mmsql 1.1 – User Manual
SQLite manager for the Colour Maximite 2

© 2026 Matthias Hollatz – MIT License (see LICENSE.txt)

This manual describes mmsql 1.1 for the Colour Maximite 2 with firmware 8. It is for everyone who works with mmsql, from the first try to exchanging databases with a PC. If you do not know SQL yet, start with the tutorial (TUTORIAL_mmsql.pdf) and use this manual for reference.

Contents

1. Introduction
2. Requirements and installation
3. Getting started
4. The line editor and the keys
5. SQL with mmsql
6. Command reference
7. Options (.set)
8. Output, export and import
9. Limits
10. Messages and hints
11. Firmware notes
12. Appendix

1. Introduction

mmsql is a manager for SQLite databases, written in MMBasic for the Colour Maximite 2 (CMM2). Since firmware 7 the CMM2 contains an SQLite database that BASIC programs can use with `SQL OPEN`, `SQL EXEC`, `SQL QUERY` and so on. mmsql makes these databases accessible directly on the CMM2: look at them, query, change, check, repair, export and import.

mmsql works like the command line tool `sqlite3` on a PC:

- Commands for mmsql itself start with a **dot**, e.g. `.tables` or `.help`.
- Anything else is **SQL** and goes straight to the database, e.g. `SELECT * FROM computer`.

mmsql runs on the CMM2's screen and on a serial terminal (e.g. Tera Term) at the same time. Both show the same, and you can type on either.

Databases that mmsql creates or changes are normal SQLite files. They can be opened on a PC with `sqlite3`, *DB Browser for SQLite* or any other SQLite tool, and the other way round.

Licence: mmsql is under the MIT License. SQLite itself is in the public domain.

2. Requirements and installation

Requirements

Computer	Colour Maximite 2, generation 1, 2 or 3
Firmware	MMBasic **8.00.00b1** or later
SQLite module	`A:/sql/sql.mod` from the firmware package
For firmware 7.00.00RC6	mmsql **1.0.1** (own version, without the new features of 1.1)

If mmsql 1.1 is started on older firmware, it shows a hint to use mmsql 1.0.1 and ends.

Files

File	Purpose
`mmsql.bas`	the program
`mmsql_en.hlp`	help texts, English
`mmsql_de.hlp`	help texts, German
`LICENSE.txt`	MIT License
`README.md`, `TUTORIAL_mmsql.pdf`, `demo.sql`	short description, tutorial, sample data

All files go into the same folder, e.g. `A:/sql/`. mmsql creates two more files in this folder:

File	Contents
`mmsql.cfg`	the options (`.set`), as a readable text file
`mmsql.his`	the history of the commands typed

Both may be deleted; mmsql then starts with the default values or an empty history.

Transfer to the CMM2

The easiest way is XMODEM over the serial cable, e.g. in Tera Term (*File* → *Transfer* → *XMODEM* → *Send*), on the CMM2:

```
chdir "a:/sql"
xmodem receive "mmsql.bas"
```

Repeat for each file. Or write the files to the SD card on a PC.

3. Getting started

Starting

```
RUN "mmsql"
RUN "mmsql", "A:/sql/computers.db"
```

Without a file mmsql starts without a database; with a file it opens it at once. At the start mmsql shows a banner:

```
mmsql V1.1 - SQLite manager for the Colour Maximite 2 (G2)
(c) 2026 Matthias Hollatz - MIT License
Type .help for help
mmsql>
```

In brackets is the generation of the CMM2 (G1, G2 or G3; “G2/G3” under `OPTION PICOMITE`, because the firmware then does not report the G3).

The prompt

Prompt	Meaning
`mmsql>`	no database open
`mmsql[club.db]>`	the database `club.db` is open
`mmsql[club.db]*>`	a transaction is open (after `BEGIN`), see chapter 5

Commands and SQL

```
mmsql> .open club.db
Opened: club.db
mmsql[club.db]> .tables
name             type  ncol
-------------------------
member           table 4
members_per_city view  2
owns             table 3
 3 row(s)
mmsql[club.db]> SELECT name, city FROM member WHERE joined < 2021
```

If you forget the dot of an mmsql command, mmsql helps:

```
mmsql> open computers.db
Did you mean .open? Commands start with a dot.
```

This also works for a comma instead of the dot (`,help`).

Help

Input	shows
`.help`	the overview of all commands in groups
`.help command`	details of one command, e.g. `.help import`
`.help keys`	the keys of the line editor
`.help sql`	notes on SQL in mmsql
`.help utf8`	how umlauts and other UTF-8 characters are shown

The help texts are in `mmsql_en.hlp` and `mmsql_de.hlp`. Switch with `.set helpfile mmsql_de.hlp`.

Quitting

`.quit` (or `.q`, `.exit`) closes the database and ends mmsql. If a transaction is open, mmsql asks first (see `.close`).

4. The line editor and the keys

Every input is edited in a line editor, on the screen and on the terminal alike.

Key	Action
← →	move the cursor
Home / End	to the start / end of the line
Backspace	delete the character left of the cursor
Del	delete the character at the cursor
Esc	clear the line
↑ ↓	earlier inputs from the history
Enter	run the input

Lines longer than the screen scroll sideways. An input is at most **255 characters** long (the length of an MMBasic string).

History: the last inputs (default 50, option `history`) are saved in `mmsql.his` and can be recalled with ↑, also after a restart.

Paging through results (`.browse`, queries, long output):

Key	Action
Enter or N	next page
B	previous page (with `.browse` and queries)
Q or Esc	stop

Ctrl-C

- **While an SQL statement runs**, Ctrl-C stops it. mmsql goes on and tells you exactly what SQLite has undone:

```
Interrupted (Ctrl-C).
The statement was undone. Your transaction is still open (COMMIT or ROLLBACK).
```

If the interrupted statement was a change (INSERT, UPDATE, DELETE …) inside a transaction, SQLite undoes the **whole transaction**; mmsql then says “The whole transaction since BEGIN was undone.” An interrupted query (SELECT) leaves the transaction open.

- **Everywhere else** (at the prompt, while writing an export file) Ctrl-C ends mmsql like any BASIC program. An open transaction is then undone; nothing of it is written. An incomplete export file is left on the card.
- With `OPTION BREAK 0` a running SQL statement cannot be stopped; mmsql shows a hint at the start. With `OPTION BREAK 4`, Ctrl-D takes the part of Ctrl-C.

Umlauts and UTF-8

SQLite stores text as UTF-8. The CMM2 font has no umlauts, so mmsql shows each UTF-8 character (e.g. ä, ß, €) as **one** placeholder: by default code 250, a middle dot (in Tera Term it appears as `ú`). This keeps columns straight, and the line editor treats such a character as one.

Setting	Effect
`.set utf8 dot`	placeholder (default)
`.set utf8 raw`	send the bytes unchanged – useful when the terminal is set to UTF-8
`.set dotchar 46`	another placeholder, here the dot (code 33 to 255)

Files (`.once`, `.output`, `.dump`) and the database always hold real UTF-8. Umlauts cannot be typed in mmsql yet.

5. SQL with mmsql

Statements

Anything that does not start with a dot is SQL. A semicolon at the end is allowed but not needed.

- Statements that return data (`SELECT`, `WITH`, `VALUES`, `PRAGMA`) show the result in the current output mode (default: table, page by page).
- All others answer `OK`; `INSERT`, `UPDATE`, `DELETE` and `REPLACE` also give the number of changed rows, and `INSERT` the new rowid:

```
mmsql[test.db]> INSERT INTO t VALUES(1)
OK, 1 row(s) changed, last ID 1
```

Several statements in one line

Several statements are separated by `;` and run **one after the other**, each with its own result – as in `sqlite3`:

```
mmsql[test.db]> CREATE TABLE t(a); INSERT INTO t VALUES(1); SELECT * FROM t
OK
OK, 1 row(s) changed, last ID 1
a
-
1
 1 row(s)
```

- The first error (or Ctrl-C) stops the rest of the line: “Not run: n more statement(s) of this line.”
- A `;` inside texts (`'a;b'`), quoted names (`"a;b"`, `` `a;b` ``, `[a;b]`) and comments (`/* … */`) does not split. From `--` on, the rest of the line is a comment.
- `CREATE TRIGGER … BEGIN …; …; END` stays in one piece, because its body has semicolons of its own.
- At most 32 statements per line.

Transactions

A transaction puts several changes together: either all of them or none. With many INSERTs it is also much faster, because the SD card is written only once.

```
BEGIN
INSERT INTO ...
INSERT INTO ...
COMMIT
```

`COMMIT` (or `END`) writes the changes, `ROLLBACK` drops them. `SAVEPOINT`, `RELEASE` and `ROLLBACK TO` work as well.

While a transaction is open, the prompt shows a **star**: `mmsql[test.db]*>`. mmsql follows `BEGIN`, `SAVEPOINT`, `COMMIT`, `END` and `ROLLBACK` for this. After errors, Ctrl-C and `RELEASE` mmsql asks SQLite whether the transaction is still open, and tells you when SQLite has undone it (“Note: SQLite has undone the open transaction.”).

`.quit`, `.close` and `.open` ask when a transaction is open: `[C] commit  [R] rollback  [Esc] cancel`.

Several databases: ATTACH

Besides the open database (`main`), up to 10 more files can be attached:

```
ATTACH 'A:/sql/computers.db' AS retro
SELECT m.name, k.name FROM owns o JOIN member m ON m.id = o.member_id JOIN retro.computer k ON k.id =
o.computer_id
CREATE TABLE computer AS SELECT * FROM retro.computer
DETACH retro
```

`.databases` lists the attached files.

What the SQLite module does not have

The SQLite module of the CMM2 (SQLite 3.47.0) is made smaller to fit the computer’s memory. Missing are:

Missing	Effect
`EXPLAIN`, `ANALYZE`	no query plans, no statistics
WAL mode	journal always in mode `delete`; files saved in WAL mode on a PC cannot be opened
Math functions	no `sqrt()`, `sin()`, `power()` etc.
Extensions	no `load_extension()`
Time zones	`datetime('now')` gives the time of the CMM2 clock (`TIME$`); the modifiers `localtime` and `utc` are missing

Available are, among others, the JSON functions, window functions, views, triggers, foreign keys, `VACUUM INTO` and `ATTACH`. Databases in UTF-8 and UTF-16 can be opened; text always arrives as UTF-8.

6. Command reference

The commands are in alphabetical order. Square brackets `[ ]` mark parts that may be left out; `a|b` means “a or b”. Short forms are given in brackets after the command. File names without drive and folder refer to the current folder of MMBasic.

Overview by group

Group	Commands
Database	`.open` `.close` `.info` `.databases` `.backup` `.vacuum` `.files` `.kill`
Look	`.tables` `.indexes` `.triggers` `.schema` `.columns` `.browse`
Data	`.insert` `.edit` `.delete` `.import` `.read`
Structure	`.create` `.alter` `.reindex`
Check	`.check` `.fkcheck` `.fk`
Output	`.mode` `.headers` `.separator` `.nullvalue` `.once` `.output` `.dump`
Options	`.set` `.timer` `.bail` `.echo` `.cls` `.help` `.quit`

.alter table

Changes the structure of a table. Shows the columns and offers:

Key	Action
A	add a column (name, type, NOT NULL, default are asked for)
R	rename a column
D	drop a column
T	rename the table

The SQL statement is shown before it runs. Example: `.alter member`

.backup file.db

Writes a clean, compact copy of the open database (`VACUUM INTO`). Asks before it overwrites an existing file. Also the best way to save what is still readable from a damaged database. Example: `.backup club_copy.db`

.bail on|off

`on` : `.read` stops at the first error. `off` (default): `.read` goes on and stops only after 10 errors.

.browse table [WHERE …] [ORDER BY …] (`.b`)

Shows the rows of a table page by page (Enter/N, B, Q). The option `rows` sets the rows per page.

```
.browse computer
.browse items WHERE city='Berlin' ORDER BY price DESC
```

Columns are at most 24 characters wide. Longer values end with `~` (shortened), values over `maxtext` appear as `<n bytes>~!` . If one of these marks appeared, a legend follows the row count. If not all columns fit on the screen, mmsql says “(+n column(s) not shown)”; `.mode line` then shows all of them.

.check [full]

`.check`	quick check (`PRAGMA quick_check`)
`.check full`	full check (`PRAGMA integrity_check`)

Shows “OK - no problems found.” or the problems found. Large databases take time, on generation 1 even minutes. If there are problems, save with `.backup` what can be saved.

.close

Closes the database and says “Closed: file”. If a transaction is open, mmsql asks `[C] commit  [R] rollback  [Esc] cancel` ; with Esc the database stays open.

.cls

Clears the screen and the terminal.

.columns table (`.c`)

Shows the columns of a table: number, name, type, NOT NULL, default and whether the column is (part of) the primary key.

.create table

Creates a table step by step: for each column name, type, primary key, NOT NULL and default. An empty name ends the list. The finished statement is shown before it runs; it may be at most 255 characters long (otherwise create the table with SQL).

.databases (`.db`)

Lists the open database (`main`), `temp` and attached databases with their files.

.delete table key

Shows the row and deletes it after a question. `key` is the value of the primary key; if the table has no single-column primary key, it is the `rowid` (visible with `SELECT rowid, * FROM table`). Long values are shown short as `(long text, n bytes)` or `(BLOB, n bytes)` . Example: `.delete computer 19`

.dump [table]

Writes the whole database (or one table) as SQL text: `CREATE TABLE` , one `INSERT` per row, then indexes, views and triggers, framed by `BEGIN TRANSACTION;` and `COMMIT;` . With `.once` or `.output` into a file; it can be read back with `.read` or used with `sqlite3` on a PC. Format see chapter 8.

```
.once club_backup.sql
.dump
```

.echo on|off

`on` : `.read` shows each statement before it runs (long lines shortened, with their length).

.edit table key

Changes a row. `key` as with `.delete` . Each field appears with its value in the line editor:

Input	Effect
Enter	keep the value
new value	change it
`NULL` or `\N`	NULL
`''`	empty text
`\NULL`	the text “NULL”

The changes are saved after “Save n change(s)? (y/n)” in one transaction; only changed columns are written. Texts over 255 characters and BLOBs appear as `(long text, n bytes - kept)` or `(BLOB, n bytes - kept)` and stay unchanged; change them with SQL (`UPDATE`). Example: `.edit computer 5`

.files [pattern]

Lists files with size and date, sorted by name, with the folder above. Without a pattern: `*.db` . Sizes below 10000 bytes are shown exactly in bytes, larger ones in KB.

```
.files *.sql
.files *
.files A:/sql/*.db
```

.fk [on|off]

Shows or switches the checking of foreign keys for this session (`PRAGMA foreign_keys`). SQLite starts with the checking switched off.

.fkcheck [table]

Finds rows that break a foreign key (`PRAGMA foreign_key_check`).

.headers on|off

Column names as the first line in `csv` and `list` (default: on).

.help [command] (`.h`)

Shows the overview or the help for one command (see chapter 3).

.import [–skip N] file.csv table

Imports a CSV file following RFC 4180 (fields in quotes, `""` for a quote, line breaks inside fields).

- **New table:** the first line gives the column names; all columns get the type TEXT.
- **Existing table:** all lines are data; `--skip 1` skips a header line.
- Separator from `.separator` (one character).
- At most 20 columns. One field per record may be longer than 255 characters (up to `maxtext`).
- The import runs in one transaction. After 10 bad records (wrong number of fields, too long …) **nothing** is imported. Also nothing after Ctrl-C.

```
mmsql[club.db]> .import members.csv member_copy
Created table member_copy with 4 column(s)
 8 row(s) imported into member_copy
```

See also chapter 8.

.indexes [table] (`.ix`, `.indices`)

Lists the indexes of all tables or of one table, with their columns.

.info

Shows file, SQLite version, page size, number of pages and size, free pages (`VACUUM` frees them), journal mode, encoding, auto vacuum, `user_version` and the number of tables, views, indexes and triggers.

.insert table

Adds a row. Asks for each column and shows its type, NOT NULL, default and primary key:

Input	Effect
Enter	default value, or a new key for INTEGER PRIMARY KEY
`NULL` or `\N`	NULL
`''`	empty text
`\NULL`	the text “NULL”

At the end “Insert? (y/n)” and the new rowid. At most 20 columns and 255 characters per value; for wider tables use `INSERT` in SQL.

.kill file

Deletes exactly one file after a question (no wildcards like `*`). The open database cannot be deleted (`.close` first). If a `file.db-journal` is next to it (left from an unfinished transaction), mmsql offers to delete it as well.

```
mmsql> .kill test.db
Delete test.db (8192 B)? (y/n) y
Deleted: test.db
```

.mode table|csv|list|insert [table]|line

Sets the output of queries (saved in `mmsql.cfg`). Details in chapter 8.

Mode	Output
`table`	aligned columns, page by page (default)
`csv`	comma separated (RFC 4180)
`list`	values with the separator, no quotes
`insert`	`INSERT` statements for the table given (default “table”)
`line`	one field per line, good for wide rows and long texts

.nullvalue text

How NULL is shown in `table`, `list`, `csv` and `line`, e.g. `.nullvalue NULL`. `.nullvalue ''` shows NULL as empty (default). Saved in `mmsql.cfg`.

.once file

The **next** query (or `.dump`) goes into the file instead of the screen. An existing file is overwritten.

```
.mode csv
.once items.csv
SELECT * FROM items
```

.open file.db (`.o`)

Opens a database. If the file does not exist, mmsql asks before it creates a new, empty one. Only one database is open at a time (more with `ATTACH`); an open one is closed first (with a question if a transaction is open). Example: `.open A:/data/club.db`

.output [file]

All following queries go into the file, until `.output` without a file switches back to the screen.

.quit (`.q`, `.exit`)

Closes the database (see `.close`) and ends mmsql.

.read file.sql

Runs SQL from a file.

- One or more statements per line (separated by `;`, as at the prompt).
- Lines over 255 characters (e.g. from `.dump`) run as a whole, up to `maxtext`; longer ones are skipped.
- Empty lines and lines starting with `--` are skipped; dot commands are not run.
- Stops after 10 errors, with `.bail on` at the first.
- Ctrl-Z or NUL at the end (XMODEM padding) end the file.
- At the end: “n statement(s) OK, n error(s), n line(s) read”, and a hint if a transaction was left open.

A statement must be in **one** line; `.read` does not understand statements spread over several lines yet (see appendix A).

.reindex [name]

Rebuilds all indexes, those of one table, or one index.

.schema [name] (`.s`)

Shows the `CREATE` statements of all objects, or of one table, index, view or trigger.

.separator x

Separator for `csv` and `list`, 1 to 3 characters; `\t` stands for tab. `.import` needs a separator of one character. Saved in `mmsql.cfg`.

.set [name value]

`.set` alone lists all options with value and explanation; `.set name value` changes one option, shows the changed line and saves it in `mmsql.cfg`. All options in chapter 7.

.tables (`.t`)

Lists the tables and views with their number of columns.

.timer on|off

Shows the run time after SQL and longer commands. Saved in `mmsql.cfg`.

.triggers

Lists the triggers and their tables.

.vacuum

Rebuilds the file without free pages and shows the size before and after. Needs about as much free space on the card as the database itself, for a short time.

7. Options (.set)

All options are changed with `.set name value` and saved at once in `mmsql.cfg` (next to the program).

Option	Default	Allowed	Meaning
`rows`	20	5 to 100	rows per page (after a restart)
`width`	screen width	40 to 250	output width in characters
`history`	50	0 to 200	lines in the history (after a restart)
`timer`	0	0, 1	show the run time
`mode`	table	table, csv, list, insert, line	output mode
`headers`	1	on/off or 1/0	column names in csv and list
`separator`	`,`	1 to 3 characters or `\t`	separator for csv and list
`nullvalue`	(empty)	any text	how NULL is shown
`utf8`	dot	dot, raw	how UTF-8 characters are shown
`dotchar`	250	code 33 to 255 or one character	placeholder for `utf8 dot`
`maxtext`	32768	1024 to 1048576	longest text or BLOB mmsql reads (bytes)
`helpfile`	mmsql_en.hlp	file name	help file, e.g. mmsql_de.hlp

width: the default is the character width of the screen, at most the terminal width set with `OPTION WIDTH`.

maxtext: mmsql reserves a memory buffer of this size. On generation 2 and 3 even 1 MB is no problem; on generation 1 with little free memory stay smaller. If there is not enough memory, mmsql sets the value back to 32768.

`mmsql.cfg` is a plain text file with lines `name=value` and can also be edited on a PC.

8. Output, export and import

Output modes

table – aligned columns with a header, page by page:

```
id name             maker       year cpu   ramkb
-----------------------------------------------
1  Apple II         Apple       1977 6502  4
2  PET 2001         Commodore   1977 6502  4
```

- A column is at most 24 characters wide. Longer values end with `~`.
- Texts over `maxtext` appear as `<n bytes>~!`.
- If not all columns fit into the width, the rest are left out: “(+n column(s) not shown)”.
- After the row count a legend explains `~` and `~!` if they appeared.

line – one field per line; shows long texts in full, line breaks indented:

```
id   = 9
name = Commodore 64
note = first line
       second line
```

csv – values separated by the separator (default comma), following RFC 4180: a field is put in quotes if it contains the separator, a quote or a line break; quotes are doubled. With `.headers on` the header line comes first.

list – like csv, but without quotes.

insert – one `INSERT INTO "table" VALUES(...);` per row, for the table given with `.mode insert table`.

Values in exports

Value	csv / list	insert / .dump
NULL	text from `.nullvalue` (default empty)	`NULL`
integer	as stored (64 bit)	as stored
decimal	15 significant digits	17 significant digits (no loss)
text	unchanged (UTF-8)	`'text'`, `'` doubled
text with line breaks	in quotes	`'a'||char(10)||'b'` – all in one line
BLOB	raw bytes	`X'0102A0…'`

Files

- Export files (`.once` , `.output` , `.dump`) get the line end **CRLF**.
- Long texts and BLOBs are written **in full**. If a value is longer than `maxtext` , the export stops with a message (row, column, size needed), and the file is reported as incomplete. Remedy: raise `.set maxtext` .

```
Stopped: a text of 40000 bytes (table lt, row 3, column note) is longer than
maxtext (32768). Use .set maxtext 40000 and try again.
lt_dump.sql is incomplete.
```

The dump format

```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE member(id INTEGER PRIMARY KEY, name TEXT NOT NULL, city TEXT, joined INTEGER);
INSERT INTO "member" VALUES(1,'Ada','London',2019);
...
CREATE INDEX owns_member ON owns(member_id);
CREATE VIEW members_per_city AS SELECT city, count(*) AS members FROM member GROUP BY city;
COMMIT;
```

Every statement is in **one** line; line breaks in texts are written with `char(10)` or `char(13)` . So a dump can be read back with `.read` and also used with `sqlite3` on a PC.

Import

CSV with .import: see the command `.import` . Note:

- In CSV, NULL cannot be told apart from an empty field; an empty field is imported as empty text.
- New tables get only columns of type TEXT. For numbers, create the table with the types you want first (`CREATE TABLE`), then import into the existing table (`--skip 1` for a header line).
- At most 20 columns, one field over 255 characters per record.

SQL with .read: dumps from mmsql and simple SQL scripts with one statement per line. A dump from `sqlite3` on a PC works as long as no statement spans several lines (often the case for `CREATE TABLE` from tools and for triggers).

Exchange with a PC

- Transfer files with XMODEM or read the SD card on the PC.
- XMODEM fills up the last block (Tera Term with Ctrl-Z, the CMM2 with NUL). `.read` and `.import` take this as the end of the file; on the PC the padding at the end of the file can be ignored or cut off.
- Databases from a PC must be saved in journal mode `delete`, not `wal` (on the PC: `PRAGMA journal_mode=delete;`).

All exports and imports were compared byte for byte with SQLite on a PC: CSV, insert and dump, the round trips CSV → database → CSV and dump → database → dump, with quotes, line breaks, tabs, UTF-8, 64-bit integers, texts up to 40000 characters and BLOBs with NUL bytes.

9. Limits

What	Limit	Reason / note
Input at the prompt	255 characters	length of an MMBasic string
Statements per line	32	
Line in `.read`	up to `maxtext`	runs as a whole
Reading and writing a text or BLOB	up to `maxtext` (1024 to 1048576 bytes)	memory buffer
Value in the line editor (`.edit`, `.insert`)	255 characters	longer: with SQL
Columns with `.insert` and `.import`	20	firmware: at most 20 values per command
Columns with `.edit`	32 (G1) / 64 (G2, G3)	column buffer
Long fields per record with `.import`	1	
Errors before stopping (`.read`, `.import`)	10	
Columns of a query in memory	32 (G1) / 64 (G2, G3)	more are not shown
Column width in `table`	24 characters	longer ones end with `~`
Column names	64 characters	
Names in lists (`.schema`, `.tables`, `.files`)	200	
`.create` statement	255 characters	otherwise with SQL
Rows per page (`rows`)	5 to 100	
Output width (`width`)	40 to 250	
History (`history`)	0 to 200 entries	
Separator	1 to 3 characters (`.import`: 1)	
Attached databases (`ATTACH`)	10	SQLite module
Database size	limited by the SD card	FAT32 / exFAT

File numbers used by mmsql: #7 (`.read`, `.import`), #8 (`.once`, `.output`, `.dump`), #9 (`mmsql.cfg`, `mmsql.his`, help file). If you take parts of mmsql into your own programs, do not use these numbers at the same time.

10. Messages and hints

Errors from SQLite appear as `Error: SQL: …` with the statement and its length. For common errors mmsql adds a hint:

Message	Meaning / remedy
`no such table`	check the table name; `.tables` lists the tables
`syntax error`	typing error in the SQL – or a dot command without the dot; input max. 255 characters
`file is not a database`	not an SQLite file, encrypted, or saved in WAL mode on a PC
`unable to open database file`	check the file name, the folder and the SD card
`disk I/O error`	SD card full or faulty (see also chapter 11)
`attempt to write a readonly database`	file or SD card write protected
`database is locked`	open transaction? `COMMIT` or `ROLLBACK`
`cannot commit - no transaction is active`	no transaction is running (maybe already undone after Ctrl-C)
`stopped by Ctrl-C`	statement stopped, see chapter 4
`Did you mean .xxx?`	dot forgotten
`Not run: n more statement(s)`	line stopped after an error
`Line n: longer than maxtext`	`.read`: line too long, raise `.set maxtext`
`Record n: … skipped`	`.import`: bad record (number of fields, length)
`Too many errors`	`.read` / `.import` stopped after 10 errors
`Note: SQLite has undone the open transaction.`	after an error SQLite has ended the transaction
`A transaction is still open`	after `.read`: type `COMMIT` or `ROLLBACK`

11. Firmware notes

OPTION CRLF: without `OPTION PICOMITE`, the Enter key sends CR and LF in firmware 8 by default. mmsql sets `OPTION CRLF CR` at the start; the setting only lasts while mmsql runs.

OPTION BREAK: sets the break key (default Ctrl-C). With `OPTION BREAK 0`, SQL statements cannot be stopped.

LGETSTR$ in 8.00.00b1: `LGETSTR$` writes one byte too many into memory when its result is exactly 255 characters long. This can disturb the open database (“disk I/O error” at COMMIT). So mmsql never asks `LGETSTR$` for more than 254 characters. Found while building mmsql 1.1, fixed by Peter Mather for the next firmware. If you write your own programs with LONGSTRING and SQL, do the same.

LINE INPUT # and tabs: `LINE INPUT #` turns tabs into spaces (intended MMBasic behaviour). mmsql therefore reads files with `INPUT$`, so tabs in data are kept.

MM.DEVICE$ under OPTION PICOMITE: gives the PicoMite name then; the G3 cannot be recognised this way (mmsql shows “G2/G3”).

12. Appendix

A. Differences to sqlite3

mmsql has the most important commands of `sqlite3`, plus its own for working on the CMM2 (`.browse`, `.insert`, `.edit`, `.delete`, `.create`, `.alter`, `.columns`, `.info`, `.check`, `.fkcheck`, `.files`, `.kill`, `.set`, `.cls`).

Not there yet, planned for later versions:

- statements over **several lines** at the prompt and in `.read` (e.g. triggers and formatted `CREATE TABLE` from PC dumps); with it, statements over 255 characters at the prompt
- `.import` with more than 20 columns
- more output modes: `markdown`, `json`, `tabs`, `quote`
- `.tables` and `.schema` with a pattern, `.restore`, `.changes`, `.print`, `.width`

Not possible on the CMM2: `.recover` (missing extension), `.eqp`/`.expert` (no EXPLAIN), `.load` (no extensions), `.shell`/`.system`, `.excel`.

B. New in 1.1 compared to 1.0

- Needs firmware 8; for 7.00.00RC6 there is mmsql 1.0.1.
- Ctrl-C stops SQL, mmsql goes on and tells exactly what was undone.
- Transactions watched: star in the prompt, question at `.quit`, `.close`, `.open`.
- Several statements per line, each with its own result.
- Long texts and BLOBs over 255 characters in display, export, dump, `.read` and `.import`; option `maxtext`; marks `~` and `~!`.
- New commands `.files` and `.kill`.
- `.info` shows the real encoding; sizes exactly in bytes or KB; G1/G2/G3 in the banner.
- No more double Enter (`OPTION CRLF CR`), hint at `OPTION BREAK 0`, hint also for `,help`.
- Fixes from 1.0.1: crash with values of exactly 255 characters ending with a quote; line breaks in `.mode line`.

C. Thanks

To Peter Mather for SQLite on the CMM2 and for the quick answers and fixes (Ctrl-C, LONGSTRING, LGETSTR$), to Geoff Graham for MMBasic, and to the community on TheBackShed.