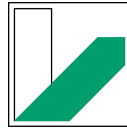


# PIC Programmierung mit PICkit 2 unter Linux

Von Manuel Lippert für den Kurs: "Prozessrechner und Elektronik der Universität Bayreuth

4. Januar 2022



## 1 Einleitung

Die Programmierung eines PIC-Microchip unter Linux hat sich für mich als Linux-Nutzer damals zu einer kleinen Herausforderung entwickelt, weswegen ich versucht habe eine möglichst nahe Programmierung über den Terminal zu suchen. Diese Methode hat den Vorteil, dass man mit einem Editor seiner Wahl Assembler-Code schreiben kann und ihn einfach über Commands im Terminal ausführen kann.

In diesen kleinen Anleitung erkläre ich, wie man dieses Verfahren unter Linux (Ubuntu) aufsetzen lässt. In einem optionalen Schritt zeige ich noch weiterhin, wie man über Visual Studio Code seine Assembler Programmierung aufsetzen kann.

Bei Fragen stehe ich Ihnen gerne über GitHub zu Verfügung. Schreiben sie einfach ein Issue zu Ihrem Problem.

## 2 Compiler

Zuallererst benötigen wir einen Compiler welcher uns `.asm`-Files in `.hex`-Files kompiliert. Als Compiler benutzen wir das Packages `gputils`, welches über den folgenden Befehl installiert werden kann:

```
sudo apt install gputils
```

Damit ist man nun in der Lage mithilfe von `gpasm` über den Terminal `.asm`-Files zu kompilieren. Bei einem gegeben File `foo.asm` wäre dann der Befehl im Terminal:

```
gpasm foo.asm
```

Dabei entstehen 3 weitere Files `foo.cod`, `foo.hex` (Dieses File wird benötigt) und `foo.lst`. Damit `foo.hex`-File auf den PIC-Microchip geladen werden kann wird das folgende Tool benötigt.

## 3 pk2cmd

Das pk2cmd Commandline-Tool ermöglicht Linux (Es gibt auch eine Mac und Windows Version) mit dem PICkit 2 PIC-Microchip über den Terminal anzusteuern. Hier wird es dafür benötigt um den compilierten Assembler-Code auf den PIC-Microchip zu laden.

### 3.1 Installation

Quelle: <https://www.making-sound.co.uk/tech-notes/pickit2-linux.html>

- 1) Zuerst ladet man die nötigen Pakete um pk2cmd zu installieren. Das erfolgt über diesen Befehl im Terminal:

```
sudo apt install build-essential libusb-dev
```

- 2) Lade pk2cmd von GitHub und entpacke Sie den Download in Ihrem Download Ordner. Öffnen Sie dann den Terminal und navigieren zum diesem Ordner-Verzeichnis.

```
cd ~/Downloads/pk2cmd/pk2cmd
```

Alternative mit git über Terminal herunterladen und in das Verzeichnis navigieren:

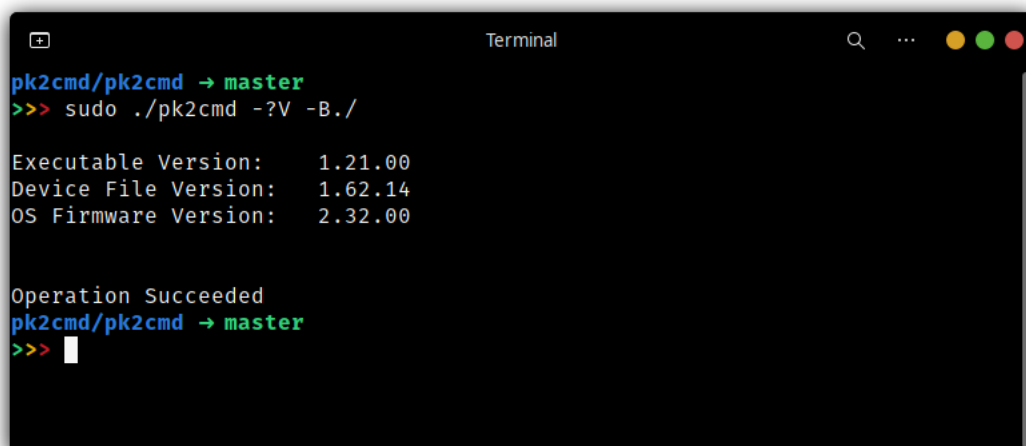
```
git clone https://github.com/psmay/pk2cmd.git  
cd pk2cmd/pk2cmd
```

- 3) Zum installieren führen Sie folgende Befehle aus:

```
make linux
```

- 4) Zum Testen von pk2cmd führen Sie diesen Command im selben Ordner aus:

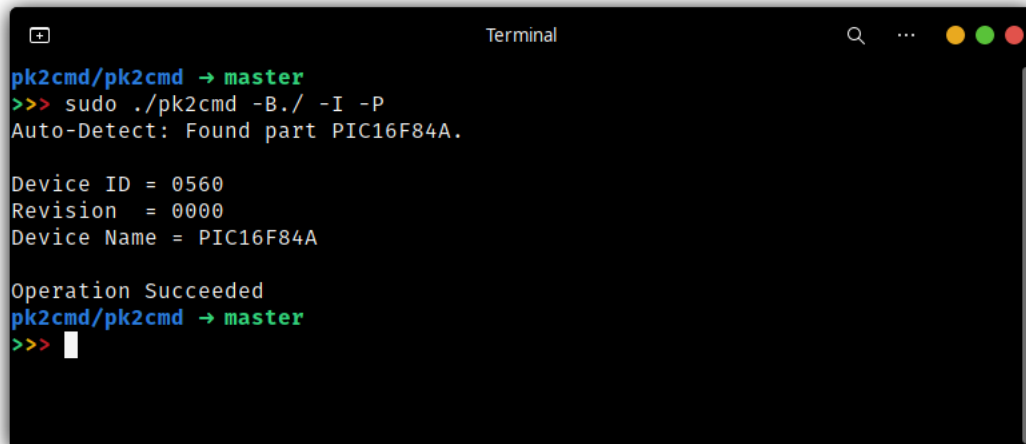
```
sudo ./pk2cmd -?V -B./
```



```
pk2cmd/pk2cmd → master  
>>> sudo ./pk2cmd -?V -B./  
  
Executable Version:    1.21.00  
Device File Version:   1.62.14  
OS Firmware Version:   2.32.00  
  
Operation Succeeded  
pk2cmd/pk2cmd → master  
>>> 
```

- 5) Zum Testen ob der PIC-Microchip von pk2cmd angesteuert wird kann dieser Command im selben Ordner ausgeführt werden:

```
sudo ./pk2cmd -B./ -I -P
```



```
pk2cmd/pk2cmd → master
>>> sudo ./pk2cmd -B./ -I -P
Auto-Detect: Found part PIC16F84A.

Device ID = 0560
Revision = 0000
Device Name = PIC16F84A

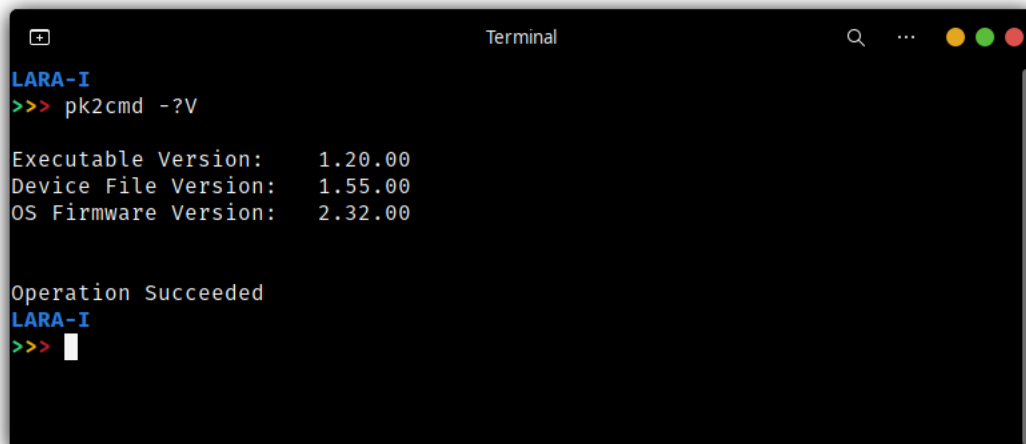
Operation Succeeded
pk2cmd/pk2cmd → master
>>> 
```

- 6) Nun wird der Command „pk2cmd“ global zugänglich gemacht mit diesem Befehl, welcher wieder im selben Ordner ausgeführt werden muss:

```
echo 'export PATH="$PATH:/usr/share/pk2"' >> ~/.bashrc
```

Danach kann man den Ordner verlassen und den globalen Befehl wie folgt testen:

```
cd
pk2cmd -?V
```



```
LARA-I
>>> pk2cmd -?V

Executable Version: 1.20.00
Device File Version: 1.55.00
OS Firmware Version: 2.32.00

Operation Succeeded
LARA-I
>>> 
```

Bei Problemen ist in der Quelle ein Troubleshooting aufgeführt, welches Ihnen bestimmt weiter helfen wird. Dies wird aber hier bewusst weggelassen, da ab nun alles funktionieren sollte.

## 3.2 Befehlspalette

Quelle: <https://github.com/kvadevack/pk2cmd/blob/master/pk2cmd/release/Readme%20For%20PK2CMD.txt>

### PICkit 2 COMMAND LINE HELP

Options	Description	Default
A<value>	Set Vdd voltage	Device Specific
B<path>	Specify the path to PK2DeviceFile.dat	Searches PATH and calling dir
C	Blank Check Device	No Blank Check
D<file>	OS Download	None
E	Erase Flash Device	Do Not Erase
F<file>	Hex File Selection	None
G<Type><range/path>	Read functions Type F: = <b>read</b> into hex file, path = full file path, range is not used Types P,E,I,C: = output <b>read</b> of Program, EEPROM, ID and/or Configuration Memory to the screen. P and E must be followed by an address range <b>in</b> the form of x-y where x is the start address and y is the end address both <b>in</b> hex, path is not used (Serial EEPROM memory is 'P')	None
H<value>	Delay before Exit K = Wait on keypress before <b>exit</b> 1 to 9 = Wait <value> seconds before <b>exit</b>	Exit immediately
I	Display Device ID & silicon revision	Do Not Display
J<newlines>	Display operation percent <b>complete</b> N = Each update on newline	Rotating slash
K	Display Hex File Checksum	Do Not Display
L<rate>	Set programming speed <rate> is a value of 1-16, with 1 being the fastest.	Fastest
M<memory region>	Program Device memory regions: P = Program memory E = EEPROM I = ID memory C = Configuration memory If no region is entered, the entire device will be erased & programmed. If a region is entered, no erase is performed and only the given region is programmed. All programmed regions are verified. (serial EEPROM memory is 'P')	Do Not Program

N<string>	Assign Unit ID string to first found PICkit 2 unit. String is limited to 14 characters maximum. May not be used with other options. Example: -NLab1B	None
P<part>	Part Selection. Example: -PPIC16f887	(Required)
P	Auto-Detect <b>in</b> all detectable families	
PF	List auto-detectable part families	
PF<id>	Auto-Detect only within the given part family, using the ID listed with -PF Example: -PF2	
Q	Disable PE <b>for</b> PIC24/dsPIC33 devices	Use PE
R	Release /MCLR after operations	Assert /MCLR
S<string/#>	<b>Use the PICkit 2 with the given Unit ID</b> string. Useful when multiple PICkit 2 units are connected. Example: -SLab1B If no <string> is entered, <b>then</b> the Unit IDs of all connected units will be displayed. In this <b>case</b> , all other options are ignored. <b>-S# will list units</b> with their firmware versions. See <b>help</b> -s? <b>for</b> more info.	<b>First found unit</b>
T	Power Target after operations	Vdd off
U<value>	Program OSCCAL memory, where: <value> is a hexadecimal number representing the OSCCAL value to be programmed. This may only be used <b>in</b> conjunction with a programming operation.	Do Not Program
V<value>	Vpp override	Device Specific
W	Externally power target	Power from Pk2
X	Use VPP first Program Entry Method	VDD first
Y<memory region>	Verify Device P = Program memory E = EEPROM I = ID memory C = Configuration memory If no region is entered, the entire device will be verified. (Serial EEPROM memory is 'P')	Do Not Verify
Z	Preserve EEData on Program	Do Not Preserve
?	Help Screen	Not Shown

Each option must be immediately preceded by a switch, Which can be either a dash <-> or a slash </> and options must be separated by a single space.

```
Example:  PK2CMD  /PPIC16F887  /Fc:\mycode  /M
           or
           PK2CMD  -PPIC16F887  -Fc:\mycode  -M
```

Any option immediately followed by a question mark will invoke a more detailed description of how to use that option.

Commands and their parameters are not **case** sensitive. Commands will be processed according to **command** order of precedence, not the order **in** which they appear on the **command** line.

Precedence:

```

-?          (first)
-B
-S
-D
-N
-P
-A -F -J -L -Q -V -W -X -Z
-C
-U
-E
-M
-Y
-G
-I -K
-R -T
-H          (last)

```

The program will **return** an **exit** code upon completion which will indicate either successful completion, or describe the reason **for** failure. To view the list of **exit** codes and their descriptions, **type** **-?E** on the **command** line.

type -?V on the command line for version information.

type -?L on the command line for license information.

type `-?P` on the `command` line for a listing of supported devices.

```
type -?P<string> to search for and display a list of supported devices
beginning with <string>.
```

---

### 3.3 Verwendung

Um nun ein gegebenes `foo.hex`-File auf einen PIC-Microchip in unseren Fall der PIC16F84A zu laden wird folgender Befehl im Terminal ausgeführt:

```
pk2cmd -A5 -PPIC16F84A -F foo.hex -M -T -R
```

Nach dem ausüben dieses Befehls sollte der Microchip das Programm ausführen. Was die jeweiligen Attribute des Befehls festlegen, kann im Kapitel 3.2 nachgelesen werden. Ab hier können Sie nun über einen Editor Ihrer Wahl `.asm`-File erstellen mit `gpasm` kompilieren und mit `pk2cmd` auf den PIC-Microchip laden.

## 4 Visual Studio Code