

MMbg – Backgammon for MMBasic

Manual · Version 1.10.36 · October 2026

© 2026 Matthias Hollatz · Free software under the GNU GPL v3 or later

One package, three editions, every RP2350. All three programs have version **1.10.36** and play the same game with the same engines:

`mmbg_pc3.bas` – boards **with PSRAM**: **Pico Computer 3 (PC3)**, also the **Colour Maximite 2** (chapter 15).
`mmbg_pc1.bas` – HDMI and VGA boards **without PSRAM**, e.g. the PicoMite HDMI/USB board ("PC1") or a Pico 2 with VGA (chapter 16).
`mmbg_pica.bas` – the **PicoCalc** (Part 2).

Tested with PicoMite **V7.00.00RC6** and CMM2 **7.00.00b4**. Memory figures (marked with ~) may change slightly with later firmware.

Contents

Part 1 – MMbg PC3, Colour Maximite 2, PC1 and the common basics

1. What is MMbg?
2. Requirements
3. Installation
4. The screen and the terminal
5. Playing a match
6. Undo, saving and resuming
7. Recording, analysis and screenshots
8. Engines and playing strength
9. The neural nets
10. Playing against gnubg over WLAN
11. Long test runs
12. Options
13. Command summary
14. Files
15. MMbg on the Colour Maximite 2
16. MMbg PC1: HDMI and VGA without PSRAM
17. Notes and limits
18. A flight through the source code

Part 2 – MMbg PiCa (PicoCalc)

- P1. MMbg PiCa at a glance
- P2. Requirements and firmware set-up
- P3. Installation and start
- P4. Screen, terminal and controls
- P5. Differences from the PC3
- P6. Nets and memory on the PicoCalc
- P7. The 16-bit nets: rounding and playing strength
- P8. Files
- P9. Notes and limits

Licence and thanks

New since the last release (1.10.34)

- **MMbg PC1:** a new edition for HDMI and VGA boards without PSRAM, with the interface and mouse of the PC3; the big nets live in the **MODBUFF** (flash, chapter 16).
- **PicoCalc:** the big nets now come from the MODBUFF as well. Move list, undo and `.mat` are as large as on the PC3, and the PiCa **plays exactly as strong** (Part 2, P6).
- **Screenshots** by the firmware: **Print Screen** or **Ctrl+** on every device, also as PNG (chapter 7.7).
- **Undo by mouse click** asks first (6.1); match end "O wins the match 5-2"; match counter in the terminal too; short prompts on the PicoCalc.

Part 1 – MMbg PC3, Colour Maximite 2, PC1 and the common basics

1. What is MMbg?

MMbg is a complete backgammon program written in MMBasic. It plays **matches** from 1 to 15 points, with doubling cube, Crawford rule and everything that goes with it.

- **Playing strength:** With the *big* engine MMbg plays at about the level of GNU Backgammon "Expert" (0-ply). Against gnubg "World class" (2-ply) it wins about **47 % of 5-point matches** (577 of 1236 measured matches).
- **Graphics and terminal:** the board is shown in colour (HDMI or VGA 640 × 480, PicoCalc: LCD 320 × 320) with animated moves. At the same time a serial terminal shows an ASCII board with move list, evaluations and log.
- **Controls:** play with the keyboard or, except on the PicoCalc, with the mouse. Sounds for dice and moves are available on request.
- **Comfort:** Multi-level undo, automatic saving and resuming. Matches are recorded as `.mat` files for analysis in gnubg. Positions can be exchanged via **GNUbg IDs**.
- **For tinkerers:** gnubg on a PC can act as opponent over WLAN. On top come overnight test runs, duplicate matches and detailed logs.

2. Requirements

Which edition for which device?

Device	Edition	tested with
Pico Computer 3 (PC3), RP2350B with PSRAM	`mmbg_pc3.bas`	PicoMitePC3 V7.00.00RC6
Colour Maximite 2 (generation 2)	`mmbg_pc3.bas`	CMM2 7.00.00b4 (chapter 15)
PicoMite HDMI/USB board ("PC1"), no PSRAM	`mmbg_pc1.bas`	PicoMiteHDMI V7.00.00RC6 (chapter 16)
Pico 2 with VGA reference board	`mmbg_pc1.bas`	PicoMiteVGA V7.00.00RC6 (chapter 16)
PicoCalc with Pico 2 W or Pico 2	`mmbg_pica.bas`	WebMite V7.00.00RC6 (Part 2)

Rule of thumb: a board **with PSRAM** takes `mmbg_pc3.bas`, an HDMI or VGA board **without PSRAM** takes `mmbg_pc1.bas`.

The table below is for the PC3. What is different on the other devices is described in chapter 15, chapter 16 and Part 2.

What	Requirement
Computer	**Pico Computer 3 (PC3)**: RP2350B with PSRAM, HDMI, USB
Firmware	**PicoMitePC3 V7.00.00** (tested with RC6)

What	Requirement
Display	HDMI, MODE 3 (640 × 480, 16 colours)
Keyboard	USB keyboard on the PC3 and/or serial terminal
Terminal (recommended)	VT100 capable (e.g. Tera Term, PuTTY), **80 × 40 characters**, colours on
Mouse (optional)	USB mouse; a PS/2 mouse is found as well (chapter 15)
Sound (optional)	`OPTION AUDIO` set up (I²S on the PC3)
WLAN (optional)	only for gnubg as opponent (chapter 10)
SD card (recommended)	drive B: for the program folder, logs and `.mat` files

On the PC3 the big nets are kept as floating-point numbers in PSRAM (~512 KB), the animation uses another ~150 KB for a framebuffer.

3. Installation

3.1 Files

All files belong **in one folder** together with the program (chapter 3.2). The package contains the files for all editions; each edition simply ignores the files it does not need (table in chapter 14 and in Part 2, chapter P8).

File	Size	Purpose	If missing …
`mmbg_pc3.bas`	~190 KB	the program for PC3 and CMM2	–
`mmbg_pc1.bas`	~185 KB	the program for HDMI/VGA without PSRAM (chapter 16)	–
`mmbg_pica.bas`	~195 KB	the program for the PicoCalc (Part 2)	–
`bgnet345.dat`	~96 KB	small nets 3, 4, 5 (gnubg)	engines 0–2 only
`bgnet02.dat`	text	output weights of the big nets	no engines 4 and 5
`bgnet0.bin`, `bgnet2.bin`	256 000 bytes each	big nets contact and crashed (PC3, CMM2)	no engines 4 and 5
`bgesc.bin`	64 KB	escape tables (gnubg; PC3, CMM2)	no engines 4 and 5
`bgnet16.mb`, `bgpk16.dat`	144 192 bytes, text	big nets with 16 bits and escape tables for the MODBUFF, check sums (PC1, PicoCalc)	no engines 4 and 5
`bgbo15.dat`	~1.6 MB	bear-off database up to 15 checkers	race with net 5 only
`bgnet35.dat`, `bgbo10.dat`		smaller fallbacks, used only if `bgnet345.dat` or `bgbo15.dat` is missing	–

MMbg creates the settings and saved-game files (`mmbg_pc3.ini` , `bgext.ini` , `mmbg_pc3.sav` ; on the PC1 `mmbg_pc1.…` , on the PicoCalc `mmbg_pica.…`) itself in the program folder, plus the folder `log/` below it.

3.2 Everything in one folder on the SD card

MMbg looks for all its files **relative to the program folder**: nets, database, settings, saved match and the log folder below it. Nothing is tied to a fixed drive. The folder can therefore **have any name and lie on any drive at any depth** – which is also why the same package runs on every computer, whether the SD card is called `B:` there (PC3, PicoCalc) or `A:` (Colour Maximite 2).

Recommended:

Folder	Contents
`/mmbg`	program, nets, database, ini files, saved match
`/mmbg/log`	logs and `.mat` files (created by MMbg itself)

To set this up, unpack the package and copy the folder `mmbg` to the root of the SD card. Alternatively transfer each file with `XMODEM RECEIVE "name"` into a folder you have created with `MKDIR "B:/mmbg"` .

3.3 Start

Always start **from the program folder**, so that MMbg finds its files there:

```
CHDIR "B:/mmbg"
LOAD "mmbg_pc3.bas"          (PC1, VGA board: "mmbg_pc1.bas")
RUN
```

Device	Start
PC3, PC1, VGA board	`LOAD "mmbg_pc3.bas"` or `"mmbg_pc1.bas"` , then `RUN` . The published programs have only short comments and load without `, C` . With `, C` MMBasic leaves out the comments and a little more program memory remains.
PicoCalc	**`LOAD "mmbg_pica.bas", C`** , then `RUN` . Here `, C` is needed, otherwise memory is too short (Part 2).
Colour Maximite 2	`CHDIR "mmbg"` , then `RUN "mmbg_pc3.bas"` . The SD card is `A:` there (chapter 15).

At start-up the logo, the licence and the loading lines for nets, database and WLAN appear. MMbg then waits on the start screen for **N** (new match).



Figure 1: The terminal after start-up: ASCII board, below it the loading lines for nets and database.

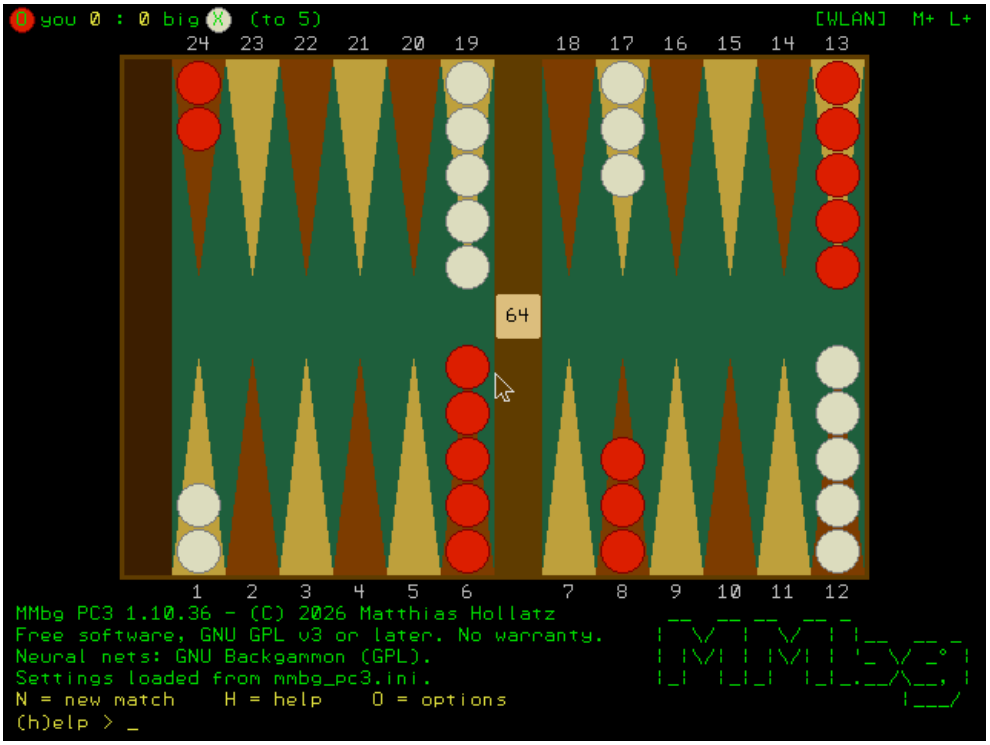


Figure 2: Start screen on HDMI. Bottom left version, licence and messages, on the right the logo, below it the most important keys.

> **Error messages with a line number** refer to the program without comment lines when it was loaded with `, C`.

3.4 The log folder

Logs and `.mat` files go to the **log folder** (screenshots, however, go to the root of the SD card, chapter 7.7). By default this is `log/` below the program folder (`B:/mmbg/log`); MMbg creates it at start-up if it does not exist. Nothing needs to be set up.

To use another folder, open the options page (**O → 7**), press **L** and enter the folder:

Input	Log folder
`log`	`log/` below the program folder (default)
`B:/games/bglog`	any existing folder, absolute
`-`	the program folder itself

Save with **S**. In `bgext.ini` the setting is the line `logdir=`: empty means `log/`, `.` means the program folder.

4. The screen and the terminal

4.1 HDMI screen

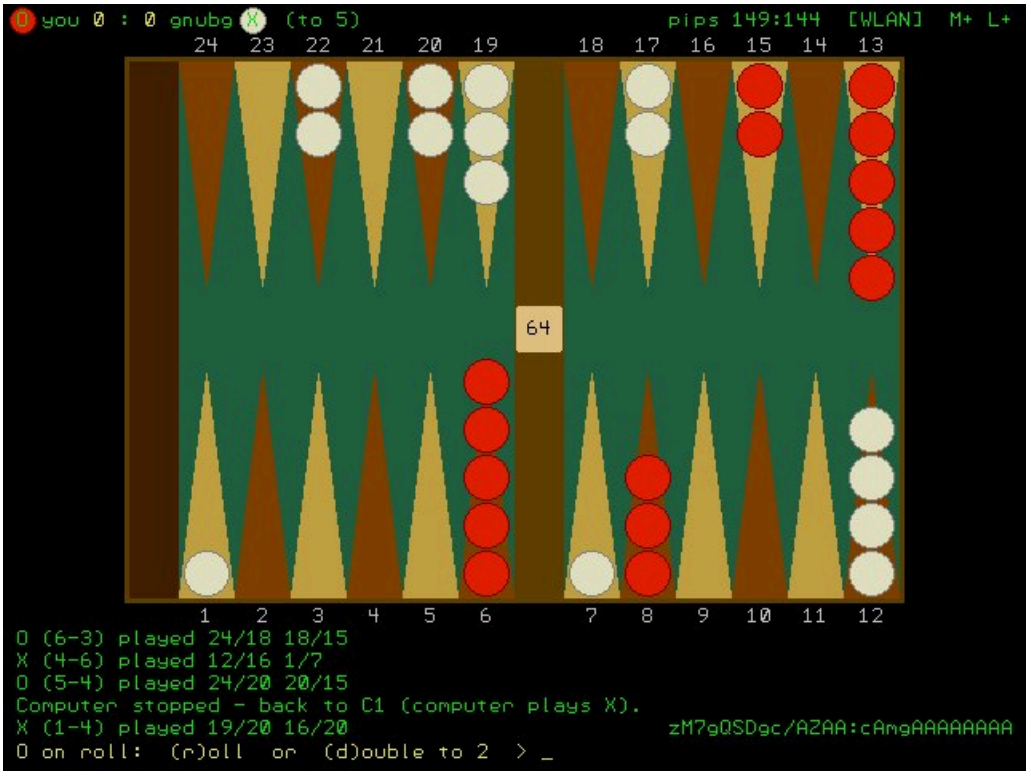


Figure 3: A game in progress. Title with players, score, pips and switches, below it the board with the cube in the bar, the five-line log, the GNUbg ID bottom right and the prompt at the very bottom (here the question before the roll).

- **Title line**
  - O and X appear as small checkers in the player's colour, next to them the player: `you` for a human, otherwise the name of the engine.
  - The **score is yellow**, followed by the match length, e.g. `(to 5)`.
  - `M 4:2` counts the matches won in this session. It appears in computer runs and as soon as a match is over.
  - In the middle: `Crawford` or `Post-Crawford`.
  - On the right: the pips (option), `[WLAN]`, `M±` for `.mat` recording and `L±` for the log.

- **Board:** 24 points, bar, tray. The dice always lie **to the right of the player** as seen from his side, so O's dice on the right and X's dice on the left.
  - The cube sits in the middle of the bar or with its owner. In the Crawford game it is grey with "64"; doubling is not allowed then.
  - An **offered double** appears as a single cube on the board, on the side of the player who has to decide.
- **Log:** The last five messages, moves and hints are shown above the prompt.
- **GNUbg ID:** The ID of the current position is shown in green at the right of the bottom log line (chapter 7.4).
- **Colours:** Text is green; keys, input and changing values are yellow.

On the Colour Maximite 2 and with MMbg PC1 (HDMI or VGA) the screen looks exactly the same.

4.2 Terminal

The terminal (80 × 40, VT100) shows the same game as text:

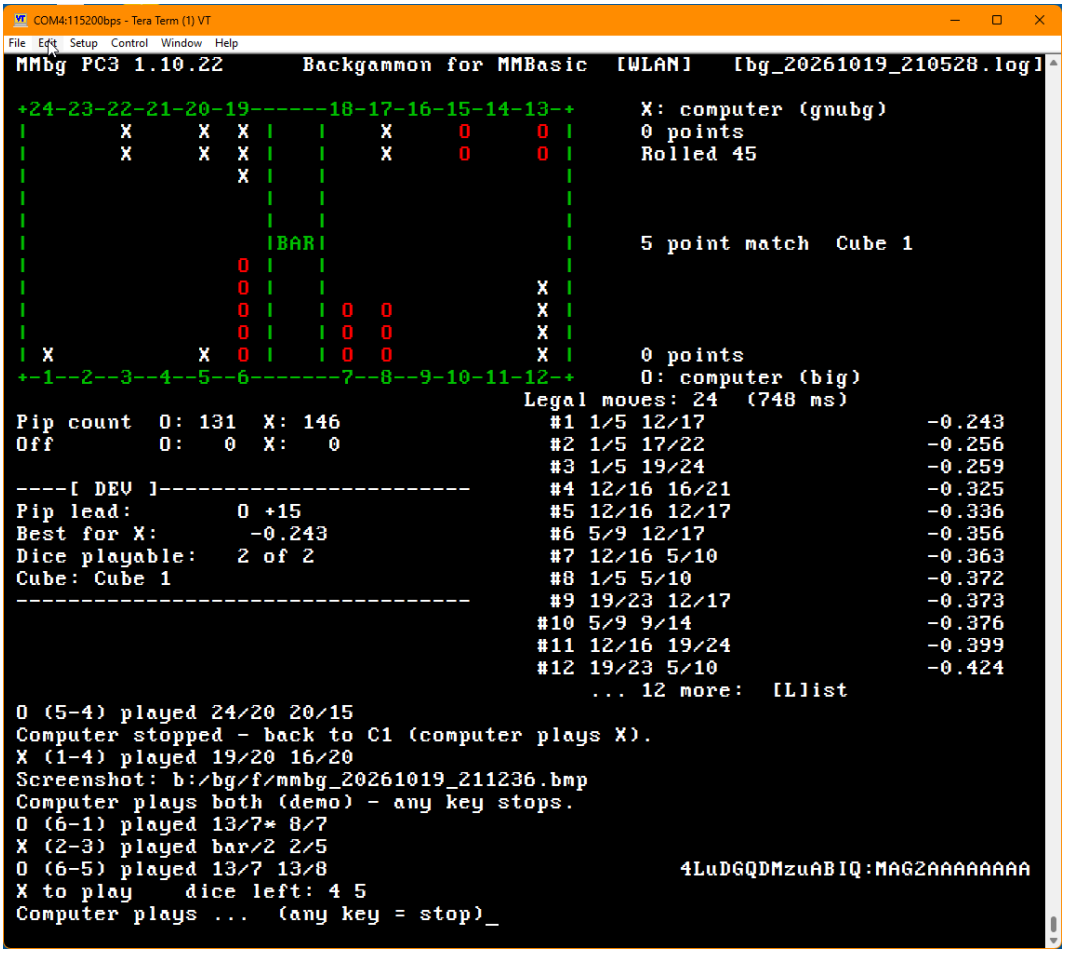


Figure 4: The terminal during a computer run (big against gnubg): ASCII board, players and dice on the right, below pips, DEV column, move list with evaluations and the log with the GNUbg ID.

- **ASCII board:** The frame is green, O checkers are white, X checkers red. The option *swap* exchanges the colours, the option *flip* mirrors the board as in the graphics.
- **Right of the board:** players, points, match length, cube and the rolled dice.
- **Below:** pips, borne-off checkers and the bar. The **DEV column** (key `P`) shows the pip lead, the best value, the playable dice and the cube.

- **Move list on the right:** the 12 best moves with evaluation (equity), the rest with `L`. Before the roll the cube analysis appears instead: match winning chances for no double, double/take and double/pass, plus the computer's recommendation.
- **Bottom:** the last **8 log lines** and the **GNUbg ID** on the right. From there it is easy to copy with the mouse.

5. Playing a match

5.1 Start

- **N** starts a new match with the set length.
- **M7** starts a match to 7 points. 1 to 15 points are possible.
- **C1** (default): the human plays O (bottom), the computer plays X (top).
- The opening roll decides who starts.

5.2 Moving

Input	Meaning
`13/7 8/7`	moves in gnubg notation, several partial moves separated by spaces
`bar/22`, `6/off`	from the bar, bearing off (short: `b`, `o`; `-` instead of `/` also works)
`13 24`	start points only: the bigger die first
`b 6`	from the bar and from the 6
`#3`	move no. 3 of the move list
`E`	end the turn if no die can be played any more
`U`	undo (chapter 6)

A partial move is only valid if the largest possible number of dice can still be played afterwards. If a player cannot move at all, MMbg passes automatically and shows the dice briefly.

With the mouse (PC3, CMM2, PC1):

Action	Meaning
Left click on an own checker (or the bar)	move it with the bigger die; if that is not possible, with the smaller
Right click on an own checker (or the bar)	move it with the smaller die; if that is not possible, with the bigger
Click on your own dice	undo, after asking (chapter 6.1)
Before the roll: click on the cube	double
Before the roll: click on the board	roll
Click on the offered cube	take. Pass only with the key `P`

Example: a checker is on the bar and the roll is 3-1. A **right click** on the bar enters with the 1, and the 3 is left for another checker.

5.3 Doubling cube

Before every roll MMbg asks the player on roll, if he may double: `(r)oll or (d)ouble`. If the opponent offers a double, the answer is `(t)ake` or `(p)ass`.

- With the option *hint* the log shows the cube recommendation and the match winning chances.
- In the **Crawford game** doubling is not allowed. Post-Crawford follows.
- Match rules apply. Money-game rules like Jacoby or beaver are not available.

5.4 End of game and match

The winner scores one point (single), two (gammon) or three (backgammon), each times the cube value. So a game brings 1, 2, 3, 4, 6, 8, 12 … points. The next game starts after three seconds.

When the match is decided, MMbg reports e.g. `*** O wins the match 5-2 ***`. A new match starts with ENTER, in computer runs by itself after three seconds.

6. Undo, saving and resuming

6.1 Undo

- **U** during a turn takes back your own partial moves.
- **U** without a moved checker jumps back to **your last turn start**, with the same dice. This also works after the computer has already moved, for example after a mis-click.
- **Another U** goes back one more of your own turns, and so on, up to the start of the game.
- After an undo the computer rolls again, so nobody can react to its dice. Undo only works within the current game.
- **With the mouse:** a click on your own dice asks first: `Undo? click the dice again or (y)es / (n)o`. A second click on the dice or **Y** takes back, anything else does not. So a mis-click costs no move. The key **U** takes back without asking.

It is based on the **move history**: for every turn start MMbg keeps the GNUbg ID, i.e. board, score, cube and dice. The PC3 and the PicoCalc keep 300 turn starts, MMbg PC1 the last 40.

6.2 Saving and resuming

- MMbg saves the current match **automatically** to `mmbg_pc3.sav` (PC1: `mmbg_pc1.sav`, PicoCalc: `mmbg_pica.sav`):
  - at every turn start of a human,
  - at the questions roll/double and take/pass,
  - in computer runs C2 and C3 at every turn. C4 does not save.
- **SG** saves immediately, **LG** loads the saved match.
- At **start-up** MMbg asks: "*Saved match found – continue it? (y/n)*".
- Saved are the whole move history (undo still works after loading), computer mode, engines and the running `.mat` recording.
- The file is deleted at the end of the match.

What is in the file: The file is plain text, one entry per line. The core are the `turn=` lines: for **every turn start** of the game they hold the **GNUbg ID** of the position (board, score, cube, dice),

plus the turn number and how many `.mat` entries had been written up to then. When loading, MMbg rebuilds the position from the last ID; the earlier IDs make undo possible after loading.

```
' MMbg PC3 saved match
version=1.10.36
date=25-10-2026 20:26:53
computer=1                    computer mode (C1)
engine_o=0                    engines O and X
engine_x=4
game=7                        game number in this session
turn=4HPwATDgc/ABMA:cAmgAAAAAAAA 0 0     turn start: GNUbg ID, turn, .mat count
turn=4HPwATDgc/ABMA:QQnrABAAGAAA 1 1
...
matfile=log/mmbg_20261025_202011.mat
mgame=3                       game number in the match (.mat)
act=2|52: 13/8 24/22          actions of the game recorded so far
now=4HPwATDgc/ABMA:cAmgAAAAAAAA          position before the roll (at roll/double)
```

Every ID can also be pasted into gnubg on its own to look at a position from a saved match. The explanations on the right are only in this manual, not in the file.

7. Recording, analysis and screenshots

7.1 Matches as `.mat` for gnubg

With the option **mat** (`M+` in the title) MMbg writes every match to its own file:

```
[log folder]mmbg_YYYYMMDD_HHMMSS.mat
```

- The format is the same one gnubg writes with `export match mat`.
- The header is written when the match starts, each game is appended at its end.
- New matches never overwrite old files.
- Moves taken back with undo do not appear in the file.
- A resumed match continues in its old file.
- In gnubg X is player 0 (top), O is player 1 (bottom).

Analysis in gnubg:

1. Transfer the file to the PC, e.g. via SD card or XMODEM. XMODEM pads the end of the file with zero bytes; gnubg does not mind.
2. In gnubg: *File* → *Import* → *.mat* or `import mat file.mat`.
3. *Analyse* → *Analyse match* or `analyse match`, then the statistics with `show statistics match`.

7.2 Log file

Switch on or off with **G** (`L±` in the title). The setting is saved with the other options. The file `bg_YYYYMMDD_HHMMSS.log` in the log folder contains:

- every move with time, dice, rank in the move list and evaluation,
- the computing time, the pips and gnubg's response time,
- cube decisions with winning chances, and game and match results.

A new file starts at about 1 MB. **GN** starts a new one immediately.

A move line looks like this (in the log it is a single line):

```
  2105.3s G12 T 14 X 6-4 big   24/18 13/9          #1/18 +0.123 best +0.123
          gen 512ms  pip O 140 X 150  ext 70ms
```

That means: seconds since start, game 12, turn 14, X rolled 6-4, engine big, the move played, its **rank in the move list** (1 of 18 legal moves), its evaluation and the best evaluation, the computing time of the move generator, the pips of both sides and, with gnubg, the response time. There are also lines with `---` (cube, engines, settings, undo), `***` (end of game), `###` (end of match and running total), `!!!` (errors) and, on the PicoCalc, `D` (board dump with key D).

7.3 `.mat` or log – which one for what?

	`.mat` recording	log file
Purpose	**replay and analyse the match**	**watch the engines**, mainly in test runs
Format	standard format, read by gnubg (and other programs)	own text format for humans and evaluation scripts
Contents	only the game: rolls, moves, cube, results	in addition rank in the move list, evaluations, computing times, pips, gnubg response times, errors, settings
Size	one small file per match	one running file, a new one from approx. 1 MB
Typical use	your own games against the computer	overnight long runs (chapter 11)

Both can be on at the same time. Players who only want to analyse their games in gnubg need only `.mat`.

7.4 GNUbg ID

The ID describes a position completely: board, match length, score, cube, Crawford, player on roll and dice. An example:

```
4HPwATDgc/ABMA:cAmgAAAAAAAA
```

- **MMbg → gnubg:** The ID is shown at the bottom right and on the dump page (key **D**). Copy it in the terminal and paste it into the *GNUbg ID* field in gnubg, or type `set gnubgid …`. This way any position can be examined in gnubg straight away.
- **gnubg → MMbg:** Simply paste an ID at the prompt, also at roll/double or take/pass. MMbg asks "*Take over the position...? (y/n)*" and sets up the position. Money game, a match longer than 15 points and a cube above 64 are rejected.
- The IDs have been cross-checked with gnubg 1.08.003. An offered double is not part of the ID, because gnubg drops it when reading.

7.5 Hint option

With **hint** (option 3) MMbg writes the best move to the log when a human is on turn, and the recommendation at the cube questions. Good for learning.

7.6 Board dump (D)

D shows the position as text: checkers per point, bar, borne-off checkers, the rolled dice, the dice still to play and the **GNUbg ID**. This is the easiest way to note or report a position, for example together with a bug report. On the PicoCalc the board and the ID are also written to the log (lines starting with `D`).

7.7 Screenshot (Print Screen or Ctrl+\)

MMbg switches on `OPTION PRINTSCREEN ON` at start-up. Then the **firmware** saves the screen as soon as the **Print Screen** key or **Ctrl+** is pressed, even in the middle of an animation. This works on every device: PC3, Colour Maximite 2, PC1, VGA board and PicoCalc (there the LCD with 320 × 320).

The file goes to the **root of the SD card**, not to the log folder:

```
SCREEN_YYYYMMDD_HHMMSS.BMP
```

As PNG: enter once at the prompt (the setting is kept):

```
OPTION PRINTSCREEN FORMAT PNG
```

Then PNG files are written, much smaller than BMP. From PicoMite **V7.00.00RC6** and CMM2 **7.00.00b4** the image viewer of the file manager (`FM`) shows the PNG pictures directly on the device. The pictures in this manual were made this way.

Older firmware without `OPTION PRINTSCREEN` skips the command at start-up; then there are simply no screenshots. The own **Ctrl+P** of earlier versions is gone.

8. Engines and playing strength

8.1 The engines

No.	Name	How it works	Strength (rough)
0	**base**	evaluation formula: pips, blots, points, bar	beginner
1	**book**	base plus opening book	beginner+
2	**tuned**	weights tuned in the lab	casual player
3	**net**	small gnubg nets (0-ply)	good club player
4	**big**	big gnubg nets, pre-filter with the small nets	**≈ gnubg "Expert" (0-ply)**
5	**deep**	big with 1 roll look-ahead (1-ply)	stronger than big, approx. 18 times slower
E	**gnubg**	gnubg on the PC over WLAN	as strong as gnubg is set

Choose with **A45**: engine for O is 4, for X is 5. For each side one character from 0 to 5 or E.

Measurements for big:

Measurement	Result
matches against gnubg 2-ply (to 5), PC3	**577 of 1236 = 46.7 %**
matches against gnubg 2-ply (to 5), PicoCalc (16 bit)	199 of 450 = 44.2 % ± 4.6 % (Part 2, P7)
error per move, measured against gnubg 2-ply	big 2.6 mEq, gnubg 0-ply 2.4 mEq
best move found	big 82 %, gnubg 0-ply 83 %

- About 90 % of the gap to gnubg 2-ply comes from move choice, about 10 % from the cube.
- The computing time averages about 0.4 to 0.6 s per move on all computers, about 10 s with *deep*.

8.2 Cube model

For the cube MMbg calculates like gnubg at 0-ply: from the win, gammon and backgammon chances of the net, the **Kazaross-XG2 match equity table** gives the chances for no double, double/take and double/pass.

8.3 Computer modes

Command	Mode
`C0`	two humans
`C1`	computer plays X (default)
`C2`	computer plays both sides (to watch, with pauses)
`C3`	both, without pauses and without animation
`C4`	**turbo**: both, no board drawing, endless matches (test runs)

In C2 to C4 **any key** stops the computer.

9. The neural nets

9.1 How does such a net work? (explained very simply)

At its core a neural net is a **big calculation with many fixed numbers**. It gets a position and returns how good it is. Nothing more – it does not think ahead and knows no rules.

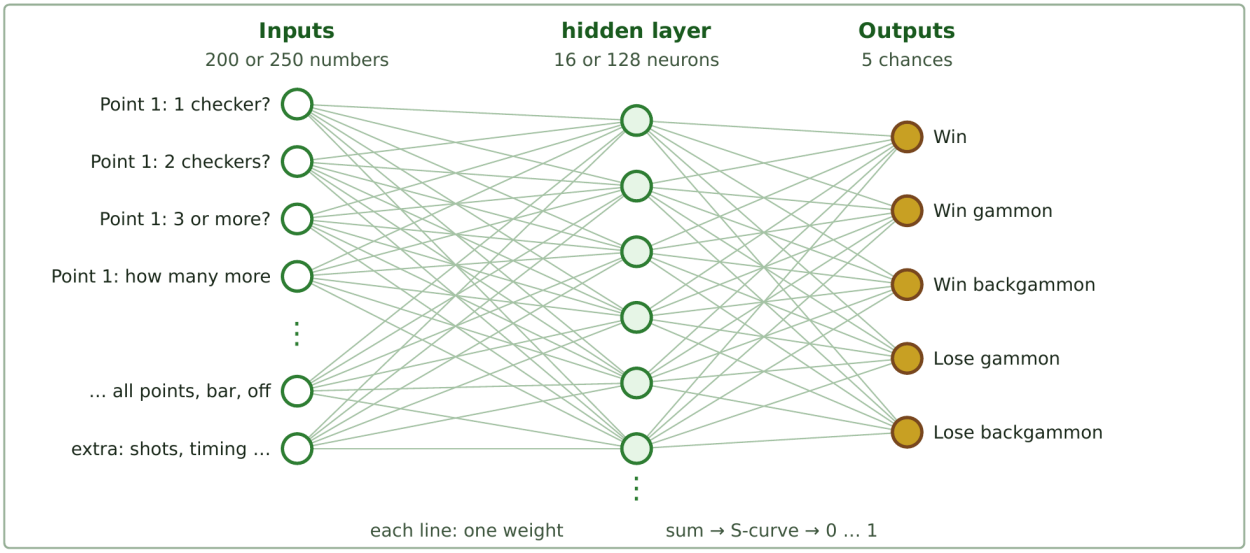


Figure 5: Structure of a gnubg net (shown reduced). On the left the position as numbers, in the middle the hidden layer, on the right the five chances.

1. The position becomes numbers. For each point and each side there are four inputs: is there exactly 1 checker (a blot)? Exactly 2? 3 or more? And how many beyond that? Bar and borne-off checkers come on top. This gives 200 numbers, mostly 0 or 1. The big net gets 50 extra features a human would "see": How often can I be hit? How easily do my back checkers escape? How much time do I have left?

2. The hidden layer. Each neuron in the middle forms a **weighted sum** of all inputs: each input times its own weight, all added up. The sum goes through an **S-curve** that squeezes it to a value between 0 and 1. In this way each neuron becomes a small "detector" for a pattern, such as "good home board" or "exposed blot" – which pattern, the net has learned by itself.

3. The outputs. The five output neurons do the same again with the values from the middle. Out come five chances between 0 and 1: win, win with gammon, win with backgammon, lose with gammon, lose with backgammon. From these one single value is formed, the **equity**:

```
Equity = 2 x win - 1 + gammon wins + backgammon wins - gammon losses - backgammon losses
```

4. Where do the weights come from? They are the actual knowledge. The gnubg authors trained them with a very large number of played-out positions: the net estimates, compares with the real result and shifts the weights a tiny bit in the right direction – millions of times. MMbg only calculates with the finished numbers; it does not learn by itself.

5. How does this become a move? The move generator finds all legal moves (often 20 to 50, with doubles sometimes several hundred). For each move the **resulting position** is sent through the net. The move with the best equity is played. Engine *big* first rates all moves with the small net and recalculates only the best 12 with the big one; *deep* additionally looks ahead at all 21 possible rolls of the opponent.

The big net needs about 32 000 multiplications per position (250 × 128 + 128 × 5). In MMBasic on the PC3 this takes about 19 ms. Since most of the 250 inputs are zero in a real position, MMbg only adds up the rows of the non-zero inputs (about 50).

9.2 The nets in MMbg

MMbg uses the **neural net weights of GNU Backgammon** (gnubg 1.08, GPL). MMbg evaluates the nets itself in MMBasic. The input encoding is the same as in gnubg.

Net	Structure	Used for
3 contact (small)	200 inputs, 16 hidden neurons, 5 outputs	engine net, pre-filter for big
4 crashed (small)	200-16-5	positions with a collapsed home board
5 race (small)	200-8-5	races, when there is no contact any more
0 contact (big)	250 inputs (200 + 50 extra features), 128 neurons, 5 outputs	engines big and deep
2 crashed (big)	250-128-5	engines big and deep

- **The 5 outputs** are: win, win with gammon, win with backgammon, lose with gammon, lose with backgammon.
- **The 50 extra features** of the big net, such as hit chances, escapes, containment and timing, are calculated by MMbg following gnubg.
- **Bear-off database `bgbo15.dat`:** computed by ourselves, it contains no gnubg data. It gives exact values in the endgame: one side, 6 points, up to 15 checkers, 54 264 positions, with gammon shares.
- **On PC1 and PicoCalc** the same weights are stored with 16 bits per weight in the MODBUFF (flash) – see chapter 16 and Part 2, chapters P6 and P7.

Which nets each engine needs:

Engine	needs
0 to 2	nothing, only the program
3 net	`bgnet345.dat`
4 big, 5 deep	additionally `bgnet02.dat` and the big nets: PC3 and CMM2 `bgnet0.bin`, `bgnet2.bin`, `bgesc.bin`; PC1 and PicoCalc `bgnet16.mb`, `bgpk16.dat`
all	`bgbo15.dat` makes the endgame exact

10. Playing against gnubg over WLAN

The PC3 (and the PicoCalc with a Pico 2 W) can use **GNU Backgammon on a PC** in the home network as opponent. MMbg asks the questions, gnubg does the computing. The Colour Maximite 2 and MMbg PC1 have no WLAN.

10.1 Setting up

Step 1 – WLAN on the PC3: set up once and restart:

```
OPTION WIFI "MyNetwork", "Password"
```

Step 2 – gnubg on the PC: start it (e.g. `gnubg-cli`) and enter:

```
external :41000
```

gnubg then waits on port 41000. The playing strength is set in gnubg itself, e.g. 0-ply or 2-ply. The package contains ready-made batch files for Windows in `gnubg_start_PC` (0-, 1- and 2-ply).

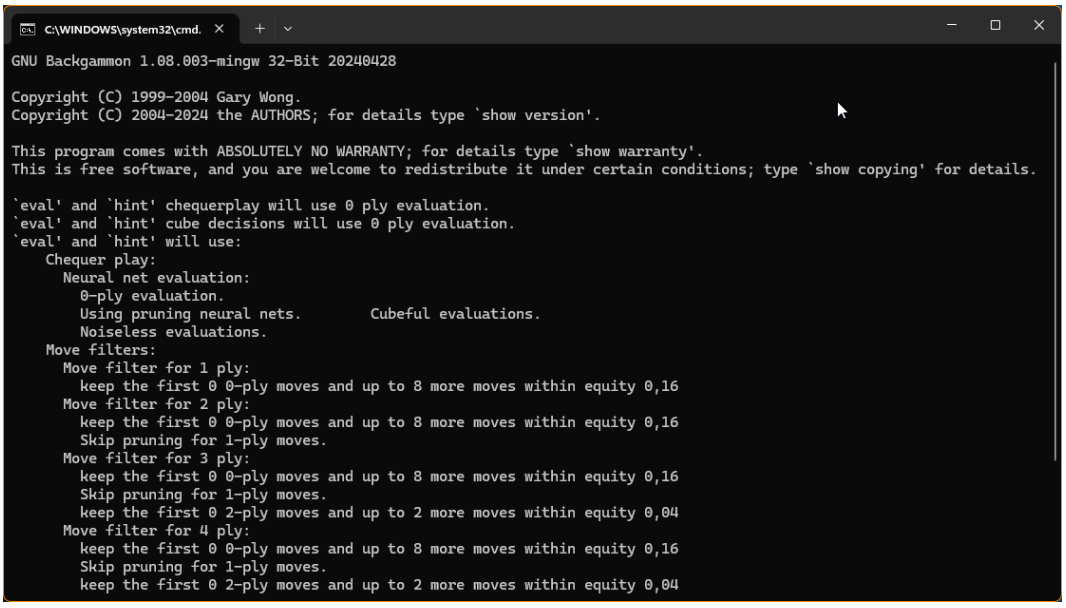


Figure 6: gnubg 1.08.003 on the PC (command line), set to 0-ply. Next comes `external :41000`.

Step 3 – in MMbg: open the options page (**O → 7**) and set:

- **H** host: the IP address of the PC
- **P** port: e.g. 41000
- **W** timeout in seconds
- **F** fallback: the engine that steps in if gnubg does not answer (e.g. 4 big)
- **T** tests the connection (version and 10 pings)
- **S** saves everything to the ini files

When `[WLAN]` appears in the title, the WLAN is up.



Figure 7: Connection test with T successful: at the bottom the gnubg version and 10 pings with response times.

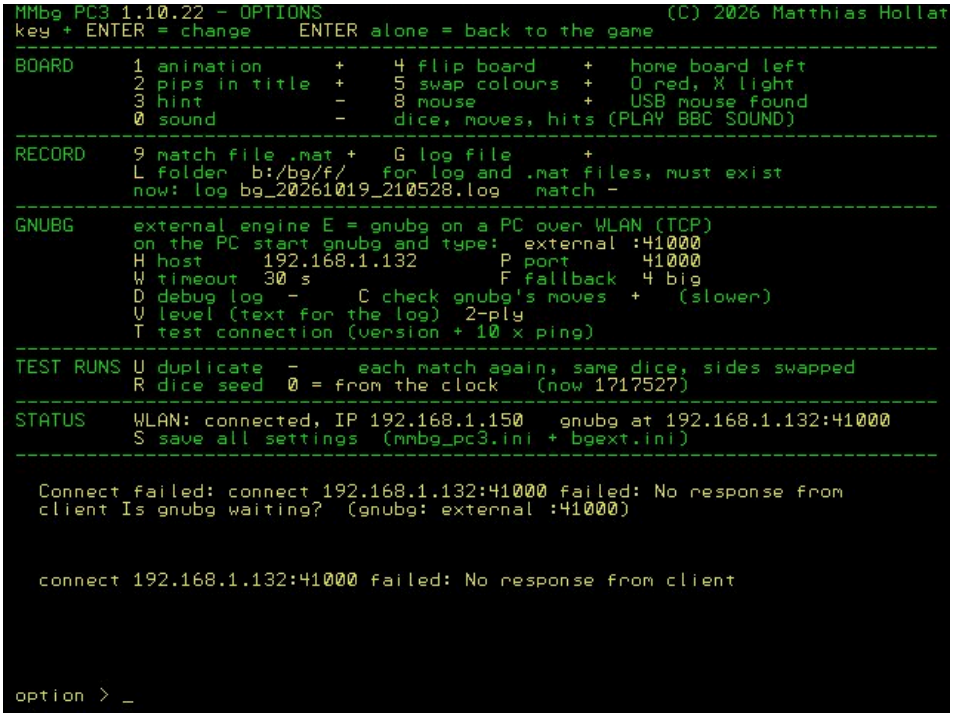


Figure 8: Connection test failed: gnubg is not running or not yet waiting (`external :41000` forgotten), or host/port are wrong.

10.2 Playing

Command	Effect
`A4E`	O = big, X = gnubg
`C1`	the human plays O, gnubg plays X: a game against gnubg
`C2` or `C3`	big against gnubg to watch
`C4E`	engines O 4, X gnubg and turbo

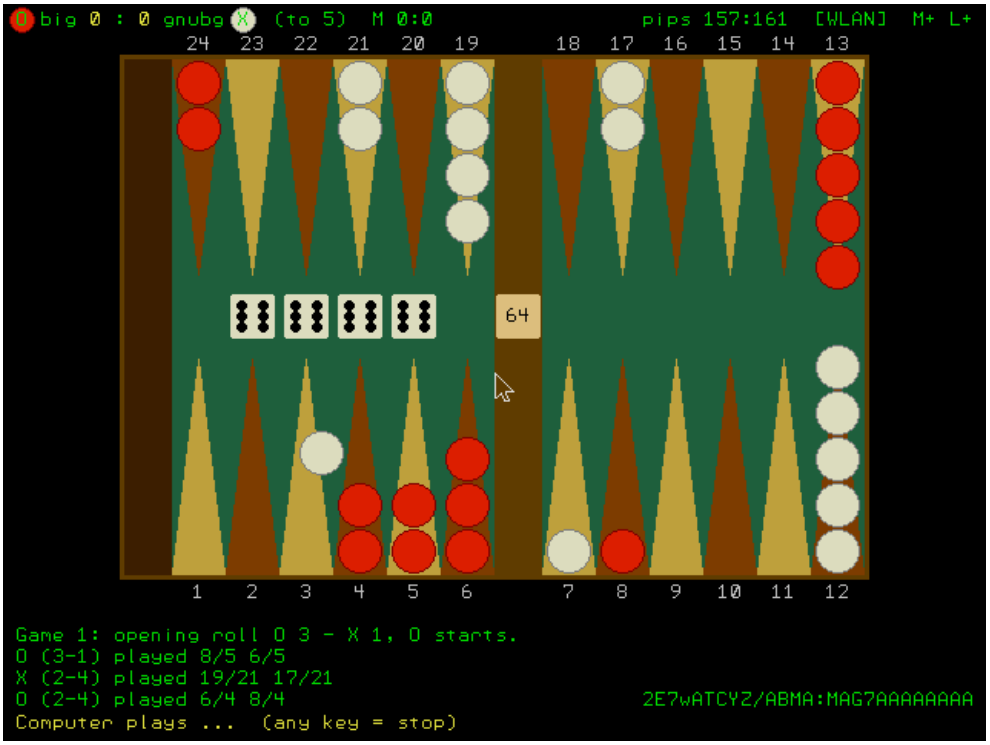


Figure 9: X is gnubg. The log reports version, level in brackets (here 2-ply), address and the fallback engine (fallback big); bottom right the GNUbg ID.

Further switches on the options page:

- **D debug log:** gnubg's requests and answers go into the log.
- **C check:** MMbg also evaluates gnubg's moves. Slower, but good for comparisons.
- **V level:** only a text for the log and the start-up message, e.g. "2-ply". If nothing is entered, it shows `(?)`, e.g. `gnubg 1.08.003-mingw 32-Bit (?) at 192.168.1.132` – this is not an error, just the missing text. MMbg **cannot query** gnubg's playing strength: it is set in gnubg itself (e.g. `set evaluation chequerplay evaluation plies 2`). Anyone who wants to record in the log which level was played against enters it here by hand and saves with S.

gnubg's move is always played. MMbg looks up gnubg's answer in its own move list. With some doubles there are several hundred legal moves, and the list can be full (PC3 and PicoCalc 500 moves). If gnubg's move is missing for that reason, MMbg puts it at the top of the list and plays it; the log notes `--- ext move not in the full list, taken: …`. Only an answer that cannot be read or played leaves the decision to the fallback engine (line `!!!` in the log).

> gnubg 1.08.003 swaps the score in `external` mode for move and double requests. MMbg compensates for this. The bug has been reported to the gnubg developers.

11. Long test runs

For strength comparisons the computer plays any number of matches against itself or against gnubg.

1. Choose engines, e.g. `A4E` (big against gnubg) or `A45` (big against deep).
2. Start mode **C4** (turbo): no board drawing, endless matches, one status line with intermediate results. **C3** runs with board but without pauses.

3. With the **log** on, every move goes into the log file in the log folder.
4. Stop with **any key**.

Aids for fair comparisons (options page):

- **U duplicate:** every match is played again with the same dice and the sides swapped. This evens out dice luck.
- **R dice seed:** a fixed seed for reproducible dice. 0 takes the seed from the clock.

Evaluation:

- At the end of a match the log writes `### MATCH` and `### total: games … matches …`.
- The title shows the match counter `M a:b`.
- Examples: on the PC3 an overnight run of 13 hours gave 117 matches big against gnubg 2-ply (46.2 %). On the PicoCalc with a Pico 2 W 450 matches ran in 8.6 hours (Part 2, P7). On the Colour Maximite 2, 404 matches big against big ran in 10.4 hours (chapter 15).

12. Options

12.1 Quick menu (key O)

```
1 anim+ 2 pips+ 3 hint- 4 flip+ 5 swap+ 8 mouse+ 9 mat+ 0 sound-  6 save 7 more
```

A key toggles the option; `+` means on, `-` off.

Key	Option
1	animation of the moves
2	pips in the title
3	hint (best move, cube recommendation in the log)
4	mirror the board (home board on the left)
5	swap colours (O red, X light)
8	mouse on/off (PC3, CMM2, PC1)
9	`.mat` recording (takes effect from the next match)
0	sounds: dice, moving, landing, hitting (only with animation, needs `OPTION AUDIO`)
6	save settings (`mmbg_pc3.ini`)
7	options page

12.2 Options page (O → 7)

The page shows **all** settings with their values. A key plus ENTER changes a setting, ENTER alone returns to the game.

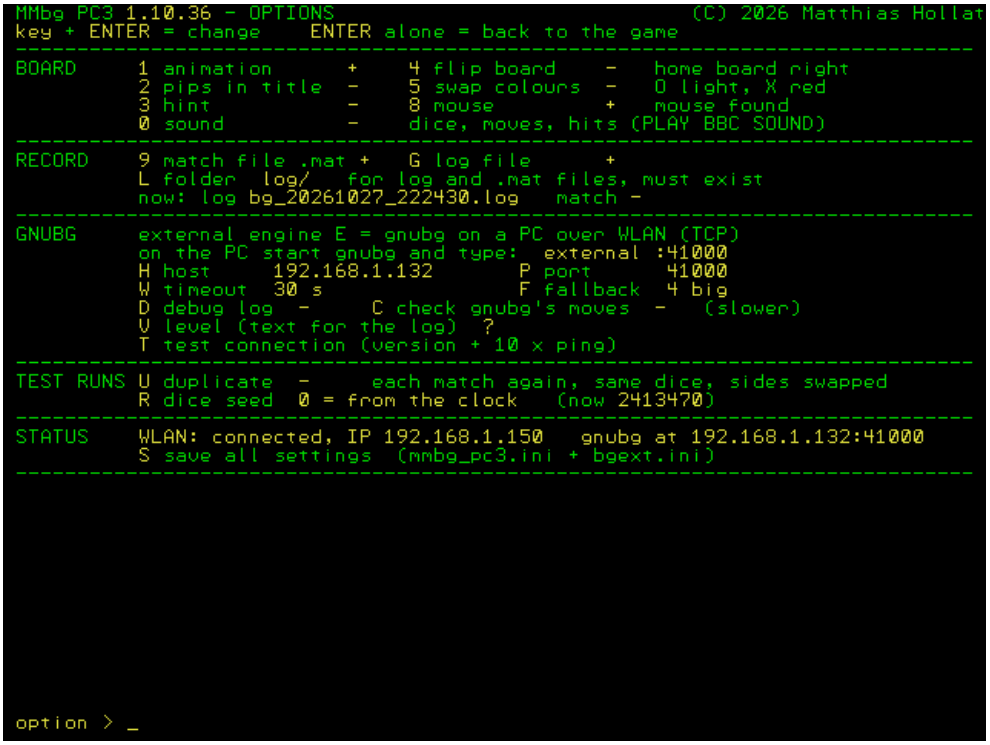


Figure 10: The options page (PC3). Logs and `.mat` are in the folder `log/`, gnubg runs on the PC 192.168.1.132.

Section	Keys
BOARD	1–5, 8, 0 as in the quick menu
RECORD	9 `.mat`, G log on/off, L log folder (chapter 3.4)
GNUBG	H host, P port, W timeout, F fallback, D debug log, C check, V level, T test
TEST RUNS	U duplicate, R dice seed
STATUS	WLAN state; S saves everything (`mmbg_pc3.ini` and `bgext.ini`)

13. Command summary

The key **H** shows all commands and, below under ACTIVE, what is currently set:



Figure 11: The help page with the ACTIVE overview (HDMI; it looks the same in the terminal).

Command	Effect
`13/7 8/7`, `13 24`, `#3`	move (chapter 5.2)
`E` / `U`	end turn / undo
`R` `D` `T` `P`	roll, double, take, pass (at the cube questions)
`N` / `M7`	new match / match to 7
`K`	cube on/off
`SG` / `LG`	save / load match
paste a GNUbg ID	take over the position
`C0` … `C4`	computer mode
`A45`, `A4E`, `C4E`	engines per side (plus turbo)
`L`	full move list
`D`	board dump with GNUbg ID (chapter 7.6)
`P`	DEV column (terminal)
`O`	options
`G` / `GN`	log on/off / new log file
`Print Screen`, `Ctrl+\`	screenshot (firmware, chapter 7.7)

Command	Effect
`H`	help with ACTIVE overview
`Q`	quit

14. Files

All files are in the program folder; the log folder is `log/` below it unless set otherwise (chapter 3.4).

File	Location	Written by	Contents
`mmbg_pc3.ini` (`mmbg_pc1.ini`, `mmbg_pica.ini`)	program folder	O → 6 or S	interface: anim, pips, hint, flip, swap, mouse, mat, sound, log, match, cube, computer, engines
`bgext.ini`	program folder	options page S	gnubg/WLAN, log folder (`logdir=`), duplicate, seed
`mmbg_pc3.sav` (`mmbg_pc1.sav`, `mmbg_pica.sav`)	program folder	automatic, SG	current match (move history, `.mat` state)
`bg_YYYYMMDD_HHMMSS.log`	log folder	log on	every move, cube, results
`mmbg_YYYYMMDD_HHMMSS.mat`	log folder	mat on	match for gnubg
`SCREEN_YYYYMMDD_HHMMSS.BMP` / `.PNG`	root of the SD card	Print Screen, Ctrl+\ (firmware)	screenshot
`bglog_err.txt`	program folder	on problems	write errors of the log

Contents of the package

Path	Contents
`mmbg/`	the three programs, nets, database, ini files – copy this folder to the SD card
`mmbg/log/`	empty log folder
`gnubg_start_PC/`	batch files to start gnubg on a Windows PC (0-, 1-, 2-ply)
`MMbg_Handbuch.pdf`, `MMbg_Manual.pdf`	manual (German, English)
`LICENSE.txt`, `README.txt`	licence, short instructions

15. MMbg on the Colour Maximite 2

Since version 1.10.34 **the same program** `mmbg_pc3.bas` also runs on the **Colour Maximite 2 (CMM2)**. This is made possible by the CMM2's firmware 7: it brings MMBasic up to the level of the PicoMite and, with `OPTION PICOMITE`, can behave like a PicoMite with HDMI. MMbg has been tested on a Generation 2 CMM2.

15.1 Requirements

What	Requirement
Computer	**Colour Maximite 2** (tested: Generation 2)
Firmware	**MMBasic 7.00.00b3 or later** (`OPTION PICOMITE` exists only from b3), tested with 7.00.00b4
Compatibility	`OPTION PICOMITE HDMI`
Display	VGA, 640 × 480 (MODE 3 as on the PC3)
Keyboard	USB keyboard in the **upper** USB socket at the back
Mouse (optional)	dual-mode mouse (USB/PS/2) in the **lower** USB socket at the back, with `OPTION MOUSE 0`
Terminal (optional)	USB type B socket (power and console), as on the PC3 80 × 40
WLAN	none – engine E (gnubg) is not available on the CMM2

The lower USB socket of the CMM2 is not a real USB socket: the mouse runs there in **PS/2 mode**. It therefore has to be a mouse that supports both modes, for example the Microsoft Basic Optical Mouse.

15.2 Setting up

Once at the prompt:

```
OPTION PICOMITE HDMI
OPTION MOUSE 0          (only with a mouse in the lower socket)
```

`OPTION LIST` then shows, among other things:

```
480MHz Colour Maximite 2 G2
MMBasic 7.00.00b4
OPTION PICOMITE PicoMiteHDMIUSB RP2350B, 640x480
```

The CMM2 firmware is installed via the upper USB socket and a USB A-to-A cable; this is described in the CMM2 manual (Appendix K).

15.3 Start

As on the PC3, the program folder is on the SD card, on the CMM2 at `A:/mmbg` (chapter 3.2). Start like this:

```
CHDIR "A:/mmbg"
RUN "mmbg_pc3.bas"
```

Or conveniently via the **file manager FM**: change to the program folder and start `mmbg_pc3.bas`.

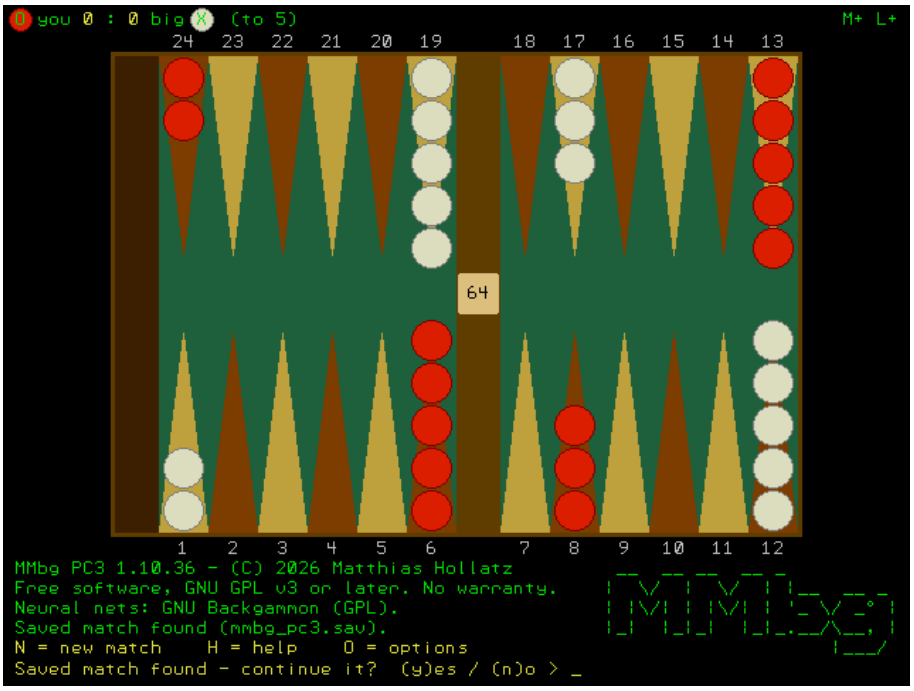


Figure 12: MMbg on the Colour Maximite 2 (screenshot): start with a saved match.

15.4 What is different

- **Enter key:** the CMM2 sends two characters for Enter (CR and LF), also over the terminal. MMbg sets `OPTION CRLF CR` at start-up and skips a left-over LF, so a running computer is not stopped by it.
- **Mouse:** MMbg finds the PS/2 mouse by itself (mouse channel 2). Without a mouse set up, you simply play with the keyboard.
- **No WLAN:** `[WLAN]` does not appear in the title; engine E and the connection test are missing. Everything else – engines 0 to 5, cube, undo, saving, `.mat`, log, test runs – is the same.
- **Speed:** the CMM2 (480 MHz) calculates about as fast as the PC3 (315 MHz): *big* needs ~0.6 s per move on average, *deep* ~10 s. A turbo run C4 big against big over 10.4 hours played 404 matches (~92 s per 5-point match) without a single error.

16. MMbg PC1: HDMI and VGA without PSRAM

`mmbg_pc1.bas` is MMbg for boards **without PSRAM**, for example the **PicoMite HDMI/USB board** (here "PC1" for short) or a **Pico 2 with VGA reference board**. The interface is that of the PC3: large board at 640 × 480, terminal, mouse, undo, saving, `.mat`, log, test runs, all engines up to *big* and *deep*. Only WLAN is missing, and with it engine E (gnubg).

The real task was memory: without PSRAM the big nets (512 KB as floating point) do not fit into RAM. MMbg PC1 therefore puts them, with 16 bits per weight, into the **MODBUFF**, a fixed area in flash, and reads them directly from there while computing. This is the same technique as on the PicoCalc (Part 2, P6 and P7) and plays just as strong.

16.1 Requirements

What	Requirement
Computer	RP2350 with HDMI or VGA, no PSRAM (tested: PicoMite HDMI/USB board with RP2350A, Pico 2 with VGA reference board)
Firmware	PicoMite **HDMI** or **VGA** V7.00.00 (tested with RC6)
MODBUFF	`OPTION MODBUFF ENABLE 192` (see 16.2)
Screen	MODE 3, 640 × 480, 16 colours
Keyboard	USB keyboard (HDMI board) or the keyboard of the VGA board, plus the serial terminal
Mouse (optional)	USB mouse on the HDMI/USB board
WLAN	none – engine E and the connection test are missing

16.2 Setting up: MODBUFF 192

Once at the prompt:

```
OPTION MODBUFF ENABLE 192
```

> **Caution:** every change of this option **formats drive A:**. Save whatever is there first. The program folder on the SD card (B:) is not touched.

The file `bgnet16.mb` (144 192 bytes) needs a MODBUFF of at least **152 KB**. **192 KB** are recommended, because most MOD music files need that much as well. With the default of 128 KB the firmware reports "File too large for modbuffer", and MMbg plays with the small nets only (engine 3).

You do not have to fill the MODBUFF yourself: at start-up MMbg checks its contents with the check sums from `bgpk16.dat`. If they do not match, e.g. at the first start, MMbg loads `bgnet16.mb` from the program folder into it by itself (`FLASH MODBUFF LOAD`, 1–2 seconds). Only then is the flash written.

> **Firmware update:** at the first start after flashing, the PicoMiteHDMI and PicoMiteRP2350VGA builds (from RC6) reset all options and clear flash and program memory. Save first with `OPTION DISK SAVE`, afterwards set `OPTION MODBUFF ENABLE 192` again.

Example for a Pico 2 with VGA reference board (`OPTION LIST`):

```
PicoMiteVGA MMBasic RP2350A Edition V7.00.00RC6
OPTION RESOLUTION 640x480 @ 315000KHz
OPTION SDCARD GP13, GP10, GP11, GP12
OPTION COMPILE ON
OPTION MODBUFF ENABLE  192
```

`OPTION COMPILE ON` works with MMbg PC1.

16.3 Start

```
CHDIR "B:/mmbg"
LOAD "mmbg_pc1.bas"
RUN
```

After the small nets the terminal shows the lines `Checking big nets in the MODBUFF ...` and, only the first time, `Loading bgnet16.mb into the MODBUFF (flash) ...`. Then MMbg reports `Big nets in the MODBUFF (flash), engine 4.` and the free RAM.

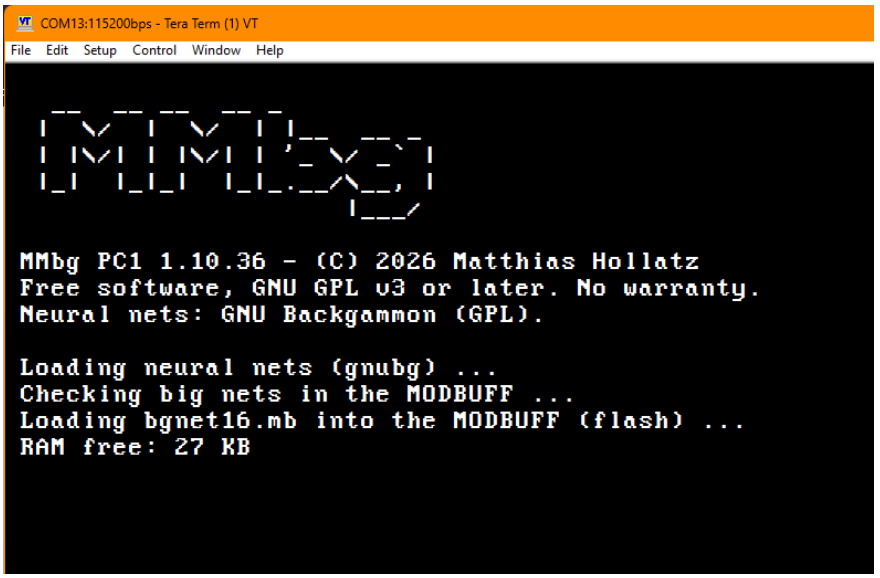


Figure 13: First start of MMbg PC1 in the terminal: the big nets are checked in the MODBUFF and loaded into it once.

If `RUN` reports "Not enough memory", `OPTION SYMBOLS OFF` as the first program line helps: it saves 3 to 5 KB, but MMBasic then finds variables only half as fast, and the computer thinks twice as long. With firmware RC6 this was not needed in the tests.

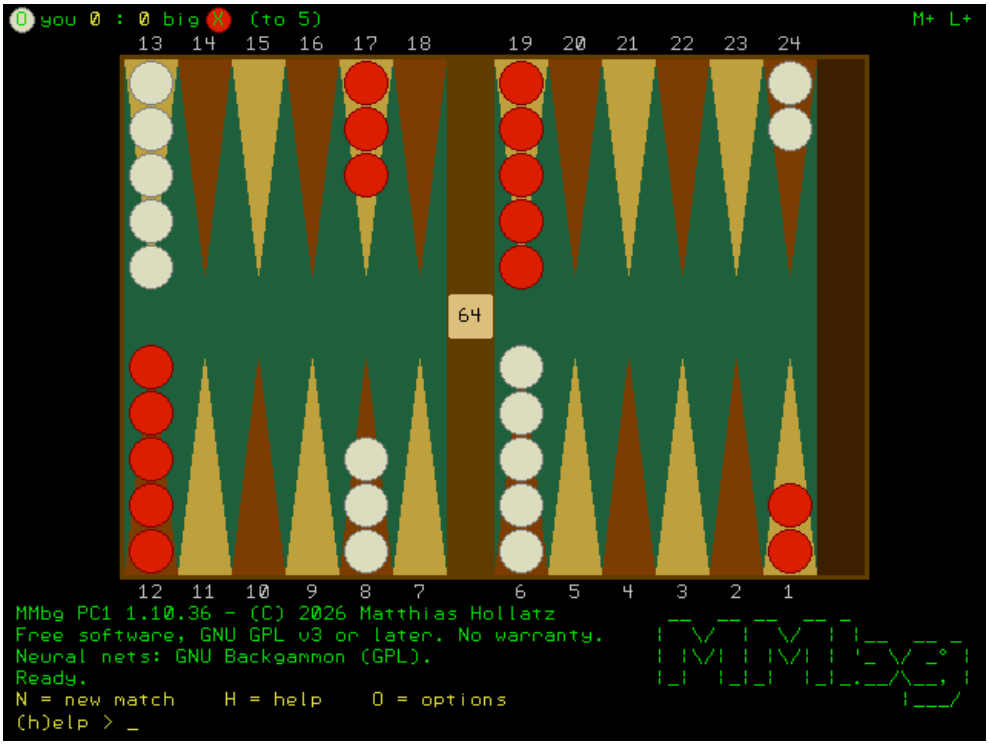


Figure 14: MMbg PC1 on a Pico 2 with VGA reference board (screenshot) – the same interface as on the PC3.

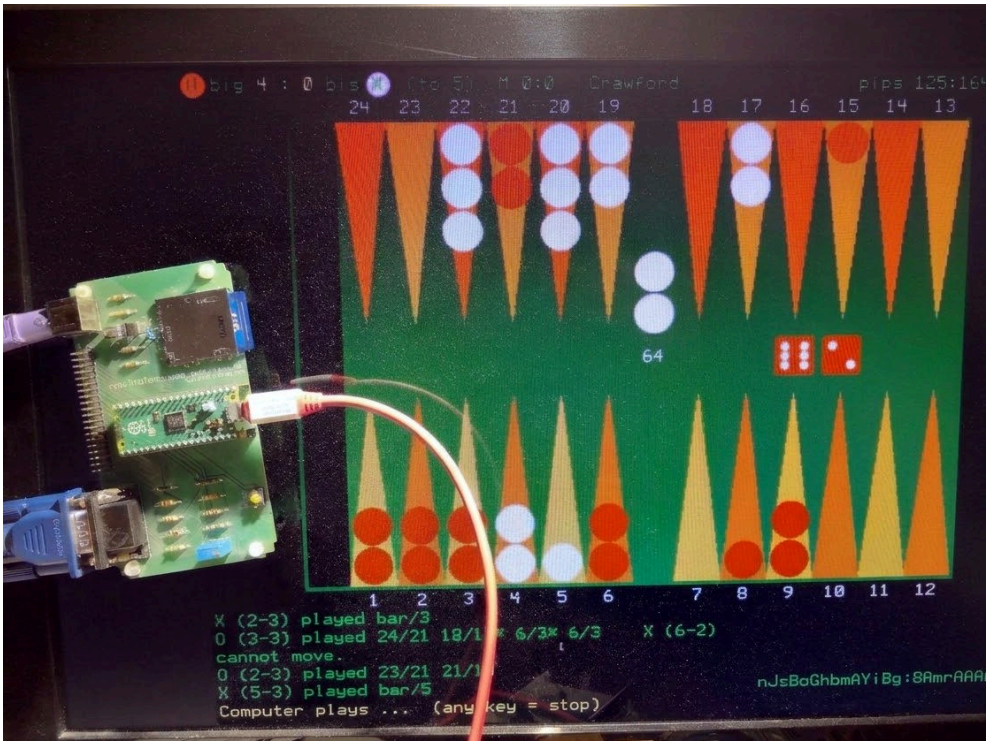


Figure 15: The VGA reference board with Pico 2 at work (big against big, Crawford game).

16.4 What is different from the PC3

	PC3	PC1 / VGA
Big nets	floating point in PSRAM	16 bit in the MODBUFF (flash), read directly
Small nets	floating point	first layer packed with 16 bits
Animation	with frame buffer	**VGA:** frame buffer F without using RAM, smooth as on the PC3. **HDMI:** no room for the frame buffer (it would need 150 KB of RAM), the field under the flying checker is redrawn; it looks the same, the mouse pointer may flicker slightly
Move list	500 moves	300 moves (only a few doubles have more, → 10.2)
Undo	300 turn starts	the last 40
`.mat`	400 actions per game	120 (enough for normal games; in very long games the rest is missing from the `.mat` file)
Log rank in the move list	all moves	the first 20
WLAN, engine E	yes	no; the options page has no GNUBG section
Help	–	an extra line `RAM … free now, lowest while thinking …`

Measured on the HDMI/USB board (RP2350A): *big* needs ~0.4 s per move on average. While computing, ~16 KB of RAM remain free.

17. Notes and limits

- **Which edition:** `mmbg_pc3.bas` for the PC3 and the Colour Maximite 2 (chapter 15), `mmbg_pc1.bas` for HDMI/VGA without PSRAM (chapter 16), `mmbg_pica.bas` for the PicoCalc (Part 2).
- **Large programs:** a program that exceeds the program memory when loaded without `, C` can damage the flash in the release candidates ("Program damaged: symbol table"). The published programs are small enough; loading with `, C` leaves extra room.
- **Stopping with Ctrl+C:** the colour palette stays changed; `MAP RESET` restores it. On a normal exit with `Q` this happens automatically.
- **Animation:** MMbg restores the background of the flying checker from a framebuffer (~150 KB PSRAM). Without the buffer it redraws the area piece by piece; that is slower but looks the same.
- **Sounds** are only available with animation (C0 to C2). Without `OPTION AUDIO` the option stays off and the log reports "no audio output". The sounds (dice, moving, landing, hit) are built from noise. Anyone who finds better settings, such as other envelopes or volumes, is welcome to report them in the TheBackShed forum. The values are in the source code in `GuSnd` and `GuSndOn`.
- **Match play only** up to 15 points, no money game. Undo only works within the current game.

- **A pasted GNUbg ID** ends the `.mat` recording for this match, because an arbitrary position cannot be represented in the `.mat` format.

18. A flight through the source code

For everyone who wants to look into `mmbg_pc3.bas` or change it: a short tour. The program has about 6 300 lines and 263 routines. MMbg PC1 and MMbg PiCa are generated from it by scripts that replace exactly defined places (other table sizes, nets in the MODBUFF, LCD interface); the routines keep their names there. The names tell the area:

Prefix	Area
`Bg…`	game: rules, input, move generator, evaluation, cube, match, log, saving
`Nn…` , `Z…` , `Bo…`	small nets, big nets, bear-off database
`Ex…`	gnubg over WLAN (TCP), options page
`Gu…`	graphics, mouse, options, sounds
`Gid…` , `BgId…`	GNUbg ID
`BgMat…`	`.mat` recording

18.1 Start-up

1. **Global variables** (`DIM …`): board `bd(0..25)` (+n = O checkers, −n = X checkers, `bd(0)` bar of O, `bd(25)` bar of X), dice `dw()` , move lists `gm$()` / `gv()` , net weights, screen state `yb()` .
2. `MODE 3` , `GuPal` (colours), `BgLogo` (start screen), `OPTION CRLF CR` (CMM2 only, skipped elsewhere).
3. **Loading:** `NnLoad` (small nets), `BoOpen` (bear-off database), `ExIni` (gnubg settings), `BgLogDir` (create the log folder), then `ZLoad` (big nets, only if the memory is available in one piece).
4. `GuInit` (framebuffer, colours), `GuCfgLoad` (`mmbg_pc3.ini`), `GuMsInit` (mouse: USB first, otherwise PS/2 on channel 2), question about the saved match (`BgLoadG`).

18.2 The main loop

```
DO
  BgShow                draw terminal and screen (computes the move list)
  Input:    computer -> best move gm$(1)   |   human -> GuIn$ (keyboard, mouse)
  Command?  H, O, N, M7, C1, A45, SG, LG, U, GNUbg ID ...   -> execute
  Move?     BgInput -> BgTry -> BgMove      (check and move)
            GuAnim -> GuMv -> GuFly        (animation, sounds)
            BgNext -> BgAuto -> BgTurn     (change sides, cube, roll)
  Game over? BgEndGame                     (points, log, .mat, next game)
LOOP
```

18.3 How the computer finds its move

1. `BgTurn` : first `BgCubeTurn` (cube before the roll; `BgCube` computes the match winning chances like gnubg 0-ply), then `BgRoll` .

2. `BgPrep` calls the **move generator** `BgGen`: it tries all dice orders and start points with its own stack instead of recursion (the stack only allows a few levels). `BgMx` first checks how many dice can be played at most.
3. Every finished move lands in `BgLeaf`; duplicate final positions are dropped. `BgEval` rates the position: formula (engines 0–2) or net via `NnEval` → `NnFwd` (small), `ZFwd` (big), `NnBo` (bear-off database).
4. `BgSort` sorts, `ZBest` recalculates the best 12 with the big net, `ZThink` looks one roll ahead for *deep*, `BgBook` puts the book move on top when needed, `ExMove` asks gnubg (`ExMvKey$` plays gnubg's answer on trial and finds it in the list, or adds it if the list is full).
5. Place 1 of the list is played.

18.4 Further building blocks

Routines	Task
`BgGid$`, `BgIdGet`, `BgIdPut`	create, check and take over a GNUbg ID
`BgHist`, `BgBack`	move history (one ID per turn start) and undo
`BgSaveG`, `BgLoadG`	write and read the saved match
`BgMatNew`, `BgMatTurn`, `BgMatGame`	`.mat` in gnubg's format
`BgLog`, `BgLogTurn`, `BgLogDir`	log file, one line per move, log folder
`ExAsk$`, `ExBoard$`, `ExCube`, `ExMove`	conversation with gnubg over TCP (FIBS board line)
`GuFull`, `GuDraw`, `GuPt`, `GuDice`	redraw the screen or only what changed
`GuFly`, `GuNear`, `GuRep`	flying checker, mouse pointer, fallback without framebuffer
`GuHit$`, `GuShort$`	translate a mouse click or short input into a move
`GuSnd`, `GuSndOn`	sounds with `PLAY BBC SOUND`
`BgHelp`, `ExMenu`, `GuOpt`	help, options page, quick menu

Part 2 – MMbg PiCa (PicoCalc)

> **New in 1.10.36:** the big nets now live in the **MODBUFF** (flash) as with MMbg PC1, no longer in a row buffer in RAM with reloading from the SD card. So move list, undo and `.mat` are as large as on the PC3. Also new: screenshots by the firmware and shorter prompts.

Part 2 describes only **what is different on the PicoCalc**. For the game itself – moving, cube, undo, saving, `.mat`, log, GNUbg ID, engines, gnubg over WLAN and test runs – Part 1 applies unchanged; the references (e.g. "→ 5.2") point to the chapters there.

P1. MMbg PiCa at a glance



Figure 16: MMbg PiCa on the PicoCalc (Pico 2 W), computer against computer.

- **One program** (`mmbg_pica.bas`) for the PicoCalc with a **Pico 2** (firmware PicoMite RP2350) or a **Pico 2 W** (firmware WebMite RP2350). It recognises the firmware at start-up by itself.
- **The same game as on the PC3:**
  - all engines up to *big* and *deep* (→ 8.1),
  - doubling cube, Crawford, saving and resuming, multi-level undo, GNUbg IDs, `.mat` recording, log, sounds, screenshots (→ 5–7),
  - computer modes C2 to C4 and long test runs (→ 8.3, 11),
  - gnubg as opponent over WLAN and the terminal over telnet (Pico 2 W only, → 10).
- **The same playing strength as the PC3:** the nets are the same gnubg weights, only stored with 16 bits; nothing measurable is lost (P7). Since 1.10.36 the move list is 500 moves long as on the PC3, so the PiCa chooses from exactly the same moves (P6.3).
- Needs **no PSRAM**: the big nets live in the MODBUFF (P6). Required once: `OPTION MODBUFF ENABLE 192`.

P2. Requirements and firmware set-up

What	Requirement
Computer	PicoCalc with **Pico 2** (RP2350, without PSRAM) or **Pico 2 W**
Firmware	**PicoMite RP2350** or **WebMite RP2350**, **V7.00.00** (tested with RC6 on the Pico 2 W)
MODBUFF	`OPTION MODBUFF ENABLE 192` (P6)
SD card	drive B: for the program folder (program, nets, logs)
Terminal (optional)	VT100, 80 × 40, over USB serial or telnet (WebMite)
WLAN (optional)	Pico 2 W only: gnubg as opponent (→ 10), terminal over telnet

Set up the firmware once at the prompt:

```
OPTION RESET PICOCALC               (LCD, keyboard, SD card, sound)
OPTION COMPILE ON                   (important for speed)
OPTION MODBUFF ENABLE 192            (formats A: - save it first!)
OPTION PRINTSCREEN FORMAT PNG        (if screenshots as PNG are wanted)
OPTION WIFI "MyNetwork", "Password" (Pico 2 W only)
OPTION TELNET CONSOLE ON            (Pico 2 W only, if wanted)
```

P3. Installation and start

P3.1 One folder for everything

As on the PC3 (→ 3.1), all files live in **one folder**, for example `B:/mmbg`; name, drive and depth are free. The package contains the files for all devices. Each edition takes only what it needs (P8); the same folder therefore works everywhere.

Folder	Contents
`mmbg` (any name)	programs, nets, database, ini files, saved match
`mmbg/log`	logs and `.mat` files; MMbg creates the folder at start-up (→ 3.4)

P3.2 Start

```
CHDIR "B:/mmbg"
LOAD "mmbg_pica.bas", C
RUN
```

On the PicoCalc **`, C` is needed**: without the comments enough program memory remains.

The start screen shows, one after another:

Line	Meaning
`Nets: bgnet345.dat (… ms).`	small nets loaded

Line	Meaning
`Checking big nets in the MODBUFF ...`	MMbg checks the big nets in the MODBUFF (terminal)
`Loading bgnet16.mb into the MODBUFF ...`	only at the first start or after a firmware update: `bgnet16.mb` is loaded into flash (1–2 s)
`Big net: MODBUFF (flash).`	big nets ready, engine 4
`Big net: MODBUFF 192 needed - engine 3.`	no MODBUFF set up: small nets only (P2)
`Bear-off db: bgbo15.dat (with gammons).`	endgame database
`Settings: mmbg_pica.ini.`	settings loaded
`RAM free: … KB.`	free memory after start-up

MMbg then waits for **N** (new match). If a match was saved, it asks whether to continue it (→ 6.2).



Figure 17: Left the start in the terminal (the big nets are loaded into the MODBUFF once), right the LCD with the short question before the roll.

P4. Screen, terminal and controls

P4.1 LCD (320 × 320)

- **Title line** (40 characters): players, score, match length and the switches as on the PC3 (→ 4.1): `W` WLAN, `M±` `.mat`, `L±` log.
- **Board:** point 23 px, checker 19 px, bar and tray. As on the PC3, the dice lie to the right of the player as seen from his side.
- **Log:** the last 4 messages of 40 characters.
- **Prompt:** at the very bottom, 40 characters. All questions are at most 39 characters long and therefore shorter than on the PC3, e.g. `O: (r)oll / (d)ouble to 2 >`, `X: O doubles to 2 (t)ake/(p)ass >` or `O wins match 5-2! ENTER=new >`. The meaning of the numbers is on the help page.

P4.2 Terminal

Over USB serial or telnet the terminal shows **the same ASCII board as on the PC3** (→ 4.2), 80 × 40, with move list, DEV column and GNUbg ID. MMbg switches the terminal to 80 × 40 at start-up if the terminal program allows it (Tera Term, PuTTY). Input works from the PicoCalc keyboard and from the terminal alike.

P4.3 Text pages (help, options, list, dump)

All text pages have **40 columns in Font 1 (8 × 12)**, 26 lines. Longer pages are split into **two pages**:

Page	Contents	Turning pages
H help 1/2	all commands (→ 13)	arrow down or any key → page 2, Q back
H status 2/2	the current state: match, cube, players, computer mode, options, log folder and file, `.mat` file, saved match, nets, bear-off database, WLAN, gnubg, GNUbg ID	arrow up → page 1, any other key back
O → 7 options 1/2	board and recording: 1–5, 0, 9, G, and L with the log folder in a line of its own	arrow down → page 2
O → 7 options 2/2	gnubg and test runs: H, P, W, F, D, C, V, T, U, R and the WLAN status	arrow up → page 1
L move list	20 moves per page	any key → more, Q back
D dump	board in two columns, dice, **GNUbg ID**, where the big net is (MODBUFF)	key → back

On the options page **all keys work on both pages**; after a change the page with the changed value is shown. **S** saves everything (`mmbg_pica.ini` and `bgext.ini`). The terminal shows the same pages, with the input line directly below them.

Copying a GNUbg ID by hand: key **D** and status page 2/2 show the ID on the LCD. In the terminal it is at the bottom right as on the PC3. An ID is pasted at the prompt as usual (→ 7.4).

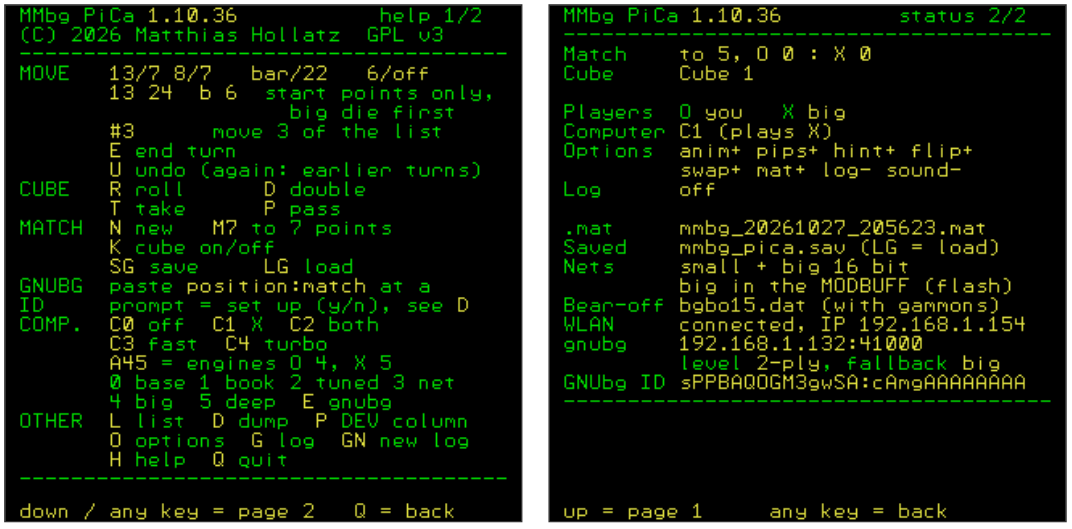


Figure 18: Text pages on the LCD: help 1/2 and status 2/2 (nets "big in the MODBUFF (flash)").

P4.4 Quick menu (key O)

As on the PC3 (→ 12.1), shown in the log area of the LCD and without the mouse switch: `1`–`5`, `9`, `0`, `6 save`, `7 more`.

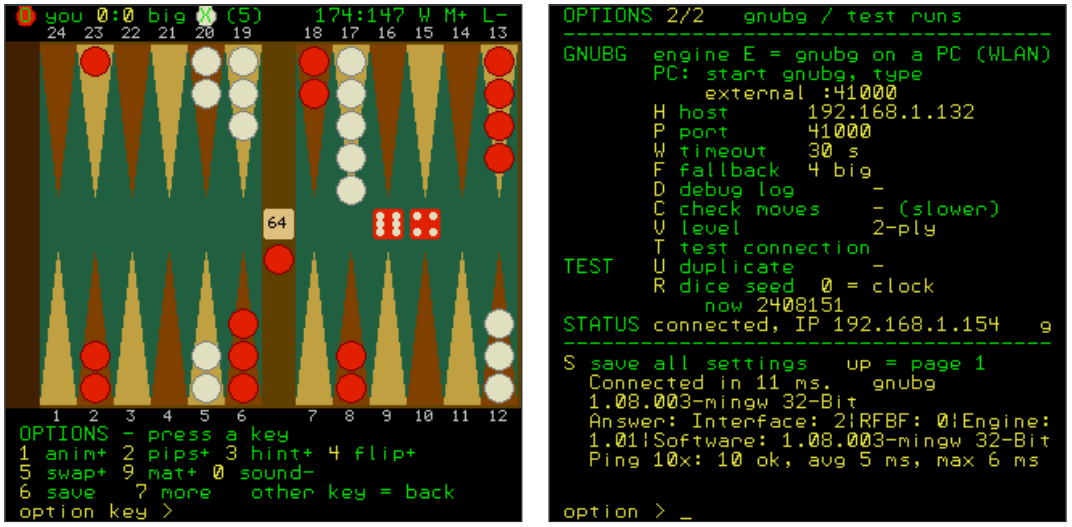


Figure 19: Left the quick menu (key O) in the log area, right the options page 2/2 after a successful connection test (Pico 2 W).

P4.5 Turbo mode C4 on the LCD

In turbo mode (→ 8.3, 11) MMbg switches the LCD **black** and the backlight to the **lowest level**, so that nothing stands still on the screen for hours. Any key stops the run; the backlight returns to its previous level and the board is redrawn.

The backlight is set by the keyboard controller of the PicoCalc, which accepts values from 16 to 240; MMbg uses `OPTION BACKLIGHT LCD`, which also stores the value in the flash. If a turbo run is ended with **Ctrl+C**, the restore is skipped: set the backlight again with `OPTION BACKLIGHT LCD n` (n from 16 to 240).

P5. Differences from the PC3

	PC3	PiCa
Screen	HDMI 640 × 480	LCD 320 × 320
Text pages	80 columns	40 columns, help and options on 2 pages each
Prompts	detailed	short, at most 39 characters (P4.1)
Mouse	yes	no
Screenshot	Print Screen, Ctrl+\ (→ 7.7)	the same, the picture has 320 × 320 pixels
Animation	with framebuffer (PSRAM)	without framebuffer, the area is redrawn; it looks the same
GNUbg ID	bottom right in the log	in the terminal, with D and on status page 2/2

	PC3	PiCa
Move list, undo, `.mat`, log rank	500 moves, 300 turn starts, 400 actions, all moves	**the same**
Big nets	floating point (8 bytes) in PSRAM	**16 bit in the MODBUFF** (flash), read directly (P6)
Small nets	floating point	first layer packed with 16 bits
Turbo C4	screen without board drawing	LCD dark, backlight lowest (P4.5)
WLAN / telnet	yes	Pico 2 W only

Everything else – game, engines, cube, saving, `.mat`, log and gnubg – is **the same**.

P6. Nets and memory on the PicoCalc

P6.1 The task

The big gnubg net has 250 inputs × 128 neurons = **32 000 weights**, and there are two of them (contact and crashed, → 9.2). As floating-point numbers (8 bytes each, as on the PC3) both together would need 512 KB. The PicoCalc has no PSRAM, and after loading MMbg only a few hundred KB of RAM remain.

P6.2 Packed weights

Each weight is stored as an unsigned 16-bit number `u` and converted back when calculating:

```
weight = lo + u × q
```

The scale applies **per input row**: `lo` is the smallest weight of the row, `q` = (largest − smallest) / 65 535. The rounding error per weight is at most q/2, typically 0.0003 to 0.0006 (P7). One net thus needs 64 000 bytes, both together with scales and escape tables **144 192 bytes**. These data are in `bgnet16.mb`, the check sums in `bgpk16.dat`.

P6.3 The nets in the MODBUFF

The **MODBUFF** is a fixed area in flash that the firmware reserves with `OPTION MODBUFF ENABLE 192`. MMbg stores `bgnet16.mb` there and reads the weights **directly from flash** while computing. None of it is in RAM.

- **At start-up** MMbg checks the contents with the check sums. If they do not match (first start, firmware newly flashed), it loads `bgnet16.mb` into it by itself with `FLASH MODBUFF LOAD` (1–2 s). Only then is the flash written.
- **While computing** the weights are stored row by row per input: one row holds the 128 weights from one input to all neurons (256 bytes). In a real position only about 50 of the 250 inputs are non-zero, so one evaluation needs only these ~50 rows. Each row is processed with three fast firmware commands:

```
MEMORY UNPACK (unpack the row from flash) -> MATH SCALE (× input × q) -> MATH C_ADD (add up)
```

- **Without MODBUFF** (or with only 128 KB) the start screen reports `Big net: MODBUFF 192 needed - engine 3.`; then the engines play with the small nets.

The computing time is ~0.5 s per move, as on the PC3. Because the nets no longer take up RAM, move list, undo and `.mat` are as large as on the PC3.

Exactly as strong as the PC3: until now the PiCa move list was limited to 300 moves. With some doubles there are more legal moves, and the rest was then missing from the selection. With 500 moves the PiCa now evaluates exactly the same candidates as the PC3 in every position, with the same pre-filter and the same nets. The only difference left is the 16-bit rounding of the weights, and that is far below anything that can be measured (P7.3).

> Up to version 1.10.35 the PiCa kept the nets in a **row buffer in RAM** and reloaded missing rows from the SD card. That worked, but left little RAM. The MODBUFF solution came up while developing MMbg PC1 (chapter 16); both editions now use the same net code.

P6.4 Small nets

The first layer of the small nets (contact, crashed, race: 200 × 16 and 200 × 8 weights) is packed with 16 bits as well, with one scale per net. It is kept in RAM. The second layer is small and stays floating point.

P7. The 16-bit nets: rounding and playing strength

P7.1 Where the rounding comes from

With 16 bits a weight can take 65 536 different values between the smallest and the largest weight **of its row**. Because every row has its own scale, the steps are fine: a typical row spans about 45 (contact net) to 80 (crashed net) units, so one step is about 0.0007 to 0.0012, and the rounding error of a weight is at most half a step – typically 0.0003 to 0.0006, in the widest row 0.0036. In one evaluation about 50 rows × 128 neurons are added up; the errors are random in sign and largely cancel out, and the S-curves of the hidden layer damp them further.

P7.2 How much does a position value change?

The 16-bit evaluation was checked against the full-precision evaluation of the PC3 with a Python model of MMbg's evaluation (inputs, both net sizes, move generator, pre-filter) and the logs of real runs on the PicoCalc:

Check	Result
1 014 positions of a run against gnubg 2-ply	deviation of the equity at most **0.0011**
2 274 positions of the overnight run	deviation at most **0.0012**
680 move decisions from 34 games	**679 identical moves**; once two moves were practically equal (difference 0.00002) and swapped places

A deviation of 0.001 equity only changes the move when two moves are almost exactly equal – and then it hardly matters which of the two is played.

P7.3 Estimated loss of playing strength

From the check above: one swapped decision in 680, costing 0.00002 equity, gives an average loss of about **0.00000003 equity per move** (3×10^{-8}). For comparison, the big net itself makes an average error of **0.0026 equity per move** (2.6 mEq) against gnubg 2-ply (→ 8.1). The rounding loss is therefore about **one hundred-thousandth** of the net's own error – far below anything that can be measured in games.

P7.4 Measured in play

An overnight run on a Pico 2 W (16 bit, big against gnubg 2-ply, 5-point matches, turbo C4):

	Pico 2 W, 16 bit	PC3, floating point
Matches won	**199 of 450 = 44.2 % ± 4.6 %**	577 of 1236 = 46.7 %
Games won	778 of 1631 = 47.7 %	–
Run time	8.6 h, ~69 s per match	–
Computing time per move	~0.5 s	~0.5 s

This was still measured with the row buffer (up to 1.10.35); the weights and the computation are the same as today in the MODBUFF. The difference of 2.5 percentage points is well within the statistical spread (about one standard deviation); with 450 matches a difference of this size is to be expected by chance alone. Together with P7.3: **the PicoCalc plays as strong as the PC3.**

P8. Files

File	PC3 / CMM2	PC1 / VGA	PicoCalc	Contents
`mmbg_pc3.bas`	✓			program PC3 and CMM2
`mmbg_pc1.bas`		✓		program PC1 (HDMI/VGA without PSRAM)
`mmbg_pica.bas`			✓	program PiCa
`bgnet345.dat`	✓	✓	✓	small nets (text)
`bgnet02.dat`	✓	✓	✓	output layer of the big nets
`bgnet0.bin`, `bgnet2.bin`, `bgesc.bin`	✓			big nets, floating point
`bgnet16.mb`, `bgpk16.dat`		✓	✓	big nets 16 bit and escape tables for the MODBUFF, check sums
`bgbo15.dat`	✓	✓	✓	bear-off database
`bgnet35.dat`, `bgbo10.dat`	✓	✓	✓	fallbacks (→ 3.1)

The earlier PiCa net files (`bgnet0.pk`, `bgnet2.pk`, `bgpk.dat`, `*.p16`, `*.s16`, `bgesc.pk`) are no longer needed and can be deleted.

Created by MMbg itself:

- in the program folder: `mmbg_pica.ini` (interface, with `log=`), `bgext.ini` (gnubg, `logdir=`, test runs; shared with the PC3), `mmbg_pica.sav` (saved match);
- in `log/`: `bg_YYYYMMDD_HHMMSS.log` and `mmbg_YYYYMMDD_HHMMSS.mat`.

P9. Notes and limits

- **Image retention on the LCD:** the PicoCalc panel can keep a faint after-image of a picture that stood unchanged for a long time, such as text at the prompt. It shows later as a shadow on dark colours, for example on the felt of the board. It fades again: switch off, or show a bright area for a while (`BOX 0,0,320,320,0,RGB(WHITE),RGB(WHITE)`). Turbo mode C4 darkens the screen for this reason (P4.5).
- **Dice:** in normal play they come from `RND`; on the RP2350 the firmware keeps reseeding it from the hardware random number generator. Only with **duplicate** (option U, → 11) are the dice deliberately repeatable from a seed. Without an own value (option R) the seed comes from the clock, which on the Pico 2 starts at 01-01-2024 after every power-on. For real test runs, set a seed.
- **Terminal size:** the automatic switch to 80 × 40 only works if the terminal program allows it. Otherwise set it by hand.
- **Telnet:** according to the firmware release notes, RC4 fixes errors in the telnet output. If characters are missing in a telnet terminal, this is the likely reason.
- **Ctrl+C in turbo mode:** see P4.5 (backlight).
- **Error messages with a line number** refer to the program without comment lines when it was loaded with `, C`.

Licence and thanks

MMbg is **free software** under the **GNU General Public License v3 or later**: it may be used, passed on and modified, also commercially. Whoever passes it on must do so under the GPL again, with source code. There is no warranty.

```
MMbg - Backgammon for MMBasic
Copyright (C) 2026 Matthias Hollatz - Soli Deo gloria
Neural network weights derived from GNU Backgammon,
(C) the GNU Backgammon authors (GPL v3 or later).
```

Thanks to

- **Geoff Graham**, the creator of MMBasic and the Maximite,
- **Peter Mather** for the PicoMite and WebMite firmware, the firmware for the PC3, the PicoCalc and the Colour Maximite 2, and the BBC sound engine,
- the authors of **GNU Backgammon** for nets, methods and testing via `external`,
- the community in the **TheBackShed forum**.

Have fun playing!