MAME Score Scanner User Guide

How to use the app, what each function does, and how to move from memory dumps to a final second-screen layout.

What The App Is For

The scanner helps you discover score RAM for arcade games so you can build a second-screen .lay file. In the best case it finds a winner automatically. In harder cases it narrows the field and creates probe layouts so you can verify the right address visually.

Main Modes Of Use

Mode	Best For	Typical Outcome
Simple Mode	Normal day-to-day use	Record full-board score dumps, analyze them, fill winner boxes, create final or probe layouts.
Advanced Mode	Debugging and discovery work	Manual block stepping, manual mode forcing, focus ranges, probe creation, database work.

The Two Core Workflows

Workflow	When To Use It	How It Works
Exact	When you think the score is stored directly as a number	You enter the visible score and the scanner searches for bytes that already match that format directly.
Delta	When you want to compare snapshots before and after score changes	You record one or more score dumps, then analyze what changed or replay many score formats against the recorded snapshots.

Supported Score Methods

- bcd2 / bcd2rev: 2-byte packed BCD variants for shorter score ranges
- bcd2_implied0 / bcd2rev_implied0: 2-byte packed BCD where the final visible zero is implied
- bcd3 / bcd4: normal packed BCD
- bcd3rev / bcd4rev: reverse-byte packed BCD
- bcd3_implied0 / bcd4_implied0: scores that drop a trailing visible zero in RAM
- bcd3rev_implied0 / bcd4rev_implied0: reversed-byte implied-zero variants
- u16/u24/u32 big- or little-endian: integer storage
- u16/u24/u32 implied0 variants: integer values stored in tens with a trailing visible zero
- galaga_digits: special display-digit decode for Galaga-style storage

Board Families In The App

The board dropdown is meant to match clues users can see in MAME, such as Nintendo or Z80, not just internal scanner preset names. Start with the family that best matches the game's hardware or source information if you know it.

- Capcom CPS-1
- Generic 64K
- Namco 8-bit / 64K
- Nintendo Z80 / 64K
- Konami 8-bit / 64K
- Taito 8-bit / 64K
- Sega Z80 / 64K
- Midway 8080 / 64K
- Irem M62 / 64K
- Capcom Pre-CPS / 64K

Typical Simple Mode Workflow

1. Load the game in MAME, then open the scanner so it can connect to the bridge.
2. Let the scanner auto-detect the game if possible, or type/load the game profile manually.
3. Choose the likely board family if it is not already set correctly.
4. Enter a known score and click Record All Blocks.
5. Score points, pause again, enter the new score, and repeat Record All Blocks several times.
6. Run Analyze Exact All to test many supported score methods across the whole board.
7. If a clean result is found, move it into the winner box and create the final layout.
8. If several candidates remain, create a probe layout and verify them visually.

Typical Advanced Mode Workflow

1. Switch to Advanced Mode.
2. Use Exact workflow when you believe the score is stored directly as BCD or integer bytes.
3. Use Delta workflow when you need to compare changes or work through a noisy game.
4. Step through blocks manually if the board-wide approach is too broad.
5. Use Apply Focus to isolate a hot cluster after Analyze or Analyze Delta All.
6. Use Create Focus Probe .LAY or Create Probe .LAY to watch uncertain survivors live.
7. Use Save To Database when you are confident enough in the final ranges and mode.

What Each Important Button Does

Button	Use
Load Game	Loads a saved game profile if it exists. Also useful after bridge auto-detection fills the game field.
Record All Blocks	Captures a full set of board blocks for the score currently entered.
Analyze Exact All	Replays many supported score modes against all-block dumps across the selected board.

Button	Use
Analyze Delta All	Ranks hot blocks and suggests a focus cluster from all-block delta data.
Create Probe .LAY	Builds a visual compare layout from remaining exact candidates.
Create Focus Probe .LAY	Builds a raw hex/live-byte probe from the current focus range.
Create Custom .LAY	Builds a centered stacked score layout from addresses you enter manually.
Create Final .LAY	Builds a centered stacked score layout from the P1, P2, and HI winner boxes.
Save To Database	Stores the current game, board, mode, ranges, and notes in the local profile database.

Winner Boxes

Winner boxes are the main handoff point between analysis and export. They should hold the full range, not just the first byte, for example 0x60B5->0x60B7. If the scanner finds a result automatically it can place it in the currently selected winner target box. You can also type or edit these boxes manually after confirming a probe. Each field can now keep its own mode, leading-zero preference, and optional display adjust helper of none, +1, or -1 for preview/export-only corrections.

Turn Flag Workflow

1. Use this only after Live, parked score fields, and HI are already confirmed for a live/parked game.
2. Turn on Live/Parked mode and, if you are only testing, turn Auto Fill Winners off first.
3. Pause on P1, click Capture, switch to P2 and pause again, then click Diff.
4. Click Capture again on P2, switch back to P1 and pause, then click Diff again.
5. Run Auto Flag Turn to rank likely turn-flag bytes and suggested P1/P2 flag values.
6. Only fill Active Flag, P1 Flag, and P2 Flag after the parked ranges themselves are already confirmed, otherwise preview/export can appear wrong even when the addresses are good.

Layout Designer

Layout Designer affects preview and export only. It lets you move the title, labels, and score rows, rename exported text, and choose row digit formatting for HI, P1, P2, and TIME using trim, minimum-2-digit, fixed-width, implied-trailing-zero, follow-winner, or timer-specific display styles.

The designer now follows the real digit count for the selected mode, uses matching label/score colors, repacks only visible rows, auto-applies presets when selected, and remembers your current working layout while the app session stays open. For example, Frogger can preview and export as a true 5-digit layout instead of being stretched to 6 digits.

Time helper rows can also follow timer-specific export styles or digit-style helper decoding, which is useful for cases like Mario World where the timer is stored as three visible HUD digits rather than a normal integer value.

The left Preview Elements list remains the safest way to manage the layout because hidden rows stay selectable there. Title, core labels, and extra helper labels can be hidden for export and later turned back on without losing the row from the designer.

How To Use Probe Layouts

1. Create a probe layout when more than one candidate remains.
2. Load the probe in MAME and compare each panel against the live in-game score.
3. Watch which panel updates immediately while playing and which one only updates later or after death.
4. Put the confirmed winning range into the matching winner box.

Saving To The Database

Save To Database is the way the scanner gets smarter over time. It stores the game name, board family, block, current mode, P1/P2/HI ranges, optional Time helper range, notes, solved status, and per-field preview/export helpers such as leading zeros and display adjust. The Solved checkbox can now force a profile to save as solved even when the game uses a helper model rather than every classic winner box. Extra helper rows are also preserved when used. For console systems, make sure the Game box contains the real game title rather than a generic system name such as genesis or snes before saving.

PreMame For Console Per-Game Layouts

Some console systems boot through one shared artwork\system\default.lay, which makes per-game console layouts awkward if you want each cart to have its own small file. PreMame is an optional wrapper that solves that by checking the loaded cart name before MAME starts, temporarily swapping in a matching per-game .lay if one exists, and restoring the original default.lay after MAME closes.

- If a matching per-game .lay exists, PreMame temporarily uses it as default.lay for that launch.
- If no matching per-game .lay exists, the normal default.lay remains in use.
- The original console default.lay is restored automatically after exit.
- Arcade games do not need this unless you explicitly route them through the wrapper.

Known Special Cases

- Galaga uses display-byte decoding instead of plain BCD.
- Galaxian uses reversed-byte 3-byte BCD.
- Pac-Man is now confirmed as a working bcd4rev Namco example using the known RAM map ranges.
- Donkey Kong can produce a strong address winner even when more than one mode still survives.
- Popeye uses live and parked score-slot behavior rather than a simple fixed P1/fixed P2 layout.
- Frogger is a strong example of a 2-byte reversed BCD score with an implied trailing zero, using bcd2rev_implied0 and a 5-digit exported layout.
- Genesis/Sonic is a good example of mixed helper data: score works as an implied-zero integer, while timer preview is better treated as a dedicated helper field.
- SNES Mario World is a good example of mixed helper data too: score/lives are numeric HUD values, but the timer is better treated as a 3-digit display helper and may need preview/export display adjustment.

Troubleshooting

Symptom	Likely Cause	Next Step
No RAM patterns found	Wrong block, wrong board, or wrong mode	Try a different block, use all-block recording, or switch workflow.
The same score candidate keeps coming back for both players	Game may reuse an active score slot	Check whether the game swaps live and parked score addresses between turns.
Layout displays correct values only after death	You found a parked or copied score slot	Keep it as a useful clue, but prefer the address that updates live while playing.
More than one winner remains	Address family or mode ambiguity	Generate a probe layout and verify the remaining candidates visually.
Generated layout shows labels but no scores	The export may still contain an old broken Lua callback	Regenerate the layout after updating the scanner to the fixed builder.
Console game keeps using the shared default layout	No matching per-game .lay file was found before launch	Check the exported filename and use PreMame if you want separate console .lay files.

Recommended Working Rule

If the scanner gives you a clean address winner, treat that as the strongest signal. If the mode is still ambiguous, keep testing live in a layout until the visual result tells you which decode is correct. When in doubt, trust the game screen more than the raw candidate list.