

Stefan's BASIC

- cc2tv Folge 363
- Stefan's IoT BASIC
- Möglichkeiten
- Rechner-System
- Arbeitsumgebung
- Was wird benötigt?
- Installation
- Konfiguration
- Installation BASICFULL
- Installation ESPSPIFFS
- Installation ARDUINOLCDI2C
- Terminal
- Filesystem SPIFFS
- BASIC Sprachen
- Interaktiv vs. Programm
- ESP32 Pinout
- I2C Pins
- GPIO Hinweise
- GPIO ESP32
- Interaktiv I/O Funktionen
- Interaktiv I/O Funktionen
- Digitaler I/O LED
- Digitaler I/O Button
- Informationsquellen
- Titel
- Anhang
- Links

cc2tv Folge 363

Die Idee:

BASIC programmieren wie früher von Thomas Rudolph.



Quelle: <https://cc2.tv/>

Dateien: <https://github.com/EKlatt/Experiences/tinybasic>

Stefan's IoT BASIC

Die Idee	Basic-Interpreter für µC. Inspiriert von Steve Wozniak, der den Apple 1 Basic Interpreter als größte Herausforderung bezeichnete.
	Erweitert um IoT-Funktionen. Einsetzbar auf AVR, ESP8266, ESP32, SAMD, RP2040 and ARM.
	Es werden 16bit und 32bit Ganzzahlen unterstützt. Je nach den Compiler-Einstellungen auch Fließkommazahlen.
Projektseite	https://github.com/slviajero/tinybasic
Versionen	„Basic1“ und „Basic2“. Hier habe ich „Basic1“ installiert.
Basic-Versionen	<ul style="list-style-type: none">• Palo Alto BASIC language set.• Apple Integer BASIC.• Stefan's extension.• Dartmouth language set.• Weitere Ergänzungen...

Möglichkeiten

Basic-Interpreter	<ul style="list-style-type: none">• Basic-Interpreters mit eingebetteten Betriebssystem.• Direkter Zugriff auf die Hardware des Rechners.
IoT-Basic	<ul style="list-style-type: none">• IoT-Basic mit Geräte-Treibern, um einen eigenständigen Rechner aufzubauen.
Filesystem	<ul style="list-style-type: none">• Filesystem auf μC.• Speichern auf SD-Karten möglich.
I ² C-LCD	<ul style="list-style-type: none">• Bestimmte Displays können angesprochen werden.
	<ul style="list-style-type: none">• Ethernet und Wifi werden unterstützt, um z.B. MQTT möglich zu machen.
Sensoren	<ul style="list-style-type: none">• Bestimmte Sensoren sind einsetzbar.

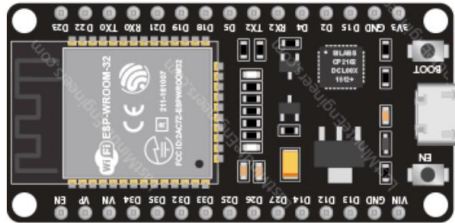
Rechner-System

μC ESP32

- Mit Hilfe der Arduino-IDE installierter
- Basic-Interpreter (*.ino Datei)

PC/RPI/MAC mit Monitor

- Arduino-IDE zum konfigurieren und hochladen des Basic-Interpreters.
- Terminal-Software

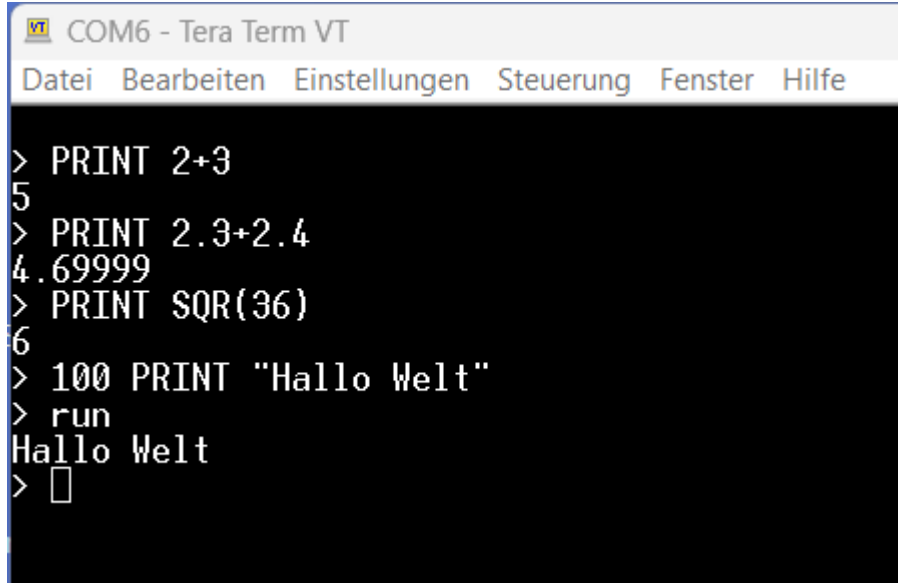


Kommunikation über USB-Kabel.

Arbeitsumgebung

Aufbau & Funktion

1. μ C mit Basic-Interpreter
2. Host (PC/RPI/MAC) mit Terminal (Monitor)-Software
3. Host (PC/RPI/MAC) über USB mit μ C verbunden
4. Steuerung des μ C über das Terminal.
d.h. Eingabe der Befehle und evtl. des Programms.



```
COM6 - Tera Term VT
Datei Bearbeiten Einstellungen Steuerung Fenster Hilfe

> PRINT 2+3
5
> PRINT 2.3+2.4
4.69999
> PRINT SQR(36)
6
> 100 PRINT "Hallo Welt"
> run
Hallo Welt
> 
```

Was wird benötigt?

Mikrokontroller	Arduino UNO (minimalistischer Einstieg)
Empfohlener μ C	ESP32 NodeMCU Module WLAN WiFi (CP2102) https://www.az-delivery.de/products/esp32-developmentboard
Basic Interpreter	Stefan's IoT BASIC (von Nuklearphysiker Stefan Lenz) https://github.com/slviajero/tinybasic/tree/main
Arduino IDE 1.8.19	https://www.arduino.cc/en/software
Terminal-Emulator Windows 11	https://filehippo.de/download_tera-term/

Installation

Arduino IDE	<ul style="list-style-type: none">• Herunterladen und installieren.• Für die ESP32-µC einen zusätzlichen Boardverwalter unter Voreinstellungen eintragen: https://dl.espressif.com/dl/package_esp32_index.json• Über Werkzeuge & Boardverwalter den „ESP32“ installieren.
LCD 16x2 library	Library für eine I ² C-LCD unter „libraries“ installieren. https://raw.githubusercontent.com/DFRobot/WikiResource/master/DFR0063/LiquidCrystal_I2C.zip
lotBasic	Die heruntergeladene Zip-Datei „tinybasic-main.zip“ entpacken und den Ordner „tinybasic-main“ an einen geeigneten Ort verschieben.
Arbeitskopie	Innerhalb „tinybasic-main“ im Ordner „Basic1“ vom Ordner „lotBasic“ eine Kopie erstellen, und diese z.B. in „IoTBasic_ESP32“ umbenennen.
	Die „IoTBasic.ino“ ebenfalls umbenennen in „IoTBasic_ESP32.ino“.

Konfiguration

BASICSIMPLE
(IotBasic_ESP32.ino)

Im Ausgangszustand ist ein minimalistischer Basic-Interpreter voreingestellt.

Man muss bedenken, dass alle Fähigkeiten des Interpreters in der hoch zu ladenden Binärdatei enthalten sein müssen.

Laufzeit

Während der Laufzeit (runtime) kann nichts mehr dazu kompiliert werden!

Preprocessor
directives

Der Basic-Interpreter ist über „Preprocessor directives“ konfigurierbar.
Es handelt sich hier um Anweisungen für den Compiler, den Code entsprechend der Directives zu kompilieren.

Basic-Interpreter

Will man ein Filesystem, ein Display, das Netzwerk und Sensoren nutzen, so müssen meist „Preprocessor directives“ angepasst werden.

Beispiel

```
#define BASICFULL
```

IDE-libraries

Will man z.B. ein LCD-Display nutzen, so muss die **Arduino-Library** ([LiquidCrystal I2C.zip](#)) unter „libraries“ vorab installiert sein.

Installation BASICFULL

(Arduino UNO)	(Wurde von mir in der Standardeinstellung „BASICSIMPLE“ getestet.) Der Arduino UNO kann nur bei minimalen Anforderungen sinnvoll genutzt werden.
ESP32	Hier kann, wegen des größeren Speichers eine leistungsfähigere Interpreter-Variante aktiviert werden.
Preprocessor directive	<code>#define BASICFULL</code>
Arduino IDE	In der Arduino-IDE im Ordner „IoTBasic_ESP32“ den Skript „IoTBasic_ESP32.ino“ öffnen.
	Hier in Zeile 49 ändern: bisher: <code>#undef BASICFULL</code> in: <code>#define BASICFULL</code> und in Zeile 51 ändern. bisher: <code>#define BASICSIMPLE</code> in: <code>#undef BASICSIMPLE</code>



Installation ESPSPIFFS

ESP32	<p>Auf dem μC (hier ESP 32) kann ein „Laufwerk“, zum Speichern von Programmen und Daten, eingerichtet werden.</p> <p>Hier können mit den Befehlen „LOAD“ und SAVE die Basic-Programme gespeichert werden. Der Speicher ist permanent.</p>
Preprocessor directive	<pre>#define ESPSPIFFS</pre>
Arduino IDE	<p>In der Arduino-IDE im Ordner „IoTBasic_ESP32“ den Skript „IoTBasic_ESP32.ino“ öffnen.</p>
Arduino IDE	<p>Den Reiter „hardware-arduino.h“ wählen.</p> <p>Hier in Zeile 88 ändern: bisher: <code>#undef ESPSPIFFS</code> in: <code>#define ESPSPIFFS</code></p>
Skript hochladen	<p>Den ESP32-Skript, wie üblich, auf den ESP32 hochladen.</p>

Installation ARDUINOLCDI2C

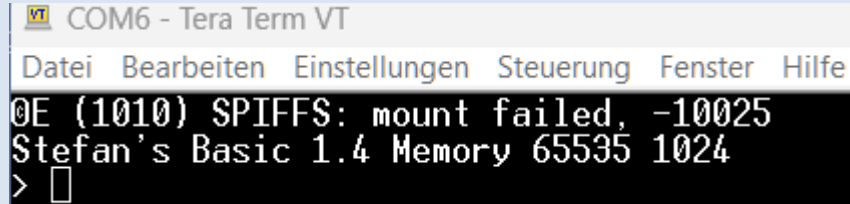
ESP32	Aktivieren der Ausgabe auf I ² C-LEDs. Hier können mit den Befehlen der Terminal-Emulation und VT52-Befehlen Ausgaben auf einem LCD erzeugt werden.
Preprocessor directive	<code>#define ARDUINOLCDI2C</code>
Arduino IDE	In der Arduino-IDE im Ordner „IoTBasic_ESP32“ den Skript „IoTBasic_ESP32.ino“ öffnen.
Arduino IDE	Den Reiter „hardware-arduino.h“ öffnen. Hier in Zeile 74 ändern: bisher: <code>#undef ARDUINOLCDI2C</code> in: <code>#define ARDUINOLCDI2C</code>
Skript hochladen	Den ESP32-Skript, wie üblich, auf den ESP32 hochladen.
IDE-Monitor	Den Arduino-IDE Monitor öffnen und auf 9600 Baud einstellen. Wird die Taste „Enter“ gedrückt, sollte der Interpreter-Prompt „>“ erscheinen.

Terminal

	Der Basic-Interpreter auf dem μ C wird über einen Host (PC) gesteuert.	
Arduino Monitor	Als erster Ansatz kann der Arduino-Monitor zum herstellen und testen einer Verbindung zum μ C genutzt werden.	
Alternative:		
Terminal „Tera Term“	Herunterladen von „Tera Term für Windows“.	
	„teraterm_v4.106.exe“ installieren.	
Tera Term konfigurieren	> Einstellungen > Terminal-Einstellungen	
Serieller Port	Seriellen Port ermitteln (z.B. Arduino-IDE) > Einstellungen > Serieller Port	
	> Einstellungen > Setup sichern	

Filesystem SPIFFS

Fehler

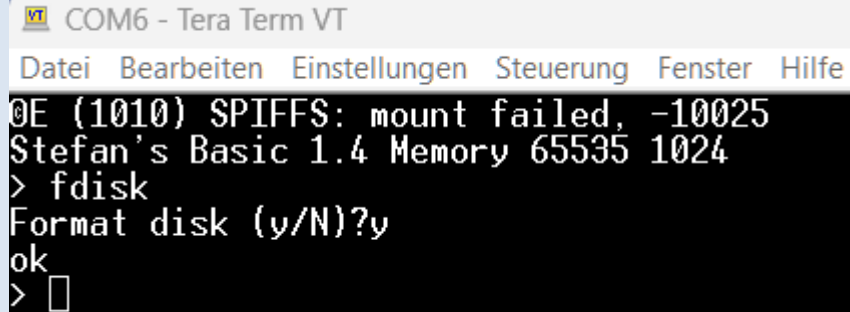


VT COM6 - Tera Term VT
Datei Bearbeiten Einstellungen Steuerung Fenster Hilfe
OE (1010) SPIFFS: mount failed, -10025
Stefan's Basic 1.4 Memory 65535 1024
>

Ursache

Laufwerk noch nicht formatiert.

Abhilfe



VT COM6 - Tera Term VT
Datei Bearbeiten Einstellungen Steuerung Fenster Hilfe
OE (1010) SPIFFS: mount failed, -10025
Stefan's Basic 1.4 Memory 65535 1024
> fdisk
Format disk (y/N)?y
ok
>

Mögliche Befehle
File I/O

catalog, save, load, delete

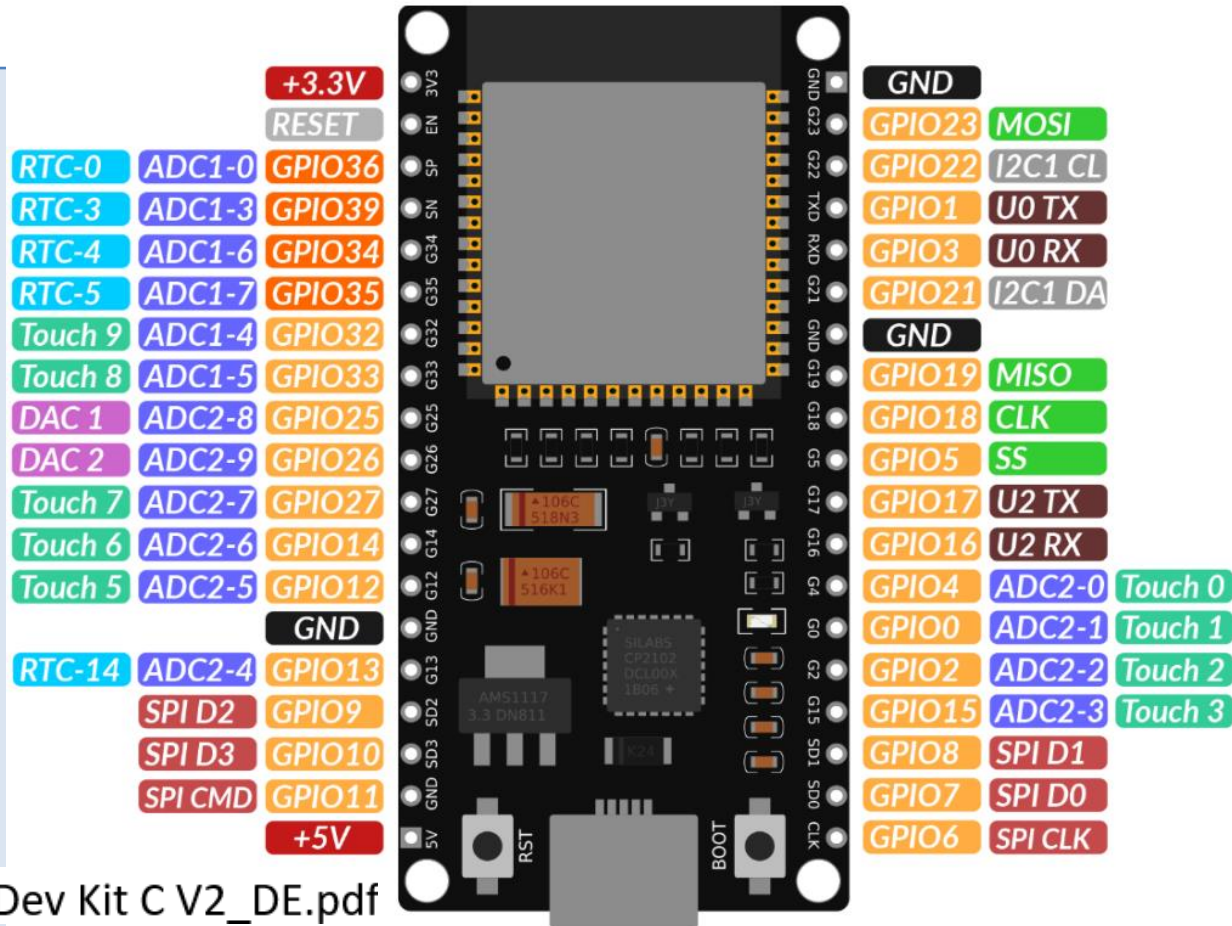
BASIC Sprachen

Palo Alto BASIC Language set	PRINT , LET , INPUT , GOTO , GOSUB , RETURN , IF , FOR , TO , NEXT , STOP , LIST , NEW , RUN , ABS , RND , SIZE , REM , 26 static Variables A-Z, @()
Apple Integer BASIC add ons	NOT , AND , OR , LEN , SGN , PEEK , DIM , CLR , HIMEM , TAB , THEN , POKE , dynamic variables A-Z[0-9,A-Z], strings A-Z[0-9,A-Z], arrays A-Z[0-9,A-Z]
Stefan's add ons	ELSE , SQR , MAP , DUMP , SAVE , LOAD , GET , PUT , SET , POW , special variables @D() , @X , @Y for display access, @T\$ and @T() for real time clock access. CONT to continue loops, BREAK
I/O add on	AWRITE , AREAD , DWRITE , DREAD , PINM , DELAY , MILLIS , PUSLE , PLAY , MAP
Floating point add ons	SIN , COS , TAN , ATAN , EXP , LOG
DARTMOUTH BASIC add ons	DATA , READ , RESTORE , ON .. GOTO/GOSUB , DEF FN A-Z[0-9, A-Z]
Graphics for TFT displays	COLOR , PLOT , LINE , RECT , FRECT , CIRCLE , FCIRCLE
File system commands	OPEN , CLOSE , DELETE , CATALOG , FDISK

Interaktiv vs. Programm

	Bei der Eingabe muss nicht auf Groß- und Kleinschreibung geachtet werden.
Prompt	„>“
	In diesem Fall hat der μ C (der Interpreter) eine Eingabeaufforderung an das Terminal gesendet. Eingaben mit „Enter“ abschließen.
Eingaben	<pre>> PRINT 2+3 5 > </pre>
Programm	<pre>> 100 REM "EBW" > 200 PRINT 2.3+3.7 > 300 END > RUN 6 > </pre>
autoexec.bas	Hat ein System ein Laufwerk (ESPSPIFFS), dann wird ein Programm mit dem Namen „autoexec.bas“ beim Systemstart sofort ausgeführt.

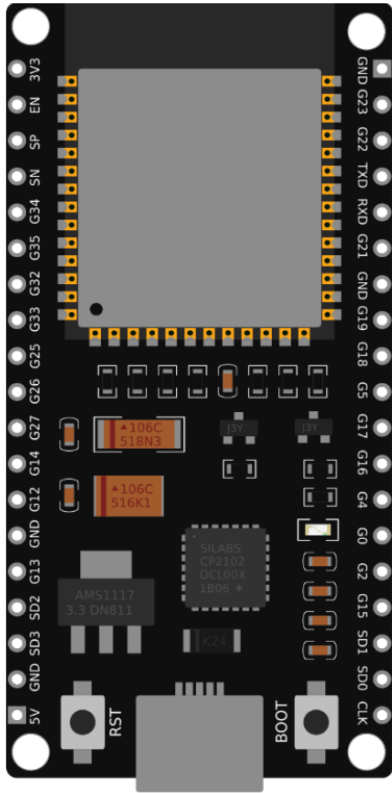
ESP32 Pinout



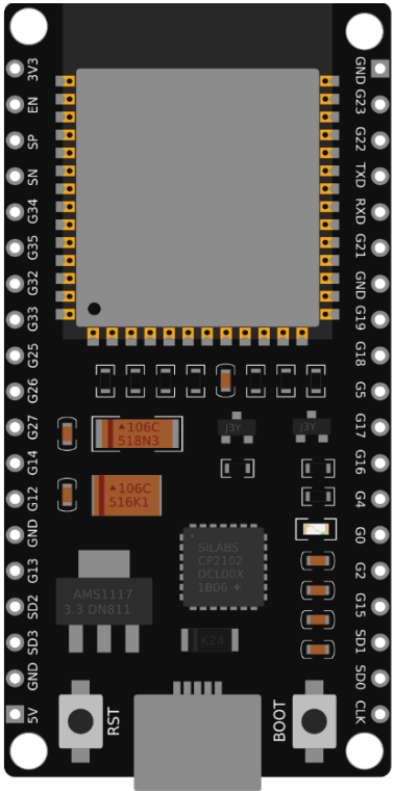
Quelle: ESP32 Dev Kit C V2_DE.pdf

I2C Pins

I2C



SPI Pins



GPIO Hinweise

Die digitalen Eingänge/Ausgänge arbeiten mit 3,3V.

Es darf keine 5V Spannung an die ESP32 Chip Pins angeschlossen werden!

Die GPIO Pins 34 bis 39 sind GPIOs - nur Eingangspins.

Der absolute Maximalstrom, der pro GPIO gezogen werden darf, beträgt **10 mA**.

GPIO-Pins maximal 50 mA

GPIO2 funktioniert nicht als interne LED!

ESP32 Dev KitC V2 Betriebsanleitung.pdf

GPIO ESP32

25 GPIO-Pins können, je nach Initialisierung, unterschiedliche Funktionen haben:

15 ADC channels

2 UART interfaces

25 PWM outputs

2 DAC channels

SPI, I2C and I2S interface

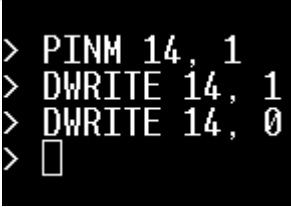
9 Touch Pads

Folgende Pins können für GPIO IN & OUT (sicher) genutzt werden:

4, 13, 14, 16, 17, (18), (19), (21), (22), (23), 25, 26, 27, 32, 33

Quelle: <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>

Interaktiv I/O Funktionen

	Bei der Eingabe muss nicht auf Groß- und Kleinschreibung geachtet werden.	
IoT Erweiterungen	PINM,DWRITE, DREAD, (LED)	
Eingaben		
Pin 14 auf output	PINM 14, 1	Pin-Modus
Pin 14 auf HIGH	DWRITE 14, 1	Digitales Schreiben auf HIGH
Pin 14 auf LOW	DWRITE 14, 0	Digitales Schreiben auf LOW

Interaktiv I/O Funktionen

	Bei der Eingabe muss nicht auf Groß- und Kleinschreibung geachtet werden.	
IoT Erweiterungen	AWRITE, AREAD, AZERO AZERO entspricht dem analogen Pin "ADC1-7"	
Eingaben	<div><pre>> PRINT AREAD(35) 98 > > PRINT AREAD(35) 4095 > □</pre></div> <div><pre>> V=AREAD(AZERO) > PRINT V 4095 > □</pre></div>	
Analog-Pin	GPIO35 entspricht AZERO (ADC1-7)	
Pin 35 lesen	AREAD(35)	Von GPIO53 lesen
Ausgabe	PRINT AREAD(35)	
AZERO lesen	V=AREAD(AZERO)	Von AZERO lesen
Ausgabe	PRINT V	

Digitaler I/O LED

ESP32	Die Board-Bezeichnungen stimmen mit den GPIO-Bezeichnungen überein!
-------	---

LED an GPIO14
blinken

```
> 100 REM "Blink from Arduino examples"
> 110 REM "setup() put your setup code here, to run once:"
> 120 L=14
> 130 PINM L ,1
> 200 REM "loop() put your main code here, to run repeatedly:"
> 210 PRINT "Type 'Q' to exit!"
> 220 FOR I
> 230 GET Q : IF Q="q" THEN END
> 240 DWRITE L, 1
> 250 DELAY 1000
> 260 DWRITE L, 0
> 270 DELAY 1000
> 300 NEXT
> □
```

Zeilennummern	Achtung: Müssen aufsteigend sein.
---------------	-----------------------------------

FOR I NEXT	Erzeugt eine Endlos-Schleife.
-----------------	-------------------------------

PINM GPIO, [0,1]	Analog zu Arduino-C++:	pinMode(pin,[INPUT,OUTPUT])
------------------	------------------------	-----------------------------

DWRITE GPIO, [1,0]	Analog zu Arduino-C++:	digitalWrite(pin, [HIGH,LOW])
--------------------	------------------------	-------------------------------

Digitaler I/O Button

Neues Programm

„> new“; weil das bisherige Programm noch vorhanden ist!

Button an GPIO13
schaltet
LED an GPIO14

```
new
> 100 REM "Button from Arduino examples"
> 110 REM "setup() put your setup code here, to run once:"
> 120 B=13
> 130 L=14
> 140 S=0
> 150 PINM L, 1
> 160 PINM B, 0
> 200 REM "loop() put your main code here, to run repeatedly:"
> 210 PRINT "Type 'Q' to exit!"
> 220 FOR I
> 230 GET Q : IF Q="q" THEN END
> 240 S=DREAD(B)
> 250 IF S=1 THEN DWRITE L, 1 ELSE DWRITE L, 0
> 260 NEXT
```

DREAD(GPIO)

Analog zu Arduino-C++:

digitalRead(pin,)

Informationsquellen

Stefan's IoT BASIC in a nutshell	https://github.com/slviajero/tinybasic/blob/main/MANUAL.md
Language features	Language features of Stefan's BASIC as compared to Dr. Wang's Palo Alto BASIC, Apple 1 BASIC, and Dartmouth BASIC · slviajero/tinybasic Wiki · GitHub
Display	Peripherals: Display LCDs, TFTs, and VGA · slviajero/tinybasic Wiki · GitHub
Sensors	Peripherals: Sensors · slviajero/tinybasic Wiki · GitHub
Apple 1 Basic	The original Apple 1 BASIC manual · slviajero/tinybasic Wiki · GitHub

Titel

Titel	
-------	--

Code	
------	--

Anhang

Links
