

Das grosse Buch für ESP32forth

version 1.1 21. Okt. 2023



Autor

- Marc PETREMANN petremann@arduino-forth.com

Mitarbeiter

- Vaclav POSSELT
- Thomas SCHREIN

Inhalt

Autor.....	1
Mitarbeiter.....	1
Einführung.....	3
Übersetzungshilfe.....	3
Entdeckung der ESP32-Karte.....	4
Präsentation.....	4
Die Stärken Punkten.....	4
GPIO-Ein-/Ausgänge auf ESP32.....	5
ESP32-Board-Peripheriegeräte.....	7
Warum auf ESP32 in FORTH-Sprache programmieren?.....	8
Präambel.....	8
Grenzen zwischen Sprache und Anwendung.....	9
Was ist ein FORTH-Wort?.....	9
Ein Wort ist eine Funktion?.....	9
FORTH-Sprache im Vergleich zur C-Sprache.....	10
Was FORTH Ihnen im Vergleich zur C-Sprache ermöglicht.....	11
Aber warum ein Stapel statt Variablen?.....	12
Sind Sie überzeugt?.....	12
Gibt es professionelle Bewerbungen, die in FORTH verfasst sind?.....	13
Ein echtes 32-Bit FORTH mit ESP32Forth.....	15
Werte auf dem Datenstapel.....	15
Werte im Gedächtnis.....	15
Textverarbeitung je nach Datengröße oder -typ.....	16
Abschluss.....	17
Installieren der OLED-Bibliothek für SSD1306.....	19
Ressourcen.....	21
Auf Englisch.....	21
Auf Französisch.....	21
GitHub.....	21

Einführung

Seit 2019 verwalte ich mehrere Websites, die sich der FORTH-Sprachentwicklung für ARDUINO- und ESP32-Karten sowie der eForth-Webversion widmen. :

- ARDUINO : <https://arduino-forth.com/>
- ESP32 : <https://esp32.arduino-forth.com/>
- eForth web : <https://eforth.arduino-forth.com/>

Diese Websites sind in zwei Sprachen verfügbar: Französisch und Englisch. Jedes Jahr bezahle ich für das Hosting der Hauptseite arduino-forth.com.

Es wird früher oder später – und zwar so spät wie möglich – passieren, dass ich die Nachhaltigkeit dieser Seiten nicht mehr gewährleisten kann. Die Folge wird sein, dass die von diesen Websites verbreiteten Informationen verschwinden.

Dieses Buch ist die Zusammenstellung von Inhalten meiner Websites. Es wird kostenlos über ein Github-Repository verteilt. Diese Verbreitungsmethode ermöglicht eine größere Nachhaltigkeit als Websites.

Wenn übrigens einige Leser dieser Seiten einen Beitrag leisten möchten, sind sie herzlich willkommen:

- Kapitel vorschlagen ;
- um Fehler zu melden oder Änderungen vorzuschlagen ;
- um bei der Übersetzung zu helfen...

Übersetzungshilfe

Mit Google Translate können Sie Texte einfach, aber mit Fehlern übersetzen. Deshalb bitte ich um Hilfe bei der Korrektur der Übersetzungen.

In der Praxis stelle ich die bereits übersetzten Kapitel im LibreOffice-Format zur Verfügung. Wenn Sie bei diesen Übersetzungen helfen möchten, besteht Ihre Aufgabe lediglich darin, diese Übersetzungen zu korrigieren und zurückzugeben.

Das Korrigieren eines Kapitels nimmt wenig Zeit in Anspruch, von einer bis zu mehreren Stunden.

Um mich zu erreichen : petremann@arduino-forth.com

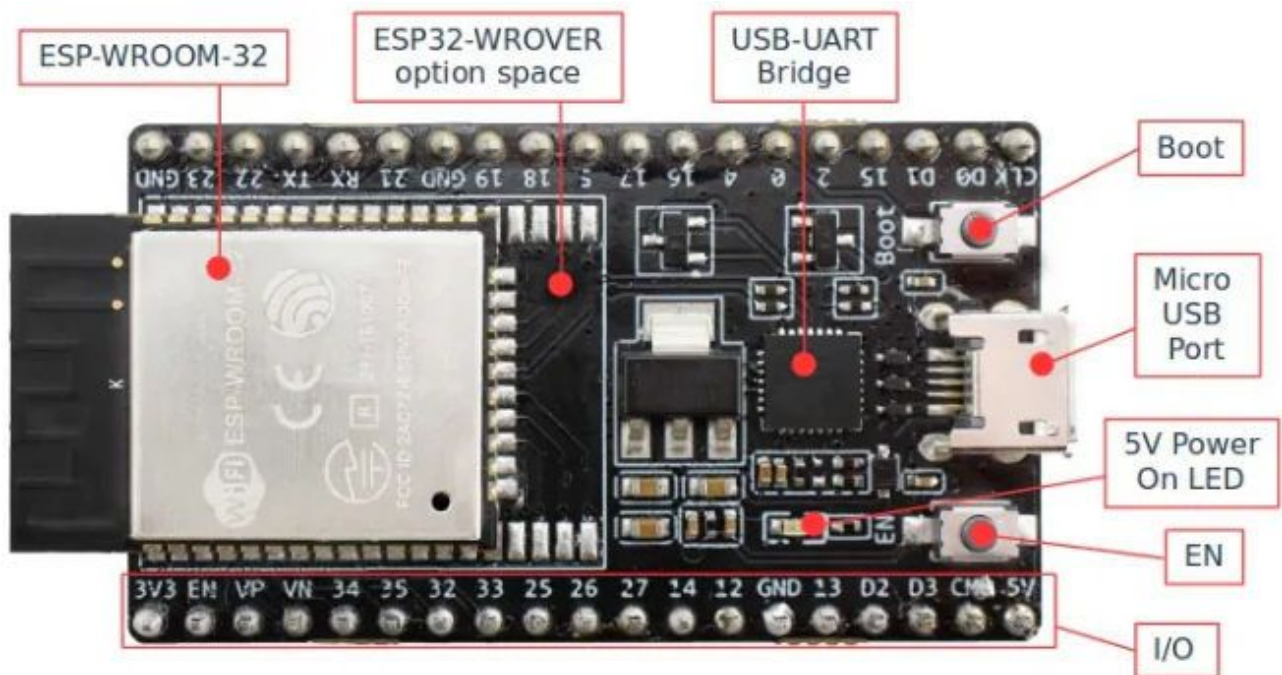
Entdeckung der ESP32-Karte

Präsentation

Das ESP32-Board ist kein ARDUINO-Board. Entwicklungstools nutzen jedoch bestimmte Elemente des ARDUINO-Ökosystems, wie beispielsweise die ARDUINO-IDE.

Die Stärken Punkten

Hinsichtlich der Anzahl der verfügbaren Ports liegt die ESP32-Karte zwischen einem



ARDUINO NANO und ARDUINO UNO. Das Basismodell verfügt über 38 Anschlüsse :

Zu den ESP32-Geräten gehören :

- 18 Analog-Digital-Wandlerkanäle (ADC).
- 3 SPI-Schnittstellen
- 3 UART-Schnittstellen
- 2 I2C-Schnittstellen
- 16 PWM-Ausgangskanäle
- 2 Digital-Analog-Wandler (DAC)
- 2 I2S-Schnittstellen

- 10 kapazitive GPIOs

Die ADC- (Analog-Digital-Wandler) und DAC-Funktionalität (Digital-Analog-Wandler) sind bestimmten statischen Pins zugewiesen. Sie können jedoch entscheiden, welche Pins UART, I2C, SPI, PWM usw. sind. Sie müssen sie nur im Code zuweisen. Dies ist dank der Multiplexing-Funktion des ESP32-Chips möglich.

Die meisten Steckverbinder haben mehrere Verwendungszwecke.

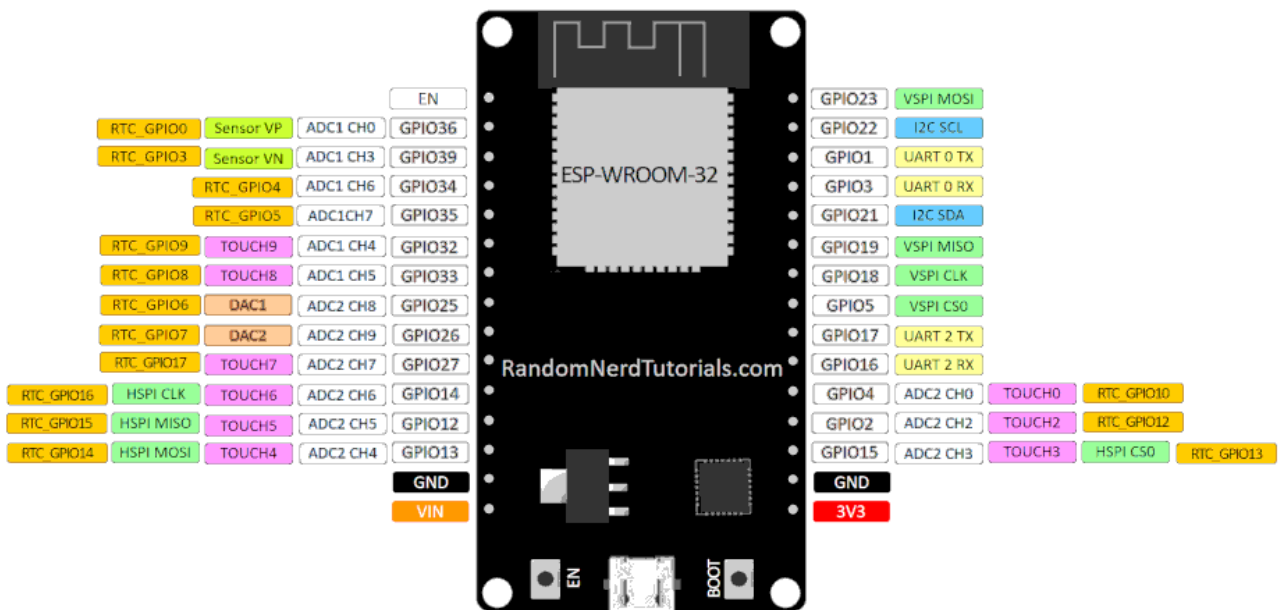
Was das ESP32-Board jedoch auszeichnet, ist, dass es standardmäßig mit WLAN- und Bluetooth-Unterstützung ausgestattet ist, was ARDUINO-Boards nur in Form von Erweiterungen bieten.

GPIO-Ein-/Ausgänge auf ESP32

Hier im Foto die ESP32-Karte, anhand derer wir die Rolle der verschiedenen GPIO-Ein-/Ausgänge erklären :



Die Position und Anzahl der GPIO-I/Os kann sich je nach Kartenmarke ändern. In diesem Fall sind nur die Angaben auf der physischen Karte authentisch. Im Bild, untere Reihe, von links nach rechts: CLK, SD0, SD1, G15, G2, G0, G4, G16.....G22, G23, GND.



In diesem Diagramm sehen wir, dass die untere Reihe mit 3V3 beginnt, während sich dieser I/O auf dem Foto am Ende der oberen Reihe befindet. Daher ist es sehr wichtig, sich nicht auf das Diagramm zu verlassen, sondern den korrekten Anschluss der Peripheriegeräte und Komponenten auf der physischen ESP32-Karte noch einmal zu überprüfen.

Entwicklungsboards auf Basis eines ESP32 verfügen neben denen für die Stromversorgung in der Regel über 33 Pins. Einige GPIO-Pins haben besondere Funktionen :

GPIO	Mögliche Namen
6	SCK/CLK
7	SCK/CLK
8	SDO/SD0
9	SDI/SD1
10	SHD/SD2
11	CSC/CMD

Wenn Ihre ESP32-Karte über I/O GPIO6, GPIO7, GPIO8, GPIO9, GPIO10, GPIO11 verfügt, sollten Sie diese auf keinen Fall verwenden, da sie mit dem Flash-Speicher des ESP32 verbunden sind. Wenn Sie sie verwenden, funktioniert der ESP32 nicht.

GPIO1(TX0) und GPIO3(RX0) I/O werden für die Kommunikation mit dem Computer in UART über den USB-Port verwendet. Wenn Sie diese verwenden, können Sie nicht mehr mit der Karte kommunizieren.

GPIO36(VP), GPIO39(VN), GPIO34, GPIO35 I/O können nur als Eingang verwendet werden. Sie verfügen auch nicht über eingebaute interne Pullup- und Pulldown-Widerstände.

Mit dem EN-Anschluss können Sie den Zündstatus des ESP32 über ein externes Kabel steuern. Es wird mit der EN-Taste auf der Karte verbunden. Wenn der ESP32 eingeschaltet ist, liegt er bei 3,3 V. Wenn wir diesen Pin mit Masse verbinden, wird der ESP32 ausgeschaltet. Sie können es verwenden, wenn sich der ESP32 in einer Box befindet und Sie ihn mit einem Schalter ein-/ausschalten möchten.

ESP32-Board-Peripheriegeräte

Um mit Modulen, Sensoren oder elektronischen Schaltkreisen zu interagieren, verfügt der ESP32 wie jeder Mikrocontroller über eine Vielzahl an Peripheriegeräten. Davon gibt es mehr als auf einem klassischen Arduino-Board.

ESP32 verfügt über die folgenden Peripheriegeräte :

- 3 UART-Schnittstellen
- 2 I2C-Schnittstellen
- 3 SPI-Schnittstellen
- 16 PWM-Ausgänge
- 10 kapazitive Sensoren
- 18 analoge Eingänge (ADC)
- 2 DAC-Ausgänge

Einige Peripheriegeräte werden bereits im Grundbetrieb von ESP32 genutzt. Somit gibt es pro Gerät weniger mögliche Schnittstellen.

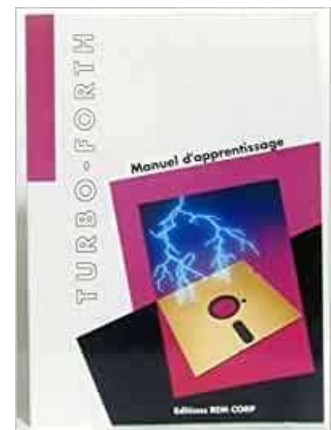
Warum auf ESP32 in FORTH-Sprache programmieren?

Präambel

Ich programmiere seit 1983 in FORTH. Ich habe 1996 mit dem Programmieren in FORTH aufgehört. Aber ich habe nie aufgehört, die Entwicklung dieser Sprache zu verfolgen. Ich habe 2019 wieder mit dem Programmieren auf ARDUINO mit FlashForth und dann mit ESP32forth begonnen.

Ich bin Co-Autor mehrerer Bücher über die FORTH-Sprache :

- Introduction au ZX-FORTH (ed Eyrolles - 1984 - ASIN:B0014IGOXO)
- Tours de FORTH (ed Eyrolles - 1985 - ISBN-13: 978-2212082258)
- FORTH pour CP/M et MSDOS (ed Loisetech - 1986)
- TURBO-Forth, manuel d'apprentissage (ed Rem CORP - 1990)
- TURBO-Forth, guide de référence (ed Rem CORP - 1991)



Das Programmieren in der FORTH-Sprache war schon immer ein Hobby, bis mich 1992 der Manager eines Unternehmens kontaktierte, das als Zulieferer für die Automobilindustrie tätig war. Sie hatten ein Interesse an der Softwareentwicklung in der Sprache C. Sie mussten einen Industrieautomaten bestellen.

Die beiden Softwareentwickler dieser Firma programmierten in der Sprache C: TURBO-C von Borland, um genau zu sein. Und ihr Code konnte nicht kompakt und schnell genug sein, um in den 64 Kilobyte großen RAM-Speicher zu passen. Es war 1992 und Flash-Speichererweiterungen gab es noch nicht. In diesen 64 KB RAM mussten wir MS-DOS 3.0 und die Anwendung unterbringen!

Einen Monat lang hatten C-Entwickler das Problem in alle Richtungen umgedreht, sogar Reverse Engineering mit SOURCER (einem Disassembler), um nicht wesentliche Teile des ausführbaren Codes zu entfernen.

Ich habe das mir vorgelegte Problem analysiert. Von Grund auf habe ich innerhalb einer Woche alleine einen perfekt funktionsfähigen Prototyp erstellt, der den Spezifikationen entsprach. Drei Jahre lang, von 1992 bis 1995, habe ich zahlreiche Versionen dieser Anwendung erstellt, die auf den Montagebändern mehrerer Automobilhersteller eingesetzt wurden.

Grenzen zwischen Sprache und Anwendung

Alle Programmiersprachen werden auf diese Weise geteilt :

- ein Interpreter und ausführbarer Quellcode: BASIC, PHP, MySQL, JavaScript usw. Die Anwendung ist in einer oder mehreren Dateien enthalten, die bei Bedarf interpretiert werden. Das System muss den Interpreter, der den Quellcode ausführt, dauerhaft hosten ;
- ein Compiler und/oder Assembler: C, Java usw. Einige Compiler generieren nativen Code, also speziell auf einem System ausführbar. Andere, wie Java, kompilieren ausführbaren Code auf einer virtuellen Java-Maschine.

Eine Ausnahme bildet die FORTH-Sprache. Es integriert :

- ein Dolmetscher, der jedes Wort in der FORTH-Sprache ausführen kann
- ein Compiler, der das Wörterbuch der FORTH-Wörter erweitern kann

Was ist ein FORTH-Wort?

Ein FORTH-Wort bezeichnet einen beliebigen Wörterbuchausdruck, der aus ASCII-Zeichen besteht und bei der Interpretation und/oder Kompilierung verwendet werden kann: Mit Wörter können Sie alle Wörter im FORTH-Wörterbuch auflisten.

Bestimmte FORTH-Wörter können nur bei der Kompilierung verwendet werden: **if else then** zum Beispiel.

Bei der FORTH-Sprache besteht das wesentliche Prinzip darin, dass wir keine Anwendung erstellen. In FORTH erweitern wir das Wörterbuch! Jedes neue Wort, das Sie definieren, ist ebenso Teil des FORTH-Wörterbuchs wie alle beim Start von FORTH vordefinierten Wörter. Beispiel :

```
: typeToLoRa ( -- )
    0 echo !      \ Deaktivieren Sie das Echo der Terminalanzeige
    ['] serial2-type is type
;
: typeToTerm ( -- )
    ['] default-type is type
    -1 echo !     \ Aktiviert das Display-Echo des Terminals
;
```

Wir erstellen zwei neue Wörter: **typeToLoRa** und **typeToTerm**, die das Wörterbuch vordefinierter Wörter vervollständigen.

Ein Wort ist eine Funktion?

Ja und nein. Tatsächlich kann ein Wort eine Konstante, eine Variable, eine Funktion sein... Hier in unserem Beispiel die folgende Sequenz :

```
: typeToLoRa ...code... ;
```

hätte sein Äquivalent in der C-Sprache :

```
void typeToLoRa() { ...code... }
```

In der FORTH-Sprache gibt es keine Grenze zwischen Sprache und Anwendung.

In FORTH können Sie wie in der Sprache C jedes bereits definierte Wort in der Definition eines neuen Worts verwenden.

Ja, aber warum dann FORTH statt C?

Ich habe diese Frage erwartet.

In der C-Sprache kann auf eine Funktion nur über die Hauptfunktion main() zugegriffen werden. Wenn diese Funktion mehrere Zusatzfunktionen integriert, wird es bei einer Fehlfunktion des Programms schwierig, einen Parameterfehler zu finden.

Im Gegenteil, mit FORTH ist es möglich, über den Interpreter jedes vordefinierte oder von Ihnen definierte Wort auszuführen, ohne das Hauptwort des Programms durchlaufen zu müssen.

Der FORTH-Interpreter ist über ein Terminalprogramm und eine USB-Verbindung zwischen der ESP32-Karte und dem PC sofort auf der ESP32-Karte zugänglich.

Die Kompilierung von in FORTH-Sprache geschriebenen Programmen erfolgt in der ESP32-Karte und nicht auf dem PC. Es erfolgt kein Upload. Beispiel:

```
: >gray ( n -- n' )  
    dup 2/ xor      \ n' = n xor ( 1 logische Verschiebung nach  
rechts )  
    ;
```

Diese Definition wird per Kopieren/Einfügen in das Terminal übertragen. Der FORTH-Interpreter/Compiler analysiert den Stream und kompiliert das neue Wort **>gray**.

In der Definition von **>gray** sehen wir die Sequenz **dup 2/ xor**. Um diese Sequenz zu testen, geben Sie sie einfach in das Terminal ein. Um **>gray** auszuführen, geben Sie einfach dieses Wort in das Terminal ein, gefolgt von der Zahl, die umgewandelt werden soll.

FORTH-Sprache im Vergleich zur C-Sprache

Das ist der Teil, den ich am wenigsten mag. Ich vergleiche die FORTH-Sprache nicht gern mit der C-Sprache. Aber da fast alle Entwickler die C-Sprache verwenden, werde ich die Übung ausprobieren.

Hier ist ein Test mit **if()** in C-Sprache:

```
if(j > 13){                // Wenn alle Bits empfangen wurden
```

```

    rc5_ok = 1;           // Der Dekodierungsprozess ist OK
    detachInterrupt(0);   // Externen Interrupt deaktivieren (INT0)
    return;
}

```

Testen Sie mit if in FORTH-Sprache (Code-Snippet) :

```

var-j @ 13 >           \ Wenn alle Bits empfangen wurden
  if
    1 rc5_ok !         \ Der Dekodierungsprozess ist OK
    di                 \ Externen Interrupt deaktivieren (INT0)
    exit
  then

```

Hier ist die Initialisierung von Registern in C-Sprache :

```

void setup() {
  // Konfigurieren des Timer1-Moduls
  TCCR1A = 0;
  TCCR1B = 0;           // Deaktiviert das Timer1-Modul
  TCNT1 = 0;           // Setzt den Vorladewert von Timer1 auf 0
  (reset)
  TIMSK1 = 1;          // Überlauf-Interrupt aktivieren Timer1
}

```

Die gleiche Definition in der FORTH-Sprache:

```

: setup ( -- )
  \ Konfigurieren des Timer1-Moduls
  0 TCCR1A !
  0 TCCR1B !           \ Deaktiviert das Timer1-Modul
  0 TCNT1 !           \ Setzt den Vorladewert von Timer1 auf 0 (reset)
  1 TIMSK1 !          \ Überlauf-Interrupt aktivieren Timer1
;

```

Was FORTH Ihnen im Vergleich zur C-Sprache ermöglicht

Wir verstehen, dass FORTH sofort Zugriff auf alle Wörter im Wörterbuch bietet, aber nicht nur darauf. Über den Interpreter greifen wir auch auf den gesamten Speicher der ESP32-Karte zu. Stellen Sie eine Verbindung zum ARDUINO-Board her, auf dem FlashForth installiert ist, und geben Sie dann einfach Folgendes ein:

```
hex here 100 dump
```

Sie sollten dies auf dem Terminalbildschirm finden :

```

3FFEE964                                DF DF 29 27 6F 59 2B 42 FA CF 9B 84
3FFEE970                                39 4E 35 F7 78 FB D2 2C A0 AD 5A AF 7C 14 E3 52
3FFEE980                                77 0C 67 CE 53 DE E9 9F 9A 49 AB F7 BC 64 AE E6
3FFEE990                                3A DF 1C BB FE B7 C2 73 18 A6 A5 3F A4 68 B5 69

```

3FFEE9A0	F9 54 68 D9 4D 7C 96 4D 66 9A 02 BF 33 46 46 45
3FFEE9B0	45 39 33 33 2F 0D 08 18 BF 95 AF 87 AC D0 C7 5D
3FFEE9C0	F2 99 B6 43 DF 19 C9 74 10 BD 8C AE 5A 7F 13 F1
3FFEE9D0	9E 00 3D 6F 7F 74 2A 2B 52 2D F4 01 2D 7D B5 1C
3FFEE9E0	4A 88 88 B5 2D BE B1 38 57 79 B2 66 11 2D A1 76
3FFEE9F0	F6 68 1F 71 37 9E C1 82 43 A6 A4 9A 57 5D AC 9A
3FFEEA00	4C AD 03 F1 F8 AF 2E 1A B4 67 9C 71 25 98 E1 A0
3FFEEA10	E6 29 EE 2D EF 6F C7 06 10 E0 33 4A E1 57 58 60
3FFEEA20	08 74 C6 70 BD 70 FE 01 5D 9D 00 9E F7 B7 E0 CA
3FFEEA30	72 6E 49 16 0E 7C 3F 23 11 8D 66 55 EC F6 18 01
3FFEEA40	20 E7 48 63 D1 FB 56 77 3E 9A 53 7D B6 A7 A5 AB
3FFEEA50	EA 65 F8 21 3D BA 54 10 06 16 E6 9E 23 CA 87 25
3FFEEA60	E7 D7 C4 45

Dies entspricht dem Inhalt des Flash-Speichers.

Und die C-Sprache konnte das nicht?

Ja, aber nicht so einfach und interaktiv wie in der FORTH-Sprache.

Aber warum ein Stapel statt Variablen?

Der Stack ist ein Mechanismus, der auf fast allen Mikrocontrollern und Mikroprozessoren implementiert ist. Sogar die C-Sprache nutzt einen Stack, aber Sie haben keinen Zugriff darauf.

Nur die FORTH-Sprache bietet vollständigen Zugriff auf den Datenstapel. Um beispielsweise eine Addition durchzuführen, stapeln wir zwei Werte, führen die Addition aus und zeigen das Ergebnis an: **2 5 + .** zeigt **7** an.

Es ist ein wenig destabilisierend, aber wenn Sie den Mechanismus des Datenstapels verstehen, werden Sie seine beeindruckende Effizienz sehr zu schätzen wissen.

Mit dem Datenstapel können Daten viel schneller zwischen FORTH-Worten übertragen werden als durch die Verarbeitung von Variablen wie in der C-Sprache oder einer anderen Sprache, die Variablen verwendet.

Sind Sie überzeugt?

Persönlich bezweifle ich, dass dieses einzelne Kapitel Sie endgültig zum Programmieren in der FORTH-Sprache bekehren wird. Wenn Sie ESP32-Boards beherrschen möchten, haben Sie zwei Möglichkeiten :

- Programmieren Sie das Programm in C-Sprache und nutzen Sie die zahlreichen verfügbaren Bibliotheken. Sie bleiben jedoch an die Möglichkeiten dieser Bibliotheken gebunden. Die Anpassung von Codes an die C-Sprache erfordert echte Programmierkenntnisse in der C-Sprache und die Beherrschung der Architektur von

ESP32-Karten. Die Entwicklung komplexer Programme wird immer ein Problem sein.

- Probieren Sie das FORTH-Abenteuer aus und erkunden Sie eine neue und aufregende Welt. Natürlich wird es nicht einfach sein. Sie müssen die Architektur von ESP32-Karten, die Register und die Registerflags im Detail verstehen. Im Gegenzug erhalten Sie Zugang zu einer Programmierung, die perfekt zu Ihren Projekten passt.

Gibt es professionelle Bewerbungen, die in FORTH verfasst sind?

Oh ja! Beginnend mit dem HUBBLE-Weltraumteleskop, dessen bestimmte Komponenten in der FORTH-Sprache geschrieben wurden.

Der deutsche TGV ICE (Intercity Express) nutzt RTX2000-Prozessoren zur Steuerung von Motoren über Leistungshalbleiter. Die Maschinensprache des RTX2000-Prozessors ist die FORTH-Sprache.

Derselbe RTX2000-Prozessor wurde für die Philae-Sonde verwendet, die versuchte, auf einem Kometen zu landen.

Die Wahl der FORTH-Sprache für professionelle Anwendungen erweist sich als interessant, wenn wir jedes Wort als Blackbox betrachten. Jedes Wort muss einfach sein, daher eine relativ kurze Definition haben und von wenigen Parametern abhängen.

Während der Debugging-Phase ist es einfach, alle möglichen Werte zu testen, die von diesem Wort verarbeitet werden. Sobald dieses Wort vollkommen zuverlässig ist, wird es zu einer Blackbox, also zu einer Funktion, deren ordnungsgemäßes Funktionieren wir absolut vertrauen können. Von Wort zu Wort ist es in FORTH einfacher, ein komplexes Programm zuverlässig zu machen als in jeder anderen Programmiersprache.

Aber wenn es uns an Genauigkeit mangelt, wenn wir Gasanlagen bauen, ist es auch sehr leicht, dass eine Anwendung schlecht funktioniert oder sogar völlig abstürzt!

Schließlich ist es in der FORTH-Sprache möglich, die von Ihnen definierten Wörter in jeder menschlichen Sprache zu schreiben. Allerdings sind die verwendbaren Zeichen auf den ASCII-Zeichensatz zwischen 33 und 127 beschränkt. So könnten wir die Wörter **high** und **low** symbolisch umschreiben :

```
\ Aktiver Port-Pin, andere nicht ändern.  
: __/ ( pinmask portadr -- )  
  mset  
;  
\ Das Deaktivieren eines Port-Pins hat keine Auswirkungen auf die  
anderen.  
: \__ ( pinmask portadr -- )
```

```
mclr  
;
```

Ab diesem Moment können Sie zum Einschalten der LED Folgendes eingeben :

```
_0_ __/ \ allume LED
```

Ja! Die Sequenz **_0_ __/** ist in FORTH-Sprache!

Mit ESP32forth stehen Ihnen hier alle Zeichen zur Verfügung, die ein FORTH-Wort bilden können:

```
~}|{zyxwvutsrqponmlkjihgfedcba`_  
^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?  
>=<;:9876543210/.-,*)( ' &%$#"!
```

Gute Programmierung.

Ein echtes 32-Bit FORTH mit ESP32Forth

ESP32Forth ist ein echtes 32-Bit FORTH. Was bedeutet das ?

Die FORTH-Sprache bevorzugt die Manipulation ganzzahliger Werte. Diese Werte können Literalwerte, Speicheradressen, Registerinhalte usw. sein.

Werte auf dem Datenstapel

Wenn ESP32Forth startet, ist der FORTH-Interpreter verfügbar. Wenn Sie eine beliebige Zahl eingeben, wird diese als 32-Bit-Ganzzahl auf dem Stapel abgelegt:

```
35
```

Wenn wir einen anderen Wert stapeln, wird dieser ebenfalls gestapelt. Der vorherige Wert wird um eine Position nach unten verschoben:

```
45
```

Um diese beiden Werte zu addieren, verwenden wir ein Wort, hier **+** :

```
+
```

Unsere beiden 32-Bit-Ganzzahlwerte werden addiert und das Ergebnis auf dem Stapel abgelegt. Um dieses Ergebnis anzuzeigen, verwenden wir das Wort **.** :

```
. \ zeigt 80 an
```

In der FORTH-Sprache können wir alle diese Operationen in einer einzigen Zeile konzentrieren:

```
35 45 +. \ 80 anzeigen
```

Im Gegensatz zur C-Sprache definieren wir keinen **int8-**, **int16-** oder **int32- Typ** .

Mit ESP32Forth wird ein ASCII-Zeichen durch eine 32-Bit-Ganzzahl bezeichnet, deren Wert jedoch auf [32..256[begrenzt ist. Beispiel :

```
67 emit \ zeigt C
```

Werte im Gedächtnis

Mit ESP32Forth können Sie Konstanten und Variablen definieren. Ihr Inhalt liegt immer im 32-Bit-Format vor. Aber es gibt Situationen, in denen uns das nicht unbedingt passt. Nehmen wir ein einfaches Beispiel und definieren ein Morsecode-Alphabet. Wir brauchen nur ein paar Bytes:

- eines, um die Anzahl der Morsezeichen zu definieren
- ein oder mehrere Bytes für jeden Buchstaben des Morsecodes

```
create mA ( -- addr )
  2 c,
  char . c,   char - c,

create mB ( -- addr )
  4 c,
  char - c,   char . c,   char . c,   char . c,

create mC ( -- addr )
  4 c,
  char - c,   char . c,   char - c,   char . c,
```

Hier definieren wir nur 3 Wörter, **mA** , **mB** und **mC** . In jedem Wort werden mehrere Bytes gespeichert. Die Frage ist : Wie werden wir die Informationen in diesen Worten abrufen ?

Die Ausführung eines dieser Wörter hinterlegt einen 32-Bit-Wert, einen Wert, der der Speicheradresse entspricht, an der wir unsere Morsecode-Informationen gespeichert haben. Es ist das Wort **c@** , das wir verwenden werden, um den Morsecode aus jedem Buchstaben zu extrahieren :

```
mA c@ . \ zeigt 2 an
mB c@ . \ zeigt 4 an
```

Das erste so extrahierte Byte wird zur Verwaltung einer Schleife zur Anzeige des Morsecodes eines Buchstabens verwendet :

```
: .morse ( addr -- )
  dup 1+ swap c@ 0 do
    dup i + c@ emit
  loop
  drop
;
mA .morse \ zeigt .-
mB .morse \ zeigt -...
mC .morse \ zeigt -.-.
```

Es gibt sicherlich viele elegantere Beispiele. Hier soll eine Möglichkeit gezeigt werden, 8-Bit-Werte, unsere Bytes, zu manipulieren, während wir diese Bytes auf einem 32-Bit-Stack verwenden.

Textverarbeitung je nach Datengröße oder -typ

In allen anderen Sprachen gibt es ein generisches Wort wie **echo** (in PHP), das jede Art von Daten anzeigt. Ob Integer, Real, String, wir verwenden immer das gleiche Wort.

Beispiel in PHP-Sprache:

```
$bread = "Gebackenes Brot";
$preis = 2,30;
echo $bread . " : " . $preis;
// zeigt Gebackenes Brot: 2,30
```

Für alle Programmierer ist diese Vorgehensweise DER STANDARD! Wie würde FORTH dieses Beispiel in PHP umsetzen?

```
: bread s" Baked bread" ;
: price s" 2.30" ;
bread type    s" : " type    price type
\ display    Baked bread: 2.30
```

Hier sagt uns der **type** , dass wir gerade eine Zeichenfolge verarbeitet haben.

Wo PHP (oder eine andere Sprache) über eine generische Funktion und einen Parser verfügt, kompensiert FORTH dies mit einem einzigen Datentyp, aber angepassten Verarbeitungsmethoden, die uns über die Art der verarbeiteten Daten informieren.

Hier ist ein absolut trivialer Fall für FORTH, bei dem eine Anzahl von Sekunden im Format HH:MM:SS angezeigt wird:

```
: :##
  # 6 base !
  # decimal
  [char] : hold
;
: .hms ( n -- )
  <# :## :## # # #> type
;
4225 .hms \ zeigt: 01:10:25
```

Ich liebe dieses Beispiel, weil bisher **KEINE ANDERE PROGRAMMIERSPRACHE** in der Lage ist, diese HH:MM:SS-Konvertierung so elegant und prägnant durchzuführen.

Sie haben verstanden, das Geheimnis von FORTH liegt in seinem Wortschatz.

Abschluss

FORTH hat keine Datentypisierung. Alle Daten durchlaufen einen Datenstapel. Jede Position im Stapel ist IMMER eine 32-Bit-Ganzzahl!

Das ist alles, was man wissen muss.

Puristen hyperstrukturierter und ausführlicher Sprachen wie C oder Java werden sicherlich Häresie ausrufen. Und hier erlaube ich mir, sie zu beantworten : Warum müssen Sie Ihre Daten eingeben ?

Denn in dieser Einfachheit liegt die Stärke von FORTH: ein einzelner Datenstapel mit einem untypisierten Format und sehr einfachen Operationen.

Und ich zeige Ihnen, was viele andere Programmiersprachen nicht können, nämlich neue Definitionswörter zu definieren :









```
: morse: ( comp: c -- | exec -- )
  create
    c,
  does>
    dup 1+ swap c@ 0 do
      dup i + c@ emit
    loop
  drop space
;
2 morse: mA      char . c,   char - c,
4 morse: mB      char - c,   char . c,   char . c,   char . c,
4 morse: mC      char - c,   char . c,   char - c,   char . c,
mA mB mC      \ zeigt   .- -... -..
```

Hier ist das Wort **morse:** zu einem Definitionswort geworden, ebenso wie **constant** oder **variable** ...

Denn FORTH ist mehr als eine Programmiersprache. Es handelt sich um eine Metasprache, also um eine Sprache zum Aufbau einer eigenen Programmiersprache....

Installieren der OLED-Bibliothek für SSD1306

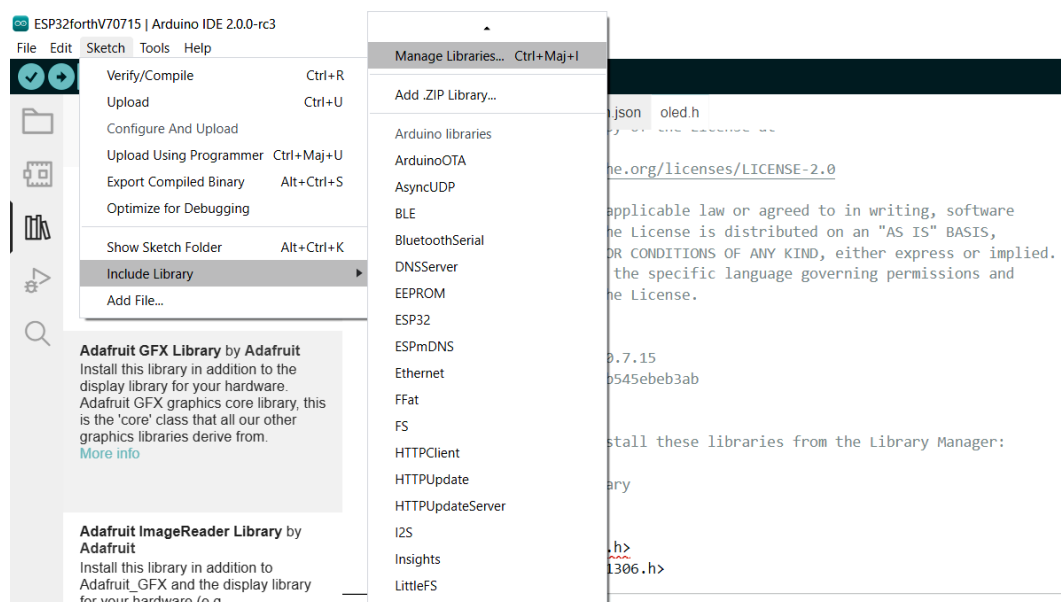
Seit ESP32forth Version 7.0.7.15 sind die Optionen im **optional** Ordner verfügbar:

Téléchargements > ESP32forth-7.0.7.15(1).zip > ESP32forth > optional		
Nom	Type	
 assemblers.h	Fichier H	
 camera.h	Fichier H	
 interrupts.h	Fichier H	
 oled.h	Fichier H	
 README-optional.txt	Document texte	
 rmt.h	Fichier H	
 serial-bluetooth.h	Fichier H	
 spi-flash.h	Fichier H	

Um das **oled** Vokabular zu erhalten, kopieren Sie die Datei **oled.h** in den Ordner, der die Datei **ESP32forth.ino** enthält.

Starten Sie dann ARDUINO IDE und wählen Sie die neueste **ESP32forth.ino**-Datei aus.

Wenn die OLED-Bibliothek nicht installiert wurde, klicken Sie in der ARDUINO IDE auf *Sketch* und wählen Sie *Include Library* und dann *Manage Libraries*.



Suchen Sie in der linken Seitenleiste nach der Bibliothek **Adafruit SSD1306 by Adafruit**.

Sie können nun mit der Zusammenstellung der Skizze beginnen, indem Sie auf *Sketch* klicken und *Upload* auswählen.

Sobald die Skizze auf die ESP32-Karte hochgeladen ist, starten Sie das TeraTerm-Terminal. Überprüfen Sie, ob das **oled**-Vokabular vorhanden ist :

```
oled vlist \ display:
OledInit SSD1306_SWITCHCAPVCC SSD1306_EXTERNALVCC WHITE BLACK OledReset
HEIGHT WIDTH OledAddr OledNew OledDelete OledBegin OledHOME OledCLS OledTextc
OledPrintln OledNumln OledNum OledDisplay OledPrint OledInvert OledTextsize
OledSetCursor OledPixel OledDrawL OledCirc OledCircF OledRect OledRectF
OledRectR OledRectRF oled-builtins
```


Ressourcen

Auf Englisch

- **ESP32forth** Seite verwaltet von Brad NELSON, dem Erfinder von ESP32forth. Dort finden Sie alle Versionen (ESP32, Windows, Web, Linux...)
<https://esp32forth.appspot.com/ESP32forth.html>

•

Auf Französisch

- **ESP32 Forth** Website in zwei Sprachen (Französisch, Englisch) mit vielen Beispielen
<https://esp32.arduino-forth.com/>

GitHub

- **Ueforth** Ressourcen verwaltet von Brad NELSON. Enthält alle Forth- und C-Sprach Quelldateien für ESP32forth
<https://github.com/flagxor/ueforth>
- **ESP32forth** Quellcodes und Dokumentation für ESP32forth. Ressourcen verwaltet von Marc PETREMANN
<https://github.com/MPETREMANN11/ESP32forth>
- **ESP32forthStation** Ressourcen verwaltet von Ulrich HOFFMAN. Eigenständiger Forth-Computer mit LillyGo TTGO VGA32-Einplatinencomputer und ESP32forth
<https://github.com/uho/ESP32forthStation>
- **ESP32Forth** Ressourcen verwaltet von F. J. RUSSO
<https://github.com/FJRusso53/ESP32Forth>
- **esp32forth-addons** Ressourcen verwaltet von Peter FORTH
<https://github.com/PeterForth/esp32forth-addons>
- **Esp32forth-org** Code-Repository für Mitglieder der Forth2020- und ESP32forth-Gruppen
<https://github.com/Esp32forth-org>

•

Index

FORTH-Wort.....9 oled.....19