

Übung zu Betriebssysteme

Bildschirm (CGA Textmode)

Wintersemester 2021/22

Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

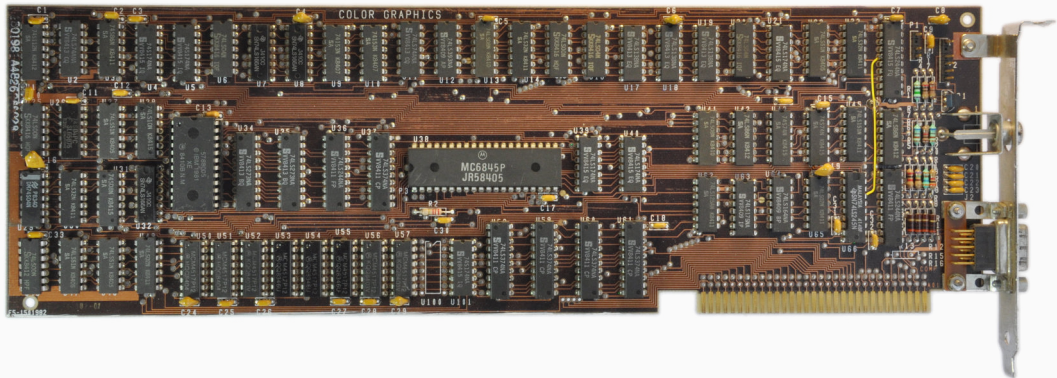


Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



Quelle: Wikipedia

Der CGA Bildschirm ...



... zwei Bytes pro Zeichen ...

1. Byte: ASCII Zeichen

☒☒♥♦♣♠•◻◼◽◾♂♀尸月※►◄↕!!¶§┐┌↑↓→←└┘▲▼
! " # \$ % & ' () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\] ^ _
` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~ ¢
Ç ü é â ã ä å ç ê ë è ì î ï Ä Å É æ ø ò ô û ù ÿ Ö Ü Ç £ ¥ ¤ ¢
á í ó ú ñ Ñ º º ¿ ¸ ½ ¼ ¡ « » ▨ ▩ ▪ ▫ ▬ ▭ ▮ ▯ ▰ ▱ ▲ △
▴ ▵ ▶ ▷ ▸ ▹ ► ▻ ▽ ▾ ▿ ▸ ▹ ► ▻ ▽ ▾ ▿ ▸ ▹ ► ▻ ▽ ▾ ▿
α β γ π Σ σ μ τ ϑ θ Ω δ ω ϕ € Π ≡ ± ≥ ≤ ∫ √ ÷ ≈ ° · √ n ² ■

... zwei Bytes pro Zeichen ...

1. Byte: **ASCII Zeichen**
2. Byte: **Darstellungsattribut**



0	schwarz	4	rot	8	dunkelgrau	12	hellrot
1	blau	5	magenta	9	hellblau	13	hellmagenta
2	grün	6	braun	10	hellgrün	14	gelb
3	cyan	7	hellgrau	11	hellcyan	15	weiß

...eingebildet im Arbeitsspeicher ab 0xb8000

⋮	
'B'	← 0xb8000
0xf4	← 0xb8001
'a'	← 0xb8002
0x74	← 0xb8003
'r'	← 0xb8004
0x74	← 0xb8005
⋮	

Bar

Ansprechen von einzelnen Bits

■ Bitmasken und logischen Bitoperationen

$\&$ Konjunktion (bitweises UND)

$$3 \& 5 = 1$$

$|$ Disjunktion (bitweises ODER)

$$3 | 5 = 7$$

$\wedge$ exklusives ODER

$$3 \wedge 5 = 6$$

$\sim$ Einerkomplement (bitweise Negation)

$$\sim 5 = 250$$

$\ll$ verschieben nach links

$$12 \ll 2 = 48$$

$\gg$ verschieben nach rechts

$$12 \gg 2 = 3$$

$x |= (1 \ll n)$ Setze das n te Bit in x

$x \&= \sim(1 \ll n)$ Lösche das n te Bit in x

Ansprechen von einzelnen Bits

- Bitmasken und logischen Bitoperationen
- Bitfelder in C/C++

```
struct cga_attr {  
    unsigned char fg : 4,  
                  bg : 3,  
                  bl : 1;  
};  
  
struct cga_attr a;  
a.fg = 4; // rot  
a.bg = 7; // hellgrau  
a.bl = 1; // blinken
```

Layout mit GCC auf x86:



Einschub: Füllen von Strukturen

```
struct Test {  
    char foo[3]; // 3 Byte  
    int bar;     // 4 Byte  
    bool baz;    // 1 Byte  
} __attribute__((packed));  
  
static_assert(sizeof(Test) == 8, "Test kaputt");
```

Schreibmarke (Cursor)

Entweder Cursorposition in Software merken...

... oder Cursorposition aus Hardware auslesen

- CGA hat 18 Steuerregister mit je 8 Bit
- Position in Register 14 (high) und 15 (low)
- Nur indirekter Zugriff über Index- (0x3d4) und Datenregister (0x3d5)

0	Horizontal Total	w
1	Horizontal Displayed	w
2	H. Sync Position	w
3	H. Sync Width	w
4	Vertical Total	w
5	V. Total Adjust	w
6	Vertical Displayed	w
7	V. Sync Position	w
8	Interlace Mode	w
9	Max Scan Line Address	w
10	Cursor Start	w
11	Cursor End	w
12	Start Address (high)	w
13	Start Address (low)	w
14	Cursor (high)	rw
15	Cursor (low)	rw
16	Light Pen (high)	r
17	Light Pen (low)	r

0x3d4

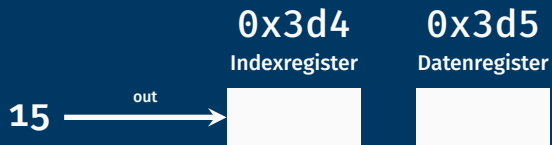
Indexregister



0x3d5

Datenregister





0x3d4
Indexregister

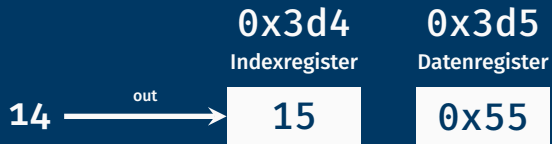
15

0x3d5
Datenregister

0x55

in





0x3d4
Indexregister

14

0x3d5
Datenregister

0x0a

in



Kleiner Exkurs:

Zugriff auf Arbeitsspeicher vs. I/O

