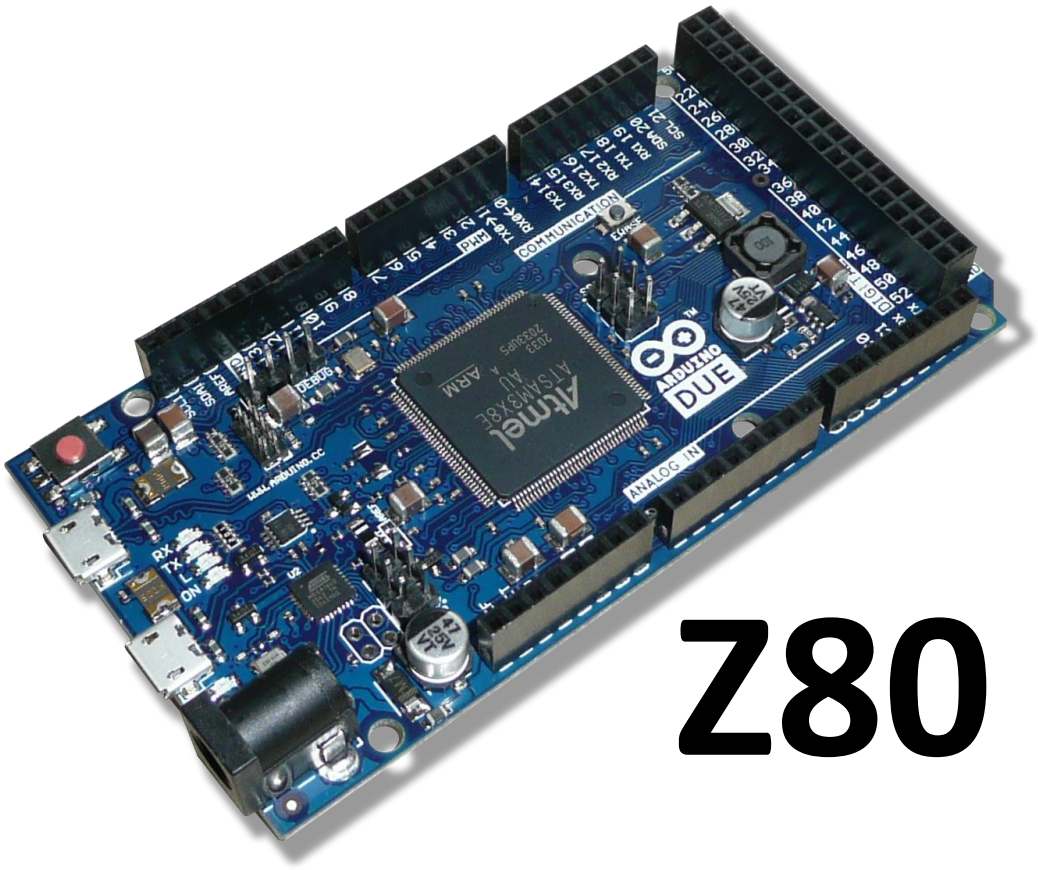
Erik Bartmann

Spaß mit dem Betriebssystem CP/M



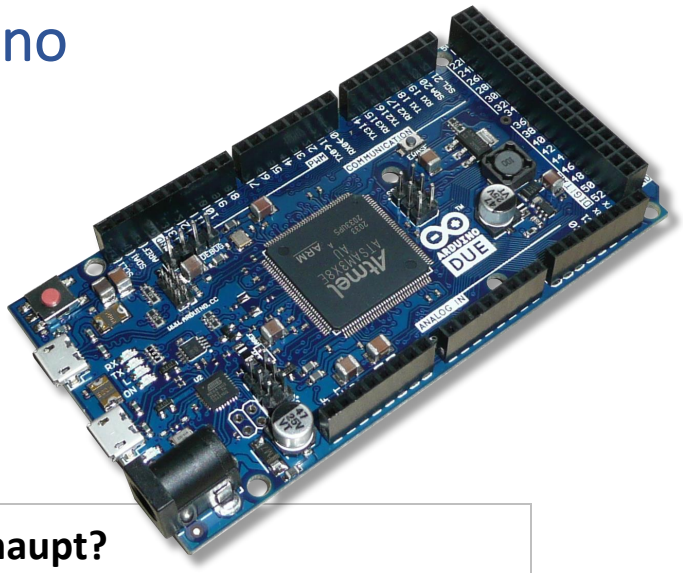
Z80

CP/M auf dem Arduino DUE

Autor	Erik Bartmann
Internet	https://erik-bartmann.de/
Thema	Spaß mit dem Betriebssystem CP/M
Version	1.01
Datum	31. Januar 2022

Inhaltsverzeichnis

CP/M mit einem Arduino



Worum geht es überhaupt?

In diesem Papier werden folgende Themen besprochen.

- Die Umsetzung eines CP/M-Systems mit einem Arduino DUE
- Der Arduino DUE
- Der SD-Card Adapter

Wenn jemand den Reiz verspürt, sich mit einer etwas älteren, aber immer noch sehr interessanten Technik der Z80-CPU, auseinanderzusetzen, dann ist er hier genau richtig gelandet. Dieses Papier stellt die Z80-CPU mithilfe des Mikrocontrollers *Arduino DUE* vor. Ich beziehe mich dabei auf eine Entwicklung, die unter der folgenden Internetadresse zu finden ist und sich *RunCPM* nennt.

Ich habe diesbezüglich keine Programmierleistung erbracht und möchte mich nicht mit fremden Federn schmücken. Mein Anliegen war es, das ganze Thema in deutscher Sprache etwas aufzuarbeiten.

Hyperlink!

https://github.com/MockbaTheBorg/RunCPM

Der Interessierte lernt sowohl etwas über das Betriebssystem CP/M, als auch ein wenig über die Z80-CPU und dient dazu da, einen geeigneten Einstieg in das Thema CP/M und Z80 zu finden. Spaß ist auf jeden Fall die oberste Prämisse und das wird in meinen Augen auch geschehen. Wer wirklich grundlegende (Er)Kenntnisse darüber erlangen möchte, wie eine CPU funktioniert, wie Bits und Bytes hin und hergeschoben werden, der sollte hier etwas Zeit investieren. Es gibt in meinen Augen nichts Schlimmeres, als sich als angehender Entwickler nur mit Programmier-Hochsprachen zu beschäftigen und dann nicht zu wissen, was quasi unter der Haube eines Computers geschieht. Das ist heutzutage im Zeitalter „moderner“ Betriebssysteme wie Windows, Mac OSX oder Linux, sehr schwierig bis unmöglich, zu erfassen beziehungsweise zu realisieren, was der Computer denn so treibt, während wir gebannt und paralysiert auf den Bildschirm starren. Alle drei bis 5 Tage werden Updates eingespielt und auch nicht die vermeintlichen Spezialisten können ergründen, was denn wirklich im Hintergrund „gespielt“ wird. Eine in meinen Augen sehr verwerfliche Situation, die den Nutzern

zugemutet wird. Das bedeutet nun nicht, alle modernen Computer mit dem Betriebssystem CP/M zu versehen, was sowieso nicht funktionieren würde. Doch zurück zum eigentlichen Thema. Der Z80 ist eine 8-Bit-CPU, was bedeutet, dass sein Datenbus eine Breite von 8 Bits besitzt. Mit diesen 8-Bits können eine Menge interessanter Dinge angestellt werden, wobei diese über einen 16-Bit breiten Adressbus quasi gemanaged werden. Auch mit Betriebssystem CP/M - DOS hat sich hier reichlich Innovationen geholt - können Dateien erstellt, gelöscht, kopiert oder umbenannt werden. Es gibt Programme für die Textverarbeitung, die Tabellenkalkulation und für Datenbanken. Sogar Spiele wurden entwickelt, die jedoch nur im Textmodus verfügbar sind, da CP/M keine Möglichkeit der Grafikausgabe hat. Das was nützen heutzutage die unzähligen hirnlosen Spiele mit wahnsinniger Grafik, die ohne Sinn und Verstand programmiert wurden. Da lobe ich mir doch ein kniffliges Text-Adventure, dass die kleinen grauen Zellen in Bewegung und einen in Grübeln bringt. Das alles ist jedoch nur meine persönliche Meinung und spiegelt nicht die öffentliche Meinung wider.

Die Vorbereitungen

Um das Ganze zu beginnen, ist natürlich etwas Hardware erforderlich. Auf der genannten Internetseite wird beschrieben, dass unterschiedliche Mikrocontroller-Boards zur Umsetzung genutzt werden können.

- Arduino DUE
- Teensy 3.5 oder höher
- ESP32
- STM32

Ich nutze in diesem Fall mein *Arduino DUE* Board. Andere Arduino-Board können aufgrund des zu geringen Speichers nicht verwendet werden. Bei Verwendung des Arduino DUE benötigt RunCPM zusätzlich noch einen SD-Karten-Adapter und eine SD- oder microSD- Karte, auf der die CP/M-Dateien gespeichert werden. Der Teensy hat einen eingebauten microSD-Adapter. Einige ESP32- und STM32-Boards benötigen möglicherweise externe SD-Kartenadapter.

Die Hardware

Doch bevor es nun wirklich losgehen kann, muss natürlich erst einmal die Hardware vorbereitet werden. Dazu ist es erforderlich, den SD-Karten-Adapter mit dem Arduino DUE zu verbinden, was sowohl über die SPI-Schnittstelle, als auch über einen digitalen I/O-Pin erfolgt. Werfen wir zunächst einen Blick auf die SPI-Schnittstelle.

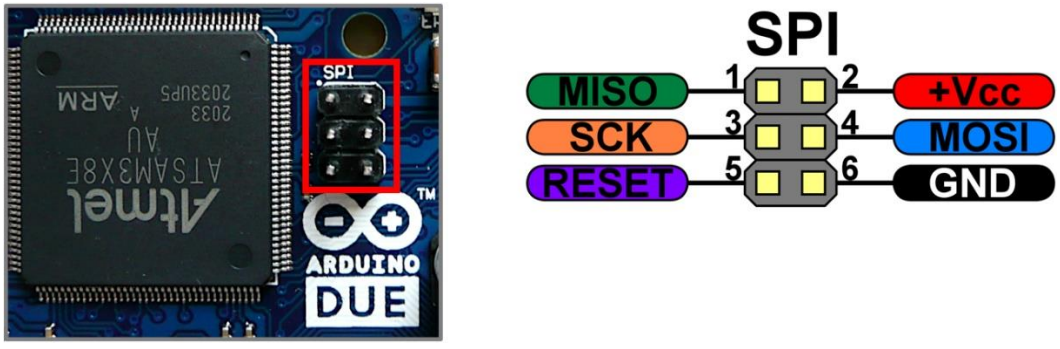


Abbildung 1 Die SPI-Schnittstelle

Der von mir verwendete SD-Karten-Adapter schaut wie folgt aus.

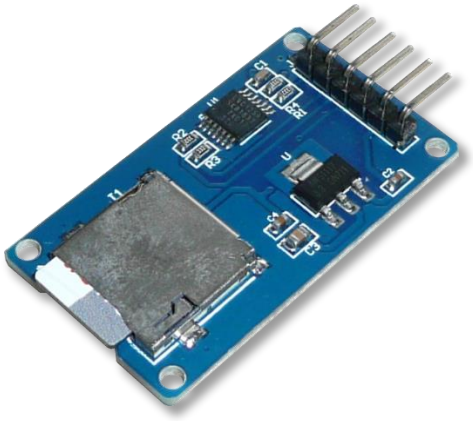


Abbildung 2 Der SD-Karten-Adapter

Es besitzt die für die SPI-Schnittstelle üblichen Pins.

- CS (Chip Select)
- SCK (Clock)
- MOSI
- MISO
- Vcc
- GND

Der Adapter muss nun in der folgenden Weise mit dem Arduino DUE-Board verbunden werden.

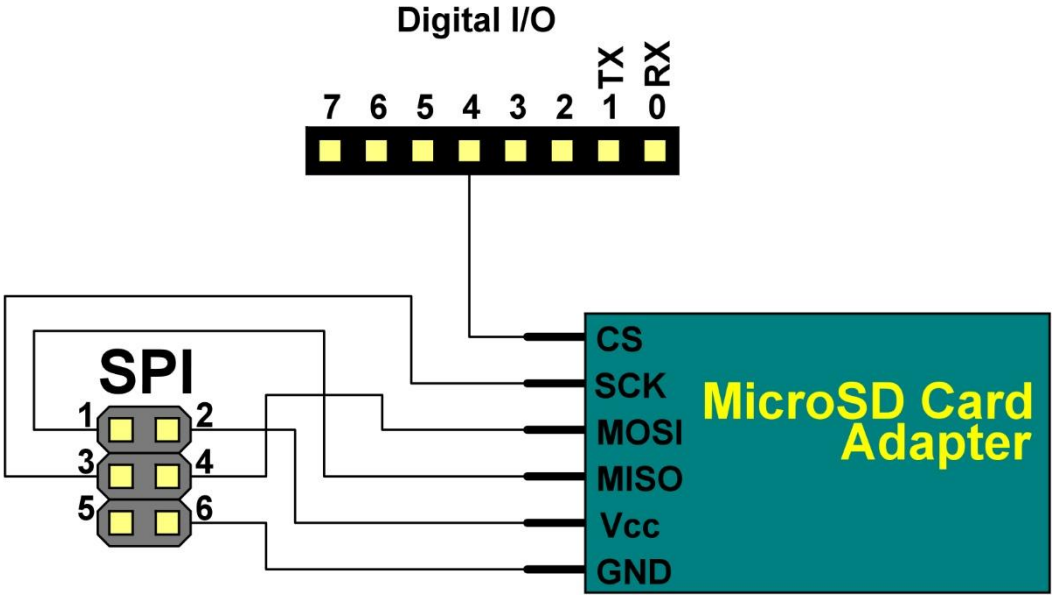


Abbildung 3 Die elektrischen Verbindung des SD-Karten Adapters mit dem Arduino Due

Im Endeffekt schaut es dann wie folgt aus.

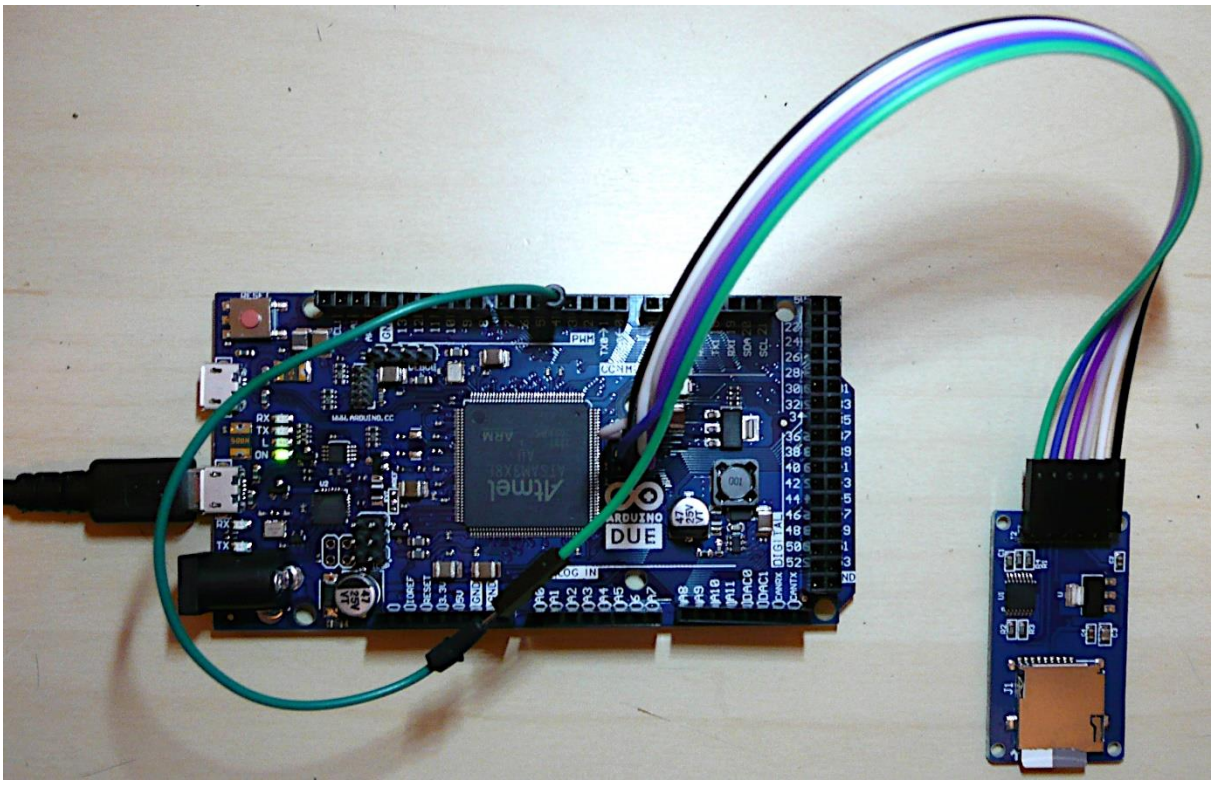


Abbildung 4 Der Arduino DUE mit dem SD-Karten Adapter

Die Arduino-Software

Um das RunCPM-Projekt auf das Mikrocontroller-Board hochzuladen, ist die Arduino-Entwicklungsumgebung notwendig, die unter der folgenden Internetadresse zu finden ist.

Hyperlink!

https://www.arduino.cc/en/software

Um die Funktionalität der SD-Karte nutzen zu können, ist zusätzlich noch die Installation einer Library mit dem Namen *SdFat* von *Bill Greiman* erforderlich. Das Hinzufügen erfolgt über den Bibliotheksverwalter der Arduino Entwicklungsumgebung.

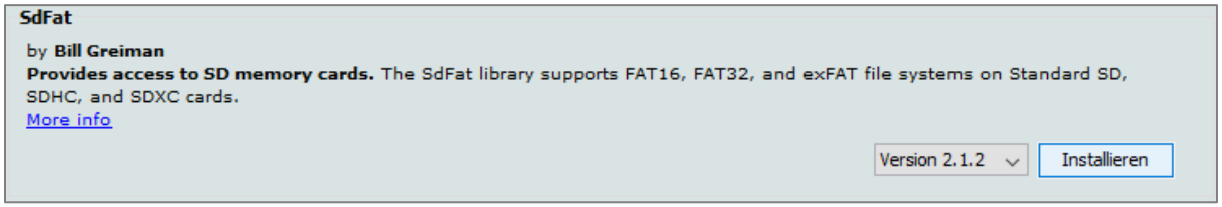


Abbildung 5 Die erforderliche SdFat-Bibliothek

Um nun das ganze Projekt beginnen zu können, ist es natürlich erforderlich, den Arduino-Quellcode (Sketch) in die Entwicklungsumgebung zu bekommen. Dazu lädt man sich am besten das gesamte Github-Repository herunter, das unter der schon genannten Internetadresse zu finden ist.

Hyperlink!

https://github.com/MockbaTheBorg/RunCPM

Dazu wird einfach die grüne *Code*-Schaltfläche rechts oben angeklickt und dann der unterste Punkt *Download ZIP* ausgewählt.

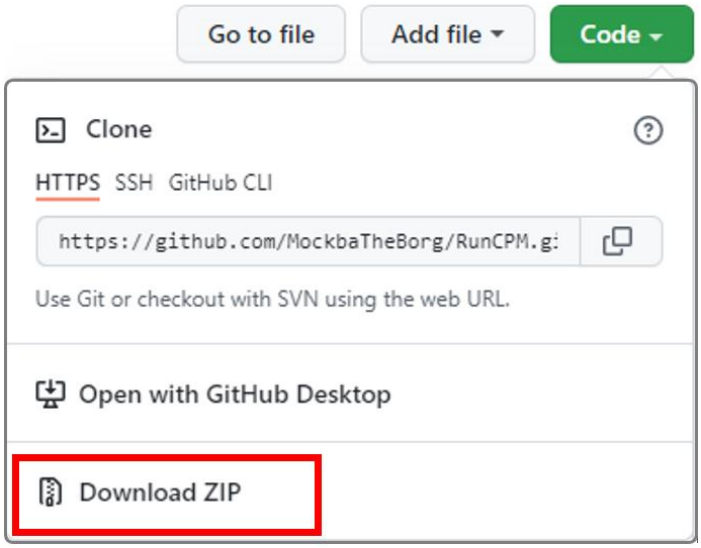


Abbildung 6 Der Download des Repositories

Nach dem Entpacken befinden sich dort sehr viele Informationen und es sollte die Datei *readme.md* auf jeden Fall studiert werden.

Für die Arduino-Entwicklungsumgebung ist das Verzeichnis *RunCPM* wichtig, das man sich in den Arduino-Sketch-Ordner kopiert. Nach dem Öffnen der Datei *RunCPM.ino* werden mehrere Dateien in die Entwicklungsumgebung geladen, was an den zahlreichen Reitern am oberen Rand zu erkennen ist.

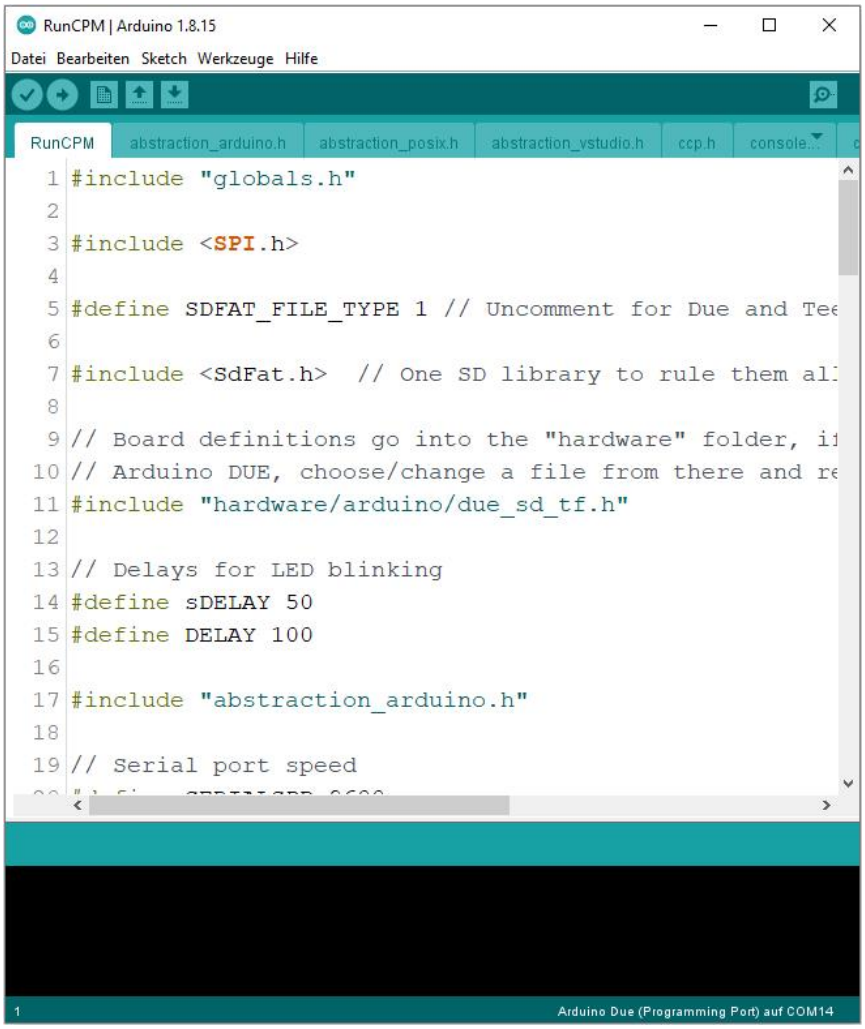
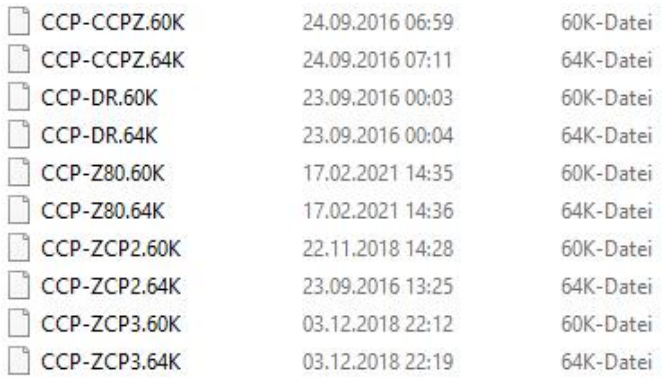


Abbildung 7 Die Arduino-IDE mit den vielen benötigten Dateien

Ich schlage vor, an dieser Stelle noch nicht den Sketch hochzuladen, denn es ist ja noch das eigentliche CP/M-Betriebssystem erforderlich, das mit einigen Programmen und Tools auf eine SD-Karte in einer bestimmten Form abgelegt werden muss.

Die CP/M-Software

Damit der CP/M-Rechner auch richtig booten kann, ist natürlich das Betriebssystem erforderlich. Auf der SD-Karte müssen sich die folgenden Dateien im Root-Verzeichnis befinden.



CCP-CCPZ.60K	24.09.2016 06:59	60K-Datei
CCP-CCPZ.64K	24.09.2016 07:11	64K-Datei
CCP-DR.60K	23.09.2016 00:03	60K-Datei
CCP-DR.64K	23.09.2016 00:04	64K-Datei
CCP-Z80.60K	17.02.2021 14:35	60K-Datei
CCP-Z80.64K	17.02.2021 14:36	64K-Datei
CCP-ZCP2.60K	22.11.2018 14:28	60K-Datei
CCP-ZCP2.64K	23.09.2016 13:25	64K-Datei
CCP-ZCP3.60K	03.12.2018 22:12	60K-Datei
CCP-ZCP3.64K	03.12.2018 22:19	64K-Datei

Abbildung 8 Die CPP-Files

Es ist zu sehen, dass hier die CCP-Binärdateien für 64K und 60K vorhanden sind. Wenn Sie eine SD-Karte verwenden, müssen sich RunCPM und seine CCPs im Stammverzeichnis der SD-Karte befinden. Die 64K-Version der CCPs stellt CP/M-Anwendungen die maximal mögliche Speichermenge zur Verfügung, aber ihre Adressierungsbereiche sind unrealistisch, wenn es um die Emulation eines echten CP/M 2.2-Computers geht. Die 60K-Version der CCPs bietet einen realistischeren Adressierungsbereich, da der CCP-Einstiegspunkt auf der gleichen Ladeadresse liegt wie auf einem physischen CP/M-Computer. Es können jedoch auch andere Speichergrößen verwendet werden, was aber die Neuerstellung der CCP-Binärdateien erforderlich macht (die Quellen befinden sich auf Diskette *A.ZIP*). Die CCP-Binärdateien sind so benannt, dass ihre Dateinamenerweiterungen mit der Speichergröße übereinstimmen, auf der sie laufen. So würde z.B. das CCP von DRI, das auf 60K Speicher läuft, CCP-DR.60K heißen. RunCPM sucht die Datei dementsprechend, je nachdem, welche Speichergröße beim Erstellen ausgewählt wurde.

RunCPM emuliert die CP/M-Disketten und -Benutzerbereiche mit Hilfe von Unterordnern unter der ausführbaren RunCPM-Datei. Um einen Ordner oder eine SD-Karte für die Ausführung von RunCPM vorzubereiten, sind folgende Schritte erforderlich

- Anlegen von mehreren Unterverzeichnissen im Root-Verzeichnis der SD-Karte mit den Namen „A“, „B“, „C“, „D“, usw. die jeweils ein physikalisches Laufwerk repräsentieren (Sowohl Ordnernamen, als auch Dateinamen sollten aus Kompatibilitätsgründen in Großbuchstaben gehalten werden!)
- Anlegen eines Unterordners unterhalb von A mit dem Namen „0“. Er repräsentiert den Benutzerbereich 0 von Laufwerk A. Das sollte auch für alle anderen Laufwerke erfolgen.
- Wenn innerhalb von CP/M zu einem anderen Benutzerbereich gewechselt wird, werden automatisch die entsprechenden Unterordner für die Benutzerbereiche "1", "2", "3", usw. angelegt, sobald sie ausgewählt werden.
- Der Inhalt der entpackten Datei *A.ZIP* (befindet sich im heruntergeladenen Github-Reporitory unterhalb des *DISK*-Ordners) muss in den Unterordner "0" des Laufwerks A kopiert werden.
- CP/M unterstützte nur 16 Laufwerke von A: bis P:, so dass das Anlegen anderer Buchstaben oberhalb von P nicht funktioniert

Um die SD-Karte direkt mit vielen Programmen zu versehen, kann die Struktur verwendet werden, die unter dem folgenden Link zu finden ist.

	Hyperlink!
https://drive.google.com/drive/folders/11WIu8rD_7pIDaET7dqTeA73CvX0jkxz2	

Der Arduino-Sketch-Upload

Ist alles so weit vorbereitet, kann der RunCPM-Sketch auf den Arduino DUE hochgeladen werden. Nach Erfolgt zeigt sich das wie folgt im Status-Fenster der Arduino-Entwicklungsumgebung.

```
Hochladen abgeschlossen.
Der Sketch verwendet 80920 Bytes (15%) des Programmspeicherplatzes. Das Maximum sind 524288 Bytes.
Atmel SMART device 0x285e0a60 found
Erase flash
done in 0.031 seconds

Write 83264 bytes to flash (326 pages)
[==============================] 100% (326/326 pages)
done in 16.081 seconds

Verify 83264 bytes of flash
[==============================] 100% (326/326 pages)
Verify successful
done in 15.055 seconds
Set boot flash true
CPU reset.
```

Abbildung 9 Der RunCPM-Upload-Erfolg

Nun kann der CP/M-Rechner endlich in Erscheinung treten. Doch eine Kleinigkeit muss natürlich noch gemacht werden.

Der Terminal-Zugriff

Um letztendlich auf den CP/M-Rechner zugreifen zu können, ist ein terminal-Programm erforderlich. Ich nutze dafür *Terra Term*, das kostenlos unter der folgenden Internetadresse zu finden ist.

	Hyperlink!
https://tera-term.de.softonic.com/	

Die folgenden Einstellungen hinsichtlich der seriellen Schnittstelle müssen vorgenommen werden.

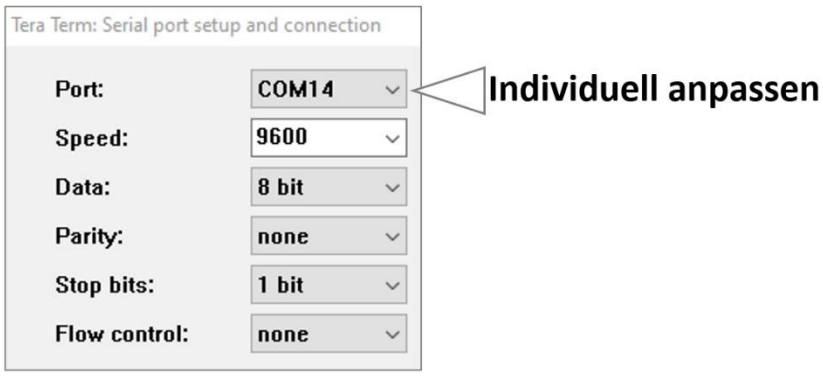


Abbildung 10 Serial-Port-Setup

Für meine Belange habe ich noch Font-Größe und Farbe ein wenig angepasst. Nach der erfolgten Verbindungsaufnahme zeigt sich das Terminal-Fenster wie folgt.

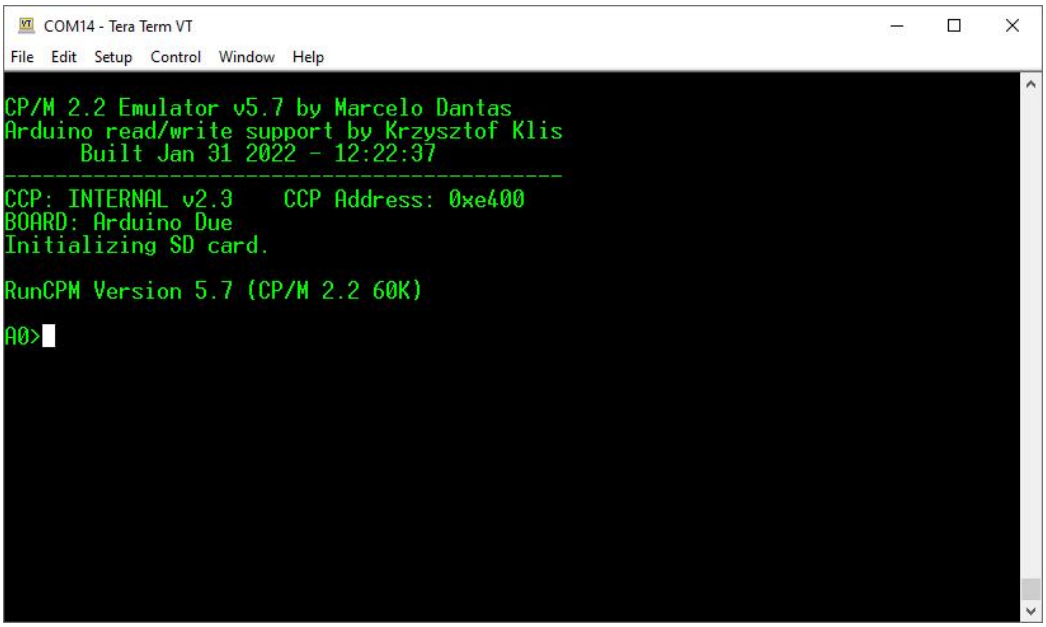


Abbildung 11 Der CP/M-Rechner im Terminal-Fenster

Frei Literatur im Netz

Im Internet gibt es Unmengen an freier Literatur, die sich mit dem Z80 und CP/M befassen. Nachfolgend liste ich einige Links dazu auf.

Z80-CPU

https://www.zilog.com/docs/z80/um0080.pdf

http://www.cpcwiki.eu/imgs/9/9f/Das_Z80-Buch.pdf

http://oldcomputers-ddns.org/public/pub/manuals/zaks_programmierung_des_z80_bw.pdf

http://www.bitsavers.org/components/zilog/z80/03-0029-01_Z80_CPU_Technical_Manual_1977.pdf

https://feldtmann.ddns.net/rc2014/doc/Z80%20Assembly%20Language%20Subroutines.pdf

http://www.fundus.org/pdf.asp?ID=6378

CP/M

http://www.cpm.z80.de/manuals/cpm22-m.pdf

http://www.cpm.z80.de/randyfiles/DRI/ASM.pdf

http://bitsavers.informatik.uni-stuttgart.de/pdf/digitalResearch/cpm/CPM_Assembly_Language_Programming_1983.pdf

http://www.cpcwiki.eu/imgs/5/5d/CPM-Handbuch.pdf

https://mirrors.apple2.org.za/ftp.apple.asimov.net/documentation/programming/cpm/Programmers%20CPM%20Handbook%20by%20Andy%20Johnson-Laird.pdf

MBASIC

http://www.bitsavers.org/pdf/dec/terminal/vt180/AA-P226A-TV_BASIC-80_Reference_Manual_VT180_V5.21_1981.pdf

http://www.msxarchive.nl/pub/msx/mirrors/msx2.com/sources/mbasic.pdf

http://www.cpm.z80.de/manuals/microsoft_softcard_volume_2.pdf

C-Programmierung

https://openbook.rheinwerk-verlag.de/c_von_a_bis_z/

Internetseiten

https://www.z80cpu.eu/mirrors/www.z80.info/index.htm

http://www.gaby.de/z80/z80lit.htm

http://www.cpm.z80.de/drilib.html

http://oldcomputers-ddns.org/public/pub/manuals/

http://www.cpm.z80.de/develop.htm

http://www.z80.eu/c-compiler.html

http://www.retroarchive.org/

https://www.ndr-nkc.de/compo/doku/books.htm

https://colorcomputerarchive.com/repo/Documents/Books/

Abschließend

Ich hoffe, ich konnte ein wenig bei der Umsetzung eines CP/M-Rechners auf einem Mikrocontroller behilflich sein und wünsche viel Spaß damit!

Weitere Informationen sind auf meiner Internetseite zu finden.

Frohes Frickeln!

Erik Bartmann