PicoMite

Benutzerhandbuch

MMBasic BASIC-Interpreter Version 6.01.00

Für den
Raspberry Pi Pico
Raspberry Pi Pico 2
Raspberry Pi Pico W
Raspberry Pi Pico 2 W
und
Module mit den Prozessoren RP2040 und RP2350

Revision 4
(22. Dezember 2025)

Aktualisierungen dieses Handbuchs und weitere Informationen zu MMBasic finden Sie unter http://geoffg.net/picomite.html
und http://mmbasic.com

Über

Peter Mather (matherp im Back Shed Forum) leitete das Projekt, portierte MMBasic auf den Raspberry Pi Pico und schrieb die Treiber für dessen Hardware-Funktionen. Der MMBasic-Interpreter und dieses Handbuch wurden von Geoff Graham (http://geoffg.net) verfasst. Darüber hinaus haben viele andere das Projekt mit speziellem Code, Tests und Vorschlägen unterstützt.

Unterstützung

Support-Fragen sollten im Back Shed-Forum (http://www.thebackshed.com/forum/Microcontrollers) gestellt werden, wo es viele begeisterte MMBasic-Anwender gibt, die Ihnen gerne weiterhelfen. Die Entwickler der PicoMite-Firmware sind ebenfalls regelmäßig in diesem Forum anzutreffen.

Urheberrecht und Danksagungen

Dieses Handbuch

Inhalt

Einführung

PicoMite ist eine Betriebsfirmware für alle Versionen des Raspberry Pi Pico, einschließlich Pico, Pico 2, Pico W und Pico 2 W.

Es enthält einen BASIC-Interpreter (MMBasic), eine mit Microsoft BASIC kompatible Implementierung der Programmiersprache BASIC mit Gleitkomma-, Ganzzahl- und Zeichenfolgenvariablen, Arrays, langen Variablennamen, einem integrierten Programmeditor und vielen anderen Funktionen.

Es gibt Versionen der PicoMite-Firmware, die für eingebettete Steuerungsanwendungen (wie Heizungssteuerungen, Einbruchmelder usw.) geeignet sind, sowie Versionen mit VGA/HDMI- und Tastaturunterstützung für den Bau eines eigenständigen Computers.

[illegible] A-Pins steuern und Kommunikationsprotokolle wie I²C oder SPI verwenden, um Daten von einer Vielzahl von Sensoren abzurufen. Sie können Daten auf kostengünstigen Farb-LCD-Displays anzeigen, Spannungen messen, digitale Eingänge erkennen und Ausgangspins ansteuern, um Leuchten, Relais usw. einzuschalten. Und mit dem Raspberry Pi Pico W können Sie auf das Internet zugreifen und einen WEB-Server auf diesem kostengünstigen Modul aufbauen.

Die PicoMite-Firmware kann völlig kostenlos heruntergeladen und

verwendet werden. Zusammenfassend sind die Funktionen der PicoMite-

Firmware:

- **Der BASIC-Interpreter ist voll ausgestattet** mit doppelter Genauigkeit bei Gleitkommazahlen, 64-Bit-Ganzzahlen und String-Variablen, langen Variablennamen, Arrays von Gleitkommazahlen, Ganzzahlen oder Strings mit mehreren Dimensionen, umfangreicher String-Verarbeitung und benutzerdefinierten Unterprogrammen und Funktionen. Darüber hinaus ermöglicht MMBasic die Einbettung von kompilierten C-Programmen für Hochleistungsfunktionen. Der Schwerpunkt liegt auf Benutzerfreundlichkeit und einfacher Entwicklung.

- **Unterstützung für alle Raspberry Pi Pico-Eingangs-/Ausgangspins**. Diese können unabhängig voneinander als digitaler Eingang oder Ausgang, analoger Eingang, Frequenz- oder Periodendauer-Messung und Zählung konfiguriert werden. Interrupts können verwendet werden, um zu benachrichtigen, wenn ein Eingangspin seinen Zustand geändert hat. PWM-Ausgänge können verwendet werden, um verschiedene Töne zu erzeugen, Servos zu steuern oder computergesteuerte Spannungen zu erzeugen.

- **Unterstützung für TFT-LCD-Bildschirme** mit parallelen, SPI- und I²C-Schnittstellen, sodass das BASIC-Programm Text anzeigen und Linien, Kreise, Rechtecke usw. in bis zu 16 Millionen Farben zeichnen kann. Resistive Touch-Controller auf diesen Bildschirmen werden ebenfalls unterstützt, sodass sie als hochentwickelte Eingabegeräte verwendet werden können.

- **Unterstützung für Internet- und WEB-Protokolle unter Verwendung des Raspberry Pi Pico W und Pico 2 W**. Dazu gehört ein WEB-Server mit TCP und HTML, der über TCP und HTTP auf andere Ressourcen zugreift. MQTT-Protokoll für die Verbindung über einen Message Broker. NTP-Protokoll zum Abrufen von Datum/Uhrzeit von einem Zeitserver. Telnet für den Fernzugriff auf die Konsole und TFTP für die schnelle Dateiübertragung.

- **Unterstützung für eine PS2- oder USB-Tastatur und HDMI- oder VGA-Videoausgang**. Dazu gehört die vollständige Unterstützung für Grafiken, Audio (Soundeffekte und Musik), internen Programmspeicher, Gamecontroller und mehr. Dadurch wird der Raspberry Pi Pico oder zu einem eigenständigen Computer wie der Apple II oder Tandy TRS-80 von gestern. Ideal zum Schreiben von Spielen, zum Erlernen von BASIC oder einfach zum Ausbalancieren Ihres Scheckbuchs.

- **Flexible Programm- und Datenspeicherung**. Programme und Daten können aus einem internen Dateisystem, das aus dem Flash-Speicher des Pico erstellt wurde, oder auf eine extern angeschlossene SD-Karte mit bis zu 32 GB, die als FAT16 oder FAT32 formatiert ist, gelesen/geschrieben werden. Dazu gehört das Öffnen von Dateien zum Lesen, Schreiben oder für den wahlfreien Zugriff sowie das Laden und Speichern von Programmen.

- In die Firmware ist **ein Vollbild-Editor** integriert, mit dem das gesamte Programm in einer Sitzung bearbeitet werden kann. Er umfasst erweiterte Funktionen wie farbcodierte Syntax, Suchen und Kopieren sowie Ausschneiden und Einfügen in die bzw. aus der Zwischenablage.

- **Programme können einfach** von einem Desktop- oder Laptop-Computer (Windows, Mac oder Linux) über die serielle Konsole oder über eine SD-Karte **übertragen werden**.

- Es wird **eine umfassende Palette an Kommunikationsprotokollen** implementiert, darunter I²C, asynchrones serielles Protokoll, RS232, SPI und 1-Wire. Diese können zur Kommunikation mit vielen Sensoren (Temperatur, Luftfeuchtigkeit, Beschleunigung usw.) sowie zum Senden von Daten an Testgeräte verwendet werden.

- **Integrierte Befehle** für die direkte Anbindung an Infrarot-Fernbedienungen, den Temperatursensor DS18B20, LCD-Anzeigemodule, batteriegepufferte Uhren, numerische Tastaturen und mehr.

Firmware-Versionen und Dateien „ ”

Die PicoMite-Firmware kann je nach geladener Firmware-Version für zwei unterschiedliche Aufgaben verwendet werden. Diese Aufgaben sind: ein eigenständiger Computer und ein eingebetteter Controller:

Eigenständiger Computer mit integrierter Steuerung

Versionen mit VGA- oder HDMI-Videoausgang sind für den Einsatz als eigenständiger Computer vorgesehen. Diese starten hoch und zeigen die Ausgabe des BASIC-Interpreters auf dem an den Videoausgang angeschlossenen Monitor an. Sie sind mit einer PS2- oder USB-Tastatur gekoppelt, und über die Tastatur und den Videoausgang können Sie ein Programm eingeben und bearbeiten, es ausführen, Optionen einstellen usw.

Da der eigenständige Computer mit der Anzeige der BASIC-Eingabeaufforderung startet, werden sie oft als „Boot-to-BASIC”-Computer bezeichnet. Sie sind einfach und machen Spaß und waren in den 70er und 80er Jahren sehr beliebt, zum Beispiel der Apple II, Tandy TRS-80, Commodore 64 und andere.

Wenn ein Programm ausgeführt wird, werden alle seine Ausgaben (Text und Grafiken) auf dem Videoausgang angezeigt. Die Textausgabe wird auch an die serielle Konsole gesendet. Dies ist ein sekundärer Kommunikationskanal, der den USB-Anschluss des Raspberry Pi Pico nutzt und eine weitere Möglichkeit darstellt, mit einem Desktop- oder Laptop-Computer mit dem MMBasic-Interpreter zu kommunizieren. Einzelheiten zur Verwendung finden Sie im nächsten Kapitel: *Serielle Konsole*.

Eingebetteter Controller „ “

Versionen der Firmware ohne Videoausgang sind in erster Linie für den Einsatz als eingebetteter Controller vorgesehen. Hier wird der Raspberry Pi Pico oder Pico 2 als „Gehirn“ in einem Gerät verwendet. Beispiele hierfür sind Einbruchmelder, Heizungsregler, Wetterstationen usw. Oftmals verfügen sie über ein angeschlossenes berührungsempfindliches LCD-Panel, über das der Benutzer das Gerät steuern und die Ausgabe beobachten kann.

Es gibt auch eine Version der Firmware, die die drahtlose Schnittstelle des Raspberry Pi Pico W (und 2 W) unterstützt. Mit dieser können Sie einen eingebetteten Controller erstellen, auf dem ein Miniatur-Webserver läuft und der auf das Internet zugreifen kann, um die Uhrzeit abzurufen, E-Mails zu versenden usw.

Um Programme einzugeben, Optionen festzulegen und den Raspberry Pi Pico allgemein als eingebetteten Controller zu verwalten, verwenden Sie die serielle Konsole, um eine Verbindung zu einem Desktop- oder Laptop-Computer herzustellen. Im Gegensatz zu dem oben beschriebenen eigenständigen Computer ist dies die einzige Möglichkeit, mit dem BASIC-Interpreter zu kommunizieren, daher ist es wichtig, dass Sie eine Verbindung herstellen können. Eine Beschreibung der seriellen Konsole finden Sie unter der Überschrift *„Serielle Konsole“* weiter unten.

Prozessor- -Unterstützung

Die PicoMite-Firmware unterstützt die ursprünglichen RP2040-Prozessoren, die im Raspberry Pi Pico verwendet werden, sowie den neueren RP2350, der im Raspberry Pi Pico 2 zum Einsatz kommt. Die Firmware ist auch für die Verwendung mit Modulen anderer Hersteller ausgelegt, die dieselben Chips verwenden.

Während es vom RP2040 nur eine Version gibt, ist der RP2350 in vier Unterversionen erhältlich: RP2350A, RP2350B, RP2354A und RP2354B. Der RP2350B entspricht dem RP2350A, verfügt jedoch über 18 zusätzliche E/A-Pins (Pins GP30 bis GP47), die automatisch in MMBasic verfügbar sind. Beide Chips werden von derselben PicoMite-Firmware unterstützt und funktionieren gleich. Daher gelten in diesem Handbuch alle Verweise auf den RP2350 gleichermaßen für die Varianten A und B, und es kann dieselbe Firmware verwendet werden.

Der RP2354A und der RP2354B werden derzeit nicht unterstützt (könnten aber in Zukunft unterstützt werden).

In diesem Handbuch beziehen sich alle Verweise auf den Raspberry Pi Pico auch auf den Raspberry Pi Pico 2, sofern dies nicht ausdrücklich ausgeschlossen ist. Bei Abweichungen wird die Teilenummer des Prozessors (RP2040 oder RP2350) verwendet, um den Unterschied deutlich zu machen.

Datei snamen

Die ZIP-Datei mit der Firmware enthält zwölf Firmware-Dateien. Ein typischer Dateiname für ein

Firmware-Image sieht wie folgt aus:

Pico Mite RP 2350 VGAUSBV 6 .00 . 02 . uf2

Wobei (in diesem Beispiel):

- **RP2350** der Prozessor ist, für den die Firmware kompiliert wurde.
- **VGAUSB** ist der unterstützte Funktionsumfang (VGA und USB).
- **V6.00.01** ist die Versionsnummer. Diese wird in zukünftigen Versionen erhöht.
- **.uf2** ist die Erweiterung, die ein ladbares Raspberry Pi Pico-Firmware-Image kennzeichnet.

In der folgenden Tabelle sind die Präfixe für die einzelnen Firmware-Dateien und die damit verbundenen Funktionen aufgeführt.

Firmware-Dateiname Beispiel: PicoMiteRP2040V60.0.01.uf2	CPU	Touch-LCD Panel	Tastatur/Maus		Videoausgang		WLAN-Internet
			PS2	USB	VGA	HDMI	
PicoMiteRP2040	RP2040	✓	✓				
PicoMiteRP2350	RP2350	✓	✓				
PicoMiteRP2040USB	RP2040	✓		✓			
PicoMiteRP2350USB	RP2350	✓		✓			
PicoMiteRP2040VGA	RP2040		✓		✓		
PicoMiteRP2350VGA	RP2350		✓		✓		
PicoMiteRP2040VGAUSB	RP2040			✓	✓		
PicoMiteRP2350VGAUSB	RP2350			✓	✓		
PicoMiteHDMI	RP2350		✓			✓	
PicoMiteHDMIUSB	RP2350			✓		✓	
WebMiteRP2040	RP2040	✓	✓				✓
WebMiteRP2350	RP2350A	✓	✓				✓

Laden der Firmware „ “

Der Raspberry Pi Pico und Pico 2 verfügen über einen eigenen integrierten Firmware-Loader, der einfach zu bedienen ist. Um die PicoMite-Firmware zu laden, gehen Sie wie folgt vor:

- Laden Sie die PicoMite-Firmware von http://geoffg.net/picomite.html herunter, entpacken Sie die Datei und suchen Sie die für Ihre Verwendung geeignete Firmware (siehe vorherige Überschriften).
- Schließen Sie den Raspberry Pi Pico mit einem USB-Kabel an Ihren Computer (Windows, Linux oder Mac) an, **während Sie die weiße BOOTSEL-Taste** oben auf dem Modul **gedrückt halten**.
- Der Raspberry Pi Pico sollte sich mit Ihrem Computer verbinden und ein virtuelles Laufwerk erstellen (genauso, als hätten Sie einen USB-Stick angeschlossen). Sie können alle Dateien, die sich auf diesem „Laufwerk" befinden, ignorieren.
- Kopieren Sie die Firmware-Datei (mit der Erweiterung .uf2) auf dieses virtuelle Laufwerk.
- Nach Abschluss des Kopiervorgangs startet der Raspberry Pi Pico neu und erstellt einen virtuellen seriellen Anschluss über USB auf Ihrem Computer. Einzelheiten zur Verwendung finden Sie im Kapitel *„Serielle Konsole"* weiter unten.
- Die LED am Raspberry Pi Pico blinkt langsam und zeigt damit an, dass die PicoMite-Firmware mit MMBasic nun ausgeführt wird.

Das vom Raspberry Pi Pico erstellte virtuelle Laufwerk sieht zwar wie ein USB-Speicherstick aus, ist aber keiner. Die Firmware-Datei verschwindet nach dem Kopieren, und wenn Sie versuchen, einen anderen Dateityp zu kopieren, wird dieser ignoriert.

Durch das Laden der PicoMite-Firmware kann der gesamte Flash-Speicher gelöscht werden, einschließlich des aktuellen Programms, aller Dateien auf Laufwerk A: und aller gespeicherten Variablen. Stellen Sie daher sicher, dass Sie diese Daten sichern, bevor Sie die Firmware aktualisieren.

Es ist möglich, dass der Flash-Speicher beschädigt wird, was zu ungewöhnlichem und unvorhersehbarem Verhalten führen kann. In diesem Fall sollten Sie die unten aufgeführte entsprechende Firmware-Datei herunterladen und wie oben beschrieben auf den Pico laden. Dadurch wird der Raspberry Pi Pico auf den Werkszustand zurückgesetzt, und Sie können die PicoMite-Firmware erneut laden:

Raspberry Pi Pico (RP2040) https://geoffg.net/Downloads/picomite/Clear_Flash.uf2 Raspberry Pi Pico 2 (RP2350) https://geoffg.net/Downloads/picomite/Clear_Flash_RP2350.uf2

Serielle S konsole

Die serielle Konsole ist eine Methode, um Ihren Desktop- oder Laptop-Computer mit dem Raspberry Pi Pico und der MMBasic-Konsole zu verbinden. Mit Zugriff auf die Konsole können Sie Programme eingeben und bearbeiten, ausführen usw. Die meisten Versionen der Firmware erstellen die serielle Konsole automatisch als virtuellen seriellen Anschluss über USB. In diesem Kapitel wird beschrieben, wie das funktioniert und wie Sie es verwenden können.

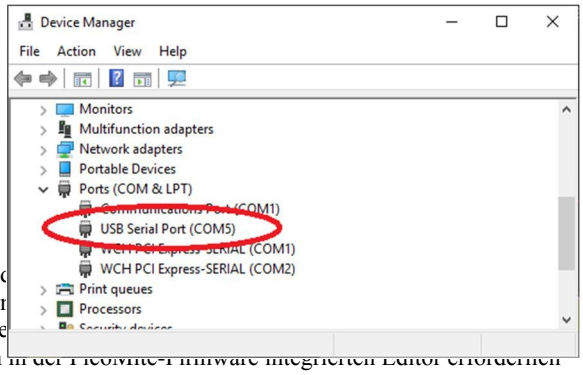
Für einen eigenständigen Computer (wie oben beschrieben) ist die serielle Konsole eine sekundäre Kommunikationsmethode, aber wenn Sie den Raspberry Pi Pico als eingebetteten Controller verwenden, ist sie die einzige verfügbare Kommunikationsmethode, daher ist es wichtig, dass Sie eine Verbindung zu ihr herstellen können.

Versionen der PicoMite-Firmware, die eine USB-Tastatur/Maus unterstützen, können den seriellen Anschluss über USB nicht erstellen. Für diese Firmware sollten Sie daher das Kapitel *„Tastatur/Maus/Gamepad"* lesen, um eine Alternative zu finden.

Virtueller serieller Port „ "

Der von der PicoMite-Firmware erstellte virtuelle serielle Anschluss über USB verwendet das CDC-Protokoll (Communication Device Class) und verhält sich wie ein normaler serieller Anschluss, funktioniert jedoch über USB. Windows 10 und 11 enthalten einen Treiber dafür, bei anderen Betriebssystemen müssen Sie jedoch möglicherweise einen Treiber laden (siehe nächste Seite).

Wenn Sie den USB-Anschluss des Raspberry Pi Pico an Ihren Desktop- oder Laptop-Computer anschließen (nachdem Sie die PicoMite-Firmware geladen haben), wird die Verbindung sofort hergestellt.

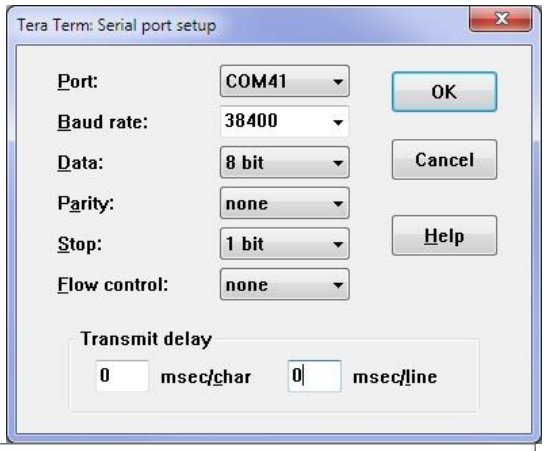


Notieren Sie sich dann die von Ihrem Computer für die virtuelle serielle Verbindung erstellte Portnummer. Unter Windows können Sie dazu den Geräte-Manager starten und unter „Anschlüsse (COM & LPT)" nach einem neuen COM-Port suchen, wie rechts dargestellt.

Terminal- -Emulator

Sie benötigen außerdem einen Terminalemulator, der auf Ihrem Desktop- od
Programm, das wie ein altmodischer Computerterminal funktioniert, auf der
alle Tastendrücke über die serielle Verbindung an den Remote-Computer ge
Terminalemulator sollte die VT100-Emulation unterstützen, da dies für den in der PicoMite-Firmware integrierten Editor erforderlich ist.

Für Windows-Benutzer wird die Verwendung von Tera Term empfohlen, da dieses Programm über einen guten VT100-Emulator verfügt und bekanntermaßen mit dem XModem-Protokoll kompatibel ist, mit dem Sie Programme zum und vom PicoMite übertragen können. Tera Term kann unter folgender Adresse heruntergeladen werden: http://tera-term.en.lo4d.com .



Der Screenshot auf der rechten Seite zeigt die Einstellungen für Tera Term. Beachten Sie, dass die Einstellung „Port:" davon abhängt, an welchen USB-Anschluss Ihr Raspberry Pi Pico angeschlossen ist.

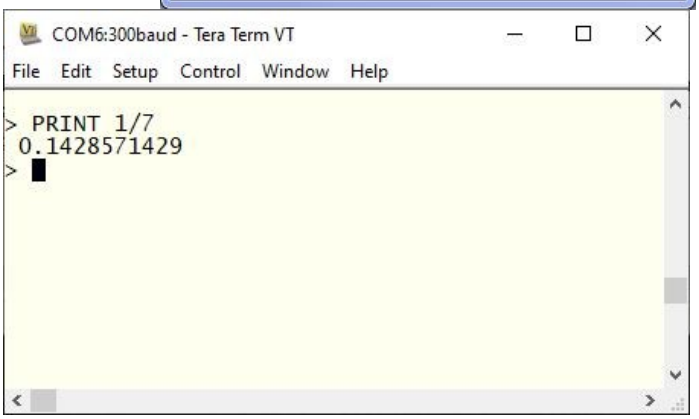
Die PicoMite-Firmware ignoriert die Baudrateneinstellung, sodass sie auf eine beliebige Geschwindigkeit eingestellt werden kann (außer 1200 Baud, wodurch der Pico in den Firmware-Upgrade-Modus versetzt wird).

Wenn Sie Tera Term verwenden, stellen Sie keine Verzögerung zwischen den Zeichen ein, und wenn Sie Putty verwenden, stellen Sie die Rücktaste so ein, dass sie das Rücktastenzeichen erzeugt.

Die „ "-Konsole

Sobald Sie den virtuellen seriellen Anschluss identifiziert und Ihren Terminalemulator damit verbunden haben, sollten Sie die Eingabetaste auf Ihrer Tastatur drücken können und die MMBasic-Eingabeaufforderung sehen, die aus dem Größer-als-Zeichen besteht (z. B. „>").

Dies ist die Konsole, über die Sie Befehle zum Konfigurieren von MMBasic, zum Laden des BASIC-Programms sowie zum Bearbeiten und Ausführen desselben eingeben können. MMBasic verwendet die Konsole auch zur Anzeige von Fehlermeldungen.



Windows 7 und 8.1

Der serielle USB-Anschluss verwendet das CDC-Protokoll. Die entsprechenden Treiber sind in Windows 10 und 11 standardmäßig enthalten und werden automatisch geladen.

Die Raspberry Pi Foundation listet Windows 7 oder 8.1 als „nicht unterstützt“ auf, jedoch können Sie ein Tool wie Zadig (https://zadig.akeo.ie) verwenden, um einen generischen Treiber für ein „usbser“-Gerät zu installieren, wodurch diese Computer eine Verbindung herstellen können sollten. Dieser Beitrag beschreibt den Vorgang: https://github.com/raspberrypi/pico-feedback/issues/118

Apple- -Macintosh

Der Apple Macintosh (OS X) ist etwas einfacher, da er über einen integrierten Gerätetreiber und Terminalemulator verfügt. Starten Sie zunächst die Anwendung „Terminal“ und listen Sie die angeschlossenen seriellen Geräte auf, indem Sie Folgendes eingeben:

```
ls /dev/tty.*.
```

Der USB-zu-Seriell-Konverter wird als etwas wie `/dev/tty.usbmodem12345` aufgelistet. Während Sie sich noch an der Terminal-Eingabeaufforderung befinden, können Sie den Terminalemulator mit 115200 Baud ausführen, indem Sie den folgenden Befehl verwenden:

```
Bildschirm /dev/tty.usbmodem12345 115200
```

Standardmäßig sind die Funktionstasten für die Verwendung im integrierten Programmeditor des PicoMite nicht korrekt definiert, sodass Sie die im Kapitel *„Vollbild-Editor”* dieses Handbuchs definierten Steuerungssequenzen verwenden müssen. Um dies zu vermeiden, können Sie den Terminalemulator so konfigurieren, dass er diese Codes generiert, wenn die entsprechenden Funktionstasten gedrückt werden.

Die Dokumentation zum Befehl „screen” finden Sie hier: https://www.systutorials.com/docs/linux/man/1-screen/

Linux

Für Linux siehe diese Beiträge:

https://www.thebackshed.com/forum/ViewTopic.php?TID=14157&PID=175474#175474#175466 und

https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=16312&LastEntry=Y#213664#213594

Android

Für Android-Geräte siehe diesen Beitrag:

https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17476&LastEntry=Y#230521#230517

Erste Schritte mit dem

Sobald Sie Zugriff auf die Konsole und die MMBasic-Eingabeaufforderung haben, können Sie einige Dinge tun, um zu überprüfen, ob Ihr Computer funktioniert. Alle diese Befehle müssen in die Eingabeaufforderung (d. h. „>“) eingegeben werden.

Was Sie eingeben, wird fett dargestellt, und die MMBasic-Ausgabe wird in normaler Schrift angezeigt.

Versuchen Sie eine einfache Berechnung:

```
> PRINT 1/7
0,1428571429
```

Sehen Sie nach, wie viel Speicher Sie haben:

```
> MEMORY
Programm:
    0K ( 0%) Programm (0 Zeilen)
 180K (100 %) frei

Gespeicherte Variablen:
   16 KB (100 %) frei

RAM:
    0K (0 %) 0 Variablen 0K
    (0 %) Allgemein
 228K (100 %) frei 112K (100 %) frei
```

Wie spät ist es gerade? Beachten Sie, dass die interne Uhr beim Einschalten auf Mitternacht zurückgesetzt wird.

```
> ZEIT DRUCKEN$
00:04:01
```

Stellen Sie die Uhr auf die aktuelle Uhrzeit ein:

```
> TIME$ = „10:45“
```

Überprüfen Sie die Uhrzeit erneut:

```
> ZEIT$
10:45:09
```

Zählen Sie bis 20:

```
> FOR a = 1 to 20 : PRINT a; : NEXT a
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Ein einfaches Programm „ “

Um ein Programm einzugeben, können Sie den Befehl EDIT verwenden, der später in diesem Handbuch beschrieben wird. Im Moment müssen Sie jedoch nur wissen, dass alles, was Sie eingeben, an der Cursorposition eingefügt wird, dass Sie den Cursor mit den Pfeiltasten bewegen können und dass Sie mit der Rücktaste das Zeichen vor dem Cursor löschen können.

Um einen schnellen Eindruck von der Funktionsweise von MMBasic zu bekommen, probieren Sie folgende Sequenz aus:

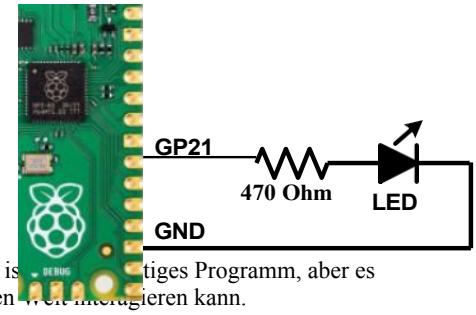
- Geben Sie an der Eingabeaufforderung `EDIT` ein und drücken Sie die ENTER-Taste.
- Der Editor sollte starten und Sie können folgende Zeile eingeben: `PRINT "Hello World"`
- Drücken Sie die Taste F1 in Ihrem Terminalemulator (oder STRG-Q, was dasselbe bewirkt). Dadurch wird der Editor angewiesen, Ihr Programm zu speichern und zur Eingabeaufforderung zurückzukehren.
- Geben Sie an der Eingabeaufforderung `RUN` ein und drücken Sie die ENTER-Taste.
- Sie sollten nun die Meldung „`Hello World`“ sehen.

Herzlichen Glückwunsch. Sie haben gerade Ihr erstes Programm in BASIC geschrieben und ausgeführt.

Wenn Sie erneut EDIT eingeben, gelangen Sie zurück in den Editor, wo Sie Ihr Programm ändern oder ergänzen können.

Blinkt eine LED „ “

Verbinden Sie eine LED mit Pin GP21 (auf der Unterseite der Platine markiert) und einem Erdungs-Pin, wie in der Abbildung rechts gezeigt.



Geben Sie dann mit dem Befehl EDIT das folgende Programm ein:

```
SETPIN GP21, DOUT
DOPIN(GP21) = 1
   PAUSE 300
   PIN(GP21) = 0
   PAUSE 300 LOOP
```

Wenn Sie dieses Programm gespeichert und ausgeführt haben, sollte die LED blinken. Es ist ein einfaches Programm, aber es veranschaulicht, wie die PicoMite-Firmware über Ihre Programmierung mit der physischen Welt interagieren kann.

Das Programm selbst ist einfach. In der ersten Zeile wird Pin GP21 als Ausgang festgelegt. Anschließend tritt das Programm in eine Endlosschleife ein, in der der Ausgang dieses Pins auf „high“ gesetzt wird, um die LED einzuschalten, gefolgt von einer kurzen Pause (300 Millisekunden). Der Ausgang wird dann auf „low“ gesetzt, gefolgt von einer weiteren Pause. Das Programm wiederholt dann die Schleife.

Wenn Sie es so belassen, bleibt der Raspberry Pi Pico für immer mit blinkender LED stehen. Wenn Sie etwas ändern möchten (z. B. die Blinkgeschwindigkeit), können Sie das Programm durch Eingabe von STRG-C auf der Konsole unterbrechen und es dann nach Bedarf bearbeiten. Das ist der große Vorteil von MMBasic: Es ist sehr einfach, ein Programm zu schreiben und zu ändern.

Wenn Sie möchten, dass dieses Programm bei jedem Einschalten automatisch gestartet wird, können Sie den folgenden Befehl an der Eingabeaufforderung oder im Programm verwenden:

```
OPTION AUTORUN ON
```

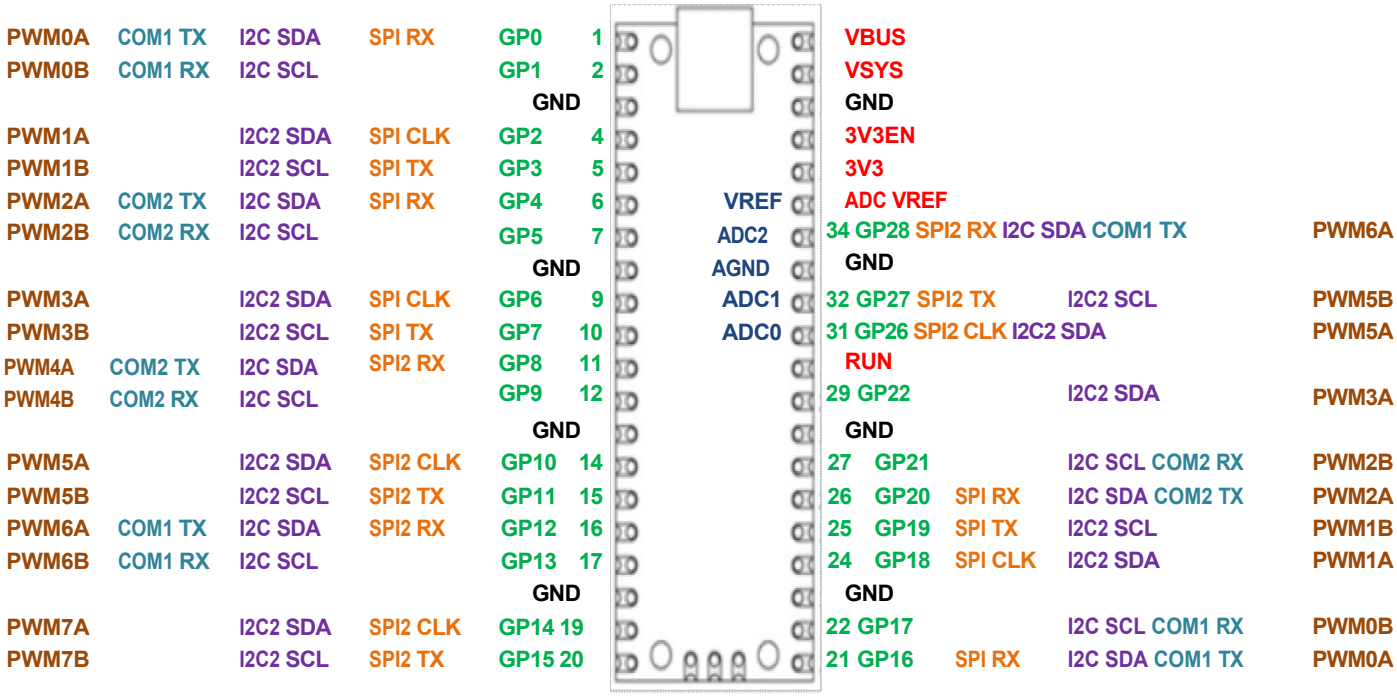
Um dies zu testen, können Sie die Stromversorgung unterbrechen und dann wiederherstellen. Das Gerät sollte starten und die LED blinken.

Tutorial zur Programmierung in der Sprache BASIC

Wenn Sie noch keine Erfahrung mit der Programmiersprache BASIC haben, sollten Sie sich jetzt Anhang J (*Programmieren in BASIC – Ein Tutorial*) am Ende dieses Handbuchs zuwenden. Dabei handelt es sich um ein umfassendes Tutorial zu dieser Sprache, das Ihnen die Grundlagen in einem leicht verständlichen Format mit vielen Beispielen vermittelt.

Hardware- sdaten

Dieses Diagramm zeigt die möglichen Verwendungszwecke innerhalb von MMBasic für jeden I/O-Pin auf dem Raspberry Pi Pico und Pico 2:



Bei Versionen mit VGA-Videoausgang sind sechs Pins (GP16 bis GP21) für diese Funktion reserviert. Ebenso sind bei HDMI-Versionen acht Pins (GP12 bis GP19) für diese Funktion reserviert. Weitere Informationen finden Sie im Kapitel *„Videoausgang“*.

Die Firmware-Version mit USB-Tastatur-/Mausunterstützung reserviert außerdem Pin 11 (GP8) für die serielle Konsole Tx und Pin 12 (GP9) für Rx. Weitere Informationen finden Sie im Kapitel *„Tastatur/Maus/Gamepad“*.

Die Notation lautet wie folgt:

GP0 bis GP28	Kann für digitale Ein- oder Ausgänge verwendet werden.
COM1, COM2	Kann für asynchrone serielle E/A verwendet werden (Pins UART0 und UART1 im Pico-Datenblatt). I2C,
I2C2	Kann für I²C-Kommunikation verwendet werden (I2C0- und I2C1-Pins im Pico-Datenblatt).
SPI, SPI2	Kann für SPI-E/A verwendet werden (siehe Anhang D). (SPI0- und SPI1-Pins im Pico-Datenblatt).
PWMnx	Kann für PWM-Ausgabe verwendet werden (siehe die Befehle PWM und SERVO).
GND	Gemeinsame Masse.
VBUS	5-V-Versorgung direkt über den USB-Anschluss.
VSYS	5-V-Versorgung, die vom SMPS zur Bereitstellung von 3,3 V verwendet wird. Diese kann als 5-V-
Ausgang oder -Eingang verwendet werden. 3V3EN Aktiviert den 3,3-V-Regler (niedrig = aus, hoch = aktiviert).	
RUN	Reset-Pin, niedrig hält das Gerät im Reset-Zustand.
ADCn	Diese Pins können zur Spannungsmessung (Analogeingang)
verwendet werden. ADC VREF	Referenzspannung für die Spannungsmessung.
AGND	Analoge Masse.

Innerhalb des MMBasic-Programms können I/O-Pins unter Verwendung der physikalischen Pin-Nummer (d. h. 1 bis 40) oder der GP-Nummer (d. h. GP0 bis GP28) referenziert werden. Die folgenden Beispiele beziehen sich auf denselben Pin und funktionieren identisch:

```
        SETPIN 32, DOUT
und     SETPIN GP27, DOUT
```

Aus Gründen der Portabilität wird empfohlen, nur die GP-Nummer zu verwenden.

In der PicoMite-Firmware sind On-Chip-Funktionen wie die SPI- und I2C-Schnittstellen nicht wie beispielsweise beim Micromite festen Pins zugewiesen. Die PicoMite-Firmware nutzt den Befehl SETPIN in großem Umfang, nicht nur zur Konfiguration von E/A-Pins, sondern auch zur Konfiguration der Pins, die für Schnittstellen wie Seriell, SPI, I²C usw. verwendet werden.

Die Pins müssen gemäß dieser Zeichnung zugewiesen werden. Beispielsweise kann SPI TX den Pins GP3, GP7 oder GP19 zugewiesen werden, jedoch nicht dem Pin GP11, der nur dem SPI2-Kanal zugewiesen werden kann. Die Zuweisungen müssen nicht im selben „Block" erfolgen, sodass Sie beispielsweise SPI2 TX dem Pin GP11 und SPI2 RX dem Pin GP28 zuweisen können.

-Module von Drittanbietern

Auf Pins, die auf dem Raspberry Pi Pico nicht freigelegt sind, kann über MMBasic weiterhin über ihre GPn-Nummer zugegriffen werden. Dadurch kann MMBasic auf anderen Modulen verwendet werden, die die Prozessoren RP2040 oder RP2350 verwenden.

Auf dem Raspberry Pi Pico werden diese versteckten Pins wie folgt für interne Funktionen verwendet:

- GP23 ist ein digitaler Ausgang, der auf den Wert von OPTION POWER eingestellt ist. (EIN=PWM, AUS=PFM).
- GP24 ist ein digitaler Eingang, der bei Vorhandensein von VBUS hoch ist.
- GP25 ist ebenfalls PWM4B. Es handelt sich um einen Ausgang, der mit der integrierten LED verbunden ist.
- GP29 ist auch ADC3, ein analoger Eingang, der ⅓ von VSYS misst.

Auf Modulen von Drittanbietern, die diese zur Verfügung stellen, können sie jedoch wie folgt verwendet werden:

- GP23: DIGITAL_IN: DIGITAL_OUT, SPI TX, I2C2 SCL, PWM3B
- GP24: DIGITAL_IN: DIGITAL_OUT, SPI2 RX, COM2 TX, I2C SDA, PWM4A
- GP25: DIGITAL_IN: DIGITAL_OUT, COM2 RX, I2C SCL, PWM4B
- GP29: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN, COM1 RX, I2C SCL, PWM6B

WebMite-Version für den Raspberry Pi Pico W oder Pico 2 W

Die WebMite-Version verfügt auch über einige GPIO-Pins, die für interne Board-Funktionen verwendet werden:

- GP29 (Eingang/Ausgang) drahtloser SPI CLK/ADC-Modus (ADC3) misst VSYS/3
- GP25 SPI CS (Ausgang) aktiviert bei hohem Pegel auch den GPIO29-ADC-Pin zum Lesen von VSYS
- GP24 (Eingang/Ausgang) drahtlose SPI-Daten / IRQ
- GP23 (Ausgang) Drahtloses Einschaltsignal

Die WebMite-Firmware erlaubt keine Neuzuweisung dieser Pins.

Im Gegensatz zum Standard-Raspberry Pi Pico ist die integrierte LED des Pico W nicht mit einem Pin des RP2040 verbunden, sondern mit einem GPIO-Pin des Wireless-Chips und kann daher nicht über ein BASIC-Programm angesteuert werden.

Die Antenne befindet sich auf der Leiterplatte am gegenüberliegenden Ende des USB-Anschlusses und sollte für eine optimale Leistung frei zugänglich sein – platzieren Sie keine Metallgegenstände unter oder in der Nähe der Antenne.

Unterschiede zwischen den CPU-

Die von der Raspberry Pi Foundation für den Pico und Pico 2 veröffentlichten Chips sind:

RP2040

Dieser ist in einem 60-Pin-Gehäuse untergebracht und wird im originalen Raspberry Pi Pico und vielen anderen Modulen von Drittanbietern verwendet. Die Pinbelegung und Funktionen der I/O-Pins sind wie oben dokumentiert.

RP2350A

Dieser ist ebenfalls in einem 60-Pin-Gehäuse untergebracht und wird im Raspberry Pi Pico 2 und in Modulen von Drittanbietern verwendet. Die Pinbelegung und Funktionen der I/O-Pins entsprechen denen des RP2040 und sind oben dokumentiert.

RP2350B

Der RP2350B ist in einem 80-Pin-Gehäuse mit zusätzlichen 18 GPIO-Pins untergebracht. Die PicoMite-Firmware unterstützt diese zusätzlichen Pins einschließlich der PWM-Kanäle 8-11 (was maximal 24 simultane PWM-Ausgänge ermöglicht). Die Konfiguration für den RP2350B erfolgt automatisch, sodass alle zusätzlichen Pins und drei PIO-Kanäle verfügbar sind (die VGA-Version verfügt über zwei freie PIO-Kanäle). Die Pin-Definitionen für den RP2350B sollten die GP-Nomenklatur verwenden (d. h. GP0 bis GP47). Die Pins GP40 bis GP47 können für analoge Eingänge verwendet werden, die Pins GP26-GP29 unterstützen keine analogen Eingänge.

Die Pinbelegung für die Pins GP0 bis GP29 auf dem RP2350B entspricht der des RP2040 und RP2350A. Die Pins GP30 bis GP47 können wie folgt verwendet werden:

GP30: DIGITAL_IN: DIGITAL_OUT, SPI2 SCK, I2C2 SDA, PWM7A GP31:
DIGITAL_IN: DIGITAL_OUT, SPI2 TX, I2C2 SCL, PWM7B
GP32: DIGITAL_IN: DIGITAL_OUT, COM1 TX, SPI RX, I2C SDA, EXT_PWM8A GP33: DIGITAL_IN:
DIGITAL_OUT, COM1 RX, I2C SCL, PWM8B

GP34: DIGITAL_IN: DIGITAL_OUT, SPI SCK, I2C2 SDA, PWM9A GP35:
DIGITAL_IN: DIGITAL_OUT, SPI TX, I2C2 SCL, PWM9B
GP36: DIGITAL_IN: DIGITAL_OUT, COM2 TX, SPI RX, I2C SDA, PWM10A GP37: DIGITAL_IN:
DIGITAL_OUT, COM2 RX, I2C SCL, PWM10B
GP38: DIGITAL_IN: DIGITAL_OUT, SPI SCK, I2C2 SDA, PWM11A GP39:
DIGITAL_IN: DIGITAL_OUT, SPI TX, I2C2 SCL, PWM11B
GP40: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN, COM2 TX, SPI2 RX, I2C SDA, PWM8A GP41: DIGITAL_IN:
DIGITAL_OUT, ANALOG_IN, COM2 RX, I2C SCL, PWM8B
GP42: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN, SPI2 SCK, I2C2 SDA, PWM9A GP43: DIGITAL_IN:
DIGITAL_OUT, ANALOG_IN, SPI2 TX, I2C2 SCL, PWM9B
GP44: DIGITAL_IN: DIGITAL_OUT, COM1TX, ANALOG_IN, SPI2 RX, I2C SDA, PWM10A GP45: DIGITAL_IN:
DIGITAL_OUT, COM1RX, ANALOG_IN, I2C SCL, PWM10B
GP46: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN, SPI2 SCK, I2C2 SDA, PWM11A GP47: DIGITAL_IN:
DIGITAL_OUT, ANALOG_IN, SPI2 TX, I2C2 SCL, PWM11B

RP2354A und RP2354B

Diese werden derzeit von der PicoMite-Firmware nicht unterstützt.

PSRAM

Der RP2350 unterstützt PSRAM, und einige kommerzielle Angebote haben 8 MB PSRAM zu Platinen mit dem 80-poligen RP2350B hinzugefügt (z. B. Pimoroni PGA2350).

Der Zugriff auf den PSRAM erfolgt über denselben Quad-SPI-Bus, der auch vom Flash-Speicher verwendet wird, sodass er vergleichsweise langsam ist, obwohl er über einen Cache gepuffert wird, der dieses Problem mindert. Wenn PSRAM vorhanden und konfiguriert ist, fügt MMBasic ihn dem allgemeinen RAM-Pool hinzu, sodass Programme über eine enorme Menge an allgemeinem RAM verfügen, mit dem sie arbeiten können, auch wenn dies möglicherweise etwas langsamer ist.

Für den Zugriff auf einen PSRAM ist ein zusätzlicher Pin für die Chip-Auswahlfunktion erforderlich, der mit dem Befehl OPTION PSRAM PIN ausgewählt wird. Gültige Pins für die PSRAM-Chip-Auswahl sind GP0, GP8, GP19 und GP47.

Nach dem Einschalten ist der Inhalt des PSRAM unbestimmt – er ist nicht Null. Sie müssen RAM ERASE verwenden, um ihn vor der Verwendung zu löschen. Dieses Verhalten ist beabsichtigt, damit der Inhalt nach einem Reset erhalten bleibt.

E/A-Pin- sbeschränkungen

Die maximale Spannung, die an einen beliebigen I/O-Pin des Raspberry Pi Pico mit RP2040-Prozessor angelegt werden kann, beträgt 3,6 V. Der Raspberry Pi Pico 2 mit RP2350-Prozessor kann 5 V aufnehmen, während der Chip mit Strom versorgt wird.

Als Ausgänge können alle I/O-Pins einzeln bis zu 8 mA liefern oder aufnehmen. Bei dieser Last sinkt die Ausgangsspannung auf etwa 2,3 V. Eine praktischere Last sind 5 mA, bei denen die Ausgangsspannung typischerweise 3 V beträgt. Um eine rote LED mit 5 mA zu betreiben, wird ein Widerstand von 220 Ω empfohlen. Andere Farben erfordern möglicherweise einen anderen Wert.

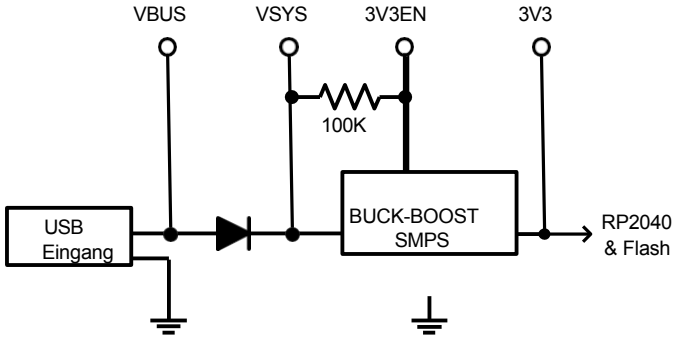
Die maximale Gesamt-I/O-Stromlast für den gesamten Chip beträgt 100 mA.

Strom versorgung

Der Raspberry Pi Pico und Pico 2 verfügen über ein flexibles Stromversorgungssystem.

Die Eingangsspannung von den USB- oder VBUS-Eingängen wird über eine Schottky-Diode an das Buck-Boost-SMPS (Switch Mode Power Supply) mit einer Ausgangsspannung von 3,3 V angeschlossen. Das SMPS akzeptiert Eingangsspannungen von 1,8 V bis 5,5 V, sodass das Gerät mit einer Vielzahl von Stromquellen, einschließlich Batterien, betrieben werden kann.

Externe Schaltungen können über VBUS (normalerweise 5 V) oder über den 3V3-Ausgang (3,3 V) mit einer Stromstärke von bis zu 300 mA versorgt werden.



Für eingebettete Controller-Anwendungen ist in der Regel eine externe Stromquelle (außer USB) erforderlich, die über eine Schottky-Diode an VSYS angeschlossen werden kann. Dadurch kann der Raspberry Pi Pico von

die Quelle mit der höheren Spannung (USB oder VSYS) versorgt werden. Die Dioden verhindern eine Rückkopplung in die Quelle mit der niedrigeren Spannung.

Um Störgeräusche der Stromversorgung zu minimieren, kann 3V3EN geerdet werden, um das SMPS auszuschalten. Beim Herunterfahren stoppt der Wandler den Schaltbetrieb, die interne Steuerschaltung wird ausgeschaltet und die Last getrennt. Sie können die Platine dann über einen linearen 3,3-V-Regler versorgen, der in den 3V3-Pin eingespeist wird.

Takt geschwindigkeit

Standardmäßig beträgt die Taktgeschwindigkeit für den im Raspberry Pi Pico verwendeten RP2040 200 MHz und für den im Raspberry Pi Pico 2 verwendeten RP2350 150 MHz. Dies sind die empfohlenen Höchstwerte.

Mit dem Befehl OPTION CPUSPEED können die meisten RP2040-CPUs auf bis zu 420 MHz und der RP2350 (nur PicoMite und WebMite) auf bis zu 396 MHz übertaktet werden. Sie können auch langsamer mit einer Mindestgeschwindigkeit von 48 MHz laufen. Diese Option wird gespeichert und beim Einschalten erneut angewendet. Beim Ändern der Taktrate wird die PicoMite-Firmware zurückgesetzt und anschließend neu gestartet, sodass die USB-Verbindung getrennt wird.

Bei VGA/HDMI-Versionen mit einer Videoauflösung von 640 x 400 wird die Taktrate auf 252 MHz eingestellt, dies kann jedoch über OPTION RESOLUTION auf 252 MHz, 315 MHz oder 378 MHz geändert werden. Bei anderen Videoauflösungen ist die Taktrate je nach ausgewählter Videoauflösung auf 283,2 MHz, 324 MHz, 360 MHz, 372 MHz oder 375 MHz festgelegt und kann nicht geändert werden.

Fast alle getesteten Raspberry Pi Picos haben bei 380 MHz oder mehr einwandfrei funktioniert, sodass eine Übertaktung sinnvoll sein kann. Wenn der Prozessor bei seiner neuen Taktrate nicht neu startet, können Sie ihn zurücksetzen, indem Sie Clear_flash.uf2 laden, um den Pico auf seinen Werkszustand zurückzusetzen (siehe Abschnitt *„Laden der Firmware“* oben).

Leistungs sverbrauch

Der Stromverbrauch hängt von der Taktrate ab, bei der Standardtaktrate (200 MHz für den RP2040 und 150 MHz für den RP2350) beträgt der typische Stromverbrauch jedoch 25 mA. Darin nicht enthalten ist der Strom, der von den I/O-Pins oder dem 3V3-Pin bezogen oder abgegeben wird:

Der Stromverbrauch der WebMite-Version für den Raspberry Pi Pico W beträgt bei deaktiviertem WLAN ebenfalls 20 mA, bei aktiviertem WLAN steigt der Stromverbrauch jedoch auf 40 bis 70 mA.

Verwendung von MMBasic

Befehle und Programm seingabe

An der Befehlszeile können Sie einen Befehl eingeben, der sofort ausgeführt wird. In den meisten Fällen werden Sie dies tun, um die PicoMite-Firmware anzuweisen, etwas zu tun, z. B. ein Programm auszuführen oder eine Option festzulegen. Mit dieser Funktion können Sie jedoch auch Befehle an der Befehlszeile testen.

Die einfachste Methode zur Eingabe eines Programms ist die Verwendung des Befehls EDIT. Dadurch wird der Vollbild-Programmeditor aufgerufen, der in die PicoMite-Firmware integriert ist und später in diesem Handbuch beschrieben wird. Er umfasst erweiterte Funktionen wie Suchen und Kopieren, Ausschneiden und Einfügen in die Zwischenablage.

Sie können das Programm auch auf Ihrem Desktop-Computer mit einem Programm wie Notepad erstellen und es dann über die Protokolle XModem oder YModem (siehe Befehl XMODEM oder YMODEM) oder per Streaming über die serielle Konsolenverbindung (siehe Befehl AUTOSAVE) auf den Raspberry Pi Pico übertragen.

Eine dritte und bequeme Methode zum Schreiben und Debuggen eines Programms ist die Verwendung von MMEdit. Dabei handelt es sich um ein Programm, das auf Ihrem Windows-Computer läuft und es Ihnen ermöglicht, Ihr Programm auf Ihrem Computer zu bearbeiten und es dann mit einem einzigen Mausklick auf den PicoMite zu übertragen. MMEdit wurde von Jim Hiley geschrieben und kann kostenlos heruntergeladen und verwendet werden. Weitere Informationen finden Sie unter: https://geoffg.net/mmedit.html

Eine Sache, die Sie nicht tun können, ist die Verwendung der alten BASIC-Methode zur Eingabe eines Programms, bei der jeder Zeile eine Zeilennummer vorangestellt wurde. Zeilennummern sind in MMBasic optional, Sie können sie also weiterhin verwenden, wenn Sie möchten, aber wenn Sie eine Zeile mit einer Zeilennummer an der Eingabeaufforderung eingeben, führt MMBasic diese einfach sofort aus.

Programm struktur

Ein BASIC-Programm beginnt an der ersten Zeile und läuft so lange, bis es das Ende des Programms erreicht oder auf einen END-Befehl stößt. An diesem Punkt zeigt MMBasic die Eingabeaufforderung (>) auf der Konsole an und wartet auf eine Eingabe.

Ein Programm besteht aus einer Reihe von Anweisungen oder Befehlen, von denen jeder den BASIC-Interpreter dazu veranlasst, etwas auszuführen (die Begriffe „Anweisung" und „Befehl" haben im Allgemeinen dieselbe Bedeutung und werden synonym verwendet). Normalerweise steht jede Anweisung in einer eigenen Zeile, aber Sie können auch mehrere Anweisungen in einer Zeile haben, die durch das Doppelpunktzeichen (:) getrennt sind. Zum Beispiel.

```
A = 24,6 : PRINT A
```

Anhang J (*Programmieren in BASIC – Ein Tutorial*) am Ende dieses Handbuchs enthält ein umfassendes Tutorial zu dieser Sprache, das Ihnen die Grundlagen in einem leicht verständlichen Format mit vielen Beispielen vermittelt.

Bearbeiten der Befehls szeile

Wenn Sie eine Zeile an der Befehlszeile eingeben, können Sie diese mit den Pfeiltasten nach links und rechts bearbeiten, indem Sie sich entlang der Zeile bewegen, mit der Entf-Taste Zeichen löschen, mit der Rücktaste Zeichen vor dem Cursor löschen und mit der Einfügen-Taste zwischen Einfüge- und Überschreibmodus wechseln. Mit Home/Ende gelangen Sie zum Anfang/Ende der Zeile, und wenn Sie zweimal Home drücken, wird die Bearbeitung beendet. Mit der Eingabetaste senden Sie die Zeile jederzeit an MMBasic, das sie ausführt.

Mit den Aufwärts- und Abwärtspfeiltasten können Sie durch den Verlauf zuvor eingegebener Befehlszeilen navigieren, die bearbeitet und wiederverwendet werden können.

Tastenkombinationen tasten

Die Funktionstasten der Tastatur, die für die Konsole verwendet werden, können an der Eingabeaufforderung verwendet werden, um häufig verwendete Befehle automatisch einzugeben. Diese Funktionstasten fügen den Text ein, gefolgt von der Eingabetaste, sodass der Befehl sofort ausgeführt wird:

F2 RUN
F3 LIST
F4 BEARBEITEN
F5 Sendet eine ESC-Sequenz, um den VT100-Bildschirm zu löschen. Löscht auch die
Konsole. F10 AUTOSAVE
F11 XMODEM EMPFANGEN
F12 XMODEM SEND

Die Funktionstasten F1 und F5 bis F9 können mit benutzerdefiniertem Text programmiert werden. Siehe den Befehl OPTION FNKey.

Unterbrechen eines laufenden MMBasic-Programms

Ein Programm wird mit dem Befehl RUN gestartet. Sie können MMBasic und das laufende Programm jederzeit unterbrechen, indem Sie STRG-C auf der Konsole eingeben. MMBasic kehrt dann zur Befehlszeile zurück.

Festlegen von MMBasic-Optionen

Viele Optionen können mit Befehlen eingestellt werden, die mit dem Schlüsselwort OPTION beginnen. Sie sind in einem eigenen Abschnitt dieses Handbuchs aufgeführt. Bei einigen Firmware-Versionen können Sie beispielsweise die CPU-Taktrate mit dem folgenden Befehl ändern:

```
OPTION CPUSPEED Geschwindigkeit
```

Gespeicherte Option-Variablen

Häufig müssen Daten gespeichert werden, die nach Wiederherstellung der Stromversorgung wiederhergestellt werden können. Dazu gehören beispielsweise Programmoptionen, Kalibrierungseinstellungen usw. Dies kann mit dem Befehl VAR SAVE erfolgen, der die in seiner Befehlszeile aufgeführten Variablen in einem nichtflüchtigen Flash-Speicher speichert. Der für gespeicherte Variablen reservierte Speicherplatz beträgt 16 KB.

Diese Variablen können mit dem Befehl VAR RESTORE wiederhergestellt werden, der alle gespeicherten Variablen zur Variablentabelle des laufenden Programms hinzufügt. Normalerweise wird dieser Befehl am Anfang eines Programms platziert, damit die Variablen für die Verwendung durch das Programm bereitstehen.

Diese Funktion dient zum Speichern von Kalibrierungsdaten, vom Benutzer ausgewählten Optionen und anderen Elementen, die sich nur selten ändern. Sie sollte nicht für Hochgeschwindigkeitsspeicherungen verwendet werden, da dies zu einer Abnutzung des Flash-Speichers führen kann. Der für den Raspberry Pi Pico verwendete Flash-Speicher hat eine hohe Lebensdauer, die jedoch durch ein Programm, das wiederholt Variablen speichert, überschritten werden kann. Wenn Sie Daten häufig speichern möchten, sollten Sie einen Echtzeituhr-Chip hinzufügen. Mit den RTC-Befehlen können dann Daten im batteriegepufferten Speicher der Echtzeituhr gespeichert und abgerufen werden. Weitere Informationen finden Sie unter dem Befehl RTC.

Gespeicherte Variablen werden gelöscht, wenn ein neues Programm geladen wird.

Watchdog- -Timer

Eine der Anwendungen für den Raspberry Pi Pico ist der Einsatz als eingebetteter Controller. Er kann in MMBasic programmiert werden, und wenn das Programm debuggt und einsatzbereit ist, kann die Konfigurationseinstellung OPTION AUTORUN aktiviert werden. Das Modul führt dann sein Programm automatisch aus, sobald es mit Strom versorgt wird, und fungiert als benutzerdefinierte Schaltung, die eine bestimmte Aufgabe ausführt. Der Benutzer muss nichts darüber wissen, was im Inneren abläuft.

Es besteht jedoch die Möglichkeit, dass ein Fehler im Programm dazu führt, dass MMBasic einen Fehler generiert und zur Eingabeaufforderung zurückkehrt. Dies wäre in einer eingebetteten Situation wenig sinnvoll, da das Gerät nicht mit der Konsole verbunden wäre. Eine weitere Möglichkeit besteht darin, dass das BASIC-Programm aus irgendeinem Grund in einer Endlosschleife hängen bleibt. In beiden Fällen wäre der sichtbare Effekt derselbe: Das Programm würde so lange nicht mehr laufen, bis die Stromversorgung unterbrochen und wiederhergestellt wird.

Um dies zu verhindern, kann der Watchdog-Timer verwendet werden. Dabei handelt es sich um einen Timer, der bis Null herunterzählt und bei Erreichen von Null die Prozessoren automatisch neu startet (genau wie beim ersten Einschalten). Dies geschieht auch dann, wenn MMBasic an der Eingabeaufforderung steht. Nach dem Neustart wird die automatische Variable MM.WATCHDOG auf „true" gesetzt, um anzuzeigen, dass der Neustart durch ein Watchdog-Timeout verursacht wurde.

Der Befehl WATCHDOG sollte an strategischen Stellen im Programm platziert werden, um den Timer immer wieder zurückzusetzen und so zu verhindern, dass er bis Null herunterzählt. Wenn dann ein Fehler auftritt, wird der Timer nicht zurückgesetzt, sondern zählt bis Null herunter und das Programm wird neu gestartet (vorausgesetzt, die Option AUTORUN ist aktiviert).

PIN- ssicherheit

Manchmal ist es wichtig, die Daten und Programme in einem eingebetteten Controller vertraulich zu behandeln. In der PicoMite-Firmware kann dies mit dem Befehl OPTION PIN erreicht werden. Dieser Befehl legt eine PIN-Nummer fest (die im Flash-Speicher gespeichert wird), und wenn die PicoMite-Firmware (aus welchem Grund auch immer) zur Eingabeaufforderung zurückkehrt, wird der Benutzer an der Konsole aufgefordert, die PIN-Nummer einzugeben. Ohne die richtige PIN kann der Benutzer nicht zur Befehlszeile gelangen und hat nur die Möglichkeit, die richtige PIN einzugeben oder die PicoMite-Firmware neu zu starten. Nach dem Neustart benötigt der Benutzer weiterhin die richtige PIN, um auf die Befehlszeile zugreifen zu können.

Da ein Eindringling die Befehlszeile nicht erreichen kann, kann er kein Programm auflisten oder kopieren, das Programm nicht ändern und auch keine Änderungen an MMBasic oder der PicoMite-Firmware vornehmen. Einmal festgelegt, kann die PIN nur durch Eingabe der ursprünglich festgelegten korrekten PIN entfernt werden. Wenn die Nummer verloren geht, besteht die einzige Möglichkeit zur Wiederherstellung darin, die PicoMite-Firmware neu zu laden (wodurch das Programm und alle Optionen gelöscht werden).

Es gibt andere zeitaufwändige Möglichkeiten, auf die Daten zuzugreifen (z. B. die Verwendung eines Programmiergeräts zur Untersuchung des Flash-Speichers), daher sollte dies nicht als ultimative Sicherheitsmaßnahme angesehen werden, aber es wirkt als erhebliche Abschreckung.

Die Bibliothek „ ”

Mit der LIBRARY-Funktion können BASIC-Funktionen, Unterprogramme und eingebettete Schriftarten erstellt und zu MMBasic hinzugefügt werden, um sie dauerhaft und zu einem Teil der Sprache zu machen. Sie haben beispielsweise eine Reihe von Unterprogrammen und Funktionen geschrieben, die komplexe Bitmanipulationen durchführen. Diese können als Bibliothek gespeichert werden, werden Teil von MMBasic und funktionieren genauso wie andere integrierte Funktionen, die bereits Teil der Sprache sind. Eine eingebettete Schriftart kann auf die gleiche Weise hinzugefügt und wie eine normale Schriftart verwendet werden.

Um Komponenten in die Bibliothek zu installieren, müssen Sie die Routinen wie normale BASIC-Routinen schreiben und testen. Wenn sie korrekt funktionieren, können Sie den Befehl LIBRARY SAVE verwenden. Dadurch werden die Routinen (so viele Sie möchten) in einen nicht sichtbaren Teil des Flash-Speichers übertragen, wo sie für jedes BASIC-Programm verfügbar sind, aber nicht angezeigt werden, wenn der Befehl LIST verwendet wird, und nicht gelöscht werden, wenn ein neues Programm geladen oder NEW verwendet wird. Die gespeicherten Unterprogramme und Funktionen können jedoch aus dem Hauptprogramm heraus aufgerufen und sogar an der Befehlszeile ausgeführt werden (genau wie ein integrierter Befehl oder eine integrierte Funktion).

Einige Punkte, die zu beachten sind:

- Bibliotheksroutinen verhalten sich genau wie normaler BASIC-Code und können aus einer beliebigen Anzahl von Unterprogrammen, Funktionen, eingebetteten C-Routinen und Schriftarten bestehen. Der einzige Unterschied besteht darin, dass sie bei der Auflistung eines Programms nicht angezeigt und beim Laden eines neuen Programms nicht gelöscht werden.
- Bibliotheksroutinen können globale Variablen erstellen und darauf zugreifen und unterliegen denselben Regeln wie das Hauptprogramm – beispielsweise müssen sie OPTION EXPLICIT beachten, wenn diese Option gesetzt ist.
- Wenn die Routinen in die Bibliothek übertragen werden, komprimiert MMBasic sie, indem Kommentare, zusätzliche Leerzeichen, Leerzeilen und die Hex-Codes in eingebetteten C-Routinen und Schriftarten entfernt werden. Dadurch wird der Bibliotheksspeicherplatz effizient genutzt, insbesondere beim Laden großer Schriftarten. Nach dem Speichern wird der Programmbereich gelöscht.
- Sie können den Befehl LIBRARY SAVE mehrmals verwenden. Bei jedem Speichern wird der neue Inhalt des Programmbereichs an den bereits vorhandenen Code in der Bibliothek angehängt.
- Sie können Zeilennummern in der Bibliothek verwenden, aber Sie können keine Zeilennummer in einer ansonsten leeren Zeile als Ziel für einen GOTO-Befehl usw. verwenden. Der Grund dafür ist, dass der Befehl LIBRARY SAVE alle Leerzeilen entfernt.
- Sie können READ-Befehle in der Bibliothek verwenden, aber diese lesen standardmäßig DATA-Anweisungen im Hauptprogrammspeicher. Wenn Sie aus DATA-Anweisungen in der Bibliothek lesen möchten, müssen Sie vor dem ersten READ-Befehl den Befehl RESTORE verwenden. Dadurch wird der Zeiger auf den Bibliotheksbereich zurückgesetzt.
- Die Bibliothek wird im Programm-Flash-Speicher Slot 3 gespeichert und steht daher nicht für die Speicherung eines Programms zur Verfügung, wenn LIBRARY SAVE verwendet wird.
- Mit dem Befehl LIBRARY LIST können Sie den Inhalt der Bibliothek anzeigen, da dieser Befehl den Inhalt des Bibliotheksspeicherplatzes auflistet.
- Der Inhalt der LIBRARY kann mit LIBRARY DISK SAVE fname$ auf der Festplatte gespeichert und mit LIBRARY DISK LOAD fname$ wiederhergestellt werden.

Um die Routinen im Bibliotheksspeicher zu löschen, verwenden Sie den Befehl LIBRARY DELETE. Dadurch wird der Speicherplatz freigegeben und der von der Bibliothek belegte Flash-Speicherplatz 3 steht wieder für die Speicherung normaler Programme zur Verfügung. Die einzige andere Möglichkeit, eine Bibliothek zu löschen, ist die Verwendung von OPTION RESET.

Initialisierung von Programm- en

Die Bibliothek kann auch Code enthalten, der nicht in einer Subroutine oder Funktion enthalten ist. Dieser Code (sofern vorhanden) wird automatisch ausgeführt, bevor ein Programm gestartet wird (d. h. über den Befehl RUN). Diese Funktion kann verwendet werden, um Konstanten zu initialisieren oder MMBasic auf bestimmte Weise einzurichten. Wenn Sie beispielsweise einige Konstanten festlegen möchten, können Sie die folgenden Zeilen in den Bibliothekscode einfügen:

```
CONST TRUE = 1
CONST FALSE = 0
```

Die Bezeichner TRUE und FALSE wurden der Sprache hinzugefügt und stehen für alle Zwecke in jedem ausgeführten Programm zur Verfügung.

MM.STARTUP

Es kann erforderlich sein, beim ersten Einschalten einen bestimmten Code auszuführen, beispielsweise um bestimmte Hardware zu initialisieren, Optionen festzulegen oder einen benutzerdefinierten Startbildschirm anzuzeigen. Dies kann durch Erstellen einer Subroutine mit dem Namen MM.STARTUP erreicht werden. Wenn die PicoMite-Firmware zum ersten Mal eingeschaltet oder zurückgesetzt wird, sucht sie nach dieser Subroutine und führt sie einmal aus, sofern sie gefunden wird.

Wenn beispielsweise an den Raspberry Pi Pico eine Echtzeituhr angeschlossen ist, könnte das Programm den folgenden Code enthalten:

```
SUB MM.STARTUP RTC
   GETTIME
END SUB
```

Dadurch würde die interne Uhr in MMBasic bei jedem Einschalten oder Zurücksetzen auf die aktuelle Uhrzeit eingestellt werden.

Nachdem der Code in MM.STARTUP ausgeführt wurde, fährt MMBasic mit der Ausführung des restlichen Programms im Programmspeicher fort. Wenn kein weiterer Code vorhanden ist, kehrt MMBasic zur Befehlszeile zurück.

Beachten Sie, dass Sie MM.STARTUP nicht für allgemeine Einstellungen von MMBasic (wie das Dimensionieren von Arrays, das Öffnen von Kommunikationskanälen usw.) vor dem Ausführen eines Programms verwenden sollten. Der Grund dafür ist, dass MMBasic bei Verwendung des Befehls RUN zunächst den Status des Interpreters löscht, um einen Neuanfang zu ermöglichen.

MM.PROMPT

Wenn eine Subroutine mit diesem Namen vorhanden ist, wird sie automatisch von MMBasic ausgeführt, anstatt die Befehlszeile anzuzeigen. Dies kann verwendet werden, um eine benutzerdefinierte Eingabeaufforderung anzuzeigen, Farben festzulegen, Variablen zu definieren usw., die alle in der Befehlszeile aktiv sind.

Beachten Sie, dass MMBasic alle Variablen und I/O-Pin-Einstellungen löscht, wenn ein Programm ausgeführt wird, sodass alle in dieser Unterroutine festgelegten Einstellungen nur für Befehle gültig sind, die an der Befehlszeile (d. h. im Sofortmodus) eingegeben werden.

Als Beispiel wird im Folgenden eine benutzerdefinierte Eingabeaufforderung angezeigt:

```
SUB MM.PROMPT PRINT
   TIME$ "> ";
END SUB
```

Beachten Sie, dass Konstanten zwar definiert werden können, aber nicht sichtbar sind, da eine innerhalb einer Subroutine definierte Konstante lokal für diese Subroutine ist. DIM erstellt jedoch globale Variablen, die stattdessen verwendet werden sollten.

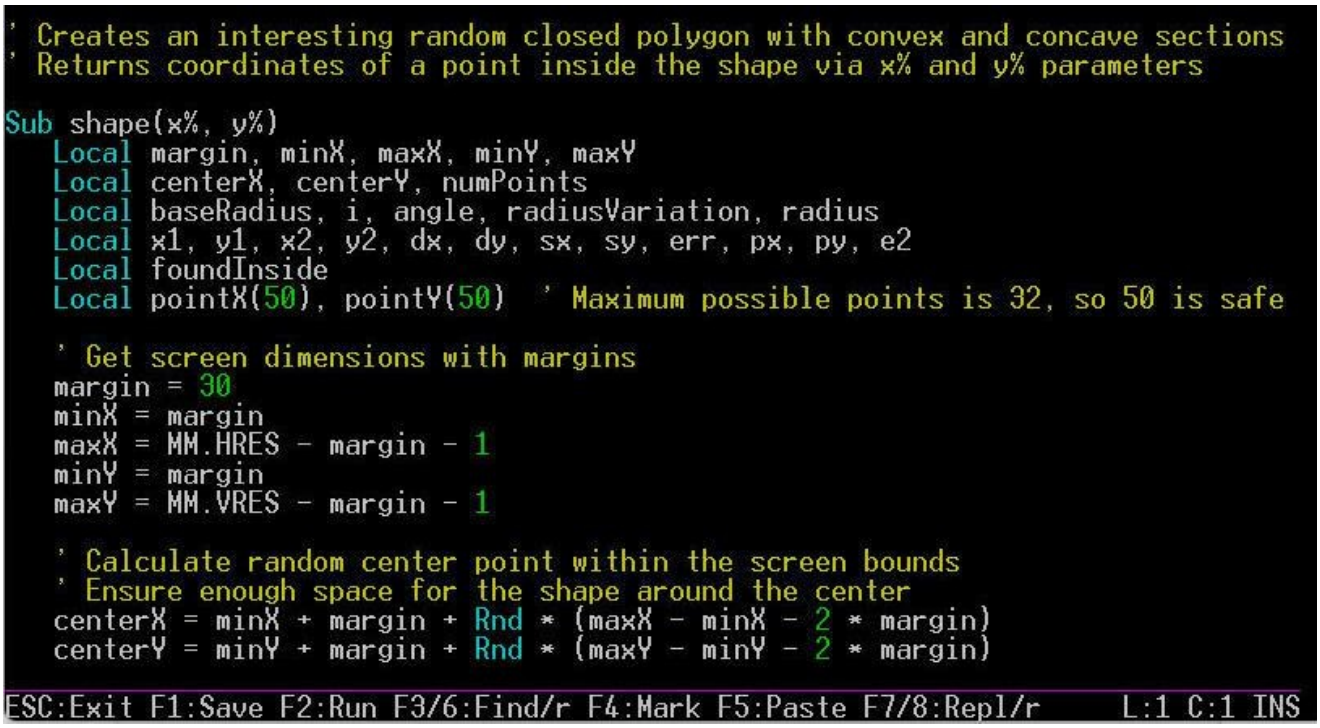
MM.END

Wenn im Programm eine Unterroutine namens MM.END vorhanden ist, wird diese immer dann ausgeführt, wenn das Programm mit einem tatsächlichen oder implizierten END-Befehl endet. Sie wird nicht ausgeführt, wenn das Programm mit Strg-C beendet wird.

Der optionale Parameter „noend” des END-Befehls kann verwendet werden, um die Ausführung der MM.END-Subroutine bei Bedarf zu blockieren (weitere Informationen finden Sie unter dem END-Befehl).

Vollbild- -Editor

Eine wichtige Produktivitätsfunktion ist der integrierte Vollbild-Editor. Wenn er ausgeführt wird, sieht er wie folgt aus:



Der Editor funktioniert mit:

- Der seriellen Konsole unter Verwendung eines VT100-unterstützten Terminalemulators (wie Tera Term) in allen Versionen.
- Der VGA- oder HDMI-Videoausgang (bei Versionen mit dieser Funktion).
- Einem angeschlossenen LCD-Panel, das mit OPTION LCDPANEL CONSOLE konfiguriert wurde.

Beim Start des Editors wird der Cursor automatisch an der Stelle positioniert, an der Sie zuletzt bearbeitet haben, oder, falls Ihr Programm gerade aufgrund eines Fehlers angehalten wurde, an der Zeile, die den Fehler verursacht hat.
Am unteren Rand des Bildschirms werden in der Statuszeile Details wie die aktuelle Cursorposition und die vom Editor unterstützten allgemeinen Funktionen angezeigt.

Wenn Sie zuvor einen Editor wie Windows Notepad verwendet haben, werden Sie feststellen, dass die Bedienung dieses Editors Ihnen vertraut ist. Mit den Pfeiltasten können Sie den Cursor im Text bewegen, mit „Home“ und „End“ gelangen Sie zum Anfang bzw. Ende der Zeile. Mit „Bild auf“ und „Bild ab“ können Sie wie der Name schon sagt die Seiten nach oben bzw. unten blättern. Mit der Entf-Taste löschen Sie das Zeichen an der Cursorposition, mit der Rücktaste das Zeichen vor dem Cursor. Mit der Einfügen-Taste können Sie zwischen dem Einfüge- und dem Überschreibmodus umschalten. Die einzige ungewöhnliche Tastenkombination ist, dass Sie mit zweimaligem Drücken der Home-Taste zum Anfang des Programms und mit zweimaligem Drücken der Ende-Taste zum Ende gelangen.

Am unteren Rand des Bildschirms werden in der Statuszeile die verschiedenen vom Editor verwendeten Funktionstasten und ihre Funktionen aufgelistet. Im Einzelnen sind dies:

ESC	Dadurch verwirft der Editor alle Änderungen und kehrt zur Befehlszeile zurück, ohne den Programmspeicher zu verändern. Wenn Sie den Text geändert haben, werden Sie gefragt, ob Sie Ihre Änderungen wirklich verwerfen möchten.
F1: SPEICHERN	Damit wird der Text im Programmspeicher gespeichert und die Eingabeaufforderung wieder angezeigt. F2: AUSFÜHREN Dadurch wird der Text im Programmspeicher gespeichert und sofort ausgeführt.
F3: SUCHEN	Hiermit wird der Text abgefragt, nach dem Sie suchen möchten. Wenn Sie die Eingabetaste drücken, wird der Cursor an den Anfang des ersten gefundenen Eintrags gesetzt.
F4: MARKIEREN	Dies wird im Folgenden ausführlich beschrieben.
F5: EINFÜGEN	Damit wird der zuvor ausgeschnittene oder kopierte Text (siehe unten) an der aktuellen Cursorposition eingefügt.
F6: ERNEUT SUCHEN	Nachdem Sie die Suchfunktion verwendet haben, können Sie die Suche durch Drücken von F6 wiederholen (Shift-F3 ist ebenfalls gültig).

F7: ERSETZEN Befindet sich der Cursor hinter F3 oder F6, öffnet sich ein Dialogfeld, in dem Sie eine Zeichenfolge direkt in die Zwischenablage eingeben und durch Drücken der Eingabetaste die gesuchte Zeichenfolge ersetzen können.

F8: ERNEUT ERSETZEN Wenn sich der Cursor hinter F3 oder F6 befindet, ersetzt das System die gesuchte Zeichenfolge durch den Inhalt des Zwischenablagepuffers.

Wenn Sie die Markierungstaste (F4) gedrückt haben, wechselt der Editor in den Markierungsmodus. In diesem Modus können Sie mit den Pfeiltasten einen Textabschnitt markieren, der dann invers hervorgehoben wird. Anschließend können Sie den markierten Text löschen, ausschneiden oder kopieren. In diesem Modus ändert sich die Statuszeile und zeigt die Funktionen der Funktionstasten im Markierungsmodus an.

Diese Tasten sind:

ESC Beendet den Markierungsmodus, ohne etwas zu ändern.

F4: AUSSCHNEIDEN Kopiert den markierten Text in die Zwischenablage und entfernt ihn aus dem

Programm. F5: KOPIEREN Kopiert den markierten Text nur in die Zwischenablage.

DELETE Löscht den markierten Text, ohne die Zwischenablage zu verändern.

Sie können auch Steuertasten anstelle der oben aufgeführten Funktionstasten verwenden. Diese Steuertasten sind:

LINKS	Strg-S	RECHTS	Strg-D	AUF	Strg-E	DOWN	Strg-X
HOME	Strg-U	END	Strg-K	Bild auf	Strg-P	Bild nach unten	Strg-L
Entf	Strg-]	EINFÜGEN	Strg-N	F1	Strg-Q	F2	Strg-W
F3	Strg-R	F4	Strg-T	F5	Strg-Y	F6	Strg-G
F7	Strg-F	F8	Strg-I				

Der beste Weg, um den Umgang mit dem Editor zu erlernen, ist, ihn einfach zu starten und damit zu experimentieren.

Der Editor ist eine sehr produktive Methode zum Schreiben eines Programms. Mit dem Befehl EDIT können Sie Ihr Programm eingeben und dann durch Drücken der Taste F2 das Programm speichern und ausführen. Wenn Ihr Programm mit einem Fehler stoppt, laden Sie durch Drücken der Funktionstaste F4 an der Eingabeaufforderung den Editor und positionieren den Cursor an der Zeile, die den Fehler verursacht hat. Dieser Zyklus aus Bearbeiten/Ausführen/Bearbeiten ist sehr schnell.

Lang zeilen

Bei langen Zeilen wird nur der erste Teil der Zeile bis zum rechten Rand des Displays angezeigt. Der Rest der Zeile jenseits des rechten Randes ist zwar noch vorhanden, wird aber nicht angezeigt und kann nicht bearbeitet werden. Wenn Sie eine sehr lange Zeile bearbeiten möchten, können Sie den Cursor nahe am rechten Rand positionieren und die Eingabetaste drücken. Dadurch wird die lange Zeile in zwei Teile geteilt, die beide separat bearbeitet werden können. Um die Zeile wieder zusammenzufügen, entfernen Sie mit der Entf- oder Rücktaste den zuvor eingegebenen Zeilenumbruch.

Alternativ können Sie vor dem Aufrufen des Editors die Fortsetzungszeilen aktivieren (OPTION CONTINUATION LINES ON). Damit können Sie am Ende einer Zeile ein Leerzeichen gefolgt von einem Unterstrich verwenden, um anzuzeigen, dass die nächste Zeile eine Fortsetzung ist und angehängt werden muss, damit das Programm ausgeführt werden kann. Das Anhängen erfolgt beim Speichern der Datei, und beim erneuten Bearbeiten werden lange Zeilen automatisch geteilt, wenn die Datei in den Editor eingelesen wird. Die Zeilenumbrüche befinden sich möglicherweise nicht an derselben Stelle, aber der Editor versucht, sie an sinnvollen Stellen (am Ende von Wörtern usw.) zu platzieren. Es gibt keine Einschränkungen hinsichtlich der Platzierung von Fortsetzungszeichen. Sie können beispielsweise in der Mitte einer Zeichenfolge in Anführungszeichen stehen. Die Begrenzung auf maximal 255 Zeichen in einer verketteten Zeile gilt weiterhin, und der Editor lässt Sie nicht aus der Datei heraus, wenn eine Zeile zu lang ist.

Verwendung einer PS/2- oder USB-Maus

Versionen der PicoMite-Firmware, die VGA/HDMI unterstützen (sowohl in der RP2040- als auch in der RP2350-Version), unterstützen auch die Verwendung einer PS2- oder USB-Maus im Editor. Einzelheiten zum Anschließen einer Maus finden Sie unter der Überschrift *„Tastatur/Maus/Gamepad“* weiter unten in diesem Handbuch.

Wenn Sie den Editor mit angeschlossener Maus starten und sich im Videomodus 1 mit aktivierter Farbcodierung befinden, sehen Sie ein Zeichen, das auf weißem Hintergrund rot hervorgehoben ist. Diese Markierung kann mit der Maus verschoben werden. Durch Klicken mit der linken Maustaste wird der Bearbeitungscursor an diese Position verschoben (d. h. wie bei Verwendung der Cursortasten). Ein Klick mit der rechten Maustaste entspricht dem Drücken der Taste F4 auf der Tastatur, und ein Klick mit dem Scrollrad entspricht der Taste F5.

Das bedeutet, dass Sie im Normalmodus des Editors den Mauszeiger positionieren können und durch einen Rechtsklick in den Markierungsmodus (Ausschneiden und Kopieren) wechseln, wobei der Cursor an der Position des Mauszeigers beginnt. Wenn Sie dann die Maus bewegen und mit der linken Maustaste klicken, werden die Zeichen von der Markierungsposition bis zur neuen Mausposition markiert.

Ein Rechtsklick (entspricht F4) schneidet den markierten Bereich in die Zwischenablage, während ein Klick auf das Scrollrad (entspricht F5) den markierten Text in die Zwischenablage kopiert, ohne ihn aus dem Text zu löschen. Beide Aktionen bringen den Editor wieder in den Normalmodus zurück.

Im normalen Modus kann der Inhalt der Zwischenablage in den Text eingefügt werden, indem Sie die Maus an die neue Position bewegen und auf das Scrollrad klicken (entspricht F5).

Farbcodierte Editor- -Anzeige

Der Editor kann das bearbeitete Programm mit Schlüsselwörtern, Zahlen und Kommentaren in verschiedenen Farben farblich kennzeichnen. Diese Funktion kann mit dem folgenden Befehl ein- oder ausgeschaltet werden:

```
OPTION COLOURCODE ON          oder    OPTION COLOURCODE OFF
```

Diese Option wird im nichtflüchtigen Speicher gespeichert und beim Start automatisch angewendet.

Variablen und Ausdrücke in „ “

In MMBasic wird bei Befehlsnamen, Funktionsnamen, Bezeichnungen, Variablennamen, Dateinamen usw. nicht zwischen Groß- und Kleinschreibung unterschieden, sodass „Run” und „RUN” gleichbedeutend sind und „dOO” und „Doo” sich auf dieselbe Variable beziehen.

Variablen

Variablen können mit einem Buchstaben oder einem Unterstrich beginnen und beliebige Buchstaben oder Ziffern, den Punkt (.) und den Unterstrich (_) enthalten. Sie können bis zu 31 Zeichen lang sein.

Ein Variablenname oder eine Bezeichnung darf nicht mit einem Befehl, einer Funktion, einer der neun PIO-Anweisungen (IN, OUT, JMP, WAIT, PUSH, PULL, MOV, IRQ, SET) oder einem der folgenden Schlüsselwörter identisch sein: THEN, ELSE, GOTO, GOSUB, TO, STEP, FOR, WHILE, UNTIL, LOAD, MOD, NOT, AND, OR, XOR, AS.
Beispielsweise ist step = 5 unzulässig, da STEP ein

Schlüsselwort ist. MMBasic unterstützt drei Arten von

Variablen:

1. Doppelte Genauigkeit mit Gleitkomma.
   Diese können eine Zahl mit Dezimalpunkt und Bruchteil speichern (z. B. 45,386), verlieren jedoch an Genauigkeit, wenn mehr als 14 Stellen verwendet werden. Gleitkommavariablen werden durch Hinzufügen des Suffixes „!“ zum Namen einer Variablen angegeben (z. B. i!, nbr! usw.). Sie sind auch die Standardeinstellung, wenn eine Variable ohne Suffix erstellt wird (z. B. i, nbr usw.).
2. 64-Bit-Ganzzahl mit Vorzeichen.
   Diese können positive oder negative Zahlen mit bis zu 19 Dezimalstellen ohne Genauigkeitsverlust speichern, jedoch keine Brüche (d. h. den Teil nach dem Dezimalpunkt). Sie werden durch Hinzufügen des Suffixes „%” zum Namen einer Variablen angegeben. Zum Beispiel i%, nbr% usw.
3. Eine Zeichenfolge.
   Eine Zeichenkette speichert eine Folge von Zeichen (z. B. „Tom”). Jedes Zeichen in der Zeichenkette wird als 8-Bit-Zahl gespeichert und kann daher einen Dezimalwert von 0 bis 255 haben. Namen von Zeichenkettenvariablen werden mit dem Symbol „$” abgeschlossen (z. B. name$, s$ usw.). Zeichenketten können bis zu 255 Zeichen lang sein.

Beachten Sie, dass es nicht zulässig ist, denselben Variablennamen für verschiedene Typen zu verwenden. Die Verwendung von `nbr!` und `nbr%` im selben Programm würde beispielsweise einen Fehler verursachen.

Die meisten Programme verwenden Fließkomma-Variablen für arithmetische Operationen, da diese mit den in typischen Situationen verwendeten Zahlen umgehen können und bei Divisionen und Brüchen intuitiver sind als Ganzzahlen. Wenn Sie sich also nicht um die Details kümmern müssen, verwenden Sie immer Fließkomma.

Konstanten

Numerische Konstanten können mit einer Ziffer (0-9) für eine Dezimalkonstante, mit &H für eine Hexadezimalzahl, mit &O für eine Oktalzahl oder mit &B für eine Binärzahl beginnen. Beispielsweise entspricht &B1000 der Dezimalkonstante 8. Konstanten, die mit &H, &O oder &B beginnen, werden immer als 64-Bit-Ganzzahlkonstanten ohne Vorzeichen behandelt.

Dezimalen können mit einem Minuszeichen (-) oder Pluszeichen (+) beginnen und mit „E” gefolgt von einer Exponentialzahl enden, um die Exponentialdarstellung anzuzeigen. Beispielsweise entspricht 1,6E+4 dem Wert 16000.

Wenn eine konstante Zahl verwendet wird, wird davon ausgegangen, dass es sich um eine Ganzzahl handelt, sofern kein Dezimalpunkt oder Exponent verwendet wird. Beispielsweise wird 1234 als Ganzzahl interpretiert, während 1234,0 als Gleitkommazahl interpretiert wird.

Zeichenfolgenkonstanten müssen in doppelte Anführungszeichen (") gesetzt werden. Beispiel: "Hello World".

OPTION DEFAULT

Eine Variable kann ohne Suffix (d. h. !, % oder $) verwendet werden. In diesem Fall verwendet MMBasic den Standardtyp „Gleitkomma”. Mit dem folgenden Befehl wird beispielsweise eine Gleitkommavariable erstellt:

```
Nbr = 1234
```

Der Standardwert kann jedoch mit dem Befehl OPTION DEFAULT geändert werden. Beispielsweise legt OPTION DEFAULT INTEGER fest, dass alle Variablen ohne einen bestimmten Typ Ganzzahlen sind. Mit dem folgenden Befehl wird also eine Ganzzahlvariable erstellt:

```
OPTION DEFAULT INTEGER Nbr =
1234
```

Der Standardwert kann auf FLOAT (Standardwert bei Ausführung eines Programms), INTEGER, STRING oder NONE gesetzt werden. Im letzteren Fall müssen alle Variablen explizit typisiert werden, da sonst ein Fehler auftritt.

Der Befehl OPTION DEFAULT kann an beliebiger Stelle im Programm platziert und jederzeit geändert werden. Es empfiehlt sich jedoch, ihn am Anfang des Programms zu platzieren und unverändert zu lassen.

OPTION EXPLICIT

Standardmäßig erstellt MMBasic automatisch eine Variable, wenn sie zum ersten Mal referenziert wird. So wird mit `Nbr = 1234` die Variable erstellt und gleichzeitig auf den Wert 1234 gesetzt. Dies ist für kurze und schnelle Programme praktisch, kann jedoch in großen Programmen zu subtilen und schwer zu findenden Fehlern führen. Beispielsweise wurde in der dritten Zeile dieses Fragments die Variable `Nbr` fälschlicherweise als `Nbrs` geschrieben. Infolgedessen würde die Variable `Nbrs` mit dem Wert Null erstellt und der Wert von `Total` wäre falsch.

```
Nbr = 1234
Incr = 2
Total = Nbrs + Incr
```

Der Befehl OPTION EXPLICIT weist MMBasic an, Variablen nicht automatisch zu erstellen. Stattdessen müssen sie vor ihrer Verwendung explizit mit den Befehlen DIM, LOCAL oder STATIC (siehe unten) definiert werden. Die Verwendung dieses Befehls wird empfohlen, um eine gute Programmierpraxis zu unterstützen. Wenn er verwendet wird, sollte er am Anfang des Programms platziert werden, bevor Variablen verwendet werden.

DIM und LOCAL ( )

Die Befehle DIM und LOCAL können zum Definieren einer Variablen und zum Festlegen ihres Typs verwendet werden und sind obligatorisch, wenn der Befehl OPTION EXPLICIT verwendet wird.

Der Befehl DIM erstellt eine globale Variable, die im gesamten Programm, einschließlich innerhalb von Unterprogrammen und Funktionen, sichtbar ist und verwendet werden kann. Wenn die Definition jedoch nur innerhalb eines Unterprogramms oder einer Funktion sichtbar sein soll, sollten Sie den Befehl LOCAL am Anfang des Unterprogramms oder der Funktion verwenden. LOCAL hat genau die gleiche Syntax wie DIM.

Wenn LOCAL verwendet wird, um eine Variable mit dem gleichen Namen wie eine globale Variable zu spezifizieren, wird die globale Variable für die Subroutine oder Funktion ausgeblendet, und alle Verweise auf die Variable beziehen sich nur auf die durch den Befehl LOCAL definierte Variable. Alle mit LOCAL erstellten Variablen verschwinden, wenn das Programm die Subroutine verlässt.

Auf der einfachsten Ebene können DIM und LOCAL verwendet werden, um eine oder mehrere Variablen basierend auf ihrem Typ-Suffix oder der zu diesem Zeitpunkt gültigen OPTION DEFAULT zu definieren. Zum Beispiel:

```
DIM nbr%, s$
```

Sie können jedoch auch verwendet werden, um eine oder mehrere Variablen mit einem bestimmten Typ zu definieren, wenn das Typ-Suffix nicht verwendet wird:

```
DIM INTEGER nbr, nbr2, nbr3 usw.
```

In diesem Fall werden nbr, nbr2, nbr3 usw. alle als Ganzzahlen erstellt. Wenn Sie die Variable innerhalb eines Programms verwenden, müssen Sie das Typ-Suffix nicht angeben. Beispielsweise funktioniert `MyStr` im folgenden Beispiel perfekt als String-Variable:

```
DIM STRING MyStr
MyStr = "Hallo"
```

Die Befehle DIM und LOCAL akzeptieren auch die Microsoft-Praxis, den Typ der Variablen nach der Variablen mit dem Schlüsselwort „AS" anzugeben. Beispiel:

```
DIM nbr AS INTEGER, s AS STRING
```

In diesem Fall wird der Typ jeder Variablen einzeln festgelegt (nicht als Gruppe, wie wenn der Typ vor der Liste der Variablen steht).

Die Variablen können auch während ihrer Definition initialisiert werden. Beispiel:

```
DIM INTEGER a = 5, b = 4, c = 3
DIM s$ = "World", i% = &H8FF8F
DIM msg AS STRING = "Hello" + " " + s$
```

Der zur Initialisierung der Variablen verwendete Wert kann ein Ausdruck sein, der benutzerdefinierte Funktionen enthält.

Die Befehle DIM oder LOCAL werden ebenfalls zum Definieren eines Arrays verwendet, wobei alle oben aufgeführten Regeln für die Definition eines Arrays gelten. Sie können beispielsweise Folgendes verwenden:

```
DIM INTEGER nbr(10), nbr2, nbr3(5,8)
```

Bei der Initialisierung eines Arrays werden die Werte als durch Kommas getrennte Werte aufgelistet, wobei die gesamte Liste in Klammern steht. Beispiel:

```
DIM INTEGER nbr(5) = (11, 12, 13, 14, 15, 16)
```

oder

```
DIM days(7) AS STRING = ("","Sun","Mon","Tue","Wed","Thu","Fri","Sat")
```

STATIC

Innerhalb einer Subroutine oder Funktion ist es manchmal nützlich, eine Variable zu erstellen, die nur innerhalb der Subroutine oder Funktion sichtbar ist (wie eine LOCAL-Variable), aber ihren Wert zwischen den Aufrufen der Subroutine oder Funktion beibehält.

Dies können Sie mit dem Befehl STATIC erreichen. STATIC kann nur innerhalb einer Subroutine oder Funktion verwendet werden und hat dieselbe Syntax wie LOCAL und DIM. Der Unterschied besteht darin, dass ihr Wert zwischen den Aufrufen der Subroutine oder Funktion beibehalten wird (d. h. sie wird beim zweiten und den folgenden Aufrufen nicht initialisiert).

Wenn Sie beispielsweise die folgende Unterroutine hätten und sie wiederholt aufrufen würden, würde der erste Aufruf 5, der zweite 6, der dritte 7 usw. ausgeben.

```
SUB Foo
   STATIC var = 5
   PRINT var var =
   var + 1
END SUB
```

Beachten Sie, dass die Initialisierung der statischen Variablen auf 5 (wie im obigen Beispiel) nur beim ersten Aufruf der Subroutine wirksam wird. Bei nachfolgenden Aufrufen wird die Initialisierung ignoriert, da die Variable bereits beim ersten Aufruf erstellt wurde.

Wie bei DIM und LOCAL können die mit STATIC erstellten Variablen Float-Werte, Ganzzahlen oder Zeichenfolgen und Arrays davon mit oder ohne Initialisierung sein. Die Länge des mit STATIC erstellten Variablennamens und die Länge des Unterprogramm- oder Funktionsnamens dürfen zusammen 31 Zeichen nicht überschreiten.

CONST

Oft ist es nützlich, einen Bezeichner zu definieren, der einen Wert repräsentiert, ohne dass die Gefahr besteht, dass der Wert versehentlich geändert wird – was passieren kann, wenn Variablen für diesen Zweck verwendet werden (diese Vorgehensweise begünstigt eine weitere Klasse von schwer zu findenden Fehlern).

Mit dem Befehl CONST können Sie einen Bezeichner erstellen, der wie eine Variable funktioniert, aber auf einen Wert gesetzt ist, der nicht geändert werden kann. Beispiel:

```
CONST Eingangsspannungs-Pin = 31
CONST Maximalwert = 2,4
```

Die Bezeichner können dann in einem Programm verwendet werden, wo sie für den gelegentlichen Leser aussagekräftiger sind als einfache Zahlen. Zum Beispiel:

```
IF PIN(InputVoltagePin) > MaxValue THEN SoundAlarm
```

Es können mehrere Konstanten in einer Zeile erstellt werden:

```
CONST InputVoltagePin = 31, MaxValue = 2.4, MinValue = 1.5
```

Der Wert, der zur Initialisierung der Konstante verwendet wird, wird bei der Erstellung der Konstante ausgewertet und kann ein Ausdruck sein, der benutzerdefinierte Funktionen enthält.

Der Typ der Konstante wird aus dem ihr zugewiesenen Wert abgeleitet; so ist beispielsweise `MaxValue` oben eine Gleitkommakonstante, da 2,4 eine Gleitkommazahl ist. Der Typ einer Konstante kann auch explizit durch Verwendung eines Typ-Suffixes (d. h. !, % oder $) festgelegt werden, muss jedoch mit ihrem zugewiesenen Wert übereinstimmen.

Sonderzeichen in Zeichenfolgen von „ “

Spezielle, nicht druckbare Zeichen können mit dem Backslash (d. h. \) als Escape-Symbol in String-Konstanten eingefügt werden. Um diese Funktion zu aktivieren, muss der Befehl OPTION ESCAPE am Anfang des Programms platziert werden.

Dies sind die gültigen Escape-Sequenzen:

	Hexadezimalwert	Beschreibung
\a	07	Alarm (Piepton, Glocke)
\b	08	Rücktaste
\e	1B	Escape-Zeichen
\f	0C	Seitenvorschub
\n	0A	Zeilenumbruch (Zeilenvorschub)
\r	0D	Wagenrücklauf
\q	22	Anführungszeichen
\t	09	Horizontaler Tabulator
\v	0B	Vertikale Registerkarte
\\	5C	Backslash

\nnn	beliebig	Das Byte, dessen Wert durch nnn als Dezimalzahl angegeben wird
\&hh	beliebig	Das Byte, dessen Wert durch hh… angegeben wird, interpretiert als Hexadezimalzahl

Beispielsweise werden mit dem folgenden Befehl die Wörter „Hello“ und „World“ in separaten Zeilen ausgegeben:

```
OPTION ESCAPE
PRINT „Hello\r\nWorld”
```

Ausdrücke und Operatoren „ “

MMBasic wertet mathematische Ausdrücke nach den üblichen mathematischen Regeln aus. Beispielsweise werden Multiplikation und Division zuerst ausgeführt, gefolgt von Addition und Subtraktion. Diese Regeln werden als Vorrangregeln bezeichnet und sind im Folgenden näher erläutert.

Das bedeutet, dass 2 + 3 * 6 zu 20 führt, ebenso wie 5 * 4 und auch 10 + 4 * 3 – 2.

Wenn Sie den Interpreter zwingen möchten, Teile des Ausdrucks zuerst zu berechnen, können Sie diesen Teil des Ausdrucks in Klammern setzen. Beispielsweise ergibt (10 + 4) * (3 – 2) 14 und nicht 20, wie es ohne Klammern der Fall gewesen wäre. Die Verwendung von Klammern verlangsamt das Programm nicht nennenswert, daher sollten Sie sie großzügig einsetzen, wenn die Möglichkeit besteht, dass MMBasic Ihre Absicht falsch interpretiert.

Die folgenden Operatoren sind in MMBasic in der Reihenfolge ihrer Priorität implementiert. Operatoren, die auf derselben Ebene stehen (z. B. + und -), werden in der Reihenfolge ihrer Erscheinung in der Programmzeile von links nach rechts verarbeitet.

Arithmetische Operatoren:

^	Potenzierung (z. B. `b^n` bedeutet b^n)
* / \ MOD	Multiplikation, Division, ganzzahlige Division und Modulo (Rest)
+ -	Addition und Subtraktion

Verschiebungsoperatoren:

x << y x >> y	Diese Operatoren funktionieren auf besondere Weise. << bedeutet, dass der zurückgegebene Wert der Wert von x ist, der um y Bits nach links verschoben wurde, während >> dasselbe bedeutet, nur dass die Verschiebung nach rechts erfolgt. Es handelt sich um Integer-Funktionen, wobei alle verschobenen Bits verworfen und alle eingefügten Bits auf Null gesetzt werden.

Logische Operatoren:

NOT INV	invertiert den logischen Wert auf der rechten Seite (z. B. `NOT(a=b)` ist `a<>b`) oder bitweise Inversion des Werts auf der rechten Seite (z. B. `a = INV b`)
<> < > <= =< >= =>	Ungleichheit, kleiner als, größer als, kleiner oder gleich, kleiner oder gleich (alternative Version), größer als oder gleich, größer als oder gleich (alternative Version)
=	Gleichheit
UND XOR ODER	Konjunktion, Disjunktion, Exklusives Oder

Aus Gründen der Kompatibilität mit Microsoft sind die Operatoren AND, OR und XOR ganzzahlige Bitoperatoren. Beispielsweise gibt PRINT (3 AND 6) die Zahl 2 aus. Da diese Operatoren sowohl als logische Operatoren (z. B. IF a=5 AND b=8 THEN …) als auch als Bitweise-Operatoren (z. B. y% = x% AND &B1010) fungieren können, kommt es zu Verwirrung beim Interpreter, wenn sie in einem Ausdruck gemischt werden. Bewerten Sie logische und bitweise Ausdrücke daher immer in separaten Ausdrücken.

Die anderen logischen Operationen ergeben die Ganzzahl 0 (Null) für falsch und 1 für wahr. Beispielsweise gibt die Anweisung PRINT 4 >= 5 die Zahl Null in der Ausgabe aus, und der Ausdruck A = 3 > 2 speichert +1 in A.

Der NOT-Operator invertiert den logischen Wert auf seiner rechten Seite (es handelt sich nicht um eine bitweise Invertierung), während der INV-Operator eine bitweise Invertierung durchführt. Beide haben die höchste Priorität, sodass sie fest mit dem nächsten Wert verbunden sind. Bei normaler Verwendung von NOT oder INV sollte der zu verarbeitende Ausdruck in Klammern gesetzt werden. Beispiel:

```
IF NOT (A = 3 OR A = 8) THEN …
```

Zeichenfolgenoperatoren:

+	Zwei Zeichenfolgen verbinden
<> < > <= =< >= =>	Ungleichheit, kleiner als, größer als, kleiner oder gleich, kleiner oder gleich (alternative Version), größer als oder gleich, größer als oder gleich (alternative Version) in der Reihenfolge des ASCII-Werts.
=	Gleichheit

Bei Zeichenfolgenvergleichen wird die Groß-/Kleinschreibung berücksichtigt. Beispielsweise ist „A" größer als „a".

Mischen von Gleitkomma- und Ganzzahlen ( )

MMBasic übernimmt automatisch die Umwandlung von Zahlen zwischen Gleitkomma- und Ganzzahlen. Wenn eine Operation sowohl Gleitkomma- als auch Ganzzahlen mischt (z. B. `PRINT A% + B!`), wird die Ganzzahl zuerst in eine Gleitkommazahl umgewandelt, dann wird die Operation ausgeführt und eine Gleitkommazahl zurückgegeben. Wenn beide Seiten des Operators Ganzzahlen sind, wird eine Ganzzahloperation ausgeführt und eine Ganzzahl zurückgegeben.

Die einzige Ausnahme bildet die normale Division („/"), bei der immer beide Seiten des Ausdrucks in eine Gleitkommazahl umgewandelt werden und anschließend eine Gleitkommazahl zurückgegeben wird. Für die ganzzahlige Division sollten Sie den ganzzahligen Divisionsoperator „\" verwenden.

MMBasic-Funktionen geben je nach ihren Eigenschaften einen Float- oder Integer-Wert zurück. Beispielsweise gibt PIN() einen Integer-Wert zurück, wenn der Pin als digitaler Eingang konfiguriert ist, aber einen Float-Wert, wenn er als analoger Eingang konfiguriert ist.

Bei Bedarf können Sie einen Float-Wert mit der Funktion INT() in einen Integer-Wert umwandeln. Es ist nicht erforderlich, einen Integer-Wert speziell in einen Float-Wert umzuwandeln, aber wenn dies erforderlich wäre, könnte der Integer-Wert einer Float-Variablen zugewiesen werden, und er würde bei der Zuweisung automatisch umgewandelt werden.

64-Bit-Ganzzahlen ohne Vorzeichen ( )

MMBasic unterstützt 64-Bit-Ganzzahlen mit Vorzeichen. Das bedeutet, dass 63 Bits für die Zahl und ein Bit (das höchstwertige Bit) für das Vorzeichen (positiv oder negativ) verwendet werden. Es ist jedoch möglich, vollständige 64-Bit-Ganzzahlen ohne Vorzeichen zu verwenden, solange Sie keine arithmetischen Operationen mit den Zahlen durchführen.

64-Bit-Ganzzahlen ohne Vorzeichen können mit den Präfixen &H, &O oder &B vor einer Zahl erstellt werden, und diese Zahlen können in einer Ganzzahlvariablen gespeichert werden. Sie haben dann einen begrenzten Bereich von Operationen, die Sie mit diesen Zahlen durchführen können.
Diese sind << (nach links verschieben), >> (nach rechts verschieben), AND (bitweise UND), OR (bitweise ODER), XOR (bitweise exklusives ODER), INV (bitweise Inversion), = (gleich) und <> (ungleich). Arithmetische Operatoren wie Division oder Addition können durch eine 64-Bit-Zahl ohne Vorzeichen verwirrt werden und unsinnige Ergebnisse liefern.

Um 64-Bit-Zahlen ohne Vorzeichen anzuzeigen, sollten Sie die Funktionen HEX$(), OCT$() oder BIN$() verwenden.

Die folgende 64-Bit-Operation ohne Vorzeichen liefert beispielsweise die erwarteten Ergebnisse:

```
X% = &HFFFF0000FFFF0044 Y% =
&H800FFFFFFFFFFFF  X%  =  X%
AND Y%
PRINT HEX$(X%, 16)
```

Zeigt „800F0000FFFF0044" an

Unterprogramme und Funktionen „ “

Eine programmdefinierte Unterroutine oder Funktion ist einfach ein Block von Programmcode, der in einem Modul enthalten ist und von überall in Ihrem Programm aufgerufen werden kann. Es ist so, als hätten Sie der Sprache einen eigenen Befehl oder eine eigene Funktion hinzugefügt.

Unterprogramme

Eine Unterprogramm verhält sich wie ein Befehl und kann Argumente (manchmal auch als Parameterliste bezeichnet) haben. In der Definition des Unterprogramms sehen sie wie folgt aus:

```
SUB MYSUB arg1, arg2$, arg3
   <Anweisungen>
   <Anweisungen>
END SUB
```

Und wenn Sie die Subroutine aufrufen, können Sie den Argumenten Werte zuweisen. Zum Beispiel:

```
MYSUB 23, „Cat”, 55
```

Innerhalb der Unterroutine hat `arg1` den Wert `23`, `arg2$` den Wert „`Cat`“ usw. Die Argumente verhalten sich wie normale Variablen, existieren jedoch nur innerhalb der Unterroutine und verschwinden, wenn die Unterroutine beendet wird. Sie können Variablen mit demselben Namen im Hauptprogramm haben, diese werden jedoch durch die für die Unterroutine definierten Argumente verdeckt.

Beim Aufruf einer Unterroutine können Sie weniger als die erforderliche Anzahl von Werten angeben. In diesem Fall werden die fehlenden Werte entweder als Null oder als leere Zeichenfolge angenommen. Sie können auch einen Wert in der Mitte der Liste weglassen, dann geschieht dasselbe. Beispiel:

```
MYSUB 23, , 55
```

Dies führt dazu, dass `arg2$` auf die leere Zeichenfolge `""` gesetzt wird.

Anstelle des Typ-Suffixes (z. B. $ in arg2$) können Sie das Suffix AS <Typ> in der Definition des Unterprogrammarguments verwenden, wodurch das Argument als der angegebene Typ erkannt wird, auch wenn das Suffix nicht verwendet wird. Beispiel:

```
SUB MYSUB arg1, arg2 AS STRING, arg3 IF arg2
   = "Cat" THEN …
END SUB
```

Innerhalb einer Unterroutine können Sie eine Variable mit LOCAL definieren (die dieselbe Syntax wie DIM hat). Diese Variable existiert nur innerhalb der Unterroutine und verschwindet, wenn die Unterroutine beendet wird.

Funktionen

Funktionen ähneln Unterprogrammen, mit dem Hauptunterschied, dass eine Funktion verwendet wird, um einen Wert in einem Ausdruck zurückzugeben. Die Regeln für die Argumentliste in einer Funktion ähneln denen für Unterprogramme. Der einzige Unterschied besteht darin, dass beim Aufruf einer Funktion Klammern um die Argumentliste gesetzt werden müssen, auch wenn keine Argumente vorhanden sind (beim Aufruf eines Unterprogramms sind Klammern optional).

Um einen Wert aus der Funktion zurückzugeben, weisen Sie dem Namen der Funktion innerhalb der Funktion einen Wert zu. Wenn der Name der Funktion mit einem $, einem % oder einem ! endet, gibt die Funktion diesen Typ zurück, andernfalls gibt sie den Wert zurück, auf den OPTION DEFAULT gesetzt ist. Sie können den Typ der Funktion auch angeben, indem Sie AS <Typ> an das Ende der Funktionsdefinition anhängen.

Beispiel:

```
FUNCTION Fahrenheit(C) AS FLOAT Fahrenheit = C *
   1.8 + 32
END FUNCTION
```

Übergabe von Argumenten durch Referenz

Wenn Sie beim Aufruf einer Subroutine oder Funktion eine gewöhnliche Variable (d. h. keinen Ausdruck) als Wert verwenden, verweist das Argument innerhalb der Subroutine/Funktion zurück auf die im Aufruf verwendete Variable, und alle Änderungen am Argument werden auch an der übergebenen Variable vorgenommen. Dies wird als Übergabe von Argumenten per Referenz bezeichnet.

Sie könnten beispielsweise eine Unterroutine definieren, um zwei Werte wie folgt zu vertauschen:

```
SUB Swap a, b
   LOCAL t
    t = a
    a = b
    b = t
  END SUB
```

In Ihrem aufrufenden Programm würden Sie für beide Argumente Variablen verwenden:

```
Swap nbr1, nbr2
```

Das Ergebnis ist, dass die Werte von `nbr1` und `nbr2` vertauscht werden.

Damit dies funktioniert, müssen der Typ der übergebenen Variablen (z. B. `nbr1`) und das definierte Argument (z. B. `a`) identisch sein (im obigen Beispiel sind beide standardmäßig float).

Wenn Sie ein Argument als allgemeine Variable innerhalb einer Subroutine oder Funktion verwenden möchten (d. h. seinen Wert ändern möchten), sollten Sie seiner Definition das Schlüsselwort BYVAL voranstellen. Dadurch wird MMBasic angewiesen, immer den Wert des Arguments zu verwenden, auch wenn es sich um eine Variable handelt, und niemals auf die im Aufruf verwendete Variable zurückzugreifen. Der Grund dafür ist, dass ein anderer Benutzer Ihrer Routine möglicherweise unwissentlich eine Variable in seinem Aufruf verwendet und diese Variable durch Ihre Routine „auf magische Weise" geändert werden könnte, wenn Sie BYVAL nicht verwenden.

-Arrays übergeben

Einzelne Elemente eines Arrays können an eine Subroutine oder Funktion übergeben werden und werden wie eine normale Variable behandelt. Dies ist beispielsweise eine gültige Methode zum Aufrufen der Subroutine Swap (siehe oben):

```
Swap dat(i), dat(i + 1)
```

Diese Art von Konstrukt wird häufig beim Sortieren von Arrays verwendet.

Sie können auch ein oder mehrere vollständige Arrays an eine Subroutine oder Funktion übergeben, indem Sie das Array mit leeren Klammern anstelle der normalen Dimensionen angeben. Zum Beispiel `a()`. In der Subroutine- oder Funktionsdefinition muss der zugehörige Parameter ebenfalls mit leeren Klammern angegeben werden. Der Typ (d. h. Float, Integer oder String) des übergebenen Arguments und des Parameters in der Definition muss identisch sein.

In der Unterroutine oder Funktion übernimmt das Array die Dimensionen des übergebenen Arrays, die bei der Indizierung des Arrays berücksichtigt werden müssen. Bei Bedarf können die Dimensionen des Arrays als zusätzliche Argumente an die Unterroutine oder Funktion übergeben werden oder über die Funktion BOUND() ermittelt werden. Das Array wird per Referenz übergeben, was bedeutet, dass alle Änderungen, die innerhalb der Unterroutine oder Funktion am Array vorgenommen werden, auch für das übergebene Array gelten.

Wenn beispielsweise Folgendes ausgeführt wird, wird der Text „Hello World" ausgegeben:

```
DIM MyStr$(5, 5)
MyStr$(4, 4) = „Hello" : MyStr$(4, 5) = „World" Concat
MyStr$()
PRINT MyStr$(0, 0)

SUB Concat arg$()
   arg$(0,0) = arg$(4, 4) + " " + arg$(4, 5) END SUB
```

Vorzeitiges Beenden von „ "

Es kann nur ein END SUB oder END FUNCTION für jede Definition einer Unterroutine oder Funktion geben. Um eine Unterroutine vorzeitig zu verlassen (d. h. bevor der Befehl END SUB erreicht wurde), können Sie den Befehl EXIT SUB verwenden. Dies hat denselben Effekt, als hätte das Programm die Anweisung END SUB erreicht. Ebenso können Sie EXIT FUNCTION verwenden, um eine Funktion vorzeitig zu verlassen.

Rekursion

Rekursion liegt vor, wenn eine Subroutine oder Funktion sich selbst aufruft. In MMBasic ist Rekursion möglich, jedoch gibt es dabei eine Reihe von Problemen (die eine direkte Folge der Einschränkungen von Mikrocontrollern und der Sprache BASIC sind):

- Die Tiefe der Rekursion ist begrenzt. In der PicoMite-Firmware sind dies 50 Ebenen.
- Wenn Sie viele Argumente für die Unterroutine oder Funktion und viele LOCAL-Variablen (insbesondere Strings) haben, könnte Ihnen leicht der Speicher ausgehen, bevor Sie die Grenze von 50 Ebenen erreichen.
- Alle FOR…NEXT-Schleifen und DO…LOOPs werden beschädigt, wenn die Unterroutine oder Funktion rekursiv aus diesen Schleifen heraus aufgerufen wird.

Beispiele

Oftmals besteht die Notwendigkeit, einen speziellen Befehl oder eine spezielle Funktion in MMBasic zu implementieren, aber in vielen Fällen können diese mit einer gewöhnlichen Subroutine oder Funktion erstellt werden, die dann genau wie ein integrierter Befehl oder eine integrierte Funktion funktioniert.

Beispielsweise wird manchmal eine TRIM-Funktion benötigt, die bestimmte Zeichen am Anfang und Ende einer Zeichenfolge entfernt. Im Folgenden finden Sie ein Beispiel dafür, wie Sie eine solche einfache Funktion in MMBasic erstellen können.

Das erste Argument der Funktion ist die zu trimmende Zeichenfolge, das zweite ist eine Zeichenfolge, die die aus der ersten Zeichenfolge zu trimmenden Zeichen enthält. RTrim$() trimmt die angegebenen Zeichen vom Ende der Zeichenfolge, LTrim$() vom Anfang und Trim$() von beiden Enden.

```
' Alle Zeichen in c$ vom Anfang und Ende von s$ entfernen Funktion
Trim$(s$, c$)
   Trim$ = RTrim$(LTrim$(s$, c$), c$)
End Function

' alle Zeichen in c$ vom Ende von s$ abschneiden Funktion
RTrim$(s$, c$)
   RTrim$ = s$
   Do While Instr(c$, Right$(RTrim$, 1)) RTrim$ =
      Mid$(RTrim$, 1, Len(RTrim$) - 1)
   Schleife
Ende Funktion

' Alle Zeichen in c$ vom Anfang von s$ abschneiden
Funktion LTrim$(s$, c$)
   LTrim$ = s$
   Do While Instr(c$, Left$(LTrim$, 1)) LTrim$ =
      Mid$(LTrim$, 2)
   Schleife
End Function
```

Beispiel für die Verwendung dieser Funktionen:

```
S$ = "     ****23.56700 "
PRINT Trim$(s$, " ")
```

Ergibt „****23.56700”

```
PRINT Trim$(s$, " *0")
```

Ergibt „23,567”

```
PRINT LTrim$(s$, " *0")
```

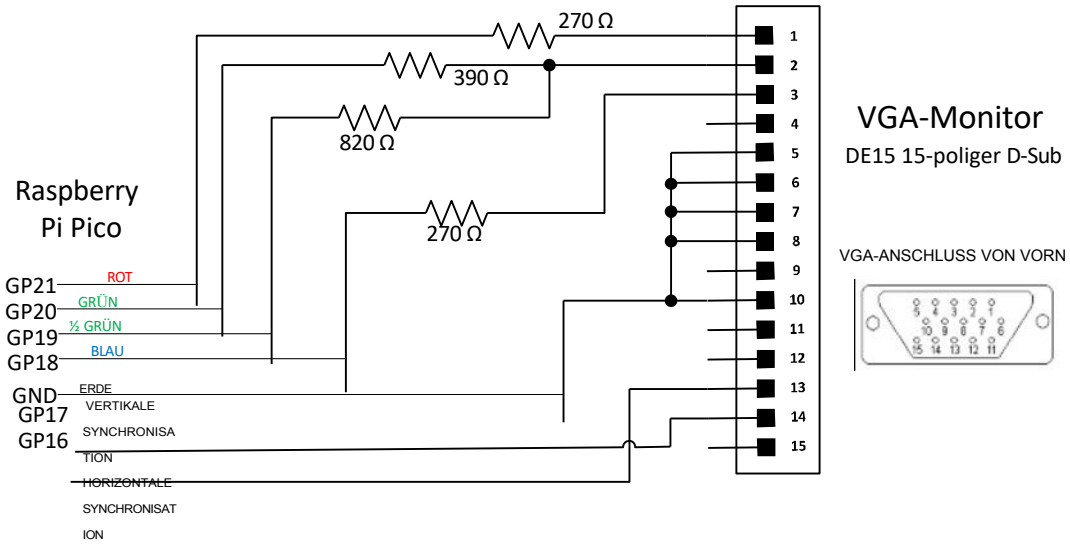
Ergibt „23,56700”

Video- -Ausgabe

VGA- -Video

Bei Firmware-Versionen, die VGA unterstützen, wird die Videoausgabe beim Start automatisch aktiviert – es müssen keine Optionen eingestellt werden.

Das folgende Diagramm veranschaulicht, wie der VGA-Monitor angeschlossen wird.



Die Ausgabe erfolgt im Standard-VGA-Format mit einer Pixelfrequenz von 25,175 MHz und einer Bildfrequenz von 60 Hz. Mit dem Befehl MODE können zwei oder drei Modi ausgewählt werden:

MODE 1 Monochrom mit einer Auflösung von 640 x 480 (Standard bei Start) MODE 2 16 Farben mit einer Auflösung von 320 x 240

MODE 3 16 Farben mit einer Auflösung von 640 x 480 (nur RP2350)

In MODUS 2 und 3 beträgt die Ausgabe 16 Farben im 4-Bit-RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). In MODUS 2 werden die Pixel entlang der x- und y-Achse dupliziert, was eine Auflösung von 320 x 240 ergibt, während der Monitor weiterhin ein Signal von 640 x 480 sieht.

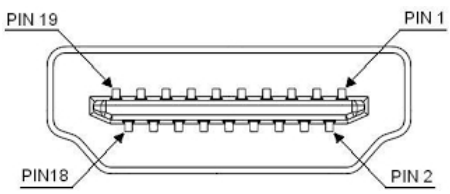
Die Ausgabe von MMBasic wird als Bitmap in einen Framebuffer geschrieben. Die Firmware nutzt dann die zweite CPU im Prozessor, um diese Framebuffer-Daten pixelweise über DMA an einen der programmierbaren I/O-Controller (PIO0) des RP2040 zu übertragen und so die Anzeige zu generieren. Da dies unabhängig vom Hauptprozessor abläuft, hat die Erzeugung der VGA-Ausgabe kaum oder gar keinen Einfluss auf die Geschwindigkeit von MMBasic.

HDMI- -Video

Für Versionen der Firmware, die HDMI-Video unterstützen, listet die folgende Tabelle die Anschlüsse an die Standard-HDMI-Buchse vom Typ A auf. Der HDMI-Ausgang wird beim Start automatisch aktiviert – es müssen keine Optionen eingestellt werden.

HDMI-Pin 1:	Pin 21 (GP16) über einen 220-Ω-Widerstand	HDMI-Pin 13:	Keine Verbindung
HDMI-Pin 2:	Masse	HDMI-Pin 14:	Keine Verbindung
HDMI-Pin 3:	Pin 22 (GP17) über einen 220-Ω-Widerstand	HDMI-Pin 15:	Keine Verbindung
HDMI-Pin 4:	Pin 24 (GP18) über einen 220-Ω-Widerstand	HDMI-Pin 16:	Keine Verbindung
HDMI-Pin 5:	Masse	HDMI-Pin 17:	Masse
HDMI-Pin 6:	Pin 25 (GP19 über einen 220-Ω-Widerstand	HDMI-Pin 18:	+5 V über Schottky-Barrier-Diode
HDMI-Pin 7:	Pin 16 (GP12) über einen 220-Ω-Widerstand	HDMI-Pin 19:	Keine Verbindung
HDMI-Pin 8:	Masse		
HDMI-Pin 9:	Pin 17 (GP13) über einen 220-Ω-Widerstand		
HDMI-Pin 10:	Pin 19 (GP14) über einen 220-Ω-Widerstand		
HDMI-Pin 11:	Masse		
HDMI-Pin 12:	Pin 20 (GP15) über einen 220-Ω-Widerstand		

HDMI Vorderseite Ansicht



Die HDMI-Signal-Pins werden mit einer hohen Frequenz angesteuert, daher sollten folgende Punkte beachtet werden:

- Halten Sie die Signalleitungen so kurz wie möglich.
- Stellen Sie sicher, dass alle Signalbahnen gleich lang sind.
- Die 220-Ω-Widerstände sollten vorzugsweise oberflächenmontiert sein.

Um das DVI/HDMI-Signal zu erzeugen, muss die Firmware den RP2350 auf bis zu 372 MHz übertakten, was für die meisten Raspberry Pi Pico 2-Module bei diesen Geschwindigkeiten kein Problem darstellt. Dies kann jedoch nicht garantiert werden, insbesondere bei Modulen von Drittanbietern. Ein Beispiel hierfür ist das Pimoroni Pico Plus 2, das bei den erforderlichen Geschwindigkeiten an seine Grenzen stößt und daher nicht für die HDMI-Versionen der PicoMite-Firmware empfohlen werden kann.

Ähnlich wie bei der Erzeugung von VGA wird die Ausgabe von MMBasic in einen Framebuffer geschrieben, der unter Verwendung der zweiten CPU und DMA an das HSTX-Peripheriegerät weitergeleitet wird, das wiederum die parallelen Videodaten erzeugt. Das erzeugte Videosignal ist eigentlich DVI (HDMI unterstützt DVI), was bedeutet, dass Audio am HDMI-Ausgang nicht unterstützt wird und auch anspruchsvolle HDMI-Funktionen wie High Definition Content Protection (HDCP) und Ethernet nicht unterstützt werden.

HDMI-Video unterstützt eine Reihe von Auflösungen. Um diese einzustellen, verwenden Sie den folgenden Befehl:

```
OPTION RESOLUTION nn
```

Wobei „nn" einer der folgenden Werte ist:

640x480 oder 640
1280x720 oder 1280
1024x768 oder 1024
800 x 600 oder 800
720 x 400 oder 720
848x480 oder 848

Jede HDMI-Auflösung kann in einer Reihe von Modi betrieben werden, die mit dem Befehl MODE eingestellt werden. Beachten Sie, dass viele Modi die angezeigte Auflösung reduzieren, um Speicherplatz für andere Funktionen zu sparen. Diese Reduzierung erfolgt durch Verdopplung oder Vervierfachung jedes Pixels, jedoch erkennt der Monitor immer die mit dem Befehl OPTION RESOLUTION eingestellte Auflösung (d. h. Pixeldichte). Die Standardeinstellung ist RESOLUTION 640x480 und MODE 1

VGA/PS2-Referenzdesign (Raspberry Pi- -Pico)

Dies ist ein einfach zu montierendes Design, das den VGA-Ausgang, die PS2-Tastaturschnittstelle und den SD-Kartensteckplatz implementiert (dieses Design wurde im Magazin *„Silicon Chip"* vorgestellt).

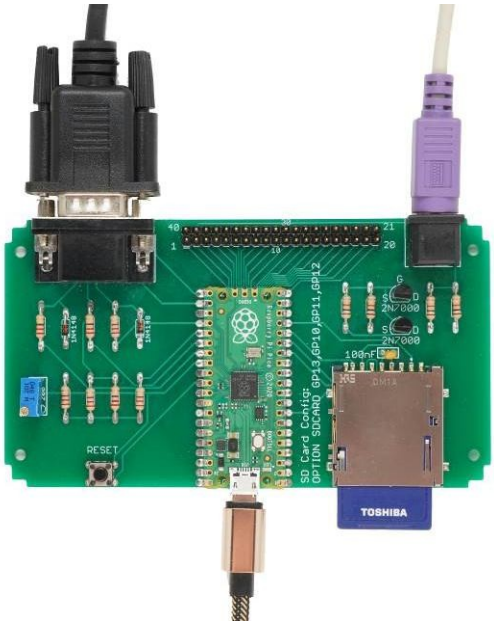
Es verwendet gängige Durchsteckkomponenten und kann in weniger als einer Stunde zusammengebaut werden.

Alle 40 Pins des Raspberry Pi Pico sind in derselben Konfiguration wie beim Pico an einen 40-poligen Stecker auf der Rückseite der Leiterplatte angeschlossen. Dies erleichtert den Anschluss externer Geräte, da Sie das Pinbelegungsdiagramm in diesem Handbuch zu Rate ziehen und dann die entsprechenden Pins am 40-poligen Stecker auswählen können.

Unter Berücksichtigung der für den VGA-Ausgang, die Tastatur und die SD-Karte reservierten I/O-Pins stehen 14 I/O-Pins für externe Schaltungen zur Verfügung.

Die Platine ist so dimensioniert, dass sie in ein Altronics-Steckgehäuse mit den Maßen 130 x 75 x 28 mm (Teilenummer H0376) passt.

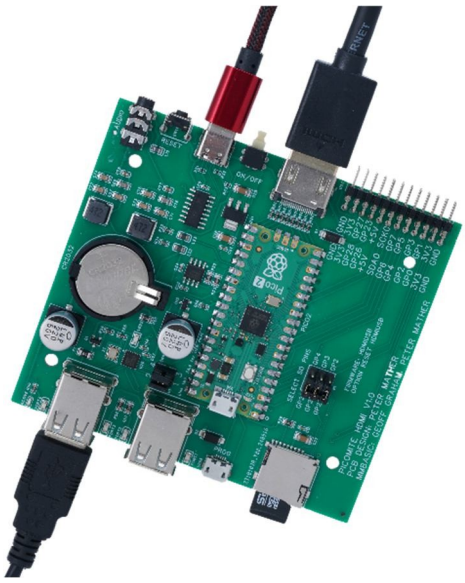
Das Konstruktionspaket für dieses Design kann unter https://geoffg.net/picomitevga.html (am Ende der Seite) heruntergeladen werden.



HDMI/USB-Referenzdesign (Raspberry Pi Pico- l 2)

Dies ist ein voll ausgestattetes Design auf Basis des Raspberry Pi Pico 2 (mit RP2350-Prozessoren), das Folgendes umfasst:

- HDMI-Videoausgang
- Vier USB-Schnittstellen für Tastatur, Maus, Gamepad usw.
- Hochwertiger Audioausgang für verstärkte Lautsprecher.
- USB-Schnittstelle zur seriellen Konsole.
- Batteriegepufferte Echtzeituhr.
- Micro-SD-Kartensteckplatz.
- 14 I/O-Pins auf der Rückseite verfügbar.
- Passend für ein Multicomp MCRM2015S- oder Hammond RM2015S-Gehäuse.



Das Konstruktionspaket für diesen Entwurf kann unter folgender Adresse heruntergeladen werden: https://geoffg.net/picomitevga.html (am Ende der Seite).

Tastatur/Maus/Gamepad

Die PicoMite-Firmware unterstützt Tastaturen und Mäuse, die entweder über eine PS2- oder eine USB-Schnittstelle angeschlossen werden. Die Wahl zwischen PS2 und USB hängt von der Version der geladenen Firmware ab. Siehe dazu das Kapitel *„Firmware-Versionen und Dateien"* am Anfang dieses Handbuchs.

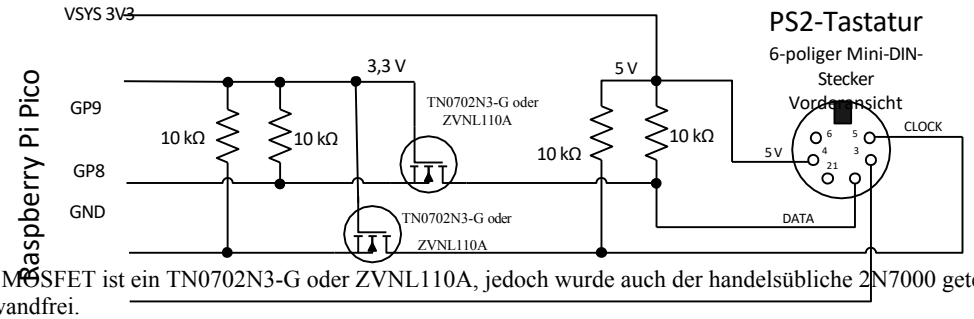
USB-Versionen der Firmware unterstützen auch ein PS3- oder PS4-Gamepad mit USB-Schnittstelle. Bei Versionen ohne USB-Unterstützung können die Befehle WII (CLASSIC) und WII NUNCHUCK verwendet werden, um ein über I^2C angeschlossenes Gamepad anzugeben.

Eine Tastatur kann zur Eingabe von Daten in das BASIC-Programm verwendet werden oder, mit einem VGA/HDMI-Videoausgang, zur Erstellung eines eigenständigen Computers mit Tastatur und Display. Anstelle eines Videoausgangs können Sie bei den Versionen, die dies unterstützen, auch ein LCD-Panel anschließen und die Ausgabe der MMBasic-Konsole auf dem LCD-Panel anzeigen, wodurch eine kompaktere Version eines eigenständigen Computers entsteht. Weitere Informationen hierzu finden Sie im Abschnitt *„LCD-Display als Konsolenausgabe*".

PS2-Tastatur auf dem Raspberry Pi Pico (RP2040)

Die Tastatur-Takt- und Datensignale der PS2 arbeiten mit 5 V, aber die I/O-Pins der RP2040-Prozessoren dürfen nicht mehr als 3,6 V ausgesetzt werden. Aus diesem Grund sollte ein Pegelumsetzer verwendet werden, damit der Raspberry Pi Pico Signalspannungen im Bereich von 0 bis 3,3 V sieht, während die Tastatur Spannungen von 0 bis 5 V sieht.

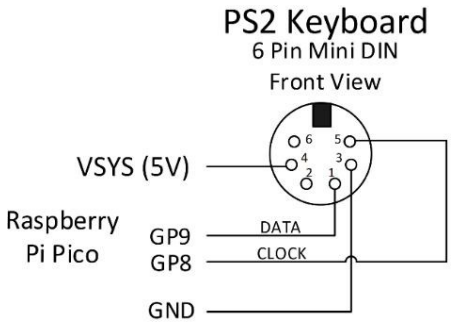
Dies kann auf viele Arten erreicht werden, aber die folgende Schaltung ist eine einfache und kostengünstige Lösung:



Der empfohlene MOSFET ist ein TN0702N3-G oder ZVNL110A, jedoch wurde auch der handelsübliche 2N7000 getestet und funktioniert einwandfrei.

Nach dem Anschließen muss die Tastatur mit dem Befehl OPTION KEYBOARD aktiviert werden.

PS2-Tastatur auf dem Raspberry Pi Pico 2 (RP2350)



Die I/O-Pins der Mikrocontroller der RP2350-Serie können 5 V (bei eingeschaltetem Gerät) aushalten, sodass die Tastatur direkt angeschlossen werden kann, wie in der Abbildung gezeigt, wobei die 5 V vom VSYS-Pin des Raspberry Pi Pico 2 geliefert werden.

Die Tastatur wird mit dem Befehl OPTION KEYBOARD aktiviert.

PS2- -Maus

Die für eine PS2-Maus verwendeten I/O-Pins müssen mit den Befehlen OPTION MOUSE (an der Eingabeaufforderung) oder MOUSE OPEN (innerhalb eines Programms) konfiguriert werden.

Eine PS2-Maus wird mit 5 V versorgt, sodass auf einem Raspberry Pi Pico (RP2040) ein Pegelumsetzer für die Takt- und Datenpins der Maus erforderlich ist – dieser kann mit der oben genannten Schaltung für eine PS2-Tastatur identisch sein. Auf einem Raspberry Pi Pico 2 (RP2350) ist kein Pegelumsetzer erforderlich, sodass die Maus direkt angeschlossen werden kann.

USB- -Schnittstelle

Versionen der Firmware (sowohl RP2040 als auch RP2350) mit USB-Unterstützung ermöglichen den Anschluss einer Tastatur, einer Maus und/oder eines Gamepads über die USB-Schnittstelle. Dazu wird der USB-Anschluss des Raspberry Pi Pico in einen USB-Host umgewandelt (im Gegensatz zum normalen Modus als USB-Client). Dies ist möglich, da der Anschluss und die Elektronik des Pico USB-OTG-kompatibel (On The Go) sind, ähnlich wie der Anschluss vieler Mobiltelefone.

Da der USB-Anschluss für andere Aufgaben verwendet wird, muss der Pico über 5 V am VSYS-Pin oder am VbUS-Pin mit Strom versorgt werden, wenn Sie einen externen Hub/eine externe Tastatur über den Pico mit Strom versorgen möchten.

Um ein USB-Gerät anzuschließen, benötigen Sie ein Konverterkabel mit einem Micro-USB-Stecker an einem Ende (für den Pico) und einer USB-Buchse vom Typ A am anderen Ende (für das Gerät). Ein typisches Beispiel ist das Jaycar Cat Nbr WC7725.

Da die USB-Schnittstelle des Pico zu einem USB-Host umgewandelt wurde, haben Sie keinen Zugriff auf die serielle MMBasic-Konsole. Die Firmware gleicht dies aus, indem sie automatisch Pin 11 (GP8) für die serielle Konsole Tx und Pin 12 (GP9) für Rx verwendet und die Baudrate auf 115200 Baud einstellt. Um auf diese Konsole zuzugreifen, benötigen Sie eine USB-zu-Seriell-Brücke, die auf der einen Seite eine serielle TTL-Schnittstelle und auf der anderen Seite eine USB-Schnittstelle bietet (suchen Sie nach Modulen mit dem CP2102- oder CH340-Chip). Bei Bedarf kann OPTION SERIAL CONSOLE verwendet werden, um die für die Konsole verwendeten Pins zu ändern.

USB- -Hub

Die Firmware unterstützt über diese Schnittstelle auch einen USB-Hub, sodass mehrere Tastaturen oder eine Tastatur plus eine Maus plus ein Gamepad usw. angeschlossen werden können. Über einen Hub können maximal 4 Geräte angeschlossen werden. Diese werden durch eine Kanalindexnummer (1 bis 4) referenziert. Verwenden Sie MM.INFO(USB n), um den Gerätecode für jedes an Kanal n angeschlossene Gerät zurückzugeben. Standardmäßig wird eine Tastatur dem Kanal 1 zugewiesen. Eine Maus wird dem Kanal 2 zugewiesen. Das erste Gamepad wird dem Kanal 3 und ein zweites Gamepad dem Kanal 4 zugewiesen.

Wenn Sie einen USB-Hub verwenden, ist es besser, einen Hub ohne eigene Stromversorgung zu verwenden (d. h. einen Hub, der vom Raspberry Pi Pico mit Strom versorgt wird). Der Grund dafür ist, dass der USB-Protokollstack den Hub nicht zurücksetzen kann und es zu Verwirrung kommen kann, wenn der Pico aus- und wieder eingeschaltet wird, ohne dass dies auch für den Hub geschieht. Der Hub kann auch verwirrt werden, wenn Geräte ausgetauscht werden, während der Hub mit Strom versorgt wird. In diesem Fall sollten Sie den Pico und anschließend den Hub aus- und wieder einschalten und dann die USB-Geräte nacheinander anschließen.

Beachten Sie, dass ein Hub nicht erforderlich ist. Wenn Sie nur ein Gerät (z. B. eine Tastatur) anschließen möchten, können Sie das Gerät (mit einem Adapterkabel) einfach direkt an den USB-Anschluss des Pico anschließen.

USB- -Tastatur

Wenn eine USB-Tastatur angeschlossen wird, wird sie sofort erkannt (keine Konfiguration erforderlich) und MMBasic ordnet sie standardmäßig Kanal 1 zu – es sind keine weiteren Schritte erforderlich.

USB- -Maus

Wenn eine USB-Maus angeschlossen wird, wird sie sofort erkannt (keine Konfiguration erforderlich) und MMBasic ordnet sie standardmäßig Kanal 2 zu – es sind keine weiteren Schritte erforderlich.

USB- -Gamepad

Ein oder mehrere PS3- oder PS4-Controller oder Gamepads wie beispielsweise ein Super Nintendo SNES-Controller mit USB-Schnittstelle können über USB angeschlossen werden (siehe Abbildung rechts).

Standardmäßig wird das erste Gamepad dem Kanal 3 und ein zweites Gamepad dem Kanal 4 zugewiesen. Innerhalb eines Programms können die Daten vom Gamepad mit der Funktion DEVICE(GAMEPAD) gelesen werden.

Konfigurieren der Tastatur „ “

Standardmäßig wird die Tastaturkonfiguration als Standard-US-Layout angenommen. Mit dem Befehl OPTION KEYBOARD können jedoch Layouts für andere Länder konfiguriert werden.

Die Syntax des Befehls lautet:

```
OPTION KEYBOARD Sprache
```

Wobei „Sprache“ ein zweistelliger Code ist, wie beispielsweise US für die in den USA, Australien und Neuseeland verwendete Standardtastatur. Andere Tastaturlayouts sind Großbritannien (UK), Frankreich (FR), Deutschland (GR), Belgien (BE), Italien (IT), Brasilien (BR) oder Spanien (ES).

Beachten Sie, dass die Nicht-US-Layouts einige der auf diesen Tastaturen vorhandenen Sondertasten abbilden, die entsprechenden Sonderzeichen jedoch nicht angezeigt werden, da sie nicht Teil der Standard-PicoMite-Schriftarten sind. Stattdessen wird ein Standard-ASCII-Zeichen verwendet.

Verwendung einer Maus mit

Die Maus ist besonders nützlich im MMBasic-Programmeditor, wo sie viele der Funktionen von GUI-Editoren wie Notepad in Windows nachbilden kann (siehe Abschnitt *„Vollbild-Editor"* weiter oben in diesem Handbuch). Dazu gehören das Positionieren der Einfügemarke sowie das Kopieren und Einfügen über die Zwischenablage.

Eine Maus kann auch in einem Programm verwendet werden, in dem ihre Position mit der Funktion DEVICE() abgefragt werden kann. Das folgende Programm meldet beispielsweise jede Mausbewegung.

Beachten Sie, dass die Maus immer dem Kanal 2 zugewiesen ist

```
„Endlosschleife, um jede Bewegung auf der Konsole auszugeben Do
  mx=DEVICE(MOUSE 2, x)
  my=DEVICE(MOUSE 2, y)
  If mx <> tx Or my <> ty Then Print mx, my tx = mx
  : ty = my
Schleife
```

Programm- und Daten sspeicher

Das BASIC-Programm wird im Flash-Speicher gehalten und von dort aus ausgeführt. Wenn ein Programm über EDIT bearbeitet oder über die Konsole geladen wird, wird es dort gespeichert. Der Flash-Speicher ist nichtflüchtig, sodass das Programm bei einem Stromausfall oder einem Reset des Prozessors nicht verloren geht.

Zusätzlich zu diesem Programmspeicher gibt es drei weitere Speicherorte, an denen Programme gespeichert werden können. Diese werden im Folgenden ausführlich beschrieben und sind Flash-Slots, das Flash-Dateisystem und eine angeschlossene SD-Karte.

Flash- -Steckplätze

Es gibt drei davon, die zum Speichern völlig unterschiedlicher Programme oder früherer Versionen des Programms, an dem Sie gerade arbeiten, verwendet werden können (für den Fall, dass Sie zu einer früheren Version zurückkehren müssen). Darüber hinaus ermöglicht MMBasic einem BASIC-Programm, ein anderes Programm zu laden und auszuführen, das an einem nummerierten Flash-Speicherort gespeichert ist, wobei alle Variablen und Einstellungen des ursprünglichen Programms beibehalten werden – dies wird als Verkettung bezeichnet und ermöglicht die Ausführung eines viel größeren Programms, als es die Programmspeicherkapazität normalerweise zulassen würde.

Um diese nummerierten Positionen im Flash-Speicher zu verwalten, können Sie die folgenden Befehle verwenden (beachten Sie, dass *n* eine Zahl zwischen 1 und 3 ist):

FLASH SAVE *n*	Speichert das Programm im Programmspeicher an der Flash-Speicherstelle *n*.
FLASH LOAD *n*	Laden Sie ein Programm aus dem Flash-Speicherplatz *n* in den Programmspeicher.
FLASH RUN *n*	Führt ein Programm aus dem Flash-Speicherplatz *n* aus, löscht alle Variablen, löscht oder ändert jedoch nicht das im Hauptprogrammspeicher gespeicherte Programm.
FLASH-LISTE	Zeigt eine Liste aller Flash-Speicherorte einschließlich der ersten Zeile des Programms an.
FLASH LIST n [,ALL]	Listet das Programm an Speicherort n auf. Verwenden Sie FLASH LIST n,ALL, um ohne Seitenumbrüche
FLASH ERASE *n*	Löscht den Flash-Speicherplatz *n*. FLASH ERASE ALL Löscht alle Flash-Speicherplätze.
FLASH CHAIN *n*	Führt das Programm im Flash-Speicherplatz *n* aus und lässt alle Variablen unverändert. Dadurch können sehr große Programme, die in zwei oder drei Teile aufgeteilt werden können, ausgeführt werden. Das im Hauptprogrammspeicher gespeicherte Programm wird dabei nicht gelöscht oder verändert.
FLASH ÜBERSCHREIBEN *n*	Löscht den Flash-Speicherplatz *n* und speichert dann das Programm im Programmspeicher an diesem Speicherplatz.
FLASH DISK LOAD f$ [,O]	Lädt einen Flash-Speicherplatz mit einer Binärdatei, die mit LIBRARY DISK SAVE erstellt wurde. Überschreibt den Speicherplatz, wenn das optionale „O“ angegeben ist.

Darüber hinaus kann mit dem Befehl OPTION AUTORUN ein Flash-Programmspeicherplatz angegeben werden, der beim Einschalten oder Neustart der CPU ausgeführt werden soll. Diese Option kann auch ohne Angabe eines Flash-Speicherplatzes verwendet werden. In diesem Fall führt MMBasic das Programm im Programmspeicher automatisch aus.

Hinweise:

- Es wird empfohlen, in der ersten Zeile des Programms einen Kommentar einzufügen, der das Programm beschreibt. Dieser wird dann vom Befehl FLASH LIST angezeigt und hilft bei der Identifizierung des Programms.
- Alle im Flash-Speicher gespeicherten BASIC-Programme können gelöscht werden, wenn Sie die PicoMite-Firmware aktualisieren (oder downgraden). Stellen Sie daher sicher, dass Sie diese zuvor sichern.
- Der Befehl LIBRARY verwendet Slot 3 zum Speichern von Bibliotheksdaten, sodass bei Verwendung der Bibliotheksfunktion nur 2 Slots verfügbar sind.

Flash- -Dateisystem

Dies ist ein Bereich des Flash-Speichers des Raspberry Pi Pico, der automatisch von der Firmware erstellt wird und für MMBasic wie ein normales Laufwerk aussieht. Er wird als Laufwerk A: bezeichnet, und Daten und Programme können mit den normalen BASIC-Dateibefehlen (SAVE, RUN, OPEN usw.) gelesen/geschrieben werden. Darüber hinaus können Unterverzeichnisse erstellt und gelöscht sowie lange Dateinamen verwendet werden.

Beispiel für die Ausführung eines Programms:

```
RUN "A:/MyProgram.bas"
```

Öffnen einer Textdatei für den wahlfreien Zugriff:

```
OPEN "A:/data/database.dat" FOR RANDOM as #4
```

Dieses Laufwerk wird automatisch erstellt, wenn die PicoMite-Firmware geladen wird, sodass es für das BASIC-Programm immer verfügbar ist. Es kann zum Speichern von Programmen, Bildern, Musik, Konfigurationsdaten, Protokolldateien und vielem mehr verwendet werden. Seine Größe variiert je nach Größe des Flash-Speichers – auf einem Raspberry Pi Pico mit 2 MB Flash-Speicher beträgt sie 200 bis 500 KB, auf einem Raspberry Pi Pico 2 mit 4 MB Flash-Speicher etwas mehr als 2 MB und auf einem Modul mit 16 MB beträgt die Größe des Flash-Dateisystems bis zu 14 MB.

Das System erstellt und verwaltet die Datei „BOOTCOUNT" im Flash-Dateisystem. Diese Datei zählt die Anzahl der Neustarts des Geräts und kann mit der Funktion MM.INFO(boot count) ausgelesen werden.

SD- -Karten

Ein SD-Kartensteckplatz kann an den Raspberry Pi Pico angeschlossen und als Laufwerk B: aufgerufen werden. Wie beim Flash-Dateisystem können die normalen BASIC-Dateibefehle zum Speichern/Laden von Programmen sowie zum Öffnen von Datendateien zum Lesen/Schreiben verwendet werden.

Es werden Karten mit bis zu 32 GB unterstützt, die mit FAT16 oder FAT32 formatiert sind. Die erstellten Dateien können auch auf PCs mit Windows-, Linux- oder Mac-Betriebssystem gelesen/geschrieben werden. Die PicoMite-Firmware verwendet das SPI-Protokoll für die Kommunikation mit der Karte. Dies wird nicht vom Kartentyp beeinflusst, sodass alle Typen (Klasse 4, 10, UHS-1 usw.) unterstützt werden. Eine Beschreibung von SPI finden Sie unter: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Das SPI-Protokoll muss vor der Verwendung speziell konfiguriert werden. Dies kann auf zwei Arten erfolgen: über den „System"-SPI-Port oder durch direkte Angabe der zu verwendenden E/A-Pins:

System-SPI-Port

Dieser Port wird für Systemzwecke verwendet (SD-Karte, LCD-Display und Touch-Controller auf einem LCD-Panel). Es gibt eine Reihe von Ports und Pins, die verwendet werden können (siehe Kapitel *„PicoMite-Hardware*"), aber in diesem Beispiel wird SPI auf den Pins GP18, GP19 und GP16 für Clock, MOSI bzw. MISO verwendet.

```
OPTION SYSTEM SPI GP18, GP19, GP16
```

Anschließend muss MMBasic mitgeteilt werden, dass eine SD-Karte angeschlossen ist und welcher Pin für das Chip-Select-Signal (CS) verwendet wird:

```
OPTION SDCARD GP22
```

Spezielle E/A-Pins

Alternativ kann die SD-Karte, wenn keine anderen Geräte den SPI-Bus mit ihr teilen, wie folgt eingerichtet werden:

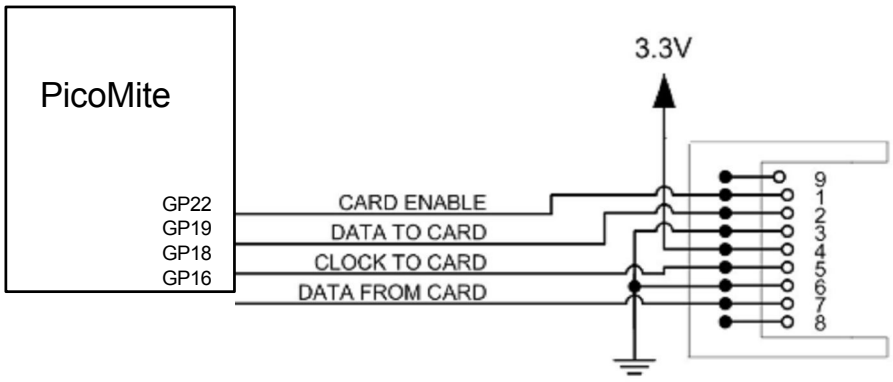
```
OPTION SDCARD CS_pin, CLK_pin, MOSI_pin, MISO_pin
```

In diesem Fall können die Pins völlig flexibel zugewiesen werden und müssen nicht für den SPI-Betrieb geeignet sein, aber die Leistung der SD-Karte ist besser, wenn gültige SPI-Pins ausgewählt werden.

Diese Befehle müssen an der Eingabeaufforderung (nicht in einem Programm) eingegeben werden und führen zu einem Neustart der PicoMite-Firmware. Dies hat den Nebeneffekt, dass die USB-Konsolenschnittstelle getrennt wird und erneut verbunden werden muss.

Wenn der Raspberry Pi Pico neu gestartet wird, initialisiert MMBasic automatisch die SD-Kartenschnittstelle. Dieser SPI-Port steht dann für BASIC-Programme nicht zur Verfügung (d. h. er ist reserviert). Um die Konfiguration zu überprüfen, können Sie den Befehl OPTION LIST verwenden, um alle eingestellten Optionen einschließlich der Konfiguration der SD-Karte aufzulisten.

Der grundlegende Schaltplan für den Anschluss des SD-Kartensteckers unter Verwendung dieser Pin-Belegungen ist unten dargestellt.

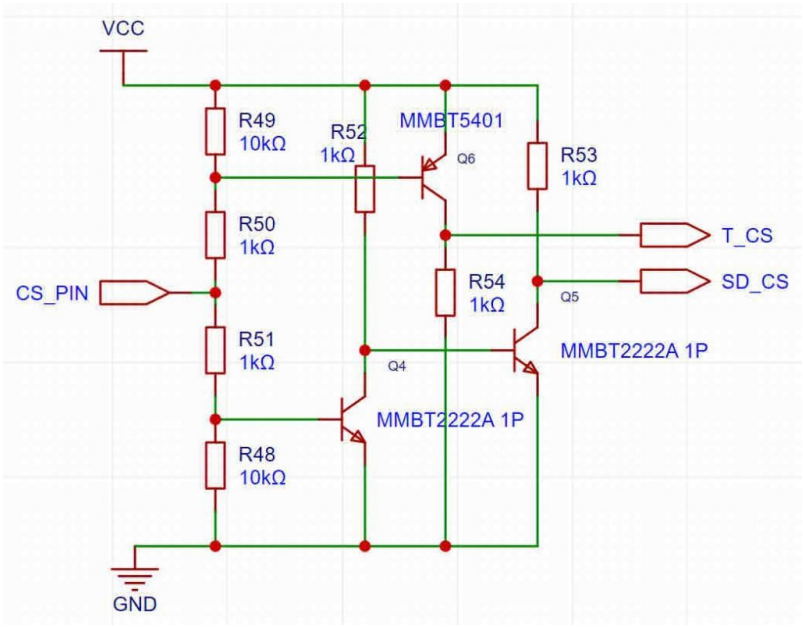


Beachten Sie, dass Sie viele verschiedene Konfigurationen mit unterschiedlichen Pin-Zuweisungen verwenden können – dies ist nur ein Beispiel, das auf den oben aufgeführten Konfigurationsbefehlen basiert.

Vorsicht ist geboten, wenn der SPI-Port von mehreren Geräten (SD-Karte, Touchscreen usw.) gemeinsam genutzt wird. In diesem Fall müssen alle Chip-Select-Signale in MMBasic konfiguriert oder alternativ durch eine permanente Verbindung mit 3,3 V deaktiviert werden. Andernfalls können schwebende Chip-Select-Signalleitungen dazu führen, dass der falsche Controller auf Befehle auf dem SPI-Bus reagiert.

Kombinierte Chip- -Auswahl

Der Chip-Select-Pin, der für die SD-Karte und den Touch-Controller auf einem LCD-Panel verwendet wird, kann mithilfe des Befehls OPTION SDCARD COMBINED CS kombiniert werden. Wenn dies angegeben ist, ist die folgende Schaltung erforderlich, um den SD-Chip-Select zu implementieren:



Die Firmware verwendet den Touch-Pin wie folgt:

TOUCH_CS niedrig: TOUCH_CS niedrig, SD_CS hoch

TOUCH CS hoch: SD_CS niedrig: TOUCH_CS hoch

TOUCH CS als Eingang (hohe Impedanz) TOUCH_CS und SD_CS hoch.

MMBasic-Unterstützung für Flash- und SD-Karten- -Dateisysteme

Die MMBasic-Unterstützung für das Flash-Dateisystem und SD-Karten ist nahezu identisch. Dadurch können Programme mit minimalen Änderungen beide Dateisysteme verwenden. Das Flash-Dateisystem wird als Laufwerk A: bezeichnet, während die SD-Karte (wenn angeschlossen) als Laufwerk B: bezeichnet wird. Das Standardlaufwerk kann mit dem Befehl DRIVE festgelegt werden, dann ist das Laufwerkspräfix nicht erforderlich.

Beachten Sie Folgendes:

- Groß-/Kleinschreibung und Leerzeichen sind zulässig. Das Dateisystem auf der SD-Karte unterscheidet **NICHT** zwischen Groß- und Kleinschreibung, das Flash-Dateisystem hingegen **schon** (dies ist der einzige Unterschied zwischen den beiden).
- Neben dem alten 8.3-Format werden auch lange Datei-/Verzeichnisnamen unterstützt.
- Die maximale Datei-/Pfadlänge beträgt 63 Zeichen (127 Zeichen für RP2350-Versionen).
- Jeder Dateipfad, der den Laufwerksbuchstaben verwendet, muss ein vollständiger Pfad vom Stammverzeichnis sein (z. B. „A:/mypath/myfile.txt“).
- Verzeichnispfade sind in Datei-/Verzeichniszeichenfolgen zulässig. (z. B. OPEN „A:\dir1\dir2\file.txt“ FOR …).
- In Pfaden können sowohl Vorwärts- als auch Rückwärtsschrägstriche verwendet werden. Beispielsweise entspricht \dir\file.txt dem Pfad /dir/file.txt.
- Beim Start ist das aktive Laufwerk (d. h. das Standardlaufwerk) A: (das Flash-Dateisystem).
- Die aktuelle PicoMite-Firmware-Zeit wird für die Erstellungs- und letzte Zugriffszeit von Dateien verwendet.
- Es können bis zu zehn Dateien gleichzeitig geöffnet sein, gemischt zwischen den Laufwerken A: und B:.
- Mit Ausnahme von INPUT, LINE INPUT und PRINT ist das # in #fnbr optional und kann weggelassen werden.

Mit diesen Befehlen können Programme aus dem Flash-Dateisystem und von SD-Karten geladen oder dort gespeichert

werden.

□ LOAD fname$ [, R]

Lädt ein BASIC-Programm. Das optionale Suffix „R“ bewirkt, dass das Programm nach dem Laden ausgeführt wird (in diesem Fall muss fname$ eine Zeichenfolgenkonstante sein).

- RUN fname$
  Laden Sie ein BASIC-Programm und führen Sie es aus. fname$ kann eine Variable sein.

- SAVE fname$
  Speichern Sie das aktuelle Programm im Flash-Dateisystem oder auf der SD-Karte.

Dies sind die grundlegenden Befehle zum Lesen und Schreiben von Daten.

- OPEN fname$ FOR mode AS #fnbr
  Öffnet eine Datei zum Lesen oder Schreiben. „fname$" ist der Dateiname. „mode" kann INPUT, OUTPUT, APPEND oder RANDOM sein. „#fnbr" ist die Dateinummer (1 bis 10).

- PRINT #fnbr, Ausdruck [[,;]Ausdruck] … usw. Gibt Text
  in die als #fnbr geöffnete Datei aus.

- INPUT #fnbr, Liste von Variablen
  Liest eine Liste von durch Kommas getrennten Daten in die angegebenen Variablen aus der zuvor als #fnbr geöffneten Datei.

- LINE INPUT #fnbr, variable$
  Liest eine vollständige Zeile in die angegebene String-Variable aus der zuvor als #fnbr geöffneten Datei.

- FLUSH #fnbr
  Erzwingt, dass alle gepufferten Schreibvorgänge in das Flash-Dateisystem oder auf die SD-Karte geschrieben werden. Es wird empfohlen, diesen
  Befehl regelmäßig verwendet wird, wenn bei einem Stromausfall Datenverlust auftreten könnte.

- CLOSE #fnbr [,#fnbr] …
  Schließt die zuvor mit der Dateinummer „#fnbr" geöffneten Dateien.

Grundlegende Datei- und Verzeichnisbearbeitung. Das meiste kann über die Eingabeaufforderung oder innerhalb eines BASIC-Programms erledigt werden.

- DRIVE Laufwerk$
  Legt das aktive Laufwerk als „drive$" fest. „drive$" kann „A:" oder „B:" sein, wobei A das Flash-Laufwerk und B die SD-Karte (sofern konfiguriert).

- DRIVE „A:/FORMAT"
  Formatiert das Flash-Dateisystem (Laufwerk A:) in seinen ursprünglichen Zustand zurück.

- FILES [Platzhalter]
  Durchsucht das aktuelle Verzeichnis und listet die gefundenen Dateien/Verzeichnisse auf.
  Hinweis: Kann nur an der Eingabeaufforderung verwendet werden, nicht innerhalb eines Programms.

- LIST fname$
  Listet den Inhalt eines Programms oder einer Textdatei auf der Konsole auf.

- KILL fname$
  Löscht eine Datei im aktuellen Verzeichnis auf dem aktuellen Laufwerk.
  Weitere Informationen zum Löschen mit Platzhaltern finden Sie in der Befehlsreferenz.

- MKDIR dname$
  Erstellt ein Unterverzeichnis im aktuellen Verzeichnis auf dem aktuellen Laufwerk.

- CHDIR dname$
  Wechseln Sie in das Verzeichnis $dname. $dname kann auch „.." (Punkt Punkt) für das übergeordnete Verzeichnis oder „\" für das Stammverzeichnis. Der Ausgangspunkt ist das aktuelle Verzeichnis auf dem aktuellen Laufwerk.

- RMDIR dir$
  Entfernen oder löschen Sie das Verzeichnis „dir$" im aktuellen Verzeichnis auf dem aktuellen Laufwerk.

- SEEK #fnbr, pos
  Positioniert den Lese-/Schreibzeiger in einer Datei, die für den zufälligen Zugriff geöffnet wurde, auf das Byte „pos".

- RENAME fromname$ AS toname$
  Benennt die Datei fromname$ im aktuellen Verzeichnis auf dem aktuellen Laufwerk in toname$ um.

- COPY [Modus] fromname$ TO toname$
  Kopiert die Datei fromname$ und ~~benennt~~ sie in toname$ um.
  Weitere Informationen zu den optionalen Kopiermodi und Platzhaltern finden Sie in der Befehlsreferenz.

Außerdem gibt es eine Reihe von Funktionen, die die oben genannten Befehle unterstützen.

- INPUT$(nbr, #fnbr)
  Gibt eine Zeichenkette zurück, die aus „nbr“ Zeichen besteht, die aus einer zuvor für INPUT oder RANDOM mit der Dateinummer „#fnbr“ gelesen wurden. Wenn weniger als „nbr“ Zeichen verfügbar sind, gibt die Funktion das zurück, was sie hat (einschließlich einer leeren Zeichenkette, wenn keine Zeichen verfügbar sind).

- DIR$(fspec, type)
  Sucht nach Dateien und gibt die Namen der gefundenen Einträge zurück.

- CWD$
  Gibt das aktuelle Arbeitsverzeichnis zurück.

- EOF(#fnbr)
  Gibt „true“ zurück, wenn die zuvor für INPUT mit der Dateinummer „#fnbr“ geöffnete Datei am Ende der Datei positioniert ist.

- LOC(#fnbr)
  Bei einer geöffneten Datei gibt dies die aktuelle Position des Lese-/Schreibzeigers in der Datei zurück.

- LOF(#fnbr)
  Gibt die aktuelle Länge der Datei in Byte zurück.

- MM.INFO(Laufwerk)
  Gibt das aktuell aktive Laufwerk zurück, d. h. „A:“ oder „B:“.

- MM.INFO(freier Speicherplatz)
  Gibt an, wie viel Speicherplatz auf dem aktiven Laufwerk noch verfügbar ist.

- MM.INFO(Festplattengröße)
  Gibt die Größe des aktiven Laufwerks zurück

- MM.INFO(exists file fname$) Gibt
  „true” zurück, wenn die Datei vorhanden ist

- MM.INFO(exists dir dirname$)
  Gibt „true“ zurück, wenn das Verzeichnis existiert

- MM.INFO(Pfad)
  Gibt den Dateipfad des ausgeführten Programms zurück. Auf diese Weise kann der Benutzer relative Pfadverweise für alle in einem Programm benötigten Ressourcendateien erstellen.

XModem- -Übertragung

Zusätzlich zur Standardmethode der XModem-Übertragung, bei der Daten in den oder aus dem Programmspeicher kopiert werden, kann die PicoMite-Firmware auch Daten in eine oder aus einer Datei auf dem Flash-Dateisystem oder der SD-Karte kopieren. Die Syntax lautet:

```
XMODEM SEND Dateiname$
```

oder

```
XMODEM RECEIVE Dateiname$
```

Dabei ist „Dateiname$” die Datei, die gespeichert oder gesendet werden soll. „Dateiname$” kann ein String-Ausdruck, eine Variable oder eine Konstante sein. Handelt es sich um eine Konstante, muss der String in Anführungszeichen gesetzt werden (z. B. XMODEM SEND „prbas.bas”). Beim Empfang einer Datei wird jede Datei mit dem gleichen Namen überschrieben.

-Image laden und speichern

Ein Bild kann aus dem Flash-Dateisystem oder von einer SD-Karte geladen und auf einem angeschlossenen LCD-Display oder VGA/HDMI-Monitor angezeigt werden. Dies kann verwendet werden, um ein Logo zu zeichnen oder einen Hintergrund auf dem Display hinzuzufügen.

Die Syntax lautet:

```
LOAD IMAGE Dateiname$ [, StartX, StartY] (BMP-Bild laden) oder
LOAD JPG Dateiname$ [, StartX, StartY]
```

```
oder    LOAD PNG Dateiname$ [, StartX, StartY]      (nur Pico2/RP2350)
```

Dabei ist „Dateiname$" das zu ladende Bild und „StartX"/„StartY" sind die Koordinaten der oberen linken Ecke des Bildes (diese sind optional und werden standardmäßig auf die obere linke Ecke des Displays gesetzt, wenn sie nicht angegeben werden).

Das Bild muss im entsprechenden Format (BMP, JPG oder PNG) vorliegen. MMBasic fügt die Erweiterung zum Dateinamen hinzu, wenn diese nicht angegeben ist. Es werden alle Bildtypen unterstützt, einschließlich Schwarzweiß- und Echtfarbenbilder mit 24 Bit.

Das aktuelle Bild auf dem Videoausgang, dem virtuellen LCD oder einem LCD-Panel, das BLIT unterstützt, kann mit dem folgenden Befehl in einer Datei gespeichert werden:

```
SAVE IMAGE Dateiname$ [,StartX, StartY, Breite, Höhe]
```

Dadurch wird das Bild oder ein Teil des Bildes als 24-Bit-True-Color-BMP-Datei gespeichert (die Erweiterung .BMP wird hinzugefügt, wenn keine Erweiterung angegeben ist).

Beispiel für sequentielle E/A-

Im folgenden Beispiel wird eine Datei erstellt und zwei Zeilen in die Datei geschrieben (mit dem Befehl PRINT). Anschließend wird die Datei geschlossen.

```
OPEN „fox.txt" FOR OUTPUT AS #1
PRINT #1, "Der schnelle braune Fuchs" PRINT
#1, "springt über den faulen Hund" CLOSE #1
```

Sie können den Inhalt der Datei mit dem Befehl LINE INPUT lesen. Beispiel:

```
OPEN "fox.txt" FOR INPUT AS #1 LINE
INPUT #1,a$
LINE INPUT #1,b$
CLOSE #1
```

LINE INPUT liest jeweils eine Zeile, sodass die Variable a$ den Text „Der schnelle braune Fuchs" und b$ den Text „springt über den faulen Hund" enthält.

Eine weitere Möglichkeit, aus einer Datei zu lesen, ist die Verwendung der Funktion INPUT$(). Diese liest eine bestimmte Anzahl von Zeichen. Beispiel:

```
OPEN "fox.txt" FOR INPUT AS #1 ta$ =
INPUT$(12, #1)
tb$ = INPUT$(3, #1) CLOSE #1
```

Die erste INPUT$()-Funktion liest 12 Zeichen und die zweite drei Zeichen. Die Variable ta$ enthält also „The quick br" und die Variable tb$ enthält „own".

Dateien enthalten normalerweise nur Text, und der Befehl „print" konvertiert Zahlen in Text. Im folgenden Beispiel enthält die erste Zeile also „123" und die zweite „56789".

```
nbr1 = 123 : nbr2 = 56789
OPEN „numbers.txt" FOR OUTPUT AS #1 PRINT #1,
nbr1
PRINT #1, nbr2
CLOSE #1
```

Sie können den Inhalt dieser Datei mit dem Befehl LINE INPUT lesen, müssen den Text dann jedoch mit VAL() in eine Zahl umwandeln.

Beispiel:

```
OPEN "numbers.txt" FOR INPUT AS #1 LINE
INPUT #1, a$
LINE INPUT #1, b$
CLOSE #1
x = VAL(a$) : y = VAL(b$)
```

Danach hätte die Variable x den Wert 123 und y den Wert 56789.

Zufällige Datei- -E/A

Für den zufälligen Zugriff sollte die Datei mit dem Schlüsselwort RANDOM geöffnet werden. Zum Beispiel:

```
OPEN „Dateiname” FOR RANDOM AS #1
```

Um einen Datensatz innerhalb der Datei zu suchen, verwenden Sie den Befehl SEEK, der den Lese-/Schreibzeiger auf ein bestimmtes Byte positioniert. Das erste Byte in einer Datei ist mit eins nummeriert, sodass beispielsweise der fünfte Datensatz in einer Datei, die 64-Byte-Datensätze verwendet, bei Byte 257 beginnt. In diesem Fall würden Sie Folgendes verwenden, um darauf zu verweisen:

```
SEEK #1, 257
```

Beim Lesen aus einer Datei mit wahlfreiem Zugriff sollte die Funktion INPUT$() verwendet werden, da diese eine feste Anzahl von Bytes (d. h. einen vollständigen Datensatz) aus der Datei liest. Um beispielsweise einen Datensatz mit 64 Bytes zu lesen, würden Sie Folgendes verwenden:

```
dat$ = INPUT$(64, #1)
```

Beim Schreiben in die Datei sollte eine feste Datensatzgröße verwendet werden. Dies lässt sich leicht erreichen, indem den zu schreibenden Daten ausreichend Füllzeichen (normalerweise Leerzeichen) hinzugefügt werden. Beispiel:

```
PRINT #1, dat$ + SPACE$(64 – LEN(dat$));
```

Die Funktion SPACE$() wird verwendet, um genügend Leerzeichen hinzuzufügen, damit die geschriebenen Daten eine bestimmte Länge haben (in diesem Beispiel 64 Byte). Das Semikolon am Ende des Druckbefehls unterdrückt das Hinzufügen der Zeichen für Wagenrücklauf und Zeilenvorschub, die den Datensatz länger als beabsichtigt machen würden.

Zwei weitere Funktionen können bei der Verwendung des zufälligen Dateizugriffs hilfreich sein. Die Funktion LOC() gibt die aktuelle Byte-Position des Lese-/Schreibzeigers zurück, und die Funktion LOF() gibt die Gesamtlänge der Datei in Bytes zurück.

Das folgende Programm demonstriert den zufälligen Dateizugriff. Mit ihm können Sie an die Datei anhängen (um zunächst einige Daten hinzuzufügen) und dann Datensätze mit zufälligen Datensatznummern lesen/schreiben. Der erste Datensatz in der Datei ist Datensatznummer 1, der zweite ist 2 usw.

```
RecLen = 64
ÖFFNE „test.dat” FÜR ZUFALL ALS #1

DO
 abbrechen: PRINT
 PRINT „Anzahl der Datensätze in der Datei =“ LOF(#1)/RecLen
 INPUT „Befehl (r = lesen, w = schreiben, a = anhängen, q = beenden): ", cmd$ IF
 cmd$ = „q" THEN CLOSE #1 : END
 WENN cmd$ = „a“ DANN SUCHEN
     #1, LOF(#1) + 1
 SONST
     INPUT "Datensatznummer: ", nbr
     WENN nbr < 1 oder nbr > LOF(#1)/RecLen DANN DRUCKEN „Ungültiger Datensatz” : GOTO Abbruch
     SUCHEN #1, RecLen * (nbr - 1) + 1
 ENDIF
 WENN cmd$ = "r" DANN
     PRINT "Der Datensatz = " INPUT$(RecLen, #1) ELSE
     LINE INPUT "Geben Sie die zu schreibenden Daten ein: ", dat$
     PRINT #1,dat$ + SPACE$(RecLen - LEN(dat$));
 ENDIF
LOOP
```

Der Direktzugriff kann auch auf eine normale Textdatei angewendet werden. Mit dem folgenden Befehl wird beispielsweise eine Datei rückwärts ausgegeben:

```
OPEN „file.txt” FOR RANDOM AS #1 FOR i
= LOF(#1) TO 1 STEP -1
    SEEK #1, i
    PRINT INPUT$(1, #1); NEXT i
CLOSE #1
```

Sound- -Ausgabe

Die PicoMite-Firmware kann Stereo-WAV-, FLAC-, MP3- oder MOD-Dateien aus dem Flash-Dateisystem oder von der SD-Karte abspielen und mit dem Befehl PLAY präzise Sinuswellen erzeugen.

Beachten Sie, dass der Schaltregler des Raspberry Pi Pico zu Störgeräuschen am Audioausgang führen kann. Diese können reduziert werden, indem Sie den Regler deaktivieren und das Modul über einen externen Linearregler mit Strom versorgen.

Pulsweitenmoduliertes (PWM) Signal „ “

Der Ton wird über PWM-Ausgänge erzeugt. Bevor die PLAY-Befehle verwendet werden können, müssen daher die PWM-Ausgangspins als Audioausgänge zugewiesen werden:

Dies geschieht mit dem Befehl OPTION AUDIO wie folgt:

```
OPTION AUDIO PWM-A-PIN, PWM-B-PIN
```

Dieser Befehl sollte an der Eingabeaufforderung eingegeben werden und wird gespeichert, sodass er nur einmal ausgeführt werden muss. Beide Pins müssen sich auf demselben PWM-Kanal befinden, wobei PWM-A-PIN der linke Audiokanal und PWM-B-PIN der rechte ist.

Beispiel:

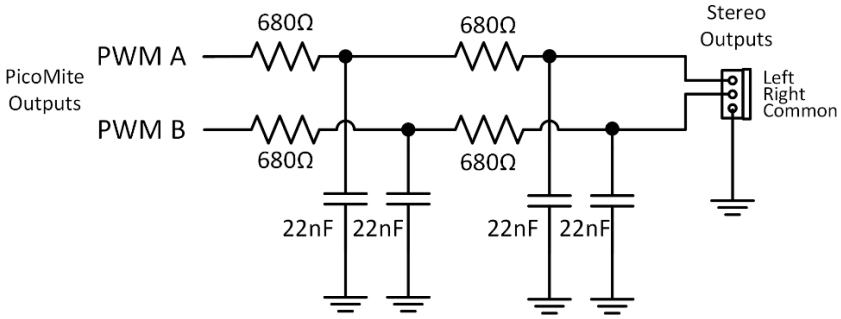
```
OPTION AUDIO GP0, GP1
```

Das Audiosignal wird als pulsweitenmoduliertes (PWM) Signal einer 44-kHz-Rechteckwelle (der sogenannten Trägerwelle) überlagert. Das bedeutet, dass ein Tiefpassfilter erforderlich ist, um die Trägerwelle zu entfernen und das Audiosignal wiederherzustellen.

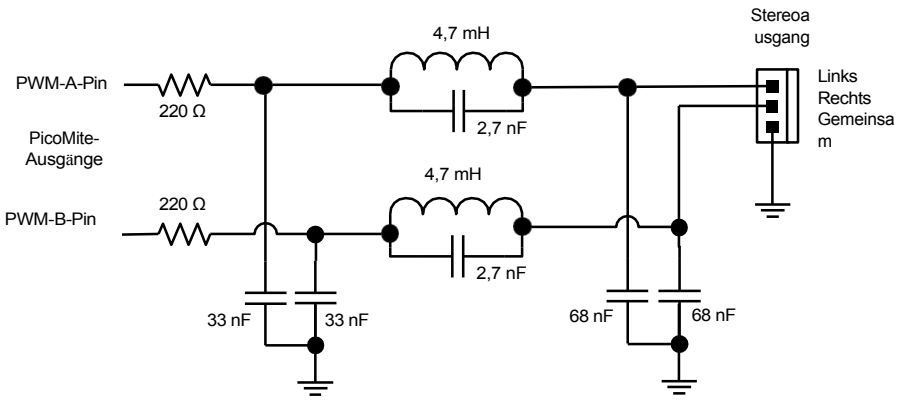
Filter sschaltungen

Die meisten kostengünstigen Verstärkerlautsprecher (für PCs) reagieren nicht auf die Trägerfrequenz, sodass sie selbst als Tiefpassfilter fungieren. Wenn Sie es also einfach halten möchten, können Sie den PWM-Ausgang direkt an den Eingang eines Verstärkerlautsprechers anschließen, um eine angemessene Tonausgabe zu erzielen.

Der hohe Pegel der 44-kHz-Trägerfrequenz kann jedoch zu Problemen für den Verstärker führen (z. B. Überhitzung oder Verzerrung), daher wird der folgende Filter empfohlen. Dieser entfernt den größten Teil der Trägerfrequenz und liefert etwa 2 V Spitze-Spitze (0,6 V RMS) mit einer angemessenen Wiedergabetreue bis zu 8 kHz (mehr als ausreichend für die meisten verstärkten Lautsprecher):



Nachstehend finden Sie eine hochwertige Schaltung, die eine hervorragende Audioqualität mit einem vernachlässigbaren Restanteil der Trägerfrequenz erzeugt. Diese eignet sich für eine anspruchsvollere HiFi-Verstärker-/Lautsprecherkonfiguration. Die Ausgangsleistung ist gut für 10 Hz bis 15 kHz bei etwa 3 V Spitze-Spitze (1 V RMS) bei 1 kHz.



Beide Schaltungen sind für die Speisung eines Verstärkers ausgelegt (nicht für den direkten Betrieb eines Kopfhörers oder Lautsprechers) und basieren auf einer Kondensatorkopplung in den nachfolgenden Verstärker (die meisten verfügen darüber).

Unterstützung für VS1053-

Der Audioausgang kann mit einem VS1053-CODEC-Modul erzeugt werden, das mit dem Befehl

```
OPTION AUDIO VS1053 CLK-Pin, MOSI-Pin, MISO-Pin, XCS-Pin, XDC-Pin, DREQ-Pin, XRST-Pin
```

Dies erfordert keine Ausgangsfilterung und kann 32-Ω-Kopfhörer direkt ansteuern. Außerdem werden zusätzliche Wiedergabefunktionen unterstützt.

Wenn ein VS1053-Codec als Audioausgabegerät verwendet wird, stehen zusätzliche Befehle zur Verfügung:

```
PLAY MP3 file$, interrupt PLAY
MIDIFILE file$, interrupt PLAY MIDI
PLAY MIDI CMD cmd%, data1% [,data2%] PLAY NOTE
ON channel, note, velocity PLAY NOTE OFF
channel, note [,velocity] PLAY HALT
PLAY CONTINUE track$
PLAY STREAM buffer%(), readpointer%, writepointer%
```

Diese werden im Abschnitt „Befehlsliste" näher erläutert.

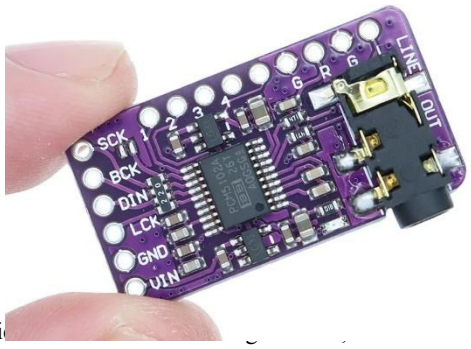
MCP48n2- -DAC

Der Audioausgang kann auch über einen angeschlossenen MCP48n2-DAC (z. B. MCP4822) erzeugt werden. In diesem Fall wird er mit dem folgenden Befehl konfiguriert:

```
OPTION AUDIO SPI CS-PIN, CLK-PIN, MOSI-PIN
```

Der DAC benötigt keinen komplexen Tiefpassfilter. Ein 120-Ω-Widerstand, der mit dem DAC-Ausgang verbunden ist und dessen anderes Ende über einen 100-nF-Kondensator mit GND verbunden ist, ist ausreichend. Bei Verwendung eines MCP4822 sollte der LDAC-Pin am DAC mit GND verbunden werden.

I2S-DAC



Der Audioausgang kann auch über einen I2S-DAC wie den PCM5102A erzeugt werden. Der DAC muss die Erstellung eines eigenen Master-Clocks unterstützen, da dieser nicht von der Firmware erstellt wird. Der I2S-DAC auf dem Pico2 (RP2350A/B) verwendet PIO2 zur Erzeugung des Ausgangs, sodass dieser nicht verfügbar ist, wenn der I2S-DAC aktiviert ist. Der I2S-DAC auf dem RP2040 verwendet PIO 0, das bei VGA-Versionen gemeinsam mit dem VGA-PIO genutzt wird.

Der I2S-DAC wird mit dem folgenden Befehl konfiguriert:

```
OPTION AUDIO I2S BCLK-PIN, DIN-PIN
```

Der I2S-Worttakt (LRCK) befindet sich dann am nächsten Pin zum BCLK. Wenn beispi
sich LRCK auf GP1. Beide Pins und der DIN-Pin müssen bei Ausgabe des Befehls unbenutzt sein.
In der Regel muss beim DAC-Modul auch ein GND- und ein Stromanschluss (in der Regel 5 V) angeschlossen werden.

Der I2S-DAC erzeugt Audio in CD-Qualität aus FLAC-Dateien und gibt MP3-Dateien aus, die nur durch die inhärente MP3-Komprimierung begrenzt sind. FLAC-Dateien mit bis zu 96000 Hz und 24 Bit wurden getestet.

Wiedergabe von WAV-, FLAC-, MP3- und MOD- -Dateien

Der Befehl PLAY kann eine WAV-, FLAC-, MP3- (nur RP2350) oder MOD-Datei, die sich auf dem Flash-Dateisystem oder der SD-Karte befindet, über den Audioausgang wiedergeben. Er kann verwendet werden, um Hintergrundmusik bereitzustellen, Programmen Soundeffekte hinzuzufügen und informative Ansagen auszugeben.

Die Befehle lauten:

```
PLAY WAV file$, interrupt PLAY
FLAC file$, interrupt PLAY MODFILE
file$, interrupt
PLAY MP3 file$, interrupt 'nur RP2350
```

„file$" ist der Name der wiederzugebenden Audiodatei. Sie muss sich im Flash-Dateisystem oder auf der SD-Karte befinden, und die entsprechende Erweiterung (z. B. .WAV) wird angehängt, wenn sie fehlt. Die Audiodatei wird im Hintergrund wiedergegeben (d. h., das Programm wird ohne Unterbrechung fortgesetzt). „interrupt" ist optional und ist der Name einer Subroutine, die aufgerufen wird, wenn die Wiedergabe der Datei beendet ist.

Erzeugen von Sinus swellen

Der Befehl PLAY TONE verwendet den Audioausgang, um Sinuswellen mit wählbaren Frequenzen für den linken und rechten Kanal zu erzeugen. Diese Funktion dient dazu, aufmerksamkeitsstarke Töne zu erzeugen, kann aber aufgrund der hohen Frequenzgenauigkeit auch für viele andere Anwendungen genutzt werden. Beispielsweise zum Senden von DTMF-Tönen über eine Telefonleitung oder zum Testen des Frequenzgangs von Lautsprechern.

Die Syntax des Befehls lautet:

```
PLAY TONE links, rechts, Dauer, Unterbrechung
```

„left" und „right" sind die Frequenzen in Hz, die für den linken und rechten Kanal verwendet werden sollen.

Der Ton wird im Hintergrund abgespielt (das Programm läuft nach diesem Befehl weiter) und „duration" gibt die Anzahl der Millisekunden an, die der Ton erklingen soll. „duration" ist optional und wenn es nicht angegeben wird, erklingt der Ton so lange, bis er explizit gestoppt wird oder das Programm beendet wird. „interrupt" (falls angegeben) wird ausgelöst, wenn die Dauer abgelaufen ist.

Die angegebene Frequenz kann zwischen 1 Hz und 20 kHz liegen und ist sehr genau (sie basiert auf einem Quarzoszillator). Die Frequenz kann jederzeit durch einen neuen Befehl „PLAY TONE" geändert werden.

Verwendung von „ " PLAY

Es ist wichtig zu wissen, dass der Befehl PLAY den Ton im Hintergrund erzeugt. Dadurch kann ein Programm (zum Beispiel) den Klang einer Glocke abspielen, während es seine Steuerungsfunktion fortsetzt. Ohne die Hintergrundfunktion würde das gesamte BASIC-Programm während der Tonwiedergabe einfrieren.

Die Audioausgabe im Hintergrund hat jedoch einige subtile Auswirkungen, die Neulinge verwirren können. Nehmen wir zum Beispiel das folgende Programm:

```
PLAY TONE 500, 500, 2000 END
```

Man könnte erwarten, dass der 500-Hz-Ton 2 Sekunden lang zu hören ist, aber in der Praxis wird überhaupt kein Ton zu hören sein. Das liegt daran, dass MMBasic den Befehl PLAY TONE ausführt (der die Tonerzeugung im Hintergrund startet) und dann sofort den Befehl END ausführt, der das Programm und den Hintergrundton beendet. Dies geschieht so schnell, dass nichts zu hören ist.

Ebenso funktioniert das folgende Programm nicht:

```
PLAY TONE 500, 500, 2000
PLAY TONE 300, 300, 5000
```

Das liegt daran, dass der erste Befehl einen Ton mit 500 Hz einstellt, der zweite PLAY-Befehl diesen jedoch sofort durch einen Ton mit 300 Hz ersetzt, woraufhin das Programm bis zum Ende läuft und das Programm (und die Hintergrundmusik) beendet, sodass nichts zu hören ist.

Wenn Sie möchten, dass MMBasic wartet, während der PLAY-Befehl ausgeführt wird, sollten Sie geeignete PAUSE-Befehle verwenden. Zum Beispiel:

```
PLAY TONE 500, 500 : PAUSE 2000
PLAY TONE 300, 300 : PAUSE 5000 PLAY
STOP
```

Dies gilt für alle Versionen des Befehls PLAY, einschließlich PLAY WAV.

Dienstprogramm- -Befehle

Es gibt eine Reihe von Befehlen, mit denen die Tonausgabe verwaltet werden kann:

`PLAY PAUSE` Die aktuell wiedergegebene Datei oder den Ton vorübergehend anhalten (pausieren). `PLAY RESUME` Fortsetzen der Wiedergabe einer zuvor angehaltenen Datei oder eines zuvor angehaltenen Tons. `PLAY NEXT` Die nächste WAV-, MP3- oder FLAC-Datei in einem Verzeichnis abspielen. `PLAY PREVIOUS` Die vorherige WAV-, MP3- oder FLAC-Datei in einem Verzeichnis abspielen

`STOP` Beenden Sie die Wiedergabe der Datei oder des Tons, wie beim Beenden des Programms.

`LAUTSTÄRKE L, R` Stellen Sie die Lautstärke zwischen 0 und 100 ein, wobei 100 die maximale Lautstärke ist. Die Lautstärke wird beim Ausführen eines Programms auf den maximalen Pegel zurückgesetzt. Es wird eine logarithmische Skala verwendet, sodass PLAY VOLUME 50,50 halb so laut wie 100,100 klingen sollte.

Spezielle Audio- -Ausgabe

Der Befehl PLAY SOUND erzeugt eine Ausgabe, die auf einer Mischung aus Sinus-, Rechteck- und anderen Wellenformen basiert. Details finden Sie in der Befehlsliste.

Verwendung der I/O- -Pins

Der Raspberry Pi Pico verfügt über 26 Ein-/Ausgangspins, die aus dem BASIC-Programm heraus gesteuert werden können, wobei 3 davon einen Hochgeschwindigkeits-ADC (Analog-Digital-Wandler) unterstützen.

Ein I/O-Pin wird anhand seiner Pin-Nummer bezeichnet, wobei es sich um die Nummer (z. B. 2) oder die GP-Nummer (z. B. GP1) handeln kann.

Digitale Ein-/Ausgänge

Ein digitaler Eingang ist die einfachste Art der Eingangskonfiguration. Wenn die Eingangsspannung höher als 2,5 V ist, ist der Logikpegel wahr (numerischer Wert 1), und alles unter 0,65 V ist falsch (numerischer Wert 0). Die Eingänge verwenden einen Schmitt-Trigger-Eingang, sodass alles zwischen diesen Pegeln den vorherigen Logikpegel beibehält.

Beachten Sie, dass die maximale Spannung an den RP2040-I/O-Pins (d. h. dem Raspberry Pi Pico) 3,3 V beträgt. Eine Pegelverschiebung ist erforderlich, wenn ein Gerät 5-V-Pegel für die Signalübertragung verwendet. Der Raspberry Pi Pico 2 mit dem RP2350 kann 5 V tolerieren (bei eingeschaltetem Gerät), sodass in diesem Fall keine Pegelverschiebung für Signale bis zu 5 V erforderlich ist.

In Ihrem BASIC-Programm würden Sie den Eingang als digitalen Eingang festlegen und die Funktion PIN() verwenden, um seinen Pegel zu ermitteln. Zum Beispiel:

```
SETPIN GP4, DIN
IF PIN(GP4) = 1 THEN PRINT „High”
```

Der Befehl SETPIN konfiguriert Pin GP4 als digitalen Eingang, und die Funktion PIN() gibt den Wert dieses Pins zurück (die Zahl 1, wenn der Pin hoch ist). Der Befehl IF führt dann den Befehl nach der THEN-Anweisung aus, wenn der Eingang hoch war. Wenn der Eingangspin niedrig war, würde das Programm einfach mit der nächsten Zeile im Programm fortfahren.

Der Befehl SETPIN erkennt auch einige Optionen, die einen internen Widerstand vom Eingang entweder mit der Versorgungsspannung oder mit Masse verbinden. Dies wird als „Pullup“- oder „Pulldown“-Widerstand bezeichnet und ist praktisch beim Anschluss an einen Schalter, da dadurch die Installation eines externen Widerstands zur Anlegung einer Spannung an die Kontakte entfällt. Aufgrund eines Hardwareproblems mit dem RP2350-Prozessor wird empfohlen, einen externen Widerstand von 8,2 K oder weniger zu verwenden, wenn ein Pulldown für diesen Prozessor erforderlich ist.

Analoge Eingänge ( )

Mit ADC gekennzeichnete Pins können so konfiguriert werden, dass sie die Spannung am Pin messen. Der Eingangsbereich reicht von null bis 3,3 V, und die Funktion PIN() gibt die Spannung zurück. Beispiel:

```
> SETPIN 31, AIN
> PRINT PIN(31)
 2,345
>
```

Sie benötigen einen Spannungsteiler, wenn Sie Spannungen über 3,3 V messen möchten. Bei kleinen Spannungen benötigen Sie möglicherweise einen Verstärker, um die Eingangsspannung in einen für die Messung geeigneten Bereich zu bringen.

Die Messung verwendet die 3,3-V-Stromversorgung der CPU als Referenz und es wird davon ausgegangen, dass diese genau 3,3 V beträgt. Dieser Wert kann mit dem Befehl OPTION VCC geändert werden. Um den bestmöglichen Messwert zu erhalten, wird der analoge Eingang 10 Mal abgetastet. Die Werte werden dann sortiert, die beiden höchsten und die beiden niedrigsten Werte verworfen und die verbleibenden 6 Werte gemittelt.

Wenn Sie den direkten Messwert vom ADC erhalten möchten, können Sie den Raw-Modus verwenden, indem Sie die Option ARAW zum Befehl SETPIN verwenden:

```
SETPIN pinno,ARAW
```

In diesem Fall wird ein Wert zwischen 0 und 4095 basierend auf einer einzelnen Probe zurückgegeben.

Die ADC-Befehle bieten eine alternative Methode zur Aufzeichnung analoger Eingänge und sind für die Hochgeschwindigkeitsaufzeichnung vieler Messwerte in einem Array vorgesehen.

Zähleingänge des

Beliebige vier Pins können als Zähleingänge verwendet werden, um die Frequenz, Periode oder einfach nur die Impulse am Eingang zu messen. Die für diese Funktion verwendeten Pins können mit dem Befehl OPTION COUNT konfiguriert werden, werden jedoch, wenn sie nicht geändert werden, standardmäßig auf GP6, GP7, GP8 und GP9 gesetzt.

Als Beispiel wird im Folgenden die Frequenz des Signals an Pin GP7 ausgegeben:

```
> SETPIN GP7, FIN
> PRINT PIN(GP7)
110374
>
```

In diesem Fall beträgt die Frequenz 110,374 kHz.

Standardmäßig beträgt die Gate-Zeit eine Sekunde. Dies ist die Zeitspanne, die MMBasic zum Zählen der Zyklen am Eingang benötigt, was bedeutet, dass der Messwert einmal pro Sekunde mit einer Auflösung von 1 Hz aktualisiert wird. Durch Angabe eines dritten Arguments für den Befehl SETPIN kann eine alternative Gate-Zeit zwischen 10 ms und 100000 ms festgelegt werden. Kürzere Zeiten führen dazu, dass die Messwerte häufiger aktualisiert werden, aber der zurückgegebene Wert hat eine geringere Auflösung. Die Funktion PIN() skaliert die zurückgegebene Zahl immer als Frequenz in Hz, unabhängig von der verwendeten Gate-Zeit.

Beispielsweise wird mit dem folgenden Befehl die Gate-Zeit auf 10 ms eingestellt, was mit einem entsprechenden Verlust an Auflösung einhergeht:

```
> SETPIN GP7, FIN, 10
> PRINT PIN(GP7)
110300
>
```

Für die genaue Messung von Signalen unter 10 Hz ist es im Allgemeinen besser, die Periode des Signals zu messen. In diesem Modus misst die PicoMite-Firmware die Anzahl der Millisekunden zwischen aufeinanderfolgenden steigenden Flanken des Eingangssignals. Der Wert wird beim Übergang von niedrig zu hoch aktualisiert. Wenn Ihr Signal also eine Periode von (sagen wir) 100 Sekunden hat, sollten Sie damit rechnen, dass Sie diese Zeit abwarten müssen, bevor die Funktion PIN() einen aktualisierten Wert zurückgibt.

Die Zählpins können auch die Anzahl der Impulse an ihrem Eingang zählen. Wenn ein Pin als Zähler konfiguriert ist (z. B. `SETPIN 7,CIN`), wird der Zähler auf Null zurückgesetzt und die PicoMite-Firmware zählt dann jeden Übergang von einer niedrigen zu einer hohen Spannung. Der Zähler kann durch Ausführen `von PIN(7) = 0` wieder auf Null zurückgesetzt werden.

Die Zähleingänge sind bei der Standardfrequenz des Prozessors bis zu etwa 200 kHz genau. Zum Auslösen des Zählers ist eine minimale Impulsbreite von etwa 40 nS erforderlich. Der RP2350 bietet außerdem die Möglichkeit, GP1 als extrem schnellen Frequenzzähl-Pin zu konfigurieren (siehe Befehl SETPIN GP1, FFIN).

Digitale Ausgänge „ “

Alle I/O-Pins können mit dem Parameter DOUT des Befehls SETPIN als digitaler Ausgang konfiguriert werden. Beispiel:

```
SETPIN GP15, DOUT
```

Das bedeutet, dass ein Ausgangspin, wenn er auf logisch niedrig gesetzt ist, seinen Ausgang auf Null zieht, und wenn er auf hoch gesetzt ist, seinen Ausgang auf 3,3 V zieht. In MMBasic wird dies mit dem Befehl PIN durchgeführt. Beispielsweise setzt `PIN(GP15) = 0` den Pin GP15 auf niedrig, während `PIN(GP15) = 1` ihn auf hoch setzt.

Pulsweiten smodulation

Der Befehl PWM (Pulsweitenmodulation) ermöglicht es der PicoMite-Firmware, Rechteckwellen mit einem programmgesteuerten Tastverhältnis zu erzeugen.

Durch Variieren des Tastverhältnisses können Sie einen programmgesteuerten Spannungsausgang zur Steuerung externer Geräte erzeugen, die einen analogen Eingang benötigen (Netzteile, Motorsteuerungen usw.). Die PWM-Ausgänge sind auch nützlich für den Antrieb von Servos und zur Erzeugung einer Tonausgabe über einen kleinen Wandler.

RP2040 Die PWM-Ausgänge bestehen aus bis zu 8 Kanälen (nummeriert von 0 bis 7), wobei jeder Kanal über zwei Ausgänge (A und B) verfügt. Für jeden Kanal kann die Frequenz ausgewählt und für jeden Ausgang ein anderer Tastgrad eingestellt werden. Mit dem Befehl SETPIN können bis zu 16 Pins als PWM-Ausgänge konfiguriert werden.

RP2350 Der RP2350 unterstützt bis zu 12 PWM-Kanäle (nummeriert von 0 bis 11) und bis zu 24 Pins können mit dem Befehl SETPIN als PWM-Ausgänge konfiguriert werden.

Kommunikationsschnittstellen (seriell, SPI und I^2C)

Diese werden in den Anhängen am Ende dieses Handbuchs beschrieben. Bevor diese Schnittstellen verwendet werden können, müssen die Pins, die für die entsprechenden Signale verwendet werden sollen, mit dem Befehl SETPIN konfiguriert werden.

Einige Geräte wie SD-Karten, LCD-Panels, Touchscreens usw. verwenden ebenfalls SPI- oder I2C-Schnittstellen, und die dafür verwendeten Pins müssen ebenfalls mit dem Befehl OPTION SYSTEM konfiguriert werden, bevor sie verwendet werden können.

Interrupts

Interrupts sind eine praktische Möglichkeit, um Ereignisse zu behandeln, die zu einem unvorhersehbaren Zeitpunkt auftreten können. Ein Beispiel hierfür ist das Drücken einer Taste durch den Benutzer. In Ihrem Programm könnten Sie nach jeder Anweisung einen Code einfügen, um zu überprüfen, ob die Taste gedrückt wurde, aber ein Interrupt sorgt für ein übersichtlicheres und besser lesbares Programm.

Wenn ein Interrupt auftritt, führt MMBasic eine definierte Subroutine aus und kehrt nach Abschluss zum Hauptprogramm zurück. Das Hauptprogramm nimmt den Interrupt überhaupt nicht wahr und läuft normal weiter.

Jeder I/O-Pin, der als digitaler Eingang verwendet werden kann, kann mit dem Befehl SETPIN so konfiguriert werden, dass er einen Interrupt generiert, wobei bis zu zehn Interrupts gleichzeitig aktiv sein können. Interrupts können so eingerichtet werden, dass sie bei einem steigenden oder fallenden digitalen Eingangssignal (oder beidem) auftreten und einen sofortigen Sprung zu der angegebenen benutzerdefinierten Subroutine bewirken. Das Ziel kann für jeden Interrupt gleich oder unterschiedlich sein. Die Rückkehr aus einem Interrupt erfolgt über die Befehle END SUB oder EXIT SUB. Beachten Sie, dass keine Parameter an die Subroutine übergeben werden können, jedoch sind innerhalb des Interrupts Aufrufe anderer Subroutinen und Funktionen zulässig.

Wenn zwei oder mehr Interrupts gleichzeitig auftreten, werden sie in der unten definierten Reihenfolge verarbeitet. Während der Verarbeitung eines Interrupts werden alle anderen Interrupts deaktiviert, bis die Interrupt-Subroutine zurückkehrt. Während eines Interrupts (und zu jeder Zeit) kann mit der Funktion PIN() auf den Wert des Interrupt-Pins zugegriffen werden.

Interrupts können jederzeit auftreten, werden jedoch während INPUT-Anweisungen deaktiviert. Außerdem werden Interrupts während einiger langer hardwarebezogener Vorgänge (z. B. der Funktion TEMPR(), LCD-Zeichenbefehlen und SD-Zugriffsbefehlen) nicht erkannt, obwohl sie erkannt werden, wenn sie nach Beendigung des Vorgangs noch vorhanden sind. Bei Verwendung von Interrupts wird das Hauptprogramm von der Interrupt-Aktivität völlig unberührt, es sei denn, eine vom Hauptprogramm verwendete Variable wird während des Interrupts geändert.

Da Interrupts im Hintergrund ausgeführt werden, können sie schwer zu diagnostizierende Fehler verursachen. Beachten Sie bei der Verwendung von Interrupts die folgenden Faktoren:

- Interrupts werden von MMBasic nur nach Abschluss jedes Befehls überprüft und nicht von der Hardware zwischengespeichert. Das bedeutet, dass ein Interrupt, der nur kurz dauert, übersehen werden kann, insbesondere wenn das Programm Befehle ausführt, deren Ausführung einige Zeit in Anspruch nimmt. Die meisten Befehle werden in weniger als 15 µs ausgeführt, einige Befehle wie die Funktion TEMPR() können jedoch bis zu 200 ms dauern, sodass ein Interrupt innerhalb dieses Zeitfensters auftreten und wieder verschwinden kann und somit nicht erkannt wird.
- Während eines Interrupts werden alle anderen Interrupts blockiert, daher sollten Ihre Interrupts kurz sein und so schnell wie möglich beendet werden. Verwenden Sie beispielsweise niemals PAUSE innerhalb eines Interrupts. Wenn Sie längere Verarbeitungsvorgänge durchführen müssen, sollten Sie einfach ein Flag setzen und den Interrupt sofort beenden, dann kann Ihre Hauptprogrammschleife das Flag erkennen und die erforderlichen Maßnahmen ergreifen.
- Die Unterroutine, die der Interrupt aufruft (und alle anderen von ihr aufgerufenen Unterroutinen oder Funktionen), sollte immer exklusiv für den Interrupt sein. Wenn Sie eine Unterroutine aufrufen müssen, die auch von einem Interrupt verwendet wird, müssen Sie den Interrupt zuerst deaktivieren (Sie können ihn wieder aktivieren, nachdem Sie die Unterroutine beendet haben).
- Denken Sie daran, einen Interrupt zu deaktivieren, wenn Sie ihn nicht mehr benötigen – Hintergrund-Interrupts können seltsame und nicht intuitive Fehler verursachen.

Zusätzlich zu den Interrupts, die durch die Zustandsänderung eines E/A-Pins ausgelöst werden, können Interrupts auch von anderen Bereichen von MMBasic ausgelöst werden, darunter Timer und Kommunikationsports. Die oben genannten Hinweise gelten auch für diese.

Die Liste aller dieser Interrupts (in der Reihenfolge ihrer Priorität von hoch nach niedrig) lautet wie folgt:

1. PID-Regelkreise
2. ON KEY individuell
3. ON KEY allgemein
4. ON PS2
5. PIO RX FIFO
6. PIO TX FIFO
7. PIO RX DMA-Abschluss
8. PIO TX DMA-Abschluss
9. GUI Int Down
10. GUI Int Up
11. Sprite-Kollision

12. WebMite: TCP-Empfang
13. WebMite: MQTT-Abschluss
14. WebMite: UDP-Empfang
15. USB-Gamecontroller/USB- oder PS2-Maus/Wii-Controller
16. ADC-Fertigstellung
17. I2C-Slave-Empfänger
18. I2C-Slave-Tx
19. I2C2-Slave-Empfänger
20. I2C2-Slave-Tx
21. WAV abgeschlossen
22. COM1: Serielle Schnittstelle
23. COM2: Serielle Schnittstelle
24. IR-Empfang
25. Tastatur
26. Interrupt-Befehl/CSub-Interrupt
27. E/A-Pin-Interrupts in der Reihenfolge ihrer Definition
28. Tick-Interrupts (1 bis 4 in dieser Reihenfolge)

Beispiel: Wenn ein ON KEY-Interrupt gleichzeitig mit einem COM1:-Interrupt auftritt, wird zuerst die ON KEY-Interrupt-Subroutine ausgeführt und nach Beendigung der Interrupt-Subroutine wird die COM1:-Interrupt-Subroutine ausgeführt.

Unterstützung für spezielle Geräte-

Um die Interaktion eines Programms mit der Außenwelt zu vereinfachen, enthält die PicoMite-Firmware Treiber für eine Reihe gängiger Peripheriegeräte.

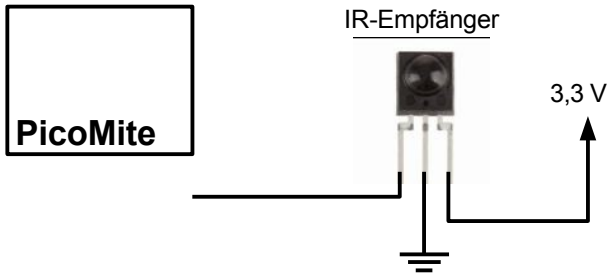
Dies sind:

- Infrarot-Fernbedienungsempfänger und -sender
- Der Temperatursensor DS18B20 und der Temperatur-/Feuchtigkeitssensor DHT22
- LCD-Anzeigemodule
- Numerische Tastaturen
- Batteriegepufferte Uhr
- Ultraschall-Abstandssensor
- WS2812 RGB-LEDs

Infrarot-Fernbedienungs- -Decoder

Mit dem IR-Befehl können Sie ganz einfach eine Fernbedienung zu Ihrem Projekt hinzufügen. Wenn diese Funktion aktiviert ist, läuft sie im Hintergrund und unterbricht das laufende Programm, sobald eine Taste auf der IR-Fernbedienung gedrückt wird.

Sie funktioniert mit allen NEC- oder Sony-kompatiblen Fernbedienungen, einschließlich solcher, die erweiterte -Nachrichten generieren. Die meisten preiswerten programmierbaren Fernbedienungen generieren eines dieser Protokolle, und mit einer solchen Fernbedienung können Sie Ihrem Pico-basierten Projekt eine raffinierte Note verleihen.



Das NEC-Protokoll wird auch von vielen anderen Herstellern wie Apple, Pioneer, Sanyo, Akai und Toshiba verwendet, sodass deren Markenfernbedienungen verwendet werden können.

Um das IR-Signal zu empfangen, benötigen Sie einen IR-Empfänger. NEC-Fernbedienungen verwenden eine 38-kHz-Modulation des IR-Signals. Zu den geeigneten Empfängern, die auf diese Frequenz abgestimmt sind, gehören der Vishay TSOP4838, Jaycar ZD1952 und Altronics Z1611A. Beachten Sie, dass die I/O-Pins des Raspberry Pi Pico nur 3,3 V tolerieren und der Empfänger daher mit maximal 3,3 V betrieben werden muss. Der Raspberry Pi Pico 2 ist anders und kann 5 V aushalten.

Hinweis: Aufgrund eines Hardwarefehlers im RP2350 vor der Version A4 wird dringend empfohlen, bei Verwendung des RP2350 einen 4K7-Pullup-Widerstand an die Datenleitung des IR-Empfängers anzuschließen.

Sony-Fernbedienungen verwenden eine 40-kHz-Modulation, aber Empfänger für diese Frequenz sind schwer zu finden. Im Allgemeinen funktionieren auch 38-kHz-Empfänger, aber die maximale Empfindlichkeit wird mit einem 40-kHz-Empfänger erreicht.

Der IR-Empfänger kann an jeden Pin des Raspberry Pi Pico angeschlossen werden. Dieser Pin muss vom Programm mit dem folgenden Befehl konfiguriert werden:

```
SETPIN n, IR
```

wobei *n* der für diese Funktion zu verwendende I/O-Pin ist. Um den Decoder einzurichten, verwenden Sie den Befehl:

```
IR dev, key, interrupt
```

wobei *dev* eine Variable ist, die mit dem Gerätecode aktualisiert wird, und *key* die Variable ist, die mit dem Schlüsselcode aktualisiert wird. *Interrupt* ist die Interrupt-Subroutine, die aufgerufen wird, wenn ein neuer Tastendruck erkannt wurde. Die IR-Decodierung erfolgt im Hintergrund, und das Programm wird nach diesem Befehl ohne Unterbrechung fortgesetzt.

Dies ist ein Beispiel für die Verwendung des an den GP6-Pin angeschlossenen IR-Decoders:

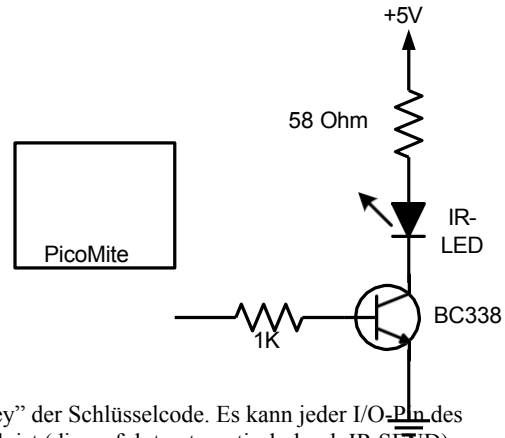
```
SETPIN GP6, IR                          ' Definieren Sie den zu verwendenden
Pin DIM INTEGER DevCode, KeyCode        ' Vom Decoder verwendete Variablen IR
DevCode, KeyCode, IRInt                 ' IR-Decoder starten
DO
   ' < Hauptteil des Programms >
LOOP

SUB IRInt                               ' Eine Tasteneingabe wurde erkannt
```

```
PRINT „Empfangenes Gerät = “ DevCode „Taste = “ KeyCode END SUB
```

IR-Fernbedienungen können viele verschiedene Geräte (VCR, TV usw.) ansprechen, daher würde das Programm normalerweise zuerst den Gerätecode überprüfen, um festzustellen, ob das Signal für das Programm bestimmt war, und wenn ja, dann würde es basierend auf der gedrückten Taste eine Aktion ausführen. Es gibt viele verschiedene Geräte und Tastencodes, daher ist die beste Methode, um festzustellen, welche Codes Ihre Fernbedienung generiert, die Verwendung des obigen Programms, um die Codes zu ermitteln.

Infrarot-Fernbedienung Sender



Mit dem Befehl IR SEND können Sie ein 12-Bit-Infrarot-Fernbedienungssignal von Sony übertragen. Dies ist für die Kommunikation zwischen Raspberry Pi Pico und Raspberry Pi Pico oder Micromite vorgesehen, funktioniert aber auch mit Sony-Geräten, die 12-Bit-Codes verwenden. Beachten Sie, dass bei allen Sony-Produkten die Nachricht dreimal mit einer Verzögerung von 26 ms zwischen den einzelnen Nachrichten gesendet werden muss.

Die Schaltung auf der rechten Seite veranschaulicht, was erforderlich ist. Der Transistor wird zum Ansteuern der Infrarot-LED verwendet, da die Ausgangsleistung des Raspberry Pi Pico begrenzt ist. Diese Schaltung liefert etwa 50 mA an die LED.

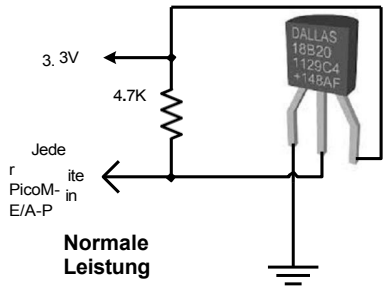
Um ein Signal zu senden, verwenden Sie den Befehl:

```
IR SEND Pin, dev, key
```

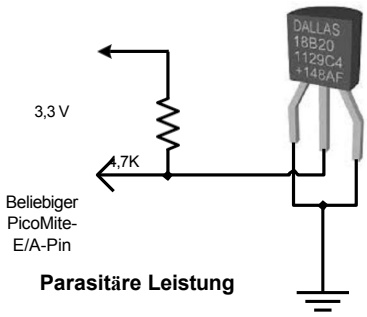
Dabei ist „pin” der verwendete I/O-Pin, „dev” der zu sendende Gerätecode und „key” der Schlüsselcode. Es kann jeder I/O-Pin des Raspberry Pi Pico verwendet werden, ohne dass zuvor eine Einrichtung erforderlich ist (dies erfolgt automatisch durch IR SEND).

Die verwendete Modulationsfrequenz beträgt 38 kHz und entspricht den gängigen IR-Empfängern (siehe vorherige Seite), um eine maximale Empfindlichkeit bei der Kommunikation zwischen zwei Raspberry Pi Picos oder mit einem Micromite zu gewährleisten.

Messung der Temperatur mit dem



Normale Leistung



Parasitäre Leistung

Die Funktion TEMPR() ermittelt die Temperatur anhand eines DS18B20-Temperatursensors. Dieses Gerät ist bei eBay für etwa 5 US-Dollar in verschiedenen Ausführungen erhältlich, darunter auch eine wasserdichte Sonde.

Der DS18B20 kann separat mit einer 3,3-V-Stromversorgung betrieben werden oder, wie rechts dargestellt, mit der parasitären Stromversorgung des Raspberry Pi Pico. Es können mehrere Sensoren verwendet werden, jedoch ist für jeden Sensor ein separater I/O-Pin und ein 4,7-K-Pullup-Widerstand erforderlich.

Um die aktuelle Temperatur zu ermitteln, verwenden Sie einfach die Funktion TEMPR() in einem Ausdruck. Beispiel:

```
PRINT "Temperatur: " TEMPR(Pin)
```

Wobei „Pin“ der I/O-Pin ist, an den der Sensor angeschlossen ist. Sie müssen den I/O-Pin nicht konfigurieren, dies wird von MMBasic übernommen.

Der zurückgegebene Wert wird in Grad Celsius mit einer Auflösung von 0,25 °C angegeben und hat eine Genauigkeit von ±0,5 °C. Wenn während der Messung ein Fehler auftritt, wird der Wert 1000 zurückgegeben.

Die für die gesamte Messung erforderliche Zeit beträgt 200 ms, und das laufende Programm wird für diesen Zeitraum angehalten, während die Messung durchgeführt wird.

Dies bedeutet auch, dass Interrupts für diesen Zeitraum deaktiviert werden. Wenn Sie dies nicht wünschen, können Sie die Umwandlung separat mit dem Befehl TEMPR START auslösen und später die Funktion TEMPR() verwenden, um
Rufen Sie den Temperaturmesswert ab. Die Funktion TEMPR() wartet immer, wenn der Sensor noch mit der Messung beschäftigt ist.

Beispiel:

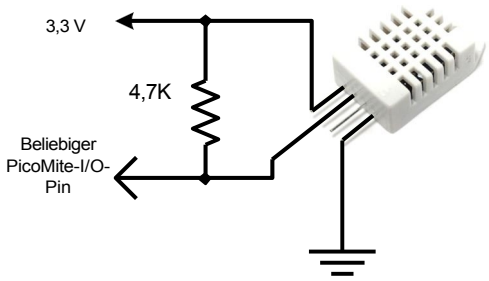
```
TEMPR START GP15
< andere Aufgaben ausführen >
PRINT „Temperatur: “ TEMPR(GP15)
```

Mit dem Befehl TEMPR START können auch die Auflösung der Messung (von standardmäßig 0,25 ºC) und die zugehörige Umrechnungszeit geändert werden.

Messung von Luftfeuchtigkeit und Temperatur mit dem Befehl „ “

Der Befehl HUMID liest die Luftfeuchtigkeit und Temperatur von einem DHT22-Feuchtigkeits-/Temperatursensor aus. Dieses Gerät wird auch als RHT03 oder AM2302 verkauft, aber alle sind kompatibel und können bei eBay für unter 5 US-Dollar erworben werden. Der DHT11-Sensor wird ebenfalls unterstützt.

Der DHT22 muss mit 3,3 V (oder bis zu 5 V mit dem Raspberry Pi Pico 2) betrieben werden und sollte, wie abgebildet, über einen Pullup-Widerstand auf der Datenleitung verfügen. Dies ist für lange Kabelstrecken (bis zu 20 Meter) geeignet, aber für kurze Strecken kann der Widerstand weggelassen werden, da die PicoMite-Firmware auch einen internen schwachen Pullup bereitstellt.



Um die Temperatur oder Luftfeuchtigkeit zu ermitteln, verwenden Sie den Befehl HUMID mit drei Argumenten wie folgt:

```
HUMID pin, tVar, hVar [,DHT11]
```

Dabei ist „pin” der I/O-Pin, an den der Sensor angeschlossen ist. Der I/O-Pin wird automatisch von MMBasic konfiguriert.

„tVar“ ist eine Gleitkommavariable, in der die Temperatur zurückgegeben wird, und „hVar“ ist eine zweite Variable für die Luftfeuchtigkeit. Die Temperatur wird in Grad Celsius mit einer Auflösung von einer Dezimalstelle (z. B. 23,4) zurückgegeben, und die Luftfeuchtigkeit wird als relative Luftfeuchtigkeit in Prozent (z. B. 54,3) zurückgegeben.

Wenn der optionale Parameter „DHT11” auf 1 gesetzt ist, verwendet der Befehl die für dieses Gerät geeigneten Gerätetimings. In diesem Fall werden die Ergebnisse mit einer Auflösung von 1 Grad und 1 % Luftfeuchtigkeit zurückgegeben.

Dieses Beispiel zeigt die Verwendung des DHT22 zur Anzeige der aktuellen Temperatur und Luftfeuchtigkeit im Sekundentakt:

```
DIM FLOAT temp, humidity DO
   HUMID GP15, temp, humidity
   PRINT „Die Temperatur beträgt“ temp „und die Luftfeuchtigkeit beträgt“ humidity
   PAUSE 1000
LOOP
```

Echtzeituhr- -Schnittstelle

Mit dem Befehl RTC GETTIME lässt sich die aktuelle Uhrzeit ganz einfach von einer Echtzeituhr vom Typ PCF8563, DS1307, DS3231 oder DS3232 sowie von kompatiblen Geräten wie dem M41T11 abrufen. Diese integrierten Schaltkreise sind beliebt und kostengünstig, zeigen auch bei ausgeschaltetem Gerät die genaue Uhrzeit an und sind bei eBay für 2 bis 8 US-Dollar erhältlich. Komplette Module einschließlich Batterie sind ebenfalls bei eBay für etwas mehr Geld erhältlich.

Der PCF8563 und der DS1307 halten die Zeit über einen Monat hinweg auf eine oder zwei Minuten genau, während der DS3231 und der DS3232 besonders präzise sind und über ein Jahr hinweg auf eine Minute genau bleiben.

Diese Chips sind I^2C-Geräte und sollten an die I^2C-I/O-Pins des Raspberry Pi Pico angeschlossen werden.

Interne Pullup-Widerstände (100 kΩ) werden an die I^2C-I/O-Pins angelegt, sodass in vielen Fällen keine externen Widerstände erforderlich sind.

Um die RTC zu aktivieren, müssen Sie zunächst die zu verwendenden I^2C-Pins mit dem folgenden Befehl zuweisen:

```
OPTION SYSTEM I2C SDApin, SCLpin
```

Die vom RTC verwendete Zeit muss ebenfalls eingestellt werden. Dies geschieht mit dem Befehl RTC SETTIME, der das folgende Format verwendet:

```
RTC SETTIME Jahr, Monat, Tag, Stunde, Minute, Sekunde
```

Beachten Sie, dass die Uhrzeit im 24-Stunden-Format angegeben werden muss.

Beispielsweise wird mit dem folgenden Befehl die Echtzeituhr auf 16:00 Uhr am 10. November 2025 eingestellt:

```
RTC SETTIME 2025, 11, 10, 16, 0, 0
```

Um die Uhrzeit abzurufen, verwenden Sie den Befehl RTC GETTIME, der die Uhrzeit vom Echtzeituhr-Chip liest und die Uhr im Raspberry Pi Pico einstellt. Normalerweise wird dieser Befehl am Anfang des Programms oder in der Subroutine MM.STARTUP platziert, damit die Uhrzeit beim Einschalten eingestellt wird.

Mit dem Befehl OPTION RTC AUTO ENABLE kann auch eine automatische Aktualisierung der schreibgeschützten Variablen TIME$ und DATE$ aus dem Echtzeituhr-Chip beim Booten und jede Stunde eingestellt werden.

Mess -Entfernung

Mit einem HC-SR04-Ultraschallsensor und der Funktion DISTANCE() können Sie die Entfernung zu einem Ziel messen.

Dieses Gerät ist bei eBay für etwa 4 US-Dollar erhältlich und misst die Entfernung zu einem Ziel von 3 cm bis 3 m. Es sendet einen Ultraschallimpuls aus und misst die Zeit, die das Echo benötigt, um zurückzukommen.

Kompatible Sensoren sind der SRF05, SRF06, Parallax PING und der DYP-ME007 (der wasserdicht ist und sich daher gut zur Überwachung des Füllstands eines Wassertanks eignet). Andere Sensoren, die laut Berichten gut funktionieren, verwenden den CS100-Chip – wie beispielsweise der HC-SR04 und der US-025.

In der PicoMite-Firmware verwenden Sie die DISTANCE-Funktion wie folgt:

```
d = DISTANCE(trig, echo)
```

Der zurückgegebene Wert ist die Entfernung zum Ziel in Zentimetern.

Dabei ist „trig“ der mit dem Eingang „trig“ des Sensors verbundene I/O-Pin und „echo“ der mit dem Ausgang „echo“ des Sensors verbundene Pin. Sie können auch 3-Pin-Geräte verwenden. In diesem Fall wird nur eine Pin-Nummer angegeben.

Beachten Sie, dass die maximale Spannung an allen I/O-Pins des Raspberry Pi Pico 3,3 V beträgt. Für diesen Sensor ist eine Pegelverschiebung erforderlich, da er für seine Echoausgabe 5-V-Pegel verwendet. Der Raspberry Pi Pico 2 verträgt 5 V (bei eingeschaltetem Gerät), sodass in diesem Fall keine Pegelverschiebung erforderlich ist.

LCD- -Anzeige

Der Befehl „LCD“ zeigt Text auf einem Standard-LCD-Modul mit minimalem Programmieraufwand an.

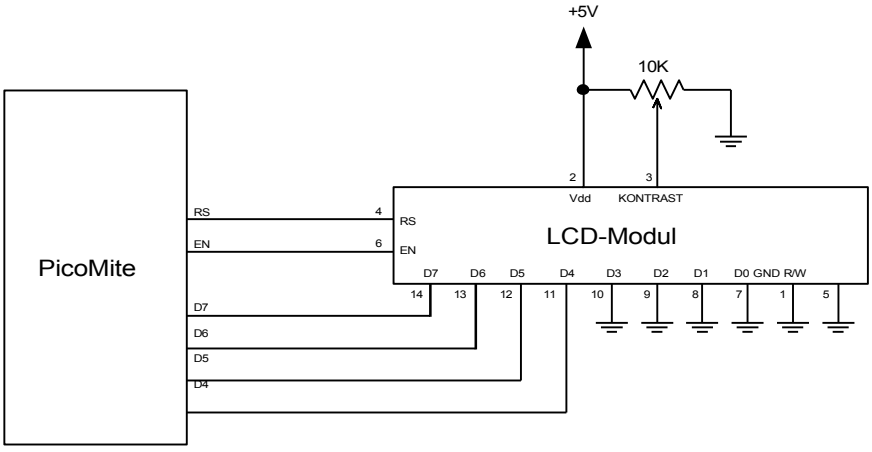
Dieser Befehl funktioniert mit LCD-Modulen, die den Controller-Chip KS0066, HD44780 oder SPLC780 verwenden und über 1, 2 oder 4 Zeilen verfügen. Typische Displays sind das LCD16X2 (futurlec.com), das Z7001 (altronics.com.au) und das QP5512 (jaycar.com.au). eBay ist eine weitere gute Bezugsquelle, wo die Preise zwischen 10 und 50 Dollar liegen können.

Um das Display einzurichten, verwenden Sie den Befehl DEVICE LCD INIT:

```
LCD INIT d4, d5, d6, d7, rs, en
```

d4, d5, d6 und d7 sind die Nummern der E/A-Pins, die mit den Eingängen D4, D5, D6 und D7 Eingänge D0 bis D3 und R/W des Moduls sollten mit Masse verbunden werden). „rs” ist der P[illegible] des Moduls verbunden ist (manchmal auch als CMD oder DAT bezeichnet). „en” ist der Pin, der mit dem Freigabe- oder Chipauswahleingang des Moduls verbunden ist.

Es können alle I/O-Pins des Raspberry Pi Pico verwendet werden, und Sie müssen diese nicht vorher einrichten (der LCD-Befehl übernimmt dies automatisch für Sie). Im Folgenden wird eine typische Konfiguration gezeigt.



Um Zeichen auf dem Modul anzuzeigen, verwenden Sie den LCD-Befehl:

```
LCD Zeile, Pos, Daten$
```

Dabei ist „Zeile” die Zeile auf dem Display (1 bis 4) und „Pos” die Position auf der Zeile, an der die Daten geschrieben werden sollen (die erste Position auf der Zeile ist 1). „Daten$” ist eine Zeichenfolge, die die Daten enthält, die auf das LCD-Display geschrieben werden sollen. Die Zeichen in „Daten$” überschreiben alles, was sich zuvor an dieser Stelle auf dem LCD befand.

Im Folgenden wird eine typische Verwendung gezeigt, bei der d4 bis d7 mit den Pins GP2 bis GP5 verbunden sind, rs mit Pin GP6 und en mit Pin GP7.

```
LCD INIT GP2, GP3, GP4, GP5, GP6, GP7
LCD 1, 2, „Temperatur”
LCD 2, 6, STR$(TEMPR(GP15))          ' DS18B20 mit Pin GP15 verbunden
```

Beachten Sie, dass in diesem Beispiel auch die Funktion TEMPR() zum Abrufen der Temperatur verwendet wird (siehe oben).

Tastatur- -Schnittstelle

Eine Tastatur ist eine einfache, aber effektive Methode zur Eingabe numerischer Daten. Die PicoMite-Firmware unterstützt entweder eine 4x3-Tastatur oder eine 4x4-Tastatur, wobei die Überwachung und Dekodierung der Tastenanschläge im Hintergrund erfolgt. Wenn ein Tastenanschlag erkannt wird, wird ein Interrupt ausgelöst, den das Programm verarbeiten kann.

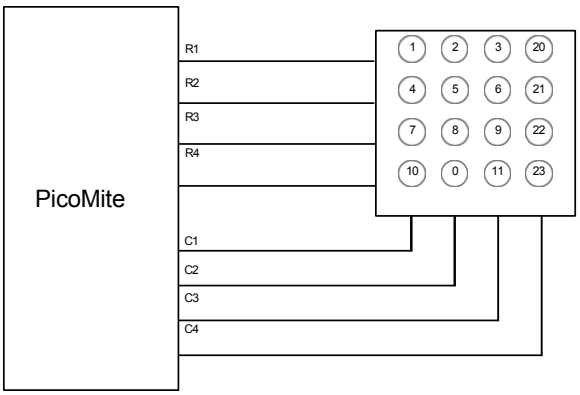
Beispiele für eine 4x3-Tastatur und eine 4x4-Tastatur sind die Altronics S5381 und S5383 (siehe www.altronics.com). Um die

Tastaturfunktion zu aktivieren, verwenden Sie den Befehl:

```
KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3, c4
```

aufgerufen werden, wenn ein neuer Tastendruck erkannt wurde. „r1“, „r2“, „r3“ und „r4“ sind die Pin-Nummern für die vier Reihenanschlüsse an die Tastatur (siehe Abbildung unten) und „c1“, „c2“, „c3“ und „c4“ sind die Spaltenanschlüsse. „c4“ wird nur bei 4x4-Tastaturen verwendet und sollte weggelassen werden, wenn Sie eine 4x3-Tastatur verwenden.

Es können alle I/O-Pins des Raspberry Pi Pico verwendet werden, und Sie müssen diese nicht vorher einrichten, da der Befehl KEYPAD dies automatisch für Sie übernimmt.



Die Erkennung und Dekodierung von Tastendrücken erfolgt im Hintergrund, und das Programm wird nach diesem Befehl ohne Unterbrechung fortgesetzt. Wenn ein Tastendruck erkannt wird, wird der Wert der Variablen var auf die Zahl gesetzt, die die Taste repräsentiert (dies ist die Zahl innerhalb der Kreise in der obigen Abbildung). Dann wird der Interrupt aufgerufen.

Beispiel:

```
Keypad KeyCode, KP_Int,GP2,GP3,GP4,GP5,GP6,GP7,GP8              ' 4x3 keybd
DO
   < Programmkörper > LOOP

SUB KP_Int                                    ' Eine Tasteneingabe wurde erkannt
   PRINT „Tasteneingabe = “ KeyCode
END SUB
```

Siehe auch den erweiterten Befehl KEYPAD in der Befehlsbeschreibung, der eine beliebige Anzahl von Zeilen und Spalten mit benutzerdefinierten Rückgabecodes ermöglicht.

WS2812- -Unterstützung

Die PicoMite-Firmware bietet integrierte Unterstützung für den mehrfarbigen LED-Chip WS2812. Dieser Chip benötigt ein sehr spezifisches Timing, um ordnungsgemäß zu funktionieren. Mit dem Befehl DEVICE WS2812 lassen sich diese Geräte mit minimalem Aufwand einfach steuern.

Dieser Befehl gibt die erforderlichen Signale aus, die zum Ansteuern einer Kette von WS2812-LED-Chips benötigt werden, die an den angegebenen Pin angeschlossen sind, und legt die Farben jeder LED in der Kette fest. Die Syntax des Befehls lautet:

```
WS2812 Typ, Pin, Anzahl%, Farben%[()]
```

Beachten Sie, dass der Pin vor Verwendung dieses Befehls auf einen digitalen Ausgang eingestellt werden muss. Das Array colours%() sollte mindestens so viele Elemente enthalten wie die Anzahl der anzusteuernden LEDs (nbr%). Jedes Element im Array sollte die Farbe im normalen RGB888-Format (0 - HFFFFFF) enthalten. Wenn eine einzelne LED angesteuert werden soll, sollte colours% eine einfache Variable sein.

Es werden bis zu 256 WS2812-Chips in einer Kette unterstützt.

„type" ist ein einzelnes Zeichen, das den Typ des angesteuerten Chips wie folgt angibt: O =

Original WS2812

B = WS2812B S

= SK6812

W = SK6812W (RGBW)

Beispiel:

```
DIM b%(4)=(RGB(rot), Rgb(grün), RGB(blau), RGB(gelb), rgb(cyan)) SETPIN GP5, DOUT
WS2812 O, GP5, 5, b%()
```

gibt die angegebenen Farben an eine Reihe von fünf WS2812-LEDs aus, die über Pin GP5 in Reihe geschaltet sind.

Es ist möglich, dass ein WS2812 mit der 3,3-V-Ausgabe des Raspberry Pi Pico nicht zuverlässig funktioniert. In diesem Fall gibt es eine Reihe von Lösungen:

- Verwenden Sie den WS2812B, der mit einer 3,3-V-Versorgung und -Eingängen funktioniert.
- Verwenden Sie den Raspberry Pi Pico 2, der 5 V (bei eingeschaltetem Gerät) verträgt, sodass in diesem Fall keine Pegelverschiebung erforderlich ist.
- Verwenden Sie einen einzelnen WS2812, der mit 3,3 V betrieben wird, als erste Stufe, um den Eingang der ersten „echten" LED in der Kette zu puffern. Die Mindestversorgung für den WS2812 beträgt 4 V, aber in vielen Fällen funktioniert er auch mit 3,3 V.

OV7670-Kamera- -Modul

Die PicoMite-Firmware unterstützt ein OV7670-Kameramodul. Weitere Informationen finden Sie unter dem Befehl CAMERA.

Display- -Panels

NICHT VERFÜGBAR BEI HDMI- ODER VGA-VERSIONEN

Die PicoMite-Firmware unterstützt viele LCD-Anzeigetafeln, die eine SPI-, I²C- oder parallele Schnittstelle verwenden.

Diese Befehle müssen an der Eingabeaufforderung (nicht in einem Programm) eingegeben werden und führen zu einem Neustart der PicoMite-Firmware. Dies hat den Nebeneffekt, dass die USB-Konsolenschnittstelle getrennt wird und erneut verbunden werden muss.

Beachten Sie, dass die maximale Spannung an allen I/O-Pins des Raspberry Pi Pico 3,3 V beträgt. Für Displays, die 5-V-Pegel für die Signalübertragung verwenden, ist eine Pegelumsetzung erforderlich. Der Raspberry Pi Pico 2 verträgt 5 V (bei eingeschaltetem Gerät), sodass in diesem Fall keine Pegelumsetzung erforderlich ist.

System-SPI-Bus

Der System-SPI-Bus ist ein dedizierter SPI-Kanal, der von vielen LCD-Panels, allen Touch-Controllern und auch zur Kommunikation mit einer SD-Karte verwendet wird. Wenn eines dieser Geräte angeschlossen ist, müssen zuerst die für den System-SPI-Bus verwendeten I/O-Pins definiert werden.

Dies geschieht mit dem folgenden Befehl:

```
OPTION SYSTEM SPI CLK-Pin, MOSI-Pin, MISO-Pin
```

Dieser Befehl muss an der Eingabeaufforderung eingegeben werden und führt dazu, dass die Firmware neu gestartet und die USB-Konsolenschnittstelle getrennt wird, die dann erneut angeschlossen werden muss. Diese Option wird beim Start erneut angewendet, und die Pins werden reserviert und stehen für andere Zwecke nicht zur Verfügung.

Ein typisches Beispiel ist:

```
OPTIONSSYSTEM SPI GP18, GP19, GP16
```

Beachten Sie, dass die Geschwindigkeit beim Zeichnen auf SPI-basierten Displays und beim Zugriff auf SD-Karten nicht von der CPU-Geschwindigkeit beeinflusst wird.

SPI-basierte Display- -Panels

Diese Display-Panels werden mit den folgenden Befehlen konfiguriert. Alle erfordern, dass der System-SPI-Bus (siehe oben) zuvor definiert wurde.

In allen Befehlen lauten die Parameter:

`ODER`	Dies ist die Ausrichtung des Displays und kann LANDSCAPE, PORTRAIT, RLANDSCAPE oder RPORTRAIT sein. Diese können mit L, P, RL oder RP abgekürzt werden. Das Präfix R steht für die umgekehrte oder „auf den Kopf gestellte" Ausrichtung.
`DC`	Anzeige Daten-/Befehlssteuerungs-Pin.
`RESET`	Anzeige-Reset-Pin (wenn auf Low gezogen).
`CS`	Anzeige-Chipauswahl-Pin (aktiv niedrig).
`BL`	Optionaler Pin zur Steuerung der Helligkeit der Hintergrundbeleuchtung mittels Pulsweitenmodulation (PWM).
`INVERT`	Diese Option bewirkt eine Invertierung der Farben, um ein nicht standardmäßiges Panel auszugleichen.

```
OPTION LCDPANEL ILI9341, OR, DC, RESET, CS [, BL] [,INVERT]
```

Initialisiert ein TFT-Display unter Verwendung des ILI9341-Controllers. Dieser unterstützt eine Auflösung von 320 × 240. Displays, die diesen Controller verwenden, sind in der Lage, transparenten Text anzuzeigen und arbeiten mit den Befehlen BLIT und BLIT READ.

```
OPTION LCDPANEL ILI9163, OR, DC, RESET, CS [, BL] [,INVERT]
```

Initialisiert ein TFT-Display mit dem ILI9163-Controller. Dieser unterstützt eine Auflösung von 128 × 128.

```
OPTION LCDPANEL ILI9481, OR, DC, RESET, CS [, BL] [,INVERT]
```

Initialisiert ein TFT-Display unter Verwendung des ILI9481-Controllers. Dieser unterstützt eine Auflösung von 480 × 320.

```
OPTION LCDPANEL ILI9481IPS, OR, DC, RESET,CS [, BL][,INVERT]
```

Initialisiert ein IPS-Display mit dem ILI9481-Controller. Dieser unterstützt eine Auflösung von 480 × 320.

```
OPTION LCDPANEL ILI9488, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein TFT-Display unter Verwendung des ILI9488-Controllers. Dieser unterstützt eine Auflösung von 480 × 320. Beachten Sie, dass dieser Controller ein Problem mit dem LCD_SDO-Pin (MISO) hat, der den Touch-Controller stört. Damit dieses Display funktioniert, darf der LCD_SDO-Pin nicht direkt mit dem System-SPI-MISO verbunden sein.

```
OPTION LCDPANEL ILI9488P, OR, DC, RESET, CS [,BL] [,INVERT]
```
Initialisiert ein TFT-Display unter Verwendung des ILI9488-Controllers. Dieser unterstützt eine Auflösung von 320 × 320. Beachten Sie, dass dieser Controller ein Problem mit dem LCD_SDO-Pin (MISO) hat, der den Touch-Controller stört. Damit dieses Display funktioniert, darf der LCD_SDO-Pin nicht direkt mit dem System-SPI-MISO verbunden sein. Diese Konfiguration unterstützt den PicoCalc mit dem Display im Hochformat.

```
OPTION LCDPANEL ILI9488W, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein TFT-Display unter Verwendung des ILI9488-Controllers. Dies unterstützt das 3,5-Zoll-Display von Waveshare, wie es auf deren Pico Eval-Board verwendet wird, sowie den normalen 3,5-Zoll-Display-Adapter.

```
OPTION LCDPANEL N5110, OR, DC, RESET, CS [,contrast] [,INVERT]
```
Initialisiert ein LCD-Display mit dem Nokia 5110-Controller. Dieser unterstützt eine Auflösung von 84 * 48. Ein zusätzlicher Parameter „contrast" kann angegeben werden, um den Kontrast des Displays zu steuern. Probieren Sie Kontrastwerte zwischen &HA8 und &HD0 aus, um das Display anzupassen. Der Standardwert bei Auslassung ist &HB1.

```
OPTION LCDPANEL SSD1306SPI, OR, DC, RESET, CS [,offset] [,INVERT]
```
Initialisiert ein OLED-Display unter Verwendung des SSD1306-Controllers mit einer SPI-Schnittstelle. Dies unterstützt eine Auflösung von 128 * 64. Ein zusätzlicher Parameter „offset" kann angegeben werden, um die Position des Displays zu steuern. 0,96-Zoll-Displays benötigen in der Regel einen Wert von 0. 1,3-Zoll-Displays benötigen in der Regel einen Wert von 2. Der Standardwert bei Auslassung ist 0.

```
OPTION LCDPANEL SSD1331, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein Farb-OLED-Display mit dem SSD1331-Controller. Dieser unterstützt eine Auflösung von 96 * 64.

```
OPTION LCDPANEL ST7735, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein TFT-Display unter Verwendung des ST7735-Controllers. Dieser unterstützt eine Auflösung von 160 * 128.

```
OPTION LCDPANEL ST7735S, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein IPS-Display mit dem ST7735S-Controller. Dieser unterstützt eine Auflösung von 160 × 80.

```
OPTION LCDPANEL ST7735S_W, OR, DC, RESET, CS [, BL][,INVERT]
```
Initialisiert ein Waveshare 128x128 ST7735S-Display. Dieses unterstützt eine Auflösung von 128 * 128.

```
OPTION LCDPANEL ST7789, OR, DC, RESET, CS [, BL] [,INVERT]
```
Initialisiert ein IPS-Display unter Verwendung des 7789-Controllers. Dieser unterstützt eine Auflösung von 240 * 240.
HINWEIS: Display-Karten ohne CS-Pin werden derzeit in der PicoMite-Firmware nicht unterstützt, sofern sie nicht modifiziert wurden.

```
OPTION LCDPANEL ST7789_135, OR, DC, RESET,CS [, BL][,INVERT]
```
Initialisiert ein IPS-Display mit dem 7789-Controller. Dieser unterstützt eine Auflösung von 240 * 135.
HINWEIS: Display-Boards ohne CS-Pin werden derzeit in der PicoMite-Firmware nicht unterstützt, sofern sie nicht modifiziert wurden.

```
OPTION LCDPANEL ST7789_320, OR, DC, RESET,CS [, BL][,INVERT]
```
Initialisiert ein IPS-Display unter Verwendung des 7789-Controllers. Dieser Typ unterstützt das Display mit einer Auflösung von 320 * 240 von Waveshare (https://www.waveshare.com/wiki/Pico-ResTouch-LCD-2.8).
Diese sind in der Lage, transparenten Text anzuzeigen und funktionieren mit den Befehlen BLIT und BLIT READ.
HINWEIS: Display-Boards ohne CS-Pin werden derzeit in der PicoMite-Firmware nicht unterstützt, sofern sie nicht modifiziert wurden.

```
OPTION LCDPANEL ST7796S, OR, DC, RESET,CS [,BL][,INVERT]
```

Initialisiert ein IPS-Display mit dem ST7796S-Controller. Dieser unterstützt eine Auflösung von 480 * 320.
HINWEIS: Damit transparente Tests und Blits ordnungsgemäß funktionieren, sollte die Diode D1 auf der Rückseite des Displays überbrückt werden. Es wird empfohlen, auch J1 zu überbrücken, um mit 3,3 V zu arbeiten.

```
OPTION LCDPANEL ST7796SP, OR, DC, RESET,CS [,BL][,INVERT]
```

Initialisiert ein TFT-Display unter Verwendung des ST7796S-Controllers. Dieser unterstützt eine Auflösung von 320 × 320. Diese Konfiguration unterstützt den PicoCalc mit dem Display im Hochformat.

```
OPTION LCDPANEL GC9A01, OR, DC, RESET, CS [, BL] [,INVERT]
```

Initialisiert ein IPS-Display mit dem GC9A01-Controller. Dieser unterstützt eine Auflösung von 240 × 240.

```
OPTION LCDPANEL ST7920, OR, DC, RESET
```

Initialisiert ein LCD-Display mit dem ST7920-Controller. Dies unterstützt eine Auflösung von 128 * 64. Beachten Sie, dass dieses Display keine Chipauswahl unterstützt, sodass der SPI-Bus bei Verwendung dieses Displays nicht gemeinsam genutzt werden kann.

I²C-basierte LCD- -Panels

Alle I² C-basierten Display-Controller verwenden die System-I² C-Pins gemäß der Pinbelegung für das jeweilige Gerät. Andere I² C-Geräte können den Bus gemeinsam nutzen, sofern ihre Adressen eindeutig sind.

Verwenden Sie zum Einrichten des System-I² C-Busses den folgenden Befehl:

```
OPTION SYSTEM I2C sdapin, sclpin
```

Wenn ein I² C-Display konfiguriert ist, muss der I² C-Port für ein zusätzliches Gerät nicht „geöffnet" werden (I2C OPEN), I2C CLOSE ist gesperrt und alle I² C-Geräte müssen für den 100-kHz-Betrieb geeignet sein. Die Geschwindigkeit des I² C-Busses wird durch Änderungen der CPU-Taktrate nicht beeinflusst.

Diese Panels werden mit den folgenden Befehlen konfiguriert. In allen Befehlen ist der Parameter OR die Ausrichtung des Displays und kann LANDSCAPE, PORTRAIT, RLANDSCAPE oder RPORTRAIT sein. Diese können mit L, P, RL oder RP abgekürzt werden. Das Präfix R steht für die umgekehrte oder „auf den Kopf gestellte" Ausrichtung.

```
OPTION LCDPANEL SSD1306I2C, OR [,offset]
```

Initialisiert ein OLED-Display unter Verwendung des SSD1306-Controllers mit einer I²C-Schnittstelle. Dies unterstützt eine Auflösung von 128 * 64. Ein zusätzlicher Parameter-Offset kann angegeben werden, um die Position des Displays zu steuern. 0,96-Zoll-Displays benötigen in der Regel einen Wert von 0. 1,3-Zoll-Displays benötigen in der Regel einen Wert von 2. Der Standardwert bei Auslassung ist 0.

Hinweis: Viele günstige I²C-Versionen von SSD1306-Displays implementieren I²C aufgrund eines Verdrahtungsfehlers nicht ordnungsgemäß. Dies scheint insbesondere bei 1,3"-Varianten der Fall zu sein.

Der SSD1306I2C-Treiber funktioniert auch mit SSD1315- und SH1106-Controllern.

```
OPTION LCDPANEL SSD1306I2C32 ODER
```

Initialisiert ein OLED-Display unter Verwendung des SSD1306-Controllers mit einer I²C-Schnittstelle. Unterstützt eine Auflösung von 128 * 32.

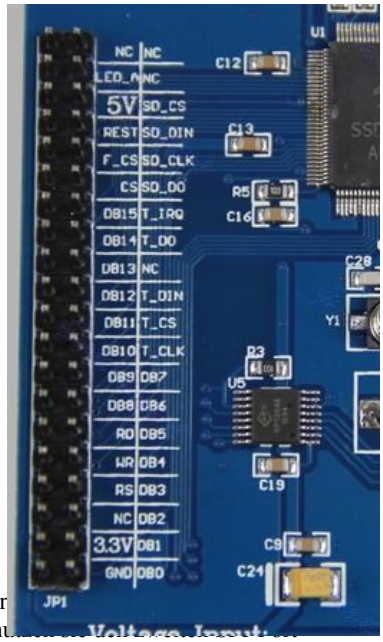
8-Bit-Parallel-LCD- -Panels

Zusätzlich zu den SPI- und I²C-basierten Controllern unterstützt die PicoMite-Firmware LCD-Displays mit dem SSD1963-Controller (wi abgebildet) und dem ILI9341-Controller.

Diese verwenden eine parallele Schnittstelle, sind in Größen von 2,8" bis 9" erhältlich und haben bessere Spezifikationen als die kleineren Displays. Alle diese Displays verfügen über einen SD-Kartensteckplatz, der von MMBasic vollständig unterstützt wird. Auf eBay finden Sie geeignete Displays, indem Sie nach dem Namen des Controllers suchen (z. B. SSD1963).

Da sie eine parallele Schnittstelle verwenden, können Daten viel schneller übertragen werden als über eine SPI-Schnittstelle, was zu einer sehr schnellen Bildschirmaktualisierung führt.

Insbesondere die SSD1963-Displays sind außerdem viel größer, haben mehr Pixel und sind heller. MMBasic kann einige von ihnen mit 24-Bit-True-Color für eine Vollfarbwiedergabe (16 Millionen Farben) ansteuern.



Die Eigenschaften dieser Displays sind:

- Ein 2,8-, 3,2-, 4,3-, 5-, 7-, 8- oder 9-Zoll-Display
- Auflösung von 320 x 240, 480 x 272 Pixel (4,3-Zoll-Version) oder 800 x 480 Pixel (5-, 7-, 8- oder 9-Zoll-Versionen).
- Ein SSD1963-Display-Controller oder ILI9341-Display-Controller mit paralleler Schnittstelle (8080-Format)
- Ein Touch-Controller (SPI-Schnittstelle).
- Ein SD-Kartensteckplatz in voller Größe.

Es gibt eine Reihe verschiedener Designs, die den SSD1963-Controller verwenden, aber glücklicherweise haben sich die meisten Anbieter auf einen einzigen Stecker standardisiert, wie rechts abgebildet.

Es wird dringend empfohlen, dass jedes gekaufte Display über einen passenden Anschluss verfügt – dies gibt Ihnen die Gewissheit, dass der Hersteller den Standard befolgt hat, für den die PicoMite-Firmware entwickelt wurde.

Anschluss eines 8-Bit-Parallel-LCD- -Panels

Der Controller verwendet eine parallele Schnittstelle, während der Touch-Controller und die SD-Kar
verwenden. Die Touch- und SD-Kartenfunktionen sind optional, aber wenn sie verwendet werden, nu
(SPI2).

Die folgende Tabelle listet die erforderlichen Verbindungen zwischen der Anzeigetafel und dem Raspberry Pi Pico auf, um die 8-Bit-Parallelschnittstelle und die LCD-Anzeige zu unterstützen. Die Touch-Controller- und SD-Karten-Schnittstellen sind unten aufgeführt.

8-Bit-Parallelanzeige	Beschreibung	Raspberry Pi Pico
DB0	Paralleler Datenbus Bit 0	Pin 1/GP0
DB1	Paralleler Datenbus Bit 1	Pin 2/GP1
DB2	Paralleler Datenbus Bit 2	Pin 4/GP2
DB3	Paralleler Datenbus Bit 3	Pin 5/GP3
DB4	Parallel-Datenbus Bit 4	Pin 6/GP4
DB5	Parallel-Datenbus Bit 5	Pin 7/GP5
DB6	Paralleler Datenbus Bit 6	Pin 9/GP6
DB7	Paralleler Datenbus Bit 7	Pin 10/GP7
CS	Chipauswahl (aktiv niedrig)	Masse (d. h. immer ausgewählt)
WR	Schreiben (aktiv niedrig)	Pin 19/GP14
RD	Lesen (aktiv niedrig)	Pin 20/GP15
DC	Befehl/Daten	Pin 17/GP13
RESET	SSD1963 zurücksetzen	Pin 21/GP16
LED_A	Hintergrundbeleuchtungssteuerung für ein unverändertes Display-Panel	Konfigurierbar, siehe OPTION LCDPANEL
5	5-V-Stromversorgung für die Hintergrundbeleuchtung einiger Displays (die meisten Displays verwenden hierfür eine 3,3-V-Stromversorgung).	
3,3 V	Stromversorgung.	
GND	Masse	

Die Pins DC, WR, RD und RESET können mit dem optionalen Parameter *DCpin* als 4er-Block anderen Pins zugewiesen werden. Beim RP2350B kann der optionale Parameter *DB0pin* verwendet werden, um den Start-Pin für die 8 oder 16 aufeinanderfolgenden Datenpins festzulegen, die vom Display verwendet werden.

Die folgende Tabelle listet die Anschlüsse auf, die zur Unterstützung der Touch-Controller-Schnittstelle erforderlich sind:

8-Bit-Parallelanzeige	Beschreibung	Raspberry Pi Pico
T_CS	Touch-Chip-Auswahl	Empfohlener Pin 24/GP18
T_IRQ	Touch-Interrupt	Empfohlener Pin 25/GP19
T_DIN	Touch-Daten-Eingang (MOSI)	Empfohlener Pin 15/GP11
T_CLK	Touch-SPI-Takt	Empfohlener Pin 14/GP10
T_DO	Touch-Datenausgang (MISO)	Empfohlener Pin 16/GP12

Die folgende Tabelle listet die für die Unterstützung des SD-Kartenanschlusses erforderlichen Anschlüsse auf:

8-Bit-Parallelanzeige	Beschreibung	Raspberry Pi Pico
SD_CS	SD-Karten-Chipauswahl	Empfohlener Pin 29/GP22
SD_DIN	SD-Kartendaten-Eingang (MOSI)	Empfohlener Pin 15/GP11
SD_CLK	SD-Karte SPI-Takt	Empfohlener Pin 14/GP10
SD_DO	SD-Karte Datenausgang (MISO)	Empfohlener Pin 16/GP12

Wenn eine Verbindung als „empfohlen" aufgeführt ist, handelt es sich lediglich um einen Vorschlag. Je nach Hardwarekonfiguration können auch andere Pins verwendet werden. Unabhängig davon sollte der spezifische Pin im entsprechenden OPTION-Befehl angegeben werden (siehe unten).

Im Allgemeinen verfügen Displays mit einer Größe von 7 Zoll und mehr über einen separaten Pin am Stecker (mit der Kennzeichnung „5V"), über den die Hintergrundbeleuchtung mit 5 V versorgt wird. Ist dieser Pin nicht vorhanden, wird die Hintergrundbeleuchtung über den 3,3-V-Pin mit Strom versorgt. Beachten Sie, dass der Stromverbrauch der Hintergrundbeleuchtung beträchtlich sein kann. Ein 7-Zoll-Display verbraucht beispielsweise in der Regel 330 mA über den 5-V-Pin.

Wenn der 3,3-V-Ausgang des Pico zur Stromversorgung eines Panels und dessen Hintergrundbeleuchtung verwendet wird, kann es leicht vorkommen, dass mehr Strom benötigt wird, als der Pico liefern kann. Zu den Symptomen einer unzureichenden Stromversorgung können Fehler bei der TOUCH-Kalibrierung oder beim SD-Zugriff gehören. In diesem Fall sollte eine externe 3,3-V-Stromversorgung verwendet werden.

Der von der Hintergrundbeleuchtung aufgenommene Strom kann auch einen Spannungsabfall am Massepin des LCD-Bildschirms verursachen, was wiederum zu einer Verschiebung der Logikpegel führen kann, die vom Bildschirmcontroller erkannt werden, was wiederum zu verfälschten Farben oder Texten führt. Eine einfache Möglichkeit, diesen Effekt zu diagnostizieren, besteht darin, die CPU-Geschwindigkeit auf (beispielsweise) 48 MHz zu reduzieren. Wenn dadurch das Problem behoben wird, ist dies ein starker Hinweis darauf, dass dies die Ursache ist. Eine mögliche Abhilfe besteht darin, die Strom- und Erdungskabel direkt an die Leiterplatte des LCD-Bildschirms anzulöten.

Bei Display-Panels, die sich den SPI-Port mit mehreren Geräten teilen (SD-Karte, Touchscreen usw.), ist Vorsicht geboten. In diesem Fall müssen alle Chip-Select-Signale in MMBasic konfiguriert oder durch eine permanente Verbindung mit 3,3 V deaktiviert werden. Andernfalls schwankt der Pin, was dazu führt, dass der falsche Controller auf Befehle auf dem SPI-Bus reagiert.

In der PicoMite-Firmware kann jeder SPI-Kanal für die Kommunikation mit dem Touch-Controller und der SD-Kartenschnittstelle verwendet werden, wie durch die Einstellung OPTION SYSTEM SPI definiert. Wenn diese Einstellung aktiviert ist, steht dieser SPI-Kanal für BASIC-Programme nicht zur Verfügung (diese können den anderen SPI-Kanal verwenden).

Konfigurieren eines 8-Bit-Parallel-LCD- -Panel

Um das Display zu verwenden, muss MMBasic mit dem Befehl OPTION LCDPANEL konfiguriert werden, der normalerweise an der Befehlszeile eingegeben wird. Bei jedem Neustart der PicoMite-Firmware initialisiert MMBasic das Display automatisch.

Die Syntax lautet:

OPTION LCDPANEL controller, orientation [,backlightpin] [,DCpin] [,NORESET] [,INVERT] [,DB0pin] Dabei kann „controller" entweder sein:

- SSD1963_4 Für ein 4,3-Zoll-Display
- SSD1963_5 Für ein 5-Zoll-Display
- SSD1963_5A Für eine alternative Version des 5-Zoll-Displays, wenn SSD1963_5 nicht funktioniert

- SSD1963_7 Für ein 7-Zoll-Display
- SSD1963_7A Für eine alternative Version des 7-Zoll-Displays, falls SSD1963_7 nicht funktioniert.
- SSD1963_8 Für 8-Zoll- oder 9-Zoll-Displays.
- ILI9341_8 Für ein 2,8-Zoll- oder 3,2-Zoll-Display

Die Ausrichtung kann LANDSCAPE, PORTRAIT, RLANDSCAPE oder RPORTRAIT sein. Diese können mit L, P, RL oder RP abgekürzt werden. Das Präfix R steht für die umgekehrte oder „auf den Kopf gestellte" Ausrichtung.

„DCpin" ist optional und bezeichnet den Daten-/Befehlspin (früher als RS-Pin bezeichnet). Wird dieser Parameter weggelassen, erfolgt die Pinbelegung wie oben in der Tabelle angegeben. Wird er angegeben, werden die Pins DC, WR, RD und RESET sequenziell vom DC-Pin aus zugewiesen.

Der optionale Parameter „NORESET" kann zum Speichern eines Pins verwendet werden. In diesem Fall sollte der Reset-Pin auf High gesetzt werden.

Der optionale Parameter „INVERT" gibt an, dass die Farbpalette invertiert werden soll (erforderlich für bestimmte EsatRisng-Displays).

Auf dem RP2350B-Prozessor kann der optionale Parameter „DB0pin" verwendet werden, um den Start-Pin für die 8 oder 16 aufeinanderfolgenden Daten-Pins festzulegen, die vom Display verwendet werden.

Dieser Befehl muss nur einmal ausgeführt werden. Von da an initialisiert MMBasic das Display automatisch beim Start oder beim Zurücksetzen. Unter bestimmten Umständen kann es erforderlich sein, die Stromversorgung des LCD-Panels zu unterbrechen, während die PicoMite-Firmware ausgeführt wird (z. B. um Batteriestrom zu sparen). In diesem Fall kann der GUI-Befehl RESET LCDPANEL verwendet werden, um das Display neu zu initialisieren.

Wenn das LCD-Panel nicht mehr benötigt wird, kann der Befehl OPTION LCDPANEL DISABLE verwendet werden, wodurch die I/O-Pins wieder für allgemeine Zwecke verfügbar werden.

Um die Konfiguration zu überprüfen, können Sie den Befehl OPTION LIST verwenden, um alle eingestellten Optionen einschließlich der Konfiguration des LCD-Panels aufzulisten.

Um das Display zu testen, können Sie den Befehl GUI TEST LCDPANEL eingeben. Sie sollten eine animierte Anzeige mit schnell übereinander gezeichneten Farbkreisen sehen. Drücken Sie die Leertaste auf der Tastatur der Konsole, um den Test zu beenden.

8- und 9-Zoll- -Displays

Die Controller-Konfiguration SSD1963_8 wurde nur mit den 8- und 9-Zoll-Displays von EastRising (erhältlich unter www.buydisplay.com) getestet. Diese müssen als TFT-LCD-Panel mit 8080-Schnittstelle, 800 x 480 Pixel LCD, SSD1963-Display-Controller und XPT2046-Touch-Controller erworben werden. Beachten Sie, dass die EastRising-Panels einen
Nicht standardmäßige Pinbelegung der Schnittstellenanschlüsse, daher müssen Sie beim Anschluss an den Raspberry Pi Pico die Datenblätter zu Rate ziehen. Ein geeigneter Adapter zur Umwandlung in den standardmäßigen 40-poligen Anschluss kann unter folgender Adresse erworben werden: https://www.rictech.nz/micromite-products

16- Bit-Parallel-LCD- -Panels

SSD1963-Panels können auch für den 16-Bit-Parallelbetrieb aktiviert werden. In diesem Fall werden standardmäßig die Pins GP0-GP15 für die Datenverbindungen und die Pins GP16 bis GP19 für die Steuersignale DC, WR, RD und RESET verwendet.

Bei Systemen, die den RP2350B verwenden, können nur die verwendeten Datenpins mithilfe des optionalen Parameters DB0pin im Konfigurationsbefehl ausgewählt werden. Um den 16-Bit-Betrieb zu aktivieren, fügen Sie „_16" an den Controller an. Beispiel: SSD1963_4_16. Die Firmware unterstützt auch ILI9341-, ILI9486-, NT35510- und OTM8009A-Panels im 16-Bit-Modus unter Verwendung der Controllertypen ILI9341_16, ILI9486_16 und IPS_4_16 (unterstützt sowohl NT35510 als auch OTM8009A).

Gültige 16-Bit-„Controller" können sein:

- SSD1963_4_16 Für ein 4,3-Zoll-Display
- SSD1963_5_16 Für ein 5-Zoll-Display
- SSD1963_5A_16 Als Alternative zum 5-Zoll-Display, falls SSD1963_5 nicht funktioniert
- SSD1963_7_16 Für ein 7-Zoll-Display
- SSD1963_7A_16 Für eine alternative Version des 7-Zoll-Displays, wenn SSD1963_7 nicht funktioniert.
- SSD1963_8_16 Für 8-Zoll- oder 9-Zoll-Displays.
- ILI9341_16 Für ein 2,8-Zoll- oder 3,2-Zoll-Display
- ILI9486_16 Unterstützt auch NXP R61529
- IPS_4_16

RP2350 Erweiterte Display- -Unterstützung

Der RP2350 verfügt über 320 KB Speicher, der dem MMBasic-Programmierer zur Verfügung steht. Dadurch können verschiedene zusätzliche Funktionen mithilfe von gepufferten Displaytreibern implementiert werden.

Die PicoMite-Firmware für den RP2350 unterstützt die folgenden neuen Displaytreiber: SPI

ILI9341BUFF: 320 x 240
ST7796SPBUFF: 320 x 320
ST7796SBUFF: 480 x 320
ILI9488PBUFF: 320 x 320
ILI9488BUFF: 480 x 320
ILI9488WBUFF: 480 x 320 'Waveshare Pico-Restouch-lcd-3.5
ST7789_320BUFF: 320 x 240 'Waveshare Pico-Restouch-LCD-2,8 Parallel
SSD1963_5_BUFF: 400 x 240 (8-Bit-Datenbus)
SSD 1963_7_ BUFF: 400 x 240 (8-Bit-Datenbus) SSD 1963_5_ 12BUFF: 400 x 240
SSD 1963_7_ 12BUFF: 400x240 SSD 1963_5_16 BUFF: 400x240 SSD 1963_7_16 BUFF: 400x240

In allen Fällen wird durch die Aktivierung eines dieser Treiber ein RGB332-Framebuffer aus dem allgemeinen Speicher entnommen, sodass im schlimmsten Fall eines ILI9488BUFF der allgemeine Speicher um 150 KB (von 320 KB) reduziert wird.

Um einen der SPI-Treiber zu verwenden, müssen Sie zunächst die zu verwendenden SPI-Pins zuweisen:

```
OPTION LCD SPI clkpin, mosipin, misopin
```

Diese Pins sind für das Display reserviert und müssen von den Pins getrennt sein, die für das SYSTEM SPI verwendet werden, das möglicherweise noch für Touch und SD-Karte benötigt wird (Hinweis: Bei den Waveshare-Boards führt die Verwendung dieses Treibers zu Konflikten mit der Verwendung von Touch und SD-Karte, da diese sich den SPI-Kanal teilen). Der Grund für die reservierten Pins ist, dass die Bildschirmaktualisierungen alle auf dem zweiten Prozessor stattfinden und dieser für maximale Leistung den SPI-Kanal nicht teilen kann.

Das Format für den Befehl zum Aktivieren eines SPI-gepufferten Treibers ist dasselbe wie bei jedem SPI-Treiber

```
OPTION LCDPANEL controller, orientation, DCpin, RESETpin, CSpin [,BLpin] [,INVERT]
```

Die SSD1963-Treiber verwenden 8, 12 oder 16 Datenpins, wie im Treibernamen angegeben. In allen Fällen beansprucht die Aktivierung eines dieser Treiber einen 400x240 RGB332-Framebuffer aus dem Heap-Speicher. Bei Verwendung dieser Treiber können Sie mit dem Befehl MODE 800/400 zwischen dem gepufferten Treiber und einem herkömmlichen Treiber umschalten, der ohne Neustart zwischen den Modi 800x480 und 400x240 umschaltet. Mit dem Befehl OPTION LCD320 ON/OFF können Sie ein 320x240-Bild auf dem 400x240-Display zentrieren.

Das Format für den Befehl zum Aktivieren eines SDD1963-Puffertreibers ist das gleiche wie für jeden SSD1963-Treiber

```
OPTION LCDPANEL Controller, Ausrichtung [,Backlightpin] [,DCpin] [,NORESET] [,INVERT]
[,DB0pin]
```

Der Vorteil der gepufferten Treiber besteht darin, dass die Grafikbefehle in MMBasic lediglich in einen Speicherpuffer geschrieben werden und die Aktualisierung der physischen Anzeige dann im Hintergrund auf dem zweiten Prozessor erfolgt. Dadurch kann das Basic-Programm die Verarbeitung fortsetzen, ohne auf die Anzeige warten zu müssen. Natürlich ändert die Verwendung eines gepufferten Treibers nichts Wesentliches am Gesamtdurchsatz zur physischen Anzeige. Kontinuierliche Schreibvorgänge müssen also wie bei einem Standardtreiber auf die Anzeige warten. Der große Vorteil kommt zum Tragen, wenn ein Programm eine rechenintensive Schleife hat, bei der Bildschirmschreibvorgänge die Verarbeitung verzögern. In diesem Fall wird der Overhead der Verarbeitungsschleife drastisch reduziert, solange die Gesamtbildschirmaktualisierungsrate überschaubar ist.

VGA222- -Treiber

Der PICOMITE RP2350 und der PICOMITEUSB RP2350 unterstützen auch einen erweiterten VGA-Treiber, der im RGB222-Modus mit 64 Farben arbeitet. Dieser wird mit dem folgenden Befehl konfiguriert:

```
OPTION LCDPANEL driver, hsyncpin, bluelpin
```

Gültige Treiber sind: VGA222_640, VGA222_320, VGA_720, VGA_360 mit den VGA222-Auflösungen 640x480, 320x240, 720x400 und 360x200.

Der Befehl OPTION überprüft die Verfügbarkeit von hysyncpin und hysyncpin+1 (vsync) bluelpin - bluelpin+5 (blue_l, blue_h, green_l, green_h, red_l, red_h) und setzt, sofern diese alle frei sind, die Ausführung der VGA222-Ausgabe auf den angegebenen Pins.

Um die Datenpins mit dem VGA-Anschluss zu verbinden, verwenden Sie 390- und 820-Ohm-Widerstände von den H- und L-Ausgängen gemäß dem grünen Kanal im Handbuch (Seite 30). RGB222 bietet eine viel schönere Farbwiedergabe als RGB121, das in der Standard-VGA-Version verfügbar ist. Beachten Sie, dass es keine Modi gibt. Die Auflösungen 640x480, 320x240, 720x400 und 360x200 sind unterschiedliche Treiber und benötigen unterschiedliche Speicherkapazitäten (deutlich mehr im Fall von 640x480 und 720x400). Zur Erzeugung des VGA-Ausgangs verwendet die Firmware PIO0 und PIO1, sodass PIO2 verfügbar bleibt verfügbar bleibt (es sei denn, Sie verwenden auch den I2S-Audiotreiber).

Steuerung der Hintergrundbeleuchtung

Für ILI9163, ILI9341, ST7735, ST7735S, SSD1331, ST7789, ILI9481, ILI9488, ILI9488W, ST7789_135
ILI9341_8 und ST7789_320 zeigen einen optionalen Parameter „, backlight“ an, der am Ende der Konfigurationsparameter hinzugefügt werden kann und einen Pin zur Steuerung der Helligkeit der Hintergrundbeleuchtung (LED_A) angibt. Dadurch wird ein PWM-Ausgang an diesem Pin mit einer Frequenz von 50 kHz und einem anfänglichen Tastverhältnis von 99 % eingerichtet.

Sie können dann mit dem Befehl BACKLIGHT die Helligkeit zwischen 0 und 100 % einstellen. Der PWM-Kanal ist für die normale PWM-Verwendung gesperrt und darf nicht mit dem PWM-Kanal in Konflikt stehen, der möglicherweise für Audio eingerichtet ist.

Beispiel:

```
OPTION LCDPANEL ILI9341, OR, DC, RESET, CS, GP11
```

Die Hintergrundbeleuchtung kann dann mit diesem Befehl auf 40 % eingestellt werden:

```
BACKLIGHT 40
```

Die meisten SSD1963-basierten LCD-Panels verfügen über drei Paare von Lötpads auf der Leiterplatte, die unter der Überschrift „Backlight Control” (Hintergrundbeleuchtungssteuerung) zusammengefasst sind, wie rechts dargestellt.

Normalerweise ist das mit „LED-A” gekennzeichnete Paar mit einem Null-Ohm-Widerstand kurzgeschlossen, wodurch die Helligkeit der Hintergrundbeleuchtung mit einem PWM-Signal (Pulsweitenmodulation) am LED-A-Pin des Hauptanschlusses des Displaypanels gesteuert werden kann.

Es ist jedoch besser, den SSD1963-Controller zur Erzeugung dieses Signals zu verwenden, da dadurch ein I/O-Pin frei wird und eine feinere Steuerung möglich ist.

Um die SSD1963-Steuerung zu verwenden, sollte der Null-Ohm-W
zum Kurzschließen des nahe gelegenen Paares von Lötpads mit der
kann dann mit dem Befehl BACKLIGHT über den SSD1963-Controller eingestellt werden (dies geschieht automatisch, es muss nichts konfiguriert werden).

Unterstützung für Touch-

Viele LCD-Panels werden mit einem resistiven Touchscreen und einem zugehörigen Controller-Chip geliefert. MMBasic unterstützt diese Schnittstelle vollständig, sodass viele der in einem Projekt verwendeten physischen Regler und Schalter als durch Berührung aktivierte Bildschirmsteuerungen implementiert werden können. Als Alternative zum resistiven Touchscreen unterstützt MMBasic auch kapazitive Touchscreens auf Basis des FT6336-Controllers.

In allen Fällen sollte zuerst das LCD-Display selbst konfiguriert und getestet werden, bevor die Touch-Funktion konfiguriert werden kann. Der Touch-Controller verwendet das SPI-Protokoll für die Kommunikation. Bei LCD-Displays, die das SPI-Protokoll verwenden, wurde dies normalerweise zuvor mit dem Befehl OPTION SYSTEM SPI durchgeführt.

Bei Displaypanels, die I^2C oder eine parallele Schnittstelle verwenden, muss der Befehl OPTION SYSTEM SPI separat verwendet werden, um den System-SPI-Bus für die Verwendung durch den Touch-Controller zu definieren. Dieser Befehl wurde am Anfang dieses Kapitels erläutert und im Kapitel *„Optionen*” dieses Handbuchs ausführlich beschrieben.

Wenn Sie beispielsweise die empfohlenen Pins für ein 8-Bit-Parallel-Display (wie oben beschrieben) verwenden, würden Sie Folgendes verwenden:

```
OPTION SYSTEM SPI GP10, GP11, GP12
```

Um die Touch-Funktion nutzen zu können, muss MMBasic mit dem Befehl OPTION TOUCH mitgeteilt werden, dass der Touch-Controller auf dem System-SPI-Bus verfügbar ist. Dadurch wird MMBasic mitgeteilt, welche Pins für die Chip-Select- und Interrupt-Signale verwendet werden.

Bei einem typischen ILI9341-Display werden beispielsweise Chip Select auf den Pin GP12 und Interrupt auf GP11 gesetzt:

```
OPTION TOUCH GP12, GP11
```

Dies muss an der Eingabeaufforderung eingegeben werden und führt dazu, dass die Firmware neu gestartet und die USB-Konsolenschnittstelle getrennt wird, die dann erneut angeschlossen werden muss.

Wenn die PicoMite-Firmware neu gestartet wird, initialisiert MMBasic automatisch den Touch-Controller. Um die Konfiguration zu überprüfen, können Sie den Befehl OPTION LIST verwenden, um alle eingestellten Optionen aufzulisten, einschließlich der Konfiguration des Display-Panels und des Touchscreens.

Vorsicht ist geboten, wenn der SPI-Port von mehreren Geräten (SD-Karte, Touchscreen usw.) gemeinsam genutzt wird. In diesem Fall müssen alle Chip-Select-Signale in MMBasic konfiguriert oder alternativ deaktiviert werden.

Kalibrieren des Touch- -Bildschirms

Bevor die Touch-Funktion verwendet werden kann, muss sie mit dem GUI-Befehl CALIBR kalibriert werden.

Dieser Befehl zeigt ein Ziel in der oberen linken Ecke des Bildschirms an (wie abgebildet). Drücken Sie mit einem spitzen, aber stumpfen Gegenstand (z. B. einem Zahnstocher) genau auf die Mitte des Ziels und halten Sie ihn mindestens eine Sekunde lang gedrückt. MMBasic zeichnet diese Position auf und setzt dann die Kalibrierung fort, indem es das Ziel nacheinander in den anderen drei Ecken des Bildschirms zur Berührung und Kalibrierung anzeigt.

Die Kalibrierungsroutine kann eine Warnung ausgeben, dass die Kalibrierung nicht genau w können die Touch-Funktion weiterhin verwenden, wenn Sie möchten, aber es wäre besser, die Kalibrierung mit größerer Sorgfalt zu wiederholen.

Nach der Kalibrierung können Sie die Touch-Funktion mit dem GUI-Befehl TEST TOUCH testen. Dieser Befehl schaltet den Bildschirm aus und wartet auf eine Berührung. Wenn der Bildschirm berührt wird, erscheint ein weißer Punkt auf dem Display, der die Position der Berührung auf dem Bildschirm markiert. Wenn die Kalibrierung erfolgreich durchgeführt wurde, sollte der Punkt genau unter der Position des Stifts auf dem Bildschirm angezeigt werden. Um die Testroutine zu beenden, können Sie die Leertaste auf der Tastatur der Konsole drücken.

Touch- sfunktionen

Um zu erkennen, ob und wo der Bildschirm berührt wird, können Sie die folgenden Funktionen in einem BASIC-Programm verwenden:

- TOUCH(X)
  Gibt die X-Koordinate der aktuell berührten Stelle zurück oder -1, wenn der Bildschirm nicht berührt wird.

- TOUCH(Y)
  Gibt die Y-Koordinate der aktuell berührten Stelle zurück oder -1, wenn der Bildschirm nicht berührt wird.

Touch- -Interrupts

Ein Interrupt kann auf die IRQ-Pin-Nummer gesetzt werden, die bei der Konfiguration der Touch-Funktion angegeben wurde. Um eine Berührung zu erkennen, sollte der Interrupt als INTL (d. h. von hoch nach niedrig) konfiguriert werden. Der Interrupt kann mit dem Befehl SETPIN pin, OFF abgebrochen werden.

Das folgende Programm veranschaulicht, wie der Touch-Interrupt verwendet werden kann. Bei jeder Berührung des Bildschirms werden die Koordinaten dieser Berührung auf der Konsole ausgegeben. Es wird davon ausgegangen, dass der Befehl OPTION TOUCH 7, 15 zur anfänglichen Konfiguration der Touch-Funktion verwendet wurde:

```
SETPIN 15, INTL, MyInt          ' geht davon aus, dass OPTION TOUCH 7, 15 verwendet
wurde DO : LOOP

SUB MyInt                       ' Unterprogramm, das bei einem Touch-Interrupt
  aufgerufen wird PRINT TOUCH(X) TOUCH(Y)
END SUB
```

LCD-Anzeige als Konsole -Ausgabe

Eine PS2- oder USB-Tastatur kann mit einem LCD-Display verwendet werden, um einen vollständig eigenständigen Computer zu erstellen, der direkt mit der MMBasic-Eingabeaufforderung startet. Diese Funktion eignet sich besonders gut für 5- und 7-Zoll-LCD-Displays mit dem SSD1963-Controller. Diese können viel Text anzeigen und schnell aktualisieren, was zu einer Benutzererfahrung führt, die mit einem PicoMite-basierten Computer mit VGA- oder HDMI-Videoausgang vergleichbar ist.

Das LCD muss zunächst mit OPTION LCDPANEL konfiguriert werden. Um dann die Konsolenausgabe auf dem LCD-Panel zu aktivieren, sollten Sie den folgenden Befehl verwenden:

```
OPTION LCDPANEL CONSOLE [Schriftart [, fc [, bc [, blight]]] [,NOSCROLL]
```

„font" ist die Standardschriftart, „fc" ist die Standard-Vordergrundfarbe, „bc" ist die Standard-Hintergrundfarbe und „blight" ist die Standardhelligkeit der Hintergrundbeleuchtung (2 bis 100). Diese Einstellungen werden im Flash-Speicher gespeichert und dienen zur Konfiguration von MMBasic beim Einschalten. Sie sind alle optional und standardmäßig auf Schriftart 2, Weiß, Schwarz und 100 % eingestellt.

Bei Displays, bei denen der Framebuffer nicht gelesen werden kann, setzt die Firmware automatisch die Option NOSCROLL. Wenn diese Option aktiviert ist und die Ausgabe den unteren Bildschirmrand erreicht, wird der Bildschirm gelöscht und die Ausgabe oben fortgesetzt. Dies gilt auch für den integrierten Editor. Bei SPI-Displays, die den Framebuffer lesen können (z. B. ILI9341), ist das Scrollen sehr langsam, sodass der Parameter NOSCROLL gesetzt werden kann, um die Ausgabe- und Bearbeitungsleistung zu verbessern.

Beachten Sie, dass dies nicht erforderlich ist, wenn das Display im Hochformat verwendet wird, da dann H/W-Scrolling verwendet wird.

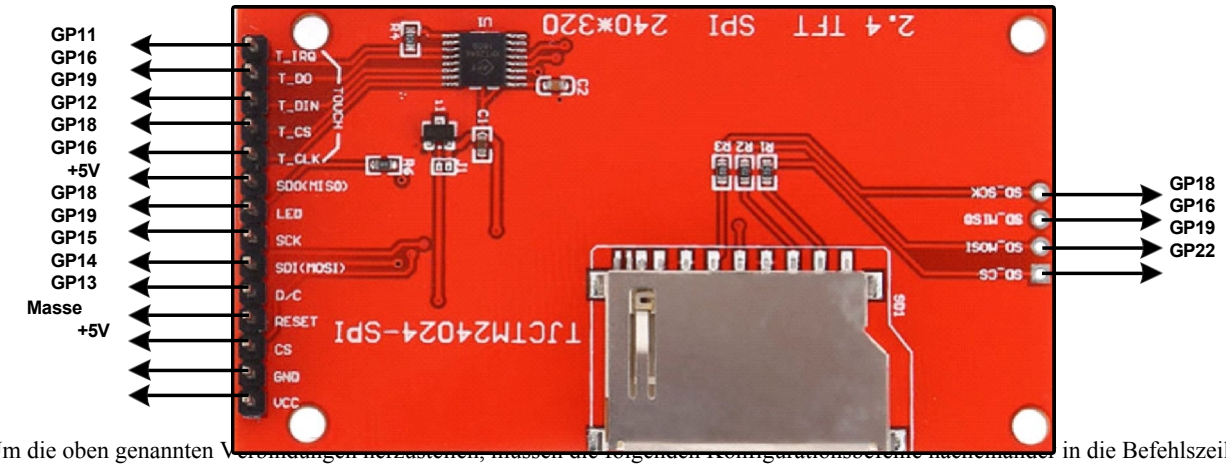
Die Farbcodierung im Editor wird ebenfalls durch diesen Befehl aktiviert (siehe Kapitel *Vollbild-Editor*). Um die Verwendung des LCD-Panels als Konsole zu deaktivieren, lautet der Befehl OPTION LCDPANEL NOCONSOLE.

Beispiel für die Konfiguration eines SPI-LCD-Panels

Im Folgenden finden Sie eine Zusammenfassung, wie ein typisches LCD-Panel mit einem ILI9341-Controller angeschlossen werden kann. Dieses Beispiel unterstützt den SD-Kartensteckplatz, das LCD-Display und die Touch-Schnittstelle.

Typische Panels finden Sie auf ebay.com und ähnlichen Websites, indem Sie nach dem Stichwort „ILI9341" suchen. Vergewissern Sie sich, dass die Anschlüsse auf der Rückseite des Panels denen in der folgenden Abbildung entsprechen:

Das Panel sollte wie abgebildet an den Raspberry Pi Pico angeschlossen werden:



Um die oben genannten Verbindungen herzustellen, müssen die folgenden Konfigurationsbefehle nacheinander in die Befehlszeile eingegeben werden:

```
OPTION SYSTEM SPI GP18, GP19, GP16
OPTION LCDPANEL ILI9341, L, GP15, GP14, GP13 OPTION
TOUCH GP12, GP11
OPTION SDCARD GP22
```

Diese Befehle werden gespeichert und beim Einschalten automatisch angewendet. Beachten Sie, dass nach der Eingabe jedes Befehls die Firmware neu gestartet wird und die USB-Verbindung unterbrochen wird und erneut hergestellt werden muss.

Als Nächstes sollte der Touchscreen kalibriert werden mit:

```
GUI CALIBRATE
```

Anschließend können Sie die verschiedenen Komponenten testen.

Mit dem folgenden Befehl werden mehrere farbige, sich überlappende Kreise auf dem LCD-Bildschirm angezeigt, um zu überprüfen, ob das LCD-Display korrekt angeschlossen ist:

```
GUI TEST LCDPANEL
```

Mit dem folgenden Befehl wird die Touch-Oberfläche getestet. Wenn Sie den LCD-Bildschirm berühren, sollte an der Stelle der Berührung ein Punkt auf dem Bildschirm erscheinen.

```
GUI TEST TOUCH
```

Wenn dies nicht korrekt ist, müssen Sie möglicherweise den GUI-Befehl CALIBRATE ein zweites Mal ausführen und dabei sorgfältiger vorgehen.

Abschließend werden die Dateien auf der SD-Karte aufgelistet. Wenn dies ohne Fehler ausgeführt wird, können Sie sicher sein, dass die SD-Karten-Schnittstelle in Ordnung ist.

```
DATEIEN
```

Wenn Sie Probleme mit der Anzeige haben, sollten Sie alle Verbindungen trennen und die Optionen mit dem Befehl OPTION RESET löschen, damit Sie mit einer sauberen Weste beginnen können. Schließen Sie dann alles Schritt für Schritt wieder an und konfigurieren und testen Sie jeden neuen Schritt, während Sie fortfahren. Beachten Sie dabei, dass jedes an den SPI-Bus angeschlossene Gerät in MMBasic konfiguriert werden muss oder dass seine Chip-Auswahlleitung auf logisch hoch gehalten werden muss. Dies ist

, weil die Spannung an einer nicht angeschlossenen Chip-Auswahlleitung schwankt und möglicherweise dazu führt, dass das falsche Gerät auf Signale reagiert, die für ein anderes Gerät bestimmt sind.

Beachten Sie auch, dass der ILI9341-Controller sehr empfindlich gegenüber elektrostatischer Entladung ist. Wenn das Panel nicht reagiert, könnte es leicht beschädigt sein, und es lohnt sich, es mit einem anderen Panel zu testen.

Grafik- sfunktionen

Diese Befehle und Funktionen wirken auf angeschlossene LCD-Panels und VGA/HDMI-Videoausgänge. Eine Anleitung zur Verwendung dieser Funktionen ist in der Firmware-Distributionsdatei enthalten. Siehe Datei: *Graphics in the PicoMite.pdf*

Unterstützte Grafik- -Hardware

LCD-Panels

Die Auflösung und die Anzahl der Farben, die von einem LCD-Panel unterstützt werden, werden vom Panel selbst und vom Treiber bestimmt – Einzelheiten finden Sie im Kapitel *„Anzeigepanels*".

VGA-Video

Es gibt eine Reihe von Modi, die mit dem Befehl MODE ausgewählt werden können:

OPTION AUFLÖSUNG 640x480

MODUS 1 640 x 480 monochrom mit RGB121-Kacheln, optionaler Ebenenpuffer
MODUS 2 320 x 240 4-Bit-Farbe, optionaler Ebenenpuffer (nur RP2350) zweiter optionaler Ebenenpuffer MODUS 3 640 x 480 4-Bit-Farbe, optionaler Layer-Puffer (nur RP2350)

OPTIONALE AUFLÖSUNG 720 x 400

MODUS 1 720 x 400 monochrom mit RGB121-Kacheln, optionaler Ebenenpuffer
MODUS 2 360 x 200 4-Bit-Farbe, optionaler Layer-Puffer (nur RP2350) zweiter optionaler Layer-Puffer MODUS 3 720 x 400 4-Bit-Farbe, optionaler Layer-Puffer (nur RP2350)

OPTIONALE AUFLÖSUNG 800 x 600 (nur RP2350)

MODUS 1 800 x 600 monochrom mit RGB121-Kacheln, optionaler Ebenenpuffer
MODUS 2 400 x 300 4-Bit-Farbe, zwei optionale Ebenen
MODUS 3 800 x 600 4-Bit-Farbe, optionaler Ebenenpuffer

OPTION AUFLÖSUNG 848 x 480 (nur RP2350)

MODUS 1 848 x 480 monochrom mit RGB121-Kacheln, optionaler Ebenenpuffer
MODUS 2 424 x 240 4-Bit-Farbe, zwei optionale Ebenen
MODUS 3 848 x 480 4-Bit-Farbe, optionaler Layer-Puffer

HDMI-Video (nur RP2350)

Jede HDMI-Auflösung kann in einer Reihe von Modi betrieben werden, die mit dem Befehl MODE eingestellt werden:

OPTION AUFLÖSUNG 640 x 480

MODUS 1 640 x 480 x 2 Farben mit RGB555, optionaler Layer-Puffer
MODUS 2 320x240x16 Farben und Farbabbildung auf RGB555-Palette, zwei optionale Ebenen MODE 3 640 x 480 x 16 Farben und Farbabbildung auf RGB555-Palette, optionaler Ebenenpuffer MODUS 4 320 x 240 x 32768 Farben, optionaler Layer-Puffer
MODUS 5 320 x 240 x 256 Farben und Farbabbildung auf RGB555-Palette, optionaler Ebenenpuffer

OPTIONALE AUFLÖSUNG 720 x 400

MODUS 1 720 x 400 monochrom mit RGB555-Kacheln, optionaler Ebenenpuffer
MODUS 2 360 x 200 4-Bit-Farbe und Farbabbildung auf RGB555-Palette, zwei optionale Ebenen MODUS 3 720 x 400 4-Bit-Farbe und Farbabbildung auf RGB555-Palette, optionaler Ebenenpuffer MODUS 4 360 x 200 x 32768 Farben, optionaler Layer-Puffer
MODUS 5 360 x 200 x 256 Farben und Farbabbildung auf RGB555-Palette, optionaler Ebenenpuffer

OPTION AUFLÖSUNG 800x600 (nur RP2350)

MODUS 1 800x600 monochrom mit RGB332-Kacheln, optionaler Layer-Puffer
MODUS 2 400x300 4-Bit-Farbe und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 3 800x600 4-Bit-Farbe und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 5 400x300x256 Farben, optionaler Ebenenpuffer

OPTION AUFLÖSUNG 848x480 (nur RP2350)

MODUS 1 848x480 monochrom mit RGB332-Kacheln, optionaler Ebenenpuffer
MODUS 2 424x240 4-Bit-Farbe und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 3 848x480 4-Bit-Farbe und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 5 424x240x256 Farben, optionaler Ebenenpuffer

OPTION AUFLÖSUNG 1280x720

MODUS 1 1280x720x2 Farben mit RGB332, optionaler Layer-Puffer
MODUS 2 320x180x16 Farben und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 3 640x360x16 Farben und Farbabbildung auf RGB332-Palette, optionaler Ebenenpuffer MODUS 5 320x180x256 Farben, optionaler Ebenenpuffer

OPTION AUFLÖSUNG 1024x768

MODUS 1 1024x768x2 Farben mit RGB332-Kacheln, optionaler Ebenenpuffer
MODUS 2 256x192x16 Farben und Farbabbildung auf RGB332-Palette, optionaler Layer-Puffer MODUS 3 512x384x16 Farben und Farbabbildung auf RGB332-Palette, optionaler Layer-Puffer MODUS 5 256x192x256 Farben, optionaler Layer-Puffer

OPTION AUFLÖSUNG 1024x600

MODUS 1 1024x600 Mono mit Kacheln
MODUS 2 256x150 RGB121 MODUS 3
512x300 RGB121 MODUS 5
256x150 RGB332

OPTION AUFLÖSUNG 800x480 (Hinweis: reduziert den verfügbaren Heap-Speicher)

MODUS 1 800x480 Mono mit Kacheln
MODUS 2 400x240 RGB121
MODUS 3 800 x 480 RGB121
MODUS 5 400x240 RGB332

Farben

Die Farbe wird als Echtfarben-24-Bit-Zahl angegeben, wobei die oberen acht Bits die Intensität der roten Farbe, die mittleren acht Bits die Intensität der grünen Farbe und die unteren acht Bits die Intensität der blauen Farbe darstellen. Am einfachsten lässt sich diese Zahl mit der Funktion RGB() generieren, die folgende Form hat:

```
RGB(rot, grün, blau)
```

Die Funktion RGB() unterstützt auch eine Abkürzung, mit der Sie gängige Farben durch Benennung angeben können. Zum Beispiel RGB(rot) oder RGB(cyan). Die Farben, die mit der Abkürzungsform benannt werden können, sind Weiß, Schwarz, Blau, Grün, Cyan, Rot, Magenta, Gelb, Braun, Weiß, Orange, Rosa, Gold, Lachs, Beige, Hellgrau und Grau (oder in der US-amerikanischen Schreibweise Gray/Lightgray).

MMBasic übersetzt alle Farben automatisch in das vom jeweiligen Display-Controller benötigte Format. Im Fall des ILI9341-LCD-Controllers sind dies beispielsweise 64K Farben im 565-Format.

Die Standardeinstellung für Befehle, die einen Farbparameter erfordern, kann mit dem Befehl COLOUR (auch COLOR geschrieben) festgelegt werden. Dies ist praktisch, wenn Ihr Programm ein einheitliches Farbschema verwendet. Sie können dann die Standardeinstellungen festlegen und die Kurzform der Zeichenbefehle in Ihrem gesamten Programm verwenden.

Der Befehl COLOUR hat das folgende Format: `COLOUR Vordergrundfarbe, Hintergrundfarbe`

Schriftarten

Es gibt acht integrierte Schriftarten. Diese sind:

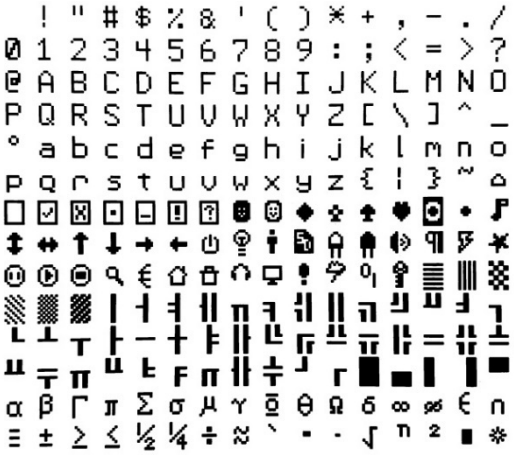
Schrift Nummer	Größe (Breite x Höhe)	Zeichen Zeichensatz	Beschreibung
1	8 x 12	Alle 95 ASCII-Zeichen plus 7F bis FF (hex)	Standardschriftart (Standard beim Start).
2	12 x 20	Alle 95 ASCII-Zeichen	Mittlere Schriftgröße.
3	16 x 16	Alle 95 ASCII-Zeichen	Eine große Schriftart für VGA-Versionen.
3	16 x 24	Alle 95 ASCII-Zeichen	Eine große Schriftart für HDMI-Versionen und LCD-Bildschirme.
4	10x16	Alle 95 ASCII-Zeichen plus 7F bis FF (hex)	Eine Schriftart mit erweiterten Grafikzeichen. Geeignet für hochauflösende Displays.
5	24 x 32	Alle 95 ASCII-Zeichen	Extra große Schriftart, sehr klar.

6	32 x 50	0 bis 9 plus einige Symbole	Zahlen plus Dezimalpunkt, Plus-, Minus-, Gleichheits-, Grad- und Doppelpunktzeichen. Sehr gut lesbar.
7	6 x 8	Alle 95 ASCII-Zeichen	Eine kleine Schriftart, die bei niedrigen Auflösungen nützlich ist.
8	4 x 6	Alle 95 ASCII-Zeichen	Eine noch kleinere Schriftart.

Beachten Sie, dass Schriftart 3 bei Verwendung mit VGA-Videoausgabe eine Größe von 16 x 16 Pixel hat, bei HDMI- und LCD-Bildschirmen jedoch eine Größe von 16 x 24.

In allen Schriftarten (einschließlich Schriftart Nr. 6) wurde das Backquote-Zeichen (60 hexadezimal oder 96 dezimal) durch das Gradzeichen (º) ersetzt.

Schriftart Nr. 1 (die Standardschriftart) und Schriftart Nr. 4 verfügen über einen erweiterten Zeichensatz, der alle Zeichen von CHR$(32) bis CHR$(255) oder 20 bis FF (hex) abdeckt, wie rechts dargestellt.



Eingebettete „ “-Schriftarten

Bei Bedarf können zusätzliche Schriftarten in ein BASIC-Programm eingebettet werden. Diese Schriftarten funktionieren genauso wie die integrierten Schriftarten (d. h. sie werden mit dem Befehl FONT ausgewählt oder im Befehl TEXT angegeben).

Das Format einer eingebetteten Schriftart lautet:

```
DefineFont #Nbr hex
     [[ hex[…]
     hex [[ hex[…] END
DefineFont
```

Es muss mit dem Schlüsselwort „DefineFont” beginnen, gefolgt von der Schriftartnummer (der optional ein #-Zeichen vorangestellt werden kann). Es kann jede Schriftartnummer im Bereich von 2 bis 5 und 8 bis 16 angegeben werden. Wenn sie mit einer integrierten Schriftart übereinstimmt, ersetzt sie diese Schriftart.

Der Hauptteil der Schriftart besteht aus einer Folge von 8-stelligen Hexadezimalwörtern, wobei jedes Wort durch ein oder mehrere Leerzeichen oder eine neue Zeile getrennt ist. Die Schriftartdefinition wird durch das Schlüsselwort „End DefineFont “ beendet. Diese können an beliebiger Stelle in einem Programm platziert werden, und MMBasic überspringt sie. Dieses Format entspricht dem von Micromite verwendeten Format.

Weitere Schriftarten und Informationen finden Sie im Ordner „Embedded Fonts“ im PicoMite-Firmware-Download. Diese Schriftarten decken eine Vielzahl von Zeichensätzen ab, darunter eine Symbolschriftart (Dingbats), die sich besonders für die Erstellung von Bildschirmsymbolen usw. eignet.

Bildschirm skoordinaten

Alle Koordinaten und Messungen auf dem Bildschirm erfolgen in Pixeln, wobei die X-Koordinate die horizontale Position und die Y-Koordinate die vertikale Position angibt. Die obere linke Ecke des Bildschirms hat die Koordinaten X = 0 und Y = 0, und die Werte steigen, wenn Sie sich nach unten und nach rechts auf dem Bildschirm bewegen.

Es gibt vier schreibgeschützte Variablen, die nützliche Informationen über das aktuell angeschlossene Display liefern:

- MM. HRES
  Gibt die Breite des Displays (die X-Achse) in Pixeln zurück.
- MM.VRES
  Gibt die Höhe des Displays (Y-Achse) in Pixeln zurück.
- MM.INFO(FONTHEIGHT)
  Gibt die Höhe der aktuellen Standardschriftart (in Pixeln) zurück. Alle Zeichen einer Schriftart haben dieselbe Höhe.
- MM.INFO(FONTWIDTH)
  Gibt die Breite eines Zeichens in der aktuellen Schriftart (in Pixeln) zurück. Alle Zeichen haben die gleiche Breite.

Befehle zum Zeichnen von en

Es gibt zehn grundlegende Zeichenbefehle, die Sie in MMBasic-Programmen zum Zeichnen von Grafiken verwenden können. Die meisten davon haben optionale Parameter. Sie können diese am Ende eines Befehls komplett weglassen oder zwei Kommas hintereinander verwenden, um einen fehlenden Parameter anzuzeigen. Der fünfte Parameter des Befehls LINE ist beispielsweise optional, sodass Sie dieses Format verwenden können:

```
LINE 0, 0, 100, 100, , rgb(red)
```

Optionale Parameter sind unten kursiv dargestellt, zum Beispiel: *font*.

In den folgenden Befehlen ist C die Zeichenfarbe und standardmäßig die aktuelle Vordergrundfarbe. FILL ist die Füllfarbe, deren Standardwert -1 ist, was bedeutet, dass keine Füllung verwendet werden soll.

Die grundlegenden Zeichenbefehle sind:

- `CLS` *C*
  Löscht den Bildschirm in der Farbe C. Wenn C nicht angegeben ist, wird die aktuelle Standard-Hintergrundfarbe verwendet.

- `PIXEL X, Y,` *C*
  Leuchtet ein Pixel auf. Wenn C nicht angegeben ist, wird die aktuelle Standard-Vordergrundfarbe verwendet.

- `LINE X1, Y1, X2, Y2,` *LW, C*
  Zeichnet eine Linie, die bei X1 und Y1 beginnt und bei X2 und Y2 endet.
  LW ist die Breite der Linie und gilt nur für horizontale oder vertikale Linien. Der Standardwert ist 1, wenn nichts angegeben ist oder wenn die Linie diagonal verläuft. Es gibt eine erweiterte Version für diagonale Linien (siehe LINE AAA).

- `BOX X, Y, W, H,` *LW, C, FILL*
  Zeichnet ein Rechteck, das bei X und Y beginnt und W Pixel breit und H Pixel hoch ist.
  LW ist die Breite der Seiten der Box und kann Null sein. Der Standardwert ist 1.

- `RBOX X, Y, W, H,` *R, C, FILL*
  Zeichnet einen Kasten mit abgerundeten Ecken, beginnend bei X und Y, der W Pixel breit und H Pixel hoch ist.
  R ist der Radius der Ecken des Kastens. Der Standardwert ist 10.

- `CIRCLE X, Y, R,` *LW, A, C, FILL*
  Zeichnet einen Kreis mit X und Y als Mittelpunkt und einem Radius R. LW ist die Breite der Linie, die für den Umfang verwendet wird und kann Null sein (Standardwert ist 1). A ist das Seitenverhältnis, das eine Gleitkommazahl ist und standardmäßig auf 1 gesetzt ist. Bei einem Seitenverhältnis von 0,5 wird beispielsweise ein Oval gezeichnet, dessen Breite halb so groß ist wie seine Höhe.

- `TEXT X, Y, STRING,` *ALIGNMENT, FONT, SCALE, C, BC*
  Zeigt eine Zeichenfolge an, die bei X und Y beginnt. ALIGNMENT ist 0, 1 oder 2 Zeichen (ein Zeichenfolgenausdruck oder eine Variable
  ist ebenfalls zulässig), wobei der erste Buchstabe die horizontale Ausrichtung um X angibt und L, C oder R für links-, zentriert- oder rechtsbündigen Text stehen kann, und der zweite Buchstabe die vertikale Ausrichtung um Y angibt und T, M oder B für oben-, mittig- oder untenbündigen Text stehen kann. Die Standardausrichtung ist links/oben. Ein zusätzlicher Code-Buchstabe kann verwendet werden, um den Text zu drehen (Details siehe unten). FONT und SCALE sind optional und standardmäßig auf die durch den Befehl FONT festgelegten Werte eingestellt. C ist die Zeichenfarbe und BC ist die Hintergrundfarbe. Sie sind optional und standardmäßig auf die durch den Befehl COLOUR festgelegten Werte eingestellt.

- `GUI BITMAP X, Y, BITS,` *WIDTH, HEIGHT, SCALE, C, BC*
  Zeigt die Bits in einer Bitmap beginnend bei X und Y an. HEIGHT und WIDTH sind die Abmessungen der Bitmap , wie sie auf dem LCD-Bildschirm angezeigt werden, und sind standardmäßig auf 8x8 eingestellt. SCALE, C und BC sind dieselben wie beim Befehl TEXT. Die Bitmap kann eine Ganzzahl oder eine Zeichenfolgenvariable oder -konstante sein und wird unter Verwendung des ersten Bytes als erste Bits der obersten Zeile (zuerst Bit 7, dann Bit 6 usw.) gezeichnet, gefolgt vom nächsten Byte usw. Wenn die oberste Zeile gefüllt ist, beginnt die nächste Zeile der angezeigten Bitmap mit dem nächsten Bit in der Ganzzahl oder Zeichenfolge.

- `POLYGON n, xarray%(), yarray%() [, bordercolour] [, fillcolour]`
  Zeichnet ein gefülltes oder umrandetes Polygon mit n xy-Koordinatenpaaren in xarray%() und yarray%(). Wenn „fillcolour” , wird nur der Polygonumriss gezeichnet. Wenn „bordercolour” weggelassen wird, wird standardmäßig die aktuelle Vordergrundfarbe verwendet.

- `ARC x, y, r1, [r2], a1, a2 [, c]`
  Zeichnet einen Kreisbogen mit einer bestimmten Farbe und Breite zwischen zwei Radien (in Grad definiert). Parameter
  Für den ARC-Befehl sind die x- und y-Koordinaten des Mittelpunkts des Bogens, der innere und äußere Radius, der Start- und Endwinkel des Bogens und die Farbe des Bogens anzugeben. Der Nullpunkt befindet sich in der 12-Uhr-Position.

Drehung von Text im Befehl „ “

Wie oben beschrieben, kann die Ausrichtung des Textes im Befehl TEXT durch Verwendung eines oder zweier Zeichen in einem Zeichenfolgenausdruck für den dritten Parameter des Befehls festgelegt werden. In dieser Zeichenfolge können Sie auch ein drittes Zeichen angeben, um die Drehung des Textes anzugeben.

Dieses Zeichen kann eines der folgenden sein:

N für normale Ausrichtung

V für vertikalen Text, wobei jedes Zeichen unter dem vorherigen von oben nach unten verläuft. I

wird der Text invertiert (d. h. auf den Kopf gestellt)

U Der Text wird um 90º gegen den Uhrzeigersinn gedreht. D

Der Text wird um 90º im Uhrzeigersinn gedreht.

Beispielsweise wird der folgende Befehl den Text „LCD Display” vertikal am linken Rand des Displayfelds und vertikal zentriert anzeigen:

```
TEXT 0, 250, „LCD Display”, „LMV”, 5
```

Die Positionierung erfolgt relativ zur oberen linken Ecke des Zeichens bei normaler Betrachtung, sodass bei einer Umkehrung von 100,100 der obere linke Pixel des ersten Zeichens bei 100,100 liegt und der Text dann über y=101 und links von x=101 angezeigt wird. Ebenso wird das „R” in der Ausrichtungszeichenfolge aus der Perspektive des Zeichens in seiner jeweiligen Ausrichtung (nicht des Bildschirms) betrachtet.

Transparenter Text „ “

Der VGA- oder HDMI-Videoausgang oder LCD-Displays, die SSD1963, ST7796S, ILI9341, ST7789_320 oder ILI9488 mit angeschlossenem MISO verwenden, sind in der Lage, transparenten Text anzuzeigen.

In diesem Fall ermöglicht der Befehl TEXT die Verwendung von -1 für die Hintergrundfarbe. Das bedeutet, dass der Text über den Hintergrund gezeichnet wird und das Hintergrundbild durch die Lücken zwischen den Buchstaben hindurchscheint.

Framebuffer und Ebenen „ “

Alle Varianten der Firmware können einen oder zwei Framebuffer im Speicher und einen oder zwei Layer-Puffer erstellen (dies ist speicherabhängig). Dabei handelt es sich um Speicherbereiche mit derselben Breite und Höhe wie das Hauptdisplay. Bei HDMI- und VGA-Displays haben sie dieselbe Farbtiefe wie der aktuelle Modus. Bei LCD-Displays haben sie 4 Bit pro Pixel (16 Farben).

Je nach Firmware-Version und aktuellem Anzeigemodus werden Framebuffer oder Layer-Puffer entweder aus vorab zugewiesenem Speicher oder aus dem Benutzerspeicher erstellt.

Framebuffer können verwendet werden, um Bilddaten zu erstellen, die auf das physische Display kopiert werden können. Layer-Puffer werden in der Regel verwendet, um Teilbilder zu erstellen, die über ein Hintergrundbild gelegt werden können und eine effiziente Methode zum Verschieben von Anzeigeelementen über einen statischen Hintergrund bieten.

Alle Standardbefehle zum Zeichnen von Grafiken können auf einem Framebuffer oder Layer-Puffer genauso verwendet werden wie beim Schreiben auf das physische Display. Der Befehl FRAMEBUFFER WRITE wird verwendet, um das Ziel der Grafikausgabe mithilfe eines *Codes* zu steuern.

Der *Code* ist ein einzelnes Zeichen, das Folgendes

sein kann: N Das physische

Ausgabegerät.

F Der Framebuffer.

2 Ein zweiter Framebuffer (nur RP2350) L

Der Layerbuffer

T Ein zweiter Layerbuffer (nur RP2350) Die

grundlegenden Framebuffer-Befehle sind:

```
FRAMEBUFFER CREATE ' Code F
FRAMEBUFFER LAYER       ' Code L
FRAMEBUFFER CLOSE FRAMEBUFFER
WRITE Code
FRAMEBUFFER COPY code1, code2 [,B]
```

Weitere Informationen zu Framebuffer-Befehlen finden Sie in den detaillierten Befehlsbeschreibungen.

Bei den VGA- und HDMI-Versionen der Firmware werden je nach Anzeigemodus und CPU-Geschwindigkeit (>=252 MHz) automatisch Ebenen über das Hauptanzeigebild gelegt, wenn dieses auf den Bildschirm ausgegeben wird.

Bei LCD-Anzeigen wird der Befehl FRAMEBUFFER MERGE verwendet, um das endgültige Bild aus einem Framebuffer und einem Layer-Puffer zu erstellen. Das Spiel PETSCII robots zeigt, wie diese Technik mit großem Effekt eingesetzt werden kann.

Die automatische Anwendung eines Ebenenpuffers ist in den VGA-Versionen Modus 2 und Modus 3 (nur RP2350) sowie in den HDMI-Modi 2, 3, 4 und 5 implementiert. Zwei Ebenenpuffer sind nur auf dem RP2350 und in den folgenden Modi verfügbar: VGA-Modus 2, HDMI-Modi 2 und 5.

BLIT- und Sprite- sbefehle

In früheren Versionen der Firmware waren die Befehle „blit“ und „sprite“ Synonyme für dieselbe Funktion. Ab Version 6.00.00 sind sie separate Befehle. Der Unterschied besteht darin, dass BLIT ein einfacher Speichervorgang ist, bei dem Daten von einem Display oder Speicher auf ein Display oder einen Speicher kopiert werden. Sprites sind komplexer und ermöglichen es dem Programmierer, Elemente über einem Hintergrund anzuzeigen und sie dann über den Hintergrund zu bewegen, ohne das Hintergrundbild zu beschädigen. Darüber hinaus kann der Programmierer die Sprite-Funktionalität nutzen, um Kollisionen zwischen Sprites sowie zwischen einem Sprite und den Rändern des Displays zu erkennen.

Sprites können nur verwendet werden, wenn das Display das Lesen aus seinem Framebuffer unterstützt, und auch die Blit-Funktionalität ist in diesem Fall eingeschränkt. Sprites sind für alle Versionen der Firmware aktiviert, wenn sie auf einem In-Memory-Framebuffer und VGA- und HDMI-Versionen der Firmware direkt mit dem Bildschirm verwendet werden.

Sprites werden immer als RGB121-Nibbles mit 2 Pixeln pro Byte gespeichert. Im Gegensatz dazu werden BLIT-Puffer als RGB888-Werte gespeichert und können daher mit Vollfarb-LCD-Displays verwendet werden. Dies geht natürlich zu Lasten eines deutlich höheren Speicherbedarfs.

Weitere Informationen zur Verwendung von Sprites finden Sie unter dem Befehl SPRITE und der Funktion sowie in Anhang G.

Wenn das Display transparenten Text anzeigen kann, ermöglicht der Befehl BLIT, einen Teil des aktuell auf dem Display angezeigten Bildes in einen Speicherpuffer zu kopieren und später wieder auf das Display zu kopieren. Dies ist nützlich, wenn etwas über den Hintergrund gezeichnet und später wieder entfernt werden muss, ohne das Bild im Hintergrund zu beschädigen. Beispiele hierfür sind ein Spiel, in dem sich eine Figur vor einer Landschaft bewegt, oder die sich bewegende Nadel eines fotorealistischen Messgeräts.

Die verfügbaren Standard-BLIT-Befehle sind:

```
BLIT READ #b, x, y, w, h
BLIT WRITE #b, x, y [,mode] BLIT
LOAD #b, f$, x, y, w, h BLIT CLOSE
#b
```

#b ist die Puffernummer im Bereich von 1 bis 64. x und y sind die Koordinaten der oberen linken Ecke und w und h sind die Breite und Höhe des Bildes. READ kopiert das Anzeigebild in den Puffer, WRITE kopiert den Puffer auf die Anzeige und CLOSE gibt den Puffer frei und holt den verwendeten Speicher zurück. LOAD lädt eine Bilddatei in den Puffer.

BLIT LOAD und BLIT WRITE funktionieren auf jedem Display, während BLIT und BLIT READ nur auf Displays funktionieren, die transparenten Text unterstützen (d. h. SSD1963, ILI9341, ST7789_320 oder ILI9488 mit angeschlossenem MISO) sowie auf VGA- und HDMI-Displays und allen In-Memory-Framebuffern.

Diese Befehle können verwendet werden, um einen Teil der Anzeige an einen anderen Ort zu kopieren (durch Kopieren in einen Puffer und anschließendes Schreiben an eine andere Stelle), aber eine einfachere Methode ist die Verwendung einer alternativen Version des BLIT-Befehls wie folgt:

```
BLIT x1, y1, x2, y2, w, h
```

Dadurch wird ein Teil des Bildes an der Position x1/y1 an die Position x2/y2 kopiert. w und h geben die Breite und Höhe des zu kopierenden Bildes an. Der Quell- und der Zielbereich können sich überlappen, und der BLIT-Befehl führt die Kopie korrekt aus.

Diese Form des BLIT-Befehls ist besonders nützlich für die Erstellung von Grafiken, die horizontal oder vertikal gescrollt werden können, wenn neue Daten hinzugefügt werden.

Darüber hinaus bietet die Firmware die Befehle BLIT MEMORY, BLIT COMPRESSED, BLIT FRAMEBUFFER und BLIT MERGE. Diese erweiterten Befehle können zum Programmieren von Spielen mit Hunderten von Anzeigeelementen verwendet werden, wie z. B. der Portierung von PETSCII-Robotern auf MMBasic.

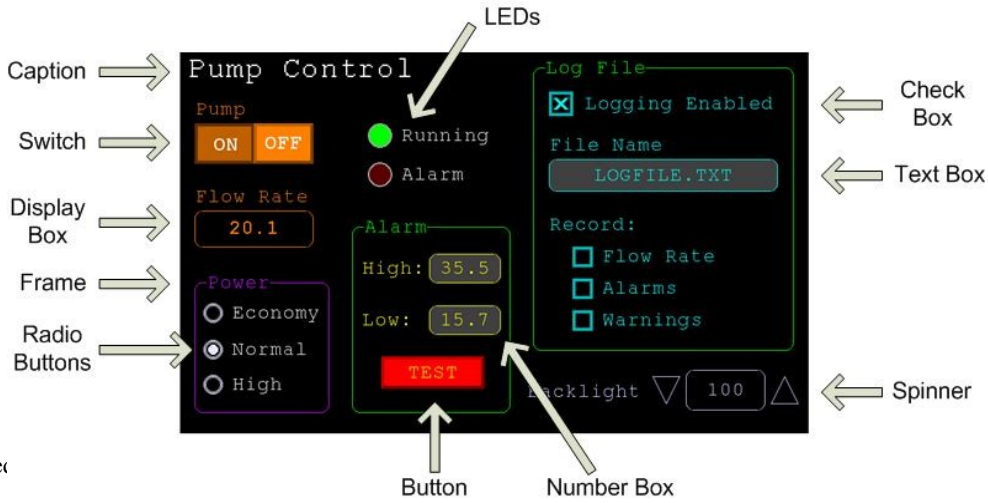
-Bild laden

Mit den Befehlen LOAD IMAGE und LOAD JPG kann ein Bild aus dem Flash-Dateisystem oder von der SD-Karte geladen und auf dem LCD-Display angezeigt werden. Dies kann verwendet werden, um ein Logo zu zeichnen oder den auf dem Display gezeichneten Grafiken einen verzierten Hintergrund hinzuzufügen.

Erweiterte Grafikfunktionen von

NICHT VERFÜGBAR IN DEN VERSIONEN VGA/HDMI UND WEBMITE RP2040

Die PicoMite-Firmware enthält eine Reihe fortschrittlicher Grafikfunktionen, mit denen Programmierer auf einfache Weise berührungsempfindliche Steuerelemente auf einem LCD-Bildschirm erstellen können, wie in der Abbildung dargestellt. Dazu gehören Bildschirmschalter, Schaltflächen, Anzeigeleuchten, Tastaturen usw.



MMBasic zeichnet die Steuerung und animiert sie (d. h., ein Schalter scheint bei Berührung gedrückt zu werden).
Der BASIC-Programmierer muss le festzulegen.

Diese Funktionen erleichtern die Erstellung eines Bedienfelds zur Steuerung beliebiger Funktionen, z. B. einer Drehmaschine, einer Motorsteuerung, einer Heizungsanlage, kleiner industrieller Prozesse usw.

Die erweiterten Grafikfunktionen werden ausführlich im Dokument *„Advanced Graphics Functions.pdf"* beschrieben beschrieben, das in der Firmware-Download-Datei enthalten ist.

3D-Engine

NICHT IN WEBMITE-VERSIONEN VERFÜGBAR

Die 3D-Engine umfasst zehn Befehle zur Bearbeitung von 3D-Bildern, darunter das Einstellen der Kamera, das Erstellen, Ausblenden, Drehen usw. Eine vollständige Beschreibung dieser Befehle und ihrer Verwendung finden Sie im Dokument *„The CMM2 3D engine.pdf*" im PicoMite-Firmware-Download.

Beispiel für LCD-Grafik

Als Beispiel für die Verwendung der einfachen Grafikbefehle zeichnet das folgende Programm eine einfache Digitaluhr auf einem ILI9341-basierten LCD-Display. Das Programm wird beendet und kehrt zur Eingabeaufforderung zurück, wenn der Bildschirm berührt wird.

Zunächst müssen die Anzeige- und Touch-Optionen durch Eingabe der am Anfang dieses Kapitels aufgeführten Befehle konfiguriert werden. Das genaue Format dieser Befehle hängt davon ab, wie Sie das Display angeschlossen haben.

Geben Sie dann das Programm ein und führen Sie es aus:

```
CONST DBlue = RGB(0, 0, 128)                    ' Eine dunkelblaue Farbe
COLOUR RGB(GREEN), RGB(BLACK)                   ' Standardfarben festlegen
FONT 1, 3                                       ' Standardschriftart festlegen

BOX 0, 0, MM.HRes-1, MM.VRes/2, 3, RGB(RED), DBlue DO
   TEXT MM.HRes/2, MM.VRes/4, TIME$, „CM", 1, 4, RGB(CYAN), DBlue TEXT
   MM.HRes/2, MM.VRes*3/4, DATE$, „CM"
   IF TOUCH(X) <> -1 THEN END LOOP
```

Dieses Programm beginnt mit der Definition einer Konstante mit einem Wert, der einer dunkelblauen Farbe entspricht, und legt dann die Standardeinstellungen für die Farben und die Schriftart fest. Anschließend zeichnet es ein Feld mit roten Wänden und einem dunkelblauen Innenraum.

Anschließend tritt das Programm in eine Endlosschleife ein, in der es drei Funktionen ausführt:

1. Es zeigt die aktuelle Uhrzeit innerhalb des zuvor gezeichneten Kastens an. Die Zeichenfolge wird sowohl horizontal als auch vertikal in der Mitte des Kastens zentriert gezeichnet. Beachten Sie, dass der Befehl TEXT sowohl die Standardschriftart als auch die Standardfarben überschreibt, um seine eigenen Parameter festzulegen.
2. Zeichnet das Datum zentriert in der unteren Hälfte des Bildschirms. In diesem Fall verwendet der Befehl TEXT die zuvor festgelegten Standardschriftarten und -farben.

3. Prüft, ob der Bildschirm berührt wurde. Dies wird angezeigt, wenn die Funktion TOUCH(X) einen anderen Wert als -1 zurückgibt. In diesem Fall wird das Programm beendet.

Die Bildschirmanzeige sollte wie folgt aussehen (die in dieser Abbildung verwendete Schriftart ist unterschiedlich):

WiFi- und Internet- sfunktionen

NUR WEBMITE-VERSION

Dieses Kapitel bietet einen Überblick über die WiFi- und Internetfunktionen, die in der WebMite-Version für den Raspberry Pi Pico W und den Raspberry Pi Pico 2 W implementiert sind.

Diese Funktionen sind komplex, daher sollten einige Punkte beachtet werden:

- Die Implementierung der Internetprotokolle beansprucht einen Großteil der Ressourcen des Prozessors (insbesondere RAM), daher sollte die WebMite-Firmware nur dort eingesetzt werden, wo WLAN- und Internetkonnektivität wichtig sind und gewisse Leistungseinbußen im Vergleich zum Standard-Raspberry Pi Pico toleriert werden können.
- Dieses Handbuch beschreibt, wie WebMite mit den WLAN- und Internetprotokollen kommuniziert, und enthält einige einfache Beispiele, geht jedoch nicht auf HTML, TCP und die vielen anderen Protokolle und Konventionen ein. Diese können für Neulinge verwirrend sein, die sich daher mit einigen der vielen im Internet verfügbaren Einführungen vertraut machen sollten.
- Bei der Verarbeitung komplexer Internetprotokolle kann die WebMite-Firmware verwirrt werden und sich aufhängen oder einen Fehler ausgeben und zur Eingabeaufforderung zurückkehren. Um dies zu ermöglichen, sollten Programme die WATCHDOG-Funktion verwenden, um solche Situationen zu beheben. Es wird außerdem empfohlen, dass das Programm von Zeit zu Zeit (z. B. einmal pro Woche) einen Neustart mit CPU RESTART erzwingt, um sicherzustellen, dass die Protokollstacks zurückgesetzt werden.

Verbindung zu einem WLAN- -Netzwerk herstellen

Bevor Sie irgendetwas tun, müssen Sie die WLAN-Funktion auf dem WebMite aktivieren. Dies geschieht mit dem folgenden Befehl, der an der Eingabeaufforderung eingegeben wird:

```
OPTION WIFI ssid, password
```

Dabei ist „ssid" der Name des WLAN-Netzwerks und „password" das Sicherheitspasswort, das für den Zugriff auf das Netzwerk verwendet wird. Beide sind Zeichenfolgen, und wenn Zeichenfolgenkonstanten verwendet werden, sollten sie wie in diesem Beispiel in Anführungszeichen gesetzt werden:

```
OPTION WIFI „MyNetwork“, „secret“
```

Wenn Sie dies eingeben, wird das WebMite neu gestartet, wodurch Sie die USB-Verbindung und den Zugriff auf die Konsole verlieren. Wenn Sie die Verbindung schnell wiederherstellen, wird die folgende Meldung angezeigt:

```
Verbindung zu WLAN wird
hergestellt... Verbunden
192.168.1.52
```

Die in der letzten Zeile angegebene Adresse ist die IP-Adresse, die dem WebMite vom Router zugewiesen wurde und je nach Netzwerk variiert. Wenn der WebMite keine Verbindung herstellen kann, wird folgende Meldung angezeigt:

```
Verbindung zu WLAN
herstellen... Verbindung
fehlgeschlagen.
```

Wenn die Verbindung fehlschlägt, sollten Sie die Konfiguration Ihres WLAN-Routers überprüfen:

- Die WLAN-Sicherheit muss WPA-PSK mit TKIP- oder AES-Verschlüsselung (oder beidem) sein.
- Für den WLAN-Zugang muss ein Passwort festgelegt werden.
- DHCP muss konfiguriert sein.

Dieser Befehl muss nur einmal eingegeben werden und wird bei jedem Neustart Ihres WebMite gespeichert. Sie können die Einstellung mit dem Befehl `OPTION LIST` überprüfen. Wenn eine Verbindung besteht, können Sie die zugewiesene IP-Adresse wie folgt überprüfen:

```
> PRINT MM.Info(IP-Adresse)
192.168.1.52
```

Bei einigen Routern kann es einige Zeit oder mehrere Versuche dauern, bis die Verbindung hergestellt ist. Wenn Sie also OPTION AUTORUN ON verwenden, empfiehlt es sich, ganz am Anfang Ihres Programms Folgendes einzufügen:

```
DO WHILE MM.INFO(IP-ADRESSE) = "0.0.0.0" IF
   TIMER > 5000 THEN CPU RESTART
LOOP
```

Dadurch würde 5 Sekunden lang auf eine Verbindung gewartet und bei weiterhin fehlender Verbindung ein Neustart durchgeführt werden.

Fernzugriff auf die Konsole des WebMite über Telnet

Sie können über Telnet eine Fernverbindung zur Konsole des WebMite über WLAN herstellen. Dies ist praktisch, wenn sich das Gerät an einem schwer zugänglichen Ort befindet. Sobald die Verbindung über Telnet hergestellt ist, können Sie alle Funktionen nutzen, die Sie normalerweise über die USB-Konsole ausführen würden, einschließlich der Ausführung des Editors.

Um diese Funktion zu aktivieren, geben Sie Folgendes in die Befehlszeile ein: `OPTION TELNET CONSOLE ON`

Dies muss nur einmal eingegeben werden und wird bei jedem Neustart Ihres WebMite gespeichert. Dadurch wird auch der WebMite neu gestartet, sodass Sie die Verbindung zur USB-Konsole erneut herstellen müssen.

Tera Term (http://tera-term.en.lo4d.com) unterstützt Telnet und wird für diese Aufgabe empfohlen, da es über eine gute VT100-Emulation verfügt und das XModem-Dateiübertragungsprotokoll unterstützt.

Datei -Übertragung

Dateien können über XModem oder TFTP zum und vom WebMite übertragen werden. XModem wird von Tera Term unterstützt und funktioniert über Telnet genauso wie über eine direkte serielle Verbindung. TFTP ist jedoch viel schneller und zuverlässiger als XModem und daher die empfohlene Methode für die Übertragung von Dateien zum und vom WebMite.

Ein TFTP-Server auf dem WebMite wird automatisch aktiviert, wenn Sie mit einem WiFi-Netzwerk verbunden sind, sodass dort nichts weiter erforderlich ist. Sie benötigen einen TFTP-Client für Ihren PC, und es sind viele verschiedene Implementierungen für Windows, Mac OS und Linux verfügbar. Beachten Sie, dass in vielen Tutorials zu TFTP die Einrichtung eines TFTP-Servers behandelt wird – dies ist nicht erforderlich, Sie benötigen lediglich einen Client.

Unter Windows ist ein TFTP-Client im Betriebssystem enthalten, muss jedoch über die Systemsteuerung aktiviert werden. Wählen Sie dazu:

```
Systemsteuerung -> Programme und Funktionen -> Windows-Funktionen aktivieren oder deaktivieren
```

Scrollen Sie dann in der Liste nach unten und aktivieren Sie `TFTP-Client`.

Um eine Datei von Ihrem PC an den WebMite zu senden, führen Sie in einem Befehls- oder PowerShell-Fenster auf Ihrem Windows-PC folgenden Befehl aus („IP-Addr“ ist die IP-Adresse des WebMite):

```
TFTP -i IP-Addr PUT Dateiname
```

Und um eine Datei vom WebMite auf Ihren PC zu kopieren:

```
TFTP -i IP-Addr GET Dateiname
```

Um die Funktionen des TFTP-Clients aufzulisten, verwenden Sie Folgendes:

```
TFTP -h
```

Eine alternative und einfache grafische Windows-TFTP-Client-Anwendung ist: http://www.3iii.dk/linux/dd-wrt/tftp2.exe

Abrufen der Zeit von

Ein üblicher erster Schritt in einem Programm ist es, die Uhrzeit/das Datum abzurufen und die Uhr im WebMite einzustellen. Diese Aktion bestätigt auch, dass der WebMite eine Verbindung zum Internet herstellen kann.

Das Abrufen der Uhrzeit erfolgt mit dem `WEB` NTP-Befehl wie folgt:

```
WEB NTP [timeoffset [, NTPserver$]]
```

Dabei ist „timeoffset” die Zeitzone, in der Sie sich befinden, und „NTPserver$” der Name oder die IP-Adresse des zu verwendenden Zeitservers. Dieser letzte Parameter ist optional. Wenn er weggelassen wird, verwendet die Firmware einen öffentlichen Zeitserver-Pool. Wenn auch der Parameter „timeoffset” weggelassen wird, wird die Uhr des WebMite auf UTC eingestellt.

Dies ist ein typisches Beispiel für ein Gerät in der Zeitzone von Los Angeles:

```
> WEB NTP -10
ntp-Adresse 27.124.125.251
NTP-Antwort erhalten: 08.03.2023 05:34:57
```

Wenn der WebMite keinen Internetzugang hat, wird eine Fehlermeldung angezeigt. Diese kann mit dem Befehl `ON ERROR SKIP` abgefangen und eine geeignete Maßnahme ergriffen werden (z. B. Neustart oder Anzeige einer Meldung für den Bediener).

Beispiel:

```
ON ERROR SKIP 3 WEB
NTP -10
IF MM.ERRNO THEN WEB NTP -10
IF MM.ERRNO THEN PRINT „Fehler beim Herstellen der Verbindung zum Internet“: CPU RESTART
```

Der Grund, warum wir den Befehl WEB NTP zweimal ausführen, ist, dass der erste Versuch aufgrund eines Zeitfehlers fehlgeschlagen sein könnte (was vorkommen kann) – beim zweiten Versuch sollte er jedoch erfolgreich sein.

Implementierung eines Web- -Servers

Eine häufig gestellte Anforderung ist die Einrichtung eines Webservers, der die vom WebMite gesammelten Daten auf einer benutzerfreundlichen Webseite anzeigt.

Der erste Schritt besteht darin, die Serverfunktion mit diesem Befehl in der Eingabeaufforderung zu konfigurieren:

```
OPTION TCP SERVER PORT nn
```

Dabei ist „nn" die zu verwendende Portnummer (normalerweise 80 für eine Webseite). Der Befehl lautet in der Regel:

```
OPTION TCP SERVER PORT 80
```

Wie bei den anderen oben aufgeführten OPTION-Befehlen muss dieser nur einmal eingegeben werden und wird bei jedem Neustart von WebMite gespeichert. Dadurch wird WebMite neu gestartet, und wenn Sie sich schnell wieder verbinden, wird Folgendes angezeigt (mit einer anderen IP-Adresse):

```
Server wird unter 192.168.1.52 auf Port 80 gestartet
```

Mit dem oben genannten Schritt wurde die WebMite-Firmware für die Unterstützung eines TCP-Servers konfiguriert. In Ihrem Programm müssen Sie den Server mit dem folgenden Befehl starten:

```
WEB TCP INTERRUPT InterruptSub
```

Dabei ist „InterruptSub" der Name Ihrer Subroutine, die immer dann aufgerufen wird, wenn eine Anfrage vom TCP-Server empfangen wird. Diese Subroutine kann den Befehl `WEB TCP READ` verwenden, um die eingehende Anfrage vom Remote-Client zu lesen, und der Befehl `WEB TRANSMIT PAGE` kann verwendet werden, um die angeforderte Webseite zu senden.

Nachfolgend finden Sie beispielsweise das vollständige Programm zur Implementierung einer einfachen Webseite (vergessen Sie nicht, zuerst den Befehl OPTION TCP SERVER zu verwenden):

```
1  DIM buff%(4096/8)
2   WEB TCP INTERRUPT WebInterrupt
3   DO
4       '<hier einige Verarbeitungsschritte durchführen>'
5   LOOP
6
7   SUB WebInterrupt
8       LOCAL a%, p%, t%, s$
9       FOR a% = 1 To MM.INFO(MAX CONNECTIONS)
10          WEB TCP READ a%, buff%()
11          p% = LINSTR(buff%(),"GET")
12          t% = LINSTR(buff%(),"HTTP")
13          Wenn (p% <> 0) Und (t% > p%) Dann
14                 WEB TRANSMIT PAGE a%,"index.html"
15          ENDIF
16      NEXT a%
17  END SUB
```

Im Folgenden wird dieses Programm ausführlich beschrieben:

Zeile 1 Zunächst wird ein 4K-Byte-Puffer für die vom Client eingehende Anfrage erstellt. In diesem Beispiel werden die langen Zeichenfolgenbefehle in MMBasic zur Verarbeitung der Daten verwendet, und dies ist der Puffer dafür (eine Beschreibung langer Zeichenfolgen finden Sie im nächsten Kapitel mit dem Titel *„Lange Zeichenfolgen"*). Die Größe dieses Puffers begrenzt die vom Client empfangene Datenmenge.

Zeile 2 Hiermit wird der Server gestartet und die Interrupt-Subroutine für die Verarbeitung eingehender Anfragen als „`WebInterrupt`" festgelegt.

Zeile 4 Dies ist Ihre Hauptprogrammschleife, die normalerweise Daten sammelt, Ausgänge umschaltet usw. Zeile 7

Dies ist die Subroutine, die jedes Mal aufgerufen wird, wenn eine Anfrage vom Browser des Remote-Clients

gestellt wird.

Zeile 9 Durchlaufen Sie alle eingehenden Verbindungen (es kann mehrere gleichzeitige Verbindungen geben). Zeile 10Lesen

Sie die eingehende Nachricht in den langen Zeichenfolgenpuffer ein.

Zeilen 11 und 12 Ermitteln Sie die Positionen der Schlüsselwörter in der Nachricht.

Zeile 13 Überprüfen Sie, ob die Schlüsselwörter vorhanden sind und in der richtigen Reihenfolge stehen.

Zeile 14 Senden Sie die Seite. Es handelt sich um eine Datei namens index.html, die sich im Standardverzeichnis auf dem internen Flash-Dateisystem oder der SD-Karte befindet. Sie ist im HTML-Format, d. h. sie kann Tags wie `<h1>Dies ist eine Überschrift</h1>` enthalten. Weitere Informationen finden Sie unter *„Eine typische Webseite*" weiter unten.

Einfügen von Daten in die Webseite

In der Regel müssen Sie die vom BASIC-Programm generierten Daten in die zu übertragende Webseite einfügen. Dies ist ganz einfach, indem Sie den Namen der BASIC-Variablen in geschweiften Klammern (d. h. { und }) in den Text der Webseite einfügen.

Wenn Ihr Programm beispielsweise eine Variable namens „`CurrentTemp`" mit dem Wert 24 enthält, die die aktuelle Temperatur angibt, würde die folgende Webseite: `Die Temperatur beträgt {CurrentTemp}` im Browser des Kunden wie folgt angezeigt werden: `Die Temperatur beträgt 24.`

Der Bezeichner zwischen den geschweiften Klammern kann eine Gleitkommazahl, eine Ganzzahl oder eine Zeichenfolge, ein Array-Element und sogar ein Ausdruck (z. B. A + B) sein. Sie können auch Funktionen verwenden. Wenn Sie also eine Gleitkommazahl mit der richtigen Anzahl von Dezimalstellen formatieren möchten, können Sie die Formatierungsfunktion Str$() verwenden. Beachten Sie, dass ein Fehler im Ausdruck einen entsprechenden Fehler verursacht, wenn der Befehl `WEB TRANSMIT PAGE` ausgeführt wird.

Wenn Sie in Ihrer Webseite eine öffnende geschweifte Klammer verwenden müssen, können Sie zwei als Paar verwenden (d. h. {{), die dann in eine einzelne öffnende Klammer umgewandelt werden. Eine schließende geschweifte Klammer ohne öffnende Klammer bleibt unverändert.

Senden mehrerer Seiten

Wenn ein Remote-Client Daten anfordert, ohne eine Seite anzugeben, sendet er die Anfrage als `GET / HTTP`, wobei der Schrägstrich die Standardseite für den Server darstellt (normalerweise index.html). Im obigen Beispiel wurde dies nicht überprüft, sondern für alle Anfragen wurde einfach dieselbe Seite gesendet.

Wenn Ihre Seite jedoch anklickbare Links wie `<a href="page2.html">Nächste Seite</a>` enthält und der Benutzer auf diesen Link klickt, sendet der Remote-Client eine weitere Anfrage mit `GET /page2.html HTTP`.

Dies lässt sich leicht bewerkstelligen, indem die angeforderten Daten zwischen den Schlüsselwörtern GET und HTML überprüft werden. Ersetzen Sie beispielsweise Zeile 15 im obigen Beispiel durch Folgendes (s$ ist die angeforderte Datei):

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5) IF s$ =
"/" THEN
    WEB TRANSMIT PAGE a%,"index.html" ELSE IF
s$ = "/page2.html" THEN
    WEB TRANSMIT PAGE a%,"page2.html" ENDIF
```

Dies kann auf beliebig viele Seiten erweitert werden.

Senden eines Bildes

Sie können ein Bild in Ihre Webseite einfügen, indem Sie den folgenden HTML-Code verwenden: `<img src="pix.jpg"/>`. Wenn der Remote-Browser dies liest, sendet er die folgende Anfrage: `GET /pix.jpg HTTP`. Sie können dann das angeforderte Bild mit dem fett gedruckten Code senden:

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5) IF s$ =
"/" THEN
    WEB TRANSMIT PAGE a%,"index.html" ELSE IF
s$ = "/page2.html" THEN
    WEB TRANSMIT PAGE a%,"page2.html"
ELSE IF s$ = "/pix.jpg" THEN
    WEB TRANSMIT FILE a%,"pix.jpg","image/jpeg"
ENDIF
```

Beachten Sie, dass `pix.jpg` ein JPEG-Bild sein muss, das sich im Standardverzeichnis des internen Flash-Dateisystems oder auf der SD-Karte befindet. Der WebMite ist kein schneller Server, daher sind kleine und einfache Bilder vorzuziehen.

Der Parameter „image/jpeg“ ist als MIME-Typ bekannt, und es gibt viele verschiedene Typen. Andere gängige Bildtypen sind image/bmp, image/png und image/gif.

Antwort „Seite nicht gefunden“ (404)

Wenn ein Remote-Client eine Seite oder Datei anfordert, die von Ihrem Programm nicht unterstützt wird, können Sie den Befehl WEB `TRANSMIT CODE` verwenden, um wie folgt einen 404-Fehler zu senden:

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5) IF s$ =
"/" THEN
    WEB TRANSMIT PAGE a%,"index.html" ELSE IF
s$ = "/page2.html" THEN
    WEB TRANSMIT PAGE a%,"page2.html" ELSE IF
s$ = "/pix.jpg" THEN
    WEB TRANSMIT FILE a%,"pix.jpg","image/jpeg" ELSE
    WEB TRANSMIT CODE a%, 404
ENDIF
```

Live-Grafikdaten in einer WEB -Seite

Die Darstellung von Zahlen und Text auf einer Webseite ist nützlich, aber oft möchte man auch grafische Elemente wie Kreisdiagramme, Liniendiagramme, historische Trends usw. einfügen, die aus den vom Programm gesammelten Daten abgeleitet wurden. Dies kann auf Umwegen erreicht werden, indem man WebMite mit einem virtuellen Anzeigefeld konfiguriert, dann auf diesem

Display mit den Standard-Zeichenbefehlen (Pixel, Linie, Kreis usw.) zu zeichnen und als BMP-Bild zu speichern. Diese Datei kann dann als Bild in die Webseite eingebunden werden. Im Einzelnen läuft dieser Prozess wie folgt ab:

Zunächst muss ein virtuelles Display konfiguriert werden. Es stehen zwei zur Auswahl: VIRTUAL_C ist ein Bild mit einer Auflösung von 320 x 240 Pixeln und 16 Farben, VIRTUAL_M ist ein monochromes Bild mit einer Auflösung von 640 x 480 Pixeln. Beispiel:

```
OPTION LCDPANEL VIRTUAL_C
```

Wie bei den anderen OPTION-Befehlen muss dieser Befehl an der Eingabeaufforderung eingegeben werden, führt zu einem Neustart und muss nur einmal eingegeben werden.

Anschließend können Sie in Ihrem Programm mit den im Kapitel *„Grafikbefehle und -funktionen"* beschriebenen Befehlen Bilder und Text auf diesem „Display" zeichnen. Beispiel:

```
CIRCLE 100, 100, 50, 1, 1, RGB(rot), RGB(blau)
LINE 10, 10, 200, 200, 1, RGB(yellow)
```

Wenn Sie fertig sind, speichern Sie dieses Bild:

```
KOMPRIMIERTES BILD „graph.bmp“ SPEICHERN
```

Innerhalb der Webseite, die Sie bereitstellen, können Sie dieses Bild mit dem folgenden HTML-Code einfügen:

```
<img src="graph.bmp"/>
```

Schließlich müssen Sie in Ihrem BASIC-Programm dafür sorgen, dass diese Datei gesendet wird, wenn die Webseite wie oben beschrieben vom Remote-Browser geladen wird (siehe *Senden eines Bildes*). Fügen Sie beispielsweise Folgendes in die oben beschriebene ELSE IF-Kette ein:

```
ELSE IF s$ = "/graph.bmp " THEN
  WEB TRANSMIT FILE a%, "graph.bmp", "image/bmp"
```

Ihr BASIC-Programm muss diese Datei aktualisieren, wenn neue Daten aufgezeichnet werden. Da dies jedoch nur erforderlich ist, wenn der Remote-Browser das Bild anfordert, sollte dies keine übermäßige Abnutzung des Flash-Speichers verursachen.

Ein vollständiger Allzweck- -Server

Auf den vorangegangenen Seiten wurden die einzelnen Komponenten eines Webservers beschrieben. Nachfolgend finden Sie ein Beispiel für einen vollständigen und integrierten Allzweckserver, der die meisten Anfragen eines Browsers bearbeiten kann. Er überprüft die Erweiterung der angeforderten Datei und sendet die angeforderten Daten dann mit dem entsprechenden WEB TRANSMIT-Befehl. Er kann als Drop-in-Modul für jedes Projekt verwendet werden, bei dem WebMite als Webserver benötigt wird.

```
WEB TCP INTERRUPT WebInterrupt DO
   '<hier einige
Verarbeitungsschritte durchführen>'
LOOP

' Subroutine zur Bearbeitung aller Webserver-
Anfragen SUB WebInterrupt
  LOCAL a%, p1%, p2%, file$, buff%(4096/8) FOR a% =
  1 To MM.INFO(MAX CONNECTIONS)
    WEB TCP READ a%, buff%() P1% =
    LINSTR(buff%(),"GET") P2% =
    LINSTR(buff%(),"HTTP")
    Wenn (p1% <> 0) Und (p2% <> 0) Und (p2% > p1%) Dann
      file$ = LCASE$(LGetStr$(buff%(), p1% + 4, p2% - p1% - 5)) IF file$ =
      "/" THEN file$ = "/index.html"
      BEI FEHLER ÜBERSPRINGEN
      OPEN file$ FOR INPUT AS #1             ' Überprüfen, ob die Datei
      existiert IF MM.ERRNO THEN WEB TRANSMIT CODE a%, 404 : CONTINUE FOR
      CLOSE #1
      WÄHLEN FALL RIGHT$(file$, 4)
        CASE "html", ".htm", ".txt" WEB
          TRANSMIT PAGE a%, file$
        CASE ".bmp", ".png", ".gif"
          WEB TRANSMIT FILE a%, file$, "image/" + RIGHT$(file$, 3) CASE
        ".jpg", "jpeg"
          WEB TRANSMIT FILE a%, file$, "image/jpeg" CASE
        ".ico"
          WEB TRANSMIT FILE a%, file$, "image/vnd.microsoft.icon" END SELECT
    ENDIF
  NEXT a%
END SUB
```

Beachten Sie, dass alle Dateien im Stammverzeichnis des Flash-Dateisystems oder der SD-Karte gespeichert werden müssen und ihre Namen ausschließlich Kleinbuchstaben enthalten dürfen (das Flash-Dateisystem unterscheidet zwischen Groß- und Kleinschreibung).

Eine typische WEB- -Seite

Die über den Befehl WEB TRANSMIT PAGE zu übertragende WEB-Seite muss gemäß dem HTML-Standard erstellt werden. Sie kann so einfach wie eine einzelne Textzeile ohne Formatierung sein, z. B.:

```
Die Temperatur beträgt {CurrentTemp}
```

Oder Sie können eine einfache Formatierung einfügen:

```
<title>WebMite</title>
<h2>Temperaturmonitor</h2> Die
Temperatur beträgt {CurrentTemp}
```

Oder Sie können eine komplexe Seite senden. In der Regel haben diese Seiten einen Kopf- und einen Hauptteil, unterteilen den Text in Absätze und verwenden das Break-Tag für Abstände. Dies ist das Grundgerüst einer solchen Seite:

```
<html>
<head>
<title>WebMite</title>
</head>
<body>
<h1>Dies ist eine Überschrift</h1>
<br>
<p>Die Temperatur beträgt {CurrentTemp}</p>
</body>
</html>
```

Im Internet gibt es viele Ressourcen, die Tutorials in HTML für Anfänger anbieten. Ein typisches Beispiel ist http://www.simplehtmlguide.com/. Es gibt auch zahlreiche WISWIG-HTML-Editoren. Zum Beispiel: https://onlinehtmleditor.dev/

Eingabefelder und Steuerelemente „ ”

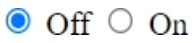
Mit HTML-Eingabefeldern können Sie Schaltflächen, Kontrollkästchen, Optionsfelder usw. auf Ihrer Webseite platzieren, über die der Benutzer Anfragen an den WebMite senden kann. Auf diese Weise kann der Benutzer aus der Ferne Dinge ein- und ausschalten, Steuerungsparameter einstellen und vieles mehr – alles über die vom WebMite bereitgestellte Webseite.

Dazu müssen Sie ein *HTML-Formular* in die Webseite einfügen, das ein oder mehrere Eingabefelder enthält. Es stehen viele Arten von Eingabefeldern zur Auswahl (siehe https://www.w3schools.com/tags/att_input_type.asp), aber in unserem einfachen Beispiel verwenden wir zwei Optionsfelder, um ein fiktives Gerät ein- und auszuschalten.

Der folgende Code muss in die Standard-Webseite (d. h. index.html) eingefügt werden:

```
<form method='get'>
     <input name='RB' type='radio' value='OFF' {offb$} onClick='this.form.submit()'> Aus
     <input name='RB' type='radio' value='ON' {onb$} onClick='this.form.submit()'> Ein
</form>
```

Dadurch werden zwei Optionsfelder erstellt, die im Browser wie folgt angezeigt werden:



Beachten Sie, dass der obige HTML-Code zwei BASIC-Variablen (`offb$` und `onb$`) enthält, die durch den Befehl WEB TRANSMIT PAGE ersetzt werden. Diese Variablen steuern, wie die Schaltfläche angezeigt wird. Wenn sie auf eine leere Zeichenfolge gesetzt sind, wird die Schaltfläche als nicht aktiviert angezeigt. Wenn sie auf die Zeichenfolge „`checked='checked'`” gesetzt sind, wird die Schaltfläche als aktiviert angezeigt. Diese Variablen sollten zu Beginn der Programmausführung initialisiert werden.

Wenn der Benutzer auf die erste Schaltfläche klickt, sendet der Browser eine Anfrage mit folgendem Inhalt: GET /?RB=OFF HTML. Wenn die zweite Schaltfläche angeklickt wird, lautet die Anfrage: GET /?RB=ON HTML.

In der ELSE IF-Kette in der TCP-Interrupt-Subroutine können wir auf diese Anfragen reagieren:

```
ELSE IF s$ = "/?RB=OFF"
     ' <Code zum Ausschalten des Geräts hier einfügen>
     offb$ = "checked='checked'" : onb$ = ""
     WEB TRANSMIT PAGE a%, "index.html" ELSE IF
s$ = "/?RB=ON"
     ' <Code zum Einschalten des Geräts hier einfügen>
     offb$ = "" : onb$ = "checked='checked'" WEB
     TRANSMIT PAGE a%, "index.html"
```

Im Wesentlichen schaltet dieser Code das Gerät wie gewünscht ein oder aus, setzt die Variablen `onb$` und `offb$,` um den neuen Status der Schaltflächen widerzuspiegeln, und sendet dann die gesamte Webseite zurück an den Browser.

In diesem Beispiel wurden viele Details ausgelassen, und Sie können es extrem kompliziert gestalten, wenn Sie möchten, mit mehreren Eingaben über Schaltflächen, Texteingaben, Dropdown-Listen, Passwortfeldern, Anfragen für Datei-Uploads und vielem mehr. Dazu müssen Sie sich jedoch tiefer mit HTML-Codierung befassen. Ein Beispiel hierfür finden Sie im Projekt *„Garden Watering Controller"* unter: https://geoffg.net/retic.html

Implementierung eines TCP- -Clients

Der WebMite kann auch als TCP-Client fungieren, um Daten von einem Remote-Server anzufordern. Dies wird mit drei Befehlen verwaltet. Der erste lautet:

```
WEB OPEN TCP CLIENT Domain$, PortNumber
```

Dadurch wird eine TCP-Verbindung zu `Domain$` (z. B. „openweathermap.org") unter Verwendung der angegebenen `PortNumber` (normalerweise 80 für eine Webseite).

Wenn die Verbindung geöffnet ist, können Sie mit diesem Befehl eine oder mehrere Anfragen senden:

```
WEB TCP CLIENT REQUEST query$, inbuf [, timeout]
```

Die zu sendende Anfrage lautet „query$" und die Antwort wird in „inbuf" gespeichert, bei dem es sich normalerweise um eine lange Zeichenfolgenvariable wie `buff%(4096/8)` handelt. Die Größe dieses Puffers (in Byte) begrenzt die vom Server empfangene Datenmenge und sollte erhöht werden, wenn mehr Daten erwartet werden.

„timeout" ist optional und gibt die Zeitüberschreitung in Millisekunden an.

Wenn Sie auf eine Website zugreifen, kann „query$" etwas so Einfaches wie „`GET / HTTP`" sein, wodurch die Standardseite dieser Website abgerufen wird. Ihr Programm ist dann dafür verantwortlich, die gewünschten Daten aus der Antwort herauszufiltern.

Schließlich schließen Sie die Verbindung mit:

```
WEB CLOSE TCP CLIENT
```

Das folgende Beispiel ist ein vollständiges Programm (unter Verwendung dieser Befehle), um die aktuelle Temperatur für die Stadt Paris von openweathermap.com abzurufen. Damit dies funktioniert, benötigen Sie ein (kostenloses) Konto bei Open Weather Map (https://openweathermap.org), das einen API-Schlüssel enthält. Dabei handelt es sich um eine 32-stellige Hexadezimalzahl, mit der Sie die Abfrage durchführen können. Diese Zahl sollte den Dummy-Schlüssel in der ersten Zeile ersetzen.

```
CONST Key = "nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn"
CONST Query = "GET /data/2.5/weather?q=Paris,fra&APPID="+Key+Chr$(13)+Chr$(10) DIM
buff%(4096/8)

WEB OPEN TCP CLIENT „api.openweathermap.org", 80 WEB TCP
CLIENT REQUEST Query, buff%()
WEB CLOSE TCP CLIENT
temp = VAL(JSON$(buff%(), "main.temp"))
PRINT „Die aktuelle Temperatur in Paris beträgt:" temp - 273
```

Die Funktion JSON() wird verwendet, um den gewünschten Wert aus der Antwort im JSON-Format (JavaScript Object Notation) zu extrahieren.

Wenn dieses Programm ausgeführt wird, sollte etwa Folgendes angezeigt werden:

```
Verbunden
Aktuelle Temperatur in Paris: 13,54
```

Verwendung von UDP

Das User Datagram Protocol (UDP) ist ein schnelles und leichtgewichtiges Protokoll, das zum Senden von Nachrichten zwischen Geräten verwendet wird. Es kann unzuverlässig sein, da es nicht garantiert, dass Datenpakete zugestellt werden oder in der richtigen Reihenfolge ankommen. Es wird jedoch häufig für zeitkritische Anwendungen verwendet, bei denen Geschwindigkeit Vorrang vor Zuverlässigkeit hat.

Die Einrichtung eines UDP-Servers ähnelt der Einrichtung eines TCP-Servers. Zunächst wird UDP mit der folgenden Option in der Eingabeaufforderung aktiviert (dies führt auch zu einem Neustart):

```
OPTION UDP SERVER PORT port_nbr
```

In Ihrem Programm müssen Sie den Server mit dem folgenden Befehl starten:

```
WEB UDP INTERRUPT InterruptSub
```

Wobei „InterruptSub" der Name Ihrer Subroutine ist, die jedes Mal aufgerufen wird, wenn eine Nachricht vom UDP-Server empfangen wird. In diesem Fall wird die IP-Adresse des sendenden Geräts in der schreibgeschützten Variablen MM.ADDRESS$ und die empfangene Nachricht in MM.MESSAGE$ gespeichert.

Sie können eine UDP-Nachricht mit folgendem Befehl senden:

```
WEB UDP send ip_address$, port_nbr, message$
```

Die folgende Demonstration verwendet zwei WebMites, die mit OPTION WIFI im selben Netzwerk konfiguriert sind. Vor dem Start müssen beide WebMites mit OPTION UDP SERVER PORT 77 konfiguriert werden.

Notieren Sie sich die IP-Adresse des ersten WebMites und geben Sie das folgende Programm ein:

```
WEB UDP interrupt myint Do
    '<hier einige Verarbeitungsschritte durchführen>'
Loop

' interrupt subroutine Sub
myint
    Drucken "Empfangen " + mm.message$ + " von " + mm.address Pause 100
    WEB UDP send mm.address$, 77, „WebMite #1 message” End Sub
```

Führen Sie dann das Programm aus, das zunächst in einer Schleife nichts tut.

Geben Sie auf dem zweiten WebMite das folgende Programm ein und ersetzen Sie dabei die IP-Adresse durch die des ersten WebMite. Führen Sie dann das Programm aus:

```
WEB UDP interrupt myint
WEB UDP send „192.168.1.127”, 77, „Starting UDP echo” Do
    '<hier einige Verarbeitungsschritte durchführen>'
Loop

' interrupt subroutine Sub
myint
    Drucken "Empfangen " + mm.message$ + " von " + mm.address$ Pause 100
    WEB UDP send mm.address$, 77, „WebMite #2 message” End Sub
```

Das Ergebnis ist ein sehr schneller Ping-Pong-Austausch von UDP-Nachrichten zwischen den beiden WebMites.

-E-Mails senden

Wenn Sie ein Remote-Gerät wie den WebMite haben, ist es nützlich, dass es E-Mails senden kann, um Fehler zu melden, über seinen Status zu berichten und so weiter. Der WebMite kann dies mithilfe des SMTP-Protokolls tun, um sich mit einem Server zu verbinden, der die E-Mail dann an ihren Bestimmungsort weiterleitet.

Das folgende Beispiel nutzt den kostenlosen SMTP-Relay-Dienst von SMTP2GO, der 1000 E-Mails pro Monat zulässt (reichlich genug für WebMite). Beachten Sie, dass Registrierungsanfragen von Personen mit einer generischen kostenlosen E-Mail-Adresse (z. B. xxx@gmail.com) nicht akzeptiert werden.

Um loszulegen, müssen Sie sich bei SMTP2GO (https://www.smtp2go.com/) kostenlos anmelden, einen verifizierten Absender registrieren und einen zugehörigen Benutzernamen und ein Passwort erstellen. Beide müssen dann in Base64-kodierte Zeichenfolgen umgewandelt werden (die folgende Website übernimmt dies für Sie: https://www.base64encode.org).

Der Base64-kodierte Benutzername sollte verwendet werden, um die Zeichenfolge `nnnnnnnnnn` in der ersten Zeile des folgenden Programms zu ersetzen, während das Base64-kodierte Passwort verwendet werden sollte, um xxxxxxxxxxxx in der zweiten Zeile zu ersetzen. Die anderen vier Zeilen am Anfang des Programms sollten ebenfalls durch Ihre Daten ersetzt werden:

```
CONST userBase64$ = "nnnnnnnnnnnn" CONST
paswdBase64$ = "xxxxxxxxxxxx" CONST
mailfrom$ =              "from@server.com "
CONST mailto$ =          "to@server.com "
CONST subject$ =         „Test-E-Mail” CONST
message$ =               "Test von SMTP2GO"

CONST cr = Chr$(13)+Chr$(10)
DIM buff%(4096/8), body$

body$ = "Von: " + mailfrom$ + cr + "An: " + mailto$ + cr body$ =
body$ + "Betreff: " + subject$ + cr + cr
body$ = body$ + message$ + cr + "." + cr

WEB OPEN TCP CLIENT „mail.smtp2go.com”, 2525 WEB TCP
CLIENT REQUEST „EHLO” + cr, buff%()
WEB-TCP-CLIENT-ANFORDERUNG „AUTH LOGIN” + cr, buff%()
WEB-TCP-CLIENT-ANFORDERUNG userBase64$ + cr, buff%() WEB-
TCP-CLIENT-ANFORDERUNG paswdBase64$ + cr, buff%()
WEB-TCP-CLIENT-ANFORDERUNG „MAIL FROM: “ + mailfrom$ + cr, buff%()
```

```
WEB-TCP-CLIENT-ANFORDERUNG „RCPT TO: “ + mailto$ + cr, buff%() WEB-
TCP-CLIENT-ANFORDERUNG „DATA“ + cr, buff%()
PAUSE 300
WEB TCP CLIENT REQUEST body$, buff%() PAUSE 300
WEB CLOSE TCP CLIENT
IF LINSTR(buff%(), "250 OK") = 0 THEN
  PRINT "E-Mail-Versand
fehlgeschlagen" Else
  Print „E-Mail erfolgreich
gesendet” EndIf
```

Beachten Sie, dass die im obigen Programm verwendete E-Mail-Adresse `mailfrom$` mit der Adresse übereinstimmen MUSS, die Sie bei der Registrierung des verifizierten Absenders bei SMTP2GO angegeben haben. Wenn sie nicht übereinstimmen, lehnt SMTP2GO die E-Mail ab (dies ist eine Anti-Spam-Maßnahme).

Heutzutage ist die Verwendung eines SMTP-Relay-Dienstes aufgrund der Unterschiede im SMTP-Protokoll der einzelnen Anbieter und der verschiedenen Schutzmaßnahmen zur Reduzierung der Spam-Menge kompliziert. Dieses Beispiel bezieht sich speziell auf das von SMTP2GO verwendete SMTP-Protokoll, es können jedoch auch andere Dienste verwendet werden, wobei das Programm jedoch an die jeweilige Version des SMTP-Protokolls angepasst werden muss.

Base-64- -Kodierung

Base64 ist ein System zur Konvertierung von Binärdaten in eine Textzeichenfolge, die nur ASCII-Zeichen verwendet (d. h. es gibt keine Steuerzeichen). Es wurde entwickelt, um das Senden von Binärdaten über das Internet mit Protokollen zu vereinfachen, die keine Binärdaten akzeptieren, und viele Protokolle erfordern seine Verwendung.

Die MATH BASE64-Kodierungs-/Dekodierungsfunktion übernimmt die Kodierung und Dekodierung für Sie (die vollständige Syntax finden Sie in der detaillierten Funktionsliste). Eine typische Anwendung ist das oben genannte Programm zum Versenden von E-Mails. Anstatt einen externen Dienst zur Konvertierung des Benutzernamens und des Passworts in Base64 zu verwenden, können Sie dies im Programm mit dieser Funktion tun.

Beispiel:

```
DIM n, userBase64$, paswdBase64$
CONST user$ = "MeinBenutzername"        ' Benutzername im Klartext
CONST paswd$ = "MyPassword"             ' Passwort im Klartext
...
' Benutzername und Passwort verschlüsseln
n = MATH(BASE64 ENCODE, user$, userBase64$) n =
MATH(BASE64 ENCODE, paswd$, paswdBase64$)
...
```

MQTT- -Client

MQTT ist ein Protokoll, mit dem Clients wie WebMite Nachrichten auf einem Server (auch MQTT-Broker genannt) veröffentlichen oder abrufen können. Dies ähnelt einem Schwarzen Brett oder einem webbasierten Forum, in dem Nutzer Nachrichten veröffentlichen, die andere später nach Belieben lesen können – der Hauptunterschied besteht darin, dass MQTT für die Kommunikation zwischen Maschinen konzipiert ist.

Ein typisches Beispiel wäre ein batteriebetriebenes WebMite, das den Wasserstand in einem Stausee überwacht. Zweimal täglich würde es sich einschalten, die Messung durchführen, das Ergebnis an einen MQTT-Broker senden und sich wieder ausschalten. Ein Client-Programm (beispielsweise auf einem PC) könnte diese Nachrichten später lesen, die Ergebnisse anzeigen und grafisch darstellen.

Es gibt viele kostenlose Broker. Suchen Sie mit Google nach „kostenloser MQTT-Broker“. Der WebMite

verwendet fünf Befehle zum Senden oder Abrufen von Nachrichten:

```
WEB MQTT CONNECT Verbindung zu einem MQTT-Broker herstellen.
WEB MQTT PUBLISH Veröffentlicht Inhalte zu einem MQTT-Thema (d. h. sendet eine
Nachricht). WEB MQTT SUBSCRIBE  Abonnieren eines MQTT-Themas (d. h. Abrufen von
Nachrichten). WEB MQTT UNSUBSCRIBE     Abonnement eines MQTT-Themas kündigen.
WEB MQTT CLOSE       Schließen Sie die MQTT-Verbindung.
```

Ping

Das WebMite reagiert auf eine Ping-Nachricht, sodass Sie überprüfen können, ob es aktiv und erreichbar ist. Wenn es mit dem öffentlichen Internet verbunden ist, kann ein kostenloser Dienst wie https://uptimerobot.com/ verwendet werden, um Sie zu benachrichtigen, wenn es nicht mehr läuft.

Streaming von Audio-

Die Befehle WEB OPEN TCP STREAM und WEB TCP CLIENT STREAM können zusammen mit dem Befehl PLAY STREAM auf sehr einfache Weise eine grundlegende Internetradiofunktion implementieren. Dies wird im folgenden Code demonstriert, der das britische Programm ClassicFM empfängt. Hinweis: Für dieses Programm ist ein VS1053-Audio-Codec erforderlich.

```
Option escape Option
default none
' Anfrage für die Radioseite (ClassicFM) erstellen Dim
a$="ice-the.musicradio.com"
Dim q$="GET "
Inc     q$,"/ClassicFMMP3"
Inc   q$,"   HTTP/1.1\r\n"
Inc q$,"Host: "
Inc q$,a$
Inc q$,"\r\nConnection: close\r\n\r\n"

'Erstellen  Sie  einen  Ringpuffer  zum  Lesen  des  Internet-Streams  und
'Lesen und Schreiben von Zeigern
Dim buff%(4095),w%,r%

' VS1053 konfigurieren und anweisen, aus dem Ringpuffer abzuspielen Stream
abspielen buff%(), r%, w%

'Internetradio-Website öffnen WEB
open tcp stream a$,80

Senden Sie die Anfrage zum Starten des Streams unter Verwendung des angegebenen
Ringpuffers WEB TCP CLIENT STREAM q$, buff%(), r%, w%

'Lehnen Sie sich zurück
und hören Sie zu. Tun
Sie Folgendes: Pause
500: Schleife
```

Lange Zeichenfolgen in

Lange Zeichenfolgen sind eine Reihe von Befehlen und Funktionen, mit denen MMBasic Zeichenfolgen unbegrenzter Länge bearbeiten kann. Sie sind besonders nützlich bei der Verarbeitung von Daten, die über WLAN und das Internet gesendet werden. Standardzeichenfolgen in MMBasic sind auf eine maximale Länge von 255 Zeichen begrenzt. Lange Zeichenfolgen duplizieren diese Funktionen, funktionieren jedoch mit Zeichenfolgen beliebiger Länge, die nur durch die verfügbare RAM-Kapazität begrenzt sind.

Variablen für die Lang

Variablen zum Speichern langer Zeichenfolgen müssen als Integer-Arrays definiert werden. Die Routinen für lange Zeichenfolgen speichern keine Zahlen in diesen Arrays, sondern verwenden sie lediglich als Speicherblöcke zum Speichern langer Zeichenfolgen.

Bei der Erstellung dieser Arrays sollten sie als eindimensionale Integer-Arrays definiert werden, wobei die Anzahl der Elemente auf die Anzahl der Zeichen festgelegt wird, die für die maximale Zeichenfolgenlänge erforderlich sind, geteilt durch acht. Der Grund für die Division durch acht ist, dass jede Ganzzahl in einem MMBasic-Array acht Bytes belegt.

Im Folgenden finden Sie ein Beispiel für die Deklaration von drei langen Zeichenfolgenvariablen, die jeweils bis zu 2048 Zeichen aufnehmen können:

```
CONST MaxLen = 2048
DIM INTEGER Str1(MaxLen/8), Str2(MaxLen/8), Str3(MaxLen/8)
```

Diese enthalten bei ihrer Erstellung leere Zeichenfolgen (d. h. ihre Länge ist Null). Wenn diese Variablen an die Funktionen für lange Zeichenfolgen übergeben werden, sollten sie als Variablenname gefolgt von leeren Klammern eingegeben werden. Beispiel:

```
LONGSTRING COPY Str1(), Str2()
```

Lange String-Variablen können als Argumente an benutzerdefinierte Unterprogramme und Funktionen übergeben werden. Beispiel:

```
Sub MySub longarg() AS INTEGER
   PRINT "Die Länge der langen Zeichenfolge beträgt"
LLEN(longarg()) END SUB
```

Und es könnte wie folgt aufgerufen werden:

```
MySub str1()
```

Befehle für lange Zeichenfolgen

Diese sind im Abschnitt *„Befehle* und *Funktionen"* dieses Handbuchs ausführlich dokumentiert. Die Befehle lauten: LONGSTRING

AES128 ENCRYPT/DECRYPT	Verschlüsselt oder entschlüsselt eine lange Zeichenfolge
LONGSTRING APPEND array%(), string$	Fügt eine normale Zeichenfolge an eine lange Zeichenfolge an
LONGSTRING BASE64 ENCODE/DECODE	Kodiert oder dekodiert eine lange Zeichenfolge unter
Verwendung von Base 64. LONGSTRING CLEAR array%()	Löscht (d. h. setzt auf leer) eine lange Zeichenfolge
LONGSTRING COPY dest%(), src%()	Kopiert eine lange Zeichenfolge
LONGSTRING CONCAT dest%(), src%()	Verknüpft zwei lange Zeichenfolgen LONGSTRING
LCASE array%()	Konvertiert eine lange Zeichenfolge in
Kleinbuchstaben LONGSTRING LEFT dest%(), src%(), nbr	Die ersten nbr Zeichen einer langen Zeichenkette
abrufen LONGSTRING LOAD array%(), nbr, string$	Zeichen in eine lange Zeichenkette kopieren
LONGSTRING MID dest%(), src%(), start, nbr	Zeichen aus der Mitte einer langen Zeichenfolge abrufen
LONGSTRING PRINT [#n,] src%() [;]	Eine lange Zeichenfolge ausgeben
LONGSTRING REPLACE array%() , string$, start	Zeichen in einer langen Zeichenkette ersetzen LONGSTRING
RESIZE addr%(), nbr	Länge einer langen Zeichenfolge festlegen
LONGSTRING RIGHT dest%(), src%(), nbr	Die rechten nbr Zeichen aus einer langen Zeichenkette
holen LONGSTRING SETBYTE addr%(), nbr, data	Ein Byte in einer langen Zeichenkette setzen
LONGSTRING TRIM array%(), nbr	Zeichen am Anfang einer langen Zeichenfolge
entfernen LONGSTRING UCASE array%()	Konvertieren Sie eine lange Zeichenfolge in
Großbuchstaben LMID(array%(),start [,num])=string$	Text in eine lange Zeichenfolge einfügen/ersetzen

Funktionen für lange Zeichenfolgen

r = LGETBYTE(array%(), n) Gibt den Wert eines Bytes in einer langen Zeichenfolge zurück

r$ = LGETSTR$(array%(), start, length) Gibt einen Teil einer langen Zeichenfolge als normale Zeichenfolge zurück. r = LINSTR(array%(), search$ [,start] [,size]) Gibt die Position einer Zeichenfolge in einer langen Zeichenfolge zurück r = LLEN(array%()) Gibt die Länge einer langen Zeichenfolge zurück

r= LINPUT(array%(),fnbr,nbr) Liest nbr Bytes aus einer Datei und gibt die gelesene Anzahl zurück. MATH(BASE64 ENCODE/DECODE) Kodiert oder dekodiert Daten unter Verwendung der Basis 64

MMBasic- -Eigenschaften

Namenskonventionen für Funktionen

Bei Befehlsnamen, Funktionsnamen, Bezeichnungen, Variablennamen usw. wird nicht zwischen Groß- und Kleinschreibung unterschieden, sodass „Run" und „RUN" gleichbedeutend sind und „dOO" und „Doo" sich auf dieselbe Variable beziehen.

Der Typ einer Variablen kann im DIM-Befehl oder durch Hinzufügen eines Suffixes am Ende des Variablennamens angegeben werden. Das Suffix für eine Ganzzahl lautet beispielsweise „%". Wenn also eine Variable namens nbr% automatisch erstellt wird, handelt es sich um eine Ganzzahl. Es gibt drei Arten von Variablen:

1. Gleitkomma. Diese können Zahlen mit Dezimalpunkt und Bruchteil (z. B. 45,386) sowie sehr große Zahlen speichern. Allerdings verlieren sie an Genauigkeit, wenn mehr als 14 signifikante Stellen gespeichert oder verarbeitet werden. Das Suffix lautet „!" und Gleitkomma ist die Standardeinstellung, wenn eine Variable ohne Suffix erstellt wird.
2. 64-Bit-Ganzzahl. Diese können Zahlen mit bis zu 19 Dezimalstellen ohne Genauigkeitsverlust speichern, jedoch keine Bruchteile (d. h. den Teil nach dem Dezimalpunkt). Das Suffix für eine Ganzzahl ist „%".
3. Zeichenfolgen. Diese speichern eine Folge von Zeichen (z. B. „Tom"). Das Suffix für eine Zeichenfolge ist das Symbol „$" (z. B. name$, s$ usw.). Zeichenfolgen können bis zu 255 Zeichen lang sein.

Variablennamen und Bezeichnungen können mit einem Buchstaben oder einem Unterstrich beginnen und beliebige Buchstaben oder Ziffern, den Punkt (.) und den Unterstrich (_) enthalten. Sie können bis zu 31 Zeichen lang sein. Ein Variablenname oder eine Bezeichnung darf nicht mit einem Befehl oder einer Funktion oder einem der folgenden Schlüsselwörter identisch sein: THEN, ELSE, TO, STEP, FOR, WHILE, UNTIL, MOD, NOT, AND, OR, XOR, AS. Beispielsweise ist step = 5 unzulässig.

Informationen zu Dateinamen finden Sie im Abschnitt *„MMBasic-Unterstützung für Flash- und SD-Karten-Dateisysteme*".

Konstanten

Numerische Konstanten können mit einer Ziffer (0-9) für eine Dezimalkonstante, mit &H für eine Hexadezimalzahl, mit &O für eine Oktalzahl oder mit &B für eine Binärzahl beginnen. Beispielsweise entspricht &B1000 der Dezimalkonstante 8. Konstanten, die mit &H, &O oder &B beginnen, werden immer als 64-Bit-Ganzzahlkonstanten behandelt.

Dezimalen können mit einem Minuszeichen (-) oder Pluszeichen (+) beginnen und mit „E" gefolgt von einer Exponentialzahl enden, um die Exponentialdarstellung anzuzeigen. Beispielsweise entspricht 1,6E+4 dem Wert 16000. Wenn die Dezimalzahl einen Dezimalpunkt oder einen Exponenten enthält, wird sie als Gleitkommakonstante behandelt, andernfalls als 64-Bit-Ganzzahlkonstante.

Zeichenfolgenkonstanten werden von doppelten Anführungszeichen (") umgeben. Beispiel: „Hello World".

Implementierungs e Eigenschaften

Maximales Programm – siehe Tabelle. Beachten Sie, dass MMBasic das Programm beim Speichern im Flash-Speicher tokenisiert, sodass die endgültige Größe im Flash-Speicher von der Größe des Klartexts abweichen kann.

Die maximale Länge einer Befehlszeile beträgt 255 Zeichen. Die maximale

Länge eines Variablennamens oder einer Bezeichnung beträgt 31 Zeichen.

Maximale Anzahl von Dimensionen – siehe Tabelle.

Die maximale Anzahl von Argumenten für Befehle, die eine variable Anzahl von Argumenten akzeptieren, beträgt 50.

Die maximale Anzahl von verschachtelten FOR…NEXT-Schleifen beträgt 20.

Die maximale Anzahl verschachtelter DO…LOOP-Befehle beträgt 20. Die

maximale Anzahl verschachtelter GOSUBs beträgt 50.

Die maximale Anzahl verschachtelter mehrzeiliger IF…ELSE…ENDIF-Befehle beträgt 20.

Die maximale Anzahl benutzerdefinierter Labels, Unterprogramme und Funktionen (zusammen) – siehe Tabelle. Die

maximale Anzahl konfigurierbarer Interrupt-Pins: 10

Zahlen werden als Gleitkommazahlen mit doppelter Genauigkeit oder als 64-Bit-Ganzzahlen mit Vorzeichen gespeichert und verarbeitet. Der Bereich der Gleitkommazahlen reicht von 1,797693134862316e+308 bis 2,225073858507201e-308.

Der Bereich der 64-Bit-Ganzzahlen (ganze Zahlen), die bearbeitet werden können, beträgt ± 9223372036854775807. Die maximale

Zeichenfolgenlänge beträgt 255 Zeichen.

Die maximale Zeilenzahl beträgt 65000.

Die maximale Anzahl von Hintergrundimpulsen, die mit dem Befehl PULSE ausgelöst werden können,

beträgt 5. Maximale Anzahl globaler Variablen und Konstanten – siehe Tabelle.

Die maximale Anzahl lokaler Variablen – siehe Tabelle

Die maximale Anzahl von Dateien, die mit dem Befehl FILES aufgelistet werden können, beträgt 1000.

Die maximale Länge eines Dateinamens/Pfads beträgt 63 Zeichen (RP2040) oder 127 Zeichen (RP2350).

Eigenschaften vs. Firmware-Funktionen:

	Maximale Programmgröße	Maximale Frequenz	Maximale Anzahl von Unterprogrammen oder Funktionen	Maximale Array-Dimensionen	Maximale Anzahl globaler Variablen	Maximale Anzahl lokaler Variablen*
PicoMite RP2040	124 KB	420 MHz	256	6	256	256
PicoMiteUSB RP2040	124 KB	420 MHz	256	6	256	256
PicoMiteVGA RP2040	100 KB	378 MHz	256	6	256	224
PicoMiteVGAUSB RP2040	100 KB	378 MHz	256	6	256	224
WebMite RP2040	88 KB	252 MHz	256	6	256	224
PicoMite RP2350	320 KB	396 MHz	512	5	512	224
PicoMiteUSB RP2350	320 KB	396 MHz	512	5	512	224
PicoMiteVGA RP2350	184 KB	378 MHz	512	5	480	256
PicoMiteVGAUSB RP2350	184 KB	378 MHz	512	5	480	256
PicoMiteHDMI RP2350	176 KB	372 MHz	512	5	480	256
PicoMiteHDMIUSB RP2350	176 KB	372 MHz	512	5	480	256
WebMite RP2350	200 KB	252 MHz	512	5	512	256

* Das Gleichgewicht zwischen globalen und lokalen Variablen kann geändert werden. Siehe OPTION LOCAL VARIABLES und MM.INFO(MAX VARS)

Kompatibilität

MMBasic implementiert einen großen Teil von Microsofts GW-BASIC. Aufgrund physikalischer und praktischer Überlegungen gibt es zahlreiche Unterschiede, aber die meisten Standard-BASIC-Befehle und -Funktionen sind im Wesentlichen identisch. Ein Online-Handbuch für GW-BASIC ist unter http://www.antonis.de/qbebooks/gwbasman/index.html verfügbar und enthält eine detailliertere Beschreibung der Befehle und Funktionen.

MMBasic implementiert auch eine Reihe moderner Programmierstrukturen, die im ANSI-Standard für Full BASIC (X3.113-1987) oder ISO/IEC 10279:1991 dokumentiert sind. Dazu gehören SUB/END SUB, DO WHILE …
LOOP, die Anweisungen SELECT…CASE und strukturierte IF . THEN … ELSE … ENDIF-Anweisungen.

Vordefinierte schreibgeschützte Variablen des

Diese Variablen werden von MMBasic festgelegt und können vom laufenden Programm nicht geändert werden. Beachten Sie, dass sie selbst keine Funktion haben, sondern ausgegeben oder einer Variablen zugewiesen werden müssen.

Beispiel:

```
> PRINT MM.VER
6.0002
>
```

oder in einem Programm:

```
Wenn MM.VER < 6.0002 Dann Fehler „Version 6.00.02 oder höher erforderlich”
```

MM.VER	Gibt die Versionsnummer der Firmware als Gleitkommazahl in der Form aa.bbcc zurück, wobei aa die Hauptversionsnummer, bb die Nebenversionsnummer und cc die Revisionsnummer ist. Beispielsweise würde Version 6.03.00 den Wert 6.03 und Version 6.03.01 den Wert 6.0301 zurückgeben.
MM.ADDRESS$	NUR WEBMITE-VERSION Diese Variable gibt die IP-Adresse des Absenders des zuletzt empfangenen UDP-Datagramms zurück.
MM.CMDLINE$	Diese Funktion gibt alle Befehlszeilenargumente zurück, die beim Ausführen eines MMBasic-Programms an das aktuelle Programm übergeben wurden. Weitere Informationen finden Sie unter den Befehlen RUN und *. Die Funktion gibt eine leere Zeichenfolge zurück, wenn Programme aus dem Editor oder mit der Option OPTION AUTORUN ausgeführt werden.
MM.DEVICE$	Eine Zeichenkette, die das Gerät oder die Plattform angibt, auf der MMBasic ausgeführt wird.
MM.DISPLAY	Gibt 1 zurück, wenn ein physisches Display konfiguriert ist, andernfalls 0.
MM.ERRNO MM.ERRMSG	Wenn eine Anweisung einen Fehler verursacht hat, der ignoriert wurde, werden diese Variablen entsprechend gesetzt. MM.ERRNO ist eine Zahl, wobei ein Wert ungleich Null bedeutet, dass ein Fehler aufgetreten ist, und MM.ERRMSG$ ist eine Zeichenfolge, die die Fehlermeldung darstellt, die normalerweise auf der Konsole angezeigt worden wäre. Sie werden durch RUN, ON ERROR IGNORE oder ON ERROR SKIP auf Null und eine leere Zeichenfolge zurückgesetzt.
MM.FLAGS	Gibt den Wert des Systemflagsregisters zurück
MM.FONTHEIGHT MM.FONTWIDTH	Gibt die Höhe oder Breite der aktuellen Schriftart in Pixeln zurück
MM.HRES MM.VRES	Ganzzahlen, die die aktuelle horizontale und vertikale Auflösung des VGA/HDMI-Videoausgangs oder des LCD-Bildschirms (sofern konfiguriert) in Pixeln angeben.
MM.HEIGHT MM.WIDTH	Gibt die Anzahl der Zeichen über die physische Anzeige mit der aktuellen Schriftart oder die Anzahl der Zeichen unter der Anzeige mit der aktuellen Schriftart zurück
MM.HPOS MM.VPOS	Gibt die aktuelle horizontale und vertikale Position (in Pixeln) nach dem letzten Grafik- oder Druckbefehl zurück.
MM.SUPPLY	Bei Modulen mit einem Pin zur Messung der Eingangsspannung, wie z. B. Pico und Pico2, gibt MM.SUPPLY die Spannung zurück.

MM.INFO() MM.INFO$()	Diese beiden Versionen können austauschbar verwendet werden, aber gemäß guter Programmierpraxis sollten Sie diejenige verwenden, die dem zurückgegebenen Datentyp entspricht.
MM.INFO$(AUTORUN)	Gibt die Einstellung des Befehls OPTION AUTORUN zurück.
MM.INFO(ADC)	Gibt die Nummer des Puffers zurück, der bei Verwendung von ADC RUN derzeit zum Lesen bereit ist (1 oder 2). Gibt 0 zurück, wenn nichts bereit ist.
MM.INFO(ADC DMA)	Gibt „true“ (1) zurück, wenn ADC DMA aktiv ist.
MM.INFO(BOOT)	Gibt den Grund für den letzten Neustart des Pico an. Gibt Folgendes zurück: Neustart - Das Gerät wurde mit CPU RESTART oder einem OPTION-Befehl neu gestartet. S/W-Watchdog – Das Gerät wurde aufgrund eines Software-Watchdog-Timeouts neu gestartet. H/W-Watchdog – Das Gerät wurde aufgrund eines Hardware-Watchdog-Timeouts neu gestartet. Firmware-Update – Das Gerät wurde nach einem Firmware-Update neu gestartet. Einschalten – Das Gerät wurde eingeschaltet Reset-Schalter – Das Gerät wurde mithilfe des Reset-Schalters neu gestartet. Unbekannter Code &Hn – Unbekannter Grund – Bitte melden Sie den Code und die Version RP2040/2350.
MM.INFO(BOOT COUNT)	Gibt die Anzahl der Neustarts des Pico seit der letzten Formatierung des Flash-Laufwerks zurück.
MM.INFO$(CALLTABLE)	Gibt die Basisadresse der MMBasic-Aufruftabelle zurück, die Zeiger auf jede MMBasic-Funktion enthält.
MM.INFO$(CPUSPEED)	Gibt die CPU-Geschwindigkeit als Zeichenfolge zurück.
MM.INFO$(LCDPANEL)	Gibt den Namen des konfigurierten LCD-Panels oder eine leere Zeichenfolge zurück.
MM.INFO(LCD320)	Gibt „true“ zurück, wenn das Display mit dem Befehl OPTION LCD320 für eine Auflösung von 320 x 240 geeignet ist.
MM.INFO$(SDCARD)	Gibt den Status der SD-Karte zurück. Gültige Rückgabewerte sind: DISABLED, NOT PRESENT, READY und UNUSED.
MM.INFO$(CURRENT)	Gibt den Namen des aktuellen Programms zurück, wenn es aus einer Datei geladen wurde, oder NONE, wenn es nach einem NEW-, AUTOSAVE-, XMODEM- oder EDIT-Befehl aufgerufen wird.
MM.INFO$(PATH)	Gibt den Pfad des aktuellen Programms zurück oder NONE, wenn nach einem NEW- oder EDIT-Befehl aufgerufen.
MM.INFO(DISK SIZE)	Gibt die Kapazität des Flash-Dateisystems oder der SD-Karte, je nachdem, welches Laufwerk aktiv ist, in Byte zurück.
MM.INFO$(DRIVE)	Gibt das aktive Laufwerk „A:“ oder „B:“ zurück.
MM.INFO(EXISTS FILE fname$)	Gibt 1 zurück, wenn die angegebene Datei existiert, gibt -1 zurück, wenn fname$ ein Verzeichnis ist, andernfalls gibt es 0 zurück.
MM.INFO(EXISTS DIR dirname$)	Gibt einen booleschen Wert zurück, der angibt, ob das angegebene Verzeichnis existiert.

MM.INFO(FREE SPACE)	Gibt den freien Speicherplatz auf dem Flash-Dateisystem oder der SD-Karte zurück, je nachdem, welches Laufwerk aktiv ist.
MM.INFO(FILESIZE file$)	Gibt die Größe von file$ in Byte zurück oder 0, wenn nicht gefunden.
MM.INFO$(MODIFIED file$)	Gibt das Datum/die Uhrzeit zurück, zu der file$ geändert wurde, leere Zeichenfolge, wenn nicht gefunden.
MM.INFO$(SYSTEM I2C)	Gibt „I2C“, „I2C2“ oder „Nicht eingestellt“ zurück, je nach Status von OPTION SYSTEM I2C
MM.INFO(FCOLOUR)	Gibt die aktuelle Vordergrundfarbe zurück.
MM.INFO(BCOLOUR)	Gibt die aktuelle Hintergrundfarbe zurück. Gibt die
MM.INFO(FONT)	Nummer der aktuell aktiven Schriftart zurück.
MM.INFO(SCHRIFTARTADRESSE n)	Gibt die Adresse des Speicherplatzes mit der Adresse von FONT n zurück.
MM.INFO(SCHRIFTARTZEIGER n)	Gibt einen ZEIGER auf den Anfang von FONT n im Speicher zurück.
MM.INFO(SCHRIFTGRÖSSE) MM.INFO(SCHRIFTBREITE)	Ganzzahlen, die die Höhe und Breite der aktuellen Schriftart (in Pixeln) angeben.
MM.INFO(FLASH)	Meldet gegebenenfalls, aus welchem Flash-Slot das Programm geladen wurde. Gibt die
MM.INFO(FLASH-ADRESSE n) MM.INFO(HEAP)	Adresse des Flash-Slots n zurück.
MM.INFO(HPOS) MM.INFO(VPOS)	Gibt die Menge des freien MMbasic-Heap-Speichers zurück. Der MMBasic-Heap wird für Zeichenfolgen, Arrays und verschiedene andere temporäre und permanente Puffer (z. B. Audio) verwendet.
MM.INFO(ID) MM.INFO$(IP-ADRESSE) MM.INFO(MAX GP)	Die aktuelle horizontale und vertikale Position (in Pixeln) nach dem letzten Grafik- oder Druckbefehl.
MM.INFO(MAX VARS)	Gibt die eindeutige ID des Pico zurück. Gibt die
MM.INFO$(MODBUFF-ADRESSE)	IP-Adresse des WebMite zurück.
MM.INFO$(OPTION Option)	Gibt die höchste gültige GPno auf dem Chip zurück
MM.INFO$(PIN pinno)	Gibt die Gesamtzahl der in der aktuellen Version verfügbaren Variablen zurück. Gibt die Adresse im Speicher des Puffers zurück, der zum Speichern von MOD-Dateien verwendet wird.
MM.INFO(PINNO GPnn)	Gibt den aktuellen Wert einer Reihe von Optionen zurück, die sich auf die Ausführung eines Programms auswirken. „option” kann einer der folgenden Werte sein: AUTORUN, AUDIO, BASE, BREAK, CONSOLE, DEFAULT, EXPLICIT, KEYBOARD, ANGLE, HEIGHT, WIDTH, FLASH SIZE
MM.INFO(PIO-RX-DMA)	Gibt den Status des E/A-Pins „pinno“ zurück. Gültige Rückgabewerte sind: OFF, DIN, DOUT, AIN usw.
MM.INFO(PIO-TX-DMA)	Gibt die physikalische Pin-Nummer für eine bestimmte GP-Nummer zurück. GPnn kann eine nicht in Anführungszeichen gesetzte Zeichenfolge (GP01), eine Zeichenfolgenliteral („GP01“) oder eine Zeichenfolgenvariable sein. D. h., A$ = „GP01“: MM.INFO(PINNO A$).
	Gibt an, ob der PIO-RX-DMA-Kanal belegt ist Gibt an, ob der PIO- TX-DMA-Kanal belegt ist

MM.INFO$(PLATFORM)	Gibt die zuvor mit OPTION PLATFORM festgelegte Zeichenfolge zurück. Gibt die
MM.INFO(PROGRAM)	Adresse des aktuell ausgeführten Programms im Speicher zurück. Meldet die letzte
MM.INFO(PS2) MM.INFO(PWM	Rohmeldung, die über die PS2-Schnittstelle empfangen wurde, sofern diese aktiviert
COUNT)	ist. Gibt die Anzahl der vom Chip unterstützten PWM-Kanäle zurück.
MM.INFO(PWM DUTY C%, n%)	Gibt den aktuellen Arbeitszyklus in Takten des PWM-Kanals C%, N% zurück, wobei N%=0 für A und 1 für B gilt.
MM.INFO$(SOUND)	Gibt die aktuelle Aktivität am Audioausgang zurück (AUS, PAUSE, TON, WAV, FLAC, MP3, SOUND)
MM.INFO(SPI-GESCHWINDIGKEIT	Gibt die tatsächliche Geschwindigkeit des SYSTEM SPI zurück oder einen Fehler, wenn nicht eingestellt
MM.INFO(STACK)	Gibt den C-Stack-Zeiger zurück. Komplexer oder rekursiver Basic-Code kann zu dem Fehler „Stack overflow, expression too complex at depth %" führen. Dies tritt auf, wenn der Stack unter &H 2003f800 liegt. Durch Überwachen des Stacks kann der Programmierer Vereinfachungen des Basic-Codes identifizieren, um den Fehler zu vermeiden.
MM.INFO$(SYSTEM I2C)	Gibt I2C, I2C2 oder NOT SET zurück, je nach
MM.INFO(SYSTEM HEAP)	Anwendbarkeit. Gibt den freien Speicherplatz auf dem
MM.INFO(SYSTICK)	System-Heap zurück.
	Gibt den aktuellen Wert des 24-Bit-Systick-Timers des Systems zurück, der mit der CPU-Taktfrequenz läuft.
MM.INFO(KACHELHÖHE)	NUR VGA- UND HDMI-VERSIONEN
MM.INFO(SPUR)	Gibt die aktuelle Einstellung der Kachelhöhe zurück.
	Gibt den Namen der FLAC-, MP3-, WAV- oder MIDI-Datei zurück, die derzeit über den Audioausgang wiedergegeben wird.
MM.INFO$(TOUCH) MM.INFO(USB	Gibt den Status des Touch-Controllers zurück. Gültige Rückgabewerte sind: „Deaktiviert", „Nicht kalibriert" und „Bereit".
n)	Gibt den Gerätecode für jedes an Kanal „n" angeschlossene Gerät zurück, wobei „n" eine Zahl zwischen 1 und 4 ist. Der zurückgegebene Gerätecode kann sein: 0=nicht verwendet, 1=Tastatur, 2=Maus, 128=PS4, 129=PS3, 130=SNES/Generisch Standardmäßig wird eine angeschlossene Tastatur dem Kanal 1, eine Maus dem Kanal 2 und Gamepads dem Kanal 3 und dann dem Kanal 4 zugewiesen. Wenn zwei oder mehr Tastaturen oder Mäuse oder drei oder mehr Gamepads angeschlossen sind, werden die zusätzlichen Geräte dem höchsten verfügbaren Kanal zugewiesen.
MM.INFO(USB VID n)	Gibt die VID des USB-Geräts auf Kanal n zurück Gibt die PID des
MM.INFO(USB PID n)	USB-Geräts auf Kanal n zurück
MM.INFO(VARCNT)	Gibt die Anzahl der im MMBasic-Programm verwendeten Variablen zurück.
MM.INFO$(LINE)	Gibt die aktuelle Zeilennummer als Zeichenfolge zurück. LIBRARY wird zurückgegeben, wenn es sich in der Bibliothek befindet, und UNKNOWN, wenn es nicht in einem Programm enthalten ist. Hilft bei der Diagnose während der Komponententests.
MM.INFO(UPTIME)	Gibt die Zeit seit dem Start in Sekunden als Gleitkommazahl zurück.

MM.INFO(VALID CPUSPEED Geschwindigkeit %)	Gibt 1 zurück, wenn „speed%“ für OPTION CPUSPEED gültig ist. „speed%“ Die
MM.INFO(VERSION)	Versionsnummer der Firmware (MM.VER konvertiert in diese).
MM.INFO(WRITEBUFF)	Gibt die Adresse im Speicher des aktuellen Puffers zurück, der für Zeichenbefehle verwendet wird.
MM.INFO(TCP-PORT) MM.INFO(UDP-PORT)	NUR WEBMITE Gibt den als Server festgelegten TCP-Port zurück oder 0, wenn keiner festgelegt ist Gibt den als Server festgelegten UDP-Port zurück oder 0, wenn keiner festgelegt ist
MM.INFO(TCPIP STATUS) MM.INFO(WIFI STATUS)	NUR WEBMITE Gibt den TCPIP-Status der Verbindung zurück. Gibt den WIFI-Status der Verbindung zurück. Gültige Rückgaben sind: 0 WiFi ist ausgefallen 1 Mit WLAN verbunden 2 Mit WLAN verbunden, aber keine IP-Adresse (nur TCPIP-STATUS) 3 Mit WLAN verbunden mit einer IP-Adresse (nur TCPIP-STATUS) -1 Verbindung fehlgeschlagen -2 Keine passende SSID gefunden (möglicherweise außerhalb der Reichweite oder nicht verfügbar) -3 Authentifizierungsfehler
MM.MESSAGE$	NUR WEBMITE Gibt den Inhalt des zuletzt empfangenen UDP-Datagramms oder des zuletzt empfangenen MQTT-Pakets zurück
MM.TOPIC$	NUR WEBMITE Gibt das Thema des zuletzt empfangenen MQTT-Pakets zurück
MM.ADDRESS$	NUR WEBMITE Gibt die Adresse des Absenders des zuletzt empfangenen UDP-Datagramms oder des zuletzt empfangenen MQTT-Pakets zurück
MM.ONEWIRE	Nach einer 1-Wire-Reset-Funktion wird diese ganzzahlige Variable gesetzt, um das Ergebnis der Operation anzuzeigen: 0 = Gerät nicht gefunden, 1 = Gerät gefunden, 2 = Zeitüberschreitung beim Gerät.
MM.I2C	Nach einem I2C-Schreib- oder Lesebefehl wird diese Ganzzahlvariable gesetzt, um das Ergebnis der Operation wie folgt anzuzeigen: 0 = Der Befehl wurde ohne Fehler ausgeführt. 1 = NACK-Antwort empfangen 2 = Befehl abgelaufen
MM.PERSISTENT	Gibt einen Wert zurück, der mit dem Befehl SAVE PERSISTENT gespeichert wurde.
MM.PS2	Gibt den letzten Code zurück, der über die PS2-Schnittstelle empfangen wurde, sofern diese aktiviert ist.
MM.WATCHDOG	Eine Ganzzahl, die wahr ist, wenn MMBasic aufgrund eines Watchdog-Timeouts neu gestartet wurde (siehe Befehl WATCHDOG), andernfalls falsch.

Optionen

Diese Tabelle listet die verschiedenen Optionsbefehle auf, mit denen MMBasic konfiguriert und seine Funktionsweise geändert werden kann. Als dauerhaft gekennzeichnete Optionen werden in einem nichtflüchtigen Speicher gespeichert und beim Neustart der PicoMite-Firmware automatisch wiederhergestellt. Optionen, die nicht dauerhaft sind, werden beim Start, beim Zurücksetzen und in vielen Fällen beim Ausführen und/oder Beenden eines Programms zurückgesetzt.

Viele OPTION-Befehle erzwingen einen Neustart der PicoMite-Firmware, wodurch die USB-Konsolenschnittstelle zurückgesetzt wird. Das Programm im Speicher geht dabei nicht verloren, da es in einem nichtflüchtigen Flash-Speicher gespeichert ist.

	Permanent?	
OPTION ANGLE RADIANS | DEGREES		Dieser Befehl schaltet die trigonometrischen Funktionen zwischen Grad und Bogenmaß um. Wirkt auf SIN, COS, TAN, ATN, ATAN2, MATH ATAN3, ACOS, ASIN
OPTION AUDIO PWMnApin, PWMnBpin oder OPTION AUDIO DEAKTIVIEREN	✓	Konfiguriert einen der PWM-Kanäle als Audioausgang. „PWMnApin" ist der linke Audiokanal, „PWMnBpin" ist der rechte. Beide Pins müssen zum selben Audiokanal gehören. Beispiel: OPTION AUDIO GP18, GP19 würde PWM1A und PWM1B an den Pins 24 und 25 verwenden. Diese Option verhindert die Verwendung dieser Pins im BASIC-Programm. Der Audioausgang wird mit PWM erzeugt, sodass ein Tiefpassfilter am Ausgang erforderlich ist. Der Audioausgang des Raspberry Pi Pico ist sehr verrauscht. Mit OPTION POWER und/oder einer Stromversorgung über einen separaten 3,3-V-Linearregler kann dies reduziert werden. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION AUDIO SPI CSpin, CLKpin, MOSIpin oder OPTION AUDIO DISABLE	✓	Konfiguriert den Audioausgang so, dass er an einen MCP48n2-DAC geleitet wird, der an die angegebenen Pins angeschlossen ist. Der LDAC-Pin am DAC sollte mit GND verbunden sein.
OPTION AUDIO VS1053 CLKpin, MOSIpin, MISOpin, XCSpin, XDCSpin, DREQpin, XRSTpin oder OPTION AUDIO DISABLE	✓	Konfiguriert den Audioausgang so, dass er an einen VS1053-CODEC weitergeleitet wird. Dies ermöglicht zusätzlich zu den anderen unterstützten Formaten die Wiedergabe von MP3- und MIDI-Dateien und unterstützt auch die Echtzeit-MIDI-Ausgabe. Weitere Informationen finden Sie unter dem Befehl PLAY.
OPTION AUDIO I2S BCLKpin, DINpin oder OPTION AUDIO DEAKTIVIEREN	✓	Konfiguriert die Audioausgabe so, dass sie an einen I2S-DAC geleitet wird, der an die angegebenen Pins angeschlossen ist. Der LRCK-Pin am DAC sollte mit dem nächsten aufeinanderfolgenden GPIO-Pin zum BCLK-Pin verbunden werden.
OPTION AUTOREFRESH OFF | ON		Schwarz-Weiß-Displays können nur bildschirmweise aktualisiert werden. Mit OPTION AUTOREFRESH OFF/ON können Sie steuern, ob ein Schreibbefehl das Display sofort aktualisiert oder nicht. Wenn AUTOREFRESH auf OFF gesetzt ist, kann der Befehl REFRESH zum Auslösen des Schreibvorgangs verwendet werden. Dies gilt für die folgenden Displays: N5110, SSD1306I2C, SSD1306I2C32, SSD1306SPI und ST7920

OPTION AUTORUN ON [,NORESET] oder OPTION AUTORUN n [,NORESET] oder OPTION AUTORUN OFF	✓	Weist MMBasic an, ein Programm beim Einschalten oder Neustart automatisch auszuführen. ON bewirkt, dass das aktuelle Programm im Programmspeicher ausgeführt wird. Durch Angabe von „n“ wird dieser Speicherort im Flash-Speicher ausgeführt. „n“ muss im Bereich von 1 bis 3 liegen. Durch Angabe des optionalen Parameters „NORESET“ bleibt AUTORUN auch dann erhalten, wenn das Programm einen Systemfehler verursacht (standardmäßig führt dies dazu, dass die Firmware alle OPTION AUTORUN-Einstellungen aufhebt). OFF deaktiviert die Autorun-Option und ist die Standardeinstellung für ein neues Programm. Durch Drücken der Unterbrechungstaste (standardmäßig STRG-C) auf der Konsole wird das laufende Programm unterbrochen und die Eingabeaufforderung wieder angezeigt.
OPTION BASE 0 | 1		Legen Sie den niedrigsten Wert für Array-Indizes auf entweder 0 oder 1 fest. Dies muss vor der Deklaration von Arrays verwendet werden und wird beim Einschalten auf den Standardwert 0 zurückgesetzt.
OPTION BAUDRATE nn		Legt die Baudrate der seriellen Konsole fest (sofern diese konfiguriert ist).
OPTION BREAK nn		Legt den Wert der Break-Taste auf den ASCII-Wert „nn” fest. Diese Taste wird verwendet, um ein laufendes Programm zu unterbrechen. Der Wert der Unterbrechungstaste ist beim Einschalten und beim Ausführen eines Programms auf die Taste STRG-C eingestellt, kann jedoch mit diesem Befehl in einem Programm auf eine beliebige Tastaturtaste geändert werden (z. B. setzt OPTION BREAK 4 die Unterbrechungstaste auf die Taste STRG-D). Wenn Sie diese Option auf Null setzen, wird die Unterbrechungsfunktion vollständig deaktiviert.
OPTION CASE LOWER | UPPER | TITLE	✓	Ändert die Groß-/Kleinschreibung für die Auflistung von Befehls- und Funktionsnamen bei Verwendung des Befehls LIST. Die Standardeinstellung ist TITLE, aber der alte Standard von MMBasic kann mit OPTION CASE UPPER wiederhergestellt werden.
OPTION FARBCODE EIN oder OPTION FARBCODE AUS	✓	Aktivieren oder deaktivieren Sie die Farbcodierung für die Ausgabe des Editors. Schlüsselwörter werden in Cyan, Kommentare in Gelb usw. angezeigt. Die Standardeinstellung ist AUS. Das Schlüsselwort COLORCODE (US-amerikanische Schreibweise) kann ebenfalls verwendet werden. Dies funktioniert bei VGA/HDMI-Video und der seriellen Konsole unter Verwendung eines Terminalemulators mit VT100-Emulation (z. B. Tera Term). Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION CONSOLE-Ausgabe		Gibt an, wo Druckanweisungen ausgegeben werden sollen. Gültige Einstellungen sind BOTH (d. h. SCREEN und SERIAL), SERIAL, SCREEN, NONE. Dies ist eine temporäre Option, die beim Beenden eines Programms zurückgesetzt wird.
OPTION CONTINUATION LINES ON/OFF	✓	Aktiviert oder deaktiviert die Verwendung von Fortsetzungszeichen im Editor und mit dem Befehl LIST. Zeilenfortsetzungen werden durch ein Leerzeichen gefolgt von einem Unterstrich am Ende einer Zeile gekennzeichnet. Wenn diese Option aktiviert ist, teilt der Editor die Zeilen beim Einlesen aus der Datei automatisch und fügt die erforderlichen Fortsetzungszeichen hinzu. Beim Beenden des Editors werden die Fortsetzungszeichen vor dem Speichern entfernt. Im Editor kann der Benutzer lange Zeilen erstellen, indem er eigene Fortsetzungszeichen hinzufügt. Dies erleichtert die Verwendung eines kleinen Bildschirms als Konsole erheblich.

OPTION CPUSPEED Geschwindigkeit	✓	NICHT BEI HDMI- ODER VGA-VERSIONEN Ändern Sie die CPU-Taktrate. „speed“ ist die CPU-Taktfrequenz in kHz im Bereich von 48000 bis 396000. Taktfrequenzen über 200 MHz (150 MHz für den RP2350) gelten als Übertaktung, da dies die angegebene maximale Taktfrequenz des Standard-Raspberry Pi Pico ist. Die Standardgeschwindigkeit beträgt 200000 für den RP2040 und 150000 für den RP2350. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION COUNT pin1, pin2, pin3, pin4	✓	Legt fest, welche Pins als Zähleingänge verwendet werden sollen. Standardmäßig sind dies GP6, GP7, GP8 und GP9. Der Befehl SETPIN definiert den Zählermodus. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION DEFAULT FLOAT | INTEGER | STRING | NONE		Wird verwendet, um den Standardtyp für eine Variable festzulegen, die nicht explizit definiert ist. Wenn OPTION DEFAULT NONE verwendet wird, muss der Typ aller Variablen explizit definiert sein, da sonst die Fehlermeldung „Variablentyp nicht angegeben” angezeigt wird. Wenn ein Programm ausgeführt wird, ist der Standardwert aus Kompatibilitätsgründen mit Microsoft BASIC und früheren Versionen von MMBasic auf FLOAT gesetzt.
OPTION DEFAULT COLOURS foreground [,background]	✓	Legen Sie die Standardfarben für Vorder- und Hintergrund sowohl für den Monochrom- als auch für den Farbmodus fest. Die Farbe muss eine der folgenden sein: Weiß, Gelb, Lila, Braun, Fuchsia, Rostrot, Magenta, Rot, Cyan, Grün, Ceruleanblau, Mittelgrün, Kobaltblau, Myrtenblau, Blau und Schwarz. Numerische Werte können nicht verwendet werden. Die Standardeinstellung ist Weiß, Schwarz. Wenn der Hintergrund weggelassen wird, ist die Standardeinstellung schwarz.
OPTION STANDARDMODUS n	✓	Hiermit wird der Standardanzeigemodus beim Booten festgelegt. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION DISK SAVE fname$ OPTION DISK LOAD fname$	✓	Mit diesen Befehlen kann der Benutzer den gesamten Satz der definierten Optionen in einer Datei speichern und aus dieser wiederherstellen. Die Datei kann dann mit XMODEM auf einen Host-Computer übertragen werden, sodass zusätzliche Geräte einfach konfiguriert oder Optionen nach einem Firmware-Upgrade wiederhergestellt werden können.
OPTION DISPLAY lines [,Zeichen]	✓	Legt die Eigenschaften des für die Konsole verwendeten Anzeigeterminals fest. Sowohl der Befehl LIST als auch der Befehl EDIT benötigen diese Informationen, um den Text für die Anzeige korrekt zu formatieren. „Zeilen” ist die Anzahl der Zeilen auf dem Display und „Zeichen” ist die Breite des Displays in Zeichen. Die Standardeinstellung ist 24 Zeilen x 80 Zeichen. Wenn diese Option geändert wird, bleibt sie auch nach dem Ausschalten des Geräts gespeichert. Die Maximalwerte sind 100 Zeilen und 240 Zeichen. Dadurch wird eine ESC-Sequenz gesendet, um das VT100-Terminal auf die entsprechende Größe einzustellen. TerraTerm, Putty und MMCC reagieren auf diese Sequenz und stellen die Terminalbreite ein (sofern die Option in den Terminaleinstellungen aktiviert ist). **Diese Option ist nicht verfügbar, wenn ein LCD-Display als Konsole verwendet wird.**
OPTION ESCAPE		Aktiviert die Möglichkeit, Escape-Sequenzen in String-Konstanten einzufügen. Siehe Abschnitt *Sonderzeichen in Strings*.
OPTION EXPLICIT		Wenn dieser Befehl am Anfang eines Programms platziert wird, muss jede Variable explizit mit den Befehlen DIM, LOCAL oder STATIC deklariert werden, bevor sie im Programm verwendet werden kann. Diese Option ist standardmäßig deaktiviert, wenn ein Programm ausgeführt wird. Wenn sie verwendet wird, muss sie vor der Verwendung von Variablen angegeben werden.

OPTION FAST AUDIO ON|OFF		Bei Verwendung des Befehls PLAY SOUND können Änderungen an Sounds, Lautstärken oder Frequenzen zu hörbaren Klicks in der Ausgabe führen. Die Firmware versucht, dies zu mildern, indem sie die Lautstärke der vorherigen Ausgabe des Kanals vor der Änderung der Ausgabe herunterfährt und sie dann wieder hochfährt. Dies verbessert die Audioausgabe erheblich, führt jedoch zu einer kurzen Verzögerung beim Befehl PLAY SOUND (im schlimmsten Fall 3 ms). Diese Verzögerung kann durch die Verwendung von OPTION FAST AUDIO ON in einem Programm vermieden werden. Die hörbaren Klickgeräusche können dann wieder auftreten, dies liegt jedoch im Ermessen des Programmierers. Dies ist eine temporäre Option, die bei jeder Ausführung eines Programms auf OFF zurückgesetzt wird.
OPTION FNKey string$	✓	Definieren Sie die Zeichenfolge, die generiert wird, wenn eine Funktionstaste an der Befehlszeile gedrückt wird. „FNKey" kann F1 und F5 bis F9 sein. Durch Setzen von string$ auf "" wird die gespeicherte Funktion gelöscht und die ursprüngliche Funktion der Taste wiederhergestellt. Beispiel: OPTION F8 „RUN "+chr$(34)+"myprog" +chr$(34)+chr$(13)+chr$(10). Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION HEARTBEAT ON/OFF [HEARTBEATpin]	✓	Aktiviert oder deaktiviert die Ausgabe der Heartbeat-LED. Beim Pico-W befindet sich der Heartbeat auf einem Pin, der vom CWY43-Chip gesteuert wird. NICHT WEBMITE-VERSION Standardmäßig ist der Heartbeat für RP2350A-Chips auf GP25 aktiviert. Wenn er deaktiviert ist, kann das Programm die LED über GP25 steuern. Bei RP2350B-Chips ist der Heartbeat nicht aktiviert. Wenn der Heartbeat deaktiviert ist, kann der Befehl sowohl zum Aktivieren als auch zum optionalen Festlegen des zu verwendenden Pins (Standard GP25) verwendet werden.
OPTION HDMI-PINS clockpositivepin, d0positivepin, d1positivepin, d2positivepin	✓	NUR HDMI-VERSION Legen Sie die für den HDMI-Videoausgang verwendeten I/O-Pins fest. Dies ist nur erforderlich, um nicht standardmäßigen PCB-Layouts gerecht zu werden. Die positiven HDMI-Signal-Pins werden gemäß „nbr" unten eingestellt. Gültige Werte sind 0-7, und die Pins dürfen sich für jeden Kanal nicht überschneiden. Wenn „nbr" eine gerade Zahl ist, befindet sich der negative Ausgang auf dem physischen Pin+1, wenn „nbr" eine ungerade Zahl ist, befindet er sich auf dem physischen Pin-1. nbr — HSTX Nbr — Physikalischer Pin 0 — HSTX0 — GP12 1 — HSTX1 — GP13 2 — HSTX2 — GP14 3 — HSTX3 — GP15 4 — HSTX4 — GP16 5 — HSTX5 — GP17 6 — HSTX6 — GP18 7 — HSTX7 — GP19 Die Standardeinstellung lautet: OPTION HDMI-PINS 2, 0, 6, 4 Das bedeutet: CK+ und CK- sind GP14 und GP15 zugewiesen D0+ und D0- sind GP12 und GP13 zugewiesen D1+ und D1- sind GP18 und GP19 zugewiesen D2+ und D2- sind GP16 und GP17 zugewiesen

OPTION KEYBOARD nn [,capslock] [,numlock] [,repeatstart] [,repeatrate] oder OPTION TASTATUR DEAKTIVIEREN	✓	Konfigurieren Sie eine Tastatur. Diese kann für die Konsoleneingabe verwendet werden, und alle eingegebenen Zeichen stehen über alle Befehle zur Verfügung, die von der Konsole gelesen werden (seriell über USB). „nn“ ist ein zweistelliger Code, der das Tastaturlayout definiert. Zur Auswahl stehen „US“ für das Standard-Tastaturlayout in den USA, Australien und Neuseeland, „UK“ für das Vereinigte Königreich, „GR“ für Deutschland, „FR“ für Frankreich, „BR“ für Brasilien und „ES“ für Spanien. Dieser Befehl muss in der Befehlszeile eingegeben werden und führt zu einem Neustart. Diese Einstellung kann mit folgendem Befehl zurückgesetzt werden: OPTION KEYBOARD DISABLE Die optionalen Parameter „capslock” und „numlock” sind True/False-Ganzzahlen, die den Ausgangszustand der Tastatur festlegen (Standardwerte sind 0 und 1). Die optionalen Parameter „repeatstart” und „repeatrate” legen die Zeit für die erste Wiederholung einer gedrückten Taste und nachfolgende Wiederholungen fest. Bei einer USB-Tastatur liegen sie zwischen 100 und 2000 ms und zwischen 25 und 2000 ms. Bei einer PS2-Tastatur liegen sie zwischen 0 und 3, was 250 ms, 500 ms, 750 ms und 1000 ms entspricht (Standardwert ist 1), und zwischen 0 und 31, was 33 ms bis 500 ms gemäß der PS2-Tastatur-Spezifikation entspricht (Standardwert ist 12 oder 100 ms).
OPTIONSTASTATUR I2C	✓	Konfiguriert die Unterstützung für die Solderparty bbq20 mini I2C-Tastatur. Hinweis: OPTION SYSTEM I2C muss vor Ausführung dieses Befehls eingestellt werden.
OPTION KEYBOARD PINS clockpin, datapin	✓	Ermöglicht dem Benutzer die Auswahl der Pins, die für den Anschluss einer PS2-Tastatur verwendet werden sollen. Die Standardeinstellung ist Pin 11 (GP8) und Pin 12 (GP9). Die PS2-Tastatur muss deaktiviert werden (OPTION KEYBOARD DISABLE).
OPTION KEYBOARD REPEAT repeatstart , repeatrate	✓	<u>NUR USB-TASTATUR</u> Legt die Zeit für die erste Wiederholung einer gedrückten Taste (100-2000) und nachfolgende Wiederholungen (25-2000) in Millisekunden fest.
OPTION LCD-PINS clkpin, mosipin, misopin	✓	Die Firmware unterstützt die Trennung der SPI-Pins, die zum Ansteuern eines LCD-Displays verwendet werden, von denen, die für Touch und/oder die SD-Karte verwendet werden. Durch diese Trennung kann die Firmware den zweiten Prozessor für die Arbeit mit dem Display nutzen, ohne dass andere Funktionen beeinträchtigt werden. Die Einstellung der LCD-SPI-Pins ist eine Voraussetzung für die Verwendung von gepufferten Treibern mit dem RP2350 PicoMite. Die angegebenen Pins müssen gültige SPI-Pins für die Funktionen clk, mosi (tx) und miso (rx) sein. Wenn auch SYSTEM-SPI-Pins angegeben sind, müssen diese auf einem anderen SPI-Kanal liegen.
OPTION LCDPANEL OPTION LCDPANEL VIRTUAL_C oder OPTION LCDPANEL VIRTUAL_M OPTION LCDPANEL-Optionen oder OPTION LCDPANEL DISABLE	✓	<u>NICHT VGA- ODER HDMI-VERSIONEN</u> Konfiguriert ein LCD-Panel bei Versionen, die einen angeschlossenen LCD-Bildschirm akzeptieren. Konfiguriert ein virtuelles LCD-Panel ohne physisch angeschlossenes Panel. VIRTUAL _C = Farbe, 4 Bit, 320 x 240 VIRTUAL _M = Monochrom, 640 x 480 Mit dieser Funktion kann ein Programm grafische Bilder auf diesem virtuellen Panel zeichnen und sie dann als BMP-Datei speichern. Nützlich zum Erstellen eines Grafikbildes für den Export ohne angeschlossenes Display Konfiguriert die PicoMite-Firmware für die Verwendung mit einem angeschlossenen LCD-Panel. Weitere Informationen finden Sie im Kapitel *„LCD-Anzeigen*”. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.

OPTION LCDPANEL CONSOLE [Schriftart [, fc [, bc [, blight]]] [,NOSCROLL] oder OPTION LCDPANEL NOCONSOLE		Konfiguriert das LCD-Display für die Verwendung als Konsolenausgabe. Das LCD muss transparenten Text unterstützen (d. h. die Controller SSD1963_x, ILI9341 oder ST7789_320). „font" ist die Standardschriftart, „fc" ist die Standard-Vordergrundfarbe, „bc" ist die Standard-Hintergrundfarbe. Diese Parameter sind optional und standardmäßig auf Schriftart 1, Weiß, Schwarz und 100 % eingestellt. Diese Einstellungen werden beim Einschalten angewendet. Der optionale Befehl NOSCROLL ändert die Firmware so, dass bei der Ausgabe auf die letzte Zeile des Displays das Display nicht scrollt, sondern gelöscht wird und die Ausgabe am oberen Rand des Displays fortgesetzt wird. Dadurch können Displays, die kein Lesen unterstützen, als Konsolengerät verwendet werden, und: Beachten Sie, dass bei anderen Displays als dem SSD1963 das Scrollen für jede Konsolenausgabe sehr langsam ist, sodass für diese Displays die Verwendung der Option NOSCROLL empfohlen wird. Diese Einstellung wird im Flash-Speicher gespeichert und beim Start automatisch angewendet. Um sie zu deaktivieren, verwenden Sie den Befehl OPTION LCDPANEL NOCONSOLE. Dieser Befehl muss an der Eingabeaufforderung ausgeführt werden.
OPTION LCDPANEL USER hres, vres		Konfiguriert einen vom Benutzer geschriebenen Anzeigetreiber in MMBasic. Eine Beschreibung zum Schreiben des Treibers finden Sie in der Datei „User Display Driver.txt" im PicoMite-Firmware-Paket.
OPTION LCDPANEL CONSOLE [Schriftart [, fc [,bc]] oder OPTION LCDPANEL NOCONSOLE	✓	NUR VGA- UND HDMI-VERSIONEN Ändert die Standardschriftart, die auf dem VGA- oder HDMI-Display verwendet wird. „fc" ist die Vordergrundfarbe und „bc" ist die Hintergrundfarbe. Deaktiviert die Konsolenausgabe auf dem VGA/HDMI-Bildschirm. Diese Option ist dauerhaft, sowohl die Druckausgabe als auch die Konsolenausgabe werden deaktiviert und nur Grafikbefehle werden auf dem VGA-Bildschirm ausgegeben. Wenn die Ausgabe in einem Programm vorübergehend deaktiviert werden soll, verwenden Sie den Befehl OPTION CONSOLE. Bei SSD1963-basierten Displays im Querformat und SPI-Displays im Hochformat verwendet die Firmware H/W-Scrolling, um die Leistung der Displaykonsole zu verbessern.
OPTION LCD320 EIN/AUS		NICHT VGA- ODER HDMI-VERSIONEN Dies aktiviert oder deaktiviert 16-Bit-LCD-Anzeigen im Modus 320x240, wodurch beispielsweise Spiele auf diesen größeren LCD-Anzeigen möglich werden. Bei Anzeigen mit einer Auflösung von 800x480 wird das Bild mit einer Auflösung von 320x240 um den Faktor 2 skaliert und nimmt den Bildschirmbereich von 80,0 bis 719,479 ein. Bei 480x272-Displays wird das 320x240-Bild fensterbasiert dargestellt und belegt den Bildschirmbereich 80,16 bis 399,255.
OPTION LEGACY ON oder OPTION LEGACY OFF		Dadurch wird der Kompatibilitätsmodus mit den im ursprünglichen Colour Maximite verwendeten Grafikbefehlen ein- oder ausgeschaltet. Die Befehle COLOUR, LINE, CIRCLE und PIXEL verwenden die alte Syntax, und alle Zeichenbefehle akzeptieren Farben im Bereich von 0 bis 7. Hinweise: • Schlüsselwörter wie RED, BLUE usw. sind nicht implementiert und sollten daher bei Bedarf als Konstanten definiert werden. • Die Syntax der alten Befehle finden Sie im Colour Maximite MMBasic Language Manual. Dieses kann unter https://geoffg.net/OriginalColourMaximite.html heruntergeladen werden.
OPTIONSLISTE		Dadurch werden die Einstellungen aller Optionen aufgelistet, die von ihrer Standardeinstellung geändert wurden und dauerhaft sind. OPTION LIST zeigt auch die Versionsnummer und die geladene Firmware an. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.

OPTION LOCAL VARIABLES n		Dieser Befehl legt die Anzahl der Gesamtvariablen (MM.INFO(MAX VARS)) fest, die als lokale Variablen zugewiesen werden sollen. Im Allgemeinen sollte dies nicht erforderlich sein, aber wenn Sie beispielsweise viele globale Konstanten zuweisen möchten und nicht viele lokale Variablen benötigen, kann dieser Befehl verwendet werden, um das Gleichgewicht zu ändern. Dies ist eine temporäre Option, die bis zum Zurücksetzen des Prozessors bestehen bleibt. Sie kann nicht aufgerufen werden, wenn bereits Variablen definiert wurden.
OPTION MILLISECONDS ON|OFF		Dies aktiviert oder deaktiviert eine Millisekundenausgabe in der Funktion TIME$. D. h. HH:MM:SS.mmm Der Millisekundenzähler wird auf Null gesetzt, wenn die Zeit mit dem Befehl TIME, dem Befehl WEB NTP oder dem Befehl RTC GETTIME aktualisiert wird. Die Standardeinstellung ist AUS.
OPTION MODBUFF ENABLE/DISABLE [sizeinK]	✓	Erstellt oder entfernt einen Bereich des Flash-Speichers, der zum Laden und Abspielen von .MOD-Dateien verwendet wird. Wenn diese Option aktiviert ist, wird ein Mod-Puffer mit einer Größe von 128 KB erstellt. Dies kann mit „sizeinK" überschrieben werden. Beachten Sie, dass diese Option einen Teil des Flash-Dateisystems reserviert (d. h. das Flash-Dateisystem wird verkleinert). Die Standardeinstellung ist deaktiviert. Hinweis: Diese Option ist bei einem RP2350 mit aktiviertem PSRAM nicht erforderlich. In diesem Fall wird die MOD-Datei in den Speicherplatz im PSRAM geladen.
OPTION MOUSE CLKpin, DATApin	✓	NUR NICHT-USB-FIRMWARE Stellen Sie die Pins so ein, dass sie zum Anschluss einer PS2-Maus verwendet werden können. Mit diesem Befehl wird die Maus beim Booten automatisch konfiguriert, und Sie können Interrupts einrichten und Werte ohne zusätzliche Befehle lesen. Dies unterscheidet sich von MOUSE OPEN, das nur während der Ausführung des Programms eine Verbindung zu einer Maus herstellt. Die PS2-Maus MUSS deaktiviert sein.
OPTION MAUS DEAKTIVIEREN	✓	Deaktiviert die automatische Verbindung zu einer PS2-Maus und gibt die Pins für die normale Verwendung frei.
OPTION MAUSEMPFINDLICHKEIT f!	✓	NUR USB-FIRMWARE Standardmäßig betreibt die Firmware eine USB-Maus im Boot-Modus. Das bedeutet, dass sie 8-Bit-X- und Y-Positionen sowie den Status der drei Standardtasten zurückgibt. Durch Einstellen von OPTION MOUSE SENSITIVITY weist die Firmware die Maus an, mit ihrer vollen Leistungsfähigkeit zu arbeiten. Das bedeutet, dass sie versucht, den gesamten von der Maus empfangenen USB-Bericht zu interpretieren, einschließlich der Radposition, aller Tasten und der 8-, 12- oder 16-Bit-x/y-Positionsinformationen. Durch Einstellen von „f$" wird der vollständige Mausbericht aktiviert und die x/y-Positionen werden um „f!" skaliert. Die Firmware unterstützt nicht alle Maustypen. Wenn dies bei einer bestimmten Maus zu Problemen führt, setzen Sie sie mit OPTION MOUSE SENSITIVITY 0 zurück in den Startmodus. Die Aktivierung dieser Option mit einem Wert von 1,0 bei Verwendung einer standardmäßigen Microsoft Basic Optical-Maus wurde vollständig getestet und ermöglicht die Verwendung des Rads in einem Programm.
OPTION NOCHECK EIN/AUS		Wenn dieser Befehl auf ON gesetzt ist, wird die standardmäßige Überprüfung auf Interrupts und Strg-C am Ende jedes Befehls deaktiviert. Durch Setzen auf ON können zeitkritische Prozesse ohne Unterbrechungsrisiko ausgeführt werden. Der Befehl sollte jedoch mit Vorsicht verwendet werden, da das Programm sonst möglicherweise nur durch einen Hardware-Reset gestoppt werden kann.

OPTION PICO EIN/AUS	✓	ALLE VERSIONEN AUSSER WEBMITE Wenn diese Option auf OFF gesetzt ist, sind die Pins GP23, GP24 und GP29 nicht für die normale Pico-Verwendung eingerichtet und sofort verfügbar. Standardmäßig ON für RP2350A und RP2040, OFF für RP2350B
OPTION PIN nbr		Legen Sie „nbr" als PIN (Persönliche Identifikationsnummer) für den Zugriff auf die Konsolenaufforderung fest. „nbr" kann eine beliebige Zahl größer als Null mit bis zu acht Stellen sein. Wenn ein laufendes Programm aus irgendeinem Grund versucht, zur Befehlszeile zu wechseln, fragt MMBasic diese Nummer ab, bevor die Eingabeaufforderung angezeigt wird. Dies ist eine Sicherheitsfunktion, da ein Eindringling ohne Zugriff auf die Befehlszeile das Programm im Speicher nicht auflisten oder ändern oder die Funktionsweise von MMBasic in irgendeiner Weise modifizieren kann. Um diese Funktion zu deaktivieren, geben Sie für die PIN-Nummer eine Null ein (d. h. OPTION PIN 0). Eine dauerhafte Sperre kann durch Eingabe der PIN-Nummer 99999999 aktiviert werden. Wenn eine dauerhafte Sperre aktiviert wurde oder die PIN-Nummer verloren gegangen ist, kann das Gerät nur durch erneutes Laden der PicoMite-Firmware wiederhergestellt werden. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.

OPTION PLATFORM name$	✓	Ermöglicht es einem Benutzer, eine bestimmte Hardwarekonfiguration zu identifizieren, die dann in Programmen zur Steuerung des Programmablaufs verwendet werden kann.
		„name$" kann bis zu 31 Zeichen lang sein. Dies ist eine permanente Option.
		MM.INFO$(PLATFORM) gibt diese Zeichenfolge zurück.
		Dies kann beispielsweise für eine bestimmte Hardwarekonfiguration verwendet werden:
		`OPTION PLATFORM „GameMite"`
		Dann können Programme, die auf dieser oder anderen Plattformen laufen, Folgendes verwenden:
		`IF MM.INFO$(PLATFORM) = "GameMite" THEN …`

OPTION POWER PFM | PWM	✓	Ändert den Betrieb des 3,3-V-Schaltnetzteils. Standardmäßig läuft dieses im PFM-Modus. PWM bietet eine bessere Rauschleistung, ist jedoch weniger energieeffizient. Beachten Sie, dass das System unter hoher Last unabhängig von dieser Einstellung im PWM-Modus läuft.
OPTION PSRAM PIN n oder OPTION PSRAM PIN DISABLE	✓	PSRAM-Unterstützung aktivieren/deaktivieren. „n" ist der PSRAM-Chipauswahl-Pin (CS) und kann GP0, GP8, GP19 oder GP47 sein. Typischerweise wird GP47 für Pimoroni-Boards verwendet. Standardmäßig ist diese Funktion deaktiviert. Nach dem Einschalten ist der Inhalt des PSRAM unbestimmt. Verwenden Sie RAM ERASE, um ihn vor der Verwendung zu löschen.
OPTION RESET	✓	Setzt alle gespeicherten Optionen auf ihre Standardwerte zurück. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION RESET cfg oder OPTION RESET LIST	✓	Setzt alle Optionen für die Konfiguration „cfg" auf die Standardwerte zurück. OPTION RESET LIST listet alle verfügbaren Konfigurationen auf. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION RESOLUTION nn [,cpuspeedinKhz]		NUR FÜR HDMI- UND VGA-VERSIONEN Bei Firmware mit HDMI-Video stellen Sie die Videoauflösung auf „nn" ein. Dabei ist „nn": 640x480 oder 640 720x400 oder 720 800x600 oder 800 (nur RP2350)

		848x480 oder 848 (nur RP2350) 1280x720 oder 1280 (nur HDMI) 1024 x 768 oder 1024 (nur HDMI) Für 640x480 kann die Bildwiederholfrequenz auf 60 Hz (252 MHz oder 378 MHz) oder 75 Hz (315 MHz) eingestellt werden, indem „cpuspeedinKHz" an den Befehl angehängt wird (d. h. 252000, 378000 oder 315000). Jede VGA- und HDMI-Auflösung kann in einer Reihe von Modi betrieben werden, die mit dem Befehl MODE eingestellt werden. Beachten Sie, dass die Auflösungen 800x600 und 848x480 sowohl die maximale Programmgröße als auch den für Basic-Programme verfügbaren variablen Speicherplatz reduzieren.
OPTION RTC AUTO ENABLE | DISABLE	✓	Aktiviert das automatische Laden von Zeit und Datum aus dem RTC beim Booten und jede Stunde. Wenn diese Option aktiviert ist und der RTC nicht reagiert, wird jedes laufende Programm mit einer Fehlermeldung abgebrochen. An der Befehlszeile wird eine Informationsmeldung ausgegeben. Dieser Befehl muss an der Befehlszeile (nicht in einem Programm) ausgeführt werden.
OPTION SDCARD CSpin [,CLKpin, MOSIpin, MISOpin] oder OPTION SDCARD DISABLE	✓	Legen Sie die für die SD-Kartenschnittstelle zu verwendenden E/A-Pins fest oder deaktivieren Sie sie. Wenn die optionalen Pins nicht angegeben sind, verwendet die SD-Karte die durch OPTION SYSTEM SPI angegebenen Pins. Hinweis: Die durch OPTION SYSTEM SPI angegebenen Pins müssen ein gültiger Satz von Hardware-SPI-Pins (SPI oder SPI2) sein, während die durch OPTION SDCARD angegebenen Pins beliebige Pins sein können. Die durch OPTION SYSTEM SPI und OPTION SDCARD angegebenen Pins dürfen nicht identisch sein. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION SDCARD COMBINED CS	✓	Hiermit wird festgelegt, dass der Touch-Chip-Auswahl-Pin auch für die SD-Karte verwendet wird. In diesem Fall ist eine externe Schaltung erforderlich, um die SD-Chip-Auswahl zu implementieren. Siehe „SD-Karten" im Kapitel „Programm- und Datenspeicherung".
OPTION SERIAL CONSOLE uartapin, uartbpin [,B] OPTION SERIAL CONSOLE DISABLE	✓	Geben Sie an, dass der Zugriff auf die Konsole über einen seriellen Hardwareanschluss (anstelle eines virtuellen seriellen Anschlusses über USB) erfolgen soll. „uartapin" und „uartbpin" können ein beliebiges gültiges Paar von rx- und tx-Pins für COM1 oder COM2 sein. Die Reihenfolge, in der sie angegeben werden, ist nicht wichtig. Die Standardgeschwindigkeit beträgt 115200 Baud, kann jedoch mit OPTION BAUDRATE geändert werden. Durch Hinzufügen des Parameters „B" wird die Ausgabe sowohl an den seriellen Anschluss als auch an den USB-Anschluss gesendet. Zurück zur normalen USB-Konsole. Diese Befehle müssen an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION SYSTEM I2C sdapin, sclpin [,SLOW/FAST]	✓	Geben Sie den I^2C-Port und die Pins an, die von Systemgeräten (LCD-Panel und RTC) verwendet werden sollen. Die PicoMite-Firmware verwendet einen bestimmten I^2C-Port für Systemgeräte, während der andere für den Programmierer reserviert ist. Dieser Befehl legt fest, welche Pins verwendet werden sollen und somit welcher der I^2C-Ports verwendet werden soll. Die dem SYSTEM I2C zugewiesenen Pins stehen für andere MMBasic SETPIN-Einstellungen nicht zur Verfügung, können jedoch für zusätzliche I^2C-Geräte unter Verwendung des Standard-I2C-Befehls verwendet werden. Hinweis: I2C(2) OPEN und I2C(2) CLOSE sind in diesem Fall nicht verfügbar. Standardmäßig wird der I^2C-Port mit einer Geschwindigkeit von 400 kHz und einer Zeitüberschreitung von 100 ms geöffnet. Die I^2C-Frequenz kann mit dem optionalen dritten

		Parameter, der die Werte FAST = 400 kHz oder SLOW = 100 kHz annehmen kann. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION SYSTEM SPI CLKpin, MOSIpin, MISOpin oder OPTION SYSTEM SPI DISABLE	✓	Legen Sie den SPI-Port und die Pins für die Verwendung durch Systemgeräte (SD-Karte, LCD-Panel usw.) fest oder deaktivieren Sie sie. Die PicoMite-Firmware verwendet einen bestimmten Hardware-SPI-Port für Systemgeräte, während der andere für den Benutzer verfügbar bleibt. Dieser Befehl legt fest, welche Pins verwendet werden sollen und somit welcher der SPI-Ports verwendet werden soll. Die dem SYSTEM SPI zugewiesenen Pins stehen für andere MMBasic-Befehle nicht zur Verfügung. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION TAB 2 | 3 | 4 | 8	✓	Legen Sie den Abstand für die Tabulatortaste fest. Der Standardwert ist 2.
OPTION TCP-SERVER-PORT n	✓	NUR WEBMITE Startet bei jedem Neustart von WebMite einen TCP-Server auf Port „n“. In der Regel verwenden HTTP-Server Port 80. Verwenden Sie „OPTION TCP SERVER PORT 0”, um diese Option zu deaktivieren. Wenn der Server läuft, kann er auf bis zu MM.INFO(MAX CONNECTIONS) reagieren.
OPTION TELNET-KONSOLE AUS|NUR|EIN	✓	NUR WEBMITE Konfiguriert die Handhabung der Konsole über Telnet. ON = Konsolenausgabe wird NUR an USB und Telnet gesendet = Konsolenausgabe wird nur an Telnet gesendet OFF = Konsolenausgabe wird nur an USB gesendet
OPTION TFTP AUS|EIN	✓	NUR WEBMITE Aktiviert oder deaktiviert den TFTP-Server. Standardmäßig aktiviert.
OPTION TOUCH T_CS-Pin, T_IRQ-Pin [, Beep] oder OPTION TOUCH DISABLE	✓	NICHT VGA- ODER HDMI-VERSIONEN Konfiguriert MMBasic für die berührungsempfindliche Funktion eines angeschlossenen LCD-Bildschirms. „T_CS-Pin” und „T_IRQ-Pin” sind die I/O-Pins, die für die Chipauswahl bzw. den Touch-Interrupt verwendet werden (es können beliebige freie Pins verwendet werden). Die übrigen Pins werden mit den Pins verbunden, die mit dem Befehl OPTION SYSTEM SPI angegeben wurden. „Beep“ ist ein optionaler Pin, der an einen kleinen Summer/Beeper angeschlossen werden kann, um ein „Klicken“ oder einen Piepton zu erzeugen, wenn ein Advanced Graphics-Steuerelement (z. B. Optionsfeld, Schalter usw.) berührt wird. Dies wird in *Advanced Graphics Functions.pdf* beschrieben. Dieser Befehl muss an der Eingabeaufforderung (nicht in einem Programm) ausgeführt werden.
OPTION TOUCH FT6336 IRQpin, RESETpin [,BEEPpin] [,sensitivity]	✓	NICHT VGA- ODER HDMI-VERSIONEN Aktiviert die Touch-Unterstützung für den kapazitiven Touch-Chip FT6336. Die Empfindlichkeit ist eine Zahl zwischen 0 und 255 – der Standardwert ist 50, je niedriger der Wert, desto empfindlicher ist das Gerät. SDA und SCK sollten an gültige I2C-Pins angeschlossen und mit OPTION SYSTEM I2C eingerichtet werden. Siehe auch die TOUCH-Funktion.

OPTION VCC-Spannung		Gibt die an den Raspberry Pi Pico gelieferte Spannung (Vcc) an. Bei Verwendung der ADC-Pins zur Spannungsmessung verwendet die PicoMite-Firmware die Spannung am Pin mit der Bezeichnung VREF als Referenz. Diese Spannung kann mit einem DMM genau gemessen und mit diesem Befehl für eine genauere Messung eingestellt werden. Der Parameter wird nicht gespeichert und sollte im Programm initialisiert werden. Der Standardwert, wenn nicht festgelegt, ist 3,3.
OPTION UDP-SERVER-PORT n	✓	NUR WEBMITE-VERSION Richtet einen Listening-Socket auf dem angegebenen Port ein. Alle an diesem Port empfangenen UDP-Datagramme werden verarbeitet und der Inhalt in MM.MESSAGE$ gespeichert. Die IP-Adresse des Absenders wird in MM.ADDRESS$ gespeichert. Hinweis: Wenn das UDP-Datagramm länger als 255 Zeichen ist, werden alle zusätzlichen Zeichen verworfen. Verwenden Sie „OPTION UDP SERVER PORT 0", um diese Option zu deaktivieren.
OPTION VGA-PINS HSYNC-Pin, BLUE-Pin	✓	NUR VGA-VERSION Ändert die für die VGA-Bildschirmausgabe verwendeten Pins und ermöglicht so mehr Flexibilität beim PCB-Design oder bei der Verdrahtung. „HSYNCpin" definiert den Anfang eines Paares zusammenhängender GP-nummerierter Pins, die mit HSYNC und VSYNC verbunden sind. „BLUEpin" definiert den Beginn von vier aufeinanderfolgenden GP-nummerierten Pins, die mit BLUE, GREEN_LSB, GREEN_MSB und RED verbunden sind.
OPTION WEB-MELDUNGEN EIN/AUS		NUR WEBMITE-VERSION Deaktiviert informative Web-Meldungen, wenn auf „OFF" gesetzt. Standardmäßig ist „ON" eingestellt.
OPTION WIFI ssid$, passwd$, [name$] [,ipaddress$, mask$, gateway$]	✓	NUR WEBMITE-VERSION Konfiguriert die Firmware so, dass sie sich beim Neustart automatisch mit einem WLAN-Netzwerk verbindet. „ssid$" ist der Name des Netzwerks und „password$" ist das Zugangspasswort. Beide sind Zeichenfolgen und sollten bei Verwendung von Zeichenfolgenkonstanten in Anführungszeichen gesetzt werden. Optional kann ein Name für das Gerät angegeben werden „name$", andernfalls wird ein Name aus der eindeutigen Geräte-ID erstellt. Optional können eine statische IP-Adresse, eine IP-Maske und eine Gateway-Adresse als „ipaddress$", „mask$" und „gateway$" angegeben werden. z. B. OPTION WIFI „mysid", „mypassword", „myPico", „192.168.1.111", „255.255.255.0", „192.168.1.1"

Befehle

Eckige Klammern zeigen an, dass der Parameter oder die Zeichen optional sind.

‘ (einfaches Anführungszeichen)	Beginnt einen Kommentar, und jeder darauf folgende Text wird ignoriert. Kommentare können an beliebiger Stelle in einer Zeile platziert werden.
*Datei	Der Sternchenbefehl ist eine Abkürzung für RUN, die nur an der MMBasic-Eingabeaufforderung verwendet werden kann. Beispiel: * RUN *foo RUN „foo” *„foo bar” RUN „foo bar” *foo –wombat RUN „foo”, „--wombat” *foo „wom” RUN „foo”, CHR$(34) + „wom” + CHR$(34) *foo --wom="bat" RUN "foo","--wom=" + CHR$(34) + "bat" + CHR$(34) String-Ausdrücke werden von diesem Befehl nicht unterstützt/ausgewertet; alle angegebenen Argumente werden als Literalstring an den Befehl RUN übergeben.
? (Fragezeichen)	Verknüpfung für den Befehl PRINT.
/* */	Beginn und Ende von mehrzeiligen Kommentaren. /* und */ müssen die ersten Nicht-Leerzeichen am Anfang einer Zeile sein und dürfen nur von einem Leerzeichen oder einem Zeilenende gefolgt werden (d. h. es handelt sich um MMBasic-Befehle). Mehrzeilige Kommentare können nicht innerhalb von Unterprogrammen und Funktionen verwendet werden. Alle Zeichen nach */ in einer Zeile werden ebenfalls als Kommentar behandelt.
A: oder B:	Verknüpfung für Laufwerk „A:“ und Laufwerk „B:“ in der Eingabeaufforderung
ADC ADC OPEN Frequenz, Anzahl Kanäle [,Interrupt] ADC FREQUENCY freq ADC CLOSE ADC START array1!() [,array2!()] [,array3!()] [,array4!()] [,C1min] [,C1max] [,C2min] [,C2max] [,C3min] [,C3max] [,C4min] [,C4max]	Die ADC-Befehle bieten eine alternative Methode zur Aufzeichnung analoger Eingänge und sind für die Hochgeschwindigkeitsaufzeichnung vieler Messwerte in einem Array vorgesehen. Dabei werden bis zu 4 ADC-Kanäle aus der Gruppe ADC0-ADC3 zur Verwendung zugewiesen und so eingestellt, dass sie mit der angegebenen Frequenz umgewandelt werden. Die Pin-Reihe umfasst GP26, GP27, GP28 und GP29 für RP2040 und RP2350A bzw. GP40, GP41, GP42 und GP43 für RP2350B. Bei einem Kanal wird immer GP26 (ADC0) verwendet, bei zwei Kanälen werden GP26 und GP27 (ADC0 und ADC1) verwendet usw. Die Abtastung mehrerer Kanäle erfolgt sequenziell (es gibt nur einen ADC). Die Pins sind auf die Funktion festgelegt, wenn ADC OPEN aktiv ist. Die maximale Gesamtfrequenz beträgt CPU-Geschwindigkeit/96 (z. B. 520 kHz, wenn alle vier Kanäle mit einer CPU-Einstellung von 200 MHz abgetastet werden sollen). Beachten Sie, dass eine Gesamtabtastfrequenz von über 500 kHz eine Übertaktung des ADC darstellt. Der optionale Interrupt-Parameter gibt einen Interrupt an, der nach Abschluss der Konvertierung aufgerufen werden soll. Wenn kein Parameter angegeben ist, wird die Konvertierung blockiert. Dadurch wird die Abtastfrequenz der ADC-Konvertierung geändert, ohne dass sie geschlossen und erneut geöffnet werden muss. Gibt die Pins für die normale Verwendung frei. Dies startet die Konvertierung in die angegebenen Arrays. Die Arrays müssen Fließkommatypen sein und dieselbe Größe haben. Die Größe der Arrays definiert die Anzahl der Konvertierungen. Start kann wiederholt aufgerufen werden, sobald der ADC OPEN ist. „Cxmin” und „Cxmax” skalieren die Messwerte. Beispielsweise erzeugen C1min=200 und C1max=100 Werte im Bereich von 200 bis 100 für äquivalente Spannungen

ADC RUN array1%(),array2%)	von 0 bis 3,3. Wenn die Skalierung nicht verwendet wird, werden die Ergebnisse als Spannung zwischen 0 und OPTION VCC (Standardwert 3,3 V) zurückgegeben. Führt den ADC kontinuierlich im doppelt gepufferten Modus aus. Der ADC füllt zuerst array1% und dann array2% und dann wieder array1% usw. Wenn im Befehl ADC OPEN mehr als ein ADC-Kanal angegeben ist, werden die Daten verschachtelt. Die Daten werden als gepackte 8-Bit-Werte zurückgegeben (verwenden Sie MEMORY UNPACK, um sie in ein normales Array zu konvertieren). MM.INFO(ADC) gibt die Nummer des derzeit zum Lesen verfügbaren Puffers zurück (1 oder 2).
ARRAY ADD in(), value ,out()	Dies fügt den Wert „value“ zu jedem Element der Matrix in() hinzu (oder hängt ihn für Zeichenfolgen an) und speichert das Ergebnis in out(). Funktioniert für Arrays beliebiger Dimensionen von Zeichenfolgen sowie für Ganzzahlen und Gleitkommazahlen (Konvertierung zwischen Ganzzahlen und Gleitkommazahlen möglich). Die Einstellung von num auf 0 oder „“ ist optimiert und stellt eine schnelle Methode zum Kopieren eines gesamten Arrays dar. in() und out() können dasselbe Array sein.

ARRAY INSERT
targetarray(), [d1] [,d2] [,d3]
[,d4] [,d5] , sourcearray()
	Dies ist das Gegenteil von ARRAY SLICE, hat eine sehr ähnliche Syntax und ermöglicht es Ihnen beispielsweise, einen einzelnen Vektor in einem Array von Vektoren mit einem einzigen Befehl oder ein eindimensionales Array von Zeichenfolgen in ein zweidimensionales Array von Zeichenfolgen zu ersetzen. Die Arrays können numerisch oder Zeichenfolgen sein, und „sourcearray” und „destinationarray” müssen identisch sein (Hinweis: Bei numerischen Arrays kann zwischen Ganzzahlen und Gleitkommazahlen konvertiert werden).
	
	Beispiel:
	`OPTION BASE 1`
	`DIM targetarray(3,4,5)`
	`DIM sourcearray(4)=(1,2,3,4)`
	`ARRAY INSERT targetarray(), 2, , 3, sourcearray()`
	Setzt die Elemente 2,1,3 = 1 und 2,2,3 = 2 und 2,3,3 = 3 und 2,4,3 = 4

ARRAY SET value, array()	Setzt alle Elemente in array() auf den Wert „value“. Der Wert kann eine Zahl oder eine Zeichenfolge sein, und „array“ muss identisch sein (Hinweis: Konvertierung zwischen Ganzzahlen und Gleitkommazahlen möglich). Beachten Sie, dass dies die schnellste Methode ist, um ein Array zu löschen, indem es auf Null oder eine leere Zeichenfolge gesetzt wird.

ARRAY SLICE sourcearray(), [d1]
[,d2] [,d3] [,d4] [,d5]
Zielarray()
	Dieser Befehl kopiert einen bestimmten Satz von Werten aus einem mehrdimensionalen Array in ein eindimensionales Array. Dies ist wesentlich schneller als die Verwendung einer FOR-Schleife. Der Ausschnitt wird durch Angabe eines Werts für alle Indizes des Quellarrays außer einem festgelegt, und der Befehl sollte so viele Indizes enthalten, einschließlich des leeren, wie das Quellarray Dimensionen hat. Die Arrays können numerisch oder Zeichenfolgen sein, und „sourcearray” und „destinationarray” müssen identisch sein (Hinweis: Bei numerischen Arrays kann zwischen Ganzzahlen und Gleitkommazahlen konvertiert werden).
	Beispiel
	`OPTION BASE 1`
	`DIM a(3,4,5) DIM`
	`b(4)`
	`ARRAY SLICE a(), 2, , 3, b()`
	Kopiert die Elemente 2,1,3 und 2,2,3 und 2,3,3 und 2,4,3 in das Array b().

ARC x, y, r1, [r2], a1, a2 [, c]	Zeichnet einen Kreisbogen mit einer bestimmten Farbe und Breite zwischen zwei Radien (in Grad definiert). Die Parameter für den Befehl ARC sind: x: X-Koordinate des Mittelpunkts des Bogens y: Y-Koordinate des Mittelpunkts des Bogens r1: Innenradius des Bogens r2: Außenradius des Bogens – kann weggelassen werden, wenn der Bogen 1 Pixel breit ist a1: Startwinkel des Bogens in Grad a2: Endwinkel des Bogens in Grad c: Farbe des Bogens (wenn nicht angegeben, wird standardmäßig die Vordergrundfarbe verwendet) Null Grad entspricht der 12-Uhr-Position.

AUTOSAVE oder AUTOSAVE CRUNCH oder AUTOSAVE APPEND oder AUTOSAVE N	Rufen Sie den automatischen Programmeingabemodus auf. Dieser Befehl übernimmt Textzeilen aus der seriellen Eingabe der Konsole und speichert sie im Programmspeicher. Dies ist eine Möglichkeit, ein BASIC-Programm auf den Raspberry Pi Pico zu übertragen. Das zu übertragende Programm kann in einen Terminalemulator eingefügt werden, und dieser Befehl erfasst den Textstrom und speichert ihn im Programmspeicher. Er kann auch zum direkten Eingeben eines kleinen Programms über die Konsoleneingabe verwendet werden. Dieser Modus wird durch Eingabe von Strg-Z oder F1 beendet, wodurch die empfangenen Daten in den Programmspeicher übertragen werden und das vorherige Programm überschreiben. Verwenden Sie F2, um den Modus zu beenden und das Programm sofort auszuführen. Die Option CRUNCH weist MMBasic an, vor dem Speichern alle Kommentare, Leerzeilen und unnötigen Leerzeichen aus dem Programm zu entfernen. Dies kann bei großen Programmen verwendet werden, damit sie in den begrenzten Speicher passen. CRUNCH kann mit dem einzelnen Buchstaben C abgekürzt werden. Die Option APPEND lässt das vorhandene Programm unverändert und hängt die neuen Daten aus der seriellen Eingabe an dessen Ende an. Die Option N führt die automatische Speicherung wie gewohnt aus, gibt die Daten jedoch nicht an die Konsole zurück. Sie wartet auf das Senden des ersten Zeichens und verwendet dann einen Timer, der überprüft, ob zwischen den Zeichen mehr als 100 ms liegen. Wenn diese Verzögerung festgestellt wird, wechselt sie zurück in den normalen Eingabemodus und gibt die Meldung „Enter ctrl-Z, F1 oder F2 zum Beenden“ Sie können dann weitere Zeichen eingeben, die gespeichert werden sollen, oder einen der normalen Befehle zum Beenden verwenden. Dieser Befehl kann jederzeit mit Strg-C abgebrochen werden, wodurch der Programmspeicher unverändert bleibt.
BACKLIGHT n [,DEFAULT] BACKLIGHT n [,FreqInHz]	NICHT-VGA- ODER HDMI-VERSIONEN Stellt die Hintergrundbeleuchtung des Displays ein, gültige Werte sind 0 bis 100. Wenn DEFAULT angegeben ist, stellt die Firmware die Hintergrundbeleuchtung beim Einschalten automatisch auf diesen Wert ein. Dies ist besonders nützlich im Batteriebetrieb, wo eine Verringerung der Hintergrundbeleuchtung die Batterielebensdauer erheblich verlängern kann. Einige Schaltungen sind zu langsam, um die Standard-PWM-Frequenz der Hintergrundbeleuchtung zu verwenden, die gewählt wurde, um Interferenzen mit dem Audio zu vermeiden. In diesem Fall kann eine Benutzerfrequenz angegeben werden. Dies ist eine temporäre Option, die bei jedem Neustart neu eingestellt werden muss.
BEZIER x%(),y%() [,n] [,Farbe]	Zeichnet eine Bezier-Kurve mit einer unbegrenzten Anzahl von Kontrollpunkten. „x%()” und „y%()” sind eindimensionale Integer-Arrays, die die Koordinaten der Kontrollpunkte enthalten. Die Bezier-Kurve beginnt immer am ersten Kontrollpunkt. Die Arrays müssen die gleiche Dimensionalität haben. Der optionale Parameter „n“ legt fest, wie viele Kontrollpunkte für die Darstellung verwendet werden sollen. Wird er weggelassen, bestimmen die Größe von „x%()“ und „y%()“ die Anzahl. N muss >=2 und <= der Array-Größe sein. Der optionale Parameter „colour“ definiert die Farbe, in der die Kurve gezeichnet wird (Standard ist Weiß).
BIT(var%, bitno) = Wert	Setzt ein bestimmtes Bit (0-63) in einer ganzzahligen Variablen. „Wert“ kann 0 oder 1 sein. Siehe auch die Funktion BIT.
BITBANG	Ersetzt durch den Befehl DEVICE. Aus Kompatibilitätsgründen kann BITBANG weiterhin in Programmen verwendet werden und wird automatisch in DEVICE konvertiert.
BLIT	BLIT ist eine einfache Speicheroperation, bei der Daten von einem Display oder Speicher auf ein Display oder einen Speicher kopiert werden. Hinweise: • Es stehen 32 Puffer von #1 bis #32 zur Verfügung. • Bei der Angabe der Puffernummer ist das Symbol # optional. • Alle anderen Argumente sind in Pixeln angegeben.

BLIT READ [#]b, x, y, w, h	BLIT READ kopiert einen Teil der Anzeige in den Speicherpuffer „#b“. Die Quellkoordinaten sind „x“ und „y“, die Breite des zu kopierenden Anzeigebereichs ist „w“ und die Höhe ist „h“. Bei Verwendung dieses Befehls wird der Speicherpuffer automatisch erstellt und ausreichend Speicher zugewiesen. Dieser Puffer kann mit dem Befehl BLIT CLOSE freigegeben und der Speicher zurückgewonnen werden.
BLIT WRITE [#]b, x, y [,mode]	BLIT WRITE kopiert den Speicherpuffer „#b” auf das Display. Die Zielkoordinaten sind „x” und „y”. Der optionale Parameter „mode” ist standardmäßig auf 0 gesetzt und legt fest, wie die gespeicherten Bilddaten beim Auslesen verändert werden. Es handelt sich um die bitweise UND-Verknüpfung der folgenden Werte: &B001 = von links nach rechts gespiegelt &B010 = von oben nach unten gespiegelt &B100 = keine transparenten Pixel kopieren
BLIT LOAD[BMP] [#]b, fname$ [,x] [,y] [,w] [,h]	BLIT LOAD lädt einen Blit-Puffer aus einer 24-Bit-BMP-Bilddatei. x,y definieren die Startposition im Bild, an der mit dem Laden begonnen werden soll, und w,h geben die Breite und Höhe des zu ladenden Bereichs an. Dieser Befehl funktioniert auf den meisten Anzeigetafeln (nicht nur auf Tafeln mit dem ILI9341-Controller). Beispiel: BLIT LOAD #1,"image1", 50,50,100,100 lädt einen Bereich von 100 Pixeln im Quadrat mit der oberen linken Ecke bei 50,50 aus dem Bild image1.bmp
BLIT CLOSE [#]b	BLIT CLOSE schließt den Speicherpuffer „#b“, damit er für einen weiteren BLIT READ-Vorgang verwendet werden kann, und gibt den verwendeten Speicher wieder frei.
	<u>NICHT VGA- ODER HDMI-VERSIONEN</u>
BLIT MERGE Farbe, x, y, w, h	Kopiert einen Bereich des Framebuffers, der durch die Pixelkoordinaten „x” und „y” der oberen linken Ecke definiert ist und eine Breite von „w” und eine Höhe von „h” hat, auf das LCD-Display. Als Teil des Kopiervorgangs überlagert es das LCD-Display mit Pixeln aus dem Layer-Puffer, die nicht auf die angegebene „Farbe” eingestellt sind. Die Farbe wird als Zahl zwischen 0 und 15 angegeben, die Folgendes darstellt: Schwarz, Blau, Myrte, Kobalt, Mittelgrün, Cerulean, Grün, Cyan, Rot, Magenta, Rost, Fuchsia, Braun, Flieder, Gelb und Weiß Erfordert sowohl einen Framebuffer als auch einen Layer-Puffer, um zu funktionieren. Wartet automatisch auf die Bildausblendung, bevor der Kopiervorgang auf ILI9341-, ST7789_320- und ILI9488-Displays gestartet wird.
BLIT FRAMEBUFFER von, zu, xin, yin, xout, yout, Breite, Höhe [,Farbe]	Kopiert einen Bereich eines bestimmten „from”-Framebuffers N, F oder L in einen anderen „to”-Framebuffer N, F oder L. „xin” und „yin” definieren die obere linke Ecke des Bereichs „width” und „height” auf dem zu kopierenden Quell-Framebuffer. „xout” und „yout” definieren die obere linke Ecke des Bereichs auf dem Ziel-Framebuffer, der die Kopie empfangen soll. Der optionale Parameter „colour” definiert eine Pixelfarbe auf der Quelle, die nicht kopiert wird. Wird dieser Parameter weggelassen, werden alle Pixel kopiert. Die Farbe wird als Zahl zwischen 0 und 15 angegeben, die Folgendes darstellt: Schwarz, Blau, Myrte, Kobalt, Mittelgrün, Cerulean, Grün, Cyan, Rot, Magenta, Rost, Fuchsia, Braun, Flieder, Gelb und Weiß Erfordert sowohl einen Framebuffer als auch einen Layer-Buffer, um zu funktionieren. Wartet automatisch auf die Bildausblendung, bevor die Kopie auf ILI9341-, ST7789_320- und ILI9488-Displays gestartet wird.
BLIT MEMORY Adresse, x, y [,col]	Kopiert einen Speicherbereich, der als gepacktes Array von Farbnibbles behandelt wird, in die aktuelle grafische Ausgabe, wie durch FRAMEBUFFER WRITE festgelegt. Die Farbe wird als Zahl zwischen 0 und 15 angegeben, die Folgendes darstellt: Schwarz, Blau, Myrte, Kobalt, Mittelgrün, Cerulean, Grün, Cyan, Rot, Magenta, Rost, Fuchsia, Braun, Flieder, Gelb und Weiß

BLIT COMPRESSED Adresse%, x, y [,Farbe] BLIT FLASH from, to, xin, yin, xout, yout, width, height [,colour]	Das erste Wort des Speicherbereichs, der bei „address%" beginnt, muss die Breite und Höhe des zu kopierenden Bereichs als 16-Bit-Ganzzahlen enthalten, wobei die Breite die unteren 16 Bits darstellt. Die Adresse muss an einer Wortgrenze ausgerichtet sein (durch 4 teilbar). Wenn der optionale Parameter „col" angegeben ist, wird diese bestimmte Farbe nicht kopiert. Wenn das oberste Bit der Breite oder Höhe auf 1 gesetzt ist, werden die Farbdaten als komprimiert behandelt (die verbleibenden 15 Bits werden als Breite und/oder Höhe verwendet). Der Komprimierungsalgorithmus ist einfach: Jedes Byte enthält eine Zählung im unteren Nibble (1–15) und eine Farbe im oberen Nibble (0–15). Falls mehr als 15 Pixel dieselbe Farbe haben, werden zusätzliche Bytes für diese Farbe verwendet. Verhält sich wie BLIT MEMORY, geht jedoch davon aus, dass die Daten komprimiert sind, und ignoriert das oberste Bit in der Breite und Höhe. Kopiert einen Bereich eines bestimmten „From"-Flash-Slots in einen „To"-Framebuffer N, F oder L. „Xin" und „Yin" definieren die obere linke Ecke des Bereichs mit der „Breite" und „Höhe" auf dem Flash-Slot, der kopiert werden soll. „xout" und „yout" definieren die obere linke Ecke des Bereichs auf dem Ziel-Framebuffer, der die Kopie empfangen soll. Der optionale Parameter „colour" definiert eine Pixelfarbe im Flash-Slot, die nicht kopiert wird. Wird dieser Parameter weggelassen, werden alle Pixel kopiert. Die Farbe wird als Zahl zwischen 0 und 15 angegeben, die Folgendes darstellt: Schwarz, Blau, Myrte, Kobalt, Mittelgrün, Cerulean, Grün, Cyan, Rot, Magenta, Rost, Fuchsia, Braun, Flieder, Gelb und Weiß
BLIT x1, y1, x2, y2, w, h	Kopieren Sie einen Abschnitt des Bildschirms in einen anderen Teil des Bildschirms. Die Quellkoordinaten sind „x1" und „y1". Die Zielkoordinaten sind „x2" und „y2". Die Breite des zu kopierenden Bildschirmbereichs ist „w" und die Höhe ist „h". Alle Argumente sind in Pixeln angegeben. Wenn die Ausgabe auf einem LCD-Panel erfolgt, muss es sich entweder um einen SSD19863-, ILI9341_8-, ILI9341-, ILI9488- (bei angeschlossenem MISO) oder ST7789_320-Controller handeln.
BOX x, y, w, h [, lw] [,c] [,fill]	Zeichnet ein Rechteck auf dem Display mit der oberen linken Ecke bei 'x' und 'y', einer Breite von 'w' Pixeln und einer Höhe von 'h' Pixeln. „lw" ist die Breite der Seiten des Kastens und kann Null sein. Der Standardwert ist 1. „c" gibt die Farbe an und ist standardmäßig auf die Standard-Vordergrundfarbe eingestellt, wenn nicht angegeben wurde. „fill" ist die Füllfarbe. Sie kann weggelassen oder auf -1 gesetzt werden, in diesem Fall wird das Feld nicht gefüllt wird. Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Kästchen entsprechend den Abmessungen des kleinsten Arrays. „x" und „y" müssen beide Arrays oder einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler ausgegeben. „w", „h", „lw", „c" und „fill" können entweder Arrays oder einzelne Variablen/Konstanten sein. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *„Grafikbefehle und -funktionen*".
BYTE(var$, byteno)=value	Setzt ein bestimmtes Byte in einer Zeichenfolge auf einen ganzzahligen Wert. „Wert" kann im Bereich von 0 bis 255 liegen. Die byteno kann zwischen 1 und der aktuellen Länge der Zeichenfolgenvariablen liegen. Dies entspricht MID$(var$,byteno,1)=CHR$(value), wird jedoch viel schneller ausgeführt. Siehe auch die Funktion BYTE.

CALL usersubname$ [,usersubparameters,..]	Dies ist eine effiziente Methode, um benutzerdefinierte Unterprogramme programmgesteuert aufzurufen (siehe auch die Funktion CALL()). In vielen Fällen können Sie damit komplexe SELECT- und IF THEN ELSEIF ENDIF-Klauseln vermeiden und die Verarbeitung erfolgt wesentlich effizienter. „usersubname$" kann eine beliebige Zeichenfolge, Variable oder Funktion sein, die zum Namen einer normalen Benutzer-Subroutine (kein integrierter Befehl) aufgelöst wird. Die „usersubparameters" sind dieselben Parameter, die zum direkten Aufruf der Unterroutine verwendet würden. Eine typische Anwendung wäre das Schreiben einer beliebigen Art von Emulator, bei dem eine von einer großen Anzahl von Unterroutinen in Abhängigkeit von einer Variablen aufgerufen werden soll. Es bietet auch die Möglichkeit, einen Unterroutinenamen als Variable an eine andere Unterroutine oder Funktion zu übergeben.
	NICHT VGA- ODER HDMI-VERSIONEN
KAMERA	Befehl zur Unterstützung des OV7670-Kameramoduls.
CAMERA OPEN XLKpin, PLKpin, HSpin, VSCpin, RETpin, D0pin	Dies initialisiert die Kamera, gibt einen 12-MHz-Takt auf XLK (PWM) aus und überprüft, ob sie Signale auf PLK, VS und HS korrekt empfängt. Die Kamera ist auf eine Auflösung von 160 x 120 (QQVGA) eingestellt, was dem Maximum entspricht, das innerhalb der Grenzen des verfügbaren Speichers erreichbar ist. Aktivieren Sie OPTION SYSTEM I2C in der PicoMite-Firmware und verbinden Sie SCL und SDA mit den entsprechenden Pins (auf dem Kameramodul möglicherweise mit SIOC und SIOD gekennzeichnet). Diese Verbindungen müssen einen Pullup auf 3,3 V haben – empfohlen wird 2K7. Andere Pins werden gemäß dem Befehl OPEN verdrahtet. (Hinweis: VS kann auf Ihrem Modul mit VSYNC, HS mit HREF, PLK mit PCLK, RET mit RESET und XLK mit XCLK gekennzeichnet sein. D0pin definiert den Beginn eines Bereichs von 8 aufeinanderfolgenden Pins (z. B. GP0 – GP7).
KAMERA-AUFNAHME [Skala, [x , y]]	Hiermit wird ein Bild von der Kamera (RGB565) aufgenommen und auf einem LCD-Bildschirm angezeigt. Damit der Befehl funktioniert, muss ein SPI-LCD angeschlossen und aktiviert sein. (ILI9341 und ST7789_320 empfohlen). Die Skalierung ist standardmäßig auf 1 und x, y jeweils auf 0 eingestellt. Standardmäßig wird ein Bild mit einer Größe von 160 x 120 Pixeln auf dem LCD-Bildschirm ausgegeben, wobei die obere linke Ecke bei 0,0 auf dem LCD-Bildschirm liegt. Wenn Sie den Maßstab auf 2 einstellen, wird das Bild auf einem 320 x 240 Pixel großen Bildschirm angezeigt. Durch Einstellen der Parameter x und y wird die obere linke Ecke des Bildes auf dem LCD-Bildschirm versetzt. Die Aktualisierungsrate in einer Endlosschleife beträgt 7 FPS auf dem Display im Maßstab 1:1 und 5 FPS skaliert auf 320 x 240. Angenommen, das Display ist mit MISO verkabelt, dann ist es möglich, das Bild mit dem Befehl SAVE IMAGE auf der Festplatte zu speichern.
CAMERA CLOSE	Schließt das Kamerasubsystem und gibt alle im Befehl OPEN zugewiesenen Pins frei.
KAMERA WECHSELN image%(),change! [,scale [,x ,y]]	Die Kamera-Firmware kann mit diesem Befehl auch Bewegungen im Sichtfeld der Kamera erkennen. Dazu wird die Kamera im YUV-Modus statt im RGB-Modus betrieben. Dies hat den Vorteil, dass die Intensitäts- und Farbinformationen getrennt sind und für ein Graustufenbild mit 256 Stufen nur ein Byte benötigt wird, was für die Bewegungserkennung ideal ist. image% ist ein Array der Größe 160x120 Bytes (DIM image%(160,120/8-1)). Beim Aufruf des Befehls enthält es ein gepacktes 8-Bit-Graustufenbild. Die Variable change! gibt den Prozentsatz zurück, um den sich das Bild seit dem letzten Aufruf des Befehls verändert hat. Optional wird bei Einstellung von „scale" das Bilddelta auf dem Bildschirm ausgegeben, d. h. die Differenz zwischen dem vorherigen Bild und diesem Bild. Wie beim Befehl CAPTURE kann das Delta-Bild nach Bedarf skaliert und positioniert werden. Wenn der Parameter „scale" weggelassen wird, wird das LCD durch diesen Befehl nicht aktualisiert.

KAMERATEST tnum	Aktiviert oder deaktiviert ein Testsignal von der Kamera. tnum=2 erzeugt Farbbalken und tnum=0 stellt die visuelle Eingabe wieder her.
CAMERA REGISTER reg%, data	Setzt das Register „reg%“ in der Kamera auf den Wert „data%“. Bei Verwendung meldet der Befehl den vorherigen Wert an die Konsole und bestätigt automatisch, dass der neue Wert wie gewünscht gesetzt wurde. Die Farbwiedergabe der Kamera ist nach der Initialisierung angemessen, könnte aber wahrscheinlich durch die Anpassung verschiedener Kameraregister noch weiter verbessert werden.
CAT S$, N$	Verbindet die Zeichenfolgen, indem N$ an S$ angehängt wird. Dies entspricht funktional `S$ = S$ + N$,` arbeitet jedoch etwas schneller.
CHAIN fname$ [cmdline$]	Ermöglicht es einem Programm, ein anderes Programm unter Beibehaltung des variablen Speicherplatzes auszuführen – es wird empfohlen, den Befehl in einem Programm der obersten Ebene und nicht innerhalb einer Unterroutine zu verwenden. Wenn der optionale Parameter „cmdline$“ angegeben ist, wird dieser in MM.CMDLINE$ an das verkettete Programm übergeben.
CHDIR dir$	Ändert das aktuelle Arbeitsverzeichnis auf dem Standardlaufwerk zu „dir$”. Der spezielle Eintrag „.“ steht für das übergeordnete Verzeichnis des aktuellen Verzeichnisses und „.“ steht für das aktuelle Verzeichnis. „/“ ist das Stammverzeichnis.
CIRCLE x, y, r [,lw] [, a] [, c] [, fill]	Zeichnet einen Kreis mit dem Mittelpunkt „x” und „y” und dem Radius „r” auf dem Bildschirm. „lw” ist optional und steht für die Linienbreite (Standardwert ist 1). „c“ ist die optionale Farbe und wird standardmäßig auf die aktuelle Vordergrundfarbe gesetzt, wenn keine Angabe erfolgt. Das optionale „a“ ist eine Gleitkommazahl, die das Seitenverhältnis definiert. Wenn das Seitenverhältnis nicht angegeben wird, ist der Standardwert 1,0, was einen Standardkreis ergibt. „fill“ ist die Füllfarbe und kann weggelassen oder auf -1 gesetzt werden. In diesem Fall wird das Feld nicht gefüllt. Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Kreise, die durch die Abmessungen des kleinsten Arrays bestimmt wird. „x”, „y“ und „r“ müssen entweder alle Arrays oder alle einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler ausgegeben. „lw“, „a“, „c“ und „fill“ können entweder Arrays oder einzelne Variablen/Konstanten sein. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *„ Grafikbefehle und -funktionen*”.
CLEAR	Löscht alle Variablen und gibt den von ihnen belegten Speicher frei. Siehe ERASE zum Löschen bestimmter Array-Variablen.
CLOSE [#]fnbr [,[#]fnbr] …	Schließt die zuvor mit der Dateinummer „#fnbr” geöffneten Dateien. Das # ist optional. Siehe auch den Befehl OPEN.
CLS [Farbe]	Löscht den Bildschirm des LCD-Panels. Optional kann „Farbe” angegeben werden, die als Hintergrundfarbe beim Löschen des Bildschirms verwendet wird.
CMM2 LOAD oder CMM2 RUN	Lädt und/oder führt ein Programm von der Festplatte unter Verwendung des CMM2-Programmlademekanismus aus. Dies umfasst eine aggressive Komprimierung des Programms und unterstützt #INCLUDE-Dateien und #DEFINE-Textersetzungen. Dies kann zur Kompatibilität mit CMM2-Programmen oder zur Strukturierung von Programmen in separate Module verwendet werden. Es ist wichtig zu beachten, dass bei Verwendung alle Bearbeitungen von Programmen offline oder direkt von und auf die Festplatte erfolgen müssen, da die Quelldateien aus der mit diesen Befehlen geladenen Version nicht rekonstruiert werden können.
COLOUR fore [, back] oder COLOR fore [, back]	Legt die Standardfarbe für Befehle (PRINT usw.) fest, die auf dem angeschlossenen LCD-Bildschirm angezeigt werden. „fore“ ist die Vordergrundfarbe, „back“ ist die Hintergrundfarbe. Der Hintergrund ist optional und wird, wenn nicht angegeben, standardmäßig auf eine zuvor festgelegte Hintergrundfarbe oder, falls zuvor nicht geändert, auf Schwarz gesetzt.

COLOUR MAP inarray%(), outarray%() [,colourmap%()]	Dieser Befehl generiert RGB888-Farben in outarray% aus Farbcodes (0-15) in inarray%. Wenn der optionale Parameter colourmap% verwendet wird, muss dieser 16 Elemente lang sein. In diesem Fall werden die Werte in inarray% den Farben für diesen Indexwert in colourmap% zugeordnet.
CONFIGURE cfg CONFIGURE LIST	Konfiguriert eine Karte gemäß dem in „cfg“ angegebenen Äquivalent von OPTION RESET). Listet alle für die Firmware-Version verfügbaren Konfigurationen auf.
CONST id = Ausdruck [, id = Ausdruck] … usw.	Erstellt einen konstanten Bezeichner, der nach seiner Erstellung nicht mehr geändert werden kann. „id” ist der Bezeichner, der denselben Regeln wie für Variablen folgt. Der Identifikator kann ein Typ-Suffix (!, %, oder $) haben, dies ist jedoch nicht erforderlich. Wenn angegeben wird, muss es mit dem Typ von „Ausdruck” übereinstimmen. „Ausdruck” ist der Wert des Bezeichners und kann ein normaler Ausdruck (einschließlich benutzerdefinierter Funktionen) sein, der bei der Erstellung der Konstante ausgewertet wird. Eine außerhalb einer Subroutine oder Funktion definierte Konstante ist global und kann im gesamten Programm gesehen werden. Eine innerhalb einer Subroutine oder Funktion definierte Konstante ist lokal für diese Routine und verdeckt eine globale Konstante mit dem gleichen Namen.
CONTINUE	Setzt die Ausführung eines Programms fort, das durch eine END-Anweisung, einen Fehler oder STRG-C angehalten wurde. Das Programm wird mit der nächsten Anweisung nach dem vorherigen Haltepunkt neu gestartet. Beachten Sie, dass es nicht immer möglich ist, das Programm korrekt fortzusetzen – dies gilt insbesondere für komplexe Programme mit Grafiken, verschachtelten Schleifen und/oder verschachtelten Unterprogrammen und Funktionen.
CONTINUE DO oder CONTINUE FOR	Springen Sie zum Ende einer DO/LOOP- oder einer FOR/NEXT-Schleife. Die Schleifenbedingung wird dann geprüft und wenn sie noch gültig ist, wird die Schleife mit der nächsten Iteration fortgesetzt.
COPY fname1$ TO fname2$ COPY fname$ TO dirname$	Kopieren Sie eine Datei von „fname1$” nach „fname2$”. Beide sind Zeichenfolgen. Ein Verzeichnispfad kann sowohl in „fname$” als auch in „fname$” verwendet werden. Wenn sich die Pfade unterscheiden, wird die in „fname$” angegebene Datei mit dem angegebenen Dateinamen in den in „fname2$” angegebenen Pfad kopiert. Die Dateinamen können die Laufwerksangabe enthalten, wenn Sie auf ein nicht aktives Laufwerk kopieren oder von einem nicht aktiven Laufwerk kopieren (siehe Befehl DRIVE). Kopieren mit Platzhaltern. Die Massenkopie wird ausgelöst, wenn fname$ ein „*“- oder ein „?“-Zeichen enthält. dirname$ muss ein gültiger Verzeichnisname sein und darf NICHT mit einem Schrägstrich enden.
CPU-NEUSTART	Erzwingt einen Neustart der Prozessoren. Dadurch werden alle Variablen gelöscht und alles zurückgesetzt (z. B. Timer, COM-Ports, I2C usw.), ähnlich wie beim Einschalten, jedoch ohne den Startbildschirm. Wenn die OPTION AUTORUN gesetzt wurde, wird das Programm an der angegebenen Flash-Position oder im Programmspeicher neu gestartet.
CPU-RUHEZUSTAND n	Veranlasst die Prozessoren, für „n“ Sekunden in den Ruhezustand zu wechseln. Beachten Sie, dass die CPU keinen echten Energiesparmodus hat, sodass die Energieeinsparung begrenzt ist.
CSUB-Name [Typ [, Typ] …] hex [[hex[…] hex [[hex[…] END CSUB	Definiert den Binärcode für ein eingebettetes Maschinencode-Programmmodul, das in C oder ARM-Assembler geschrieben ist. Das Modul erscheint in MMBasic als Befehl „name” und kann wie ein integrierter Befehl verwendet werden. In einem Programm können mehrere eingebettete Routinen verwendet werden, wobei jede ein anderes Modul mit einem anderen „name” definiert. Das erste „Hex“-Wort ist ein 32-Bit-Wort, das den Offset in Bytes vom Beginn des CSUB bis zum Einstiegspunkt der eingebetteten Routine (in der Regel die Funktion

	main()). Die folgenden Hex-Wörter sind der kompilierte Binärcode für das Modul. Diese werden beim Speichern des Programms automatisch in MMBasic programmiert. Jedes „Hex“ muss genau acht Hexadezimalziffern umfassen, die die Bits in einem 32-Bit-Wort darstellen, und durch ein oder mehrere Leerzeichen oder Zeilenumbrüche voneinander getrennt sein. Der Befehl muss mit einem passenden END CSUB abgeschlossen werden. Fehler im Datenformat werden bei der Ausführung des Programms gemeldet. Während der Ausführung überspringt MMBasic alle CSUB-Befehle, sodass diese an beliebiger Stelle im Programm platziert werden können. Der Typ jedes Parameters kann in der Definition angegeben werden. Beispiel: `CSUB MySub integer, integer, string` Dies gibt an, dass es drei Parameter geben wird, wobei die ersten beiden Ganzzahlen und der dritte eine Zeichenfolge sind. Hinweis: • Es können bis zu zehn Argumente angegeben werden („arg1”, „arg2” usw.). • Wenn eine Variable oder ein Array als Argument angegeben wird, erhält die C-Routine einen Zeiger auf den der Variable oder dem Array zugewiesenen Speicherplatz und kann diesen Speicherplatz ändern, um einen Wert an den Aufrufer zurückzugeben. Im Falle von Arrays sollten diese mit leeren Klammern übergeben werden, z. B. arg(). In der CSUB wird das Argument als Zeiger auf das erste Element des Arrays übergeben. • Konstanten und Ausdrücke werden als Zeiger auf einen temporären Speicherplatz, der den Wert enthält, an die eingebettete C-Routine übergeben.
DATA Konstante[,Konstante]..	Speichert numerische und Zeichenfolgenkonstanten, auf die READ zugreifen kann. Im Allgemeinen sollten String-Konstanten in doppelte Anführungszeichen ("") gesetzt werden. Eine Ausnahme bildet der Fall, dass der String nur aus alphanumerischen Zeichen besteht, die keine MMBasic-Schlüsselwörter darstellen (wie THEN, WHILE usw.). In diesem Fall sind keine Anführungszeichen erforderlich. Numerische Konstanten können auch Ausdrücke wie 5 * 60 sein.
DATE$ = "DD-MM-YY[YY]" oder DATE$ = "DD/MM/YY[YY]" oder DATE$ =”YYYY-MM-DD” oder DATE$ = „JJJJ/MM/TT”	Stellen Sie das Datum der internen Uhr/des internen Kalenders ein. DD, MM und YY sind Zahlen, zum Beispiel: DATE$ = "28-7-2014" Mit OPTION RTC AUTO ENABLE startet die PicoMite-Firmware mit dem in RTC programmierten DATE$. Ohne OPTION RTC AUTO ENABLE startet die PicoMite-Firmware beim Einschalten mit dem Datum „01-01-2024”.
DEFINEFONT #Nbr hex [[hex[…] hex [[hex[…] END DEFINEFONT	Hiermit wird eine eingebettete Schriftart definiert, die zusammen mit den integrierten Schriftarten auf einem angeschlossenen LCD-Bildschirm verwendet werden kann oder diese ersetzt. Diese funktionieren genau wie die integrierten Schriftarten (d. h. sie werden mit dem Befehl FONT ausgewählt oder im Befehl TEXT angegeben). Eine Auswahl eingebetteter Schriftarten und eine vollständige Beschreibung ihrer Erstellung finden Sie im Ordner *„Embedded Fonts”* in der ZIP-Datei mit der PicoMite-Firmware. „#Nbr” ist die Referenznummer der Schriftart (von 1 bis 16). Sie kann mit einer integrierten Schriftart übereinstimmen. In diesem Fall ersetzt sie die integrierte Schriftart. Jedes „hex” muss genau acht Hexadezimalziffern umfassen und durch Leerzeichen oder Zeilenumbrüche vom nächsten getrennt sein. • Es können mehrere Zeilen mit „hex”-Wörtern verwendet werden, wobei der Befehl mit einem entsprechenden END DEFINEFONT beendet wird. • In einem Programm können mehrere eingebettete Schriftarten verwendet werden, wobei jede eine andere Schriftart mit einer anderen Schriftartnummer definiert. • Während der Ausführung überspringt MMBasic alle DEFINEFONT-Befehle, sodass diese an beliebiger Stelle im Programm platziert werden können. • Fehler im Datenformat werden beim Speichern des Programms gemeldet.

DEVICE BITSTREAM pinno, n_transitions, array%()	Dieser Befehl wird verwendet, um eine äußerst genaue Bitsequenz auf dem angegebenen Pin zu erzeugen. Der Pin muss zuvor als Ausgang eingerichtet und auf den erforderlichen Startpegel eingestellt worden sein. Hinweise: • Das Array enthält die Länge jedes Pegels im Bitstrom in Mikrosekunden. Die maximal zulässige Periode beträgt 65,5 ms. • Der erste Übergang erfolgt unmittelbar nach Ausführung des Befehls. • Der letzte Zeitraum im Array wird ignoriert, außer zur Definition der Zeit, bevor die Steuerung zum Programm oder zur Befehlszeile zurückkehrt. • Der Pin bleibt im Startzustand, wenn die Anzahl der Übergänge gerade ist, und im entgegengesetzten Zustand, wenn die Anzahl der Übergänge ungerade ist.
DEVICE CAMERA	Siehe Befehl CAMERA
GERÄT GAMEPAD	Siehe Befehl GAMEPAD
GERÄT HUMID	Siehe Befehl HUMID
GERÄT KEYPAD	Siehe Befehl KEYPAD
DEVICE MOUSE	Siehe MOUSE-Befehl
GERÄT LCD	Siehe Befehl LCD
DEVICE SERIALTX pinno, baudrate, ostring$	Gibt „ostring$“ als seriellen Datenstrom auf „pinno“ aus. „baudrate“ kann zwischen 110 und 230400 liegen (für 230400 muss die CPU möglicherweise übertaktet werden). Beachten Sie, dass das Programm während der Übertragung angehalten wird und Unterbrechungen ignoriert werden.
DEVICE SERIALRX pinno, baudrate, istring$, timeout_in_ms, status% [,nbr] [,terminators$]	Gibt serielle Daten auf „pinno” ein. „baudrate” kann zwischen 110 und 230400 liegen (für 230400 muss die CPU möglicherweise übertaktet werden). „status%” gibt Folgendes zurück: -1 = Zeitüberschreitung (Hinweis: Verwenden Sie LEN(istring$), um die Anzahl der empfangenen Zeichen anzuzeigen) 2 = Anzahl der angeforderten Zeichen erfüllt 3 = Endzeichen erfüllt „nbr“ gibt die Anzahl der Zeichen an, die empfangen werden müssen, bevor der Befehl zurückkehrt. „terminators$“ gibt ein oder mehrere einzelne Zeichen an, die zum Beenden des Empfangs verwendet werden können. Das Programm wird angehalten und Unterbrechungen werden ignoriert, während dieser Befehl ausgeführt wird.
GERÄT WII	Siehe Befehl WII
GERÄT WS2812	Siehe WS2812-Befehl
DIM [Typ] Deklaration [,Deklaration].. wobei „Deklaration” Folgendes ist: var [Länge] [Typ] [Init] „var” ist ein Variablenname mit optionalen Dimensionen „length” wird verwendet, um die maximale Größe der Zeichenfolge auf „n” zu setzen, wie in LENGTH n „type” ist entweder FLOAT, INTEGER oder STRING (dem Typ kann mit dem Schlüsselwort AS vorangestellt werden – wie in AS FLOAT)	Deklariert eine oder mehrere Variablen (d. h. macht den Variablennamen und seine Eigenschaften dem Interpreter bekannt). Wenn OPTION EXPLICIT verwendet wird (wie empfohlen), können Variablen nur mit den Befehlen DIM, LOCAL oder STATIC erstellt werden. Wenn diese Option nicht verwendet wird, ist die Verwendung des Befehls DIM optional, und wenn er nicht verwendet wird, wird die Variable automatisch erstellt, wenn sie zum ersten Mal referenziert wird. Der Typ der Variablen (d. h. Zeichenfolge, Gleitkomma oder Ganzzahl) kann auf drei Arten angegeben werden: Durch Verwendung eines Typ-Suffixes (d. h. !, % oder $ für Gleitkomma, Ganzzahl oder Zeichenfolge). Beispiel: `DIM nbr%, amount!, name$` Durch Verwendung eines der Schlüsselwörter FLOAT, INTEGER oder STRING unmittelbar nach dem Befehl DIM und vor der Auflistung der Variablen. Der angegebene Wert

„init" ist der Wert, mit dem die Variable initialisiert wird, und besteht aus:
= <Ausdruck>
Für eine einfache Variable wird ein Ausdruck verwendet, für ein Array eine Liste von durch Kommas getrennten Ausdrücken, die in Klammern stehen.

Beispiele:
DIM nbr(50)
DIM INTEGER nbr(50) DIM name AS STRING
DIM a, b$, nbr(100), strn$(20)
DIM a(5,5,5), b(1000)

DIM strn$(200) LENGTH 20
DIM STRING strn(200) LENGTH 20
DIM a = 1234, b = 345

DIM STRING strn = "text"
DIM x%(3) = (11, 22, 33, 44)

Der Typ gilt dann für alle aufgeführten Variablen (d. h. er muss nicht wiederholt werden). Beispiel:

```
DIM STRING first_name, last_name, city
```

Es wird die Microsoft-Konvention verwendet, nach jeder Variablen das Schlüsselwort „AS" und das Typ-Schlüsselwort (d. h. FLOAT, INTEGER oder STRING) anzugeben. Bei dieser Methode muss der Typ für jede Variable angegeben werden und kann von Variable zu Variable geändert werden.
Beispiel:

```
DIM Betrag AS FLOAT, Name AS STRING
```

Gleitkomma- oder Ganzzahlvariablen werden bei ihrer Erstellung auf Null gesetzt, und Zeichenfolgen werden auf eine leere Zeichenfolge (d. h. "") gesetzt. Sie können den Wert der Variablen initialisieren, indem Sie ein Gleichheitszeichen (=) und einen Ausdruck nach der Variablendefinition verwenden. Beispiel:

```
DIM STRING city = "Perth", house = "Brick"
```

Der Initialisierungswert kann ein Ausdruck (einschließlich anderer Variablen) sein und wird bei Ausführung des DIM-Befehls ausgewertet. Weitere Beispiele für die Syntax finden Sie im Kapitel *„Definieren und Verwenden von Variablen*".

Neben einfachen Variablen deklariert der Befehl DIM auch Array-Variablen (d. h. indizierte Variablen mit mehreren Dimensionen). Nach dem Namen der Variablen werden die Dimensionen durch eine Liste von Zahlen angegeben, die durch Kommas getrennt und in Klammern gesetzt sind. Beispiel:

```
DIM array(10, 20)
```

Jede Zahl gibt den Indexbereich in jeder Dimension an. Normalerweise beginnt die Indizierung jeder Dimension bei 0, aber mit dem Befehl OPTION BASE kann dies auf 1 geändert werden.

Das obige Beispiel gibt ein zweidimensionales Array mit 11 Elementen (0 bis 10) in der ersten Dimension und 21 Elementen (0 bis 20) in der zweiten Dimension an. Die Gesamtzahl der Elemente beträgt 231, und da jede Gleitkommazahl 8 Byte benötigt, werden insgesamt 1848 Byte Speicher zugewiesen.

Zeichenfolgen belegen standardmäßig 255 Byte (d. h. Zeichen) Speicherplatz für jedes Element, was bei der Definition von Zeichenfolgen-Arrays schnell zu einer hohen Speicherauslastung führen kann. In diesem Fall kann das Schlüsselwort LENGTH verwendet werden, um die jedem Element zuzuweisende Speichermenge und damit die maximale Länge der zu speichernden Zeichenfolge anzugeben. Diese Zuweisung („n") kann zwischen 1 und 255 Zeichen betragen.

Beispiel: `DIM STRING s(5, 10)` deklariert ein String-Array mit 66 Elementen, das 16.896 Byte Speicherplatz belegt, während

```
DIM STRING s(5, 10) LENGTH 20
```

nur 1.386 Byte Speicherplatz belegt. Beachten Sie, dass die jedem Element zugewiesene Speichermenge n + 1 beträgt, da das zusätzliche Byte zur Verfolgung der tatsächlichen Länge der in jedem Element gespeicherten Zeichenfolge verwendet wird.

Wenn einem Element des Arrays eine Zeichenfolge zugewiesen wird, die länger als 'n' ist, wird ein Fehler ausgegeben. Ansonsten verhalten sich mit dem Schlüsselwort LENGTH erstellte Zeichenfolgen-Arrays genauso wie andere Zeichenfolgen-Arrays. Dieses Schlüsselwort kann auch mit Nicht-Array-Zeichenfolgenvariablen verwendet werden, spart jedoch keinen Speicherplatz.
Im obigen Beispiel können Sie auch die Microsoft-Syntax verwenden, bei der der Typ nach dem Längenqualifizierer angegeben wird. Beispiel:

```
DIM s(5, 10) LENGTH 20 AS STRING
```

Der Längenparameter kann für einfache (nicht als Array definierte) Zeichenfolgen mit den folgenden Einschränkungen verwendet werden: Beim RP2040 wird eine Länge von mehr als 9 ignoriert und die Standardlänge von 255 Byte zugewiesen. Beim RP2350 wird eine Länge von mehr als 15 ignoriert und die Standardlänge von 255 Byte zugewiesen. Wenn die Länge kleiner oder gleich dem Grenzwert ist, wird die Zeichenfolge in der Variablen

	Header gespeichert und 256 Bytes werden gespeichert. Dies ist wahrscheinlich besonders nützlich für kurze konstante Strings. Arrays können auch bei ihrer Deklaration initialisiert werden, indem am Ende der Deklaration ein Gleichheitszeichen (=) gefolgt von einer in Klammern gesetzten Werteliste hinzugefügt wird. Beispiel: `DIM INTEGER nbr(4) = (22, 44, 55, 66, 88)` oder `DIM s$(3) = ("foo", "boo", "doo", "zoo")` Beachten Sie, dass die Anzahl der Initialisierungswerte mit der Anzahl der Elemente im Array übereinstimmen muss, einschließlich des durch OPTION BASE festgelegten Basiswerts. Wenn ein mehrdimensionales Array initialisiert wird, wird zuerst die erste Dimension initialisiert, dann die zweite usw. Beachten Sie auch, dass die Initialisierungswerte nach dem LENGTH-Qualifizierer (falls verwendet) und nach der Typdeklaration (falls verwendet) stehen müssen.
DO <Anweisungen> LOOP	Diese Struktur wird endlos wiederholt; der Befehl EXIT DO kann verwendet werden, um die Schleife zu beenden, oder die Steuerung muss explizit durch Befehle wie GOTO oder EXIT SUB (in einer Unterroutine) außerhalb der Schleife übertragen werden.
DO WHILE Ausdruck <Anweisungen> LOOP	Wiederholt die Schleife, solange „Ausdruck" wahr ist (dies entspricht der älteren WHILE-WEND-Schleife). Ist der Ausdruck zu Beginn falsch, werden die Anweisungen in der Schleife nicht ausgeführt, auch nicht einmal.
DO <Anweisungen> LOOP UNTIL Ausdruck	Wiederholt die Schleife, bis der Ausdruck nach UNTIL wahr ist. Da die Prüfung am Ende der Schleife erfolgt, werden die Anweisungen innerhalb der Schleife mindestens einmal ausgeführt, auch wenn der Ausdruck wahr ist.
DO <Anweisungen> LOOP WHILE Ausdruck	Wiederholt die Schleife, bis der Ausdruck nach WHILE falsch ist. Da die Prüfung am Ende der Schleife erfolgt, werden die Anweisungen innerhalb der Schleife mindestens einmal ausgeführt, auch wenn der Ausdruck falsch ist.
DRAW3D	IN DER WEBMITE-VERSION NICHT VERFÜGBAR Die 3D-Engine enthält Befehle zur Bearbeitung von 3D-Bildern, darunter das Einstellen der Kamera, das Erstellen, Ausblenden, Drehen usw. Eine vollständige Beschreibung finden Sie im Dokument *„The CMM2 3D engine.*pdf" im PicoMite-Firmware-Download.
DRIVE drive	Legt das aktive Laufwerk als „drive$" fest. „drive$" kann „A:" oder „B:" sein, wobei A das Flash-Laufwerk und B die SD-Karte ist, sofern diese konfiguriert ist.
EDIT oder EDIT fname$ oder EDIT FILE fname$	Ruft den Vollbild-Editor auf. Wenn ein Dateiname angegeben ist, lädt der Editor die Datei von der aktuellen Festplatte (A: oder B:) zum Bearbeiten und speichert sie beim Beenden mit F1 oder F2 auf der Festplatte. Wenn die Datei nicht existiert, wird sie beim Beenden erstellt. Das aktuelle Programm, das im Flash-Speicher gespeichert ist, bleibt davon unberührt. Wenn eine vorhandene Datei bearbeitet wird, wird beim Beenden ebenfalls eine Sicherungskopie mit der Endung .bak erstellt. Wenn fname$ eine andere Erweiterung als .bas enthält, wird die Farbcodierung während der Bearbeitung vorübergehend deaktiviert. Wenn keine Erweiterung angegeben ist, geht die Firmware von .bas aus. Durch das Bearbeiten einer Datei von der Festplatte können Nicht-Basic-Dateien wie HTML- oder Sprite-Dateien ohne Beschädigung während des Tokenisierungsprozesses bearbeitet werden, der beim Speichern im Flash-Speicher stattfindet. EDIT und EDIT fname$ können nur an der Befehlszeile aufgerufen werden. Wenn Sie eine Datei in einem Programm bearbeiten möchten, können Sie den Befehl EDIT FILE fname$ verwenden. Der Befehl muss im Programm der obersten Ebene und nicht innerhalb einer Subroutine verwendet werden. EDIT FILE fname$ unterscheidet sich von EDIT fname$ dadurch, dass es automatisch den gesamten Variablenbereich auf dem Laufwerk A: speichert und beim Beenden wiederherstellt. Der Befehl

	schlägt fehl, wenn auf dem Laufwerk A: nicht genügend freier Speicherplatz vorhanden ist. Bei einem RP2350 mit PSRAM wird der Variablenbereich in einem reservierten Bereich im PSRAM gespeichert, und das Laufwerk A: wird nicht verwendet. Weitere Informationen zur Verwendung des Editors finden Sie im Kapitel *Vollbild-Editor*.
ELSE	Führt eine optionale Standardbedingung in einer mehrzeiligen IF-Anweisung ein. Weitere Informationen finden Sie unter „Mehrzeilige IF-Anweisung".
ELSEIF Ausdruck THEN oder ELSE IF Ausdruck THEN	Führt eine optionale sekundäre Bedingung in einer mehrzeiligen IF-Anweisung ein. Weitere Informationen finden Sie unter „Mehrzeilige IF-Anweisung".
END [noend] oder END cmd$	Beenden Sie das laufende Programm und kehren Sie zur Eingabeaufforderung zurück. Wenn im Programm eine Subroutine namens MM.END vorhanden ist, wird diese immer dann ausgeführt, wenn das Programm mit einem tatsächlichen oder implizierten END-Befehl endet. Sie wird nicht ausgeführt, wenn das Programm mit dem Abbruchzeichen (d. h. Strg-C) beendet wird. Der optionale Parameter „noend" kann verwendet werden, um die Ausführung der Unterroutine MM.END zu blockieren, z. B. „END noend". Wenn „cmd$" angegeben ist, wird es nach Beendigung des Programms wie an der Eingabeaufforderung ausgeführt. Hinweis: Wenn „END cmd$" verwendet wird, aber eine Unterroutine MM.END vorhanden ist, wird diese ausgeführt und cmd$ ignoriert.
END CSUB	Markiert das Ende einer C-Subroutine. Siehe den Befehl CSUB. Jedes CSUB muss genau eine passende END CSUB-Anweisung haben.
END FUNCTION	Markiert das Ende einer benutzerdefinierten Funktion. Siehe den Befehl FUNCTION. Jede Funktion muss genau eine passende END FUNCTION-Anweisung haben. Verwenden Sie EXIT FUNCTION, wenn Sie aus einer Funktion innerhalb ihres Körpers zurückkehren müssen.

ENDIF
oder
END IF
	Beendet eine mehrzeilige IF-Anweisung.
	Weitere Informationen finden Sie unter mehrzeilige IF-Anweisung.

END SUB	Markiert das Ende einer benutzerdefinierten Subroutine. Siehe den Befehl SUB. Jede Subroutine muss genau eine passende END SUB-Anweisung haben. Verwenden Sie EXIT SUB, wenn Sie aus einer Subroutine innerhalb ihres Körpers zurückkehren müssen.
ERASE Variable [,Variable]..	Löscht globale Variablen und gibt den ihnen zugewiesenen Speicher frei. Dies funktioniert sowohl mit Array-Variablen als auch mit normalen (Nicht-Array-)Variablen. Arrays können mit leeren Klammern (z. B. `dat()`) oder einfach durch Angabe des Variablennamens (z. B. `dat`) angegeben werden. Verwenden Sie CLEAR, um alle Variablen gleichzeitig zu löschen (einschließlich Arrays).
ERROR [error_msg$]	Löst einen Fehler aus und beendet das Programm. Dies wird normalerweise beim Debuggen oder zum Abfangen von Ereignissen verwendet, die nicht auftreten sollten. 'error_msg$' ist optional und ist die Meldung, die auf der Konsole angezeigt werden soll.
EXECUTE Befehl$	Dies führt den Basic-Befehl „command$" aus. Die Verwendung sollte auf Basic-Befehle beschränkt sein, die sequenziell ausgeführt werden, da beispielsweise die GOTO-Anweisung nicht ordnungsgemäß funktioniert. Zu den getesteten und funktionierenden Elementen gehören GOSUB, Unterprogrammaufrufe und andere einfache Anweisungen (wie PRINT und einfache Zuweisungen). Mehrere Anweisungen, die durch : getrennt sind, sind nicht zulässig und führen zu einem Fehler. Der Befehl setzt vor der Ausführung des angeforderten Befehls einen internen Watchdog, und wenn die Steuerung nicht zum Befehl zurückkehrt, wie beispielsweise bei einer GOTO-Anweisung

	, läuft der Timer ab. In diesem Fall erhalten Sie die Meldung „Befehlszeitüberschreitung". Sie können EXECUTE nicht aus Code heraus aufrufen, der bereits mit EXECUTE ausgeführt wurde. RUN ist ein Sonderfall und bricht den Timer ab, sodass Sie den Befehl bei Bedarf zum Verketten von Programmen verwenden können.
EXIT DO EXIT FOR EXIT FUNCTION EXIT SUB	EXIT DO ermöglicht einen vorzeitigen Ausstieg aus einer DO..LOOP-Schleife EXIT FOR ermöglicht einen vorzeitigen Ausstieg aus einer FOR..NEXT-Schleife. EXIT FUNCTION ermöglicht einen vorzeitigen Ausstieg aus einer definierten Funktion. EXIT SUB ermöglicht einen vorzeitigen Ausstieg aus einer definierten Unterroutine. Der alte Standard von EXIT allein (Beenden einer do-Schleife) wird ebenfalls unterstützt.
FILES [fspec$] [,sort]	Listet Dateien in beliebigen Verzeichnissen auf dem Standard-Flash-Dateisystem oder der SD-Karte auf. „fspec$" (falls angegeben) kann einen Pfad und Suchplatzhalter im Dateiname. Fragezeichen (?) entsprechen jedem beliebigen Zeichen und ein Sternchen (*) entspricht einer beliebigen Anzahl von Zeichen. Wenn nichts angegeben ist, werden alle Dateien aufgelistet. Beispiel: * Alle Einträge suchen *.TXT Alle Einträge mit der Erweiterung TXT suchen E*.* Alle Einträge suchen, die mit E beginnen X?X.* Alle Dateinamen mit drei Buchstaben finden, die mit X beginnen und enden mydir/* Alle Einträge im Verzeichnis mydir finden Hinweis: Die Verwendung von Platzhaltern im Pfadnamen führt zu einem Fehler. „sort" gibt die Sortierreihenfolge wie folgt an: Größe in aufsteigender Reihenfolge Zeit in absteigender Reihenfolge nach Datum/Uhrzeit Name nach Dateiname (Standard, wenn nicht anders angegeben) Typ nach Dateierweiterung
FILL x, y, fillcolour [,bordercolour]	Füllt einen Bereich eines Displays mit einer Farbe. Wenn der Befehl ohne die optionale Angabe „bordercolour" verwendet wird, liest er die Farbe an der Position „x", „y" auf dem Display und füllt dann den Bereich ab diesem Punkt, an dem die aktuelle Farbe mit der neuen Farbe „fillcolour" übereinstimmt. Wenn die optionale Option „bordercolour" angegeben ist, ersetzt „fillcolour" alle bereits vorhandenen Farben, bis die angegebene „bordercolour" erreicht ist. Beachten Sie, dass dies auf TFT-Displays mit ungepufferten Treibern langsam sein kann.
FLAG(n%)=Wert	Setzt ein Bit in einem Systemflag-Register. N% kann zwischen 0 und 63 liegen (d. h. es stehen 64 Flag-Bits zur Verfügung). Der Wert kann 0 oder 1 sein. Siehe auch den Befehl FLAGS und die Funktion FLAG sowie MM.FLAGS
FLAGS=Wert	Setzt alle Bits im Systemflagregister auf den angegebenen Wert. Siehe auch den Befehl FLAG und die Funktion FLAGS sowie MM.FLAGS
FLASH FLASH LIST FLASH LIST n [,all] FLASH ERASE n	Verwaltet die Speicherung von Programmen im Flash-Speicher. Bis zu drei Programme können im Flash-Speicher gespeichert und bei Bedarf abgerufen werden. Beachten Sie, dass diese gespeicherten Programme bei einer Firmware-Aktualisierung gelöscht werden. Einer dieser Flash-Speicherplätze kann mit dem Befehl OPTION AUTORUN n automatisch geladen und ausgeführt werden, wenn die Stromversorgung eingeschaltet wird. Im Folgenden ist „n" eine Zahl zwischen 1 und 3. Zeigt eine Liste aller Flash-Speicherorte einschließlich der ersten Zeile des Programms an. Listet das auf Speicherplatz n gespeicherte Programm auf. Verwenden Sie ALL, um ohne Seitenumbrüche aufzulisten. Löschen Sie einen Flash-Programmspeicherplatz.

FLASH ERASE ALL	Löschen Sie alle Flash-Programmspeicherplätze.
FLASH SAVE n FLASH	Speichert das aktuelle Programm an der angegebenen Flash-Speicherstelle.
LOAD n FLASH RUN n	Laden Sie ein Programm aus dem angegebenen Flash-Speicherplatz in den Programmspeicher.
FLASH CHAIN n	Führt das Programm im Flash-Speicherplatz n aus, löscht alle Variablen. Ändert den Programmspeicher nicht. Führt das Programm an der Flash-Speicherstelle n aus und lässt dabei alle Variablen unverändert (ermöglicht ein Programm, das viel größer ist als der Programmspeicher). Ändert den Programmspeicher nicht. Hinweis: Wenn das verkettete Programm den Befehl READ verwendet, muss es vor dem ersten Lesevorgang den Befehl RESTORE aufrufen.
FLASH ÜBERSCHREIBEN n	Löscht einen Flash-Programmspeicherplatz und speichert dann das aktuelle Programm an dem angegebenen Flash-Speicherplatz.
FLASH DISK LOAD n, fname$ [,O[VERWRITE]]	Lädt den Inhalt der Datei fname$ als Binärbild in den Flash-Speicherplatz n. Die Datei kann mit LIBRARY DISK SAVE erstellt werden. Außerdem kann jede extern erstellte Datei mit Daten, die von einem Programm benötigt werden, mit Befehlen wie PEEK und MEMORY COPY unter Verwendung der Adresse des Flash-Speicherplatzes geladen und abgerufen werden. Wenn der optionale Parameter OVERWRITE (oder O) angegeben ist, wird der Inhalt des Flash-Steckplatzes ohne Fehlermeldung überschrieben.
FLASH LOAD IMAGE n, Dateiname$ [,O/ÜBERSCHREIBEN]	Dieser Befehl lädt eine angegebene BMP-Datei im RGB121-Format in den Flash-Speicherplatz. Der Flash-Speicherplatz sollte zuvor gelöscht worden sein, oder die Angabe des optionalen Parameters O oder OVERWRITE erzwingt eine Löschung. Beachten Sie, dass die Firmware die ersten beiden Wörter im Flash-Slot auf die Breite und Höhe des gespeicherten Bildes setzt. Die Bildgröße muss nicht mit den Abmessungen der aktuellen Anzeige übereinstimmen.
FLUSH [#]fnbr	Bewirkt, dass alle gepufferten Schreibvorgänge in eine zuvor mit der Dateinummer „#fnbr" geöffnete Datei auf die Festplatte geschrieben werden. Das # ist optional. Mit diesem Befehl wird sichergestellt, dass keine Daten verloren gehen, wenn es nach einem Schreibbefehl zu einem Stromausfall kommt.
FONT [#]font-number, scaling	Hiermit wird die Standardschriftart für die Anzeige von Text auf einem LCD-Bildschirm oder der Videoausgabe festgelegt. Schriftarten werden als Zahl angegeben. Zum Beispiel #2 (das # ist optional). Weitere Informationen zu den verfügbaren Schriftarten finden Sie im Kapitel *„Grafikbefehle und -funktionen*". „Skalierung" kann zwischen 1 und 15 liegen und multipliziert die Größe der Pixel, wodurch das angezeigte Zeichen entsprechend breiter und höher wird. Beispielsweise verdoppelt eine Skalierung von 2 die Höhe und Breite.
FOR counter = start TO finish [STEP increment]	Startet eine FOR-NEXT-Schleife, wobei der „Zähler" zunächst auf „start" gesetzt wird und in Schritten von „inkrement" (Standardwert ist 1) erhöht wird, bis der „Zähler" größer als „end" ist. Der „Inkrementwert" kann eine ganze Zahl oder eine Gleitkommazahl sein. Beachten Sie, dass die Verwendung einer Gleitkommazahl für den „Inkrementwert" zu Rundungsfehlern im „Zähler" führen kann, wodurch die Schleife vorzeitig oder verspätet beendet werden könnte. „Inkrement" kann negativ sein. In diesem Fall sollte „Ende" kleiner als „Start" sein, und die Schleife zählt rückwärts. Siehe auch den Befehl NEXT.

FRAMEBUFFER	NICHT HDMI- UND VGA-VERSIONEN Mit dem Framebuffer-Befehl können Sie einen Teil des variablen Speichers entweder einem Framebuffer, einer zweiten Anzeigeebene oder beiden zuweisen und diese dann auf interessante Weise nutzen, um Tearing-Artefakte zu vermeiden und/oder Grafikobjekte über der Hintergrundanzeige abzuspielen.
FRAMEBUFFER CREATE	Erstellt einen Framebuffer „F“ mit einem RGB121-Farbraum und einer Auflösung, die dem konfigurierten SPI-Farbdisplay entspricht.
FRAMEBUFFER-EBENE	Erstellt einen Framebuffer „L“ mit einem RGB121-Farbraum und einer Auflösung, die dem konfigurierten SPI-Farbdisplay entspricht.
FRAMEBUFFER WRITE where/where$	Gibt das Ziel für nachfolgende Grafikbefehle an. „where“ kann N, F oder L sein, wobei N die tatsächliche Anzeige ist. Es kann eine Zeichenfolgenvariable oder ein Literal verwendet werden.
FRAMEBUFFER CLOSE [which]	Schließt einen Framebuffer und gibt den Speicher frei. Der optionale Parameter „which“ kann F oder L sein. Wenn er weggelassen wird, werden beide geschlossen.
FRAMEBUFFER COPY from, to [,b]	Führt eine hochoptimierte Vollbildkopie von einem Framebuffer in einen anderen durch. „from” und „to” können N, F oder L sein, wobei N die physische Anzeige ist. Sie können nur von N auf Displays kopieren, die BLIT und transparenten Text unterstützen. Die Firmware komprimiert oder erweitert automatisch die RGB-Auflösung beim Kopieren von und zu nicht übereinstimmenden Framebuffern. Beim Kopieren von RGB565 nach RGB121 gehen natürlich Informationen verloren, aber für viele Anwendungen (z. B. Spiele) sind 16 Farbstufen mehr als ausreichend. Beim Kopieren auf ein LCD-Display kann der optionale Parameter „b” verwendet werden (FRAMEBUFFER COPY F/L, N, B). Dadurch wird die Firmware angewiesen, den Kopiervorgang mit der zweiten CPU im Raspberry Pi Pico durchzuführen, und die Steuerung kehrt sofort zum Basic-Programm zurück.
FRAMEBUFFER WAIT	Unterbricht die Verarbeitung, bis das LCD-Display in den Frame-Blanking-Modus wechselt. Implementiert für ILI9341-, ST7789_320- und ILI9488-Displays. Wird verwendet, um Artefakte beim Schreiben auf den Bildschirm zu reduzieren.
FRAMEBUFFER MERGE [Farbe] [,Modus] [,Aktualisierungsrate]	Kopiert den Inhalt des Layer-Puffers und des Framebuffers auf das LCD-Display und lässt dabei alle Pixel einer bestimmten Farbe weg. Voraussetzungen für den Befehl sind, dass sowohl FRAMEBUFFER als auch LAYERBUFFER erstellt werden FRAMEBUFFER MERGE – schreibt den Inhalt des Framebuffers auf das physische Display und überschreibt dabei alle Pixel im Framebuffer, die im Layerbuffer gesetzt sind (nicht Null). FRAMEBUFER MERGE col – schreibt den Inhalt des Framebuffers auf das physische Display und überschreibt dabei alle Pixel im Framebuffer, die sich im Layerbuffer befinden und nicht auf die transparente Farbe „col” gesetzt sind. Die Farbe wird als Zahl zwischen 0 und 15 angegeben, die Folgendes darstellt: 0:SCHWARZ,1:BLAU,2:MYRTLE,3:KOBALT,4:MITTLERES GRÜN,5:CERULEAN,6: GRÜN,7:CYAN,8:ROT,9:MAGENTA,10:ROST,11:FUCHSIA,12:BRAUN, 13:LILA,14:GELB,15:WEISS FRAMEBUFFER MERGE col,B – wie oben, außer dass die Übertragung auf das physische Display auf der zweiten CPU stattfindet und die Steuerung sofort zu Basic zurückkehrt FRAMEBUFFER MERGE col,R [,updaterate] – stellt die zweite CPU so ein, dass sie die physische Anzeige kontinuierlich mit der Zusammenführung der beiden Puffer aktualisiert. Setzt FRAMEBUFFER WRITE F automatisch, wenn nicht bereits F oder L gesetzt ist. Standardmäßig wird der Bildschirm so schnell wie möglich aktualisiert (bei 200 MHz aktualisiert ein ILI9341 im SPI-Modus etwa 13 Mal pro Sekunde, im 8-Bit-Parallelmodus erreicht der ILI9341 27 FPS).

	Wenn „updaterate" eingestellt ist, wird der Bildschirm mit der in Millisekunden angegebenen Rate aktualisiert (es sei denn, diese ist geringer als die schnellstmögliche auf dem Display). Hinweis: FRAMEBUFFER WRITE kann nicht auf N gesetzt werden, während die kontinuierliche zusammengeführte Aktualisierung aktiv ist. FRAMEBUFFER MERGE col,A – bricht die kontinuierlichen Aktualisierungen ab Darüber hinaus wird die automatische Aktualisierung auch durch Löschen des Layerbuf oder Framebuffers, durch Strg+C oder durch END abgebrochen.
FRAMEBUFFER SYNC	Wartet auf den Abschluss der letzten Hintergrundzusammenführung oder Hintergrundkopie auf der zweiten CPU, um ein Zeichnen ohne Tearing zu ermöglichen
FRAMEBUFFER	<u>NUR HDMI- UND VGA-VERSIONEN</u> Mit dem Framebuffer-Befehl können Sie einen Teil des variablen Speichers für Framebuffer, Layer-Puffer oder beides reservieren und diese dann auf interessante Weise nutzen, um Tearing-Artefakte zu vermeiden und/oder Grafikobjekte über die Hintergrundanzeige abzuspielen.
FRAMEBUFFER CREATE	Erstellt einen Framebuffer „F" mit einem Farbraum und einer Auflösung, die dem aktuellen Anzeigemodus entsprechen.
FRAMEBUFFER CREATE 2	Nur RP2350: Erstellt einen zweiten Framebuffer „2" mit einem Farbraum und einer Auflösung, die dem aktuellen Anzeigemodus entsprechen.
FRAMEBUFFER LAYER [Farbe]	Erstellt einen Ebenenpuffer „L" mit einem Farbraum und einer Auflösung, die dem aktuellen Anzeigemodus entsprechen. Der optionale Parameter „colour" wird als Zahl zwischen 0 und 15 (Modi 2 und 3), als RGB888-Farbe (Modus 4) oder als Zahl zwischen 0 und 255 (Modus 5) angegeben und legt eine Farbe fest, die ignoriert wird, wenn die Ebene auf die Anzeige angewendet wird. In Anzeigemodi, in denen die automatische Ebenenanwendung nicht unterstützt wird, fungiert ein Ebenenpuffer als weiterer Bildspeicher.
FRAMEBUFFER-EBENE TOP [Farbe]	Nur RP2350: Erstellt einen zweiten Ebenenpuffer „T" mit einem Farbraum und einer Auflösung, die dem aktuellen Anzeigemodus entsprechen. Der optionale Parameter Farbe wird als Zahl zwischen 0 und 15 (Modi 2 und 3) bzw. zwischen 0 und 255 (Modus 5) angegeben und legt eine Farbe fest, die ignoriert wird, wenn die Ebene auf die Anzeige angewendet wird. In Anzeigemodi, in denen die automatische Anwendung der[zweiten]Ebene nicht unterstützt wird, fungiert sie als weiterer Bildspeicher.
FRAMEBUFFER WRITE where/where$	Gibt das Ziel für nachfolgende Grafikbefehle an. „where" kann N, F, 2, T oder L sein, wobei N die tatsächliche Anzeige ist. Es kann eine Zeichenfolgenvariable verwendet werden.
FRAMEBUFFER CLOSE [which]	Schließt einen Framebuffer und gibt den Speicher frei. Der optionale Parameter „which" kann F, 2, T oder L sein. Wenn er weggelassen wird, werden alle geschlossen.
FRAMEBUFFER COPY from, to [,b]	Führt eine hochoptimierte Vollbildkopie von einem Framebuffer auf einen anderen durch. „from" und „to" können N, F, 2, T oder L sein, wobei N die physische Anzeige ist. Wenn der optionale Parameter „b" angegeben ist, wird die Verarbeitung angehalten, bis der Monitor in die Bildausblendung wechselt.
FRAMEBUFFER WAIT	Unterbricht die Verarbeitung, bis die nächste Bildausblendung beginnt.

FUNCTION xxx (arg1 [,arg2, …]) [AS <type>}
<Anweisungen>
<Anweisungen>
xxx = <Rückgabewert>
END FUNCTION
	Definiert eine aufrufbare Funktion. Dies entspricht dem Hinzufügen einer neuen Funktion zu MMBasic während der Ausführung Ihres Programms.
	„xxx" ist der Name der Funktion und muss den Spezifikationen für die Benennung einer Variablen entsprechen. Der Typ der Funktion kann durch Verwendung eines Typ-Suffixes (z. B. xxx$) oder durch Angabe des Typs mit AS <Typ> am Ende der Funktionsdefinition angegeben werden. Beispiel:
	`FUNCTION xxx (arg1, arg2) AS STRING`
	'arg1', 'arg2' usw. sind die Argumente oder Parameter der Funktion (die Klammern sind immer erforderlich, auch wenn keine Argumente vorhanden sind). Ein Array wird durch leere Klammern angegeben, z. B. arg3(). Der Typ des Arguments kann durch Verwendung eines Typ-Suffixes (z. B. arg1$) oder durch Angabe des Typs mit `AS <Typ> (z. B. arg1 AS STRING)` angegeben werden.
	Das Argument kann auch eine andere definierte Funktion oder dieselbe Funktion sein, wenn Rekursion verwendet werden soll (der Rekursionsstapel ist begrenzt).
	Um den Rückgabewert der Funktion festzulegen, weisen Sie den Wert dem Namen der Funktion zu. Beispiel:
	`FUNCTION SQUARE(a) SQUARE =`
	`  a * a`
	`END FUNCTION`
	Jede Definition muss eine END FUNCTION-Anweisung enthalten. Wenn diese erreicht wird, gibt die Funktion ihren Wert an den Ausdruck zurück, von dem aus sie aufgerufen wurde. Der Befehl EXIT FUNCTION kann für einen vorzeitigen Abbruch verwendet werden.
	Sie verwenden die Funktion, indem Sie ihren Namen und ihre Argumente in einem Programm genauso wie eine normale MMBasic-Funktion verwenden. Beispiel:
	`PRINT SQUARE(56.8)`
	Wenn die Funktion aufgerufen wird, wird jedes Argument im Aufrufer mit dem Argument in der Funktionsdefinition abgeglichen. Diese Argumente sind nur innerhalb der Funktion verfügbar.
	Funktionen können mit einer variablen Anzahl von Argumenten aufgerufen werden. Alle in der Funktionsliste ausgelassenen Argumente werden auf Null oder eine Null-Zeichenkette gesetzt.
	Argumente in der Liste des Aufrufers, die eine Variable sind und den richtigen Typ haben, werden per Referenz an die Funktion übergeben. Das bedeutet, dass alle Änderungen am entsprechenden Argument in der Funktion auch in die Variable des Aufrufers kopiert werden und daher nach Beendigung der Funktion darauf zugegriffen werden kann. Dem Argument kann das Präfix BYVAL vorangestellt werden, wodurch dieser Mechanismus verhindert wird und nur der Wert verwendet wird. Alternativ weist das Präfix BYREF MMBasic an, dass eine Referenz erforderlich ist, und es wird ein Fehler generiert, wenn dies nicht möglich ist.
	Arrays werden durch Angabe des Array-Namens mit leeren Klammern (z. B. arg()) übergeben, werden immer per Referenz übergeben und müssen den richtigen Typ haben.
	Sie dürfen nicht mit Befehlen wie GOTO in eine Funktion springen oder aus ihr heraus springen. Dies hat undefinierte Nebenwirkungen, darunter auch, dass Ihnen der Tag verdorben wird.

GAMEPAD-FARBE Kanal, Farbe	Ändert die Farbe des Display-Panels auf einem PS4-Controller auf dem USB-Kanal „channel". „colour" wird als Standard-RGB888-Wert festgelegt, z. B. RGB(RED).
GAMEPAD HAPTIC Kanal links, rechts	Bewirkt, dass die linken und rechten Vibrationsmotoren eines PS4-Controllers auf dem USB-Kanal „channel" betrieben werden. „left" und „right" müssen eine Zahl zwischen 0 (aus) und 255 (maximal) sein.
GAMEPAD INTERRUPT ENABLE Kanal, int [,Maske]	Aktiviert Interrupts beim Drücken der Tasten eines USB-Gamecontrollers. Der optionale Parameter „mask" definiert, welche der Tasten den Interrupt auslösen (Standard ist „alle"). „mask" ist eine Bitmap, die der Ausgabe der Funktion DEVICE(GAMEPAD channel,B) entspricht.

GAMEPAD-INTERRUPT DISABLE Kanal GAMEPAD MONITOR GAMEPAD KONFIGURIEREN vid, pid, i0, c0, i1, c1, i2, c2, i3, c3, i 4,c4,i5,c5,i6,c6,i7,c7,i8,c8,i9,c 9,i10,c10,i11,c11,i12,c12,i13,c 13,i14,c14,i15,c15	Deaktiviert Interrupts vom Gamepad auf dem angegebenen Kanal Verwenden Sie GAMEPAD MONITOR, bevor Sie ein Gamepad anschließen. Nach dem Anschließen wird bei jeder Änderung der Tasten der Bericht vor und nach der Änderung angezeigt. Verwenden Sie diese Option, um ein Gamepad zu konfigurieren, das von der Firmware nicht unterstützt wird. Führen Sie den Befehl aus, bevor Sie das Gamepad anschließen. Alle 34 Parameter sind obligatorisch. In jedem Fall definieren die i/c-Parameter den Index im Bericht und die Bitnummer an diesem Index für die Daten, die dem entsprechenden Bit entsprechen. Weitere Informationen zur Bitverwendung (0-15) finden Sie unter DEVICE(GAMEPAD n,B).
GOTO Ziel	Verzweigt die Programmausführung zum Ziel, das eine Zeilennummer oder eine Bezeichnung sein kann.
GUI BITMAP x, y, Bits [, Breite] [, Höhe] [, Skalierung] [, c] [, bc]	Zeigt die Bits in einer Bitmap auf einem VGA/HDMI-Monitor oder LCD-Bildschirm an, beginnend bei „x" und „y" auf einem angeschlossenen Gerät. „height" und „width" sind die Abmessungen der Bitmap, wie sie auf dem Gerät angezeigt werden, und standardmäßig auf 8x8 eingestellt. „scale" ist optional und standardmäßig auf den Wert gesetzt, der mit dem Befehl FONT festgelegt wurde. „c" ist die Zeichenfarbe und „bc" ist die Hintergrundfarbe. Sie sind optional und standardmäßig auf die aktuellen Vordergrund- und Hintergrundfarben eingestellt. Die Bitmap kann eine Ganzzahl- oder eine Zeichenfolgenvariable oder -konstante sein und wird unter Verwendung des ersten Bytes als erste Bits der obersten Zeile (zuerst Bit 7, dann Bit 6 usw.) gezeichnet, gefolgt vom nächsten Byte usw. Wenn die oberste Zeile gefüllt ist, beginnt die nächste Zeile der angezeigten Bitmap mit dem nächsten Bit in der Ganzzahl oder Zeichenfolge. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *„Grafikbefehle und -funktionen*".
GUI CALIBRATE oder GUI CALIBRATE a,b,c,d,d	NICHT VGA- UND HDMI-VERSIONEN Dieser Befehl dient zur Kalibrierung der Touch-Funktion eines LCD-Bildschirms. Er zeigt eine Reihe von Zielen auf dem Bildschirm an und wartet, bis jedes Ziel präzise berührt wurde. Der Befehl kann auch mit fünf Argumenten verwendet werden, die die Kalibrierungswerte angeben. In diesem Fall wird die Kalibrierung ohne Anzeige von Zielen und ohne Eingabe durch den Benutzer durchgeführt. Um die Werte zu ermitteln, verwenden Sie **nach** der normalen Kalibrierung des Displays die OPTION LIST. Beachten Sie, dass diese Werte für das jeweilige Display spezifisch sind und erheblich variieren können.
GUI-TEST LCDPANEL	Testet ein Anzeigegerät (LCD, VGA usw.). Es zeichnet kontinuierlich eine animierte Anzeige mit Farbcirkeln auf dem Display.
GUI RESET LCDPANEL	NICHT VGA- UND HDMI-VERSIONEN Initialisiert das konfigurierte LCD-Panel neu. Die Initialisierung erfolgt automatisch beim Start der PicoMite-Firmware, aber unter bestimmten Umständen kann es erforderlich sein, die Stromversorgung des LCD-Panels zu unterbrechen (z. B. um Batteriestrom zu sparen). In diesem Fall kann dieser Befehl verwendet werden, um das Display neu zu initialisieren.
GUI TEST TOUCH	NICHT VGA- UND HDMI-VERSIONEN Testet die Touch-Funktion des LCD-Panels. Der Bildschirm wird gelöscht und MMBasic wartet auf eine Berührung, wodurch ein weißer Punkt auf dem Display platziert wird, der die genaue Berührungsposition auf dem Bildschirm markiert. Jedes in die Konsole eingegebene Zeichen beendet den Test.

HELP Suchtext	Der Befehl „help“ sucht nach einer Datei namens „help.txt“ auf Laufwerk A:. Diese Datei kann vom Benutzer oder von der Community erstellt worden sein und muss ein bestimmtes Format aufweisen. Für jeden Hilfseintrag muss die erste Zeile eine Suchzeichenfolge sein, der ein ~-Zeichen vorangestellt ist. Diese wird von der Hilfefunktion zum Auffinden eines Eintrags verwendet und nicht angezeigt. Der „Suchtext“ kann ? für die Ersetzung eines einzelnen Zeichens oder * für die Ersetzung mehrerer Zeichen (oder keiner) enthalten. Nach dem Suchstring gibt die nächste Zeile in der Regel die Syntax eines bestimmten Befehls oder einer bestimmten Funktion an. Alle folgenden Zeilen enthalten weitere Erläuterungen. Beispiel *~COLOR* *COLOR fore [, back]* *Legt die Standardfarbe für Befehle (PRINT usw.) fest, die auf dem angeschlossenen LCD-Bildschirm angezeigt werden.* *„fore” ist die Vordergrundfarbe, „back” ist die Hintergrundfarbe.* *Der Hintergrund ist optional und wird standardmäßig schwarz angezeigt, wenn keine Angabe erfolgt.* Der Befehl gibt alle Einträge zurück, die mit dem „Suchtext“ übereinstimmen, und diese werden entsprechend dem Konsolengerät paginiert. Verschiedene Versionen von help.txt sind verfügbar unter https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17865
HUMID pin, tvar, hvar [,DHT11]	Gibt die Temperatur und Luftfeuchtigkeit unter Verwendung des DHT22-Sensors zurück. Alternative Versionen des DHT22 sind der AM2303 oder der RHT03 (alle sind kompatibel). „pin” ist der mit dem Sensor verbundene I/O-Pin. Es kann jeder beliebige I/O-Pin verwendet werden. „tvar” ist die Variable, die die gemessene Temperatur enthält, und „hvar” ist das Gleiche für die Luftfeuchtigkeit. Beide müssen vorhanden sein und beide müssen Fließkommavariablen sein. Beispiel: `HUMID 3, TEMP!, HUMIDITY!` Die Temperatur wird in °C gemessen, die Luftfeuchtigkeit in Prozent relativer Luftfeuchtigkeit. Beide Werte werden mit einer Auflösung von 0,1 gemessen. Bei Auftreten eines Fehlers (Sensor nicht angeschlossen oder fehlerhaftes Signal) werden beide Werte mit 1000,0 angezeigt. Normalerweise sollte der DHT22 mit 3,3 V betrieben werden, um seinen Ausgang für den Raspberry Pi Pico unter 3,6 V zu halten (der Pico 2 hat dieses Problem nicht), und der Signalpin sollte durch einen 1K- bis 10K-Widerstand (4,7K empfohlen) auf 3,3 V hochgezogen werden. Der optionale Parameter DHT11 ändert die Zeitsteuerung für die Verwendung mit dem DHT11. Setzen Sie ihn für DHT11 auf 1 und für DHT22 auf 0 oder lassen Sie ihn weg.
I2C	Weitere Details finden Sie in Anhang B.
I2C OPEN Geschwindigkeit, Zeitlimit	Aktiviert das erste I^2C-Modul im Master-Modus. „speed“ ist die zu verwendende Taktrate (in kHz) und muss entweder 100, 400 oder 1000 sein. „timeout“ ist ein Wert in Millisekunden, nach dessen Ablauf die Sende- und Empfangskommandos des Masters unterbrochen werden, wenn sie nicht abgeschlossen sind. Der Mindestwert beträgt 100. Der Wert Null deaktiviert das Timeout (dies wird jedoch nicht empfohlen).
I2C WRITE addr, option, sendlen, senddata [,sendata ..]	Senden Sie Daten an das I^2C-Slave-Gerät. „addr“ ist die I^2C-Adresse des Slaves. „option“ kann 0 für den normalen Betrieb oder 1 sein, um die Kontrolle über den Bus nach dem Befehl zu behalten (eine Stoppbedingung wird nach Abschluss des Befehls nicht gesendet). „sendlen“ ist die Anzahl der zu sendenden Bytes. „senddata“ sind die zu sendenden Daten – diese können auf verschiedene Weise angegeben werden (alle gesendeten Werte liegen zwischen 0 und 255). Hinweise: • Die Daten können als einzelne Bytes in der Befehlszeile angegeben werden. Beispiel: `I2C WRITE &H6F, 0, 3, &H23, &H43, &H25`

	• Die Daten können in einem eindimensionalen Array mit leeren Klammern (d. h. ohne Dimensionen) angegeben werden. Die „sendlen"-Bytes des Arrays werden beginnend mit dem ersten Element gesendet. Beispiel: `I2C WRITE &H6F, 0, 3, ARRAY()` Die Daten können eine String-Variable (keine Konstante) sein. Beispiel: `I2C WRITE &H6F, 0, 3, STRING$`
I2C READ addr, option, rcvlen, rcvbuf	Daten vom I2C-Slave-Gerät abrufen. „addr" ist die I²C-Adresse des Slaves. „option" kann 0 für den normalen Betrieb oder 1 sein, um die Kontrolle über den Bus nach dem Befehl zu behalten (eine Stoppbedingung wird nach Abschluss des Befehls nicht gesendet). „rcvlen" ist die Anzahl der zu empfangenden Bytes. „rcvbuf" ist die Variable oder das Array, das zum Speichern der empfangenen Daten verwendet wird – dies kann sein: • Eine Zeichenfolgenvariable. Die Bytes werden als aufeinanderfolgende Zeichen gespeichert. • Ein eindimensionales Array von Zahlen, das mit leeren Klammern angegeben wird. Die empfangenen Bytes werden in aufeinanderfolgenden Elementen des Arrays gespeichert, beginnend mit dem ersten. Beispiel: `I2C READ &H6F, 0, 3, ARRAY()` Eine normale numerische Variable (in diesem Fall muss rcvlen 1 sein).
I2C CHECK addr I2C	Setzt die schreibgeschützte Variable MM.I2C auf 0, wenn ein Gerät auf die Adresse „addr" antwortet. MM.I2C wird auf 1 gesetzt, wenn keine Antwort erfolgt.
CLOSE	Deaktiviert das Master-I²C-Modul. Dieser Befehl sendet auch einen Stopp, wenn der Bus noch gehalten wird.
I2C SLAVE	Siehe Anhang B
I2C2	Dieselbe Reihe von Befehlen wie für I2C (oben), jedoch für den zweiten I²C-Kanal.
IF expr THEN stmt [: stmt] oder IF expr THEN stmt ELSE stmt	Wertet den Ausdruck „expr" aus und führt die Anweisung nach dem Schlüsselwort THEN aus, wenn er wahr ist, oder springt zur nächsten Zeile, wenn er falsch ist. Wenn weitere Anweisungen in der Zeile vorhanden sind (getrennt durch Doppelpunkte (:)), werden diese ebenfalls ausgeführt, wenn sie wahr sind, oder übersprungen, wenn sie falsch sind. Das Schlüsselwort ELSE ist optional. Wenn es vorhanden ist, werden die darauf folgenden Anweisungen ausgeführt, wenn „expr" als falsch ausgewertet wird. Die Konstruktion „THEN-Anweisung" kann auch durch „GOTO Zeilennummer | Label" ersetzt werden. Diese Art von IF-Anweisung steht vollständig in einer Zeile.

IF Ausdruck THEN
<Anweisungen>
[ELSEIF Ausdruck THEN
<Anweisungen>]
[ELSE
<Anweisungen>]
ENDIF
	Mehrzeilige IF-Anweisung mit optionalen ELSE- und ELSEIF-Fällen, die mit ENDIF endet. Jede Komponente steht in einer separaten Zeile.
	Wertet „Ausdruck" aus und führt die Anweisung(en) nach THEN aus, wenn der Ausdruck wahr ist, oder optional die Anweisung(en) nach der ELSE-Anweisung, wenn er falsch ist. Die ELSEIF-Anweisung (falls vorhanden) wird ausgeführt, wenn die vorherige Bedingung falsch ist, und sie startet eine neue IF-Kette mit weiteren ELSE- und/oder ELSEIF-Anweisungen, je nach Bedarf. Ein ENDIF wird verwendet, um das mehrzeilige IF zu beenden.

INC var [,Inkrement]	Inkrementiert die Variable „var" entweder um 1 oder, falls angegeben, um den Wert in increment. „increment" kann negativ sein, was zu einer Dekrementierung führt. Dies entspricht funktional `var = var + increment,` wird jedoch viel schneller verarbeitet.

INPUT ["prompt$";] var1 [,var2 [, var3 [, etc]]]	Nimmt eine Liste von Werten, die durch Kommas (,) getrennt sind und an der Konsole eingegeben werden, und weist sie einer sequenziellen Liste von Variablen zu. Beispiel: Wenn der Befehl lautet: INPUT a, b, c Und Folgendes wird auf der Tastatur eingegeben: 23, 87, 66 Dann ist a = 23 und b = 87 und c = 66 Die Liste der Variablen kann eine Mischung aus Float-, Integer- oder String-Variablen sein. Die in der Konsole eingegebenen Werte müssen dem Typ der Variablen entsprechen. Wenn ein einzelner Wert eingegeben wird, ist kein Komma erforderlich (dieser Wert darf jedoch kein Komma enthalten). „prompt$“ ist eine Zeichenfolgenkonstante (keine Variable oder kein Ausdruck) und wird, wenn angegeben, zuerst ausgegeben. Normalerweise wird die Eingabeaufforderung mit einem Semikolon (;) beendet, und in diesem Fall wird nach der Eingabeaufforderung ein Fragezeichen ausgegeben. Wenn die Eingabeaufforderung mit einem Komma (,) statt mit einem Semikolon (;) beendet wird, wird das Fragezeichen unterdrückt.
INPUT #nbr, Liste von Variablen	Wie oben, außer dass die Eingabe von einem seriellen Port oder einer zuvor für INPUT als „nbr” geöffneten Datei gelesen wird. Siehe Befehl OPEN.
INTERRUPT [myint]	Dieser Befehl löst einen Software-Interrupt aus. Der Interrupt wird mit INTERRUPT „myint“ eingerichtet, wobei „myint“ der Name einer Subroutine ist, die ausgeführt wird, wenn der Interrupt ausgelöst wird. Verwenden Sie INTERRUPT 0, um den Interrupt zu deaktivieren. Verwenden Sie INTERRUPT ohne Parameter, um den Interrupt auszulösen. Hinweis: Der Interrupt kann auch aus einem CSUB
IR dev, key , int oder IR CLOSE	Decodiert Infrarot-Fernbedienungssignale von NEC oder Sony. Ein IR-Empfängermodul ist erforderlich, um das IR-Licht zu erfassen und das Signal zu demodulieren. Es kann an jeden Pin angeschlossen werden, dieser Pin muss jedoch zuvor mit dem Befehl `SETPIN n, IR` konfiguriert werden. Die Dekodierung des IR-Signals erfolgt im Hintergrund, und das Programm wird nach diesem Befehl ohne Unterbrechung fortgesetzt. „dev” und „key” sollten numerische Variablen sein, deren Werte bei jedem Empfang eines neuen Signals aktualisiert werden („dev” ist der von der Fernbedienung gesendete Gerätecode und „key” ist die gedrückte Taste). „int” ist eine benutzerdefinierte Subroutine, die aufgerufen wird, wenn eine neue Taste gedrückt wird oder wenn die vorhandene Taste für die automatische Wiederholung gedrückt gehalten wird. In der Interrupt-Subroutine kann das Programm die Variablen „dev” und „key” überprüfen und entsprechende Maßnahmen ergreifen. Der Befehl „IR CLOSE” beendet den IR-Decoder. Beachten Sie, dass beim NEC-Protokoll die Bits in „dev” und „key” vertauscht sind. Beispielsweise sollte in „key” Bit 0 Bit 7 sein, Bit 1 sollte Bit 6 sein usw. Dies hat keinen Einfluss auf die normale Verwendung, aber wenn Sie nach einem bestimmten numerischen Code eines Herstellers suchen, sollten Sie die Bits umkehren. Hier finden Sie eine Beschreibung, wie Sie dies tun können: http://www.thebackshed.com/forum/forum_posts.asp?TID=8367 Weitere Informationen finden Sie im Kapitel *„Spezielle Hardwaregeräte*”.
IR-SEND-Pin, dev, key	Erzeugt ein 12-Bit-Infrarotsignal nach dem Sony-Fernbedienungsprotokoll. „Pin” ist der zu verwendende I/O-Pin. Dies kann ein beliebiger I/O-Pin sein, der automatisch als Ausgang konfiguriert wird und an eine Infrarot-LED angeschlossen werden sollte. Im Ruhezustand ist der Pegel niedrig, hohe Pegel zeigen an, wann die LED eingeschaltet werden soll. „dev“ ist das zu steuernde Gerät und eine Zahl zwischen 0 und 31, „key“ ist der simulierte Tastendruck und eine Zahl zwischen 0 und 127. Das IR-Signal wird mit etwa 38 kHz moduliert, und das Senden des Signals dauert etwa 25 ms, währenddessen die Programmausführung angehalten wird.

KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3 [, c4] oder KEYPAD CLOSE	Überwacht und decodiert Tastendrücke auf einer 4x3- oder 4x4-Tastatur. Die Überwachung der Tastatur erfolgt im Hintergrund, und das Programm wird nach diesem Befehl ohne Unterbrechung fortgesetzt. „var" sollte eine numerische Variable sein, deren Wert bei jeder erkannten Tastenbetätigung aktualisiert wird. „int" ist eine benutzerdefinierte Subroutine, die aufgerufen wird, wenn ein neuer Tastendruck empfangen wird. In der Interrupt-Subroutine kann das Programm die Variable „var" überprüfen und entsprechende Maßnahmen ergreifen. r1, r2, r3 und r4 sind Pin-Nummern, die für die vier Reihenverbindungen zum Tastenfeld verwendet werden, und c1, c2, c3 und c4 sind die Spaltenverbindungen. c4 ist optional und wird nur bei 4x4-Tastenfeldern verwendet. Dieser Befehl konfiguriert diese Pins automatisch nach Bedarf. Bei einem Tastendruck ist der Wert, der „var" zugewiesen ist, die Nummer einer Zifferntaste (z. B. gibt „6" den Wert 6 zurück) oder 10 für die Taste * und 11 für die Taste #. Bei 4x4-Tastaturen wird für A der Wert 20, für B der Wert 21, für C der Wert 22 und für D der Wert 23 zurückgegeben. Der Befehl KEYPAD CLOSE beendet die Tastaturfunktion und versetzt den E/A-Pin wieder in einen nicht konfigurierten Zustand. Weitere Informationen finden Sie im Abschnitt *„Spezielle Hardwaregeräte*".

KEYPAD keymapmap!(), var!,int, startcolpin, nocols, startrowpin, norows	NUR RP2350-VERSIONEN
	Konfigurieren und verwenden Sie eine Tastatur mit einer beliebigen Anzahl von Zeilen und Spalten und weisen Sie jeder Taste benutzerdefinierte Rückgabecodes zu.
	„keymapmap!()" ist ein zweidimensionales Array aus Floats, das die vom Benutzer angegebene Zuordnung für jede Taste enthält. Die Array-Dimensionen müssen mit der angegebenen Anzahl von Spalten und Zeilen übereinstimmen.
	„var!" ist die Float-Variable, die den zurückgegebenen Wert enthält. „int" ist die Interrupt-Routine, die beim Drücken einer Taste aufgerufen wird.
	„startcolpin" ist der Pin, der den Bereich der zusammenhängenden GP-Pins beginnt, die für den Spaltenscan verwendet werden, während „nocols" die Anzahl der Pins in diesem Bereich ist. „startrowpin" ist der Pin, der „norows" zusammenhängender GP-Pins beginnt, die für den Zeilenscan verwendet werden.
	Beispiel:
	Mit einer Tastatur, die aus 3 Spalten und 4 Zeilen besteht (d. h. 123 / 456 / 789 / *0#). Verbinden Sie GP1 mit der linken Spalte, GP2 mit der mittleren und GP3 mit der rechten, dann die 4 Zeilen von oben nach unten mit GP4 bis GP7.
	Das Programm könnte wie folgt aussehen:
	`OPTION BASE 1`
	`DIM keymap(3,4)=(1,4,7,10, 4,5,6,0, 3,6,9,11)`
	`KEYPAD keymap(), keyret, myint, GP1, 3, GP4, 4 DO`
	`  ' Hauptprogramm-Verarbeitungsschleife`
	`LOOP`
	`SUB myint PRINT`
	`  keyret`
	`END SUB`

KILL file$ [,all]	Löscht die durch „file$" angegebene Datei. Die Dateiendung muss angegeben werden. Die Massenlöschung wird ausgelöst, wenn fname$ ein '*'- oder ein '?'-Zeichen enthält. Wenn der optionale Parameter 'all' verwendet wird, werden Sie zu einer einzigen Bestätigung aufgefordert. Wenn 'all' nicht angegeben ist, werden Sie für jede Datei einzeln aufgefordert.
LCD INIT d4, d5, d6, d7, rs, en od er LCD line, pos, text$ oder LCD CLEAR oder	Zeigt Text auf einem LCD-Zeichenanzeigemodul an. Dieser Befehl funktioniert mit den meisten 1-zeiligen, 2-zeiligen oder 4-zeiligen LCD-Modulen, die den Controller KS0066, HD44780 oder SPLC780 verwenden (dies kann jedoch nicht garantiert werden). Der Befehl LCD INIT wird verwendet, um das LCD-Modul für die Verwendung zu initialisieren. „d4" bis „d7" sind die I/O-Pins, die mit den Eingängen D4 bis D7 des LCD-Moduls verbunden sind (die Eingänge D0 bis D3 sollten mit Masse verbunden sein). „rs" ist der Pin, der mit dem

LCD CLOSE	Registerauswahleingang am Modul (manchmal auch als CMD bezeichnet). „en" ist der Pin, der mit dem Freigabe- oder Chipauswahleingang am Modul verbunden ist. Der R/W-Eingang am Modul sollte immer geerdet sein. Die oben genannten E/A-Pins werden durch diesen Befehl automatisch als Ausgänge festgelegt. Nachdem das Modul initialisiert wurde, können mit dem LCD-Befehl Daten darauf geschrieben werden. „line" ist die Zeile auf dem Display (1 bis 4) und „pos" ist die Zeichenposition in der Zeile (die erste Position ist 1). „text$" ist eine Zeichenfolge, die den Text enthält, der auf das LCD-Display geschrieben werden soll. „pos" kann auch C8, C16, C20 oder C40 sein. In diesem Fall wird die Zeile gelöscht und der Text auf einem Display mit 8, 16, 20 oder 40 Zeilen zentriert. Beispiel: `LCD 1, C16, „Hallo"` LCD CLEAR löscht alle auf dem LCD angezeigten Daten, und LCD CLOSE beendet die LCD-Funktion und versetzt alle E/A-Pins wieder in den nicht konfigurierten Zustand. Weitere Informationen finden Sie im Kapitel *„Spezielle Hardwaregeräte*".
LCD CMD d1 [, d2 [, etc]] oder LCD DATA d1 [, d2 [, etc]]	Diese Befehle senden ein oder mehrere Bytes als Befehl (LCD CMD) oder als Daten (LCD DATA) an ein LCD-Display. Jedes Byte ist eine Zahl zwischen 0 und 255 und muss durch Kommas getrennt werden. Das LCD muss zuvor mit dem Befehl LCD INIT initialisiert worden sein (siehe oben). Diese Befehle können verwendet werden, um ein nicht standardmäßiges LCD im „Rohmodus" anzusteuern, oder um spezielle Funktionen wie Scrollen, Cursor und benutzerdefinierte Zeichensätze zu aktivieren. Die erforderlichen Befehle und Datenwerte finden Sie im Datenblatt Ihres LCD.
LET Variable = Ausdruck	Weist der Variablen den Wert von „Ausdruck" zu. LET wird automatisch angenommen, wenn eine Anweisung nicht mit einem Befehl beginnt. Beispiel: `Var = 56`
LIBRARY SAVE oder LIBRARY DELETE oder BIBLIOTHEK AUFZEICHNEN oder LIBRARY LIST ALL oder BIBLIOTHEKS-DISK SPEICHERN fname$ oder LIBRARY DISK LOAD fname$	Die Bibliothek ist ein spezieller Bereich des Programmspeichers, der Programmcode wie Unterprogramme, Funktionen und CFunktionen enthalten kann. Diese Routinen sind für den Programmierer nicht sichtbar, stehen jedoch dem laufenden Programm zur Verfügung und verhalten sich genauso wie die in MMBasic integrierten Befehle und Funktionen. Jeder Code in der Bibliothek, der nicht in einer Unterroutine oder Funktion enthalten ist, wird unmittelbar vor der Ausführung eines Programms ausgeführt. Dies kann zum Initialisieren von Konstanten, Festlegen von Optionen usw. verwendet werden. Eine ausführliche Erläuterung finden Sie unter der Überschrift *„Die Bibliothek"* in diesem Handbuch. Die Bibliothek wird im Programmspeicher Flash Slot 3 gespeichert, der dann nicht mehr zum Speichern eines Programms zur Verfügung steht (die Slots 1 bis 2 sind weiterhin verfügbar). LIBRARY SAVE komprimiert den gesamten Inhalt des normalen Programmspeichers (redundante Daten wie Kommentare werden entfernt) und hängt ihn an den Bibliotheksbereich an (der Hauptprogrammspeicher ist dann leer). Der Code in der Bibliothek wird nicht in LIST oder EDIT angezeigt und wird nicht gelöscht, wenn ein neues Programm geladen oder NEW verwendet wird. LIBRARY DELETE entfernt die Bibliothek und gibt Flash-Steckplatz 3 für die normale Verwendung zurück (OPTION RESET hat die gleiche Wirkung). LIBRARY LIST listet den Inhalt der Bibliothek auf. Verwenden Sie ALL, um ohne Seitenbestätigungen aufzulisten. LIBRARY DISK SAVE fname$ speichert die aktuelle Bibliothek als Binärdatei, sodass sie später mit LIBRARY DISK LOAD fname$ wiederhergestellt werden kann. Auf diese Weise lassen sich programmspezifische Bibliotheken einfach speichern, wiederherstellen und verteilen. Abgesehen von versionsspezifischen Funktionen in der Bibliothek (WEB, VGA, GUI) können Bibliotheken versionsübergreifend gemeinsam genutzt werden.

LINE x1, y1, x2, y2 [,[-] LW [, C]]	Zeichnet eine Linie auf dem Display, die bei den Koordinaten „x1" und „y1" beginnt und bei „x2" und „y2" endet. „LW" ist die Breite der Linie. Wenn nicht angegeben, ist der Standardwert 1. Bei Linien mit einer definierten Breite definieren die Koordinaten x1 und y1 den Pixel oben links der dicken Linie. Das heißt, die Linie befindet sich rechts von der angegebenen Position oder darunter auf dem Bildschirm. Wenn die Breite als negativer Wert angegeben wird, gilt die Breite für Linien in alle Richtungen, und diese werden auf die angegebenen Start- und Zielkoordinaten zentriert. „C" ist eine ganze Zahl, die die Farbe angibt und standardmäßig auf die aktuelle Vordergrundfarbe eingestellt ist. Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Linien, die durch die Abmessungen des kleinsten Arrays bestimmt wird. „x1", „y1", „x2" und „y2" müssen alle Arrays oder alle einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler generiert. „lw" und „c" können entweder Arrays oder einzelne Variablen/Konstanten sein.
LINE AA x1, y1, x2, y2 [, LW [, C]]	Zeichnet eine Linie mit Anti-Aliasing. Die Parameter entsprechen denen des Befehls LINE oben. Diese Version verwendet jedoch variable Intensitätswerte der angegebenen Farbe, um die „versetzte" Qualität diagonaler Linien zu reduzieren. Darüber hinaus kann diese Version diagonale Linien beliebiger Breite zeichnen. Beachten Sie, dass sie keine Arrays als Parameter akzeptiert.
LINE GRAPH x(), y(), Farbe	Dieser Befehl erzeugt ein Liniendiagramm der in „x()" und „y()" angegebenen Koordinatenpaare. Das Diagramm hat n-1 Segmente, wobei die Arrays x und y jeweils n Elemente enthalten.
LINE INPUT [prompt$,] string-variable$	Liest eine gesamte Zeile aus der Konsoleneingabe in „string-variable$". „prompt$" ist eine Zeichenfolgenkonstante (keine Variable oder kein Ausdruck) und wird, sofern angegeben, zuerst ausgegeben. Ein Fragezeichen wird nur ausgegeben, wenn es Teil von „prompt$" ist. Im Gegensatz zu INPUT liest dieser Befehl eine ganze Zeile und hält nicht bei durch Kommas getrennten Datenelementen an.
LINE INPUT #nbr, Zeichenfolgenvariable$	Wie oben, außer dass die Eingabe von einem seriellen Kommunikationsport oder einer zuvor für INPUT als „nbr" geöffneten Datei gelesen wird. Siehe Befehl OPEN.
LINE PLOT ydata() [,nbr] [,xstart] [,xinc] [,ystart] [,yinc] [,colour]	Zeichnet ein Liniendiagramm aus einem Array von Datenpunkten der y-Achse. „ydata()" ist ein Array aus Floats oder Integers, die gezeichnet werden sollen. „nbr" ist die Anzahl der zu zeichnenden Liniensegmente – standardmäßig ist dies der kleinere Wert aus der Array-Größe und MM.HRES-2, wenn nichts angegeben ist. „xstart" ist die x-Koordinate, bei der mit dem Zeichnen begonnen werden soll – Standardwert ist 0. „xinc" ist die Schrittweite entlang der x-Achse, um jede Koordinate zu zeichnen – Standardwert ist 1 „ystart" ist die Position in ydata, an der die Darstellung beginnen soll – standardmäßig ist dies der Anfang des Arrays. Hinweis: berücksichtigt OPTION BASE „yinc" ist die Inkrementierung des Index in ydata, die für jeden zu zeichnenden Punkt hinzugefügt wird. „colour" ist die Farbe, in der die Linie gezeichnet wird.
LIST [fname$] oder LIST ALL [fname$]	Listet ein Programm auf der Konsole auf. LIST allein listet das Programm mit einer Pause nach jedem Bildschirm voll auf. LIST ALL listet das Programm ohne Pausen auf. Dies ist nützlich, wenn Sie das Programm an einen Terminalemulator auf einem PC übertragen möchten, der seinen Eingabestrom in einer Datei erfassen kann. Wenn die Option „fname$" angegeben ist, wird diese Datei im Flash-Dateisystem oder auf der SD-Karte aufgelistet.
LIST PINS	Listet den aktuellen Status aller Pins auf dem Prozessor auf

LIST SYSTEM I2C	Listet eine Übersicht aller I2C-Geräte auf, die mit dem System-I2C-Bus verbunden sind.
LIST COMMANDS oder LIST FUNCTIONS	Listet alle gültigen Befehle oder Funktionen auf
LIST VARIABLES [s%()]	Listet alle globalen Variablen und Konstanten auf und, wenn in einer Subroutine aufgerufen, die von dieser Subroutine und allen sie aufrufenden Subroutinen verwendeten Variablen. Wenn der optionale Parameter s$() verwendet wird, werden die Variablen in s%() als Longstring aufgelistet (siehe Befehl LONGSTRING).
LMID(array%(),start [,num])=string$	fügt „string$" an der Position „start" in den Langstring „array%()" ein und ersetzt dabei „num" vorhandene Zeichen. Wenn „num" nicht angegeben ist, wird es aus der Länge von „string$" berechnet. „num" kann 0 sein. In diesem Fall wird „string$" an der angegebenen Position eingefügt.
LOAD file$ [,R]	Lädt ein Programm namens „file$" aus dem Flash-Dateisystem oder von der SD-Karte in den Programmspeicher. Wenn das optionale Suffix ‚R hinzugefügt wird, wird das Programm sofort ohne Aufforderung ausgeführt (in diesem Fall muss „file$" eine String-Konstante sein). Der Befehl RUN hat dieselbe Funktion und ermöglicht die Verwendung einer String-Variablen. Wenn keine Erweiterung angegeben ist, wird „.BAS" an den Dateinamen angehängt.
LOAD CONTEXT [KEEP]	Stellt den Variablenbereich auf den zuvor gespeicherten Zustand zurück und behält optional die gespeicherten Variablen bei, um bei Bedarf ein zweites LOAD zu ermöglichen. Siehe auch SAVE CONTEXT
LOAD IMAGE/BMP fname$ [,x] [,y] [,mode] [,ximage] [,yimage]	Lädt eine BMP-Datei aus „fname$" und schreibt sie auf das aktuelle Ausgabegerät (Anzeige oder Framebuffer), beginnend bei „x", „y" (Standardwert ist 0,0). Der optionale Parameter „mode" gibt an, ob die Ausgabe gedithert werden soll. Die Bits 0 und 1 legen das Ausgabeformat fest, Bit 2 legt die Art des zu verwendenden Ditherings fest. Standardmäßig ist der Modus auf -1 eingestellt, was bedeutet, dass kein Dithering angewendet werden soll. DITHER_FLOYD_STEINBERG_RGB121 0 DITHER_FLOYD_STEINBERG_RGB222 1 DITHER_FLOYD_STEINBERG_RGB332 2 DITHER_ATKINSON_RGB121 4 DITHER_ATKINSON_RGB222 5 DITHER_ATKINSON_RGB332 6 „ximage" und „yimage" geben an, an welcher Stelle in der Bilddatei mit dem Schreiben auf das Display begonnen werden soll (Standardwert ist 0,0). Wenn keine Erweiterung angegeben ist, wird „.BMP" an den Dateinamen angehängt. Alle Arten des BMP-Formats werden unterstützt, einschließlich Schwarzweiß- und Echtfarben-24-Bit-Bildern.
LOAD JPG file$ [, x] [, y] [,mode] [,ximage] [,yimage]	Lädt eine JPG-Datei aus „fname$" und schreibt sie auf das aktuelle Ausgabegerät (Anzeige oder Framebuffer), beginnend bei „x", „y" (Standardwert ist 0,0). Der optionale Parameter „mode" gibt an, ob die Ausgabe gedithert werden soll. Die Bits 0 und 1 legen das Ausgabeformat fest, Bit 2 legt die Art des zu verwendenden Ditherings fest. Standardmäßig ist der Modus auf -1 gesetzt, was bedeutet, dass kein Dithering angewendet werden soll. DITHER_FLOYD_STEINBERG_RGB121 0 DITHER_FLOYD_STEINBERG_RGB222 1 DITHER_FLOYD_STEINBERG_RGB332 2 DITHER_ATKINSON_RGB121 4 DITHER_ATKINSON_RGB222 5 DITHER_ATKINSON_RGB332 6

	„ximag" und „yimage" geben an, an welcher Stelle in der Bilddatei mit dem Schreiben auf das Display begonnen werden soll (Standardwert ist 0,0). Wenn keine Erweiterung angegeben ist, wird „.JPG" an den Dateinamen angehängt. Progressive JPG-Bilder werden nicht unterstützt.
LOAD PNG fname$ [, x] [, y] [,transparent] [,alphacut]	NUR RP2350-VERSIONEN Lädt und zeigt eine PNG-Datei „fname" an Wenn keine Erweiterung angegeben ist, wird automatisch die Erweiterung .png an den Dateinamen angehängt. Die Datei muss im RGBA8888-Format vorliegen, was die normale Standardeinstellung ist. Wenn angegeben, geben „x" und „y" an, wo auf dem Display oder im Framebuffer das Bild angezeigt wird. Der optionale Parameter „transparent" (Standardwert 0) gibt einen der Farbcodes (0–15) an, die den Pixeln in der PNG-Datei mit einem Alpha-Wert unter „alphacut" (Standardwert 20) zugewiesen werden. Wenn „transparent" auf -1 gesetzt ist, wird das PNG-Bild so geschrieben, dass Pixel mit einem Alpha-Wert unter „alphacut" vollständig weggelassen werden.
LOCAL variable [, variables] Die vollständige Syntax finden Sie unter DIM.	Definiert eine Liste von Variablennamen als lokal für die Subroutine oder Funktion. Dieser Befehl verwendet genau die gleiche Syntax wie DIM und erstellt Variablen, die nur innerhalb der Subroutine oder Funktion sichtbar sind. Sie werden automatisch verworfen, wenn die Subroutine oder Funktion beendet wird.
LONGSTRING	Die LONGSTRING-Befehle ermöglichen die Bearbeitung von Zeichenfolgen, die länger sind als die normale MMBasic-Begrenzung von 255 Zeichen. Variablen zum Speichern langer Zeichenfolgen müssen als eindimensionale Integer-Arrays definiert werden, wobei die Anzahl der Elemente auf die Anzahl der Zeichen festgelegt wird, die für die maximale Zeichenfolgenlänge erforderlich sind, geteilt durch acht. Der Grund für die Division durch acht ist, dass jede Ganzzahl in einem MMBasic-Array acht Bytes belegt. Beachten Sie, dass die Routinen für lange Zeichenfolgen nicht auf Überlauf in der Länge der Zeichenfolgen prüfen. Wenn versucht wird, eine Zeichenfolge zu erstellen, die länger als die Größe einer Variablen für lange Zeichenfolgen ist, ist das Ergebnis undefiniert.
LONGSTRING AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key[!/%](), in%(), out%() [,iv$/iv[!/%]()]	Verschlüsselt oder entschlüsselt die lange Zeichenkette in in%() und speichert das Ergebnis in out%(). Für die Modi CBC und CTR generiert die Verschlüsselung einen zufälligen Initialisierungsvektor und hängt out%() mit dem IV vor. Wenn ein expliziter IV angegeben ist, wird dieser anstelle des Zufallsvektors verwendet und an out%() angehängt. Bei der CBC- und CTR-Entschlüsselung geht die Firmware davon aus, dass die ersten 16 Bytes von in%() der Initialisierungsvektor sind. Wenn Sie eine Nachricht ohne IV übertragen möchten, können Sie LONGSTRING RIGHT verwenden, um die IV vor dem Senden der Nachricht zu entfernen. In diesem Fall muss der Empfänger sowohl die IV als auch den Schlüssel kennen und einen vollständigen Longstring erstellen, bevor er DECRYPT verwendet. Dies kann mit LONGSTRING CONCAT erfolgen, um die eingehende Nachricht zu einem Longstring hinzuzufügen, der die IV enthält.
LONGSTRING APPEND array%(), string$	Fügen Sie eine normale MMBasic-Zeichenkette an eine Longstring-Variable an. „array%()" ist eine Longstring-Variable, während „string$" ein normaler MMBasic-Zeichenkettenausdruck ist.
LONGSTRING BASE64 ENCODE/DECODE in%(), out%()	Diese BASE64-Funktion codiert oder decodiert die Longstring-Variable in in%() und speichert das Ergebnis in out%(). Das als Ausgabe verwendete Array muss im Verhältnis zur Eingabe und zur Richtung groß genug sein. Durch die Codierung erhöht sich die Länge um 4/3, durch die Decodierung verringert sie sich um 3/4.
LONGSTRING CLEAR array%()	Löscht die lange Zeichenfolgenvariable „array%()". Das heißt, sie wird auf eine leere Zeichenfolge gesetzt.
LONGSTRING COPY dest%(), src%()	Kopieren Sie eine lange Zeichenfolge in eine andere. „dest%()" ist die Zielvariable und „src%()" ist die Quellvariable. Der Inhalt von „dest%()" wird überschrieben.

LONGSTRING CONCAT dest%(), src%()	Verknüpfen Sie eine lange Zeichenfolge mit einer anderen. „dest%()“ ist die Zielvariable und „src%()“ ist die Quellvariable. „src%()“ wird an das Ende von „dest%()“ angehängt (das Ziel wird **nicht** überschrieben).
LONGSTRING LCASE array%()	Konvertiert alle Großbuchstaben in „array%()” in Kleinbuchstaben. „array%()” muss eine lange Zeichenfolgenvariable sein.
LONGSTRING LEFT dest%(), src%(), nbr	Kopiert die ersten „nbr” Zeichen aus „src%()” nach „dest%()” und überschreibt dabei den bisherigen Inhalt von „dest%()”. Das heißt, es wird vom Anfang von „src%()” kopiert. „src%()” und „dest%()” müssen lange Zeichenfolgenvariablen sein. „nbr” muss eine ganzzahlige Konstante oder ein Ausdruck sein.
LONGSTRING LOAD array%(), nbr, string$	Kopiert 'nbr' Zeichen aus 'string$' in die lange String-Variable 'array%()' und überschreibt dabei den bisherigen Inhalt von 'array%()'.
LONGSTRING MID dest%(), src%(), start, nbr	Kopiert 'nbr' Zeichen aus „src%()” nach „dest%()”, beginnend an der Zeichenposition 'start', und überschreibt dabei den gesamten Inhalt von „dest%()”. Kopiert also aus der Mitte von „src%()”. „nbr” ist optional. Wird es weggelassen, werden die Zeichen von „start” bis zum Ende der Zeichenfolge kopiert. „src%()” und „dest%()” müssen lange Zeichenfolgenvariablen sein. „start” und „nbr” müssen ganzzahlige Konstanten oder Ausdrücke sein.
LONGSTRING PRINT [#n,] src%() [;]	Druckt die in „src%()” gespeicherte Longstring in die Datei oder den COM-Port, die bzw. der als „#n” geöffnet ist. Wenn „#n” nicht angegeben ist, wird die Ausgabe an die Konsole gesendet. Fügen Sie ein Semikolon hinzu, um CR/LF zu unterdrücken.
LONGSTRING REPLACE array%() , string$, start	Ersetzt Zeichen in der normalen MMBasic-Zeichenkette „string$“ durch eine vorhandene lange Zeichenkette „array%()“, beginnend an der Position „start“ in der langen Zeichenkette.
LONGSTRING RESIZE addr%(), nbr	Setzt die Größe der langen Zeichenkette auf „nbr“. Dies überschreibt die durch andere Befehle für lange Zeichenketten festgelegte Größe und sollte daher mit Vorsicht verwendet werden. Eine typische Anwendung wäre die Verwendung einer langen Zeichenkette als Byte-Array.
LONGSTRING RIGHT dest%(), src%(), nbr	Kopiert die rechten „nbr” Zeichen von „src%()” nach „dest%()” und überschreibt dabei den Inhalt von „dest%()”. Das heißt, es wird vom Ende von „src%()” kopiert. „src%()” und „dest%()” müssen Longstring-Variablen sein. „nbr” muss eine ganzzahlige Konstante oder ein Ausdruck sein.
LONGSTRING SETBYTE addr%(), nbr, data	Setzt das Byte „nbr” auf den Wert „data”, wobei „nbr” OPTION BASE berücksichtigt.
LONGSTRING TRIM array%(), nbr	Schneidet „nbr” Zeichen von der linken Seite einer langen Zeichenkette ab. „array%()” muss eine lange Zeichenkettenvariable sein. „nbr” muss eine ganzzahlige Konstante oder ein Ausdruck sein.
LONGSTRING UCASE array%()	Wandelt alle Kleinbuchstaben in „array%()“ in Großbuchstaben um. „array%()“ muss eine lange Zeichenfolgenvariable sein.
LOOP [UNTIL Ausdruck]	Beendet eine Programmschleife: siehe DO.
MAP	NUR HDMI-VERSION Mit den MAP-Befehlen kann der Programmierer die Farben festlegen, die in 4- oder 8-Bit-Farbmodi verwendet werden. Jeder Wert in der 4- oder 8-Bit-Farbpalette kann auf eine unabhängige 24-Bit-Farbe (d. h. RGB555-Format) eingestellt werden. Weitere Informationen finden Sie unter der Funktion MAP.
MAP(n) = rgb%	Dadurch wird allen Pixeln mit dem 4- oder 8-Bit-Farbwert „n” die 24-Bit-Farbe „rgb%” zugewiesen. Die Änderung wird nach dem Befehl MAP SET aktiviert.
MAP MAXIMITE	Dadurch wird die Farbtabelle auf die Farben eingestellt, die im ursprünglichen Colour Maximite implementiert sind.

MAP GREYSCALE	Dadurch wird die Farbtabelle auf 16 oder 32 Graustufen eingestellt (abhängig vom MODUS). MAP GRAYSCALE ist ebenfalls gültig.
MAP SET	Dadurch aktualisiert MMBasic die Farbtabelle (festgelegt mit MAP(n)=rgb%) während des nächsten Austastintervalls.
MAP RESET	Dadurch wird die Farbtabelle auf die Standardfarben zurückgesetzt.
MAP	PICOMITE RP2350 NUR GEPUFFERTE TREIBER Mit den MAP-Befehlen kann der Programmierer die im 8-Bit-Farbmodus RGB332 verwendeten Farben festlegen. Jeder Wert in der 8-Bit-Farbpalette kann auf eine unabhängige 24-Bit-Farbe (d. h. RGB888-Format) eingestellt werden. Weitere Informationen finden Sie unter der MAP-Funktion.
MAP(n) = rgb%	Dadurch wird allen Pixeln mit einem 8-Bit-Farbwert von „n" die 24-Bit-Farbe „rgb%" zugewiesen. Die Änderung wird nach dem Befehl MAP SET aktiviert.
MAP SET	Dadurch aktualisiert MMBasic die Farbtabelle (festgelegt mit MAP(n)=rgb%) während des nächsten Austastintervalls.
MAP RESET	Dadurch wird die Farbtabelle auf die Standardfarben zurückgesetzt.
MAP	NUR VGA-VERSION Mit den MAP-Befehlen kann der Programmierer die in 4-Bit verwendeten Farben auswählen. Farbmodi. Jeder Wert in der 4-Bit-Farbpalette kann auf eine der 16 verfügbaren Farben eingestellt werden. Weitere Informationen finden Sie unter der Funktion MAP.
MAP(n) = rgb%	Dadurch wird allen Pixeln mit dem 4-Bit-Farbwert „n" die 24-Bit-Farbe „rgb%" zugewiesen. Der RGB-Wert wird in eine der 16 verfügbaren VGA-RGB121-Farben Farben umgewandelt, wie durch das Widerstandsnetzwerk festgelegt. Die Änderung wird nach dem MAP SET-Befehl aktiviert.
MAP MAXIMITE	Dadurch wird die Farbtabelle auf die Farben eingestellt, die in der ursprünglichen Maximite-Farbpalette implementiert sind. Maximite.
MAP SET	Dadurch aktualisiert MMBasic die Farbtabelle (festgelegt mit MAP(n)=rgb%) während des nächsten Austastintervalls zu aktualisieren.
MAP RESET	Dadurch wird die Farbtabelle auf die Standardfarben zurückgesetzt, die im 4-Bit-Modus wie folgt lauten:

„n"	Farbe	Wert
15	WEISS	RGB(255, 255, 255)
14	GELB	RGB(255, 255, 0)
13	LILA	RGB(255, 128, 255)
12	BRAUN	RGB(255, 128, 0)
11	FUCHSIA	RGB(255, 64, 255)
10	ROST	RGB(255, 64, 0)
9	MAGENTA	RGB(255, 0, 255)
8	ROT	RGB(255, 0, 0)
7	CYAN	RGB(0, 255, 255)
6	GRÜN	RGB(0, 255, 0)
5	CERULEAN	RGB(0, 128, 255)
4	MITTLERES GRÜN	RGB(0, 128, 0)

	3 COBALT RGB(0, 64, 255) 2 MYRTLE RGB(0, 64, 0)
	1 BLAU RGB(0, 0, 255) 0 SCHWARZ RGB(0, 0, 0)
MATH	Der Befehl „math“ führt viele einfache mathematische Berechnungen aus, die in BASIC programmiert werden können. Es gibt jedoch Geschwindigkeitsvorteile, wenn Schleifenstrukturen in der Firmware codiert werden, und es hat den Vorteil, dass sie nach dem Debuggen für alle verfügbar sind, ohne dass das Rad neu erfunden werden muss. Hinweis: Zweidimensionale mathematische Matrizen werden immer mit DIM matrix(n_columns, n_rows) angegeben, wobei die Dimensionen natürlich OPTION BASE entsprechen. Quaternionen werden als Array mit 5 Elementen gespeichert: w, x, y, z, magnitude.
MATH RANDOMIZE [n]	Sät den Mersenne-Twister-Algorithmus. Wenn n nicht angegeben ist, ist der Startwert die Zeit in Mikrosekunden seit dem Systemstart. Der Mersenne-Twister-Algorithmus liefert viel bessere Zufallszahlen als die integrierte Funktion der C-Bibliothek. Hinweis: Der RP2350 verfügt über einen Hardware-Zufallszahlengenerator.
Einfache Array-Arithmetik	
MATH SET nbr, array()	Siehe ARRAY SET
MATH SCALE in(), scale ,out()	Dies skaliert die Matrix in() um den Skalar scale und speichert das Ergebnis in out(). Funktioniert für Arrays beliebiger Dimensionen sowohl für Ganzzahlen als auch für Gleitkommazahlen und kann zwischen diesen konvertieren. Die Einstellung b auf 1 ist optimiert und die schnellste Methode, um ein gesamtes Array zu kopieren.
MATH ADD in(), num ,out()	Siehe ARRAY ADD
MATH INTERPOLATE in1(), in2(), Verhältnis, out()	Dieser Befehl wendet die folgende Gleichung auf jedes Array-Element an: out = (in2 - in1) * ratio + in1 Arrays können eine beliebige Anzahl von Dimensionen haben, müssen jedoch eindeutig sein und die gleiche Gesamtzahl an Elementen aufweisen. Der Befehl funktioniert sowohl mit Ganzzahl- als auch mit Gleitkomma-Arrays in beliebiger Kombination.
MATH WINDOW in(), minout, maxout, out() [,minin!, maxin!]	Dieser Befehl nimmt das Array „in” und skaliert es zwischen „minout” und „maxout”, wobei das Ergebnis in „out” zurückgegeben wird. Optional kann er auch die minimalen und maximalen Gleitkommawerte aus den Originaldaten zurückgeben („minin!” und „maxin!”). Hinweis: „minout” kann größer als „maxout” sein. In diesem Fall werden die Daten sowohl skaliert als auch invertiert. Dieser Befehl kann die Skalierung von Daten für die Darstellung usw. erheblich vereinfachen. Beispiel: DIM IN(2)=(1,2,3) DIM OUT(2) MATH WINDOW IN(),7,3,OUT(),LOW,HIGH Gibt OUT(0)=7, OUT(1)=5,OUT(2)=3,LOW=1,HIGH=3 zurück. Siehe ARRAY
MATH SLICE sourcearray(), [d1] [,d2] [,d3] [,d4] [,d5] , Zielarray()	SLICE
MATH INSERT Zielarray(), [d1] [,d2] [,d3] [,d4] [,d5] , sourcearray()	Siehe ARRAY INSERT

MATH POWER inarray(), power, outarray()	Erhebt jedes Element in „inarray()" zur definierten „Potenz" und speichert die Ausgabe in „outarray()".
MATH SHIFT inarray%(), nbr, outarray%() [,U]	Dieser Befehl führt eine Bitverschiebung aller Elemente von inarray%() durch und speichert das Ergebnis in outarray%() (kann mit inarray%()) identisch sein. nbr kann zwischen -63 und 63 liegen. Positive Zahlen werden nach links verschoben (mit der Potenz von 2 multipliziert). Negative Zahlen werden nach rechts verschoben. Der optionale Parameter ‚U erzwingt eine vorzeichenlose Verschiebung.
Matrixarithmetik	
MATH M_INVERSE array!(), inversearray!()	Dies gibt die Umkehrung von array!() in inversearray!() zurück. Das Array muss quadratisch sein, und Sie erhalten eine Fehlermeldung, wenn das Array nicht umgekehrt werden kann (Determinante = 0). array!() und inversearray!() können nicht identisch sein.
MATH M_PRINT array()	Schneller Mechanismus zum Drucken einer 2D-Matrix mit einer Zeile pro Zeile.
MATH M_TRANSPOSE in(), out()	Transponiere die Matrix in() und speichere das Ergebnis in der Matrix out(). Beide Arrays müssen zweidimensional sein, müssen jedoch nicht quadratisch sein. Sind sie nicht quadratisch, müssen die Arrays die Dimensionen in(m,n) und out(n,m) haben.
MATH M_MULT in1(), in2(), out()	Multipliziert die Arrays in1() und in2() und speichert das Ergebnis in out()c. Alle Arrays müssen zweidimensional sein, müssen jedoch nicht quadratisch sein. Sind sie nicht quadratisch, müssen die Arrays die Dimensionen in1(m,n), in2(p,m) und out(p,n) haben.
Vektorarithmetik	
MATH V_PRINT array() [,hex]	Schneller Mechanismus zum Drucken eines kleinen Arrays in einer einzigen Zeile. „hex" druckt in Hexadezimal.
MATH V_NORMALISE inV(), outV()	Konvertiert einen Vektor in V() in die Einheitsskala und speichert das Ergebnis in outV() (sqr(x*x + y*y +)=1 Die Anzahl der Elemente im Vektor ist unbegrenzt
MATH V_MULT matrix(), inV(), outV()	Multipliziert matrix() und den Vektor inV() und gibt den Vektor outV() zurück. Die Vektoren und die 2D-Matrix können beliebig groß sein, müssen jedoch dieselbe Kardinalität aufweisen.
MATH V_CROSS inV1(), inV2(), outV()	Berechnet das Kreuzprodukt der beiden dreielementigen Vektoren inV1() und inV2() und speichert das Ergebnis in outV().
MATH V_ROTATE x, y, a, xin(), yin(), xout(), yout()	Dieser Befehl dreht die Koordinatenpaare in „xin()" und „yin()" um den durch „x" und „y" definierten Mittelpunkt um den Winkel „a" und speichert die Ergebnisse in „xout()" und „yout()". Hinweis: Die Eingabe- und Ausgabe-Arrays können identisch sein, und der Drehwinkel ist standardmäßig in Radianten angegeben, kann jedoch mit dem Befehl OPTION ANGLE geändert werden.
Quaternion-Arithmetik	
MATH Q_INVERT inQ(), outQ()	Invertiert das Quaternion in inQ() und speichert das Ergebnis in outQ().
MATH Q_VECTOR x, y, z, outVQ()	Konvertiert einen durch x, y und z angegebenen Vektor in einen normalisierten Quaternion-Vektor outVQ() mit der ursprünglichen Größe
MATH Q_CREATE theta, x, y, z, outRQ()	Erzeugt einen normalisierten Rotationsquaternion outRQ(), um Quaternion-Vektoren um die Achsen x, y, z um einen Winkel theta zu drehen. Theta wird in Radianten angegeben.

MATH Q_EULER Gierwinkel, Neigungswinkel, Rollwinkel, outRQ()	Erzeugt einen normalisierten Rotationsquaternion outRQ(), um Quaternion-Vektoren gemäß den definierten Gier-, Nick- und Rollwinkeln zu drehen. Wenn sich der Vektor vor dem „Betrachter" befindet, schaut Yaw von der Spitze des Vektors aus und dreht sich im Uhrzeigersinn, Pitch dreht die Spitze von der Kamera weg und Roll dreht sich um die Z-Achse im Uhrzeigersinn.
MATH Q_MULT inQ1() inQ2(), outQ() MATH Q_ROTATE , RQ(), inVQ(), outVQ()	Die Gier-, Neigungs- und Rollwinkel sind standardmäßig auf Radianten eingestellt, berücksichtigen jedoch die Einstellung von OPTION ANGLE Multipliziert zwei Quaternionen inQ1() und inQ2() und speichert das Ergebnis in outQ() Dreht den Quaternion-Vektor inVQ() um den Drehquaternion RQ() und speichert die Antwort in outVQ()
MATH C_ADD array1(), array2(), array3() MATH C_SUB array1(), array2(), array3() MATH C_MUL array1(), array2(), array3() MATH C_DIV array1(), array2(), array3() MATH C_AND array1(), array2(), array3() MATH C_OR array1(), array2(), array3() MATH C_XOR array1(), array2(), array3()	Diese Befehle führen zellweise Operationen auf Arrays durch. array1 und array2 sind die Parameter und array3 ist die Ausgabe. Alle Arrays müssen dieselbe Größe und denselben Typ (Float oder Integer) haben. Es gibt keine Beschränkungen hinsichtlich der Anzahl der Dimensionen und keine Beschränkungen hinsichtlich der Verwendung desselben Arrays zweimal oder sogar dreimal in den Parametern. Beispiel: MATH C_MUL a%(),a%(),a%() quadriert alle Werte im Array a%().
MATH FFT signalarray!(), FFTarray!()	Führt eine schnelle Fourier-Transformation der Daten in „signalarray!" durch. „signalarray" muss ein Fließkommawert sein und die Größe muss eine Potenz von 2 sein (z. B. s(1023), vorausgesetzt OPTION BASE ist Null). „FFTarray" muss ein Gleitkommawert sein und die Dimension 2*N haben, wobei N dem Signalarray entspricht (z. B. f(1,1023), vorausgesetzt OPTION BASE ist Null). Der Befehl gibt die FFT als komplexe Zahlen zurück, wobei der Realteil in f(0,n) und der Imaginärteil in f(1,n) enthalten ist.
MATH FFT INVERSE FFTarray!(), signalarray!()	Führt eine inverse schnelle Fourier-Transformation der Daten in „FFTarray!" durch. „FFTarray" muss ein Fließkommawert sein und die Dimension 2*N haben, wobei N eine Potenz von 2 sein muss (z. B. f(1,1023), vorausgesetzt, OPTION BASE ist Null), mit dem Realteil in f(0,n) und dem Imaginärteil in f(1,n). „signalarray" muss ein Gleitkommawert sein und die einzelne Dimension muss mit dem FFT-Array übereinstimmen. Der Befehl gibt den Realteil der inversen Transformation in „signalarray" zurück.
MATH FFT MAGNITUDE signalarray!(),magnitudearray! ()	Erzeugt Magnituden für Frequenzen für die Daten in „signalarray!". „signalarray" muss ein Fließkommawert sein und die Größe muss eine Potenz von 2 sein (z. B. s(1023), wenn OPTION BASE Null ist). „magnitudearray" muss ein Fließkommawert sein und die Größe muss mit der des Signal-Arrays übereinstimmen. Der Befehl gibt die Amplitude des Signals bei verschiedenen Frequenzen gemäß der folgenden Formel zurück: Frequenz an der Array-Position N = N * Abtastfrequenz / Anzahl der Abtastwerte
MATH FFT PHASE signalarray!(), phasearray!()	Erzeugt Phasen für Frequenzen für die Daten in „signalarray!". „signalarray" muss ein Fließkommawert sein und die Größe muss eine Potenz von 2 sein (z. B. s(1023), wenn OPTION BASE Null ist). „phasearray" muss ein Fließkommawert sein und die Größe muss mit der des Signal-Arrays übereinstimmen. Der Befehl gibt den Phasenwinkel des Signals bei verschiedenen Frequenzen gemäß der obigen Formel zurück.

MATH SENSORFUSION Typ ax, ay, az, gx, gy, gz, mx, my, mz, Neigung, Rollwinkel, Gierwinkel [,p1] [,p2]	Typ kann MAHONY oder MADGWICK sein. Ax, ay und az sind die Beschleunigungen in den drei Richtungen und sollten in Einheiten der Standardbeschleunigung angegeben werden. Gx, gy und gz sind die Momentanwerte der Drehzahl, die in Radiant pro Sekunde angegeben werden sollten.
	Mx, my und mz sind die Magnetfelder in den drei Richtungen und sollten in Nano-Tesla (nT) angegeben werden. Es muss darauf geachtet werden, dass die x-, y- und z-Komponenten zwischen den drei Eingaben konsistent sind. Bei Verwendung des MPU-9250 lautet die korrekte Eingabe beispielsweise ax, ay, az, gx, gy, gz, my, mx, -mz, basierend auf den Messwerten des Sensors. Pitch, Roll und Yaw sollten Fließkommavariablen sein und die Ausgaben der Sensorfusion enthalten. Die Routine SENSORFUSION misst automatisch die Zeit zwischen aufeinanderfolgenden Aufrufen und verwendet diese für ihre internen Berechnungen. Der Madwick-Algorithmus verwendet einen optionalen Parameter p1. Dieser wird bei der Berechnung als Beta verwendet. Wenn er nicht angegeben wird, ist der Standardwert 0,5. Der Mahony-Algorithmus verwendet zwei optionale Parameter p1 und p2. Diese werden in der Berechnung als Kp und Ki verwendet. Wenn sie nicht angegeben werden, sind die Standardwerte 10,0 bzw. 0,0. Ein vollständig ausgearbeitetes Beispiel für die Verwendung des Codes finden Sie im BackShed-Forum unter: https://www.thebackshed.com/forum/ViewTopic.php?TID=13459&PID=166962
MATH SINC x_in(), y_in(), n , m, window, freq, x_out(), y_out() MATH SINC x_in(), y_in(), n, window, freq, x_out(), y_out()	Wendet einen Fenster-Sinc-Filter an, um Koordinatendaten zu glätten oder zu interpolieren. Ein Sinc-Filter ist ein idealer Tiefpassfilter, der hochfrequente Störgeräusche entfernt und gleichzeitig die Signalform beibehält. Er eignet sich besonders gut für das Resampling (Interpolation). Parameter: x_in(), y_in() — Eingabekoordinaten-Arrays (float) n — Anzahl der Eingabepunkte m — Anzahl der Ausgabepunkte (optional). Wenn weggelassen oder m=n: Glättungsmodus (Ausgabegröße n). Wenn m!=n: Interpolations-/Resampling-Modus (Ausgabegröße m) window — Filterkernelgröße (muss ungerade sein; gerade Werte werden erhöht). Typische Werte: 15 bis 101. Größere Werte ergeben eine schärfere Trennlinie, sind jedoch langsamer. freq — Normalisierte Grenzfrequenz (0,0 < Freq <= 0,5). 0,5 ist die Nyquist-Frequenz (keine Filterung). Typische Werte: 0,1 (starke Glättung) bis 0,4 (leichte Glättung). x_out(), y_out() — Ausgabe-Koordinaten-Arrays (müssen groß genug sein, um die Ausgabe aufzunehmen)

MATH PID INIT-Kanal, pid_params!(), Callback	Dieser Befehl richtet einen PID-Regler ein, der automatisch im Hintergrund arbeiten kann. Es können bis zu 8 PID-Regler gleichzeitig laufen (Kanäle 1 bis 8). „callback" ist eine MMbasic-Subroutine, die mit der durch die Abtastzeit definierten Rate aufgerufen wird. Einzelheiten dazu, was in der Subroutine enthalten sein sollte, finden Sie unter der Funktion MATH(PID …). Das Array pid_params!() muss für alle aufgeführten Elemente dimensioniert werden, einschließlich der Reglerspeicherparameter (DIM pid_params!(13)), und mit den erforderlichen Einstellungen initialisiert werden. PID-Konfiguration Element 0 = Kp Element 1 = Ki Element 2 = Kd Element 3 = tau ' Zeitkonstante des derivativen Tiefpassfilters Element 4 = limMin 'Ausgangsbegrenzungen Element 5 = limMax
MATH PID START-Kanal MATH PID STOP-Kanal	Element 6 = limMinInt 'Integrator-Grenzen Element 7 = limMaxInt Element 8 = T 'Abtastzeit (in Sekunden) Controller-„Speicher" Element 9 = Integrator Element 10 = prevError Element 11 = Differentiator Element 12 = prevMeasurement Element 13 = out Startet einen zuvor initialisierten PID-Regler auf dem angegebenen Kanal Stoppt einen zuvor initialisierten PID-Regler auf dem angegebenen Kanal und löscht die internen Datenstrukturen Siehe https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17263 Beispiel für die Einrichtung und den Betrieb eines PID-Reglers

MATH AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key(), in$/in(), out$/out() [,iv$/iv()]	Dieser Befehl verschlüsselt oder entschlüsselt die Daten in „in” und speichert das Ergebnis unter Verwendung der angegebenen AES128-Verschlüsselungsmethode in „out”. Die Parameter können jeweils entweder eine Zeichenfolge, ein Integer-Array oder ein Float-Array sein, wobei jede Kombination möglich ist. Der Schlüssel muss 16 Elemente lang sein (16*8=128 Bit), „in” und „out” müssen ein Vielfaches von 16 Elementen lang sein. Wenn „out” als Zeichenfolge angegeben wird (z. B. out$), muss die Zeichenfolgenvariable vorhanden sein und sollte auf leer gesetzt werden (DIM out$=""). Die maximale Anzahl von Elementen in „in” und „out” ist durch den Speicher begrenzt, wenn sie als Arrays definiert sind. Zeichenfolgen für die Verschlüsselung sind auf 240 Byte (EBR) und 224 Byte (CTR und CBC) begrenzt. Für die CBC- und CTR-Verschlüsselung können Sie optional einen Initialisierungsvektor „iv” angeben. „iv” muss 16 Elemente lang sein (16*8=128 Bit). Wenn kein Initialisierungsvektor angegeben ist, generiert die Verschlüsselung einen zufälligen Initialisierungsvektor. In beiden Fällen wird der IV vor die Ausgabe gestellt. Bei CBC und CTR müssen die ersten 16 Elemente der Eingabe der Initialisierungsvektor sein, damit die Entschlüsselung funktioniert. Wenn Sie eine Nachricht ohne IV übertragen möchten, sollten Sie die IV vor dem Senden der Nachricht mit Standard-MMBasic-Manipulationen entfernen. In diesem Fall muss der Empfänger sowohl die IV als auch den Schlüssel kennen und eine vollständige Nachricht erstellen, bevor er DECRYPT verwendet, indem er die IV an die eingehende Nachricht anhängt.

MEMORY	Listet die derzeit belegte Speichermenge auf. Beispiel:
	`Programm:`
	`  0K ( 0%) Programm (0 Zeilen)`
	`180K (100 %) Frei`
	
	`Gespeicherte`
	`  Variablen: 16K`
	`  (100 %) Frei`
	
	`RAM:`
	`  0K ( 0%) 0 Variablen 0K (`
	`  0%) Allgemein`
	`228K (100 %) Frei`

	Hinweise: • Die Speichernutzung wird auf die nächsten 1 KB gerundet. • Der allgemeine Speicher (RAM) wird von Arrays, Strings, seriellen E/A-Puffern usw. verwendet.

MEMORY SET Adresse, Byte, Anzahl der Bytes MEMORY SET BYTE Adresse, Byte, Anzahl der Bytes MEMORY SET SHORT Adresse, Short, Anzahl der Shorts MEMORY SET WORD Adresse, Wort, Anzahl der Wörter SPEICHER SET INTEGER Adresse, ganzzahliger Wert ,Anzahl der Ganzzahlen [,Inkrement] SPEICHER SET FLOAT Adresse, Gleitkommawert ,AnzahlderGleitkommazahlen [,Inkrement]	Dieser Befehl setzt einen Speicherbereich auf einen Wert. BYTE = Ein Byte pro Speicheradresse. SHORT = Zwei Bytes pro Speicheradresse. WORD = Vier Bytes pro Speicheradresse. FLOAT = Acht Bytes pro Speicheradresse. „increment“ ist optional und steuert die Inkrementierung des Zeigers „address“ während der Ausführung der Operation. Wenn beispielsweise increment=3 ist, wird nur jedes dritte Element des Ziels gesetzt. Der Standardwert ist 1.
SPEICHERKOPIE Quelladresse, Zieladresse, Anzahl der Bytes [,Quellinkrement][,Zielinkrement] SPEICHERKOPIE INTEGER Quelladresse, Zieladresse, Anzahl der Ganzzahlen [,Quellinkrement][,Zielinkrement] SPEICHERKOPIERUNG FLOAT Quelladresse, Zieladresse, Anzahl der Fließkommazahlen [,Quellinkrement][,Zielinkrement]	Dieser Befehl kopiert einen Speicherbereich in einen anderen. COPY INTEGER und FLOAT kopieren acht Bytes pro Vorgang. „sourceincrement” ist optional und steuert die Inkrementierung des Zeigers „sourceaddress” während der Ausführung des Vorgangs. Wenn beispielsweise sourceincrement=3 ist, wird nur jedes dritte Element der Quelle kopiert. Der Standardwert ist 1. „destinationincrement” funktioniert ähnlich und wirkt sich auf den Zeiger „destinationaddress” aus.
MEMORY PRINT #]fnbr , nbr, address%/array() SPEICHEREINGABE [#]fnbr , nbr, Adresse%/Array()	Diese Befehle speichern oder lesen „nbr” Datenbytes aus dem Speicher oder in einen offenen Datei auf der Festplatte. Der zu speichernde Speicher kann als ganzzahliges Array angegeben werden. In diesem Fall wird die Anzahl der zu speichernden oder zu lesenden Bytes mit der Array-Größe verglichen. Alternativ kann eine Speicheradresse verwendet werden. In diesem Fall findet keine Überprüfung statt, und Benutzerfehler können zu einem Absturz der Firmware führen.

MEMORY PACK source%()/sourceaddress%, dest%()/destaddress%, number, size MEMORY UNPACK source%()/sourceaddress%, dest%()/destaddress%, number, size	Mit Memory Pack und Unpack können ganzzahlige Werte aus einem Array in ein anderes komprimiert oder von einem in das andere dekomprimiert werden. Die beiden Arrays sind immer normale Integer-Arrays, aber das gepackte Array kann 2, 4, 8, 16 oder 64 Werte „gepackt“ enthalten. Somit kann ein einzelnes Integer-Array-Element 2 32-Bit-Wörter, 4 16-Bit-Werte, 8 Bytes, 16 Nibbles oder 64 Boolesche Werte (Bits) speichern. „number“ gibt die Anzahl der zu packenden oder zu entpackenden Werte an und „size“ gibt die Anzahl der Bits (1, 4, 8, 16 oder 32) an. Alternativ können Speicheradressen verwendet werden. In diesem Fall kann keine Überprüfung stattfinden, und Benutzerfehler können zu einem Absturz der Firmware führen.
MKDIR dir$	Erstellen Sie das Verzeichnis „dir$“ im Standard-Flash-Dateisystem oder auf der SD-Karte.
MID$(str$, start, num) = str2$	Die „num” Zeichen in „str$”, beginnend an der Position „start”, werden durch die Zeichen in „str2$” ersetzt. „num” kann auf 0 gesetzt werden, d. h. die neue Zeichenfolge wird an der angegebenen Position eingefügt. Wenn „num” nicht angegeben ist, wird standardmäßig die Länge von „str2$” verwendet.
MODUS 1 oder MODUS 2 oder MODUS 3 (nur RP2350)	<u>NUR VGA-VERSIONEN</u> VGA-Video unterstützt eine Reihe von Auflösungen (siehe OPTION AUFLÖSUNG). Dieser Befehl wählt den Modus „n” in Abhängigkeit von der Auflösung aus: <u>OPTION AUFLÖSUNG 640 x 480</u> MODUS 1 640 x 480 x 2 Farben (monochrom). Standard bei Start. Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann jedoch zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden unter Verwendung der Standardnotation RGB888 angegeben. Diese wird in RGB121 umgewandelt. Es können ein Framebuffer (F) und ein Layer-Buffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und belegen keinen Speicherplatz, können jedoch beide zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. MODUS 2 320 x 240 x 16 Farben. RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheint. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind. MODUS 3 640 x 480 x 16 Farben. RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheint. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind.

OPTION AUFLÖSUNG 720 x 400

MODUS 1 720 x 400 x 2 Farben (monochrom). Standard bei Start.
Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann aber zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden in der Standardnotation RGB888 angegeben. Diese wird in RGB121 umgewandelt. Es können ein Framebuffer (F) und ein Layer-Buffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und belegen keinen Benutzerspeicher, können jedoch beide zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.

MODUS 2 360 x 200 x 16 Farben.
RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern alles, was sich im Hauptanzeigepuffer befindet. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheinen kann. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Bei VGA ist die Hardware auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind.

MODUS 3 720 x 400 x 16 Farben.
RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern alles, was sich im Hauptanzeigepuffer befindet. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheinen kann. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Im Falle von VGA ist die Hardware auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind.

OPTION AUFLÖSUNG 800 x 600 (nur RP2350)

MODUS 1 800 x 600 x 2 Farben (monochrom). Standard bei Start.
Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann jedoch zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden unter Verwendung der Standardnotation RGB888 angegeben. Diese wird in RGB121 konvertiert. Es können ein Framebuffer (F) und ein Layer-Buffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und belegen keinen Benutzerspeicher, können jedoch beide zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.

MODUS 2 400 x 300 x 16 Farben.
RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheint. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind.

	MODUS 3 800 x 600 x 16 Farben. RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser beansprucht keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheinen kann. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind. <u>OPTION AUFLÖSUNG 848 x 480 (nur RP2350)</u> MODUS 1 848 x 480 x 2 Farben (monochrom). Standard bei Start. Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann jedoch zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden in der Standardnotation RGB888 angegeben. Diese wird in RGB121 umgewandelt. Es können ein Framebuffer (F) und ein Layer-Buffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und belegen keinen Benutzerspeicher, können jedoch beide zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. MODUS 2 424 x 240 x 16 Farben. RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheint. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind. MODUS 3 848 x 48 x 16 Farben. RGB121-Format (d. h. 1 Bit für Rot, 2 Bits für Grün und 1 Bit für Blau). Es kann ein Bildspeicher (F) erstellt werden. Dieser hat keinen Einfluss auf die Anzeige und belegt keinen Benutzerspeicher, kann jedoch zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Auch dieser belegt keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und überlagern den Inhalt des Hauptdisplaypuffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptdisplaypuffer durchscheint. Mit der Map-Funktion können die Standardfarben der 16 verfügbaren Farben überschrieben werden. Die Hardware ist auf die 16 Farben beschränkt, die durch das Widerstandsnetzwerk definiert sind.
MODUS n	<u>NUR HDMI-VERSIONEN</u> HDMI-Video unterstützt eine Reihe von Auflösungen (siehe OPTION AUFLÖSUNG). Dieser Befehl wählt je nach Auflösung den Modus „n“ aus: <u>OPTION AUFLÖSUNG 640 x 480</u> MODUS 1 640 x 480 x 2 Farben (monochrom). Standard bei Start. Verwenden Sie den Befehl TILE wie gewohnt. Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann jedoch zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden mithilfe der Standardnotation

RGB888-Notation festgelegt. Diese wird in RGB555 konvertiert. Es können ein Framebuffer (F) und ein Layer-Buffer (L) erstellt werden. Diese können zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.

MODUS 2 320 x 240 x 16 Farben.
Es kann ein Framebuffer (F) erstellt werden. Dieser kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Layer-Puffer erstellt werden. Alle in den Layer-Puffer geschriebenen Pixel werden automatisch auf dem Bildschirm angezeigt und überlagern alles, was sich im Hauptbildschirm-Puffer befindet. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptbildschirm-Puffer durchscheinen kann. Die Kartenfunktion kann verwendet werden, um die Standardfarben zu überschreiben.

MODUS 3 640 x 480 x 16 Farben.
Farbzuordnung zur RGB555-Palette. Es kann ein Framebuffer (F) erstellt werden. Dieser kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Layer-Puffer erstellt werden. Alle in den Layer-Puffer geschriebenen Pixel werden automatisch auf dem Display angezeigt und liegen über den Daten im Hauptdisplay-Puffer. Es kann eine Farbe festgelegt werden (0–15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptdisplay-Puffer durchscheinen kann.

MODUS 4 320 x 240 x 32768 Farben.
Dies ist Voll-RGB555, wodurch Farbbilder in guter Qualität angezeigt werden können. Es können ein Bildspeicher (F) und ein Ebenenpuffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und können zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.
Es kann nur einer erstellt

werden MODUS 5 320 x 240 x 256
Farben.

Es kann ein Framebuffer (F) erstellt werden. Dies hat keine Auswirkungen auf die
Anzeige. Er kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Dieser verwendet keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Bildschirm angezeigt und überlagern alles, was sich im Hauptanzeigepuffer befindet. Es kann eine Farbe festgelegt werden (0-255: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheinen kann. Die Kartenfunktion kann verwendet werden, um die Standardfarben der 256 verfügbaren Farben zu überschreiben.
Jede der 256 Farben kann einer beliebigen RGB555-Farbe zugeordnet werden.

OPTION AUFLÖSUNG 720 x 400

MODUS 1 720 x 400 x 2 Farben (monochrom). Standard bei Start. Verwenden Sie den Befehl TILE wie gewohnt. Die Breite der Kacheln ist auf 8 Pixel festgelegt. Die Höhe der Kacheln beträgt standardmäßig 12 Pixel, kann jedoch zwischen 8 und MM.HRES liegen. Die Farben der Kacheln werden in der Standardnotation RGB888 angegeben. Diese wird in RGB555 umgewandelt. Es können ein Framebuffer (F) und ein Layer-Puffer (L) erstellt werden. Diese können zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.

MODUS 2 360 x 200 x 16 Farben.
Es kann ein Framebuffer (F) erstellt werden. Dieser kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Layer-Puffer erstellt werden. Alle in den Layer-Puffer geschriebenen Pixel werden automatisch auf dem Bildschirm angezeigt und überlagern alles, was sich im Hauptbildschirm-Puffer befindet. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptbildschirm-Puffer durchscheinen kann. Die Kartenfunktionalität kann verwendet werden, um die Standardfarben zu überschreiben.

MODUS 3 720 x 400 x 16 Farben.
Farbzuordnung zur RGB555-Palette. Es kann ein Framebuffer (F) erstellt werden. Dieser kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Layer-Puffer erstellt werden. Alle in den Layer-Puffer geschriebenen Pixel werden automatisch auf dem Bildschirm angezeigt und überlagern den Inhalt des Hauptbildschirm-Puffers. Es kann eine Farbe festgelegt werden (0-15: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptbildschirm-Puffer durchscheinen kann.

MODUS 4 360 x 200 x 32768 Farben.
Dies ist Voll-RGB555, wodurch Farbbilder in guter Qualität angezeigt werden können. Es können ein Bildspeicher (F) und ein Ebenenpuffer (L) erstellt werden. Diese haben keinen Einfluss auf die Anzeige und können zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden.
Es kann nur einer erstellt

werden. MODUS 5 360 x 200 x 256

Farben.

Es kann ein Framebuffer (F) erstellt werden. Dieser hat keinen Einfluss auf die
Anzeige. Es kann zum Erstellen von Bildern und zum Kopieren auf den Bildschirm (N) verwendet werden. Zusätzlich kann ein Ebenenpuffer erstellt werden. Dieser beansprucht keinen Benutzerspeicher. Alle in den Ebenenpuffer geschriebenen Pixel werden automatisch auf dem Bildschirm angezeigt und überlagern den Inhalt des Hauptanzeigepuffers. Es kann eine Farbe festgelegt werden (0-255: Standardwert ist 0), die nicht angezeigt wird, sodass der Hauptanzeigepuffer durchscheinen kann. Die Map-Funktion kann verwendet werden, um die Standardfarben der 256 verfügbaren Farben zu überschreiben.
Jede der 256 Farben kann einer beliebigen RGB555-Farbe zugeordnet werden.

OPTION AUFLÖSUNG 800 x 600 (Hinweis: reduziert den verfügbaren Heap-Speicher)

MODUS 1 800 x 600 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt)

MODUS 2 400 x 300 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette

MODUS 3 800 x 400 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette

MODUS 5 400 x 300 x 256 Farben mit optionaler Ebene (kein Speicherverbrauch)

OPTIONALE AUFLÖSUNG 848 x 480 (Hinweis: reduziert den verfügbaren Heap-Speicher)

MODUS 1 848 x 480 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt)

MODUS 2 424 x 240 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette

MODUS 3 848 x 480 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette

MODUS 5 424 x 240 x 256 Farben mit optionaler Ebene (kein Speicherverbrauch)

OPTIONALE AUFLÖSUNG 1280 x 720

MODUS 1 1280 x 720 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt)

MODUS 2 320 x 180 x 16 Farben mit 2 optionalen Ebenen und Farbabbildung auf RGB332-Palette

MODUS 3 640 x 360 x 16 Farben mit optionaler Ebenen- und Farbabbildung auf RGB332-Palette

MODUS 5 320 x 180 x 256 Farben mit optionaler Ebene (kein Speicherverbrauch)

	OPTIONALE AUFLÖSUNG 1024 x 768 MODUS 1 1024 x 768 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt) MODUS 2 256 x 192 x 16 Farben mit 2 optionalen Ebenen und Farbabbildung auf RGB332-Palette MODUS 3 512 x 384 x 16 Farben MODUS 5 256 x 192 x 256 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette OPTION AUFLÖSUNG 1024 x 600 MODUS 1 1024 x 600 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt) MODUS 2 256 x 150 x 16 Farben mit 2 optionalen Ebenen und Farbabbildung auf RGB332-Palette MODUS 3 512 x 300 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette MODUS 5 256 x 150 x 256 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette OPTIONALE AUFLÖSUNG 800 x 480 (Hinweis: reduziert den verfügbaren Heap-Speicher) MODUS 1 800 x 480 x 2 Farben mit RGB332-Kacheln (verwenden Sie den Befehl TILE wie gewohnt) MODUS 2 400 x 240 x 16 Farben mit 2 optionalen Ebenen und Farbabbildung auf RGB332-Palette MODUS 3 800 x 480 x 16 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette MODUS 5 400 x 240 x 256 Farben mit optionaler Ebene und Farbabbildung auf RGB332-Palette
MODUS 400 oder MODUS 800	NUR RP2350 PICOMITE-VERSIONEN Wechselt zwischen den Modi 400x240 und 800x480 für SSD1963-Puffertreiber
MAUS MAUS-INTERRUPT Kanal aktivieren, int MOUSE INTERRUPT Kanal DEAKTIVIEREN MOUSE SET Kanal, x-Koordinate, y-Koordinate [, Radanzahl]	Für alle Varianten des Befehls. Bei USB-Firmware ist „Kanal" der USB-Anschluss, an den die Maus angeschlossen ist (1-4). Weitere Informationen finden Sie unter MM.INFO(USB n). Bei PS2-Firmware ist „Kanal" fest auf den Wert 2 eingestellt. „int" ist eine benutzerdefinierte Subroutine, die aufgerufen wird, wenn die linke Maustaste gedrückt wird. Deaktiviert einen Interrupt für die linke Maustaste. Legt die aktuelle Position fest, die von der Maus zurückgegeben wird x-, y- und optional Radpositionen
MOUSE OPEN-Kanal, CLKpin, DATApin	NICHT-USB-VERSIONEN – NUR FÜR EINE PS2-MAUS Öffnet eine Verbindung zu einer PS2-Maus, die an die beiden angegebenen Pins angeschlossen ist. Dieser Befehl kann in einem Programm verwendet werden, um die Maus während der Ausführung des Programms zu konfigurieren, im Gegensatz zu OPTION MOUSE, das die Maus dauerhaft konfiguriert.

MOUSE CLOSE channel	Der Kanal ist aus Kompatibilitätsgründen mit der USB-Mausfunktionalität enthalten und muss auf 2 gesetzt werden. Wenn keine Maus angeschlossen ist, erhalten Sie eine Fehlermeldung und können den Befehl erneut aufrufen, sobald die Maus angeschlossen ist. Schließt den Zugriff auf die Maus und stellt die Pins wieder auf normale Verwendung zurück. Der Befehl schlägt fehl, wenn OPTION MOUSE eingestellt wurde.
NEU	Löscht den Programmspeicher und alle Variablen, einschließlich gespeicherter Variablen.
NEXT [Zählvariable] [, Zählvariable] usw.	NEXT steht am Ende einer FOR-NEXT-Schleife; siehe FOR. Die „Zählvariable" gibt genau an, welche Schleife bearbeitet wird. Wenn keine „Zählvariable" angegeben ist, wird NEXT standardmäßig auf die innerste Schleife angewendet. Es ist auch möglich, mehrere Zählvariablen anzugeben, wie in: `NEXT x, y, z`
ON ERROR ABORT oder ON ERROR IGNORE oder BEI FEHLER ÜBERSPRINGEN [nn] oder BEI FEHLER LÖSCHEN oder BEI FEHLER NEU STARTEN	Dies steuert die Aktion, die bei einem Fehler während der Ausführung eines Programms durchgeführt wird, und gilt für alle von MMBasic entdeckten Fehler, einschließlich Syntaxfehlern, falschen Daten, fehlender Hardware usw. ON ERROR ABORT bewirkt, dass MMBasic eine Fehlermeldung anzeigt, das Programm abbricht und zur Eingabeaufforderung zurückkehrt. Dies ist das normale Verhalten und die Standardeinstellung, wenn ein Programm gestartet wird. ON ERROR RESTART bewirkt, dass MMbasic einen Hardware-Reset durchführt, und wenn Sie OPTION AUTORUN eingestellt haben, wird das Programm natürlich sauber neu gestartet. ON ERROR IGNORE bewirkt, dass alle Fehler ignoriert werden. ON ERROR SKIP ignoriert einen Fehler in einer Reihe von Befehlen (angegeben durch die Zahl „nn"), die nach diesem Befehl ausgeführt werden. „nn" ist optional, der Standardwert ist eins, wenn nichts angegeben ist. Nachdem die Anzahl der Befehle abgeschlossen ist (mit oder ohne Fehler), kehrt MMBasic zum Verhalten von ON ERROR ABORT zurück. Wenn ein Fehler auftritt und ignoriert/übersprungen wird, wird die schreibgeschützte Variable MM.ERRNO auf einen Wert ungleich Null gesetzt und MM.ERRMSG$ wird auf die Fehlermeldung gesetzt, die normalerweise generiert würde. Diese werden durch ON ERROR CLEAR auf Null und eine leere Zeichenfolge zurückgesetzt. Sie werden auch gelöscht, wenn das Programm ausgeführt wird und wenn ON ERROR IGNORE und ON ERROR SKIP verwendet werden. ON ERROR IGNORE kann die Fehlersuche in einem Programm erheblich erschweren, daher wird dringend empfohlen, nur ON ERROR SKIP zu verwenden.
ON KEY target oder ON KEY ASCIIcode, Ziel	Die erste Version des Befehls setzt einen Interrupt, der die benutzerdefinierte Subroutine „target" aufruft, sobald ein oder mehrere Zeichen im Eingabepuffer der seriellen Konsole warten. Beachten Sie, dass alle im Eingabepuffer wartenden Zeichen in der Interrupt-Subroutine gelesen werden müssen, da sonst automatisch ein weiterer Interrupt generiert wird, sobald das Programm aus dem Interrupt zurückkehrt. Die zweite Version ermöglicht es Ihnen, eine Interrupt-Routine mit einem bestimmten Tastendruck zu verknüpfen. Diese funktioniert auf einer niedrigen Ebene für die serielle Konsole, und wenn sie aktiviert ist, wird die Taste nicht in den Eingabepuffer gestellt, sondern löst lediglich den Interrupt aus. Sie verwendet einen separaten Interrupt vom einfachen Befehl ON KEY, sodass sie bei Bedarf gleichzeitig verwendet werden kann. In beiden Varianten verwenden Sie zum Deaktivieren des Interrupts die Ziffer Null als Ziel, d. h.: ON KEY 0. oder ON KEY ASCIIcode, 0
ON PS2-Ziel	Dies löst einen Interrupt aus, sobald die PicoMite-Firmware eine Nachricht von der PS2-Schnittstelle empfängt. Verwenden Sie MM.info(PS2), um die empfangene Rohmeldung zu melden. Auf diese Weise kann der Programmierer sowohl Tastendrücke als auch Tastenloslassen erfassen. Die Scancodes (Set 2) finden Sie unter https://wiki.osdev.org/PS/2_Keyboard.

ONEWIRE RESET-Pin oder ONEWIRE WRITE-Pin, Flag, Länge, Daten [, Daten…] oder ONEWIRE READ-Pin, Flag, Länge, Daten [, Daten…]	Befehle für die Kommunikation mit 1-Wire-Geräten. ONEWIRE RESET setzt den 1-Wire-Bus zurück. ONEWIRE WRITE sendet eine bestimmte Anzahl von Bytes. ONEWIRE READ liest eine bestimmte Anzahl von Bytes. „Pin" ist der zu verwendende I/O-Pin (befindet sich im hinteren Anschluss). Es kann jeder Pin sein, der für digitale I/O geeignet ist. „flag" ist eine Kombination aus den folgenden Optionen: 1 – Reset vor Befehl senden 2 – Reset nach Befehl senden 4 - Senden/Empfangen Sie nur ein Bit statt eines Bytes an Daten 8 – Rufen Sie nach dem Befehl einen starken Pullup auf (der Pin wird auf High gesetzt und Open Drain deaktiviert). „length" ist die Länge der zu sendenden oder zu empfangenden Daten „data" sind die zu sendenden Daten oder die zu empfangende Variable. Die Anzahl der Datenelemente muss mit dem Parameter „length" übereinstimmen. Siehe auch *Anhang C*.
OPEN fname$ FOR mode AS [#]fnbr	Öffnet eine Datei zum Lesen oder Schreiben. „fname" ist der Dateiname mit einer optionalen Erweiterung, die durch einen Punkt (.) getrennt ist. Lange Dateinamen mit Groß- und Kleinbuchstaben werden unterstützt. Das Dateisystem auf der SD-Karte unterscheidet NICHT zwischen Groß- und Kleinschreibung, das Flash-Dateisystem hingegen schon. Ein Verzeichnispfad kann mit einem Backslash als Verzeichnistrennzeichen angegeben werden. Das übergeordnete Verzeichnis des aktuellen Verzeichnisses kann mit dem Verzeichnisnamen „.." (zwei Punkte) und das aktuelle Verzeichnis mit „." (ein Punkt) angegeben werden. Beispiel: OPEN „.\dir1\dir2\filename.txt" FOR INPUT AS #1 „mode" ist INPUT, OUTPUT, APPEND oder RANDOM. INPUT öffnet die Datei zum Lesen und gibt eine Fehlermeldung aus, wenn die Datei nicht existiert. OUTPUT öffnet die Datei zum Schreiben und überschreibt automatisch alle vorhandenen Dateien mit demselben Namen. APPEND öffnet die Datei ebenfalls zum Schreiben, überschreibt jedoch keine vorhandene Datei, sondern hängt alle Schreibvorgänge an das Ende der Datei an. Wenn keine Datei vorhanden ist, verhält sich der Modus APPEND wie der Modus OUTPUT (d. h. die Datei wird erstellt und dann zum Schreiben geöffnet). RANDOM öffnet die Datei sowohl zum Lesen als auch zum Schreiben und ermöglicht den zufälligen Zugriff mit dem Befehl SEEK. Beim Öffnen wird der Lese-/Schreibzeiger am Ende der Datei positioniert. Wenn die Datei nicht existiert, wird sie erstellt. „fnbr" ist die Dateinummer (1 bis 10). Das # ist optional. Es können bis zu 10 Dateien gleichzeitig geöffnet sein, die sich entweder auf dem Laufwerk A: oder C: oder auf beiden Laufwerken befinden können. Die Befehle INPUT, LINE INPUT, PRINT, WRITE und CLOSE sowie die Funktionen EOF() und INPUT$() verwenden alle „fnbr", um die Datei zu identifizieren, auf die sie angewendet werden. Siehe auch ON ERROR und MM.ERRNO für die Fehlerbehandlung.
OPEN comspec$ AS [#]fnbr	Öffnet einen seriellen Kommunikationsport zum Lesen und Schreiben. Es stehen zwei Ports zur Verfügung (COM1: und COM2:), die beide gleichzeitig geöffnet werden können. Eine vollständige Beschreibung mit Beispielen finden Sie in *Anhang A*. Mit „fnbr" kann der Port mit jedem Befehl oder jeder Funktion, die eine Dateinummer verwendet, beschrieben und gelesen werden.
OPEN comspec$ AS GPS [,timezone_offset] [,monitor]	Öffnet einen seriellen Kommunikationsport zum Lesen von einem GPS-Empfänger. Weitere Informationen finden Sie unter der GPS-Funktion. Die interpretierten Sätze sind GPRMC, GNRMC, GPGGA und GNGGA.

	Der Parameter timezone_offset wird verwendet, um die vom GPS empfangene UTC-Zeit in die lokale Zeitzone umzuwandeln. Wenn er weggelassen wird, wird standardmäßig die UTC-Zeitzone verwendet. Der Parameter timezone_offset kann ein beliebiger Wert zwischen -12 und 14 sein, sodass die Zeit sogar für die Chatham-Inseln in Neuseeland (UTC +12:45) korrekt eingestellt werden kann. Wenn der Parameter „monitor" auf 1 gesetzt ist, werden alle GPS-Eingaben an die Konsole weitergeleitet. Dies kann durch Schließen des GPS-Kanals gestoppt werden.
OPTION	Siehe den Abschnitt *„Optionen"* weiter oben in diesem Handbuch.
PAUSE Verzögerung	Halten Sie die Ausführung des laufenden Programms für „delay" ms an. Dies kann ein Bruchteil sein. Beispielsweise entspricht 0,2 200 µs. Die maximale Verzögerung beträgt 2147483647 ms (etwa 24 Tage). Beachten Sie, dass Interrupts während einer Pause erkannt und verarbeitet werden.
PIN(Pin) = Wert	Bei einem als digitaler Ausgang konfigurierten „Pin" wird der Ausgang auf niedrig gesetzt („Wert" ist Null) oder auf High („Wert" ist ungleich Null). Sie können einen Ausgang auf High oder Low setzen, bevor er als Ausgang konfiguriert wird, und diese Einstellung ist dann die Standardausgabe, wenn der Befehl SETPIN wirksam wird. Siehe die Funktion PIN() zum Lesen von einem Pin und den Befehl SETPIN zum Konfigurieren. Eine allgemeine Beschreibung der Ein-/Ausgabefunktionen der PicoMite-Firmware finden Sie im Kapitel *„Verwendung der E/A-Pins*".
PIO	Der Prozessorchip im Raspberry Pi Pico mit den RP2040-Prozessoren enthält ein programmierbares E/A-System mit zwei identischen PIO-Geräten (pio%=0 oder pio%=1), die wie spezialisierte CPU-Kerne funktionieren. Der Raspberry Pi Pico 2 mit den RP2350-Prozessoren verfügt über drei PIO-Geräte. Eine detailliertere Beschreibung der Programmierung von PIOs finden Sie im Anhang F. Die PIO-Befehle werden als MMBasic-Befehle behandelt. Gültige Befehle sind: jmp (<cond>) <target> wait <Polarität> gpio <gpio_num> wait <Polarität> pin <pin_num> wait <Polarität> irq (vorherige | nächste) <irq_num> (rel) wait <Polarität> jmppin (+ <Pin-Offset>) in <Quelle>, <Bitanzahl> out <Ziel>, <Bitanzahl> push (iffull) push (iffull) block push (iffull) noblock pull (ifempty) ziehen (wenn leer) block ziehen (wenn leer) noblock mov <Ziel>, (op) <Quelle> irq (vorherige | nächste) <irq_num> (rel) irq (vorherige | nächste) set <irq_num> (rel) irq (vorherige | nächste) nowait <irq_num> (rel) irq (vorherige | nächste) wait <irq_num> (rel) irq (vorherige | nächste) clear <irq_num> (rel) set <Ziel>, <Wert> Nur RP2350 mov rxfifo[y], isr

PIO-Assembler pio,linedata$	mov rxfifo[<Index>], isr mov osr, rxfifo[y] NB mov osr, rxfifo[<index>] NB: Nur RP2350 Gültige Direktiven: .Programmname .line n/next .label name .wrap Ziel .wrap .Seite festlegen n .end program [Liste] PIO-Befehle werden zwischen einem Befehl „PIO ASSEMBLE n" und einer Anweisung „.end program" verwendet. Zwischen diesen Befehlen können auch andere Anweisungen stehen. Dieser Befehl assembliert und lädt textbasierten PIO-Assembler-Code einschließlich Sprungmarken. Verwendung: PIO assemble pio,".program anything", um den Assembler zu initialisieren. Verwendung: PIO assemble pio,".side_set n [opt] [pindirs]", wenn Side Set verwendet wird. Dies ist zwingend erforderlich, um die Op-Codes korrekt zu erstellen, wenn ein oder mehrere Side Set-Pins verwendet werden. Das pinctrl-Register wird nicht geladen, da dies spezifisch für die Zustandsmaschine ist. Beachten Sie auch, dass der Parameter „opt" den Op-Code bei Anweisungen mit einem Side-Parameter ändert. Verwendung: PIO assemble pio,".line n", um ab einer anderen Zeile als 1 zu assemblieren – dies ist optional. Verwendung: PIO assemble pio,".end program [list]", um die Assemblierung zu beenden und das PIO zu programmieren. Der optionale Parameter LIST bewirkt eine Hex-Ausgabe der Op-Codes auf dem Terminal. Verwendung: PIO assemble pio,"label:" zum Definieren einer Bezeichnung. Dies muss als separater Befehl erscheinen. Verwendung: PIO assemble „'.wrap target", um anzugeben, wohin das Programm umbrechen soll. Informationen zur Verwendung finden Sie unter PIO(.wrap target). Verwendung: PIO assemble „.wrap", um anzugeben, wo das Programm von „.wrap target" zurückkehren soll. Informationen zur Verwendung finden Sie unter PIO(.wrap). Verwendung: PIO assemble pio „instruction [parameters]", um die tatsächlichen PIO-Befehle zu definieren, die in Maschinencode umgewandelt werden sollen.
PIO DMA RX pio, sm, nbr, data%() [,completioninterrupt] [,transfersize] [,loopbackcount] PIO DMA TX pio, sm, nbr, data%() [,completioninterrupt] [,transfersize] [,loopbackcount]	Richtet DMA-Übertragungen von PIO zum MMBasic-Speicher ein. pio gibt an, welche der beiden pio-Instanzen verwendet werden soll (0 oder 1). sm gibt an, welche der Zustandsmaschinen verwendet werden soll (0-3). nbr gibt an, wie viele 32-Bit-Wörter übertragen werden sollen. Siehe unten für den Sonderfall, dass nbr auf Null gesetzt wird. data%() ist das Array, das die PIO-Daten entweder bereitstellt oder empfängt. Der optionale Parameter completioninterrupt ist der Name einer MMBasic-Subroutine r, die aufgerufen wird, wenn der DMA abgeschlossen ist und im Fall von DMA_OUT der FIFO geleert wurde. Wenn der optionale Interrupt nicht verwendet wird, kann der Status des DMA mit den folgenden Funktionen überprüft werden: MM.INFO(PIO RX DMA)

	MM.INFO(PIO TX DMA) Der optionale Parameter transfersize ermöglicht es dem Benutzer, die normalen 32-Bit-Übertragungen zu überschreiben und 8, 16 oder 32 auszuwählen. Der optionale Parameter loopbackcount gibt an, wie viele Datenelemente gelesen oder geschrieben werden sollen, bevor der DMA am Anfang des Puffers erneut startet. Der Parameter muss eine Potenz von 2 zwischen 2 und 32768 sein. Aufgrund einer Einschränkung im RP2040/RP2350 muss bei Verwendung von loopbackcount das MMBasic-Array im Speicher an die Anzahl der Bytes in der Schleife ausgerichtet werden. Wenn das Array also 64 Ganzzahlen lang ist, was 512 Bytes entspricht, muss das Array auf eine 512-Byte-Grenze im Speicher ausgerichtet werden. Alle MMBasic-Arrays werden an einer 256-Byte-Grenze ausgerichtet, aber um ein Array zu erstellen, das garantiert an einer 512-Byte-Grenze oder größer ausgerichtet ist, muss der PIO-Befehl MAKE RING BUFFER verwendet werden. Wenn „loopbackcount“ gesetzt ist, kann „nbr“ auf 0 gesetzt werden. In diesem Fall läuft die Übertragung kontinuierlich und füllt den Puffer wiederholt, bis sie explizit gestoppt wird. Wenn „nbr” und „loopbackcount” beide angegeben und identisch sind, startet der DMA nach Abschluss automatisch neu (nur TX).
	Bricht einen laufenden DMA ab.
PIO DMA RX OFF PIO DMA TX OFF	Legt grundlegende Interrupts für PIO-Aktivitäten fest. Verwenden Sie den Wert 0 für RXinterrupt oder TXinterrupt, um einen Interrupt zu deaktivieren. Lassen Sie nicht benötigte Werte weg.
PIO INTERRUPT pio, sm [,RXinterrupt] [,TXinterrupt]	Der RX-Interrupt wird ausgelöst, wenn ein Wort vom PIO-Code in den angegebenen FIFO „geschoben” wurde. Die Daten MÜSSEN im Interrupt gelesen werden, um ihn zu löschen. Der TX-Interrupt wird ausgelöst, wenn der angegebene FIFO VOLL ist und der PIO-Code ihn nun „gezogen” hat.
PIO INIT MACHINE pio%, Zustandsmaschine%, Taktfrequenz [,Pinsteuerung] [,Ausführungssteuerung] [,Verschiebungssteuerung] [,startinstruction] [,sideout [,setout] [,outout]	Initialisiert PIO „pio%“ mit der Zustandsmaschine „statemachine%“. „clockspeed“ ist die Taktrate der Zustandsmaschine in kHz. Die ersten vier optionalen Argumente sind Variablen, die die Initialisierungswerte der Zustandsmaschinenregister und die Adresse der ersten auszuführenden Anweisung (standardmäßig Null) enthalten. Diese bestimmen, wie die PIO funktioniert. sideout, setout und outout können auf 0 (Standard) oder 1 gesetzt werden, um anzugeben, ob die in pinctrl definierten Pins als Eingänge (0) oder Ausgänge (1) initialisiert werden sollen. Führt die Anweisung sofort auf dem angegebenen PIO und der angegebenen Zustandsmaschine aus.
PIO EXECUTE pio, Zustandsmaschine, Befehl%	Schreibt die Datenelemente in die angegebene PIO und Zustandsmaschine. Der Schreibvorgang ist blockierend, sodass die Zustandsmaschine in der Lage sein muss, die bereitgestellten Daten zu übernehmen. Hinweis: Dieser Befehl wird in zukünftigen Versionen wahrscheinlich zusätzliche Funktionen erfordern.
PIO WRITE pio, Zustandsmaschine, Zählung, Daten0 [,Daten1..]	Schreibt in eines der 4 einzelnen FIFO-Register. „a” ist der PIO (0 oder 1), „b” ist die Zustandsmaschine (0...3), „c” ist das FIFO-Register. *0…3), „d“ ist der Datenprozentsatz (32-Bit-Ganzzahlwert).
PIO WRITEFIFO a,b,c,d	Liest die Datenelemente aus dem angegebenen PIO und der angegebenen Zustandsmaschine. Der Lesevorgang ist nicht blockierend, daher muss die Zustandsmaschine in der Lage sein, die angeforderten Daten bereitzustellen. Wenn count gleich eins ist, kann eine Ganzzahl zum Empfangen der Daten verwendet werden, andernfalls sollte ein Ganzzahl-Array angegeben werden.
PIO READ pio, Zustandsmaschine, Zähler, Daten%[()]	Hinweis: Dieser Befehl wird in zukünftigen Versionen wahrscheinlich zusätzliche Funktionen benötigen.

PIO START pio, Zustandsmaschine PIO STOP pio, Zustandsmaschine PIO CLEAR pio PIO PROGRAM pio PIO PROGRAM pio,array%() PIO PROGRAM LINE pio, line, instruction	Startet eine bestimmte Zustandsmaschine auf pio. Stoppt eine bestimmte Zustandsmaschine auf pio. Dadurch wird das auf allen Zustandsmaschinen angegebene pio gestoppt und die Steuerregister und Interrupt-Flags für die Zustandsmaschinen PINCTRL, EXECTRL und SHIFTCTRL auf die Standardwerte zurückgesetzt. Gibt an, dass die nächsten Zeilen PIO-Assemblerbefehle für den PIO „pio" sind, bis eine „.end program"-Anweisung folgt. Programmier den gesamten pio-Programmspeicher mit den Daten in array%(). Siehe Anhang F. Programmier nur die angegebene Zeile in einem PIO-Programm.
PIO SYNC pio, Zustandsmaschinen, [,vorherige Zustandsmaschinen] [,nächste Zustandsmaschinen]	Synchronisiert die Uhren der PIO-Zustandsmaschine. „pio" gibt den Referenz-PIO für den Befehl an „statemachines", „prevstatemachines" und „nextstatemachines" sind Bitmaps (0 bis 15), die angeben, welche Zustandsmaschinenuhren synchronisiert werden sollen. Um also anzugeben, dass alle Zustandsmaschinen von PIO0 und PIO1 synchronisiert werden sollen, könnten Sie Folgendes verwenden: PIO SYNC 0,15,,15 oder PIO SYNC 1,15,15
PIO SET BASE 0/16	PIO-Befehle können nur mit 32 GPIO-Ports verwendet werden. Beim RP2350B weist dieser Befehl das System an, GP0-GP31 (0) oder GP16-GP47 (16) zu verwenden.
PIO CONFIGURE pio, sm, clock [,startaddress] [,sidesetbase] [,sidesetno] [,sidesetout] [,setbase] [,setno] [,setout] [,outbase] [,outno] [,outout] [,inbase] [,jmppin] [,wraptarget] [,wrap] [,sideenable] [,sidepindir] [,pushthreshold] [,pullthreshold] [,autopush] [,autopull] [,inshiftdir] [,outshiftdir] [,joinrxfifo] [,jointxfifo] [,joinrxfifoget] [,joinrxfifoput]	Die Parameter in diesem Befehl entsprechen im Wesentlichen denen, die Sie im Befehl PIO INIT verwenden würden, zuzüglich der Hilfsfunktionen PINCTRL, SHIFTCTRL und EXECCTRL, sind jedoch in einem einzigen Befehl zusammengefasst. Dies ist erforderlich, da das Pico SDK im Hintergrund einige sehr clevere Verarbeitungsschritte durchführt, um den RP2350B zu verarbeiten. „sidesetbase", „sidebase outbase", „inbase" und „jmppin" sind Pin-Definitionen. Sie können diese entweder als GPno oder als Pin-Nummer (z. B. GP3 oder 5) angeben. Geben Sie in allen Fällen den tatsächlichen Pin an. Wenn also PIO SET BASE für diesen PIO auf 16 gesetzt ist, sind die Werte GP16 bis GP47 gültig. Wenn PIO SET BASE nicht gesetzt oder auf 0 gesetzt ist, sind die Pins GP0 bis GP31 gültig. Sie sind alle standardmäßig auf den Basiswert gesetzt, mit Ausnahme von „jmppin" (Standardwert -1), das explizit gesetzt werden muss, wenn Sie ein „jmppin" verwenden möchten, da dies die Einstellung des erforderlichen Statusbits auslöst. „clock" ist die gewünschte PIO-Taktrate in Hz „startaddress" ist die PIO-Anweisung, die die Ausführung startet – Standardwert ist 0. „sidesetno", „setno" und „outno" geben die Anzahl der Pins an, die für diese Funktionen verwendet werden können – Standardwert ist 0. „sidesetout", „setout" und „outout" geben an, ob diese Pins als Ausgänge konfiguriert werden sollen (1=ja, 0=nein) – Standardwert ist 0 „wraptarget" und „wrap" liegen im Bereich von 0 bis 31 und sind standardmäßig auf 0 und 31 eingestellt. „inshiftdir" und „outshiftdir" sind standardmäßig auf 1 eingestellt – Verschiebung des Ausgangs-Schieberegisters nach rechts und Verschiebung des Eingangs-Schieberegisters nach rechts (Daten werden von links eingegeben). Alle anderen Parameter sind Boolesche Werte, die eine bestimmte Funktion aktivieren können – 1 zum Aktivieren, 0 zum Deaktivieren – alle sind standardmäßig auf 0 gesetzt. Einfaches Beispiel: *'PIO Konfigurieren Sie pio, sm, clock, startaddress,* *'sidesetbase, sidesetno, sidesetout,* *'setbase, setno, setout, outbase, outno, outout, inbase, 'jmppin,* *wraptarget, wrap, sideenable*, *sidepindir,*

	'pushthreshold, pullthreshold, autopush, autopull, inshiftdir, outshiftdir, 'joinrxfifo, jointxfifo, joinrxfifoget, joinrxfifoput *PIO-Assembler 1* *.Programmtest* *.Zeile 0* *.wrap Ziel* *Pins setzen,1* *Pins setzen,0* *.wrap* *.Ende Programm* *SetPin gp45,pio1 PIO* *Basis 1,16 einstellen* *PIO konfigurieren 1,0,1000000,,,,gp45,1,1,,,,,Pio(.wrap Ziel),Pio(.wrap) PIO starten 1,0* *Do-* *Schleife* Obwohl der Befehl PIO CONFIGURE viele Parameter hat, ist er sehr einfach zu verwenden, wenn Sie diesen einfachen Ansatz anwenden: Kopieren Sie die Kommentarzeilen im Beispiel in Ihr Programm. Ersetzen Sie für jeden Parameter den gewünschten Wert oder löschen Sie den Parameter, wobei Sie die Kommas unverändert lassen. Wenn Sie alle Ersetzungen vorgenommen haben, löschen Sie alle nachgestellten Kommas. Wenn Sie davon ausgehen, dass die Zeile für den Editor zu lang ist, löschen Sie die CRs nacheinander, beginnend am Ende der vorletzten Zeile und nach oben arbeitend. Auf diese Weise erhalten Sie einen gültigen Befehl, der leicht einzugeben und zu bearbeiten ist. Hinweis: Sie können auch Fortsetzungszeilen verwenden, um die Bearbeitung zu vereinfachen (siehe OPTION CONTINUATION LINES).
PIXEL x, y [,c]	Legen Sie einen Pixel auf einem Videoausgang oder einem angeschlossenen LCD-Bildschirm auf eine Farbe fest. „x“ ist die horizontale Koordinate und „y“ die vertikale Koordinate des Pixels. „c“ ist eine 24-Bit-Zahl, die die Farbe angibt. „c“ ist optional. Wird es weggelassen, wird die aktuelle Vordergrundfarbe verwendet. Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Pixel entsprechend den Abmessungen des kleinsten Arrays. „x” und „y” müssen entweder beide Arrays oder beide einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler ausgegeben. „c” kann entweder ein Array oder eine einzelne Variable oder Konstante sein. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *„Grafikbefehle und -funktionen*“.
PLAY PLAY TONE links, rechts [,dauer] [,unterbrechen]	Dieser Befehl erzeugt eine Vielzahl von Audioausgaben. Siehe den Befehl OPTION AUDIO zum Einstellen der für die Ausgabe zu verwendenden I/O-Pins. Das Audiosignal ist ein pulsweitenmoduliertes Signal (PWM), sodass ein Tiefpassfilter erforderlich ist, um die Trägerfrequenz zu entfernen. Erzeugt zwei separate Frequenzen auf dem linken und rechten Kanal der Audioausgabe. „left” und „right” sind die Frequenzen in Hz, die für den linken und rechten Kanal verwendet werden sollen. Der Ton wird im Hintergrund abgespielt (das Programm läuft nach diesem Befehl weiter) und „dur” gibt die Anzahl der Millisekunden an, für die der Ton erklingen soll. Wenn die Dauer nicht angegeben ist, wird der Ton so lange wiedergegeben, bis er explizit gestoppt wird oder das Programm beendet wird. „interrupt“ ist eine optionale Subroutine, die aufgerufen wird, wenn die Wiedergabe beendet wird. Die Frequenz kann zwischen 1 Hz und 20 kHz liegen und ist sehr genau (sie basiert auf einem Quarzoszillator). Die Frequenz kann jederzeit durch einen neuen PLAY TONE-Befehl geändert werden.

PLAY FLAC file$ [, interrupt]	Spielt eine FLAC-Datei über den Audioausgang ab. „file$" ist die abzuspielende FLAC-Datei (die Erweiterung .flac wird angehängt, falls sie fehlt). Die Abtastrate kann bis zu 48 kHz in Stereo betragen (96 kHz, wenn der Pico übertaktet ist). Die FLAC-Datei wird im Hintergrund abgespielt. „interrupt" ist optional und ist der Name einer Subroutine, die aufgerufen wird, wenn die Wiedergabe der Datei beendet ist. Wenn file$ ein Verzeichnis auf dem Laufwerk B: ist, spielt der Pico alle Dateien in diesem Verzeichnis nacheinander ab.
PLAY WAV file$ [, interrupt]	Spielt eine WAV-Datei über den Audioausgang ab. „file$" ist die abzuspielende WAV-Datei (die Erweiterung .wav wird angehängt, falls sie fehlt). Die WAV-Datei muss PCM-codiert in Mono oder Stereo mit 8 oder 16 Bit Abtastrate sein. Die Abtastrate kann bis zu 48 kHz in Stereo betragen (96 kHz, wenn der Pico übertaktet ist). Die WAV-Datei wird im Hintergrund abgespielt. 'interrupt' ist optional und ist der Name einer Subroutine, die aufgerufen wird, wenn die Datei fertig abgespielt ist.
PLAY ARRAY l%(), r%(), freq [,start] [,end] [,terminationinterrupt]	Gibt Daten über den Audioausgang aus, wobei Daten in einem oder zwei Arrays verwendet werden: Dazu sind gepackte Arrays „l%" und „r%" (können für links und rechts dasselbe Array sein) mit 16-Bit-Werten im Bereich von -32768 bis 32767 erforderlich, und die Werte im Array werden mit der angegebenen Abtastfrequenz ausgegeben. Wenn der optionale Parameter „start" angegeben ist, werden die Arrays ab der gepackten Position „start" ausgegeben. Wenn der optionale Parameter „end" angegeben ist, werden die Arrays bis zum Erreichen der gepackten Position „end" ausgegeben. Wenn der optionale Parameter „terminationinterrupt" angegeben ist, wird die angegebene Subroutine ausgeführt, sobald das letzte Array-Element ausgegeben wurde. Es ist wichtig, den Parameter „freq" zu verstehen. Dies ist die Rate, mit der jedes Array-Element ausgegeben wird. Wenn das Array beispielsweise 10 Zyklen einer Sinuswelle mit insgesamt 18000 Samples enthält, würde die Einstellung der Frequenz auf 1800 eine 1-Hz-Sinuswelle für 10 Sekunden ausgeben. Die Einstellung der Frequenz auf 18000 würde eine 10-Hz-Sinuswelle für 1 Sekunde ausgeben.
MP3-Datei abspielen$ [, Interrupt]	Spielt eine MP3-Datei über den Soundausgang ab (NUR RP2350). „file$" ist die abzuspielende MP3-Datei (die Erweiterung .mp3 wird angehängt, falls sie fehlt). Die Abtastrate kann bis zu 48 kHz betragen. Die MP3-Datei wird im Hintergrund abgespielt. „interrupt" ist optional und bezeichnet den Namen einer Subroutine, die aufgerufen wird, wenn die Datei vollständig abgespielt wurde. Wenn file$ ein Verzeichnis auf Laufwerk B: ist, spielt der Pico alle Dateien in diesem Verzeichnis nacheinander ab.
PLAY MODFILE file$ [,interrupt]	Spielt eine MOD-Datei über den Soundausgang ab. „file$" ist die abzuspielende MOD-Datei (die Erweiterung .mod wird angehängt, falls sie fehlt). Die MOD-Datei wird im Hintergrund abgespielt und läuft kontinuierlich in einer Schleife. Wenn die optionale Option „interrupt" angegeben ist, wird diese aufgerufen, wenn die Datei einmal durch die Sequenz abgespielt wurde, und die Wiedergabe wird dann beendet. Dieser Befehl nutzt vorzugsweise Speicherplatz im PSRAM, wenn er für den Dateipuffer aktiviert ist (nur RP2350). In diesem Fall muss ein Modbuffer nicht mit dem Befehl OPTION aktiviert werden.
PLAY MODSAMPLE Samplenum, Kanal [,Lautstärke]	Spielt ein bestimmtes Sample in der Mod-Datei auf dem angegebenen Kanal ab. Die Lautstärke ist optional und kann zwischen 0 und 64 liegen. Dieser Befehl kann nur verwendet werden, wenn

	bereits eine Mod-Datei abgespielt wird und ermöglicht die Ausgabe von Soundeffekten, während die Hintergrundmusik noch abgespielt wird.
PLAY LOAD SOUND array%()	Lädt ein Array mit 1024 Elementen, das aus 4096 16-Bit-Werten zwischen 0 und 4095 besteht. Dies liefert die Daten für jede beliebige Wellenform, die mit dem Befehl PLAY SOUND abgespielt werden kann. Mit dem Befehl MEMORY PACK können Sie die Arrays aus einem normalen Integer-Array mit 4096 Elementen erstellen.
PLAY SOUND soundno, channelno, type [,frequency] [,volume]	Spielen Sie eine Reihe von Sounds gleichzeitig über den Audioausgang ab. „soundno" ist die Soundnummer und kann zwischen 1 und 4 liegen, sodass vier Sounds gleichzeitig auf jedem Kanal abgespielt werden können. „channelno" gibt den Ausgangskanal an. Er kann L (linker Lautsprecher), R (rechter Lautsprecher) oder B (beide Lautsprecher) sein. „type" gibt die zu verwendende Wellenform an. Sie kann S (Sinuswelle), Q (Rechteckwelle), T (Dreieckwelle), W (Sägezahnwelle), O (Null-Ausgabe), P (periodisches Rauschen), N (zufälliges Rauschen) oder U (benutzerdefiniert mit PLAY LOAD SOUND) sein. „frequency" ist die Frequenz von 1 bis 20000 (Hz) und muss angegeben werden, außer wenn „type" O ist. Im Modus „Type U" kann dieser Parameter auch Dezimalwerte annehmen. Beispielsweise geben alle folgenden Angaben die Wellenform in ihrer ursprünglichen Tonhöhe wieder: Abtastrate von 4000, Frequenz = 1 Abtastrate von 8000, Frequenz = 2 Abtastrate von 16000, Frequenz = 4 „volume" ist optional und muss zwischen 1 und 25 liegen. Der Standardwert ist 25. Wenn PLAY SOUND aufgerufen wird, werden alle anderen Audiofunktionen blockiert und bleiben blockiert, bis PLAY STOP aufgerufen wird. Die Ausgabe kann mit PLAY PAUSE und PLAY RESUME vorübergehend angehalten werden. Der Aufruf von SOUND bei einem bereits laufenden „soundno" ersetzt sofort die vorherige Ausgabe. Einzelne Sounds werden mit dem Typ „O" ausgeschaltet. Die gleichzeitige Wiedergabe von 4 Sounds auf beiden Kanälen der Audioausgabe verbraucht etwa 23 % der CPU-Leistung.
PLAY PAUSE PLAY RESUME PLAY STOP	PLAY PAUSE hält die aktuell wiedergegebene Datei oder den Ton vorübergehend an. PLAY RESUME setzt die Wiedergabe eines angehaltenen Sounds fort. PLAY STOP beendet die Wiedergabe der Datei oder des Tons. Wenn das Programm aus irgendeinem Grund beendet wird, wird auch die Tonausgabe automatisch gestoppt.
PLAY VOLUME links, rechts	Stellt die Lautstärke der Audioausgabe ein. „links" und „rechts" sind die Pegel für den linken und rechten Kanal und können zwischen 0 und 100 liegen, wobei 100 die maximale Lautstärke ist. Es besteht ein linearer Zusammenhang zwischen dem angegebenen Pegel und der Ausgabe. Die Lautstärke ist standardmäßig auf Maximum eingestellt, wenn ein Programm ausgeführt wird.
NÄCHSTES ABSPIELEN	Beendet die Wiedergabe der aktuellen Audiodatei und startet die nächste Datei im Verzeichnis.
VORHERIGE WIEDERGEBEN	Beendet die Wiedergabe der aktuellen Audiodatei und startet die vorherige Datei im Verzeichnis.
	VS1053-spezifische PLAY-Befehle
MP3-Datei abspielen$ [, Unterbrechung]	Spielt eine MP3-Datei über den Audioausgang ab. „file$" ist die abzuspielende MP3-Datei (die Erweiterung .mp3 wird angehängt, falls sie fehlt). Die Abtastrate sollte 44100 Hz Stereo betragen.

	Die MP3-Datei wird im Hintergrund abgespielt. „interrupt" ist optional und ist der Name einer Subroutine, die aufgerufen wird, wenn die Wiedergabe der Datei beendet ist. Wenn file$ ein Verzeichnis auf dem Laufwerk B: ist, spielt der Pico alle Dateien in diesem Verzeichnis nacheinander ab.
PLAY HALT	Dieser Befehl funktioniert, wenn eine MP3-Datei abgespielt wird. Er stoppt die Wiedergabe und speichert die aktuelle Dateiposition, damit die Wiedergabe an derselben Stelle fortgesetzt werden kann. Dieser Befehl wurde speziell für die Unterstützung von MP3-Hörbüchern entwickelt.
PLAY CONTINUE track	Setzt die Wiedergabe des angegebenen MP3-Titels fort. „track$" ist der Name der Datei, die bei der Unterbrechung abgespielt wurde, wobei alle Dateiattribute entfernt wurden. z. B. PLAY MP3 „B:/mp3/mymp3.mp3" etwas später PLAY HALT später erneut PLAY CONTINUE „mymp3"
MIDIFILE-Datei abspielen file$ [, unterbrechen]	Spielt eine MIDI-Datei über den Soundausgang ab. „file$" ist die abzuspielende MIDI-Datei (die Erweiterung .mid wird angehängt, falls sie fehlt). Die MIDI-Datei wird im Hintergrund abgespielt. „interrupt" ist optional und ist der Name einer Subroutine, die aufgerufen wird, wenn die Datei vollständig abgespielt wurde. Wenn file$ ein Verzeichnis auf Laufwerk B: ist, spielt der Pico alle Dateien in diesem Verzeichnis nacheinander ab.
MIDI abspielen	Startet den Echtzeit-MIDI-Modus. In diesem Modus können MIDI-Befehle an den VS1053 gesendet werden, um auszuwählen, welche Instrumente auf welchen Kanälen gespielt werden sollen, Noten einzuschalten und sie in Echtzeit auszuschalten.
PLAY MIDI CMD cmd%, data1%, data2%	Sendet einen MIDI-Befehl im Echtzeit-MIDI-Modus. Ein Beispiel wäre die Zuweisung eines Instruments zu einem Kanal. Z. B. PLAY MIDI CMD &B11000001,4 „Kanal 1 auf Instrument 4 einstellen
PLAY MIDI TEST n	Spielt eine MIDI-Testsequenz ab, n=0 bis 3, 0 = normale Echtzeit, die anderen spielen Noten- und Instrumentensamples ab
PLAY NOTE ON Kanal%, Note%, Anschlagstärke%	Schaltet die Note auf dem angegebenen Kanal im Echtzeit-MIDI-Modus ein
PLAY NOTE OFF Kanal%, Note% [,Velocity%]	Schaltet die Note auf dem angegebenen Kanal im Echtzeit-MIDI-Modus aus
PLAY STREAM buffer%(), readpointer%, writepointer%	Sendet Daten aus dem Ringpuffer „buffer%" an den VS1053-CODEC. Dieser Befehl initiiert einen Hintergrund-Ausgabestrom, bei dem alles, was sich im Puffer zwischen dem Lesezeiger und dem Schreibzeiger befindet, an den VS1053 gesendet wird, wobei der Lesezeiger fortlaufend aktualisiert wird. Kann für die Ausgabe beliebiger Wellenformen verwendet werden.

POKE BYTE addr%, byte oder POKE SHORT addr%, short% oder POKE WORD addr%, word% oder POKE INTEGER addr%, int% oder POKE FLOAT addr%, float! oder POKE VAR var, offset, byte oder POKE VARTBL, offset, byte	Setzt ein Byte oder ein Wort im Speicherbereich des Pico. Wenn mehr als ein Byte geschrieben wird, muss die Adresse genau durch die Anzahl der Bytes teilbar sein: 2, 4 oder 8, andernfalls wird ein Fehler gemeldet. POKE BYTE setzt das Byte (d. h. 8 Bit) an der Speicherstelle „addr%" auf „byte". „addr%" sollte eine ganze Zahl sein. POKE SHORT setzt die kurze Ganzzahl (d. h. 16 Bit) an der Speicherstelle „addr%" auf „word%". „addr%" und „short%" sollten ganze Zahlen sein. POKE WORD setzt das Wort (d. h. 32 Bit) an der Speicherstelle „addr%" auf „word%". „addr%" und „word%" sollten Ganzzahlen sein. POKE INTEGER setzt die MMBasic-Ganzzahl (d. h. 64 Bit) an der Speicherstelle „addr%" auf „int%". „addr%" und „int%" sollten Ganzzahlen sein. POKE FLOAT setzt das Wort (d. h. 64 Bit) an der Speicherstelle „addr%" auf „float!". „addr%" sollte eine ganze Zahl und „float!" eine Gleitkommazahl sein. POKE VAR setzt ein Byte an der Speicheradresse von 'var'. 'offset' ist der ±Offset von der Adresse der Variablen. Ein Array wird als var() angegeben. POKE VARTBL setzt ein Byte in der Variablentabelle von MMBasic. „offset" ist der ±Offset vom Anfang der Variablentabelle. Beachten Sie, dass nach dem Schlüsselwort VARTBL ein Komma erforderlich ist.
POKE DISPLAY-Befehl [,Daten1] [,Daten2] [,Daten3]	Dieser Befehl sendet Befehle und zugehörige Daten an den Display-Controller eines angeschlossenen Displays. Dadurch kann der Programmierer Parameter für die Konfiguration des Displays ändern. Beispielsweise schaltet POKE DISPLAY &H28 ein SSD1963-Display aus und POKE DISPLAY &H29 schaltet es wieder ein. Funktioniert für alle Displays außer dem ST7790.
POKE DISPLAY HRES n POKE DISPLAY VRES n	Diese Befehle ändern den gespeicherten Wert von MM.HRES und MM.VRES, sodass der Programmierer nicht standardmäßige Displays konfigurieren kann.
POLYGON n, xarray%(), yarray%() [, bordercolour] [, fillcolour] POLYGON n(), xarray%(), yarray%() [, bordercolour()] [, fillcolour()] POLYGON n(), xarray%(), yarray%() [, bordercolour] [, fillcolour]	Zeichnet ein gefülltes oder umrandetes Polygon mit „n" xy-Koordinatenpaaren in „xarray%()" und „yarray%()". Wenn „fillcolour" weggelassen wird, wird nur die Polygonumrandung gezeichnet. Wenn „bordercolour" weggelassen wird, wird standardmäßig die aktuelle Vordergrundfarbe verwendet. Wenn das letzte xy-Koordinatenpaar nicht mit dem ersten übereinstimmt, erstellt die Firmware automatisch ein zusätzliches xy-Koordinatenpaar, um das Polygon zu vervollständigen. Die Größe der Arrays sollte mindestens so groß sein wie die Anzahl der x,y-Koordinatenpaare. „n" kann ein Array sein, und die Farben können optional auch Arrays sein, wie folgt: POLYGON n(), xarray%(), yarray%() [, bordercolour()] [, fillcolour()] POLYGON n(), xarray%(), yarray%() [, bordercolour] [, fillcolour] Die Elemente von „array n()" definieren die Anzahl der xy-Koordinatenpaare in jedem der Polygone. Beispielsweise würde DIM n(1)=(3,3) definieren, dass zwei Polygone mit jeweils drei Eckpunkten gezeichnet werden sollen. Die Größe des n-Arrays bestimmt die Anzahl der zu zeichnenden Polygone, es sei denn, es wird ein Element mit dem Wert Null gefunden. In diesem Fall verarbeitet die Firmware nur Polygone bis zu diesem Punkt. Die x-, y-Koordinatenpaare für alle Polygone werden in „xarray%()" und „yarray%()" gespeichert. Die Parameter „xarray%()" und „yarray%()" müssen mindestens so viele Elemente enthalten wie die Gesamtzahl der Werte im Array n. Jedes Polygon kann geschlossen werden, wenn das erste und das letzte Element identisch sind. Wenn das letzte Element nicht mit dem ersten übereinstimmt, erstellt die Firmware automatisch ein zusätzliches x,y-Koordinatenpaar, um das Polygon zu vervollständigen. Wenn die Füllfarbe weggelassen wird, werden nur die Polygonkonturen gezeichnet. Die Farbparameter können ein einzelner Wert sein, wobei in diesem Fall alle Polygone in derselben Farbe gezeichnet werden, oder sie können Arrays mit derselben Kardinalität wie „n" sein.

	In diesem Fall kann jedes gezeichnete Polygon eine andere Farbe für den Rand und/oder die Füllung haben.
	Beispielsweise werden dadurch drei Dreiecke in Gelb, Grün und Rot gezeichnet:
	`DIM c%(2)=(3,3,3)`
	`DIM x%(8)=(100,50,150,100,50,150,100,50,150) DIM y%(8)=(50,100,100,150,200,200,250,300,300)`
	`DIM fc%(2)=(rgb(gelb),rgb(grün),rgb(rot)) POLYGON c%(),x%(),y%(),fc%(),fc%()`

PORT(start, nbr [,start, nbr]…) = Wert	Legt mehrere E/A-Pins gleichzeitig fest (d. h. mit einem Befehl). „start“ ist eine I/O-Pin-Nummer, und das niedrigste Bit in „value“ (Bit 0) wird zum Einstellen dieses Pins verwendet. Bit 1 wird zum Setzen des Pins „start“ plus 1 verwendet, Bit 2 setzt den Pin „start“+2 und so weiter für die Anzahl der Bits „nbr“. Die verwendeten I/O-Pins müssen fortlaufend nummeriert sein, und jeder I/O-Pin, der ungültig oder nicht als Ausgang konfiguriert ist, verursacht einen Fehler. Das Paar start/nbr kann wiederholt werden, wenn eine zusätzliche Gruppe von Ausgangspins hinzugefügt werden muss. Beispiel: `PORT(15, 4, 23, 4) = &B10000011` Setzt acht I/O-Pins. Die Pins 15 und 16 werden auf High gesetzt, während 17, 18, 23, 24 und 25 auf Low gesetzt werden und schließlich 26 auf High gesetzt wird. Dieser Befehl kann verwendet werden, um bequem mit parallelen Geräten wie LCD-Displays zu kommunizieren. Es kann eine beliebige Anzahl von I/O-Pins (und damit Bits) von 1 bis zur Anzahl der I/O-Pins auf dem Chip verwendet werden. Hinweis: Wenn die Pins mit der GPn-Syntax definiert werden, ignoriert die Firmware ungültige Pins, sodass `PORT(GP0, 8) = &B10000011` acht I/O-Pins setzt. Die Pins GP0, GP1 und GP7 werden auf High gesetzt, während GP2, GP3, GP4, GP5 und GP6 auf Low gesetzt werden. Siehe die Funktion PORT, um gleichzeitig von mehreren Pins zu lesen.
PRINT Ausdruck [[,;]Ausdruck] … usw.	Gibt Text an die serielle Konsole aus, gefolgt von einem Zeilenumbruch/Zeilenvorschub-Paar. Es können mehrere Ausdrücke verwendet werden, die durch eines der folgenden Zeichen getrennt werden müssen: • Komma (,) getrennt werden, wodurch das Tabulatorzeichen ausgegeben wird • Semikolon (;), das nichts ausgibt (es dient nur zur Trennung von Ausdrücken). • Nichts oder ein Leerzeichen, das sich wie ein Semikolon verhält. Ein Semikolon (;) oder ein Komma (,) am Ende der Ausdrucksliste unterdrückt die Ausgabe des Zeilenumbruch-/Zeilenvorschub-Paares am Ende einer Druckanweisung. Bei der Ausgabe wird einer positiven Zahl ein Leerzeichen vorangestellt, einer negativen Zahl ein Minuszeichen (-), jedoch folgt kein Leerzeichen. Ganzzahlen werden ohne Dezimalpunkt ausgegeben, während Bruchzahlen mit Dezimalpunkt und den signifikanten Dezimalstellen ausgegeben werden. Große oder kleine Gleitkommazahlen werden automatisch im wissenschaftlichen Zahlenformat ausgegeben. Die Funktion TAB() kann verwendet werden, um einen Abstand zu einer bestimmten Spalte zu schaffen, und die Funktion STR$() kann verwendet werden, um Zeichenfolgen auszurichten oder anderweitig zu formatieren.
PRINT #nbr, Ausdruck [[,;]Ausdruck] … usw.	Wie oben, außer dass die Ausgabe an einen seriellen Kommunikationsport oder eine Datei geleitet wird, die für OUTPUT oder APPEND mit der Dateinummer „nbr” geöffnet wurde. Siehe den Befehl OPEN.
PRINT #GPS, Ausdruck [[,;]Ausdruck] … usw.	Gibt eine NMEA-Zeichenkette an ein geöffnetes GPS-Gerät aus. Die Zeichenkette muss mit einem $ beginnen und mit einem * enden. Die Prüfsumme wird automatisch berechnet und zusammen mit den CR/LF-Zeichen an die Zeichenfolge angehängt.

PRINT @(x [, y]) Ausdruck oder PRINT @(x, [y], m) Ausdruck	Funktioniert auf der Terminal-Konsole eines angeschlossenen Computers oder auf einem VGA/HDMI-Bildschirm oder dem Display, wenn die OPTION LCDPANEL CONSOLE aktiviert ist. Entspricht dem Standardbefehl PRINT, außer dass der Cursor an den Koordinaten x, y positioniert wird, die in Pixeln angegeben sind. Wenn y weggelassen wird, wird der Cursor an der Position „x" in der aktuellen Zeile positioniert. Beispiel: PRINT @(150, 45) „Hello World" Die @-Funktion kann an jeder Stelle eines Druckbefehls verwendet werden. Beispiel: PRINT @(150, 45) „Hello" @(150, 55) „World" Mit der Funktion @(x,y) kann der Cursor an einer beliebigen Stelle auf oder außerhalb des Bildschirms positioniert werden. Beispielsweise wird bei PRINT @(-10, 0) „Hello" nur „llo" angezeigt, da die ersten beiden Zeichen nicht angezeigt werden können, weil sie sich außerhalb des Bildschirms befinden. Die Funktion @(x,y) unterdrückt automatisch den automatischen Zeilenumbruch, der normalerweise erfolgt, wenn der Cursor über den rechten Bildschirmrand hinausgeht. Wenn „m" angegeben ist, ist der Modus der Videofunktion wie folgt: m = 0 Normaler Text (weiße Buchstaben, schwarzer Hintergrund) m = 1 Der Hintergrund wird nicht gezeichnet (d. h. transparent). m = 2 Das Video wird invertiert (schwarze Buchstaben, weißer Hintergrund). m = 5 Die aktuellen Pixel werden invertiert (transparenter Hintergrund).
PULSE-Pin, Breite	Erzeugt einen Impuls auf „Pin" mit einer Dauer von „Breite" ms. „Breite" kann ein Bruchteil sein. Beispielsweise entspricht 0,01 10 µs, wodurch sehr schmale Impulse erzeugt werden können. Der erzeugte Impuls hat die entgegengesetzte Polarität zum Zustand des I/O-Pins, wenn der Befehl ausgeführt wird. Wenn beispielsweise der Ausgang auf „high" gesetzt ist, erzeugt der Befehl PULSE einen negativen Impuls. Hinweise: • „Pin" muss als Ausgang konfiguriert sein. • Bei einem Impuls von weniger als 3 ms beträgt die Genauigkeit ± 1 µs. • Bei einem Impuls von 3 ms oder mehr beträgt die Genauigkeit ± 0,5 ms. • Ein Impuls von 3 ms oder mehr läuft im Hintergrund. Bis zu fünf verschiedene Impulse können gleichzeitig im Hintergrund laufen, und jeder Impuls kann durch einen neuen PULSE-Befehl zeitlich verändert oder durch einen PULSE-Befehl mit dem Wert Null für „width" beendet werden.
PWM-Kanal, Frequenz, [dutyA] [,dutyB][,phase][,defer]	Es stehen 8 separate PWM-Frequenzen (Kanäle 0 bis 7) und bis zu 16 Ausgänge mit individuell gesteuerter Einschaltdauer zur Verfügung. Sie können für jeden Kanal entweder auf PWMnA oder PWMnB oder auf beiden ausgeben – ohne Einschränkung. Die Einschaltdauer wird als Prozentsatz angegeben, und Sie können einen negativen Wert verwenden, um die Ausgabe umzukehren (-100,0 <= duty <=100,0). Um nur Kanal B zu verwenden, verwenden Sie die Syntax: „PWM-Kanal, Frequenz, , dutyB" – beachten Sie das doppelte Komma vor dem gewünschten Arbeitszyklus. Minimale Frequenz = (cpuspeed + 1) / (2^24) Hz. Die maximale Geschwindigkeit beträgt OPTION CPUSPEED/4. Bei sehr hohen Geschwindigkeiten werden die Arbeitszyklen zunehmend eingeschränkt. Phase ist ein Parameter, der bewirkt, dass die Wellenformen so zentriert werden, dass eine Wellenform mit einem kürzeren Arbeitszyklus zu gleichen Zeiten wie eine längere beginnt und endet. Verwenden Sie 1, um diesen Modus zu aktivieren, und 0 (oder lassen Sie ihn weg), um normal zu arbeiten. Der Parameter „deferredstart" konfiguriert bei Einstellung auf 1 die PWM-Kanäle, startet jedoch nicht die Ausgabe. Sie können dann mit dem Befehl PWM SYNC gestartet werden. Dies kann verwendet werden, um unerwünschte Startartefakte zu vermeiden. Der PWM-Befehl kann auch Servos wie folgt ansteuern: `PWM 1,50,(Position_als_Prozentsatz * 0,05 + 5)`

PWM SYNC s0 [,s1][,s2][,s3][,s4][,s5][,s6][,s7]	Dies initiiert die PWM auf Kanälen, für die ein verzögerter Start definiert wurde, oder synchronisiert einfach bereits laufende Kanäle. Die Stärke liegt jedoch in der Möglichkeit, die Kanäle gegeneinander zu versetzen (definiert als Prozentsatz der Zeitdauer gemäß dem Arbeitszyklus – kann ein Float sein). Sie können einen Versatz von -1 verwenden, um einen Kanal aus der Synchronisierung auszuschließen.
PWM-Kanal, AUS	Stoppen Sie die Ausgabe auf „Kanal”.
RAM RAM-LISTE RAM-LISTE n [,alle] RAM-LÖSCHEN n RAM-ALLES- LÖSCHEN RAM- SPEICHERN n RAM- LADEN n RAM- AUSFÜHREN n RAM-KETTE n RAM ÜBERSCHREIBEN n RAM-DATEI LADEN n, fname$ [,ÜBERSCHREIBEN]]	RP2350 nur mit aktiviertem PSRAM Der RAM-Befehl ermöglicht den Zugriff auf bis zu 5 RAM-Programmslots (ähnlich wie Flash-Slots). RAM-Slots überstehen einen Hardware- und Software-Reset, jedoch keinen Neustart. Zeigt eine Liste aller RAM-Speicherplätze einschließlich der ersten Zeile des Programms an. Listet das auf Speicherplatz n gespeicherte Programm auf. Verwenden Sie ALL, um ohne Seitenumbrüche aufzulisten. Löschen Sie einen RAM- Programmspeicherplatz. Löschen Sie alle RAM-Programmspeicherplätze. Speichern Sie das aktuelle Programm an der angegebenen RAM-Speicherstelle. Laden Sie ein Programm aus dem angegebenen RAM-Speicherplatz in den Programmspeicher. Führt das Programm an Speicherplatz n aus, löscht alle Variablen. Ändert den Programmspeicher nicht. Führt das Programm im RAM-Speicherplatz n aus und lässt alle Variablen unverändert (ermöglicht ein Programm, das viel größer ist als der Programmspeicher). Ändert den Programmspeicher nicht. Hinweis: Wenn das verkettete Programm den Befehl READ verwendet, muss es vor dem ersten Lesevorgang RESTORE aufrufen. Löschen Sie einen RAM-Programmspeicherplatz und speichern Sie dann das aktuelle Programm an dem angegebenen RAM-Speicherplatz. Lädt die MMBasic-Datei fname$ in den angegebenen RAM-Speicherplatz. Wenn der optionale Parameter OVERWRITE (oder O) angegeben ist, wird der Inhalt des Flash-Speicherplatzes ohne Fehlermeldung überschrieben.
RANDOMIZE nbr	Sät den Zufallszahlengenerator mit „nbr”. Beim RP2040 wird der Zufallszahlengenerator beim Einschalten mit Null initialisiert und erzeugt jedes Mal die gleiche Folge von Zufallszahlen. Um jedes Mal eine andere Zufallsfolge zu erzeugen, müssen Sie einen anderen Wert für „nbr” verwenden (die TIMER-Funktion ist dafür sehr praktisch). Dieser Befehl hat keine Wirkung auf den RP2350, der über einen Hardware-Zufallsgenerator verfügt, der keine Initialisierung erfordert.
REDIM [PRESERVE] array1(Dimensionen) [, array2(Dimensionen]…[,arrayn(Dimensionen]	Damit wird die Größe der angegebenen Arrays mit den angegebenen Dimensionen geändert. Wenn der optionale Unterbefehl PRESERVE angegeben ist, werden die vorhandenen Daten in das neue Array kopiert. Das neue Array kann größer oder kleiner als das ursprüngliche sein. Bei Zeichenfolgen-Arrays bleibt die ursprünglich angegebene LÄNGE erhalten. Beachten Sie, dass bei mehrdimensionalen Arrays nur die letzte Dimension geändert werden kann, wenn PRESERVE verwendet wird. Bei Verwendung von PRESERVE muss genügend Speicherplatz vorhanden sein, damit sowohl das ursprüngliche Array als auch seine geänderte Version gleichzeitig existieren können. Der dem ursprünglichen Array zugewiesene Speicher wird beim Beenden des Befehls freigegeben.

RBOX x, y, w, h [, r] [,c] [,fill]	Zeichnet ein Rechteck mit abgerundeten Ecken auf dem Videoausgang oder dem angeschlossenen LCD-Bildschirm, beginnend bei „x" und „y", mit einer Breite von „w" Pixeln und einer Höhe von „h" Pixeln. „r" ist der Radius der Ecken des Kastens. Der Standardwert ist 10. „c" gibt die Farbe an und ist standardmäßig auf die Standard-Vordergrundfarbe eingestellt, wenn nichts anderes angegeben ist. „fill" ist die Füllfarbe. Sie kann weggelassen oder auf -1 gesetzt werden. In diesem Fall wird das Feld nicht gefüllt. Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Kästchen entsprechend den Abmessungen des kleinsten Arrays. „x", „y", „w" und „h" müssen entweder alle Arrays oder alle einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler ausgegeben. „r", „c" und „fill" können entweder Arrays oder einzelne Variablen/Konstanten sein. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *„Grafikbefehle und -funktionen*".
READ variable[, variable]..	Liest Werte aus DATA-Anweisungen und weist diese Werte den benannten Variablen zu. Die Variablentypen in einer READ-Anweisung müssen mit den Datentypen in den DATA-Anweisungen übereinstimmen, wenn sie gelesen werden. Arrays können als Variablen verwendet werden (angegeben mit leeren Klammern, z. B. a()) und in diesem Fall wird die Größe des Arrays verwendet, um zu bestimmen, wie viele Elemente gelesen werden sollen. Wenn das Array mehrdimensional ist, wird die Dimension ganz links am schnellsten verschoben. Siehe auch DATA und RESTORE.
READ SAVE oder READ RESTORE	READ SAVE speichert den vom Befehl READ verwendeten virtuellen Zeiger, um auf die nächsten zu lesenden DATEN zu zeigen. READ RESTORE stellt den zuvor gespeicherten Zeiger wieder her. Dadurch können Unterprogramme Daten lesen und anschließend den Lesepointer wiederherstellen, um andere Teile des Programms, die möglicherweise dieselben Datenanweisungen lesen, nicht zu stören. Diese Befehle können verschachtelt werden.
REFRESH	Initiiert eine Aktualisierung des Bildschirms für E-Ink-Schwarzweiß-Displays. Diese können nur bildschirmweise aktualisiert werden. Wenn OPTION AUTOREFRESH auf OFF gesetzt ist, kann dieser Befehl zum Auslösen des Schreibvorgangs verwendet werden. Dieser Befehl funktioniert mit den folgenden Displays: N5110, SSD1306I2C, SSD1306I2C32, SSD1306SPI, ST7920.
REM-Zeichenfolge	REM ermöglicht das Einfügen von Anmerkungen in ein Programm. Beachten Sie, dass die Verwendung von einfachen Anführungszeichen (') im Microsoft-Stil zur Kennzeichnung von Anmerkungen ebenfalls unterstützt wird und bevorzugt wird.
RENAME old$ AS new$	Benennen Sie eine Datei oder ein Verzeichnis von „old$" in „new$" um. Beide sind Zeichenfolgen. Sowohl in „old$" als auch in „new$" kann ein Verzeichnispfad verwendet werden. Wenn sich die Pfade unterscheiden, wird die in „old$" angegebene Datei mit dem angegebenen Dateinamen in den in „new$" angegebenen Pfad verschoben.
RESTORE [Zeile]	Setzt die Zeilen- und Positionszähler für die READ-Anweisung zurück. Wenn „Zeile" angegeben ist, werden die Zähler auf den Anfang der angegebenen Zeile zurückgesetzt. „Zeile" kann eine Zeilennummer, eine Bezeichnung oder eine Variable mit diesen Werten sein. Wenn „line" nicht angegeben ist, werden die Zähler auf den Beginn des Programms zurückgesetzt.
RMDIR dir$	Entfernen oder löschen Sie das Verzeichnis „dir$" auf dem Standard-Flash-Dateisystem oder der SD-Karte.
RTC GETTIME	RTC GETTIME ruft das aktuelle Datum/die aktuelle Uhrzeit von einer Echtzeituhr PCF8563, DS1307 oder DS3231 ab und stellt die interne MMBasic-Uhr entsprechend ein. Das Datum/die Uhrzeit kann dann mit den Funktionen DATE$ und TIME$ abgerufen werden.

RTC SETTIME Jahr, Monat, Tag, Stunde, Minute, Sekunde RTC SETREG reg, Wert RTC GETREG reg, var	RTC SETTIME stellt die Zeit im Uhrenchip ein. „Stunde" muss im 24-Stunden-Format angegeben werden. „Jahr" kann zwei- oder vierstellig sein. Der Befehl RTC SETTIME akzeptiert auch ein einzelnes String-Argument im Format *TT/MM/JJ HH:MM*. Das bedeutet, dass das Datum/die Uhrzeit vom Benutzer über eine GUI-FORMATBOX mit dem Format DATETIME2 eingegeben werden kann (siehe *Advanced Graphics Functions.pdf*). Die Befehle RTC SETREG und GETREG können verwendet werden, um den Inhalt von Registern innerhalb des Echtzeituhr-Chips zu setzen oder zu lesen. „reg" ist die Nummer des Registers, „value" ist die Zahl, die im Register gespeichert werden soll, und „var" ist eine Variable, die die aus dem Register gelesene Zahl empfängt. Diese Befehle sind für den normalen Betrieb nicht erforderlich, können jedoch zur Steuerung spezieller Funktionen des Chips (Alarme, Ausgangssignale usw.) verwendet werden. Sie sind auch nützlich, um temporäre Informationen im batteriegepufferten RAM des Chips zu speichern. Diese Chips sind I²C-Geräte und müssen gemäß OPTION SYSTEM I2C mit geeigneten Pullup-Widerständen an die beiden I²C-Pins angeschlossen werden. Siehe auch den Befehl OPTION RTC AUTO ENABLE.
RUN oder RUN [Datei$] [, Befehlszeile$]	Führt ein Programm aus. Wenn „file$" nicht angegeben ist, wird das derzeit im Programmspeicher befindliche Programm ausgeführt. Wenn „file$" angegeben ist, wird die genannte Datei aus dem Flash- oder SD-Karten-Dateisystem ausgeführt. Wenn „file$" keine Erweiterung „.BAS" enthält, wird diese automatisch hinzugefügt. Wenn „cmdline$" angegeben ist, übergeben Sie dessen Wert an die Konstante MM.CMDLINE$ des Programms, wenn es ausgeführt wird. Wenn „cmdline$" nicht angegeben ist, wird ein leerer String-Wert an MM.CMDLINE$ übergeben. Hinweise: • Sowohl „file$" als auch „cmdline$" können als Zeichenfolgenausdrücke angegeben werden. • Verwenden Sie FLASH RUN n, um ein in einem Flash-Slot gespeichertes Programm auszuführen.
SAVE file	Speichert das Programm im aktuellen Arbeitsverzeichnis des Flash-Dateisystems oder auf der SD-Karte als „file$". Beispiel: SAVE „TEST.BAS" Wenn keine Erweiterung angegeben ist, wird „.BAS" an den Dateinamen angehängt. Siehe auch FLASH SAVE *n* zum Speichern in einem Flash-Speicherplatz.
SAVE CONTEXT [CLEAR]	Speichert den Variablenbereich und löscht ihn optional – der Befehl sollte im Top-Level-Programm und nicht innerhalb einer Subroutine verwendet werden. Dadurch wird der gesamte Variablenbereich auf dem Laufwerk A: gespeichert. Der Befehl schlägt fehl, wenn auf dem Laufwerk A: nicht genügend Speicherplatz vorhanden ist. Bei einem RP2350 mit PSRAM wird der Variablenbereich in einem reservierten Bereich im PSRAM gespeichert und das Laufwerk A: wird nicht verwendet. Siehe auch LOAD CONTEXT
SAVE IMAGE file$ [,x, y, w, h] oder SAVE COMPRESSED IMAGE file$ [,x, y, w, h]	Speichert das aktuelle Bild auf dem Videoausgang oder LCD-Panel als BMP-Datei. Das LCD-Panel muss lesbar sein, z. B. ein Panel auf ILI9341-Basis oder ein VIRTUAL_M- oder VIRTUAL_C-Panel. „file$" ist der Name der Datei. Wenn keine Erweiterung angegeben ist, wird „.BMP" an den Dateinamen angehängt. Das Bild wird als Echtfarbenbild mit 24 Bit gespeichert. „x", „y", „w" und „h" sind optional und bezeichnen die Koordinaten („x" und „y" sind die Koordinaten oben links) und Abmessungen (Breite und Höhe) des zu speichernden Bereichs. Wenn keine Angaben gemacht werden, wird der gesamte Bildschirm gespeichert. Beachten Sie, dass „width" (Breite), falls verwendet, ein Vielfaches von 2 sein muss. SAVE COMPRESSED IMAGE funktioniert genauso, außer dass die Dateigröße durch RLE-Komprimierung reduziert wird.

SAVE PERSISTENT n%	Speichert den Wert n% an einem speziellen Speicherort, der einen Watchdog-Reset oder einen physischen Reset übersteht, jedoch nicht einen Neustart. Siehe auch MM.PERSISTENT oder MM.INFO(PERSISTENT).
SUCHEN [#]fnbr, pos	Positioniert den Lese-/Schreibzeiger in einer Datei, die im Flash-Dateisystem oder auf der SD-Karte geöffnet wurde, für den ZUFÄLLIGEN Zugriff auf das Byte „pos“. Das erste Byte in einer Datei ist mit eins nummeriert, sodass SEEK #5,1 den Lese-/Schreibzeiger an den Anfang der als #5 geöffneten Datei positioniert.

SELECT CASE Wert
CASE testexp [[, testexp] …]
<Anweisungen>
<Anweisungen>
CASE test-n [[, test-n] …]
<Anweisungen>
<Anweisungen>
CASE ELSE
<Anweisungen>
<Anweisungen>
END SELECT
	Führt eine von mehreren Anweisungsgruppen aus, abhängig vom Wert eines Ausdrucks. „Wert” ist der zu testende Ausdruck. Es kann sich um eine Zahlen- oder Zeichenfolgenvariable oder einen komplexen Ausdruck handeln.
	„testexp” (oder test-n) ist der Wert, mit dem verglichen werden soll. Dies kann sein:
	• Ein einzelner Ausdruck (z. B. 34, „Zeichenfolge” oder PIN(4)*5), dem er entsprechen kann
	• Ein Wertebereich in Form von zwei einzelnen Ausdrücken, die durch das Schlüsselwort „TO“ getrennt sind (z. B. 5 TO 9 oder „aa“ TO „cc“)
	• Ein Vergleich, der mit dem Schlüsselwort „IS“ beginnt (optional). Beispiel: IS > 5, IS <= 10.
	Wenn mehrere Testausdrücke (durch Kommas getrennt) verwendet werden, ist die CASE-Anweisung wahr, wenn einer dieser Tests wahr ist.
	Wenn „value“ nicht mit einem „testexp“ übereinstimmt, wird es automatisch mit CASE ELSE abgeglichen. Wenn CASE ELSE nicht vorhanden ist, führt das Programm keine <Anweisungen> und fährt mit dem Code nach END SELECT fort. Bei einer Übereinstimmung werden die <Anweisungen> nach der CASE-Anweisung bis END SELECT oder bis zum nächsten CASE ausgeführt. Danach fährt das Programm mit dem Code nach END SELECT fort.
	Es kann eine unbegrenzte Anzahl von CASE-Anweisungen verwendet werden, aber es darf nur ein CASE ELSE vorhanden sein, und dieses sollte das letzte vor dem END SELECT sein.
	Beispiel:
	`SELECT CASE nbr%`
	`CASE 4, 9, 22, 33 TO 88`
	`statements`
	`CASE IS < 4, IS > 88, 5 TO 8`
	`Anweisungen`
	`CASE ELSE`
	`Anweisungen END`
	`SELECT`
	Jedes SELECT CASE muss genau eine passende END SELECT-Anweisung haben. Eine beliebige Anzahl von SELECT…CASE-Anweisungen kann innerhalb der CASE-Anweisungen anderer SELECT…CASE-Anweisungen verschachtelt werden.

SERVO-Kanal [PositionA] [,PositionB]	Steuert einen Standard-Servo. „positionA” und „positionB” können zwischen -20 und 120 liegen und erzeugen ein 50-Hz-Signal zwischen 800 µs und 2,2 ms. Wie beim PWM-Befehl müssen die Pins mit SETPIN n,PWM eingerichtet werden. Um nur Kanal B zu verwenden, verwenden Sie die Syntax: SERVO Kanal,,Position. Beachten Sie die beiden Kommas, die anzeigen, dass Kanal A nicht eingestellt wird. Beziehen Sie sich auf die Pinbelegung, um den Kanal und den Unterkanal (A oder B) für jeden Pin anzugeben. Beispiele: 90°-Servo: 0 = 0° und 90 = 90° 180°-Servo: 0 = 0° und 90 = 180 Hinweis: Bei Werten < 0 oder > 90 kann der Strom ansteigen, wenn das Servo seine Endposition erreicht. 360°-Servo: Geschwindigkeit 50 = Stopp, links = L->H:51-100, rechts = L->H:49-0.

SETPIN-Pin, cfg [, Option]	Konfiguriert einen externen I/O-Pin. Eine allgemeine Beschreibung der Ein-/Ausgabefunktionen des Pico finden Sie im Kapitel *„Verwendung der I/O-Pins*“. „pin“ ist der zu konfigurierende E/A-Pin, „cfg“ ist der Modus, in den der Pin versetzt werden soll, und „option“ ist ein optionaler Parameter. „cfg“ ist ein Schlüsselwort und kann einer der folgenden Werte sein: OFF Nicht konfiguriert oder inaktiv AIN Analogeingang (d. h. Messung der Spannung am Eingang). ARAW Schneller Analogeingang, der einen Wert zwischen 0 und 4095 zurückgibt. DIN Digitaler Eingang Wenn „option” weggelassen wird, ist der Eingang hochohmig. Wenn „Option” das Schlüsselwort „PULLUP” oder „PULLDOWN” ist, wird ein konstanter Strom von etwa 50 µA verwendet, um den Eingangs-Pin auf 3,3 V hoch- oder herunterzuziehen. Aufgrund eines Fehlers in den RP2350-Chips wird empfohlen, einen Pulldown mit einem Widerstand von 8,2 K oder weniger zu implementieren. FIN Frequenzeingang Mit „option“ kann die Gate-Zeit (die Zeitspanne, die zum Zählen der Eingangszyklen verwendet wird) festgelegt werden. Es kann sich um eine beliebige Zahl zwischen 10 ms und 100000 ms handeln. Die Funktion PIN() gibt unabhängig von der verwendeten Gate-Zeit immer die korrekt skalierte Frequenz in Hz zurück. Wenn „option“ weggelassen wird, beträgt die Gate-Zeit 1 Sekunde. Die Pins können GP6, GP7, GP8 oder GP9 sein (kann mit OPTION COUNT geändert werden). PIN Periodeneingabe Mit „option” kann die Anzahl der Eingangszyklen angegeben werden, über die die Periodenmessung gemittelt werden soll. Es kann eine beliebige Zahl zwischen 1 und 10000 sein. Die Funktion PIN() gibt immer die durchschnittliche Periode eines Zyklus in ms zurück, unabhängig von der Anzahl der für den Durchschnitt verwendeten Zyklen. Wenn „option” weggelassen wird, wird die Periode eines einzigen Zyklus verwendet. Die Pins können GP6, GP7, GP8 oder GP9 sein (kann mit OPTION COUNT geändert werden). CIN Zähleingang Mit „option“ kann festgelegt werden, welche Flanke den Zählvorgang auslöst und ob Pullup oder Pulldown aktiviert ist. 2 gibt eine fallende Flanke mit Pullup an, 3 gibt an, dass sowohl eine fallende als auch eine steigende Flanke einen Zählvorgang auslöst, ohne dass ein Pullup angewendet wird. 5 gibt beide Flanken an, jedoch mit Pullup. Wenn „Option“ weggelassen wird, löst eine steigende Flanke den Zählvorgang aus. Aufgrund eines Fehlers in den RP2350-Chips wird Pulldown nicht empfohlen. Die Pins können GP6, GP7, GP8 oder GP9 sein (kann mit OPTION COUNT geändert werden). DOUT Digitaler Ausgang „option“ wird in diesem Modus nicht verwendet. Die Funktionen PIN() und PORT() können auch verwendet werden, um den Wert an einem oder mehreren Ausgangspins zurückzugeben. Siehe die Funktion PIN() zum Lesen von Eingängen und die Anweisung PIN()= zum Einstellen eines Ausgangs. Siehe den folgenden Befehl, wenn ein Interrupt konfiguriert ist.
SETPIN pin, cfg, target [, option]	Konfiguriert „pin” so, dass ein Interrupt gemäß „cfg” generiert wird. Jeder I/O-Pin, der für digitale Eingaben geeignet ist, kann so konfiguriert werden, dass er einen Interrupt generiert, wobei maximal zehn Interrupts gleichzeitig konfiguriert werden können. „cfg” ist ein Schlüsselwort und kann einer der folgenden Werte sein: OFF Nicht konfiguriert oder inaktiv INTH Interrupt bei Eingang von niedrig auf hoch

	INTL Interrupt bei Eingang von hoch nach niedrig INTB Interrupt bei beiden (d. h. bei jeder Änderung des Eingangs) „target“ ist eine benutzerdefinierte Subroutine, die aufgerufen wird, wenn das Ereignis eintritt. Die Rückkehr aus der Unterbrechung erfolgt über die Befehle END SUB oder EXIT SUB. „option“ entspricht der Verwendung in SETPIN pin DIN (oben). In diesem Modus wird der Pin auch als digitaler Eingang konfiguriert, sodass der Wert des Pins jederzeit mit der Funktion PIN() abgerufen werden kann. Eine allgemeine Beschreibung finden Sie im Kapitel *„Verwendung der E/A-Pins*“.
SETPIN GP25, DOUT | HEARTBEAT	NICHT IN DER WEBMITE-VERSION Diese Version von SETPIN steuert die integrierte LED. Wenn es als DOUT konfiguriert ist, kann es programmgesteuert ein- und ausgeschaltet werden. Wenn es als HEARTBEAT konfiguriert ist, blinkt es bei eingeschaltetem Gerät kontinuierlich 1 Sekunde lang und ist dann 1 Sekunde lang ausgeschaltet. Dies ist der Standardzustand, der wiederhergestellt wird, wenn das Benutzerprogramm nicht mehr ausgeführt wird.
SETPIN p1[, p2 [, p3]], Gerät	Diese Befehle werden für die Pin-Zuweisung für spezielle Geräte verwendet. Die Pins müssen aus dem Pin-Bezeichnungsdiagramm ausgewählt und zugewiesen werden, bevor die Geräte verwendet werden können. Beachten Sie, dass die Pins (z. B. rx, tx usw.) in beliebiger Reihenfolge deklariert werden können und dass die Pins anhand ihrer Pin-Nummer (z. B. 1, 2) oder GP-Nummer (z. B. GP0, GP1) referenziert werden können. Beachten Sie, dass in der WebMite-Version • SPI1 und SPI2 bei GP20 bis GP28 nicht verfügbar sind • COM1 und COM2 sind auf P20 bis GP28 nicht verfügbar • I2C auf Pin 34 (GP28) nicht verfügbar ist • Folgende Pins sind nicht verfügbar: GP29, GP25, GP24 und GP23
SETPIN rx, tx, COM1	Weisen Sie die Pins zu, die für die serielle Schnittstelle COM1 verwendet werden sollen. Gültige Pins sind RX: GP1, GP13 oder GP17 TX: GP0, GP12, GP16 oder GP28
SETPIN rx, tx, COM2	Weisen Sie die Pins zu, die für die serielle Schnittstelle COM2 verwendet werden sollen. Gültige Pins sind RX: GP5, GP9 oder GP21 TX: GP4, GP8 oder GP20
SETPIN rx, tx, clk, SPI	Weisen Sie die Pins zu, die für den SPI-Port SPI verwendet werden sollen. Gültige Pins sind RX: GP0, GP4, GP16 oder GP20 TX: GP3, GP7 oder GP19 CLK: GP2, GP6 oder GP18
SETPIN rx, tx, clk, SPI2	Weisen Sie die Pins zu, die für den SPI-Port SPI2 verwendet werden sollen. Gültige Pins sind RX: GP8, GP12 oder GP28 TX: GP11, GP15 oder GP27 CLK: GP10, GP14 oder GP26
SETPIN sda, scl, I2C	Weisen Sie die Pins zu, die für den I^2C-Port I2C verwendet werden sollen. Gültige Pins sind SDA: GP0, GP4, GP8, GP12, GP16, GP20 oder GP28 SCL: GP1, GP5, GP9, GP13, GP17 oder GP21
SETPIN sda, scl, I2C2	Weisen Sie die Pins dem I^2C-Port I2C2 zu. Gültige Pins sind SDA: GP2, GP6, GP10, GP14, GP18, GP22 oder GP26 SCL: GP3, GP7, GP11, GP15, GP19 oder GP27

SETPIN-Pin, PWM[nx]	Pin zu PWMnx zuweisen „n" ist die PWM-Nummer (0 bis 7) und „x" ist der Kanal (A oder B). n und x sind optional. Der Setpin kann geändert werden, bis der PWM-Befehl ausgegeben wird. Zu diesem Zeitpunkt wird der Pin für PWM gesperrt, bis PWMn,OFF ausgegeben wird.
SETPIN Pin, IR	Weisen Sie Pins für die Infrarotkommunikation (IR) zu (kann jeder Pin sein).
SETPIN-Pin, PIOn	Pin für die Verwendung durch PIO0, PIO1 oder PIO2 reservieren (nur RP2350) (siehe *Anhang F* für Details zu PIO).
SETPIN GP1, FFIN [,gate]	<u>NUR RP2350</u> Legt GP1 als schnellen Frequenzeingang fest. Es können Eingänge bis zur CPU-Geschwindigkeit /2 aufgezeichnet werden. Mit „gate" kann die Gate-Zeit (die Zeitdauer, die zum Zählen der Eingangszyklen verwendet wird) festgelegt werden. Es kann sich um eine beliebige Zahl zwischen 10 ms und 100000 ms handeln. Die Funktion PIN() gibt unabhängig von der verwendeten Gate-Zeit immer die korrekt skalierte Frequenz in Hz zurück. Wenn „option" weggelassen wird, beträgt die Gate-Zeit 1 Sekunde. Die Funktion verwendet den PWM-Kanal 0 zum Zählen und ist daher mit keiner anderen Verwendung dieses PWM-Kanals kompatibel.
SETTICK Periode, Ziel [, Anzahl]	Hiermit wird ein periodischer Interrupt (oder „Tick") eingerichtet. Es stehen vier Tick-Timer zur Verfügung („nbr" ist 1, 2, 3 oder 4). „nbr" ist optional. Wenn es nicht angegeben wird, wird Timer Nummer 1 verwendet. Die Zeit zwischen den Interrupts beträgt „period" Millisekunden und „target" ist die Interrupt-Subroutine, die aufgerufen wird, wenn das zeitgesteuerte Ereignis eintritt. Die Periode kann zwischen 1 und 2147483647 ms (etwa 24 Tage) liegen. Diese Interrupts können deaktiviert werden, indem „period" auf Null gesetzt wird (d. h. SETTICK 0, 0, 3 deaktiviert den Tick-Timer Nummer 3).
SETTICK PAUSE, Ziel [, Anzahl] oder SETTICK RESUME, Ziel [, Anzahl]	Den angegebenen Timer anhalten oder fortsetzen. Im angehaltenen Zustand wird der Interrupt verzögert, aber der aktuelle Zählwert bleibt erhalten.
SORT array() [,indexarray()] [,flags] [,startposition] [,elementstosort]	Dieser Befehl nimmt ein Array beliebigen Typs (Ganzzahl, Gleitkomma oder Zeichenfolge) und sortiert es an Ort und Stelle in aufsteigender Reihenfolge. Er hat einen optionalen Parameter „indexarray%()". Wenn dieser verwendet wird, muss es sich um ein Integer-Array handeln, das dieselbe Größe wie das zu sortierende Array hat. Nach der Sortierung enthält dieses Array die ursprüngliche Indexposition jedes Elements im zu sortierenden Array vor der Sortierung. Alle Daten im Array werden überschrieben. Auf diese Weise können verbundene Arrays sortiert werden. Der Parameter „flag" ist optional. Gültige Flag-Werte sind: bit0: 0 (Standardwert, wenn nicht angegeben) Normale Sortierung - 1 Umgekehrte Sortierung bit1: 0 (Standard) groß-/kleinschreibungsabhängig – 1 Sortierung ist groß-/kleinschreibungsunabhängig (nur Zeichenfolgen). bit2: 0 (Standard) Normale Sortierung – 1 Leere Zeichenfolgen werden an das Ende des Arrays verschoben Der optionale Parameter „startposition" legt fest, bei welchem Element im Array die Sortierung beginnen soll. Der Standardwert ist 0 (OPTION BASE 0) oder 1 (OPTION BASE 1). Die optionale Angabe „elementstosort" legt fest, wie viele Elemente im Array sortiert werden sollen. Der Standardwert ist alle Elemente nach der „startposition". Alle optionalen Parameter können weggelassen werden. Um beispielsweise nur die ersten 50 Elemente eines Arrays zu sortieren, könnten Sie Folgendes verwenden: `SORT array(), , , ,50`

	Beispiel:
	Das Array city$() könnte die Namen von Städten weltweit enthalten und lässt sich mit dem folgenden Befehl ganz einfach in aufsteigender alphabetischer Reihenfolge sortieren: SORT city$()
	Der Befehl SORT funktioniert mit Zeichenfolgen, Gleitkommazahlen und Ganzzahlen, jedoch muss das zu sortierende Array eindimensional sein.
	Häufig werden Daten in mehreren Arrays gespeichert, beispielsweise könnte der Name jeder Stadt im Array city$() gespeichert sein, die Einwohnerzahl im Array pop%() und die Größe der Stadt im Array area!(). Der gleiche Index würde sich auf den Namen, die Einwohnerzahl und die Fläche der Stadt beziehen.
	Das Sortieren und Abrufen dieser Daten ist etwas komplexer, lässt sich jedoch relativ einfach mit einem optionalen Parameter für den Sortierbefehl wie folgt durchführen:
	SORT array(), indexarray%()
	indexarray%() muss ein eindimensionales Integer-Array sein, das dieselbe Größe wie das zu sortierende Array hat. Nach dem Sortieren enthält indexarray%() den entsprechenden Index zu den Originaldaten vor dem Sortieren. (Alle zuvor in indexarray%() enthaltenen Daten werden überschrieben).
	Um auf die sortierten Daten zuzugreifen, kopieren Sie zunächst das Array, das den Hauptschlüssel enthält, in ein temporäres Array und sortieren dieses unter Angabe von indexarray%(). Nach dem Sortieren kann indexarray%() zum Indizieren der ursprünglichen Arrays verwendet werden.
	Beispiel:
	`DIM city$(100),pop%(100),area!(100),sindex%(100),t$(100)`
	`FOR i = 0 to 100`
	`  t$(i) = city$(i)  ' temporäre Kopie der Schlüssel`
	`NEXT i`
	`SORT t$(), sindex%()  ' Sortiere das temporäre Array,`
	`FOR i = 0 bis 100`
	`  k = sindex%(i)  ' Index zum ursprünglichen Array`
	`  PRINT city$(k),pop%(k),area!(k)' in sortierter Reihenfolge ausgeben`
	`NEXT i`

SPI OPEN Geschwindigkeit, Modus, Bits oder SPI READ nbr, array() oder SPI WRITE nbr, data1, data2, data3, … etc oder SPI WRITE nbr, string$ oder SPI WRITE nbr, array() oder SPI CLOSE	Kommunikation über einen SPI-Kanal. Details finden Sie in *Anhang D*. „nbr" ist die Anzahl der zu sendenden oder empfangenden Datenelemente. „data1", „data2" usw. können Float- oder Integer-Werte sein und im Fall von WRITE eine Konstante oder ein Ausdruck sein. Wenn „string$" verwendet wird, werden „nbr" Zeichen gesendet. „array" muss ein eindimensionales Float- oder Integer-Array sein, und „nbr" Elemente werden gesendet oder empfangen.
SPI2	Dieselbe Reihe von Befehlen wie für SPI (oben), jedoch für den zweiten SPI-Kanal.
SPRITE	NUR VGA- UND HDMI-VERSIONEN Die SPRITE-Befehle dienen zur Bearbeitung kleiner Grafiken auf dem VGA- oder HDMI-Bildschirm und sind beim Schreiben von Spielen nützlich. Sprites funktionieren nur in Framebuffern in den Modi 2 und 3. Sprites werden aus Effizienzgründen immer als RGB121-„Nibbles" gespeichert. Die maximale Größe eines Sprites beträgt MM.HRES-1 und MM.VRES-1. Siehe auch den Befehl BLIT und die Funktionen SPRITE().

SPRITE CLOSE [#]n	Schließt Sprite „n“ und gibt dessen Speicherressourcen frei, sodass die Sprite-Nummer wiederverwendet werden kann. Der Befehl gibt eine Fehlermeldung aus, wenn andere Sprites aus diesem Sprite kopiert werden, sofern diese nicht zuvor geschlossen wurden.
SPRITE CLOSE ALL	Schließt alle Sprites und gibt den gesamten Sprite-Speicher frei. Der Bildschirm wird nicht verändert.
SPRITE COPY [#]n, [#]m, nbr	Erstellt eine Kopie von Sprite „n“ in „nbr“ neuer Sprites, beginnend mit der Nummer „m“. Kopierte Sprites verwenden dasselbe geladene Bild wie das Original, um Speicherplatz zu sparen.
SPRITE HIDE [#]n	Entfernt Sprite n aus der Anzeige und ersetzt den gespeicherten Hintergrund. Um einen Bildschirm in einen früheren Zustand zurückzusetzen, sollten Sprites in umgekehrter Reihenfolge zu ihrer Schreibweise „LIFO” ausgeblendet werden.
SPRITE ALLE AUSBLENDEN	Blendet alle Sprites aus, sodass der Hintergrund bearbeitet werden kann. Die folgenden Befehle können nicht verwendet werden, wenn alle Sprites ausgeblendet sind: SPRITE ANZEIGEN (SICHER) SPRITE HIDE (SAFE, ALL) SPRITE SWAP SPRITE MOVE SPRITE SCROLLR SPRITE SCROLL
SPRITE WIEDERHERSTELLEN	Stellt die Sprites wieder her, die zuvor mit SPRITE HIDE ALL ausgeblendet wurden.
SPRITE HIDE SAFE [#]n	Entfernt Sprite n aus der Anzeige und ersetzt den gespeicherten Hintergrund. Blendet automatisch alle neueren Sprites sowie das angeforderte Sprite aus und ersetzt sie anschließend. Dadurch wird sichergestellt, dass Sprites, die von anderen Sprites überdeckt werden, entfernt werden können, ohne dass der Benutzer die Schreibreihenfolge verfolgen muss. Natürlich ist diese Version weniger leistungsfähig als die einfache Version und sollte nur verwendet werden, wenn die Gefahr besteht, dass das Sprite teilweise überdeckt wird.
SPRITE INTERRUPT sub	Gibt den Namen der Unterroutine an, die aufgerufen wird, wenn eine Sprite-Kollision auftritt. In Anhang G wird beschrieben, wie Sie mit der Funktion SPRITE Details zu den kollidierten Objekten abfragen können.
SPRITE READ [#]b, x, y, w, h	Damit wird ein Teil der Anzeige in den Speicherpuffer „#b” kopiert. Die Quellkoordinaten sind „x” und „y”, die Breite des zu kopierenden Anzeigebereichs ist „w” und die Höhe ist „h”. Bei Verwendung dieses Befehls wird der Speicherpuffer automatisch erstellt und ausreichend Speicher zugewiesen. Dieser Puffer kann mit dem Befehl SPRITE CLOSE freigegeben und der Speicher zurückgewonnen werden.
SPRITE WRITE [#]b, x, y [,mode]	Kopiert das Sprite „#b” auf das Display. Die Zielkoordinaten sind „x” und „y”. Der optionale Parameter „mode” ist standardmäßig auf 4 gesetzt und legt fest, wie die gespeicherten Bilddaten beim Auslesen verändert werden. Es handelt sich um die bitweise UND-Verknüpfung der folgenden Werte: &B001 = von links nach rechts gespiegelt &B010 = von oben nach unten gespiegelt &B100 = keine transparenten Pixel kopieren
SPRITE LOAD fname$ [,start_sprite_number] [,mode]	Lädt die Datei „fname$”, die als originale Colour Maximite-Sprite-Datei formatiert sein muss. Das Dateiformat finden Sie im *Original-Handbuch* zur *Sprache* Colour Maximite *MMBasic*. Es können mehrere Sprite-Dateien geladen werden, indem für jede Datei eine andere „start_sprite_number” angegeben wird. Der Programmierer muss sicherstellen, dass sich die Sprites nicht überlappen. Der Modus ist standardmäßig auf Null eingestellt. In diesem Fall werden die CMM1/CMM2-Farbcodes verwendet (Schwarz, Blau, Grün, Cyan, Rot, Magenta, Gelb, Weiß, Myrte, Kobalt, Mittelgrün, Cerulean, Rost, Fuchsia, Braun, Flieder). Wenn der Modus als 1 angegeben ist, werden die RGB121-Farbcodes verwendet: (Schwarz, Blau, Myrte, Kobalt, Mittelgrün, Cerulean, Grün, Cyan, Rot, Magenta, Rost, Fuchsia, Braun, Flieder, Gelb, Weiß).

SPRITE LOADARRAY [#]n, w, h, array%()	Erstellt das Sprite „n" mit der Breite „w" und der Höhe „h", indem w*h RGB888-Werte aus „array%()" gelesen werden. Die RGB888-Werte müssen in der Reihenfolge der Spalten und dann der Zeilen beginnend oben links gespeichert werden. Auf diese Weise kann der Programmierer einfache Sprites in einem Programm erstellen, ohne sie von der Festplatte laden oder vom Display lesen zu müssen. Die Firmware generiert einen Fehler, wenn „array%()" nicht groß genug ist, um die erforderliche Anzahl von Werten aufzunehmen.
SPRITE LOADBMP [#]b, fname$ [,x] [,y] [,w] [,h]	Lädt einen Blit-Puffer aus einer 24-Bit-BMP-Bilddatei. „x" und „y" definieren die Startposition im Bild, an der mit dem Laden begonnen werden soll, und „w" und „h" geben die Breite und Höhe des zu ladenden Bereichs an. Beispiel: `SPRITE LOAD #1,"image1", 50,50,100,100` lädt einen Bereich von 100 Pixeln im Quadrat mit der oberen linken Ecke bei 50, 50 aus dem Bild image1.bmp.
SPRITE LOADPNG [#]b, fname$ [,transparent] [,alphacut]	Lädt SPRITE Nummer „b" aus der PNG-Datei „fname$". Wenn keine Erweiterung angegeben ist, wird automatisch .png an den Dateinamen angehängt. Die Datei muss im RGBA8888-Format vorliegen, was die normale Standardeinstellung ist. Der optionale Parameter „transparent" (Standardwert 0) gibt einen der Farbcodes (0–15) an, der den Pixeln in der PNG-Datei mit einem Alpha-Wert unter „alphacut" (Standardwert 20) zugewiesen wird. Die variable Transparenz kann dann mit dem Befehl SPRITE SET TRANSPARENT n oder FRAMEBUFFER LAYER n verwendet werden, um das Sprite mit dem transparenten Bereich ausgeblendet anzuzeigen.
SPRITE MOVE	Führt eine einzelne atomare Transaktion aus, die alle Sprites neu positioniert, für die zuvor mit dem Befehl SPRITE NEXT eine Positionsänderung festgelegt wurde. Kollisionen werden erkannt, sobald alle Sprites verschoben sind, und auf die gleiche Weise wie bei einem Bildlauf gemeldet.
SPRITE NEXT [#]n, x, y	Legt die X- und Y-Koordinaten des Sprites fest, die beim nächsten Bildlauf oder bei Ausführung des Befehls SPRITE MOVE verwendet werden sollen. Durch die Verwendung von SPRITE NEXT anstelle von SPRITE SHOW können mehrere Sprites als Teil derselben atomaren Transaktion verschoben werden.
SPRITE SCROLL x, y [,col]	Verschiebt den Hintergrund und alle Sprites auf dem aktiven Framebuffer (L oder N) um „x" Pixel nach rechts und „y" Pixel nach oben. „x" kann eine beliebige Zahl zwischen - MM.HRES-1 und MM.HRES-1 sein, „y" kann eine beliebige Zahl zwischen -MM.VRES-1 und MM.VRES-1 sein. Sprites auf einer anderen Ebene als Null bleiben an ihrer Position auf dem Bildschirm fixiert. Standardmäßig wird das Bild beim Scrollen umgebrochen. Wenn „col" angegeben ist, ersetzt die Farbe den Bereich hinter dem gescrollten Bild. Wenn „col" auf -1 gesetzt ist, bleibt der gescrollte Bereich unverändert.
SPRITE SET TRANSPARENT n	Legt den Farbcode (0-15) fest, der als transparent verwendet wird, wenn Sprites über einem Hintergrund angezeigt werden (Standardwert ist 0).
SPRITE SHOW [#]n, x,y, layer [,options]	Zeigt das Sprite „n" auf dem Bildschirm an, wobei sich die obere linke Ecke an den Koordinaten „x" und „y" befindet. Sprites kollidieren nur mit anderen Sprites auf derselben Ebene, Ebene Null, oder mit dem Bildschirmrand. Wenn ein Sprite bereits auf dem Bildschirm angezeigt wird, verschiebt der Befehl SPRITE SHOW das Sprite an die neue Position. Der Anzeigehintergrund wird als Teil des Befehls gespeichert und ersetzt, wenn das Sprite ausgeblendet oder weiter verschoben wird. Der Parameter „options" ist optional und kann wie folgt eingestellt werden: Bit 0 gesetzt – von links nach rechts gespiegelt Bit 1 gesetzt – von oben nach unten gespiegelt Bit 2 gesetzt – schwarze Pixel werden nicht als transparent behandelt, Standardwert ist 0
SPRITE SHOW SAFE [#]n, x,y, Ebene [,Ausrichtung] [,ontop]	Zeigt ein Sprite an und gleicht automatisch alle anderen Sprites aus, die es überlappen. Wenn das Sprite noch nicht angezeigt wird, verhält sich der Befehl genau wie SPRITE SHOW. Wenn das Sprite bereits angezeigt wird, wird es verschoben und bleibt in seiner Position relativ zu

	Andere Sprites basieren auf der ursprünglichen Reihenfolge, in der sie geschrieben wurden. Das heißt, wenn Sprite 1 vor Sprite 2 geschrieben wurde und so verschoben wird, dass es Sprite 2 überlappt, wird es unter Sprite 2 angezeigt. Wenn der optionale Parameter „ontop" auf 1 gesetzt ist, wird das verschobene Sprite zum neuesten Sprite und liegt über allen anderen Sprites, die es überlappt. Weitere Informationen zum Orientierungsparameter finden Sie unter SPRITE SHOW.
SPRITE SWAP [#]n1, [#]n2 [,Ausrichtung]	Ersetzt das Sprite „n1" durch das Sprite „n2". Die Sprites müssen dieselbe Breite und Höhe haben, und „n1" muss angezeigt werden, sonst wird ein Fehler generiert. Weitere Informationen zum Orientierungsparameter finden Sie unter SPRITE SHOW. Das Ersatz-Sprite übernimmt den Hintergrund des Originals sowie dessen Position in der Reihenfolge der Zeichnung.
STATIC variable [, variables] Die vollständige Syntax finden Sie unter DIM.	Definiert eine Liste von Variablennamen, die lokal für die Subroutine oder Funktion sind. Diese Variablen behalten ihren Wert zwischen den Aufrufen der Subroutine oder Funktion (im Gegensatz zu Variablen, die mit dem Befehl LOCAL erstellt wurden). Dieser Befehl verwendet genau dieselbe Syntax wie DIM. Der einzige Unterschied besteht darin, dass die Länge des durch STATIC erstellten Variablennamens und die Länge des Unterprogramm- oder Funktionsnamens zusammen nicht mehr als 31 Zeichen betragen dürfen. Statische Variablen können mit einem Wert initialisiert werden. Diese Initialisierung wird nur beim ersten Aufruf der Unterroutine wirksam (nicht bei nachfolgenden Aufrufen).
SUB xxx (arg1 [,arg2, …]) <Anweisungen> <Anweisungen> END SUB	Definiert eine aufrufbare Unterroutine. Dies entspricht dem Hinzufügen eines neuen Befehls zu MMBasic während der Ausführung Ihres Programms. „xxx" ist der Name der Unterroutine und muss den Spezifikationen für die Benennung einer Variablen entsprechen. 'arg1', 'arg2' usw. sind die Argumente oder Parameter für die Subroutine. Ein Array wird durch leere Klammern angegeben, z. B. arg3(). Der Typ des Arguments kann durch Verwendung eines Typ-Suffixes (z. B. arg1$) oder durch Angabe des Typs mit `AS <Typ> (z. B. arg1 AS STRING)` angegeben werden. Argumente in der Liste des Aufrufers, die eine Variable sind und den richtigen Typ haben, werden per Referenz an die Unterroutine übergeben. Das bedeutet, dass alle Änderungen am entsprechenden Argument in der Unterroutine auch in die Variable des Aufrufers kopiert werden und daher nach Beendigung der Unterroutine darauf zugegriffen werden kann. Dem Argument kann das Präfix BYVAL vorangestellt werden, wodurch dieser Mechanismus verhindert wird und nur der Wert verwendet wird. Alternativ weist das Präfix BYREF MMBasic an, dass eine Referenz erforderlich ist, und es wird ein Fehler generiert, wenn dies nicht möglich ist. Arrays werden durch Angabe des Array-Namens mit leeren Klammern (z. B. arg()) übergeben, werden immer per Referenz übergeben und müssen vom richtigen Typ sein. Jede Definition muss eine END SUB-Anweisung enthalten. Wenn diese erreicht ist, kehrt das Programm nach dem Aufruf der Unterroutine zur nächsten Anweisung zurück. Der Befehl EXIT SUB kann für einen vorzeitigen Abbruch verwendet werden. Sie verwenden die Subroutine, indem Sie ihren Namen und ihre Argumente in einem Programm genauso wie einen normalen Befehl verwenden. Beispiel: `MySub a1, a2` Wenn die Unterroutine aufgerufen wird, wird jedes Argument im Aufrufer mit dem Argument in der Unterroutinen-Definition abgeglichen. Diese Argumente sind nur innerhalb der Unterroutine verfügbar. Unterroutinen können mit einer variablen Anzahl von Argumenten aufgerufen werden. Alle in der Liste der Unterroutine ausgelassenen Argumente werden auf Null oder eine Null-Zeichenkette gesetzt. Klammern um die Argumentliste sowohl im Aufrufer als auch in der Definition sind optional.
SYNC time% [,period] oder SYNC	Mit dem Befehl SYNC kann der Benutzer sehr präzise zeitgesteuerte wiederholte Aktionen implementieren (Genauigkeit von 1–2 Mikrosekunden). Um dies zu ermöglichen, wird der Befehl zunächst mit dem Parameter „time%" aufgerufen. Dadurch wird eine sich wiederholende Uhr für „time%" Mikrosekunden eingerichtet. Der optionale Parameter „period"

	ändert die Zeit und kann „U“ für Mikrosekunden, „M“ für Millisekunden oder „S“ für Sekunden sein.
	Sobald die Uhr eingerichtet ist, wird das Programm mit dem Befehl SYNC ohne Parameter daran synchronisiert. Dieser wartet, bis die Uhrperiode abgelaufen ist. Bei Perioden unter 2 ms ist dies nicht unterbrechbar. Bei Perioden über 2 ms reagiert das Programm auf Strg-C, aber nicht auf MMBasic-Interrupts.
	Typischerweise wird die Uhr außerhalb einer Schleife eingestellt und dann am Anfang der Schleife der Befehl SYNC ohne Parameter aufgerufen. Das bedeutet, dass der Inhalt der Schleife genau einmal pro eingestellter Taktperiode ausgeführt wird. Das folgende Beispiel würde beispielsweise einen Servo mit der erforderlichen präzisen 50-Hz-Taktung ansteuern:
	`SYNC 20, M`
	`DO`
	`  SYNC`
	`  PULSE GP0,n LOOP`

TEMPR START-Pin [, Präzision] [,Zeitlimit]	Mit diesem Befehl kann eine Umwandlung gestartet werden, die auf einem an „Pin” angeschlossenen DS18B20-Temperatursensor ausgeführt wird. Normalerweise reicht die Funktion TEMPR() allein aus, um eine Temperaturmessung durchzuführen, sodass die Verwendung dieses Befehls optional ist. Weitere Informationen finden Sie im Abschnitt *„Temperaturmessung*”. Dieser Befehl startet die Messung am Temperatursensor. Das Programm kann dann während der Messung andere Aufgaben ausführen und später die Funktion TEMPR() verwenden, um den Messwert abzurufen. Wenn die Funktion TEMPR() vor Ablauf der Umwandlungszeit verwendet wird, wartet die Funktion die verbleibende Umwandlungszeit ab, bevor sie den Wert zurückgibt. Es kann eine beliebige Anzahl dieser Umwandlungen (an verschiedenen Pins) gestartet werden, die gleichzeitig ausgeführt werden. „precision“ ist die Auflösung der Messung und optional. Es handelt sich um eine Zahl zwischen 0 und 3 mit folgender Bedeutung: 0 = Auflösung 0,5 °C, Umwandlungszeit 100 ms. 1 = Auflösung 0,25 °C, Umwandlungszeit 200 ms (dies ist die Standardeinstellung). 2 = Auflösung 0,125 °C, Umwandlungszeit 400 ms. 3 = Auflösung 0,0625 °C, Umwandlungszeit 800 ms. Der optionale Timeout-Parameter überschreibt die oben genannten Umwandlungszeiten, um langsame Geräte zu berücksichtigen.
TEXT x, y, string$ [,alignment$] [, font] [, scale] [, c] [, bc]	Zeigt eine Zeichenfolge auf dem Videoausgang oder dem angeschlossenen LCD-Panel beginnend bei „x” und „y” an. „string$“ ist die anzuzeigende Zeichenfolge. Numerische Daten sollten in eine Zeichenfolge konvertiert und mit der Funktion Str$() formatiert werden. „alignment$“ ist ein String-Ausdruck oder eine String-Variable, die aus 0, 1 oder 2 Buchstaben besteht, wobei der erste Buchstabe die horizontale Ausrichtung um „x“ angibt und L, C oder R für LEFT (links), CENTER (zentriert) oder RIGHT (rechts) stehen kann, und der zweite Buchstabe die vertikale Ausrichtung um „y“ angibt und T, M oder B für TOP (oben), MIDDLE (Mitte) oder BOTTOM (unten) stehen kann. Die Standardausrichtung ist links/oben. Beispiel: „CM” zentriert den Text vertikal und horizontal. Die Zeichenfolge „alignment$“ kann eine Konstante (z. B. „CM“) oder eine Zeichenfolgenvariable sein. Aus Gründen der Abwärtskompatibilität mit früheren Versionen von MMBasic kann die Zeichenfolge auch ohne Anführungszeichen angegeben werden (z. B. CM). Ein dritter Buchstabe kann in der Ausrichtungszeichenfolge verwendet werden, um die Drehung des Textes anzugeben. Dies kann „N“ für normale Ausrichtung, „V“ für vertikalen Text, bei dem jedes Zeichen unter dem vorherigen von oben nach unten läuft, „I“ für umgekehrten Text (d. h. auf dem Kopf stehend), „U“ für eine Drehung des Textes um 90º gegen den Uhrzeigersinn und „D“ für eine Drehung des Textes um 90º im Uhrzeigersinn sein. „font“ und „scale“ sind optional und werden standardmäßig durch die mit dem Befehl FONT festgelegten Werte ersetzt.

	„c“ ist die Zeichenfarbe und „bc“ ist die Hintergrundfarbe. Sie sind optional und werden standardmäßig auf die aktuellen Vordergrund- und Hintergrundfarben gesetzt. Eine Definition der Farben und Grafikkoordinaten finden Sie im Kapitel *Grafikbefehle und -funktionen*.
TILE x, y [,foreground] [,background] [,nbr_tiles_wide] [,nbr_tiles_high] TILE HEIGHT n	NUR VGA- ODER HDMI-VERSIONEN MODUS 1 Legt die Farbe für eine oder mehrere Kacheln auf dem Bildschirm fest. Im Standard-Monochrom-Modus ist der Bildschirm in 80x40 Kacheln mit jeweils 8x12 Pixeln unterteilt. Dies entspricht der Schriftart 1 und ermöglicht eine vollständige Farbcodierung im Editor im Monochrom-Modus. Jeder Kachel kann eine andere Vordergrund- und Hintergrundfarbe zugewiesen bekommen, die aus den folgenden Farben ausgewählt werden kann: Weiß, Gelb, Lila, Braun, Fuchsia, Rostrot, Magenta, Rot, Cyan, Grün, Ceruleanblau, Mittelgrün, Kobaltblau, Myrtenblau, Blau und Schwarz. „x” und „y” sind die Koordinaten des Startblocks (0-79, 0-39). „foreground” und „background” sind die neu ausgewählten Farben. „nbr_tiles_wide” und „nbr_tiles_high” sind die Anzahl der zu ändernden Kacheln. Die Änderung erfolgt sofort und hat keine Auswirkungen auf den aktuell in den Kacheln angezeigten Text oder die Grafiken (nur auf die Farben). Legt die Höhe der Kacheln fest. „n“ kann zwischen 12 und 480 (RP2040) oder zwischen 8 und 480 (RP2350) liegen.
TIME$ = „HH:MM:SS” oder TIME$ = „HH:MM” oder TIME$ = „HH“	Stellt die Zeit der internen Uhr ein. MM und SS sind optional und werden standardmäßig auf Null gesetzt, wenn sie nicht angegeben werden. Beispielsweise setzt TIME$ = „14:30“ die Uhr auf 14:30 Uhr mit null Sekunden. Mit OPTION RTC AUTO ENABLE startet die PicoMite-Firmware mit der in RTC programmierten TIME$. Ohne OPTION RTC AUTO ENABLE startet die Firmware mit TIME$="00:00:00".
TIMER = msec	Setzt den Timer auf eine bestimmte Anzahl von Millisekunden zurück. Normalerweise wird dies nur verwendet, um den Timer auf Null zurückzusetzen, aber Sie können ihn auf jede beliebige positive Zahl einstellen. Weitere Informationen finden Sie unter der Funktion TIMER.
TRACE ON oder TRACE OFF oder TRACE LIST nn	TRACE ON/OFF schaltet die Trace-Funktion ein bzw. aus. Diese Funktion gibt während der Ausführung des Programms die Nummer jeder Zeile (gezählt vom Beginn des Programms) in eckigen Klammern aus. Dies ist beim Debuggen von Programmen hilfreich. TRACE LIST listet die letzten „nn” ausgeführten Zeilen in dem oben beschriebenen Format auf. MMBasic protokolliert immer die ausgeführten Zeilen, sodass diese Funktion immer verfügbar ist (d. h. sie muss nicht eingeschaltet werden).
TRIANGLE X1, Y1, X2, Y2, X3, Y3 [, C [, FILL]]	Zeichnet ein Dreieck auf dem angeschlossenen Videoausgang oder LCD-Display mit den Ecken bei X1, Y1 und X2, Y2 und X3, Y3. „C” ist die Farbe des Dreiecks und standardmäßig die aktuelle Vordergrundfarbe. „FILL” ist die Füllfarbe und standardmäßig keine Füllung (sie kann auch auf -1 für keine Füllung gesetzt werden). Alle Parameter können als Arrays ausgedrückt werden, und die Software zeichnet die Anzahl der Dreiecke, die durch die Abmessungen des kleinsten Arrays bestimmt wird, es sei denn, X1 = Y1 = X2 = Y2 = X3 = Y3 = -1. In diesem Fall wird die Verarbeitung an diesem Punkt „x1”, „y1”, „x2”, „y2“, „x3“ und „y3“ müssen alle Arrays oder alle einzelne Variablen/Konstanten sein, andernfalls wird ein Fehler generiert. „c“ und „fill“ können entweder Arrays oder einzelne Variablen/Konstanten sein.
TRIANGLE SAVE [#]n, x1,y1,x2,y2,x3,y3 TRIANGLE RESTORE [#]n	Speichert einen dreieckigen Bereich des Bildschirms im Puffer #n. Stellt einen gespeicherten dreieckigen Bereich des Bildschirms wieder her und löscht den gespeicherten Puffer.
TURTLE	Die Firmware implementiert eine vollständige Turtle-Grafik-Engine. Siehe Anhang H

FIRMWARE AKTUALISIEREN	NICHT BEI USB-VERSIONEN Versetzt die PicoMite-Firmware in den Firmware-Aktualisierungsmodus (entspricht dem Einschalten bei gedrückter BOOTSEL-Taste). Dieser Befehl ist nur an der Eingabeaufforderung verfügbar.
VAR SAVE var [, var]… oder VAR-Wiederherstellung oder VAR CLEAR	VAR SAVE speichert eine oder mehrere Variablen in einem nichtflüchtigen Flash-Speicher, wo sie später (normalerweise nach einer Stromunterbrechung) wiederhergestellt werden können. „var" kann eine beliebige Anzahl von numerischen oder Zeichenfolgenvariablen und/oder Arrays sein. Arrays werden durch leere Klammern angegeben. Beispiel: var() Der Befehl VAR SAVE kann wiederholt verwendet werden. Zuvor gespeicherte Variablen werden mit ihrem neuen Wert aktualisiert, und alle neuen Variablen (die zuvor nicht gespeichert wurden) werden der gespeicherten Liste hinzugefügt, um später wiederhergestellt zu werden. VAR RESTORE ruft die zuvor gespeicherten Variablen ab und fügt sie (und ihre Werte) in die Variablentabelle ein. VAR CLEAR löscht alle gespeicherten Variablen. Dieser Befehl wird normalerweise zum Speichern von Kalibrierungsdaten, Optionen und anderen Daten verwendet, die sich nicht oft ändern, aber auch bei einem Stromausfall erhalten bleiben müssen. Normalerweise wird der Befehl VAR RESTORE am Anfang des Programms platziert, damit zuvor gespeicherte Variablen wiederhergestellt werden und dem Programm beim Start sofort zur Verfügung stehen. Hinweise: • Der für diesen Befehl verfügbare Speicherplatz beträgt 16 KB. • Die Verwendung von VAR RESTORE ohne vorherige Speicherung hat keine Auswirkungen und führt nicht zu einer Fehlermeldung. • Wenn bei Verwendung von RESTORE bereits eine Variable mit demselben Namen vorhanden ist, wird deren Wert überschrieben. • Gespeicherte Arrays müssen (mit DIM) deklariert werden, bevor sie wiederhergestellt werden können. • Beachten Sie, dass String-Arrays schnell den gesamten für diesen Befehl zugewiesenen Speicherplatz belegen können. Der Qualifizierer LENGTH kann bei der Deklaration eines String-Arrays verwendet werden, um die Größe des Arrays zu reduzieren (siehe Befehl DIM). Bei normalen String-Variablen ist dies nicht erforderlich. • Die gespeicherten Variablen werden durch ein Firmware-Upgrade, durch den Befehl NEW oder beim Laden eines neuen Programms über AUTOSAVE, XMODEM usw. automatisch gelöscht.
WATCHDOG-Timeout oder WATCHDOG OFF oder WATCHDOG HW-Timeout oder WATCHDOG HW OFF	Startet den Watchdog-Timer, der die Prozessoren nach Ablauf der Zeit automatisch neu startet. Dies kann verwendet werden, um nach einem Ereignis, das das laufende Programm deaktiviert hat (z. B. eine Endlosschleife oder ein Programmier- oder anderer Fehler, der ein laufendes Programm anhält), wiederherzustellen. Dies kann in einer unbeaufsichtigten Kontrollsituation wichtig sein Situation. Die Zeitüberschreitung kann entweder im System-Timer-Interrupt (WATCHDOG-Befehl) oder als echter CPU/Hardware-Watchdog (WATCHDOG HW-Befehl) verarbeitet werden. Wenn der Hardware-Watchdog verwendet wird, hat der Timer eine maximale Dauer von 8,3 Sekunden. Für den Software-Watchdog gibt es keine solche Begrenzung. „timeout" ist die Zeit in Millisekunden (ms), bevor ein Neustart erzwungen wird. Dieser Befehl sollte an strategischen Stellen im laufenden BASIC-Programm platziert werden, um den Watchdog-Timer ständig zurückzusetzen (auf „timeout") und somit zu verhindern, dass er bis Null herunterzählt. Wenn der Timer tatsächlich Null erreicht (möglicherweise weil das BASIC-Programm nicht mehr ausgeführt wird), wird die PicoMite-Firmware automatisch neu gestartet und die automatische Variable MM.WATCHDOG wird auf „true" (d. h. 1) gesetzt, was anzeigt, dass ein Fehler aufgetreten ist. Bei einem normalen Start wird MM.WATCHDOG auf „false" (d. h. 0) gesetzt. Beachten Sie, dass OPTION AUTORUN angegeben werden muss, damit das Programm neu gestartet wird.

	Mit WATCHDOG OFF kann der Watchdog-Timer deaktiviert werden (dies ist die Standardeinstellung bei einem Reset oder beim Einschalten). Der Timer wird auch ausgeschaltet, wenn das Break-Zeichen (STRG-C) auf der Konsole verwendet wird, um ein laufendes Programm zu unterbrechen.
WII [CLASSIC] OPEN [,interrupt]	Öffnet einen WII Classic-Controller und implementiert eine Hintergrundabfrage des Geräts. Der Wii Classic muss an die von OPTION SYSTEM I2C angegebenen Pins angeschlossen sein, was eine Voraussetzung ist. Open versucht, mit dem Wii Classic zu kommunizieren, und gibt einen Fehler zurück, wenn er nicht gefunden wird. Wenn er gefunden wird, tastet die Firmware die Wii-Daten im Hintergrund mit einer Rate von 50 Hz ab. Wenn eine optionale Benutzerunterbrechung angegeben ist, wird diese ausgelöst, wenn sich eine der Tasten ändert (sowohl ein- als auch ausgeschaltet). Informationen zum Auslesen von Daten aus dem Wii Classic finden Sie unter der Funktion DEVICE.
WII [CLASSIC] CLOSE	CLOSE stoppt die Hintergrundabfrage und deaktiviert alle angegebenen Interrupts.
WII NUNCHUCK OPEN [,interrupt]	Öffnet einen WII Nunchuck-Controller und implementiert die Hintergrundabfrage des Geräts. Der Wii Nunchuck muss an die durch OPTION SYSTEM I2C angegebenen Pins angeschlossen sein, was eine Voraussetzung ist. Versucht, mit dem Wii Nunchuck zu kommunizieren, und gibt einen Fehler zurück, wenn dieser nicht gefunden wird. Wird er gefunden, tastet die Firmware die Wii-Daten im Hintergrund mit einer Rate von 50 Hz ab. Wenn eine optionale Benutzerunterbrechung angegeben ist, wird diese ausgelöst, wenn sich eine der Tasten ändert (sowohl ein- als auch ausgeschaltet). Informationen zum Auslesen von Daten aus dem Wii Nunchuck finden Sie unter der Funktion DEVICE.
WII NUNCHUCK CLOSE	CLOSE stoppt die Hintergrundabfrage und deaktiviert alle angegebenen Unterbrechungen.
WEB	NUR WEBMITE Die WEB-Befehle dienen zur Verwaltung der Internetfunktionen des WebMite.
WEB CONNECT [ssid$, passwd$, [name$] [,ipaddress$, mask$, gateway$]]	Dieser Befehl ohne optionale Parameter stellt nach Möglichkeit eine Verbindung zum Standardnetzwerk her (wie zuvor mit OPTION WIFI festgelegt) oder stellt mit den optionalen Parametern eine Verbindung zum angegebenen Netzwerk her und richtet OPTION WIFI für die zukünftige Verwendung ein.
WEB MQTT CONNECT addr$, port, user$, passwd$ [, interrupt]	Verbindung zu einem MQTT-Broker herstellen. „addr$" ist die IP-Adresse, „port" ist die zu verwendende Portnummer, „user$" ist der Benutzername, „passwd$" ist das Passwort des Kontos und „interrupt" ist optional und, falls angegeben, die Subroutine, die beim Empfang einer Nachricht aufgerufen wird. WEB CONNECT trennt die Verbindung zu einem zuvor verbundenen Netzwerk nicht, daher sollte es nur verwendet werden, wenn zuvor nichts eingerichtet wurde oder wenn ein zuvor konfiguriertes Netzwerk nicht aktiv ist oder ein zuvor konfiguriertes Netzwerk beim Booten keine Verbindung herstellen konnte (keine Parameter).
WEB MQTT PUBLISH topic$, msg$, [,qos] [,retain]	Veröffentlicht Inhalte in einem MQTT-Broker-Thema. „topic$" ist der Name des Themas und „msg$" ist die Nachricht/ „qos" ist die optionale Dienstgüte mit den Werten 0, 1 oder 2 (Standardwert ist 1).
WEB MQTT SUBSCRIBE topic$ [,qos]	Abonnieren Sie ein MQTT-Broker-Thema. „topic$" ist der Name des Themas und „qos" ist die optionale Dienstgüte mit den Werten 0, 1 oder 2 (Standard ist 1).
WEB MQTT UNSUBSCRIBE topic$	Abonnement eines MQTT-Broker-Themas kündigen. „topic$" ist der Name des Themas.
WEB MQTT CLOSE	Schließt eine persistente MQTT-Verbindung.

WEB NTP [timeoffset [, NTPserver$]] [,timeout]]]	Rufen Sie das Datum/die Uhrzeit von einem NTP-Server ab und stellen Sie die interne WebMite-Uhr ein. „timeoffset" ist die lokale Zeitzone. Wird diese Angabe weggelassen, wird das Datum/die Uhrzeit auf GMT eingestellt. „NTPserver$" ist der zu verwendende Zeitserver. Wird diese Angabe weggelassen, wird standardmäßig ein internationaler Zeitserverpool verwendet. „timeout" ist die optionale Zeitüberschreitung in Millisekunden und standardmäßig auf 5000 eingestellt.
WEB OPEN TCP CLIENT address$, port	Öffnet eine TCP-Client-Verbindung zu einem WEB-Server. „address$" ist eine Zeichenfolge und die Adresse des Servers, zu dem eine Verbindung hergestellt werden soll. Dies kann entweder eine URL (z. B. „api.openweathermap.org") oder eine IP-Adresse (z. B. „192.168.1.111") sein. „port" ist die Nummer des zu verwendenden Ports. Wird zusammen mit WEB TCP CLIENT REQUEST verwendet, um den Server abzufragen. Beachten Sie, dass nur eine CLIENT-Verbindung zulässig ist.
WEB OPEN TCP STREAM Adresse$, Port	Öffnet eine TCP-Client-Verbindung zu einem WEB-Server wie WEB OPEN TCP CLIENT, verbindet jedoch die Empfängerlogik WEB TCP CLIENT STREAM anstelle der Logik für WEB TCP CLIENT REQUEST. „address$" ist eine Zeichenfolge und die Adresse des Servers, zu dem eine Verbindung hergestellt werden soll. Es kann sich entweder um eine URL (z. B. „api.openweathermap.org") oder eine IP-Adresse (z. B. „192.168.1.111") handeln. „port" ist die Nummer des zu verwendenden Ports. Beachten Sie, dass eine CLIENT-Verbindung zulässig ist.
WEB SCAN [array%()]	Sucht nach allen verfügbaren WLAN-Verbindungen. Wenn „array%()" angegeben ist, wird die Ausgabe in einem Longstring gespeichert, andernfalls erfolgt die Ausgabe auf der Konsole. Der Befehl kann unabhängig davon verwendet werden, ob bereits eine Netzwerkverbindung aktiv ist.
WEB TCP CLIENT REQUEST request$, buff%() [,timeout]	Senden Sie eine Anfrage an den mit WEB OPEN TCP CLIENT geöffneten Remote-Server und warten Sie auf eine Antwort. „request$" ist eine Zeichenfolge und stellt die an den Server zu sendende Anfrage dar. „buff%()" ist ein Integer-Array, das die Antwort als LONGSTRING empfängt. Die Größe dieses Puffers begrenzt die vom Server empfangene Datenmenge. „timeout" ist die optionale Zeitüberschreitung in Millisekunden und standardmäßig auf 5000 eingestellt. Wenn die Anfrage zeitlich begrenzt ist, tritt ein Fehler auf, andernfalls werden die empfangenen Daten im LONGSTRING „buff%()" gespeichert. Wenn es sich bei den empfangenen Daten um eine JSON-Zeichenkette handelt, kann die Funktion JSON$() zum Parsen verwendet werden.
WEB-TCP-CLIENT-STREAM command$, buffer%(), readpointer%, writepointer%	Stellt eine Verbindung zu einem zuvor mit WEB OPEN TCP STREAM geöffneten Server her. „command$" ist eine Zeichenfolge und stellt die an den Server zu sendende Anfrage dar. „buffer%()" ist ein Integer-Array, das die laufenden Antworten empfängt und als zirkulärer Puffer für empfangene Bytes fungiert. Die Firmware verwaltet den Parameter „writepointer%", sobald die Daten vom Server eintreffen. „readpointer%" sollte vom Basic-Programm verwaltet werden, da es Daten aus dem Ringpuffer entfernt. Wenn „writepointer%" „readpointer%" einholt, wird „readpointer%" erhöht, um ein Byte voraus zu bleiben, und eingehende Daten gehen verloren. Dieser Befehl ist so konzipiert, dass er mit dem Befehl PLAY STREAM kompatibel ist, um die Implementierung von Streaming-Internet-Audio zu ermöglichen.
WEB CLOSE TCP CLIENT	Schließt die mit WEB OPEN TCP CLIENT geöffnete Verbindung zum Remote-Server. Dies muss erfolgen, bevor ein weiterer Öffnungsversuch unternommen wird.

WEB-TCP-INTERRUPT InterruptSub	Starten Sie den TCP-Server. „InterruptSub" ist die Subroutine, die aufgerufen wird, wenn eine Anfrage an den TCP-Server gestellt wird (d. h. ein Interrupt). Beachten Sie, dass zuerst der Befehl OPTION WIFI und anschließend der Befehl OPTION TCP SERVER PORT verwendet werden muss, um den TCP-Server zu aktivieren.
WEB TCP READ cb%, buff%()	Lesen Sie die Daten aus einer potenziellen TCP-Verbindung „cb%". „buff%()" ist ein Array, um alle Daten aus dieser Verbindung als Longstring zu empfangen. Die Größe dieses Puffers begrenzt die Datenmenge, die vom Remote-Client empfangen wird. Wenn über diese Verbindung nichts empfangen wird, wird eine leere Zeichenfolge zurückgegeben (d. h. `LLEN(buff%())=0`). Wenn Daten empfangen wurden, muss das BASIC-Programm mit einem der Befehle WEB TRANSMIT antworten, um zu reagieren und die Verbindung zu schließen.
WEB TCP SEND cb%, data%() WEB TCP CLOSE cb%	Diese beiden Befehle ermöglichen eine flexiblere Nutzung des TCP-Servers. Im Gegensatz zu WEB TRANSMIT PAGE oder WEB TRANSMIT FILE erstellt WEB TCP SEND keine Kopfzeile und schließt die TCP-Verbindung nach der Übertragung nicht. Es sendet lediglich genau den Inhalt von LONGSTRING data%() und es ist Aufgabe des Basic-Programmierers, die Verbindung zum geeigneten Zeitpunkt zu schließen.
WEB TRANSMIT CODE cb%, nnn%	Sendet eine numerische Antwort an die offene TCP-Verbindung „cb%" und schließt anschließend die Verbindung. Eine typische Verwendung wäre `TRANSMIT CODE cb%, 404,` um anzuzeigen, dass die Seite nicht gefunden wurde.
WEB-ÜBERTRAGUNGSDATEI cb%, Dateiname$, Inhaltstyp$	Erstellt einen HTTP 1.1-Header mit dem angegebenen „content-type$", sendet ihn und sendet anschließend den Inhalt der Datei an die offene TCP-Verbindung cb%. Nach Abschluss wird die Verbindung geschlossen. „content-type$" ist ein MIME-Typ, der als Zeichenfolge ausgedrückt wird. Beispiel: „image/jpeg"
WEB TRANSMIT PAGE cb%, Dateiname$ [,Puffergröße]	Erstellt einen HTTP 1.1-Header, sendet ihn und sendet anschließend den Inhalt der Datei an die offene TCP-Verbindung cb%. Nach Abschluss wird die Verbindung geschlossen. MMBasic ersetzt alle MMBasic-Variablen oder Ausdrücke, die in der Datei innerhalb von geschweiften Klammern definiert sind, z. B. {myvar%}, durch aktuelle Werte. Variablen können einfach, Array-Elemente oder Ausdrücke sein. Eine öffnende geschweifte Klammer kann mithilfe von {{ in die Ausgabe aufgenommen werden. Standardmäßig weist der Befehl einen Puffer in der Größe der Datei + 4096 Byte zu, um die zu übertragende Seite zu erstellen. Wenn die Seite jedoch komplex ist und viele MMBasic-Variablen enthält, deren Text größer ist als der Variablenname, ist es möglich, dass der Puffer nicht groß genug ist. In diesem Fall kann der Benutzer den zusätzlich benötigten Speicherplatz angeben (Standardwert ist 4096, wenn nicht angegeben).
WEB UDP INTERRUPT intname	Richtet eine BASIC-Interrupt-Routine ein, die bei jedem Empfang eines UDP-Datagramms ausgelöst wird. Der Inhalt wird in MM.MESSAGE$ gespeichert. Die IP-Adresse des Absenders wird in MM.ADDRESS$ gespeichert.
WEB UDP SEND addr$, port, data	Wird verwendet, um ein Datagramm an einen entfernten Empfänger zu senden. In diesem Fall muss die IP-Adresse angegeben werden, die entweder eine numerische Adresse (z. B. „192.168.1.147") oder eine normale Textadresse (z. B. „google.com") sein kann. Die Portnummer des Empfängers muss ebenfalls angegeben werden, ebenso wie die Nachricht selbst. Der Befehl SEND kann als Antwort auf eine eingehende Nachricht oder eigenständig verwendet werden.
WS2812 Typ, Pin, Nr., Wert%[()]	Dieser Befehl steuert einen oder mehrere WS2812-LED-Chips, die an „Pin" angeschlossen sind. Beachten Sie, dass der Pin vor Verwendung dieses Befehls auf einen digitalen Ausgang eingestellt werden muss. „type" ist ein einzelnes Zeichen, das den Typ des angesteuerten Chips angibt: O = Original WS2812 B = WS2812B

	S = SK6812 W = SK6812W (RGBW) „nbr“ ist die Anzahl der LEDs in der Kette (1 bis 256). Das Array „value%()“ sollte ein Integer-Array sein, dessen Größe genau der Anzahl der anzusteuernden LEDs entspricht. Bei den ersten drei Varianten sollte jedes Element im Array die Farbe im normalen RGB888-Format enthalten (d. h. 0 bis &HFFFFFF). Für Typ W verwenden Sie einen RGBW-Wert (0-&HFFFFFFFF). Wenn nur eine LED angeschlossen ist, sollte für „value%“ eine einzelne Ganzzahl verwendet werden (d. h. kein Array).
XMODEM SEND oder XMODEM SEND file$ oder XMODEM RECEIVE oder XMODEM RECEIVE Datei$ oder XMODEM CRUNCH	Überträgt ein BASIC-Programm unter Verwendung des XModem-Protokolls zu oder von einem Remote-Computer. Die Übertragung erfolgt über die USB-Konsolenverbindung. XMODEM SEND sendet das aktuelle Programm, das im Programmspeicher des PicoMite gespeichert ist, an das Remote-Gerät. XMODEM RECEIVE empfängt ein vom Remote-Gerät gesendetes Programm und speichert es im Programmspeicher des PicoMite, wobei das dort aktuell gespeicherte Programm überschrieben wird. In beiden Fällen können Sie auch „file$“ angeben, wodurch die Daten in eine Datei auf dem Flash-Dateisystem oder der SD-Karte übertragen werden. Wenn die Datei bereits vorhanden ist, wird sie beim Empfang einer Datei überschrieben. Beachten Sie, dass die Daten im RAM gepuffert werden, wodurch die maximale Übertragungsgröße begrenzt ist. Dieser Befehl erstellt auch eine Sicherungskopie des Programms im Flash-Speicher, die automatisch abgerufen wird, wenn die CPU zurückgesetzt wird oder die Stromversorgung ausfällt. Die Option CRUNCH funktioniert wie RECEIVE, entfernt jedoch vor dem Speichern alle Kommentare, Leerzeilen und unnötigen Leerzeichen aus dem Programm. Dies kann bei großen Programmen verwendet werden, damit sie in den begrenzten Speicher passen. SEND, RECEIVE und CRUNCH können mit S, R und C abgekürzt werden. Das XModem-Protokoll erfordert ein kooperierendes Softwareprogramm, das auf dem Remote-Computer ausgeführt wird und mit dessen serieller Schnittstelle verbunden ist. Es wurde unter Windows mit Tera Term getestet, dessen Verwendung empfohlen wird. Nachdem Sie den Befehl XMODEM in MMBasic ausgeführt haben, wählen Sie: Datei -> Übertragung -> XMODEM -> Empfangen/Senden aus dem Tera Term-Menü, um die Übertragung zu starten. Der Start der Übertragung kann bis zu 15 Sekunden dauern. Wenn der XMODEM-Befehl die Kommunikation nicht herstellen kann, kehrt er nach 60 Sekunden zur MMBasic-Eingabeaufforderung zurück und lässt den Programmspeicher unverändert. Laden Sie Tera Term von http://ttssh2.sourceforge.jp/ herunter.
YMODEM SEND oder YMODEM SEND file$ oder YMODEM RECEIVE oder YMODEM RECEIVE Datei$ oder YMODEM CRUNCH	RP2350 NUR NICHT-USB-VERSIONEN Dieser Befehl ahmt XMODEM nach, überträgt Dateien jedoch viel schneller (>10x) über eine serielle CDC-Verbindung. YMODEM unterscheidet sich von XMODEM dadurch, dass die Nachricht den Dateinamen enthält. Das bedeutet, dass beim Senden einer Datei vom Pico an einen Computer das empfangende Programm automatisch denselben Namen verwendet, den Sie beim Senden angegeben haben. Falls die Datei bereits vorhanden ist, erhöht der Empfänger automatisch die Versionsnummer im Namen der empfangenen Datei. fred1.bas, fred2.bas usw. Beim Senden aus dem Speicher mit YMODEM S gibt es zwei Möglichkeiten. Wenn die Datei vom Laufwerk A: oder B: geladen wurde, enthält sie einen versteckten Dateinamen, der verwendet wird. Bei einer mit dem Editor erstellten oder automatisch gespeicherten Datei erhält die empfangene Datei automatisch den Namen FILEn.DAT. Es liegt in der Verantwortung des empfangenden Programms, anzugeben, wo die Datei gespeichert werden soll

	gespeichert wird. Bei Teraterm ist dies eine Option, die Sie auf dem Bildschirm „Allgemeine Einstellungen“ unter „Dateiübertragungsordner“ festlegen können. Wenn Sie eine Datei auf dem Pico empfangen, können Sie den Namen und das Verzeichnis ganz normal frei wählen Hinweis: YMODEM reagiert sehr empfindlich auf die Qualität der Verbindung und die USB-Implementierung auf dem Host. Wenn es bei Ihnen nicht funktioniert, verwenden Sie weiterhin XMODEM

Funktionen

Beachten Sie, dass die Funktionen im Zusammenhang mit Kommunikationsfunktionen (I^2C, 1-Wire und SPI) hier nicht aufgeführt sind, sondern in den Anhängen am Ende dieses Dokuments beschrieben werden.

Eckige Klammern weisen darauf hin, dass der Parameter oder die Zeichen optional sind.

ABS(Zahl)	Gibt den absoluten Wert des Arguments „Zahl" zurück (d. h. alle negativen Vorzeichen werden entfernt und eine positive Zahl zurückgegeben).
ACOS(Zahl)	Gibt den inversen Kosinus des Arguments „Zahl" in Radianten zurück.
ASC(Zeichenfolge$)	Gibt den ASCII-Code (d. h. den Byte-Wert) für den ersten Buchstaben in „Zeichenfolge$" zurück.
ASIN(Zahl)	Gibt den inversen Sinuswert des Arguments „number" in Radianten zurück.
ATN(Zahl)	Gibt den Arkustangens des Arguments „number" in Radianten zurück.
ATAN2(y, x)	Gibt den Arkustangens der beiden Zahlen x und y als Winkel in Radianten zurück. Dies ähnelt der Berechnung des Arkustangens von y / x, mit dem Unterschied, dass die Vorzeichen beider Argumente zur Bestimmung des Quadranten des Ergebnisses herangezogen werden.
BIN$(Zahl [, Zeichen])	Gibt eine Zeichenfolge zurück, die den binären Wert (Basis 2) für „number" angibt. „chars" ist optional und gibt die Anzahl der Zeichen in der Zeichenfolge an, wobei Null als führendes Auffüllzeichen verwendet wird.
BIN2STR$(Typ, Wert [,BIG])	Gibt eine Zeichenfolge zurück, die die binäre Darstellung von „value" enthält. „type" kann sein: INT64 vorzeichenbehaftete 64-Bit-Ganzzahl, konvertiert in eine 8-Byte-Zeichenfolge UINT64 vorzeichenlose 64-Bit-Ganzzahl, konvertiert in eine 8-Byte-Zeichenkette INT32 vorzeichenbehaftete 32-Bit-Ganzzahl, konvertiert in eine 4-Byte-Zeichenfolge UINT32 vorzeichenlose 32-Bit-Ganzzahl, konvertiert in eine 4-Byte-Zeichenkette INT16 vorzeichenbehaftete 16-Bit-Ganzzahl, konvertiert in eine 2-Byte-Zeichenkette UINT16 vorzeichenlose 16-Bit-Ganzzahl, konvertiert in eine 2-Byte-Zeichenkette INT8 vorzeichenbehaftete 8-Bit-Ganzzahl, konvertiert in eine 1-Byte-Zeichenkette UINT8 vorzeichenlose 8-Bit-Ganzzahl, konvertiert in eine 1-Byte-Zeichenkette SINGLE Gleitkommazahl mit einfacher Genauigkeit, konvertiert in eine 4-Byte-Zeichenkette DOUBLE Doppelt genaue Gleitkommazahl, konvertiert in eine 8-Byte-Zeichenkette Standardmäßig enthält die Zeichenfolge die Zahl im Little-Endian-Format (d. h. das niedrigstwertige Byte steht an erster Stelle in der Zeichenfolge). Wenn Sie den dritten Parameter auf „BIG" setzen, wird die Zeichenfolge im Big-Endian-Format zurückgegeben (d. h. das höchstwertige Byte steht an erster Stelle in der Zeichenfolge). Bei der Konvertierung von Ganzzahlen wird ein Fehler ausgegeben, wenn der „Wert" nicht in den „Typ" passt (z. B. bei dem Versuch, den Wert 400 in einem INT8 zu speichern). Diese Funktion erleichtert die Vorbereitung von Daten für eine effiziente binäre Datei-E/A oder die Vorbereitung von Zahlen für die Ausgabe an Sensoren und das Speichern im Flash-Speicher. Siehe auch die Funktion STR2BIN
BIT(var%, bitno)	Gibt den Wert eines bestimmten Bits (0-63) in einer Integer-Variablen (0 oder 1) zurück. Siehe auch den Befehl BIT

BOUND(array() [,dimension]	Gibt die Obergrenze des Arrays für die angeforderte Dimension zurück. Die Dimension ist standardmäßig auf eins gesetzt, wenn nichts anderes angegeben ist. Bei Angabe eines Dimensionswerts von 0 wird der aktuelle Wert von OPTION BASE zurückgegeben. Nicht verwendete Dimensionen geben den Wert Null zurück. Beispiel: DIM myarray(44,45) BOUND(myarray(),2) gibt 45 zurück
BYTE(var$, byteno)	Gibt den ganzzahligen Wert eines bestimmten Bytes in einer Zeichenfolge zurück (0-255). Dies entspricht ASC(MID$(var$,byteno,1)), arbeitet jedoch wesentlich schneller. Siehe auch den Befehl BYTE
CALL(userfunname$, [,userfunparameters,..])	Dies ist eine effiziente Methode, um benutzerdefinierte Funktionen programmgesteuert aufzurufen. (Siehe auch den Befehl CALL). In vielen Fällen kann damit komplexe SELECT- und IF THEN ELSEIF ENDIF-Klauseln vermieden werden, und die Verarbeitung erfolgt wesentlich effizienter. „userfunname$" kann eine beliebige Zeichenfolge, Variable oder Funktion sein, die zu einem normalen Benutzernamen (kein integrierter Befehl) aufgelöst wird. „userfunparameters" sind dieselben Parameter, die zum direkten Aufruf der Funktion verwendet würden. Eine typische Anwendung für diesen Befehl wäre das Schreiben eines beliebigen Emulators, bei dem eine von vielen Funktionen in Abhängigkeit von einer bestimmten Variablen aufgerufen werden soll. Er bietet auch eine Methode, um einen Funktionsnamen als Variable an eine andere Subroutine oder Funktion zu übergeben.
CHOICE(Bedingung, AusdruckWennWahr, AusdruckWennFalsch)	Mit dieser Funktion können Sie einfache Entweder-oder-Auswahlen effizienter und schneller durchführen als mit IF THEN ELSE ENDIF-Klauseln. Die Bedingung ist alles, was zu einem Wert ungleich Null (wahr) oder Null (falsch) führt. Die Ausdrücke sind alles, was Sie normalerweise einer Variablen zuweisen oder in einem Befehl verwenden können, und können Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen sein. Beispiele: PRINT CHOICE(1, „hello", „bye") gibt „Hello" aus. PRINT CHOICE (0, „hello", „bye") gibt „Bye" aus. a=1 : b=1 : PRINT CHOICE (a=b, 4, 5) gibt 4 aus.
CHR$(Zahl)	Gibt eine einstellige Zeichenkette zurück, die aus dem Zeichen besteht, das dem durch das Argument „number" angegebenen ASCII-Code (d. h. Byte-Wert) entspricht.
CINT(Zahl)	Rundet Zahlen mit Bruchteilen auf oder ab auf die nächste ganze Zahl oder ganze Zahl. Beispiel: wird 45,47 auf 45 gerundet 45,57 wird auf 46 gerundet -34,45 wird auf -34 gerundet -34,55 wird auf -35 gerundet Siehe auch INT() und FIX().
COS(Zahl)	Gibt den Kosinus des Arguments „Zahl" in Radianten zurück.
CWD$	Gibt das aktuelle Arbeitsverzeichnis auf dem Flash-Dateisystem oder der SD-Karte zurück. Ungültig für das exFAT-Format. Das Format lautet: `A:/dir1/dir2`.
DATE$	Gibt das aktuelle Datum basierend auf der internen Uhr von MMBasic als Zeichenfolge im Format „TT-MM-JJJJ" zurück. Beispiel: „28-07-2012".

	Die interne Uhr/der interne Kalender erfasst die Uhrzeit und das Datum einschließlich Schaltjahren. Um das Datum festzulegen, verwenden Sie den Befehl DATE$ =.
DATETIME$(n)	Gibt das Datum und die Uhrzeit zurück, die der Epoche Nummer „n" entsprechen (Anzahl der Sekunden, die seit Mitternacht GMT am 1. Januar 1970 vergangen sind). Das Format der zurückgegebenen Zeichenfolge lautet „dd-mm-yyyy hh:mm:ss". Verwenden Sie den Text NOW, um die aktuelle Datums- und Zeitzeichenfolge zu erhalten, d. h. DATETIME$(NOW)
DAY$(date$)	Gibt den Wochentag für ein bestimmtes Datum als Zeichenfolge zurück. Zum Beispiel „Montag", „Dienstag" usw. „date$" ist eine Zeichenfolge, deren Format DD-MM-YY, DD-MM-YYYY oder YYYY-MM-DD lauten kann. Sie können auch NOW verwenden, um den Tag des aktuellen Datums zu erhalten, z. B. PRINT DAY$(NOW)
DEG(Radiant)	Konvertiert „Radiant" in Grad.
DEVICE(GAMEPAD channel, funct)	Gibt Daten von einem USB-PS3- oder PS4-Controller zurück. „funct" ist ein 1- oder 2-stelliger Code, der die zurückzugebenden Informationen wie folgt angibt: LX t die Position der x-Achse des analogen linken Joysticks LY iert die Position der analogen linken Joystick-Y-Achse RX die Position der x-Achse des analogen rechten Joysticks RY die Position der y-Achse des analogen rechten Joysticks GX der Messwert des X-Achsen-Gyroskops (sofern unterstützt) GY der Messwert des Y-Achsen-Gyroskops (sofern unterstützt) GZ der Messwert des Z-Achsen-Gyroskops (sofern unterstützt) AX der Messwert des Beschleunigungsmessers der X-Achse (sofern unterstützt) AY der Messwert des Beschleunigungsmessers der Y-Achse (sofern unterstützt) AZ der Messwert des Beschleunigungsmessers der Z-Achse (sofern unterstützt) L die Position der analogen linken Taste R t die Position der analogen rechten Taste B eine Bitmap des Status aller Tasten. Ein Bit wird auf 1 gesetzt, wenn die Taste gedrückt wird. T der ID-Code des Controllers Die Tasten-Bitmap sieht wie folgt aus: BIT 0 -Taste R/R1 BIT 1 Taste Start/Optionen BIT 2 Taste Home BIT 3 Taste Auswählen/Teilen BIT 4 Taste L/L1 BIT 5 Taste Cursor nach unten BIT 6 Taste Cursor nach rechts BIT 7 Taste Cursor nach oben BIT 8 Taste Cursor links BIT 9 Rechte Schultertaste 2/R2 BIT 10 Taste x/Dreieck BIT 11 Taste a/Kreis BIT 12 Taste y/Quadrat BIT 13 Taste b/Kreuz BIT 14 Linker Schulterknopf 2/L2 BIT 15 Touchpad
GERÄT(MAUS-Kanal, Funktion)	Gibt Daten von einer über „Kanal" angeschlossenen Maus zurück. Eine PS2-Maus wird immer Kanal 2 zugewiesen. Normalerweise wird auch eine USB-Maus Kanal 2 zugewiesen, dies kann jedoch variieren. Weitere Informationen finden Sie unter MM.INFO(USB n).

	„funct“ ist ein einstelliger Code, der die zurückzugebenden Informationen wie folgt angibt: X die X-Koordinate (0 bis MM.HRES-1) Y die Y-Koordinate (0 bis MM.VRES-1) L den Status der linken Maustaste R t der Status der rechten Maustaste M: Status der mittleren Maustaste (Scrollrad) D 1, wenn ein Doppelklick mit der linken Maustaste erfolgt ist W gibt die Delta-Position des Rads seit dem letzten Aufruf an (PS2-Maus oder USB-Maus mit eingestellter OPTION MOUSE SENSITIVITY) B t den Status aller Tasten als Bitmap (nur USB-Maus)
DEVICE(WII [CLASSIC] funct)	Gibt Daten von einem Wii Classic Controller zurück. „funct” ist ein 1- oder 2-Buchstaben-Code, der die zurückzugebenden Informationen wie folgt angibt: LX die Position der x-Achse des analogen linken Joysticks LY die Position der y-Achse des analogen linken Joysticks RX die Position der analogen rechten Joystick-X-Achse RY die Position der analogen rechten Joystick-Y-Achse L die Position des analogen linken Knopfes R die Position der analogen rechten Taste B eine Bitmap des Status aller Tasten. Ein Bit wird auf 1 gesetzt, wenn die Taste gedrückt wird. T der ID-Code des Controllers – sollte hex &HA4200101 lauten Die Tasten-Bitmap sieht wie folgt aus: BIT 0 Taste R BIT 1 Taste Start BIT 2 Taste Home BIT 3 Taste „Auswählen“ BIT 4 Taste L BIT 5 Taste „Cursor nach unten“ BIT 6Taste Rechts-Cursor BIT 7 Taste Cursor nach oben BIT 8 Taste links Cursor BIT 9 Taste ZR BIT 10 Taste x BIT 11 Taste a BIT 12 Taste y BIT 13 Taste b BIT 14 Taste ZL
DEVICE(WII NUNCHUCK funct)	Gibt Daten von einem Wii Nunchuck-Controller zurück. „funct” ist ein 1- oder 2-stelliger Code, der die zurückzugebenden Informationen wie folgt angibt: AX die Beschleunigung der x-Achse AY die Beschleunigung der y- Achse AZ die Beschleunigung der Z-Achse JX die Position der X-Achse des Joysticks JY die Position der Y-Achse des Joysticks C der Status der C-Taste Z der Status der Z-Taste

	T der ID-Code des Controllers – sollte hex &HA4200000 sein

DIR$(fspec, type) oder
DIR$(fspec)
oder
DIR$( )
	Durchsucht das Standard-Flash-Dateisystem oder die SD-Karte nach Dateien und gibt die Namen der gefundenen Einträge zurück.
	„fspec" ist eine Dateispezifikation mit Platzhaltern, die denen des Befehls FILES entsprechen. Beispielsweise gibt „*.*" alle Einträge zurück, „*.TXT" gibt Textdateien zurück. Beachten Sie, dass der Platzhalter *.* keine Dateien oder Ordner ohne Erweiterung findet.
	„type" ist der Typ des zurückzugebenden Eintrags und kann einer der folgenden sein:
	ALL Sucht nach allen Dateien und Verzeichnissen. DIR Nur nach Verzeichnissen suchen
	FILE Nur nach Dateien suchen (Standard, wenn „type" nicht angegeben ist)
	Die Funktion gibt den ersten gefundenen Eintrag zurück. Um nachfolgende Einträge abzurufen, verwenden Sie die Funktion ohne Argumente, d. h. DIR$( ). Die Rückgabe einer leeren Zeichenfolge bedeutet, dass keine weiteren Einträge mehr abgerufen werden können.
	In diesem Beispiel werden alle Dateien in einem Verzeichnis ausgegeben:
	`f$ = DIR$("*.*", FILE) DO`
	`WHILE f$ <> ""`
	`PRINT f$`
	`f$ = DIR$() LOOP`
	Sie müssen vor dem Aufruf dieses Befehls in das gewünschte Verzeichnis wechseln.

DISTANCE(trigger, echo) oder DISTANCE(trig-echo)	Messen Sie die Entfernung zu einem Ziel mit dem Ultraschall-Entfernungssensor HC-SR04. Vierpolige Sensoren verfügen über separate Trigger- und Echo-Anschlüsse. „Trigger" ist der I/O-Pin, der mit dem „Trig"-Eingang des Sensors verbunden ist, und „Echo" ist der Pin, der mit dem „Echo"-Ausgang des Sensors verbunden ist. Dreipolige Sensoren haben einen kombinierten Trigger- und Echoanschluss. In diesem Fall müssen Sie nur einen I/O-Pin für die Schnittstelle zum Sensor angeben. Beachten Sie, dass der HC-SR04 ein 5-V-Gerät ist, sodass bei Pico-Prozessoren (RP2040) eine Pegelumsetzung erforderlich ist, bei Pico-2-Prozessoren (RP2350) jedoch nicht. Die I/O-Pins werden durch diese Funktion automatisch konfiguriert, und es können mehrere Sensoren an verschiedenen I/O-Pins verwendet werden. Der zurückgegebene Wert ist die Entfernung zum Ziel in Zentimetern oder -1, wenn kein Ziel erkannt wurde, oder -2, wenn ein Fehler aufgetreten ist (z. B. Sensor nicht angeschlossen).
EOF([#]fnbr)	Gibt „true" zurück, wenn die zuvor im Flash-Dateisystem oder auf der SD-Karte für INPUT geöffnete Datei mit der Dateinummer „#fnbr" am Ende der Datei positioniert ist. Das # ist optional. Siehe auch die Befehle OPEN, INPUT und LINE INPUT sowie die Funktion INPUT$.
EPOCH(DATETIME$)	Gibt die Epochenzahl (Anzahl der Sekunden, die seit Mitternacht GMT am 1. Januar 1970 vergangen sind) für die angegebene DATETIME$-Zeichenkette zurück. Das Format für DATETIME$ ist „dd-mm-yyyy hh:mm:ss", „dd-mm-yy hh:mm:ss" oder „yyyy-mm-dd hh:mm:ss". Verwenden Sie NOW, um die Epochenzahl für das aktuelle Datum und die aktuelle Uhrzeit zu erhalten, d. h. PRINT EPOCH(NOW)
EVAL(string$)	Wertet „string$" wie einen BASIC-Ausdruck aus und gibt das Ergebnis zurück. „string$" kann eine Konstante, eine Variable oder ein String-Ausdruck sein. Der Ausdruck kann alle Operatoren, Funktionen, Variablen, Unterprogramme usw. verwenden, die zum Zeitpunkt der Ausführung bekannt sind. Der zurückgegebene Wert ist je nach Ergebnis der Auswertung eine Ganzzahl, eine Gleitkommazahl oder ein String. Beispiel: S$ = "COS(RAD(30)) * 100" : PRINT EVAL(S$) Anzeige: 86,6025
EXP(Zahl)	Gibt den Exponentialwert von „Zahl" zurück, d. h. e^x, wobei x „Zahl" ist.

FIELD$(string1, nbr, string2 [, string3])	Gibt ein bestimmtes Feld in einer Zeichenfolge zurück, wobei die Felder durch Trennzeichen getrennt sind. Beachten Sie, dass ein Leerzeichen nicht als Trennzeichen verwendet werden kann. „nbr" ist das zurückzugebende Feld (das erste ist nbr 1). „string1" ist die zu suchende Zeichenfolge und „string2" ist eine Zeichenfolge, die die Trennzeichen enthält (es können mehrere verwendet werden). Das Leerzeichen darf nicht als Trennzeichen verwendet werden. „string3" ist optional und enthält, falls angegeben, Zeichen, die zum Zitieren von Text in „string1" verwendet werden (d. h., zitierter Text wird nicht nach einem Trennzeichen durchsucht). Beispiel: S$ = "foo, boo, zoo, doo" r$ = FIELD$(s$, 2, ",") ergibt r$ = „boo". Während: s$ = „foo, 'boo, zoo', doo" r$ = FIELD$(s$, 2, ",", """) ergibt r$ = „boo, zoo".
FIX(Zahl)	Eine Zahl auf eine ganze Zahl kürzen, indem der Dezimalpunkt und alle Zeichen rechts vom Dezimalpunkt entfernt werden. Beispielsweise ergibt 9,89 den Wert 9 und -2,11 den Wert -2. Der wesentliche Unterschied zwischen FIX() und INT() besteht darin, dass FIX() eine echte Ganzzahlfunktion bereitstellt (d. h. gibt bei negativen Zahlen nicht wie INT() die nächstkleinere Zahl zurück). Dieses Verhalten dient der Kompatibilität mit Microsoft. Siehe auch CINT().
FLAG(n%)	Gibt den Wert (0 oder 1) des Bits n% (0-63) im Systemflagregister zurück. Siehe auch MM.FLAGS und die Befehle FLAG und FLAGS.
FORMAT$(nbr [, fmt$])	Gibt eine Zeichenfolge zurück, die „nbr" gemäß den Angaben in der Zeichenfolge „fmt$" formatiert darstellt. Die Formatspezifikation beginnt mit einem %-Zeichen und endet mit einem Buchstaben. Alles außerhalb dieser Konstruktion wird unverändert in die Ausgabe kopiert. Die Struktur einer Formatspezifikation lautet: % [Flags] [Breite] [.Genauigkeit] Typ Dabei können „Flags" sein: - Den Wert innerhalb einer bestimmten Feldbreite linksbündig ausrichten 0 Verwenden Sie 0 als Füllzeichen anstelle von Leerzeichen + Erzwingt die Anzeige des Pluszeichens für positive Zahlen Leerzeichen Bewirkt, dass bei positiven Werten ein Leerzeichen anstelle des Vorzeichens angezeigt wird. Negative Werte zeigen weiterhin das – Zeichen an. „width" ist die Mindestanzahl der auszugebenden Zeichen. Bei einer geringeren Anzahl wird die Zahl aufgefüllt, bei einer höheren Anzahl wird die Breite erweitert. „precision" gibt die Anzahl der zu generierenden Dezimalstellen mit einem e- oder f-Typ oder die maximale Anzahl der zu generierenden signifikanten Stellen mit einem g-Typ an und ist standardmäßig auf 4 Stellen eingestellt. Wenn angegeben, muss der Genauigkeit ein Punkt (.) vorangestellt werden. „type" kann einer der folgenden Werte sein: g Formatiert die Zahl automatisch für die beste Darstellung. f Formatiert die Zahl mit Dezimalpunkt und nachfolgenden Stellen. e Formatiert die Zahl im Exponentialformat. Wenn Großbuchstaben G oder F verwendet werden, wird für die Exponentialausgabe ein Großbuchstabe E verwendet. Wenn die Formatangabe nicht angegeben ist, wird „%g" angenommen. Beispiele: format$(45) gibt 45 zurück format$(45, „%g") gibt 45 zurück

GETSCANLINE	NUR VGA UND HDMI Dies meldet die Zeile, die derzeit auf dem VGA/HDMI-Monitor im Bereich von 0 bis 525 gezeichnet wird. Dies ist unabhängig vom aktuellen MODUS. Durch die Verwendung dieser Funktion zur zeitlichen Abstimmung von Aktualisierungen auf dem Bildschirm können Timing-Effekte vermieden werden, die durch Aktualisierungen während der Bildschirmaktualisierung verursacht werden. Die erste sichtbare Zeile gibt den Wert 0 zurück. Alle Zeilennummern über 479 befinden sich in der Bildausblendungsperiode.
GPS() GPS(HÖHE) GPS(DATUM) GPS(DOP) GPS(FIX) GPS(GEOID) GPS(LATITUDE) GPS(LONGITUDE) GPS(SATELLITEN) GPS(GESCHWINDIGKEIT) GPS(ZEIT) GPS(SPUR) GPS(GÜLTIG)	Die GPS-Funktionen dienen dazu, Daten aus einem als GPS geöffneten seriellen Kommunikationskanal zurückzugeben. Die Funktion GPS(VALID) sollte vor der Verwendung einer dieser Funktionen überprüft werden, um sicherzustellen, dass der zurückgegebene Wert gültig ist. Gibt die aktuelle Höhe zurück (wenn der Satz GGA aktiviert ist). Gibt die normale Datumszeichenfolge zurück, korrigiert um die Ortszeit, z. B. „12-01-2020“. Gibt den DOP-Wert (Dilution of Precision) zurück (wenn der Satz GGA aktiviert ist). Gibt einen Wert ungleich Null (wahr) zurück, wenn das GPS eine ausreichende Anzahl von Satelliten erfasst hat und gültige Daten liefert. Gibt den Geoid-Ellipsoid-Abstand zurück (wenn der Satz GGA aktiviert ist). Gibt den Breitengrad in Grad als Fließkommazahl zurück, Werte südlich des Äquators sind negativ Gibt die Länge in Grad als Fließkommazahl zurück, Werte westlich des Meridians sind negativ. Gibt die Anzahl der sichtbaren Satelliten zurück (wenn der Satz GGA aktiviert ist). Gibt die Geschwindigkeit über Grund in Knoten als Fließkommazahl zurück. Gibt die normale Zeitangabe korrigiert um die Ortszeit zurück, z. B. „12:09:33“. Gibt die Bodenstrecke (Grad wahr) als Gleitkommazahl zurück. Gibt zurück: 0 = ungültige Daten, 1 = gültige Daten
HEX$(Zahl [, Zeichen])	Gibt eine Zeichenfolge zurück, die den Hexadezimalwert (Basis 16) für die „Zahl“ angibt. „chars“ ist optional und gibt die Anzahl der Zeichen in der Zeichenfolge an, wobei Null als führendes Auffüllzeichen verwendet wird.
INKEY$	Überprüft den Konsoleneingabepuffer und entfernt, wenn ein oder mehrere Zeichen in der Warteschlange stehen, das erste Zeichen und gibt es als einzelnes Zeichen in einer Zeichenkette zurück. Wenn der Eingabepuffer leer ist, gibt diese Funktion sofort eine leere Zeichenkette (d. h. „“) zurück.
INPUT$(nbr, [#]fnbr)	Gibt eine Zeichenkette zurück, die aus „nbr“ Zeichen besteht, die aus einer Datei oder einem seriellen Kommunikationsport gelesen wurden, der als „fnbr“ geöffnet wurde. Diese Funktion gibt so viele Zeichen zurück, wie sich in der Datei oder im Empfangspuffer befinden, bis zu „nbr“. Wenn keine Zeichen verfügbar sind, wird sofort eine leere Zeichenkette zurückgegeben. Es kann #0 verwendet werden, was sich auf den Eingabepuffer der Konsole bezieht. Das # ist optional. Siehe auch den Befehl OPEN.

INSTR([Startposition,] gesuchter-String$, String-Muster$ [,Größe])	Gibt die Position zurück, an der „string-pattern$" in „string-searched$" vorkommt, beginnend bei „start-position". Wenn „start-position" nicht angegeben ist, wird standardmäßig 1 verwendet. Sowohl die zurückgegebene Position als auch „start-position" verwenden 1 für das erste Zeichen, 2 für das zweite usw. Die Funktion gibt Null zurück, wenn „string-pattern$" nicht gefunden wird. Wenn der optionale Parameter „size" angegeben ist, wird „string-pattern" als regulärer Ausdruck behandelt. Details finden Sie in *Anhang E*.
INT(Zahl)	Kürzt einen Ausdruck auf die nächste ganze Zahl, die kleiner oder gleich dem Argument ist. Beispielsweise gibt 9,89 den Wert 9 und -2,11 den Wert -3 zurück. Dieses Verhalten dient der Kompatibilität mit Microsoft. Die Funktion FIX() bietet eine echte Integer-Funktion. Siehe auch CINT() .
JSON$(array%(), string$)	Gibt eine Zeichenfolge zurück, die ein bestimmtes Element aus der JSON-Eingabe darstellt, die im Longstring-Array %() gespeichert ist. Beachten Sie, dass viele JSON-Datensätze sehr groß sind und möglicherweise zu groß sind, um mit dem verfügbaren Speicherplatz analysiert zu werden. Beispiele (entnommen aus api.openweathermap.org): JSON$(a%(), „name") JSON$(a%(), „coord.lat") JSON$(a%(), „weather[0].description") JSON$(a%(),"list[4].weather[0].description
KEYDOWN(n)	Gibt den dezimalen ASCII-Wert der derzeit gedrückten Taste der USB-Tastatur zurück oder Null, wenn keine Taste gedrückt ist. Die Dezimalwerte für die Funktions- und Pfeiltasten sind in Anhang I aufgeführt. Diese Funktion meldet mehrere gleichzeitige Tastendrücke, wobei der Parameter „n" die Nummer des zu meldenden Tastendrucks angibt. KEYDOWN(0) gibt die Anzahl der gedrückten Tasten zurück. Wenn beispielsweise „c", „g" und „p" gleichzeitig gedrückt werden, gibt KEYDOWN(0) 3 zurück, KEYDOWN(1) gibt 99 zurück, KEYDOWN(2) gibt 103 zurück usw. Die Tasten müssen nicht gleichzeitig gedrückt werden und werden in der Reihenfolge ihrer Betätigung gemeldet. Wenn Sie einen Finger von einer Taste nehmen, wird die nächste gedrückte Taste auf Platz 1 verschoben. Die erste Taste („n" = 1) wird in den Tastaturpuffer eingegeben (zugänglich über INKEY$), während auf die Tasten 2 bis 6 nur über diese Funktion zugegriffen werden kann. Durch Verwendung dieser Funktion wird der Konsoleneingabepuffer gelöscht. KEYDOWN(7) gibt alle gedrückten Modifikatortasten aus. Diese Tasten werden nicht zur Zählung in keydown(0) hinzugefügt. Der Rückgabewert ist eine Bitmaske wie folgt: lalt = 1, lctrl = 2, lgui = 4, lshift = 8, ralt = 16, rctrl = 32, rgui = 64, rshift = 128 KEYDOWN(8) gibt den aktuellen Status der Sperrtasten an. Diese Tasten werden nicht zur Zählung in keydown(0) hinzugefügt. Der Rückgabewert ist eine Bitmaske wie folgt: caps_lock = 1, num_lock = 2, scroll_lock = 4 Beachten Sie, dass einige Tastaturen die Anzahl der aktiven Tasten, die sie melden können, begrenzen.
LCASE$(string$)	Gibt „string$" in Kleinbuchstaben zurück.
LCOMPARE(array1%(), array2%())	Vergleichen Sie den Inhalt der beiden langen String-Variablen „array1%()" und „array2%()". Die Rückgabe ist eine Ganzzahl und beträgt -1, wenn „array1%()" kleiner als „array2%()" ist. Sie beträgt Null, wenn Länge und Inhalt gleich sind, und +1, wenn „array1%()" größer als „array2%()" ist. Der Vergleich verwendet den ASCII-Zeichensatz und unterscheidet zwischen Groß- und Kleinschreibung.

LEFT$(string$, nbr)	Gibt eine Teilzeichenfolge von „string$“ mit „nbr“ Zeichen vom Anfang (von links) der Zeichenfolge zurück.
LEN(string$)	Gibt die Anzahl der Zeichen in „string$“ zurück.
LGETBYTE(array%(), n)	Gibt den numerischen Wert des n-ten Bytes in der LONGSTRING zurück, die in „array%()” gespeichert ist. Diese Funktion berücksichtigt die Einstellung von OPTION BASE bei der Bestimmung des zurückzugebenden Bytes.
LGETSTR$(array%(), start, length)	Gibt einen Teil einer langen Zeichenfolge zurück, die in „array%()” als normale MMBasic-Zeichenfolge gespeichert ist. Die Parameter start und length definieren den Teil der Zeichenfolge, der zurückgegeben werden soll.
LINSTR(array%(), search$ [,start] [,size]))	Gibt die Position einer Suchzeichenfolge in einer langen Zeichenfolge zurück. Der zurückgegebene Wert ist eine Ganzzahl und beträgt Null, wenn die Teilzeichenfolge nicht gefunden werden kann. „array%()” ist die zu durchsuchende Zeichenfolge und muss eine lange Zeichenfolgenvariable sein. „search$” ist die zu suchende Teilzeichenfolge und muss eine normale MMBasic-Zeichenfolge oder ein Ausdruck sein (keine lange Zeichenfolge). Bei der Suche wird zwischen Groß- und Kleinschreibung unterschieden. Normalerweise beginnt die Suche beim ersten Zeichen in „array%()”, aber mit dem optionalen dritten Parameter kann die Startposition der Suche angegeben werden. Wenn der optionale Parameter „size“ angegeben wird, wird „search$“ als regulärer Ausdruck behandelt. Details finden Sie in *Anhang E*.
LLEN(array%())	Gibt die Länge einer langen Zeichenfolge zurück, die in „array%()“ gespeichert ist.
LINPUT(array%(),fnbr,nbr)	Dies liest „nbr“ Bytes aus einer als „fnbr“ geöffneten Datei in den LONGSTRING „array%()“. Die Funktion gibt die Anzahl der tatsächlich gelesenen Bytes zurück. Wenn Sie sich also dem Ende der Datei nähern, kann die Anzahl geringer sein als die angeforderte Anzahl. Dies funktioniert nur mit Datei-I/O und nicht mit serieller oder Konsolen-I/O.
LOC([#]fnbr)	Für eine Datei im Flash-Dateisystem oder auf einer SD-Karte, die als „fnbr” geöffnet ist, gibt diese Funktion die aktuelle Position des Lese-/Schreibzeigers in der Datei zurück. Beachten Sie, dass das erste Byte in einer Datei mit 1 nummeriert ist. Für einen seriellen Kommunikationsport, der als „fnbr” geöffnet ist, gibt diese Funktion die Anzahl der empfangenen Bytes zurück, die im Empfangspuffer zum Lesen bereitstehen. #0 kann verwendet werden, was sich auf den Eingabepuffer der Konsole bezieht. Das # ist optional.
LOF([#]fnbr)	Für eine Datei im Flash-Dateisystem oder auf der SD-Karte, die als „fnbr” geöffnet wurde, gibt diese Funktion die aktuelle Länge der Datei in Byte zurück. Für einen seriellen Kommunikationsport, der als „fnbr” geöffnet ist, gibt diese Funktion den verbleibenden Platz (in Zeichen) im Sendepuffer zurück. Beachten Sie, dass MMBasic bei vollem Puffer beim Hinzufügen eines neuen Zeichens pausiert und wartet, bis wieder Platz verfügbar ist. Diese Funktion kann verwendet werden, um dies zu vermeiden. Das # ist optional.
LOG(Zahl)	Gibt den natürlichen Logarithmus des Arguments „number” zurück.
MAP(n)	NUR FÜR HDMI-, VGA- UND PICOMITE RP2350-PUFFERTreiber Gibt den 24-Bit-RGB-Wert für den Index „n” in der Farbtabelle zurück. Siehe Befehl MAP. Damit kann der Basic-Programmierer eine durch den Befehl MAP festgelegte Farbe verwenden. z. B. MAP(8) = RGB(100,100,100) MAP SET Pixel x,y,map(8)

	Hinweis: Bei VGA werden alle mit dem Befehl „map“ festgelegten Farben in die nächstgelegene RGB121-Farbe umgewandelt, die durch das VGA-Widerstandsnetzwerk bestimmt wird. Bei HDMI-Bildschirmen werden die Farben in die nächstgelegene RGB555-Farbe (Auflösung 640 x 480) oder RGB332-Farbe (Auflösung 1024 x 768 oder 1280 x 720) umgewandelt. Bei gepufferten Treibern für PIcoMite RP2350 werden die Farben in die nächstgelegene RGB332-Farbe umgewandelt.
MATH **Einfache Funktionen**	Die Math-Funktion führt viele einfache mathematische Berechnungen aus, die in Basic programmiert werden können, aber es gibt Geschwindigkeitsvorteile beim Codieren von Schleifenstrukturen in C und den Vorteil, dass sie nach dem Debuggen für alle verfügbar sind, ohne dass das Rad neu erfunden werden muss.
MATH(ATAN3 x,y)	Gibt ATAN3 von x und y zurück
MATH(COSH a)	Gibt den hyperbolischen Kosinus von a zurück.
MATH(LOG10 a)	Gibt den Logarithmus zur Basis 10 von a zurück
MATH(SINH a)	Gibt den hyperbolischen Sinus von a zurück
MATH(TANH a)	Gibt den hyperbolischen Tangens von a zurück
MATH(CRCn data [,length] [,polynome] [,startmask] [,endmask] [,reverseIn] [,reverseOut]	Berechnet den CRC auf n Bits (8, 12, 16, 32) von „data”. „data” kann ein ganzzahliges oder Gleitkomma-Array oder eine Zeichenfolgenvariable sein. „Length” ist optional und wird, wenn nicht angegeben, die Größe des Arrays oder die Länge der Zeichenfolge verwendet. Die Standardwerte für startmask, endmask reverseIn und reversOut sind alle Null. reverseIn und reversOut sind beide Boolesche Werte und nehmen den Wert 1 oder 0 an. Die Standardwerte für polynome sind CRC8=&H07, CRC12=&H80D, CRC16=&H1021, crc32=&H04C11DB7 Beispiel: Für crc16_CCITT verwenden Sie MATH(CRC16 array(), n,, &HFFFF)
MATH(RAND)	Gibt eine Zufallszahl 0,0 <= n < 1,0 unter Verwendung des „Mersenne-Twister-Algorithmus” zurück. Wenn nicht mit MATH RANDOMIZE initialisiert, wird bei der ersten Verwendung die Zeit in Mikrosekunden seit dem Start als Startwert verwendet. Hinweis: Der RP2350 verfügt über einen Hardware-Zufallszahlengenerator.
Einfache Statistik	
MATH(CHI a())	Gibt den Pearson-Chi-Quadrat-Wert des zweidimensionalen Arrays a()) zurück.
MATH(CHI_p a())	Gibt die zugehörige Wahrscheinlichkeit in % des Pearson-Chi-Quadrat-Werts des zweidimensionalen Arrays a()) zurück.
MATH(CROSSING array() [,level] [,direction]	Gibt den Array-Index zurück, bei dem die Werte im Array den „level” in Die angegebene Richtung. Der Standardwert für „level“ ist 0. Der Standardwert für „direction“ ist 1 (gültige Werte sind -1 oder 1).
MATH(CORREL a(), a())	Gibt den Pearson-Korrelationskoeffizienten zwischen den Arrays a() und b() zurück.
MATH(MAX a() [,index%])	

MATH(MEAN a())	Gibt das Maximum aller Werte im Array a() zurück, wobei a() eine beliebige Anzahl von Dimensionen haben kann. Wenn die Ganzzahlvariable angegeben ist, wird sie mit dem Index des Maximalwerts im Array aktualisiert. Dies ist nur bei eindimensionalen Arrays verfügbar.
MATH(MEDIAN a())	Gibt den Durchschnitt aller Werte im Array a() zurück. a() kann eine beliebige Anzahl von Dimensionen haben.
MATH(MIN a(), [index%])	Gibt den Median aller Werte im Array a() zurück, wobei a() eine beliebige Anzahl von Dimensionen haben kann.
	Gibt das Minimum aller Werte im Array a() zurück, wobei a() eine beliebige Anzahl von Dimensionen haben kann. Wenn die Ganzzahlvariable angegeben ist, wird sie mit dem Index des Minimalwerts im Array aktualisiert. Dies ist nur bei eindimensionalen dimensionalen Arrays verfügbar.
MATH(SD a())	
MATH(SUM a())	Gibt die Stichprobenstandardabweichung aller Werte im Array a() zurück, wobei a() eine beliebige Anzahl von Dimensionen haben kann.
Vektorarithmetik	Gibt die Summe aller Werte im Array a() zurück. a() kann eine beliebige Anzahl von Dimensionen haben.
MATH(MAGNITUDE v())	
MATH(DOTPRODUCT v1(), v2())	Gibt die Größe des Vektors v() zurück. Der Vektor kann eine beliebige Anzahl von Elementen haben.
Matrixarithmetik	Gibt das Skalarprodukt zweier Vektoren v1() und v2() zurück. Die Vektoren können eine beliebige Anzahl von Elementen haben, müssen jedoch dieselbe Kardinalität aufweisen.
MATH(M_DETERMINANT array!())	Gibt die Determinante des Arrays zurück. Das Array muss quadratisch sein.

Erstellung complex% = MATH(C_CPLX r!, i!) complex% = MATH(C_POLAR radius!, angle!) **Floating gibt zurück** real! = MATH(C_REAL complex%) imag! = MATH(C_IMAG complex%) arg! = MATH(C_ARG complex%) mod! = MATH(C_MOD komplex%) phase! = MATH(C_PHASE komplex%) **Unäre Funktionen** complex1% = MATH(C_CONJ complex2%) complex1% = MATH(C_SIN complex2%) complex1% = MATH(C_COS complex2%) complex1% = MATH(C_TAN complex2%) complex1% = MATH(C_ASIN complex2%)	MMBasic unterstützt eine ganze Reihe von Funktionen, die die Bearbeitung komplexer Zahlen ermöglichen. In dieser Implementierung haben komplexe Zahlen einen 32-Bit-Realteil und einen 32-Bit-Imaginärteil. Damit dies in MMBasic funktioniert, werden diese mit ganzen Zahlen (64 Bit) gespeichert.

complex1% = MATH(C_ACOS complex2%) complex1% = MATH(C_ATAN complex2%) complex1% = MATH(C_SINH complex2%) complex1% = MATH(C_COSH complex2%) complex1% = MATH(C_TANH complex2%) complex1% = MATH(C_ASINH complex2%) complex1% = MATH(C_ACOSH complex2%) complex1% = MATH(C_ATANH complex2%) complex1% = MATH(C_PROJ complex2%) **Grundlegende Arithmetik** complex1% = MATH(C_ADD complex2%,complex3%) complex1% = MATH(C_SUB complex2%,complex3%) complex1% = MATH(C_MUL complex2%,complex3%) complex1% = MATH(C_DIV complex2%,complex3%) complex1% = MATH(C_POW complex2%,Komplex3 %) Komplex1 % = MATH(C_AND Komplex2 %, Komplex3 %) Komplex1 % = MATH(C_OR Komplex2 %, Komplex3 %) Komplex1 % = MATH(C_XOR Komplex2 %, Komplex3 %)	
MATH(PID-Kanal, Sollwert!, Messwert))	Diese Funktion muss in der PID-Callback-Subroutine für den angegebenen „Kanal" aufgerufen werden und gibt die Ausgabe der Reglerfunktion zurück. Der Wert „setpoint" ist der gewünschte Zustand, den der Regler zu erreichen versucht. Der Wert „measurement" ist der aktuelle Wert in der realen Welt. https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17263 Ein Beispiel für die Einrichtung und den Betrieb eines PID-Reglers
MATH(BASE64 ENCODE/DECODE in$/in(), out$/out())	Gibt die Länge von out$/out() zurück. Diese Base64-Funktion codiert oder decodiert die Daten in „in" und speichert das Ergebnis in „out". Wenn Arrays als Ausgabe verwendet werden, müssen sie im Verhältnis zur Eingabe und zur Richtung groß genug sein. Durch die Verschlüsselung erhöht sich die Länge um 4/3, durch die Entschlüsselung verringert sie sich um 3/4.
MAX(arg1 [, arg2 [, …]]) oder MIN(arg1 [, arg2 [, …]])	Gibt die maximale oder minimale Zahl in der Argumentliste zurück. Beachten Sie, dass es sich um einen Fließkomma-Vergleich handelt (ganzzahlige Argumente werden in Fließkommazahlen umgewandelt) und eine Fließkommazahl zurückgegeben wird.
MID$(string$, start) oder MID$(string$, start, nbr)	Gibt eine Teilzeichenfolge von „string$" zurück, die bei „start" beginnt und sich über „nbr" Zeichen erstreckt. Das erste Zeichen in der Zeichenfolge ist die Zahl 1. Wenn „nbr" weggelassen wird, erstreckt sich die zurückgegebene Zeichenfolge bis zum Ende von „string$".
OCT$(Zahl [, Zeichen])	Gibt eine Zeichenkette zurück, die die oktale (Basis 8) Darstellung von „number" angibt. „chars" ist optional und gibt die Anzahl der Zeichen in der Zeichenfolge an, wobei Null als führendes Auffüllzeichen verwendet wird.
PEEK(BYTE addr%) oder PEEK(SHORT addr%) oder PEEK(WORD addr%) oder PEEK(INTEGER addr%) oder PEEK(FLOAT addr%)	Hinweis: Adressen werden abgerundet, um dem angeforderten Datentyp zu entsprechen (z. B. PEEK(SHORT) oder es wird ein Fehler ausgegeben, wenn sie nicht ausgerichtet sind, z. B. PEEK(SP Gibt ein Byte oder ein Wort innerhalb des virtuellen Speicherbereichs des Prozessors zurück. BYTE gibt das Byte (8 Bit) zurück, das sich an der Adresse „addr%" befindet. SHORT gibt die kurze Ganzzahl (16 Bit) an der Adresse „addr%" zurück. WORD gibt das Wort (32 Bit) an der Adresse „addr%" zurück. INTEGER gibt die Ganzzahl (64 Bit) an der Adresse „addr%" zurück. FLOAT gibt die Gleitkommazahl (64 Bit) an der Adresse „addr%" zurück.

oder PEEK(VARADDR var) oder	VARADDR gibt die Adresse (32 Bit) der Variablen „var" im Speicher zurück. Ein Array wird als var() angegeben.
PEEK(VARHEADER var) oder	VARHEADER gibt die Adresse (32 Bit) des Headers der Variablen „var" im Speicher zurück. Ein Array wird als var() angegeben. Dies ist die Adresse des ersten Bytes des Variablennamens.
PEEK(CFUNADDR cfun) oder	CFUNADDR gibt die Adresse (32 Bit) der CFunktion „cfun" im Speicher zurück. Diese Adresse kann an eine andere CFunktion übergeben werden, die sie dann aufrufen kann, um einen gemeinsamen Prozess auszuführen.
PEEK(VAR var, ±offset) oder	VAR gibt ein Byte im Speicher zurück, das 'var' zugewiesen ist. Ein Array wird als var() angegeben.
PEEK(VARTBL, ±offset) oder	VARTBL gibt ein Byte in dem Speicher zurück, der der von MMBasic verwalteten Variablentabelle zugewiesen ist. Beachten Sie, dass nach dem Schlüsselwort VARTBL ein Komma steht.
PEEK(PROGMEM, ±offset)	PROGMEM gibt ein Byte in dem für das Programm zugewiesenen Speicher zurück. Beachten Sie, dass nach dem Schlüsselwort PROGMEM ein Komma steht. Beachten Sie, dass „addr%" eine ganze Zahl sein sollte.
PEEK(BP n%)	PEEK(bp n%) gibt das Byte an der Adresse n% zurück und erhöht n%, um auf das nächste Byte zu zeigen.
PEEK(SP n%)	PEEK(sp n%) gibt das Short an der Adresse n% zurück und erhöht n%, um auf das nächste Short zu zeigen.
PEEK(WP n%)	PEEK(wp n%) ' gibt das Wort an der Adresse n% zurück und erhöht n%, um auf das nächste Wort zu zeigen.
PI	Gibt den Wert von Pi zurück.
PIN(pin)	Gibt den Wert am externen E/A-Pin zurück. Null bedeutet digital niedrig, 1 bedeutet digital hoch, und bei analogen Eingängen wird die gemessene Spannung als Gleitkommazahl zurückgegeben. Frequenzeingänge geben die Frequenz in Hz zurück. Ein Periodeneingang gibt die Periode in Millisekunden zurück, während ein Zähleingang die Anzahl seit dem Zurücksetzen zurückgibt (die Zählung erfolgt an der positiven Flanke). Der Zähleingang kann durch Zurücksetzen des Pins auf Zähleingang auf Null zurückgesetzt werden (auch wenn er bereits so konfiguriert ist). Wenn ein Pin als Ausgang konfiguriert ist, gibt diese Funktion den Wert der Ausgangseinstellung zurück (d. h. hoch oder niedrig). Siehe auch die Befehle SETPIN und PIN() =. Eine allgemeine Beschreibung der Ein-/Ausgabefunktionen des PicoMite finden Sie im Kapitel *„Verwendung der I/O-Pins*".
PIN(BOOTSEL)	Gibt den Status des Boot-Auswahlschalters zurück, sodass dieser als Benutzereingabe in einem Programm verwendet werden kann.
PIN(TEMP)	Gibt die Temperatur des RP2040/RP2350-Chips zurück (Einzelheiten finden Sie im Datenblatt).
PIO(DMA RX POINTER) PIO(DMA TX POINTER)	Gibt das aktuelle Datenelement zurück, das vom PIO geschrieben oder gelesen wird.
PIO (SHIFTCTRL push_threshold [,pull_threshold] [,autopush]	Hilfsfunktion zur Berechnung des Werts von shiftctrl für INIT MACHINE Befehl berechnet.

[,autopull] [,in_shiftdir] [,out_shiftdir] [,fjoin_tx] [,fjoin_rx])	
PIO (PINCTRL no_side_set_pins [,no_set_pins] [,no_out_pins] [,IN base] [,side_set_base] [,set_base][, out_base])	Hilfsfunktion zur Berechnung des Werts von pinctrl für den Befehl INIT MACHINE. Hinweis: Die Pin-Parameter müssen im Format GPn angegeben werden.
PIO (EXECCTRL jmp_pin ,wrap_target, wrap [,side_pindir] [,side_en])	Hilfsfunktion zur Berechnung des Werts von execctrl für den Befehl INIT MACHINE
PIO(READFIFO a, b, c)	Auslesen aus einem PIO-FIFO „a" ist der PIO (0 oder 1), „b" ist die Zustandsmaschine (0...3), „c" ist das FIFO-Register *0…3)
PIO (FDEBUG pio)	Gibt den Wert des FSDEBUG-Registers für den angegebenen PIO zurück
PIO (FSTAT pio)	Gibt den Wert des FSTAT-Registers für den angegebenen PIO zurück
PIO (FLEVEL pio)	Gibt den Wert des FLEVEL-Registers für die angegebene PIO zurück PIO(FLEVEL pio)
PIO(FLEVEL pio ,sm, DIR)	dir kann RX oder TX sein. Gibt den Pegel des angegebenen FIFO zurück
PIO(.WRAP) PIO(.WRAP TARGET)	Gibt die Position der .wrap-Direktive in PIO ASSEMBLE zurück. Gibt die Position der .wrap-Ziel-Direktive in PIO ASSEMBLE zurück. Diese können in der PIO(EXECCTRL-Funktion wie folgt verwendet werden: PIO (EXECCTRL jmp_pin PIO(.WRAP TARGET), PIO(.WRAP) [,side_pindir] [,side_en])
PIO(NEXT LINE)	Gibt den nächsten ungenutzten PIO-Befehlsslot nach einem Block von PIO-Befehlen zurück, der mit END PROGRAM beendet wurde.
PIXEL(x, y)	Gibt die Farbe eines Pixels auf dem Videoausgang oder dem LCD-Display zurück. „x" ist die horizontale Koordinate und „y" ist die vertikale Koordinate des Pixels. Wenn ein LCD-Display verwendet wird, muss es einen der Controller SSD1963, ILI9341, ILI9488 oder ST7789_320 verwenden.
PORT(start, nbr [,start, nbr]…)	Gibt den Wert einer Reihe von E/A-Pins in einem Vorgang zurück. „start" ist eine I/O-Pin-Nummer, deren Wert als Bit 0 zurückgegeben wird. „start"+1 wird als Bit 1 zurückgegeben, „start"+2 als Bit 2 usw. für die Anzahl der Bits „nbr". Die verwendeten I/O-Pins müssen fortlaufend nummeriert sein. Jeder I/O-Pin, der ungültig oder nicht als Eingang konfiguriert ist, verursacht einen Fehler. Das Start-/Anzahl-Paar kann bis zu 25 Mal wiederholt werden, wenn zusätzliche Gruppen von Eingangs-Pins hinzugefügt werden müssen. Diese Funktion gibt auch den Ausgangsstatus eines als Ausgang konfigurierten Pins zurück. Dies kann zur bequemen Kommunikation mit parallelen Geräten wie Speicherchips verwendet werden. Es kann eine beliebige Anzahl von E/A-Pins (und damit Bits) von 1 bis zur Anzahl der E/A-Pins auf dem Chip verwendet werden. Hinweis: Wenn die Pins mit der GPn-Syntax definiert sind, ignoriert die Firmware ungültige Pins, sodass `PORT(GP0, 8)` acht I/O-Pins liest: GP0, GP1, GP2, GP3, GP4, GP5, GP6 und GP7.

	Siehe den Befehl PORT, um gleichzeitig an mehrere Pins auszugeben.
PULSIN(Pin, Polarität) oder PULSIN(Pin, Polarität, t1) oder PULSIN(Pin, Polarität, t1, t2)	Misst die Breite eines Eingangsimpulses von 1 µs bis 1 Sekunde mit einer Auflösung von 0,1 µs. Auflösung. „pin" ist der für die Messung zu verwendende E/A-Pin, der zuvor als digitaler Eingang konfiguriert werden muss. „polarity" ist der zu messende Impulstyp. Bei Null gibt die Funktion die Breite des nächsten negativen Impulses zurück, bei einem Wert ungleich Null wird der nächste positive Impuls gemessen. „t1" ist die Zeitüberschreitung, die beim Warten auf das Eintreffen des Impulses angewendet wird, „t2" ist die Zeitüberschreitung, die beim Messen des Impulses verwendet wird. Beide Angaben erfolgen in Mikrosekunden (µs) und sind optional. Wenn „t2" weggelassen wird, wird der Wert von „t1" für beide Zeitüberschreitungen verwendet. Wenn sowohl „t1" als auch „t2" weggelassen werden, werden die Zeitüberschreitungen auf 100000 (d. h. 100 ms) gesetzt. Diese Funktion gibt die Breite des Impulses in Mikrosekunden (µs) zurück oder -1, wenn ein Zeitüberschreitung aufgetreten. Die Messung ist auf ±0,5 % und ±0,5 µs genau. Beachten Sie, dass diese Funktion das laufende Programm während der Messung unterbricht und Interrupts während dieses Zeitraums ignoriert werden.
RAD(Grad)	Konvertiert „Grad" in Radiant.
RGB(rot, grün, blau) oder RGB(Shortcut)	Erzeugt einen RGB-Echtfarbenwert. „Rot", „Blau" und „Grün" stehen für die Intensität der jeweiligen Farbe. Der Wert Null steht für Schwarz und 255 für volle Intensität. Mit „Abkürzung" können gängige Farben durch Benennung angegeben werden. Die Farben, die benannt werden können, sind Weiß, Schwarz, Blau, Grün, Cyan, Rot, Magenta, Gelb, Braun, Weiß, Orange, Rosa, Gold, Lachs, Beige, Hellgrau und Grau (oder die US-amerikanische Schreibweise Gray/Lightgray). Zum Beispiel RGB(Rot) oder RGB(Cyan).
RIGHT$(Zeichenfolge$, Anzahl der Zeichen)	Gibt eine Teilzeichenfolge von „string$" mit „number-of-chars" Zeichen von rechts (Ende) der Zeichenfolge zurück.
RND(Zahl) oder RND	Gibt eine Zufallszahl (RP2350) oder Pseudozufallszahl (RP2040) im Bereich von 0 bis 0,999999 zurück. Der Wert „number" wird ignoriert, wenn er angegeben wird. Siehe auch den Befehl RANDOMIZE (nur RP2040).
SGN(Zahl)	Gibt das Vorzeichen des Arguments „Zahl" zurück, +1 für positive Zahlen, 0 für 0 und -1 für negative Zahlen.
SIN(Zahl)	Gibt den Sinus des Arguments „Zahl" in Radianten zurück.
SPACE$(Zahl)	Gibt eine Zeichenfolge mit Leerzeichen zurück, die 'number' Zeichen lang ist.
SPI (Daten) oder SPI2 (Daten)	Senden und Empfangen von Daten über einen SPI-Kanal. Eine einzelne SPI-Transaktion sendet Daten und empfängt gleichzeitig Daten vom Slave. „data" sind die zu sendenden Daten, und die Funktion gibt die während der Transaktion empfangenen Daten zurück. „data" kann eine Ganzzahl, eine Gleitkommavariable oder eine Konstante sein.
SPRITE() SPRITE(C, [#]n)	NUR VGA- UND HDMI-VERSIONEN Die SPRITE-Funktionen geben Informationen zu Sprites zurück, bei denen es sich um kleine Grafiken auf dem VGA/HDMI-Bildschirm handelt. Diese sind beim Schreiben von Spielen nützlich. Siehe auch die SPRITE-Befehle. Gibt die Anzahl der derzeit aktiven Kollisionen für Sprite n zurück. Wenn n=0, gibt es die Anzahl der Sprites zurück, die nach einem SPRITE SCROLL-Befehl derzeit eine aktive Kollision haben.

SPRITE(C, [#]n, m)	Gibt die Nummer des Sprites zurück, das die „m“-te Kollision von Sprite n verursacht hat. Wenn n=0, wird die Sprite-Nummer des „m“-ten Sprites zurückgegeben, das nach einem SPRITE SCROLL-Befehl derzeit eine aktive Kollision hat. Wenn die Kollision mit dem Rand des Bildschirms stattfand, lautet der Rückgabewert: &HF1 Kollision mit der linken Seite des Bildschirms &HF2 Kollision mit dem oberen Bildschirmrand &HF4 Kollision mit der rechten Seite des Bildschirms &HF8 Kollision mit dem unteren Rand des Bildschirms
SPRITE(D ,[#]s1, [#]s2)	Gibt den Abstand zwischen den Mittelpunkten der Sprites „s1” und „s2” zurück (gibt -1 zurück, wenn eines der Sprites nicht aktiv ist)
SPRITE(E, [#]n)	Gibt eine Bitmap zurück, die alle Kanten des Bildschirms angibt, mit denen das Sprite in Kontakt steht: 1 = linke Bildschirmseite, 2 = obere Bildschirmseite, 4 = rechte Bildschirmseite, 8 = untere Bildschirmseite
SPRITE(H,[#]n)	Gibt die Höhe von Sprite n zurück. Diese Funktion ist unabhängig davon aktiv, ob das Sprite gerade angezeigt wird (aktiv ist).
SPRITE(L, [#]n)	Gibt die Ebenennummer des aktiven Sprites Nummer n zurück Gibt
SPRITE(N)	die Anzahl der angezeigten (aktiven) Sprites zurück
SPRITE(N,n) SPRITE(S)	Gibt die Anzahl der angezeigten (aktiven) Sprites auf Ebene n zurück
SPRITE(T, [#]n)	Gibt die Nummer des Sprites zurück, das zuletzt eine Kollision verursacht hat. Hinweis: Wenn die zurückgegebene Nummer Null ist, ist die Kollision das Ergebnis eines SPRITE SCROLL-Befehls, und die Funktion SPRITE(C…) sollte verwendet werden, um herauszufinden, wie viele und welche Sprites kollidiert sind.
SPRITE(V,[#]s1, [#]s2)	Gibt eine Bitmap zurück, die alle Sprites anzeigt, die derzeit das angeforderte Sprite berühren. Die Bits 0-63 in der zurückgegebenen Ganzzahl stellen jeweils eine aktuelle Kollision mit den Sprites 1 bis 64 dar. Gibt den Vektor vom Sprite „s1” zum Sprite „s2” in Radianten zurück. Der Winkel basiert auf der Uhr, wenn also „s2” auf dem Bildschirm über „s1” liegt, ist das Ergebnis Null. Dies kann für jedes Paar sichtbarer Sprites verwendet werden. Ist eines der Sprites nicht sichtbar, gibt die Funktion -1 zurück. Dies ist besonders nützlich nach einer Kollision, wenn der Programmierer eine differenzierte Entscheidung basierend auf dem Ort der Kollision treffen möchte. Der Winkel wird zwischen den Mittelpunkten der einzelnen Sprites berechnet, die natürlich unterschiedliche Größen haben können.
SPRITE(W, [#]n)	Gibt die Breite von Sprite n zurück. Diese Funktion ist unabhängig davon aktiv, ob das Sprite derzeit angezeigt wird (aktiv) oder nicht.
SPRITE(X, [#]n)	Gibt die X-Koordinate von Sprite n zurück. Diese Funktion ist nur aktiv, wenn das Sprite gerade angezeigt wird (aktiv). Andernfalls wird 10000 zurückgegeben.
SPRITE(Y, [#]n)	Gibt die Y-Koordinate von Sprite n zurück. Diese Funktion ist nur aktiv, wenn das Sprite gerade angezeigt wird (aktiv). Andernfalls wird 10000 zurückgegeben.
SQR(Zahl)	Gibt die Quadratwurzel des Arguments „Zahl” zurück.

STR$(Zahl) oder STR$(Zahl, m) oder STR$(Zahl, m, n) oder STR$(Zahl, m, n, c$)	Gibt eine Zeichenfolge in der Dezimaldarstellung (Basis 10) von „number" zurück. Wenn „m" angegeben ist, werden am Anfang der Zahl ausreichend Leerzeichen hinzugefügt, um sicherzustellen, dass die Anzahl der Zeichen vor dem Dezimalpunkt (einschließlich des Vorzeichens) mindestens „m" Zeichen beträgt. Wenn „m" Null ist oder die Zahl mehr als „m" signifikante Stellen hat, werden keine Leerzeichen hinzugefügt. Wenn „m" negativ ist, werden positive Zahlen mit dem Pluszeichen und negative Zahlen mit dem Minuszeichen versehen. Wenn „m" positiv ist, wird nur das Minuszeichen verwendet. „n" ist die Anzahl der Stellen, die nach dem Dezimalpunkt folgen müssen. Ist „n" gleich Null, wird die Zeichenfolge ohne Dezimalpunkt zurückgegeben. Ist „n" negativ, wird für die Ausgabe immer das Exponentialformat mit einer Auflösung von „n" Stellen verwendet. Ist „n" nicht angegeben, variieren die Anzahl der Dezimalstellen und das Ausgabeformat automatisch entsprechend der Zahl. „c$" ist eine Zeichenfolge. Wenn sie angegeben ist, wird das erste Zeichen dieser Zeichenfolge anstelle eines Leerzeichens als Auffüllzeichen verwendet (siehe Argument „m"). Beispiele: STR$(123,456) gibt „123,456" zurück STR$(-123,456) gibt „-123,456" zurück STR$(123,456, 1) gibt „123,456" zurück STR$(123,456, -1) gibt „+123,456" zurück STR$(123,456, 6) gibt „ 123,456" STR$(123.456, -6) gibt „ +123,456" zurück STR$(-123,456, 6) gibt „ -123,456" zurück STR$(-123,456, 6, 5) gibt „-123,45600" zurück STR$(-123,456, 6, -5) gibt „ -1,23456e+02" STR$(53, 6) gibt „ 53" STR$(53, 6, 2) gibt „ 53,00 STR$(53, 6, 2, "*") gibt „****53,00" zurück
STR2BIN(Typ, Zeichenfolge$ [,BIG])	Gibt eine Zahl zurück, die der binären Darstellung in „string$" entspricht. „type" kann sein: INT64 konvertiert eine 8-Byte-Zeichenkette, die eine vorzeichenbehaftete 64-Bit-Ganzzahl darstellt, in eine Ganzzahl UINT64 konvertiert eine 8-Byte-Zeichenkette, die eine vorzeichenlose 64-Bit-Ganzzahl darstellt, in eine Ganzzahl INT32 konvertiert eine 4-Byte-Zeichenkette, die eine vorzeichenbehaftete 32-Bit-Ganzzahl darstellt, in eine Ganzzahl UINT32 konvertiert eine 4-Byte-Zeichenkette, die eine vorzeichenlose 32-Bit-Ganzzahl darstellt, in eine Ganzzahl INT16 konvertiert eine 2-Byte-Zeichenkette, die eine vorzeichenbehaftete 16-Bit-Ganzzahl darstellt, in eine Ganzzahl UINT16 konvertiert eine 2-Byte-Zeichenkette, die eine vorzeichenlose 16-Bit-Ganzzahl darstellt, in eine Ganzzahl. INT8 konvertiert eine 1-Byte-Zeichenkette, die eine vorzeichenbehaftete 8-Bit-Ganzzahl darstellt, in eine Ganzzahl UINT8 konvertiert eine 1-Byte-Zeichenkette, die eine vorzeichenlose 8-Bit-Ganzzahl darstellt, in eine Ganzzahl SINGLE konvertiert eine 4-Byte-Zeichenkette, die eine Gleitkommazahl mit einfacher Genauigkeit darstellt, in eine Gleitkommazahl DOUBLE konvertiert eine 8-Byte-Zeichenkette, die eine Gleitkommazahl mit einfacher Genauigkeit darstellt, in eine Gleitkommazahl Standardmäßig muss die Zeichenkette die Zahl im Little-Endian-Format enthalten (d. h. das das niedrigstwertige Byte das erste in der Zeichenfolge ist). Wenn Sie den dritten Parameter auf „BIG" setzen, wird die Zeichenfolge im Big-Endian-Format interpretiert (d. h. das Byte das erste in der Zeichenkette ist). Diese Funktion erleichtert das Auslesen von Daten aus Binärdateien, das Interpretieren von Zahlen aus Sensoren oder das effiziente Auslesen von Binärdaten aus Flash-Speicherchips. Wenn die Zeichenfolge die falsche Länge für die

	gewünschte Konvertierung Siehe auch die Funktion BIN2STR$
STRING$(nbr, ascii) oder STRING$(nbr, string$)	Gibt eine Zeichenfolge mit einer Länge von „nbr" Bytes zurück, die entweder aus dem ersten Zeichen von string$ oder dem Zeichen besteht, das den ASCII-Wert „ascii" darstellt, bei dem es sich um eine ganze Zahl oder eine Gleitkommazahl im Bereich von 0 bis 255 handelt.
TAB(Zahl)	Gibt Leerzeichen aus, bis die durch „number" angegebene Spalte in der Konsolenausgabe erreicht ist.
TAN(Zahl)	Gibt den Tangens des Arguments „number" in Radianten zurück.
TEMPR(Pin [,Timeout])	Gibt die Temperatur zurück, die von einem an „pin" angeschlossenen DS18B20-Temperatursensor gemessen wurde (der nicht konfiguriert werden muss). Der zurückgegebene Wert ist in Grad Celsius mit einer Standardauflösung von 0,25 °C. Wenn während der Messung ein Fehler auftritt, ist der zurückgegebene Wert 1000. Die für die gesamte Messung erforderliche Zeit beträgt 200 ms, und Interrupts werden während dieses Zeitraums ignoriert. Der optionale Parameter „timeout" kann verwendet werden, um den Standardwert (200 ms) zu überschreiben und langsame Geräte zu berücksichtigen. Alternativ kann der Befehl TEMPR START verwendet werden, um die Messung zu starten, während Ihr Programm andere Aufgaben ausführt, während die Umwandlung läuft. Wenn diese Funktion aufgerufen wird, wird der Wert sofort zurückgegeben, vorausgesetzt, die Umwandlungszeit ist abgelaufen. Ist dies nicht der Fall, wartet diese Funktion die restliche Umwandlungszeit ab, bevor sie den Wert zurückgibt. Der DS18B20 kann separat mit einer 3-V- bis 5-V-Stromversorgung betrieben werden oder mit der parasitären Stromversorgung des Raspberry Pi Pico. Weitere Informationen finden Sie im Kapitel *„Spezielle Hardwaregeräte*".
TIME$	Gibt die aktuelle Uhrzeit basierend auf der internen Uhr von MMBasic als Zeichenfolge im Format „HH:MM:SS" in 24-Stunden-Notation zurück. Zum Beispiel „14:30:00". Um die aktuelle Uhrzeit einzustellen, verwenden Sie den Befehl TIME$ = .
TIMER	Gibt die seit dem Zurücksetzen verstrichene Zeit in Millisekunden (z. B. 1/1000 einer Sekunde) zurück. Der Timer wird beim Einschalten oder bei einem Neustart der CPU auf Null zurückgesetzt. Sie können ihn auch mit dem Befehl TIMER zurücksetzen. Wenn er nicht ausdrücklich zurückgesetzt wird, zählt er unbegrenzt weiter (es handelt sich um eine 64-Bit-Zahl, die erst nach 200 Millionen Jahren auf Null zurückspringt).
TOUCH(X) oder TOUCH(Y oder nur FT6336 TOUCH(X2) oder TOUCH(Y2)	Gibt die X- oder Y-Koordinate der Stelle zurück, die gerade auf einem LCD-Bildschirm berührt wird. Wenn der Bildschirm nicht berührt wird, gibt die Funktion -1 zurück. Bei FT6336 geben TOUCH(X2) und TOUCH(Y2) die Position einer zweiten Berührungsstelle zurück oder -1, wenn keine zweite Stelle berührt wurde.
TRIM$(Quelle$ [,Maske$] [,wo/wo$])	Diese Funktion kann Zeichen am Anfang oder am Ende einer Zeichenfolge oder an beiden Stellen entfernen. „source$" ist die Eingabezeichenfolge „mask$" ist eine Zeichenkette, die eine Liste der zu entfernenden Zeichen enthält. Wenn sie weggelassen wird, wird standardmäßig ein Leerzeichen verwendet. „where/where$" kann L, R oder B sein oder eine Zeichenfolge, die mit L, R oder B beginnt, um

	angeben, ob Zeichen links von der Quelle, rechts von der Quelle oder an beiden Stellen entfernt werden sollen. Wenn nichts angegeben ist, wird standardmäßig L verwendet.
UCASE$(string$)	Gibt „string$“ in Großbuchstaben zurück.
VAL(string$)	Gibt den numerischen Wert von „string$“ zurück. Wenn „string$“ eine ungültige Zahl ist, gibt die Funktion Null zurück. Diese Funktion erkennt das Präfix &H für eine Hexadezimalzahl, &O für eine Oktalzahl und &B für eine Binärzahl.

Veraltete Befehle und Funktionen von „ ”

Diese Befehle und Funktionen dienen hauptsächlich dazu, die Konvertierung von Programmen zu unterstützen, die für Microsoft BASIC geschrieben wurden. Für neue Programme sollten die entsprechenden modernen Befehle in MMBasic verwendet werden.

Beachten Sie, dass diese Befehle/Funktionen in Zukunft möglicherweise entfernt werden, um Speicherplatz für andere Funktionen freizugeben.

BITBANG	Ersetzt durch den Befehl DEVICE. Aus Kompatibilitätsgründen kann BITBANG weiterhin in Programmen verwendet werden und wird automatisch in DEVICE konvertiert.
DEVICE CAMERA	Geändert in den Befehl CAMERA.
DEVICE GAMEPAD	In den Befehl GAMEPAD geändert.
DEVICE HUMID	Geändert zu HUMID-Befehl
DEVICE KEYPAD	Geändert zu KEYPAD-Befehl
DEVICE MOUSE	Geändert in den Befehl MOUSE
GERÄT LCD	Auf den Befehl „LCD“ geändert
DEVICE WII	Geändert zu WII-Befehl
DEVICE WS2812	Geändert zu WS2812-Befehl
GOSUB-Ziel	Initiiert einen Unterprogrammaufruf zum Ziel, das eine Zeilennummer oder eine Bezeichnung sein kann. Das Unterprogramm muss mit RETURN enden. Neue Programme sollten definierte Unterprogramme verwenden (d. h. SUB…END SUB).
IF-Bedingung THEN Zeilenumbruch	Aus Gründen der Kompatibilität mit Microsoft wird ein GOTO angenommen, wenn auf die THEN-Anweisung eine Zahl folgt. Ein Label ist in dieser Konstruktion ungültig. Neue Programme sollten Folgendes verwenden: IF-Bedingung THEN GOTO Zeilenumbruch | Label
IRETURN	Kehrt von einer Unterbrechung zurück, wenn das Unterbrechungsziel eine Zeilennummer oder eine Markierung war. Neue Programme sollten eine benutzerdefinierte Subroutine als Interrupt-Ziel verwenden. In diesem Fall bewirkt END SUB oder EXIT SUB eine Rückkehr aus dem Interrupt.
ON nbr GOTO | GOSUB Ziel[,Ziel, Ziel,..]	ON verzweigt entweder (GOTO) oder ruft eine Unterroutine auf (GOSUB), basierend auf dem gerundeten Wert von 'nbr'; wenn dieser 1 ist, wird das erste Ziel aufgerufen, wenn 2, das zweite Ziel usw. Das Ziel kann eine Zeilennummer oder eine Markierung sein. Neue Programme sollten SELECT CASE verwenden.
POS	Gibt für die Konsole die aktuelle Cursorposition in der Zeile in Zeichen zurück.
RETURN	RETURN beendet eine von GOSUB aufgerufene Unterroutine und kehrt zur Anweisung nach dem GOSUB zurück.

Anhang A
Serielle Kommunikation

Für die asynchrone serielle Kommunikation stehen zwei serielle Schnittstellen zur Verfügung. Sie sind mit COM1: und COM2: gekennzeichnet.

I/O- -Pins

Bevor eine serielle Schnittstelle verwendet werden kann, müssen die I/O-Pins mit dem folgenden Befehl für den ersten Kanal (bezeichnet als COM1) definiert werden:

SETPIN rx, tx, COM1

Gültige Pins sind RX: GP1, GP13 oder GP17

TX: GP0, GP12, GP16 oder GP28

Und der folgende Befehl für den zweiten Kanal (bezeichnet als COM2): SETPIN rx, tx, COM2

Gültige Pins sind RX: GP5, GP9 oder GP21

TX: GP4, GP8 oder GP20

TX sind Daten vom Raspberry Pi Pico und RX sind Daten an ihn.

Beachten Sie, dass in der WebMite-Version COM1 und COM2 auf GP20 bis GP28 nicht verfügbar sind.

Die Signalpolarität entspricht dem Standard für Geräte, die mit TTL-Spannungen betrieben werden. Im Ruhezustand ist die Spannung hoch, das Startbit ist niedrig, Daten verwenden eine hohe Spannung für Logik 1 und das Stoppbit ist hoch. Diese Signalpegel ermöglichen den direkten Anschluss an Geräte wie GPS-Module (die in der Regel TTL-Spannungspegel verwenden).

Befehle

Nach dem Öffnen wird dem seriellen Port eine zugehörige Dateinummer zugewiesen, und Sie können alle Befehle verwenden, die mit einer Dateinummer arbeiten, um darin zu lesen und zu schreiben. Ein serieller Port kann mit dem Befehl CLOSE geschlossen werden.

Hier ein Beispiel:

```
SETPIN GP13, GP16, COM1          ' Weisen Sie die E/A-Pins für die erste serielle Schnittstelle zu.
OPEN „COM1:4800” AS #5           ' Öffnen der ersten seriellen Schnittstelle mit einer
Geschwindigkeit von 4800 Baud PRINT #5, "Hello"       ' Senden der Zeichenfolge „Hello” über den
seriellen Port
dat$ = INPUT$(20, #5)            ' bis zu 20 Zeichen vom seriellen Port abrufen CLOSE
#5                               ' Schließen Sie die serielle Schnittstelle.
```

Der Befehl OPEN

Eine serielle Schnittstelle wird mit dem folgenden Befehl geöffnet:

```
OPEN comspec$ AS #fnbr
```

„fnbr” ist die zu verwendende Dateinummer. Sie muss im Bereich von 1 bis 10 liegen. Das # ist optional.

„comspec$“ ist die Kommunikationsspezifikation und eine Zeichenfolge (kann eine Zeichenfolgenvariable sein), die die zu öffnende serielle Schnittstelle und optionale Parameter angibt. Die Standardeinstellung ist 9600 Baud, 8 Datenbits, keine Parität und ein Stoppbit.

Es hat die Form „COMn: baud, buf, int, int-trigger, EVEN, ODD, S2, 7BIT” wobei:

- „n” die Nummer des seriellen Anschlusses für entweder COM1: oder COM2: ist.
- „baud” die Baudrate ist. Diese kann zwischen 1200 und 921600 liegen. Der Standardwert ist 9600.
- „buf” die Größe des Empfangspuffers in Byte ist (die Standardgröße beträgt 256). Der Sendepuffer ist auf 256 Byte festgelegt.
- „int” ist die Interrupt-Subroutine, die aufgerufen wird, wenn der serielle Port Daten empfangen hat.
- „int-trigger“ ist die Anzahl der empfangenen Zeichen, die einen Interrupt auslösen.

Alle Parameter außer dem Namen der seriellen Schnittstelle (COMn:) sind optional. Wenn ein Parameter weggelassen wird, müssen auch alle folgenden Parameter weggelassen werden, und es werden die Standardwerte verwendet.

Am Ende von „comspec$“ können fünf Optionen hinzugefügt werden. Diese sind:

- „S2“ gibt an, dass nach jedem übertragenen Zeichen zwei Stoppbits gesendet werden.
- EVEN gibt an, dass ein gerades Paritätsbit angewendet wird, was zu einer 9-Bit-Übertragung führt, sofern nicht 7BIT gesetzt ist.
- ODD gibt an, dass ein ungerades Paritätsbit angewendet wird, was zu einer 9-Bit-Übertragung führt, sofern nicht 7BIT gesetzt ist.
- 7BIT gibt an, dass es sich um 7 Datenbits handelt. Dies wird normalerweise zusammen mit EVEN oder ODD verwendet.
- INV gibt an, dass die Ausgangssignale invertiert werden und die Eingabe als invertiert angenommen wird.

Beispiele

Öffnen einer seriellen Schnittstelle mit allen Standardeinstellungen:

```
OPEN „COM1:“ AS #2
```

Öffnen einer seriellen Schnittstelle unter Angabe nur der Baudrate (4800 Bit pro Sekunde):

```
ÖFFNE „COM1:4800” ALS #1
```

Öffnen einer seriellen Schnittstelle unter Angabe der Baudrate (9600 Bit pro Sekunde) und der Empfangspuffergröße (1 KB):

```
OPEN „COM2:9600, 1024” AS #8
```

Wie oben, jedoch mit zwei Stoppbits aktiviert:

```
OPEN "COM2:9600, 1024, S2" AS #8
```

Ein Beispiel, in dem alles angegeben ist, einschließlich einer Unterbrechung, einer Unterbrechungsstufe und zwei Stoppbits:

```
OPEN "COM2:19200, 1024, ComIntLabel, 256, S2" AS #5
```

Lesen und Schreiben von

Sobald eine serielle Schnittstelle geöffnet wurde, können Sie jeden Befehl oder jede Funktion verwenden, die eine Dateinummer zum Lesen und Schreiben auf die Schnittstelle verwendet. Die von der seriellen Schnittstelle empfangenen Daten werden von MMBasic automatisch im Speicher gepuffert, bis sie vom Programm gelesen werden. Die Funktion INPUT$() ist hierfür am besten geeignet. Bei Verwendung der Funktion INPUT$() entspricht die angegebene Zeichenanzahl der maximalen Anzahl der zurückgegebenen Zeichen, kann jedoch geringer sein, wenn sich weniger Zeichen im Empfangspuffer befinden. Tatsächlich gibt die Funktion INPUT$() sofort eine leere Zeichenfolge zurück, wenn im Empfangspuffer keine Zeichen verfügbar sind.

Die Funktion LOC() ist ebenfalls nützlich; sie gibt die Anzahl der im Empfangspuffer wartenden Zeichen zurück (d. h. die maximale Anzahl von Zeichen, die mit der Funktion INPUT$() abgerufen werden können). Beachten Sie, dass die serielle Schnittstelle bei einem Überlauf des Empfangspuffers durch eingehende Daten automatisch die ältesten Daten verwirft, um Platz für die neuen Daten zu schaffen.

Der Befehl PRINT wird für die Ausgabe an eine serielle Schnittstelle verwendet. Alle zu sendenden Daten werden in einem Speicherpuffer gehalten, während die serielle Schnittstelle sie sendet. Das bedeutet, dass MMBasic nach dem Befehl PRINT mit der Ausführung der Befehle fortfährt, während die Daten übertragen werden. Die einzige Ausnahme ist, wenn der Ausgabepuffer voll ist. In diesem Fall hält MMBasic an und wartet, bis genügend Platz vorhanden ist, bevor es fortfährt. Die Funktion LOF() gibt den verbleibenden Speicherplatz im Sendepuffer zurück. Damit können Sie vermeiden, dass das Programm ins Stocken gerät, während es darauf wartet, dass Speicherplatz im Puffer verfügbar wird.

Wenn Sie sicherstellen möchten, dass alle Daten gesendet wurden (z. B. weil Sie die Antwort vom Remote-Gerät lesen möchten), sollten Sie warten, bis die Funktion LOF() den Wert 256 (die Größe des Sendepuffers) zurückgibt, was bedeutet, dass nichts mehr zu senden ist.

Serielle Schnittstellen können mit dem Befehl CLOSE geschlossen werden. Dadurch wird gewartet, bis der Sendepuffer geleert ist, dann wird der von den Puffern belegte Speicher freigegeben und der Interrupt (falls gesetzt) abgebrochen. Eine serielle Schnittstelle wird auch automatisch geschlossen, wenn Befehle wie RUN und NEW ausgegeben werden.

Interrupts

Die Interrupt-Subroutine (sofern angegeben) funktioniert genauso wie ein allgemeiner Interrupt an einem externen E/A-Pin (eine Beschreibung finden Sie im Kapitel *„Verwendung der E/A-Pins*“).

Bei der Verwendung von Interrupts müssen Sie beachten, dass es einige Zeit dauert, bis MMBasic auf den Interrupt reagiert, und dass in der Zwischenzeit weitere Zeichen eingegangen sein können, insbesondere bei hohen Baudraten. Wenn Sie beispielsweise die Interrupt-Stufe auf 200 Zeichen und einen Puffer von 256 Zeichen festgelegt haben, kann es leicht passieren, dass der Puffer überläuft, bevor die Interrupt-Subroutine die Daten lesen kann. In diesem Fall sollte der Puffer auf 512 Zeichen oder mehr erhöht werden.

Anhang B
I2C-Kommunikation

Es gibt zwei I²C-Kanäle. Sie können im Master- oder Slave-Modus betrieben werden.

I/O- -Pins

Bevor die I²C-Schnittstelle verwendet werden kann, müssen die I/O-Pins mit dem folgenden Befehl für den ersten Kanal (bezeichnet als I2C) definiert werden:

SETPIN sda, scl, I2C

Gültige Pins sind SDA: GP0, GP4, GP8, GP12, GP16, GP20 oder GP28 SCL: GP1, GP5, GP9, GP13, GP17 oder GP21

Beachten Sie, dass bei der WebMite-Version I2C SDA auf GP28 nicht verfügbar ist. Und

der folgende Befehl für den zweiten Kanal (bezeichnet als I2C2):

SETPIN sda, scl, I2C2

Gültige Pins sind SDA: GP2, GP6, GP10, GP14, GP18, GP22 oder GP26 SCL: GP3, GP7, GP11, GP15, GP19 oder GP27

Wenn der I²C-Bus mit mehr als 100 kHz betrieben wird, ist die Verkabelung zwischen den Geräten von großer Bedeutung. Idealerweise sollten die Kabel so kurz wie möglich sein (um die Kapazität zu reduzieren) und die Daten- und Taktleitungen sollten nicht nebeneinander verlaufen, sondern durch ein Erdungskabel voneinander getrennt sein (um Übersprechen zu reduzieren).

Wenn die Datenleitung bei hohem Takt nicht stabil ist oder die Taktleitung unruhig ist, können die I²C-Peripheriegeräte „verwirrt" werden und den Bus sperren (normalerweise durch Halten der Taktleitung auf niedrigem Niveau). Wenn Sie keine höheren Geschwindigkeiten benötigen, ist der Betrieb mit 100 kHz die sicherste Wahl.

Mit dem Befehl I2C CHECK addr kann überprüft werden, ob ein Gerät unter der Adresse „addr" vorhanden ist. Dadurch wird die schreibgeschützte Variable MM.I2C auf 0 gesetzt, wenn ein Gerät antwortet, oder auf 1, wenn keine Antwort erfolgt.

I² C-Master-Befehle

Es gibt vier Befehle, die für den ersten Kanal (I2C) im Master-Modus verwendet werden können. Die Befehle für den zweiten Kanal (I2C2) sind identisch, außer dass der Befehl I2C2 lautet.

I2C OPEN Geschwindigkeit, Zeitüberschreitung

Aktiviert das I² C-Modul im Master-Modus. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich mit derselben Syntax auf Kanal 2 bezieht.

„speed" ist die zu verwendende Taktrate (in KHz) und muss entweder 100, 400 oder 1000 sein.

„timeout" ist ein Wert in Millisekunden, nach dessen Ablauf die Sende- und Empfangskommandos des Masters unterbrochen werden, wenn sie nicht abgeschlossen sind. Der Mindestwert beträgt 100. Der Wert Null deaktiviert das Timeout (dies wird jedoch nicht empfohlen).

I2C WRITE addr, option, sendlen, senddata [,sendata ..]

Sendet Daten an das I²C-Slave-Gerät. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich mit derselben Syntax auf Kanal 2 bezieht.

„addr" ist die I²C-Adresse des Slaves.

„option" kann 0 für den normalen Betrieb oder 1 sein, um die Kontrolle über den Bus nach dem Befehl zu behalten (eine Stoppbedingung wird nach Abschluss des Befehls nicht gesendet).

„sendlen" ist die Anzahl der zu sendenden Bytes.

„senddata" sind die zu sendenden Daten – diese können auf verschiedene Weise angegeben werden (alle gesendeten Daten werden als Bytes mit einem Wert zwischen 0 und 255 gesendet):

- Die Daten können als einzelne Bytes in der Befehlszeile angegeben werden. Beispiel: I2C WRITE &H6F, 0, 3, &H23, &H43, &H25
- Die Daten können in einem eindimensionalen Array angegeben werden, das mit leeren Klammern (d. h. ohne Dimensionen) angegeben wird. „sendlen" Bytes des Arrays werden beginnend mit dem ersten Element gesendet. Beispiel: I2C WRITE &H6F, 0, 3, ARRAY()
- Die Daten können eine String-Variable (keine Konstante) sein. Beispiel: I2C WRITE &H6F, 0, 3, STRING$

I2C READ addr, option, rcvlen, rcvbuf

Daten vom I²C-Slave-Gerät abrufen. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich unter Verwendung derselben Syntax auf Kanal 2 bezieht.

„addr" ist die I²C-Adresse des Slaves.

„option" kann 0 für den normalen Betrieb oder 1 sein, um die Kontrolle über den Bus nach dem Befehl zu behalten (eine Stoppbedingung wird nach Abschluss des Befehls nicht gesendet).

„rcvlen" ist die Anzahl der zu empfangenden Bytes.

„rcvbuf" ist die Variable oder das Array, das zum Speichern der empfangenen Daten verwendet wird – dies kann sein:

- Eine Zeichenfolgenvariable. Bytes werden als aufeinanderfolgende Zeichen in der Zeichenfolge gespeichert.
- Ein eindimensionales Array von Zahlen, das mit leeren Klammern angegeben wird. Empfangene Bytes werden in aufeinanderfolgenden Elementen des Arrays gespeichert, beginnend mit dem ersten. Beispiel: I2C READ &H6F, 0, 3, ARRAY()
- Eine normale numerische Variable (in diesem Fall muss „rcvlen" 1 sein).

I2C CLOSE

Deaktiviert das Master-I²C-Modul und versetzt die E/A-Pins in einen „nicht konfigurierten" Zustand. Dieser Befehl sendet auch ein Stopp-Signal, wenn der Bus noch gehalten wird.

I²C-Slave-Befehle

I2C SLAVE OPEN addr, send_int, rcv_int

Aktiviert das I² C-Modul im Slave-Modus. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich unter Verwendung derselben Syntax auf Kanal 2 bezieht.

„addr" ist die Slave-I²C-Adresse.

„send_int" ist die Subroutine, die aufgerufen wird, wenn das Modul erkannt hat, dass der Master Daten erwartet.

„rcv_int" ist die Subroutine, die aufgerufen wird, wenn das Modul Daten vom Master empfangen hat. Beachten Sie, dass dies beim Empfang des ersten Bytes ausgelöst wird, sodass Ihr Programm möglicherweise warten muss, bis alle Daten empfangen wurden.

I2C-SLAVE-SCHREIBEN sendlen, senddata [,sendata ..]

Sendet die Daten an den I²C-Master. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich mit derselben Syntax auf Kanal 2 bezieht.

Dieser Befehl sollte im Send-Interrupt verwendet werden (d. h. in der Subroutine „send_int", wenn der Master Daten angefordert hat). Alternativ kann in der Interrupt-Subroutine ein Flag gesetzt und der Befehl aus der Hauptprogrammschleife aufgerufen werden, wenn das Flag gesetzt ist.

„sendlen" ist die Anzahl der zu sendenden Bytes.

„senddata" sind die zu sendenden Daten. Diese können auf verschiedene Weise angegeben werden, siehe die I2C-WRITE-Befehle für Details.

I2C SLAVE READ rcvlen, rcvbuf, rcvd

Empfangen von Daten vom I²C-Master-Gerät. Der Befehl I2C bezieht sich auf Kanal 1, während der Befehl I2C2 sich mit derselben Syntax auf Kanal 2 bezieht.

Dieser Befehl sollte im Empfangsinterrupt verwendet werden (d. h. in der Subroutine „rcv_int", wenn der Master Daten gesendet hat). Alternativ kann in der Empfangsinterrupt-Subroutine ein Flag gesetzt und der Befehl aus der Hauptprogrammschleife aufgerufen werden, wenn das Flag gesetzt ist.

„rcvlen" ist die maximale Anzahl der zu empfangenden Bytes.

„rcvbuf" ist die Variable zum Empfangen der Daten. Diese kann auf verschiedene Weise angegeben werden, siehe die I2C-READ-Befehle für Details.

„rcvd" ist eine Variable, die nach Abschluss des Befehls die tatsächlich empfangene Anzahl von Bytes enthält (die von „rcvlen" abweichen kann).

I2C SLAVE CLOSE

Deaktiviert das Slave-I²-C-Modul und versetzt die externen E/A-Pins in einen „nicht konfigurierten" Zustand. Sie können dann mit SETPIN konfiguriert werden.

Fehler

Nach einem I²C-Schreib- oder Lesevorgang wird die automatische Variable MM.I2C wie folgt gesetzt, um das Ergebnis anzuzeigen:

0 = Der Befehl wurde ohne Fehler ausgeführt.

1 = Eine NACK-Antwort wurde empfangen.

2 = Befehl abgelaufen

7-Bit- -Adressierung

Die in diesen Befehlen verwendeten Standardadressen sind 7-Bit-Adressen (ohne Lese-/Schreibbit). MMBasic fügt das Lese-/Schreibbit hinzu und manipuliert es während der Übertragungen entsprechend.

Einige Anbieter stellen 8-Bit-Adressen zur Verfügung, die das Lese-/Schreibbit enthalten. Sie können feststellen, ob dies der Fall ist, da sie eine Adresse zum Schreiben auf das Slave-Gerät und eine andere zum Lesen vom Slave bereitstellen. In diesen Situationen sollten Sie nur die oberen sieben Bits der Adresse verwenden. Beispiel: Wenn die Leseadresse 9B (hex) und die Schreibadresse 9A (hex) lautet, erhalten Sie durch Verwendung nur der oberen sieben Bits die Adresse 4D (hex).

Ein weiterer Hinweis darauf, dass ein Anbieter 8-Bit-Adressen anstelle von 7-Bit-Adressen verwendet, ist die Überprüfung des Adressbereichs. Alle 7-Bit-Adressen sollten im Bereich von 08 bis 77 (hex) liegen. Wenn Ihre Slave-Adresse größer als dieser Bereich ist, hat Ihr Anbieter wahrscheinlich eine 8-Bit-Adresse bereitgestellt.

Beispiele

Als Beispiel für eine einfache Kommunikation, bei der der Raspberry Pi Pico der Master ist, liest und zeigt das folgende Programm die aktuelle Uhrzeit (Stunden und Minuten) an, die von einem PCF8563-Echtzeituhr-Chip gespeichert wird, der an den zweiten I2C-Kanal angeschlossen ist:

```
DIM AS INTEGER RData(2)                    ' hier werden die empfangenen Daten
gespeichert SETPIN GP6, GP7, I2C2          ' Weist die I/O-Pins für I2C2 zu
I2C2 OPEN 100, 1000                        ' Öffnen des I2C-Kanals
I2C2 WRITE &H51, 0, 1, 3                   ' Setze das erste Register auf 3
I2C2 READ &H51, 0, 2, RData()              ' Zwei Register lesen
I2C2 CLOSE                                 ' Schließe den I2C-Kanal
PRINT "Die Zeit ist " hex$(RData(1),2) ":" hex$(RData(0),2) 'BCD-codierte Daten
anzeigen
```

Dies ist ein Beispiel für die Kommunikation zwischen zwei Raspberry Pi Picos. Der Master sendet die Zeichen „A" bis „Z" und sendet nach jeder Übertragung eine Anfrage an den Slave und gibt die Antwort aus. Der Slave liest das vom Master gesendete Byte und gibt es aus. Als Antwort auf die Anfrage des Masters sendet er die aktuelle Uhrzeit.

Zuerst der Master:

```
SETPIN GP2, GP3, I2C2 I2C2
OPEN 100, 1000
  FOR i = 65 to 90
    a$ = CHR$(i)
   I2C2 WRITE &H50, 0, 1, a$
   PAUSE 500
   I2C2 LESEN &H50, 0, 8, a$
   DRUCKEN a$
   PAUSE 500
NECT i
```

Dann der Slave:

```
SETPIN GP2, GP3, I2C2
I2C2 SLAVE OPEN &H50, tint, rint DO :
LOOP

SUB rint
   LOCAL count, a$
   I2C2 SLAVE READ 10, a$, count
   PRINT LEFT$(a$, count) END
SUB

SUB tint
   LOCAL a$ = Time$
   I2C2 SLAVE WRITE LEN(a$), a$ END
SUB
```

Anhang C
1-Wire-Kommunikation

Das 1-Wire-Protokoll wurde von Dallas Semiconductor entwickelt, um über eine einzige Signalleitung mit Chips zu kommunizieren. Diese Implementierung wurde von Gerard Sexton für MMBasic geschrieben.

Es gibt drei Befehle, die Sie verwenden können:

ONEWIRE RESET pin	Setzen Sie den 1-Wire-Bus
zurück. ONEWIRE WRITE pin, flag, length, data [, data…]	Senden einer Anzahl von
Bytes ONEWIRE READ pin, flag, length, data [, data…]	Eine Anzahl von Bytes
abrufen	

Wobei:

Pin – Der zu verwendende I/O-Pin. Es kann jeder Pin sein, der für digitale I/O

geeignet ist. Flag – Eine Kombination der folgenden Optionen:

1 – Reset vor Befehl senden

2 – Reset nach Befehl senden

4 – Nur ein Bit statt eines Bytes an Daten senden/empfangen

8 - Nach dem Befehl einen starken Pullup auslösen (der Pin wird auf High gesetzt und Open Drain deaktiviert) Länge -

Länge der zu sendenden oder zu empfangenden Daten

data – Zu sendende Daten oder zu empfangende Variable.
Die Anzahl der Datenelemente muss mit dem Parameter „length“ übereinstimmen.

Die automatische Variable MM.ONEWIRE gibt „true“ zurück, wenn ein Gerät gefunden wurde

Nach Ausführung des Befehls wird der I/O-Pin auf den nicht konfigurierten Zustand gesetzt, sofern nicht die Flag-Option 8 verwendet wird.

Wenn ein Reset angefordert wird, gibt die automatische Variable MM.ONEWIRE den Wert „true“ zurück, wenn ein Gerät gefunden wurde. Dies geschieht mit dem Befehl ONEWIRE RESET und den Befehlen ONEWIRE READ und ONEWIRE WRITE, wenn ein Reset angefordert wurde (Flag = 1 oder 2).

Das 1-Wire-Protokoll wird häufig für die Kommunikation mit dem Temperaturmesssensor DS18B20 verwendet. Zu diesem Zweck enthält MMBasic die Funktion TEMPR(), die eine bequeme Methode zum direkten Auslesen der Temperatur eines DS18B20 ohne Verwendung dieser Funktionen bietet.

Anhang D
SPI-Kommunikation

Das Kommunikationsprotokoll Serial Peripheral Interface (SPI) wird zum Senden und Empfangen von Daten zwischen integrierten Schaltkreisen verwendet. Der Raspberry Pi Pico fungiert als Master (d. h. er generiert den Takt).

I/O- -Pins

Bevor eine SPI-Schnittstelle verwendet werden kann, müssen die I/O-Pins für den Kanal mit den folgenden Befehlen zugewiesen werden. Für den ersten Kanal (bezeichnet als SPI) lautet der Befehl:

SETPIN rx, tx, clk, SPI

Gültige Pins sind	RX:	GP0, GP4, GP16 oder GP20
	TX:	GP3, GP7 oder GP19
	CLK:	GP2, GP6 oder GP18

Und der folgende Befehl für den zweiten Kanal (bezeichnet als SPI2) lautet: SETPIN rx, tx, clk,

SPI2

Gültige Pins sind	RX:	GP8, GP12 oder GP28
	TX:	GP11, GP15 oder GP27
	CLK:	GP10, GP14 oder GP26

TX sind Daten vom Raspberry Pi Pico und RX sind Daten an ihn.

Beachten Sie, dass in der WebMite-Version SPI1 und SPI2 auf GP20 bis GP28 nicht verfügbar sind.

SPI- t offen

Um die SPI-Funktion nutzen zu können, muss der SPI-Kanal zunächst geöffnet werden.

Die Syntax zum Öffnen des ersten SPI-Kanals lautet (verwenden Sie SPI2 für den zweiten Kanal):

```
SPI OPEN Geschwindigkeit, Modus, Bits
```

Wobei:

- „speed“ ist die angeforderte Geschwindigkeit für den SPI-Takt. Es kann jeder beliebige Wert angefordert werden, die Firmware wählt die nächste erreichbare Geschwindigkeit aus, die gleich oder langsamer als die angeforderte Geschwindigkeit ist. Die tatsächlich eingestellte Geschwindigkeit ist CPU-Geschwindigkeit/2 geteilt durch 1 bis 256.
- „mode“ ist eine einzelne Ziffer, die den Übertragungsmodus angibt – siehe Übertragungsformat unten.
- „bits“ ist die Anzahl der zu sendenden/empfangenden Bits. Dies kann eine beliebige Zahl im Bereich von 4 bis 16 Bits sein.
- Es liegt in der Verantwortung des Programms, den CS-Pin (Chip Select) bei Bedarf separat zu manipulieren.

Übertragungs sformat

Das signifikanteste Bit wird zuerst gesendet und empfangen. Das Format der Übertragung kann durch den „Modus“ wie unten gezeigt festgelegt werden. Modus 0 ist das gängigste Format.

Modus	Beschreibung	CPOL	CPHA
0	Der Takt ist hochaktiv, die Daten werden an der steigenden Flanke erfasst und an der fallenden Flanke ausgegeben.	0	0
1	Der Takt ist hochaktiv, die Daten werden an der fallenden Flanke erfasst und an der steigenden Flanke ausgegeben.	0	1
2	Der Takt ist aktiv niedrig, Daten werden an der fallenden Flanke erfasst und an der steigenden Flanke ausgegeben.	1	0
3	Die Uhr ist aktiv niedrig, Daten werden an der steigenden Flanke erfasst und an der fallenden Flanke ausgegeben.	1	1

Eine ausführlichere Erklärung finden Sie unter: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Standard- -Senden/Empfangen

Wenn der erste SPI-Kanal geöffnet ist, können Daten mit der SPI-Funktion gesendet und empfangen werden (verwenden Sie SPI2 für den zweiten Kanal). Die Syntax lautet:

```
received_data = SPI(data_to_send)
```

Beachten Sie, dass bei einer einzelnen SPI-Transaktion Daten gesendet und gleichzeitig Daten vom Slave empfangen werden. „data_to_send” sind die zu sendenden Daten, und die Funktion gibt die während der Transaktion empfangenen Daten zurück. „data_to_send” kann eine Ganzzahl, eine Gleitkommavariable oder eine Konstante sein.

Wenn Sie keine Daten senden möchten (d. h. nur empfangen möchten), kann für die zu sendenden Daten eine beliebige Zahl (z. B. Null) verwendet werden. Wenn Sie die empfangenen Daten nicht verwenden möchten, können Sie sie einer Variablen zuweisen und ignorieren.

Massen en senden/empfangen

Daten können auch in großen Mengen gesendet werden (verwenden Sie SPI2 für den zweiten Kanal):

```
SPI WRITE nbr, data1, data2, data3, … etc
```

oder

```
SPI WRITE nbr, string$
```

oder

```
SPI WRITE nbr, array()
```

Bei der ersten Methode ist „nbr" die Anzahl der zu sendenden Datenelemente und die Daten sind die Ausdrücke in der Argumentliste (d. h. „data1", „data2" usw.). Die Daten können eine Ganzzahl, eine Gleitkommavariable oder eine Konstante sein.

Bei der zweiten oder dritten oben aufgeführten Methode sind die zu sendenden Daten in „string$" oder dem Inhalt von „array()" enthalten (das muss ein eindimensionales Array aus Ganzzahlen oder Gleitkommazahlen sein). Die Zeichenfolgenlänge oder die Größe des Arrays muss gleich oder größer als nbr sein. Alle vom Slave zurückgegebenen Daten werden verworfen.

Daten können auch in großen Mengen empfangen werden (verwenden Sie SPI2 für den zweiten Kanal):

```
SPI READ nbr, array()
```

Dabei ist „nbr" die Anzahl der zu empfangenden Datenelemente und array() ein eindimensionales Array mit Ganzzahlen, in dem die empfangenen Datenelemente gespeichert werden. Dieser Befehl sendet Nullen, während die Daten vom Slave gelesen werden.

SPI- -Close

Bei Bedarf kann der erste SPI-Kanal wie folgt geschlossen werden (die I/O-Pins werden auf inaktiv gesetzt):

```
SPI CLOSE
```

Verwenden Sie SPI2 für den zweiten Kanal.

Beispiele

Das folgende Beispiel zeigt, wie der SPI-Port für allgemeine E/A verwendet werden kann. Es sendet einen Befehl 80 (hex) und empfängt zwei Bytes vom Slave-SPI-Gerät unter Verwendung der Standard-Sende-/Empfangsfunktion:

```
PIN(10) = 1 : SETPIN 10, DOUT          ' Pin 10 wird als Freigabesignal verwendet SETPIN
GP20, GP3, GP2, SPI                    ' I/O-Pins zuweisen
SPI OPEN 5000000, 3, 8                 ' Geschwindigkeit beträgt 5 MHz und die Datengröße
8 Bit PIN(10) = 0                      ' Freigabeleitung aktivieren (aktiv niedrig)
junk = SPI(&H80)                       ' Befehl senden und Rückgabe ignorieren
byte1 = SPI(0)                         ' erstes Byte vom Slave abrufen
byte2 = SPI(0)                         ' Zweites Byte vom Slave abrufen
PIN(10) = 1                            ' Slave abwählen
SPI CLOSE                              ' und Kanal schließen
```

Das Folgende ähnelt dem oben angegebenen Beispiel, jedoch erfolgt die Übertragung diesmal mithilfe der Befehle zum Massenversand/-empfang:

```
OPTION BASE 1                          ' Unser Array beginnt mit dem Index 1
DIM-Daten%(2)                          ' Array zum Empfangen der Daten definieren
SETPIN GP20, GP3, GP2, SPI             ' Weisen Sie die E/A-Pins zu
PIN(10) = 1 : SETPIN 10, DOUT          ' Pin 10 wird als Freigabesignal verwendet SPI
OPEN 5000000, 3, 8                     ' Geschwindigkeit beträgt 5 MHz, 8 Bit Daten
PIN(10) = 0                            ' Freigabeleitung aktivieren (aktiv niedrig)
SPI WRITE 1, &H80                      ' Befehl senden
SPI READ 2, data%()                    ' Zwei Bytes vom Slave empfangen
PIN(10) = 1                            ' Slave abwählen
SPI CLOSE                              ' und Kanal schließen
```

Anhang E
Regex-Syntax

Die alternativen Formen der Funktionen INSTR() und LINSTR() können einen regulären Ausdruck als Suchmuster verwenden. Die alternativen Formen der Funktionen lauten:

INSTR([start],text$, search$ [,size]) LINSTR(text%(),search$ [,start] [,size]

In beiden Fällen bewirkt die Angabe des Größenparameters, dass die Firmware die Suchzeichenfolge als regulären Ausdruck interpretiert. Der Größenparameter ist eine Gleitkomma- oder Ganzzahlvariable, die von der Firmware verwendet wird, um die Größe einer übereinstimmenden Zeichenfolge zurückzugeben. Wie in MMBasic implementiert, müssen Sie die zurückgegebenen ***Start-*** und Größenwerte auf die MID$-Funktion anwenden, um die übereinstimmende Zeichenfolge zu extrahieren. z. B.

IF start THEN match$=MID$(text$,start,size) ELSE match$="" ENDIF

Die Syntax regulärer Ausdrücke kann je nach Implementierung leicht variieren. Dieser Anhang ist eine Zusammenfassung der Syntax und der unterstützten Operationen, die in der MMBasic-Implementierung verwendet werden.

Hinweis: Die Verwendung der Syntax für reguläre Ausdrücke mit OPTION ESCAPE sollte möglichst vermieden werden, da dies sonst zu Verwirrung führen kann!

Anker

^ Beginn der Zeichenfolge
$ Ende der Zeichenfolge
\b Wortgrenze
\B Keine Wortgrenze

Qualifizierer

* 0 oder mehr (nicht maskiert)
+ 1 oder mehr
? 0 oder 1
{3} Genau 3
{3,} 3 oder mehr
{,5} 5 oder weniger
{3,5\} 3,4 oder 5

Gruppen und Bereiche

(a|b|c) a oder b
oder c (…) Gruppe
[abc] Bereich (a oder b oder
c) [^abc] Nicht (a oder b oder c]
[a-q] Kleinbuchstaben a bis q [A-Q]
Großbuchstaben A bis Q [0-7] Ziffern

Zeichenklassen

\w Ziffern und Buchstaben plus _
\W Nicht-Alphanumerische Zeichen
\s Leerzeichen \t \f \r \n \v Leerzeichen
\S Nicht-Leerzeichen
\d Ziffern
\D Nicht-Ziffern
\xXX hexadezimal codiertes Byte

Escape-Zeichen, die erforderlich sind, um normale Zeichen abzugleichen

\^ entspricht ^ (Caret)
\. entspricht dem Zeichen „ “ (Punkt)
\ entspricht * (Sternchen)
\$ entspricht $ (Dollar)
\[entspricht [(linke Klammer)
\\ entspricht \ (Backslash)
\? to match ? (Fragezeichen)
\{ entspricht { (linke geschweifte Klammer)
\} entspricht } (rechte geschweifte Klammer)
\| entspricht | (Pipe)
\(entspricht (
\) zu übereinstimmen) (rechte Klammer)
\ + entspricht + (Plus)

Übereinstimmungsregeln

Nicht-Sonderzeichen entsprechen sich selbst
. passt auf jedes Zeichen
Eine Wortgrenze befindet sich am Anfang oder Ende einer Zeichenfolge oder dort, wo ein \w-Zeichen ein \W-Zeichen angrenzt.

Einschränkungen

Anker innerhalb einer Gruppe werden nicht unterstützt.

Beispielsweise werden (^hello) oder (hello$) nicht wie erwartet mit „hello” am Anfang oder Ende der Zeile abgeglichen.

Anker außerhalb der Gruppe sind jedoch zulässig. Beispielsweise ^(hello) oder (hello)$ werden

Beispielausdruck zum Abgleichen einer IP-Adresse.

```
„[\d]+\.[\d]+\.[\d]+\.[\d]+“
```

Verwendung regulärer Ausdrücke mit OPTION ESCAPE

Wenn ein regulärer Ausdruck wörtlich in die Funktion INSTR oder LINSTR eingebettet ist, wird jede Syntax, die das Escape-Zeichen „\“ verwendet, korrekt verarbeitet, ohne dass der Backslash weiter escaped werden muss, unabhängig davon, ob OPTION ESCAPE verwendet wird oder nicht. z. B.

```
 ? instr("test123","\d",size)
```

Die Funktionen INSTR / LINSTR deaktivieren OPTION ESCAPE vorübergehend, während der reguläre Ausdruck gelesen wird.

Wenn der reguläre Ausdruck jedoch als String-Variable übergeben wird und OPTION ESCAPE aktiviert ist, wenn Sie den regulären Ausdruck der Variable zuweisen, müssen Sie alle Backslashes im regulären Ausdruck escapen. Beispiel:

```
s$=”\\d”
? instr("test123",s$,size)
```

Anhang F
Das PIO-Programmierpaket

Einführung in das PIO-

Der RP2040 und der RP2350 verfügen über zahlreiche integrierte Peripheriegeräte wie PWM, UART, ADC und SPI. Mit Hilfe von PIOs lassen sich spezielle Funktionen/Peripheriegeräte wie serielle Hochgeschwindigkeits-Datenschnittstellen und Bitströme hinzufügen.

PIOs können als reduzierte, hochspezialisierte CPU-Kerne betrachtet werden. Der RP2040 enthält zwei PIO-Blöcke, während der RP2350 über drei Blöcke verfügt. MMBasic bezeichnet sie in Übereinstimmung mit der Raspberry Pi-Dokumentation als PIO0, PIO1 und PIO2. Die PIOs laufen völlig unabhängig vom Hauptsystem und voneinander und sind mit einem Durchsatz von bis zu 32 Bit pro Taktzyklus extrem schnell.

PIOs implementieren Zustandsmaschinen. Bevor eine Zustandsmaschine ihr Programm ausführen kann, muss das Programm in den PIO-Speicher geschrieben und die Zustandsmaschine konfiguriert werden.

Dieser Anhang beschreibt die Unterstützung, die MMBasic bei der Verwendung von PIOs bieten kann. Er enthält keine Erklärung, wie PIO-Zustandsmaschinenprogramme geschrieben werden. Um besser zu verstehen, wie diese funktionieren, lesen Sie bitte den folgenden Thread „PIO explained PICOMITE" im Forum von thebackshed.com: https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=15385

Verfügbarkeit von PIOs für die

PicoMite-Plattform	I2S Audio	PIO0	PIO1	PIO2
2040		✓	✓	
2040	ja	X	✓	
2350		✓	✓	✓
2350	ja	✓	✓	X
2040 VGA		X	✓	
2040 VGA	Ja	X	✓	
2350 VGA		X	✓	✓
2350 VGA	ja	X	✓	X
2350 HDMI		✓	✓	✓
2350 HDMI	Ja	✓	✓	X
WebMite2040		✓	X	
WebMite2040	ja	X	X	
WebMite2350		✓	X	✓
WebMite2350	ja	✓	X	X

✓ = VERFÜGBAR X = NICHT VERFÜGBAR

Übersicht über die PIO-

Ein einzelner PIO-Block verfügt über vier unabhängige Zustandsmaschinen. Alle vier Zustandsmaschinen teilen sich einen einzigen 32-Befehls-Programmbereich des Flash-Speichers. Dieser Speicher ist vom Hauptsystem aus schreibgeschützt, verfügt jedoch über vier Leseports, einen für jede Zustandsmaschine, sodass jede unabhängig mit ihrer eigenen Geschwindigkeit darauf zugreifen kann. Jede Zustandsmaschine hat ihren eigenen Programmzähler.

Jede Zustandsmaschine verfügt außerdem über zwei 32-Bit-„Scratchpad"-Register, X und Y, die als temporäre Datenspeicher verwendet werden können.

Auf die E/A-Pins wird über ein Eingangs-/Ausgangs-Zuordnungsmodul zugegriffen, das auf 32 Pins zugreifen kann (bei RP2040 jedoch auf 30 beschränkt). Alle Zustandsmaschinen können unabhängig voneinander und gleichzeitig auf alle Pins zugreifen.

MMBasic kann Daten in das Eingangsende eines 4-Wort-32-Bit-breiten TX-FIFO-Puffers schreiben. Die Zustandsmaschine kann dann PULL verwenden, um das Ausgangswort des FIFO in das OSR (Output Shift Register) zu verschieben. Sie kann auch OUT verwenden, um jeweils 1-32 Bits aus dem OSR in das Ausgangs-Mapping-Modul oder andere Ziele zu verschieben. Mit AUTOPULL können Daten automatisch gezogen werden, bis der TX-FIFO leer ist oder einen voreingestellten Pegel erreicht.

MMBasic kann Daten vom Ausgangsende eines 4-Wort-RX-FIFO-Puffers mit einer Breite von 32 Bit lesen. Die Zustandsmaschine kann dann IN verwenden, um jeweils 1 bis 32 Datenbits vom Eingangsabbildungsmodul in das ISR (Input Shift Register) zu verschieben. Anschließend kann sie PUSH verwenden, um den Inhalt des ISR in den FIFO zu verschieben. Mit AUTOPUSH können Daten automatisch verschoben werden, bis der RX-FIFO voll ist oder einen voreingestellten Pegel erreicht.

Die FIFO-Puffer können so umkonfiguriert werden, dass sie einen unidirektionalen 8-Wort-32-Bit-FIFO bilden. Die Puffer ermöglichen die Übertragung von Daten zu und von den Zustandsmaschinen, ohne dass das System oder die Zustandsmaschine aufeinander warten müssen.

Jede der vier Zustandsmaschinen im PIO verfügt über vier zugehörige Register:

- CLKDIV ist der Taktteiler, der über einen 16-Bit-Ganzzahlteiler und einen 8-Bit-Bruchteilteiler verfügt. Dieser legt fest, wie schnell die Zustandsmaschine läuft. Er teilt den Hauptsystemtakt herunter.
- EXECCTRL enthält Informationen zur Steuerung der Übersetzung und Ausführung des Programmspeichers.
- SHIFTCTRL steuert die Anordnung und Verwendung der Schieberegister.
- PINCTRL steuert, welche GPIO-Pins wie verwendet werden.

Die vier Zustandsmaschinen eines PIO haben gemeinsamen Zugriff auf seinen Block mit 8 Interrupt-Flags. Jede Zustandsmaschine kann jedes Flag verwenden. Sie können diese setzen, zurücksetzen oder auf ihre Änderung warten. Auf diese Weise können sie bei Bedarf synchron ausgeführt werden. Die unteren vier Flags sind auch für das Hauptsystem zugänglich, sodass der PIO von MMBasic aus gesteuert werden kann oder Interrupts zurückgeben kann.

DMA kann verwendet werden, um Informationen über seinen FIFO vom Speicher des RP2040 zum und vom PIO-Block zu übertragen.

Ein PIO verfügt über neun mögliche Programmierbefehle, aber es kann viele Variationen für jeden einzelnen geben. Zum Beispiel kann Mov bis zu 8 Quellen, 8 Ziele, 3 Prozessoperationen während des Kopiervorgangs mit optionalen Verzögerungs- und/oder Side-Set-Operationen haben!

- Jmp Springt zu einer absoluten Adresse im Programmspeicher, wenn eine Bedingung wahr ist (oder sofort).
- Warten Den Betrieb der Zustandsmaschine anhalten, bis eine Bedingung wahr ist.
- In Verschiebt eine Anzahl von Bits aus einer Quelle in den ISR.
- Aus Verschiebt eine Anzahl von Bits aus dem OSR an ein Ziel.
- Push Den Inhalt des ISR als einzelnes 32-Bit-Wort in den RX-FIFO schieben.
- Pull Laden Sie ein 32-Bit-Wort aus dem TX-FIFO in den OSR.
- Bewegen Kopieren Sie das Datum von einer Quelle zu einem Ziel.
- Irq Setzen oder Löschen eines Interrupt-Flags.
- Set Daten sofort an einen Zielort schreiben.

Alle Befehle sind 16-Bit-Befehle und enthalten sowohl den Befehl als auch alle damit verbundenen Daten. Alle Befehle werden in einem Taktzyklus ausgeführt, es ist jedoch möglich, zwischen einem Befehl und dem nächsten eine Verzögerung von mehreren Leerlauf-Taktzyklen einzufügen.

Zusätzlich gibt es eine Funktion namens „Side-Set", mit der ein Wert in einige vordefinierte Ausgangspins geschrieben werden kann, während ein Befehl aus dem Speicher gelesen wird. Dies ist für das Programm transparent.

Programmierung der PIO v

Die PicoMite-Firmware programmiert den PIO-Zustandsmaschinen-Speicher mit einer von drei Methoden. Jede Methode wird anhand eines Beispiels für genau dasselbe Programm erläutert, das einen der GPIO-Pins des Raspberry Pi Pico umschaltet. Welcher GPIO-Pin umgeschaltet wird, wird durch die Konfiguration bestimmt, nicht durch das Programm selbst.

PIO ASSEMBLE

Dieser Befehl wird verwendet, um mit dem integrierten Assembler das Programm aus Mnemonik zu generieren und es dann direkt in den PIO-Speicher zu schreiben.

```
PIO ASSEMBLE 1,".program test"                'Ein Programm muss einen Namen
haben PIO ASSEMBLE 1,".line 0"                'das Programm beginnt bei Zeile 0
PIO ASSEMBLE 1,"SET PINDIRS,1"                'GPIO-Leitung auf Ausgabe setzen
PIO ASSEMBLE 1,"label:"                       'Definiere eine Bezeichnung namens
„label"
PIO ASSEMBLE 1, „SET PINS,1"                  'GPIO-Pin auf High setzen
PIO ASSEMBLE 1,"SET PINS,0"                   'GPIO-Pin auf niedrig setzen
PIO ASSEMBLE 1, „JMP label"                   'Springe zu „label"
PIO ASSEMBLE 1, „.end program list"           'Programm beenden, list=Ergebnis anzeigen
```

Der Assembler ermöglicht auch ein besser lesbares Format wie dieses

```
PIO ASSEMBLE 1,".program test"          Ein Programm muss einen Namen haben
.Zeile 0                                'Programm bei Zeile 0 starten SET
   PINDIRS,1                            'GPIO-Leitung auf Ausgabe
   einstellen
.LABEL label:                           'Definiere eine Bezeichnung namens
   „label" SET PINS,1                   'GPIO-Pin auf High setzen
   SET PINS,0                           'GPIO-Pin auf Low setzen
```

```
   JMP label                          'Springe zu „label"
.end Programmliste                    'Programm beenden, list=Ergebnis anzeigen
```

PIO-PROGRAMMZEILE

Mit diesem Befehl können 16-Bit-Werte für einzelne Zeilen (Speicherplätze) im PIO-Speicher programmiert werden.

```
PIO PROGRAM LINE 1,0,&hE081                  'Pin-Ausgang einstellen
PIO-PROGRAMMZEILE 1,1,&hE001                 'SET Pin hoch
PIO-PROGRAMMIERLEITUNG 1,2,&hE000            'Pin niedrig setzen
PIO PROGRAM LINE 1,3,&h0001                  'JMP zu Zeile 1
```

PIO PROGRAM

Dieser Befehl schreibt alle 32 Zeilen in einem PIO aus einem Array. Dies ist nützlich, sobald ein PIO-Programm debuggt wurde. Es ist äußerst kompakt.

```
DIM a%(7)=(&h0001E0000E001E081,0,0,0,0,0,0,0) PIO
PROGRAM 1,a%()
```

-PIO konfigurieren

Die PicoMite-Firmware kann jede Zustandsmaschine einzeln konfigurieren. Die Konfiguration ermöglicht es, dass zwei Zustandsmaschinen genau dieselben Programmzeilen ausführen (z. B. eine SPI-Schnittstelle), jedoch mit unterschiedlichen GPIO-Pins und unterschiedlichen Geschwindigkeiten arbeiten. Es gibt mehrere Konfigurationsfelder.

FREQUENZ

Die PicoMite-Firmware enthält eine Standardkonfiguration für jedes Konfigurationsfeld, mit Ausnahme der Frequenz. Die Frequenz wird durch einen 16-Bit-CLKDIV-Teiler vom ARM-Taktgeber festgelegt. Beispiel: Wenn OPTION CPUSPEED 126000 eingestellt ist, kann die PIO mit Geschwindigkeiten zwischen 126 MHz und 1,922 kHz (126000000 / 65536) laufen. Beachten Sie, dass höhere CPU-Geschwindigkeiten (Übertaktung) sich direkt auf die Frequenz der Zustandsmaschine auswirken.

PIN-STEUERUNG

Die PicoMite-Firmware verwendet standardmäßig die GPIO-Pins für MMBasic. Damit PIO die Kontrolle über einen GPIO-Pin übernehmen kann, muss MMBasic ihn wie unten gezeigt PIO zuweisen.

SETPIN GPxx,PIOx (z. B. SETPIN gp0,pio1)

Eine Zustandsmaschine kann den Zustand eines Pins SETZEN (SET ist eine Zustandsmaschinenanweisung), aber auch serielle Daten mit der OUT-Anweisung an einen oder mehrere GPIO-Pins ausgeben. Oder serielle Daten mit der IN-Anweisung lesen. Und GPIO-Pins können als Nebeneffekt jeder Zustandsmaschinenanweisung (SIDE SET) gesetzt werden. Für jede Schnittstellenmethode können der Zustandsmaschine unterschiedliche Pins zugeordnet werden.

Es ist wichtig zu verstehen, dass diese Befehle auf aufeinanderfolgenden Pins funktionieren. Das bedeutet, dass es einen Bereich von Pins gibt, die gesteuert werden können, beginnend mit der niedrigsten GPx-Pin-Nummer (z. B. GP0), und dass benachbarte Pins einbezogen werden können (insgesamt bis zu 5 Pins). GP0, GP1, GP2 ist also ein gültiger Satz von IO-Pins. GP0, GP1, GP6 ist es nicht. Berücksichtigen Sie dies bei der Entwicklung einer PIO-Anwendung.

Die Zuweisung von GPIO-Pins zu einer Zustandsmaschine erfolgt über die PIO-Hilfsfunktion PINCTRL:

PIO(PINCTRL a,b,c,d,e,f,g)
- a/ Anzahl der SIDE SET-Pins (0...5), SIDE SET kann 5 Pins gleichzeitig schreiben b/ Anzahl der SET-Pins (0...5), SET kann 5 Pins gleichzeitig schreiben
- c/ die Anzahl der OUT-Pins (0...31), OUT kann 32 Pins gleichzeitig schreiben
- d/ der niedrigste Pin für IN-Pins (GP0 .....GP31) IN kann bis zu 32 Pins gleichzeitig lesen
- e/ der niedrigste Pin für SIDE SET (GP0.....GP31)
- f/ der unterste Pin für SET (GP0 .....GP31)
- g/ der unterste Pin für OUT (GP0 .....GP31)

Die Bereiche für die verschiedenen Funktionen (SET/OUT/SIDESET/IN) können sich überschneiden, identisch sein

oder nebeneinander liegen. AUSFÜHRUNGSSTEUERUNG

Das Ausführungssteuerungsregister EXECCTRL konfiguriert den Programmablauf. Es gibt ein Feld, das einen GPIO Pin mit einem bedingten Sprung (JMP-Befehl) verbindet, und Felder, die die Zeilenadresse des Hauptprogramm-Loops Anfang (.WRAP TARGET) und Ende (.WRAP) enthalten.

Wenn wir möchten, dass sich der Programmablauf als Reaktion auf den Status eines GPIO-Pins ändert, wird ein JMP-PIN verwendet. Der spezifische Pin wird in der Ausführungssteuerungskonfiguration zugewiesen (es kann nur 1 Pin pro Zustandsmaschine geben) und der JMP erfolgt nur, wenn der Pin hoch ist.

Das Zustandsmaschinenprogramm beginnt am Anfang und läuft bis zum Ende. Im obigen Demoprogramm

Das Programm durchläuft eine Schleife vom Ende zum Anfang unter Verwendung einer (bedingungslosen) JMP-Anweisung. Eine Alternative zur Verwendung der JMP-Anweisung besteht darin, den Anfang der Schleife (WRAP TARGET = Zeile 1) und das Ende der Schleife (WRAP = Zeile 2) zu definieren und die Zustandsmaschine so zu konfigurieren, dass sie nur die dazwischen liegenden Anweisungen ausführt. Die JMP-Anweisung in Zeile 3 ist bei Verwendung von WRAP/WRAP TARGET überflüssig.

PIO(EXECCTRL a,b,c)

a/ der GPIO-Pin für bedingtes JMP	(z. B. GP0)
b/ die WRAP TARGET-Zeilennummer	(z. B. 1)
c/ die WRAP-Zeilennummer	(z. B. 2)

SHIFT-STEUERUNG

Die Befehle IN und OUT verschieben Daten vom FIFO-Register zu den GPIO-Pins. Zwischen MMBasic und dem PIO können 32-Bit-Worte übertragen werden. Da sowohl die ARM-Kerne als auch die PIO-Mikrocontroller unabhängig voneinander arbeiten, werden die Daten über FIFOs ausgetauscht. Der ARM (MMBasic) legt Daten im FIFO ab, der PIO liest sie. Dabei wird der TX-FIFO verwendet. Umgekehrt wird der RX-FIFO verwendet. Die FIFOs sind normalerweise 4 Wörter tief, können aber auch als einzelner 8 Wörter tiefer RX- oder TX-FIFO konfiguriert werden.

Der PIO kann Daten im RX-FIFO von der MSB-Seite oder von der LSB-Seite „verschieben". Dies wird mit dem IN SHIFTDIR-Bit eingestellt. Ähnlich verhält es sich mit dem OUT SHIFTDIR-Bit für OUT-Daten. Die Autopull- und Autopush-Flags bestimmen in Kombination mit den Pull- und Push-Schwellenwerten, wann der FIFO aufgefüllt wird.

Im RP2350 können die FIFOs auch als einzelne Register verwendet werden, was eine flexiblere Kommunikation zwischen MMBasic und den Zustandsmaschinen ermöglicht. Dies wird durch FJOIN_RX_GET und FJOIN_RX_PUT im SHIFTCTRL-Register erreicht.

PIO(SHIFTCTRL a,b,c,d,e,f,g,h)

a/ Push-Schwelle	(1..31 Bits vor autoPUSH in ISR verschoben)
b/ Pull-Schwelle	(1..31 Bits werden vor autoPULL aus OSR verschoben) c/
autopush	(1 = automatisches Drücken zulassen)
d/ AutoPull	(1 = automatisches Ziehen zulassen)
e/ IN-Shiftdir	(1 = MSB verschieben, 0 = LSB verschieben)
f/ OUT-shiftdir	(1 = MSB verschieben, 0 = LSB verschieben)
g/ fjoin_tx	(TX- und RX-FIFO zu 1 RX-FIFO verbinden)
h/ fjoin_rx	(TX- und RX-FIFO zu 1 TX-FIFO verbinden)
i/ fjoin_rx_get	(1=ARM schreibt einzelne RX-FIFO-Register, nur 2350)
j/ fjoin_rx_put	(1=ARM liest einzelne RX-FIFO-Register, nur 2350)

SCHREIBEN DER ZUSTANDSMASCHINENKONFIGURATION

Eine Zustandsmaschinenkonfiguration wird mit dem folgenden Befehl geschrieben:

PIO INIT MACHINE a,b,c,d,e,f,g

a/ die PIO	(0,1 oder 2 (nur 2350))
b/ die Zustandsmaschinen-Nummer	(0...3)
c/ Frequenz	(CPUSPEED/65536...CPUSPEED in Hz)
d/ Pinsteuerungswert	(PIO(PINCTRL .....))
e/ Exekutionssteuerungswert	(PIO(EXECCTRL .....))
f/ Schaltsteuerungswert	(PIO(SHIFCTRL.....))
g/ Startadresse	(0.....31, die Zeile, in der die Zustandsmaschine mit der Ausführung beginnt, kann eine Bezeichnung sein)

SCHREIBEN DER ZUSTANDSMASCHINENKONFIGURATION IN KOMPAKTER FORM

Eine Zustandsmaschinenkonfiguration wird mit diesem einzigen Befehl konfiguriert und geschrieben:

PIO CONFIGURE a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,aa,bb,cc

a/ pio	(0,1 oder 2 (nur 2350))
b/ sm	(0....3)
c/ Frequenz	(CPUSPEED/65535 bis zu CPUSPEED in Hz)
d/ Startadresse	(0..31)
e/ Seitenbasis	(SIDE SET-Bereich beginnt mit GPx)
f/ sidesetno	(0..5.....Auswirkung auf DELAY-Bereich)
g/ sidesetout	(1=automatisch die oben definierten Pins als Ausgang festlegen)
h/ setbase	(SET-Bereich beginnt mit GPy)
i/ setno	(0....5)
j/ -Ausgabe	(1=automatisch die oben definierten Pins als Ausgang festlegen)
k/ outbase	(OUT/MOV-Bereich beginnt mit GPz)
l/ outno	(0....5)

m/ outout (1=automatisch die oben definierten Pins als Ausgang festlegen)
n/ inbase (IN-Bereich beginnt bei GPxx)
o/ jmppin (Bedingter JMP-Pin ist GPyy)
p/ wraptarget (0..31 = Zeilennummer für WRAP TARGET)
q/ wrap (0..31 = Zeilennummer für WRAP)
r/ sideenable (1 = nur Pins explizit aktualisieren, wenn Side Set in der Anweisung enthalten ist) s/ sidepindir (1 = Side Set ändert die Pin-Richtung, nicht den Pin-Status)
t/ pushthreshold (1..31 Bits vor autoPUSH in ISR verschoben)
u/ pullthreshold (1..31 Bits werden vor autoPULL aus OSR verschoben) v/ autopush (1 = automatisches Drücken zulassen)
w/ autopull (1 = automatisches Ziehen zulassen)
x/ inshiftdir (1 = Verschiebung MSB→LSB, 0 = Verschiebung LSB→MSB)
y/ outshiftdir (1 = Verschiebung MSB→LSB, 0 = Verschiebung LSB→MSB Richtung)
z/ joinrxfifo (1 = RX-FIFO ist 8 tief, TX-FIFO = 0)
aa/ jointxfifo (1 = TX-FIFO ist 8 tief, RX-FIFO = 0)
bb/ joinrxfifoget (1 = RX-FIFO ändert sich zu 4 lesbaren Registern (nur 2350) cc/ joinrxfifoput (1 = RX-FIFO ändert sich zu 4 beschreibbaren Registern (nur 2350)

STARTEN UND STOPPEN EINER ZUSTANDSMASCHINE

Sobald die PIO konfiguriert ist, können Sie die Zustandsmaschine mit folgenden Befehlen starten und stoppen:

PIO START a,b PIO
STOP a,b
a/ die PIO-Nummer (0,1 oder 2 (nur 2350))
b/ die Zustandsmaschine (0...3)

Beachten Sie, dass eine Zustandsmaschine beim Anhalten genau an der Stelle anhält, an der sie sich gerade befindet. Um die Zustandsmaschine neu zu starten, ist es ratsam, zuerst PIO INIT MACHINE auszuführen.

BEISPIELPROGRAMM 1

Eine vollständige PIO-Implementierung, die einen GPIO-Pin umschaltet, kann in MMBasic wie unten gezeigt implementiert werden. Schließen Sie einen Summer an GP0 an und hören Sie den vom PIO erzeugten Audioton.

```
'ARM von GP0 trennen
setpin gp0,pio1                         'GP0 als Ausgangspin für PIO 1 verwenden

'PIO-Programm verwendet
'0 E081     'Pin-Ausgang
einstellen '1 E001     'Pin
hoch setzen '2 E000    'Pin
niedrig setzen '3 0001 'jmp 1

'PIO 1 programmieren, indem ein Array verwendet wird, um das Programm in den PIO-Speicher
zu schreiben, und starten Dim a%(7)=(&h0001E000E001E081,0,0,0,0,0,0,0)
PIO-Programm 1,a%()

'PIO 1-Zustandsmaschine 0 konfigurieren
p=Pio(pinctrl 0,1,,,,gp0,)        'SET verwendet 1 Pin, und zwar GP0 f=3100    '3051
Hz ist die niedrigste Frequenz CPUSPEED 200000 PIO-Initialisierungsmaschine 1,0,f,p
                                 'Standardwerte für execctrl, shiftctrl, start
verwenden
                                       'Adresse(=0)

'PIO 1-Zustandsmaschine starten 0 PIO
starten 1,0
```

Beachten Sie, dass das MMBasic-Programm beendet wird, der Summer jedoch weiterhin ertönt. PIO ist unabhängig von der ARM-CPU und läuft weiter, bis es gestoppt wird. Durch Aufrufen des MMBasic-Editors wird PIO gestoppt.

FIFOs

MMBasic und PIO tauschen Informationen über FIFOs aus. Die PIOs schieben Daten in den RX-FIFO (MMBasic ist der Empfänger) oder ziehen Daten aus dem TX-FIFO (MMBasic ist der Sender).

Wenn PIO Daten aus dem FIFO abruft, werden die Daten an das OSR (Output Shift Register) übertragen, von wo aus sie verarbeitet werden können. PIO kann die Daten aus dem ISR (Input Shift Register) in den FIFO schieben. Zusätzlich

Der PIO verfügt über zwei Register X und Y, die zur Speicherung oder zum Zählen verwendet werden können. Der PIO kann weder addieren noch subtrahieren oder vergleichen.

Datenfluss:

MMBasic -> FIFO -> OSR -> PIO (oder Pins) PIO
(oder Pins) -> ISR -> FIFO -> MMBasic

MMBasic kann Daten in den TX-FIFO schreiben und Daten aus dem RX-FIFO lesen, indem es Folgendes verwendet:

PIO READ a,b,c,d PIO
WRITE a,b,c,d

a/ PIO-Nummer	(0,1 oder 2 (nur 2350))
b/ Zustandsmaschinen-Nummer	(0...3)
c/ Anzahl der 32-Bit-Wörter	(1...4)
d/ Name der ganzzahligen Variablen	(d. h. Variable% oder Array%())

PIO CLEAR löscht alle PIO-FIFOs, ebenso wie PIO START und PIO INIT MACHINE.

Das MMBAsic-Programm muss nicht auf das Erscheinen von Daten im FIFO warten, da dem RX-FIFO ein Interrupt zugewiesen werden kann. Die MMBasic-Interrupt-Routine kann die Daten aus dem FIFO abrufen.

Ähnlich verhält es sich mit dem TX-Interrupt, bei dem MMBasic einen Interrupt erhält, wenn Daten für den TX-FIFO benötigt werden.

PIO INTERRUPT a,b,c,d

a/ PIO	(0,1 oder 2 (nur 2350))
b/ Zustandsmaschine	(0...3)
c/ Name des RX-Interrupt-Handlers	(z. B. „myRX_Interrupt" oder 0 zum Deaktivieren)
d/ Name des TX-Interrupt-Handlers	(z. B. „myTX_Interrupt" oder 0 zum Deaktivieren)

BEISPIELPROGRAMM 2

Das folgende Programm erläutert viele der oben vorgestellten MMbasic-Funktionen und -Befehle. Das Programm liest einen NES-Controller (SPI), der an den Raspberry Pi Pico angeschlossen ist. Der NES-Controller besteht aus einem HEF4021-Schieberegister, das mit 8 Druckschaltern verbunden ist.

Das Programm verwendet: **wrap** und **wrap target**, IN, side set und delay, PUSH, PIO READ. GP0 und GP1 befinden sich in SET für die Pin-Richtung und in side set für kompakten Code.

Die Verdrahtung entspricht der Definition im Code:

```
'ARM von GP0/1/2 trennen
  setpin gp0,pio1                      'Takt ausgeben
  setpin gp1,pio1                      'Auslesen
  setpin gp2,pio1                      'Daten in

'PIO-Programm

PIO assemble 1,".program NES"          'Ein Programm braucht einen Namen
.Seite setzen 2                        '2 Bits für Seiteneinstellung verwenden, 3 für Verzögerung
.Zeile 0                               'Startcode bei Zeile 0
   SET pindirs,&b11                    'GP0,GP1-Ausgang einstellen, Seite GP0,GP1 niedrig
.wrap Ziel                             'Ziel umbrechen = Anfang der Schleife
   IN null,32 Seite 2                  'set ISR auf 0, GP1 hoch (laden), GP0 niedrig
   SET X,7 Seite 0                     'X-Zähler auf 7 setzen, GP0,GP1 niedrig
.label Schleife:                       'innere Schleifenbezeichnung
   IN Pins,1 Seite 0                   '1 Datenbit verschieben, GP0,GP1 niedrig halten
   JMP X-- Schleife Seite 1            'jmp zu Schleife, dec. X, GP0 'hoch(Takt) PUSH
   Seite 0 [7]                         'jetzt X=0, Ergebnis in FIFO schieben,
   Verzögerung 7
.wrap"                                 'Ende äußere Schleife, wiederholen
.end program list                      'Ende des Programms, Ergebnis auflisten

'pio1 konfigurieren
  p=Pio(pinctrl 2,2,,gp2,gp0,gp0,)'GP0,GP1 aus (SET und SIDESET), GP2 IN f=1e5 '100
  kHz
  s=PIO(shiftctrl 0,0,0,0,0,0)          'Verschiebung von LSB für IN (und OUT)
  e=PIO(execctrl gp0,PIO(.wrap target),PIO(.wrap)) 'Wrap und Wrap-Ziel

'Konfiguration schreiben
  PIO init machine 1,0,f,p,e,s,0 'Start bei Zeile 0
```

```
'PIO1-Code starten PIO
  start 1,0


'Lesedaten in MMBasic überprüfen und ausgeben dim d%
  do
    pio lesen 1,0,1,d%
    bin$(d%) drucken
    Pause 200
  Schle
  ife
  END
```

DMA zu und von den FIFOs „ ”

DMA funktioniert wie folgt:

Beim Lesen aus dem FIFO wartet der DMA-Controller darauf, dass Daten im FIFO vorhanden sind, und überträgt diese Daten dann in den Prozessorspeicher. Bei jedem Lesevorgang erhöht er den Zeiger im Prozessorspeicher, sodass er beispielsweise ein Array schrittweise füllen kann, sobald jedes einzelne Datenelement verfügbar ist.

Beim Schreiben in den FIFO schreibt der DMA-Controller Daten aus dem Prozessorspeicher automatisch in den FIFO und wartet, wenn der FIFO voll ist. So können Daten in einem Array vorbereitet werden, und der DMA-Controller überträgt diese Daten so schnell, wie es das PIO-Programm erfordert, an den PIO-FIFO.

DMA kann ein 32-Bit-Wort, ein 16-Bit-Short oder ein 8-Bit-Byte übertragen. Bei der Einrichtung von DMA müssen Sie die Größe der Übertragung und die Anzahl der durchzuführenden Übertragungen angeben. Da jede Übertragung den Speicherzeiger um 1, 2 oder 4 Byte erhöht, muss MMBasic mit den im Speicher gepackten Daten arbeiten und nicht mit den 64 Bit, die für MMbasic-Ganzzahlen und -Gleitkommazahlen verwendet werden. Glücklicherweise implementiert MMBasic zwei Befehle, MEMORY PACK und MEMORY UNPACK, um dies sehr effizient zu tun, aber es könnte genauso gut mit Standard-BASIC-Arithmetik durchgeführt werden.

Der DMA kann so konfiguriert werden, dass er Daten wiederholt in einen Speicherbereich (einen Ringpuffer) ein- oder aus diesem ausliest. Die Befehle lauten:

PIO DMA RX a, b, c, d, e, f, g PIO
DMA TX a, b, c, d, e, f, g

Dabei gilt: a = pio	(0,1 oder 2 (nur 2350))
b = Zustandsmaschine	(0...3)
c = nbr	(Anzahl der zu übertragenden Wörter)
d = data%()	(Name des Integer-Arrays)
e = completioninterrupt	(wohin nach Abschluss, optional)
f = Übertragungsgröße	(8 = 16 = 32, optional)
g = Loopback-Zählung	(verwendet data%() als Ringpuffer, optional, loopbackcount = 2^n)

Der DMA startet die Zustandsmaschine automatisch, sodass kein PIO-START-Befehl erforderlich ist. Bevor Sie jedoch mit der Übertragung beginnen, stellen Sie sicher, dass ein neues PIO INIT MACHINE ausgeführt wird, damit die Zustandsmaschine an der erforderlichen Startadresse beginnt.

Wenn ein Ringpuffer verwendet wird, sind besondere Vorbereitungen erforderlich:

PIO MAKE RING BUFFER a, b
Wobei: a = Name des Integer-Puffers
b = Größe des Arrays in Byte

Beispiel:

```
DIM packed%                                  'Achtung: KEINE Klammern()
PIO MAKE RING BUFFER packed%,4096
```

packed% ist dann ein Integer-Array, das 4096/8 = 512 Integer-Werte enthält.

Dies kann dann vom DMA für einen Loopback-Zähler mit DMA von 1024 32-Bit-Wörtern, 2048 16-Bit-Shorts oder 4096 8-Bit-Bytes verwendet werden.

BEISPIELPROGRAMM 3

Dieses Programm fasst alles zusammen und verwendet DMA, um 128 Samples aus dem PIO RX FIFO zu lesen. Für die Demonstration sind GP0 bis GP5 Ausgänge von 3 PWMS und werden gleichzeitig vom PIO als 6-Kanal-Logikanalysator oder Oszilloskop abgetastet. Die 128 Samples werden als Wellenformen an den seriellen Port gesendet.

Dieses Logikanalysatorprogramm demonstriert auch PIO DMA RX, MEMORY UNPACK und die Verwendung von Puffern. Es verwendet PWMs, um zu Demonstrationszwecken ein Testsignal auf gp0..gp5 zu erzeugen. Dieselben Pins werden vom Logikanalysator gelesen und an die Konsole ausgegeben.

Um diesen Logikanalysator zu verwenden, kommentieren Sie zunächst die ersten 14 Zeilen aus.

```
'Erzeugen Sie ein 50-Hz-Dreiphasen-Testsignal, um den DMA an 6 GPIO-Pins zu
demonstrieren. SetPin gp0,pwm 'CH 0a
SetPin   gp1,pwm   'CH   0b
SetPin   gp2,pwm   'CH   1a
SetPin   gp3,pwm   'CH   1b
SetPin   gp4,pwm   'CH   2a
SetPin gp5,pwm 'CH 2b

Fpwm = 50: PW = 100 / 3
PWM 0, Fpwm, PW, PW - 100, 1, 1
PWM 1, Fpwm, PW, PW – 100, 1, 1
PWM 2, Fpwm, PW, PW - 100, 1, 1 PWM-
Synchronisation 0, 100/3, 200/3

'---------------------------------------------------------------------- LA-Code PIO    ----------------------------------------------------------------
'PIO-Code zum Abtasten von GP0..GP6 als elementarer Logikanalysator
PIO löschen 1

'In diesem Programm liest der PIO GP0..GP5 mit Brute-Force
'und schiebt Daten in den FIFO. Die Taktrate bestimmt die
'Abtastrate. Es gibt 2 Befehle pro Zyklus, 'die 10000/2 / 50 = 100
Abtastungen pro 50-Hz-Zyklus durchführen.

PIO assemble 1,".program push"
    .Zeile 0
    .wrap Ziel
    IN Pins,6                    „6 Bits von GP0..GP5 abrufen
    PUSH block                   'Daten schieben, wenn FIFO Platz hat
    .wrap
.Ende Programmliste

'Konfiguration
f=1e4                                          „PIO läuft mit 10 kHz
p=Pio(pinctrl 0,0,0,gp0,,,)                    'IN-Basis = GP0
e=Pio(execctrl gp0,PIO(.wrap target),PIO(.wrap)) 'Wrap 1 bis 0, gp0 ist
                                                          'Standard s=Pio(shiftctrl
0,0,0,0,0,0)                              'Shift-in durch LSB, Out wird nicht verwendet

'Konfiguration schreiben, Geschwindigkeit 10 kHz (Daten in FIFO 10 us nach Flanke GP0)
PIO init Maschine 1,0,f,p,e,s,0                'Startadresse = 0

'---------------------------------------------------------- LA-Code MMBasic    ----------------------------------------------------------------
'Speicherpuffer definieren
Dim a$(1)=("_","-")                            'Zeichen für den Ausdruck
length%=64                                     'Größe des gepackten Arrays
Dim data%(2*length%-1)                         'Array zum Speichern der 32-Bit-
                                               Samples im FIFO-Format
Dim packed%(length%-1)                         'DMA-Array zum Packen von 32-Bit-Samples in 64-
                                               Bit-Ganzzahlen

'DMA-Maschine laufen lassen und nach Belieben
wiederholen Do
  PIO DMA RX 1,0,2*Länge%,gepackt%(),ReadyInt print
  "Drücken Sie eine beliebige Taste, um die Abtastung
  neu zu starten" do:loop while inkey$=""
Schleife
beenden

'-----------------------------------------------------------------------------SUBS MMBasic    ----------------------------------------------------------------
Sub ReadyInt
'PIO stoppen und für nächsten Durchlauf neu initialisieren
```

```
PIO stop 1,0
PIO init machine 1,0,f,p,e,s,0                'Startadresse = 0

'Daten aus dem gepackten DMA-Puffer holen und in das ursprüngliche 32-Bit-Format entpacken
Speicher entpacken gepackt%(),Daten%(),2*Länge%,32

Serielle Ausgabe wie bei Logikanalysator-Traces Für
j=0 bis 5
  Maske%=2^j
  Für i=0 bis 2*Länge%-1
    Wenn i<106 Dann Drucken a$(((Daten%(i) Und Maske%)=Maske%));
  Nächstes i
  Drucken: Drucken
Nächstes j
Ende Sub
```

BEISPIELPROGRAMM 4

Dieses Programm läuft nur auf RP2350 und demonstriert die Verwendung der FIFOs als einzelne Register. Der PIO des RP2350 verfügt über spezielle Befehle zur Unterstützung von MOV RXFIFO[y],ISR und MOV OSR,RXFIFO[y].

MMBasic kann die 4 einzelnen FIFO-Register mit folgenden Befehlen lesen/schreiben:

PIO WRITEFIFO a, b, c, d PIO
READFIFO(a, b, c)

a/ pio	(0,1 oder 2 (nur 2350))
b/ Zustandsmaschine	(0...3)
c/ nbr	(FIFO-Register 0…3)
d/ Daten%	(32-Bit-Ganzzahlwert)

Das Programm konfiguriert den FIFO für einzelne Lesevorgänge und schreibt dann einige Werte in diese Register. Es startet den PIO, der 2 der 4 einzelnen FIFO-Register aktualisiert. Dann liest MMBasic die Werte, um anzuzeigen, dass sich nur 2 Register geändert haben und keine Daten verschoben wurden (wie es beim RP2040 FIFO der Fall wäre).

```
Nur für RP2350-Assembler. Dies funktioniert nicht auf RP2040. pio

clear 1

pio assemble 1,".program test"
.Zeile 0
set y,4                     'nur ein Wert 4 in Y
mov isr,y                   'Y nach isr kopieren
jmp y--,next                'y=y-1 immer zum nächsten
.label next                 'Bezeichnung
mov rxfifo[y],isr           'sollte Programm 4 in FIFO [3]
programmieren mov isr,y     'Y nach isr kopieren
mov rxfifo[2],isr           'sollte 3 in FIFO [2]
programmieren jmp 0         'wiederholen
.end program

f=1e6 '1 MHz

'PIO(EXECCTRL a,b,c)
e=pio(execctrl gp0,0,31)                  'Standardwert, wird nicht tatsächlich geändert

'PIO(SHIFTCTRL a,b,c,d,e,f,g,h,i,j)
sr=pio(shiftctrl 0,0,0,0,0,0,0,0,0,1) 'einzelnes RX lesen, RX=4 tief sw=pio(shiftctrl
0,0,0,0,0,0,0,0,1,0) 'einzelnes RX schreiben, RX=4 tief

'PIO(PINCTRL a,b,c,d,e,f,g)
p=pio(pinctrl 0,0,0,gp0,gp0,gp0,gp0) 'Standardwert, wird nicht tatsächlich geändert

'Verwendung von FIFO als einzelne Register testen 'FIFO
mit vorgegebenen Werten füllen
pio init machine 1,0,f,p,e,sw,0         'Maschine zum Schreiben in RX-FIFO
initialisieren für i=0 bis &h3          'die 4 RXFIFO-Register schreiben
```

```
 pio writefifo 1,0,i,&h100*(i+1) 'Werte &h100, &h200, &h300, &h400 schreiben next

'Überprüfen, ob die Werte korrekt geschrieben wurden
print „3 RXFIFO-Register vor dem Ausführen des Programms”
pio init machine 1,0,f,p,e,sr,0         'Maschine zum Lesen des RX-FIFO für i=0 bis
3 initialisieren                        'die 4 RXFIFO-Register lesen
   print i,hex$(pio(readfifo 1,0,i)) 'überprüfen, ob sie korrekt geschrieben wurden
next
ausgeben

'PIO-Programm ausführen. Das sollte (kontinuierlich) in die Register 2 und 3 des FIFO schreiben,
aber die Register 0 und 1 nicht verändern
pio start 1,0

'aktualisierte FIFO-Register anzeigen
print "3 RXFIFO-Register nach Ausführung des Programms"
für i=0 bis 3        'die 4 FIFO-Register lesen, um zu überprüfen, ob das Programm
   funktioniert print i,hex$(pio(readfifo 1,0,i))
next
```

Die erwartete Ausgabe lautet:

```
3 RXFIFO-Register vor dem Ausführen des Programms
0        100
1        200
2        300
3        400

3 RXFIFO-Register nach Ausführung des Programms
0       100
1       200
2       3
3       4
```

BEISPIELPROGRAMM 5

Abschließend ein Beispielprogramm, das zeigt, wie mehrere Zustandsmaschinen zusammenarbeiten können, wobei jede durch einen Interrupt der anderen ausgelöst wird. Beachten Sie auch die Verwendung von .LINE NEXT als Startpunkt des zweiten Programms und die Verwendung von x=PIO(NEXT LINE) zur Bestimmung des Startpunkts des zweiten Programms.

IRQ 1 setzt und IRQ. Wenn die andere Zustandsmaschine auf WAIT 1 IRQ 1 wartet, fährt sie fort, nachdem IRQ 1 gesetzt wurde, und löscht gleichzeitig IRQ 1.

```
'Test für PIO-IRQs auf den PIO-0-Zustandsmaschinen 0 und 1

  'verwendet GP0 und GP1
  setpin    gp0,pio0
  setpin    gp1,pio0
  pio clear 0

  'Zielfrequenz für PIO f=1e5
  '100 kHz

  pio assemble 0
   .Programm sm0
  .Zeile 0
    Pindirs,1 setzen                    'GP0 ausgeben, siehe pinctrl
  .Ziel umschließen
    Pins setzen, 1 [31]                 „GP0 auf High setzen, 31 Zyklen warten
    Pins setzen, 0 [31]                 GP0 auf niedrig setzen, 31 Zyklen warten
    irq 0                               „IRQ 0 setzen
    1 IRQ 1 warten                      'Warten, bis sm1 IRQ 1 setzt
  .wrap
.end programmliste
```

```
  ln=pio(nächste Zeile)                    „Zeile im Programm merken
  p0=pio(pinctrl 0,1,,,,gp0)               „GP0 ist Pin für SET e0=pio(execctrl
  gp0,pio(.wrap target),pio(.wrap))

   pio assemble 0
    .Programm sm1
  .Zeile nächste
     pindirs,1 setzen                      „GP1-Ausgang festlegen
  .wrap Ziel
     wait 1 irq 0                          'auf IRQ 0 warten
     Pins setzen, 1 [31]                   „GP1 hoch setzen und 31 Zyklen warten
     Pins,0 [31] setzen                    „GP1 auf niedrig setzen und 31 Zyklen warten
     irq 1                                 „IRQ 1 setzen
  .wrap
.end programmliste

  p1=pio(pinctrl 0,1,,,,gp1)
  e1=pio(execctrl gp0,pio(.wrap target),pio(.wrap))

  pio init machine 0,1,f,p1,e1,,ln 'sm1 an Zeile ln starten pio
  init machine 0,0,f,p0,e0,,0              'sm0 in Zeile 0 starten

  pio start 0,1
  pio start 0,0

do:loop 'nichts tun, PIO die Signale generieren lassen

* Ende *
```

Anhang G
Sprites

VGA-, HDMI- und LCD-FRAMEBUFFER

Sie können ein Sprite auf verschiedene Arten erstellen, aber im Wesentlichen speichern Sie nur ein Bild in einem Puffer. Der Unterschied besteht darin, wie Sie das Sprite ANZEIGEN. In diesem Fall speichert die Firmware beim ersten Aufruf den Speicherbereich (oder den Bildschirmbereich), der durch das Sprite ersetzt wird, und zeichnet dann das Sprite an dieser Stelle.

Nachfolgende SHOW-Befehle ersetzen das Sprite durch den gespeicherten Hintergrund, speichern den Hintergrund für die neue Position und zeichnen schließlich das Sprite. Auf diese Weise können Sie das Sprite ohne zusätzlichen Code über den Hintergrund bewegen.

Die Kollisionserkennung sitzt dann darüber und sucht nach den rechteckigen Begrenzungen von Sprites, die sich berühren, um einen Interrupt zu erzeugen, oder nach Sprites, die den Rand des Rahmens berühren.

Sprites werden so angeordnet, dass die Zeichenreihenfolge in einem LIFO-Verfahren beibehalten wird. Angenommen, Sie haben Sprite 1, das von Sprite 2 und dann von Sprite 3 überlagert wird. Wenn Sie Sprite 1 einfach verschieben würden, würde sein Hintergrund Teile von 2 und 3 überschreiben – was nicht in unserem Sinne ist. SPRITE SHOW SAFE löst den LIFO auf, indem es jedes Sprite in umgekehrter Reihenfolge entfernt, Sprite 1 verschiebt und dann zuerst 2 und dann 3 darüber wiederherstellt. Schließlich gibt es noch das Konzept der Ebenen (dies ist der vierte Parameter in SPRITE SHOW).

Das Konzept der Sprite-Implementierung ist wie folgt:

- Sprites sind vollfarbig und können jede beliebige Größe haben. Die Kollisionsgrenze ist das umschließende Rechteck.
- Sprites werden bis zu einer bestimmten Anzahl (1 bis 64) geladen.
- Sprites werden mit dem Befehl SPRITE SHOW angezeigt.
- Für jeden SHOW-Befehl muss der Benutzer eine „Ebene” auswählen. Diese kann zwischen 0 und 10 liegen.
- Sprites kollidieren mit Sprites auf derselben Ebene, Ebene 0 oder dem Bildschirmrand.
- Layer 0 ist ein Sonderfall, Sprites auf allen anderen Layern kollidieren mit ihm.
- Die SCROLL-Befehle lassen Sprites auf allen Ebenen außer Ebene 0 unbewegt.
- Sprites der Ebene 0 scrollen mit dem Hintergrund, was zu Kollisionen führen kann.
- Es gibt keine praktische Begrenzung für die Anzahl der durch SHOW- oder SCROLL-Befehle verursachten Kollisionen.
- Mit der Funktion SPRITE() kann der Benutzer die Details einer Kollision vollständig abfragen.
- Ein SHOW-Befehl überschreibt die Details aller vorherigen Kollisionen für dieses Sprite.
- Ein SCROLL-Befehl überschreibt die Details früherer Kollisionen für ALLE Sprites.
- Um einen Bildschirm in einen früheren Zustand zurückzusetzen, sollten Sprites in umgekehrter Reihenfolge zu ihrer Schreibweise entfernt werden (d. h. Last In First Out).

Da das Verschieben eines Sprites oder insbesondere das Scrollen des Hintergrunds zu mehreren Sprite-Kollisionen führen kann, ist es wichtig zu verstehen, wie diese abgefragt werden können.

Der beste Weg, um mit einer Sprite-Kollision umzugehen, ist die Verwendung der Interrupt-Funktion. Eine Kollisions-Interrupt-Routine wird mit dem Befehl SPRITE INTERRUPT eingerichtet. Beispiel:

```
SPRITE INTERRUPT collision
```

Das folgende Beispielprogramm dient zur Identifizierung aller Kollisionen, die entweder durch einen SPRITE SHOW-Befehl oder einen SCROLL-Befehl verursacht wurden

```
'
' Diese Routine demonstriert eine vollständige Abfrage von Kollisionen '
SUB collision LOCAL
  INTEGER i
' Verwenden Sie zunächst die Funktion SPRITE(S), um festzustellen, was die
  Unterbrechung verursacht hat IF SPRITE(S) <> 0 THEN 'Kollision eines
  bestimmten einzelnen Sprites
    'SPRITE(S) gibt das Sprite zurück, dessen Bewegung die Kollision
    verursacht hat PRINT "Kollision bei Sprite ", SPRITE(S)
    process_collision(SPRITE(S))
```

```
    PRINT
  SONST      '0 bedeutet Kollision eines oder mehrerer Sprites, verursacht durch
             Hintergrundbewegung ' SPRITE(C, 0) gibt an, wie viele Sprites eine Kollision
             hatten
    PRINT "Durch das Scrollen kam es zu insgesamt ", SPRITE(C,0)," Sprite-Kollisionen" FOR I =
    1 TO SPRITE(C, 0)
      ' SPRITE(C, 0, i) gibt uns die Sprite-Nummer des „I"-ten Sprites an PRINT "Sprite ",
      SPRITE(C, 0, i)
      process_collision(SPRITE(C, 0, i)) NEXT i
    PRINT
  ENDIF
END SUB

' Details zu den spezifischen Kollisionen für ein bestimmtes Sprite
abrufen SUB process_collision(S AS INTEGER)
  LOCAL INTEGER i, j
  ' SPRITE(C, #n) gibt die Anzahl der aktuellen Kollisionen für Sprite n zurück PRINT
  "Insgesamt " SPRITE(C, S) " Kollisionen"
   FOR I = 1 TO SPRITE(C, S)
    ' SPRITE(C, S, i) gibt die Sprite-Nummer des „I"-ten Sprites an j = SPRITE(C, S,
    i)
    IF j = &HF1 THEN
      PRINT "Kollision mit der linken
    Bildschirmseite" ELSE IF j = &HF2 THEN
      PRINT "Kollision mit dem oberen
    Bildschirmrand" ELSE IF j = &HF4 THEN
      PRINT „Kollision mit der rechten Seite des
    Bildschirms" ELSE IF j = &HF8 THEN
      PRINT "Kollision mit dem unteren
    Bildschirmrand" ELSE
      ' SPRITE(C, #n, #m) gibt Details zur m-ten Kollision zurück PRINT
      "Kollision mit Sprite ", SPRITE(C, S, i)
    ENDIF
  NEXT i
END SUB
```

Anhang H
Turtle-Grafik

Diese Version enthält eine sehr umfassende Turtle-Grafik-Implementierung für alle Versionen außer WebMite RP2040 und WebMite RP2350. Die unterstützten Befehle mit dem Präfix „TURTLE “ sind:

Bewegungsbefehle

FORWARD distance	(FD) – Um die angegebene Pixelanzahl vorwärts
bewegen BACK distance	(BK) – Um distance Pixel rückwärts bewegen
LEFT [Winkel]	(LT) – Um Winkelgrad nach links drehen, 90 Grad, wenn nicht angegeben
RECHTS [Winkel]	(RT) – Nach rechts um Winkelgrad drehen, 90 Grad, wenn nicht
angegeben	

Positionsbefehle

SET XY x, y	- Zur absoluten Position (x,y) bewegen
SET X x	- X-Koordinate festlegen, Y beibehalten
SET Y y	- Y-Koordinate festlegen, X beibehalten
SET HEADING Winkel	(SETH) - Absolute Richtung einstellen (0=oben, 90=rechts)
HOME	- Zurück zur Mitte (MM.HRES\2,MM.VRES\2) Kurs 0

Stiftsteuerungsbefehle

STIFT HOCH	(PU) – Stift anheben (Zeichnen beenden)
PEN DOWN	(PD) – Stift absenken (Zeichnen starten)
PEN COLOUR Farbe	(PC) – Stiftfarbe festlegen
STIFTBREITE Breite	(PW) – Stiftlinienbreite

festlegen Befehle für Bögen und Kurven

BOGEN Radius Winkel	– Zeichnen Sie einen Bogen mit dem angegebenen Radius und Winkel.
ARCLEFT Radius, Winkel	(ARCL) – Bogen nach links zeichnen
ARCRIGHT Radius,Winkel	(ARCR) – Rechtsdrehenden Bogen
zeichnen	
BEZIER cp1 , cp1winkel, cp2, cp2winkel, ende, endekwinkel	– Zeichnen einer Bezierkurve mit

Kontrollpunkten Grundlegende Formbefehle

CIRCLE Radius	– Kreis an aktueller Position zeichnen
DOT Größe	- Gefüllten Punkt zeichnen (Standardgröße = 5)
FCIRCLE Radius	- Gefüllten Kreis zeichnen
FRECTANGLE Breite, Höhe (FRECT)	- Gefülltes Rechteck zeichnen
WEDGE Radius Anfang Ende	- Gefüllten Keil/Tortenstück zeichnen

Füllbefehle

FILL COLOUR Farbe	(FC) – Füllfarbe festlegen und Füllung
aktivieren FILL PATTERN Muster	(FP) - Füllmuster festlegen (0-31)
KEINE FÜLLUNG	– Füllung deaktivieren
FÜLLEN	– Flächige Füllung an der aktuellen Position
FÜLLUNG BEGINNEN	(BF) - Aufzeichnung des Polygons für die Füllung starten
END FILL	(EF) – Aufnahme beenden und Polygon füllen

Cursor-Befehle

SCHILDKRÖTE ANZEIGEN	(ST) – Schildkrötencursor anzeigen
HIDE TURTLE	(HT) – Schildkrötencursor
ausblenden CURSORGRÖSSE Größe	(CS) – Cursorgröße
festlegen CURSOR COLOUR Farbe	CC) – Cursorfarbe festlegen
STAMP	- Zeichnen Sie eine Schildkröte an der aktuellen x,y-Position

Befehle zur Statusverwaltung

RESET [anzeigen]	- Bildschirm löschen und alles zurücksetzen, Schildkröte anzeigen, wenn
show = 1 PUSH	- Aktuelle Position und Richtung im Stapel speichern
POP	- Position und Richtung aus dem Stapel wiederherstellen

Der Code enthält die Möglichkeit, Kreise, Rechtecke und Polygone mit einer strukturierten Füllung zu füllen. Führen Sie ihn im Modus 2 auf einem RP2040-VGA-System oder im Modus 3 auf einem RP2350-VGA- oder HDMI-System aus.

Füllmuster

0: Vollflächige Füllung
1: Schachbrettmuster
2: Vertikale Linien
3: Horizontale Linien
4: Diagonales Kreuz
5: Diagonale Streifen
6: Kreuzschraffur
7: Feine Diagonale
8: Dichtes Schachbrettmuster
9: Diagonal rechts mittel 10: Diagonal links mittel 11: Vertikale Linien mittel 12: Horizontale Linien mittel 13: Großes Schachbrettmuster
14: Vertikal gepunktet
15: Horizontale Streifen eng 16: Gitter
17: Webmuster
18: Rautenmuster
19: Diagonaler Farbverlauf
20: Diagonaler Farbverlauf umgekehrt 21: Rand/Rahmen
22: Vertikale Teilung
23: Gewebt
24: Spärliche Punkte
25: Diagonal sehr fein 26: Pfeil nach oben
27: Dichte Punkte
28: Chevron
29: Hohlrautenmuster
30: Kreis
31: Gefüllter Kreis

Anhang I
Spezielle Tastaturtasten

MMBasic generiert ein einzelnes eindeutiges Zeichen für die Funktionstasten und andere Sondertasten auf der Tastatur. Hinweis: Die DE-USB-Tastatur gibt bei normaler Eingabe (input$/inkey$) die Codes 200-209 für Tastaturzeichen mit Akzenten zurück.

Die Codes werden in dieser Tabelle als Hexadezimal- und Dezimalzahlen angezeigt:

Tastaturtaste	Tastencode (Hex)	Tastencode (dezimal)
DEL	7F	127
Pfeil nach oben	80	128
Pfeil nach unten	81	129
Pfeil nach links	82	130
Pfeil nach rechts	83	131
Einfügen	84	132
Startseite	86	134
Ende	87	135
Seite nach oben	88	136
Seite nach unten	89	137
Alt	8B	139
F1/Umschalt F1	91/B1	145/177
F2/Umschalt F2	92/B2	146/178
F3/Umschalt F3 **	93/B3	147/179
F4/Umschalt F4 **	94/B4	148/180
F5/Umschalt F5 **	95/B5	149/181
F6/Umschalt F6 **	96/B6	150/182
F7/Umschalt F7 **	97/B7	151/183
F8/Umschalt F8 **	98/B8	152/184
F9/Umschalt F9	99/B9	153/185
F10/Umschalt F10	9A/BA	154/186
F11/Umschalt F11	9B/BB	155/187
F12/Umschalt F12	9C/BC	156/188
PrtScr/SysRq	9D	157
PAUSE/BREAK	9E	158
SHIFT_TAB	9F	159
SHIFT_DEL	A0	160
SHIFT_DOWN_ARROW	A1	161
SHIFT_RIGHT_ARROW	A3	163

** zeigt auch die Funktion für VT100-Emulatoren an

Bei angeschlossenen PS2- und USB-Tastaturen wird, wenn die Umschalttaste gleichzeitig mit den Funktionstasten F1 bis F12 gedrückt wird, 20 (hex) zum Code hinzugefügt (dies entspricht der Einstellung von Bit 5). Beispielsweise erzeugt Shift-F10 BA (hex).

Der Umschaltmodifikator funktioniert mit den Funktionstasten F1 bis F12; er wird für die anderen Tasten außer TAB, DEL, DOWN_ARROW und RIGHT_ARROW, wie oben angegeben, ignoriert. MMBasic übersetzt die meisten VT100-Escape-Codes, die von Terminalemulatoren wie TeraTerm und Putty generiert werden, in diese Codes (der Umschaltmodifikator funktioniert nur für F3-F8). Das bedeutet, dass ein Terminalemulator, der über einen USB- oder seriellen Anschluss als Konsole betrieben wird, die gleichen Tastencodes generiert wie eine direkt angeschlossene Tastatur.

Anhang J
Programmieren in BASIC – Ein Tutorial

Die Sprache BASIC wurde 1964 vom Dartmouth College in den USA als Computersprache für den Programmierunterricht eingeführt und ist dementsprechend einfach zu verwenden und zu erlernen. Gleichzeitig hat sie sich als kompetente und leistungsfähige Programmiersprache bewährt und war daher in den späten 70er und frühen 80er Jahren sehr beliebt. Auch heute noch sind einige große kommerzielle Datensysteme in der Sprache BASIC (hauptsächlich Pick Basic) geschrieben.

Der in der PicoMite-Firmware verwendete BASIC-Interpreter heißt MMBasic und ist eine moderne Version der Sprache BASIC, die in etwa den vor Jahren beliebten Microsoft BASIC-Interpreter emuliert.

Für Programmierer ist der größte Vorteil von BASIC seine Benutzerfreundlichkeit. Einige modernere Sprachen wie C und C++ können wirklich verwirrend sein, aber mit BASIC können Sie mit einem einzeiligen Programm beginnen und etwas Sinnvolles daraus machen. MMBasic ist auch insofern leistungsstark, als Sie damit anspruchsvolle Grafiken zeichnen, die externen E/A-Pins zur Steuerung anderer Geräte manipulieren und mit einer Reihe von integrierten Kommunikationsprotokollen mit anderen Geräten kommunizieren können.

Befehls e Eingabeaufforderung

Die Interaktion mit MMBasic erfolgt über die Konsole an der Befehlszeile (d. h. das Größer-als-Zeichen (>) auf der Konsole). Beim Start gibt MMBasic die Befehlszeile aus und wartet auf die Eingabe eines Befehls. Es kehrt auch zur Befehlszeile zurück, wenn Ihr Programm beendet wird oder wenn es eine Fehlermeldung generiert hat.

Wenn die Eingabeaufforderung angezeigt wird, stehen Ihnen eine Vielzahl von Befehlen zur Verfügung, die Sie eingeben und ausführen können. In der Regel werden damit die im Speicher befindlichen Programme aufgelistet (LIST) oder bearbeitet (EDIT) oder bestimmte Optionen festgelegt (Befehl OPTION). Meistens lautet der Befehl einfach RUN, wodurch MMBasic angewiesen wird, das im Programmspeicher befindliche Programm auszuführen.

Fast jeder Befehl kann an der Befehlszeile eingegeben werden, was häufig genutzt wird, um einen Befehl zu testen und zu sehen, wie er funktioniert. Ein einfaches Beispiel ist der Befehl PRINT (mehr dazu später), den Sie testen können, indem Sie Folgendes an der Befehlszeile eingeben:

```
PRINT 2 + 2
```

Es überrascht nicht, dass MMBasic die Zahl 4 ausgibt, bevor es zur Befehlszeile zurückkehrt.

Diese Möglichkeit, einen Befehl an der Eingabeaufforderung zu testen, ist nützlich, wenn Sie das Programmieren in BASIC lernen. Es lohnt sich daher, einen Raspberry Pi Pico mit der PicoMite-Firmware zur Hand zu haben, um während der Arbeit mit diesem Tutorial gelegentlich Tests durchzuführen.

Struktur eines BASIC- -Programms

Ein BASIC-Programm beginnt an der ersten Zeile und läuft bis zum Ende oder bis es auf einen END-Befehl trifft. An diesem Punkt zeigt MMBasic die Befehlszeile (>) auf der Konsole an und wartet auf eine Eingabe.

Ein Programm besteht aus einer Reihe von Anweisungen oder Befehlen, von denen jeder den BASIC-Interpreter dazu veranlasst, etwas auszuführen (die Begriffe *„Anweisung"* und *„Befehl"* haben im Allgemeinen dieselbe Bedeutung und werden in diesem Tutorial synonym verwendet).

Normalerweise steht jede Anweisung in einer eigenen Zeile, aber Sie können auch mehrere Anweisungen in einer Zeile haben, die durch das Doppelpunktzeichen (:) getrennt sind.

Beispiel:

```
A = 24.6 : PRINT A
```

Jede Zeile kann mit einer Zeilennummer beginnen. Zeilennummern waren in den frühen BASIC-Interpretern obligatorisch, moderne Implementierungen (wie MMBasic) benötigen sie jedoch nicht. Sie können sie weiterhin verwenden, wenn Sie möchten, aber sie haben keinen Nutzen und überladen in der Regel nur Ihre Programme.

Dies ist ein Beispiel für ein Programm, das Zeilennummern verwendet:

```
50 A = 24,6
60 PRINT A
```

Eine Zeile kann auch mit einem Label beginnen, das als Ziel für einen Programmsprung mit dem Befehl GOTO verwendet werden kann. Dies wird näher erläutert, wenn wir den Befehl GOTO behandeln, aber hier ist ein Beispiel (der Labelname lautet `JmpBack`):

```
JmpBack: A = A + 1 PRINT
A
GOTO JmpBack
```

Kommentare

Ein Kommentar ist jeder Text, der auf das einfache Anführungszeichen (') folgt. Ein Kommentar kann an beliebiger Stelle eingefügt werden und erstreckt sich bis zum Ende der Zeile. Wenn MMBasic auf einen Kommentar stößt, springt es einfach zum Ende des Kommentars (d. h. es führt keine Aktion in Bezug auf den Kommentar aus).

Kommentare sollten verwendet werden, um nicht offensichtliche Teile des Programms zu erklären und allgemein Personen, die mit dem Programm nicht vertraut sind, darüber zu informieren, wie es funktioniert und was es bewirkt. Denken Sie daran, dass Sie sich nach nur wenigen Monaten nicht mehr an ein von Ihnen geschriebenes Programm erinnern können und es Ihnen seltsam vorkommen wird, wenn Sie es wieder zur Hand nehmen. Aus diesem Grund werden Sie sich später dafür danken, wenn Sie viele Kommentare verwenden.

Im Folgenden finden Sie einige Beispiele für Kommentare:

```
' berechne die Hypotenuse PRINT
SQR(a * a + b * b)
```

oder

```
INPUT var       ' Temperatur abrufen
```

Ältere BASIC-Programme verwendeten den Befehl REM, um einen Kommentar zu beginnen. Sie können diesen Befehl ebenfalls verwenden, wenn Sie möchten, aber das einfache Anführungszeichen ist einfacher zu verwenden und praktischer.

Der Befehl PRINT

Es gibt eine Reihe von grundlegenden Befehlen, die wir in diesem Tutorial behandeln werden, aber der wohl nützlichste ist der Befehl PRINT. Seine Aufgabe ist einfach: etwas auf der Konsole auszugeben. Er wird hauptsächlich verwendet, um Daten auszugeben, die Sie sehen können (wie das Ergebnis von Berechnungen), oder um informative Meldungen anzuzeigen.

PRINT ist auch nützlich, wenn Sie einen Fehler in Ihrem Programm suchen. Sie können damit die Werte von Variablen ausgeben und Meldungen an wichtigen Stellen während der Ausführung des Programms anzeigen.

In seiner einfachsten Form gibt der Befehl einfach alles aus, was in seiner Befehlszeile steht. Zum Beispiel:

```
PRINT 54
```

wird auf der Konsole die Zahl 54 gefolgt von einer neuen Zeile angezeigt.

Die zu druckenden Daten können etwas Einfaches wie dies sein oder ein Ausdruck, der etwas zu berechnendes bedeutet. Wir werden später noch genauer auf Ausdrücke eingehen, aber als Beispiel sei Folgendes genannt:

```
> PRINT 3/21
 0.1428571429
>
```

würde das Ergebnis von drei geteilt durch einundzwanzig berechnen und anzeigen. Beachten Sie, dass das Größer-als-Zeichen (>) die von MMBasic erzeugte Eingabeaufforderung ist – Sie müssen es nicht eingeben.

Weitere Beispiele für den Befehl PRINT sind:

```
> PRINT „Wonderful World”
Wonderful World
> PRINT (999 + 1) / 5
 200
>
```

Sie können diese Beispiele in der Befehlszeile ausprobieren.

Der Befehl PRINT funktioniert auch mit mehreren Werten gleichzeitig, zum Beispiel:

```
> PRINT „Die erste Zahl ist“ 20+25 „und die zweite ist“ 18/3 Die erste
Zahl ist 45 und die zweite ist 6
>
```

Normalerweise werden die Werte wie im vorherigen Beispiel durch ein Leerzeichen voneinander getrennt, Sie können die Werte jedoch auch durch ein Komma (,) trennen. Durch das Komma wird zwischen den beiden Werten ein Tabulator eingefügt. In MMBasic sind die Tabulatoren im Befehl PRINT acht Zeichen voneinander entfernt.

Um die Tabulatorfunktion zu veranschaulichen, gibt der folgende Befehl eine tabulatorgetrennte Liste von Zahlen aus:

```
> PRINT 12, 34, 9.4, 1000
 12        34        9,4       1000
>
```

Beachten Sie, dass vor jeder Zahl ein Leerzeichen gedruckt ist. Dieses Leerzeichen ist ein Platzhalter für das Minuszeichen (-), falls der Wert negativ ist. Sie können den Unterschied anhand der Zahlen 12 und 9,4 in diesem Beispiel sehen:

```
> PRINT -12, 34, -9,4, 1000
-12        34       -9,4       1000
>
```

Die PRINT-Anweisung kann mit einem Semikolon (;) beendet werden. Dadurch wird verhindert, dass der PRINT-Befehl nach dem Drucken des gesamten Textes in eine neue Zeile springt. Beispiel:

```
PRINT "Dies wird";
PRINT „ in einer einzigen Zeile gedruckt.“
```

Dies führt zu folgender Ausgabe:

```
Dies wird in einer einzigen Zeile gedruckt.
```

Ohne das Semikolon am Ende der ersten Zeile würde die Meldung wie folgt aussehen:

```
Dies wird
 in einer einzigen Zeile gedruckt.
```

Variablen

Bevor wir weitermachen, müssen wir definieren, was eine „Variable” ist, da sie für die Funktionsweise der Sprache BASIC (und eigentlich der meisten Programmiersprachen) von grundlegender Bedeutung ist. Eine Variable ist einfach ein Ort, an dem ein Datenelement (d. h. sein „Wert”) gespeichert wird. Dieser Wert kann während der Ausführung des Programms geändert werden, weshalb er als „Variable” bezeichnet wird.

Variablen in MMBasic können drei verschiedene Typen haben. Der häufigste Typ ist der Gleitkommawert, der automatisch angenommen wird, wenn der Typ der Variablen nicht angegeben ist. Die beiden anderen Typen sind Ganzzahl und Zeichenfolge, auf die wir später noch eingehen werden. Eine Gleitkommazahl ist eine gewöhnliche Zahl, die einen Dezimalpunkt enthalten kann. Beispielsweise sind 3,45, -0,023 oder 100,00 allesamt Gleitkommazahlen.

Eine Variable kann zum Speichern einer Zahl verwendet werden und dann auf dieselbe Weise wie die Zahl selbst verwendet werden. In diesem Fall repräsentiert sie den Wert der letzten Zahl, die ihr zugewiesen wurde.

Ein einfaches Beispiel:

```
A = 3
B = 4
PRINT A + B
```

gibt die Zahl 7 aus. In diesem Fall sind sowohl A als auch B Variablen, und MMBasic verwendet ihre aktuellen Werte in der PRINT-Anweisung. MMBasic erstellt automatisch eine Variable, wenn es zum ersten Mal auf sie trifft. Die Anweisung `A = 3` hat also eine Gleitkommavariable (der Standardtyp) mit dem Namen `A` erstellt und ihr dann den Wert 3 zugewiesen.

Der Name einer Variablen muss mit einem Buchstaben beginnen, während der Rest des Namens aus Buchstaben, Zahlen, Unterstrichen oder Punkten bestehen kann. Der Name kann bis zu 31 Zeichen lang sein, wobei die Groß- und Kleinschreibung keine Rolle spielt. Hier einige Beispiele:

```
Gesamtanzahl
Vorfarben temp3
count
x
DiesIstEinSehrLangerVariablenname
increment.value
```

Sie können den Wert einer Variablen an jeder beliebigen Stelle in Ihrem Programm mithilfe des Zuweisungsbefehls ändern, z. B.:

```
Variable = Ausdruck
```

Beispiel:

```
temp3 = 24,6
count = 5
CTemp = (FTemp - 32) * 0,5556
```

Im letzten Beispiel sind sowohl CTemp als auch FTemp Variablen, und diese Zeile konvertiert den Wert von FTemp (in Grad Fahrenheit) in Grad Celsius und speichert das Ergebnis in der Variablen CTemp.

Ausdrücke

Wir sind in diesem Tutorial bereits auf den Begriff „Ausdruck" gestoßen, der in der Programmierung eine bestimmte Bedeutung hat. Es handelt sich um eine Formel, die vom BASIC-Interpreter in eine einzelne Zahl oder einen Wert aufgelöst werden kann.

MMBasic wertet numerische Ausdrücke nach denselben Regeln aus, die wir in der Schule gelernt haben. Beispielsweise werden Multiplikation und Division zuerst ausgeführt, gefolgt von Addition und Subtraktion. Diese Regeln werden als Vorrangregeln bezeichnet und sind zuvor in diesem Handbuch ausführlich beschrieben worden (siehe Kapitel *„Ausdrücke und Operatoren*").

Das bedeutet, dass MMBasic 2 + 3 * 6 berechnet, indem es zuerst 3 mit 6 multipliziert, was 18 ergibt, und dann 2 addiert , was zu einem Endwert von 20 führt. In ähnlicher Weise werden sowohl 5 * 4 als auch 10 + 4 * 3 – 2 ebenfalls zu 20 ausgewertet.

Wenn Sie den Interpreter zwingen möchten, Teile des Ausdrucks zuerst auszuwerten, können Sie diesen Teil des Ausdrucks in Klammern setzen. Beispielsweise ergibt (10 + 4) * (3 – 2) den Wert 14 und nicht 20, wie es ohne Klammern der Fall gewesen wäre. Die Verwendung von Klammern verlangsamt das Programm nicht nennenswert, daher sollten Sie sie großzügig einsetzen, wenn die Möglichkeit besteht, dass MMBasic Ihre Absicht falsch interpretiert.

Wie bereits erwähnt, können Sie Variablen in einem Ausdruck genauso wie reine Zahlen verwenden. Beispielsweise erhöht dies den Wert der Variablen `temp` um eins:

```
temp = temp + 1
```

Sie können in Ausdrücken auch Funktionen verwenden. Dabei handelt es sich um spezielle Operationen, die von MMBasic bereitgestellt werden, beispielsweise zur Berechnung trigonometrischer Werte.

Ein Beispiel für die Verwendung einer Funktion ist das folgende, das die Länge der Hypotenuse eines rechtwinkligen Dreiecks ausgibt. Dabei wird die Funktion SQR() verwendet, die die Quadratwurzel einer Zahl zurückgibt (`a` und `b` sind Variablen, die die Längen der anderen Seiten enthalten):

```
PRINT SQR(a * a + b * b)
```

MMBasic wertet diesen Ausdruck zunächst aus, indem es `a` mit `a` multipliziert, dann `b` mit `b` multipliziert und anschließend die Ergebnisse addiert. Die resultierende Zahl wird dann an die Funktion `SQR()` übergeben, die die Quadratwurzel dieser Zahl (d. h. die Hypotenuse) berechnet und sie zur Anzeige durch den Befehl PRINT zurückgibt.

Einige weitere mathematische Funktionen, die MMBasic bereitstellt, sind: SIN(r)

– der Sinus von r

COS(r) – der Kosinus von r

TAN(r) – der Tangens von r

Es stehen Ihnen noch viele weitere Funktionen zur Verfügung, die alle weiter oben in diesem Handbuch aufgeführt sind.

Beachten Sie, dass bei den oben genannten trigonometrischen Funktionen der an die Funktion übergebene Wert (d. h. „r") der Winkel in Radianten ist. In MMBasic können Sie die Funktion RAD(d) verwenden, um einen Winkel von Grad in Radianten umzurechnen („d" ist der Winkel in Grad).

Eine weitere Eigenschaft der meisten Programmiersprachen (einschließlich BASIC) ist, dass Sie Funktionsaufrufe ineinander verschachteln können. Wenn beispielsweise der Winkel in Grad (d. h. „d") gegeben ist, kann der Sinus dieses Winkels mit diesem Ausdruck ermittelt werden:

```
PRINT SIN(RAD(d))
```

In diesem Fall nimmt MMBasic zunächst den Wert von d und wandelt ihn mit der Funktion RAD() in Radianten um. Die Ausgabe dieser Funktion wird dann zur Eingabe für die Funktion SIN().

Die IF- -Anweisung

Das Treffen von Entscheidungen ist der Kern der meisten Computerprogramme, und in BASIC geschieht dies in der Regel mit der IF-Anweisung. Diese wird fast wie ein englischer Satz geschrieben:

```
IF Bedingung THEN Aktion
```

Die *Bedingung* ist in der Regel ein Vergleich wie „gleich", „kleiner als", „größer als" usw.

Beispiel:

```
IF Temp < 25 THEN PRINT "Kalt"
```

`Temp` wäre eine Variable, die die aktuelle Temperatur (in ºC) enthält, und `PRINT „Cold"` die auszuführende Aktion.

Es gibt eine Reihe von Tests, die Sie durchführen können:

=	gleich	<>	ungleich
<	kleiner als	<=	kleiner oder gleich
>	größer als	>=	größer als oder gleich

Sie können auch eine ELSE-Klausel hinzufügen, die ausgeführt wird, wenn die ursprüngliche Bedingung als falsch getestet wurde:

```
IF Bedingung THEN wahre Aktion ELSE falsche Aktion
```

Beispielsweise werden verschiedene Aktionen ausgeführt, wenn die Temperatur unter 25 oder 25 oder mehr liegt:

```
IF Temp < 25 THEN PRINT „Kalt" ELSE PRINT „Heiß"
```

In den vorherigen Beispielen wurden alle einzeilige IF-Anweisungen verwendet, aber Sie können auch mehrzeilige IF-Anweisungen verwenden. Diese sehen wie folgt aus:

```
IF Bedingung THEN
     wahre-Aktion
     wahre-Aktion
ENDIF
```

oder

```
IF Bedingung THEN
     wahre-Aktion
     wahre-Aktion
ELSE
     false-action
     false-action
ENDIF
```

Im Gegensatz zur einzeiligen IF-Anweisung können Sie viele wahre Aktionen haben, die jeweils in einer eigenen Zeile stehen, und ebenso viele falsche Aktionen. Im Allgemeinen ist die einzeilige IF-Anweisung praktisch, wenn Sie eine einfache Aktion ausführen müssen, während die mehrzeilige Version viel leichter zu verstehen ist, wenn die Aktionen zahlreich und komplizierter sind.

Ein Beispiel für eine mehrzeilige IF-Anweisung mit mehr als einer Aktion ist:

```
IF Amount < 100 THEN
     PRINT "Zu niedrig"
     PRINT „Mindestwert ist 100”
ELSE
     PRINT „Eingabe akzeptiert”
     SaveToSDCard
     PRINT „Zweiten Betrag eingeben”
ENDIF
```

Beachten Sie, dass im obigen Beispiel jede Aktion eingerückt ist, um zu zeigen, zu welchem Teil der IF-Struktur sie gehört. Einrückungen sind nicht zwingend erforderlich, machen ein Programm jedoch für jemanden, der damit nicht vertraut ist, viel verständlicher und sind daher sehr zu empfehlen.

In einer mehrzeiligen IF-Anweisung können Sie mit dem Befehl ELSE IF zusätzliche Tests durchführen. Dies lässt sich am besten anhand eines Beispiels erklären (die Temperaturen sind alle in °C angegeben):

```
IF Temp < 0 THEN
     PRINT „Frost“ ELSE IF
Temp < 20 THEN
     PRINT „Kalt”
ELSE IF Temp < 35 THEN
     PRINT „Warm”
ELSE
     DRUCKE „Heiß”
ENDIF
```

ELSE IF verwendet dieselben Tests wie ein gewöhnliches IF (d. h. <, <= usw.), aber dieser Test wird nur durchgeführt, wenn der vorhergehende Test falsch war. So erhalten Sie beispielsweise nur die Meldung *„Warm“*, wenn `Temp < 0` fehlgeschlagen ist und `Temp < 20` fehlgeschlagen ist, aber `Temp < 35` wahr war. Das letzte ELSE fängt den Fall ab, in dem alle Tests falsch waren.

Ein Ausdruck wie `Temp < 20` wird von MMBasic entweder als wahr oder falsch ausgewertet, wobei wahr den Wert eins und falsch den Wert null hat. Sie können dies sehen, wenn Sie Folgendes in die Konsole eingeben:

```
PRINT 30 > 20
```

MMBasic gibt `1` aus, was bedeutet, dass der Wert des Ausdrucks wahr ist.

Ebenso wird im folgenden Fall `die Zahl 0` ausgegeben, was bedeutet, dass der Ausdruck als falsch ausgewertet wurde.

```
PRINT 30 < 20
```

Die IF-Anweisung kümmert sich nicht wirklich darum, wie die Bedingung tatsächlich lautet, sondern wertet lediglich die Bedingung aus und nimmt bei einem Ergebnis von Null den Wert als falsch und bei einem Ergebnis ungleich Null den Wert als wahr an.

Dies ermöglicht einige praktische Abkürzungen. Wenn beispielsweise `BalanceCorrect` eine Variable ist, die wahr (ungleich Null) ist, wenn eine bestimmte Funktion des Programms korrekt ist, kann Folgendes verwendet werden, um eine Entscheidung auf der Grundlage dieses Werts zu treffen:

```
IF BalanceCorrect THEN …etwas tun…
```

FOR- -Schleifen

Eine weitere häufige Anforderung in der Programmierung ist die Wiederholung einer Reihe von Aktionen. Beispielsweise möchten Sie vielleicht alle sieben Tage der Woche durchlaufen und für jeden Tag dieselbe Funktion ausführen. BASIC bietet für diese Art von Aufgabe die FOR-Schleife, die wie folgt funktioniert:

```
FOR day = 1 TO 7
   Führen Sie eine Aktion basierend auf dem Wert von „day" aus
NEXT day
```

Zunächst wird die Variable „`day`" erstellt und ihr der Wert 1 zugewiesen. Das Programm führt dann die folgenden Anweisungen aus, bis es zur NEXT-Anweisung gelangt. Diese weist den BASIC-Interpreter an, den Wert von „`day`" zu erhöhen, zur vorherigen FOR-Anweisung zurückzukehren und die folgenden Anweisungen ein zweites Mal auszuführen. Dies wird so lange wiederholt, bis der Wert von „`day`" 7 überschreitet. Dann verlässt das Programm die Schleife und fährt mit den Anweisungen nach der NEXT-Anweisung fort.

Als einfaches Beispiel können Sie die Zahlen von eins bis zehn wie folgt ausgeben:

```
FOR nbr = 1 TO 10 PRINT
   nbr,;
NEXT nbr
```

Das Komma am Ende der PRINT-Anweisung weist den Interpreter an, nach dem Drucken der Zahl zur nächsten Tabulator-Spalte zu springen, und das Semikolon lässt den Cursor in dieser Zeile stehen, anstatt automatisch zur nächsten Zeile zu springen. Dadurch werden die Zahlen in ordentlichen Spalten über die Seite gedruckt.

Das Ergebnis sieht wie folgt aus:

```
1         2         3         4         5         6         7         8         9         10
```

Die FOR-Schleife hat noch ein paar zusätzliche Tricks auf Lager. Mit dem Schlüsselwort STEP können Sie den Wert ändern, um den die Variable erhöht wird. So werden beispielsweise mit dem folgenden Code nur die ungeraden Zahlen ausgegeben:

```
FOR nbr = 1 TO 10 STEP 2 PRINT
   nbr,;
NEXT nbr
```

Der Wert des Schritts (oder Inkrementwerts) ist standardmäßig eins, wenn das Schlüsselwort STEP nicht verwendet wird, aber Sie können ihn auf einen beliebigen Wert setzen.

Wenn MMBasic die Variable inkrementiert, überprüft es, ob die Variable den TO-Wert überschritten hat, und wenn ja, verlässt es die Schleife. Im obigen Beispiel erreicht der Wert von `nbr` also neun und wird ausgegeben, aber in der nächsten Schleife ist `nbr` elf, und an diesem Punkt verlässt die Ausführung die Schleife. Diese Prüfung wird auch zu Beginn der Schleife durchgeführt. Wenn beispielsweise der Wert der Variablen zu Beginn den TO-Wert überschreitet, wird die Schleife nie ausgeführt, nicht einmal.

Durch Setzen des STEP-Wertes auf eine negative Zahl können Sie die FOR-Schleife verwenden, um von einer hohen Zahl zu einer niedrigen Zahl zu gelangen. In diesem Fall muss die Startzahl größer als die TO-Zahl sein.

Beispielsweise werden mit dem folgenden Code die Zahlen von 1 bis 10 in umgekehrter Reihenfolge ausgegeben:

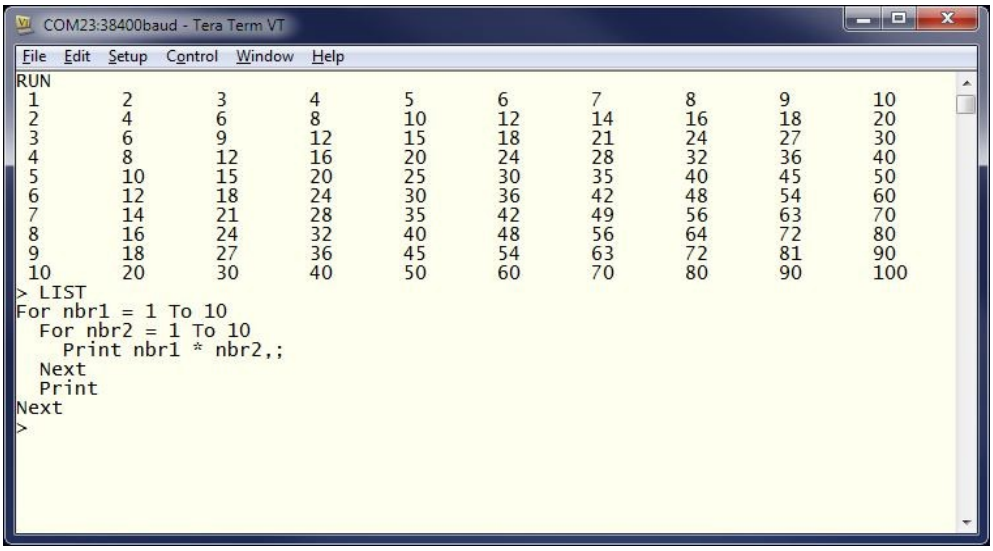
```
FOR nbr = 10 TO 1 STEP -1 PRINT
   nbr,;
NEXT nbr
```

Multiplikations stabelle

Um die Funktionsweise von Schleifen und ihren Nutzen weiter zu veranschaulichen, verwendet das folgende kurze Programm zwei FOR-Schleifen, um die Multiplikationstabelle auszudrucken, die wir alle in der Schule gelernt haben. Das Programm dafür ist nicht kompliziert:

```
FOR nbr1 = 1 to 10
   FOR nbr2 = 1 to 10 PRINT
      nbr1 * nbr2,;
   NEXT nbr2 PRINT
NEXT nbr1
```

Die Ausgabe ist im folgenden Screenshot zu sehen, der auch eine Auflistung des Programms enthält.



Sie müssen die Logik dieses Beispiels Zeile für Zeile durcharbeiten, um zu verstehen, was es tut. Im Wesentlichen besteht es aus einer Schleife innerhalb einer anderen. Die innere Schleife, die die Variable `nbr2` inkrementiert, druckt eine horizontale Zeile der Tabelle. Wenn diese Schleife beendet ist, wird der folgende PRINT-Befehl ausgeführt, der nichts zu drucken hat – er gibt also einfach eine neue Zeile aus (d. h. beendet die von der inneren Schleife gedruckte Zeile).

Das Programm führt dann eine weitere Iteration der äußeren Schleife aus, indem es `nbr1` erhöht und die innere Schleife erneut ausführt. Wenn die äußere Schleife schließlich erschöpft ist (wenn `nbr1` größer als 10 ist), erreicht das Programm das Ende und wird beendet.

Ein letzter Hinweis: Sie können den Variablennamen in der NEXT-Anweisung weglassen, und MMBasic erät, auf welche Variable Sie sich beziehen. Es empfiehlt sich jedoch, den Namen anzugeben, damit andere Personen, die das Programm lesen, es leichter verstehen können. Sie können auch mehrere Schleifen beenden, indem Sie in der NEXT-Anweisung eine durch Kommas getrennte Liste von Variablen angeben. Beispiel:

```
FOR var1 = 1 TO 5
   FÜR var2 = 10 bis 13
      DRUCKEN var1 * var2
NEXT var1, var2
```

DO- -Schleifen

Eine weitere Methode zum Erstellen von Schleifen ist die DO…LOOP-Struktur, die wie folgt aussieht:

```
DO WHILE Bedingung
   <Anweisung>
   <Anweisung>
LOOP
```

Zunächst wird die *Bedingung* geprüft. Ist sie wahr, werden die Anweisungen bis zum LOOP-Befehl ausgeführt. An dieser Stelle kehrt das Programm zur DO-Anweisung zurück und die *Bedingung* wird erneut geprüft. Ist sie weiterhin wahr, wird die Schleife erneut ausgeführt. Die Bedingung ist dieselbe wie im IF-Befehl (d. h. `X < Y`).

Beispielsweise wird mit dem folgenden Code das Wort „Hello“ 4 Sekunden lang auf der Konsole ausgegeben und dann angehalten:

```
Timer = 0
DO WHILE Timer < 4000
   PRINT „Hello“
LOOP
```

Beachten Sie, dass `Timer` eine Funktion innerhalb von MMBasic ist, die die Zeit in Millisekunden seit dem Zurücksetzen des Timers zurückgibt. Ein Zurücksetzen erfolgt durch Zuweisen von Null zu `Timer` (wie oben) oder beim Einschalten des PicoMite.

Eine Variante der DO-LOOP-Struktur ist die folgende:

```
DO
    <Anweisung>
    <Anweisung>
LOOP UNTIL Bedingung
```

In dieser Anordnung wird die Schleife zunächst einmal ausgeführt, dann wird die *Bedingung* geprüft, und wenn die Bedingung falsch ist, wird die Schleife so lange wiederholt, bis die *Bedingung* wahr wird. Beachten Sie, dass die Prüfung in LOOP UNTIL das Gegenteil von DO WHILE ist.

Ähnlich wie im vorherigen Beispiel wird auch im folgenden Beispiel vier Sekunden lang „Hello” ausgegeben:

```
Timer = 0 DO
   PRINT „Hello”
LOOP UNTIL Timer >= 4000
```

Beide Formen der DO-LOOP-Anweisung führen im Wesentlichen dasselbe aus, sodass Sie die Struktur verwenden können, die am besten zu der von Ihnen gewünschten Logik passt.

Schließlich ist es möglich, eine DO-Schleife ohne Bedingungen zu verwenden, d. h.

```
TUN
    <Anweisung>
    <Anweisung>
LOOP
```

Diese Konstruktion führt eine Endlosschleife aus, und Sie als Programmierer müssen eine Möglichkeit bereitstellen, um die Schleife explizit zu verlassen (dies geschieht mit dem Befehl EXIT DO). Beispiel:

```
Timer = 0 DO
   PRINT „Hallo“
   IF Timer >= 4000 THEN EXIT DO LOOP
```

Konsolen- seingabe

Neben der Ausgabe von Daten für den Benutzer möchten Ihre Programme auch Eingaben vom Benutzer erhalten. Damit dies funktioniert, müssen Sie Tastenanschläge von der Konsole erfassen. Dies kann mit dem Befehl INPUT erfolgen. In seiner einfachsten Form lautet der Befehl:

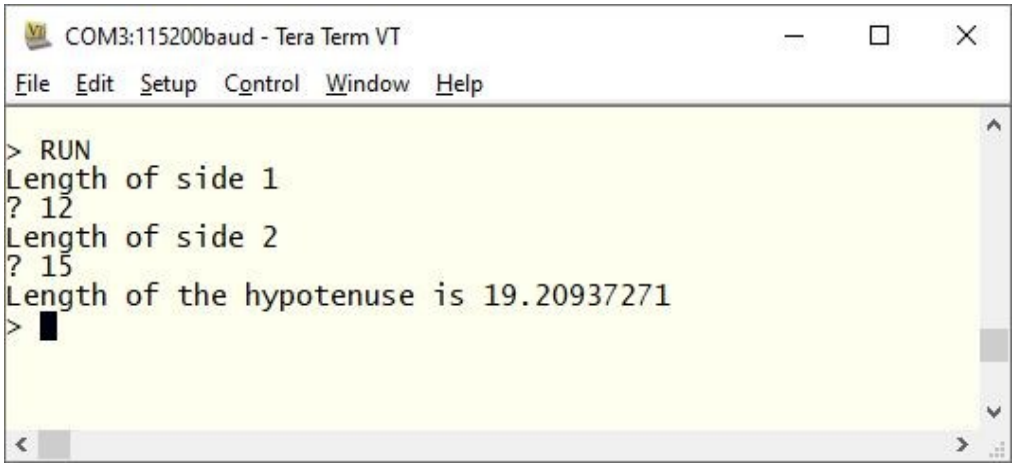
```
INPUT var
```

Dieser Befehl gibt ein Fragezeichen auf dem Konsolenbildschirm aus und wartet darauf, dass eine Zahl eingegeben und die Eingabetaste gedrückt wird. Diese Zahl wird dann der Variablen `var` zugewiesen.

Das folgende Programm erweitert beispielsweise den Ausdruck zur Berechnung der Hypotenuse eines Dreiecks, indem es dem Benutzer ermöglicht, die Längen der anderen Seiten über die Konsole einzugeben.

```
PRINT „Länge der Seite 1” INPUT a
PRINT "Länge der Seite 2" INPUT b
PRINT „Länge der Hypotenuse ist“ SQR(a * a + b * b)
```

Dies ist ein Screenshot einer typischen Sitzung:



Der Befehl INPUT kann auch Ihre Eingabeaufforderung ausgeben, sodass Sie keinen separaten PRINT-Befehl benötigen. Dies funktioniert beispielsweise genauso wie das obige Programm:

```
INPUT "Länge der Seite 1"; a
INPUT "Länge der Seite 2"; b
PRINT "Die Länge der Hypotenuse beträgt" SQR(a * a + b * b)
```

Mit dem Befehl INPUT können Sie schließlich eine Reihe von durch Kommas getrennten Zahlen eingeben, wobei jede Zahl in einer anderen Variablen gespeichert wird.

Beispiel:

```
INPUT „Geben Sie die Länge der beiden Seiten ein: “, a, b PRINT
„Die Länge der Hypotenuse beträgt“ SQR(a * a + b * b)
```

Wenn der Benutzer `12,15` eingibt, wird die Zahl `12` in der Variablen `a` und `15` in `der Variablen b` gespeichert.

Eine weitere Methode, um Eingaben von der Konsole zu erhalten, ist der Befehl LINE INPUT. Dieser Befehl ruft die gesamte vom Benutzer eingegebene Zeile ab und weist sie einer String-Variablen zu. Wie beim Befehl INPUT können Sie auch hier eine Eingabeaufforderung festlegen. Hier ein einfaches Beispiel:

```
LINE INPUT "Wie heißt du? ", s$ PRINT "Hallo "
s$
```

Wir werden später in diesem Tutorial auf String-Variablen eingehen, aber im Moment können Sie sich diese als Variablen vorstellen, die eine Folge von Zeichen enthalten. Wenn Sie das obige Programm ausführen und bei der Eingabeaufforderung „`John`“ eingeben, antwortet das Programm mit „`Hallo John`“.

Manchmal möchten Sie nicht warten, bis der Benutzer die Eingabetaste drückt, sondern jedes Zeichen sofort nach der Eingabe erhalten. Dies ist mit der Funktion INKEY$ möglich, die den Wert des Zeichens als Zeichenkette mit nur einem Zeichen oder als leere Zeichenkette (d. h. ohne Zeichen) zurückgibt, wenn nichts eingegeben wurde.

GOTO- und „ “-Bezeichnungen

Eine Methode zur Steuerung des Programmablaufs ist der Befehl GOTO. Dieser weist MMBasic im Wesentlichen an, zu einem anderen Teil des Programms zu springen und die Ausführung dort fortzusetzen. Das Ziel von GOTO ist ein Label, das zunächst erklärt werden muss.

Ein Label ist ein Bezeichner, der einen Teil des Programms markiert. Es muss an erster Stelle in der Zeile stehen und mit einem Doppelpunkt (:) enden. Der Name, den Sie verwenden, kann bis zu 31 Zeichen lang sein und muss denselben Regeln folgen wie der Name einer Variablen. In der folgenden Programmzeile ist beispielsweise LoopBack ein Label:

```
LoopBack: a = a + 1
```

Wenn Sie den Befehl GOTO verwenden, um zu diesem bestimmten Teil des Programms zu springen, würden Sie den Befehl wie folgt verwenden:

```
GOTO LoopBack
```

Um all dies in einen Zusammenhang zu bringen, gibt das folgende Programm alle Zahlen von 1 bis 10 aus:

```
z = 0
LoopBack: z = z + 1
PRINT z
IF z < 10 THEN GOTO LoopBack
```

Das Programm beginnt damit, die Variable `z` auf Null zu setzen und sie dann in der nächsten Zeile auf 1 zu erhöhen. Der Wert von `z` wird ausgegeben und dann geprüft, ob er kleiner als 10 ist. Ist er kleiner als 10, springt die Programmausführung zurück zum Label `LoopBack`, wo der Vorgang wiederholt wird. Schließlich erreicht der Wert von `z` den Wert 10, und das Programm läuft bis zum Ende durch und wird beendet.

Beachten Sie, dass eine FOR-Schleife dasselbe leisten kann (und einfacher ist), sodass dieses Beispiel lediglich dazu dient, die Funktionsweise des Befehls GOTO zu veranschaulichen.

In der Vergangenheit hat der GOTO-Befehl einen schlechten Ruf bekommen. Das liegt daran, dass man mit GOTOs ein Programm erstellen kann, das ständig von einem Punkt zum anderen springt (oft als „Spaghetti-Code" bezeichnet), und solche Programme sind für andere Programmierer fast unmöglich zu verstehen. Mit Konstrukten wie den mehrzeiligen IF-Anweisungen ist der Bedarf an GOTO-Anweisungen zurückgegangen, und sie sollten nur verwendet werden, wenn es keine andere Möglichkeit gibt, den Programmablauf zu ändern.

Prüfung auf Primzahlen

Das folgende einfache Programm vereint viele der zuvor besprochenen Programmierfunktionen.

```
DO
  InpErr:
  PRINT
  INPUT „Geben Sie eine Zahl ein: “;
  a IF a < 2 THEN
    PRINT „Die Zahl muss größer oder gleich 2 sein“ GOTO InpErr
  ENDIF

  Divs = 0
  FOR x = 2 TO SQR(a)
    r = a/x
    IF r = FIX(r) THEN Divs = Divs + 1 NEXT x

  PRINT a " ist ";
  WENN Divs > 0 DANN DRUCKE „nicht “;
  PRINT "eine Primzahl." LOOP
```

Zunächst wird (auf der Konsole) zur Eingabe einer Zahl aufgefordert. Nach der Eingabe wird geprüft, ob es sich um eine Primzahl handelt, und eine entsprechende Meldung angezeigt.

Es beginnt mit einer DO-Schleife ohne Bedingung – sie wird also endlos fortgesetzt. Das ist genau das, was wir wollen. Das bedeutet, dass nach der Eingabe einer Zahl durch den Benutzer gemeldet wird, ob es sich um eine Primzahl handelt oder nicht, und dann die Schleife fortgesetzt wird, um eine weitere Zahl abzufragen. Der Benutzer kann das Programm (falls gewünscht) durch Eingabe des Abbruchzeichens (normalerweise STRG-C) beenden.

Das Programm gibt dann eine Eingabeaufforderung für den Benutzer aus, die mit einem Semikolon beendet wird. Das bedeutet, dass der Cursor am Ende der Eingabeaufforderung für den Befehl INPUT stehen bleibt, der die Zahl abruft und in der Variablen a speichert.

Anschließend wird die Zahl überprüft. Ist sie kleiner als 2, wird eine Fehlermeldung ausgegeben, das Programm springt zurück und fordert erneut zur Eingabe einer Zahl auf.

Nun können wir prüfen, ob die Zahl eine Primzahl ist. Das Programm verwendet eine FOR-Schleife, um die möglichen Teiler durchzugehen und zu prüfen, ob jeder einzelne die eingegebene Zahl ohne Rest teilen kann. Bei jedem Durchlauf erhöht das Programm die Variable `Divs`.

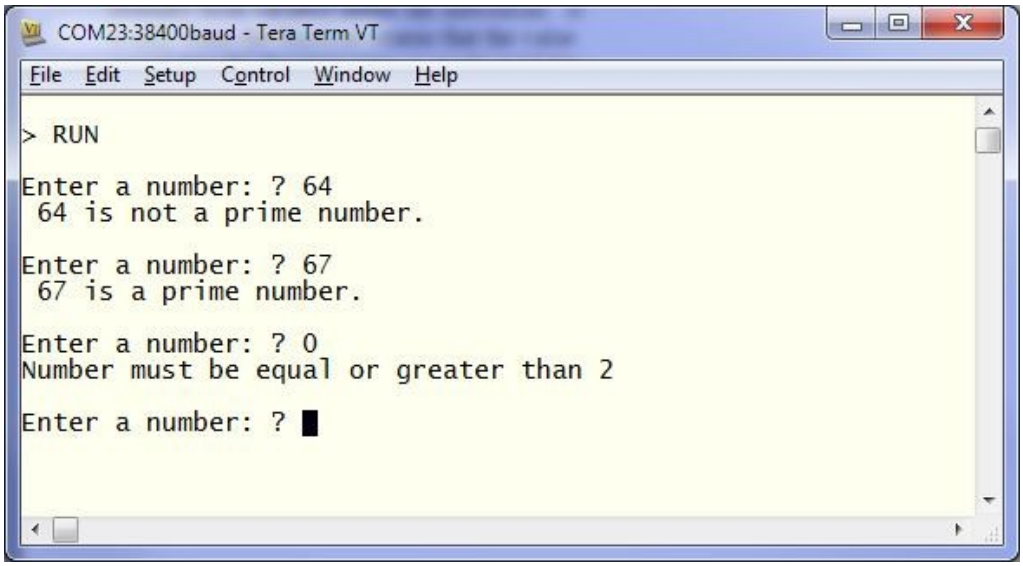
Beachten Sie, dass der Test mit der Funktion FIX(r) durchgeführt wird, die einfach alle Ziffern nach dem Dezimalpunkt entfernt. Die Bedingung `r = FIX(r)` ist also wahr, wenn `r` eine ganze Zahl ist (d. h. keine Ziffern nach dem Dezimalpunkt hat).

Schließlich erstellt das Programm die Nachricht für den Benutzer. Der entscheidende Punkt ist, dass, wenn die Variable `Divs` größer als Null ist, dies bedeutet, dass eine oder mehrere Zahlen gefunden wurden, die sich gleichmäßig durch die Testzahl teilen lassen. In diesem Fall fügt die IF-Anweisung das Wort „nicht“ in die Ausgabemeldung ein.

Wenn beispielsweise die eingegebene Zahl 21 war, erhält der Benutzer folgende Antwort:

```
21 ist keine Primzahl.
```

Dies ist das Ergebnis der Ausführung des Programms und ein Teil der Ausgabe:



Sie können dieses Programm testen, indem Sie es mit dem Editor (dem Befehl EDIT) eingeben.

Mit Ihren neu erworbenen Kenntnissen könnten Sie dann versuchen, das Programm effizienter zu gestalten. Da das Programm beispielsweise zählt, wie oft eine Zahl durch die Testzahl teilbar ist, dauert es viel länger als nötig, um eine Nicht-Primzahl zu erkennen. Das Programm würde viel effizienter laufen, wenn es bei der ersten Zahl, die sich gleichmäßig teilen lässt, aus der FOR-Schleife springen würde. Dazu könnten Sie den Befehl GOTO verwenden oder den Befehl EXIT FOR, der die FOR-Schleife sofort beendet.

Weitere Effizienzsteigerungen sind möglich, indem nur die Division mit ungeraden Zahlen getestet wird (durch einen ersten Test für eine gerade Zahl, dann den Start der FOR-Schleife bei 3 und die Verwendung von STEP 2) oder indem nur Primzahlen für den Test verwendet werden (was jedoch wesentlich komplizierter wäre).

Arrays

Arrays sind etwas, das Sie auf den ersten Blick wahrscheinlich nicht für nützlich halten, aber wenn Sie sie einmal brauchen, werden Sie sie als sehr praktisch empfinden.

Ein Array lässt sich am besten als eine Reihe von Briefkästen für einen Block von Wohnungen oder Eigentumswohnungen vorstellen, wie rechts dargestellt. Die Briefkästen befinden sich alle an derselben Adresse, und jeder Kasten steht für eine Wohnung oder Eigentumswohnung an dieser Adresse. Sie können einen Brief in den Kasten für Wohnung eins, Wohnung zwei usw. legen.

Ähnlich ist ein Array in BASIC eine einzelne Variable mit mehreren Untereinheiten (in BASIC als *Elemente* bezeichnet), die nummeriert sind. Sie können Daten in Element eins, Element zwei usw. ablegen. In BASIC wird ein Array mit dem Befehl DIM erstellt, zum Beispiel:

```
DIM numarr(300)
```

Dadurch wird ein Array mit dem Namen `numarr` erstellt, das 301 Elemente (stellen Sie sich diese als Briefkästen vor) im Bereich von 0 bis 300 enthält. Standardmäßig beginnt ein Array bei Null, daher gibt es ein zusätzliches Element, sodass die Gesamtzahl 301 beträgt. Um ein bestimmtes Element im Array (d. h. einen bestimmten Briefkasten) anzugeben, verwenden Sie einen Index, der einfach die Nummer des Array-Elements ist, auf das Sie zugreifen möchten. Wenn Sie beispielsweise das Element Nummer 100 in diesem Array auf (sagen wir) die Zahl 876 setzen möchten, würden Sie dies wie folgt tun:

```
numarr(100) = 876
```

Normalerweise ist der Index eines Arrays keine konstante Zahl wie in diesem Beispiel (d. h. 100), sondern eine Variable, die geändert werden kann, um auf verschiedene Array-Elemente zuzugreifen.

Als Beispiel für die Verwendung eines Arrays betrachten wir den Fall, dass Sie die Höchsttemperatur für jeden Tag des Jahres aufzeichnen und am Ende des Jahres den Gesamtdurchschnitt berechnen möchten. Sie könnten normale Variablen verwenden, um die Temperatur für jeden Tag aufzuzeichnen, aber Sie würden 365 davon benötigen, was Ihr Programm ziemlich unhandlich machen würde. Stattdessen könnten Sie ein Array definieren, um die Werte wie folgt zu speichern:

```
DIM days(365)
```

Jeden Tag müssten Sie die Temperatur an der richtigen Stelle im Array speichern. Wenn die Nummer des Tages im Jahr in der Variablen `doy` und die Höchsttemperatur in der Variablen `maxtemp` gespeichert wäre, würden Sie den Messwert wie folgt speichern:

```
days(doy) = maxtemp
```

Am Ende des Jahres wäre es einfach, den Jahresdurchschnitt zu berechnen. Zum Beispiel:

```
total = 0
FOR i = 1 bis 365
   total = total + days(i) NEXT i
PRINT „Durchschnitt ist:“ Gesamtwert / 365
```

Das ist viel einfacher, als 365 einzelne Variablen zu addieren und zu mitteln.

Das obige Array war eindimensional, aber Sie können auch mehrdimensionale Arrays verwenden. Um auf unsere Analogie mit den Briefkästen zurückzukommen: Ein zweidimensionales Array könnte man sich wie ein mehrstöckiges Wohnhaus vorstellen. Ein Block könnte eine Reihe von vier Briefkästen für die erste Etage, eine weitere Reihe von vier Briefkästen für die zweite Etage usw. haben. Um einen Brief in einen Briefkasten zu werfen, müssen Sie die Etagennummer und die Nummer der Einheit auf dieser Etage angeben.

In BASIC wird ein solches Array mit zwei durch ein Komma getrennten Indizes angegeben. Zum Beispiel:

```
LetterBox(Etage, Einheit)
```

Als praktisches Beispiel nehmen wir an, Sie müssten die Höchsttemperatur für jeden Tag über einen Zeitraum von fünf Jahren aufzeichnen. Dazu könnten Sie das Array wie folgt dimensionieren:

```
DIM-Tage (365, 5)
```

Der erste Index ist der Tag im Jahr und der zweite ist eine Zahl, die das Jahr angibt. Wenn Sie den Tag 100 im Jahr 3 auf 24 Grad einstellen möchten, würden Sie dies wie folgt tun:

```
days(100, 3) = 24
```

In MMBasic für die PicoMite-Firmware können Sie mit dem RP2040-Prozessor bis zu sechs Dimensionen und mit dem RP2350-Prozessor bis zu fünf Dimensionen verwenden. Die Größe eines Arrays ist nur durch die Menge des verfügbaren freien RAM begrenzt.

Ganzzahlen

Bisher waren alle Zahlen und Variablen, die wir verwendet haben, Gleitkommazahlen. Wie bereits erläutert, sind Gleitkommazahlen praktisch, da sie die Stellen nach dem Komma verfolgen und bei Divisionen ein sinnvolles Ergebnis liefern. Wenn Sie also einfach nur Ihre Arbeit erledigen möchten und sich nicht um die Details kümmern, sollten Sie bei Gleitkommazahlen bleiben.

Die Einschränkung von Gleitkommazahlen besteht jedoch darin, dass sie Zahlen als Annäherungswerte mit einer Genauigkeit von 14 Stellen in der PicoMite-Firmware speichern. In den meisten Fällen ist diese Eigenschaft von Gleitkommazahlen kein Problem, aber es gibt einige Fälle, in denen Sie größere Zahlen genau speichern müssen.

Nehmen wir als Beispiel an, Sie möchten die Zeit auf die Mikrosekunde genau manipulieren, um zwei verschiedene Datums-/Zeitangaben zu vergleichen und herauszufinden, welche davon früher ist. Der einfachste Weg, dies zu tun, besteht darin, das Datum/die Uhrzeit in die Anzahl der Mikrosekunden seit einem bestimmten Datum (z. B. dem 1. Januar im Jahr Null) umzuwandeln – dann ist es nur noch eine Frage der arithmetischen Vergleichung in einer IF-Anweisung, um das frühere der beiden Daten zu ermitteln.

Das Problem besteht darin, dass die Anzahl der Mikrosekunden seit diesem Datum den Genauigkeitsbereich von Gleitkomma-Variablen überschreiten wird, und hier kommen Integer-Variablen ins Spiel. Eine Integer-Variable kann sehr große Zahlen bis zu neun Millionen Millionen Millionen (oder ±9223372036854775807, um genau zu sein) genau speichern.

Der Nachteil der Verwendung einer Ganzzahl ist, dass sie keine Brüche (d. h. Zahlen nach dem Dezimalpunkt) speichern kann. Jede Berechnung, die ein gebrochenes Ergebnis liefert, wird bei der Zuweisung zu einer Ganzzahl auf die nächste ganze Zahl auf- oder abgerundet. Diese Eigenschaft kann jedoch bei Geldgeschäften nützlich sein – beispielsweise möchten Sie niemandem eine Rechnung über 100,13333333333 $ schicken.

Es ist einfach, eine Ganzzahlvariable zu erstellen. Fügen Sie einfach das Prozentzeichen (%) als Suffix zum Variablennamen hinzu. Beispielsweise ist `sec%` eine Ganzzahlvariable. Innerhalb eines Programms können Sie Ganzzahlen und Gleitkommazahlen mischen, und MMBasic nimmt die erforderlichen Umrechnungen vor. Wenn Sie jedoch die volle Genauigkeit von Ganzzahlen beibehalten möchten, sollten Sie eine Vermischung der beiden vermeiden.

Genau wie bei Gleitkommazahlen können Sie auch Arrays von Ganzzahlen erstellen. Dazu müssen Sie lediglich das Prozentzeichen als Suffix an den Array-Namen anhängen. Beispiel: `days%(365, 5)`.

Anfänger sind oft verwirrt, wann sie Fließkommazahlen oder Ganzzahlen verwenden sollen, und die Antwort ist einfach: Verwenden Sie immer Fließkommazahlen, es sei denn, Sie benötigen eine extrem hohe Genauigkeit. Das kommt nicht oft vor, aber wenn Sie sie brauchen, werden Sie feststellen, dass Ganzzahlen sehr nützlich sind.

Zeichenfolgen

Zeichenfolgen sind ein weiterer Variablentyp (wie Gleitkommazahlen und Ganzzahlen). Zeichenfolgen werden verwendet, um eine Folge von Zeichen zu speichern. Beispielsweise im Befehl:

```
PRINT „Hallo”
```

ist die Zeichenkette „Hello” eine Zeichenkettenkonstante. Beachten Sie, dass eine Konstante etwas ist, das sich nicht ändert (im Gegensatz zu einer Variablen, die sich ändern kann), und dass Zeichenkettenkonstanten immer in doppelte Anführungszeichen gesetzt werden.

String-Variablennamen verwenden das Dollarzeichen ($) als Suffix, um sie als String statt als normale Gleitkommavariable zu kennzeichnen, und Sie können ihren Wert mit einer gewöhnlichen Zuweisung festlegen. Nachfolgend finden Sie einige Beispiele (beachten Sie, dass im zweiten Beispiel ein Array von Strings verwendet wird):

```
Car$ = "Holden"
Country$(12) = "India"
Name$ = "Fred"
```

Sie können Zeichenfolgen auch mit dem Pluszeichen verbinden:

```
Wort1$ = "Hallo" Wort2$
= "Welt"
Greeting$ = Word1$ + " " + Word2$
```

In diesem Fall lautet der Wert von `Greeting$` „`Hello World`".

Zeichenfolgen können auch mit Operatoren wie = (gleich), <> (ungleich), < (kleiner als) usw. verglichen werden. Beispiel:

```
IF Car$ = "Holden" THEN PRINT "War ein in Australien hergestelltes Auto"
```

Der Vergleich erfolgt unter Verwendung des vollständigen ASCII-Zeichensatzes, sodass vor einem druckbaren Zeichen ein Leerzeichen steht. Außerdem wird bei dem Vergleich zwischen Groß- und Kleinschreibung unterschieden, sodass „holden" nicht mit „Holden" übereinstimmt. Mit der Funktion UCASE() können Sie die Zeichenfolge in Großbuchstaben umwandeln und so einen Vergleich ohne Berücksichtigung der Groß-/Kleinschreibung durchführen. Beispiel:

```
IF UCASE$(Car$) = "HOLDEN" THEN PRINT "War ein in Australien hergestelltes Auto"
```

Sie können Zeichenfolgen-Arrays verwenden, müssen jedoch bei deren Deklaration vorsichtig sein, da Ihnen schnell der RAM-Speicher (allgemeiner Speicher zum Speichern von Variablen usw.) ausgehen kann. Der Grund dafür ist, dass MMBasic standardmäßig 255 Byte RAM für jedes Element des Arrays zuweist. Ein Zeichenfolgen-Array mit 100 Elementen belegt beispielsweise standardmäßig 25 KB RAM.

Um dies zu vermeiden, können Sie den Qualifizierer LENGTH verwenden, um die maximale Größe jedes Elements zu begrenzen. Wenn Sie beispielsweise wissen, dass die maximale Länge aller Zeichenfolgen, die im Array gespeichert werden, weniger als 20 Zeichen beträgt, können Sie die folgende Deklaration verwenden, um nur 20 Byte für jedes Element zuzuweisen:

```
DIM MyArray$(100) LENGTH 20
```

Das resultierende Array belegt dann nur 2 KB RAM.

Sie können den LENGTH-Qualifizierer bei der Deklaration einer normalen (nicht als Array definierten) Zeichenfolgenvariablen verwenden und sparen damit 256 Byte RAM, wenn die Länge 9 oder weniger (RP2040) bzw. 15 oder weniger (RP2350) beträgt.

Bearbeiten von Zeichenfolgen in „ "

Die Zeichenfolgenverarbeitung ist eine der Stärken von MMBasic. Mit einigen einfachen Funktionen können Sie Zeichenfolgen zerlegen und allgemein bearbeiten. Die grundlegenden Zeichenfolgenfunktionen sind:

LEFT$(Zeichenfolge$, Anzahl)	Gibt eine Teilzeichenfolge von *string$* mit *nbr* Zeichen vom Anfang (Anfang) der Zeichenfolge.
RIGHT$(Zeichenfolge$, nbr)	Wie oben, gibt jedoch *nbr* Zeichen von rechts (Ende) der Zeichenkette zurück.
MID$(Zeichenfolge$, pos, Anzahl)	Gibt eine Teilzeichenfolge von *string$* mit *nbr* Zeichen zurück, beginnend mit der Zeichen *pos* in der Zeichenfolge (d. h. in der Mitte der Zeichenfolge).

Beispiel: Wenn `S$ = „Dies ist eine Zeichenfolge"`

Dann: `R$ = LEFT$(S$, 7)` würde dazu führen, dass der Wert von R$ auf „This is" gesetzt wird,
und: `R$ = RIGHT$(S$, 8)` würde dazu führen, dass der Wert von R$ auf „a string" gesetzt wird.
Schließlich würde `R$ = MID$(S$, 6, 2)` dazu führen, dass der Wert von R$ auf „is" gesetzt wird.

Beachten Sie, dass in MID$() die erste Zeichenposition in einer Zeichenfolge die Nummer 1 ist, die zweite die Nummer 2 und so weiter. Wenn man also das erste Zeichen als eins zählt, ist die sechste Position der Anfang des Wortes „is".

Eine weitere nützliche Funktion ist:

INSTR(Zeichenfolge$, Muster$) Gibt eine Zahl zurück, die die Position angibt, an der *pattern$* in *string*$.

Dies kann verwendet werden, um innerhalb einer Zeichenfolge nach einer anderen Zeichenfolge zu suchen. Die zurückgegebene Zahl ist die Position der Teilzeichenfolge innerhalb der Hauptzeichenfolge. Wie bei MID$() ist der Anfang der Zeichenfolge Position 1.

Beispiel: Wenn S$ = „Dies ist eine Zeichenfolge“
Dann: `pos = INSTR(S$, „ “)`
würde dazu führen, dass `pos` auf die Position des ersten Leerzeichens in S$ (d. h. 5) gesetzt wird.

INSTR() kann mit anderen Funktionen kombiniert werden, sodass dies das erste **Wort** in S$ zurückgeben würde:

```
R$ = LEFT$(S$, INSTR(S$, " ") - 1)
```

Es gibt auch eine erweiterte Version von INSTR():

INSTR(pos, string$, pattern$) Gibt eine Zahl zurück, die die Position angibt, an der *pattern$* in *string$* auftritt, wenn die Suche an der Zeichenposition pos

So können wir das zweite Wort in S$ mit folgendem Befehl finden:

```
pos1 = INSTR(S$, " ")
pos2 = INSTR(pos1 + 1, S$, " ")
r$ = MID$(S$, pos1 + 1, pos2 – pos1)
```

Dieses letzte Beispiel ist ziemlich kompliziert, daher lohnt es sich vielleicht, es im Detail durchzugehen, damit Sie verstehen, wie es funktioniert.

Beachten Sie, dass INSTR() die Zahl Null zurückgibt, wenn die Teilzeichenfolge nicht gefunden wird, und dass jede Zeichenfolgenfunktion einen Fehler ausgibt (und das Programm anhält), wenn diese als Zeichenposition verwendet wird. In einem praktischen Programm würden Sie also zunächst überprüfen, ob INSTR() die Zahl Null zurückgibt, bevor Sie diesen Wert verwenden.

Beispiel:

```
r$ = ""
pos1 = INSTR(S$, " ")
if pos1 > 0 THEN
  pos2 = INSTR(pos1 + 1, S$, " ")
  if pos2 > 0 THEN r$ = MID$(S$, pos1 + 1, pos2 – pos1) ENDIF
```

Wissenschaftliche Notation ( )

Bevor wir das Thema Datentypen abschließen, müssen wir noch auf Fließkommazahlen und wissenschaftliche Notation eingehen.

Die meisten Zahlen können normal geschrieben werden, zum Beispiel 11 oder 24,5, aber sehr große oder kleine Zahlen sind schwieriger. Beispielsweise wird geschätzt, dass die Anzahl der Sandkörner auf der Erde 7500000000000000000 beträgt. Das Problem bei dieser Zahl ist, dass man leicht den Überblick darüber verliert, wie viele Nullen sie enthält, und es daher schwierig ist, sie mit einer ähnlich großen Zahl zu vergleichen.

Ein Wissenschaftler würde diese Zahl als 7,5 x 10^{18}schreiben, was als wissenschaftliche Notation bezeichnet wird und viel leichter zu verstehen ist.

Bei Verwendung des Befehls PRINT verwendet MMBasic automatisch die wissenschaftliche Notation, wenn sehr große oder kleine Gleitkommazahlen ausgegeben werden. Wenn beispielsweise die oben genannte Zahl in einer Gleitkommavariablen gespeichert wäre, würde der Befehl PRINT sie als `7,5E+18` anzeigen (dies ist die BASIC-Darstellung für 7,5 x 10^{18}). Ein weiteres Beispiel: Die Zahl 0,0000000456 würde als `4,56E-8` angezeigt werden, was 4,56 x 10^{-8}entspricht.

Sie können bei der Eingabe von Konstanten in MMBasic auch die wissenschaftliche Schreibweise verwenden. Beispiel:

```
SandGrains = 7,5E+18
```

MMBasic verwendet die wissenschaftliche Notation nur für Gleitkommazahlen (nicht für Ganzzahlen). Wenn Sie beispielsweise die Anzahl der Sandkörner einer Ganzzahlvariablen zuweisen, würde der Befehl PRINT diese als normale Zahl (mit vielen Nullen) anzeigen.

DIM- -Befehl

Wir haben den Befehl DIM bereits zum Definieren von Arrays verwendet, aber er kann auch zum Erstellen gewöhnlicher Variablen verwendet werden. Sie können beispielsweise gleichzeitig vier String-Variablen wie folgt erstellen:

```
DIM STRING Car, Name, Street, City
```

Beachten Sie, dass diese Variablen mit dem Befehl DIM als Zeichenfolgen definiert wurden und daher kein Suffix $ erforderlich ist. Die Definition allein reicht aus, damit MMBasic ihren Typ identifizieren kann. Ebenso benötigen Sie kein Typ-Suffix, wenn Sie diese Variablen in einem Ausdruck verwenden. Beispiel:

```
City = "Sydney"
```

Sie können auch das Schlüsselwort INTEGER verwenden, um eine Reihe von Ganzzahlvariablen zu definieren, und FLOAT, um dasselbe für Gleitkommavariablen zu tun. Diese Art der Notation kann in ähnlicher Weise zur Definition von Arrays verwendet werden.

Beispiel:

```
DIM INTEGER Sekunden(200)
```

Eine weitere Methode zur Definition des Variablentyps ist die Verwendung des Schlüsselworts AS. Beispiel:

```
DIM Car AS STRING, Name AS STRING, Street AS STRING
```

Diese Methode wird von Microsoft verwendet (MMBasic versucht, die Kompatibilität mit Microsoft aufrechtzuerhalten) und ist nützlich, wenn die Variablen unterschiedliche Typen haben. Beispiel:

```
DIM Car AS STRING, Age AS INTEGER, Value AS FLOAT
```

Sie können jede dieser Methoden zur Definition des Variablentyps verwenden, sie funktionieren alle gleich.

Der Vorteil der Definition von Variablen mit dem Befehl DIM besteht darin, dass sie klar definiert sind (vorzugsweise am Anfang des Programms) und ihr Typ (Float, Integer oder String) nicht falsch interpretiert werden kann.

Sie können dies verstärken, indem Sie die folgenden Befehle ganz oben in Ihrem Programm verwenden:

```
OPTION EXPLICIT OPTION
DEFAULT NONE
```

Der erste Befehl weist MMBasic an, dass alle Variablen vor ihrer Verwendung explizit mit DIM definiert werden müssen. Der zweite Befehl legt fest, dass der Typ aller Variablen bei ihrer Erstellung angegeben werden muss.

Warum sind diese beiden Befehle wichtig?

Ersteres kann dabei helfen, einen häufigen Programmierfehler zu vermeiden, bei dem Sie versehentlich den Namen einer Variablen falsch schreiben. Beispielsweise könnte Ihr Programm die aktuelle Temperatur in einer Variablen namens „`Temp`" gespeichert haben, aber an einer Stelle schreiben Sie diese versehentlich falsch als „`Tmp`". Dies führt dazu, dass MMBasic automatisch eine Variable namens „`Tmp`" erstellt und deren Wert auf Null setzt.

Dies ist natürlich nicht das, was Sie wollen, und es führt zu einem subtilen Fehler, der schwer zu finden sein könnte, selbst wenn Sie wissen, dass etwas nicht stimmt. Wenn Sie hingegen den Befehl OPTION EXPLICIT am Anfang Ihres Programms verwenden, weigert sich MMBasic, die Variable automatisch zu erstellen, und zeigt stattdessen eine Fehlermeldung an, wodurch Ihnen wahrscheinlich Kopfzerbrechen erspart bleibt.

Der Befehl OPTION DEFAULT NONE ist ebenfalls hilfreich, da er MMBasic mitteilt, dass der Programmierer den Typ jeder Variablen bei ihrer Deklaration ausdrücklich angeben muss. Es kann leicht passieren, dass man vergisst, den Typ anzugeben, und wenn MMBasic den Typ automatisch annimmt, kann dies zu unerwarteten Folgen führen.

Bei kleinen, schnellen und einfachen Programmen ist es in Ordnung, MMBasic die automatische Erstellung von Variablen zu überlassen, aber bei größeren Programmen sollten Sie diese Funktion immer mit OPTION EXPLICIT deaktivieren und mit OPTION DEFAULT NONE verstärken.

Wenn eine Variable erstellt wird, wird sie für Float- und Integer-Variablen auf Null und für String-Variablen auf eine leere Zeichenfolge (d. h. ohne Zeichen) gesetzt. Sie können ihren Anfangswert bei der Erstellung mit DIM auf einen anderen Wert setzen.

Beispiel:

```
DIM FLOAT nbr = 12,56
DIM STRING Car = "Ford", City = "Perth"
```

Sie können Arrays auch initialisieren, indem Sie die Initialisierungswerte wie folgt in Klammern setzen:

```
DIM s$(2) = ("zero", "one", "two")
```

Beachten Sie, dass Arrays standardmäßig bei Null beginnen und dieses Array daher tatsächlich drei Elemente mit den Indexnummern 0, 1 und 2 enthält. Aus diesem Grund benötigten wir drei String-Konstanten, um es zu initialisieren.

Konstanten

Eine häufige Anforderung in der Programmierung ist die Definition eines Bezeichners, der einen Wert repräsentiert, ohne dass die Gefahr besteht, dass dieser Wert versehentlich geändert wird – was passieren kann, wenn Variablen für diesen Zweck verwendet werden. Diese werden als Konstanten bezeichnet und können I/O-Pin-Nummern, Signalgrenzen, mathematische Konstanten usw. repräsentieren.

Mit dem Befehl CONST können Sie eine Konstante erstellen. Damit definieren Sie einen Bezeichner, der wie eine Variable funktioniert, aber auf einen Wert festgelegt ist, der nicht geändert werden kann.

Wenn Sie beispielsweise die Spannung einer an Pin 31 angeschlossenen Batterie überprüfen möchten, könnten Sie die entsprechenden Werte wie folgt definieren:

```
CONST BatteryVoltagePin = 31 CONST
BatteryMinimum = 1,5
```

Diese Konstanten können dann im Programm verwendet werden, wo sie für den gelegentlichen Leser aussagekräftiger sind als einfache Zahlen. Zum Beispiel:

```
SETPIN BatteryVoltagePin, AIN
IF PIN(BatteryVoltagePin) < BatteryMinimum THEN SoundAlarm
```

Es ist eine gute Programmierpraxis, Konstanten für alle festen Zahlen zu verwenden, die einen wichtigen Wert darstellen. Normalerweise werden sie am Anfang eines Programms definiert, wo sie gut sichtbar sind und von anderen Programmierern (falls erforderlich) bequem angepasst werden können.

Unterprogramme

Eine Subroutine ist ein eigenständiger Programmcode-Block (ähnlich einem Modul), der von jeder Stelle Ihres Programms aus aufgerufen werden kann. Für Ihr Programm sieht sie wie ein integrierter MMBasic-Befehl aus und kann auch genauso verwendet werden. Angenommen, Sie benötigen einen Befehl, der einen Fehler durch die Ausgabe einer Meldung auf der Konsole signalisiert. Dann könnten Sie die Subroutine wie folgt definieren:

```
SUB ErrMsg
  PRINT "Fehler erkannt" END
SUB
```

Wenn diese Unterroutine in Ihr Programm eingebettet ist, müssen Sie nur den Befehl ErrMsg verwenden, wenn Sie die Meldung anzeigen möchten. Beispiel:

```
IF A < B THEN ErrMsg
```

Die Definition einer Subroutine kann an beliebiger Stelle im Programm stehen, befindet sich jedoch in der Regel am Ende. Wenn MMBasic während der Ausführung Ihres Programms auf die Definition stößt, überspringt es diese einfach.

Das obige Beispiel ist zwar ausreichend, aber es wäre besser, wenn eine nützlichere Meldung angezeigt werden könnte, die bei jedem Aufruf der Unterroutine angepasst werden kann. Dies kann erreicht werden, indem eine Zeichenfolge als Argument (manchmal auch als Parameter bezeichnet) an die Unterroutine übergeben wird.

In diesem Fall würde die Definition der Unterroutine wie folgt aussehen:

```
SUB ErrMsg Msg$
   PRINT "Fehler: " + Msg$ END
SUB
```

Wenn Sie dann die Subroutine aufrufen, können Sie die auszugebende Zeichenfolge in der Befehlszeile der Subroutine angeben.

Beispiel:

```
IF A < B THEN ErrMsg "Zahl zu klein"
```

Wenn die Subroutine wie oben aufgerufen wird, wird die Meldung „`Fehler: Zahl zu` klein" auf der Konsole ausgegeben. Innerhalb der Subroutine hat `Msg$` bei einem solchen Aufruf den Wert „Zahl zu klein" und wird in der PRINT-Anweisung verkettet, um die vollständige Fehlermeldung zu bilden.

Eine Unterroutine kann eine beliebige Anzahl von Argumenten haben, die Float-, Integer- oder String-Werte sein können, wobei jedes Argument durch ein Komma getrennt wird.

Innerhalb der Unterprogramm verhalten sich die Argumente wie normale Variablen, aber sie existieren nur innerhalb der Unterprogramm und verschwinden, wenn die Unterprogramm endet. Sie können Variablen mit dem gleichen Namen im Hauptprogramm haben, die innerhalb der Unterprogramm ausgeblendet werden und sich von den für die Unterprogramm definierten Argumenten unterscheiden.

Der Typ des zu übergebenden Arguments kann mit einem Typ-Suffix angegeben werden (d. h. $, % oder ! für Zeichenfolge, Ganzzahl und Gleitkomma). Im folgenden Beispiel muss das erste Argument eine Zeichenfolge und das zweite eine Ganzzahl sein:

```
SUB MySub Msg$, Nbr%
   … END
SUB
```

MMBasic konvertiert die angegebenen Werte, sofern dies möglich ist. Wenn Ihr Programm also einen Gleitkommawert als zweites Argument angibt, konvertiert MMBasic diesen in eine Ganzzahl. Wenn MMBasic den Wert nicht konvertieren kann, wird eine Fehlermeldung angezeigt und das Programm kehrt zur Eingabeaufforderung zurück. Wenn Sie beispielsweise eine Zeichenkette als zweites Argument angegeben haben, wird Ihr Programm mit einer Fehlermeldung beendet.

Sie müssen die Typ-Suffixe nicht verwenden, sondern können den Typ der Argumente stattdessen mit dem Schlüsselwort AS definieren, ähnlich wie es im Befehl DIM verwendet wird.

Das folgende Beispiel ist beispielsweise identisch mit dem obigen Beispiel:

```
SUB MySub Msg AS STRING, Nbr AS INTEGER
   … END
SUB
```

Wenn Sie im gesamten Programm nur einen Variablentyp verwenden und diesen Typ mit OPTION DEFAULT festlegen, können Sie die Frage nach den Variablentypen natürlich komplett ignorieren.

Wenn eine Subroutine mit einem Argument aufgerufen wird, das eine Variable ist (d. h. keine Konstante oder kein Ausdruck), erstellt MMBasic innerhalb der Subroutine eine entsprechende Variable, *die auf diese Variable zurückverweist.* Jede Änderung an der Variable, die das Argument innerhalb der Subroutine darstellt, ändert auch die im Aufruf verwendete Variable. Dies wird als Übergabe von Argumenten durch Referenz bezeichnet.

Am besten lässt sich dies anhand eines Beispiels erklären:

```
DIM MyNumber = 5           ' setze die Variable auf 5
CalcSquare MyNumber        ' Die Unterroutine quadriert ihren Wert PRINT
MyNumber                   ' Dies gibt die Zahl 25 aus
END

SUB CalcSquare nbr
   nbr = nbr * nbr         ' das Argument quadrieren und zurückgeben END
SUB
```

Die Subroutine CalcSquare nimmt ihr Argument, quadriert es und schreibt es zurück in die Variable, die das Argument repräsentiert (`nbr`). Da die Subroutine mit einer Variablen (`MyNumber`) aufgerufen wurde, verweist die Variable `nbr` zurück auf `MyNumber,` und jede Änderung an `nbr` ändert auch `MyNumber` entsprechend. Als Ergebnis gibt die PRINT-Anweisung 25 aus.

Die Übergabe von Argumenten per Referenz ist praktisch, da sie es einer Unterroutine ermöglicht, Werte an den Code zurückzugeben, der sie aufgerufen hat. Es könnte jedoch zu Problemen führen, wenn eine Unterroutine die Variable, die ein Argument darstellt, als allgemeine Variable verwendet und ihren Wert ändert. Wenn sie dann mit einer Variablen als Argument aufgerufen würde, würde diese Variable unbeabsichtigt geändert werden. Um dies zu vermeiden, sollten Sie ihrer Definition das Schlüsselwort BYVAL voranstellen. Dadurch wird MMBasic angewiesen, immer den Wert des Arguments zu verwenden, auch wenn es sich um eine Variable handelt, und niemals auf die im Aufruf verwendete Variable zurückzuweisen.

Wenn Sie eine Subroutine aufrufen, können Sie einige (oder alle) Parameter weglassen, die dann den Wert Null (für Fließkommazahlen oder Ganzzahlen) oder eine leere Zeichenfolge annehmen. Dies ist praktisch, da Ihre Subroutine erkennen kann, ob ein Parameter fehlt, und entsprechend reagieren kann.

Hier ist beispielsweise unsere Unterroutine zum Generieren einer Fehlermeldung, die jedoch auch ohne Angabe einer Fehlermeldung als Parameter verwendet werden kann:

```
SUB ErrMsg    Msg$ IF
  Msg$ = "" THEN
     PRINT "Fehler erkannt" ELSE
     PRINT "Fehler: " + Msg$
  ENDIF
END SUB
```

Innerhalb einer Subroutine können Sie die meisten Funktionen von MMBasic verwenden, darunter das Aufrufen anderer Subroutinen, IF…THEN-Befehle, FOR…NEXT-Schleifen und so weiter. Eine Sache, die Sie jedoch nicht tun können, ist, mit GOTO aus einer Subroutine zu springen (wenn Sie dies tun, ist das Ergebnis undefiniert und kann Ihnen graue Haare bescheren).

Normalerweise wird die Subroutine beendet, wenn der Befehl END SUB erreicht wird, aber Sie können die Subroutine auch vorzeitig mit dem Befehl EXIT SUB beenden.

Funktionen

Funktionen ähneln Unterprogrammen, mit dem Hauptunterschied, dass eine Funktion dazu dient, einen Wert in einem Ausdruck zurückzugeben. Wenn Sie beispielsweise eine Funktion zur Umrechnung einer Temperatur von Grad Celsius in Fahrenheit wünschen, könnten Sie Folgendes definieren:

```
FUNCTION Fahrenheit(C) Fahrenheit = C
  * 1,8 + 32
END FUNCTION
```

Dann könnten Sie sie in einem Ausdruck verwenden:

```
Input "Geben Sie eine Temperatur in Celsius ein: ", t
PRINT "Das entspricht" Fahrenheit(t) "F"
```

Oder ein weiteres Beispiel:

```
IF Fahrenheit(temp) <= 32 THEN PRINT "Freezing"
```

Sie könnten auch das Gegenteil definieren:

```
FUNKTION Celsius(F)
  Celsius = (F - 32) * 0,5556 END
FUNCTION
```

Wie Sie sehen können, wird der Funktionsname als gewöhnliche lokale Variable innerhalb der Unterroutine verwendet. Erst wenn die Funktion zurückkehrt, wird der Wert für den Ausdruck verfügbar, der sie aufgerufen hat.

Die Regeln für die Argumentliste in einer Funktion ähneln denen für Unterprogramme. Der einzige Unterschied besteht darin, dass beim Aufruf einer Funktion immer Klammern um die Argumentliste gesetzt werden müssen, auch wenn keine Argumente vorhanden sind (beim Aufruf eines Unterprogramms sind Klammern optional).

Um einen Wert aus der Funktion zurückzugeben, weisen Sie dem Namen der Funktion innerhalb der Funktion einen Wert zu. Wenn der Name der Funktion mit einem Typ-Suffix endet (d. h. $, % oder !), gibt die Funktion diesen Typ (Zeichenkette, Ganzzahl oder Gleitkommazahl) zurück, andernfalls gibt sie den Wert zurück, auf den OPTION DEFAULT gesetzt ist. Die folgende Funktion gibt beispielsweise eine Zeichenkette zurück:

```
FUNCTION LVal$(nbr)
  IF nbr = 0 THEN LVal$ = "False" ELSE LVal$ = "True" END FUNCTION
```

Sie können den Typ der Funktion explizit mit dem Schlüsselwort AS angeben, dann müssen Sie kein Typ-Suffix verwenden (ähnlich wie bei der Definition einer Variablen mit DIM).

Das obige Beispiel wurde unter Verwendung dieser Funktion umgeschrieben:

```
FUNCTION LVal(nbr) AS STRING
  IF nbr = 0 THEN LVal = "False" ELSE LVal = "True" END
FUNCTION
```

In diesem Fall ist der von der Funktion `LVal` zurückgegebene Typ eine Zeichenfolge.

Was Unterprogramme betrifft, so können Sie die meisten Funktionen von MMBasic innerhalb von Funktionen verwenden. Dazu gehören FOR…NEXT-Schleifen, der Aufruf anderer Funktionen und Unterprogramme usw. Außerdem kehrt die Funktion zu dem Ausdruck zurück, der sie aufgerufen hat, wenn der Befehl END FUNCTION erreicht wird, aber Sie können auch vorzeitig zurückkehren, indem Sie den Befehl EXIT FUNCTION verwenden.

Lokale Variablen der Funktion

Variablen, die mit DIM erstellt oder einfach automatisch erstellt werden, werden als *globale* Variablen bezeichnet. Das bedeutet, dass sie überall im Programm, einschließlich in Unterprogrammen und Funktionen, sichtbar sind und verwendet werden können. Innerhalb eines Unterprogramms oder einer Funktion müssen Sie jedoch häufig Variablen für verschiedene Aufgaben verwenden, die innerhalb des Unterprogramms/der Funktion intern sind. In portierbarem Code möchten Sie vermeiden, dass der Name, den Sie für eine solche Variable gewählt haben, mit einer globalen Variable gleichen Namens kollidiert. Zu diesem Zweck können Sie eine Variable mit dem Befehl LOCAL innerhalb der Unterroutine/Funktion definieren.

Die Syntax für LOCAL ist identisch mit dem DIM-Befehl, das heißt, die Variable kann ein Array sein, Sie können den Typ der Variable festlegen und sie mit einem beliebigen Wert initialisieren.

Dies ist beispielsweise unsere ErrMsg-Subroutine, die jedoch diesmal erweitert wurde, um eine lokale Variable zum Verknüpfen der Fehlermeldungszeichenfolgen zu verwenden.

```
SUB ErrMsg Msg$ LOCAL
  STRING tstr
  tstr = "Fehler: " + Msg$ PRINT
  tstr
END SUB
```

Die Variable `tstr` wird innerhalb der Unterroutine als LOCAL deklariert, was bedeutet, dass sie (wie die Argumentliste) nur innerhalb der Unterroutine existiert und beim Beenden der Unterroutine verschwindet. Sie können in Ihrem Hauptprogramm eine globale Variable namens `tstr` haben, die sich von der Variable `tstr` in der Unterroutine unterscheidet (in diesem Fall wird die globale `tstr` innerhalb der Unterroutine ausgeblendet).

Sie sollten für Operationen innerhalb Ihrer Unterprogramm oder Funktion immer lokale Variablen verwenden, da diese dazu beitragen, das Modul wesentlich eigenständiger und portabler zu machen.

Statische Variablen „ “

LOKALE Variablen werden bei jedem Start der Unterroutine oder Funktion auf ihre Anfangswerte (normalerweise Null oder eine leere Zeichenfolge) zurückgesetzt. Es gibt jedoch Fälle, in denen Sie möchten, dass die Variable ihren Wert zwischen den Aufrufen beibehält. Dieser Variablentyp wird mit dem Befehl STATIC definiert.

Wir können die Nützlichkeit von STATIC-Variablen demonstrieren, indem wir die Subroutine ErrMsg erweitern, um doppelte Aufrufe der Subroutine zu verhindern, die wiederholt dieselbe Meldung anzeigen. Unser Programm könnte diese Subroutine beispielsweise an mehreren Stellen aufrufen, aber wenn die Meldung in mehreren aufeinanderfolgenden Aufrufen identisch ist, möchten wir die Meldung nur einmal sehen.

Dies ist unsere neue Subroutine:

```
SUB ErrMsg Msg$
   STATIC STRING lastmsg
   LOCAL STRING tstr
   IF Msg$ <> lastmsg THEN tstr =
      "Fehler: " + Msg$ PRINT tstr
      lastmsg = Msg$
   ENDIF
END SUB
```

Um die zuletzt angezeigte Meldung zu verfolgen, verwenden wir eine statische Variable namens `lastmsg`. Diese enthält den Text der letzten Meldung, den wir mit dem aktuellen Meldungstext vergleichen können, um festzustellen, ob er sich unterscheidet und daher ausgegeben werden sollte. Auf diese Weise würde bei jedem Aufruf mit einem doppelten Meldungstext nur eine Meldung ausgegeben werden.

Der Befehl STATIC verwendet genau dieselbe Syntax wie DIM. Das bedeutet, dass Sie verschiedene Arten von statischen Variablen einschließlich Arrays definieren und diese auch mit einem bestimmten Wert initialisieren können.

Die statische Variable wird beim ersten Auftreten des Befehls STATIC erstellt und automatisch auf Null (bei Float- oder Integer-Typen) oder eine leere Zeichenfolge gesetzt. Bei nachfolgenden Aufrufen der Subroutine oder Funktion erkennt MMBasic, dass die Variable bereits erstellt wurde, und lässt ihren Wert unverändert (d. h. so, wie er beim vorherigen Aufruf war). Wie bei DIM können Sie auch eine statische Variable mit einem bestimmten Wert initialisieren. Beispiel:

```
STATIC INTEGER var = 123
```

Beim ersten Aufruf (wenn die Variable erstellt wird) wird sie auf 123 initialisiert, bei nachfolgenden Aufrufen behält sie jedoch den zuvor festgelegten Wert bei.

Statische Variablen werden meist verwendet, um den *Status* von etwas innerhalb einer Subroutine oder Funktion zu verfolgen. Ein *Status* ist eine Aufzeichnung von etwas, das zuvor geschehen ist.

Beispiele hierfür sind:

- Wurde der COM-Port bereits geöffnet?
- Welche Schritte einer Sequenz haben wir bereits abgeschlossen?
- Welcher Text wurde bereits angezeigt?

Normalerweise verwenden Sie globale Variablen (erstellt mit DIM), um einen *Status* zu verfolgen, aber manchmal möchten Sie, dass dieser in einem Modul enthalten ist, und hier sind statische Variablen von Vorteil. Genau wie LOCAL trägt die Verwendung von STATIC dazu bei, Ihre Unterprogramme und Funktionen eigenständiger und portabler zu machen.

Berechnen Sie die Tage v

Wir haben in diesem Tutorial bisher viele Programmierbefehle und -techniken behandelt, und bevor wir es beenden, wäre es sinnvoll, ein Beispiel dafür zu geben, wie sie zusammenwirken. Das folgende Beispiel nutzt viele Funktionen der Sprache BASIC, um die Anzahl der Tage zwischen zwei Daten zu berechnen:

```
' Beispielprogramm zur Berechnung der Anzahl der Tage zwischen zwei Daten

OPTION EXPLICIT OPTION
DEFAULT NONE

DIM STRING s DIM
FLOAT d1, d2
```

```
DO
   ' Hauptprogrammschleife
   PRINT : PRINT „Geben Sie das Datum im Format TT.MMM.JJJJ
   ein" PRINT „Erstes Datum";
   INPUT s
   d1 = GetDays(s)
   IF d1 = 0 THEN PRINT "Ungültiges Datum!" : CONTINUE DO
   PRINT "Zweites Datum";
   INPUT s
   d2 = GetDays(s)
   WENN d2 = 0 DANN DRUCKEN „Ungültiges Datum!“ : WEITER
   PRINT "Die Differenz beträgt" ABS(d2 - d1) " Tage"
LOOP

' Berechne die Anzahl der Tage seit dem 1.1.1900 FUNCTION
GetDays(d$) AS FLOAT
   LOCAL STRING Monat(11) =
("jan","feb","mar","apr","may","jun","jul","aug","sep","okt","nov","dez") LOCAL FLOAT
   Tage(11) = (0,31,59,90,120,151,181,212,243,273,304,334)
   LOCAL FLOAT Tag, Monat, Jahr, s1, s2

   ' Trennzeichen innerhalb eines Datums suchen s1 = INSTR(d$, "
   ")
   IF s1 = 0 THEN EXIT FUNCTION s2 =
   INSTR(s1 + 1, d$, " ")
   WENN s2 = 0 DANN FUNKTION BEENDEN

   ' Tag, Monat und Jahr als Zahlen abrufen day =
   VAL(MID$(d$, 1, s2 - 1)) - 1
   WENN Tag < 0 ODER Tag > 30 DANN FUNKTION BEENDEN
   FÜR Monat = 0 BIS 11
      WENN LCASE$(MID$(d$, s1 + 1, 3)) = Monat(mth) DANN BEENDE WÄHREND
   NEXT mth
   WENN Monat > 11 DANN FUNKTION BEENDEN
   Jahr = VAL(MID$(d$, s2 + 1)) - 1900
   WENN Jahr < 1 ODER Jahr >= 200 DANN FUNKTION BEENDEN

   Berechnen Sie die Anzahl der Tage einschließlich der Anpassung für Schaltjahre
   GetDays = (Jahr * 365) + FIX((Jahr - 1) / 4)
   IF Jahr MOD 4 = 0 AND Monat >= 2 THEN GetDays = GetDays + 1 GetDays =
   GetDays + Tage(Monat) + Tag
END FUNCTION
```

Beachten Sie, dass die Zeile, die mit `LOCAL STRING Month(11)` beginnt, aufgrund der begrenzten Seitenbreite umgebrochen wurde – sie besteht aus einer einzigen Zeile wie folgt:

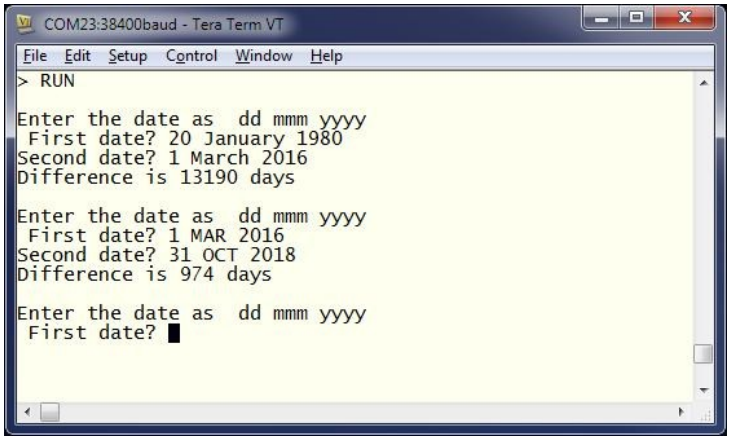
```
LOCAL STRING Month(11) = ("jan","feb","mar","apr","may","jun","jul","aug","sep","oct","nov","dec")
```

Dieses Programm fragt den Benutzer an der Konsole nach zwei Datumsangaben und wandelt diese dann in die Anzahl der Tage seit 1900 um. Mit diesen beiden Zahlen ergibt eine einfache Subtraktion die Anzahl der Tage zwischen ihnen.

Wenn dieses Programm ausgeführt wird, werden Sie aufgefordert, die beiden Daten einzugeben. Verwenden Sie dazu das Format: `dd mmm yyyy`.

Der Screenshot auf der rechten Seite zeigt, wie das laufende Programm aussieht.

Das Hauptmerkmal des Programms ist die definierte Funktion `GetDays()`, die eine in der Konsole eingegebene Zeichenfolge nimmt, sie in ihre Bestandteile Tag, Monat und Jahr aufteilt und dann die Anzahl der Tage seit dem 1. Januar 1900 berechnet.



Diese Funktion wird zweimal aufgerufen, einmal für das erste Datum und dann erneut für das zweite Datum. Anschließend muss nur noch das eine Datum (in Tagen) vom anderen abgezogen werden, um die Differenz in Tagen zu erhalten.

Wir werden nicht im Detail darauf eingehen, wie die Berechnungen durchgeführt werden (z. B. die Behandlung von Schaltjahren), da dies als Übung für den Leser dienen kann. Es ist jedoch angebracht, auf einige Funktionen von MMBasic hinzuweisen, die vom Programm verwendet werden.

Es zeigt, wie lokale Variablen verwendet und initialisiert werden können. In der Funktion `GetDays()` werden zwei Arrays gleichzeitig deklariert und initialisiert. Diese sind lediglich eine praktische Methode, um die Namen der Monate und die kumulierte Anzahl der Tage für jeden Monat nachzuschlagen. Später in der Funktion (in der FOR-Schleife) können Sie sehen, wie sie den Umgang mit zwölf verschiedenen Monaten recht effizient gestalten.

Eine weitere Besonderheit dieses Programms sind die Zeichenfolgenverarbeitungsfunktionen von MMBasic. Mit der Funktion INSTR() werden die beiden Leerzeichen in der Datumszeichenfolge gesucht, und anschließend werden mit der Funktion MID$() daraus die Komponenten Tag, Monat und Jahr des Datums extrahiert. Mit der Funktion VAL() wird eine Zeichenfolge aus Ziffern (wie das Jahr) in eine Zahl umgewandelt, die in einer numerischen Variablen gespeichert werden kann.

Beachten Sie, dass der Wert einer Funktion bei jedem Aufruf der Funktion auf Null initialisiert wird und dass sie einen Wert von Null zurückgibt, sofern ihr kein Wert zugewiesen wurde. Dies erleichtert die Fehlerbehandlung, da wir die Funktion einfach beenden können, wenn ein Fehler entdeckt wird. Es liegt dann in der Verantwortung des aufrufenden Programmcodes, auf einen Rückgabewert von Null zu prüfen, der einen Fehler anzeigt.

Dieses Programm veranschaulicht einen der Vorteile der Verwendung von Unterprogrammen und Funktionen: Wenn sie geschrieben und vollständig getestet sind, können sie als vertrauenswürdige „Black Box" behandelt werden, die nicht geöffnet werden muss. Aus diesem Grund sollten Funktionen wie diese ordnungsgemäß getestet und dann, wenn möglich, unverändert gelassen werden (für den Fall, dass Sie einen Fehler hinzufügen).

Es gibt einige Funktionen dieses Programms, die wir bisher noch nicht behandelt haben. Die erste ist der MOD-Operator, der den Rest der Division einer Zahl durch eine andere berechnet. Wenn Sie beispielsweise 4 durch 15 dividieren, erhalten Sie einen Rest von 3, was genau dem Ergebnis des Ausdrucks 15 MOD 4 entspricht. Die Funktion ABS() ist ebenfalls neu und gibt ihr Argument als positive Zahl zurück (z. B. gibt ABS(-15) +15 zurück, ebenso wie ABS(15)).

Der Befehl EXIT FOR beendet eine FOR-Schleife, auch wenn sie noch nicht das Ende erreicht hat, EXIT FUNCTION beendet eine Funktion sofort, auch wenn die Ausführung noch nicht das Ende der Funktion erreicht hat, und CONTINUE DO bewirkt, dass das Programm sofort zum Ende einer DO-Schleife springt und diese erneut ausführt.

Warum sollte dieses Programm nützlich sein? Nun, manche Menschen zählen ihr Alter gerne in Tagen, denn so ist jeder Tag ein Geburtstag! Sie können Ihr Alter in Tagen berechnen, indem Sie einfach Ihr Geburtsdatum und das heutige Datum eingeben. Das ist zwar nicht besonders nützlich, aber das Programm selbst ist wertvoll, da es viele der Eigenschaften der Programmierung in MMBasic demonstriert. Arbeiten Sie sich also durch das Programm und gehen Sie jeden Abschnitt durch, bis Sie ihn verstanden haben – es sollte eine lohnende Erfahrung sein.