +2126,7 @@ CPU:
 只读存储器

 5x M27C2001 1,2,3,4,5 已倾销
-4x maskrom KA,KB,KC,KD 未转储
+4x 掩模 ROM KA、KB、KC、KD 未转储

 内存：

@@ -2149,7 +2149,7 @@ 注：

 */
 /* 盗版 */
-/* FIXME - GFX ROM 错误，从其他版本复制 */
+/* FIXME - 图形 ROM 错误，从其他版本复制 */
 /* 缺少的 ROM 为 KA.IC91 KB.IC92 KC.IC93 KD.IC94 */
 ROM_START（骑士b）
        ROM_REGION( CODE_SIZE, "maincpu", 0 ) /* 68000 代码 */
@@ -2158,7 +2158,7 @@ ROM_START( 骑士b )
        ROM_LOAD16_BYTE（“2.ic175”，0x80001，0x40000，CRC（1eb91343）SHA1（e02cfbbd7689346f14f2e3455ed17e7f0b51bad0））
        ROM_LOAD16_BYTE（“4.ic176”，0x80000，0x40000，CRC（af352703）SHA1（7855ac65752203f45af4ef41af8c291540a1c8a8））

- ROM_REGION( 0x400000, "gfx", 0 ) /* bootleg 有 4x 1meg MASKrom，这些需要转储以便知道格式 */
+ ROM_REGION( 0x400000, "gfx", 0 ) /* bootleg 有 4x 1meg mask ROM，这些需要转储以便知道格式 */
       ROMX_LOAD（“kr_gfx1.rom”，0x000000，0x80000，BAD_DUMP CRC（9e36c1a4）SHA1（772daae74e119371dfb76fde9775bda78a8ba125），ROM_GROUPWORD |
ROM_SKIP（6））
       ROMX_LOAD（“kr_gfx3.rom”，0x000002，0x80000，BAD_DUMP CRC（c5832cae）SHA1（a188cf401cd3a2909b377d3059f14d22ec3b0643），ROM_GROUPWORD |
ROM_SKIP（6））
       ROMX_LOAD（“kr_gfx2.rom”，0x000004，0x80000，BAD_DUMP CRC（f095be2d）SHA1（0427d1574062f277a9d04440019d5638b05de561），ROM_GROUPWORD |
ROM_SKIP（6））
@@ -2311,7 +2311,7 @@ ROM_START( 迪诺皮克 )
        ROM_LOAD16_BYTE（“2.bin”，0x100001，0x80000，CRC（0e4058ba）SHA1（346f9e34ea53dd1bf5cdafa1e38bf2edb09b9a7f））
        ROM_LOAD16_BYTE( "7.bin", 0x100000, 0x80000, CRC(6133f349) SHA1(d13af99910623f62c090d25372a2253dbc2f8cbe) )

- ROM_REGION( 0x400000, "gfx", 0 ) // 相同的数据，不同的格式，除了 8 是 99% 匹配（坏的 rom？）
+ ROM_REGION( 0x400000, "gfx", 0 ) // 相同的数据，不同的格式，除了 8 是 99% 匹配（坏 ROM？）
        ROMX_LOAD( "4.bin", 0x000000, 0x40000, CRC(f3c2c98d) SHA1(98ae51a67fa4159456a4a205eebdd8d1775888d1), ROM_SKIP(7) )
        ROM_继续（0x000004，0x40000）
        ROMX_LOAD( "8.bin", 0x000001, 0x40000, CRC(d574befc) SHA1(56482e7a9aa8439f30e3cf72311495ce677a083d), ROM_SKIP(7) )
@@ -2639,7 +2639,7 @@ ROM_START( 布尼皮克 )
        ROM_LOAD( "声音.bin", 0x000000, 0x80000, CRC(aeec9dc6) SHA1(56fd62e8db8aa96cdd242d8c705849a413567780) )
 ROM_END

-/* alt bootleg with PIC, 与上面相同的程序 rom，更大的 GFX rom
+/* alt bootleg with PIC, 与上面相同的程序 ROM，更大的图形 ROM
```

```
 惩罚者
 1993年，卡普空
@@ -2672,14 +2672,14 @@ 注：
        ----------------
        68000时钟：12.000MHz（24/2）
        M6295时钟：937.5kHz（30 / 32），采样率 = 30000000 / 32 / 132
- 16C57 时钟：3.75MHz (30 / 8) 注意！4096字节内部ROM受到保护，无法读取。
+ 16C57 时钟：3.75MHz (30 / 8) 注意！4096字节内部ROM受到保护，无法读取。
        垂直同步：60Hz

        只读存储器
        ----
- PRG* - 4M掩膜ROM（读为27C040）
- 声音 - 4M 掩模 ROM（读为 27C040）
- PU* - 16M掩膜ROM（读为27C160）
+ PRG* - 4M掩膜ROM（读为27C040）
+ 声音 - 4M 掩模 ROM（读为 27C040）
+ PU* - 16M掩膜ROM（读为27C160）

 */

@@ -2710,7 +2710,7 @@ ROM_START( punipic2 )
         ROM_LOAD（“93c46.bin”，0x00，0x80，CRC（36ab4e7d）SHA1（60bea43051d86d9aefcbb7a390cf0c7d8b905a4b））
 ROM_END

-/* 自述文件实际上并没有说明它有 PIC，并且没有声音 ROM
+/* 自述文件实际上并没有说明它有 PIC，并且没有声音 ROM
    所以可能会有所不同*/

 ROM_START( punipic3 )
@@ -3325,7 +3325,7 @@ 游戏( 1993, dinopic2, 迪诺, 迪诺皮克, 迪诺, cps_state, init_dinopic,

 游戏（1990，fcrash，ffight，fcrash，fcrash，cps_state，init_cps1，ROT0，“盗版（Playmark）”，“最终崩溃（最终战斗的盗版）”，
MACHINE_SUPPORTS_SAVE）
 游戏（1990，ffightbl，ffight，fcrash，fcrash，cps_state，init_cps1，ROT0，“盗版”，“最终之战（盗版）”，MACHINE_SUPPORTS_SAVE）
-GAME( 1990, ffightbla, ffight, fcrash, fcrash, cps_state, init_cps1, ROT0, "bootleg", "Final Fight (bootleg on Final Crash PCB)",
MACHINE_SUPPORTS_SAVE ) // 与没有修改 gfx 的 Final Crash 相同
+GAME( 1990, ffightbla, ffight, fcrash, fcrash, cps_state, init_cps1, ROT0, "bootleg", "Final Fight (bootleg on Final Crash PCB)",
MACHINE_SUPPORTS_SAVE ) // 与没有修改图形的 Final Crash 相同

 GAME( 1991, kodb, kod, kodb, kodb, cps_state, init_kodb, ROT0, "盗版 (Playmark)", "龙王 (盗版)", MACHINE_IMPERFECT_GRAPHICS |
MACHINE_SUPPORTS_SAVE ) // 910731 "ETC"

@@ -3351,6 +3351,6 @@ 游戏( 1992, sf2m9, sf2ce, sf2m1, sf2, cps_state, init_sf2m1,
 游戏( 1993, slampic, slammast, slampic, slammast, cps_state, init_dinopic, ROT0, "盗版", "周六夜大满贯大师 (盗版PIC16c57)",
MACHINE_IMPERFECT_GRAPHICS | MACHINE_NO_SOUND | MACHINE_SUPPORTS_SAVE ) // 930713 E TC
```

```
 GAME( 1999, sgyxz, wof, sgyxz, sgyxz, cps_state, init_cps1, ROT0, "盗版 (All-In Electronic)", "命运战士 ('sgyxz' 盗版)",
MACHINE_IMPERFECT_GRAPHICS | MACHINE_SUPPORTS_SAVE ) // 921005 - 三国志 2
-game（1999，wofabl，wof，wofabl，wofabl，cps_state，init_wofabl，rot0，“ bootleg”，“ sangokushi ii（bootleg）”，machine_not_working |
machine_imperfect_grect_graphics
+GAME( 1999, wofabl, wof, wofabl, wofabl, cps_state, init_wofabl, ROT0, "bootleg", "Sangokushi II (bootleg)", MACHINE_NOT_WORKING |
MACHINE_IMPERFECT_GRAPHICS | MACHINE_SUPPORTS_SAVE ) // 严重的图形故障 - 921005 - 三国志 2

 游戏（1992，varthb，varth，varthb，varth，cps_state，init_dinopic，ROT270，“盗版”，“Varth：雷暴行动（盗版）”，MACHINE_SUPPORTS_SAVE）
diff --git a/src/mame/drivers/gcpinbal.cpp b/src/mame/drivers/gcpinbal.cpp
索引 c866984fa59..46e435851ac 100644
---a/src/mame/drivers/gcpinbal.cpp
+++ b/src/mame/drivers/gcpinbal.cpp
@@ -76,7 +76,7 @@ 定制：优秀系统 ES-9208 347102 (QFP160)

 * 表示未填充的组件

-注意：来自 Power Flipper Pinball Shooting 的面具 ROM 尚未被丢弃，但假设已被丢弃
+注意：来自 Power Flipper Pinball Shooting 的 Mask ROM 尚未被转储，但假设已
       是相同的数据。

 ********************************************** *************************/
@@ -419,20 +419,20 @@ ROM_START( pwrflip ) /* Grand Cross Pinball 的更新版本还是半续集？*
        ROM_LOAD16_WORD_SWAP（“pfu46”，0x180000，0x80000，CRC（e0f3a1b4）SHA1（761dddf374a92c1a1e4a211ead215d5be461a082））

        ROM_REGION( 0x200000, "bg0", 0 ) /* BG0 (16 x 16) */
- ROM_LOAD16_WORD_SWAP( "u1", 0x000000, 0x100000, CRC(afa459bb) SHA1(7a7c64bcb80d71b8cf3fdd3209ef109997b6417c) ) /* 23C8000 MASK ROM */
+ ROM_LOAD16_WORD_SWAP( "u1", 0x000000, 0x100000, CRC(afa459bb) SHA1(7a7c64bcb80d71b8cf3fdd3209ef109997b6417c) ) /* 23C8000 掩码 ROM */
        ROM_LOAD16_WORD_SWAP（“u6”，0x100000，0x100000，CRC（c3f024e5）SHA1（d197e2b715b154fc64ff9a61f8c6df111d6fd446））

        ROM_REGION( 0x020000, "fg0", 0 ) /* FG0 (8 x 8) */
        ROM_LOAD16_WORD_SWAP（“pfu10”，0x000000，0x020000，CRC（50e34549）SHA1（ca1808513ff3feb8bcd34d9aafd7b374e4244732））

        ROM_REGION( 0x200000, "sprite", 0 ) /* 精灵 (16 x 16) */
- ROM_LOAD16_WORD_SWAP( "u13", 0x000000, 0x200000, CRC(62f3952f) SHA1(7dc9ccb753d46b6aaa791bcbf6e18e6d872f6b79) ) /* 23C16000 掩码ROM */
+ ROM_LOAD16_WORD_SWAP( "u13", 0x000000, 0x200000, CRC(62f3952f) SHA1(7dc9ccb753d46b6aaa791bcbf6e18e6d872f6b79) ) /* 23C16000掩膜ROM */

        ROM_REGION( 0x080000, "oki", 0 ) /* M6295 符合 Raine */
- ROM_LOAD( "u55", 0x000000, 0x080000, CRC(b3063351) SHA1(825e63e8a824d67d235178897528e5b0b41e4485) ) /* OKI M534001B 掩膜ROM */
+ ROM_LOAD( "u55", 0x000000, 0x080000, CRC(b3063351) SHA1(825e63e8a824d67d235178897528e5b0b41e4485) ) /* OKI M534001B掩膜ROM */

        ROM_REGION( 0x200000, "essnd", 0 ) /* M6585 符合 Raine 但应该适用于 ES-8712？？？*/
- ROM_LOAD( "u56", 0x000000, 0x200000, CRC(092b2c0f) SHA1(2ec1904e473ddddb50dbeaa0b561642064d45336) ) /* 23C16000 掩码ROM */
+ ROM_LOAD( "u56", 0x000000, 0x200000, CRC(092b2c0f) SHA1(2ec1904e473ddddb50dbeaa0b561642064d45336) ) /* 23C16000 掩码 ROM */

        ROM_REGION( 0x000400, "plds", 0 ) /* 2x TIBPAL16L8-15CN */
        ROM_LOAD（“a.u72”，0x000，0x104，NO_DUMP）
@@ -447,20 +447,20 @@ ROM_START( gcpinbal )
```

```
        ROM_LOAD16_WORD_SWAP( "4_excellent.u46", 0x180000, 0x80000, CRC(e0f3a1b4) SHA1(761dddf374a92c1a1e4a211ead215d5be461a082) )

        ROM_REGION( 0x200000, "bg0", 0 ) /* BG0 (16 x 16) */
- ROM_LOAD16_WORD_SWAP( "u1", 0x000000, 0x100000, CRC(afa459bb) SHA1(7a7c64bcb80d71b8cf3fdd3209ef109997b6417c) ) /* 23C8000 MASK ROM */
+ ROM_LOAD16_WORD_SWAP( "u1", 0x000000, 0x100000, CRC(afa459bb) SHA1(7a7c64bcb80d71b8cf3fdd3209ef109997b6417c) ) /* 23C8000 掩码 ROM */
        ROM_LOAD16_WORD_SWAP（“u6”，0x100000，0x100000，CRC（c3f024e5）SHA1（d197e2b715b154fc64ff9a61f8c6df111d6fd446））

        ROM_REGION( 0x020000, "fg0", 0 ) /* FG0 (8 x 8) */
        ROM_LOAD16_WORD_SWAP( "1_excellent.u10", 0x000000, 0x020000, CRC(79321550) SHA1(61f1b772ed8cf95bfee9df8394b0c3ff727e8702) )

        ROM_REGION( 0x200000, "sprite", 0 ) /* 精灵 (16 x 16) */
- ROM_LOAD16_WORD_SWAP( "u13", 0x000000, 0x200000, CRC(62f3952f) SHA1(7dc9ccb753d46b6aaa791bcbf6e18e6d872f6b79) ) /* 23C16000 掩码ROM */
+ ROM_LOAD16_WORD_SWAP( "u13", 0x000000, 0x200000, CRC(62f3952f) SHA1(7dc9ccb753d46b6aaa791bcbf6e18e6d872f6b79) ) /* 23C16000掩膜ROM */

        ROM_REGION( 0x080000, "oki", 0 ) /* M6295 符合 Raine */
- ROM_LOAD( "u55", 0x000000, 0x080000, CRC(b3063351) SHA1(825e63e8a824d67d235178897528e5b0b41e4485) ) /* OKI M534001B 掩膜ROM */
+ ROM_LOAD( "u55", 0x000000, 0x080000, CRC(b3063351) SHA1(825e63e8a824d67d235178897528e5b0b41e4485) ) /* OKI M534001B掩膜ROM */

        ROM_REGION( 0x200000, "essnd", 0 ) /* M6585 符合 Raine 但应该适用于 ES-8712? ? ? */
- ROM_LOAD( "u56", 0x000000, 0x200000, CRC(092b2c0f) SHA1(2ec1904e473ddddb50dbeaa0b561642064d45336) ) /* 23C16000 掩码ROM */
+ ROM_LOAD( "u56", 0x000000, 0x200000, CRC(092b2c0f) SHA1(2ec1904e473ddddb50dbeaa0b561642064d45336) ) /* 23C16000 掩码 ROM */

        ROM_REGION( 0x000400, "plds", 0 ) /* 2x TIBPAL16L8-15CN */
        ROM_LOAD（“a.u72”，0x000，0x104，NO_DUMP）
diff --git a/src/mame/drivers/hikaru.cpp b/src/mame/drivers/hikaru.cpp
索引 633e05ff366..166fd2625aa 100644
---a/src/mame/drivers/hikaru.cpp
+++ b/src/mame/drivers/hikaru.cpp
@@ -110,7 +110,7 @@ ROM 使用 - CRC（来自 ROM 测试
            游戏世嘉零件号 ROM 类型 字节 字
            ------------------------------------------------ ------------
            空气特里克斯 -
- MPR-23573.IC37 128M TSOP48 掩模ROM B9A5 9E67
+ MPR-23573.IC37 128M TSOP48 掩模 ROM B9A5 9E67
                       MPR-23577.IC38“A52A BCE0
                       MPR-23574.IC41“DABB B621
                       MPR-23578.IC42“4BD4 5E6B
@@ -133,7 +133,7 @@ ROM 使用 - CRC（来自 ROM 测试
            游戏世嘉零件号 ROM 类型 字节 字
            ------------------------------------------------ ------------
            行星鹞 -
- MPR-23549.IC37 128M TSOP48 掩模ROM 7F16 2C37
+ MPR-23549.IC37 128M TSOP48掩膜ROM 7F16 2C37
                       MPR-23553.IC38“1F9F AAE5
                       MPR-23550.IC41“986C 8D7A
                       MPR-23554.IC42“BD1D 5304
@@ -186,7 +186,7 @@ ROM 使用 - CRC（来自 ROM 测试
```

```
            游戏世嘉零件号 ROM 类型 字节 字
            ------------------------------------------------ ------------
            星球大战赛车街机
- MPR-23086.IC37 64M SOP44 掩模ROM 7993 8E18
+ MPR-23086.IC37 64M SOP44掩膜ROM 7993 8E18
                        MPR-23087.IC38“4D44 D239
                        MPR-23088.IC39“4135 BEAB
                        MPR-23089.IC40“F0C8 04E2
@@ -583,7 +583,7 @@ ROM_START（airtrix）
        /* ic35 未填充 */
        /* ic36 未填充 */

- /* 使用 128M TSOP48 MASKROM 的 ROM 板 */
+ /* ROM板使用128M TSOP48掩模ROM */
        ROM_REGION( 0x10000000, "用户2", 0)
        ROM_LOAD32_WORD（“mpr-23573.ic37”，0x0000000，0x1000000，CRC（e22a0734）SHA1（fc06d5972d285d09473874aaeb1efed2d19c8f36））
        ROM_LOAD32_WORD（“mpr-23577.ic38”，0x0000002，0x1000000，CRC（d007680d）SHA1（a795057c40b1851adb0e19e5dfb39e16206215bf））
@@ -614,7 +614,7 @@ ROM_START（airtrixo）
        ROM_LOAD32_WORD（“epr-23601.ic29”，0x0000000，0x0400000，CRC（e0c642cb）SHA1（f04f8e13cc46d462c79ecebcded7dee9b3500bdc））
        ROM_LOAD32_WORD（“epr-23602.ic30”，0x0000002，0x0400000，CRC（fac11d21）SHA1（70b48a7e1ac4268fc09d96d6845c5a5099d4e301））

- /* 使用 128M TSOP48 MASKROM 的 ROM 板 */
+ /* ROM板使用128M TSOP48掩模ROM */
        ROM_REGION( 0x10000000, "用户2", 0)
        ROM_LOAD32_WORD（“mpr-23573.ic37”，0x0000000，0x1000000，CRC（e22a0734）SHA1（fc06d5972d285d09473874aaeb1efed2d19c8f36））
        ROM_LOAD32_WORD（“mpr-23577.ic38”，0x0000002，0x1000000，CRC（d007680d）SHA1（a795057c40b1851adb0e19e5dfb39e16206215bf））
@@ -644,7 +644,7 @@ ROM_START( 法里尔 )
        ROM_LOAD32_WORD("epr-23571.ic35", 0x1800000, 0x0400000, CRC(5a75fa92) SHA1(b5e0c8c995ecc954b74d5eb36f3ae2a732a5986b) )
        ROM_LOAD32_WORD（“epr-23572.ic36”，0x1800002，0x0400000，CRC（46054067）SHA1（449800bdc2c40c76aed9bc5e7e8831d8f03ef286））

- /* 使用 128M TSOP48 MASKROM 的 ROM 板 */
+ /* ROM板使用128M TSOP48掩模ROM */
        ROM_REGION( 0x10000000, "用户2", 0)
        ROM_LOAD32_WORD（“mpr-23549.ic37”，0x0000000，0x1000000，CRC（ed764200）SHA1（ad840a40347345f72a443f284b1bb0ae2b37f7ac））
        ROM_LOAD32_WORD（“mpr-23553.ic38”，0x0000002，0x1000000，CRC（5e70ae78）SHA1（2ae6bdb5aa1434bb60b2b9bca7af12d6476cd35f））
@@ -681,7 +681,7 @@ ROM_START( swracer )
        /* ic35 未填充 */
        /* ic36 未填充 */

- /* ROM板使用64M SOP44 MASKROM */
+ /* ROM板使用64M SOP44掩膜ROM */
        ROM_REGION( 0x10000000, "用户2", 0)
        ROM_LOAD32_WORD（“mpr-23086.ic37”，0x0000000，0x0800000，CRC（ef6f20f1）SHA1（11fb66bf71223b4c6650d3adaea21e8709b8d67b））
        ROM_LOAD32_WORD（“mpr-23087.ic38”，0x0000002，0x0800000，CRC（54389822）SHA1（6357f0aa77ef0a5a08a751e085fa026d26ba47d1））
@@ -734,7 +734,7 @@ ROM_START( 勇敢 )
        /* ic35 未填充 */
        /* ic36 未填充 */
```

```
- /* ROM板使用64M SOP44 MASKROM */
+ /* ROM板使用64M SOP44掩膜ROM */
        ROM_REGION（0xc000000, “用户2”, ROMREGION_ERASE00）
        ROM_LOAD32_WORD（“mpr-22000.ic37”, 0x0000000, 0x800000, CRC（53d641d6）SHA1（f47d7c77d0e36c4ec3b7171fd7a017f9f58ca5a0））
        ROM_LOAD32_WORD（“mpr-22001.ic38”, 0x0000002, 0x800000, CRC（234bc48f）SHA1（177c46884de0ba4bac1f9b778f99c905410a9345））
@@ -772,7 +772,7 @@ ROM_START( sgnascar )
        ROM_LOAD32_WORD（“epr-23485a.ic35”, 0x000000, 0x400000, CRC（1072f531）SHA1（ca07a8bfb7247e4aec57e18cb091d24dcef666c1））
        ROM_LOAD32_WORD（“epr-23486a.ic36”, 0x000002, 0x400000, CRC（02d4aab6）SHA1（b1b0e07dc71dc124177e27dfd8b459444e8ae4d3））

- /* 使用 128M TSOP48 MASKROM 的 ROM 板 */
+ /* ROM板使用128M TSOP48掩模ROM */
        ROM_REGION（0x10000000, “用户2”, ROMREGION_ERASE00）
        ROM_LOAD32_WORD（“mpr-23469.ic19”, 0x0000000, 0x1000000, CRC（89cbad8d）SHA1（e4f103b96a3a842a90182172ddcf3bc5dfe6cca8））
        ROM_LOAD32_WORD（“mpr-23473.ic20”, 0x0000002, 0x1000000, CRC（977b87d6）SHA1（079eeebc6f9c60d0a016a46386bbe846d8a354da））
@@ -804,7 +804,7 @@ ROM_START( sgnascaro )
        ROM_LOAD32_WORD（“epr-23485.ic35”, 0x000000, 0x400000, CRC（13b44fbf）SHA1（73416fa7b671ec5c96f0b084a427ff701bf6c399））
        ROM_LOAD32_WORD（“epr-23486.ic36”, 0x000002, 0x400000, CRC（ac3acd19）SHA1（1ec96be0bfceb2f1f808d78b07425d32056fbde0））

- /* 使用 128M TSOP48 MASKROM 的 ROM 板 */
+ /* ROM板使用128M TSOP48掩模ROM */
        ROM_REGION（0x10000000, “用户2”, ROMREGION_ERASE00）
        ROM_LOAD32_WORD（“mpr-23469.ic19”, 0x0000000, 0x1000000, CRC（89cbad8d）SHA1（e4f103b96a3a842a90182172ddcf3bc5dfe6cca8））
        ROM_LOAD32_WORD（“mpr-23473.ic20”, 0x0000002, 0x1000000, CRC（977b87d6）SHA1（079eeebc6f9c60d0a016a46386bbe846d8a354da））
diff --git a/src/mame/drivers/hp_ipc.cpp b/src/mame/drivers/hp_ipc.cpp
索引 8684662ef3d..a8c503788da 100644
---a/src/mame/drivers/hp_ipc.cpp
+++ b/src/mame/drivers/hp_ipc.cpp
@@ -210,7 +210,7 @@ ROM 板（操作系统ROM PCA。组件# HP82991A 或 HP82995A）
 笔记：
       J1/J2 - 连接到“Option ROM PCA”的 20 针连接器
       J3/J4 - 连接到“LOGIC A PCA”的 20 针连接器
- U1-U4 - 28 引脚 EPROM/MASKROM 0L/1L/0H/1H（注 1Mbit：128Kx8 28 引脚）
+ U1-U4 - 28 pin EPROM/mask ROM 0L/1L/0H/1H (注 1Mbit: 128Kx8 28 pin)


 ROM 板（选项 ROM PCA）
@@ -229,7 +229,7 @@ 请注意，此 PCB 颠倒插入操作系统 ROM PCB 的顶部
 笔记：
       J1/J2 - 连接到“操作系统 ROM PCA”的 20 针连接器
       J3/J4 - 连接到“LOGIC A PCA”的 20 针连接器
- U1-U4 - 28 引脚 EPROM/MASKROM 0L/1L/0H/1H（注 1Mbit：128Kx8 28 引脚）
+ U1-U4 - 28 pin EPROM/mask ROM 0L/1L/0H/1H (注 1Mbit: 128Kx8 28 pin)


 物理内存映射
diff --git a/src/mame/drivers/itech32.cpp b/src/mame/drivers/itech32.cpp
```

```
索引 cb22dd9ff38..b0605d43815 100644
---a/src/mame/drivers/itech32.cpp
+++ b/src/mame/drivers/itech32.cpp
@@ -2871,7 +2871,7 @@ ROM_START( sftm ) /* 版本 1.12, P/N 1064 REV 1 主板, P/N 1073 REV 0 Ro
        ROM_LOAD32_BYTE (“rm1-2.grm1_2”, 0x1000002, 0x400000, CRC (903e56c2) SHA1 (843ed9855ffdf37b100b3c5614139d552fd9cd6d) )
        ROM_LOAD32_BYTE (“rm1-3.grm1_3”, 0x1000003, 0x400000, CRC (fac35686) SHA1 (ba99ab265620575c14c46806dc543d1f9fd24462) )

- /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */
+ /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */

        ROM_LOAD32_BYTE( "sfm_grm3_0.grm3_0", 0x2000000, 0x020000, CRC(3e1f76f7) SHA1(8aefe376e7248a583a6af02e5f9b2a4b48cc91d7) )
        ROM_LOAD32_BYTE (“sfm_grm3_1.grm3_1”, 0x2000001, 0x020000, CRC (578054b6) SHA1 (99201959de28dbfd7692cedea4485751d3d4788f) )
@@ -2907,7 +2907,7 @@ ROM_START( sftm111 ) /* 版本 1.11, P/N 1064 REV 1 主板, P/N 1073 REV
        ROM_LOAD32_BYTE (“rm1-2.grm1_2”, 0x1000002, 0x400000, CRC (903e56c2) SHA1 (843ed9855ffdf37b100b3c5614139d552fd9cd6d) )
        ROM_LOAD32_BYTE (“rm1-3.grm1_3”, 0x1000003, 0x400000, CRC (fac35686) SHA1 (ba99ab265620575c14c46806dc543d1f9fd24462) )

- /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */
+ /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */

        ROM_LOAD32_BYTE( "sfm_grm3_0.grm3_0", 0x2000000, 0x020000, CRC(3e1f76f7) SHA1(8aefe376e7248a583a6af02e5f9b2a4b48cc91d7) )
        ROM_LOAD32_BYTE (“sfm_grm3_1.grm3_1”, 0x2000001, 0x020000, CRC (578054b6) SHA1 (99201959de28dbfd7692cedea4485751d3d4788f) )
@@ -2943,7 +2943,7 @@ ROM_START( sftm110 ) /* 版本 1.10, P/N 1064 REV 1 主板, P/N 1073 REV
        ROM_LOAD32_BYTE (“rm1-2.grm1_2”, 0x1000002, 0x400000, CRC (903e56c2) SHA1 (843ed9855ffdf37b100b3c5614139d552fd9cd6d) )
        ROM_LOAD32_BYTE (“rm1-3.grm1_3”, 0x1000003, 0x400000, CRC (fac35686) SHA1 (ba99ab265620575c14c46806dc543d1f9fd24462) )

- /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */
+ /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */

        ROM_LOAD32_BYTE( "sfm_grm3_0.grm3_0", 0x2000000, 0x020000, CRC(3e1f76f7) SHA1(8aefe376e7248a583a6af02e5f9b2a4b48cc91d7) )
        ROM_LOAD32_BYTE (“sfm_grm3_1.grm3_1”, 0x2000001, 0x020000, CRC (578054b6) SHA1 (99201959de28dbfd7692cedea4485751d3d4788f) )
@@ -2979,7 +2979,7 @@ ROM_START( sftmj ) /* 版本 1.12N（日本）, P/N 1064 REV 1 主板, P/N 1073
        ROM_LOAD32_BYTE (“rm1-2.grm1_2”, 0x1000002, 0x400000, CRC (903e56c2) SHA1 (843ed9855ffdf37b100b3c5614139d552fd9cd6d) )
        ROM_LOAD32_BYTE (“rm1-3.grm1_3”, 0x1000003, 0x400000, CRC (fac35686) SHA1 (ba99ab265620575c14c46806dc543d1f9fd24462) )

- /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */
+ /* GROM2_0 到 GROM2_3 是未填充的 23C32000 掩码 ROM 位置 */

        ROM_LOAD32_BYTE( "sfm_grm3_0.grm3_0", 0x2000000, 0x020000, CRC(3e1f76f7) SHA1(8aefe376e7248a583a6af02e5f9b2a4b48cc91d7) )
        ROM_LOAD32_BYTE (“sfm_grm3_1.grm3_1”, 0x2000001, 0x020000, CRC (578054b6) SHA1 (99201959de28dbfd7692cedea4485751d3d4788f) )
diff --git a/src/mame/drivers/jchan.cpp b/src/mame/drivers/jchan.cpp
索引 f595198d3b1..e654e534c6a 100644
---a/src/mame/drivers/jchan.cpp
+++ b/src/mame/drivers/jchan.cpp
@@ -113,7 +113,7 @@ CG24143/CG24173 - 富士通定制图形生成器
           垂直同步 - 59.6010Hz
           水平同步 - 15.55610kHz
```

```
 -EPROM:
 +EPROM:
  位置 ROM 类型 PCB 标签
  --------------------------------
  U164 27C2001 SPA-7A
 @@ -648,7 +648,7 @@ MACHINE_CONFIG_START(jchan_state::jchan)
         MCFG_SOUND_ROUTE(1, "扬声器", 1.0)
  MACHINE_CONFIG_END

 -/* ROM加载*/
 +/* ROM加载*/

  ROM_START（jchan）
         ROM_REGION( 0x200000, "maincpu", 0 ) /* 68000 代码 */
 @@ -690,7 +690,7 @@ ROM_START( jchan )
  ROM_END


 -ROM_START( jchan2 ) /* 某种半续集？MASK ROM 转储并确认是相同的 */
 +ROM_START( jchan2 ) /* 某种半续集？掩码 ROM 转储并确认是相同的 */
         ROM_REGION( 0x200000, "maincpu", 0 ) /* 68000 代码 */
         ROM_LOAD16_BYTE（“j2p1x1.u67”，0x000001，0x080000，CRC（5448c4bc）SHA1（447835275d5454f86a51879490a6b22b06a23e81））
         ROM_LOAD16_BYTE（“j2p1x2.u68”，0x000000，0x080000，CRC（52104ab9）SHA1（d6647e628662bdb832270540ece18b265b7ce62d））
 diff --git a/src/mame/drivers/kaneko16.cpp b/src/mame/drivers/kaneko16.cpp
 索引 b706ffaedab..3f87ce5e926 100644
 ---a/src/mame/drivers/kaneko16.cpp
 +++ b/src/mame/drivers/kaneko16.cpp
 @@ -3575,7 +3575,7 @@ 注意：
              M2B0X0.U93 27C010 (1M) | M2B0X0.U93 27C010 (1M)
              M2B1X0.U94 27C010（1M）/

 - M2-100-00.U48 8M MASKROM（32针）\
 + M2-100-00.U48 8M掩膜ROM（32针）\
              M2W1A1.U47 27C040 (4M) / 冲电气样品

  ************************************************** ************************/
 diff --git a/src/mame/drivers/konamigv.cpp b/src/mame/drivers/konamigv.cpp
 索引 858e6becf38..770768474c9 100644
 ---a/src/mame/drivers/konamigv.cpp
 +++ b/src/mame/drivers/konamigv.cpp
 @@ -96,7 +96,7 @@ 注意：

        - CXD2922 和 CXD2925 是 SPU。

 - - ZV610 和 GV999 上的 BIOS 是相同的。它是一个4M MASK ROM，与27C040兼容。
 + - ZV610 和 GV999 上的 BIOS 是相同的。它是一个4M掩膜ROM，与27C040兼容。
```

```
        - CD 包含一个 MODE 1 数据轨道和多个 Redbook 音轨，这些音轨通过 CN8 传输到扬声器。

@@ -705,13 +705,13 @@ INPUT_PORTS_END
 ROM_START（科纳米）
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", ROMREGION_ERASE00 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", ROMREGION_ERASE00 ) /* 默认 EEPROM */
 ROM_END

 ROM_START( lacrazyc )
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD（“lacrazyc.25c”，0x000000，0x000080，CRC（e20e5730）SHA1（066b49236c658a4ef2930f7bacc4b2354dd7f240））

        DISK_REGION（“scsi：”SCSI_PORT_DEVICE1“：cdrom”）
@@ -721,7 +721,7 @@ ROM_END
 ROM_START（继续）
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD("susume.25c", 0x000000, 0x000080, CRC(52f17df7) SHA1(b8ad7787b0692713439d7d9bebfa0c801c806006) )

        DISK_REGION（“scsi：”SCSI_PORT_DEVICE1“：cdrom”）
@@ -731,7 +731,7 @@ ROM_END
 ROM_START（海马）
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "hyperath.25c", 0x000000, 0x000080, CRC(20a8c435) SHA1(a0f203a999757fba68b391c525ac4b9684a57ba9) )

        DISK_REGION（“scsi：”SCSI_PORT_DEVICE1“：cdrom”）
@@ -741,7 +741,7 @@ ROM_END
 ROM_START（powyak96）
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD（“powyak96.25c”，0x000000，0x000080，CRC（405a7fc9）SHA1（e2d978f49748ba3c4a425188abcd3d272ec23907））

        DISK_REGION（“scsi：”SCSI_PORT_DEVICE1“：cdrom”）
@@ -751,7 +751,7 @@ ROM_END
 ROM_START（婚礼者）
```

```
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "weddingr.25c", 0x000000, 0x000080, CRC(b90509a0) SHA1(41510a0ceded81dcb26a70eba97636d38d3742c3) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -761,7 +761,7 @@ ROM_END
 ROM_START(简单碗)
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "simpbowl.25c", 0x000000, 0x000080, CRC(2c61050c) SHA1(16ae7f81cbe841c429c5c7326cf83e87db1782bf) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -771,7 +771,7 @@ ROM_END
 ROM_START(btchamp)
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "btchmp.25c", 0x000000, 0x000080, CRC(6d02ea54) SHA1(d3babf481fd89db3aec17f589d0d3d999a2aa6e1) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -781,7 +781,7 @@ ROM_END
 ROM_START(kdeadeye)
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "kdeadeye.25c", 0x000000, 0x000080, CRC(3935d2df) SHA1(cbb855c475269077803c380dbc3621e522efe51e) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -791,7 +791,7 @@ ROM_END
 ROM_START(nagano98)
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "nagano98.25c", 0x000000, 0x000080, CRC(b64b7451) SHA1(a77a37e0cc580934d1e7e05d523bae0acd2c1480) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -801,7 +801,7 @@ ROM_END
 ROM_START(长野)
        GV_BIOS
```

```
- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "720ja.25c", 0x000000, 0x000080, CRC(34c473ba) SHA1(768225b04a293bdbc114a092d14dee28d52044e9) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -811,7 +811,7 @@ ROM_END
 ROM_START( tmosh )
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "tmosh.25c", 0x000000, 0x000080, NO_DUMP )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -821,7 +821,7 @@ ROM_END
 ROM_START( tmosh )
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "tmoshs.25c", 0x000000, 0x000080, CRC(e57b833f) SHA1(f18a0974a6be69dc179706643aab837ff61c2738) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -831,7 +831,7 @@ ROM_END
 ROM_START( tmoshsp )
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "tmoshsp.25c", 0x000000, 0x000080, CRC(af4cdd87) SHA1(97041e287e4c80066043967450779b81b62b2b8e) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
@@ -841,7 +841,7 @@ ROM_END
 ROM_START( tmoshspa )
        GV_BIOS

- ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认eeprom */
+ ROM_REGION16_BE( 0x0000080, "eeprom", 0 ) /* 默认 EEPROM */
        ROM_LOAD( "tmoshsp.25c", 0x000000, 0x000080, CRC(af4cdd87) SHA1(97041e287e4c80066043967450779b81b62b2b8e) )

        DISK_REGION( "scsi:" SCSI_PORT_DEVICE1 ":cdrom" )
diff --git a/src/mame/drivers/konamim2.cpp b/src/mame/drivers/konamim2.cpp
索引 5bfed0207f0..1aeac3f1c40 100644
---a/src/mame/drivers/konamim2.cpp
+++ b/src/mame/drivers/konamim2.cpp
@@ -89,7 +89,7 @@ 注意：
                    连接至 Panasonic CR-583 8 速 CDROM 驱动器。
```

```
       LA4705 - LA4705 功率放大器
       DSW-8 位拨码开关
- BOOTROM.8Q - 16MBit MASKROM。位置 8Q (DIP42)
+ BOOTROM.8Q - 16MBit 掩码 ROM。位置 8Q (DIP42)
                    战斗幽会 - 636A01.8Q
                    邪恶之夜 - .8Q
                    十一人热火 '98 - .8Q
diff --git a/src/mame/drivers/lastfght.cpp b/src/mame/drivers/lastfght.cpp
索引 db1107ec162..ba14903c9ee 100644
---a/src/mame/drivers/lastfght.cpp
+++ b/src/mame/drivers/lastfght.cpp
@@ -37,8 +37,8 @@ PCB 布局

 笔记：
       H8/3044 - Subsino 重新贴标 Hitachi H8/3044 HD6433044A22F 微控制器 (QFP100)
- H8/3044 是具有 24 位地址总线的 H8/3002，具有 32k MASKROM 和 2k RAM，时钟输入为 16.000MHz [32/2]
- MD0、MD1 和 MD2 配置为 MODE 6 16M 字节扩展模式，并启用片上 32k MASKROM。
+ H8/3044 是具有 24 位地址总线的 H8/3002，具有 32k 掩模 ROM 和 2k RAM，时钟输入为 16.000MHz [32/2]
+ MD0、MD1 和 MD2 配置为 MODE 6 16M 字节扩展模式，并启用片上 32k 掩膜 ROM。
       EPM7032 - Altera EPM7032LC44-15T CPLD (PLCC44)
      CXK58257 - 索尼 CXK58257 32k x8 SRAM (SOP28)
     KM428C256 - 三星半导体 KM428C256 256k x8 双端口 DRAM (SOJ40)
@@ -52,7 +52,7 @@ 注意：
          水平同步 - 15.75kHz
          垂直同步 - 60Hz
- ROM 板 - 小型子板，包含 8 个 16MBit SOP44 MASKROM 的位置。仅填充位置 1-4。
+ ROM 板 - 小型子板，包含 8 个 16MBit SOP44 掩模 ROM 的位置。仅填充位置 1-4。
     定制 IC -
                 U19 - SUBSINO 9623EX008 (QFP208)
                 H8/3044 - SUBSINO SS9689 6433044A22F，重新贴牌日立 H8/3044 MCU (QFP100)
diff --git a/src/mame/drivers/m107.cpp b/src/mame/drivers/m107.cpp
索引 9eab26207e9..c2c2b523660 100644
---a/src/mame/drivers/m107.cpp
+++ b/src/mame/drivers/m107.cpp
@@ -898,13 +898,13 @@ ROM_START( dsoccr94 )
        ROM_LOAD16_BYTE（“a3-s10-c-0.ic37”，0x00000，0x10000，CRC（768132e5）SHA1（1bb64516eb58d3b246f08e1c07f091e78085689f））

        ROM_REGION( 0x400000, "gfx1", 0 ) /* 字符 */
- ROM_LOAD16_BYTE( "ds_c00.ic29", 0x000000, 0x100000, CRC(2d31d418) SHA1(6cd0e362bc2e3f2b20d96ee97a04bff46ee3016a) ) /* 没有“官方”ROM 标签的 MASK
ROM */
+ ROM_LOAD16_BYTE( "ds_c00.ic29", 0x000000, 0x100000, CRC(2d31d418) SHA1(6cd0e362bc2e3f2b20d96ee97a04bff46ee3016a) ) /* 屏蔽没有“官方”ROM 标签的
ROM */
        ROM_LOAD16_BYTE（“ds_c10.ic28”，0x000001，0x100000，CRC（57f7bcd3）SHA1（a38e7cdfdea72d882fba414cae391ba09443e73c））
        ROM_LOAD16_BYTE（“ds_c01.ic21”，0x200000，0x100000，CRC（9d31a464）SHA1（1e38ac296f64d77fabfc0d5f7921a9b7a8424875））
        ROM_LOAD16_BYTE（“ds_c11.ic20”，0x200001，0x100000，CRC（a372e79f）SHA1（6b0889cfc2970028832566e25257927ddc461ea6））
```

```
         ROM_REGION( 0x400000, "gfx2", 0 ) /* 精灵 */
- ROM_LOAD( "ds_000.ic11", 0x000000, 0x100000, CRC(366b3e29) SHA1(cb016dcbdc6e8ea56c28c00135263666b07df991) ) /* 没有“官方”ROM 标签的 MASK ROM */
+ ROM_LOAD( "ds_000.ic11", 0x000000, 0x100000, CRC(366b3e29) SHA1(cb016dcbdc6e8ea56c28c00135263666b07df991) ) /* 没有“官方”ROM 标签的掩码 ROM */
         ROM_LOAD（“ds_010.ic12”，0x100000，0x100000，CRC（28a4cc40）SHA1（7f4e1ef995eaadf1945ee22ab3270cb8a21c601d））
         ROM_LOAD（“ds_020.ic13”，0x200000，0x100000，CRC（5a310f7f）SHA1（21969e4247c8328d27118d00604096deaf6700af））
         ROM_LOAD（“ds_030.ic14”，0x300000，0x100000，CRC（328b1f45）SHA1（4cbbd4d9be4fc151d426175bdbd35d8481bf2966））
@@ -925,13 +925,13 @@ ROM_START( dsoccr94k )
         ROM_LOAD16_BYTE（“a3-sl0-c-0.ic37”，0x00000，0x10000，CRC（768132e5）SHA1（1bb64516eb58d3b246f08e1c07f091e78085689f））

         ROM_REGION( 0x400000, "gfx1", 0 ) /* 字符 */
- ROM_LOAD16_BYTE( "ds_c00.ic29", 0x000000, 0x100000, CRC(2d31d418) SHA1(6cd0e362bc2e3f2b20d96ee97a04bff46ee3016a) ) /* 没有“官方”ROM 标签的 MASK
ROM */
+ ROM_LOAD16_BYTE( "ds_c00.ic29", 0x000000, 0x100000, CRC(2d31d418) SHA1(6cd0e362bc2e3f2b20d96ee97a04bff46ee3016a) ) /* 屏蔽没有“官方”ROM 标签的
ROM */
         ROM_LOAD16_BYTE（“ds_c10.ic28”，0x000001，0x100000，CRC（57f7bcd3）SHA1（a38e7cdfdea72d882fba414cae391ba09443e73c））
         ROM_LOAD16_BYTE（“ds_c01.ic21”，0x200000，0x100000，CRC（9d31a464）SHA1（1e38ac296f64d77fabfc0d5f7921a9b7a8424875））
         ROM_LOAD16_BYTE（“ds_c11.ic20”，0x200001，0x100000，CRC（a372e79f）SHA1（6b0889cfc2970028832566e25257927ddc461ea6））

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 精灵 */
- ROM_LOAD( "ds_000.ic11", 0x000000, 0x100000, CRC(366b3e29) SHA1(cb016dcbdc6e8ea56c28c00135263666b07df991) ) /* 没有“官方”ROM 标签的 MASK ROM */
+ ROM_LOAD( "ds_000.ic11", 0x000000, 0x100000, CRC(366b3e29) SHA1(cb016dcbdc6e8ea56c28c00135263666b07df991) ) /* 没有“官方”ROM 标签的掩码 ROM */
         ROM_LOAD（“ds_010.ic12”，0x100000，0x100000，CRC（28a4cc40）SHA1（7f4e1ef995eaadf1945ee22ab3270cb8a21c601d））
         ROM_LOAD（“ds_020.ic13”，0x200000，0x100000，CRC（5a310f7f）SHA1（21969e4247c8328d27118d00604096deaf6700af））
         ROM_LOAD（“ds_030.ic14”，0x300000，0x100000，CRC（328b1f45）SHA1（4cbbd4d9be4fc151d426175bdbd35d8481bf2966））
@@ -998,7 +998,7 @@ ROM_START( kftgoal )
         ROM_REGION( 0x100000, "irem", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "pk-da0.da0", 0x000000, 0x80000, BAD_DUMP CRC(26a34cf4) SHA1(a8a7cd91cdc6d644ee02ca16e7fdc8debf8f3a5f) ) //显然取自 World PK
Soccer，标题屏幕上显示“World PK Soccer”

- ROM_REGION( 0x2000, "eeprom", 0 ) /* ST M28C64C-20PI Eeprom */
+ ROM_REGION( 0x2000, "eeprom", 0 ) /* ST M28C64C-20PI EEPROM */
         ROM_LOAD（“st-m28c64c.eeprom”，0x000，0x2000，CRC（8e0c8b7c）SHA1（0b57290d709e6d54ce1bb3a5c01b80590203c1dd））
 ROM_END

diff --git a/src/mame/drivers/macrossp.cpp b/src/mame/drivers/macrossp.cpp
索引 4232d3ed722..0ac3136fece 100644
---a/src/mame/drivers/macrossp.cpp
+++ b/src/mame/drivers/macrossp.cpp
@@ -8,7 +8,7 @@ 测验 Bisyoujo Senshi Sailor Moon (c)1997 Banpresto
 司机：大卫•海伍德

 去做：
- -  “BIOS”ROM 有什么用？它似乎是数据表，并且在游戏之间有很大不同，但我们没有将其映射到任何地方
+ -  “BIOS”ROM 的用途是什么？它似乎是数据表，并且在游戏之间有很大不同，但我们没有将其映射到任何地方
  - 将图块地图转换为设备？

  68020 中断
```

```
diff --git a/src/mame/drivers/midzeus.cpp b/src/mame/drivers/midzeus.cpp
索引 ba02c534e00..b9363d648a6 100644
---a/src/mame/drivers/midzeus.cpp
+++ b/src/mame/drivers/midzeus.cpp
@@ -379,7 +379,7 @@ WRITE32_MEMBER(midzeus_state::disk_asic_jr_w)
                        membank("bank1")->set_entry(disk_asic_jr[偏移] & 3);
                        休息;

- /* 宙斯2 ws; 0=zeus 访问 1 等待状态，2=解锁 ROM；crusnexo/thegrid 在启动时写入 1 */
+ /* 宙斯2 ws; 0=zeus 访问 1 等待状态，2=解锁 ROM；crusnexo/thegrid 在启动时写入 1 */
                案例7：
                        休息;

@@ -1344,10 +1344,10 @@ MACHINE_CONFIG_END

 ROM_START（mk4）
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
- ROM_LOAD16_BYTE( "mk4_l2.u2", 0x000000, 0x100000, CRC(f9d410b4) SHA1(49bcacf83430ed26c08789b2f3ed9f946c3a0e5e) ) /* 标记为 v2.0，ROM 类型 M27C800
*/
- ROM_LOAD16_BYTE( "mk4_l2.u3", 0x400000, 0x200000, CRC(8fbcf0ac) SHA1(c53704e72cfcba800c7af3a03267041f1e29a784) ) /* 标记为 v2.0，ROM 类型 M27C160
*/
- ROM_LOAD16_BYTE( "mk4_l1.u4", 0x800000, 0x200000, CRC(dee91696) SHA1(00a182a36a414744cd014fcfc53c2e1a66ab5189) ) /* 标记为 v1.0，rom 类型 M27C160
*/
- ROM_LOAD16_BYTE( "mk4_l1.u5", 0xc00000, 0x200000, CRC(44d072be) SHA1(8a636c2801d799dfb84e69607ade76d2b49cf09f) ) /* 标记为 v1.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l2.u2", 0x000000, 0x100000, CRC(f9d410b4) SHA1(49bcacf83430ed26c08789b2f3ed9f946c3a0e5e) ) /* 标记为 v2.0，ROM 类型 M27C800
*/
+ ROM_LOAD16_BYTE( "mk4_l2.u3", 0x400000, 0x200000, CRC(8fbcf0ac) SHA1(c53704e72cfcba800c7af3a03267041f1e29a784) ) /* 标记为 v2.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l1.u4", 0x800000, 0x200000, CRC(dee91696) SHA1(00a182a36a414744cd014fcfc53c2e1a66ab5189) ) /* 标记为 v1.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l1.u5", 0xc00000, 0x200000, CRC(44d072be) SHA1(8a636c2801d799dfb84e69607ade76d2b49cf09f) ) /* 标记为 v1.0，ROM 类型 M27C160
*/

        ROM_REGION32_LE( 0x1800000, "用户1", 0 )
        ROM_LOAD32_WORD( "mk4_l3.u10", 0x0000000, 0x200000, CRC(84efe5a9) SHA1(e2a9bf6fab971691017371a87ab87b1bf66f96d0) ) /* ROM U10 和 U11 被标记
为 v3.0 */
@@ -1355,22 +1355,22 @@ ROM_START( mk4 )
        ROM_LOAD32_WORD( "mk4_l2.u12", 0x0400000, 0x200000, CRC(7816c07f) SHA1(da94b4391e671f915c61b5eb9bece4acb3382e31) ) /* ROM U12 到 U17 全部标
记为 v2.0 */
        ROM_LOAD32_WORD（“mk4_l2.u13”，0x0400002，0x200000，CRC（b3c237cd）SHA1（9e71e60cc92c17524f85f36543c174ca138104cd））
        ROM_LOAD32_WORD( "mk4_l2.u14", 0x0800000, 0x200000, CRC(fd33eb1a) SHA1(59d9d2e5251679d19cab031f51731c85f429ba18) ) /* 在后期生产中，这些 */
- ROM_LOAD32_WORD( "mk4_l2.u15", 0x0800002, 0x200000, CRC(b907518f) SHA1(cfb56538746895bdca779957fec6a872019b23c3) ) /* rom 也被标记为 v3.0，但标签
*/
+ ROM_LOAD32_WORD( "mk4_l2.u15", 0x0800002, 0x200000, CRC(b907518f) SHA1(cfb56538746895bdca779957fec6a872019b23c3) ) /* ROM 也被标记为 v3.0，但标签
*/
        ROM_LOAD32_WORD( "mk4_l2.u16", 0x0c00000, 0x200000, CRC(24371d57) SHA1(c90134b17c23a182d391d1679bf457d251e641f7) ) /* v2.0 已在多块板上验证
```

```
 */
- ROM_LOAD32_WORD( "mk4_l2.u17", 0x0c00002, 0x200000, CRC(3a1a082c) SHA1(5f8e8ce760d8ebadd1240ef08f1382a37cf11d0b) ) /* 有些 PCB 可能全部都是 MASK
ROM。*/
+ ROM_LOAD32_WORD( "mk4_l2.u17", 0x0c00002, 0x200000, CRC(3a1a082c) SHA1(5f8e8ce760d8ebadd1240ef08f1382a37cf11d0b) ) /* 某些 PCB 可能全部包含掩模
ROM。*/
 ROM_END

 ROM_START（mk4a）
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
- ROM_LOAD16_BYTE( "mk4_l2.u2", 0x000000, 0x100000, CRC(f9d410b4) SHA1(49bcacf83430ed26c08789b2f3ed9f946c3a0e5e) ) /* 标记为 v2.0，ROM 类型 M27C800
*/
- ROM_LOAD16_BYTE( "mk4_l2.u3", 0x400000, 0x200000, CRC(8fbcf0ac) SHA1(c53704e72cfcba800c7af3a03267041f1e29a784) ) /* 标记为 v2.0，ROM 类型 M27C160
*/
- ROM_LOAD16_BYTE( "mk4_l1.u4", 0x800000, 0x200000, CRC(dee91696) SHA1(00a182a36a414744cd014fcfc53c2e1a66ab5189) ) /* 标记为 v1.0, rom 类型 M27C160
*/
- ROM_LOAD16_BYTE( "mk4_l1.u5", 0xc00000, 0x200000, CRC(44d072be) SHA1(8a636c2801d799dfb84e69607ade76d2b49cf09f) ) /* 标记为 v1.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l2.u2", 0x000000, 0x100000, CRC(f9d410b4) SHA1(49bcacf83430ed26c08789b2f3ed9f946c3a0e5e) ) /* 标记为 v2.0，ROM 类型 M27C800
*/
+ ROM_LOAD16_BYTE( "mk4_l2.u3", 0x400000, 0x200000, CRC(8fbcf0ac) SHA1(c53704e72cfcba800c7af3a03267041f1e29a784) ) /* 标记为 v2.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l1.u4", 0x800000, 0x200000, CRC(dee91696) SHA1(00a182a36a414744cd014fcfc53c2e1a66ab5189) ) /* 标记为 v1.0，ROM 类型 M27C160
*/
+ ROM_LOAD16_BYTE( "mk4_l1.u5", 0xc00000, 0x200000, CRC(44d072be) SHA1(8a636c2801d799dfb84e69607ade76d2b49cf09f) ) /* 标记为 v1.0，ROM 类型 M27C160
*/

        ROM_REGION32_LE( 0x1800000, "用户1", 0 )
- ROM_LOAD32_WORD( "mk4_l2.1.u10", 0x000000, 0x200000, CRC(42d0f1c9) SHA1(5ac0ded8bf6e756319be2691e3b555eac079ebdc) ) /* ROM U10 和 U11 被标记为
v2.1 */
+ ROM_LOAD32_WORD( "mk4_l2.1.u10", 0x000000, 0x200000, CRC(42d0f1c9) SHA1(5ac0ded8bf6e756319be2691e3b555eac079ebdc) ) /* ROM U10 和 U11 被标记为
v2.1 */
        ROM_LOAD32_WORD（“mk4_l2.1.u11”，0x000002，0x200000，CRC（6e21b243）SHA1（6d4768a5972db05c1409e0d16e79df9eff8918a0））
- ROM_LOAD32_WORD( "mk4_l2.u12", 0x0400000, 0x200000, CRC(7816c07f) SHA1(da94b4391e671f915c61b5eb9bece4acb3382e31) ) /* ROM U12 到 U17 全部标记为
v2.0 */
+ ROM_LOAD32_WORD( "mk4_l2.u12", 0x0400000, 0x200000, CRC(7816c07f) SHA1(da94b4391e671f915c61b5eb9bece4acb3382e31) ) /* ROM U12 到 U17 全部标记为
v2.0 */
        ROM_LOAD32_WORD（“mk4_l2.u13”，0x0400002，0x200000，CRC（b3c237cd）SHA1（9e71e60cc92c17524f85f36543c174ca138104cd））
        ROM_LOAD32_WORD（“mk4_l2.u14”，0x0800000，0x200000，CRC（fd33eb1a）SHA1（59d9d2e5251679d19cab031f51731c85f429ba18））
        ROM_LOAD32_WORD（“mk4_l2.u15”，0x0800002，0x200000，CRC（b907518f）SHA1（cfb56538746895bdca779957fec6a872019b23c3））
@@ -1380,24 +1380,24 @@ ROM_END

 ROM_START（mk4b）
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
- ROM_LOAD16_BYTE( "mk4_l1.u2", 0x000000, 0x200000, CRC(daac8ab5) SHA1(b93aa205868212077a9b6ac8e93205e1ebf8c05e) ) /* 所有声音 ROM 均标记为 v1.0 且
为 M27C160 类型*/
+ ROM_LOAD16_BYTE( "mk4_l1.u2", 0x000000, 0x200000, CRC(daac8ab5) SHA1(b93aa205868212077a9b6ac8e93205e1ebf8c05e) ) /* 所有声音 ROM 均标记为 v1.0 且
为 M27C160 类型*/
```

```
        ROM_LOAD16_BYTE（“mk4_l1.u3”, 0x400000, 0x200000, CRC（cb59413e）SHA1（f7e5c589a8f6a2e7dceee4881594e7403be4d4ad））
        ROM_LOAD16_BYTE（“mk4_l1.u4”, 0x800000, 0x200000, CRC（dee91696）SHA1（00a182a36a414744cd014fcfc53c2e1a66ab5189））
        ROM_LOAD16_BYTE（“mk4_l1.u5”, 0xc00000, 0x200000, CRC（44d072be）SHA1（8a636c2801d799dfb84e69607ade76d2b49cf09f））

        ROM_REGION32_LE( 0x1800000, "用户1", 0 )
- ROM_LOAD32_WORD( "mk4_l1.u10", 0x0000000, 0x200000, CRC(6fcc86dd) SHA1(b3b2b463daf51450fbcd5d2922ac1b091bd91c4a) ) /* 所有ROM都标记为v1.0 */
+ ROM_LOAD32_WORD( "mk4_l1.u10", 0x0000000, 0x200000, CRC(6fcc86dd) SHA1(b3b2b463daf51450fbcd5d2922ac1b091bd91c4a) ) /* 所有ROM都标记为v1.0 */
        ROM_LOAD32_WORD（“mk4_l1.u11”, 0x0000002, 0x200000, CRC（04895940）SHA1（55d368905f5986587c4e3da236401fdd5e2c269c））
        ROM_LOAD32_WORD（“mk4_l1.u12”, 0x0400000, 0x200000, CRC（323ddc5c）SHA1（4303c109c68a7cc15ff6fe91b6d34383b6066351））
        ROM_LOAD32_WORD（“mk4_l1.u13”, 0x0400002, 0x200000, CRC（0b95bdf0）SHA1（a25d48b33a861b5e52736720c7a79291fa837f78））
        ROM_LOAD32_WORD（“mk4_l1.u14”, 0x0800000, 0x200000, CRC（cb6816ef）SHA1（9c828c188d297aee0f211acc283035289e80b5a8））
        ROM_LOAD32_WORD（“mk4_l1.u15”, 0x0800002, 0x200000, CRC（cde47df7）SHA1（63383d983c03703b2f3f1973ce2a7553654836d4））
- /* 此版本中不存在 U16 或 U17 ROM */
+ /* 此版本中不存在 U16 或 U17 ROM */
 ROM_END

-ROM_START( invasnab ) /* 版本 5.0 程序 ROM、v4.0 图形 ROM、v2.0 声音 ROM */
+ROM_START( invasnab ) /* 版本 5.0 程序 ROM、v4.0 图形 ROM、v2.0 声音 ROM */
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
- ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 rom 被标记为
v2.0 */
+ ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 ROM 被标记为
v2.0 */
        ROM_LOAD16_BYTE（“入侵2.u3”, 0x400000, 0x200000, CRC（86b956ae）SHA1（f7fd4601a2ce3e7e9b67e7d77908bfa206ee7e62））
        ROM_LOAD16_BYTE（“入侵2.u4”, 0x800000, 0x200000, CRC（5ef1fab5）SHA1（987afa0672fa89b18cf20d28644848a9e5ee9b17））
        ROM_LOAD16_BYTE（“入侵2.u5”, 0xc00000, 0x200000, CRC（e42805c9）SHA1（e5b71eb1852809a649ac43a82168b3bdaf4b1526））
@@ -1418,9 +1418,9 @@ ROM_START( invasnab ) /* 版本 5.0 程序 ROM、v4.0 图形 ROM、v2.0 Soun
        ROM_LOAD32_WORD（“入侵4.u19”, 0x1000002, 0x200000, CRC（89fa6ee5）SHA1（572565e1308142b0b062aa72315c68e928f2419c））
 ROM_END

-ROM_START( invasnab4 ) /* 版本 4.0 程序 ROM 和图形 ROM，v2.0 声音 ROM */
+ROM_START( invasnab4 ) /* 版本 4.0 程序 ROM 和图形 ROM、v2.0 声音 ROM */
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
- ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 rom 被标记为
v2.0 */
+ ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 ROM 被标记为
v2.0 */
        ROM_LOAD16_BYTE（“入侵2.u3”, 0x400000, 0x200000, CRC（86b956ae）SHA1（f7fd4601a2ce3e7e9b67e7d77908bfa206ee7e62））
        ROM_LOAD16_BYTE（“入侵2.u4”, 0x800000, 0x200000, CRC（5ef1fab5）SHA1（987afa0672fa89b18cf20d28644848a9e5ee9b17））
        ROM_LOAD16_BYTE（“入侵2.u5”, 0xc00000, 0x200000, CRC（e42805c9）SHA1（e5b71eb1852809a649ac43a82168b3bdaf4b1526））
@@ -1429,7 +1429,7 @@ ROM_START( invasnab4 ) /* 版本 4.0 程序 ROM 和图形 ROM，v2.0 声音 r
        ROM_LOAD( "pic16c57.u76", 0x00000, 0x2000, CRC(f62729c9) SHA1(9642c53dd7eceeb7eb178497d367691c44abc5c5) ) // 这是否是有效的转储？

        ROM_REGION32_LE( 0x1800000, "用户1", 0 )
- ROM_LOAD32_WORD( "invasion4.u10", 0x0000000, 0x200000, CRC(b3ce958b) SHA1(ed51c167d85bc5f6155b8046ec056a4f4ad5cf9d) ) /* 这些ROM都标记为v4.0 */
+ ROM_LOAD32_WORD( "invasion4.u10", 0x0000000, 0x200000, CRC(b3ce958b) SHA1(ed51c167d85bc5f6155b8046ec056a4f4ad5cf9d) ) /* 这些ROM都标记为v4.0 */
        ROM_LOAD32_WORD（“入侵4.u11”, 0x0000002, 0x200000, CRC（0bd09359）SHA1（f40886bd2e5f5fbf506580e5baa2f733be200852））
```

```
        ROM_LOAD32_WORD（“入侵4.u12”，0x0400000，0x200000，CRC（ce1eb06a）SHA1（ff17690a0cbca6dcccccde70e2c5812ae03db5bb））
        ROM_LOAD32_WORD（“入侵4.u13”，0x0400002，0x200000，CRC（33fc6707）SHA1（11a39ad980ec320547319eca6ffa5aef3ab8b010））
@@ -1441,9 +1441,9 @@ ROM_START( invasnab4 ) /* 版本 4.0 程序 ROM 和图形 ROM，v2.0 声音 r
        ROM_LOAD32_WORD（“入侵4.u19”，0x1000002，0x200000，CRC（89fa6ee5）SHA1（572565e1308142b0b062aa72315c68e928f2419c））
 ROM_END

-ROM_START( invasnab3 ) /* 版本 3.0 程序 ROM 和 v2.0 图形 ROM、v2.0 声音 ROM */
+ROM_START( invasnab3 ) /* 版本 3.0 程序 ROM 和 v2.0 图形 ROM、v2.0 声音 ROM */
        ROM_REGION16_LE( 0x1000000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
-       ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 rom 被标记为 v2.0 日期 6/24 /99 */
+       ROM_LOAD16_BYTE( "invasion2.u2", 0x000000, 0x200000, CRC(59d2e1d6) SHA1(994a4311ac4841d4341449c0c7480952b6f3855d) ) /* 这四个声音 ROM 被标记为 v2.0 日期 6/24 /99 */
        ROM_LOAD16_BYTE（“入侵2.u3”，0x400000，0x200000，CRC（86b956ae）SHA1（f7fd4601a2ce3e7e9b67e7d77908bfa206ee7e62））
        ROM_LOAD16_BYTE（“入侵2.u4”，0x800000，0x200000，CRC（5ef1fab5）SHA1（987afa0672fa89b18cf20d28644848a9e5ee9b17））
        ROM_LOAD16_BYTE（“入侵2.u5”，0xc00000，0x200000，CRC（e42805c9）SHA1（e5b71eb1852809a649ac43a82168b3bdaf4b1526））
@@ -1474,7 +1474,7 @@ ROM_START( crusnexo )
        ROM_REGION32_LE( 0x0800000, "用户1", 0 )
        ROM_LOAD32_WORD( "exotica-24.u10", 0x0000000, 0x200000, CRC(5e702f7c) SHA1(98c76fb46b304d4d21656d0505d5e5e99c8335bf) ) /* 版本 2.4 2000 年 8 月 23 日星期三17:26:53 */
        ROM_LOAD32_WORD（“exotica-24.u11”，0x0000002，0x200000，CRC（5ecb2cbc）SHA1（57283167e48ca96579d0712d9fec23a36fa2b496））
-       ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
+       ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
        ROM_LOAD32_WORD( "exotica-10.u13", 0x0400002, 0x200000, CRC(cf9d3609) SHA1(6376891f478185d26370466bef92f0c5304d58d3) ) /* 但数据没有改变。已验证 v1.3 和 v1.6 */

        ROM_REGION32_LE( 0x3000000, "用户2", 0 )
@@ -1502,7 +1502,7 @@ ROM_START( crusnexoa )
        ROM_REGION32_LE( 0x0800000, "用户1", 0 )
        ROM_LOAD32_WORD( "exotica-20.u10", 0x0000000, 0x200000, CRC(43d80f54) SHA1(25683d835f3ed3dee99da33280ae6e21865801e4) ) /* 版本 2.0 2000 年 4 月 7 日星期五 17:5 5:07 */
        ROM_LOAD32_WORD（“exotica-20.u11”，0x0000002，0x200000，CRC（dba26b69）SHA1（4900ac3fe67664a543dcd66e41793874f6cdc07f））
-       ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
+       ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
        ROM_LOAD32_WORD( "exotica-10.u13", 0x0400002, 0x200000, CRC(cf9d3609) SHA1(6376891f478185d26370466bef92f0c5304d58d3) ) /* 但数据没有改变。已验证 v1.3 和 v1.6 */

        ROM_REGION32_LE( 0x3000000, "用户2", 0 )
@@ -1530,7 +1530,7 @@ ROM_START( crusnexob )
        ROM_REGION32_LE( 0x0800000, "用户1", 0 )
        ROM_LOAD32_WORD( "exotica-16.u10", 0x0000000, 0x200000, CRC(65450140) SHA1(cad41a2cad48426de01feb78d3f71f768e3fc872) ) /* 版本 1.6 2000 年 2 月 22 日星期二 10:25: 01*/
        ROM_LOAD32_WORD（“exotica-16.u11”，0x0000002，0x200000，CRC（e994891f）SHA1（bb088729b665864c7f3b79b97c3c86f9c8f68770））
```

```
- ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
+ ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
        ROM_LOAD32_WORD( "exotica-10.u13", 0x0400002, 0x200000, CRC(cf9d3609) SHA1(6376891f478185d26370466bef92f0c5304d58d3) ) /* 但数据没有改变。已验证 v1.3 和 v1.6 */

        ROM_REGION32_LE( 0x3000000, "用户2", 0 )
@@ -1558,7 +1558,7 @@ ROM_START( crusnexoc )
        ROM_REGION32_LE( 0x0800000, "用户1", 0 )
        ROM_LOAD32_WORD( "exotica-13.u10", 0x0000000, 0x200000, CRC(ab7f1b5e) SHA1(c0c561e8cb15fd97465278b4b3b15acb27380c5d) ) /* 版本 1.3 2000 年 2 月 11 日星期五 16: 19:13 */
        ROM_LOAD32_WORD（“exotica-13.u11”，0x0000002，0x200000，CRC（62d3c966）SHA1（9a485892295984a292501424d2c78caafac99a75））
- ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
+ ROM_LOAD32_WORD( "exotica-10.u12", 0x0400000, 0x200000, CRC(21f122b2) SHA1(5473401ec954bf9ab66a8283bd08d17c7960cd29) ) /* 这 2 个 ROM 可能被标记为不同的版本，*/
        ROM_LOAD32_WORD( "exotica-10.u13", 0x0400002, 0x200000, CRC(cf9d3609) SHA1(6376891f478185d26370466bef92f0c5304d58d3) ) /* 但数据没有改变。已验证 v1.3 和 v1.6 */

        ROM_REGION32_LE( 0x3000000, "用户2", 0 )
@@ -1605,7 +1605,7 @@ ROM_START( crusnexod )
 ROM_END


-ROM_START( thegrid ) /* 版本 1.2 程序 rom */
+ROM_START( thegrid ) /* 版本 1.2 程序 ROM */
        ROM_REGION16_LE( 0xc00000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
        ROM_LOAD（“the_grid.u2”，0x000000，0x400000，CRC（e6a39ee9）SHA1（4ddc62f5d278ea9791205098fa5f018ab1e698b4））
        ROM_LOAD（“the_grid.u3”，0x400000，0x400000，CRC（40be7585）SHA1（e481081edffa07945412a6eab17b4d3e7b42cfd3））
@@ -1627,7 +1627,7 @@ ROM_START( thegrid ) /* 版本 1.2 程序 ROM */
 ROM_END


-ROM_START( thegrida ) /* 版本 1.1 程序 rom */
+ROM_START( thegrida ) /* 版本 1.1 程序 ROM */
        ROM_REGION16_LE( 0xc00000, "dcs", ROMREGION_ERASEFF ) /* 声音数据 */
        ROM_LOAD（“the_grid.u2”，0x000000，0x400000，CRC（e6a39ee9）SHA1（4ddc62f5d278ea9791205098fa5f018ab1e698b4））
        ROM_LOAD（“the_grid.u3”，0x400000，0x400000，CRC（40be7585）SHA1（e481081edffa07945412a6eab17b4d3e7b42cfd3））
diff --git a/src/mame/drivers/model1.cpp b/src/mame/drivers/model1.cpp
索引 5079d0a93f0..ea63a17904f 100644
---a/src/mame/drivers/model1.cpp
+++ b/src/mame/drivers/model1.cpp
@@ -72,13 +72,13 @@ 注意：

       IC6、IC7、IC8、\
       IC9、IC10、\
```

```
- IC11、IC12、| 834000 4M 掩模ROM (DIP32)
+ IC11，IC12，| 834000 4M掩膜ROM（DIP32）
       IC13，|
       IC39、IC40、/
       IC41、IC42 /

       IC26、IC27、\
- IC28、IC29、| 8316200 16M 掩模ROM (DIP42)
+ IC28，IC29，| 8316200 16M掩膜ROM（DIP42）
       IC30、IC31、/
       IC32、IC33 /

@@ -195,13 +195,13 @@ 注：
       J4、J5、J6 - 跳线，全部设置为 2-3

       OPR-14742.44 \
- OPR-14743.45 - 1M SOP40 MASKROM，绑定到 315-5464
+ OPR-14743.45 - 1M SOP40 掩模 ROM，绑定到 315-5464

       OPR-14744.64 \
- OPR-14745.65 - 1M SOP40 MASKROM，与 315-5572 绑定
+ OPR-14745.65 - 1M SOP40 掩模 ROM，与 315-5572 绑定

       OPR-14746.68 \
- OPR-14747.69 - 1M SOP40 MASKROM，连接至 315-5572 @ IC66
+ OPR-14747.69 - 1M SOP40 掩模 ROM，连接到 315-5572 @ IC66


 视频PCB
@@ -280,7 +280,7 @@ 注：
       315-5292 - 世嘉定制 (QFP160)

       OPR-14748.15 \
- OPR-14748.16 - 1M SOP40 MASKROM，连接到 315-5423 和 315-5424。请注意，两个 ROM 是相同的。
+ OPR-14748.16 - 1M SOP40 掩模 ROM，连接到 315-5423 和 315-5424。请注意，两个 ROM 是相同的。


 电机PCB
diff --git a/src/mame/drivers/model3.cpp b/src/mame/drivers/model3.cpp
索引 22317f05e91..f224bdbf676 100644
---a/src/mame/drivers/model3.cpp
+++ b/src/mame/drivers/model3.cpp
@@ -191,7 +191,7 @@ ROM板
 笔记：
       CN1/2/3/4 - 将 ROM 板连接到 CPU 板的连接器（下）
       跳线 - 这些跳线设置与 Scud Race 和 VF3TB 相匹配。跳投可能与其他游戏不同，但是
- 如果不同，则可能仅适用于使用 64MBit MASKROM 的游戏。
```

```
+ 如果不同，则可能仅适用于使用 64MBit mask ROM 的游戏。
        ROM - 并非所有插槽均已填充。请参阅 MAME src 以了解确切的 ROM 使用情况。

  （供倾销参考）
@@ -215,8 +215,8 @@ 跳线位置。3是+5V
  ----------------------------------------
  JP10：ic 26至ic 41的1-2 pin32

-所有CROM ROM均为32M MASK
-所有 VROM ROM 均为 16M MASK
+所有CROM ROM均为32M掩码
+所有VROM ROM均为16M掩码


  中央处理器板
@@ -440,7 +440,7 @@ VROM03.28 mpr-20379 |
  VROM04.31 mpr-20382 |
  VROM05.30 mpr-20381 |
  VROM06.33 mpr-20384 |
-VROM07.32 mpr-20383 \ 32MBit DIP42 掩码ROM
+VROM07.32 mpr-20383 \ 32MBit DIP42 掩模 ROM
  VROM10.35 mpr-20386 /
  VROM11.34 mpr-20385 |
  VROM12.37 mpr-20388 |
@@ -453,7 +453,7 @@ VROM17.40 mpr-20391 /
  CROM01.3 mpr-20363 |
  CROM02.2 mpr-20362 |
-CROM03.1 mpr-20361 \ 32MBit DIP42 掩码ROM
+CROM03.1 mpr-20361 \ 32MBit DIP42 掩模 ROM
  CROM10.8 mpr-20368 /
  CROM11.7 mpr-20367 |
  CROM12.6 mpr-20366 |
@@ -474,7 +474,7 @@ CROM3.17 epr-20393A /
  SROM1.22 mpr-20373 \
 -SROM2.23 mpr-20374 | 32MBit DIP42 掩码ROM
+SROM2.23 mpr-20374 | 32MBit DIP42掩膜ROM
  SROM3.24 mpr-20375 |
  SROM4.25 mpr-20376 /

@@ -532,10 +532,10 @@ ROM板
  |---------------------------------------------- ----------------------------------------------- |

  用于 Spikeout FE。主要区别是修订后的 315-6090B GAL 和 JP2 设置。
-IC1 到 IC12 的 CROM 是 64Mbit 掩码 ROM，所有其他掩码 ROM 都是 32Mbit。
```

```
-64Mbit 掩码 ROM 读取为 27C322，引脚 11 +5v 和 27C322，引脚 11 GND
+ IC1 至 IC12 处的 CROM 为 64Mbit 掩膜 ROM，所有其他掩膜 ROM 均为 32Mbit。
+64Mbit 掩模 ROM 读取为 27C322，引脚 11 +5v 和 27C322，引脚 11 GND

-具有 64Mbit mask rom 的 Model 3 游戏有 Daytona 2、Spikeout 和 Spikeout FE
+具有 64Mbit mask ROM 的 Model 3 游戏有 Daytona 2、Spikeout 和 Spikeout FE

 中央处理器板
 ---------
@@ -2205,7 +2205,7 @@ ROM_START( lemans24 ) /* 步骤1.5，Sega游戏ID#为833-13159，ROM板ID#8
 ROM_END

 ROM_START( scud ) /* 步骤 1.5，Sega 游戏 ID# 为 833-13041，ROM 主板 ID# 834-13072 SPG COMM AUS */
- /* 已知有一块 ROM 板 ID# 834-13034 SPG DX AUS，其程序 roms EPR-19634 至 EPR-19637 */
+ /* 已知 ROM 板 ID# 834-13034 SPG DX AUS 具有程序 ROM EPR-19634 至 EPR-19637 */
         ROM_REGION64_BE( 0x8800000, "user1", 0 ) /* 程序 + 数据 ROM */
         // 只读存储器
         ROM_LOAD64_WORD_SWAP（“epr-19731.17”，0x0600006，0x80000，CRC（3ee6447e）SHA1（124697791d90c1b352dd6e33bd3b45535aa92bb5））
@@ -2966,27 +2966,27 @@ ROM_END

 /*
 在 Mame getbass 中标记为 GET BASS STD，而我的 PCB 来自 DLX 驾驶室。
- ROM板833-13317
-834-13318 贴纸也位于 ROM 板上。
-笼子上有以下贴纸
+ROM板833-13317
+834-13318 贴纸也位于 ROM 板上。
+笼子上有以下贴纸：
 BSS-4500-CVT2
 833-13317 游戏BD BSS-CVT2

 I/O 板 837-13283（手册中的 GET BASS MEC CONT BD） 171-7558c

-epr20690.ic11 是控制器板程序
-CPU是k15c80a16cf
- 该板有 4 个开关（sw3 至 sw6）
- 一个复位开关
+epr20690.ic11 是控制器板程序ROM
+CPU为KL5C80A16CF
+该板有 4 个开关（sw3 至 sw6）
+一个复位开关
 2 组 8 个拨码开关
 SW1全部关闭
 SW2全部关闭
-lh52256cn-70 内存（超级电容备份）
-世嘉315-5296
-sega 315-5649 都是I/O芯片
```

```
 -gal16v8d（世嘉315-6126）
 -32 MHz 晶体
 -93c45 EEPROM
 +LH52256CN-70 RAM（超级容量备份）
 +世嘉315-5296
 +Sega 315-5649（两者似乎都是I/O芯片）
 +GAL16V8D（世嘉315-6126）
 +32 Mhz 晶体
 +93C45 EEPROM
  */

  ROM_START( getbass ) /* 步骤 1.0, Sega 游戏 ID# 为 833-13416 GET BASS STD, ROM 板 ID# 834-13417 */
 diff --git a/src/mame/drivers/namcofl.cpp b/src/mame/drivers/namcofl.cpp
 索引 7947692be99..5cc06e34a82 100644
 ---a/src/mame/drivers/namcofl.cpp
 +++ b/src/mame/drivers/namcofl.cpp
 @@ -106,17 +106,17 @@ 定制

  ROM - 主板
  ----------------
 -23S：MASK ROM - SE1_VOI.23S（PCB 标签“VOICE”），安装在小型插件 PCB 上
 +23S：掩模 ROM - SE1_VOI.23S（PCB 标签“VOICE”），安装在小型插件 PCB 上
       标记为 MEMEXT 32M MROM PCB 8635909200 (8635909300)。该芯片以 BYTE 模式编程。
 -18U：MB834000 掩模 ROM - SE1_SSH.18U（PCB 标签“SSHAPE”）
 -21P：MB838000 掩模 ROM - SE1_SCH0.21P（PCB 标签“SCHA0”）
 -20P：MB838000 掩模 ROM - SE1_SCH1.20P（PCB 标签“SCHA1”）
 -19P：MB838000 掩模 ROM - SE1_SCH2.19P（PCB 标签“SCHA2”）
 -18P：MB838000 掩模 ROM - SE1_SCH3.18P（PCB 标签“SCHA3”）
 +18U：MB834000 掩模 ROM - SE1_SSH.18U（PCB 标签“SSHAPE”）
 +21P：MB838000 掩模 ROM - SE1_SCH0.21P（PCB 标签“SCHA0”）
 +20P：MB838000 掩模 ROM - SE1_SCH1.20P（PCB 标签“SCHA1”）
 +19P：MB838000 掩模 ROM - SE1_SCH2.19P（PCB 标签“SCHA2”）
 +18P：MB838000掩膜ROM - SE1_SCH3.18P（PCB标签“SCHA3”）
  21L：M27C4002 EPROM - SE1_SPR.21L（PCB 标签“SPROG”）
 -14K：MB834000 掩模 ROM - SE1_RSH.14K（PCB 标签“RSHAPE”）
 -19J：MB838000 掩模 ROM - SE1_RCH0.19J（PCB 标签“RCHA0”）
 -18J：MB838000 掩模 ROM - SE1_RCH1.18J（PCB 标签“RCHA1”）
 +14K：MB834000 掩模 ROM - SE1_RSH.14K（PCB 标签“RSHAPE”）
 +19J：MB838000 掩模 ROM - SE1_RCH0.19J（PCB 标签“RCHA0”）
 +18J：MB838000 掩模 ROM - SE1_RCH1.18J（PCB 标签“RCHA1”）
  17J、16J：RCH2、RCH3，但插座未安装
  19A：D27C4096 EPROM - SE2MPEA4.19A（PCB 标签“PROGE”）
  18A：D27C4096 EPROM - SE2MPOA4.18A（PCB 标签“PROGO”）
 @@ -128,10 +128,10 @@ ROM - 主板

  ROM - 子板
  ----------------
```

```
-IC1：MB8316200 SOP44 掩模 ROM - SE1OBJ0L.IC1（PCB 标签“OBJ0L”）
-IC2：MB8316200 SOP44 掩模 ROM - SE1OBJ0U.IC2（PCB 标签“OBJ0U”）
-IC3：MB8316200 SOP44 掩模 ROM - SE1OBJ1L.IC3（PCB 标签“OBJ1L”）
-IC4：MB8316200 SOP44 掩模 ROM - SE1OBJ1U.IC4（PCB 标签“OBJ1U”）
+IC1：MB8316200 SOP44 掩模 ROM - SE1OBJ0L.IC1（PCB 标签“OBJ0L”）
+IC2：MB8316200 SOP44 掩模 ROM - SE1OBJ0U.IC2（PCB 标签“OBJ0U”）
+IC3：MB8316200 SOP44 掩模 ROM - SE1OBJ1L.IC3（PCB 标签“OBJ1L”）
+IC4：MB8316200 SOP44 掩模 ROM - SE1OBJ1U.IC4（PCB 标签“OBJ1U”）


 PAL
diff --git a/src/mame/drivers/namconb1.cpp b/src/mame/drivers/namconb1.cpp
索引 eaaf3ecfb89..1d280536d50 100644
---a/src/mame/drivers/namconb1.cpp
+++ b/src/mame/drivers/namconb1.cpp
@@ -1131,13 +1131,13 @@ MACHINE_CONFIG_END

 /*************************************************** **************/

-ROM_START( ptblank ) /* 使用 4Mb 声音数据 ROM 的世界集（已验证） */
+ROM_START( ptblank ) /* 使用 4Mb 声音数据 ROM 的世界集（已验证） */
        ROM_REGION( 0x100000, "maincpu", 0 ) /* 主程序 */
        ROM_LOAD32_WORD（“gn2_mprlb.15b”，0x00002，0x80000，CRC（fe2d9425）SHA1（51b166a629cbb522720d63720558816b496b6b76））
        ROM_LOAD32_WORD( "gn2_mprub.13b", 0x00000, 0x80000, CRC(3bf4985a) SHA1(f559e0d5f55d23d886fe61bd7d5ca556acc7f87c) )

- ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
-// ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
+ ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
+// ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
        ROM_LOAD( "gn1-spr0.5b", 0, 0x80000, CRC(71773811) SHA1(e482784d9b9ebf8c2e4a2a3f6f6c4dc8304d2251) ) /* 4Megabit, 0x00000-0x1ffff, 0x20000-
处的相同数据0x7ffff为0xff填充*/

        ROM_REGION( 0x1000000, "c352", 0 ) // 样本
@@ -1167,8 +1167,8 @@ ROM_START( ptblanka ) /* 使用非标准 ROM 标签的世界集（NR 是 Namco 的）
        ROM_LOAD32_WORD( "nr3_spr0.15b", 0x00002, 0x80000, CRC(fe2d9425) SHA1(51b166a629cbb522720d63720558816b496b6b76) ) // == gn2_mprlb.15b
        ROM_LOAD32_WORD( "nr2_spr0.13b", 0x00000, 0x80000, CRC(3bf4985a) SHA1(f559e0d5f55d23d886fe61bd7d5ca556acc7f87c) ) // == gn2_mprub.12b

- ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
-// ROM_LOAD( "nr1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
+ ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
+// ROM_LOAD( "nr1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
        ROM_LOAD( "nr1_spr0.5b", 0, 0x80000, CRC(a0bde3fb) SHA1(b5fac1d0339b1df6b8880fcd7aa2725a774765a4) ) /* 4Megabit, 0x00000-0x1ffff 处的相同数
据重复 4 次 */
```

```
        ROM_REGION( 0x1000000, "c352", 0 ) // 样本
@@ -1193,13 +1193,13 @@ ROM_START( ptblanka ) /* 使用非标准 ROM 标签的世界集（NR 是 Namco 的）
        ROM_LOAD（“eeprom”，0x0000，0x0800，CRC（95760d0f）SHA1（94ac5a261d9afc77c2a163a50950b0e86b1f8041））
 ROM_END

-ROM_START( Gunbuletw ) /* 使用 4Mb 声音数据 ROM 的世界设定（已验证） */
+ROM_START( Gunbuletw ) /* 使用 4Mb 声音数据 ROM 的世界设定（已验证） */
        ROM_REGION( 0x100000, "maincpu", 0 ) /* 主程序 */
        ROM_LOAD32_WORD（“gn3_mprlb.15b”，0x00002，0x80000，CRC（9260fce5）SHA1（064579be1ac90e04082a8b403c6adf35dbb46a7e））
        ROM_LOAD32_WORD（“gn3_mprub.13b”，0x00000，0x80000，CRC（6c1ac697）SHA1（7b52b5ef8154a5d741ac24673f3e6bbfa246a494））

- ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
-// ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
+ ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
+// ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb
ROM 相同的数据 */
        ROM_LOAD( "gn1-spr0.5b", 0, 0x80000, CRC(71773811) SHA1(e482784d9b9ebf8c2e4a2a3f6f6c4dc8304d2251) ) /* 4Megabit, 0x00000-0x1ffff, 0x20000-
处的相同数据0x7ffff为0xff填充*/

        ROM_REGION( 0x1000000, "c352", 0 ) // 样本
@@ -1224,13 +1224,13 @@ ROM_START( Gunbuletw ) /* 使用 4Mb 声音数据 ROM 的世界设置（已验证） */
        ROM_LOAD（“eeprom”，0x0000，0x0800，CRC（95760d0f）SHA1（94ac5a261d9afc77c2a163a50950b0e86b1f8041））
 ROM_END

-ROM_START(gunbuletj) /* 使用 1Mb 声音数据 ROM 的日语设置（已验证）*/
+ROM_START(gunbuletj) /* 使用 1Mb 声音数据 ROM 的日语设置（已验证）*/
        ROM_REGION( 0x100000, "maincpu", 0 ) /* 主程序 */
        ROM_LOAD32_WORD( "gn1_mprl.15b", 0x00002, 0x80000, CRC(f99e309e) SHA1(3fe0ddf756e6849f8effc7672456cbe32f65c98a) )
        ROM_LOAD32_WORD( "gn1_mpru.13b", 0x00000, 0x80000, CRC(72a4db07) SHA1(8c5e1e51cd961b311d03f7b21f36a5bd5e8e9104) )

- ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
- ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb ROM
相同的数据 */
+ ROM_REGION16_LE( 0x80000, "c75data", 0 ) /* 声音数据 - JP1 跳线可在 1Mb (27C1024) 或 4Mb (27C4096) 之间选择，ROM 正确 */
+ ROM_LOAD( "gn1_spr0.5b", 0, 0x20000, CRC(6836ba38) SHA1(6ea17ea4bbb59be108e8887acd7871409580732f) ) /* 1Megabit，与 0x00000-0x1ffff 处的 4Mb ROM
相同的数据 */
 // ROM_LOAD( "gn1-spr0.5b", 0, 0x80000, CRC(71773811) SHA1(e482784d9b9ebf8c2e4a2a3f6f6c4dc8304d2251) ) /* 4Megabit, 0x00000-0x1ffff, 0x20000 处的
相同数据-0x7ffff为0xff填充*/

        ROM_REGION( 0x1000000, "c352", 0 ) // 样本
@@ -1481,20 +1481,20 @@ ROM 标签 标签类型
 GS1MPRU.13B PRGU 27C240 \ 主程序
 GS1MPRL.15B PRGL 27C240 /
 GS1SPR0.5B SPRG 27C240 声音程序，链接到 75、C351 和 C352
-GS1VOI-0.5J VOICE 16M MASK 声音
```

```
 -GS1CHR-0.8J CHR0 8M 面具角色
 -GS1CHR-1.9J CHR1 8M 面具角色
 -GS1CHR-2.10J CHR2 8M 面具角色
 -GS1CHR-3.11J CHR3 8M 面具角色
 -GS1SHA-0.5M 形状 4M 面罩形状
 +GS1VOI-0.5J VOICE 16M 面罩 声音 声音
 +GS1CHR-0.8J CHR0 8M 掩码字符
 +GS1CHR-1.9J CHR1 8M 掩模字符
 +GS1CHR-2.10J CHR2 8M 掩模字符
 +GS1CHR-3.11J CHR3 8M 掩模字符
 +GS1SHA-0.5M SHAPE 4M 面罩形状

  ROM、MEMEXT OBJ2 PCB（所有 ROM 表面安装）
  --------------------
  文件名/PCB ROM
  ROM 标签 标签类型
  ---------------------------------------------------
 -GS1OBJ-0.IC1 OBJL 16M 掩模 SOP44
 -GS1OBJ-1.IC2 OBJU 16M 面罩 SOP44
 +GS1OBJ-0.IC1 OBJL 16M 掩模 SOP44
 +GS1OBJ-1.IC2 OBJU 16M 掩模 SOP44

  笔记！所有 ROM 均与 Great Sluggers '94 套装不同。

 @@ -1761,10 +1761,10 @@ ou2mprl.11c PRGL 27C4002 \ 主程序
  ou2mpru.11d PRGU 27C4002 /
  ou1spr0.5b SPRG 27C240 声音程序，链接到 C352 和 C382
  ou1voi0.6n VOICE0 MB8316200B 声音
 -ou1shas.12s 形状-S 16M 面膜形状
 -ou1shar.18s SHAPE-R 16M 面膜形状
 +ou1shas.12s SHAPE-S 16M 面膜形状
 +ou1shar.18s SHAPE-R 16M 面膜形状

 -ROM、MASK ROM PCB（所有 ROM 表面安装）
 +ROM、掩膜 ROM PCB（所有 ROM 表面安装）
  ------------------
  文件名/PCB ROM
  ROM 标签 标签类型
 diff --git a/src/mame/drivers/namcos10.cpp b/src/mame/drivers/namcos10.cpp
 索引 58930b534a5..97a64523422 100644
 ---a/src/mame/drivers/namcos10.cpp
 +++ b/src/mame/drivers/namcos10.cpp
 @@ -177,7 +177,7 @@ ROM 子板 PCB
  该 PCB 包含所有 ROM。
  S10 游戏使用了三种已知类型的 ROM 子板（到目前为止）。
  所有 PCB 的尺寸相同（约 5″ x 5″），包含一个表面安装在 PCB 底部的定制连接器
 -PCB、一些 MASKROM/FlashROM、一个 CPLD（这似乎是惯用的“KEYCUS”芯片。第二个类型是 RAM
```

```
+PCB、一些掩膜 ROM/闪存 ROM、一个 CPLD（这似乎是惯用的“KEYCUS”芯片。第二个类型是 RAM
 芯片也存在。第三种类型具有额外的硬件来解码 MP3 音频和无 ROM 微控制器。

 ******
@@ -209,7 +209,7 @@ 系统10 MEM(M) PCB 8906961000 (8906970700)
 笔记：
       CY37128VP160：CY37128VP160 赛普拉斯复杂可编程逻辑器件 (TQFP160)
       1A - 5A：英特尔闪存 DA28F640J5 64MBit 闪存 EEPROM (SSOP56)
- 1D - 7E：Samsung Electronics K3N9V1000A-YC 128MBit MASK ROM (TSOP48)（参见注释 3）
+ 1D - 7E：Samsung Electronics K3N9V1000A-YC 128MBit 掩模 ROM (TSOP48)（见注释 3）
       J1：6 引脚接头，用于通过 JTAG 对 CPLD 进行编程

 该 PCB 用于：
@@ -926,7 +926,7 @@ MACHINE_CONFIG_START(namcos10_state::namcos10_memm)
        MCFG_DEVICE_ADD（“主CPU”，CXD8606BQ，XTAL（101’491’200））
        MCFG_DEVICE_PROGRAM_MAP（ namcos10_memm_map ）

- // BIOS首先将rom窗口配置为80000-big，然后
+ // BIOS首先将ROM窗口配置为80000-big，然后
        //切换到400000。如果berr处于活动状态，则第一个配置
        // 擦除 1fc80000 之后的所有处理程序，从而终止系统
        // 然后
@@ -953,7 +953,7 @@ MACHINE_CONFIG_START(namcos10_state::namcos10_memn)
        MCFG_DEVICE_ADD（“主CPU”，CXD8606BQ，XTAL（101’491’200））
        MCFG_DEVICE_PROGRAM_MAP（ namcos10_memn_map ）

- // BIOS首先将rom窗口配置为80000-big，然后
+ // BIOS首先将ROM窗口配置为80000-big，然后
        //切换到400000。如果berr处于活动状态，则第一个配置
        // 擦除 1fc80000 之后的所有处理程序，从而终止系统
        // 然后
diff --git a/src/mame/drivers/namcos11.cpp b/src/mame/drivers/namcos11.cpp
索引 4220c4e207d..62c43d77c0c 100644
---a/src/mame/drivers/namcos11.cpp
+++ b/src/mame/drivers/namcos11.cpp
@@ -64,8 +64,8 @@ MOTHER PCB-这是主PCB。它包含所有声音电路、声音 ROM、prog
             用于一些较小的组件改组。第 2 版仅适用于 Kosodate Quiz My Angel 3 和 Star Sweep。
 CPU PCB - 该 PCB 有两个已知版本。任何游戏都可以使用任一 PCB。包含主 CPU/RAM 和 GPU/视频 RAM
             区别仅在于 RAM 类型，一种使用 4 个 16MBit 芯片，另一种则使用 2 个 32MBit 芯片。
-ROM PCB - 该 PCB 有两个已知版本。它们基本上是相同的，除了一个使用所有 32MBit SOP44 MASKROM，另一个使用所有 32MBit SOP44 MASKROM
- 使用 64MBit SOP44 MASKROM。64MBit ROM 板还具有用于 PAL 和 KEYCUS 的空间。
+ROM PCB - 该 PCB 有两个已知版本。它们基本上是相同的，除了一个使用所有 32MBit SOP44 掩模 ROM，另一个使用所有 32MBit SOP44 掩模 ROM
+ 使用 64MBit SOP44 掩模 ROM。64MBit ROM 板还具有用于 PAL 和 KEYCUS 的空间。

 每个游戏都分配有一个多字母代码，该代码印在小贴纸上并放置在母 PCB 的底部。
 该代码后面是一个数字（到目前为止看到的是 1、2、3 和 4），然后是“Ver.” 然后A/B/C/D/E 表示软件
@@ -112,7 +112,7 @@ 注意：
```

```
                    * 引脚 9 VCLKOUT - 40.0264MHz (==2x MCLKOUT)。与 C195 相关
                    * 引脚 7 XTALOUT - 16.93426MHz。这与 C76 的时钟输入相关
       S11MOT* - 标准系统 11 PAL (DIP20)
- WAVE.8K - 声音样本，42 针 DIP MASKROM，16MBit 或 32MBit。如果是 32MBit，则以字节模式编程。
+ WAVE.8K - 声音样本，42 针 DIP 掩模 ROM，16MBit 或 32MBit。如果是 32MBit，则以字节模式编程。
       SPROG.6D - 声音程序，Intel PA28F200BX 2MBit Flash ROM (SOP44)
       PRG.2* - 主程序，Intel E28F008SA 8MBit Flash ROM (TSOP40)
       CONN1 - 用于连接 ROM 板
@@ -164,7 +164,7 @@ SYSTEM11 ROM8 PCB 8645960202 (8645970202)
 | |
 |-----------------------------------------------------|
 笔记：
- 该 ROM 板最多可连接 8 个 8 位 32MBit SOP44 MASK ROM。
+ 该 ROM 板连接最多可容纳 8 个 8 位 32MBit SOP44 掩模 ROM。


 系统11 ROM8(64) PCB 8645960500 (8645970500)
@@ -177,7 +177,7 @@ SYSTEM11 ROM8(64) PCB 8645960500 (8645970500)
 | *PRG3L.IC9 |
 |-----------------------------------------------------|
 笔记：
- 该 ROM 板最多可连接 8 个 8 位 64MBit SOP44 MASK ROM。
+ 该 ROM 板连接最多可容纳 8 个 8 位 64MBit SOP44 掩模 ROM。
       有空间容纳 PLCC44 KEYCUS IC（通常是 CPLD，但未安装）和 PLCC20
       IC 类型 PAL16V8H（已填充并标记为“ROM8 DEC0”）
       * - 这些 ROM 位于 PCB 的另一侧。
@@ -474,9 +474,9 @@ WRITE16_MEMBER( namcos11_state::c76_shared_w )

 无效 namcos11_state::namcos11_map(address_map &map)
 {
- 地图(0x1fa04000, 0x1fa0ffff).rw(FUNC(namcos11_state::c76_shared_r), FUNC(namcos11_state::c76_shared_w)); /* 与 C76 共享内存 */
+ 地图(0x1fa04000, 0x1fa0ffff).rw(FUNC(namcos11_state::c76_shared_r), FUNC(namcos11_state::c76_shared_w)); /* 与 C76 共享 RAM */
        映射(0x1fa20000, 0x1fa2001f).rw("keycus", FUNC(ns11_keycus_device::read), FUNC(ns11_keycus_device::write));
- 映射（0x1fa30000，0x1fa30fff）.rw（“at28c16”，FUNC（at28c16_device::read），FUNC（at28c16_device::write））.umask32（0x00ff00ff）；/* EEPROM */
+ 映射(0x1fa30000, 0x1fa30fff).rw("at28c16", FUNC(at28c16_device::read), FUNC(at28c16_device::write)).umask32(0x00ff00ff); /* EEPROM */
        地图(0x1fb00000, 0x1fb00003).nopw(); /* ?? */
        地图(0x1fbf6000, 0x1fbf6003).nopw(); /* ?? */
 }
diff --git a/src/mame/drivers/namcos12.cpp b/src/mame/drivers/namcos12.cpp
索引 c0b79c58126..737f4ece210 100644
---a/src/mame/drivers/namcos12.cpp
+++ b/src/mame/drivers/namcos12.cpp
@@ -299,9 +299,9 @@ ROM 子板 PCB
 ----------------------
 该 PCB 包含剩余的 ROM，用于图形和 3D 几何。
 S12 游戏使用的 ROM 子板有 8 种已知类型（到目前为止）。
-所有 PCB 尺寸相同（约 2″ x 7″），包含一个定制连接器和一些 MASKROM/FlashROM、一个 PLCC
```

```
+所有 PCB 尺寸相同（约 2″ x 7″），包含一个定制连接器和一些掩模 ROM/闪存 ROM、一个 PLCC
 PAL 和 KEYCUS（PLCC44 CPLD），在某些情况下还有额外的 TQFP CPLD。
- PCB 以特殊编码命名。首先是一个字母 M，表示 MASKROM（始终是 SOP44），然后是一个数字，表示如何
+ PCB 以特殊编码命名。首先是一个字母 M，表示掩模 ROM（始终是 SOP44），然后是一个数字，表示如何
 许多该类型的 ROM，然后是另一个字母 F，表示 FLASHROM（始终为 TSOP40/48/56），然后是一个数字，表示有多少
 该类型的 ROM。该数字始终是 PCB 上可使用的该 ROM 类型的最大数量。实际数量
 PCB 上填充的 ROM 数量可以更少。
@@ -330,9 +330,9 @@ 注：
        | 64M | 哦| X | |32M闪存| 哦|
        |----------|----|----| |----------|----|

- WAVEx：64M SOP44 MASKROM（R6 填充）
+ WAVEx：64M SOP44 掩模 ROM（R6 填充）
        FLx：Intel E28F016 TSOP40 16M FlashROM 或 Fujitsu 29F016A TSOP48 16M FlashROM（R8 未填充）
- ROM0x：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ ROM0x：64M SOP44掩膜ROM（大小固定为64M，无配置选项）
        IC3：MACH211 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和每个游戏不同的 3 位数字。）
        IC4：PALCE 16V8H（PLCC20，PCB 标记为“A_DECO”，芯片标记为“A DECO”）

@@ -419,9 +419,9 @@ 注意：
        |64M 地层| 哦| | 3.3V| X | X | 哦| 哦| 哦| 哦|
        |----------|----| |------|---|---|---|---|---|----|

- WAVEx：64M SOP44 MASKROM（R6 填充）
+ WAVEx：64M SOP44 掩模 ROM（R6 填充）
        FLx：Intel E28F320J5 TSOP56 32M FlashROM（R11 和 R12 填充）
- ROMx：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ ROMx：64M SOP44 mask ROM（大小固定为64M，无配置选项）
        IC3：MACH211 CPLD 或 Cypress CY37064 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和一个 3 位数字，
               每个游戏都不同。）
        IC4：PALCE 16V8H（PLCC20，PCB 标记为“A_DECO”，芯片标记为“M5F4”）
@@ -475,9 +475,9 @@ 注意：
        | 64M | 哦| X | |64M 地层| 哦|
        |----------|----|----| |----------|----|

- WAVEx：64M SOP44 MASKROM（R6 填充）
+ WAVEx：64M SOP44 掩模 ROM（R6 填充）
        FLx：Fujitsu 29F016 TSOP48 16M FlashROM（R8 未填充）
- ROMx：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ ROMx：64M SOP44 mask ROM（大小固定为64M，无配置选项）
        IC3：MACH211 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和每个游戏不同的 3 位数字。）
        IC4：PALCE 16V8H（PLCC20，PCB 标记为“A_DECO”，芯片标记为“M5F4”）

@@ -542,13 +542,13 @@ 注意：
        | W128M | R3 | 如果填充 R3，Wave ROM 为 128M
        | W32M | R4 | 如果填充 R4，Wave ROM 为 32M
        | W64M | R5 | 如果填充 R5，Wave ROM 为 64M
```

```
- | M64M | R6 | 如果填充 R6，MASK ROM 为 64M
- | M32M | R7 | 如果填充 R7，MASK ROM 为 32M
+ | M64M | R6 | 如果填充 R6，掩模 ROM 为 64M
+ | M32M | R7 | 如果填充 R7，掩模 ROM 为 32M
       |--------|----|

- WAVEx：32M/64M SOP44 掩模ROM
+ WAVEx：32M/64M SOP44 掩模 ROM
       FLx：Fujitsu 29F016 TSOP48 16M FlashROM（大小固定为16M，无配置选项）
- ROMx：32M/64M SOP44 掩模ROM
+ ROMx：32M/64M SOP44掩膜ROM
       IC1：MACH211 CPLD 或 Cypress CY37064 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和一个 3 位数字，
              每个游戏都不同。）
       IC2：74F139逻辑IC
@@ -626,9 +626,9 @@ 注意：
       |----------|----|----|

       * : 这些部件位于 PCB 的另一侧
- WAVEx：32M/64M SOP44 掩模ROM
+ WAVEx：32M/64M SOP44 掩模 ROM
       FLx：Fujitsu 29F016 TSOP48 16M FlashROM（大小固定为16M，无配置选项）
- ROMx：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ ROMx：64M SOP44 mask ROM（大小固定为64M，无配置选项）
       IC2：MACH211 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和每个游戏不同的 3 位数字。）
       IC3：PALCE 22V10H（PLCC28，标记为“S12M840A”）
       IC8：ALTERA MAX EPM7128STC100-10 FPGA（TQFP100，标记为“S12M841”）
@@ -677,9 +677,9 @@ 注意：
       |----------|----|----|

       * : 这些部件位于 PCB 的另一侧
- WAVEx：32M/64M SOP44 掩模ROM
+ WAVEx：32M/64M SOP44 掩模 ROM
       FLx：Fujitsu 29F016 TSOP48 16M FlashROM（大小固定为16M，无配置选项）
- ROMx：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ ROMx：64M SOP44 mask ROM（大小固定为64M，无配置选项）
       IC2：赛普拉斯 CY37064 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和一个不同的 3 位数字
              每场比赛。）
       IC3：PALCE 22V10H（PLCC28，PCB 标记为“A_DECO”，芯片标记为“S12M8F6”）
@@ -736,8 +736,8 @@ 注意：
       | 64M | X | 哦|
       |----------|----|----|

- WAVEx：32M/64M SOP44 掩模ROM
- ROMx：64M SOP44 MASKROM（大小固定为64M，无配置选项）
+ WAVEx：32M/64M SOP44 掩模 ROM
+ ROMx：64M SOP44 mask ROM（大小固定为64M，无配置选项）
       KEYCUS：CY37064 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和每个游戏不同的 3 位数字。）
```

```
        IC4：PALCE 16V8H（PLCC20，PCB 标记为“A_DECO”，芯片标记为“S12M10X”）

@@ -781,9 +781,9 @@ 注意：
        该 PCB 上 ROM 的实际转储与原始 Soul Calibur 转储完全匹配（2 型 ROM PCB），尽管
        ROM 大小和类型不同。

- WAVEx：MR27C3252CZ 32M SOP44 掩模ROM
- ROMx：MR27C3252CZ 32M SOP44 掩模ROM
- FLx：MR27C3252CZ 32M SOP44 掩模ROM
+ WAVEx：MR27C3252CZ 32M SOP44 掩模 ROM
+ ROMx：MR27C3252CZ 32M SOP44 掩模 ROM
+ FLx：MR27C3252CZ 32M SOP44 掩模 ROM
        KEYCUS：未填充
        IC4：GAL16V8B（PLCC20，无其他标记）

@@ -812,9 +812,9 @@ 系统 12 F2M5 PCB 8661962600 (8661972600)
  | |--------------------| |
  |-------------------------------------------------- ---------|
 笔记：
- WAVEx：64M SOP44 掩模ROM
+ WAVEx：64M SOP44掩模ROM
        FLx：英特尔闪存 DA28F640J5 64M FlashROM (SSOP56)
- ROMx：64M SOP44 MASKROM
+ ROMx：64M SOP44掩膜ROM
        IC3：CY37064 CPLD（PLCC44，标记为“KEYCUS”并印有“KC”和每个游戏不同的 3 位数字。）
        IC4：GAL22V10（PLCC28，PCB 标记为“S12F2M”，芯片标记为“S12F2M”）

@@ -1359,7 +1359,7 @@ void namcos12_state::namcos12_map(address_map &map)
         地图(0x1f000000, 0x1f000003).nopr();
         地图(0x1f000000, 0x1f000001).w(FUNC(namcos12_state::bankoffset_w)); /* 银行业务 */
         映射（0x1f080000，0x1f083fff）.rw（FUNC（namcos12_state :: sharedram_r），FUNC（namcos12_state :: sharedram_w））；/* 共享内存?? */
- 映射（0x1f140000，0x1f140fff）.rw（“at28c16”，FUNC（at28c16_device::read），FUNC（at28c16_device::write））.umask32（0x00ff00ff）；/* EEPROM */
+ 映射(0x1f140000, 0x1f140fff).rw("at28c16", FUNC(at28c16_device::read), FUNC(at28c16_device::write)).umask32(0x00ff00ff); /* EEPROM */
         地图(0x1f1bff08, 0x1f1bff0f).nopw(); /* ?? */
         地图(0x1f700000, 0x1f70ffff).w(FUNC(namcos12_state::dmaoffset_w)); /* DMA */
         /* 网络区域 */
@@ -1367,7 +1367,7 @@ void namcos12_state::namcos12_map(address_map &map)
 // 映射(0x1f796002, 0x1f796003).w(FUNC(namcos12_state::linkcpu_enable_w));
 // 映射(0x1f796022, 0x1f796023).w(FUNC(namcos12_state::linkcpu_disable_w));

- map(0x1fa00000, 0x1fbfffff).bankr("主银行"); /* 存储的 ROM */
+ map(0x1fa00000, 0x1fbfffff).bankr("主银行"); /* 存储的 ROM */
 }

 void namcos12_state::ptblank2_map(address_map &map)
@@ -2939,7 +2939,7 @@ ROM_END
```

```
 ROM_START（泰克塔格）
        ROM_REGION32_LE( 0x00400000, "maincpu:rom", 0 ) /* 主程序 */
- ROM_LOAD16_BYTE( "teg2verc1.2e", 0x0000000, 0x200000, CRC(c6da0717) SHA1(9e01ae64710d85eb9899d6fa6fd0a2152aee8c11) ) /* 修改为与 alt romboard 一起使用？*/
+ ROM_LOAD16_BYTE( "teg2verc1.2e", 0x0000000, 0x200000, CRC(c6da0717) SHA1(9e01ae64710d85eb9899d6fa6fd0a2 152aee8c11) ) /* 修改为与替代 ROM 板一起使用？*/
        ROM_LOAD16_BYTE（“teg2verc1.2j”，0x0000001，0x200000，CRC（25a1d2ff）SHA1（529a11a1bbb8655534d7ec371f1c09e9e387ed11））

        ROM_REGION32_LE( 0x3800000, "bankedroms", 0 ) /* 主要数据 */
@@ -2950,7 +2950,7 @@ ROM_START( tektagt )
        ROM_LOAD32_WORD（“teg1rom2e.11”，0x2000000，0x800000，CRC（6e5c3428）SHA1（e3cdb60a4445406877b2e273385f34bfb0974220））
        ROM_LOAD32_WORD（“teg1rom2o.15”，0x2000002，0x800000，CRC（21ce9dfa）SHA1（f27e8210ee236c327aa3e1ce4dd408abc6580a1b））

- ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 没有标签的闪存 */
+ ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 无标签的 Flash ROM */
        ROM_LOAD32_BYTE（“teg_fleu.5”，0x3000001，0x200000，CRC（36df0867）SHA1（6bec8560ad4c122dc909daa83aa9089ba5b281f7））
        ROM_LOAD32_BYTE（“teg_flol.6”，0x3000002，0x200000，CRC（03a76765）SHA1（ae35ae28375f2a3e52d72b77ec09750c326cc269））
        ROM_LOAD32_BYTE（“teg_flou.7”，0x3000003，0x200000，CRC（6d6947d1）SHA1（2f307bc4070fadb510c0473bc91d917b2d845ca5））
@@ -3007,7 +3007,7 @@ ROM_START( tektagtuc1 )
        ROM_LOAD32_WORD（“teg1rom2e.11”，0x2000000，0x800000，CRC（6e5c3428）SHA1（e3cdb60a4445406877b2e273385f34bfb0974220））
        ROM_LOAD32_WORD（“teg1rom2o.15”，0x2000002，0x800000，CRC（21ce9dfa）SHA1（f27e8210ee236c327aa3e1ce4dd408abc6580a1b））

- ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 没有标签的闪存 */
+ ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 无标签的 Flash ROM */
        ROM_LOAD32_BYTE（“teg_fleu.5”，0x3000001，0x200000，CRC（36df0867）SHA1（6bec8560ad4c122dc909daa83aa9089ba5b281f7））
        ROM_LOAD32_BYTE（“teg_flol.6”，0x3000002，0x200000，CRC（03a76765）SHA1（ae35ae28375f2a3e52d72b77ec09750c326cc269））
        ROM_LOAD32_BYTE（“teg_flou.7”，0x3000003，0x200000，CRC（6d6947d1）SHA1（2f307bc4070fadb510c0473bc91d917b2d845ca5））
@@ -3033,7 +3033,7 @@ ROM_START( tektagtub )
        ROM_LOAD32_WORD（“teg1rom2e.11”，0x2000000，0x800000，CRC（6e5c3428）SHA1（e3cdb60a4445406877b2e273385f34bfb0974220））
        ROM_LOAD32_WORD（“teg1rom2o.15”，0x2000002，0x800000，CRC（21ce9dfa）SHA1（f27e8210ee236c327aa3e1ce4dd408abc6580a1b））

- ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 没有标签的闪存 */
+ ROM_LOAD32_BYTE( "teg_flel.4", 0x3000000, 0x200000, CRC(88b3823c) SHA1(6f31acb642c57daccbfdb87b790037e261c8c73c) ) /* 无标签的 Flash ROM */
        ROM_LOAD32_BYTE（“teg_fleu.5”，0x3000001，0x200000，CRC（36df0867）SHA1（6bec8560ad4c122dc909daa83aa9089ba5b281f7））
        ROM_LOAD32_BYTE（“teg_flol.6”，0x3000002，0x200000，CRC（03a76765）SHA1（ae35ae28375f2a3e52d72b77ec09750c326cc269））
        ROM_LOAD32_BYTE（“teg_flou.7”，0x3000003，0x200000，CRC（6d6947d1）SHA1（2f307bc4070fadb510c0473bc91d917b2d845ca5））
@@ -3052,9 +3052,9 @@ ROM_START( tektagtjc1 )
        ROM_LOAD16_BYTE（“teg1verc1.2j”，0x0000001，0x200000，CRC（4ece9b9a）SHA1（7091dadfe3a2954e684fcc9e5a3337ecd26609f6））

        ROM_REGION32_LE( 0x3800000, "bankedroms", 0 ) /* 主要数据 */
- ROM_LOAD32_WORD( "teg1_rom0e.ic9", 0x0000000, 0x800000, BAD_DUMP CRC(c962a373) SHA1(d662dbd89ef62c5ac3150a018fc2d35ef2ee94ac) ) // 这些 rom 被转储一半大小，
+ ROM_LOAD32_WORD( "teg1_rom0e.ic9", 0x0000000, 0x800000, BAD_DUMP CRC(c962a373) SHA1(d662dbd89ef62c5ac3150a018fc2d35ef2ee94ac) ) // 这些 ROM 被转储一半大小，
        ROM_LOAD32_WORD( "teg1_rom0o.ic13", 0x0000002, 0x800000, BAD_DUMP CRC(badb7dcf) SHA1(8c0bf7f6351c5a2a0996df371a901cf90c68cd8c) ) // 可能是后半部分
- ROM_LOAD32_WORD( "teg1_rom1e.ic10", 0x1000000, 0x800000, BAD_DUMP CRC(b3d56124) SHA1(4df20c74ba63f7362caf15e9b8949fab655704fb) ) // 日本版 ROM 包
```

```
含
+ ROM_LOAD32_WORD( "teg1_rom1e.ic10", 0x1000000, 0x800000, BAD_DUMP CRC(b3d56124) SHA1(4df20c74ba63f7362caf15e9b8949fab655704fb) ) // 日本版 ROM 包
含
        ROM_LOAD32_WORD( "teg1_rom1o.ic14", 0x1000002, 0x800000, BAD_DUMP CRC(2434ceb6) SHA1(f19f1599acbd6fd48793a2ee5a500ca817d9df56) ) // 不同的
图形。
        ROM_LOAD32_WORD( "teg1_rom2e.ic11", 0x2000000, 0x800000, BAD_DUMP CRC(6e5c3428) SHA1(e3cdb60a4445406877b2e273385f34bfb0974220) ) //
        ROM_LOAD32_WORD( "teg1_rom2o.ic15", 0x2000002, 0x800000, BAD_DUMP CRC(21ce9dfa) SHA1(f27e8210ee236c327aa3e1ce4dd408abc6580a1b) ) //
diff --git a/src/mame/drivers/namcos2.cpp b/src/mame/drivers/namcos2.cpp
索引 ad7cebf00bc..2dd96a14440 100644
---a/src/mame/drivers/namcos2.cpp
+++ b/src/mame/drivers/namcos2.cpp
@@ -456,7 +456,7 @@ System 21 这里大概指的是Winning Run PCB，而不是后来的游戏？
 定制芯片： Final Lap Assault LuckyWld System21 NA1/2 NB1/2
     C45 陆地发电机 * *
     C65 I/O 控制器（旧版） * *
- C67 TMS320C25（DSP int ROM）
+ C67 TMS320C25（DSP 内部 ROM）
     C68 I/O 控制器（较新） * *
     C70 *
     C95 **
@@ -2335,7 +2335,7 @@ ROM_START(攻击)
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“atshape.bin”，0x000000，CRC（dfcad82b）SHA1（9c3826b8dc36fa0d71c0de7f8be3479d9a025803））

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K（“at1dat0.bin”，0x000000，CRC（844890f4）SHA1（1be30760acd81fae836301d81d6adbb3e5941373））
        NAMCOS2_DATA_LOAD_O_128K（“at1dat1.bin”，0x000000，CRC（21715313）SHA1（97c6edae6a5f1df434f1dcf7be307b5e006e72a6））

@@ -2386,7 +2386,7 @@ ROM_START( 攻击j )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“atshape.bin”，0x000000，CRC（dfcad82b）SHA1（9c3826b8dc36fa0d71c0de7f8be3479d9a025803））

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K（“at1dat0.bin”，0x000000，CRC（844890f4）SHA1（1be30760acd81fae836301d81d6adbb3e5941373））
        NAMCOS2_DATA_LOAD_O_128K（“at1dat1.bin”，0x000000，CRC（21715313）SHA1（97c6edae6a5f1df434f1dcf7be307b5e006e72a6））

@@ -2437,7 +2437,7 @@ ROM_START( 攻击 )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“atshape.bin”，0x000000，CRC（dfcad82b）SHA1（9c3826b8dc36fa0d71c0de7f8be3479d9a025803））

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K（“at1dat0.bin”，0x000000，CRC（844890f4）SHA1（1be30760acd81fae836301d81d6adbb3e5941373））
        NAMCOS2_DATA_LOAD_O_128K（“at1dat1.bin”，0x000000，CRC（21715313）SHA1（97c6edae6a5f1df434f1dcf7be307b5e006e72a6））
```

```
@@ -2487,7 +2487,7 @@ ROM_START(burnforc)
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K( "bu_shape.bin", 0x000000, CRC(80a6b722) SHA1(2c24327a890310c5e8086dc6821627108a88c62e) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "bu1_dat0.bin", 0x000000, CRC(e0a9d92f) SHA1(15042e6d7b31bec08ccdf36e89fdb4b6fb62fa4b) )
        NAMCOS2_DATA_LOAD_O_128K( "bu1_dat1.bin", 0x000000, CRC(5fe54b73) SHA1(a5d4895f0a4523be20de40ccaa74f8fad0d5df7d) )

@@ -2536,7 +2536,7 @@ ROM_START(burnforco)
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K( "bu_shape.bin", 0x000000, CRC(80a6b722) SHA1(2c24327a890310c5e8086dc6821627108a88c62e) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "bu1_dat0.bin", 0x000000, CRC(e0a9d92f) SHA1(15042e6d7b31bec08ccdf36e89fdb4b6fb62fa4b) )
        NAMCOS2_DATA_LOAD_O_128K( "bu1_dat1.bin", 0x000000, CRC(5fe54b73) SHA1(a5d4895f0a4523be20de40ccaa74f8fad0d5df7d) )

@@ -2578,7 +2578,7 @@ ROM_START( 科斯莫格 )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        ROM_LOAD( "co1sha0.bin", 0x000000, 0x80000, CRC(063a70cc) SHA1(c3179d55d57c47d3fef49d45e45b88c4d8250548) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "co1dat0.bin", 0x000000, CRC(b53da2ae) SHA1(a7fe63668d50928d5d2e2249a5f377c7e8dfc6a5) )
        NAMCOS2_DATA_LOAD_O_128K( "co1dat1.bin", 0x000000, CRC(d21ad10b) SHA1(dcf2d4cc048ea57507952a9a35390af7de5cfe34) )

@@ -2621,7 +2621,7 @@ ROM_START( cosmongj )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        ROM_LOAD( "co1sha0.bin", 0x000000, 0x80000, CRC(063a70cc) SHA1(c3179d55d57c47d3fef49d45e45b88c4d8250548) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "co1dat0.bin", 0x000000, CRC(b53da2ae) SHA1(a7fe63668d50928d5d2e2249a5f377c7e8dfc6a5) )
        NAMCOS2_DATA_LOAD_O_128K( "co1dat1.bin", 0x000000, CRC(d21ad10b) SHA1(dcf2d4cc048ea57507952a9a35390af7de5cfe34) )

@@ -2671,7 +2671,7 @@ ROM_START( dirtyfoxj )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K( "df1_sha.bin", 0x000000, CRC(9a7c9a9b) SHA1(06221ae8d3f6bebbb5a7ab2eaaf35b9922389115) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_256K( "df1_dat0.bin", 0x000000, CRC(f5851c85) SHA1(e99c05891622cdaab394630b7b2678968e6761d7) )
        NAMCOS2_DATA_LOAD_O_256K( "df1_dat1.bin", 0x000000, CRC(1a31e46b) SHA1(4be7115893b27d6a3dc38c97dcb41eafebb423cd) )

@@ -2715,7 +2715,7 @@ ROM_START( dsaber )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
```

```
        ROM_LOAD( "shape.bin", 0x000000, 0x80000, CRC(698e7a3e) SHA1(4d41bf0242626ca1448d1f650c84b5987a7f6597) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "do1 dat0.data0", 0x000000, CRC(3e53331f) SHA1(3dd4c133f587361f30ab1b890f5b05749d5838e3) )
        NAMCOS2_DATA_LOAD_O_128K（“do1 dat1.data1”，0x000000，CRC（d5427f11）SHA1（af8d8153dc60044616a6b0571831c53c09fefda1））

@@ -2763,7 +2763,7 @@ ROM_START( dsabera )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        ROM_LOAD( "shape.bin", 0x000000, 0x80000, CRC(698e7a3e) SHA1(4d41bf0242626ca1448d1f650c84b5987a7f6597) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "do1 dat0.data0", 0x000000, CRC(3e53331f) SHA1(3dd4c133f587361f30ab1b890f5b05749d5838e3) )
        NAMCOS2_DATA_LOAD_O_128K（“do1 dat1.data1”，0x000000，CRC（d5427f11）SHA1（af8d8153dc60044616a6b0571831c53c09fefda1））

@@ -2810,7 +2810,7 @@ ROM_START( dsaberj )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        ROM_LOAD( "shape.bin", 0x000000, 0x80000, CRC(698e7a3e) SHA1(4d41bf0242626ca1448d1f650c84b5987a7f6597) )

- ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
        NAMCOS2_DATA_LOAD_E_128K( "do1 dat0.data0", 0x000000, CRC(3e53331f) SHA1(3dd4c133f587361f30ab1b890f5b05749d5838e3) )
        NAMCOS2_DATA_LOAD_O_128K（“do1 dat1.data1”，0x000000，CRC（d5427f11）SHA1（af8d8153dc60044616a6b0571831c53c09fefda1））

@@ -2859,10 +2859,10 @@ ROM_START（最后一圈）
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“fl2-sha”，0x000000，CRC（5fda0b6d）SHA1（92c0410e159977ea73a8e8c0cb1321c3056f6c2f））

- ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
        /* ZIP 存档中不存在 DAT 文件 */

- ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
+ ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
        ROM_LOAD( "fl1-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

        ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
@@ -2911,10 +2911,10 @@ ROM_START( Finallapd )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“fl2-sha”，0x000000，CRC（5fda0b6d）SHA1（92c0410e159977ea73a8e8c0cb1321c3056f6c2f））

- ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
        /* ZIP 存档中不存在 DAT 文件 */

- ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
```

```
+ ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
        ROM_LOAD( "f11-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

        ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
@@ -2963,10 +2963,10 @@ ROM_START( 最终lapc )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“f12-sha”，0x000000，CRC（5fda0b6d）SHA1（92c0410e159977ea73a8e8c0cb1321c3056f6c2f））

- ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
        /* ZIP 存档中不存在 DAT 文件 */

- ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
+ ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
        ROM_LOAD( "f11-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

        ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
@@ -3015,10 +3015,10 @@ ROM_START( Finallapjc )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“f11_sha.bin”，0x000000，CRC（b7e1c7a3）SHA1（b82f9b340d95b80a12286647adba8c139b4d081a））

- ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
        /* ZIP 存档中不存在 DAT 文件 */

- ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
+ ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
        ROM_LOAD( "f11-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

        ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
@@ -3067,10 +3067,10 @@ ROM_START( Finallapjb )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_128K（“f11_sha.bin”，0x000000，CRC（b7e1c7a3）SHA1（b82f9b340d95b80a12286647adba8c139b4d081a））

- ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
+ ROM_REGION16_BE( 0x200000, "data_rom", ROMREGION_ERASEFF ) /* 共享数据 ROM */
        /* ZIP 存档中不存在 DAT 文件 */

- ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
+ ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
        ROM_LOAD( "f11-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

        ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
@@ -3122,11 +3122,11 @@ ROM_START( Finalap2 )
        ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
        NAMCOS2_GFXROM_LOAD_256K（“fls2sha”，0x000000，CRC（f7b40a85）SHA1（a458a1cc0dae757fe8a15cb5f5ae46d3c033df00））
```

```
 - ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
 + ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
         NAMCOS2_DATA_LOAD_E_256K（“fls2dat0”，0x000000，CRC（f1af432c）SHA1（c514261a49ceb5c3ba0246519ba5d02e9a20d950））
         NAMCOS2_DATA_LOAD_O_256K（“fls2dat1”，0x000000，CRC（8719533e）SHA1（98d2767da6f7f67da7af15e8cfed95adb04b7427））

 - ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
 + ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
         ROM_LOAD( "fl1-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

         ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
 @@ -3177,11 +3177,11 @@ ROM_START( Finalap2j )
         ROM_REGION( 0x080000, "c123tmap:mask", 0 ) /* 掩模形状 */
         NAMCOS2_GFXROM_LOAD_256K（“fls2sha”，0x000000，BAD_DUMP CRC（f7b40a85）SHA1（a458a1cc0dae757fe8a15cb5f5ae46d3c033df00））

 - ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据ROM */
 + ROM_REGION16_BE( 0x200000, "data_rom", 0 ) /* 共享数据 ROM */
         NAMCOS2_DATA_LOAD_E_256K（“fls2dat0”，0x000000，CRC（f1af432c）SHA1（c514261a49ceb5c3ba0246519ba5d02e9a20d950））
         NAMCOS2_DATA_LOAD_O_256K（“fls2dat1”，0x000000，CRC（8719533e）SHA1（98d2767da6f7f67da7af15e8cfed95adb04b7427））

 - ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的舞会 */
 + ROM_REGION( 0x100, "c45_road:clut", 0 ) /* 道路颜色的 PROM */
         ROM_LOAD( "fl1-3.5b", 0, 0x100, CRC(d179d99a) SHA1(4e64f284c74d2b77f893bd28aaa6489084056aa2) )

         ROM_REGION( 0x100000, "c140", 0 ) /* 声音 */
 @@ -3690,7 +3690,7 @@ 注：（显示所有 IC）
        J2 - 2 针连接器，用 1 条电线连接到视频板上的逻辑芯片。如果连接器
             拔掉电源后，所有自行车精灵都坏了。场景和观众对象不受影响。
        所有 ROM 均为 2MBit。1A、2A、5A、2C 和 6C 处的 ROM 为 27C2001 EPROM，所有其他
 - 是 2M 32 针 MASK ROM。
 + 是 2M 32 引脚掩模 ROM。
  */

  ROM_START（四线）
 diff --git a/src/mame/drivers/namcos22.cpp b/src/mame/drivers/namcos22.cpp
 索引 d0ed4ed556f..a625242023b 100644
 ---a/src/mame/drivers/namcos22.cpp
 +++ b/src/mame/drivers/namcos22.cpp
 @@ -275,7 +275,7 @@
   *
   *笔记：
   * J6 : 用于插件程序 ROM PCB 的定制 Namco 连接器
 - * J11：用于可选插件 WAVE ROM PCB 的定制 Namco 连接器（容纳一些 SOP44 MASKROM）
 + * J11：定制 Namco 连接器，用于可选插件 WAVE ROM PCB（容纳一些 SOP44 掩模 ROM）
   * JC410：定制 Namco 连接器，用于可选插入式辅助 PCB（例如，时间危机中使用的枪支控制 PCB）
   * ETC）
   * 连接器仅安装在第 2 版 CPU (B) PCB 上 8646962600 (8646972600)
 @@ -340,20 +340,20 @@
```

```
   * 东京大战“TW1 DATA”
   *
   * WAVEA.2L \
 - * WAVEB.1L / : 16M/32M WAVE MASKROM。如果是 32MBit DIP42，则以字节模式 (DIP42/SOP44) 进行编程
 + * WAVEB.1L / : 16M/32M WAVE 掩模 ROM。如果是 32MBit DIP42，则以字节模式 (DIP42/SOP44) 进行编程
   * 游戏A波B类型
   * ---------------------------------------------- ---------------------------
 - * Air Combat 22 'ACS1 WAVE0'、'ACS1 WAVE1'，均为 SOP44 32M MASKROM
 - * Alpine Racer 1 'AR1 WAVEA' , , DIP42 16M MASKROM
 - * Alpine Racer 2 'ARS1 WAVEA'、'ARS2 WAVE B'，均为 DIP42 32M MASKROM
 - * 高山冲浪者'AF1 WAVEA' , , DIP42 32M MASKROM
 - * Aqua Jet“AJ1 WAVEA”、“AJ1 WAVEB”，均为 DIP42 32M MASKROM
 - * Armadillo Racing 'AM1 WAVEA' 、 'AM1 WAVEB' ，均为 DIP42 32M MASKROM。原型版本使用 TSOP56，安装在 DIP48 适配器板上
 - * Cyber Cycles 'CB1 WAVEA' , 'CB1 WAVEB' , WAVE A DIP42 32M MASKROM, WAVE B DIP42 16M MASKROM
 - * Dirt Dash 'DT1 WAVEA' 、 'DT1 WAVEB' ，均为 DIP42 32M MASKROM
 - * Prop Cycle 'PR1 WAVE A', 'PR1 WAVE B' , 均为 DIP42 32M MASKROM
 - * 时间危机“TS1 WAVE A”、“TS1 WAVE B”、WAVE A DIP42 32M MASKROM、WAVE B DIP42 16M MASKROM
 - * 东京大战 'TW1 WAVE A',, DIP42 32M MASKROM
 + * Air Combat 22 'ACS1 WAVE0'、'ACS1 WAVE1' ，均为 SOP44 32M 掩模 ROM
 + * Alpine Racer 1 'AR1 WAVEA' , , DIP42 16M 掩模 ROM
 + * Alpine Racer 2 'ARS1 WAVEA'、'ARS2 WAVE B'，均为 DIP42 32M 掩模 ROM
 + * Alpine Surfer 'AF1 WAVEA' , , DIP42 32M 面罩 ROM
 + * Aqua Jet 'AJ1 WAVEA' 、 'AJ1 WAVEB' ，均为 DIP42 32M 掩模 ROM
 + * Armadillo Racing 'AM1 WAVEA' 、 'AM1 WAVEB' ，均为 DIP42 32M 掩模 ROM。原型版本使用 TSOP56，安装在 DIP48 适配器板上
 + * Cyber Cycles 'CB1 WAVEA' , 'CB1 WAVEB' , WAVE A DIP42 32M mask ROM, WAVE B DIP42 16M mask ROM
 + * Dirt Dash 'DT1 WAVEA' 、 'DT1 WAVEB' ，均为 DIP42 32M 掩模 ROM
 + * Prop Cycle 'PR1 WAVE A', 'PR1 WAVE B' , 均为 DIP42 32M mask ROM
 + * 时间危机 'TS1 WAVE A'、'TS1 WAVE B'、WAVE A DIP42 32M mask ROM、WAVE B DIP42 16M mask ROM
 + * 东京大战 'TW1 WAVE A',, DIP42 32M 面具 ROM
   *
   *
   *程序ROM子板PCB
 diff --git a/src/mame/drivers/namcos23.cpp b/src/mame/drivers/namcos23.cpp
 索引 dfdfad54fcc..907028c18aa 100644
 ---a/src/mame/drivers/namcos23.cpp
 +++ b/src/mame/drivers/namcos23.cpp
 @@ -30,7 +30,7 @@
      - 序列号数据位于 BIOS 中的偏移量 0x201 处。直到比赛开始
        并展示它，但我不会干涉它。

 - - 在Gunmen Wars中添加sh2（无rom，控制相机）
 + - 在Gunmen Wars中添加sh2（无ROM，控制相机）

      - 超级系统 23 在帖子中测试 irqs。timecrs2v4a的代码可以
      可能测试 7 个源，但实际上只测试 5 个。
 @@ -151,7 +151,7 @@ System 23 单元由以下一些部分组成.....
                                    模拟控件（按钮/操纵杆/电位器等）。
```

```
 - V185 I/O PCB 枪 I/O 板与 Time Crisis II 一起使用
 - V221 MIU PCB Gun I/O 板与 Crisis Zone（系统 23 Evolution 2）和 Time Crisis 3（系统 246 上）一起使用
-- SYSTEM23 MEM(M) PCB 容纳用于 GFX/Sound 和相关逻辑的 MASKROM
+- SYSTEM23 MEM(M) PCB 保存用于 GFX/Sound 和相关逻辑的掩模 ROM
                                    请注意，在 Super System23 中，重复使用 System23 中的 MEM(M) PCB。
                                    在 Super System23 上，System23 部件上有一张标有“SystemSuper23”的贴纸，还有一张
                                    PAL 未填充。
@@ -164,7 +164,7 @@ System 23 单元由以下一些部分组成.....
 容纳这些 PCB 的金属盒的尺寸与 Super System 22 大致相同。但是，该盒大部分是
 空的。所有 CPU/视频/DSP 硬件均位于主 PCB 上，其尺寸与
 超级系统 22 CPU 板。ROM PCB 的尺寸是 Super System22 ROM PCB 的一半。ROM 位于其上
- 可配置为 32MBit 或 64MBit SOP44 MASK ROM，最大容量为 1664MBits。
+可配置为32MBit或64MBit SOP44掩膜ROM，最大容量为1664MBit。
 该系统还使用类似于 Super System 22 的双管道图形总线，并具有两个图形 ROM 副本
 PCB 上。
 System 23 硬件是第一个需要外部 3.3V 电源的 NAMCO 系统。以前是3.3V
@@ -3888,7 +3888,7 @@ ROM_START(rapidrvr)
        ROM_LOAD( "rd1wavel.2s", 0x000000, 0x800000, CRC(bf52c08c) SHA1(6745062e078e520484390fad1f723124aa4076d0) )
        ROM_LOAD( "rd1waveh.3s", 0x800000, 0x800000, CRC(ef0136b5) SHA1(a6d923ededca168fe555e0b86a72f53bec5424cc) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“rd1cgll.8f”，0x000000，0x800000，CRC（b58b92ac）SHA1（70ee6e0e5347e05817aa30d53d766b8ce0fc44e4））
        ROM_LOAD（“rd1cglm.7f”，0x000000，0x800000，CRC（447067fa）SHA1（e2052373773594feb303e1924a4a820cf34ab55b））
        ROM_LOAD（“rd1cgum.6f”，0x000000，0x800000，CRC（c50de2ef）SHA1（24758a72b3569ce6a643a5786fce7c34b8aa692d））
@@ -3949,7 +3949,7 @@ ROM_START(rapidrvrv2c)
        ROM_LOAD( "rd1wavel.2s", 0x000000, 0x800000, CRC(bf52c08c) SHA1(6745062e078e520484390fad1f723124aa4076d0) )
        ROM_LOAD( "rd1waveh.3s", 0x800000, 0x800000, CRC(ef0136b5) SHA1(a6d923ededca168fe555e0b86a72f53bec5424cc) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“rd1cgll.8f”，0x000000，0x800000，CRC（b58b92ac）SHA1（70ee6e0e5347e05817aa30d53d766b8ce0fc44e4））
        ROM_LOAD（“rd1cglm.7f”，0x000000，0x800000，CRC（447067fa）SHA1（e2052373773594feb303e1924a4a820cf34ab55b））
        ROM_LOAD（“rd1cgum.6f”，0x000000，0x800000，CRC（c50de2ef）SHA1（24758a72b3569ce6a643a5786fce7c34b8aa692d））
@@ -4010,7 +4010,7 @@ ROM_START(rapidrvrp) // 原型板
        ROM_LOAD( "rd1wavel.2s", 0x000000, 0x800000, CRC(bf52c08c) SHA1(6745062e078e520484390fad1f723124aa4076d0) )
        ROM_LOAD( "rd1waveh.3s", 0x800000, 0x800000, CRC(ef0136b5) SHA1(a6d923ededca168fe555e0b86a72f53bec5424cc) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“rd1cgll.8f”，0x000000，0x800000，CRC（b58b92ac）SHA1（70ee6e0e5347e05817aa30d53d766b8ce0fc44e4））
        ROM_LOAD（“rd1cglm.7f”，0x000000，0x800000，CRC（447067fa）SHA1（e2052373773594feb303e1924a4a820cf34ab55b））
        ROM_LOAD（“rd1cgum.6f”，0x000000，0x800000，CRC（c50de2ef）SHA1（24758a72b3569ce6a643a5786fce7c34b8aa692d））
@@ -4065,7 +4065,7 @@ ROM_START( finfurl )
        ROM_LOAD( "ff2wavel.2s", 0x000000, 0x800000, CRC(6235c605) SHA1(521eaee80ac17c0936877d49394e5390fa0ff8a0) )
        ROM_LOAD( "ff2waveh.3s", 0x800000, 0x800000, CRC(2a59492a) SHA1(886ec0a4a71048d65f93c52df96416e74d23b3ec) )
```

```
- ROM_REGION( 0x400000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x400000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “ff2cguu.5f”, 0x000000, 0x400000, CRC（595deee4）SHA1（b29ff9c6ba17737f1f87c05b2d899d80b0b72dbb）)
        ROM_LOAD ( “ff2cgum.6f”, 0x000000, 0x400000, CRC（b808be59）SHA1（906bfbb5d34feef9697da545a93930fe6e56685c）)
        ROM_LOAD ( “ff2cgll.8f”, 0x000000, 0x400000, CRC（8e6c34eb）SHA1（795631c8019011246ed1e5546de4433dc22dd9e7）)
@@ -4093,7 +4093,7 @@ ROM_START( motoxgo )
        ROM_REGION( 0x20000, "exioboard", 0 ) /* “额外” I/O 板（使用 Fujitsu MB90611A MCU） */
        ROM_LOAD ( “mg1prog0a.3a”, 0x000000, 0x020000, CRC（b2b5be8f）SHA1（803652b7b8fde2196b7fb742ba8b9843e4fcd2de）)

- ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “mg1mtah.2j”, 0x000000, 0x800000, CRC（845f4768）SHA1（9c03b1f6dcd9d1f43c2958d855221be7f9415c47）)
        ROM_LOAD16_BYTE ( “mg1mtal.2h”, 0x000001, 0x800000, CRC（fdad0f0a）SHA1（420d50f012af40f80b196d3aae320376e6c32367）)

@@ -4118,7 +4118,7 @@ ROM_START( motoxgo )
        ROM_LOAD( "mg1wavel.2c", 0x000000, 0x800000, CRC(f78b1b4d) SHA1(47cd654ec0a69de0dc81b8d83692eebf5611228b) )
        ROM_LOAD( "mg1waveh.2a", 0x800000, 0x800000, CRC(8cb73877) SHA1(2e2b170c7ff889770c13b4ab7ac316b386ada153) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “mg1cgll.5m”, 0x000000, 0x800000, CRC（175dfe34）SHA1（66ae35b0084159aea1afeb1a6486fffa635992b5）)
        ROM_LOAD ( “mg1cglm.5k”, 0x000000, 0x800000, CRC（b3e648e7）SHA1（98018ae2276f905a7f74e1dab540a44247524436）)
        ROM_LOAD ( “mg1cgum.5j”, 0x000000, 0x800000, CRC（46a77d73）SHA1（132ce2452ee68ba374e98b59032ac0a1a277078d）)
@@ -4141,7 +4141,7 @@ ROM_START( motoxgov2a )
        ROM_REGION( 0x20000, "exioboard", 0 ) /* “额外” I/O 板（使用 Fujitsu MB90611A MCU） */
        ROM_LOAD ( “mg1prog0a.3a”, 0x000000, 0x020000, CRC（b2b5be8f）SHA1（803652b7b8fde2196b7fb742ba8b9843e4fcd2de）)

- ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “mg1mtah.2j”, 0x000000, 0x800000, CRC（845f4768）SHA1（9c03b1f6dcd9d1f43c2958d855221be7f9415c47）)
        ROM_LOAD16_BYTE ( “mg1mtal.2h”, 0x000001, 0x800000, CRC（fdad0f0a）SHA1（420d50f012af40f80b196d3aae320376e6c32367）)

@@ -4166,7 +4166,7 @@ ROM_START( motoxgov2a )
        ROM_LOAD( "mg1wavel.2c", 0x000000, 0x800000, CRC(f78b1b4d) SHA1(47cd654ec0a69de0dc81b8d83692eebf5611228b) )
        ROM_LOAD( "mg1waveh.2a", 0x800000, 0x800000, CRC(8cb73877) SHA1(2e2b170c7ff889770c13b4ab7ac316b386ada153) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “mg1cgll.5m”, 0x000000, 0x800000, CRC（175dfe34）SHA1（66ae35b0084159aea1afeb1a6486fffa635992b5）)
        ROM_LOAD ( “mg1cglm.5k”, 0x000000, 0x800000, CRC（b3e648e7）SHA1（98018ae2276f905a7f74e1dab540a44247524436）)
        ROM_LOAD ( “mg1cgum.5j”, 0x000000, 0x800000, CRC（46a77d73）SHA1（132ce2452ee68ba374e98b59032ac0a1a277078d）)
@@ -4188,7 +4188,7 @@ ROM_START( motoxgov2a2 )
        ROM_REGION( 0x20000, "exioboard", 0 ) /* “额外” I/O 板（使用 Fujitsu MB90611A MCU） */
        ROM_LOAD ( “mg1prog0a.3a”, 0x000000, 0x020000, CRC（b2b5be8f）SHA1（803652b7b8fde2196b7fb742ba8b9843e4fcd2de）)

- ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据 ROM */
```

```
        ROM_LOAD16_BYTE（“mg1mtah.2j”, 0x000000, 0x800000, CRC（845f4768）SHA1（9c03b1f6dcd9d1f43c2958d855221be7f9415c47））
        ROM_LOAD16_BYTE（“mg1mtal.2h”, 0x000001, 0x800000, CRC（fdad0f0a）SHA1（420d50f012af40f80b196d3aae320376e6c32367））

@@ -4213,7 +4213,7 @@ ROM_START( motoxgov2a2 )
        ROM_LOAD( "mg1wavel.2c", 0x000000, 0x800000, CRC(f78b1b4d) SHA1(47cd654ec0a69de0dc81b8d83692eebf5611228b) )
        ROM_LOAD( "mg1waveh.2a", 0x800000, 0x800000, CRC(8cb73877) SHA1(2e2b170c7ff889770c13b4ab7ac316b386ada153) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“mg1cgll.5m”, 0x000000, 0x800000, CRC（175dfe34）SHA1（66ae35b0084159aea1afeb1a6486fffa635992b5））
        ROM_LOAD（“mg1cglm.5k”, 0x000000, 0x800000, CRC（b3e648e7）SHA1（98018ae2276f905a7f74e1dab540a44247524436））
        ROM_LOAD（“mg1cgum.5j”, 0x000000, 0x800000, CRC（46a77d73）SHA1（132ce2452ee68ba374e98b59032ac0a1a277078d））
@@ -4235,7 +4235,7 @@ ROM_START( motoxgov1a )
        ROM_REGION( 0x20000, "exioboard", 0 ) /* “额外” I/O 板（使用 Fujitsu MB90611A MCU） */
        ROM_LOAD（“mg1prog0a.3a”, 0x000000, 0x020000, CRC（b2b5be8f）SHA1（803652b7b8fde2196b7fb742ba8b9843e4fcd2de））

- ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据 ROM */
        ROM_LOAD16_BYTE（“mg1mtah.2j”, 0x000000, 0x800000, CRC（845f4768）SHA1（9c03b1f6dcd9d1f43c2958d855221be7f9415c47））
        ROM_LOAD16_BYTE（“mg1mtal.2h”, 0x000001, 0x800000, CRC（fdad0f0a）SHA1（420d50f012af40f80b196d3aae320376e6c32367））

@@ -4260,7 +4260,7 @@ ROM_START( motoxgov1a )
        ROM_LOAD( "mg1wavel.2c", 0x000000, 0x800000, CRC(f78b1b4d) SHA1(47cd654ec0a69de0dc81b8d83692eebf5611228b) )
        ROM_LOAD( "mg1waveh.2a", 0x800000, 0x800000, CRC(8cb73877) SHA1(2e2b170c7ff889770c13b4ab7ac316b386ada153) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“mg1cgll.5m”, 0x000000, 0x800000, CRC（175dfe34）SHA1（66ae35b0084159aea1afeb1a6486fffa635992b5））
        ROM_LOAD（“mg1cglm.5k”, 0x000000, 0x800000, CRC（b3e648e7）SHA1（98018ae2276f905a7f74e1dab540a44247524436））
        ROM_LOAD（“mg1cgum.5j”, 0x000000, 0x800000, CRC（46a77d73）SHA1（132ce2452ee68ba374e98b59032ac0a1a277078d））
@@ -4283,7 +4283,7 @@ ROM_START( motoxgov1a2 )
        ROM_REGION( 0x20000, "exioboard", 0 ) /* “额外” I/O 板（使用 Fujitsu MB90611A MCU） */
        ROM_LOAD（“mg1prog0a.3a”, 0x000000, 0x020000, CRC（b2b5be8f）SHA1（803652b7b8fde2196b7fb742ba8b9843e4fcd2de））

- ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", ROMREGION_ERASEFF ) /* 数据 ROM */
        ROM_LOAD16_BYTE（“mg1mtah.2j”, 0x000000, 0x800000, CRC（845f4768）SHA1（9c03b1f6dcd9d1f43c2958d855221be7f9415c47））
        ROM_LOAD16_BYTE（“mg1mtal.2h”, 0x000001, 0x800000, CRC（fdad0f0a）SHA1（420d50f012af40f80b196d3aae320376e6c32367））

@@ -4308,7 +4308,7 @@ ROM_START( motoxgov1a2 )
        ROM_LOAD( "mg1wavel.2c", 0x000000, 0x800000, CRC(f78b1b4d) SHA1(47cd654ec0a69de0dc81b8d83692eebf5611228b) )
        ROM_LOAD( "mg1waveh.2a", 0x800000, 0x800000, CRC(8cb73877) SHA1(2e2b170c7ff889770c13b4ab7ac316b386ada153) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD（“mg1cgll.5m”, 0x000000, 0x800000, CRC（175dfe34）SHA1（66ae35b0084159aea1afeb1a6486fffa635992b5））
        ROM_LOAD（“mg1cglm.5k”, 0x000000, 0x800000, CRC（b3e648e7）SHA1（98018ae2276f905a7f74e1dab540a44247524436））
```

```
         ROM_LOAD（“mg1cgum.5j”, 0x000000, 0x800000, CRC（46a77d73）SHA1（132ce2452ee68ba374e98b59032ac0a1a277078d））
@@ -4328,7 +4328,7 @@ ROM_START( timecrs2 )
         ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
         ROM_LOAD（“tssioprog.ic3”, 0x000000, 0x040000, CRC（edad4538）SHA1（1330189184a636328d956c0e435f8d9ad2e96a80））

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
         ROM_LOAD16_BYTE（“tss1mtah.2j”, 0x0000000, 0x800000, CRC（697c26ed）SHA1（72f6f69e89496ba0c6183b35c3bde71f5a3c721f））
         ROM_LOAD16_BYTE（“tss1mtal.2h”, 0x0000001, 0x800000, CRC（bfc79190）SHA1（04bda00c4cc5660d27af4f3b0ee3550dea8d3805））
         ROM_LOAD16_BYTE（“tss1mtbh.2m”, 0x1000000, 0x800000, CRC（82582776）SHA1（7c790d09bac660ea1c62da3ffb21ab43f2461594））
@@ -4371,7 +4371,7 @@ ROM_START( timecrs2v2b )
         ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
         ROM_LOAD（“tssioprog.ic3”, 0x000000, 0x040000, CRC（edad4538）SHA1（1330189184a636328d956c0e435f8d9ad2e96a80））

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
         ROM_LOAD16_BYTE（“tss1mtah.2j”, 0x0000000, 0x800000, CRC（697c26ed）SHA1（72f6f69e89496ba0c6183b35c3bde71f5a3c721f））
         ROM_LOAD16_BYTE（“tss1mtal.2h”, 0x0000001, 0x800000, CRC（bfc79190）SHA1（04bda00c4cc5660d27af4f3b0ee3550dea8d3805））
         ROM_LOAD16_BYTE（“tss1mtbh.2m”, 0x1000000, 0x800000, CRC（82582776）SHA1（7c790d09bac660ea1c62da3ffb21ab43f2461594））
@@ -4414,7 +4414,7 @@ ROM_START( timecrs2v1b )
         ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
         ROM_LOAD（“tssioprog.ic3”, 0x000000, 0x040000, CRC（edad4538）SHA1（1330189184a636328d956c0e435f8d9ad2e96a80））

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
         ROM_LOAD16_BYTE（“tss1mtah.2j”, 0x0000000, 0x800000, CRC（697c26ed）SHA1（72f6f69e89496ba0c6183b35c3bde71f5a3c721f））
         ROM_LOAD16_BYTE（“tss1mtal.2h”, 0x0000001, 0x800000, CRC（bfc79190）SHA1（04bda00c4cc5660d27af4f3b0ee3550dea8d3805））
         ROM_LOAD16_BYTE（“tss1mtbh.2m”, 0x1000000, 0x800000, CRC（82582776）SHA1（7c790d09bac660ea1c62da3ffb21ab43f2461594））
@@ -4457,7 +4457,7 @@ ROM_START( timecrs2v4a )
         ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
         ROM_LOAD（“tssioprog.ic3”, 0x000000, 0x040000, CRC（edad4538）SHA1（1330189184a636328d956c0e435f8d9ad2e96a80））

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
         ROM_LOAD16_BYTE（“tss1mtah.2j”, 0x0000000, 0x800000, CRC（697c26ed）SHA1（72f6f69e89496ba0c6183b35c3bde71f5a3c721f））
         ROM_LOAD16_BYTE（“tss1mtal.2h”, 0x0000001, 0x800000, CRC（bfc79190）SHA1（04bda00c4cc5660d27af4f3b0ee3550dea8d3805））
         ROM_LOAD16_BYTE（“tss1mtbh.2m”, 0x1000000, 0x800000, CRC（82582776）SHA1（7c790d09bac660ea1c62da3ffb21ab43f2461594））
@@ -4500,7 +4500,7 @@ ROM_START( timecrs2v5a )
         ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
         ROM_LOAD（“tssioprog.ic3”, 0x000000, 0x040000, CRC（edad4538）SHA1（1330189184a636328d956c0e435f8d9ad2e96a80））

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
         ROM_LOAD16_BYTE（“tss1mtah.2j”, 0x0000000, 0x800000, CRC（697c26ed）SHA1（72f6f69e89496ba0c6183b35c3bde71f5a3c721f））
         ROM_LOAD16_BYTE（“tss1mtal.2h”, 0x0000001, 0x800000, CRC（bfc79190）SHA1（04bda00c4cc5660d27af4f3b0ee3550dea8d3805））
         ROM_LOAD16_BYTE（“tss1mtbh.2m”, 0x1000000, 0x800000, CRC（82582776）SHA1（7c790d09bac660ea1c62da3ffb21ab43f2461594））
@@ -4541,9 +4541,9 @@ ROM_START( aking )
```

```
        ROM_LOAD16_WORD_SWAP( "ag1vera.ic3", 0x000000, 0x080000, CRC(266ac71c) SHA1(648a64adc0e4a2cefd71c31a6a71359b6c196430) )

        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 MB90F574 MCU 代码 */
- ROM_LOAD( "fcaf10.bin", 0x000000, 0x040000, NO_DUMP ) // 256KB 内部闪存
+ ROM_LOAD( "fcaf10.bin", 0x000000, 0x040000, NO_DUMP ) // 256KB 内部闪存 ROM

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "ag1mtah.2j", 0x0000000, 0x800000, CRC(f2d8ca9d) SHA1(8158d13d74f2aae7c0d1238619ce1ad3a17d8047) )
        ROM_LOAD16_BYTE( "ag1mtal.2h", 0x0000001, 0x800000, CRC(7facbfd4) SHA1(c42988e274a1b4f40f4b4379e94653ef07429c58) )
        ROM_LOAD16_BYTE( "ag1mtbh.2m", 0x1000000, 0x800000, CRC(890bdb52) SHA1(a38f039187448ee328547582eab22813ce625615) )
@@ -4587,9 +4587,9 @@ ROM_START( 500gp )
        ROM_LOAD16_WORD_SWAP( "5gp3verc.3", 0x000000, 0x080000, CRC(b323abdf) SHA1(8962e39b48a7074a2d492afb5db3f5f3e5ae2389) )

        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 MB90F574 MCU 代码 */
- ROM_LOAD( "fcaf10.bin", 0x000000, 0x040000, NO_DUMP ) // 256KB 内部闪存
+ ROM_LOAD( "fcaf10.bin", 0x000000, 0x040000, NO_DUMP ) // 256KB 内部闪存 ROM

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "5gp1mtah.2j", 0x0000000, 0x800000, CRC(246e4b7a) SHA1(75743294b8f48bffb84f062febfbc02230d49ce9) )
        ROM_LOAD16_BYTE( "5gp1mtal.2h", 0x0000001, 0x800000, CRC(1bb00c7b) SHA1(922be45d57330c31853b2dc1642c589952b09188) )
        ROM_LOAD16_BYTE( "5gp1mtbh.2m", 0x1000000, 0x800000, CRC(352360e8) SHA1(d621dfac3385059c52d215f6623901589a8658a3) )
@@ -4637,7 +4637,7 @@ ROM_START( 比赛 )
        ROM_REGION( 0x80000, "ffb", 0 ) /* STR 转向力反馈板代码 */
        ROM_LOAD( "ro1_str-0a.ic16", 0x000000, 0x080000, CRC(27d39e1f) SHA1(6161cbb27c964ffab1db3b3c1f073ec514876e61) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "ro1mtah.2j", 0x000000, 0x800000, CRC(216abfb1) SHA1(8db7b17dc6441adc7a4ec8b941d5a84d73c735d6) )
        ROM_LOAD16_BYTE( "ro1mtal.2h", 0x000001, 0x800000, CRC(17646306) SHA1(8d1af777f8e884b650efee8e4c26e032e1c088b7) )

@@ -4688,7 +4688,7 @@ ROM_START( finfurl2 )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
        ROM_LOAD( "asca-3a.ic14", 0x000000, 0x040000, CRC(8e9266e5) SHA1(ffa8782ca641d71d57df23ed1c5911db05d3df97) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "ffs1mtah.2j", 0x0000000, 0x800000, CRC(f336d81d) SHA1(a9177091e1412dea1b6ea6c53530ae31361b32d0) )
        ROM_LOAD16_BYTE( "ffs1mtal.2h", 0x0000001, 0x800000, CRC(98730ad5) SHA1(9ba276ad88ec8730edbacab80cdacc34a99593e4) )
        ROM_LOAD16_BYTE( "ffs1mtbh.2m", 0x1000000, 0x800000, CRC(0f42c93b) SHA1(26b313fc5c33afb0a1ee42243486e38f052c95c2) )
@@ -4733,7 +4733,7 @@ ROM_START( finfurl2j )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
        ROM_LOAD( "asca-3a.ic14", 0x000000, 0x040000, CRC(8e9266e5) SHA1(ffa8782ca641d71d57df23ed1c5911db05d3df97) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
```

```
        ROM_LOAD16_BYTE( "ffs1mtah.2j", 0x0000000, 0x800000, CRC(f336d81d) SHA1(a9177091e1412dea1b6ea6c53530ae31361b32d0) )
        ROM_LOAD16_BYTE( "ffs1mtal.2h", 0x0000001, 0x800000, CRC(98730ad5) SHA1(9ba276ad88ec8730edbacab80cdacc34a99593e4) )
        ROM_LOAD16_BYTE( "ffs1mtbh.2m", 0x1000000, 0x800000, CRC(0f42c93b) SHA1(26b313fc5c33afb0a1ee42243486e38f052c95c2) )
@@ -4777,7 +4777,7 @@ ROM_START(panicprk)
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
        ROM_LOAD( "asca-3a.ic14", 0x000000, 0x040000, CRC(8e9266e5) SHA1(ffa8782ca641d71d57df23ed1c5911db05d3df97) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "pnp1mtah.2j", 0x000000, 0x800000, CRC(37addddd) SHA1(3032989653304417df80606bc3fde6e9425d8cbb) )
        ROM_LOAD16_BYTE( "pnp1mtal.2h", 0x000001, 0x800000, CRC(6490faaa) SHA1(03443746009b434e5d4074ea6314910418907360) )

@@ -4803,7 +4803,7 @@ ROM_START(panicprk)
        ROM_LOAD( "pnp1wavel.2c", 0x000000, 0x800000, CRC(35c6a9bd) SHA1(4b56fdc37525c15e57d93091e6609d6a6905fc5c) )
        ROM_LOAD( "pnp1waveh.2a", 0x800000, 0x800000, CRC(6fa1826a) SHA1(20a5af49e65ae2bc57c016b5cd9bafa5a5220d35) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD( "pnp1cguu.4f", 0x000000, 0x800000, CRC(cd64f57f) SHA1(8780270298e0823db1acbbf79396788df0c3c19c) )
        ROM_LOAD( "pnp1cgum.5j", 0x000000, 0x800000, CRC(206217ca) SHA1(9c095bba7764f3405c3fab10513b9b78981ec44d) )
        ROM_LOAD( "pnp1cgll.5m", 0x000000, 0x800000, CRC(d03932cf) SHA1(49240e44923cc6e815e9457b6290fd18466658af) )
@@ -4824,7 +4824,7 @@ ROM_START(panicprkj)
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O板HD643334 H8/3334 MCU代码 */
        ROM_LOAD( "asca-3a.ic14", 0x000000, 0x040000, CRC(8e9266e5) SHA1(ffa8782ca641d71d57df23ed1c5911db05d3df97) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "pnp1mtah.2j", 0x000000, 0x800000, CRC(37addddd) SHA1(3032989653304417df80606bc3fde6e9425d8cbb) )
        ROM_LOAD16_BYTE( "pnp1mtal.2h", 0x000001, 0x800000, CRC(6490faaa) SHA1(03443746009b434e5d4074ea6314910418907360) )

@@ -4850,7 +4850,7 @@ ROM_START(panicprkj)
        ROM_LOAD( "pnp1wavel.2c", 0x000000, 0x800000, CRC(35c6a9bd) SHA1(4b56fdc37525c15e57d93091e6609d6a6905fc5c) )
        ROM_LOAD( "pnp1waveh.2a", 0x800000, 0x800000, CRC(6fa1826a) SHA1(20a5af49e65ae2bc57c016b5cd9bafa5a5220d35) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD( "pnp1cguu.4f", 0x000000, 0x800000, CRC(cd64f57f) SHA1(8780270298e0823db1acbbf79396788df0c3c19c) )
        ROM_LOAD( "pnp1cgum.5j", 0x000000, 0x800000, CRC(206217ca) SHA1(9c095bba7764f3405c3fab10513b9b78981ec44d) )
        ROM_LOAD( "pnp1cgll.5m", 0x000000, 0x800000, CRC(d03932cf) SHA1(49240e44923cc6e815e9457b6290fd18466658af) )
@@ -4871,7 +4871,7 @@ ROM_START( 枪战 )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。"ASCA-5；版本 2.09；日本，多用途"*/
        ROM_LOAD( "asc5_io-a.ic14", 0x000000, 0x020000, CRC(5964767f) SHA1(320db5e78ae23c5f94e368432d51573b409995db) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE( "gm1mtah.2j", 0x000000, 0x800000, CRC(3cea9094) SHA1(497395425e409de47e1114de9aeeaf05e4f6a9a1) )
        ROM_LOAD16_BYTE( "gm1mtal.2h", 0x000001, 0x800000, CRC(d531dfcd) SHA1(9f7cbe9a03c1f7649bf05a7a30d47511573b50ba) )
```

```
@@ -4896,7 +4896,7 @@ ROM_START( 枪战 )
        ROM_REGION( 0x1000000, "c352", 0 ) /* C352 PCM 样本 */
        ROM_LOAD( "gm1wave.2c", 0x000000, 0x800000, CRC(7d5c79a4) SHA1(b800a46bcca10cb0d0d9e0acfa68af63ae64dcaf) )

-       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “gm1cguu.4f”, 0x000000, 0x800000, CRC（26a74698）SHA1（3f07d273abb3f2552dc6a29300f5dc2f2744c852）)
        ROM_LOAD ( “gm1cgum.5j”, 0x000000, 0x800000, CRC（a7728944）SHA1（c187c6d66128554fcecc96e81d4f5396197e8280）)
        ROM_LOAD ( “gm1cgll.5m”, 0x000000, 0x800000, CRC（936c0079）SHA1（3aec8caada35b7ed790bb3a8bcf6e01cad068fcd）)
@@ -4917,7 +4917,7 @@ ROM_START(gunwarsa)
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“ASCA-5；版本 2.09；日本，多用途”*/
        ROM_LOAD ( “asc5_io-a.ic14”, 0x000000, 0x020000, CRC（5964767f）SHA1（320db5e78ae23c5f94e368432d51573b409995db）)

-       ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+       ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “gm1mtah.2j”, 0x000000, 0x800000, CRC（3cea9094）SHA1（497395425e409de47e1114de9aeeaf05e4f6a9a1）)
        ROM_LOAD16_BYTE ( “gm1mtal.2h”, 0x000001, 0x800000, CRC（d531dfcd）SHA1（9f7cbe9a03c1f7649bf05a7a30d47511573b50ba）)

@@ -4942,7 +4942,7 @@ ROM_START(gunwarsa)
        ROM_REGION( 0x1000000, "c352", 0 ) /* C352 PCM 样本 */
        ROM_LOAD( "gm1wave.2c", 0x000000, 0x800000, CRC(7d5c79a4) SHA1(b800a46bcca10cb0d0d9e0acfa68af63ae64dcaf) )

-       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “gm1cguu.4f”, 0x000000, 0x800000, CRC（26a74698）SHA1（3f07d273abb3f2552dc6a29300f5dc2f2744c852）)
        ROM_LOAD ( “gm1cgum.5j”, 0x000000, 0x800000, CRC（a7728944）SHA1（c187c6d66128554fcecc96e81d4f5396197e8280）)
        ROM_LOAD ( “gm1cgll.5m”, 0x000000, 0x800000, CRC（936c0079）SHA1（3aec8caada35b7ed790bb3a8bcf6e01cad068fcd）)
@@ -4963,7 +4963,7 @@ ROM_START( 下坡 )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“ASCA-3；Ver 2.04；JPN，多用途 + 旋转编码器”*/
        ROM_LOAD ( “asc3_io-c.ic14”, 0x000000, 0x020000, CRC（2f272a7b）SHA1（9d7ebe274c0d26f5f38747224d42d0375e2ed14c）)

-       ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+       ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “dh1mtah.2j”, 0x000000, 0x800000, CRC（3b56faa7）SHA1（861db7f549bedbb2b837516fcc966ad5890007ce）)
        ROM_LOAD16_BYTE ( “dh1mtal.2h”, 0x000001, 0x800000, CRC（9fa07bfe）SHA1（a6b847ff7d5eadbf60b434a0d905051ea4227113）)

@@ -4991,7 +4991,7 @@ ROM_START( 下坡 )
        ROM_LOAD( "dh1wavel.2c", 0x000000, 0x800000, CRC(10954726) SHA1(50ee0346c46194dada7b5c0d8b1efe9a7f211b90) )
        ROM_LOAD( "dh1waveh.2a", 0x800000, 0x800000, CRC(2adfa312) SHA1(d01a46af2c95d1ea64e9778979ae147298d921e3) )

-       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+       ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD( "dh1cguu.4f", 0x000000, 0x800000, CRC(66cb0dd7) SHA1(1f67320f150f1b55c97eae4b9fe4890fabc8dc7e) )
        ROM_LOAD ( “dh1cgum.5j”, 0x000000, 0x800000, CRC（1044d0a0）SHA1（e0bf843616e166495fcdc76f076eb53a28287d30）)
        ROM_LOAD ( “dh1cgll.5m”, 0x000000, 0x800000, CRC（c0d5ad87）SHA1（bc1992516c63aebdae0322def77f082d799a327a）)
@@ -5012,7 +5012,7 @@ ROM_START( crszone )
```

```
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “csz1mtah.2j” , 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( “csz1mtal.2h” , 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( “csz1mtbh.2m” , 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5044,7 +5044,7 @@ ROM_START( crszone )
        ROM_LOAD ( “csz1wavel.2c” , 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( “csz1waveh.2a” , 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “csz1cguu.4f” , 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( “csz1cgum.5j” , 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( “csz1cgll.5m” , 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
@@ -5065,7 +5065,7 @@ ROM_START( crszonev4a )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “csz1mtah.2j” , 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( “csz1mtal.2h” , 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( “csz1mtbh.2m” , 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5097,7 +5097,7 @@ ROM_START( crszonev4a )
        ROM_LOAD ( “csz1wavel.2c” , 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( “csz1waveh.2a” , 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “csz1cguu.4f” , 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( “csz1cgum.5j” , 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( “csz1cgll.5m” , 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
@@ -5118,7 +5118,7 @@ ROM_START( crszonev3b )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “csz1mtah.2j” , 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( “csz1mtal.2h” , 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( “csz1mtbh.2m” , 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5150,7 +5150,7 @@ ROM_START( crszonev3b )
        ROM_LOAD ( “csz1wavel.2c” , 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( “csz1waveh.2a” , 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )
```

```
- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “csz1cguu.4f” , 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( “csz1cgum.5j” , 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( “csz1cgll.5m” , 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
@@ -5171,7 +5171,7 @@ ROM_START( crszonev3b2 )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “csz1mtah.2j” , 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( “csz1mtal.2h” , 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( “csz1mtbh.2m” , 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5203,7 +5203,7 @@ ROM_START( crszonev3b2 )
        ROM_LOAD ( “csz1wavel.2c” , 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( “csz1waveh.2a” , 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “csz1cguu.4f” , 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( “csz1cgum.5j” , 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( “csz1cgll.5m” , 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
@@ -5224,7 +5224,7 @@ ROM_START( crszonev3a )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( “csz1mtah.2j” , 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( “csz1mtal.2h” , 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( “csz1mtbh.2m” , 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5256,7 +5256,7 @@ ROM_START( crszonev3a )
        ROM_LOAD ( “csz1wavel.2c” , 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( “csz1waveh.2a” , 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( “csz1cguu.4f” , 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( “csz1cgum.5j” , 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( “csz1cgll.5m” , 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
@@ -5277,7 +5277,7 @@ ROM_START( crszonev2a )
        ROM_REGION( 0x40000, "iocpu", 0 ) /* I/O 板 HD643334 H8/3334 MCU 代码。“MIU-I/O; Ver2.05; JPN, 枪扩展”*/
        ROM_LOAD ( “csz1prg0a.8f” , 0x000000, 0x020000, CRC (8edc36b3) SHA1 (b5df211988d856572fcc313480e693c8561784e4) )

- ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据ROM */
```

```
+ ROM_REGION32_BE( 0x2000000, "data", 0 ) /* 数据 ROM */
        ROM_LOAD16_BYTE ( "csz1mtah.2j", 0x0000000, 0x800000, CRC (66b076ad) SHA1 (edd32e0b380f01a9626d32f5eec860f841c8be8a) )
        ROM_LOAD16_BYTE ( "csz1mtal.2h", 0x0000001, 0x800000, CRC (38dc639a) SHA1 (aa9b5b35174c1b007a57a4bd7a53bc3f479b5b71) )
        ROM_LOAD16_BYTE ( "csz1mtbh.2m", 0x1000000, 0x800000, CRC (bdec4188) SHA1 (a098651fbd8a69a0afc17f4b6c93350926cacd6b) )
@@ -5309,7 +5309,7 @@ ROM_START( crszonev2a )
        ROM_LOAD ( "csz1wavel.2c", 0x000000, 0x800000, CRC (d0d74132) SHA1 (a293d93bca8e12e388a088a592cfa7bcb9a976f7) )
        ROM_LOAD ( "csz1waveh.2a", 0x800000, 0x800000, CRC (de9d14a8) SHA1 (e5006861928bb1d29bf80c7304f1a6d044b094fd) )

- ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
+ ROM_REGION( 0x800000, "dups", 0 ) /* 重复的 ROM */
        ROM_LOAD ( "csz1cguu.4f", 0x000000, 0x800000, CRC (e1d1bf24) SHA1 (daf2c68e2d9a8f313d262d221cc990c93dfdf22f) )
        ROM_LOAD ( "csz1cgum.5j", 0x000000, 0x800000, CRC (913c98b5) SHA1 (b952dbc19053796077d4f33e8da836893e933b12) )
        ROM_LOAD ( "csz1cgll.5m", 0x000000, 0x800000, CRC (0bcd41f2) SHA1 (80b74f9398e8bd074f79a14490d06cfeb875c874) )
diff --git a/src/mame/drivers/naomi.cpp b/src/mame/drivers/naomi.cpp
索引 768c3c0d4b2..ba034b07f82 100644
---a/src/mame/drivers/naomi.cpp
+++ b/src/mame/drivers/naomi.cpp
@@ -361,7 +361,7 @@ PFSB 64M 掩模ROM板
 |------------------------------------------------ ---------|
 笔记：
       购物车 PCB 的另一侧有更多位置
- SOP44 MASKROM...IC12S 至 IC21S（ROM12 至 ROM21）
+ SOP44 掩膜 ROM...IC12S 至 IC21S（ROM12 至 ROM21）

   IC1-IC21S - MaskROM (SOP44), 32Mb 或 64Mb。并非所有位置都已填充
       IC22 - EPROM (DIP42), 27C160 或 27C322
@@ -374,7 +374,7 @@ 注意：
    CN1/2/3 - 连接到主板的连接器

 已知使用该 PCB 的游戏包括.....
- 贴纸 EPROM MASKROM X76F100 XC9536 315-5881
+ 贴纸 EPROM 掩模 ROM X76F100 XC9536 315-5881
 购物车上的游戏 IC22# SOP44 IC37# IC41# IC42# 备注
 ------------------------------------------------ ------------------------------------------------ --------------------------
 18 Wheeler（豪华版）（修订版 A）840-0023C 22185A 20 (64Mb) 现有 315-6213 317-0273-COM
@@ -538,7 +538,7 @@ 从NAOMI这边读为X76F100型，可能是由ACTEL FPGA模拟的。
 实际数据已打乱，地址位 3 至 6 的顺序相反，即 8 7 3 4 5 6 2 1 0。

 已知使用该 PCB 的游戏包括.....
- 贴纸 EPROM MASKROM 25LC040 A54SX32
+ 贴纸 EPROM 掩模 ROM 25LC040 A54SX32
 游戏推车 IC11# SOP44 IC13S# IC1# 备注
 ------------------------------------------------ ------------------------------------------------ ----------------------------------------
 卡丁车俱乐部：欧洲会议（2003 年，修订版 A）840-0139C 24173A 18 (64Mb) 目前 317-0382-COM
@@ -591,7 +591,7 @@ 注意：
    CN1/2/3 - 连接到主板的连接器
```

```
    已知使用该 PCB 的游戏包括......
- 贴纸 EPROM MASKROM EPM7032 XC9536 315-5881 X76F100
+ 贴纸 EPROM 掩模 ROM EPM7032 XC9536 315-5881 X76F100
 购物车上的游戏 IC22# SOP44 IC27# IC41# IC42# IC46# 备注
 ---------------------------------------------- ---------------------------------------------------- ----------------------------------------
----
 外星人前线（修订版 A）840-0048C 23586A 5 (128Mb) 315-6319A 315-6213 317-0293-COM 存在
@@ -723,7 +723,7 @@ 注意：
                丝印 VOYAGER64。看起来相当于 Sega 推车上的 IC11/22
           2K - NAODEC1B (QFP100) Altera MAX EPM7064S。丝印NAODEC1A
           3J - 振荡器 28.000MHz
-4B-4N, 6B-6P - 掩模ROM (TSOP48), 128Mb。并非所有位置都已填充。丝印MASK128MT
+4B-4N, 6B-6P - 掩模 ROM (TSOP48), 128Mb。并非所有位置都已填充。丝印MASK128MT
       4P,5P - 静态随机存储器 (SOJ28) 32kx8, ISSI IS61C256AH-15J
    CN1/2/3 - 连接到主板的连接器

@@ -779,7 +779,7 @@ 注意：
                数字末尾，-JPN 表示需要日文 BIOS，-COM 可以运行任何 BIOS
 2B、2C、2D、2F - DA28F640J5 FlashROM (SSOP56), 32Mb 或 64Mb。并非所有位置都已填充。
                丝印 VOYAGER64。看起来相当于 Sega 推车上的 IC11/22
-4B-4M, 6B-6N - 掩模ROM (TSOP48), 128Mb。并非所有位置都已填充。丝印MASK128MT
+4B-4M, 6B-6N - 掩模 ROM (TSOP48), 128Mb。并非所有位置都已填充。丝印MASK128MT
       4N,4P - 静态随机存储器 (SOJ28) 32kx8, ISSI IS61C256AH-15J
    CN1/2/3 - 连接到主板的连接器

@@ -1288,7 +1288,7 @@ 注意：
       XC9536 - Xilinx XC9536 系统内可编程 CPLD (PLCC44), 印有
                游戏代码。对于每个不同的游戏，此代码都不同。
                最后 3 位数字似乎是用于用途。
- F01 = CPLD/保护器件且 M01 = MASKROM
+ F01 = CPLD/保护器件且 M01 = 掩模 ROM

                游戏（按代码排序） 代码
                ------------------------------------------------------------
@@ -1370,7 +1370,7 @@ 注意：
    XCR3128XL - Xilinx XCR3128XL 系统内可编程 128 宏单元 CPLD (TQFP100)
                刻有游戏代码。对于每个不同的游戏，此代码都不同。
                最后 3 位数字似乎是用于用途。
- F01 = CPLD/保护器件且 M01 = MASKROM
+ F01 = CPLD/保护器件且 M01 = 掩模 ROM

                游戏（按代码排序） 代码
                ------------------------------------------------------------
diff --git a/src/mame/drivers/nitedrvr.cpp b/src/mame/drivers/nitedrvr.cpp
索引 a65d3ac2cbc..b7c7af0ab08 100644
---a/src/mame/drivers/nitedrvr.cpp
+++ b/src/mame/drivers/nitedrvr.cpp
```

```
@@ -187,8 +187,8 @@ ROM_END

 ROM_START（nitedrvr）
        ROM_REGION( 0x10000, "主CPU", 0 )
- ROM_LOAD( "006569-01.d2", 0x9000, 0x0800, CRC(7afa7542) SHA1(81018e25ebdeae1daf1308676661063b6fd7fd22) ) // 掩码 ROM 1
- ROM_LOAD( "006570-01.f2", 0x9800, 0x0800, CRC(bf5d77b1) SHA1(6f603f8b0973bd89e0e721b66944aac8e9f904d9) ) // 掩码 ROM 2
+ ROM_LOAD( "006569-01.d2", 0x9000, 0x0800, CRC(7afa7542) SHA1(81018e25ebdeae1daf1308676661063b6fd7fd22) ) // 掩码 ROM 1
+ ROM_LOAD( "006570-01.f2", 0x9800, 0x0800, CRC(bf5d77b1) SHA1(6f603f8b0973bd89e0e721b66944aac8e9f904d9) ) // 掩码 ROM 2
        ROM_RELOAD( 0xf800, 0x0800 ) // 向量

        ROM_REGION( 0x200, "gfx1", 0 )
diff --git a/src/mame/drivers/nmk16.cpp b/src/mame/drivers/nmk16.cpp
索引 6edff8f8de2..7baf37ce6d6 100644
---a/src/mame/drivers/nmk16.cpp
+++ b/src/mame/drivers/nmk16.cpp
@@ -1868,7 +1868,7 @@ INPUT_PORTS_END

 静态 INPUT_PORTS_START（ hachamfp ）
        PORT_INCLUDE(hachamfb)
-
+
        PORT_MODIFY（“DSW1”）
        PORT_DIPNAME( 0x0001, 0x0001, DEF_STR( 翻转屏幕 ) ) PORT_DIPLOCATION("SW1:8")
        PORT_DIPSETTING（0x0001, DEF_STR（关））
@@ -5108,7 +5108,7 @@ void nmk16_state::init_ssmissin()

 无效 nmk16_state::init_bjtwin()
 {
- /* 修补 ROM 以启用测试模式 */
+ /* 修补 ROM 以启用测试模式 */

 /* 008F54: 33F9 0008 0000 000F FFFC move.w $80000.l, $ffffc.l
  * 008F5E: 3639 0008 0002 move.w $80002.l, D3
@@ -5207,7 +5207,7 @@ void nmk16_state::afega_map(address_map &map)
        映射（0x0f0000, 0x0fffff）.ram（）.w（FUNC（nmk16_state :: nmk16_mainram_strange_w））.share（“mainram”）;
 }

-// firehawk 有 0x100000 字节的程序 ROM（至少是可切换版本），所以上面的方法无法工作。
+// firehawk 有 0x100000 字节的程序 ROM（至少是可切换版本），所以上面的方法不起作用。
 无效 nmk16_state::firehawk_map(address_map &map)
 {
        map.global_mask(0x3fffff);
@@ -5900,7 +5900,7 @@ ROM_END

 /*

-SBS Gomorrah（以及已安装正确 rom 的 Bio-ship Paladin）
```

```
 +SBS Gomorrah（以及具有正确 ROM 的 Bio-ship Paladin）
  UPL, 1993

  PCB布局
 @@ -6449,7 +6449,7 @@ ROM_START( hachamfb ) /* 雷龙转换 - 未受保护的原型或 bo
         /* 0x20000 - 0x80000 存入 */
  ROM_END

 -ROM_START( hachamfp ) /* 原型位置测试发布; 带有各种日期的手写标签。68K 程序 rom 有 19th Sep. 1991 字符串。*/
 +ROM_START( hachamfp ) /* 原型位置测试发布; 带有各种日期的手写标签。68K 程序 ROM 有 1991 年 9 月 19 日字符串。*/
         ROM_REGION( 0x40000, "maincpu", 0 ) /* 68000 代码 */
         rom_load16_byte（“ KF-68-PE-B.IC7”，0x00000，0x20000，CRC（B98A525E）SHA1（161C3B3360068E606E4D4D4104CC172B9736A52EEB）
         ROM_LOAD16_BYTE( "kf-68-po-b.ic6", 0x00001, 0x20000, CRC(b62ad179) SHA1(60a66fb9eb3fc792d172e1f4507a806ac2ad4217) ) /* 标签显示“KF 9/25 II
 68 PO B” */
 @@ -6634,13 +6634,13 @@ ROM_END

  ROM_START( Macross2k ) /* 标题画面只显示超时空要塞 II，没有汉字。疑似韩语版本 - 舞台信息屏幕仍使用语言倾斜 */
         ROM_REGION( 0x80000, "maincpu", 0 ) /* 68000 代码 */
 - ROM_LOAD16_WORD_SWAP( "1.3", 0x00000, 0x80000, CRC(1506fcfc) SHA1(638ccc90effde3be20ab9b4da3a0d75af2577e51) ) /* 非描述性 ROM 标签 "1" */
 + ROM_LOAD16_WORD_SWAP( "1.3", 0x00000, 0x80000, CRC(1506fcfc) SHA1(638ccc90effde3be20ab9b4da3a0d75af2577e51) ) /* 非描述性 ROM 标签 "1" */

         ROM_REGION( 0x20000, "audiocpu", 0 ) /* Z80 代码 */
         ROM_LOAD( "mcrs2j.2", 0x00000, 0x20000, CRC(b4aa8ac7) SHA1(73a6de56cbfb468450d9b39fcbae0362f242f37b) ) /* 存储 */

         ROM_REGION( 0x020000, "fgtile", 0 )
 - ROM_LOAD( "2.1", 0x000000, 0x020000, CRC(e8ab17f9) SHA1(9396e29a134698db59b7faae19dd8fb947cde752) ) /* 8x8 块 - 非描述性 ROM 标签“2”*/
 + ROM_LOAD( "2.1", 0x000000, 0x020000, CRC(e8ab17f9) SHA1(9396e29a134698db59b7faae19dd8fb947cde752) ) /* 8x8 块 - 非描述性 ROM 标签 "2" */

         ROM_REGION( 0x200000, "bgtile", 0 )
         ROM_LOAD( "bp932an.a04", 0x000000, 0x200000, CRC(c4d77ff0) SHA1(aca60a3f5f89265e7e3799e5d80ea8196fb11ff3) ) /* 16x16 块 */
 @@ -6728,7 +6728,7 @@ ROM_START( tdragon3h )
         ROM_REGION( 0x020000, "fgtile", 0 )
         ROM_LOAD( "12.27c1000", 0x000000, 0x020000, CRC(f809d616) SHA1(c6a4d776fee770ec197204b855b85bcc719469a5) ) /* 8x8 块 */

 - // 所有其他 ROM 都是标记为“CONNY”的 MASK 部件，但并未从该 PCB 中转储，因此仅假定内容相同
 + // 所有其他 ROM 都是标记为“CONNY”的掩模部件，但并未从该 PCB 中转储，因此仅假定内容相同

         ROM_REGION( 0x200000, "bgtile", 0 )
         ROM_LOAD( "ww930914.2", 0x000000, 0x200000, CRC(f968c65d) SHA1(fd6d21bba53f945b1597d7d0735bc62dd44d5498) ) /* 16x16 块 */
 @@ -6812,7 +6812,7 @@ NMK, 1994

  除了 3 个 PROM 之外，主板根本没有 ROM。有插件女儿
  包含所有 ROM 的板。它具有 3 个插槽 EPROMS 和 7 个插槽的容量
 总共 -16M MASK ROM。
 总共 +16M 掩模 ROM。
```

```
 PCB布局（主板）
@@ -7551,8 +7551,8 @@ AFEGA4.U112 27C020 + M68000 程序代码
 AFEGA5.U107 27C020 |

 AFEGA3.UC13 ST M27C160 - 精灵
-AF1-B2.UC8 MASK ROM 读取为 27C160 - 背景
-AF1-B1.UC3 MASK ROM 读取为 27C160 - 背景
+AF1-B2.UC8 掩模 ROM 读取为 27C160 - 背景
+AF1-B1.UC3 掩模 ROM 读取为 27C160 - 背景

 森金的ROM：

@@ -7564,9 +7564,9 @@ GST-03.U4 27C512 - 图形/文本层
 GST-04.U112 27C2000+M68000程序代码
 GST-05.U107 27C2000 |

-AF1-SP.UC13 MASK ROM 读取为 27C160 - 精灵
-AF1-B2.UC8 MASK ROM 读取为 27C160 - 背景
-AF1-B1.UC3 MASK ROM 读取为 27C160 - 背景
+AF1-SP.UC13 掩模 ROM 读取为 27C160 - 精灵
+AF1-B2.UC8 掩模 ROM 读取为 27C160 - 背景
+AF1-B1.UC3 掩模 ROM 读取为 27C160 - 背景

 **************************************************** **************************/

@@ -7581,14 +7581,14 @@ ROM_START( grdnstrm )
         ROM_LOAD16_BYTE（“afega5.u107”，0x000001，0x040000，CRC（5815c806）SHA1（f6b7809b2e3b29b89289ecc994909434fe34e10d））

         ROM_REGION( 0x10000, "audiocpu", 0 ) /* Z80 代码 */
-        ROM_LOAD( "afega7.u92", 0x00000, 0x10000, CRC(5d8cf28e) SHA1(2a440bf5136f95af137b6688e566a14e65be94b1) ) /* MASK ROM (读为 27C020) */
+        ROM_LOAD( "afega7.u92", 0x00000, 0x10000, CRC(5d8cf28e) SHA1(2a440bf5136f95af137b6688e566a14e65be94b1) ) /* 掩码 ROM (读为 27C020) */

         ROM_REGION( 0x200000, "精灵", 0 ) /* 精灵, 16x16x4 */
         ROM_LOAD( "afega3.uc13", 0x000000, 0x200000, CRC(0218017c) SHA1(5a8a4f07cd3f9dcf62455ddaceaec0cfba8c2de9) ) /* ST M27C160 EPROM */

         ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层, 16x16x8 */
-        ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
-        ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* MASK ROM (读为 27C160) */
+        ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
+        ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* 掩码 ROM (读为 27C160) */

         ROM_REGION( 0x10000, "fgtile", 0 ) /* 第 1 层，8x8x4 */
         ROM_LOAD（“afega1.u4”，0x00000，0x10000，CRC（9e7ef086）SHA1（db086bb2ceb11f3e24548aa131cc74fe79a2b516））
@@ -7606,11 +7606,11 @@ ROM_START( grdnstrmk )
         ROM_LOAD（“afega7.u92”，0x00000，0x10000，CRC（5d8cf28e）SHA1（2a440bf5136f95af137b6688e566a14e65be94b1））

         ROM_REGION( 0x200000, "精灵", 0 ) /* 精灵, 16x16x4 */
```

```
- ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* MASK ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* 掩码 ROM (读为 27C160) */

        ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层, 16x16x8 */
- ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
- ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* MASK ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* 掩码 ROM (读为 27C160) */

        ROM_REGION( 0x10000, "fgtile", 0 ) /* 第 1 层, 8x8x4 */
        ROM_LOAD（“gst-03.u4”，0x00000，0x10000，CRC（a1347297）SHA1（583f4da991eeedeb523cf4fa3b6900d40e342063））
@@ -7628,11 +7628,11 @@ ROM_START( grdnstrmj )
        ROM_LOAD（“afega7.u92”，0x00000，0x10000，CRC（5d8cf28e）SHA1（2a440bf5136f95af137b6688e566a14e65be94b1））

        ROM_REGION( 0x200000, "精灵", 0 ) /* 精灵, 16x16x4 */
- ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* MASK ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* 掩码 ROM (读为 27C160) */

        ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层, 16x16x8 */
- ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
- ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* MASK ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* 掩码 ROM (读为 27C160) */

        ROM_REGION( 0x10000, "fgtile", 0 ) /* 第 1 层, 8x8x4 */
        ROM_LOAD（“gst-03.u4”，0x00000，0x10000，CRC（a1347297）SHA1（583f4da991eeedeb523cf4fa3b6900d40e342063））
@@ -7647,14 +7647,14 @@ ROM_START( grdnstrmv ) /* Apples Industries 许可证 - 垂直版本 */
        ROM_LOAD16_BYTE（“afega3.u107”，0x000001，0x040000，CRC（05920a99）SHA1（ee77da303d6b766c529c426a836777827ac31676））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* Z80 代码 */
- ROM_LOAD( "afega7.u92", 0x00000, 0x10000, CRC(5d8cf28e) SHA1(2a440bf5136f95af137b6688e566a14e65be94b1) ) /* MASK ROM (读为 27C020) */
+ ROM_LOAD( "afega7.u92", 0x00000, 0x10000, CRC(5d8cf28e) SHA1(2a440bf5136f95af137b6688e566a14e65be94b1) ) /* 掩码 ROM (读为 27C020) */

        ROM_REGION( 0x200000, "精灵", 0 ) /* 精灵, 16x16x4 */
        ROM_LOAD( "afega6.uc13", 0x000000, 0x200000, CRC(9b54ff84) SHA1(9e120d85cf2fa899e6426dcb4302c8051746facc) ) /* ST M27C160 EPROM */

        ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层, 16x16x8 */
- ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
- ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* MASK ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM (读为 27C160) */
+ ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* 掩码 ROM (读为 27C160) */

        ROM_REGION( 0x10000, "fgtile", 0 ) /* 第 1 层, 8x8x4 */
        ROM_LOAD（“afega1.u4”，0x00000，0x10000，CRC（9e7ef086）SHA1（db086bb2ceb11f3e24548aa131cc74fe79a2b516））
@@ -7687,7 +7687,7 @@ ROM_START( grdnstrmg ) /* 德国 */
        ROM_LOAD16_BYTE（“gs10_br4.uc11”，0x100001，0x080000，CRC（1d3b57e1）SHA1（a2da598d6cbe257de5b66905a5ad9de90711ccc7））
        ROM_LOAD16_BYTE（“gs9_br2.uc4”，0x100000，0x080000，CRC（4d2c220b）SHA1（066067f7e80973ba0483559ac04f99292cc82dce））
```

```
- // 其他组在这里有更大的区域，因为它们在 rom 中包含 2 组图块，每个方向一组。
+ // 其他组在这里有更大的区域，因为它们在 ROM 中包含 2 组图块，每个方向一组。
         // 该集合仅包含所需方向的图块数据。
         ROM_REGION( 0x200000, "bgtile", 0 ) /* 第 0 层，16x16x8 */
         ROM_LOAD（“gs10_cr5.uc15”，0x000000，0x080000，CRC（2c8c23e3）SHA1（4c1a460dfc250f9aea77e2ddd82278ee816365be））
@@ -7712,11 +7712,11 @@ ROM_START( redfoxwp2 )
         ROM_LOAD( "u92", 0x00000, 0x10000, CRC(864b55c2) SHA1(43475b05e35549ad301c3d4a25d4f4f0bcbe3f2c) ) /* 华邦 W27E512-12 无标签 */

         ROM_REGION( 0x200000, "精灵", 0 ) /* 精灵，16x16x4 */
- ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* MASK ROM（读为 27C160）*/
+ ROM_LOAD( "afega_af1-sp.uc13", 0x000000, 0x200000, CRC(7d4d4985) SHA1(15c6c1aecd3f12050c1db2376f929f1a26a1d1cf) ) /* 掩码 ROM（读为 27C160）*/

         ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层，16x16x8 */
- ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM（读为 27C160）*/
- ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* MASK ROM（读为 27C160）*/
+ ROM_LOAD( "afega_af1-b2.uc8", 0x000000, 0x200000, CRC(d68588c2) SHA1(c5f397d74a6ecfd2e375082f82e37c5a330fba62) ) /* 掩码 ROM（读为 27C160）*/
+ ROM_LOAD( "afega_af1-b1.uc3", 0x200000, 0x200000, CRC(f8b200a8) SHA1(a6c43dd57b752d87138d7125b47dc0df83df8987) ) /* 掩码 ROM（读为 27C160）*/

         ROM_REGION( 0x10000, "fgtile", 0 ) /* 第 1 层，8x8x4 */
         ROM_LOAD( "u4", 0x00000, 0x10000, CRC(19239401) SHA1(7876335dd97418bd9130dc894a517f3ceca20135) ) /* 华邦 W27E512-12 无标签 */
@@ -7760,7 +7760,7 @@ ROM_END

 Afega 的流行音乐 (1999)

- PCB 插座中可能缺少 eprom
+ PCB 插座中可能缺少 EPROM
   --- 我只是认为它使用通用 PCB，但在这种情况下没有精灵。

 1x 68k
@@ -7784,7 +7784,7 @@ ROM_START( popspops )
         ROM_LOAD（“afega1.u92”，0x00000，0x10000，CRC（5d8cf28e）SHA1（2a440bf5136f95af137b6688e566a14e65be94b1））

         ROM_REGION( 0x400000, "精灵", ROMREGION_ERASEFF ) /* 精灵，16x16x4 */
- /* 没有精灵ROM？*/
+ /* 没有精灵 ROM？*/

         ROM_REGION( 0x400000, "bgtile", 0 ) /* 第 0 层，16x16x8 */
         ROM_LOAD（“afega6.uc8”，0x000000，0x200000，CRC（6d506c97）SHA1（4909c0b530f9526c8bf76e502c914ef10a50d1fc））
@@ -7846,18 +7846,18 @@ CPU：TMP68HC000P-10 (68000)
 声音：Z840006（Z80，44 针 QFP），YM2151，OKI M6295
 振荡器：4.000MHz、12.000MHz
 DIPSW：8 位 (x2)
-RAM：6116（x5，gfx相关？）6116（x1，声音程序RAM），6116（x1，靠近rom3）
+RAM：6116（x5，gfx 相关？）6116（x1，声音程序 RAM），6116（x1，靠近 ROM 3）
          64256（x4，gfx相关？），62256（x2，主程序RAM），6264（x2，gfx相关？）
 PAL/PROM：无
```

```
 定制：未知 208 引脚 QFP，标记为 LTC2（图形生成器）
- 未知 68 针 PLCC，标记为 LTC1（？，靠近 rom 2 和 rom 3）
+ 未知 68 针 PLCC，标记为 LTC1（？，靠近 ROM 2 和 ROM 3）
 ROM：

 文件名类型 可能的用途
 ----------------------------------------------------------
 rom01.92 27C512 声音程序
 rom02.95 27C020 冲电气样品
-rom03.4 27C512? （位于 rom 1 和 2 附近以及 LTC1 附近）
+rom03.4 27C512? （位于 ROM 1 和 2 附近以及 LTC1 附近）
 ROM04.1 27C040 \
 ROM05.3 27C040 |
 ROM06.6 27C040 |
@@ -8197,7 +8197,7 @@ YONATech3 是 MX27C4000
 YONATech2 和 YONATech4 是 TMS27C010A
 YONATech5 和 YONATech6 是 TMS27C020

-UC1、UC2 和 UC3 均为 Micronix MX29F1610ML 16Mb Flash rom
+UC1、UC2 和 UC3 均为 Micronix MX29F1610ML 16Mb Flash ROM

 UC1、UC2 和 UC3 具有适用于 MX29F1610 闪存和 27C160 EPROM 的焊盘

@@ -8364,7 +8364,7 @@ 游戏( 1996, tdragon3h, tdragon2, tdragon3h, tdragon2, nmk16_state, init_

 GAME( 1994, arcadian, 0, raphero, raphero, nmk16_state, init_banked_audiocpu, ROT270, "NMK", "Arcadia (NMK)", 0 ) // 1993 年 7 月 23 日测试模式，(c)1994 标题屏幕
 游戏( 1994, raphero, arcadian, raphero, raphero, nmk16_state, init_banked_audiocpu, ROT270, "NMK", "快速英雄 (NMK)", 0 ) // ^^
-GAME( 1994, rapheroa, arcadian, raphero, raphero, nmk16_state, init_banked_audiocpu, ROT270, "NMK (Media Trading License)", "Rapid Hero (Media Trading)", 0 ) // ^^ - 请注意所有 romset 都有媒体平铺图形中的交易（又名 Media Shoji），但这是唯一在标题屏幕上显示它的集合
+GAME( 1994, rapheroa, arcadian, raphero, raphero, nmk16_state, init_banked_audiocpu, ROT270, "NMK (Media Trading License)", "Rapid Hero (Media Trading)", 0 ) // ^^ - 请注意，所有 ROM 集都有平铺图形中的媒体交易（又名 Media Shoji），但这是唯一在标题屏幕上显示它的集合

 /* 这两款游戏在测试模式下都显示日期为 1992 年 3 月 9 日，它们看起来像是不同的修订版，所以我怀疑这是准确的 */
 游戏( 1992, sabotenb, 0, bjtwin, sabotenb, nmk16_state, init_nmk, ROT0, "NMK / Tecmo", "Saboten 轰炸机 (set 1)", MACHINE_NO_COCKTAIL )
@@ -8384,7 +8384,7 @@ 游戏( 1995, nouryokup, nouryoku, bjtwin, nouryoku, nmk16_state, 空

 // 这些使用西武音响系统（从雷电窃取的声音/音乐）而不是盗版者复制 nmk004
 GAME( 1990, Mustangb, Mustang, Mustangb, Mustang, nmk16_state,empty_init, ROT0, "盗版", "美国 AAF 野马 (盗版)", 0 )
-GAME( 1990, Mustangb2, Mustang, Mustangb, Mustang, nmk16_state,empty_init, ROT0, "bootleg (TAB Austria)", "US AAF Mustang (TAB Austria bootleg)", 0 ) // PCB 和 ROM 上有 TAB Austria 贴纸
+GAME( 1990, Mustangb2, Mustang, Mustangb, Mustang, nmk16_state,empty_init, ROT0, "bootleg (TAB Austria)", "US AAF Mustang (TAB Austria bootleg)", 0 ) // PCB 和 ROM 上有 TAB Austria 贴纸
 游戏（1991，tdragonb，tdragon，tdragonb，tdragonb，nmk16_state，init_tdragonb，ROT270，“盗版”，“雷龙（盗版）”，0）

 // 这些来自Comad，基于雷龙代码？
diff --git a/src/mame/drivers/pgm.cpp b/src/mame/drivers/pgm.cpp
```

```
索引 e2bf9190c86..60c623592d5 100644
---a/src/mame/drivers/pgm.cpp
+++ b/src/mame/drivers/pgm.cpp
@@ -160,9 +160,9 @@ 注意：

       只读存储器
       ----
- PGM_M01S.U18 - 16MBit 掩码ROM (TSOP48)
- PGM_P01S.U20 - 1MBit MASKROM（DIP40，带插座，相当于 27C1024 EPROM）
- PGM_T01S.U29 - 16MBit 掩码ROM (SOP44)
+ PGM_M01S.U18 - 16MBit 掩模 ROM (TSOP48)
+ PGM_P01S.U20 - 1MBit 掩码 ROM（DIP40，带插座，相当于 27C1024 EPROM）
+ PGM_T01S.U29 - 16MBit 掩模 ROM (SOP44)

       定制IC
       ------------
@@ -922,8 +922,8 @@ 注意：
       V-110X.U2 - AM27C4096 4MBit EPROM（DIP42，标记为“DRAGON II V-100C”）

       PAL - x3，标记为“CZ U3”、“CZ U4”、“CZ U6”
- *1 - MX23C4100 SOP40 MASKROM 的未填充位置
- *2 - MX23C4100 DIP40 EPROM/MASKROM 的未填充位置
+ *1 - MX23C4100 SOP40 掩模 ROM 的未填充位置
+ *2 - MX23C4100 DIP40 EPROM/掩模 ROM 的未填充位置


 IGS PCB NO-0135
@@ -941,7 +941,7 @@ IGS PCB NO-0135
   |--------------------------------||--------------------|

 笔记：
- 该 PCB 仅包含 SOP44 MASKROMS 和 2 个逻辑 IC
+ 该 PCB 仅包含 SOP44 掩模 ROM 和 2 个逻辑 IC
       仅填充 U5 和 U9

       实际板上存在选择屏幕上的故障。
@@ -1151,10 +1151,10 @@ IGS025 ASIC
 1 个 PAL
 2x 27C040 EPROM（主 68k 程序）
 1x 27C512 EPROM（保护代码？）
-1x 32MBit smt MASKROM (T0400)
+1x 32MBit smt 掩模 ROM (T0400)

 底板包含.....
-4x 32MBit smt MASKROM（A0400、A0401、B0400、M0400）
+4x 32MBit smt 掩模 ROM（A0400、A0401、B0400、M0400）
```

```
 */

@@ -1987,7 +1987,7 @@ 注：
       U5 - 27C4000 4MBit EPROM（DIP32，标记为“KB U5 V104”）
       U6 - 27C4000 4MBit EPROM（DIP32，标记为“KB U6 V104”）
       PAL - x3，标记为“DH U8”、“DH U1”、“DH U7”
- * - DIP42 EPROM/MASKROM 的未填充位置（标记为“P0300”）
+ * - DIP42 EPROM/掩模 ROM 的未填充位置（标记为“P0300”）


 IGS PCB NO-0178
@@ -2005,19 +2005,19 @@ IGS PCB NO-0178
   |---------------------------------||--------------------|

 笔记：
- U1 - 32MBit MASKROM（SOP44，标记为“M0300”）
- U2 - 32MBit MASKROM（SOP44，标记为“A0307”）
- U3 - 16MBit MASKROM（DIP42，标记为“A0302”）
- U4 - 16MBit MASKROM（DIP42，标记为“A0304”）
- U5 - 16MBit MASKROM（DIP42，标记为“A0305”）
- U8 - 16MBit MASKROM（DIP42，标记为“B0301”）
- U9 - 32MBit MASKROM（SOP44，标记为“A0300”）
- U10 - 32MBit MASKROM（SOP44，标记为“A0301”）
- U11 - 32MBit MASKROM（SOP44，标记为“A0303”）
- U12 - 32MBit MASKROM（SOP44，标记为“A0306”）
- U13 - 32MBit MASKROM（SOP44，标记为“B0300”）
- U14 - 32MBit MASKROM（SOP44，标记为“B0302”）
- U15 - 32MBit MASKROM（SOP44，标记为“B0303”）
+ U1 - 32MBit 掩模 ROM（SOP44，标记为“M0300”）
+ U2 - 32MBit 掩模 ROM（SOP44，标记为“A0307”）
+ U3 - 16MBit 掩模 ROM（DIP42，标记为“A0302”）
+ U4 - 16MBit 掩模 ROM（DIP42，标记为“A0304”）
+ U5 - 16MBit 掩模 ROM（DIP42，标记为“A0305”）
+ U8 - 16MBit 掩模 ROM（DIP42，标记为“B0301”）
+ U9 - 32MBit 掩模 ROM（SOP44，标记为“A0300”）
+ U10 - 32MBit 掩模 ROM（SOP44，标记为“A0301”）
+ U11 - 32MBit 掩模 ROM（SOP44，标记为“A0303”）
+ U12 - 32MBit 掩模 ROM（SOP44，标记为“A0306”）
+ U13 - 32MBit 掩模 ROM（SOP44，标记为“B0300”）
+ U14 - 32MBit 掩模 ROM（SOP44，标记为“B0302”）
+ U15 - 32MBit 掩模 ROM（SOP44，标记为“B0303”）

 */

@@ -2169,7 +2169,7 @@ 注意：
       U1_V100MG.U1 - MX27C4000 512K x8 EPROM（DIP32，标记为“PuzzleStar U1 V100MG”）
       U2_V100MG.U2 - MX27C4000 512K x8 EPROM（DIP32，标记为“PuzzleStar U2 V100MG”）
```

```
       PAL - Atmel ATF22V10B PAL（DIP24，标记为“EA U4”）
- U3 - 32MBit MASKROM (DIP42) 的未填充位置
+ U3 - 32MBit 掩码 ROM (DIP42) 的未填充位置
       U6、U7 - 74LS245 逻辑芯片的未安装位置 (x2)


@@ -2248,7 +2248,7 @@ 注：
       V101.U2/3/4/5- MX27C4000 4MBit EPROM (DIP32)
       PAL - x2，标记为“CW-2 U8”、“CW-2 U7”
       6264 - 8K x8 SRAM
- *1 - 标记为“P0500”的 SOP44 MASKROM 的未填充位置
+ *1 - 标记为“P0500”的 SOP44 掩模 ROM 的未填充位置


 IGS PCB NO-0135
@@ -5029,6 +5029,6 @@ 游戏( 2008, kovshxas, kovshp, pgm_arm_type1, kovsh, pgm_arm_ty
 //乱世拳皇/Luànshì quánhuáng
 GAME( 200?, kovlsqh, kovshp, pgm_arm_type1, kovsh, pgm_arm_type1_state, init_kovlsqh2, ROT0, "bootleg", "乱世拳皇 (盗版《勇者英雄超级英雄 Plus》，
版本 200CN)", MACHINE_IMPERFECT_SOUND | MACHINE_UNEMULATED_PROTECTION | MACH INE_NOT_WORKING | MACHINE_SUPPORTS_SAVE ) /* 需要 IGS027A 的内部 rom
*/
 GAME( 200?, kovlsqh2, kovshp, pgm_arm_type1, kovsh, pgm_arm_type1_state, init_kovlsqh2, ROT0, "bootleg", "乱世拳皇2 (盗版《三国志超级英雄Plus》，
版本200CN)", MACHINE_IMPERFECT_SOUND | MACHINE_UNEMULATED_PROTECTION | MACHINE_NOT_WORKING | MACHINE_SUPPORTS_SAVE ) /* 需要 IGS027A 的内部 rom */
-//乱世街霸/Luànshì jiē bà
+//乱世街霸/Luànshì jiē b
 GAME( 200?, kovlsjb, kovshp, pgm_arm_type1, kovsh, pgm_arm_type1_state, init_kovlsqh2, ROT0, "bootleg", "乱世解霸 (盗版《勇者超级英雄 Plus》，版本
200CN，第 1 集)", MACHINE_IMPERFECT_SOUND | MACHINE_UNEMULATED_PROTEC TION | MACHINE_NOT_WORKING | MACHINE_SUPPORTS_SAVE ) /* 需要 IGS027A 的内部
rom */
 GAME( 200?, kovlsjba, kovshp, pgm_arm_type1, kovsh, pgm_arm_type1_state, init_kovlsqh2, ROT0, "bootleg", "乱世解霸 (盗版《勇者超级英雄 Plus》，版
本 200CN，第 2 集)", MACHINE_IMPERFECT_SOUND | MACHINE_UNEMULATED_PROTEC TION | MACHINE_NOT_WORKING | MACHINE_SUPPORTS_SAVE ) /* 需要 IGS027A 的内
部 rom */
diff --git a/src/mame/drivers/powerins.cpp b/src/mame/drivers/powerins.cpp
索引 b2525527c34..8b0fc3d885a 100644
---a/src/mame/drivers/powerins.cpp
+++ b/src/mame/drivers/powerins.cpp
@@ -423,9 +423,9 @@ 注意：
       ROM -
             -1、-2：27C1001 EPROM
             -3、-4：27C4096 EPROM
- -5, -6 : 8M 42 针 MASKROM (578200)
- -7 : 4M 40 针掩码ROM (574200)
- -8 至 -19：8M 42 引脚 MASKROM (578200)
+ -5, -6 : 8M 42 引脚掩模 ROM (578200)
+ -7 : 4M 40 针掩模 ROM (574200)
+ -8 至 -19：8M 42 引脚掩码 ROM (578200)
             20：82S129可编程只读存储器
             21：82S135可编程只读存储器
             22：82S123可编程只读存储器
```

```
@@ -624,9 +624,9 @@ 力量本能
 阿特拉斯，1993

 这是盗版美国版本，其声音硬件与现有盗版集不同。
- PCB非常大，有2个插件子板和许多MASK ROM。
- MASK ROM 内容的添加可能等于大概的内容
-在原始 PCB 上发现了更大的 MASK ROM...
+PCB非常大，有2个插件子板和许多掩模ROM。
+掩膜 ROM 的内容相加可能等于大概的内容
+在原始 PCB 上发现了更大的掩膜 ROM...

 PCB布局

@@ -662,7 +662,7 @@ 注意：
       M6295时钟：4.000MHz（两者）；采样率 = 4000000/165（两者）
       垂直同步：60Hz

- ROM 1F 和 6N 为 1M MASK (MX27C1000)，所有其他 ROM 均为 4M MASK (MX27C4000)。
+ ROM 1F 和 6N 为 1M 掩码 (MX27C1000)，所有其他 ROM 均为 4M 掩码 (MX27C4000)。
       5* 处的 ROMS 位于插入式子板上。
       11*、12*、13G、13P 和 14* 处的 ROMS 位于插入式子板上。
       82S123 和 82S147 是 PROM。
diff --git a/src/mame/drivers/psikyo.cpp b/src/mame/drivers/psikyo.cpp
索引 9086dd9e776..8b3516a9770 100644
---a/src/mame/drivers/psikyo.cpp
+++ b/src/mame/drivers/psikyo.cpp
@@ -1610,11 +1610,11 @@ 注：
       HA1388 - 日立 HA1388 音响放大器
       2058D - JRC 2058 运算放大器
       U61、U34、\
- U20、U21、- 16M 掩模ROM (SOP44)
+ U20, U21,- 16M掩膜ROM (SOP44)
       U22、U23 /
- U1 - 4M 掩模ROM (SOP44)
+ U1 - 4M掩膜ROM (SOP44)
       VOL - 主音量电位器
- * - 16M SOP44 MASKROM 的未填充位置
+ * - 16M SOP44 mask ROM 的未填充位置


 CPU：MC68EC020FG16
diff --git a/src/mame/drivers/psikyo4.cpp b/src/mame/drivers/psikyo4.cpp
索引 5cc4dc7dd81..d98a18fbb7a 100644
---a/src/mame/drivers/psikyo4.cpp
+++ b/src/mame/drivers/psikyo4.cpp
@@ -121,9 +121,9 @@ ROM -
        1.U23-主程序，ST 27C4002 EPROM（DIP40）
```

```
         2.U22/

- 其余 ROM 为表面贴装 TSOP48 Type II MASKROM，
+ 其余 ROM 是表面贴装 TSOP48 Type II 掩膜 ROM，
         OKI MSM27C3252 (32MBit) 或 OKI MSM27C1652 (16MBit)。
- 这些 MASKROM 是非标准的，需要定制适配器
+ 这些掩模 ROM 是非标准的，需要定制适配器
         阅读它们。并非每场比赛的所有位置都已填充。看
         以下来源了解具体信息。

diff --git a/src/mame/drivers/raiden2.cpp b/src/mame/drivers/raiden2.cpp
索引 73362ded4ea..62105e47c63 100644
--- a/src/mame/drivers/raiden2.cpp
+++ b/src/mame/drivers/raiden2.cpp
@@ -19,7 +19,7 @@

 raiden 2 板测试说明 17/04/08（基于 dox 测试）

- ROM 银行位于 6c9，位 0x80
+ ROM 银行位于 6c9，位 0x80
    -- 游戏只在启动时直接写入，必须通过间接写入
       保护命令之一？或镜像？
    0x80 的值将 0x00000-0x1ffff 置于 0x20000 - 0x3ffff
@@ -159,7 +159,7 @@ 这似乎不存在于任何集合中。

 保护注意事项：
  这些游戏使用第二代（和第三代）西武的“COP”保护，
- 利用外部“COPX_D2”和“COPX_D3”查找ROM（可能用于
+ 利用外部“COPX_D2”和“COPX_D3”查找 ROM（可能用于
  数学运算）这些标记为 (c)1992 RISE Corp. 的芯片不被认为是
  是实际的 MCU，可能位于 Seibu 之一的内部
  海关。
@@ -1602,12 +1602,12 @@ 注：
       PAL1 - MMIPAL16L8B 印有“JJ4B01”（DIP20）
       PAL2 - AMI 18CV8 印有“JJ4B02”（DIP20）
       ROM - *PCM - 2M MaskROM 在位置 U1018 (DIP32) 处标记为“RAIDEN 2 PCM”，PCB 标记为 VOI2
- 6 - 23C020 MASK ROM 标记为“SEIBU 6”，位置 U1017 (DIP32)，PCB 标记为 VOI1
+ 6 - 23C020 掩膜 ROM 标记为“SEIBU 6”，位置 U1017 (DIP32)，PCB 标记为 VOI1
                    5 - 27C512 EPROM 标记为“SEIBU 5”，位置 U1110 (DIP28)
- *OBJ-1 - 16M MaskROM 标记为“RAIDEN 2 OBJ-1”，位置 U0811 (DIP42)
- *OBJ-2 - 16M MaskROM 标记为“RAIDEN 2 OBJ-2”，位置 U082 (DIP42)
- *OBJ-3 - 16M MaskROM 标记为“RAIDEN 2 OBJ-3”，位置 U0837 (DIP42)
- *OBJ-4 - 16M MaskROM 标记为“RAIDEN 2 OBJ-4”，位置 U0836 (DIP42)
+ *OBJ-1 - 16M 掩模 ROM 标记为“RAIDEN 2 OBJ-1”，位置 U0811 (DIP42)
+ *OBJ-2 - 16M 掩模 ROM 在位置 U082 (DIP42) 处标记为“RAIDEN 2 OBJ-2”
+ *OBJ-3 - 16M 掩模 ROM 标记为“RAIDEN 2 OBJ-3”，位置 U0837 (DIP42)
+ *OBJ-4 - 16M 掩模 ROM 标记为“RAIDEN 2 OBJ-4”，位置 U0836 (DIP42)
```

```
                    / 1x - 27C2001 EPROM 标记为“SEIBU 1”，位置 U1210 (DIP32)
      早期板| 2x - 27C2001 EPROM 标记为“SEIBU 2”，位置 U1211 (DIP32)
                    | 3x - 27C2001 EPROM 标记为“SEIBU 3”，位置 U129 (DIP32)
@@ -1632,9 +1632,9 @@ 注意：

 /* 注意：一些 raiden 2 fabtek usa 主板（至少是 Hammad 发送给 LN 和 Balrog 的主板）具有
     ROM_LOAD（“seibu5.u1110”，0x000000，0x08000，CRC（8f130589）SHA1（e58c8beaf9f27f063ffbcb0ab4600123c25ce6f3））
- 在Raiden2hk中使用的z80声音ROM而不是
+ z80 声音 ROM 在 raiden2hk 中使用，而不是
     ROM_LOAD（“snd.u1110”，0x000000，0x08000，CRC（f51a28f9）SHA1（7ae2e2ba0c8159a544a8fd2bb0c2c694ba849302））
- 来自Raiden2的ROM。版本略有差异，我不知道哪个是旧的/新的。- 闪电网络
+ 雷电2 的ROM。版本略有差异，我不知道哪个是旧的/新的。- 闪电网络

 ROMSET组织：
 注意：类型编号不一定按时间版本顺序排列。
@@ -1648,10 +1648,10 @@ raiden2e (设置 5 简单) 3 5’ 4(6bad0a3e) 2(488d050f) 2(c709)
 raiden2ea (set 6 easy) 4 6’ 5(f5f835af) 3(fab9f8e4) 3(c7aa4d00) 红色战斗机 在 hiscore
 raiden2eu (set 7 easy fabtek) 4 7’ 5(f5f835af) 3(fab9f8e4) 3(c7aa4d00) 红色战斗机 在 hiscore
 raiden2eua (set 8 easy fabtek) 3 8’ 6(6d362472) 3(fab9f8e4) 3(c7aa4d00) hiscore 上的红色战斗机，SN 0003068，aama 0557135
-^ 该套件有 4 个 prg rom：1 和 3 对应 seibu1/prg0，2 和 4 对应 seibu2/prg1
-balrog+ln (set x fabtek) 1 1’ 2(8f130589) 1(fb0fca23) 1(c9ec9469) hiscore 上的棕褐色战斗机，sn 0012739，aama 0600565，由于 rom 匹配第 1 组和第 2 组的混合，尚未在 mame 中
+^ 该组有 4 个程序 ROM：1 和 3 对应 seibu1/prg0，2 和 4 对应 seibu2/prg1
+balrog+ln (set x fabtek) 1 1’ 2(8f130589) 1(fb0fca23) 1(c9ec9469) hiscore 上的棕褐色战斗机，sn 0012739，aama 0600565，尚未在 mame 中，因为 ROM 匹配集合 1 和 2 的混合

-SND/u1110 ROM 之间的差异：
+SND/u1110 ROM 之间的差异：
    前半部分结束，ff 填充之前的最后一个字节以 7fff 结束
    | ff 填充之前的最后一个字节以 8fff 结尾
    | | ff 填充之前的最后一个字节以 ffff 结尾
@@ -1676,7 +1676,7 @@ ROM_START( 雷电2 )
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD（“snd.u1110”，0x000000，0x08000，CRC（f51a28f9）SHA1（7ae2e2ba0c8159a544a8fd2bb0c2c694ba849302））
@@ -1687,20 +1687,20 @@ ROM_START( 雷电2 )
        ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
```

```
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
@@ -1715,7 +1715,7 @@ ROM_START(raiden2g)
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD（“snd.u1110”，0x000000，0x08000，CRC（f51a28f9）SHA1（7ae2e2ba0c8159a544a8fd2bb0c2c694ba849302））
@@ -1726,20 +1726,20 @@ ROM_START(raiden2g)
        ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB 丝印 FX0 - sldh
w/raiden2u */

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
```

```
        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
@@ -1772,7 +1772,7 @@ ROM_START(raiden2hk)
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD( "seibu5.u1110", 0x000000, 0x08000, CRC(8f130589) SHA1(e58c8beaf9f27f063ffbcb0ab4600123c25ce6f3) ) // sldh w/raiden2u
@@ -1783,20 +1783,20 @@ ROM_START(raiden2hk)
        ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
```

```
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
@@ -1846,7 +1846,7 @@ ROM_START(raiden2j)
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD（“seibu5.u1110”，0x000000，0x08000，CRC（8f130589）SHA1（e58c8beaf9f27f063ffbcb0ab4600123c25ce6f3））
@@ -1857,20 +1857,20 @@ ROM_START(raiden2j)
        ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
```

```
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB 丝印 VOICE1 - sldh
w/raiden2u */

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
@@ -1885,7 +1885,7 @@ ROM_START( raiden2sw ) // 带序列号的原始板 # 0008307
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD( "seibu_5.u1110", 0x000000, 0x08000, CRC(c2028ba2) SHA1(f6a9322b669ff82dea6ecf52ad3bd5d0901cce1b) ) // 99.993896% 匹配
@@ -1896,20 +1896,20 @@ ROM_START( raiden2sw ) // 带序列号的原始板 # 0008307
        ROM_LOAD( "seibu_7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
```

```
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "seibu_6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -1924,7 +1924,7 @@ ROM_START( raiden2f ) // 序列号为 12476 且与 raiden2 匹配的原始板
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD( "seibu5_u1110.bin", 0x000000, 0x08000, CRC(f51a28f9) SHA1(7ae2e2ba0c8159a544a8fd2bb0c2c694ba849302) ) // == 雷电2
 @@ -1935,20 +1935,20 @@ ROM_START( raiden2f ) // 序列号为 12476 且与 raiden2 匹配的原始板
         ROM_LOAD( "7_u0724.bin", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
```

```
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "6_u1017.bin", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -1963,7 +1963,7 @@ ROM_START(raiden2nl)
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD（“5_u1110.bin”，0x000000，0x08000，CRC（8f130589）SHA1（e58c8beaf9f27f063ffbcb0ab4600123c25ce6f3））
 @@ -1974,20 +1974,20 @@ ROM_START(raiden2nl)
         ROM_LOAD( "7_u0724.bin", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
```

```
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "6_u1017.bin", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -2002,7 +2002,7 @@ ROM_START(raiden2u)
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", 0 ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD( "seibu5.u1110", 0x000000, 0x08000, CRC(6d362472) SHA1(a362e500bb9492affde1f7a4da7e08dd16e755df) ) // sldh w/raiden2hk
 @@ -2013,20 +2013,20 @@ ROM_START(raiden2u)
         ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c7aa4d00) SHA1(9ad99d3891598c1ea3f12318400ee67666da56dd) ) // sldh w/raiden2g

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
```

```
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fab9f8e4) SHA1(b1eff154c4f766b2d272ac6a57f8d54c9e39e3bb) ) // sldh w/raiden2j

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -2041,7 +2041,7 @@ ROM_START(raiden2i)
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD( "seibu5.c.u1110", 0x000000, 0x08000, CRC(5db9f922) SHA1(8257aab98657fe44df19d2a48d85fcf65b3d98c6) )
 @@ -2052,20 +2052,20 @@ ROM_START(raiden2i)
         ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
```

```
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -2081,7 +2081,7 @@ ROM_START(raiden2k)
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD（“snd.u1110”，0x000000，0x08000，CRC（f51a28f9）SHA1（7ae2e2ba0c8159a544a8fd2bb0c2c694ba849302））
 @@ -2092,20 +2092,20 @@ ROM_START(raiden2k)
         ROM_LOAD( "seibu7.u0724", 0x000000, 0x020000, CRC(c9ec9469) SHA1(a29f480a1bee073be7a177096ef58e1887a5af24) ) /* PCB丝印FX0 */

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
```

```
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "seibu6.u1017", 0x00000, 0x40000, CRC(fb0fca23) SHA1(4b2217b121a66c5ab6015537609cf908ffedaf86) ) /* PCB丝印VOICE1 */

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -2137,7 +2137,7 @@ ROM_START(raiden2e)
         ROM_RELOAD(0x100001, 0x80000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD（“r2_snd.u1110”，0x000000，0x08000，CRC（6bad0a3e）SHA1（eb7ae42353e1984cd60b569c26cdbc3b025a7da6））
 @@ -2148,20 +2148,20 @@ ROM_START(raiden2e)
         ROM_LOAD（“r2_fx0.u0724”，0x000000，0x020000，CRC（c709bdf6）SHA1（0468d90412b7590d67eaadc0a5e3537cd5e73943））

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
```

```
 */

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD ( "r2_voi1.u1017" , 0x00000, 0x40000, CRC (488d050f) SHA1 (fde2fd64fea6bc39e1a42885d21d362bc6be2ac2) )

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD ( "jj4b02__ami18cv8-15.u0342.jed" , 0x0000, 0x288, NO_DUMP)
@@ -2176,7 +2176,7 @@ ROM_START(raiden2ea)
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", 0 ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD ( "r2.5.u1110" , 0x000000, 0x08000, CRC (f5f835af) SHA1 (5be82ebc582d0da919e9ae1b9e64528bb295efc7) )
@@ -2187,20 +2187,20 @@ ROM_START(raiden2ea)
        ROM_LOAD ( "r2.7.u0724" , 0x000000, 0x020000, CRC (c7aa4d00) SHA1 (9ad99d3891598c1ea3f12318400ee67666da56dd) )

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
```

```
        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD ( "r2.6.u1017" , 0x00000, 0x40000, CRC (fab9f8e4) SHA1 (b1eff154c4f766b2d272ac6a57f8d54c9e39e3bb) )

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD ( "jj4b02__ami18cv8-15.u0342.jed" , 0x0000, 0x288, NO_DUMP)
@@ -2215,7 +2215,7 @@ ROM_START( raiden2eu ) // 与raiden2ea相同，不同地区
        ROM_RELOAD(0x100001, 0x80000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", 0 ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD ( "r2.5.u1110" , 0x000000, 0x08000, CRC (f5f835af) SHA1 (5be82ebc582d0da919e9ae1b9e64528bb295efc7) )
@@ -2226,27 +2226,27 @@ ROM_START( raiden2eu ) // 与raiden2ea相同，不同区域
        ROM_LOAD ( "r2.7.u0724" , 0x000000, 0x020000, CRC (c7aa4d00) SHA1 (9ad99d3891598c1ea3f12318400ee67666da56dd) )

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD ( "r2.6.u1017" , 0x00000, 0x40000, CRC (fab9f8e4) SHA1 (b1eff154c4f766b2d272ac6a57f8d54c9e39e3bb) )
```

```
        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
        ROM_LOAD（“jj4b01__mmipal16l8bcn.u0341.jed”，0x0000，0x335，CRC（d1a039af）SHA1（f88ff8674d5be17ae9085b51aefcf6abf0574883））
 ROM_END

-ROM_START( raiden2eua ) // 有点像 raiden2e 简易设置与 raiden2ea 和 2f 的语音 rom 以及独特的声音 rom 的混合
+ROM_START( raiden2eua ) // 有点像 raiden2e 简易设置与 raiden2ea 和 2f 的语音 ROM 以及独特的声音 ROM 的混合
        ROM_REGION( 0x200000, "maincpu", 0 ) /* v30 主CPU */
        ROM_LOAD32_BYTE("seibu__1.27c020j.u1210", 0x000000, 0x40000, CRC(ed1514e3) SHA1(296125bfe3c4f3033f7aa319dd8554bc978c4a00) )
        ROM_RELOAD(0x100000, 0x40000)
@@ -2258,7 +2258,7 @@ ROM_START( raiden2eua ) // 有点像 raiden2e easy set 与语音 rom 的混合
        ROM_RELOAD(0x100003, 0x40000)

        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
- ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x20000, "audiocpu", 0 ) /* 声音 Z80 的 64k 代码 */
        ROM_LOAD( "seibu__5.27c512.u1110", 0x000000, 0x08000, CRC(6d362472) SHA1(a362e500bb9492affde1f7a4da7e08dd16e755df) )
@@ -2269,20 +2269,20 @@ ROM_START( raiden2eua ) // 有点像 raiden2e easy set 与语音 rom 的混合
        ROM_LOAD( "seibu__7.fx0.27c210.u0724", 0x000000, 0x020000, CRC(c7aa4d00) SHA1(9ad99d3891598c1ea3f12318400ee67666da56dd) )

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
- ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
- ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

        ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
*/
- ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
*/
+ ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
```

```
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */

         ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
         ROM_LOAD( "seibu__6.voice1.23c020.u1017", 0x00000, 0x40000, CRC(fab9f8e4) SHA1(b1eff154c4f766b2d272ac6a57f8d54c9e39e3bb) )

         ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
 - ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
         ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
 @@ -2301,7 +2301,7 @@ ROM_START( raiden2eg ) // 这是与 raiden2eua 相同的代码修订版，但是是一个 ger
         ROM_RELOAD(0x100003, 0x40000)

         ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
 - ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 焊接掩模 ROM */

         ROM_REGION( 0x20000, "audiocpu", 0 ) /* 声音 Z80 的 64k 代码 */
         ROM_LOAD（“raiden_2_5.bin”，0x000000，0x08000，CRC（6d362472）SHA1（a362e500bb9492affde1f7a4da7e08dd16e755df））
 @@ -2312,20 +2312,20 @@ ROM_START( raiden2eg ) // 这是与 raiden2eua 相同的代码修订版，但是是一个 ger
         ROM_LOAD（“raiden_2_7.bin”，0x000000，0x020000，CRC（c7aa4d00）SHA1（9ad99d3891598c1ea3f12318400ee67666da56dd））

         ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
 - ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 - ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-1.u0714", 0x000000, 0x200000, CRC(e61ad38e) SHA1(63b06cd38db946ad3fc5c1482dc863ef80b58fec) ) /* 焊接掩模 ROM */
 + ROM_LOAD( "raiden_2_seibu_bg-2.u075", 0x200000, 0x200000, CRC(a694a4bb) SHA1(39c2614d0effc899fe58f735604283097769df77) ) /* 焊接掩模 ROM */

         ROM_REGION32_LE( 0x800000, "gfx3", 0 ) /* 精灵 gfx (加密) */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 - ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-1.u0811", 0x000000, 0x200000, CRC(ff08ef0b) SHA1(a1858430e8171ca8bab785457ef60e151b5e5cf1) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-2.u082", 0x000002, 0x200000, CRC(638eb771) SHA1(9774cc070e71668d7d1d20795502dccd21ca557b) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-3.u0837", 0x400000, 0x200000, CRC(897a0322) SHA1(abb2737a2446da5b364fc2d96524b43d808f4126) ) /* 焊接掩模 ROM
 */
 + ROM_LOAD32_WORD( "raiden_2_seibu_obj-4.u0836", 0x400002, 0x200000, CRC(b676e188) SHA1(19cc838f1ccf9c4203cd0e5365e5d99ff3a4ff0f) ) /* 焊接掩模 ROM
```

```
 */

        ROM_REGION( 0x100000, "oki1", 0 ) /* ADPCM 样本 */
        ROM_LOAD( "raiden_2_6.bin", 0x00000, 0x40000, CRC(fab9f8e4) SHA1(b1eff154c4f766b2d272ac6a57f8d54c9e39e3bb) )

        ROM_REGION( 0x100000, "oki2", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */
+ ROM_LOAD( "raiden_2_pcm.u1018", 0x00000, 0x40000, CRC(8cf0d17e) SHA1(0fbe0b1e1ca5360c7c8329331408e3d799b4714c) ) /* 焊接掩模 ROM */

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD（“jj4b02__ami18cv8-15.u0342.jed”，0x0000，0x288，NO_DUMP）
@@ -2727,7 +2727,7 @@ ROM_START( raidendxch )
        ROM_LOAD32_BYTE（“rdxc_3.u129”，0x000002，0x80000，CRC（50f0a6aa）SHA1（68579f8e73fe06b458368ac9cac0b33370cf3b4e））
        ROM_LOAD32_BYTE（“rdxc_4.u1212”，0x000003，0x80000，CRC（00071e70）SHA1（8a03ea0e650936e48cdd21ff84132742649920fe））

- // 此套件中没有其他 ROM，因此下面的可能是错误的
+ // 该组中没有其他 ROM，因此下面的可能是错误的
        ROM_REGION( 0x40000, "user2", 0 ) /* COPX */
        ROM_LOAD( "copx-d2.u0313", 0x00000, 0x40000, CRC(a6732ff9) SHA1(c4856ec77869d9098da24b1bb3d7d58bb74b4cda) ) /* 与原始 Raiden 2 共享 */

@@ -2762,15 +2762,15 @@ ROM_END
 /* 零队的硬件稍旧（93 年初而不是 93 后期），但是
 与 Raiden 2 几乎相同，但有一些关键区别：
 零队：雷电2：
-BG/FG ROM 标记为 MUSHA BG/FG ROM 标记为 RAIDEN 2
+标记为 MUSHA 的 BG/FG ROM 标记为 RAIDEN 2 的 BG/FG ROM
 SEI251 fg/sprite 门阵列 SEI252 fg/sprite 门阵列
 约15个74xx逻辑芯片SEI360门阵列
 3x 拨码开关阵列 2x 拨码开关阵列
-4x 8位程序ROM 2x 16位程序ROM（一些较旧的PCB有4x 8位，如zt）
+4x 8 位程序 ROM 2x 16 位程序 ROM（一些较旧的 PCB 有 4x 8 位，如 zt）
 YM3812加Y3014 YM2151加Y3012加NJM4550（一些较旧的PCB有YM2151、Y3014）
-1x OKI M6295 和语音 ROM 2x OKI M6295s 和 2x 语音 ROM
-2x 8 位被许可人背景 1x 16 位被许可人背景
-2x fg/sprite mask roms 4x fg/sprite mask roms
+1x OKI M6295 和语音 ROM 2x OKI M6295 和 2x 语音 ROM
+2x 8 位授权 bg ROM 1x 16 位授权 bg ROM
+2x fg/sprite mask ROM 4x fg/sprite mask ROM
 4x 朋友（两个堆叠） 2x 朋友
 */
 /* ZERO TEAM 西武开发 1993
@@ -2923,13 +2923,13 @@ ROM_START( Zeroteama ) // 无授权，原装日本？
        ROM_LOAD( "v3c004x.ami18cv8pc-25.u0310.jed", 0x0000, 0x288, NO_DUMP) // 位于 v3c001 上并连接到一些返工线
 ROM_END

-/* 该套件由更新的程序 rom 组成，是零团队的更高版本或 hack，以纳入编写内容
+/* 该套件由更新的程序 ROM 组成，是零团队的更高版本或 hack，以纳入编写内容
```

```
 fg sei251 的“关键数据”在启动时传输到 PCB（如 Raiden 2 那样），而不是依靠 sram 来保存
- 工厂编程的按键（或通过下面的自杀复活套件）；因此这个romset不受常见的影响
+工厂编程的按键（或通过下面的自杀复活套件）；因此这个 ROMset 不受常见的影响
 3.6v锂电池电量耗尽和缺少按键导致精灵显示为乱码的问题*/
 // 注意：mame 中的这个特定集合有可能*但尚未证明*是日本和美国的弗兰肯斯坦混合体
-// 集，使用我们集的声音和字符 ROM 以及后来的日本集的代码。如果它被丢弃，这将是有意义的
-//来自一个“已修复、无自杀”的修改过的美国主板，其中有人交换了后来的无自杀日本代码ROM。
+// 集，使用美国集的声音和字符 ROM 以及后来日本集的代码。如果它被丢弃，这将是有意义的
+// 来自“已修复、无自杀”的修改过的美国主板，其中有人交换了后来的无自杀日本代码 ROM。
 ROM_START( Zeroteamb ) // 没有被许可人，后来的日本？
        ROM_REGION( 0x200000, "maincpu", 0 ) /* v30 主CPU */
        ROM_LOAD32_BYTE("1b.u024.5k", 0x000000, 0x40000, CRC(157743d0) SHA1(f9c84c9025319f76807ef0e79f1ee1599f915b45) )
@@ -3005,7 +3005,7 @@ ROM_START( Zeroteamc ) // 梁华，台湾授权商，标志下无特殊字样
 ROM_END

 ROM_START( Zeroteamd ) // Dream Soft，韩国授权商，标题徽标下无特殊字样；主板上有序列号“no 1041”。
- // 这很奇怪，在其他 zt 设置上 rom 顺序是 1 3 2 4，但这个是 1 3 4 2。责怪 seibu 或标记 rom 的人，这些 rom 是用钢笔标记的
+ // 这很奇怪，在其他 zt 设置上，ROM 顺序是 1 3 2 4，但这个是 1 3 4 2。责怪 seibu 或标记 ROM 的人，这些 ROM 是用钢笔标记的
        ROM_REGION( 0x200000, "maincpu", 0 ) /* v30 主CPU */
        ROM_LOAD32_BYTE("1.d.u024.5k", 0x000000, 0x40000, CRC(6cc279be) SHA1(63143ba3105d24d133e60ffdb3edc2ceb2d5dc5b) )
        ROM_LOAD32_BYTE("3.d.u023.6k", 0x000002, 0x40000, CRC(0212400d) SHA1(28f77b5fddb9d724b735c3ff2255bd518b166e67) )
@@ -3016,13 +3016,13 @@ ROM_START( Zeroteamd ) // Dream Soft，韩国授权商，徽标下无特殊字样
        ROM_LOAD（“copx-d2.u0313.6n”，0x00000，0x40000，CRC（a6732ff9）SHA1（c4856ec77869d9098da24b1bb3d7d58bb74b4cda））

        ROM_REGION( 0x20000, "audiocpu", ROMREGION_ERASEFF ) /* 声音 Z80 的 64k 代码 */
- ROM_LOAD( "512kb.u1110.5b", 0x000000, 0x08000, CRC(efc484ca) SHA1(c34b8e3e7f4c2967bc6414348993478ed637d338) ) // 这是此 PCB 版本上的焊接掩模
ROM! 内容与台版EPROM相符；面膜ROM没有标签
+ ROM_LOAD( "512kb.u1110.5b", 0x000000, 0x08000, CRC(efc484ca) SHA1(c34b8e3e7f4c2967bc6414348993478ed637d338) ) // 这是此 PCB 版本上的焊接掩模
ROM! 内容与台湾版EPROM相符；掩膜ROM没有标签
        ROM_CONTINUE(0x10000,0x8000)
        ROM_COPY（“音频CPU”，0x000000，0x018000，0x08000）

        ROM_REGION( 0x020000, "gfx1", 0 ) /* 字符 */
- ROM_LOAD16_BYTE( "512kb.u072.5s", 0x000000, 0x010000, CRC(30ec0241) SHA1(a0d0be9458bf97cb9764fb85c988bb816710475e) ) // 这是此 PCB 版本上的焊接掩
模 ROM! 内容与台版EPROM相符；面膜ROM没有标签
- ROM_LOAD16_BYTE( "512kb.u077.5r", 0x000001, 0x010000, CRC(e18b3a75) SHA1(3d52bba8d47d0d9108ee79014fd64d6e856a3fde) ) // 这是此 PCB 版本上的焊接掩
模 ROM! 内容与台版EPROM相符；面膜ROM没有标签
+ ROM_LOAD16_BYTE( "512kb.u072.5s", 0x000000, 0x010000, CRC(30ec0241) SHA1(a0d0be9458bf97cb9764fb85c988bb816710475e) ) // 这是此 PCB 版本上的焊接掩
模 ROM! 内容与台湾版EPROM相符；掩膜ROM没有标签
+ ROM_LOAD16_BYTE( "512kb.u077.5r", 0x000001, 0x010000, CRC(e18b3a75) SHA1(3d52bba8d47d0d9108ee79014fd64d6e856a3fde) ) // 这是此 PCB 版本上的焊接掩
模 ROM! 内容与台湾版EPROM相符；掩膜ROM没有标签

        ROM_REGION( 0x400000, "gfx2", 0 ) /* 背景 gfx */
        ROM_LOAD（“musha_back-1.u075.4s”，0x000000，0x100000，CRC（8b7f9219）SHA1（3412b6f8a4fe245e521ddcf185a53f2f4520eb57））
@@ -3033,7 +3033,7 @@ ROM_START( Zeroteamd ) // Dream Soft，韩国授权商，徽标下无特殊字样
        ROM_LOAD32_WORD（“musha_obj-2.u082.5f”，0x000002，0x200000，CRC（cb61c19d）SHA1（151a2ce9c32f3321a974819e9b165dddc31c8153））
```

```
        ROM_REGION( 0x100000, "oki", 0 ) /* ADPCM 样本 */
- ROM_LOAD( "8.u105.4a", 0x00000, 0x40000, CRC(b4a6e899) SHA1(175ab656db3c 3258ff10eede89890f62435d2298) ) // 与上面 Zeroteamc 中标记为 1 的 '6' 相同，但在 pen 的标签上写有 '8'
+ ROM_LOAD( "8.u105.4a", 0x00000, 0x40000, CRC(b4a6e899) SHA1(175ab656db3c3258ff10eede89890f62435d2298) ) // 与上面的 Zeroteamc 中标记为 1 的“6”相同，但在 Pen 的标签上写有“8”

        ROM_REGION( 0x10000, "朋友", 0 ) /* 朋友 */
        ROM_LOAD( "v3c001.pal.u0310.jed", 0x0000, 0x288, NO_DUMP) // 位于 v3c004x 下，未知 pal 类型
@@ -3086,10 +3086,10 @@ ROM_END
 顾名思义，这是用来让“自杀”的 ZT PCB 再次复活，其中 3.6v
 备份FG/精灵加密密钥的锂电池没电了，精灵显示
 作为垃圾块。
-使用方法：将3.6V电池更换为正常的电池，然后取出正常的四码ROM
+使用方法：将3.6V电池更换为正常的电池，然后取出正常的四个代码ROM
 并安装这些。
 启动 PCB，它应该重写 sei251 解密密钥并在屏幕上显示一条消息。
-接下来，关闭电源并重新插入旧的代码ROM，PCB现在应该有工作精灵。
+接下来，关闭电源并重新插入旧的代码 ROM，PCB 现在应该具有工作精灵。
 */

 ROM_START（零团队）
@@ -3315,9 +3315,9 @@ void raiden2_state::init_zeroteam()

 /* 游戏驱动 */

-/* Raiden 2 / DX 集按非区域 ROM 的校验和排序（最终程序 ROM 包含区域字节）
+/* Raiden 2 / DX 集按非区域 ROM 的校验和排序（最终程序 ROM 包含区域字节）

- 有趣的是，大多数 Raiden DX 集都是独一无二的，很少有仅在区域字节上有所不同，但对于 Raiden 2，有很多程序 ROM 仅在区域字节上有所不同
+ 有趣的是，大多数 Raiden DX 集都是独一无二的，很少有仅在区域字节上有所不同，但对于 Raiden 2，有很多程序 ROM 仅在区域字节上有所不同
     目前出售的两套《雷电 2》“较难”套装均来自韩国地区，但韩国地区字节并未确定难度。
 */

@@ -3325,60 +3325,60 @@ void raiden2_state::init_zeroteam()

 // 常规版本 - 棕褐色高分表背景，第一桥上的常规坦克

-// 第一个 ROM 的代码 rev 具有校验和 09475ec4
+// 第一个 ROM 的代码 rev 具有校验和 09475ec4
 游戏（1993，raiden2，0，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（Fabtek许可证）”，“Raiden II（美国，设置1）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2g，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（调整许可证）”，“Raiden II（德国）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2hk，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（Metrotainment许可证）”，“Raiden II（香港）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2j，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“西武开发”，“雷电II（日本）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2sw，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“西武开发”，“雷电II（瑞士）”，MACHINE_SUPPORTS_SAVE）
```

```
-// 第一个 ROM 的代码 rev 具有校验和 b16df955
+// 代码版本，第一个 ROM 具有校验和 b16df955
 游戏（1993，raiden2u，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（Fabtek许可证）”，“Raiden II（美国，套装2）”，MACHINE_SUPPORTS_SAVE）
-// 第一个 ROM 的代码 rev 具有校验和 53be3dd0
+// 第一个 ROM 的代码版本具有校验和 53be3dd0
 游戏（1993，raiden2f，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“西武开发”，“雷电II（法国）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2nl，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“西武开发”，“Raiden II（荷兰）”，MACHINE_SUPPORTS_SAVE）
-// 第一个 ROM 的代码版本具有校验和 c1fc70f5
+// 第一个 ROM 的代码版本具有校验和 c1fc70f5
 游戏（1993，raiden2i，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“西武开发”，“雷电II（意大利）”，MACHINE_SUPPORTS_SAVE）

 // 简易版 - 彩色高分表背景，不同的敌人放置

-// 第一个 ROM 的代码 rev 具有校验和 2abc848c
+// 第一个 ROM 的代码版本具有校验和 2abc848c
 GAME( 1993, raiden2e, raiden2, raiden2, raiden2, raiden2_state, init_raiden2, ROT270, "Seibu Kaihatsu", "Raiden II (更简单, 韩国)", MACHINE_SUPPORTS_SAVE ) // (区域 0x04) - 韩国，如果区域与RDX，没有许可证或区域消息
-// 第一个 ROM 具有校验和 ed1514e3 的代码版本（使用 4x 程序 ROM 配置，而不是 2）将在 2 ROM 配置中具有 crc 2abc848c，因此与上面相同的版本
+// 第一个 ROM 具有校验和 ed1514e3 的代码版本（使用 4x 程序 ROM 配置，而不是 2）将在 2 ROM 配置中具有 crc 2abc848c，因此与上面相同的版本
 游戏（1993，raiden2eua，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（Fabtek许可证）”，“Raiden II（更简单，美国套装1）”，MACHINE_SUPPORTS_SAVE）
 游戏（1993，raiden2eg，raiden2，raiden2，raiden2，raiden2_state，init_raiden2，ROT270，“Seibu Kaihatsu（调整许可证）”，“Raiden II（更容易，德国）”，MACHINE_SUPPORTS_SAVE）
-// 第一个 ROM 的代码 rev 具有校验和 d7041be4
+// 第一个 ROM 的代码 rev 具有校验和 d7041be4
 GAME( 1993, raiden2ea, raiden2, raiden2, raiden2, raiden2_state, init_raiden2, ROT270, "Seibu Kaihatsu", "Raiden II (easyer, Japan)", MACHINE_SUPPORTS_SAVE ) // (区域 0x00) - 日本，但简单集没有即使地区为 00，也会显示“FOR USE IN JAPAN ONLY”
 GAME( 1993, raiden2eu, raiden2, raiden2, raiden2, raiden2_state, init_raiden2, ROT270, "Seibu Kaihatsu (Fabtek 许可证)", "Raiden II (更简单, US set 2)", MACHINE_SUPPORTS_SAVE ) // ^

 // 更难的版本 - 棕褐色高分表背景，第一座桥上有红色坦克

-// 第一个 ROM 的代码 rev 具有校验和 1fcc08cf
+// 第一个 ROM 的代码版本具有校验和 1fcc08cf
 GAME( 1993, raiden2k, raiden2, raiden2, raiden2, raiden2_state, init_raiden2, ROT270, "Seibu Kaihatsu", "Raiden II (harder, Korean)", MACHINE_SUPPORTS_SAVE ) // (区域 0x04) - 韩国，没有显示消息
-// 第一个 ROM 的代码版本具有校验和 413241e0（在 Raiden DX 硬件上使用 4x 程序 ROM 配置，而不是 2）
+// 第一个 ROM 的代码版本具有校验和 413241e0（在 Raiden DX 硬件上使用 4x 程序 ROM 配置，而不是 2）
 GAME( 1993, raiden2dx, raiden2, raidendx, raiden2, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu", "Raiden II (更难，Raiden DX 硬件，韩国)", MACHINE_SUPPORTS_SAVE ) // ^


 // 雷电 DX 套装

-// 代码 rev，前 3 个 ROM 的校验和为 14d725fc、5e7e45cb、f0a47e67
```

```
+// 前 3 个 ROM 的代码版本具有校验和 14d725fc、5e7e45cb、f0a47e67
 游戏（1994，raidendx，0，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“西武开发”，“雷电DX（英国）”，MACHINE_SUPPORTS_SAVE）
 游戏（1994，raidendxg，raidendx，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“Seibu Kaihatsu（调整许可证）”，“Raiden DX（德国）”，MACHINE_SUPPORTS_SAVE）
 游戏（1994，raidendxpt，raidendx，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“西武开发”，“Raiden DX（葡萄牙）”，MACHINE_SUPPORTS_SAVE）
-// 代码版本，前 3 个 ROM 的校验和为 7624c36b、4940fdf3、6c495bcf
+// 前 3 个 ROM 的代码版本具有校验和 7624c36b、4940fdf3、6c495bcf
 GAME( 1994, raidendxa1, raidendx, raidendx, raidendx, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu (Metrotainment 许可证)", "Raiden DX (香港, set 1)", MACHINE_SUPPORTS_SAVE )
-// 代码 rev，前 3 个 ROM 的校验和为 22b155ae、2be98ca8、b4785576
+// 代码版本，前 3 个 ROM 的校验和为 22b155ae、2be98ca8、b4785576
 GAME( 1994, raidendxa2, raidendx, raidendx, raidendx, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu (Metrotainment 许可证)", "Raiden DX (香港, set 2)", MACHINE_SUPPORTS_SAVE )
-// 前 3 个 ROM 的代码版本具有校验和 b5b32885、7efd581d、55ec0e1d
+// 前 3 个 ROM 的代码版本具有校验和 b5b32885、7efd581d、55ec0e1d
 游戏（1994，raidendxk，raidendx，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“西武开发”，“Raiden DX（韩国）”，MACHINE_SUPPORTS_SAVE）
-// 前 3 个 ROM 的代码版本具有校验和 53e63194、ec8d1647、7dbfd73d
+// 前 3 个 ROM 的代码版本具有校验和 53e63194、ec8d1647、7dbfd73d
 游戏（1994，raidendxu，raidendx，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“Seibu Kaihatsu（Fabtek许可证）”，“Raiden DX（美国）”，MACHINE_SUPPORTS_SAVE）
-// 代码版本，前 3 个 ROM 具有校验和 c589019a、b2222254、60f04634
+// 前 3 个 ROM 的代码版本具有校验和 c589019a、b2222254、60f04634
 游戏（1994，raidendxnl，raidendx，raidendx，raidendx，raiden2_state，init_raidendx，ROT270，“西武开发”，“雷电DX（荷兰）”，MACHINE_SUPPORTS_SAVE）
-// 代码版本，前 3 个 ROM 的校验和为 5af382e1、899966fc、e7f08013
+// 前 3 个 ROM 的代码版本具有校验和 5af382e1、899966fc、e7f08013
 游戏( 1994, raidendxj, raidendx, raidendx, raidendx, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu", "Raiden DX (日本, set 1)", MACHINE_SUPPORTS_SAVE )
-// 前 3 个 ROM 的代码版本具有校验和 247e21c7、f2e9855a、fbab727f
+// 前 3 个 ROM 的代码版本具有校验和 247e21c7、f2e9855a、fbab727f
 游戏( 1994, raidendxja, raidendx, raidendx, raidendx, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu", "Raiden DX (日本, set 2)", MACHINE_SUPPORTS_SAVE )
-// 代码 rev，前 3 个 ROM 的校验和为 2154c6ae、73bb74b7、50f0a6aa
+// 前 3 个 ROM 的代码版本具有校验和 2154c6ae、73bb74b7、50f0a6aa
 GAME( 1994, raidendxch, raidendx, raidendx, raidendx, raiden2_state, init_raidendx, ROT270, "Seibu Kaihatsu (Ideal International Development Corp 许可证)", "Raiden DX (中国)", MACHINE_SUPPORTS_SAVE ) // 区域字节为 0x16，定义为 "中国大陆”仅此套装


diff --git a/src/mame/drivers/segahang.cpp b/src/mame/drivers/segahang.cpp
索引 30b4d3e14a5..ee7834d9350 100644
---a/src/mame/drivers/segahang.cpp
+++ b/src/mame/drivers/segahang.cpp
@@ -1143,11 +1143,11 @@ ROM_END
 // 组件 ROM BD 834-5669
 // 声音组件 BD 834-5670-01（贴纸 834-5706-01 挂在上面）
```

```
 //
-// ROM 和声音板丢失，但手册编号 420-5244 第二次印刷确认它们是相同的，具有相同的数据/ROM。
+// ROM 和声音板丢失，但手册第 420-5244 号第二次印刷确认它们是相同的，具有相同的数据/ROM。
 //
-// 主 CPU ROM 上没有标签，只有黑色贴纸，可能是 1995 年维修工作的结果
+// 主 CPU ROM 上没有标签，只有黑色贴纸，可能是 1995 年维修工作的结果
 // 已看到第二个 PCB 带有手写标签，其中列出了匹配的 sum16 值，但未标记为“A”修订版
-// 虽然两个 PCB 都贴有 834-5667-01 REV.A，但对于此 ROM 集来说它不太可能正确
+// 虽然两个 PCB 都贴有 834-5667-01 REV.A，但对于该 ROM 集来说它不太可能正确
 //
 ROM_START（汉贡2）
        ROM_REGION( 0x40000, "maincpu", 0 ) // 68000 代码
@@ -1948,7 +1948,7 @@ ROM_START( endurobl )
        ROM_REGION( 0x40000, "maincpu", 0 ) // 68000 代码
        ROM_LOAD16_BYTE（“7.13j”，0x030000，0x08000，CRC（f1d6b4b7）SHA1（32bd966191cbb36d1e60ed1a06d4caa023dd6b88））
        ROM_继续（0x000000，0x08000）
- ROM_LOAD16_BYTE( "4.13h", 0x030001, 0x08000, CRC(43bff873) SHA1(04e906c1965a6211fb8e13987db52f1f99cc0203) ) // ROM 解码
+ ROM_LOAD16_BYTE( "4.13h", 0x030001, 0x08000, CRC(43bff873) SHA1(04e906c1965a6211fb8e13987db52f1f99cc0203) ) // ROM 解码
        ROM_CONTINUE( 0x000001, 0x08000 ) // 数据解码
        ROM_LOAD16_BYTE（“8.14j”，0x010000，0x08000，CRC（2153154a）SHA1（145d8ed59812d26ca412a01ae77cd7872adaba5a））
        ROM_LOAD16_BYTE（“5.14h”，0x010001，0x08000，CRC（0a97992c）SHA1（7a6fc8c575637107ed07a30f6f0f8cb8877cbb43））
@@ -2020,7 +2020,7 @@ ROM_END
 //
 ROM_START（endurob2）
        ROM_REGION( 0x040000, "maincpu", 0 ) // 68000 代码
- // 程序 rom 的大小应该是两倍
+ // 程序 ROM 的大小应该是两倍
        ROM_LOAD16_BYTE（“enduro.a07”，0x000000，0x08000，BAD_DUMP CRC（259069bc）SHA1（42fa47ce4a29294f9eff3eddbba6c305d750aaa5））
 // ROM_CONTINUE( 0x030000, 0x08000 )
        ROM_LOAD16_BYTE（“enduro.a04”，0x000001，0x08000，BAD_DUMP CRC（f584fbd9）SHA1（6c9ddcd1d9cf95c6250b705b27865644da45d197））
diff --git a/src/mame/drivers/seta2.cpp b/src/mame/drivers/seta2.cpp
索引 54e17bf2560..d4b778bf03c 100644
--- a/src/mame/drivers/seta2.cpp
+++ b/src/mame/drivers/seta2.cpp
@@ -89,8 +89,8 @@猎鹿，wschamp：
   进入鹿。在 wschamp 介绍中，GPS 装置应缩放至高分。

 wschampb：
-- 程序 ROM 的转储与每个芯片的手写校验和相匹配，但是
- 两个 ROM 的启动屏幕均报告 NG。- 这是正确的吗？是来自
+- 程序 ROM 的转储与每个芯片的手写校验和相匹配，但是
+ 两个 ROM 的启动屏幕均报告 NG。- 这是正确的吗？是来自
   原始版本？这就是为什么下一个错误修复版本是 v1.01 吗？即：这样一个
   版本号略有增加。

diff --git a/src/mame/drivers/shangha3.cpp b/src/mame/drivers/shangha3.cpp
索引 d59b2bb27b7..00509daa46d 100644
```

```
---a/src/mame/drivers/shangha3.cpp
+++ b/src/mame/drivers/shangha3.cpp
@@ -614,7 +614,7 @@ 内存：M1 = TMM2018AP-45 (2K x 8 SRAM)
  其他：SW1 和 SW2 - 8 位拨码开关
         VOL - 音量罐

-* = 未填充 32 针 ROM 插座，丝印 27C040
+* = 未填充的 32 针 ROM 插座，丝印 27C040

 注意：对于“World”集，它与 US 集不同（除了具有重复数据的 US 集）2 个字节。

@@ -629,8 +629,8 @@ ROM_START( shangha3 ) /* PCB 标记为 SUN04C - 有两个额外的块集可供 c
        ROM_LOAD16_BYTE( "ic2", 0x0001, 0x40000, CRC(714bfdbc) SHA1(0ce611624e8a5e28cba5443b63b8872eed9f68fc) ) /* ST M27C2001 无标签 EPROM */

        ROM_REGION( 0x400000, "gfx1", 0 )
- ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin 掩膜ROM */
- ROM_LOAD( "27c4000.ic44", 0x200000, 0x080000, CRC(6344ffb7) SHA1(06bc5bcf94973ec152e7abf9cc658ef319eb4b65) ) // 韩文字体，vs模式如何玩等？（可能
对于韩国程序 ROM，我们没有，但在世界板上）
+ ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin掩码ROM */
+ ROM_LOAD( "27c4000.ic44", 0x200000, 0x080000, CRC(6344ffb7) SHA1(06bc5bcf94973ec152e7abf9cc658ef319eb4b65) ) // 韩文字体，vs模式如何玩等？（可能
对于韩国程序 ROM，我们没有，但在世界板上）

        ROM_REGION( 0x40000, "oki", 0 ) /* M6295 的样本 */
        ROM_LOAD（“s3j_v10.ic75”，0x0000，0x40000，CRC（f0cdc86a）SHA1（b1017a9841a56e0f5d2714f550f64ed1f4e238e6））
@@ -643,7 +643,7 @@ ROM_START( shangha3u ) /* PCB 标记为 SUN04C - 显示 FBI “获胜者不使用毒品
        /* 两个程序 ROM 都是双倍大小，具有相同的两半 */

        ROM_REGION( 0x200000, "gfx1", 0 )
- ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin 掩膜ROM */
+ ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin掩码ROM */

        ROM_REGION( 0x80000, "oki", 0 ) /* M6295 的样本 */
        ROM_LOAD（“ IC75.IC75”，0x0000，0x80000，CRC（A8136D8C）SHA1（8028BDA5642C2546C1AC8DA78DBFF4084829F03B）） / * 27c4001 an
@@ -654,7 +654,7 @@ ROM_START( shangha3up ) /* PCB 标记为 SUN04，贴纸标记为 PCB 001, ap
        ROM_LOAD16_BYTE( "syan3u_evn_10-7.ic3", 0x0000, 0x40000, CRC(a1f5275a) SHA1(71a024205bd5e6385bd9d746c339f0327bd1c1d6) ) /* ST M27C2001
EPROM 手写标签：SYAN3U EVN 10/7 */
        ROM_LOAD16_BYTE( "syan3u_odd_10-7.ic2", 0x0001, 0x40000, CRC(fe3960bf) SHA1(545473260d959b8ed8145263d54f5f4523a844c4) ) /* ST M27C2001
EPROM 手写标签: SYAN3U 奇数 10/7 */

- ROM_REGION( 0x200000, "gfx1", 0 ) /* 与 42 pin MASK S3J CHAR-A1 相同的数据 */
+ ROM_REGION( 0x200000, "gfx1", 0 ) /* 与 42 引脚掩码 S3J CHAR-A1 相同的数据 */
        ROM_LOAD( "s3j_chr-a1_1_sum_53b1_93.9.20.ic80", 0x000000, 0x80000, CRC(fcaf795b) SHA1(312d85f39087564d67f12e0287f508b94b1493af) ) /* HN27C4
001 手写标签：S3J-CHR-A1 #1 SUM：53B1 93.9.20 */
        ROM_LOAD( "s3j_chr-a1_2_sum_0e32_93.9.20.ic81", 0x080000, 0x80000, CRC(5a564f50) SHA1(34ca2ecd7101961e657034082802d89db5b4b7bd) ) /*
HN27C40 01 手写标签：S3J-CHR-A1 #2 SUM：0E32 93.9.20 */
        ROM_LOAD( "s3j_chr-a1_3_sum_0d9a_93.9.20.ic82", 0x100000, 0x80000, CRC(2b333c69) SHA1(6e720de5d222be25857ab18902636587e8c6afb8) ) /*
HN27C400 1 个手写标签：S3J-CHR-A1 #3 SUM：0D9A 93.9.20 */
```

```
 @@ -670,7 +670,7 @@ ROM_START( shangha3j ) /* PCB 标记为 SUN04C */
         ROM_LOAD16_BYTE（“s3j_v11.ic2”，0x0001，0x40000，CRC（09174620）SHA1（1d1639c07895f715facfe153fbdb6ae0f3cdd876））

         ROM_REGION( 0x200000, "gfx1", 0 )
 - ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin 掩膜ROM */
 + ROM_LOAD( "s3j_char-a1.ic43", 0x0000, 0x200000, CRC(2dbf9d17) SHA1(dd94ddc4bb02ab544aa3f89b614afc46678cc48d) ) /* 42pin掩码ROM */

         ROM_REGION( 0x40000, "oki", 0 ) /* M6295 的样本 */
         ROM_LOAD（“s3j_v10.ic75”，0x0000，0x40000，CRC（f0cdc86a）SHA1（b1017a9841a56e0f5d2714f550f64ed1f4e238e6））
 diff --git a/src/mame/drivers/speglsht.cpp b/src/mame/drivers/seglsht.cpp
 索引 3515d9dc1e5..aae854edba4 100644
 ---a/src/mame/drivers/speglsht.cpp
 +++ b/src/mame/drivers/speglsht.cpp
 @@ -99,9 +99,9 @@ 注意：
        SW4：8 位拨码开关
        U30,U31,
        U32、U33：Macronix MX27C4000 512k x8 EPROM（DIP32，PCB 标记为“RPRO0”、“RPRO1”、“RPRO2”、“RPRO3”）
 - U34、U35：8M MASKROM（DIP42，PCB 标记为“RD0”、“RD1”）
 - U70：16M MASKROM（DIP42，PCB 标记为“ZPRO0”）
 - * ：16M DIP42 MASKROM 的未填充位置（PCB 标记为“ZPRO1”）
 + U34、U35：8M 掩膜 ROM（DIP42，PCB 标记为“RD0”、“RD1”）
 + U70：16M 掩模 ROM（DIP42，PCB 标记为“ZPRO0”）
 + * : 16M DIP42 mask ROM 的未填充位置（PCB 标记为“ZPRO1”）

  */

 diff --git a/src/mame/drivers/ssv.cpp b/src/mame/drivers/ssv.cpp
 索引 7300cc1863c..f8513657799 100644
 ---a/src/mame/drivers/ssv.cpp
 +++ b/src/mame/drivers/ssv.cpp
 @@ -497,7 +497,7 @@ void ssv_state::gdfs_map(address_map &map)
      在 0x580000-0x5bffff。测试被跳过并且该内存未被使用
      尽管。我猜这要么是剩下的，要么是有不同的
      带有一些电池支持 RAM 的版本（这确实是在
 - ROM板，据我所知）
 + ROM 板，据我所知）
  */

  READ16_MEMBER(ssv_state::hypreact_input_r)
 @@ -776,7 +776,7 @@ void ssv_state::sxyreact_map(address_map &map)
                                 双鹰II
  ********************************************************** **************************/

 -/* 作为独立板或标准 SSV rom 板（已验证）*/
 +/* 作为独立板或标准 SSV ROM 板（已验证）*/

  无效 ssv_state::twinag2_map(address_map &map)
```

```
  {
@@ -2441,8 +2441,8 @@ static const gfx_layout layout_16x8x6_ram =
 };

 静态 GFXDECODE_START( gfx_eaglshot )
- GFXDECODE_ENTRY( nullptr, 0,layout_16x8x8_ram, 0, 0x8000/64 ) // [0] 精灵（256 色，从 ram 解码）
- GFXDECODE_ENTRY( nullptr, 0,layout_16x8x6_ram, 0, 0x8000/64 ) // [1] 精灵（64 色，从 ram 解码）
+ GFXDECODE_ENTRY( nullptr, 0,layout_16x8x8_ram, 0, 0x8000/64 ) // [0] 精灵（256 色，从 RAM 解码）
+ GFXDECODE_ENTRY( nullptr, 0,layout_16x8x6_ram, 0, 0x8000/64 ) // [1] 精灵（64 色，从 RAM 解码）
 GFXDECODE_END

 静态常量 gfx_layout 布局_16x16x8 =
@@ -2473,7 +2473,7 @@ GFXDECODE_END
 /************************************************** *************************

     一些游戏（例如 hypreac2）奇怪地映射了图块代码的高位
- 对于 gfx rom：相应地排列 rom 会浪费数十
+ gfx ROM：相应地排列 ROM 会浪费数十
     兆字节。所以我们使用查找表。

     我们还需要为精灵和图层设置游戏特定的偏移量
@@ -3191,7 +3191,7 @@ ROM：U18 和 U20 用于主程序。
             其余均为16M面膜
             U23 和 U24 与声音相关，所有其他与 GFX 相关。

-Loc ROM 使用和 eprom 类型
+Loc ROM 使用和 EPROM 类型
 ----------------------------------------------------
 U18 si003-09.prl - V60 程序 (27C4001)
 U20 si003-10.prh /
@@ -3206,7 +3206,7 @@ U10 si003-04.d3 |
 U30 si003-05.d4 |
 U31 si003-06.d5 /

-注：上面的“s”和“d”名称是丝印在 ROM PCB 上的。
+注：上面的“s”和“d”名称是丝印在 ROM PCB 上的。

 值得注意的芯片：mc14584b - Motorola HEX 施密特触发器

@@ -4288,9 +4288,9 @@ ADC0809CCN：8位微处理器兼容A/D转换器，具有8通道乘法
 * 表示未填充的组件

 U37、U33 = 27c040
-U22、U41、U35、U25、U21、U11、U7 = 16 MEG 掩模 ROM
-U32、U18、U4 = 4 MEG 面具 ROM
-U26 = 8 MEG 掩模 ROM
+U22、U41、U35、U25、U21、U11、U7 = 16 兆掩码 ROM
```

```
 +U32、U18、U4 = 4 兆掩模 ROM
 +U26 = 8兆掩模ROM

  ********************************************** **********************/

 @@ -4390,7 +4390,7 @@ STS0003 双鹰

    SX002-13：GAL16V8B（未转储）
  SETA ST010：用于数学计算的定制编程 uPD96050 MCU
 -所有ROM均为16M Mask ROM
 +所有ROM均为16M掩膜ROM

  ********************************************** **********************/

 @@ -4512,7 +4512,7 @@ 维斯科游戏，2000 / 2001
  这是一款适合任何标准 SSV 主板的子板。

  ROM板上的东西很少。只需 2 个 27C040 EPROM，
 -4x 64Mbit SOP44 MASK ROM、3x 16Mbit SOP44 MASK ROM、
 +4x 64Mbit SOP44 掩膜 ROM、3x 16Mbit SOP44 掩膜 ROM、
  PROG & DATA ROM 附近有一些逻辑和 2 个 PAL。

  实际的 ROM PCB 能够接受 SOP44 和
 @@ -4568,10 +4568,10 @@ 文件名标记为 Loc。印刷* ROM 类型
  prg-h.u31 PRG-H U31 U31 PRG H | 27C040
  prg-l.u30 PRG-L U20 U30 PRG L /
  s1.u37 C DAT VASARA-1 U37 S1 \
 -s0.u36 B DAT VASARA-1 U36 S0 | 表面安装 16Mbit SOP44 MASK ROM
 +s0.u36 B DAT VASARA-1 U36 S0 | 表面贴装 16Mbit SOP44 掩模 ROM
  data.u34 A SND 1 VASARA-1 U34 数据 ROM /
  d0.u4 VASARA-2-D0 U4 D0.D1 \
 -c0.u3 VASARA-2-C0 U3 C0.C1 | 表面安装 64Mbit SOP44 MASK ROM
 +c0.u3 VASARA-2-C0 U3 C0.C1 | 表面贴装 64Mbit SOP44 掩模 ROM
  b0.u2 VASARA-2-B0 U2 B0.B1 | b0.u2 VASARA-2-B0 U2 B0.B1 |
  a0.u1 VASARA-2-A0 U1 A0.A1 /

 @@ -4762,7 +4762,7 @@ 游戏( 1993, dynagear, 0, dynagear, dynagear, ssv_state, init_dynagear,
  GAME( 1993, keithlcy, 0, keithlcy, keithlcy, ssv_state, init_keithlcy, ROT0, "Visco", "戏剧性冒险测验 Keith & Lucy (日本)", MACHINE_NO_COCKTAIL |
 MACHINE_SUPPORTS_SAVE )

  游戏( 1993, srmp4, 0, srmp4, srmp4, ssv_state, init_srmp4, ROT0, "Seta", "超级真实麻将 PIV (日本)", MACHINE_NO_COCKTAIL | MACHINE_SUPPORTS_SAVE )
 -GAME( 1993, srmp4o, srmp4, srmp4, srmp4, ssv_state, init_srmp4, ROT0, "Seta", "Super Real Mahjong PIV (Japan, old set)", MACHINE_NO_COCKTAIL |
 MACHINE_SUPPORTS_SAVE ) // 通过程序 ROM 的编号应该更老
 +GAME( 1993, srmp4o, srmp4, srmp4, srmp4, ssv_state, init_srmp4, ROT0, "Seta", "Super Real Mahjong PIV (Japan, old set)", MACHINE_NO_COCKTAIL |
 MACHINE_SUPPORTS_SAVE ) // 通过程序 ROM 的编号应该更老

  游戏（1993，survarts，0，survarts，survarts，ssv_state，init_survarts，ROT0，“萨米”，“生存艺术（世界）”，MACHINE_NO_COCKTAIL |
```

```
MACHINE_SUPPORTS_SAVE)
 游戏（1993，survartsu，survarts，survarts，survarts，ssv_state，init_survarts，ROT0，“美国萨米”，“生存艺术（美国）”，MACHINE_NO_COCKTAIL |
MACHINE_SUPPORTS_SAVE ）
diff --git a/src/mame/drivers/studio2.cpp b/src/mame/drivers/studio2.cpp
索引 40282dc6999..ab9a8fac273 100644
---a/src/mame/drivers/studio2.cpp
+++ b/src/mame/drivers/studio2.cpp
@@ -35,7 +35,7 @@ 注意：
       CDP1802 - RCA CDP1802CE 微处理器
       TA10171V1 - RCA TA10171V1 NTSC 视频显示控制器 (VDC) (= RCA CDP1861)
       CDP1822 - RCA CDP1822NCE 256 x4 RAM（= 三菱 M58721P）
- ROM.x - RCA CDP1831CE 512 x8 掩模ROM。所有 ROM 均标有“程序版权 (C) RCA CORP. 1977”
+ ROM.x - RCA CDP1831CE 512 x8 掩模 ROM。所有 ROM 均标有“程序版权 (C) RCA CORP. 1977”
       CD4001 - 4001 四路 2 输入 NOR 缓冲 B 系列门（4000 系列 CMOS TTL 逻辑 IC）
       CD4042 - 4042 四时钟 D 锁存器（4000 系列 CMOS TTL 逻辑 IC）
       CD4515 - 4515 4 位锁存/4 至 16 线路解码器（4000 系列 CMOS TTL 逻辑 IC）
@@ -77,7 +77,7 @@ 注：（上面显示的所有芯片）
                 时钟 - 223.721562kHz [3.579545/16]（在引脚 1 上测量）
       2111 - NEC D2111AL-4 256 字节 x4 SRAM（DIP18，x6）。总计 1.5k
       C - 从电视调制器到电视的复合视频输出
- TMM331 - 东芝 TMM331AP 2k x8 MASKROM (DIP24)
+ TMM331 - 东芝 TMM331AP 2k x8 掩模 ROM (DIP24)
                 引脚排列：
                           TMM331
                         |----\/----|
@@ -165,7 +165,7 @@ 注意：
       CDP1802 - RCA CDP1802CE 微处理器
       CDP1864 - RCA CDP1864CE PAL 视频显示控制器 (VDC)
       CDP1822 - RCA CDP1822NCE 256 x4 RAM（= 三菱 M58721P）
- ROM.ICx - RCA CDP1833 1k x8 MASKROM。所有 ROM 均标有“程序版权 (C) RCA CORP. 1978”
+ ROM.ICx - RCA CDP1833 1k x8 掩模 ROM。所有 ROM 均标有“程序版权 (C) RCA CORP. 1978”
       CD4019 - 4019 四路与或选择门（4000 系列 CMOS TTL 逻辑 IC）
       CDP1858 - RCA CDP1858E 锁存器/解码器 - 4 位
       CD4081 - 4081 四路 2 输入与缓冲 B 系列门（4000 系列 CMOS TTL 逻辑 IC）
diff --git a/src/mame/drivers/subsino2.cpp b/src/mame/drivers/subsino2.cpp
索引 232e3e614b1..59885ec840a 100644
---a/src/mame/drivers/subsino2.cpp
+++ b/src/mame/drivers/subsino2.cpp
@@ -2589,8 +2589,8 @@ PCB 布局
 |------------------------------------------- -----|
 笔记：
       H8/3044 - Subsino 重新贴标 Hitachi H8/3044 HD6433044A22F 微控制器 (QFP100)
- H8/3044 是具有 24 位地址总线的 H8/3002，具有 32k MASKROM 和 2k RAM，时钟输入为 14.7MHz [44.1/3]
- MD0、MD1 和 MD2 配置为模式 6 16MByte 扩展模式，并启用片上 32k MASKROM。
+ H8/3044 是具有 24 位地址总线的 H8/3002，具有 32k mask ROM 和 2k RAM，时钟输入为 14.7MHz [44.1/3]
+ MD0、MD1 和 MD2 配置为模式 6 16MByte 扩展模式，并启用片上 32k 掩模 ROM。
      CXK58257 - 索尼 CXK58257 32k x8 SRAM (SOP28)
```

```
        HM86171 - 华隆微电子 HMC HM86171 VGA 256色 RAMDAC (DIP28)
            S-1 - ?? 可能是某种音频运算放大器或 DAC? (DIP8)
diff --git a/src/mame/drivers/suprnova.cpp b/src/mame/drivers/suprnova.cpp
索引 30642ef142d..7278a77fefb 100644
---a/src/mame/drivers/suprnova.cpp
+++ b/src/mame/drivers/suprnova.cpp
@@ -89,9 +89,9 @@ NEP-16
  \- \-

  笔记：
- *: 表面贴装 16MBit SOP44 MASK ROM 的未填充位置
+ *: 表面贴装 16MBit SOP44 掩模 ROM 的未填充位置
        U8和U10插接27C040 EPROM
- 所有其他 ROM 均为表面贴装 SOP44 MASK ROM
+ 所有其他 ROM 均为表面贴装 SOP44 掩膜 ROM

 购物车布局
 ------------
@@ -1036,7 +1036,7 @@ void skns_state::init_galpans3() { m_spritegen->skns_sprite_kludge(-1,-1); .ini
 #define ROM_LOAD_BIOS(BIOS,名称,偏移量,长度,哈希值) \
                ROMX_LOAD(名称、偏移量、长度、散列、ROM_BIOS(BIOS))

- /* 注意：欧洲 BIOS ROM 已被发现标记为 SKNSE1 和 SKNSE2，但数据是相同的 */
+ /* 注意：欧洲 BIOS ROM 已被发现标记为 SKNSE1 和 SKNSE2，但数据相同 */
 #定义 SKNS_BIOS \
        ROM_REGION( 0x0100000, "主CPU", 0 ) \
        ROM_SYSTEM_BIOS( 0, "日本", "日本" ) \
@@ -1161,7 +1161,7 @@ ROM_START( galpani4 ) // 仅提供主 CPU 和 plds 转储

        ROM_REGION( 0x400000, "ymz", 0 ) /* 样本 */
        ROM_LOAD( "gp4-300-00.u4", 0x000000, 0x200000, CRC(8374663a) SHA1(095512564f4de25dc3752d9fbd254b9dabd16d1b) ) /* 似乎根本没有使用这些样本
*/
- ROM_LOAD( "gp4-301-00.u7", 0x200000, 0x200000, NO_DUMP ) /* 与 GP4-301-01 不同 - 与 U4 rom 相比更改了一些示例 */
+ ROM_LOAD( "gp4-301-00.u7", 0x200000, 0x200000, NO_DUMP ) /* 与 GP4-301-01 不同 - 与 U4 ROM 相比更改了一些示例 */

        ROM_REGION( 0x400, "plds", 0 )
        ROM_LOAD( "skns-r09.u9", 0x000, 0x117, CRC(b02058d9) SHA1(77d07e0f329fb1969aa4543cd124e36ad34b07ba) ) // Atmel ATF16V8B
@@ -1191,7 +1191,7 @@ ROM_START( galpani4j )
        ROM_LOAD（“gp4-300-00.u4”，0x000000，0x200000，CRC（8374663a）SHA1（095512564f4de25dc3752d9fbd254b9dabd16d1b））
 ROM_END

-ROM_START( galpani4k ) /* ROM-BOARD NEP-16 部件号 GP04K00372，在 U7 处带有额外的声音样本 ROM */
+ROM_START( galpani4k ) /* ROM-BOARD NEP-16 部件号 GP04K00372，在 U7 处带有额外的声音样本 ROM */
        SKNS_韩国

        ROM_REGION32_BE( 0x200000, "user1", 0 ) /* SH-2 代码映射到 0x04000000 */
@@ -1212,7 +1212,7 @@ ROM_START( galpani4k ) /* ROM-BOARD NEP-16 部件号 GP04K00372 带有额外的 sou
```

```
        ROM_REGION( 0x400000, "ymz", 0 ) /* 样本 */
        ROM_LOAD( "gp4-300-00.u4", 0x000000, 0x200000, CRC(8374663a) SHA1(095512564f4de25dc3752d9fbd254b9dabd16d1b) ) /* 似乎根本没有使用这些样本
*/
- ROM_LOAD( "gp4-301-01.u7", 0x200000, 0x200000, CRC(886ef77f) SHA1(047d5fecf2034339c69b2cb605b623a814a18f0d) ) /* 与 U4 rom 相比更改了一些示例 */
+ ROM_LOAD( "gp4-301-01.u7", 0x200000, 0x200000, CRC(886ef77f) SHA1(047d5fecf2034339c69b2cb605b623a814a18f0d) ) /* 与 U4 ROM 相比更改了一些示例 */
 ROM_END

 ROM_START( galpanidx )
@@ -1236,7 +1236,7 @@ ROM_START( galpanidx )

        ROM_REGION( 0x400000, "ymz", 0 ) /* 样本 */
        ROM_LOAD( "gp4-300-00.u4", 0x000000, 0x200000, CRC(8374663a) SHA1(095512564f4de25dc3752d9fbd254b9dabd16d1b) ) /* 似乎根本没有使用这些样本
*/
- ROM_LOAD( "gp4-301-01.u7", 0x200000, 0x200000, CRC(886ef77f) SHA1(047d5fecf2034339c69b2cb605b623a814a18f0d) ) /* 与 U4 rom 相比更改了一些示例 */
+ ROM_LOAD( "gp4-301-01.u7", 0x200000, 0x200000, CRC(886ef77f) SHA1(047d5fecf2034339c69b2cb605b623a814a18f0d) ) /* 与 U4 ROM 相比更改了一些示例 */
 ROM_END

 ROM_START（加尔帕尼斯）
@@ -1267,8 +1267,8 @@ ROM_START( galpanise )
        SKNS_欧洲

        ROM_REGION32_BE( 0x200000, "user1", 0 ) /* SH-2 代码映射到 0x04000000 */
- ROM_LOAD16_BYTE( "u10", 0x000000, 0x100000, CRC(e78e1623) SHA1(f68346b65d2613c8515894d9a239fcbb0b5cb52d) ) /* 没有标签的 MASK rom */
- ROM_LOAD16_BYTE( "u8", 0x000001, 0x100000, CRC(098eff7c) SHA1(3cac22cbb11905a46afaa62c0470624b3b554fc0) ) /* 没有标签的 MASK rom */
+ ROM_LOAD16_BYTE( "u10", 0x000000, 0x100000, CRC(e78e1623) SHA1(f68346b65d2613c8515894d9a239fcbb0b5cb52d) ) /* 没有标签的掩码 ROM */
+ ROM_LOAD16_BYTE( "u8", 0x000001, 0x100000, CRC(098eff7c) SHA1(3cac22cbb11905a46afaa62c0470624b3b554fc0) ) /* 没有标签的掩码 ROM */

        ROM_REGION( 0x1000000, "精灵", 0 )
        ROM_LOAD（“gps10000.u24”，0x000000，0x400000，CRC（a1a7acf2）SHA1（52c86ae907f0c0236808c19f652955b09e90ec5a））
@@ -1438,7 +1438,7 @@ ROM_START( galpansu )
        ROM_LOAD16_BYTE（“su.u10”，0x000000，0x100000，CRC（5ae66218）SHA1（c3f32603e1da945efb984ff99e1a30202e535773））
        ROM_LOAD16_BYTE（“su.u8”，0x000001，0x100000，CRC（10977a03）SHA1（2ab95398d6b88d8819f368ee6104d7f8b485778d））

- /* 其余的 rom 与 Gals Panic S2 匹配，但位于不同的位置 */
+ /* 其余 ROM 与 Gals Panic S2 匹配，但位于不同位置 */
        ROM_REGION( 0x1000000, "精灵", 0 )
        ROM_LOAD（“24”，0x000000，0x400000，CRC（294b2f14）SHA1（90cbd0acdaa2d89d208c28aae33ab57c03624089））
        ROM_LOAD（“20”，0x400000，0x400000，CRC（f75c5a9a）SHA1（3919643cee6c88185a1aa3c58c5bc80599bf734e））
diff --git a/src/mame/drivers/taito_f3.cpp b/src/mame/drivers/taito_f3.cpp
索引 703933526c4..9270c17a389 100644
--- a/src/mame/drivers/taito_f3.cpp
+++ b/src/mame/drivers/taito_f3.cpp
@@ -1592,8 +1592,8 @@ 注：
                        PAL.7 PALCE16V8H
                        D49-12-1 PAL16L8B
```

```
- D69-01 至 D69-08 16M 掩模只读存储器 (DIP42)
- d69-09 至 d69-11 8M 掩模ROM (DIP42)
+ D69-01 至 D69-08 16M 掩模 ROM (DIP42)
+ d69-09 至 d69-11 8M 掩膜 ROM (DIP42)
       D69-13 至 D69-15/D69-20 27C040 EPROM (DIP32)
       D69-18/D69-19 27C1001 EPROM (DIP32)
 */
diff --git a/src/mame/drivers/tnzs.cpp b/src/mame/drivers/tnzs.cpp
索引 721e116a340..2f06eed84a8 100644
---a/src/mame/drivers/tnzs.cpp
+++ b/src/mame/drivers/tnzs.cpp
@@ -545,7 +545,7 @@ f000-f003 输入（仅由打砖块 2 使用）
 e001=dip-sw A e399=硬币计数器值 e72c-d=1P 桨（低-高）
 e002=dip-sw B e3a0-2=1P 分数/10 (BCD) e72e-f=2P 桨 (lo-hi)
 e008=level=2*(shown_level-1)+x <- 记住它是一个二叉树（最后 42 个）
-e7f0=国家代码（来自声音ROM中的9fde）
+e7f0=国家代码（来自声音ROM中的9fde）
 e807=计数器，由声音CPU重置，每个vblank由主CPU增加
 e80b=测试进度=0(开始) 1(前8) 2(全部正常) 3(错误)
 ec09-a~=ed05-6=高分中光标的 xy 位置
@@ -753,7 +753,7 @@ void tnzsb_state::tnzsb_main_map(address_map &map)
        映射（0xf300，0xf303）.mirror（0xfc）.w（m_seta001，FUNC（seta001_device :: spritectrl_w8））；/* 控制寄存器（Arkanoid 2 使用的 0x80 镜像）
*/
        地图（0xf400，0xf400）。w（m_seta001，FUNC（seta001_device :: spritebgflag_w8））；/* 启用/禁用背景透明度 */
        地图（0xf600，0xf600）。w（FUNC（tnzsb_state :: ramrom_bankswitch_w））；
- /* kabukiz 仍然写在这里，但它没有被使用（它是 type1 地图中的 Paletteram）*/
+ /* kabukiz 仍然在这里写入，但没有使用（它是 type1 映射中的调色板 RAM）*/
        地图(0xf800, 0xfbff).nopw();
 }

@@ -1743,10 +1743,10 @@ MACHINE_CONFIG_END
 ********************************************** ****************************/
 /* TNZS/Seta 硬件有多种略有不同的 PCB，全部
     具有 Seta 和 Taito 零件号。
- 所有 PCB 均配有 Z80B 处理器和一个 6264 mainram 芯片以及一个 X1-001
+ 所有 PCB 均配有 Z80B 处理器和一个 6264 主 RAM 芯片以及一个 X1-001
     和 X1-002 视频芯片和 X1-004 I/O? 芯片和四个 PAL

-Seta# Taito#s CPUS RxM2 ROM1 MCU? 视频 RAM PROM SETA X1 GFXROM QUADRATURE ESD。PROT游戏图片
+Seta# Taito#s CPUS RxM2 ROM1 MCU? 视频 RAM PROM SETA X1 GFXROM 正交 ESD。PROT游戏图片
 PO-022-A K1100245A J1100108A 2xZ80B 512/256 512/256 8042 4x6116 是，2 03 23c1000 uPD4701AC 3x X2-003*4 arkanoid2
http://www.classicarcaderesource.com/RevengeOfDoh3.jpg
 PO-022-B K1100234A J1100108A 2xZ80B 512/256 512/256 8042 4x6116 有，2 03 27c512(A) uPD4701AC 3x X2-003*4 丰满 不适用
 PO-025-A K1100241A J1100107A 2xZ80B 512/256 512/256 8042 4x6116 有，2 03 23c1000 N/A 3x X2-003 drtoppel,extermatn,chukatai(B)
http://arcade.ym2149.com/pcb/taito / drtoppel_pcb_partside.jpg
@@ -1757,16 +1757,16 @@ PO-041-A K1100356A J1100156A 2xZ80B 61256 27c1000 8042 1x6164 无
 PO-043A M6100356A 3xZ80B* 61256 27512** 无 1x6164 无 05,06 LH534000* N/A 4x X2-004 tnzs(j,u)，kabukiz
```

```
http://arcade.ym2149.com/pcb/taito/tnzs_pcb2_mainboard_partside。 .jpg
 PO-056A K1100476A J1100201A 3xZ80B 空*3 27c1000 无 1x6164 无 05,06 LH534000 N/A 5x X2-005 昆虫x(D)
http://www.jammarcade.net/images/2014/04/InsectorX.jpg

-(A) GFX ROM 映射与 PO-022-A PCB 略有不同，可能已配置
+(A) GFX ROM 映射与 PO-022-A PCB 略有不同，可能已配置
     通过跳线。
 (B) chukatai 有一套与早期的套不同，它使用 PO-025-A
     PCB，但带有可转换四个 23c1000 gfx ROM 的子板
- 插入 8 个 27c1000 eprom 插槽，并且使用彩色 PROM!
- 另一组 PCB 使用 PO-028-A PCB 和 23c1000 mask rom 和彩色 RAM，
- 但 ROM ID 号较低。可能创建了编号较高的集合
+ 插槽插入 8 个 27c1000 EPROM 插槽，并且确实使用彩色 PROM!
+ 另一组 PCB 使用 PO-028-A PCB 和 23c1000 掩模 ROM 和彩色 RAM，
+ 但 ROM ID 号较低。可能创建了编号较高的集合
     由 Taito 来“用完”旧的 PO-025-A PCB 的库存。
 (C) 这是一个开发/原型 PCB，因此它有 32 针插座
- gfx ROM 作为 27c1000 eprom，而不是 23c1000 掩码的 28 针插座
+ gfx ROM 作为 27c1000 EPROM，而不是 23c1000 掩模的 28 针插座
     ROM。它还使用（未受保护？）8742 MCU。
     另一个奇怪的事情是 Taito ID 号可能被意外地泄露了
     按向后顺序打印，即应该是符合图案的 C1100304A
@@ -1821,7 +1821,7 @@ ROM_START( 丰满的)
        ROM_LOAD( "a98__08.mbm27c512.2a", 0xe0000, 0x10000, CRC(bfa7609a) SHA1(0b9aa89b5954334f40dda1f14b1691852c74fc37) )
        ROM_RELOAD（0xf0000，0x10000）

- ROM_REGION( 0x0400, "proms", 0 ) /* 颜色 proms */
+ ROM_REGION( 0x0400, "proms", 0 ) /* 彩色 PROM */
        ROM_LOAD( "a98-13.15f", 0x0000, 0x200, CRC(7cde2da5) SHA1(0cccfc35fb716ebb4cffa85c75681f33ca80a56e) ) /* hi 字节，AM27S29 或兼容 MB7124 */
        ROM_LOAD( "a98-12.17f", 0x0200, 0x200, CRC(90dc9da7) SHA1(f719dead7f4597e5ee6f1103599505b98cb58299) ) /* lo 字节，AM27S29 或兼容 MB7124 */

@@ -2095,7 +2095,7 @@ ROM_START( drtoppel )
        ROM_LOAD（“b19__10.9c”，0x10000，0x10000，CRC（7e72fd25）SHA1（6035e4db75e6dc57b13bb6e92217d1c2d0ffdfd2））

        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
- ROM_LOAD( "b19__15.3e", 0x00000, 0x10000, BAD_DUMP CRC(37a0d3fb) SHA1(f65fb9382af5f5b09725c39b660c5138b3912f53) ) /* Region-Hacked??，需要正确的
Taito rom 编号 */
+ ROM_LOAD( "b19__15.3e", 0x00000, 0x10000, BAD_DUMP CRC(37a0d3fb) SHA1(f65fb9382af5f5b09725c39b660c5138b3912f53) ) /* Region-Hacked??，需要正确的
Taito ROM 编号 */

        ROM_REGION( 0x10000, "mcu", 0 ) /* M-Chip (i8x42 内部 ROM) */
        ROM_LOAD( "b06__14.1g", 0x0000, 0x0800, CRC(28907072) SHA1(21c7017af8a8ceb8e43d7e798f48518b136fd45c) ) /* 标记为 B06 // 14 及印刷标签下
"Taito M-001, 128P, 7 20100”，是掩码8042 */
@@ -2110,7 +2110,7 @@ ROM_START( drtoppel )
        ROM_LOAD（“b19-07.23c1000.4a”，0xc0000，0x20000，CRC（8bb06f41）SHA1（a0c182d473317f2cdb31bdf39a2593c032002305））
        ROM_LOAD（“b19-08.23c1000.2a”，0xe0000，0x20000，CRC（3584b491）SHA1（d0aca90708be241bbd3a1097220a85083337a4bc））
```

```
- ROM_REGION( 0x0400, "proms", 0 ) /* 颜色 proms */
+ ROM_REGION( 0x0400, "proms", 0 ) /* 彩色 PROM */
        ROM_LOAD( "b19-13.am27s29.15f", 0x0000, 0x200, CRC(6a547980) SHA1(c82f8dfad028565b4b4e5be1167f2f290c929090) ) /* hi 字节, AM27S29 或兼容
MB7124 */
        ROM_LOAD( "b19-12.am27s29.16f", 0x0200, 0x200, CRC(5754e9d8) SHA1(8c7d29e22c90b1f72929b95675dc15e431aae044) ) /* lo 字节, AM27S29 或兼容
MB7124 */

@@ -2127,7 +2127,7 @@ ROM_START( drtoppelu )
        ROM_LOAD（“b19__10.9c”，0x10000，0x10000，CRC（7e72fd25）SHA1（6035e4db75e6dc57b13bb6e92217d1c2d0ffdfd2））

        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
- ROM_LOAD( "b19__14.3e", 0x00000, 0x10000, BAD_DUMP CRC(05565b22) SHA1(d1aa47b438d3b44c5177337809e38b50f6445c36) ) /* Region-Hacked??, 需要正确的
Taito rom 编号 */
+ ROM_LOAD( "b19__14.3e", 0x00000, 0x10000, BAD_DUMP CRC(05565b22) SHA1(d1aa47b438d3b44c5177337809e38b50f6445c36) ) /* Region-Hacked??, 需要正确的
Taito ROM 编号 */

        ROM_REGION( 0x10000, "mcu", 0 ) /* M-Chip (i8x42 内部 ROM) */
        ROM_LOAD( "b06__14.1g", 0x0000, 0x0800, CRC(28907072) SHA1(21c7017af8a8ceb8e43d7e798f48518b136fd45c) ) /* 标记为 B06 // 14 及印刷标签下
"Taito M-001, 128P, 7 20100”，是掩码8042 */
@@ -2142,7 +2142,7 @@ ROM_START( drtoppelu )
        ROM_LOAD（“b19-07.23c1000.4a”，0xc0000，0x20000，CRC（8bb06f41）SHA1（a0c182d473317f2cdb31bdf39a2593c032002305））
        ROM_LOAD（“b19-08.23c1000.2a”，0xe0000，0x20000，CRC（3584b491）SHA1（d0aca90708be241bbd3a1097220a85083337a4bc））

- ROM_REGION( 0x0400, "proms", 0 ) /* 颜色 proms */
+ ROM_REGION( 0x0400, "proms", 0 ) /* 彩色 PROM */
        ROM_LOAD( "b19-13.am27s29.15f", 0x0000, 0x200, CRC(6a547980) SHA1(c82f8dfad028565b4b4e5be1167f2f290c929090) ) /* hi 字节, AM27S29 或兼容
MB7124 */
        ROM_LOAD( "b19-12.am27s29.16f", 0x0200, 0x200, CRC(5754e9d8) SHA1(8c7d29e22c90b1f72929b95675dc15e431aae044) ) /* lo 字节, AM27S29 或兼容
MB7124 */

@@ -2174,7 +2174,7 @@ ROM_START( drtoppelj )
        ROM_LOAD（“b19-07.23c1000.4a”，0xc0000，0x20000，CRC（8bb06f41）SHA1（a0c182d473317f2cdb31bdf39a2593c032002305））
        ROM_LOAD（“b19-08.23c1000.2a”，0xe0000，0x20000，CRC（3584b491）SHA1（d0aca90708be241bbd3a1097220a85083337a4bc））

- ROM_REGION( 0x0400, "proms", 0 ) /* 颜色 proms */
+ ROM_REGION( 0x0400, "proms", 0 ) /* 彩色 PROM */
        ROM_LOAD( "b19-13.am27s29.15f", 0x0000, 0x200, CRC(6a547980) SHA1(c82f8dfad028565b4b4e5be1167f2f290c929090) ) /* hi 字节, AM27S29 或兼容
MB7124 */
        ROM_LOAD( "b19-12.am27s29.16f", 0x0200, 0x200, CRC(5754e9d8) SHA1(8c7d29e22c90b1f72929b95675dc15e431aae044) ) /* lo 字节, AM27S29 或兼容
MB7124 */

@@ -2271,7 +2271,7 @@ ROM_START( kagekij )
        ROM_LOAD（“b06-13.pal16l8a.c2.jed”，0x03000，0x01000，NO_DUMP）
 ROM_END

-/* 主板 ID 为 M6100309A - 程序 ROM 已被黑客入侵，显示 1992 年：/
+/* 主板 ID 为 M6100309A - 程序 ROM 已被黑客入侵，说 1992 :/
```

```
      支持，因为它似乎是与其他受支持集不同的代码修订版
 */

@@ -2341,7 +2341,7 @@ 台东: K1100416A J1100332A
 笔记：
       6264：8K x8 SRAM
       6116：2K x8 SRAM
- 图形 ROM 为 23C1000/TC531000 MASK ROM
+ 图形 ROM 为 23C1000/TC531000 掩模 ROM


 中国大战“后期”版本：
@@ -2350,7 +2350,7 @@中华大战‘后期’版本：
 贴纸：K1100364A 中国大战

 技术上较新的 ROM ID#，但用于摆脱旧的 PCB 库存。
- 该套件与 PO-028-A PCB 版本不同，使用两种颜色的 proms。
+与 PO-028-A PCB 版本不同，该套件使用两种颜色的 PROM。


 **********************************************
@@ -2390,7 +2390,7 @@ ROM_START( chukatai )
        ROM_LOAD（“b44-11”，0x10000，0x10000，CRC（32484094）SHA1（f320fea2910816b5085ca9aa37e30af665fb6be1））

        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
- ROM_LOAD( "b44-12w", 0x00000, 0x10000, CRC(e80ecdca) SHA1(cd96403ca97f18f630118dcb3dc2179c01147213) ) /* 被黑客攻击？？，需要正确的 Taito rom 编号 */
+ ROM_LOAD( "b44-12w", 0x00000, 0x10000, CRC(e80ecdca) SHA1(cd96403ca97f18f630118dcb3dc2179c01147213) ) /* 被黑客攻击？？，需要正确的 Taito ROM 编号 */

        ROM_REGION( 0x10000, "mcu", 0 ) /* M-Chip (i8x42 内部 ROM) */
        ROM_LOAD( "b44-8742.mcu", 0x0000, 0x0800, CRC(7dff3f9f) SHA1(bbf4e036d025fe8179b053d639f9b8ad401e6e68) ) /* B44 // 09 是标签？标签下的口罩编号是多少？也许是Taito M-011？最后一位数字肯定是1 */
@@ -2418,7 +2418,7 @@ ROM_START( chukataiu )
        ROM_LOAD（“b44-11”，0x10000，0x10000，CRC（32484094）SHA1（f320fea2910816b5085ca9aa37e30af665fb6be1））

        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
- ROM_LOAD( "b44-12u", 0x00000, 0x10000, BAD_DUMP CRC(9f09fd5c) SHA1(ae92f2e893e1e666dcabbd793f1a778c5e3d7bab) ) /* 被黑??，需要正确的 Taito rom 编号 */
+ ROM_LOAD( "b44-12u", 0x00000, 0x10000, BAD_DUMP CRC(9f09fd5c) SHA1(ae92f2e893e1e666dcabbd793f1a778c5e3d7bab) ) /* 被黑客攻击？？，需要正确的 Taito ROM 编号 */

        ROM_REGION( 0x1000, "mcu", 0 ) /* M-Chip (i8x42 内部 ROM) */
        ROM_LOAD( "b44-8742.mcu", 0x0000, 0x0800, CRC(7dff3f9f) SHA1(bbf4e036d025fe8179b053d639f9b8ad401e6e68) ) /* B44 // 09 是标签？标签下的口罩编号是多少？也许是Taito M-011？最后一位数字肯定是1 */
@@ -2505,7 +2505,7 @@ ROM_END
 新西兰的故事
```

```
 台东，1988

-PCB 布局（“新型 PCB”，带有 3x z80，无 M 芯片，以及带有 ROM 和 z80 的子板）
+PCB 布局（“新型 PCB”，带有 3x z80，无 M 芯片，以及带有 ROM 和 z80 的子板）
 ----------
 tnzs PCB 有一个贴纸标签，上面写着“M6100409A // NZLAND STORY”

@@ -2583,7 +2583,7 @@ ROM_START( tnzs )
        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于第三个 CPU */
        ROM_LOAD（“b53-26.u34”，0x00000，0x10000，CRC（cfd5649c）SHA1（4f6afccd535d39b41661dc3ccd17af125bfac015））

- ROM_REGION( 0x100000, "gfx1", 0 ) /* 较新的 PCB 已更新 GFX rom 标签，内容相同。位于 SUB PCB 上*/
+ ROM_REGION( 0x100000, "gfx1", 0 ) /* 新的 PCB 更新了 GFX ROM 标签，内容相同。位于 SUB PCB 上 */
        ROM_LOAD( "b53-16.ic7", 0x00000, 0x20000, CRC(c3519c2a) SHA1(30fe7946fbc95ab6b3ccb6944fb24bf47bf3d743) ) /* 也标记为 U35L */
        ROM_LOAD( "b53-17.ic8", 0x20000, 0x20000, CRC(2bf199e8) SHA1(4ed73e4f00ae2f5f4028a0ea5ae3cd238863a370) ) /* 也标记为 U35U */
        ROM_LOAD( "b53-18.ic9", 0x40000, 0x20000, CRC(92f35ed9) SHA1(5fdd8d6ddbb7be9887af3c8dea9ad3b58c4e86f9) ) /* 也标记为 U39L */
@@ -2610,7 +2610,7 @@ ROM_START( tnzsj )
        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于第三个 CPU */
        ROM_LOAD（“b53-26.u34”，0x00000，0x10000，CRC（cfd5649c）SHA1（4f6afccd535d39b41661dc3ccd17af125bfac015））

- ROM_REGION( 0x100000, "gfx1", 0 ) /* 较新的 PCB 已更新 GFX rom 标签，内容相同。位于 SUB PCB 上*/
+ ROM_REGION( 0x100000, "gfx1", 0 ) /* 新的 PCB 更新了 GFX ROM 标签，内容相同。位于 SUB PCB 上 */
        ROM_LOAD( "b53-16.ic7", 0x00000, 0x20000, CRC(c3519c2a) SHA1(30fe7946fbc95ab6b3ccb6944fb24bf47bf3d743) ) /* 也标记为 U35L */
        ROM_LOAD( "b53-17.ic8", 0x20000, 0x20000, CRC(2bf199e8) SHA1(4ed73e4f00ae2f5f4028a0ea5ae3cd238863a370) ) /* 也标记为 U35U */
        ROM_LOAD( "b53-18.ic9", 0x40000, 0x20000, CRC(92f35ed9) SHA1(5fdd8d6ddbb7be9887af3c8dea9ad3b58c4e86f9) ) /* 也标记为 U39L */
@@ -2636,10 +2636,10 @@ 台东 ID: K1100356A
          主电路板
 濑田 ID：PO-041A
 */
-/* 该 pcb 源自 Chuka Taisen、DrToppel 和 Arkanoid 2 pcb，用 color ram 替换两个 color proms；
+/* 该 pcb 源自 Chuka Taisen、DrToppel 和 Arkanoid 2 pcb，用彩色 RAM 替换了两个彩色 PROM；
    有一个 M 芯片 i8x42（带 Taito 丝印），没有第三个 z80。
    没有像后来的TNZS PCB那样的子PCB。
- PCB上的GFX ROM是28针23C1000/TC531000 128K掩模ROM */
+ PCB 上的 GFX ROM 为 28 针 23C1000/TC531000 128K 掩模 ROM */

 ROM_START（tnzso）
        ROM_REGION( 0x20000, "maincpu", 0 ) /* 第一个 CPU 的 64k + Bankswitch 区域 */
@@ -2753,16 +2753,16 @@ ROM_START( tnzsoa ) // 这是一个合法的集合，还是一个 hack，或者一个接近最终的（稍后）
 ROM_END

 /* 这是原型 CA403001A PCB (Seta: PO-041-1)，与上面的 K1100356A/J1100156A (Seta: PO-041A) 'tnzsuo/tnzsjo/arkanoid2/etc' PCB 几乎但不完全相同：
-此 PCB 使用 32 针 27C1000D EPROM 来支持 8 个 GFX ROM，最终的 K1100356A/J1100156A PCB 使用 28 针 23C1000 掩模 ROM。jamma 连接器附近的一些电容器也被移动。
+此 PCB 使用 32 引脚 27c1000d EPROM 作为 8 个 gfx ROM，最终的 K1100356A/J1100156A PCB 使用 28 引脚 23c1000 掩模 ROM。jamma 连接器附近的一些电容器也被移动。
 不存在其他明显明显的路由/接线变化。
```

```
 这种类型的 PCB 可能已用于该硬件上所有游戏的内部测试。
 */
-ROM_START( tnzsop ) // 原型（位置测试？）版本；有不同的 rom 标签，Seta X1-001 芯片有原型标记，表明它是由 Yamaha 制造的，名为“YM3906”
+ROM_START( tnzsop ) // 原型（位置测试？）版本；有不同的 ROM 标签，Seta X1-001 芯片有原型标记，表明它是由 Yamaha 制造的，为“YM3906”
        ROM_REGION( 0x20000, "maincpu", 0 ) /* 第一个 CPU 的 64k + Bankswitch 区域 */
- ROM_LOAD( "c-11__6-24__1959h.d27c1000d-15.u32", 0x00000, 0x20000, CRC(3c1dae7b) SHA1(0004fccc171714c80565326f8690f9662c5b75d9) ) // 标记为 PCB 位
置、6/24 日期和校验和 - NEC D271000d eprom
+ ROM_LOAD( "c-11__6-24__1959h.d27c1000d-15.u32", 0x00000, 0x20000, CRC(3c1dae7b) SHA1(0004fccc171714c80565326f8690f9662c5b75d9) ) // 标记为 PCB 位
置、6/24 日期和校验和 - NEC D271000d EPROM

        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
- ROM_LOAD( "e-3__6-24__c4ach.tmm27512d-20.u38", 0x00000, 0x10000, CRC(c7662e96) SHA1(be28298bfde4e3867cfe75633ffb0f8611dbbd8b) ) // 标记为 PCB 位
置，日期为 6/2 4 & 校验和 - TMM27512D eprom
+ ROM_LOAD( "e-3__6-24__c4ach.tmm27512d-20.u38", 0x00000, 0x10000, CRC(c7662e96) SHA1(be28298bfde4e3867cfe75633ffb0f8611dbbd8b) ) // 标记为 PCB 位
置，日期为 6/2 4 & 校验和 - TMM27512D EPROM

        ROM_REGION( 0x10000, "mcu", 0 ) /* M-Chip (i8x42 内部 ROM) */
        ROM_LOAD( "b8042h___88-6-22__0fcc.d8742.u46", 0x0000, 0x0800, CRC(a4bfce19) SHA1(9340862d5bdc1ad4799dc92cae9bce1428b47478) ) // 日期为
'88/6/22，带校验和 -英特尔D8742微控制器
@@ -2872,7 +2872,7 @@ ROM_START( 昆虫x )
        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
        ROM_LOAD( "b97__07.u38", 0x00000, 0x10000, CRC(324b28c9) SHA1(db77a4ac60196d0f0f35dbc5c951ec29d6392463) ) /* 标签为 B97 07* 带星号 */

- ROM_REGION( 0x100000, "gfx1", 0 ) /* 掩码 rom */
+ ROM_REGION( 0x100000, "gfx1", 0 ) /* 掩码 ROM */
        ROM_LOAD（“b97__01.u1”，0x00000，0x80000，CRC（d00294b1）SHA1（f43a4f7d13193ddbbcdef71a5085c1db0fc062d4））
        ROM_LOAD（“b97__02.u2”，0x80000，0x80000，CRC（db5a7434）SHA1（71fac872b19a13a7ad25c8ad895c322ec9573fdc））
 ROM_END
@@ -2884,7 +2884,7 @@ ROM_START( 昆虫xj )
        ROM_REGION( 0x10000, "sub", 0 ) /* 64k 用于第二个 CPU */
        ROM_LOAD( "b97__04.u38", 0x00000, 0x10000, CRC(dc4549e5) SHA1(9920f7c12e047ee165418d33b3add51ea615df7e) ) /* 标签为 B97 04* 带星号 */

- ROM_REGION( 0x100000, "gfx1", 0 ) /* 掩码 rom */
+ ROM_REGION( 0x100000, "gfx1", 0 ) /* 掩码 ROM */
        ROM_LOAD（“b97__01.u1”，0x00000，0x80000，CRC（d00294b1）SHA1（f43a4f7d13193ddbbcdef71a5085c1db0fc062d4））
        ROM_LOAD（“b97__02.u2”，0x80000，0x80000，CRC（db5a7434）SHA1（71fac872b19a13a7ad25c8ad895c322ec9573fdc））
 ROM_END
@@ -2916,7 +2916,7 @@ 游戏( 1992, kagekih, kageki, kageki, kageki, kageki_state,empty_init,
 GAME( 1988, chukatai, 0, tnzs, chukatai, tnzs_state,empty_init, ROT0, "Taito Corporation Japan", "Chuka Taisen (World) (P0-028-A PCB)",
MACHINE_SUPPORTS_SAVE ) /* 可能的区域破解 */
 GAME( 1988, chukataiu, chukatai, tnzs, chukatau, tnzs_state,empty_init, ROT0, "Taito America Corporation", "Chuka Taisen (US) (P0-028-A PCB)",
MACHINE_SUPPORTS_SAVE ) /* 可能的区域破解 */
 游戏（1988，chukataij，chukatai，tnzs，chukatau，tnzs_state，empty_init，ROT0，“Taito Corporation”，“Chuka Taisen（日本）（P0-028-A PCB）”，
MACHINE_SUPPORTS_SAVE）
-GAME( 1988, chukataija,chukatai, extrmatn, chukatau, extrmatn_state,empty_init, ROT0, "Taito Corporation", "Chuka Taisen (Japan) (P0-025-A PCB)",
MACHINE_SUPPORTS_SAVE ) /* 更高的 rom ID# 但更旧PCB库存*/
+GAME( 1988, chukataija,chukatai, extrmatn, chukatau, extrmatn_state,empty_init, ROT0, "Taito Corporation", "Chuka Taisen (Japan) (P0-025-A PCB)",
```

```
MACHINE_SUPPORTS_SAVE ) /* 更高的 ROM ID# 但较旧PCB库存*/

 游戏（1988，tnzs，0，tnzsb，tnzs，tnzsb_state，empty_init，ROT0，“日本台东公司”，“新西兰故事（世界，新版本）（PO-043A PCB）”，
MACHINE_SUPPORTS_SAVE）
 游戏（1988，tnzsj，tnzs，tnzsb，tnzsj，tnzsb_state，empty_init，ROT0，“Taito Corporation”，“新西兰故事（日本，新版本）（PO-043A PCB）”，
MACHINE_SUPPORTS_SAVE）
diff --git a/src/mame/drivers/tsispch.cpp b/src/mame/drivers/tsispch.cpp
索引 468b4971d8f..c6157793ff2 100644
---a/src/mame/drivers/tsispch.cpp
+++ b/src/mame/drivers/tsispch.cpp
@@ -19,12 +19,12 @@
 *
 * 完毕：
 * 骨架书写
-* 加载 cpu 和 dsp rom 以及映射器 proms
+* 加载 cpu 和 dsp ROM 以及映射器 PROM
 * 编译成功
 * 运行成功
-* 正确交错 8086 CPU ROM
+* 正确交错 8086 CPU ROM
 * 调试与 popmessage 挂钩的 LED
-* 正确加载 UPD7720 ROM 作为 UPD7725 数据 - 完成；这是完全恶心的代码，但似乎有效。
+* 正确加载 UPD7720 ROM 作为 UPD7725 数据 - 完成；这是完全恶心的代码，但似乎有效。
 * 在 u15 处附加 i8251a UART
 * 添加拨码开关阵列 S4
 * 附8259 PIC
@@ -41,7 +41,7 @@
 * UPD7720：连接串行输出和SCK，并将SO连接到DAC；这需要修复 upd7725 核心才能真正支持 SCK 和串行输出/SO!
 * 连接另一个 i8251a uart（假设它已连接到主硬件！）
 * 正确实现UPD7720 cpu核心以避免需要令人反感的转换代码；这可能涉及覆盖和复制 7725 核心的大部分 exec_xx 部分
-* 正确的内存映射和 io 映射，并找出所有 prom 的作用 - 大部分已完成
+* 正确的内存映射和 io 映射，并找出所有 PROM 的作用 - 大部分已完成
 * 8259 PIC：找出 IR4-7 的来源（如果有的话）。
 * UPD7720 和 8259：将 p0 和 p1 连接为输出，并弄清楚 8259 IR0 如何从 7720 p0 中屏蔽。
 * 添加其他拨码开关和跳线（这些实际上可能只是控制两个 8251 的时钟分频器）
@@ -49,7 +49,7 @@
 * 其他一切
 *
 * 注意事项：
-* ROM 中的文本表示有一个测试模式“由开关 s4 dash 7 激活”
+* ROM 中的文本表示有一个测试模式“由开关 s4 dash 7 激活”
 * 当开关 s4-7 打开时，硬件一遍又一遍地说：
 *“这是版本 3.4.1 测试模式，由开关 s4 dash 7 激活”
 *
@@ -94,7 +94,7 @@
 * 检查bp是否为1，如果为1则跳转到D318F
 * 将 0x14 (0 0 0 [1 0 1 0] 0) 写入 3401
```

```
 * 调用E3987：初始化UPD7720，返回
-* D33D2：校验 5 中的 ROM? 通过，在 D33DA 循环，在 D33E6 测试（通过）
+* D33D2：5 中 ROM 的校验和？通过，在 D33DA 循环，在 D33E6 测试（通过）
 * 如果测试 DID 失败：将 0x10 (0 0 0 [1 0 0 0] 0) 写入 3401
 * 更多东西
 * 将 0xFF 写入 3401
@@ -288,7 +288,7 @@ void tsispch_state::init_prose2k()
 /********************************************************** ******************************
  地址图
 ********************************************************** ************************/
-/* Prose 2020的地址映射由2个proms控制，参见rom部分
+/*散文2020的地址映射由2个PROM控制，参见ROM部分
    有关这些的详细信息。
    （x = 忽略；* = 选择此范围内的地址；s = 选择一对芯片中的一个）
    A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
@@ -308,7 +308,7 @@ void tsispch_state::init_prose2k()
 void tsispch_state::i8086_mem(地址映射&映射)
 {
        map.unmap_value_high();
- 地图(0x00000, 0x02FFF).mirror(0x34000).ram(); // 已验证；6264*2 sram，仅使用前3/4
+ 地图(0x00000, 0x02FFF).mirror(0x34000).ram(); // 已验证；6264*2 SRAM，仅使用前3/4
        映射(0x03000, 0x03003).mirror(0x341FC).rw("i8251a_u15", FUNC(i8251_device::read), FUNC(i8251_device::write)).umask16(0x00ff);
        映射(0x03200, 0x03203).mirror(0x341FC).rw(m_pic, FUNC(pic8259_device::read), FUNC(pic8259_device::write)).umask16(0x00ff); // AMD P8259 PIC
@ U5（读为04和7c，高字节为开放总线）
        映射(0x03400, 0x03400).mirror(0x341FE).r(FUNC(tsispch_state::dsw_r)); // 已验证，从拨码开关 s4 读取
@@ -433,7 +433,7 @@ ROM_START( 散文2k )
        ROM_REGION（0x400，“DSP数据”，0）
        ROM_LOAD（“v3.12__8-9-88__dsp_data.u29”，0x0000，0x0400，CRC（f4e4dd16）SHA1（6e184747db2f26e45d0e02907105ff192e51baba））

- // 映射舞会：
+ // 映射 PROM:
        // 全部都是am27s19 32x8 TriState PROM（相当于82s123/6331）
        // L - 始终为低；H——始终高
        // U77：未知（这是做什么的？可能与多总线和等待状态有关？）
@@ -452,7 +452,7 @@ ROM_START( 散文2k )
        // 位 7-4 始终为低电平，位 2 和 1 始终为高电平。
        // SRAM 仅填充在 U61 和 U64 中。
        // 在早期转储的 Prose 2000 主板上，位 3,2,1,0 都被使用；
- // 位 7-4 始终为低电平。sram位于6 6116s中，映射与后来板上的2 6264s相同。
+ // 位 7-4 始终为低电平。SRAM 为 6 6116，映射与后续板上的 2 6264 相同。
        // 输出位 0bLLLL3210
        // 7,6,5,4 - 看起来没有连接？
        // 3 - 至 U80 处 74S138N 的 /EN3（引脚 4）
@@ -474,7 +474,7 @@ ROM_START( 散文2k )
        // /Y0 - ?
        // /Y1 - ?
        // /Y2 - ?
```

```
- // /Y3 - 连接某处，仅在 A18 和 A19 为高电平时有效，可能是 ROM 总线缓冲区启用？（TODO：弄清楚这是做什么的）
+ // /Y3 - 连接某处，仅在 A18 和 A19 为高电平时有效，可能是 ROM 总线缓冲区启用？（TODO：弄清楚这是做什么的）
        // /Y4-/Y7 - 从未使用过，因为 S2 被拉至 GND
        // 2 - 到 U63 和 U66 的 6264 SRAM 上的 /CS1
        // 1 - 到 U62 和 U65 的 6264 SRAM 上的 /CS1
@@ -482,9 +482,9 @@ ROM_START( 散文2k )
        //
        // U81：（可选）映射 ROMS：对于 I4,3,2,1,0 输入为 A19-A15
        // 在转储的 Prose 2000 板上，仅使用了位 6 和 5，
- // 其余的始终为高；将 ROM 0、1、2、3 映射到 C0000-FFFFF。
- // Prose 2000 板有用于 rom 4-15 的空的未填充插槽；
- // 如果存在，这些将由该位置的不同舞会驱动。
+ // 其余的始终为高；将 ROM 0、1、2、3 映射到 C0000-FFFFF。
+ // Prose 2000 板具有用于 ROM 4-15 的空的未填充插槽；
+ // 如果存在，这些将由该位置中的不同 PROM 驱动。
        // 位函数
        // 7 - 到 ROM 14(U28) 和 15(U51) 的 /CE
        // 6 - 到 ROM 0(U21) 和 1(U44) 的 /CE
@@ -495,8 +495,8 @@ ROM_START( 散文2k )
        // 1 - 到 ROM 10(U26) 和 11(U49) 的 /CE
        // ROM 12(U27) 和 13(U50) 的 0 - 到 /CE
        //
- // 注意U81是可选的；可以用 74s138 代替 prom，
- // 以 A19、A18、A17 作为输入，用于将 rom 解码为：
+ // 注意 U81 是可选的；它可以用 74s138 代替 PROM 代替，
+ // 以 A19、A18、A17 作为输入，用于将 ROM 解码为：
        // 7 - 到 ROM 0(U21) 和 1(U44) (0xE0000-0xE3FFF) 的 /CE
        // 6 - 到 ROM 2(U22) 和 3(U45) 的 /CE (0xE4000-0xE7FFF)
        // 5 - 到 ROM 4(U23) 和 5(U46) 的 /CE (0xE8000-0xEBFFF)
@@ -528,7 +528,7 @@ ROM_START（散文2ko）
        ROMX_LOAD( "v1.1__15__speech__plus__=c=1983.am2764.15.u51", 0xfc001, 0x2000, CRC(beb1fa19)
SHA1(72130fe45c3fd3de7cf794936dc68ed2d4193daf),ROM_SKIP(1))

        // TSI/语音加 DSP 固件 v?.?（没有贴纸，但芯片上印有 S140025），未标记芯片，但显然是 NEC UPD7720C 陶瓷
- // 尚未转储，使用 3.12 dsp 固件作为占位符，因为旧板上的 dsp 是 MASK ROM 并且不容易转储
+ // 尚未转储，使用 3.12 dsp 固件作为占位符，因为旧板上的 dsp 是掩模 ROM 并且不容易转储
        ROM_REGION( 0x600, "dspprgload", 0) // 打包 24 位数据
        ROM_LOAD（“s140025__dsp_prog.u29”，0x0000，0x0600，NO_DUMP）
        ROM_LOAD( "v3.12__8-9-88__dsp_prog.u29", 0x0000, 0x0600, CRC(9e46425a) SHA1(80a915d731f5b6863aeeb448261149ff15e5b786)) // 临时占位符
@@ -540,7 +540,7 @@ ROM_START（散文2ko）
        ROM_REGION（0x1000，“舞会”，0）
        ROM_LOAD( "dm74s288n.u77", 0x0000, 0x0020, CRC(a88757fc) SHA1(9066d6dbc009d7a126d75b8461ca464ddf134412)) // == am27s19.u77
        ROM_LOAD（“dm74s288n.whitespot.u79”，0x0020，0x0020，CRC（7faee6cb）SHA1（b6dd2a6909dac9e89e7317c006a013ff0866382d））
- // 该组中没有第三个 prom，使用 74S138 代替 e0000-fffff rom 映射
+ // 该组中没有第三个 PROM，使用 74S138 代替 e0000-fffff ROM 映射
        ROM_END
```

```
 /*************************************************************** ******************************
diff --git a/src/mame/drivers/vamphalf.cpp b/src/mame/drivers/vamphalf.cpp
索引 b1b83d1ebbf..dd1fd3dda7e 100644
---a/src/mame/drivers/vamphalf.cpp
+++ b/src/mame/drivers/vamphalf.cpp
@@ -36,8 +36,8 @@

  笔记：

- Kicker 先生：如果没有有效的默认 eeprom，则无法启动，但似乎不再失败
- 获得高分后（自 eeprom 重写后）。
+ Kicker 先生：如果没有有效的默认 EEPROM，则无法启动，但似乎不再失败
+ 获得高分后（自 EEPROM 重写后）。

  Boong-Ga Boong-Ga：测试模式可与“通用”输入配置等标准输入配置一起使用

@@ -1035,7 +1035,7 @@ 静态 INPUT_PORTS_START( 哦 )
        PORT_BIT（0x00000002，IP_ACTIVE_LOW，IPT_START2）
        PORT_BIT（0x00000004，IP_ACTIVE_LOW，IPT_UNKNOWN）
        PORT_BIT（0x00000008，IP_ACTIVE_LOW，IPT_UNKNOWN）
- PORT_BIT( 0x00000010, IP_ACTIVE_HIGH, IPT_CUSTOM ) PORT_READ_LINE_DEVICE_MEMBER("eeprom", eeprom_serial_93cxx_device, do_read) // eeprom 位
+ PORT_BIT( 0x00000010, IP_ACTIVE_HIGH, IPT_CUSTOM ) PORT_READ_LINE_DEVICE_MEMBER("eeprom", eeprom_serial_93cxx_device, do_read) // EEPROM 位
        PORT_BIT（0x00000020，IP_ACTIVE_LOW，IPT_SERVICE1）
        PORT_BIT（0x00000040，IP_ACTIVE_LOW，IPT_UNKNOWN）
        PORT_BIT（0x00000080，IP_ACTIVE_LOW，IPT_UNKNOWN）
@@ -1409,7 +1409,7 @@ 早期DANBI PCB：
 显卡：Actel A40MX04-F PL84
    声音：Oki M6295 重新包装为 U6295
          YM3012/YM2151 重新包装为 KA3002/KA51
- ROM：ROML01、ROMU01 - 用于 ELC 和 EVI 的 SOP44 32MBit MASK ROM
+ ROM：ROML01、ROMU01 - 用于 ELC 和 EVI 的 SOP44 32MBit 掩码 ROM
          ROML00、ROMU00 - 未填充
    DRAM1：LG Semi GM71C18163 1M x16 EDO DRAM (SOJ44)

@@ -1447,7 +1447,7 @@ ROM_START( vamphalfr1 )
        ROM_LOAD( "ws1-01201.rom1", 0x80000, 0x80000, CRC(afa75c19) SHA1(5dac104d1b3c026b6fce4d1f9126c048ebb557ef) ) /* 在 0x162B8: 欧洲版本
1.0.0903 */

        ROM_REGION( 0x800000, "gfx", 0 ) /* 16x16x8 精灵 */
- ROM_LOAD32_WORD( "elc.roml01", 0x000000, 0x400000, CRC(19df4056) SHA1(8b05769d8e245f8b25bf92013b98c9d7e5ab4548) ) /* 只有 2 个 rom，尽管是其他组
的两倍 */
+ ROM_LOAD32_WORD( "elc.roml01", 0x000000, 0x400000, CRC(19df4056) SHA1(8b05769d8e245f8b25bf92013b98c9d7e5ab4548) ) /* 只有 2 个 ROM，尽管是其他组
的两倍 */
        ROM_LOAD32_WORD( "evi.romu01", 0x000002, 0x400000, CRC(f9803923) SHA1(adc1d4fa2c6283bc24829f924b58fbd9d1bacdd2) )

        ROM_REGION( 0x40000, "oki1", 0 ) /* Oki 样本 */
@@ -2213,7 +2213,7 @@ Wivern Wings (c) 2001 SemiCom / Wyvern Wings (c) 2001 SemiCom，游戏视觉许可证
```

```
     CPU：  海派世通 E1-32T
   视频：2 个 QuickLogic QL12x16B-XPL84 FPGA
-  声音：AdMOS QDSP1000 和 QDSP QS1001A 示例 ROM
+  声音：AdMOS QDSP1000 和 QDSP QS1001A 示例 ROM
     振荡器：50MHz、28MHz 和 24MHz
  EEPROM：93C46

@@ -2256,8 +2256,8 @@ F-E1-32-010-D
  S1是设置按钮
  S2是复位按钮

-ROMH和ROML均为MX 29F1610MC-16闪存ROM
-u15A 是 MX 29F1610MC-16 闪存 ROM
+ROMH & ROML 均为 MX 29F1610MC-16 闪存 ROM
+u15A 是 MX 29F1610MC-16 闪存 ROM
  u7 是 ST 27c1001
  ROM 1 & ROM2 均为 ST 27C4000D

@@ -2275,7 +2275,7 @@ ROM_START( Wivernwg )
         ROM_RELOAD（0x60000，0x20000）

         ROM_REGION( 0x1000000, "gfx", 0 ) /* gfx 数据 */
- ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
         ROM_LOAD32_WORD（“romh00”，0x000002，0x200000，CRC（516aca48）SHA1（42cf5678eb4c0ee7da2ab0bd66e4e34b2735c75a））
         ROM_LOAD32_WORD（“roml01”，0x400000，0x200000，CRC（1c764f95）SHA1（ba6ac1376e837b491bc0269f2a1d10577a3d40cb））
         ROM_LOAD32_WORD（“romh01”，0x400002，0x200000，CRC（fee42c63）SHA1（a27b5cbca0defa9be85fee91dde1273f445d3372））
@@ -2285,7 +2285,7 @@ ROM_START( Wivernwg )
         ROM_LOAD32_WORD（“h03”，0xc00002，0x200000，CRC（ade8af9f）SHA1（05cdc1b38dec9d8a86302f2de794391fd3e376a5））

         ROM_REGION( 0x1000000, "qs1000", 0 ) /* 音乐数据/QDSP 样本 (SFX) */
- ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
         ROM_LOAD（“qs1001a”，0x200000，0x080000，CRC（d13c6407）SHA1（57b14f97c7d4f9b5d9745d3571a0b7115fbe3176））
  ROM_END

@@ -2301,7 +2301,7 @@ ROM_START( wyvernwg )
         ROM_RELOAD（0x60000，0x20000）

         ROM_REGION( 0x1000000, "gfx", 0 ) /* gfx 数据 */
- ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
         ROM_LOAD32_WORD（“romh00”，0x000002，0x200000，CRC（516aca48）SHA1（42cf5678eb4c0ee7da2ab0bd66e4e34b2735c75a））
         ROM_LOAD32_WORD（“roml01”，0x400000，0x200000，CRC（1c764f95）SHA1（ba6ac1376e837b491bc0269f2a1d10577a3d40cb））
         ROM_LOAD32_WORD（“romh01”，0x400002，0x200000，CRC（fee42c63）SHA1（a27b5cbca0defa9be85fee91dde1273f445d3372））
@@ -2311,7 +2311,7 @@ ROM_START( wyvernwg )
```

```
        ROM_LOAD32_WORD（“romh03”，0xc00002，0x200000，CRC（e01c2a92）SHA1（f53c2db92d62f595d473b1835c46d426f0dbe6b3））

        ROM_REGION( 0x1000000, "qs1000", 0 ) /* 音乐数据/QDSP 样本 (SFX) */
- ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
        ROM_LOAD（“qs1001a”，0x200000，0x080000，CRC（d13c6407）SHA1（57b14f97c7d4f9b5d9745d3571a0b7115fbe3176））
 ROM_END

@@ -2327,7 +2327,7 @@ ROM_START( wyvernwga )
        ROM_RELOAD（0x60000，0x20000）

        ROM_REGION( 0x1000000, "gfx", 0 ) /* gfx 数据 */
- ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD32_WORD( "roml00", 0x000000, 0x200000, CRC(fb3541b6) SHA1(4f569ac7bde92c5febf005ab73f76552421ec223) ) /* MX 29F1610MC-16 无标签闪存 */
        ROM_LOAD32_WORD（“romh00”，0x000002，0x200000，CRC（516aca48）SHA1（42cf5678eb4c0ee7da2ab0bd66e4e34b2735c75a））
        ROM_LOAD32_WORD（“roml01”，0x400000，0x200000，CRC（1c764f95）SHA1（ba6ac1376e837b491bc0269f2a1d10577a3d40cb））
        ROM_LOAD32_WORD（“romh01”，0x400002，0x200000，CRC（fee42c63）SHA1（a27b5cbca0defa9be85fee91dde1273f445d3372））
@@ -2337,7 +2337,7 @@ ROM_START( wyvernwga )
        ROM_LOAD32_WORD（“romh03”，0xc00002，0x200000，CRC（e01c2a92）SHA1（f53c2db92d62f595d473b1835c46d426f0dbe6b3））

        ROM_REGION( 0x1000000, "qs1000", 0 ) /* 音乐数据/QDSP 样本 (SFX) */
- ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
+ ROM_LOAD( "romsnd.u15a", 0x000000, 0x200000, CRC(fc89eedc) SHA1(2ce28bdb773cfa5b5660e4c0a9ef454cb658f2da) ) /* MX 29F1610MC-16 无标签闪存 */
        ROM_LOAD（“qs1001a”，0x200000，0x080000，CRC（d13c6407）SHA1（57b14f97c7d4f9b5d9745d3571a0b7115fbe3176））
 ROM_END

@@ -2469,7 +2469,7 @@ CPU - Hyperstone E1-32T @ 50.000MHz
 OSC - 50MHz、27MHz、24MHz 和 7.3728MHz（未填充）

 QDSP QS1000 @ 24MHz（丝印为 SND1）
- QS1001A 示例 rom（丝印为 SND3）
+ QS1001A 示例 ROM（丝印为 SND3）
      SND2 附加声音样本
      QS1000 的 SND5 8052 CPU 代码？

@@ -2558,7 +2558,7 @@ SEMICOM-003a
 +--------------------------------------------------------------------+

 ROM1和U7是27C040
-ROML00 和 ROMH00 是 MX 29F1610MC 闪存
+ROML00 和 ROMH00 是 MX 29F1610MC 闪存 ROM
 ROM0、ROML01 和 ROMH01 未填充
 YM2151、YM3012 和 M6295 标记为 BS901、BS902 和 U6295
 CRAM是MCM6206BAEJ15
@@ -2622,7 +2622,7 @@ SEMICOM-003b
 +--------------------------------------------------------------------+
```

```
 ROM1和U7是27C040
-ROML00 和 ROMH00 是 MX 29F1610MC 闪存
+ROML00 和 ROMH00 是 MX 29F1610MC 闪存 ROM
 ROM0、ROML01 和 ROMH01 未填充
 YM2151、YM3012 和 M6295 标记为 U6651、U6612 和 AD-65
 CRAM是MCM6206BAEJ15
@@ -3589,7 +3589,7 @@ 游戏( 2001, dtfamily, 0, mrkicker, 普通, vamphalf_state, init
 GAME（2001，finalgdr，0，finalgdr，finalgdr，vamphalf_nvram_state，init_finalgdr，ROT0，“SemiCom”，“Final Godori（韩国，版本2.20.5915）”，
MACHINE_SUPPORTS_SAVE）

 GAME( 2001, mrkicker, 0, mrkicker, common, vamphalf_state, init_mrkicker, ROT0, "SemiCom", "踢球先生 (F-E1-16-010 PCB)", MACHINE_SUPPORTS_SAVE )
-GAME( 2001, mrkickera, mrkicker, mrkickera, Finalgdr, vamphalf_nvram_state,init_mrkickera, ROT0, "SemiCom", "Mr. Kicker (SEMICOM-003b PCB)",
MACHINE_SUPPORTS_SAVE | MACHINE_NOT_WORKING ) // 如果允许 eeprom 保存，则此设置有效损坏 eeprom，然后无法启动
+GAME( 2001, mrkickera, mrkicker, mrkickera, Finalgdr, vamphalf_nvram_state,init_mrkickera, ROT0, "SemiCom", "Mr. Kicker (SEMICOM-003b PCB)",
MACHINE_SUPPORTS_SAVE | MACHINE_NOT_WORKING ) // 如果允许 EEPROM 保存，则此设置损坏 EEPROM，然后无法启动

 GAME( 2001、玩具乐园、0、coolmini、common、vamphalf_state、init_toyland、ROT0、“SemiCom”、“玩具乐园冒险”、MACHINE_SUPPORTS_SAVE )

diff --git a/src/mame/drivers/vendetta.cpp b/src/mame/drivers/vendetta.cpp
索引 5778bdbc4d6..597a56880ed 100644
---a/src/mame/drivers/vendetta.cpp
+++ b/src/mame/drivers/vendetta.cpp
@@ -76,13 +76,13 @@
   *** 转储 ROM ***
      1) ROM1 (17C) 32Pin 1Mbit UV-EPROM -> 保存“975r01”文件
      2) ROM2 ( 5F) 28Pin 512Kbit 一次性 PROM -> 保存“975f02”文件
- 3) ROM3 ( 1D) 40Pin 4Mbit MASK ROM -> 保存“975c03”文件
- 4) ROM4 ( 3K) 42Pin 8Mbit MASK ROM -> 保存“975c04”文件
- 5) ROM5 (8L) 42Pin 8Mbit MASK ROM -> 保存“975c05”文件
- 6) ROM6 (12M) 42Pin 8Mbit MASK ROM -> 保存“975c06”文件
- 7) ROM7 (16K) 42Pin 8Mbit MASK ROM -> 保存“975c07”文件
- 8) ROM8 (16I) 40Pin 4Mbit MASK ROM -> 保存“975c08”文件
- 9) ROM9 (18I) 40Pin 4Mbit MASK ROM -> 保存“975c09”文件
+ 3) ROM3 ( 1D) 40Pin 4Mbit mask ROM -> 保存“975c03”文件
+ 4) ROM4 ( 3K) 42Pin 8Mbit mask ROM -> 保存“975c04”文件
+ 5) ROM5 (8L) 42Pin 8Mbit mask ROM -> 保存“975c05”文件
+ 6) ROM6 (12M) 42Pin 8Mbit mask ROM -> 保存“975c06”文件
+ 7) ROM7 (16K) 42Pin 8Mbit mask ROM -> 保存“975c07”文件
+ 8) ROM8 (16I) 40Pin 4Mbit mask ROM -> 保存“975c08”文件
+ 9) ROM9 (18I) 40Pin 4Mbit mask ROM -> 保存“975c09”文件
                                                    呜呜呜呜呜呜呜呜呜
                                                    esckidsj.zip

@@ -112,14 +112,14 @@ WRITE8_MEMBER(vendetta_state::eeprom_w)
        /* 位 0 - VOC0 - 视频银行相关 */
        /* 位 1 - VOC1 - 视频银行相关 */
        /* 位 2 - MSCHNG - 单声道声音选择（放大器）*/
```

```
- /* 位 3 - EEPCS - Eeprom CS */
- /* 位 4 - EEPCLK - Eeprom CLK */
- /* 位 5 - EEPDI - Eeprom 数据 */
+ /* 位 3 - EEPCS - EEPROM CS */
+ /* 位 4 - EEPCLK - EEPROM 时钟 */
+ /* 位 5 - EEPDI - EEPROM 数据 */
        /* 位 6 - IRQ 使能 */
        /* 位 7 - 未使用 */


- if (data == 0xff ) /* 这是 eeprom 写入代码中的一个错误 */
+ if (data == 0xff ) /* 这是 EEPROM 写入代码中的一个错误 */
                返回;

        /* EEPROM */
@@ -142,7 +142,7 @@ WRITE8_MEMBER(vendetta_state::K052109_w)
 {
        // ************************************************ ***************************************
        // * Escape Kids 使用 052109 的镜像 Tilemap ROM 库选择器，但仅在 * 期间
- // * Tilemap MASK-ROM 测试 (0x1d80<->0x3d80, 0x1e00<->0x3e00, 0x1f00<->0x3f00) *
+ // * Tilemap 掩码 ROM 测试 (0x1d80<->0x3d80, 0x1e00<->0x3e00, 0x1f00<->0x3f00) *
        // ************************************************ ***************************************
        if ((偏移量== 0x1d80) || (偏移量== 0x1e00) || (偏移量== 0x1f00))
                m_k052109->write(空间、偏移量、数据);
@@ -250,7 +250,7 @@ void vendetta_state::esckids_map(address_map &map)
        映射（0x3fd6，0x3fd7）.rw（“k053260”，FUNC（k053260_device :: main_read），FUNC（k053260_device :: main_write））；// 声音
        映射（0x3fd8，0x3fd9）。r（m_k053246，FUNC（k053247_设备::k053246_r））；// 053246（雪碧）
        地图（0x3fda，0x3fda）.nopw（）；// 未模拟（看门狗？？？）
- 地图（0x4000，0x4fff）.m（m_videobank1，FUNC（address_map_bank_device::amap8））；// 0x2000-0x3fff，Tilemap MASK-ROM 库选择器（MASK-ROM 测试）
+ 地图(0x4000，0x4fff).m(m_videobank1，FUNC(address_map_bank_device::amap8))；// 0x2000-0x3fff，Tilemap 掩码 ROM 库选择器（掩码 ROM 测试）
        地图(0x6000，0x7fff).bankr("银行1")；// 053248 '975r01' 1M ROM（已存储）
        地图（0x8000，0xffff）.rom（）.region（“主CPU”，0x18000）；// 053248 '975r01' 1M ROM (0x18000-0x1ffff)
 }
@@ -519,7 +519,7 @@ MACHINE_CONFIG_END
 ************************************************ *************************/

 ROM_START（仇杀）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD（“081t01.17c”，0x00000，0x40000，CRC（e76267f5）SHA1（efef6c2edb4c181374661f358dad09123741b63d））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -538,12 +538,12 @@ ROM_START( 仇杀 )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
```

```
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START（复仇者）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD（“081r01.17c”，0x00000，0x40000，CRC（84796281）SHA1（e4330c6eaa17adda5b4bd3eb824388c89fb07918））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -562,12 +562,12 @@ ROM_START( 供应商 )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendettar.nv", 0x0000, 0x080, CRC(ec3f0449) SHA1(da35b98cd10bfabe9df3ede05462fabeb0e01ca9) )
 ROM_END

 ROM_START（仇杀）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD（“081z01.17c”，0x00000，0x40000，CRC（4d225a8d）SHA1（fe8f6e63d033cf04c9a287d870db244fddb81f03））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -586,13 +586,13 @@ ROM_START(仇杀)
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START（仇杀）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
- ROM_LOAD( "1.17c", 0x00000, 0x40000, CRC(1a7ceb1b) SHA1(c7454e11b7a06d10c94fe44ba6f83208bca4ced9) ) /* World 4 播放器，程序 ROM 被发现简单标记为
“1”*/
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
+ ROM_LOAD( "1.17c", 0x00000, 0x40000, CRC(1a7ceb1b) SHA1(c7454e11b7a06d10c94fe44ba6f83208bca4ced9) ) /* World 4 播放器，找到的程序 ROM 简单标记为
“1” */

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
        ROM_LOAD（“081b02”，0x000000，0x10000，CRC（4c604d9b）SHA1（22d979f5dbde7912dd927bf5538fdbfc5b82905e））
@@ -610,12 +610,12 @@ ROM_START( 复仇 )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））
```

```
- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START（仇杀队）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD（“081w01.17c”，0x00000，0x40000，CRC（cee57132）SHA1（8b6413877e127511daa76278910c2ee3247d613a））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -634,12 +634,12 @@ ROM_START( vendetta2pw )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START（仇杀队）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD( "081-eb-a01.17c", 0x00000, 0x40000, CRC(8430bb52) SHA1(54e896510fa44e76b0640b17150210fbf6b3b5bc)) // 除了中线上的 EB 标记外，标签
不清楚。底线看起来像 401，但可能是 A01

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -658,13 +658,13 @@ ROM_START( vendetta2peba )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START（仇杀双关）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
- ROM_LOAD( "1.17c", 0x00000, 0x40000, CRC(b4edde48) SHA1(bf6342cfeb0560cdf9c943f6d112fd89ee5a4f6b) ) /* World 2 播放器，程序 rom 被发现简单标记为
“1”*/
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
+ ROM_LOAD( "1.17c", 0x00000, 0x40000, CRC(b4edde48) SHA1(bf6342cfeb0560cdf9c943f6d112fd89ee5a4f6b) ) /* World 2 播放器，找到的程序 ROM 简单标记为
“1” */

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
        ROM_LOAD（“081b02”，0x000000，0x10000，CRC（4c604d9b）SHA1（22d979f5dbde7912dd927bf5538fdbfc5b82905e））
```

```
@@ -682,12 +682,12 @@ ROM_START( vendetta2pun )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD ( "081a03", 0x000000, 0x100000, CRC (14b6baea) SHA1 (fe15ee57f19f5acaad6c1642d51f390046a7468a) )

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START (仇杀2pu)
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD ( "081u01.17c", 0x00000, 0x40000, CRC (b4d9ade5) SHA1 (fbd543738cb0b68c80ff05eed7849b608de03395) )

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -706,12 +706,12 @@ ROM_START( vendetta2pu )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD ( "081a03", 0x000000, 0x100000, CRC (14b6baea) SHA1 (fe15ee57f19f5acaad6c1642d51f390046a7468a) )

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START (仇杀2pd)
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD ( "081d01.17c", 0x00000, 0x40000, CRC (335da495) SHA1 (ea74680eb898aeecf9f1eec95f151bcf66e6b6cb) )

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -730,12 +730,12 @@ ROM_START( vendetta2pd )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD ( "081a03", 0x000000, 0x100000, CRC (14b6baea) SHA1 (fe15ee57f19f5acaad6c1642d51f390046a7468a) )

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendetta.nv", 0x0000, 0x080, CRC(fbac4e30) SHA1(d3ff3a392550d9b06400b9292a44bdac7ba5c801) )
 ROM_END

 ROM_START (文德坦)
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD ( "081n01.17c", 0x00000, 0x40000, CRC (fc766fab) SHA1 (a22c82810f2a2b66fc112e2d043e8025d0dc2841) )

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -754,12 +754,12 @@ ROM_START( 文德坦 )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
```

```
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendettaj.nv", 0x0000, 0x080, CRC(3550a54e) SHA1(370cd40a12c471b3b6690ecbdde9c7979bc2a652) )
 ROM_END

 ROM_START（仇杀2pp）
- ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 rom + 存储 ram */
+ ROM_REGION( 0x40000, "maincpu", 0 ) /* 代码 + 存储 ROM + 存储 RAM */
        ROM_LOAD（“081p01.17c”，0x00000，0x40000，CRC（5fe30242）SHA1（2ea98e66637fa2ad60044b1a2b0dd158a82403a2））

        ROM_REGION( 0x10000, "audiocpu", 0 ) /* 64k 用于声音 CPU */
@@ -778,7 +778,7 @@ ROM_START( vendetta2pp )
        ROM_REGION( 0x100000, "k053260", 0 ) /* 053260 个样本 */
        ROM_LOAD（“081a03”，0x000000，0x100000，CRC（14b6baea）SHA1（fe15ee57f19f5acaad6c1642d51f390046a7468a））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "vendettaj.nv", 0x0000, 0x080, CRC(3550a54e) SHA1(370cd40a12c471b3b6690ecbdde9c7979bc2a652) )
 ROM_END

@@ -790,20 +790,20 @@ ROM_START( esckids )
        ROM_REGION( 0x010000, "audiocpu", 0 ) // 声音 CPU (Z80) 代码 (512K x 1)
        ROM_LOAD( "975f02", 0x000000, 0x010000, CRC(994fb229) SHA1(bf194ae91240225b8edb647b1a62cd83abfa215e) )

- ROM_REGION( 0x100000, "k052109", 0 ) // Tilemap MASK-ROM (4M x 2)
+ ROM_REGION( 0x100000, "k052109", 0 ) // Tilemap 掩码 ROM (4M x 2)
        ROM_LOAD32_WORD（“975c09”，0x000000，0x080000，CRC（bc52210e）SHA1（301a3892d250495c2e849d67fea5f01fb0196bed））
        ROM_LOAD32_WORD（“975c08”，0x000002，0x080000，CRC（fcff9256）SHA1（b60d29f4d04f074120d4bb7f2a71b9e9bf252d33））

- ROM_REGION( 0x400000, "gfx2", 0 ) // Sprite MASK-ROM (8M x 4)
+ ROM_REGION( 0x400000, "gfx2", 0 ) // 精灵掩码 ROM (8M x 4)
        ROM_LOAD64_WORD（“975c04”，0x000000，0x100000，CRC（15688a6f）SHA1（a445237a11e5f98f0f9b2573a7ef0583366a137e））
        ROM_LOAD64_WORD（“975c05”，0x000002，0x100000，CRC（1ff33bb7）SHA1（eb17da33ba2769ea02f91fece27de2e61705e75a））
        ROM_LOAD64_WORD（“975c06”，0x000004，0x100000，CRC（36d410f9）SHA1（2b1fd93c11839480aa05a8bf27feef7591704f3d））
        ROM_LOAD64_WORD（“975c07”，0x000006，0x100000，CRC（97ec541e）SHA1（d1aa186b17cfe6e505f5b305703319299fa54518））

- ROM_REGION( 0x100000, "k053260", 0 ) // 样本 MASK-ROM (4M x 1)
+ ROM_REGION( 0x100000, "k053260", 0 ) // 样本掩码 ROM (4M x 1)
        ROM_LOAD（“975c03”，0x000000，0x080000，CRC（dc4a1707）SHA1（f252d08483fd664f8fc03bf8f174efd452b4cdc5））

- ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
+ ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
        ROM_LOAD( "esckids.nv", 0x0000, 0x080, CRC(a8522e1f) SHA1(43f82fce3c3b854bc8898c63dffc7c01b288c8aa) )
 ROM_END
```

```
 @@ -815,20 +815,20 @@ ROM_START( esckidsj )
         ROM_REGION( 0x010000, "audiocpu", 0 ) // 声音 CPU (Z80) 代码 (512K x 1)
         ROM_LOAD( "975f02", 0x000000, 0x010000, CRC(994fb229) SHA1(bf194ae91240225b8edb647b1a62cd83abfa215e) )

 - ROM_REGION( 0x100000, "k052109", 0 ) // Tilemap MASK-ROM (4M x 2)
 + ROM_REGION( 0x100000, "k052109", 0 ) // Tilemap 掩码 ROM (4M x 2)
         ROM_LOAD32_WORD ( "975c09" , 0x000000, 0x080000, CRC (bc52210e) SHA1 (301a3892d250495c2e849d67fea5f01fb0196bed) )
         ROM_LOAD32_WORD ( "975c08" , 0x000002, 0x080000, CRC (fcff9256) SHA1 (b60d29f4d04f074120d4bb7f2a71b9e9bf252d33) )

 - ROM_REGION( 0x400000, "gfx2", 0 ) // Sprite MASK-ROM (8M x 4)
 + ROM_REGION( 0x400000, "gfx2", 0 ) // 精灵掩码 ROM (8M x 4)
         ROM_LOAD64_WORD ( "975c04" , 0x000000, 0x100000, CRC (15688a6f) SHA1 (a445237a11e5f98f0f9b2573a7ef0583366a137e) )
         ROM_LOAD64_WORD ( "975c05" , 0x000002, 0x100000, CRC (1ff33bb7) SHA1 (eb17da33ba2769ea02f91fece27de2e61705e75a) )
         ROM_LOAD64_WORD ( "975c06" , 0x000004, 0x100000, CRC (36d410f9) SHA1 (2b1fd93c11839480aa05a8bf27feef7591704f3d) )
         ROM_LOAD64_WORD ( "975c07" , 0x000006, 0x100000, CRC (97ec541e) SHA1 (d1aa186b17cfe6e505f5b305703319299fa54518) )

 - ROM_REGION( 0x100000, "k053260", 0 ) // 样本 MASK-ROM (4M x 1)
 + ROM_REGION( 0x100000, "k053260", 0 ) // 样本掩码 ROM (4M x 1)
         ROM_LOAD ( "975c03" , 0x000000, 0x080000, CRC (dc4a1707) SHA1 (f252d08483fd664f8fc03bf8f174efd452b4cdc5) )

 - ROM_REGION( 0x80, "eeprom", 0 ) // 默认 eeprom 以防止游戏启动颠倒并出现错误
 + ROM_REGION( 0x80, "eeprom", 0 ) // 默认 EEPROM，以防止游戏启动颠倒并出错
         ROM_LOAD( "esckidsj.nv", 0x0000, 0x080, CRC(985e2a2d) SHA1(afd9e5fc014d593d0a384326f32caf2a73fba867) )
  ROM_END

 @@ -843,10 +843,10 @@ ROM_END
  游戏 ( 1991, 仇杀, 0, 仇杀, vendet4p, vendetta_state, empty_init, ROT0, "科乐美", "仇杀 (世界, 4 名玩家, 版本 T)", MACHINE_SUPPORTS_SAVE )
  游戏( 1991, vendettar, vendetta, vendetta, vendet4p, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (US, 4 Players, ver. R)",
 MACHINE_SUPPORTS_SAVE )
  游戏( 1991, 仇杀队, 仇杀队, 仇杀队, vendet4p, vendetta_state,empty_init, ROT0, "Konami", "仇杀队 (亚洲, 4 名玩家, ver. Z)", MACHINE_SUPPORTS_SAVE
 )
 -GAME( 1991, vendettaun, vendetta, vendetta, vendet4p, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (World, 4 Players, ver. ?)",
 MACHINE_SUPPORTS_SAVE ) /* 程序 ROM 标记为 1 */
 +GAME( 1991, vendettaun, vendetta, vendetta, vendet4p, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (World, 4 Players, ver. ?)",
 MACHINE_SUPPORTS_SAVE ) /* 程序 ROM 标记为 1 */
  游戏( 1991, vendetta2pw, vendetta, vendetta, vendetta, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (世界, 2 名玩家, ver. W)",
 MACHINE_SUPPORTS_SAVE )
  游戏 (1991, vendetta2peba, vendetta, vendetta, vendetta, vendetta_state, empty_init, ROT0, "Konami", "Vendetta (世界, 2名玩家, 版本EB-A? )",
 MACHINE_SUPPORTS_SAVE)
 -GAME( 1991, vendetta2pun,vendetta, vendetta, vendetta, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (World, 2 Players, ver. ?)",
 MACHINE_SUPPORTS_SAVE ) /* 程序 ROM 标记为 1 */
 +GAME( 1991, vendetta2pun,vendetta, vendetta, vendetta, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (World, 2 Players, ver. ?)",
 MACHINE_SUPPORTS_SAVE ) /* 程序 ROM 标记为 1 */
  游戏( 1991, vendetta2pu, vendetta, vendetta, vendetta, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (亚洲, 2 名玩家, ver. U)",
 MACHINE_SUPPORTS_SAVE )
  游戏( 1991, vendetta2pd, vendetta, vendetta, vendetta, vendetta_state,empty_init, ROT0, "Konami", "Vendetta (亚洲, 2 名玩家, ver. D)",
 MACHINE_SUPPORTS_SAVE )
```

```
 游戏( 1991, vendettan, vendetta, vendetta, vendet4p, vendetta_state,empty_init, ROT0, "Konami", "犯罪斗士 2 (日本, 4 名玩家, ver. N)",
MACHINE_SUPPORTS_SAVE )
diff --git a/src/mame/drivers/vt100.cpp b/src/mame/drivers/vt100.cpp
索引 5d10f0d04f5..d3f9a2b9e1f 100644
---a/src/mame/drivers/vt100.cpp
+++ b/src/mame/drivers/vt100.cpp
@@ -7,8 +7,8 @@
         29/04/2009 初级驾驶员。

         TODO：某些视频属性尚未完全支持
- TODO：支持 AVO 字符集 ROM
- TODO：完成对 CPU 板上备用字符集 ROM 的支持
+ TODO：支持 AVO 字符集 ROM
+ TODO：完成对 CPU 板上备用字符集 ROM 的支持
         TODO：VT1XX-AC 和 VT125 的 STP（标准终端端口）总线

         从 VT125 技术手册中可以获得大量有用的信息：
@@ -456,57 +456,57 @@ MACHINE_CONFIG_START(vt100_state::vt102)
 MACHINE_CONFIG_END

 /* VT1xx 型号：
- * VT100 - 1978 年基本型号。“后来的”ROM 是 1979 年或 1980 年的。
+ * VT100 - 1978 年基本型号。“后来的”ROM 是 1979 年或 1980 年的。
   * vt100 有一系列 -XX 型号的分支；这
         这里描述了我所知道的，以及任何特别的内容
         他们：
   * VT100-AA - 标准型号带120VAC电缆\__电压可切换
   * VT100-AB - 标准型号，带 240vac 电缆/任何 VT100 装置内
   * VT100-W* - 文字处理系列：
- * VT100-WA/WB - 预装了特殊的 LA120 AVO 板，WP romset？，
- 英文 WP 键盘，没有替代字符集 ROM，LA120？23-069E2 AVO ROM。
+ * VT100-WA/WB - 预装了特殊的 LA120 AVO 板、WP ROMset？、
+ 英文 WP 键盘，无替代字符集 ROM，LA120？23-069E2 AVO ROM。
         （WA 和 WB 变体在原理图中称为“-02”变体）
   * VT100-WC至WZ：外文文字处理系列：
         （WC 到 WK 变体在示意图上称为“-03”变体）
- * VT100-WC/WD - 预装了 AVO 板、WP romset？、法语加拿大语
- WP 键盘，具有 23-094E2 alt 字符集 rom、23-093E2 AVO rom。
- * VT100-WE/WF - 预装了 AVO 板，WP romset？，法语
- WP 键盘，具有 23-094E2 alt 字符集 rom、23-093E2 AVO rom。
- * VT100-WG/WH - 预装了 AVO 板、WP romset？，荷兰语
- WP 键盘，具有 23-094E2 alt 字符集 rom、23-093E2 AVO rom。
- * VT100-WJ/WK - 预装了 AVO 板，WP romset？，德语
- WP 键盘，具有 23-094E2 alt 字符集 rom、23-093E2 AVO rom。
- * VT100-WY/WZ - 预装了 AVO 板、WP romset？，英语
- WP 键盘，具有 23-094E2 alt 字符集 rom、23-093E2 AVO rom。
- WP romset 支持英语、法语、荷兰语和德语，但
```

```
+ * VT100-WC/WD - 预装了 AVO 板、WP ROMset? 、加拿大法语
+ WP 键盘，具有 23-094E2 alt 字符集 ROM、23-093E2 AVO ROM。
+ * VT100-WE/WF - 预装了 AVO 板，WP ROMset? ，法语
+ WP 键盘，具有 23-094E2 alt 字符集 ROM、23-093E2 AVO ROM。
+ * VT100-WG/WH - 预装了 AVO 板、WP ROMset? ，荷兰语
+ WP 键盘，具有 23-094E2 alt 字符集 ROM、23-093E2 AVO ROM。
+ * VT100-WJ/WK - 预装了 AVO 板、WP ROMset? 、德语
+ WP 键盘，具有 23-094E2 alt 字符集 ROM、23-093E2 AVO ROM。
+ * VT100-WY/WZ - 预装了 AVO 板、WP ROMset? ，英语
+ WP 键盘，具有 23-094E2 alt 字符集 ROM、23-093E2 AVO ROM。
+ WP ROMset 支持英语、法语、荷兰语和德语，但
         只有在以下情况下才会正确显示非英语语言的文本：
- 23-094E2 alt 字符集 ROM 和外语 23-093E2
- AVO ROM 已填充。
- * VT100-NA/NB - ? 带 DECFORM 键帽的 romset
+ 23-094E2 alt 字符集 ROM 和外语 23-093E2
+ AVO ROM 已填充。
+ * VT100-NA/NB - ? 带 DECFORM 键帽的 ROMset
  * 带有 vt1xx-ac 套件的 VT100 - 添加串行打印机接口 (STP)
- PCB，用 095e2/096e2/139e2/140e2 STP 套件替换 ROM
+ PCB，用 095e2/096e2/139e2/140e2 STP 套件替换 ROM
  * VT101 - 1981 成本降低不可扩展的 VT100；和股票一样
    未展开的 vt100。它没有 AVO 或升级连接器，也没有
    视频输入端口。）有自己的固件。
    与vt102和vt131共享相同的PCB，但STP/AVO未填充；
  * VT102 - 1981 年成本降低的不可扩展 vt100，内置 AVO 和 STP
    与安装了 AVO 和 STP 扩展的库存 vt100 相同，
- 但全部在一块 PCB 上。不支持 AVO 扩展字符 ROM，也不支持
- 文字处理ROM集。有自己的固件。
+ 但全部在一块 PCB 上。不支持 AVO 扩展字符 ROM，也不支持
+ 字处理ROM集。有自己的固件。
    与 vt101 和 vt131 共享相同的 PCB，已填充 STP 和 AVO。
  * VT103 - 1980 基本型号 VT100，带有集成 TU58 磁带机和
- LSI-11背板，其中使用LSI-11 cpu卡，因此是计算机
- 实际上是一个内置在 vt100 机箱中的微型 lsi-11 (pdp-11)。使用相同的ROM
- vt100部分为vt100，tu58有自己的cpu和rom。它可以
- 有正常的 vt100 romset 变体，也可以有多个单词
- 处理变化（使用与 vt100 相同的 ROM）。
+ LSI-11 背板，其中使用 LSI-11 CPU 卡，因此计算机
+ 实际上是一个内置于 vt100 机箱中的微型 lsi-11 (pdp-11)。使用相同的 ROM
+ 作为 vt100 的 vt100 部分，而 tu58 有自己的 CPU 和 ROM。它可以
+ 有正常的 vt100 ROMset 变体，也可以有多个字
+ 处理变化（使用与 vt100 相同的 ROM）。
  * VT104 不存在。
  * VT105 - 1978 VT100 安装了 WG 波形发生器板
    （对于简单的图表类型线比较制作的光栅图形，使用一些内置的
    在函数中），AVO 可选；旨在用于 MINC 模拟数据
```

```
     采集计算机。
   * VT110 - 1978 vt100 安装了 DPM01 DECDataway 串行多路复用器
 - DPM01 据说有自己的处理器和 ROM。
 + DPM01 据说有自己的处理器和 ROM。
   * vt125 - 1982 年？基本型号（库存 vt100 固件加上额外的 gfx 板
     固件和处理器）带有 ReGIS 图形语言板的 vt100
     （又名 GPO）已安装（几乎字面上是 vk100-on-a-board，但添加了
 @@ -517,87 +517,87 @@ MACHINE_CONFIG_END
     作为并口接口板，支持串行块模式。
     与 vt101 和 vt102 共享相同的 PCB，已填充 STP 和 AVO。
   * vt132 - 1980？带有 AVO、STP 的基础 vt100 及其自己的 23-099e2/23-100e2
 - AVO 角色 ROM 集。有自己的基础固件ROM，支持块
 + AVO 字符 ROM 集。拥有自己的基础固件 ROM，支持块
     串行模式。
   * vt180 - 1980 vt10x（带 vt100 扩展背板），带 z80 子板
     安装；
 - 子板上有两个 ROM：23-017e3-00 和 23-021e3-00
 -（两者都是 0x1000 长，2332 个掩码 ROM）
 + 子板上有两个 ROM：23-017e3-00 和 23-021e3-00
 +（两者都是 0x1000 长，2332 个掩模 ROM）
   * vk100 'gigi'-图形终端；vt125 GPO 板是非常接近的衍生产品；
     到目前为止，信息相对较少，但已经取得了进展。
     有关当前驱动程序的信息，请参阅 vk100.c

   * vt1xx 升级套件：
   * VT1xx-AA：部件号 5413206 20ma 电流环路接口 PCB，适用于 VT100
 - * VT1xx-AB：部件号 5413097 AVO 板（AVO ROM 可以选择性地与
 + * VT1xx-AB：部件号 5413097 AVO 板（AVO ROM 可以选择性地与
                如果需要的话，这个板）
 - * VT1xx-AC：STP 串行打印机板（包括特殊的 romset）
 + * VT1xx-AC : STP 串行打印板（包括特殊的 ROMset）
   * VT1xx-CA：部件号 5413206？适用于 vt101/vt102/vt131 的 20ma 电流环接口 PCB
   * VT1xx-CB 或 CL：GPO“ReGIS”板 vt100->vt125 升级套件（部件号 5414275 桨板和 5414277 gpo 板）
   * VT1xx-CE：DECWord 转换套件
   * VT1xx-FB：防眩光套件

 - * 有关面具 ROM 和其他令人讨厌的内容的信息：
 - * 普通 2716 ROM 的引脚 18：/CE；引脚 20：/OE；引脚 21：VPP（充当 CE2）
 - * vt100 23-031e2/23-061e2、23-032e2、23-033e2 和 23-034e2 掩码 ROM
 + * 关于掩模 ROM 和其他讨厌的东西的信息：
 + * 普通 2716 ROM 的引脚 18：/CE；引脚 20：/OE；引脚 21：VPP（充当 CE2）
 + * vt100 23-031e2/23-061e2、23-032e2、23-033e2 和 23-034e2 掩模 ROM
     具有以下功能：
         23-031e2/23-061e2：引脚 18：CS2；引脚 20：CS1；引脚 21：CS3
         23-032e2：引脚 18：/CS2；引脚 20：CS1；引脚 21：CS3
         23-033e2：引脚 18：CS2；引脚 20：CS1；引脚 21：/CS3
         23-034e2：引脚 18：/CS2；引脚 20：CS1；引脚 21：/CS3
```

```
-（这很可爱，因为从技术上讲，这意味着 ROM 可以放入
+（这很可爱，因为从技术上讲，它意味着 ROM 可以放入
        任何顺序的 4 个插座仍然可以正常工作，因为 cs2 和
        cs3 引脚使它们能够自解码并在正确的地址激活）
         （几乎可以肯定，同样的可爱技巧也是用
- 23-180e2、181e2、182e2 183e2 romset，以及
+ 23-180e2、181e2、182e2 183e2 ROMset，以及
        23-095e2,096e2,139e2,140e2 设置，可能还有其他）
- * 位于 e4 位置的 vt100/101/102/103/etc 23-018e2-00 字符集 ROM 是一个 24 引脚 2316 掩码 ROM，其启用如下：引脚 18：CS2；引脚 20：/CS1；引脚
21：/CS3
- * 位于 e9 位置的可选 23-094e2-00 备用字符集 ROM 是一个 24 引脚 2316 掩码 ROM，其启用如下：引脚 18：/CS2；引脚 20：/CS1；引脚 21：/CS3
- 据说 23-094e2 rom 适用于 vt100-WC 或 -WF 系统（分别是加拿大法语和法语），这意味着它具有欧洲语言特定的重音字符。它可能用于所有 -W* 系统。
- 通过移除跳线 w4 并插入跳线 w5，可以在位置 e9 将该插座的引脚 21 跳接到 +5v，从而允许使用普通的 2716 eprom。
- * 可选的 AVO 字符集 ROM（见下文）具有：引脚 18：/CS2*；引脚 20：/CS1；引脚 21：CS3，因此它们与普通 2716 匹配
+ * 位于 e4 位置的 vt100/101/102/103/etc 23-018e2-00 字符集 ROM 是一个 24 引脚 2316 掩码 ROM，其使能如下：引脚 18：CS2；引脚 20：/CS1；引脚
21：/CS3
+ * 位置 e9 处的可选 23-094e2-00 备用字符集 ROM 是一个 24 引脚 2316 掩码 ROM，其使能如下：引脚 18：/CS2；引脚 20：/CS1；引脚 21：/CS3
+ 据说 23-094e2 ROM 适用于 vt100-WC 或 -WF 系统（分别是加拿大法语和法语），这意味着它具有欧洲语言特定的重音字符。它可能用于所有 -W* 系统。
+ 通过移除跳线 w4 并插入跳线 w5，可以将该插座的引脚 21 在位置 e9 跳接到 +5v，从而允许使用普通的 2716 EPROM。
+ * 可选的 AVO 字符集 ROM（见下文）具有： 引脚 18：/CS2*；引脚 20：/CS1；引脚 21：CS3，因此它们与普通 2716 匹配
    *（这在图像上标记为 CS2，但输入与 gnd 绑定，意味着它必须是 /CS2）

- * AVO 本身最多可容纳四个 ROM（请参阅 http://www.bitsavers.org/pdf/dec/terminal/vt100/MP00633_VT100_Mar80.pdf
+ * AVO 本身最多可以容纳四个 ROM（请参阅 http://www.bitsavers.org/pdf/dec/terminal/vt100/MP00633_VT100_Mar80.pdf
    和http://vt100.net/dec/ek-vt1ac-ug-002.pdf）
- 这些 ROM 可以根据跳线映射到 0x8000，或者将主代码 ROM 覆盖在 0x0000-0x1fff!
- 他们甚至可能允许在主代码 ROM 和覆盖 ROM 之间进行存储，我还没有追踪原理图。
- 至少制作了 16 个 AVO ROM，并按如下方式使用：
+ 并且这些 ROM 可以根据跳线映射到 0x8000，或者将主代码 ROM 覆盖在 0x0000-0x1fff!
+ 它们甚至可能允许主代码 ROM 和覆盖 ROM 之间的存储，我还没有追踪原理图。
+ 至少制作了 16 个 AVO ROM，并按如下方式使用：
    （基于 EK-VT100-TM-003_VT100_Technical_Manual_Jul82.pdf）
- * 无 ROM - 安装了 AVO 的普通 vt100 系统
- * 23-069E2（位置 e21） - 用于 vt100-wa 和 -wb ’LA120’ ’字处理’ 系统（该系统的 ROM 映射与以下系统不同）
- * 23-099E2（位置 e21）和 23-100E2（位置 e17）- 适用于 vt132，但仅适用于旧 vt132 主 romset 095,096,097,098E2
+ * 无 ROM - 安装了 AVO 的普通 vt100 系统
+ * 23-069E2（位置 e21） - 用于 vt100-wa 和 -wb ’LA120’ ’字处理’系统（该系统的 ROM 映射与以下系统不同）
+ * 23-099E2（位置 e21）和 23-100E2（位置 e17）- 适用于 vt132，但仅适用于旧 vt132 主 ROM 集 095,096,097,098E2
  * 23-093E2（位置 e21）- 适用于 vt100 wc 至 wz“外语”字处理系统
  * 23-184E2 和 23-185E2 - 适用于安装了 STP 打印机选件板的 vt100，版本 1，附带 vt1xx-ac 套件
  * 23-186E2 和 23-187E2 - 适用于安装了 STP 打印机选件板的 vt100，版本 2，附带 vt1xx-ac 套件
- * 23-224E2、23-225E2、23-226E2、23-227E2 - 适用于 vt132，但仅适用于新的 vt132 主 romset 180,181,182,183E2
- * 23-236E2、23-237E2、23-238E2、23-239E2 - 适用于 vt132，但仅适用于新的 vt132 主 romset 180,181,182,183E2，与上面的未知差异（PROM VS MASK ROM？内
容相同？）
+ * 23-224E2、23-225E2、23-226E2、23-227E2 - 适用于 vt132，但仅适用于新的 vt132 主 ROMset 180,181,182,183E2
+ * 23-236E2、23-237E2、23-238E2、23-239E2 - 适用于 vt132，但仅适用于新的 vt132 主 ROM 集 180,181,182,183E2，与上面的未知差异（PROM 与掩模 ROM？内
容相同？）
```

```
   */

  /* ROM定义 */
  ROM_START( vt100 ) // 这是来自 http://www.bitsavers.org/pdf/dec/terminal/vt100/MP00633_VT100_Mar80.pdf 的原理图
 -// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来是 EPROM 0 的版本）
 -// 此 romset 也用于 vt103、vt105、vt110、vt125 和 vt180
 +// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来版本为 EPROM 0）
 +// 此 ROMset 也用于 vt103、vt105、vt110、vt125 和 vt180
         ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
         ROM_DEFAULT_BIOS(“vt100”)
 - ROM_SYSTEM_BIOS（0, “vt100o”, “VT100旧版ROM”）
 - ROMX_LOAD( "23-031e2-00.e56", 0x0000, 0x0800, NO_DUMP, ROM_BIOS(0)) // 版本 1 1978 '早期 rom', 需要转储，适用于早期 vt100
 - ROM_SYSTEM_BIOS（1, “vt100”, “VT100较新的ROM”）
 - ROMX_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15), ROM_BIOS(1)) // 版本 2 1979 或 1980'
以后rom', 适用于以后的 vt100
 + ROM_SYSTEM_BIOS( 0, "vt100o", "VT100 较旧的 ROM" )
 + ROMX_LOAD( "23-031e2-00.e56", 0x0000, 0x0800, NO_DUMP, ROM_BIOS(0)) // 版本 1 1978 '早期 ROM', 需要转储，适用于早期 vt100
 + ROM_SYSTEM_BIOS( 1, "vt100", "VT100 较新的 ROM" )
 + ROMX_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15), ROM_BIOS(1)) // 版本 2 1979 或 1980'
以后ROM', 适用于以后的 vt100
         ROM_LOAD（“23-032e2-00.e52”, 0x0800, 0x0800, CRC（3d86db99）SHA1（cdd8bdecdc643442f6e7d2c83cf002baf8101867））
         ROM_LOAD（“23-033e2-00.e45”, 0x1000, 0x0800, CRC（384dac0a）SHA1（22aaf5ab5f9555a61ec43f91d4dea3029f613e64））
         ROM_LOAD（“23-034e2-00.e40”, 0x1800, 0x0800, CRC（4643184d）SHA1（27e6c19d9932bf13fdb70305ef4d806e90d60833））

         ROM_REGION(0x1000, "充电", 0)
         ROM_LOAD（“23-018e2-00.e4”, 0x0000, 0x0800, CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
 - ROM_LOAD_OPTIONAL("23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选？字处理？替代字符集 ROM
 + ROM_LOAD_OPTIONAL("23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选？字处理？备用字符集 ROM
  ROM_END

  #如果0
  ROM_START( vt100wp ) // 这是来自 http://www.bitsavers.org/pdf/dec/terminal/vt100/MP00633_VT100_Mar80.pdf 的原理图
 -// 这是标准的 vt100 cpu 板，带有“字处理”功能？romset，包含在 VT1xx-CE 套件中吗？
 -// vt103也可以使用这个ROM集（-04和-05 revs默认有它，-05 rev默认也有可选的alt charset rom）
 -// 注意：这实际上与较新的 VT132 romset 相同；vt132 也有不同的 AVO ROM。
 +// 这是标准 vt100 CPU 板，带有“字处理”功能？ROMset，包含在 VT1xx-CE 套件中吗？
 +// vt103也可以使用这个ROM集（-04和-05 revs默认有它，-05 rev默认也有可选的alt字符集ROM）
 +// 注意：这实际上与较新的 VT132 ROMset 相同；vt132 也有不同的 AVO ROM。
         ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
         ROM_LOAD（“23-180e2-00.e56”, 0x0000, 0x0800, NO_DUMP）
         ROM_LOAD（“23-181e2-00.e52”, 0x0800, 0x0800, NO_DUMP）
 @@ -612,37 +612,37 @@ ROM_START( vt100wp ) // 这来自 http://www.bitsavers.org/ 上的原理图

         ROM_REGION(0x1000, "充电", 0)
         ROM_LOAD（“23-018e2-00.e4”, 0x0000, 0x0800, CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
 - ROM_LOAD( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 需要外语备用字符集 rom
 + ROM_LOAD( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 需要外语备用字符集 ROM
```

```
 ROM_END

 ROM_START( vt132 ) // 这是来自轶事证据和 vt100.net，因为 vt132 原理图没有被扫描
 // 但在 http://bitsavers.trailing-edge.com/www.computer.museum.uq.edu.au/pdf/EK-VT100-TM-003%20VT100%20Series%20Video 中的第 433 页几乎得到了证
实%20终端%20技术%20手册.pdf
-// 带块串行 ROM 的 VT100 板、带特殊 ROM 的 AVO、STP、带块串行模式的自定义固件
+// 带块串行 ROM 的 VT100 板、带特殊 ROM 的 AVO、STP、带块串行模式的自定义固件
 // ROM 上有设置页面 C
- // 旧版 vt132 romset
+ // 旧版 vt132 ROMset
        ROM_LOAD（“23-095e2-00.e56”，0x0000，0x0800，NO_DUMP）
        ROM_LOAD（“23-096e2-00.e52”，0x0800，0x0800，NO_DUMP）
        ROM_LOAD（“23-097e2-00.e45”，0x1000，0x0800，NO_DUMP）
        ROM_LOAD（“23-098e2-00.e40”，0x1800，0x0800，NO_DUMP）

- // 较新的 vt132（和 STP？）romset
+ // 较新的 vt132（和 STP？）ROMset
        ROM_LOAD（“23-180e2-00.e56”，0x0000，0x0800，NO_DUMP）
        ROM_LOAD（“23-181e2-00.e52”，0x0800，0x0800，NO_DUMP）
        ROM_LOAD（“23-182e2-00.e45”，0x1000，0x0800，NO_DUMP）
        ROM_LOAD（“23-183e2-00.e40”，0x1800，0x0800，NO_DUMP）

- // AVO rom 仅适用于旧版 romset
+ // AVO ROM 仅适用于旧版 ROMset
        ROM_REGION(0x1000,“avo”,0)
        ROM_LOAD（“23-099e2-00.e21”，0x0000，0x0800，NO_DUMP）
        ROM_LOAD（“23-100e2-00.e17”，0x0800，0x0800，NO_DUMP）
        // 另外2个socket是空的

- // AVO rom 仅适用于较新的 romset
+ // AVO ROM 仅适用于较新的 ROMset
        ROM_LOAD（“23-224e2-00.e21”，0x0000，0x0800，NO_DUMP）
        ROM_LOAD（“23-225e2-00.e17”，0x0800，0x0800，NO_DUMP）
        ROM_LOAD( "23-226e2-00.e15", 0x1000, 0x0800, NO_DUMP) // loc 是一个猜测
        ROM_LOAD( "23-227e2-00.e13", 0x1800, 0x0800, NO_DUMP) // loc 是一个猜测
- // 较新的 avo ROM 的替代版本，技术手册暗示上面是 PROMS 下面是 MASK ROMS？相同的数据？
+ // 较新 avo ROM 的替代版本，技术手册暗示上面是 PROM，下面是掩码 ROM？相同的数据？
        ROM_LOAD（“23-236e2-00.e21”，0x0000，0x0800，NO_DUMP）
        ROM_LOAD（“23-237e2-00.e17”，0x0800，0x0800，NO_DUMP）
        ROM_LOAD( "23-238e2-00.e15", 0x1000, 0x0800, NO_DUMP) // loc 是一个猜测
@@ -650,16 +650,16 @@ ROM_START( vt132 ) // 这来自轶事证据和 vt100.net，如 vt13

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
 ROM_END
```

```
 #万一

 ROM_START( vt100ac ) // 这是来自 VT180 技术手册：http://www.bitsavers.org/pdf/dec/terminal/vt180/EK-VT18X-TM-001_VT180_Technical_Man_Feb83.pdf
-// 这是标准的 vt100 CPU 板，但带有 VT1xx-AC 套件中包含的 ROM 集
+// 这是标准 vt100 CPU 板，但带有 VT1xx-AC 套件中包含的 ROM 集
 // 仅当部件 54-14260-00 STP“打印机端口扩展”卡安装到端子板中时才使用。
 // 或者作为http://bitsavers.trailing-edge.com/www.computer.museum.uq.edu.au/pdf/EK-VT100-TM-003%20VT100%20Series%20Video%20Terminal%20Technical%20Manual。 pdf
 // 第 433 页：VT100 WC 或 WK 也使用这些。
-// 此 romset 将设置 C 页面添加到设置菜单（点击设置后按键盘 5 两次）
+// 此 ROMset 将设置 C 页面添加到设置菜单（点击设置后按键盘 5 两次）
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
        ROM_LOAD( "23-095e2.e40", 0x0000, 0x0800, CRC(6c8acf44) SHA1(b3ef5af920995a40a316c6dc008960c461853bfc)) // 标签: "23095E2 // (C)DEC // (M)QQ8227" @E40
        ROM_LOAD( "23-096e2.e45", 0x0800, 0x0800, CRC(77f21473) SHA1(6f10b250777c12cca63ee611735f9f36cc05a7ef)) // 标签: "23096E2 // (C)DEC // (M)QQ8227" @E45
@@ -667,127 +667,127 @@ ROM_START( vt100ac ) // 这是来自 http://www.bi 的 VT180 技术手册
        ROM_LOAD( "23-140e2.e56", 0x1800, 0x0800, CRC(4bf1ce4e) SHA1(279f47ec9a68c801c3c05005dd782202ac9e51a4)) // 标签: "AMD // 37109 8230DHP // 23-140E2 // AM9 218CPC // (C)1979 年 12 月" @ E56 // 修订版 2?; 修订版 1 可能是 23-098e2

        ROM_REGION(0x1000, "avo", 0) // "54-13097-00 // PN2280402L" AVO 上的所有开关均打开，除了 S2-3；该映射是否映射到主CPU空间中的0xa000-0xcfff（镜像）？
- // 这组相同的 ROM 也出现在“PN1030385J-F”AVO 上，该 AVO 没有拨码开关；而不是单件跳线？安装在 NDIP20 封装中 E16 和 E22 之间、封装引脚 6 和 15 之间的电阻器
- //注意：对于这两个 avo ROM，引脚 18 为正使能 CE，引脚 20 为负使能 /CE1，引脚 21 为负使能 /CE2，
+ // 这组 ROM 也出现在“PN1030385J-F”AVO 上，该 AVO 没有拨码开关；而不是单件跳线？安装在 NDIP20 封装中 E16 和 E22 之间、封装引脚 6 和 15 之间的电阻器
+ //注意：对于这两个 avo ROM，引脚 18 为正使能 CE，引脚 20 为负使能 /CE1，引脚 21 为负使能 /CE2，
        ROM_LOAD( "23-186e2.avo.e21", 0x0000, 0x0800, CRC(1592dec1) SHA1(c4b8fc9fc0514e0cd46ad2de03abe72271ce460b)) // 标签: "S 8218 // C69063 // 23186E2" @E21
        ROM_LOAD( "23-187e2.avo.e17", 0x0800, 0x0800, CRC(c6d72a41) SHA1(956f9eb945a250fd05c76100b38c0ba381ab8fde)) // 标签: "S 8228 // C69062 // 23187E2" @E 17 号
        // 184 和 185 是 VT100-AC AVO 固件的旧版本吗？

        ROM_REGION(0x2000, "stp", 0) // stp开关1和5闭合，2,3,4打开
        ROM_LOAD( "23-029e4.stp.e14", 0x0000, 0x2000, CRC(da55c62b) SHA1(261b02b774d57253d1dedecab8ca0e368c2a96cd)) // 标签: "S 8218 // C43020 // 23029E4 (C) DEC // TP02" @E14
- // 上面的 ROM 转储顺序可能错误：它是用 A11 转储到引脚 18、A12 转储到引脚 21、A13 转储到引脚 20，但我不确定这些引脚的分配是否正确。
+ // 上面的 ROM 转储顺序可能错误：它是通过 A11 转储到引脚 18、A12 转储到引脚 21、A13 转储到引脚 20，但我不确定这些引脚的分配是否正确。
        // 最坏的情况是它的 8 个部分的顺序是错误的。

        ROM_REGION(0x1000, "充电",0)
        ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选？字处理？替代字符集 ROM
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选？字处理？备用字符集 ROM
        ROM_REGION（0x10000，“stpcpu”，ROMREGION_ERASEFF）
 /* 智能 STP 板：以下是“旧版本”的有限信息。或电路板原型
```

```
-// vt100 的扩展板，上面有处理器和 DMA，旨在充当 STP 打印机板的 ram/发送缓冲区。
-// 它可以填充两组，每组两个 eprom，每个组包含 2k 或 4k eprom，具体取决于 w2/w3 和 w4/w5 跳线。
-// CPU 板上还有两个 prom。我不知道如果安装了 STP 模块，技术上是否有必要安装该板，但由于 alt stp romset，可能是这样。
+// vt100 的扩展板，上面有处理器和 DMA，旨在充当 STP 打印机板的 RAM/发送缓冲区。
+// 它可以填充两个组，每个组有两个 EPROM，每个组包含 2k 或 4k EPROM，具体取决于 w2/w3 和 w4/w5 跳线。
+// CPU 板上还有两个 PROM。我不知道如果安装了 STP 模块，技术上是否有必要安装该板，但由于 alt stp ROMset，可能是这样。
     ROM_LOAD( "23-003e3-00.e10", 0x0000, 0x1000, NO_DUMP) // "EPROM 0" 存储区 0
     ROM_LOAD( "23-004e3-00.e4", 0x1000, 0x1000, NO_DUMP) // "EPROM 1" 存储区 0
     ROM_LOAD( "23-005e3-00.e9", 0x2000, 0x1000, NO_DUMP) // "EPROM 2" 存储区 1
     ROM_LOAD( "23-006e3-00.e3", 0x3000, 0x1000, NO_DUMP) // "EPROM 3" 存储区 1
     //ROM_REGION(0x0800, "avo",0)
- //ROM_LOAD("23-???e2-00.e34", 0x0000, 0x0800, NO_DUMP) // ? 第二个gfx rom?
+ //ROM_LOAD("23-???e2-00.e34", 0x0000, 0x0800, NO_DUMP) // ? 第二个gfx ROM?
     ROM_REGION（0x0400, “舞会”, 0）
- ROM_LOAD( "23-312a1-07.e26", 0x0000, 0x0200, NO_DUMP) // "PROM A"; 处理 8085 I/O? 映射（usart、定时器、dma、comm 等）
- ROM_LOAD( "23-313a1-07.e15", 0x0200, 0x0200, NO_DUMP) // "PROM B"; 处理固件 ROM 映射和内存大小/页面选择；位 0 = RAM 页，位 1-3 未使用，位 4-7 各
选择一个 EPROM
+ ROM_LOAD( "23-312a1-07.e26", 0x0000, 0x0200, NO_DUMP) // "PROM A"; 处理 8085 I/O 映射？（USART、定时器、DMA、通讯等）
+ ROM_LOAD( "23-313a1-07.e15", 0x0200, 0x0200, NO_DUMP) // "PROM B"; 处理固件 ROM 映射和内存大小/页面选择；位 0 = RAM 页，位 1-3 未使用，位 4-7 各
选择一个 EPROM
     */
 ROM_END

 #如果0
 ROM_START( vt103 ) // 这是来自 http://www.bitsavers.org/pdf/dec/terminal/vt103/MP00731_VT103_Aug80.pdf 的原理图
-// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来是 EPROM 0 的版本），但带有
+// 这是标准 VT100 CPU 板，其中填充了“正常”ROM（但后来版本为 EPROM 0），但带有
 // LSI-11 背板（而不是普通的 VT100 背板，因此它不能使用 AVO、WG、GPO 或 VT180 Z80 板）和
-// DEC TU58 双 256k 集成磁带机；本来打算在里面放一块 LSI-11 cpu 卡，
-// 将与终端对话作为其输入/输出设备。有几种 LSI-11 cpu 卡可用？
+// DEC TU58 双 256k 集成磁带机；本来打算在里面放一块 LSI-11 CPU 卡，
+// 将与终端对话作为其输入/输出设备。有几种 LSI-11 CPU 卡可用？
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
- ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 rom'
+ ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 ROM'
        ROM_LOAD（“23-032e2-00.e52”，0x0800，0x0800，CRC（3d86db99）SHA1（cdd8bdecdc643442f6e7d2c83cf002baf8101867））
        ROM_LOAD（“23-033e2-00.e45”，0x1000，0x0800，CRC（384dac0a）SHA1（22aaf5ab5f9555a61ec43f91d4dea3029f613e64））
        ROM_LOAD（“23-034e2-00.e40”，0x1800，0x0800，CRC（4643184d）SHA1（27e6c19d9932bf13fdb70305ef4d806e90d60833））

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM

- ROM_REGION(0x0800, "tapecpu", 0) // 集成串行 tu58-xa 驱动器中 8085 cpu 的 rom
+ ROM_REGION(0x0800, "tapecpu", 0) // 集成串行 tu58-xa 驱动器中 8085 CPU 的 ROM
        ROM_LOAD( "23-089e2.e1", 0x0000, 0x0800, CRC(8614dd4c) SHA1(1b554e6c98bddfc6bc48d81c990deea43cf9df7f)) // 标签: "23-089E2 // P8316E - AMD
```

```
 // 35227 8008NPP ”

 - ROM_REGION(0x80000, "lsi11cpu", 0) // LSI-11 cpu 板的 rom
 + ROM_REGION(0x80000, "lsi11cpu", 0) // LSI-11 CPU 板的 ROM
         ROM_LOAD_OPTIONAL(“未知.bin”, 0x00000, 0x80000, NO_DUMP)
  ROM_END
  #万一

  ROM_START( vt105 ) // 这是来自铁事证据和 vt100.net，因为 vt105 原理图没有被扫描
 -// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来是 EPROM 0 的版本），但带有
 +// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来版本为 EPROM 0），但带有
  // WG波形发生器板工厂安装；这使得终端就像具有 vt100 终端功能的 vt55 一样
  // VT105 旨在用于 MINC 模拟数据采集计算机
         ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
 - ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 rom'
 + ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 ROM'
         ROM_LOAD（“23-032e2-00.e52”，0x0800，0x0800，CRC（3d86db99）SHA1（cdd8bdecdc643442f6e7d2c83cf002baf8101867））
         ROM_LOAD（“23-033e2-00.e45”，0x1000，0x0800，CRC（384dac0a）SHA1（22aaf5ab5f9555a61ec43f91d4dea3029f613e64））
         ROM_LOAD（“23-034e2-00.e40”，0x1800，0x0800，CRC（4643184d）SHA1（27e6c19d9932bf13fdb70305ef4d806e90d60833））

         ROM_REGION(0x1000, "充电", 0)
         ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
 - ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
 + ROM_LOAD_OPTIONAL( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
  ROM_END

  #如果0
  ROM_START（vt110）
 -// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来是 EPROM 0 的版本），但带有
 +// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来版本为 EPROM 0），但带有
  // DECDataway DPM01 板，添加 4 或 5 个特殊网络可寻址 50 欧姆？电流环串行线
 -// 并且可以添加自己的处理器和内存来控制它们。请参阅http://bitsavers.org/pdf/dec/terminal/EK-VT110_UG-001_Dec78.pdf
 +// 并且可以添加自己的处理器和 RAM 来控制它们。请参阅http://bitsavers.org/pdf/dec/terminal/EK-VT110_UG-001_Dec78.pdf
         ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
 - ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 rom'
 + ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 ROM'
         ROM_LOAD（“23-032e2-00.e52”，0x0800，0x0800，CRC（3d86db99）SHA1（cdd8bdecdc643442f6e7d2c83cf002baf8101867））
         ROM_LOAD（“23-033e2-00.e45”，0x1000，0x0800，CRC（384dac0a）SHA1（22aaf5ab5f9555a61ec43f91d4dea3029f613e64））
         ROM_LOAD（“23-034e2-00.e40”，0x1800，0x0800，CRC（4643184d）SHA1（27e6c19d9932bf13fdb70305ef4d806e90d60833））

         ROM_REGION(0x1000, "充电", 0)
         ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
 - ROM_LOAD_OPTIONAL("23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
 -//DECDataway 板 rom 在这里！
 + ROM_LOAD_OPTIONAL ( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
 +//DECDataway 板 ROM 位于此处！
  ROM_END
```

```
 ROM_START( vt125 ) // 这是来自 Bitavers 和 vt100.net，因为 vt125 原理图未被扫描
-// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来是 EPROM 0 的版本），但带有
-// 安装了特殊的“GPO”ReGIS cpu+ram 卡 54-14277，它提供了帧缓冲区、文本旋转、自定义 RAM 字体和许多其他功能。
+// 这是标准 VT100 CPU 板，填充了“正常”ROM（但后来版本为 EPROM 0），但带有
+// 安装了特殊的“GPO”ReGIS CPU+RAM 卡 54-14277，它提供了帧缓冲区、文本旋转、自定义 RAM 字体和许多其他功能。
 // 还配有定制“哑”STP 卡 54-14275。
 // VT125升级套件（从vt100或vt105升级）被称为VT1xx-CB或CL
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
- ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 rom'
+ ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 ROM'
        ROM_LOAD（“23-032e2-00.e52”，0x0800，0x0800，CRC（3d86db99）SHA1（cdd8bdecdc643442f6e7d2c83cf002baf8101867））
        ROM_LOAD（“23-033e2-00.e45”，0x1000，0x0800，CRC（384dac0a）SHA1（22aaf5ab5f9555a61ec43f91d4dea3029f613e64））
        ROM_LOAD（“23-034e2-00.e40”，0x1800，0x0800，CRC（4643184d）SHA1（27e6c19d9932bf13fdb70305ef4d806e90d60833））

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD（“23-018e2-00.e4”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL("23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL ( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM

- //“GPO”又名 vt125“单板”rom 和 proms
- ROM_REGION(0x10000, "monocpu", ROMREGION_ERASEFF) // 8085 subcpu 的 rom
- ROM_LOAD( "23-043e4-00.e22", 0x0000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
- ROM_LOAD( "23-044e4-00.e23", 0x2000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
- ROM_LOAD( "23-045e4-00.e24", 0x4000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
+ //“GPO”又名 vt125“单板”ROM 和 PROM
+ ROM_REGION(0x10000, "monocpu", ROMREGION_ERASEFF) // 8085 子 CPU 的 ROM
+ ROM_LOAD( "23-043e4-00.e22", 0x0000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
+ ROM_LOAD( "23-044e4-00.e23", 0x2000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
+ ROM_LOAD( "23-045e4-00.e24", 0x4000, 0x2000, NO_DUMP) // 2364/MK36xxx 掩码 ROM
        // E25套接字为空

- ROM_REGION(0x100, "dir", ROMREGION_ERASEFF ) // vt125 方向 prom，与 vk100、82s135 等效上相同
+ ROM_REGION(0x100, "dir", ROMREGION_ERASEFF ) // vt125 方向 PROM，与 vk100、82s135 等效上相同
        ROM_LOAD（“23-059b1.e41”，0x0000，0x0100，CRC（4b63857a）SHA1（3217247d983521f0b0499b5c4ef6b5de9844c465））

- ROM_REGION(0x100, "trans", ROMREGION_ERASEFF ) // vt125 x 转换 prom，与 vk100、82s135 等效上相同
+ ROM_REGION(0x100, "trans", ROMREGION_ERASEFF ) // vt125 x 转换 PROM，与 vk100、82s135 等效上相同
        ROM_LOAD（“23-060b1.e60”，0x0000，0x0100，CRC（198317fc）SHA1（00e97104952b3fbe03a4f18d800d608b837d10ae））

- ROM_REGION(0x500, "proms", ROMREGION_ERASEFF) // vt125 单板 proms
- ROM_LOAD（“23-067b1.e135”，0x0000，0x0100，NO_DUMP）//82s135，等待状态舞会
- ROM_LOAD（“23-068b1.e64”，0x0100，0x0100，NO_DUMP）//82s135，sync_a prom
- ROM_LOAD（“23-069b1.e66”，0x0200，0x0100，NO_DUMP）//82s135，sync_b prom
+ ROM_REGION(0x500, "proms", ROMREGION_ERASEFF) // vt125 单板 PROM
+ ROM_LOAD( "23-067b1.e135", 0x0000, 0x0100, NO_DUMP) // 82s135，等待状态 PROM
+ ROM_LOAD( "23-068b1.e64", 0x0100, 0x0100, NO_DUMP) // 82s135,sync_a PROM
```

```
+ ROM_LOAD( "23-069b1.e66", 0x0200, 0x0100, NO_DUMP) // 82s135,sync_b PROM
        ROM_LOAD( "23-070b1.e71", 0x0300, 0x0100, NO_DUMP) // 82s135, 矢量舞会
- ROM_LOAD("23-582a2.e93", 0x0400, 0x0100, NO_DUMP) // 82s129, ras/擦除舞会
+ ROM_LOAD( "23-582a2.e93", 0x0400, 0x0100, NO_DUMP) // 82s129, RAS/擦除 PROM
 ROM_END
 #万一

@@ -795,14 +795,14 @@ ROM_START( vt101 ) // p/n 5414185-01 '不可升级/低成本' vt101/vt102/vt131 m
 // 没有集成的 STP 或 AVO 填充
 // 基于8085而不是I8080
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
- ROM_LOAD( "23-028e4-00.e71", 0x0000, 0x2000, CRC(fccce02c) SHA1(f3e3e93a857443685b816cab4fb52e34c0bc72b1)) // rom 是 vt101 独有的；“CN55004N
8232 // DEC TP03 // 23-028E4-00” 24 引脚掩模 ROM（mc68764 引脚排列）
+ ROM_LOAD( "23-028e4-00.e71", 0x0000, 0x2000, CRC(fccce02c) SHA1(f3e3e93a857443685b816cab4fb52e34c0bc72b1)) // ROM 是 vt101 独有的；“CN55004N
8232 // DEC TP03 // 23-028E4-00” 24 引脚掩模 ROM（mc68764 引脚排列）
        // E69 套接字在 vt101 上为空/未填充
        // vt101 上的 E67 套接字为空/未填充
        // vt101（WD8250 UART）中MB右下角E74处的DIP40缺失

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD（“23-018e2-00.e3”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
 ROM_END


@@ -812,17 +812,17 @@ ROM_START( vt102 ) // p/n 5414185-01 '不可升级/低成本' vt101/vt102/vt131 m
 // 基于8085而不是I8080
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
        ROM_DEFAULT_BIOS(“vt102”)
- ROM_SYSTEM_BIOS（0，“vt102o”，“VT102旧版ROM”）
+ ROM_SYSTEM_BIOS( 0, "vt102o", "VT102 较旧的 ROM" )
        ROMX_LOAD( "23-042e4-00.e71", 0x0000, 0x2000, CRC(e8aa006c) SHA1(8ac2a84a8d2a9fa0c6cd583ae35e4c21f863b45b), ROM_BIOS(0)) // 与 vt131 共享
        ROMX_LOAD( "23-041e4-00.e69", 0x8000, 0x2000, CRC(b11d331e) SHA1(8b0f885c7e032d1d709e3913d279d6950bbd4b6a), ROM_BIOS(0)) // 与 vt131 共享
- ROM_SYSTEM_BIOS（1，“vt102”，“VT102较新的ROM”）
+ ROM_SYSTEM_BIOS( 1, "vt102", "VT102 较新的 ROM" )
        ROMX_LOAD( "23-226e4-00.e71", 0x0000, 0x2000, CRC(85c9279a) SHA1(3283d27e9c45d9e384227a7e6e98ee8d54b92bcb), ROM_BIOS(1)) // 与 vt131 共享
        ROMX_LOAD( "23-225e4-00.e69", 0x8000, 0x2000, CRC(3567c760) SHA1(672473162e9c92cd237e4dbf92c2700a31c5374b), ROM_BIOS(1)) // 与 vt131 共享
        //e67 套接字在 vt102 上为空，但在下面的 vt131 上填充

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD（“23-018e2-00.e3”，0x0000，0x0800，CRC（6958458b）SHA1（103429674fc01c215bbc2c91962ae99231f8ae53））
- ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
 ROM_END

 ROM_START( vt131 ) // p/n 5414185-01 'unupgra dable/low cost' vt101/vt131 主板，具有 vt132 型块串行模式
```

```
@@ -831,31 +831,31 @@ ROM_START( vt131 ) // p/n 5414185-01 '不可升级/低成本' vt101/vt131 主板
 // 基于8085而不是I8080
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
        ROM_DEFAULT_BIOS("vt131")
- ROM_SYSTEM_BIOS(0, "vt131o", "VT131旧版ROM")
+ ROM_SYSTEM_BIOS( 0, "vt131o", "VT131 较旧的 ROM")
        ROMX_LOAD( "23-042e4-00.e71", 0x0000, 0x2000, CRC(e8aa006c) SHA1(8ac2a84a8d2a9fa0c6cd583ae35e4c21f863b45b), ROM_BIOS(0)) // 与 vt102 共享
        ROMX_LOAD( "23-041e4-00.e69", 0x8000, 0x2000, CRC(b11d331e) SHA1(8b0f885c7e032d1d709e3913d279d6950bbd4b6a), ROM_BIOS(0)) // 与 vt102 共享
- ROM_SYSTEM_BIOS(1, "vt131", "VT131较新的ROM")
+ ROM_SYSTEM_BIOS( 1, "vt131", "VT131 较新的 ROM")
        ROMX_LOAD( "23-226e4-00.e71", 0x0000, 0x2000, CRC(85c9279a) SHA1(3283d27e9c45d9e384227a7e6e98ee8d54b92bcb), ROM_BIOS(1)) // 与 vt102 共享
        ROMX_LOAD( "23-225e4-00.e69", 0x8000, 0x2000, CRC(3567c760) SHA1(672473162e9c92cd237e4dbf92c2700a31c5374b), ROM_BIOS(1)) // 与 vt102 共享
- ROM_LOAD( "23-280e2-00.e67", 0xA000, 0x0800, CRC(71b4172e) SHA1(5a82c7dc313bb92b9829eb8350840e072825a797)) // 在 vt101 快速参考指南中称为"VT131
ROM"；引脚 20、18 和 21 是此掩码 ROM 上的 /CE /CE2 和 /CE3
+ ROM_LOAD( "23-280e2-00.e67", 0xA000, 0x0800, CRC(71b4172e) SHA1(5a82c7dc313bb92b9829eb8350840e072825a797)) // 在 vt101 快速参考指南中称为"VT131
ROM"；引脚 20、18 和 21 是此掩膜 ROM 上的 /CE /CE2 和 /CE3

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD ("23-018e2-00.e3", 0x0000, 0x0800, CRC (6958458b) SHA1 (103429674fc01c215bbc2c91962ae99231f8ae53) )
- ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL( "23-094e2-00.e4", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM
 ROM_END

 ROM_START(vt180)
-// 这是标准 VT100 CPU 板，填充了"正常"ROM（但后来是 EPROM 0 的版本），但带有
-// Z80子板添加到扩展槽，并更换STP适配器（STP rom替换为正常设置）
+// 这是标准 VT100 CPU 板，填充了"正常"ROM（但后来版本为 EPROM 0），但带有
+// Z80子板添加到扩展槽，并更换STP适配器（STP ROM替换为正常设置）
        ROM_REGION( 0x10000, "主CPU", ROMREGION_ERASEFF )
- ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 rom'
+ ROM_LOAD( "23-061e2-00.e56", 0x0000, 0x0800, CRC(3dae97ff) SHA1(e3437850c33565751b86af6c2fe270a491246d15)) // 版本 2 1980 '后来的 ROM'
        ROM_LOAD ("23-032e2-00.e52", 0x0800, 0x0800, CRC (3d86db99) SHA1 (cdd8bdecdc643442f6e7d2c83cf002baf8101867) )
        ROM_LOAD ("23-033e2-00.e45", 0x1000, 0x0800, CRC (384dac0a) SHA1 (22aaf5ab5f9555a61ec43f91d4dea3029f613e64) )
        ROM_LOAD ("23-034e2-00.e40", 0x1800, 0x0800, CRC (4643184d) SHA1 (27e6c19d9932bf13fdb70305ef4d806e90d60833) )

        ROM_REGION(0x1000, "充电", 0)
        ROM_LOAD ("23-018e2-00.e4", 0x0000, 0x0800, CRC (6958458b) SHA1 (103429674fc01c215bbc2c91962ae99231f8ae53) )
- ROM_LOAD_OPTIONAL("23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 rom
+ ROM_LOAD_OPTIONAL ( "23-094e2-00.e9", 0x0800, 0x0800, NO_DUMP) // 可选（某些型号默认）备用字符集 ROM

        ROM_REGION(0x10000, "z80cpu", 0) // z80 子板
        ROM_LOAD ("23-021e3-00.bin", 0x0000, 0x1000, CRC (a2a575d2) SHA1 (47a2c40aaec89e8476240f25515d75ab157f2911) )
diff --git a/src/mame/drivers/zn.cpp b/src/mame/drivers/zn.cpp
索引 106468653f9..2b250e8602b 100644
---a/src/mame/drivers/zn.cpp
+++ b/src/mame/drivers/zn.cpp
@@ -370,7 +370,7 @@ void zn_state::zn_map(address_map &map)
```

```
        地图(0x1fa30000, 0x1fa30003).noprw(); /* ?? */
        地图(0x1fa40000, 0x1fa40003).nopr(); /* ?? */
        地图(0x1fa60000, 0x1fa60003).nopr(); /* ?? */
-map(0x1faf0000, 0x1faf07ff).rw("at28c16", FUNC(at28c16_device::read), FUNC(at28c16_device::write)); /* EEPROM */
+ map(0x1faf0000, 0x1faf07ff).rw("at28c16", FUNC(at28c16_device::read), FUNC(at28c16_device::write)); /* EEPROM */
        地图(0x1fb20000, 0x1fb20007).r(FUNC(zn_state::unknown_r));
 }

@@ -625,8 +625,8 @@ void zn_state::coh1000c_map(address_map &map)

 MACHINE_START_MEMBER（zn_state，coh1000c）
 {
- m_rombank[0]->configure_entries( 0, 16, m_bankedroms->base() + 0x400000, 0x400000 ); /* 银行游戏 ROM */
- m_soundbank->configure_entries( 0, 16, memregion("audiocpu")->base() + 0x8000, 0x4000 ); /* 存储音频 ROM */
+ m_rombank[0]->configure_entries( 0, 16, m_bankedroms->base() + 0x400000, 0x400000 ); /* 银行游戏 ROM */
+ m_soundbank->configure_entries( 0, 16, memregion("audiocpu")->base() + 0x8000, 0x4000 ); /* 存储音频 ROM */
 }

 MACHINE_RESET_MEMBER（zn_state，coh1000c）
@@ -637,7 +637,7 @@ MACHINE_RESET_MEMBER(zn_state,coh1000c)

 MACHINE_RESET_MEMBER（zn_state，glpracr）
 {
- m_audiocpu->set_input_line(INPUT_LINE_RESET, ASSERT_LINE); // glpracr qsound rom 套接字为空
+ m_audiocpu->set_input_line(INPUT_LINE_RESET, ASSERT_LINE); // glpracr QSound ROM 插槽为空
        MACHINE_RESET_CALL_MEMBER(coh1000c);
 }

@@ -1112,7 +1112,7 @@ void zn_state::coh1000ta_map(address_map &map)

 MACHINE_START_MEMBER（zn_state，coh1000ta）
 {
- m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
        if (m_soundbank.found())
        {
                m_soundbank->configure_entry( 0, memregion( "audiocpu" )->base() + 0x20000 ); /* TODO : Bank 0 正在寻址 ROM 的前 16 KB? */
@@ -1580,7 +1580,7 @@ RA9701 SUB
 |---------------------------------------|
 笔记：
       CAT702 - 标记为“ET02”的保护芯片（DIP20）
- ROM 217、216 和 326 - 表面安装 32MBit MASK ROM (SOP44)
+ ROM 217、216 和 326 - 表面安装 32MBit 掩模 ROM (SOP44)
       ROM 042 和 046 - 27C2001 EPROM
       ROM 212 至 215 - 27C4001 EPROM
       MAIN_IF2 和 SUB_IF2 - AMD Mach211 CPLD (PLCC44)
@@ -1624,7 +1624,7 @@ PS9805
```

```
 笔记：
       * - 未填充的 ROM 位置。
       CAT702 - 保护芯片标有“MG11”（DIP20）
- ROM-x - 表面安装 32MBit MASK ROM (SOP44)
+ ROM-x - 表面贴装 32MBit 掩膜 ROM (SOP44)
       ROM 412 和 049 - 27C040 EPROM
       MASK4A - smt 焊盘（未填充）
       MASK4B - DIP42 插座（未安装）
@@ -1661,7 +1661,7 @@ void zn_state::coh1002e_map(address_map &map)

 MACHINE_START_MEMBER（zn_state，coh1002e）
 {
- m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
        if (m_okibank.found())
                m_okibank->configure_entries( 0, memregion( "oki" )->bytes()/0x10000, memregion( "oki" )->base(), 0x10000 ); /* 未经审核的 */
 }
@@ -1844,7 +1844,7 @@ void zn_state::init_bam2()

 MACHINE_START_MEMBER（zn_state，bam2）
 {
- m_rombank[0]->configure_entries( 0, 16, m_bankedroms->base(), 0x400000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 16, m_bankedroms->base(), 0x400000 ); /* 银行游戏 ROM */
 }

 MACHINE_RESET_MEMBER(zn_state,bam2)
@@ -1871,7 +1871,7 @@ 光枪类射击游戏
 使用带有 Rom 板的 Sony ZN-1 硬件和
 硬盘驱动器

-U35和U36 eprom是27c1001，被认为是BIOS
+U35和U36 EPROM是27c1001，被认为是BIOS
 数据。

 磁盘驱动器是Quantum ????2.1 GB??
@@ -2339,10 +2339,10 @@ 注：
       ATHG-03.22 - 声音程序（27C010 EPROM）
       ATHG-04.21 /

- ATHG-05.4136 - 声音数据（16MBit DIP42 MASKROM）
+ ATHG-05.4136 - 声音数据（16MBit DIP42 mask ROM）
       ATHG-06.4134 /

- ATHG-07.027 - 图形数据（32MBit DIP42 MASKROM）
+ ATHG-07.027 - 图形数据（32MBit DIP42 掩模 ROM）
       ATHG-08.028 /
       ATHG-09.210 /
```

```
       ATHG-10.029 /
@@ -2375,7 +2375,7 @@ void zn_state::coh1001l_map(address_map &map)

 MACHINE_START_MEMBER(zn_state, coh1001l)
 {
- m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 4, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
 }

 MACHINE_RESET_MEMBER(zn_state, coh1001l)
@@ -2441,7 +2441,7 @@ void zn_state::coh1002v_map(address_map &map)

 MACHINE_START_MEMBER(zn_state, coh1002v)
 {
- m_rombank[0]->configure_entries( 0, 24, m_bankedroms->base(), 0x100000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 24, m_bankedroms->base(), 0x100000 ); /* 银行游戏 ROM */
 }

 MACHINE_RESET_MEMBER(zn_state, coh1002v)
@@ -2550,7 +2550,7 @@ 特库摩 TPS1-7
 |-------------------------------------|
 笔记：
       该游戏板上有一些无人居住的位置，包括
- 4 个未填充位置用于 4x 32MBit smt SOP44 MASKROM
+ 4 个未填充位置，用于 4x 32MBit smt SOP44 掩模 ROM
       D43001 RAM 附近有 1 个未填充的 uPD72103AG 位置
       Z80 ROM 附近有 2 个未安装的位置，用于 2 个连接器，可能用于网络链接？
       ROM 'CBAJ2' 附近 PAL16V8 的 1 个未填充位置
@@ -2566,8 +2566,8 @@ 注：
       主程序 ROM 附近有 3 个逻辑芯片。
       2x 4MBit EPROM 标记为“CBAJ1”和“CBAJ2”
       1x 2MBit EPROM 标记为“CBAJZ80”
- 9x 32MBit smt SOP44 MASKROM 标记为“CB-00”至“CB-08”（图形）
- 2x 32MBit smt SOP44 MASKROM 标记为“CB-SE”和“CB-V0”（连接到 YMZ280B）
+ 9x 32MBit smt SOP44 掩模 ROM 标记为“CB-00”至“CB-08”（图形）
+ 2x 32MBit smt SOP44 掩模 ROM 标记为“CB-SE”和“CB-V0”（连接到 YMZ280B）
       LH540202 - CMOS 1024 x 9 异步 FIFO (PLCC32)
       D43001 - 32K x8 SRAM，相当于 62256 SRAM

@@ -2613,8 +2613,8 @@ 注：
       3个逻辑芯片
       2x 4MBit EPROM 标记为“SHMJ-B”和“SHMJ-A”
- 4x 32MBit smt SOP44 MASKROM 标记为“SH03”、“SH02”、“SH01”和“SH00”。有空间
- 用于另外 11 个 32MBit smt SOP44 MASKROM。
+ 4x 32MBit smt SOP44 掩模 ROM 标记为“SH03”、“SH02”、“SH01”和“SH00”。有空间
+ 用于另外 11 个 32MBit smt SOP44 掩模 ROM。
```

```
 */

 WRITE8_MEMBER(zn_state::coh1002m_bank_w)
@@ -2632,7 +2632,7 @@ void zn_state::coh1002m_map(address_map &map)

 MACHINE_START_MEMBER（zn_state，coh1002m）
 {
- m_rombank[0]->configure_entries( 0, 8, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
+ m_rombank[0]->configure_entries( 0, 8, m_bankedroms->base(), 0x800000 ); /* 银行游戏 ROM */
 }

 MACHINE_RESET_MEMBER（zn_state，coh1002m）
@@ -4901,7 +4901,7 @@ ROM_START(beastrzr)
        PSARC95_BIOS

        ROM_REGION32_LE( 0x1800000, "bankedroms", 0 )
- ROM_LOAD16_BYTE( "b.roar_u0213", 0x000001, 0x080000, CRC(2c586534) SHA1(a38dfc3a45446d24a1caac89b0f560989d46ded5) ) /* 对于 U0212 和 U0213，8ing
使用相同ROM标签*/
+ ROM_LOAD16_BYTE( "b.roar_u0213", 0x000001, 0x080000, CRC(2c586534) SHA1(a38dfc3a45446d24a1caac89b0f560989d46ded5) ) /* 对于 U0212 和 U0213，8ing
使用相同ROM标签*/
        ROM_LOAD16_BYTE( "b.roar_u0212", 0x000000, 0x080000, CRC(1c85d7fb) SHA1(aa406a42c424cc16a9e5330c68dda9acf8760088) ) /* 即使版本之间的内容发
生变化 */
        ROM_LOAD16_BYTE(“b.roar-u0215”,0x100001,0x080000,CRC(31c8e055)SHA1(2811789ab6221b972d1e3ffe98916587990f7564))
        ROM_LOAD16_BYTE(“b.roar-u0214”,0x100000,0x080000,CRC(1cdc450a)SHA1(9215e5fec52f7c5c0070feb621eb9c77f98e2362))
@@ -4923,7 +4923,7 @@ ROM_START( bldyroar )
        PSARC95_BIOS

        ROM_REGION32_LE( 0x1800000, "bankedroms", 0 )
- ROM_LOAD16_BYTE( "b.roar-u0213", 0x000001, 0x080000, CRC(63769342) SHA1(7231188073b997b039467db85ce7c85383daf591) ) /* 对于 U0212 和 U0213, 8ing
使用缩进ical ROM 标签 */
+ ROM_LOAD16_BYTE( "b.roar-u0213", 0x000001, 0x080000, CRC(63769342) SHA1(7231188073b997b039467db85ce7c85383daf591) ) /* 对于 U0212 & U0213, 8ing
使用缩进物理ROM标签*/
        ROM_LOAD16_BYTE( "b.roar-u0212", 0x000000, 0x080000, CRC(966b7169) SHA1(63e025cacb84e89d30b40ed6cfa5c63d84c298c4) ) /* 即使版本之间的内容发
生变化 */
        ROM_LOAD16_BYTE(“b.roar-u0215”,0x100001,0x080000,CRC(31c8e055)SHA1(2811789ab6221b972d1e3ffe98916587990f7564))
        ROM_LOAD16_BYTE(“b.roar-u0214”,0x100000,0x080000,CRC(1cdc450a)SHA1(9215e5fec52f7c5c0070feb621eb9c77f98e2362))
diff --git a/src/mame/drivers/zr107.cpp b/src/mame/drivers/zr107.cpp
索引 74328b8d75b..379172b5bda 100644
---a/src/mame/drivers/zr107.cpp
+++ b/src/mame/drivers/zr107.cpp
@@ -79,7 +79,7 @@ 注意：
      403GA：时钟32.000MHz（64/2）
      68000：时钟8.000MHz（64/8）
     TSOP56：2Mx8 TSOP56 FlashROM 的未填充位置
- DIP42：2Mx8 DIP42 MASKROM 的未填充位置
+ DIP42：2Mx8 DIP42 掩码 ROM 的未填充位置
      DIP32：512kx8 EPROM 的未填充位置
```

```
       SOJ40：DRAM 814260-70 的未填充位置
       QFP44：MB89371FL 的未填充位置
@@ -148,7 +148,7 @@ 注意：
        CY7C199：32kx8 SRAM
        CY7C109：128kx8 SRAM
        62256：32kx8 静态存储器
- DIP42：1Mx8 DIP42 MASKROM 的未填充位置
+ DIP42：1Mx8 DIP42 掩码 ROM 的未填充位置
        MC88916 : Motorola MC88916 低偏移 CMOS PLL 时钟驱动器

 ROM使用
--
cgit v1.2.3
```