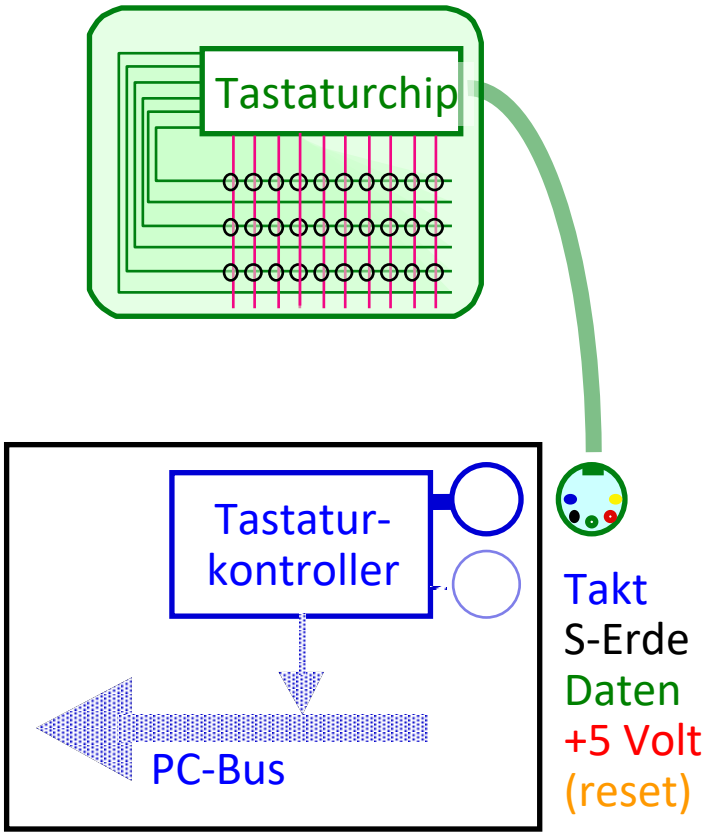
Tastatur-Programmierung

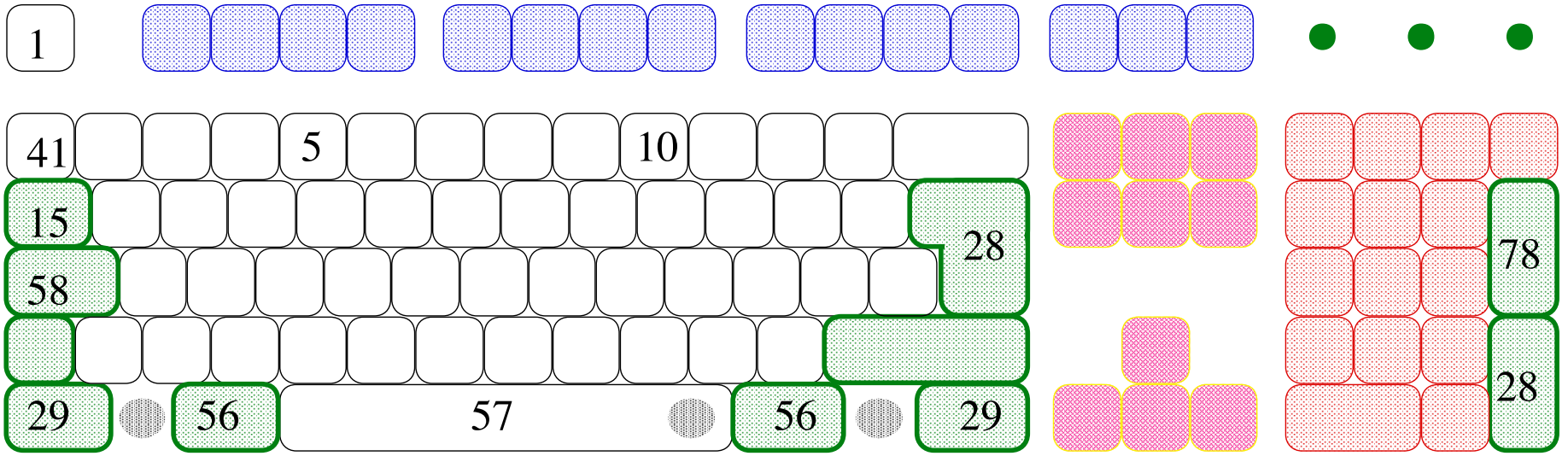
Komponenten

- Tastaturchip in der Tastatur:
  - Serielle Kommunikation mit dem Tastaturkontroller
  - Abfrage der Tastatur über Scanmatrix
  - (Puffer & Entprellen der Tasten)
- Tastaturkontroller auf der Hauptplatine:
  - Kontroller heute in South Bridge integriert.
  - Zusatzeingang für Maus bei den PS/2 Modellen.
  - IRQ1, Ports 0x60 und 0x64
- Moderner PC: USB-Tastatur
  - USB Legacy Support: Ansteuerung funktioniert immer noch zusätzlich über den Tastaturcontroller (Rückwärtskompatibilität)



Tastenkodierung

- Tastaturen liefern nur "Scancodes“= Nummer der Taste (7 Bit)
- Treiber übersetzt Scancode in Zeichen (gemäß konfig. Alphabet).
- Dauerumschaltung durch Treiber realisiert: Caps-Lock, Num-Lock.
- Beispiel MF II - Tastatur:



Beispiel: Scancodes & ASCII-Codes

- Beispiel: Scancodes & ASCII-Codes

Darstellung im Programm (und im CGA-Speicher!):
Zeichencodes (ASCII)

Zeichen	ASCII-Code
(	40
0	48
1	49
2	50
A	65
B	66
a	97

Darstellung in der Hardware:
Tastencodes

Taste	Scan-Code
A	30
a	30
S	31
D	32
Cursor hoch	72
Cursor runter	80

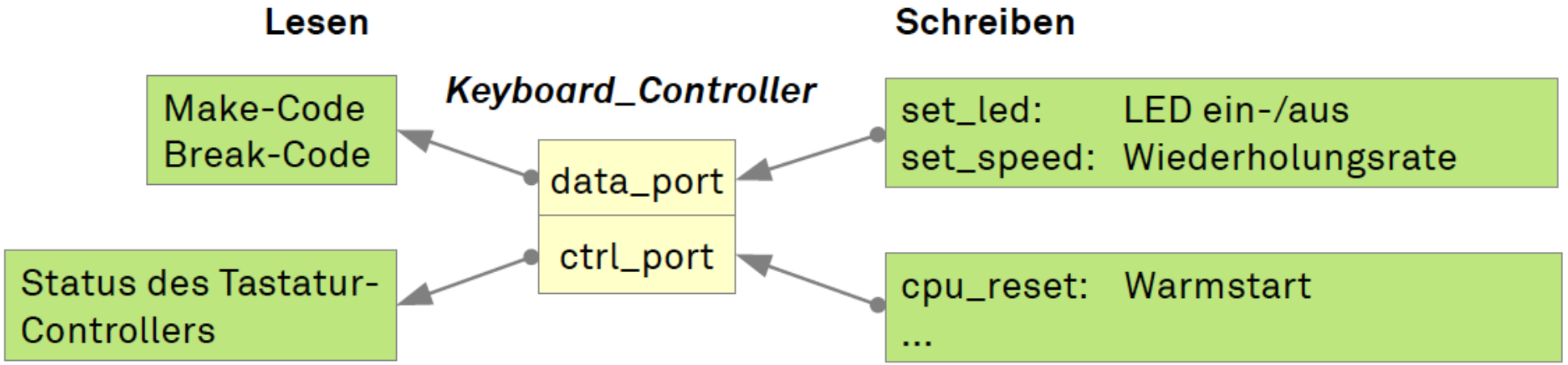
- Tastaturcontroller sendet zusätzliche Informationen
  - Make-Code beim Drücken / Halten einer Taste
  - Break-Code beim Loslassen

Make- und Break-Codes

- Üblicherweise gilt
  - Make-Code (Taste gedrückt) = Scan-Code
  - Break-Code (Taste losgelassen) = Scan-Code + 128 (Bit 7)
- Einige Tasten senden jedoch mehrere Codes
  - z.B. Funktionstasten (F1-F12)
  - bis zu drei Make/Break-Codes pro Taste
- Eingebaute Wiederholungsfunktion
  - Hardware sendet zusätzliche Make-Codes, wenn eine Taste länger gehalten wird
- Dekodierung ist mühsam
  - hhuTOS entält fertige Funktion: `key_decoded()`

Datenaustausch mit der Tastatur

- Tastaturcontroller wird über zwei E/A-Ports
  - Ein-/Ausgabe-Register (data_port) 0x60
  - Status-/Steuer-Register (ctrl_port) 0x64



Status-Register der Tastatur

- Zugriff durch Lesen von Port 0x64
  - Bit-7: PARE Paritäts-Error (Bit#7)
  - Bit-6: TIM Zeitüberschreitung für Antwort
  - **Bit-5: AUXB Byte von Zusatzeinheit (Maus) abholen bitte**
  - Bit-4: KEYL Tastatur frei, Tastatur gesperrt (1,0)
  - Bit-3: C/D Befehlsbyte oder Datenbyte geschrieben
  - Bit-2: SYSF Selbsttest erfolgreich
  - **Bit-1: INPB Eingabepuffer beschäftigt, bitte warten**
  - **Bit-0: OUTB Byte von Ausgabepuffer abholen bitte**

Steuer-Register der Tastatur

- Schreiben in Port 0x64
  - Befehle an den Tastatur-Kontroller (je 1 Byte):
    - 0xAA: Selbsttest
    - 0xAE: Aktivieren der Tastatur
    - 0xED: LEDs an/ausschalten
    - 0xF3: Wiederholrate einstellen
  - Achtung: Port 0x64 erst schreiben, wenn der Eingabepuffer leer ist
    → Bit-1 INP im Statusregister prüfen

- Eine Übersicht über die Befehle gibt es hier:
  https://wiki.osdev.org/PS/2_Keyboard

Aktive Tastaturabfrage

- In einer Endlos-Schleife folgende Befehle ausführen:
  - In einer Endlos-Schleife warten, bis ein Byte von der Tastatur abholbereit ist → Prüfen von Bit-0 `OUTB` im Status-Register
  - Byte vom Daten-Port einlesen
  - Falls dieses Byte nicht von der PS/2-Maus ist (kann man am `AUXB` Bit im Status-Register) erkennen, dann `key_decode()` aufrufen
  - `key_decode()` gibt `true` zurück, fall ein Zeichen komplett dekodiert wurde. Dann soll die Funktion verlassen werden und `gather` zurückgegeben werden
  - (Da manche Zeichen aus mehreren Bytes bestehen kann es öfters vorkommen, dass mehrere Schleifendurchläufe notwendig sind, bis `key_decode()` ein Zeichen dekodieren kann.)
  - Ansonsten erfolgt der nächste Schleifendurchlauf
- Das Abfragen der Tastatur mithilfe einer Endlos-Schleife ist nicht optimal und wird später im Projekt auf Interrupt-Betrieb umgestellt

Befehle an die Tastatur schicken (LEDs, Wiederholrate)

- Genereller Ablauf:
  - 1. Warten bis der Eingabepuffer leer ist → Bit-1 `INPB` im Status-Register prüfen
  - 2. Befehlsbyte in Data-Port schreiben
  - 3. Warten bis eine Antwort vorliegt → Bit-0 `OUTB` im Status-Register prüfen
  - 4. Antwort vom Data-Port einlesen
  - 5. Falls kein ACK vorliegt (= 0xFA) abbrechen und -1 zurückgeben
  - 6. Sonst Datenbyte in Data-Port schreiben
  - 7. Warten bis eine Antwort vorliegt → Bit-0 im Status-Register prüfen
  - 8. Antwort vom Data-Port einlesen
  - 9. Falls kein ACK vorliegt (= 0xFA) -1 zurückgeben, sonst 0

Tastatur – LED-Ansteuerung

- Aufbau des Befehlsbytes:

Bit	Bedeutung
7	-
6	-
5	-
4	-
3	-
2	Caps Lock
1	Num Lock
0	Scroll Lock

Tastatur – Einstellen der Wiederholrate

- **Typematic-Rate**: legt fest wie schnell der gleiche Scan-Code geschickt wird, wenn eine Taste längere Zeit gedrückt wird
  - Hiermit wird die mehrfache Betätigung der gleichen Taste simuliert
  - Kann zwischen 2 bis 30 Wiederholungen pro Sekunde liegen

- **Delay-Rate**: legt fest nach welcher Zeit-Verzögerung die Wiederholfunktion einsetzen soll (Toleranz ca. 20%)

Code	Delay-Rate
00	1/4 s
01	1/2 s
10	3/4 s
11	1 s

Tastatur – Einstellen der Wiederholrate (2)

- **Typematic-Rate**

Code	WpS*	Code	WpS*	Code	WpS*	Code	WpS*
11111b	2,0	10111b	4,0	01111b	8,0	00111b	16,0
11110b	2,1	10110b	4,3	01110b	8,6	00110b	17,1
11101b	2,3	10101b	4,6	01101b	9,2	00101b	18,5
11100b	2,5	10100b	5,0	01100b	10,0	00100b	20,0
11011b	2,7	10011b	5,5	01011b	10,9	00011b	21,8
11010b	3,0	10010b	6,0	01010b	12,0	00010b	24,0
11001b	3,3	10001b	6,7	01001b	13,3	00001b	26,7
11000b	3,7	10000b	7,5	01000b	15,0	00000b	30,0

* Wiederholungen pro Sekunde

- Die Codes entstehen durch folgende Formel:

$$WpS = 1 / ((8 + A) * 2^B * 0.00417)$$

A = Bits 0, 1, 2 von Code
B = Bits 3, 4 von Code

Tastatur – Einstellen der Wiederholrate (3)

- Aufbau des kompletten Befehlsbytes:

Bit	Bedeutung
7	-
6	Delay-Rate
5	Delay-Rate
4	Typematic-Rate
3	Typematic-Rate
2	Typematic-Rate
1	Typematic-Rate
0	Typematic-Rate

Maus & Tastatur

- Am Keyboard-Controller hängt die PS/2-Tastatur und die PS/2-Maus
- Die Tastatur verwendet IRQ1 und die Maus IRQ12
- Ob von der Maus oder Tastatur Daten anliegen kann im Statusregister abgefragt werden -> AUXB

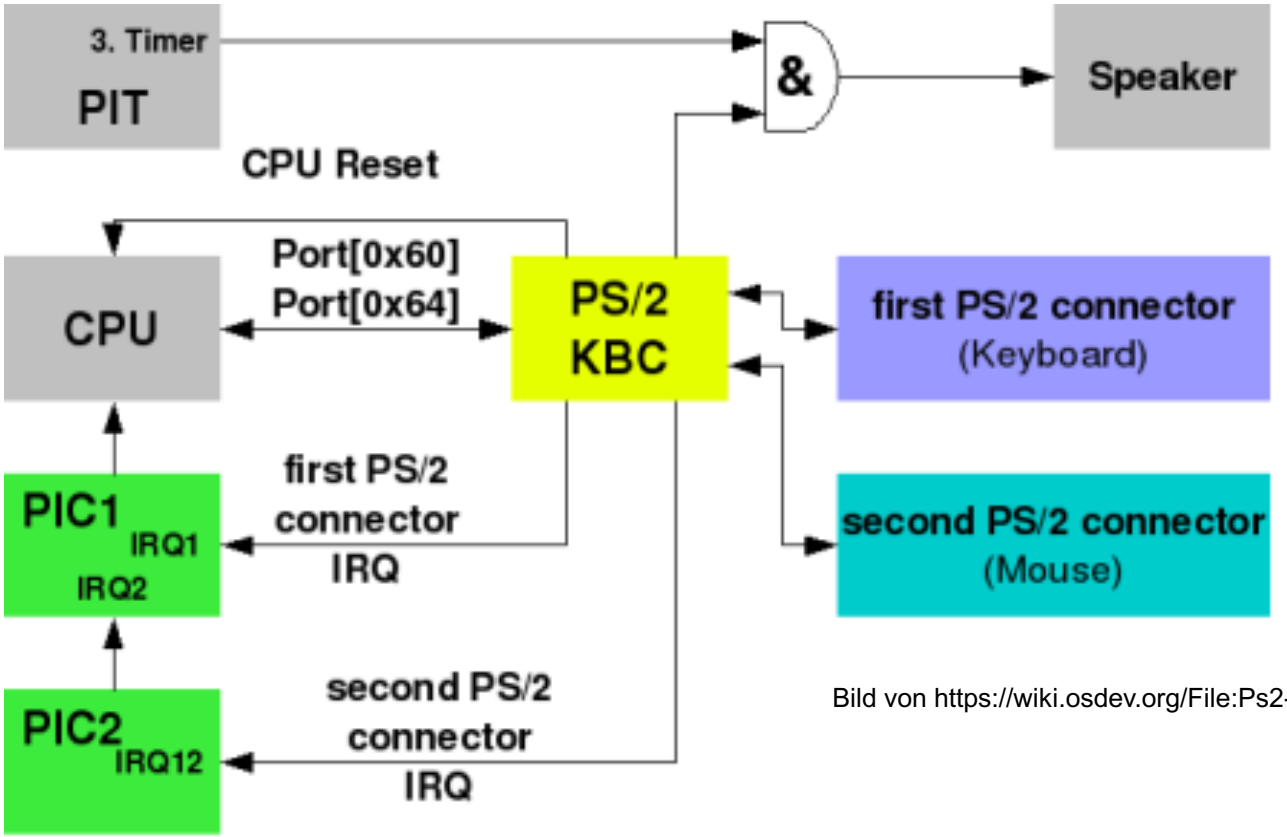


Bild von https://wiki.osdev.org/File:Ps2-kbc.png