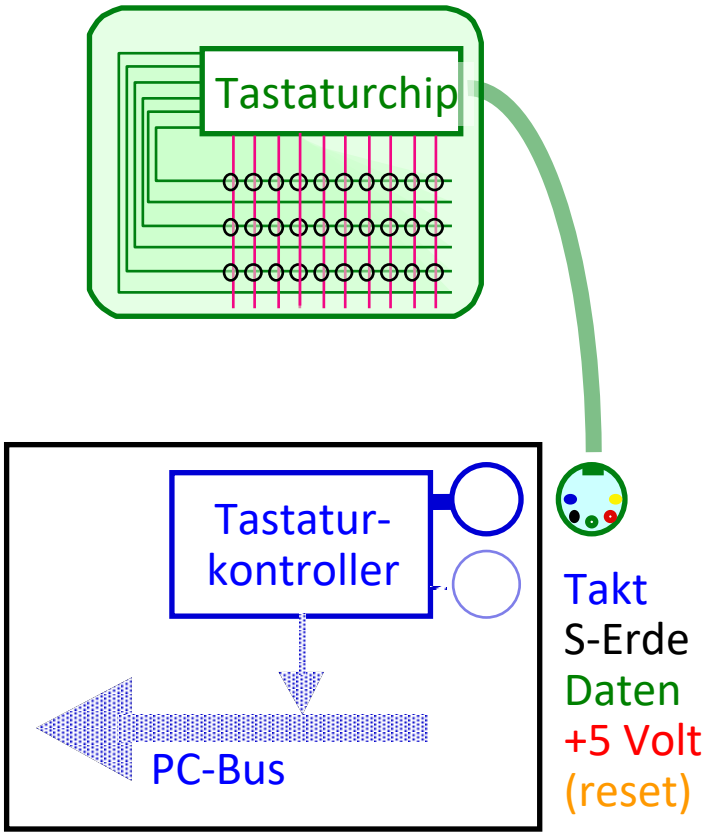
Tastatur-Programmierung

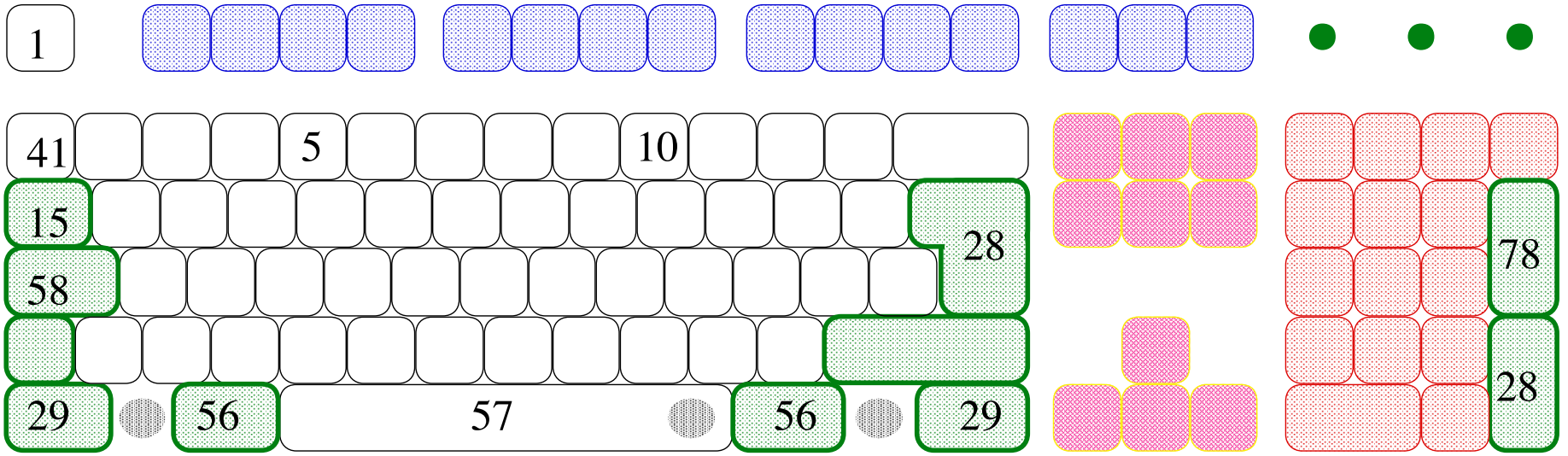
Komponenten

- 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

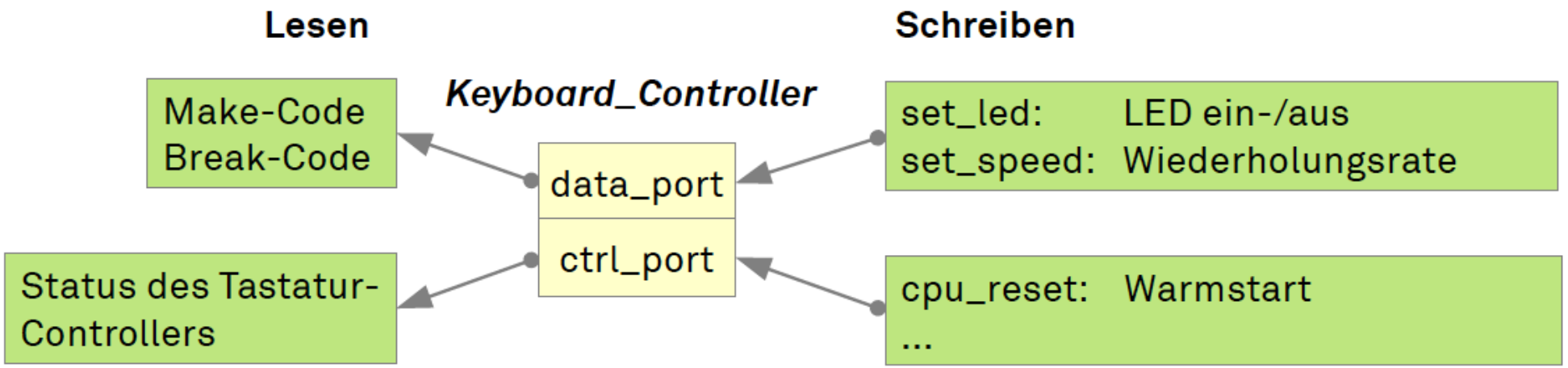
- Tastaturcontroller sendet zusätzliche Informationen
  - Make-Code beim Drücken / Halten einer Taste
  - Break-Code beim Loslassen

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
  - HHUos Vorgabe entält: `bool key_decoded()`

Datenaustausch mit der Tastatur

- Tastaturcontroller wird über zwei E/A-Ports
  - Ein-/Ausgaberegister (data_port) 0x60
  - Status-/Steuerregister (ctrl_port) 0x64



Status-/Steuerregister der Tastatur

- **Statusregister:** Lesen vom Ctrl-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**

- **Steuerregister:** Schreiben in Ctrl-Port 0x64
  - Befehle an den Tastatur-Kontroller (je 1 Byte), siehe auch nächste Seite
    * Befehlsbeispiele: Selbsttest = 0xAA, Aktivieren der Tastatur = 0xAE, …
  - Erst schreiben, wenn der Eingabepuffer leer ist
    → Bit-1 im Statusregister prüfen

Weitere Informationen

- Aktive Tastaturabfrage (nicht durch Unterbrechungen):
  - Warten, bis OUTB im Statusregister gesetzt ist
  - Make-/Break-Code vom Data-Port 0x60 lesen (löscht automatisch OUTB)

- Tastatur programmieren (set_led, set_speed)
  - 1. Befehlsbyte in Data-Port schreiben
    - ❖ Tastatur antwortet mit ack (0xFA), ggfs. auf Antwort warten (s.o.)
  - 2. Datenbyte in Data-Port schreiben (LED-Codes, Repeat-Rate)
    - ❖ Tastatur antwortet mit ack, ggfs. auf Antwort warten

Tastatur - Programmierung

- set_led 0xED, <led_mask> in data_port

Parameter für set_led Befehl: (led_mask)

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

- set_speed 0xF3, <config_byte> in data_port

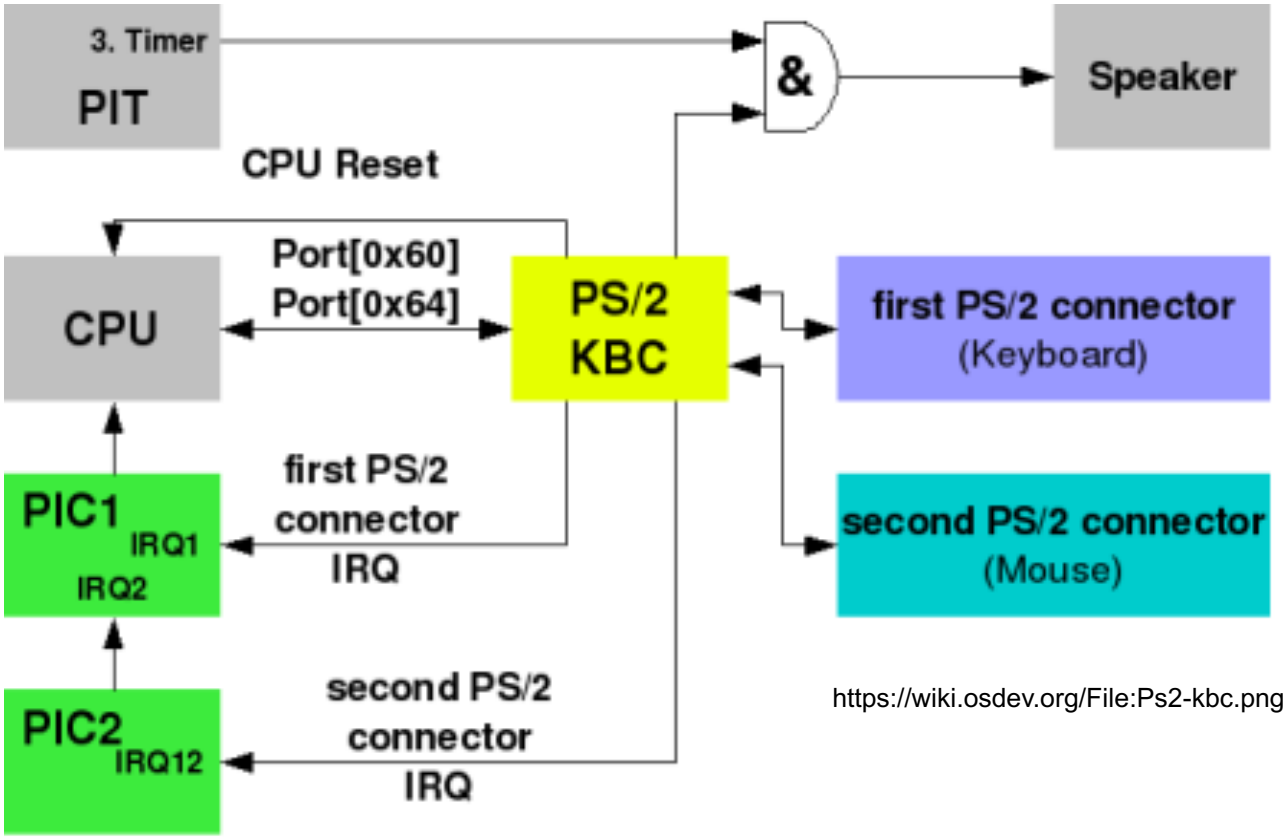
Parameter für set_speed Befehl: (config_byte)

Bits 5 und 6 (hex)	Verzögerung (s)
0x00	0.25
0x01	0.5
0x02	0.75
0x03	1.0

Bits 0-4 (hex)	Wiederholungsrate (cps)
0x00	30
0x02	25
0x04	20
0x08	15
0x0c	10
0x10	7
0x14	5

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



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