

Programmering av grafisk display

- Arbetsboken avsnitt 5
 - "Drivrutrin för grafisk display", uppg. 5.10-5.16.
- "Datablad LCD – Grafisk" under resurser på kurshemsidan, s.14-23.
- Lab 3
 - Drivrutiner
 - Enkel grafik
- Lab 5
 - Spel



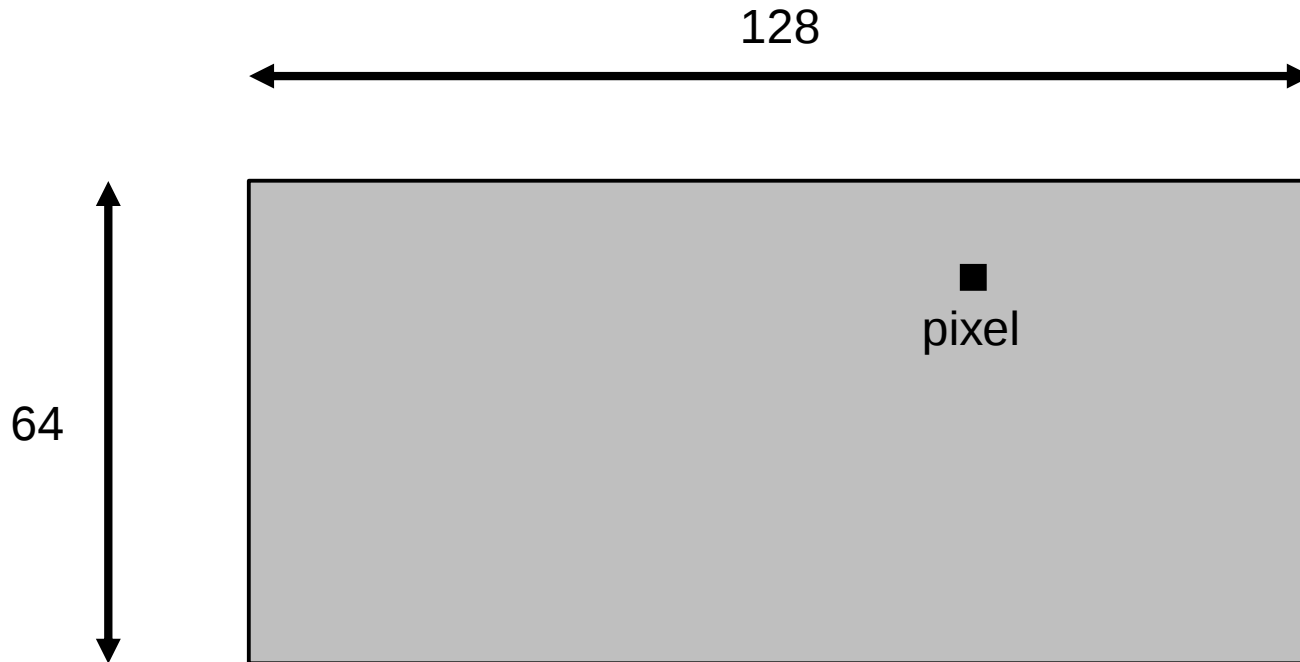
Översikt

- Egenskaper
 - Implementation av drivrutiner
 - Skriva till display
-
- Prestanda
 - Design av grafik



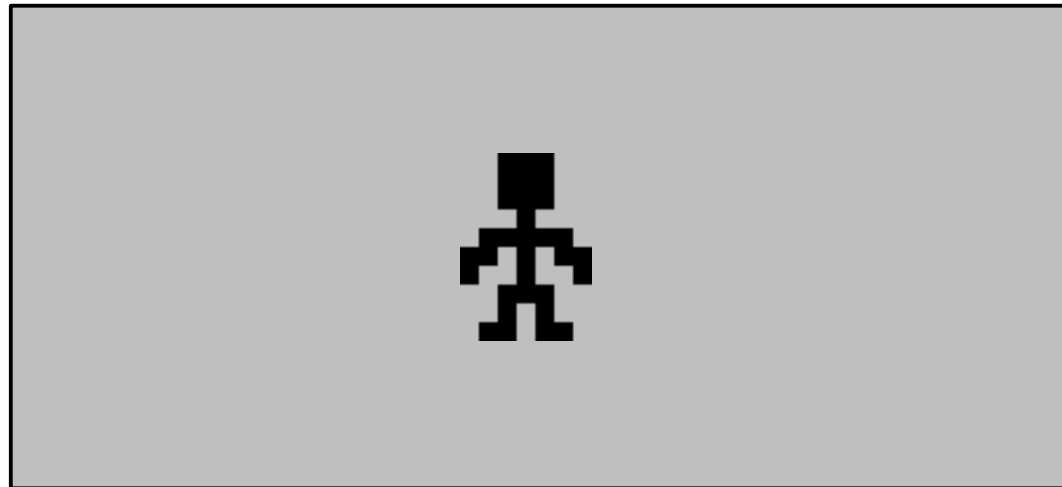
Egenskaper

Upplösning på 128 x 64 pixlar.



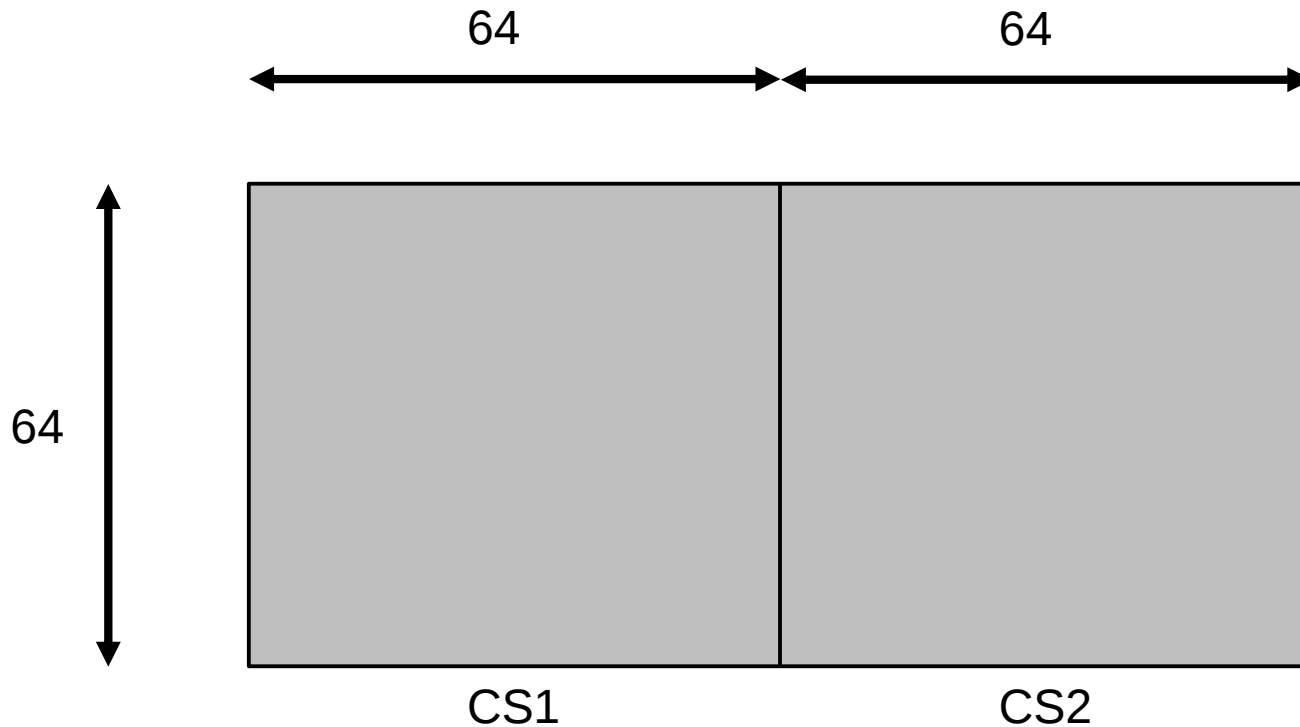
Egenskaper

- Inga färger eller ens gråskala.
- Varje pixel är antingen transparent eller fylld.
- Med andra ord: 1 bit / pixel.



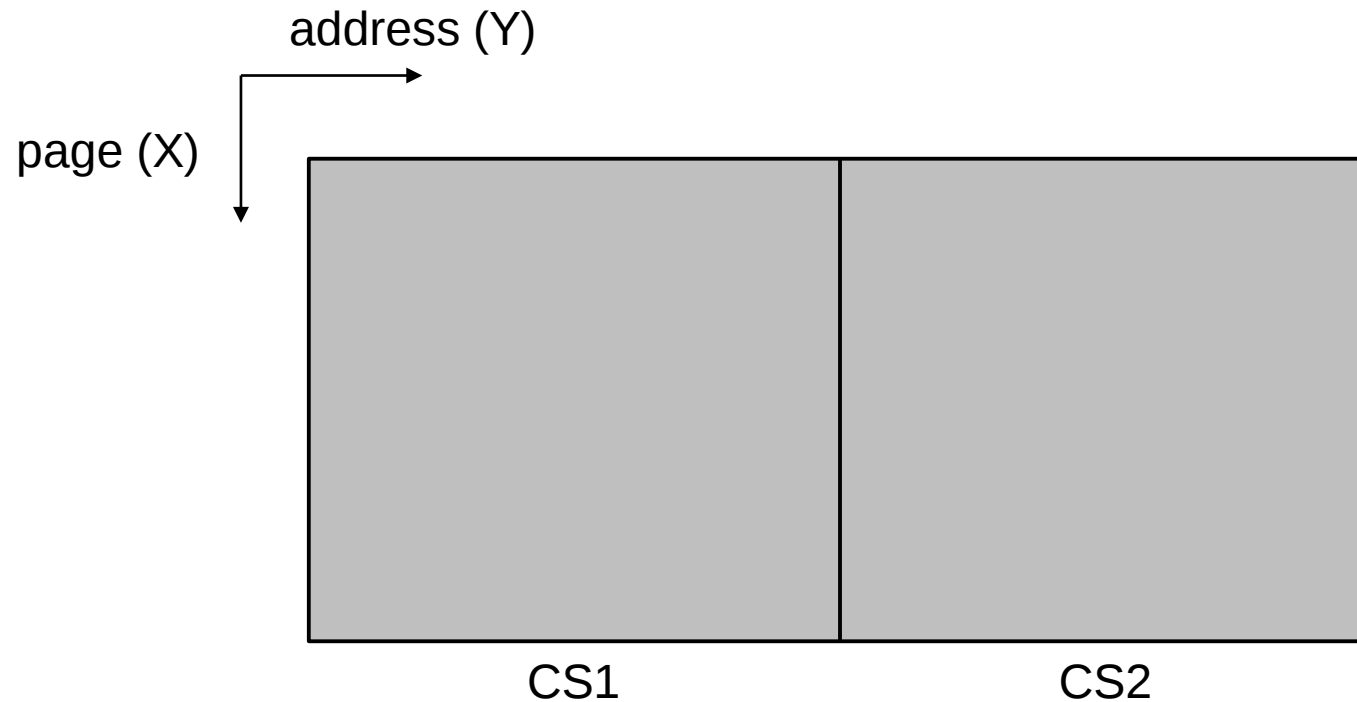
Egenskaper

- Displayen består av två separata enheter.
- CS = "Chip Select".



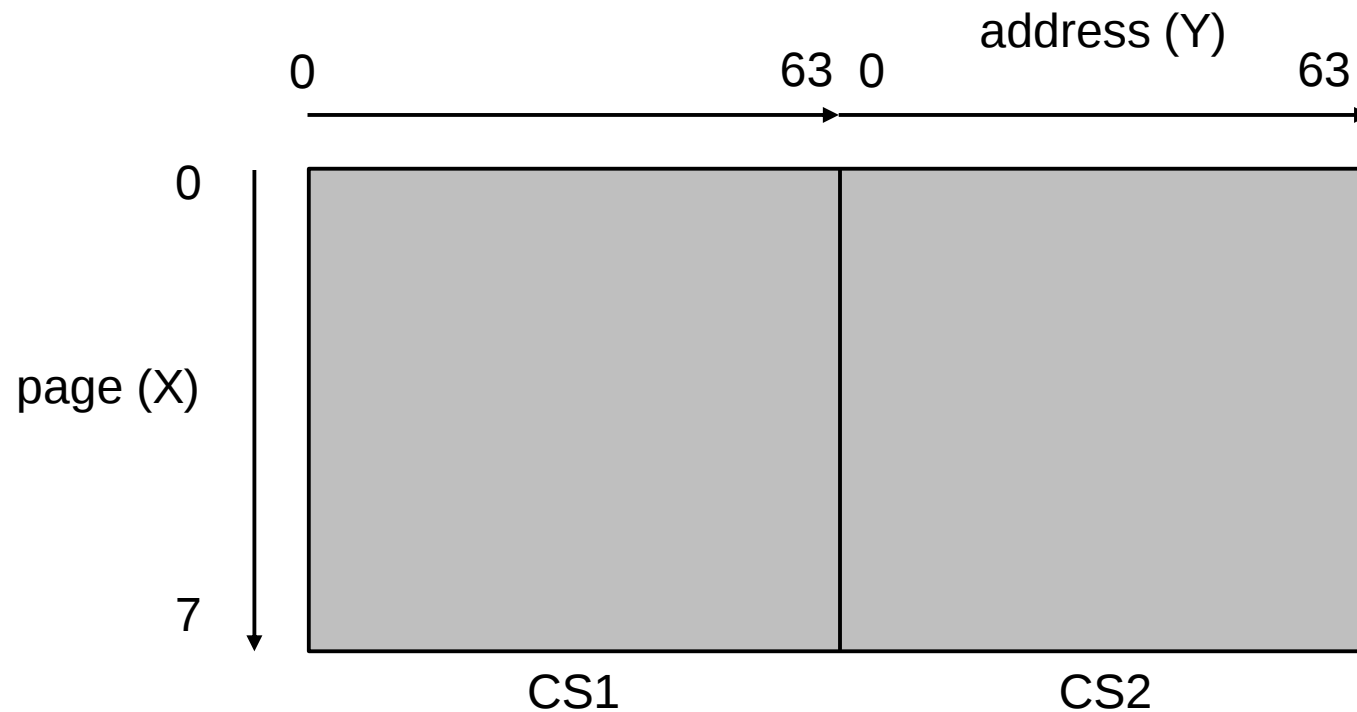
Egenskaper

- Fysiska koordinater för displayen anges i "page" och "address".
- I databladet kallas page X och address Y.



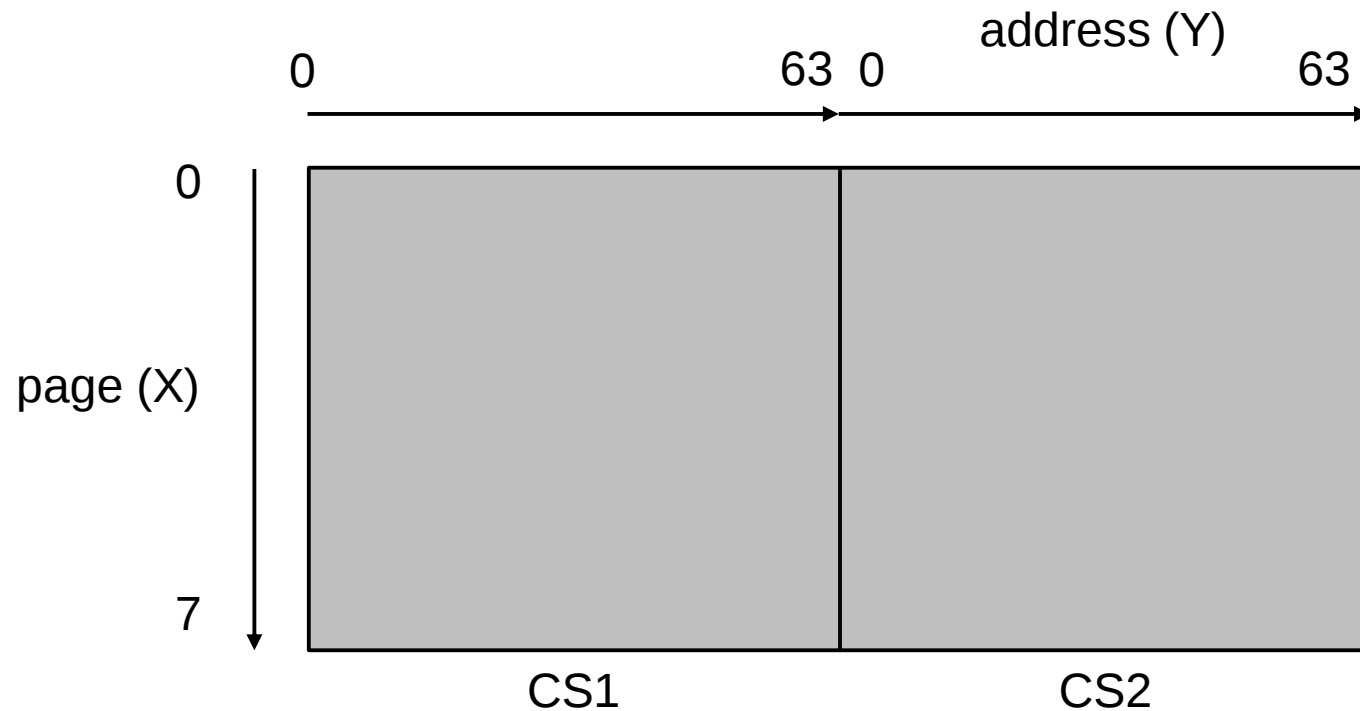
Egenskaper

- Address anger vilken pixel inom varje halva i horisontalled. Från 0-63 för varje halva.
- Page anger 0-7 i vertikalled.



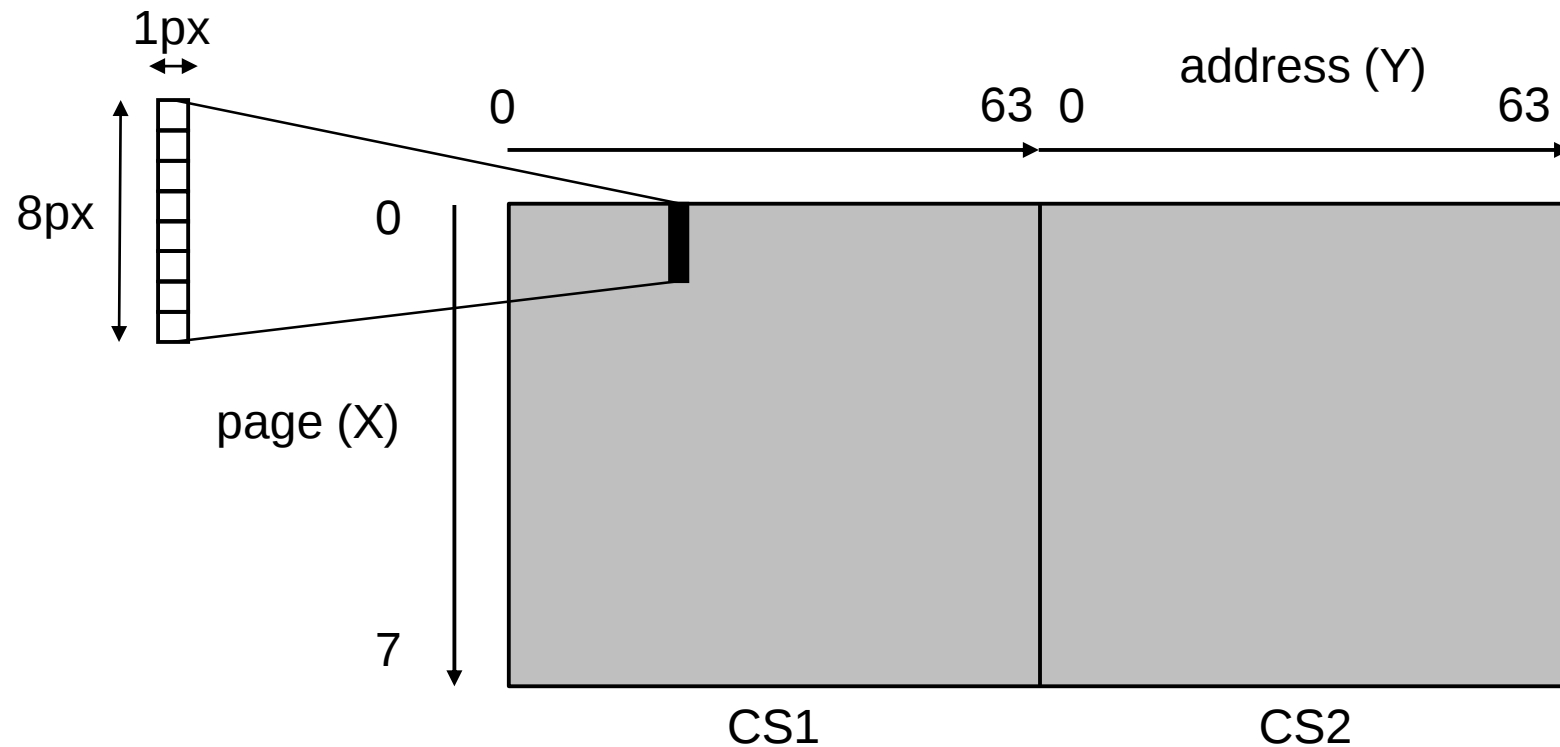
Egenskaper

- Vi kan bara läsa och skriva i hela alignade bytes i vertikalled!
- Exempel:



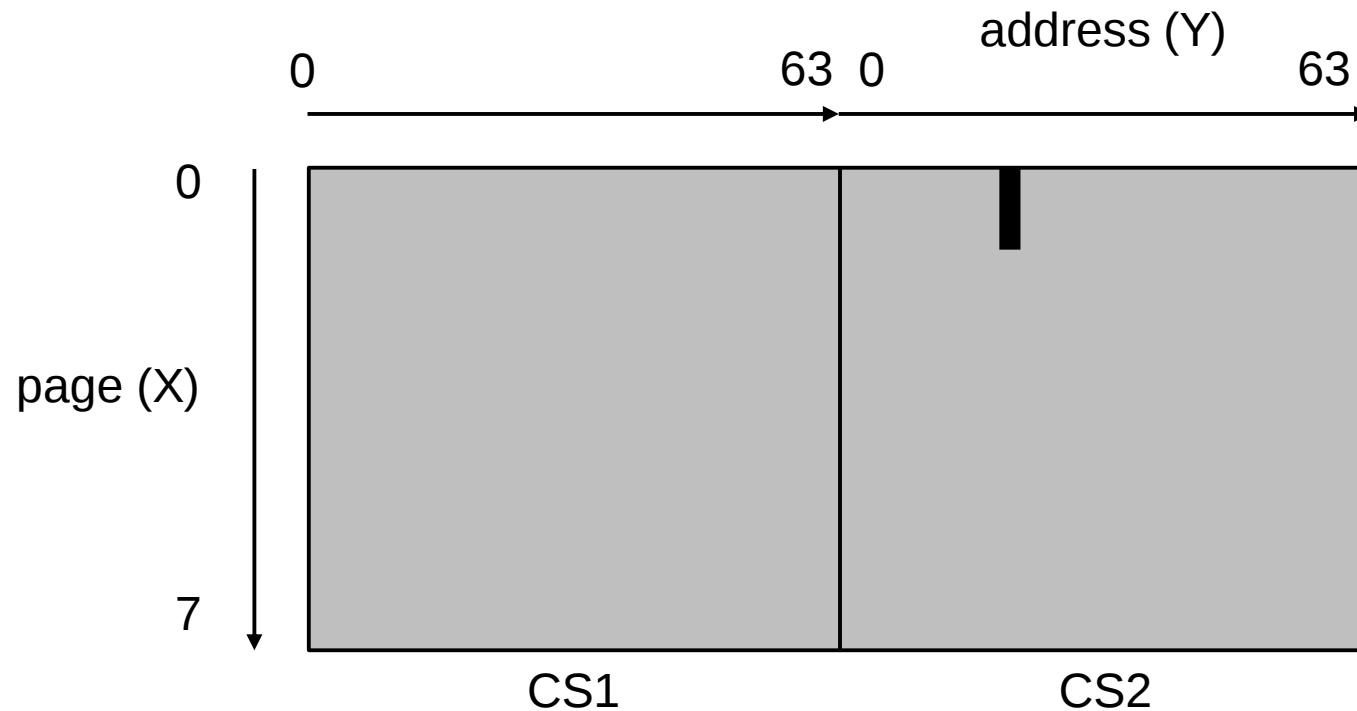
Egenskaper

- Vi kan bara läsa och skriva i hela alignade bytes i vertikalled!
- Exempel: CS1 = 1, CS2 = 0, page = 0, address = 20.



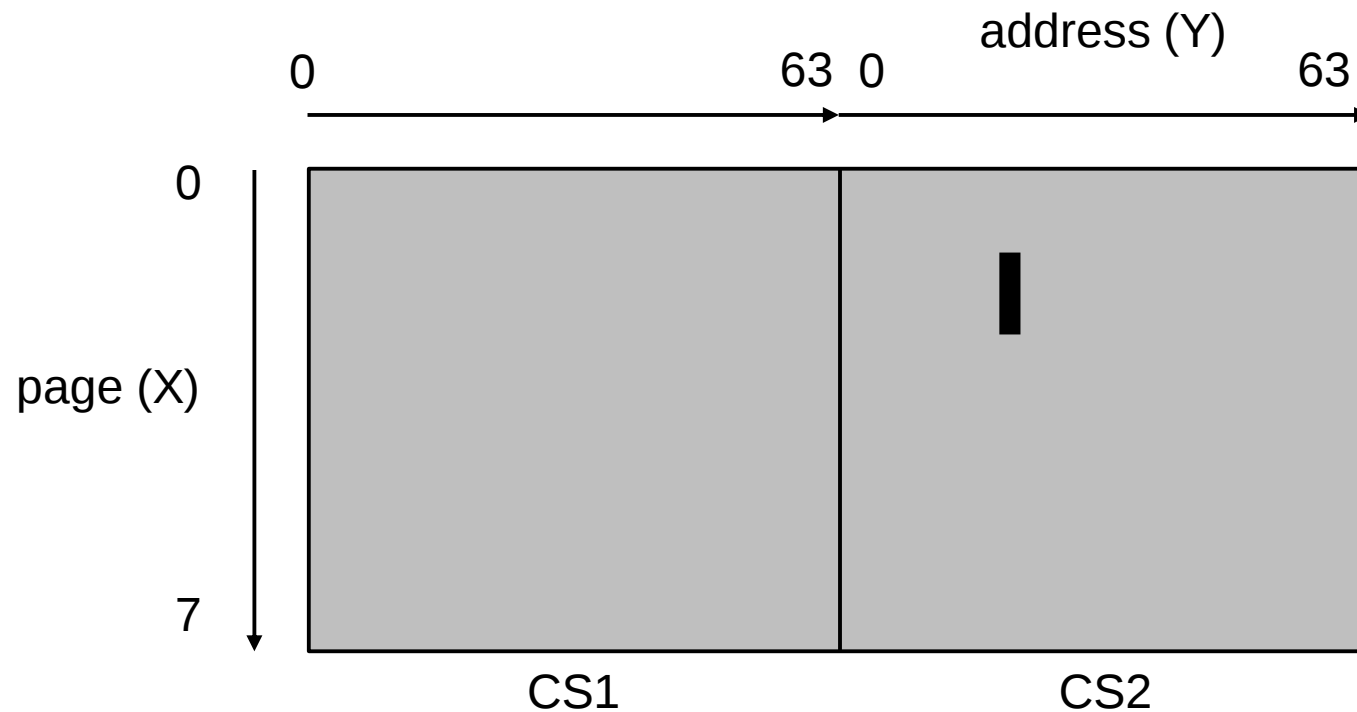
Egenskaper

- Vi kan bara läsa och skriva i hela alignade bytes i vertikalled!
- Exempel: CS1 = 0, CS2 = 1, page = 0, address = 20.



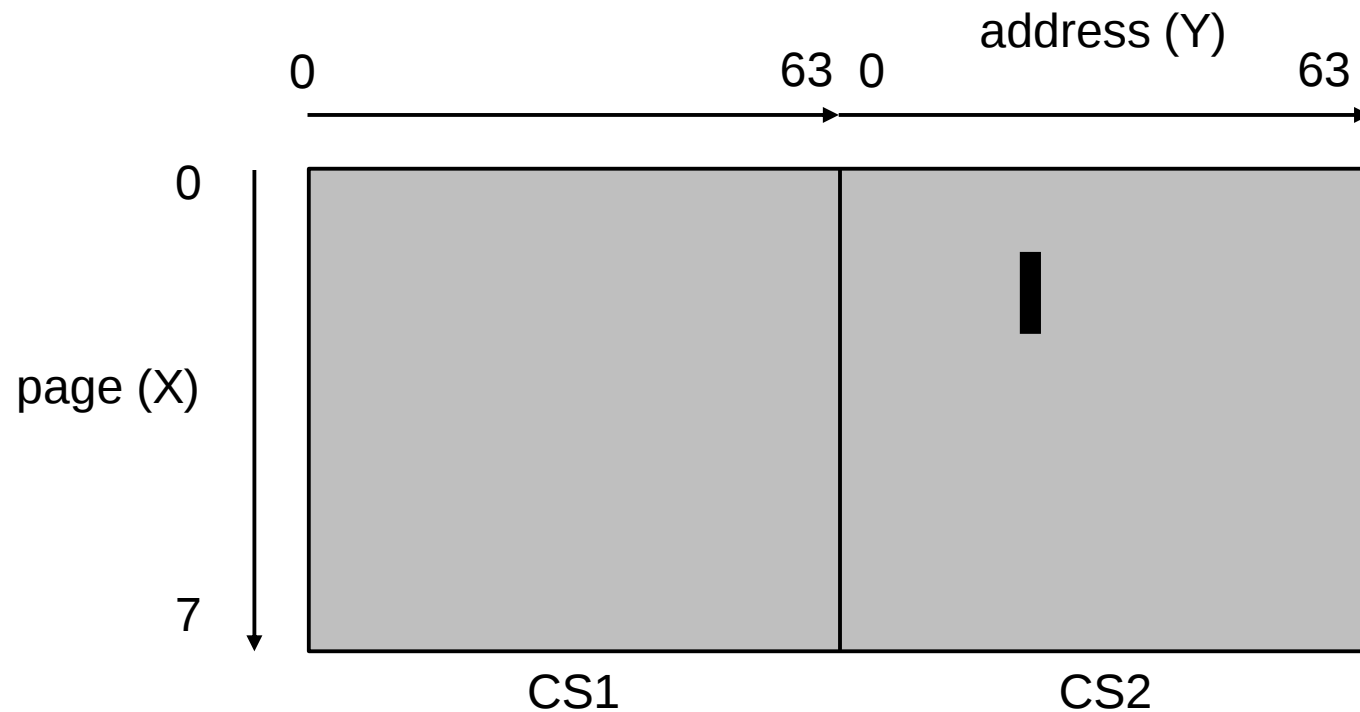
Egenskaper

- Vi kan bara läsa och skriva i hela alignade bytes i vertikalled!
- Exempel: CS1 = 0, CS2 = 1, page = 1, address = 20.

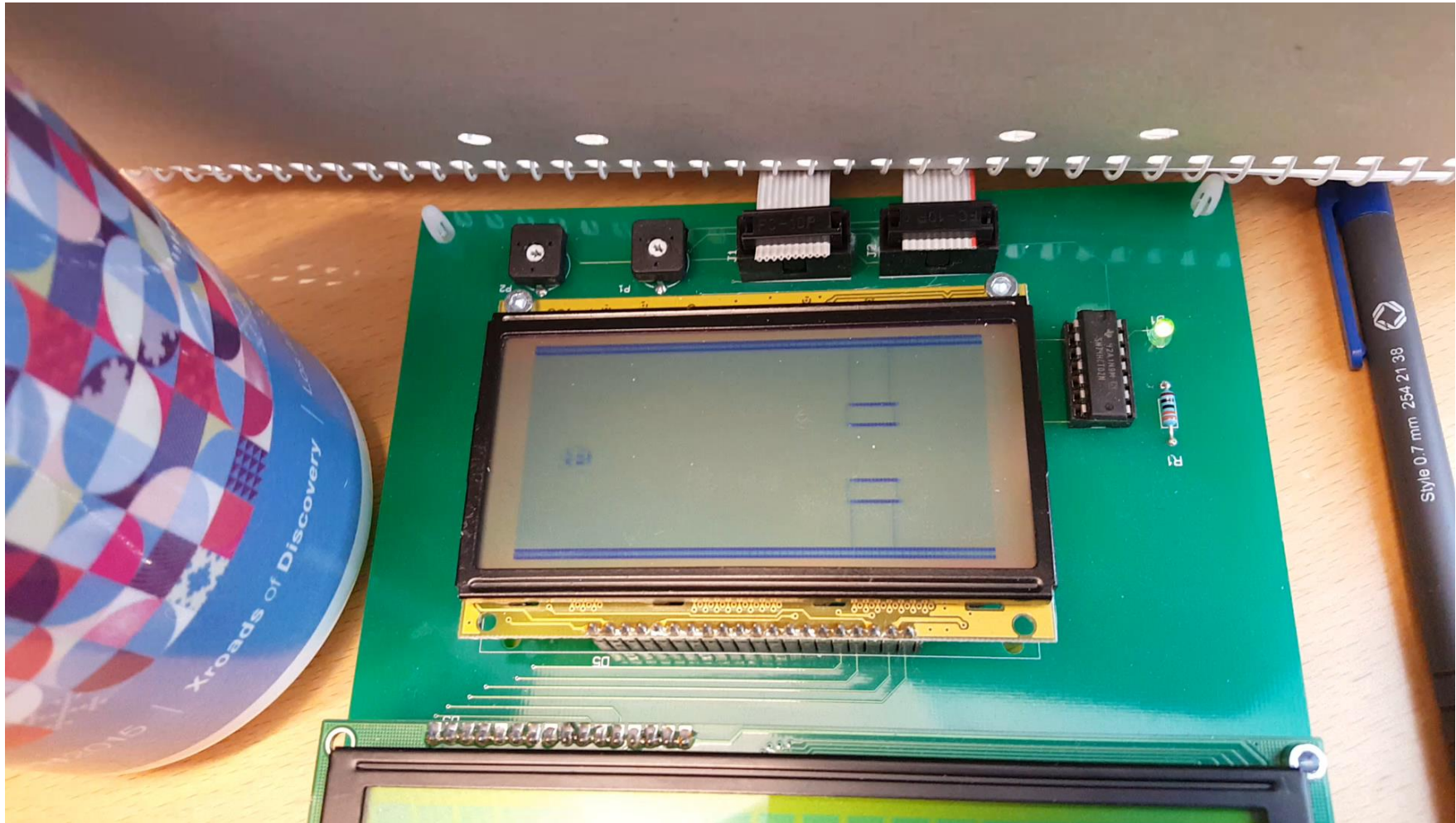


Egenskaper

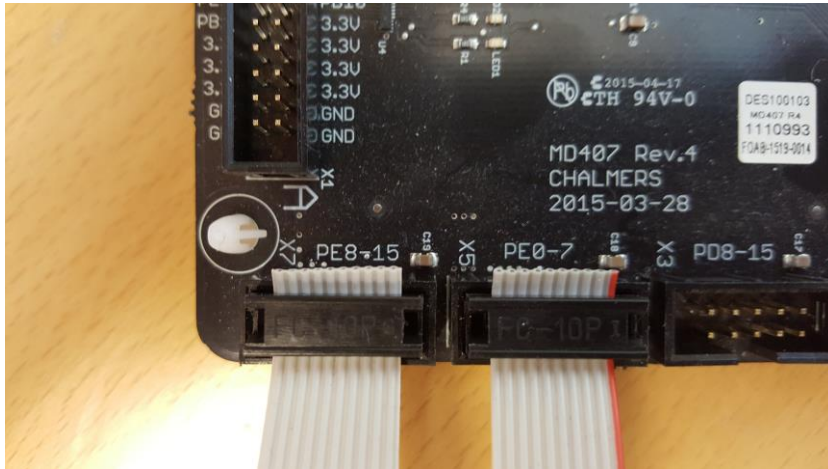
- Vi kan bara läsa och skriva i hela alignade bytes i vertikalled!
- Exempel: CS1 = 0, CS2 = 1, page = 1, address = 21.



Exempel



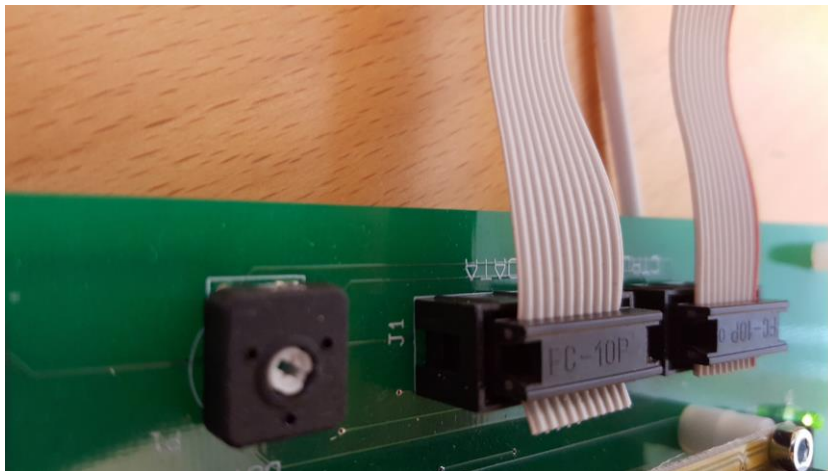
Implementation av drivrutiner



Port E

Låga bitar (0-7) = styrregister

Höga bitar (8-15) = dataregister



Implementation av drivrutiner

Definition av portar

```
#define PORT_DISPLAY_BASE 0x40021000 // MD407 port E

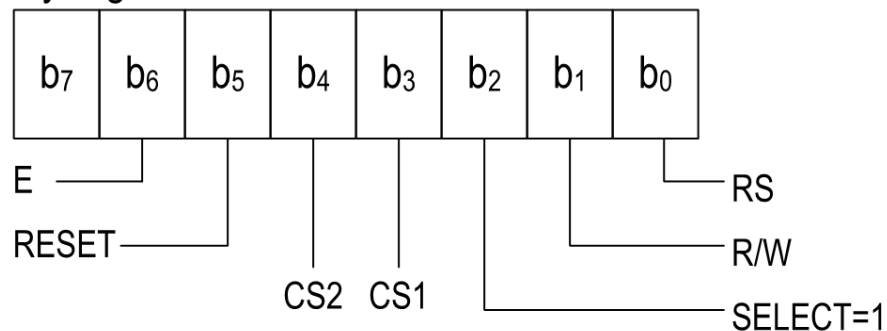
#define portModer      ((volatile unsigned int *)    (PORT_DISPLAY_BASE))
#define portOtyper     ((volatile unsigned short *) (PORT_DISPLAY_BASE+0x4))
#define portOspeedr    ((volatile unsigned int *)   (PORT_DISPLAY_BASE+0x8))
#define portPupdr      ((volatile unsigned int *)   (PORT_DISPLAY_BASE+0xC))
#define portIdr        ((volatile unsigned short *) (PORT_DISPLAY_BASE+0x10))
#define portIdrHigh    ((volatile unsigned char *)  (PORT_DISPLAY_BASE+0x11))
#define portOdrLow     ((volatile unsigned char *)  (PORT_DISPLAY_BASE+0x14))
#define portOdrHigh    ((volatile unsigned char *)  (PORT_DISPLAY_BASE+0x14+1))
```

Implementation av drivrutiner

Definition av kommandon till styrregister. Hantera synkronisering.

```
#define B_E      0x40    // Enable
#define B_RST    0x20    // Reset
#define B_CS2    0x10    // Controller Select 2
#define B_CS1    8       // Controller Select 1
#define B_SELECT 4       // 0 Graphics, 1 ASCII
#define B_RW     2       // 0 Write, 1 Read
#define B_RS     1       // 0 Command, 1 Data
```

styrregister



Implementation av drivrutiner

Definition av kommandon till display, sänds via dataregister.

```
#define LCD_ON          0x3F          // Display on
#define LCD_OFF         0x3E          // Display off
#define LCD_SET_ADD     0x40          // Set horizontal coordinate
#define LCD_SET_PAGE    0xB8          // Set vertical coordinate
#define LCD_DISP_START  0xC0          // Start address
#define LCD_BUSY        0x80          // Read busy status
```

Implementation av drivrutiner

Från datablad.

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display on/off	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is not affected. L: OFF, H: ON
Set address (Y address)	L	L	L	H	Y address (0 - 63)						Sets the Y address in the Y address counter.
Set page (X address)	L	L	H	L	H	H	H	Page (0 - 7)			Sets the X address at the X address register.
Display start line (Z address)	L	L	H	H	Display start line (0 - 63)						Indicates the display data RAM displayed at the top of the screen.
Status read	L	H	Busy	L	On/Off	Reset	L	L	L	L	Read status. BUSY L: Ready H: In operation ON/OFF L: Display ON H: Display OFF RESET L: Normal H: Reset
Write display data	H	L	Write data								Writes data (DB0:7) into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read display data	H	H	Read data								Reads data (DB0:7) from display data RAM to the data bus.

Implementation av drivrutiner

```
static void graphic_ctrl_bit_set(uint8_t x);
```

```
static void graphic_ctrl_bit_clear(uint8_t x);
```

```
static void select_controller(uint8_t controller);
```

Implementation av drivrutiner

```
static void select_controller(uint8_t controller) {  
    switch(controller) {  
        case 0:  
            graphic_ctrl_bit_clear(B_CS1|B_CS2);  
            break;  
        case B_CS1 :  
            graphic_ctrl_bit_set(B_CS1);  
            graphic_ctrl_bit_clear(B_CS2);  
            break;  
        case B_CS2 :  
            graphic_ctrl_bit_set(B_CS2);  
            graphic_ctrl_bit_clear(B_CS1);  
            break;  
        case B_CS1|B_CS2 :  
            graphic_ctrl_bit_set(B_CS1|B_CS2);  
            break;  
    }  
}
```

Implementation av drivrutiner

```
void graphic_initialize(void) ;
```

```
static void graphic_wait_ready(void) ;
```

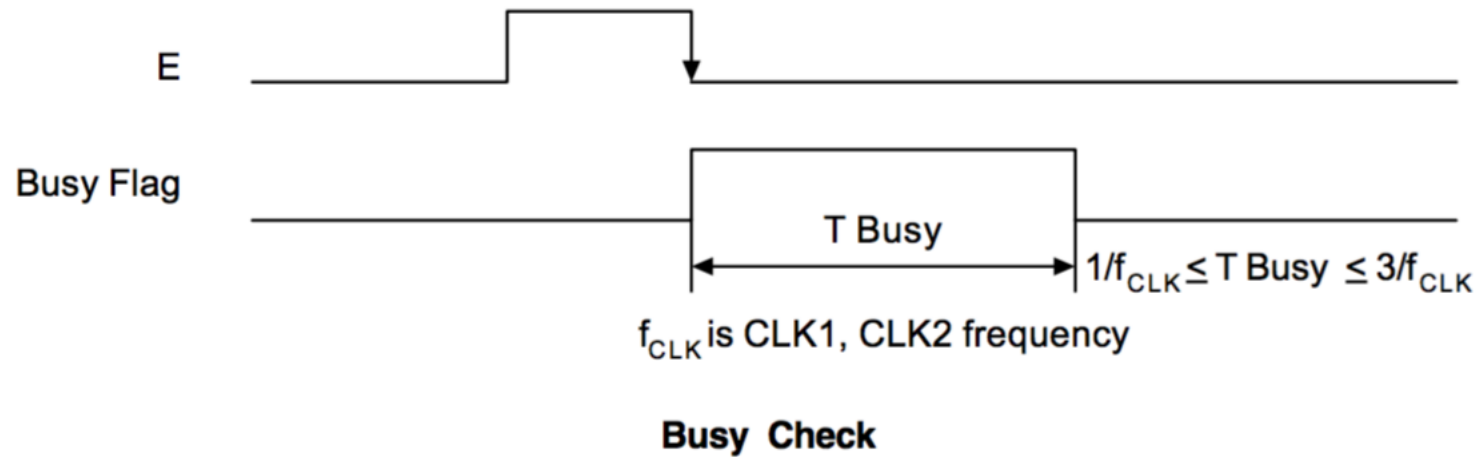
Implementation av drivrutiner

```
void graphic_initialize(void) {  
    graphic_ctrl_bit_set(B_E);  
    delay_micro(10);  
  
    graphic_ctrl_bit_clear(B_CS1 | B_CS2 | B_RST | B_E);  
    delay_milli(30);  
    graphic_ctrl_bit_set(B_RST);  
    delay_milli(100);  
    graphic_write_command(LCD_OFF,          B_CS1 | B_CS2);  
    graphic_write_command(LCD_ON,           B_CS1 | B_CS2);  
    graphic_write_command(LCD_DISP_START,   B_CS1 | B_CS2);  
    graphic_write_command(LCD_SET_ADD,      B_CS1 | B_CS2);  
    graphic_write_command(LCD_SET_PAGE,     B_CS1 | B_CS2);  
    select_controller(0);  
}
```

Implementation av drivrutiner

```
static void graphic_wait_ready(void) {
    uint8_t c;
    graphic_ctrl_bit_clear(B_E);
    *portModer = 0x00005555;           // 15-8 inputs, 7-0 outputs
    graphic_ctrl_bit_clear(B_RS);
    graphic_ctrl_bit_set(B_RW);
    delay_500ns();

    while(1) {
        graphic_ctrl_bit_set(B_E);
        delay_500ns();
        c = *portIdrHigh & LCD_BUSY;
        graphic_ctrl_bit_clear(B_E);
        delay_500ns();
        if( c == 0 ) break;
    }
    *portModer = 0x55555555;           // 15-0 outputs
}
```



När “Busy flag” är låg så kan displayen acceptera kommandon och data. Se databladet sida 19.

Implementation av drivrutiner

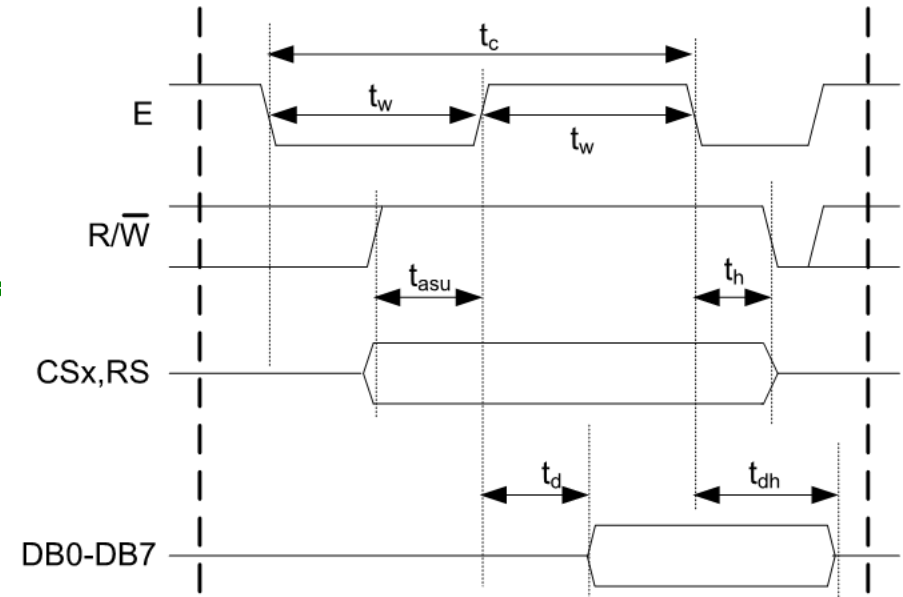
```
static uint8_t graphic_read(uint8_t controller);
```

```
static uint8_t graphic_read_data(uint8_t controller);
```

Implementation av drivrutiner

```
static uint8_t graphic_read(uint8_t controller) {
    uint8_t c;
    graphic_ctrl_bit_clear(B_E);
    *portMod0 = 0x0005555; // 15-8 inputs, 7-0 outputs
    graphic_ctrl_bit_set(B_RS|B_RW);
    select_controller(controller);
    delay_500ns();
    graphic_ctrl_bit_set(B_E);
    delay_500ns();
    c = *portIdrHigh;
    graphic_ctrl_bit_clear(B_E);
    *portMod0 = 0x5555555; // 15-0 outputs

    if( controller & B_CS1 ) {
        select_controller(B_CS1);
        graphic_wait_ready();
    }
    if( controller & B_CS2 ) {
        select_controller(B_CS2);
        graphic_wait_ready();
    }
    return c;
}
```



Implementation av drivrutiner

```
static uint8_t graphic_read_data(uint8_t controller) {  
    graphic_read(controller);  
    return graphic_read(controller);  
}
```

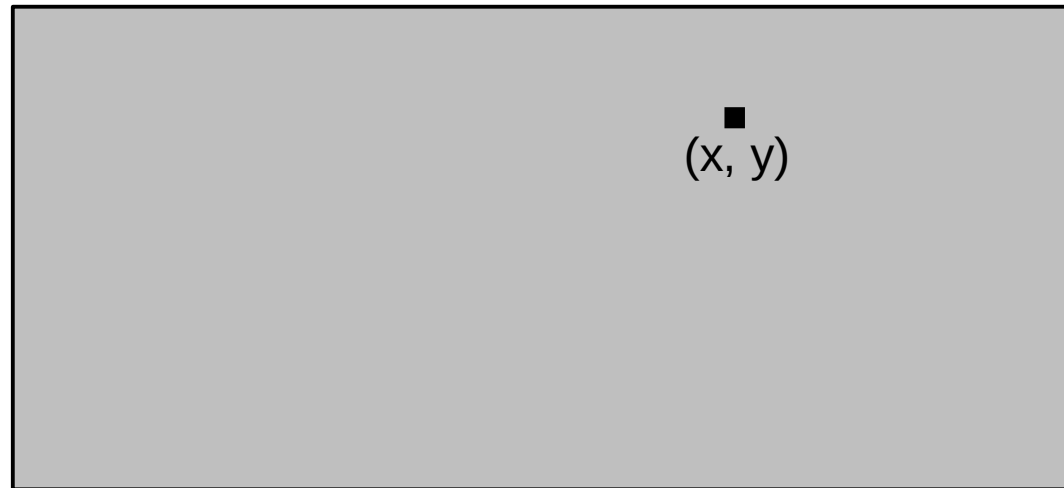
Enligt databladet måste vi läsa ut värdet två gånger för att det ska få effekt, se s. 19.

Implementation av drivrutiner

```
static void graphic_write(uint8_t value, uint8_t controller);  
  
static void graphic_write_command(uint8_t command, uint8_t controller);  
  
static void graphic_write_data(uint8_t data, uint8_t controller);  
  
void graphic_clear_screen(void);
```

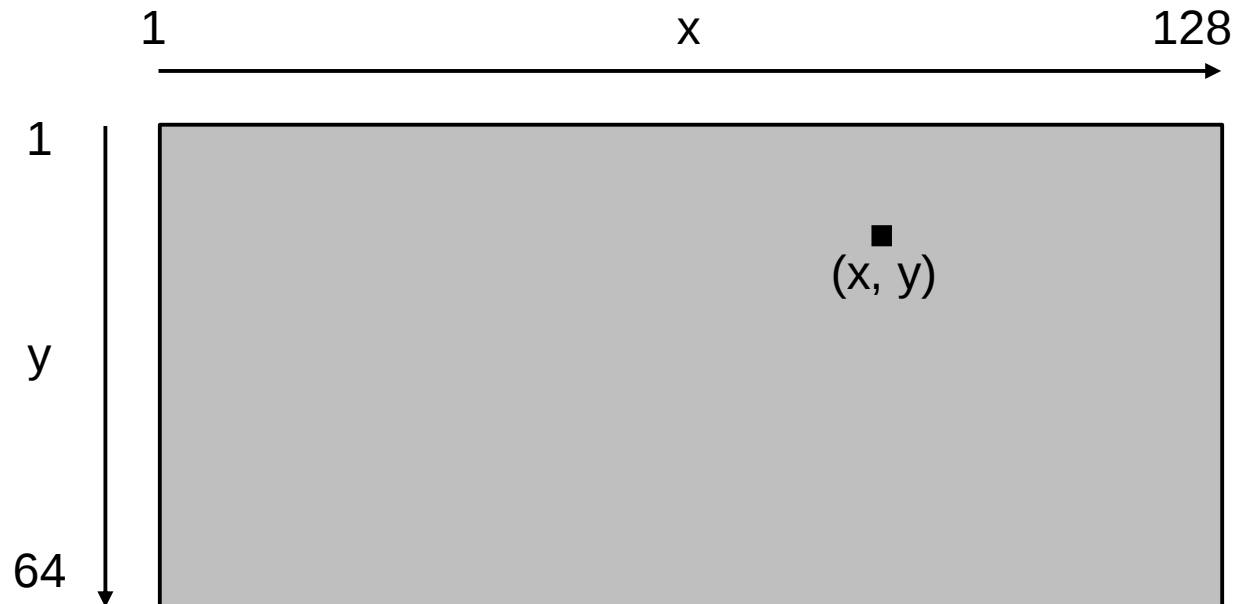
Skriva till display

- Funkar ok om vi vill rensa hela displayen, men inte så smidigt annars.
- Vi vill kunna skriva till displayen varje pixel för sig.



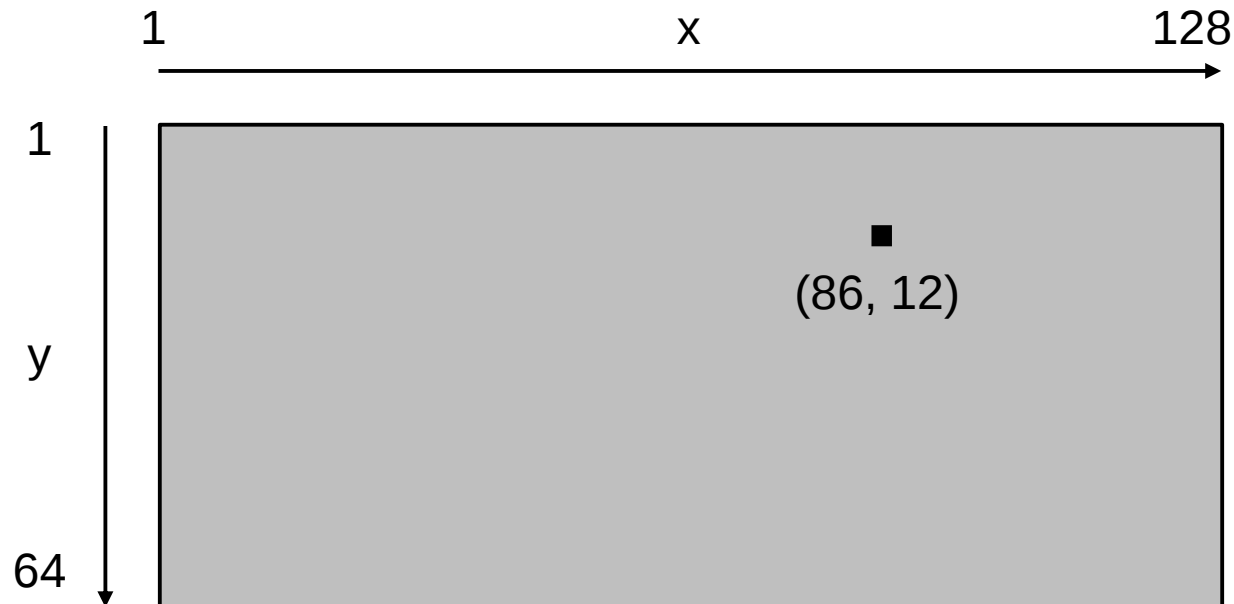
Skriva till display

- Vi vill ha en funktion `void pixel(int x, int y, int set)`
- Inför *logiska koordinater* med x i [1..128] och y i [1..64]



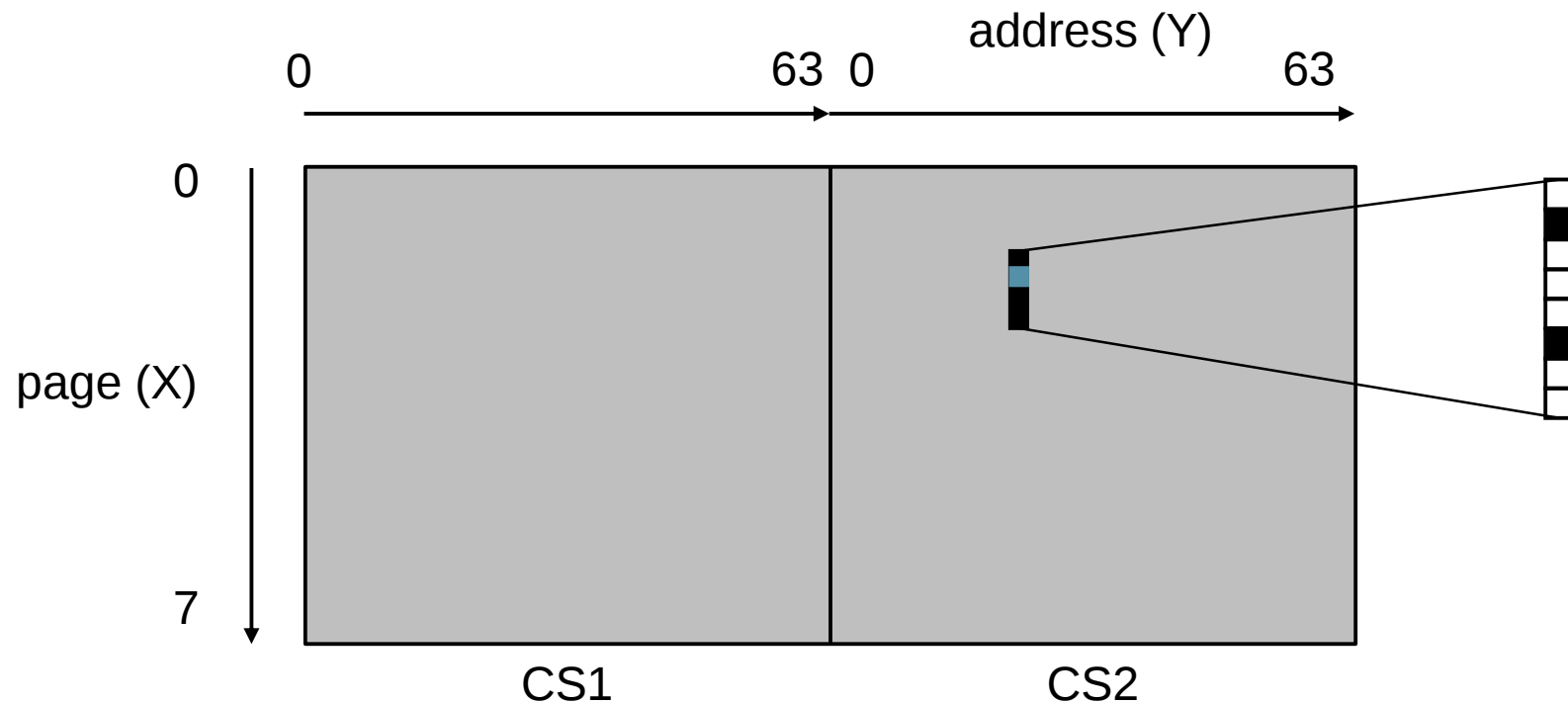
Skriva till display

- Hur skriver vi en sådan funktion när vi bara kan läsa/skriva i hela bytes?
- Exempel: Vi vill skriva till logiska koordinater (86, 12).



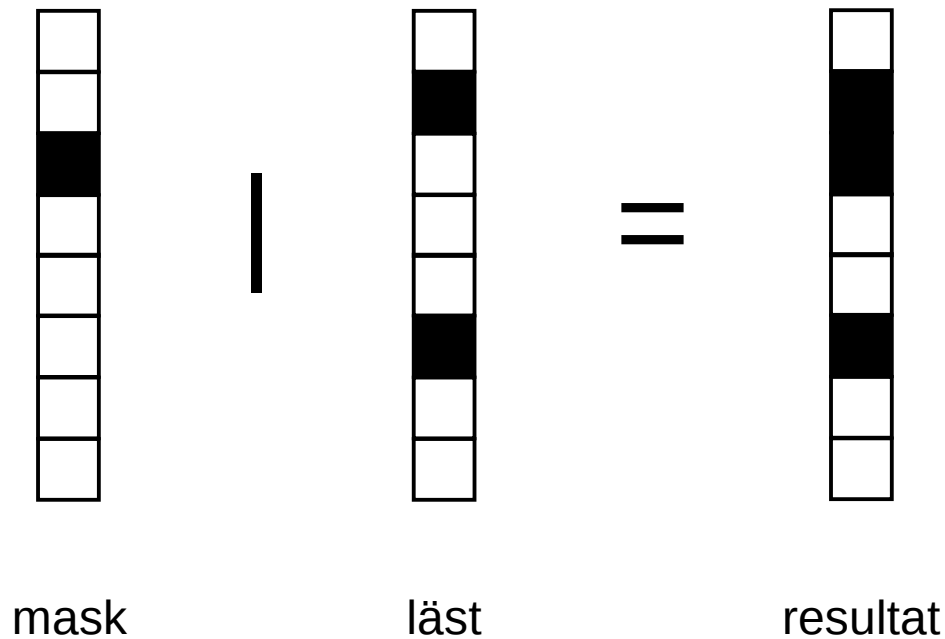
Egenskaper

- Vår address blir då $86 - 65 = 21$. Page blir $(12-1) / 8 = 1$.
- Vår bit i vår vertikala byte blir $(12-1) \% 8 = 3$.



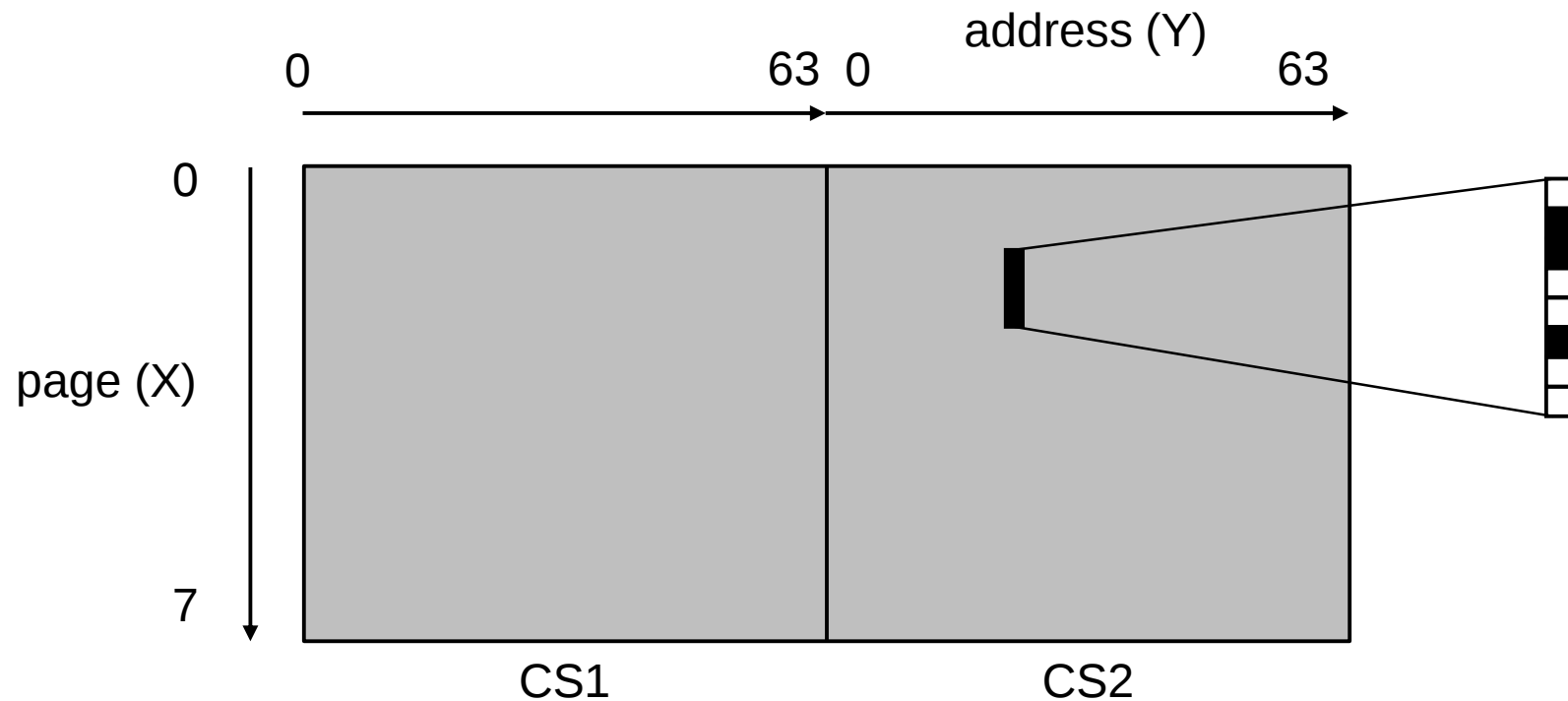
Egenskaper

- Vi läser då en byte med fysiska koordinater (CS2, 21, 1) .
- Vi vill skriva till bit 3 i denna byte.



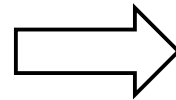
Egenskaper

- Skriv denna resulterande byte tillbaka till displayen.



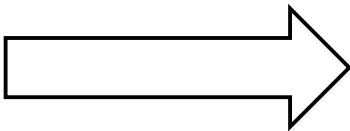
Skriva till display

```
void pixel(int x, int y, int set) {  
    uint8_t mask, c, controller;  
    int index;  
  
    if((x < 1) || (y < 1) || (x > 128) || (y > 64)) return;  
  
    index = (y-1)/8;  
  
    switch( (y-1)%8 ) {  
        case 0: mask = 1; break;  
        case 1: mask = 2; break;  
        case 2: mask = 4; break;  
        case 3: mask = 8; break;  
        case 4: mask = 0x10; break;  
        case 5: mask = 0x20; break;  
        case 6: mask = 0x40; break;  
        case 7: mask = 0x80; break;  
    }  
}
```



`mask = 1 << ((y-1)%8);`

Forts.



Skriva till display

```
if(set == 0)
    mask = ~mask;

if(x > 64){
    controller = B_CS2;
    x = x - 65;
} else {
    controller = B_CS1;
    x = x-1;
}

graphic_write_command(LCD_SET_ADD | x, controller );
graphic_write_command(LCD_SET_PAGE | index, controller );
c = graphic_read_data(controller);
graphic_write_command(LCD_SET_ADD | x, controller);

if(set)
    mask = mask | c;
else
    mask = mask & c;

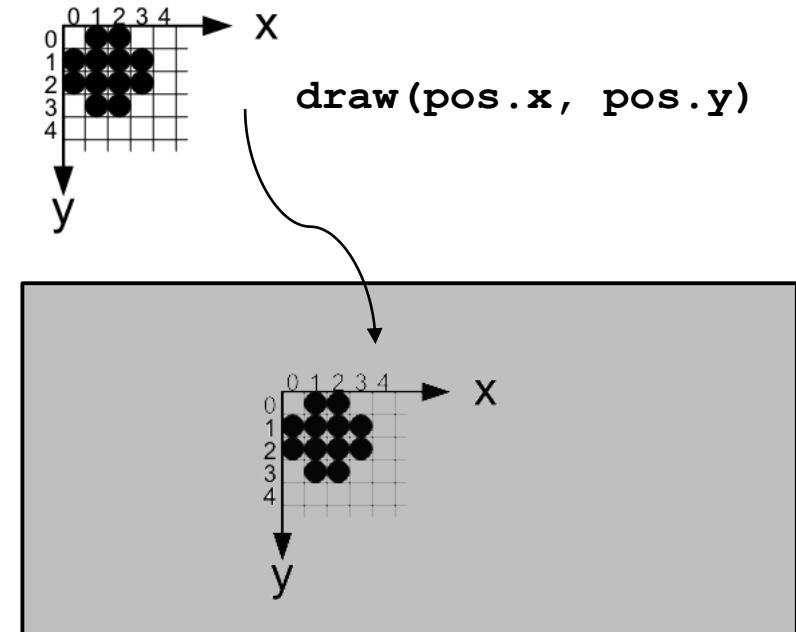
graphic_write_data(mask, controller);
}
```

Notera: på grund av
autoinkrementering av address

Skriva till display

```
int main(int argc, char **argv)
{
    POBJECT p = &ball;
    init_app();
    graphic_initialize();
    graphic_clear_screen();

    p->set_speed(p, 4, 1);
    while(1)
    {
        p->move(p);
    }
}
```



Lista av logiska koordinater beskriver geometrin.

Skicka till `pixel()` med offset för nuvarande position av objektet.

Prestanda

Upplevd bildkvalité

- Flimmer.
- Hackar.
- Kan vi se enskilda bildrutor?

Uppdateringsfrekvens (eng. frame rate)

- Vad är en lämplig uppdateringsfrekvens för spel med realtidsgrafik?
- Hur kan vi specificera en uppdateringsfrekvens för vårt spel?

Prestanda

```
#define TARGET_FPS 30

int main(int argc, char **argv)
{
    POBJECT p = &ball;
    init_app();
    graphic_initialize();
    graphic_clear_screen();

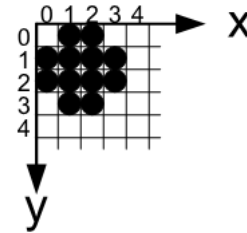
    p->set_speed(p, 4, 1);
    while(1)
    {
        p->move(p);
    }
}
```

Prestanda

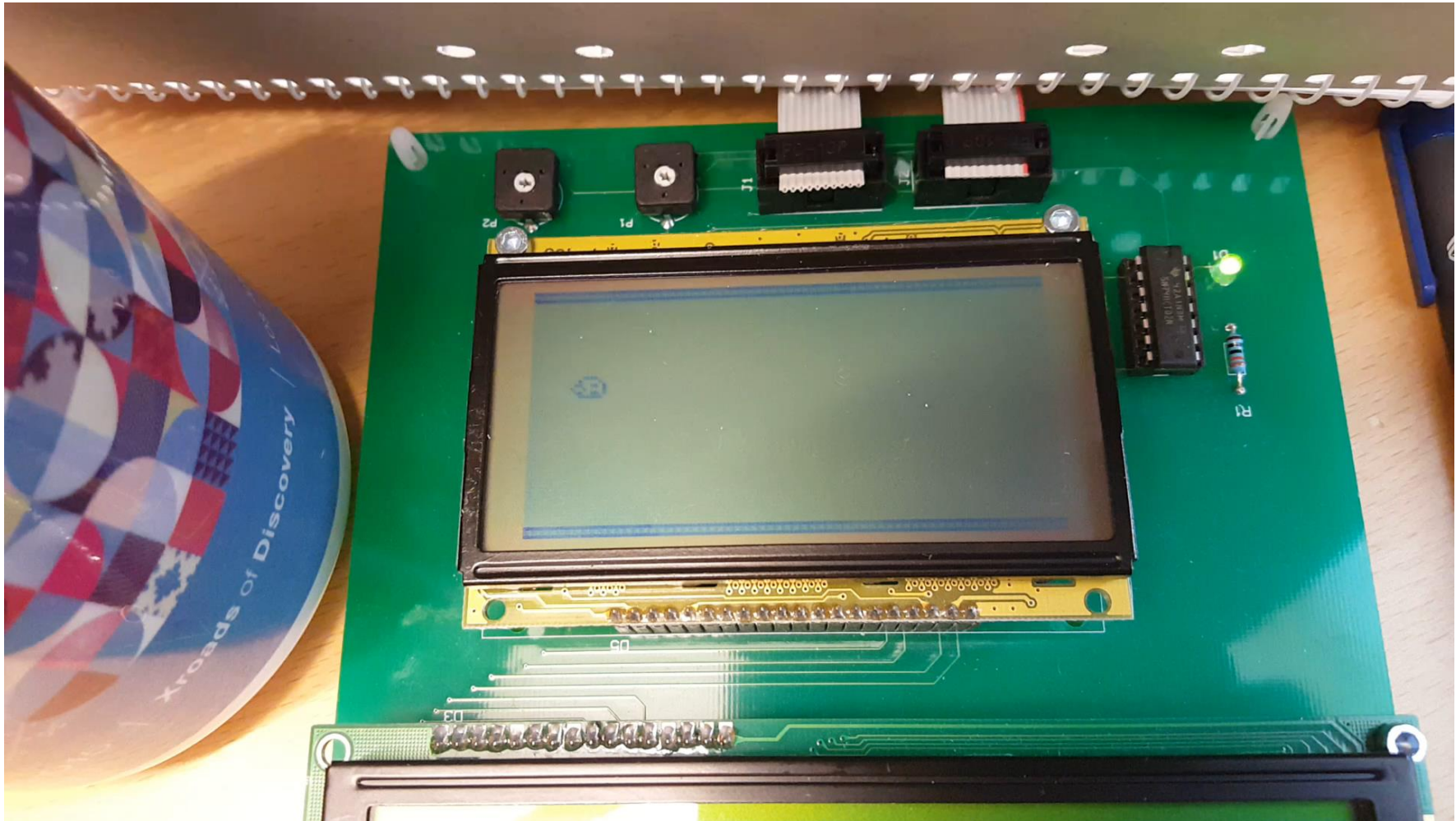
```
#define TARGET_FPS 30

int main(int argc, char **argv)
{
    POBJECT p = &ball;
    init_app();
    graphic_initialize();
    graphic_clear_screen();

    p->set_speed(p, 4, 1);
    while(1)
    {
        p->move(p);
        delay_milli(1000 / TARGET_FPS);
    }
}
```



Prestanda



Prestanda

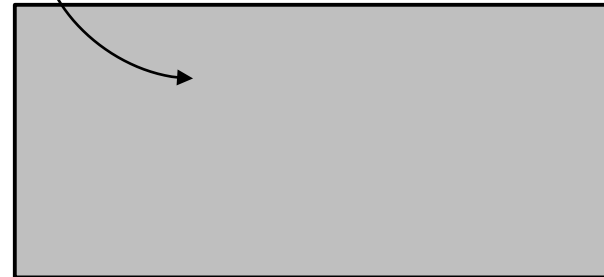
Varför flimrar det så?

På grund av att vi ritar
direkt till skärmen.

Lösning: använd
dubbelbuffering.

```
...  
draw()  
draw()  
draw()  
draw()  
draw()  
draw()  
...
```

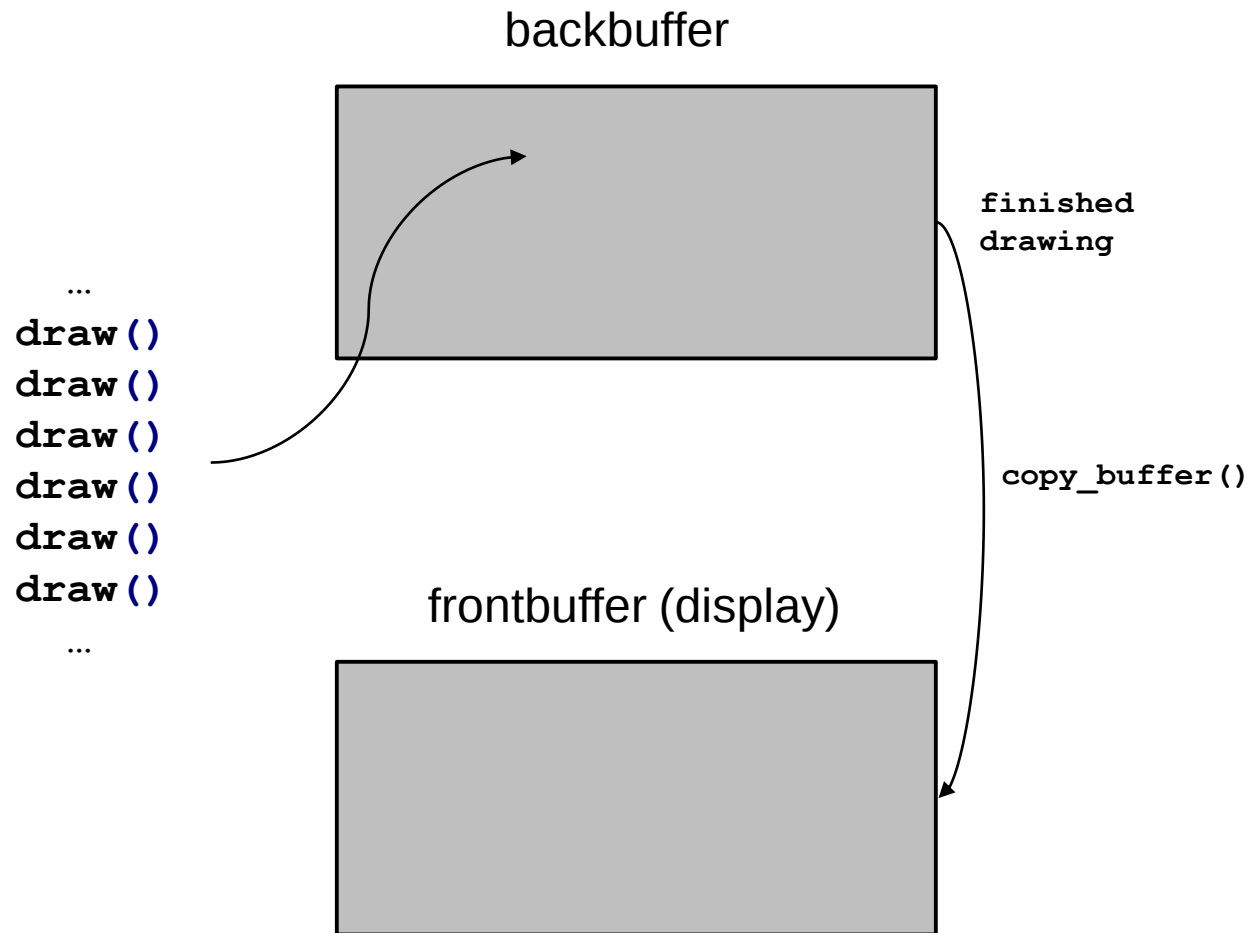
frontbuffer (display)



Prestanda

Rita till en backbuffer –
alltså till en buffer i
RAM-minnet.

Rita sedan hela
backbuffern i ett svep
när vi är klara med vår
nuvarande frame.



Prestanda

En enkel implementation.

```
uint8_t backBuffer[1024]; // 128 * 64 / 8

void clear_backBuffer() {
    int i;
    for (i = 0; i < 1024; i++)
        backBuffer[i] = 0;
}
```

Prestanda

Skriv till backbuffern istället för till skärmen.

```
void pixel(int x, int y) {
    uint8_t mask;
    int index = 0;
    if( (x > 128) || (x < 1) || (y > 64) || (y < 1) ) return;

    mask = 1 << ((y-1)%8);

    if(x > 64) {
        x -= 65;
        index = 512;
    }

    index += x + ((y-1)/8)*64;

    backBuffer[index] |= mask;
}
```

Prestanda

Kopiering av backbuffer till frontbuffer. Jämför med `graphic_clear_screen()`.

```
void graphic_draw_screen(void) {
    uint8_t i, j, controller, c;
    unsigned int k = 0;

    for(c = 0; c < 2; c++) {
        controller = (c == 0) ? B_CS1 : B_CS2;
        for(j = 0; j < 8; j++) {
            graphic_writeCommand(LCD_SET_PAGE | j, controller);
            graphic_writeCommand(LCD_SET_ADD | 0, controller);
            for(i = 0; i <= 63; i++, k++) {
                graphic_write_data(backBuffer[k], controller);
            }
        }
    }
}
```

Prestanda

```
int main(int argc, char **argv)
{
    POBJECT p = &ball;
    init_app();
    graphic_initialize();
    graphic_clear_screen();

    p->set_speed(p, 4, 1);
    while(1)
    {
        p->move(p);
        delay_milli(40);
    }
}
```

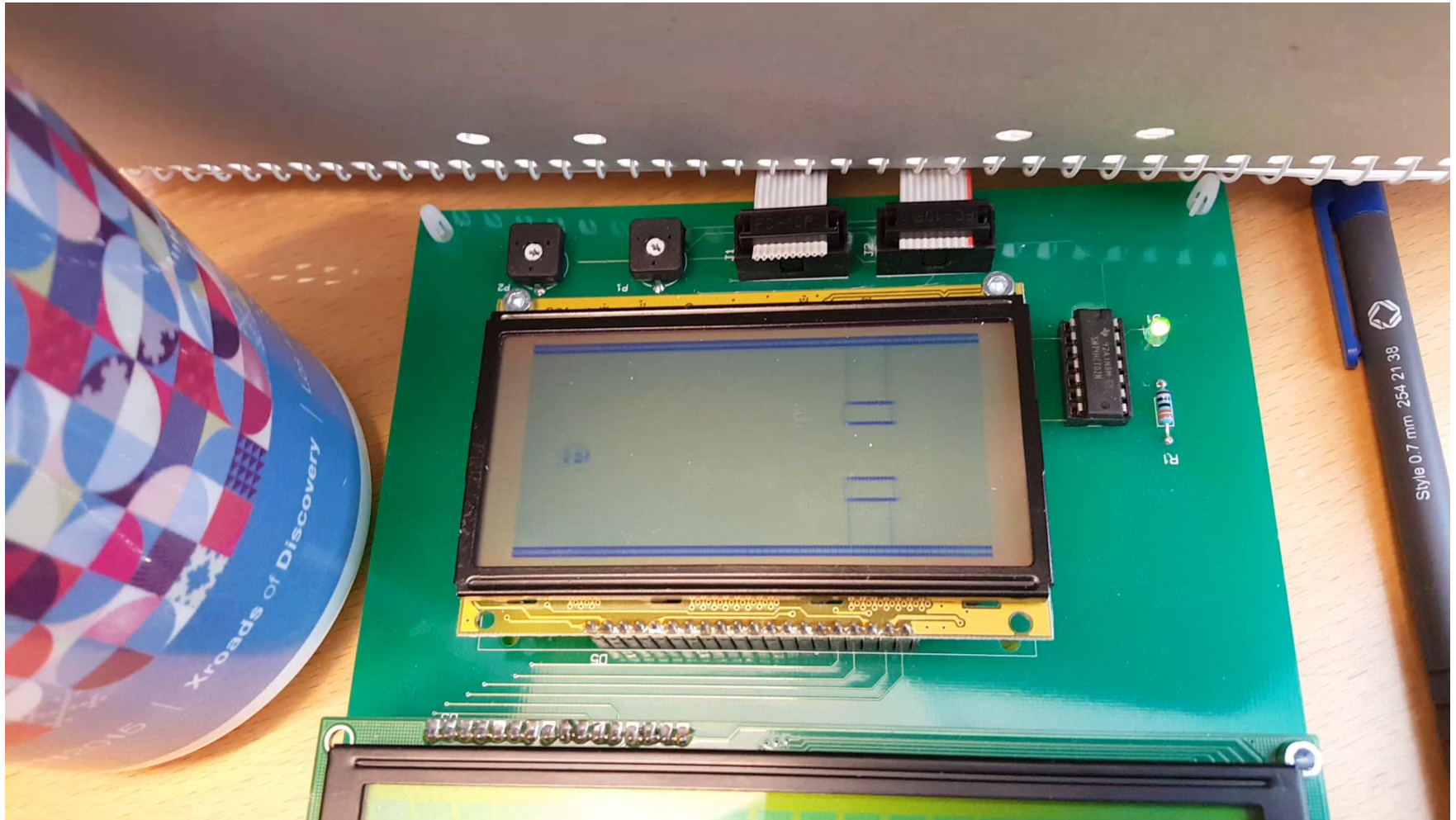
```
int main(int argc, char **argv)
{
    init_app();
    graphic_initialize();
    graphic_clear_screen();

    while(1)
    {
        clear_backBuffer();

        // DO STUFF.

        graphic_draw_screen();
        delay_milli(40);
    }
}
```

Exempel

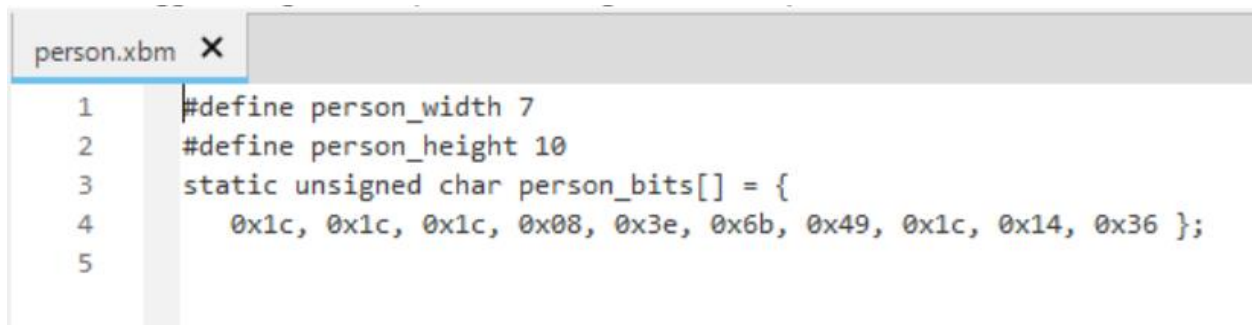


Design av grafik

- Arbetsbok: lista av logiska koordinater.
 - Svårt att "rita" mer avancerad grafik.
 - Inte heller minneseffektivt: två unsigned char per pixel.
- Vill designa sprites i externt bildredigeringsprogram.
 - Problem 1: vilket format stödjer 1 bit per pixel?
- Vill ladda dessa bildfiler i vårt program.
 - Problem 2: vi har inget standardbibliotek med fil-i/o i vårt inbyggda system.

Design av grafik

- Förslag till lösning
 - Använd filformat X Bitmap (.xbm).
 - Formatet specificeras av bredd, höjd och ett antal bytes, allt direkt i c-kod!
 - Kan därför användas direkt i källkoden med
`#include "image.xbm"`



```
person.xbm X
1  #define person_width 7
2  #define person_height 10
3  static unsigned char person_bits[] = {
4      0x1c, 0x1c, 0x1c, 0x08, 0x3e, 0x6b, 0x49, 0x1c, 0x14, 0x36 };
5
```

Design av grafik

- Filformatet är standard och stöds av många bildbehandlingsprogram.
- Tänkte visa två exempel som använder programmet GIMP (<https://www.gimp.org/>), som är gratis och finns till många plattformar.



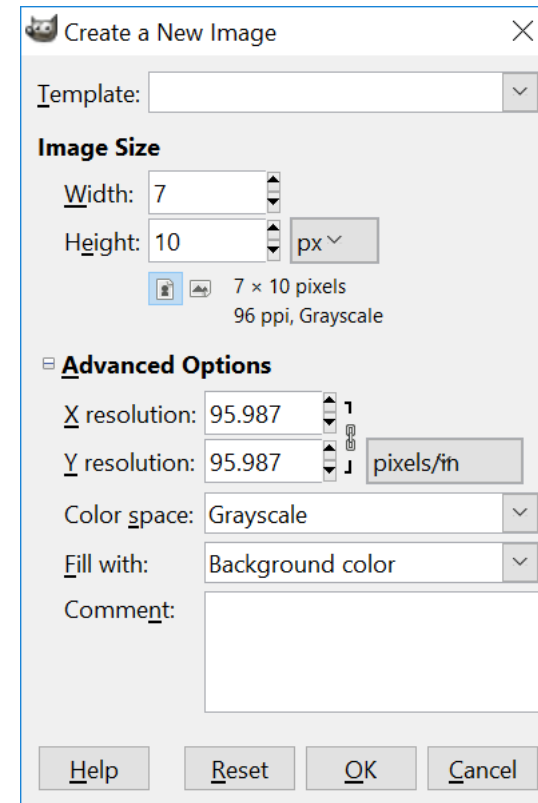
Design av grafik

Designa egen sprite

Gå till File->New...

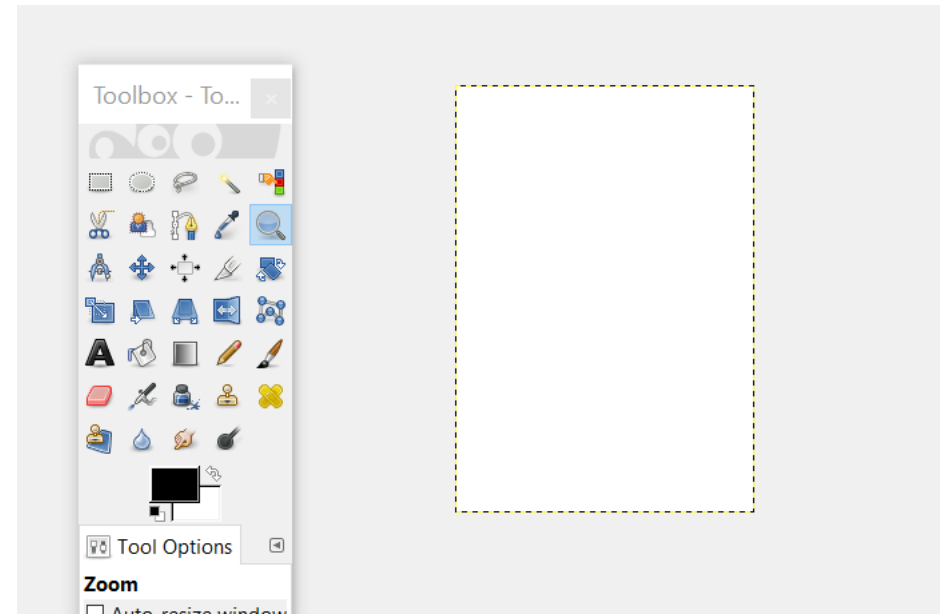
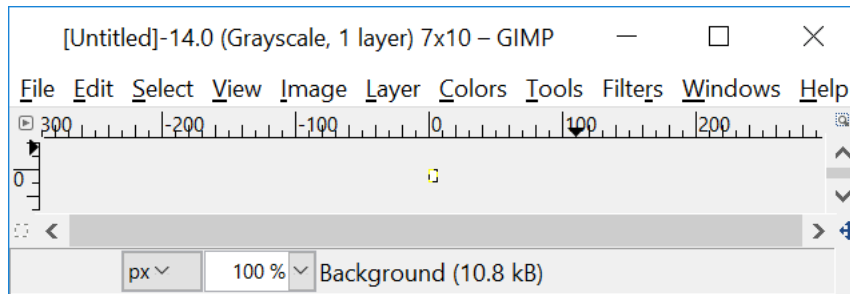
Sätt bredd och höjd i pixlar

Välj Color space: Grayscale



Design av grafik

Bilden är extremt liten på vanlig bildskärm.
Zooma in.

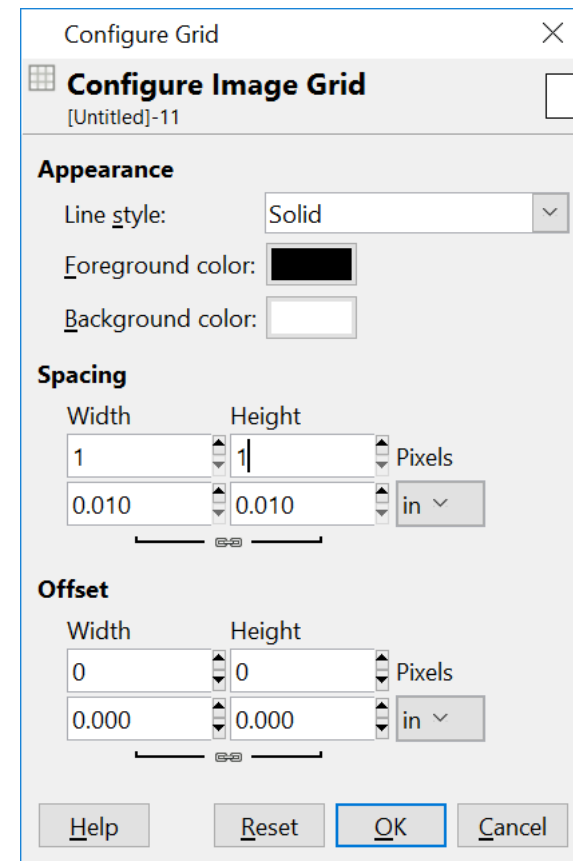


Design av grafik

1x1 pixels rutnät hjälper till vid ritning.

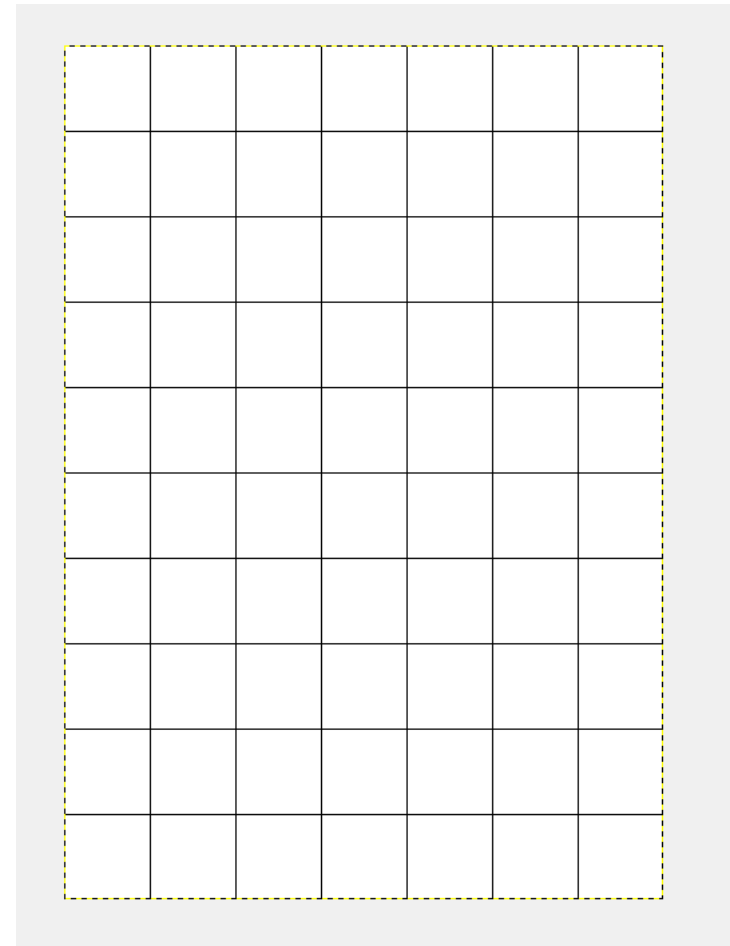
Välj Image->Configure Grid...

Välj Width = Height = 1 px



Design av grafik

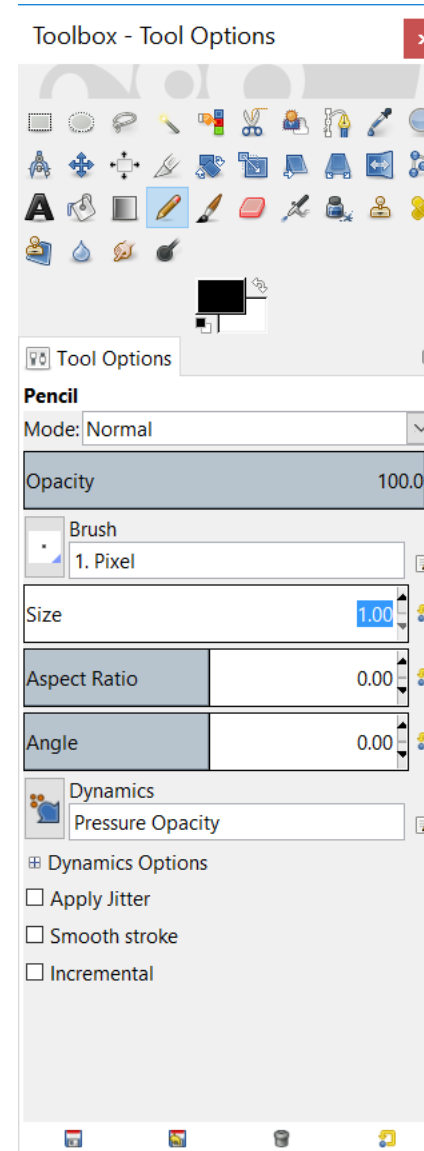
Togglar View->Show Grid



Design av grafik

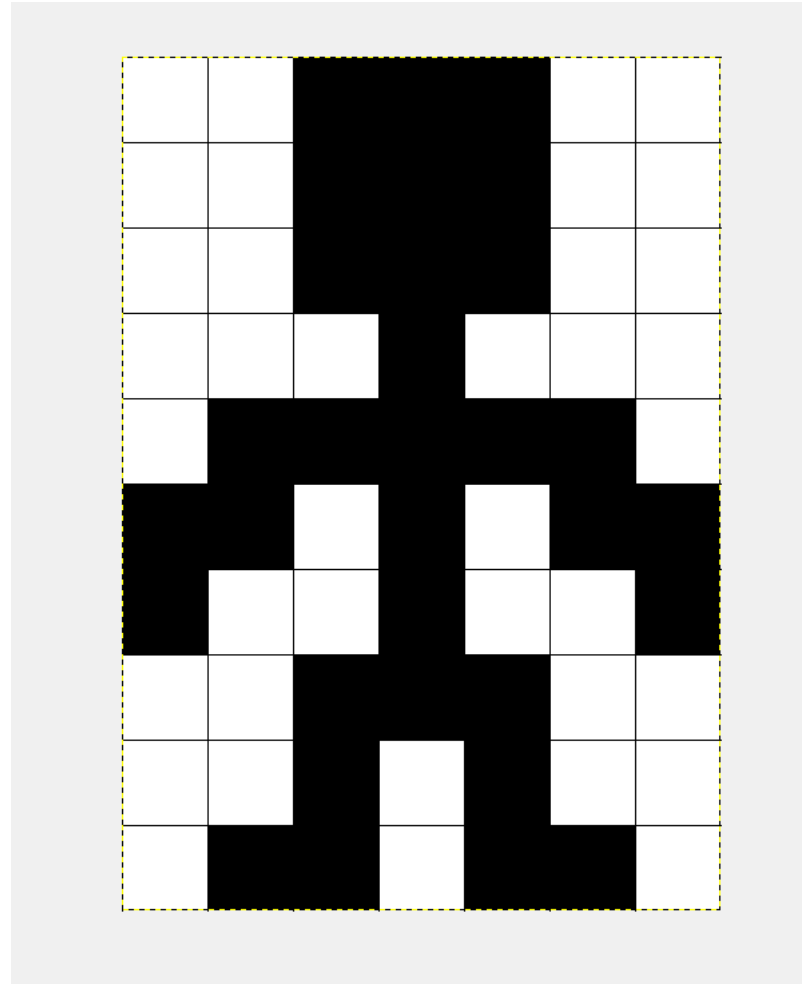
Använd "Pencil Tool"

Sätt Size = 1 px



Design av grafik

Rita!



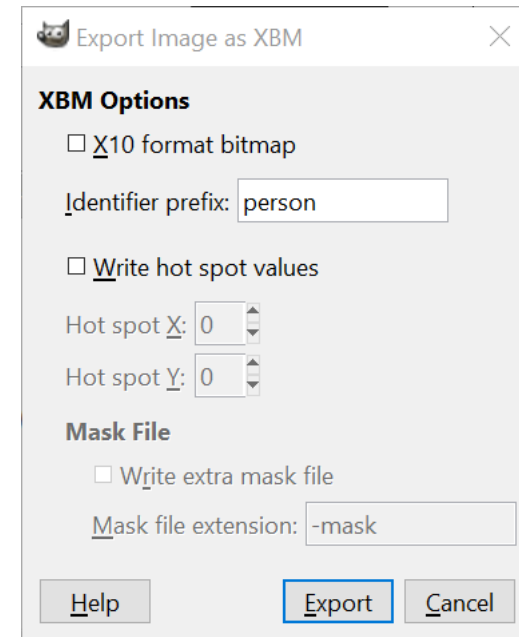
Design av grafik

Spara sedan som X Bitmap-fil.

Välj File->Export as...

Spara med ändelsen .xbm

Klicka på Export i dialogruten



Design av grafik

Resultat:

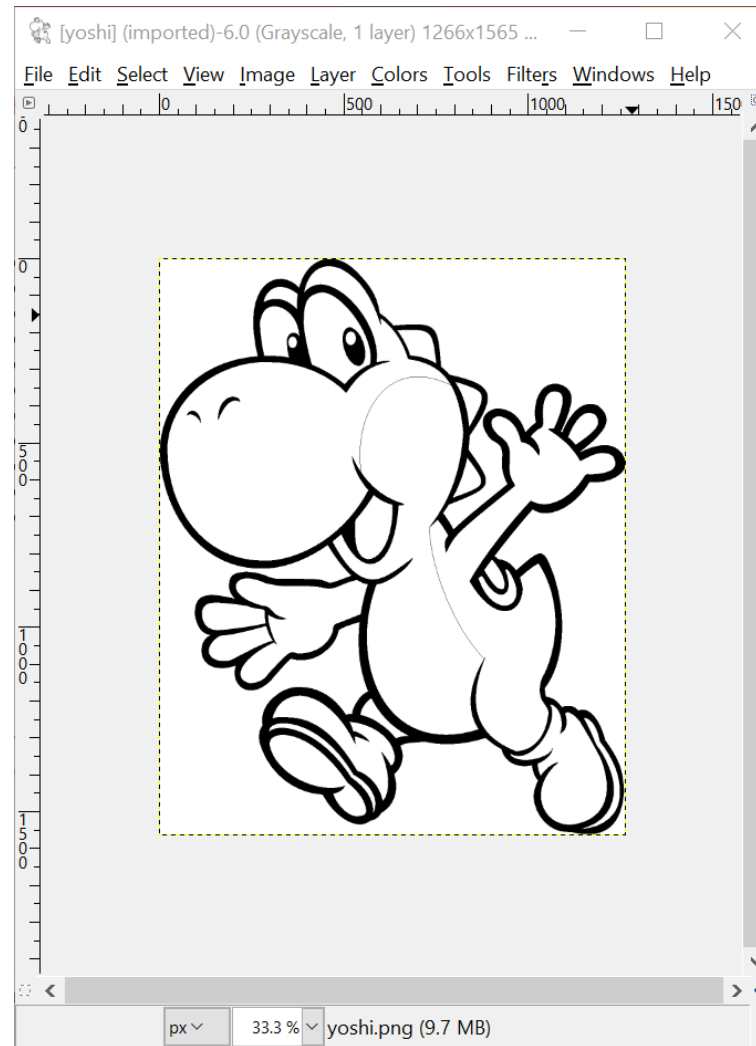
```
person.xbm X
1  #define person_width 7
2  #define person_height 10
3  static unsigned char person_bits[] = {
4      0x1c, 0x1c, 0x1c, 0x08, 0x3e, 0x6b, 0x49, 0x1c, 0x14, 0x36 };
5
```

Design av grafik

Vill designa en större eller mer komplicerad sprite.

Hitta en bild du gillar på internet och sampla ner färgrymd och upplösning!

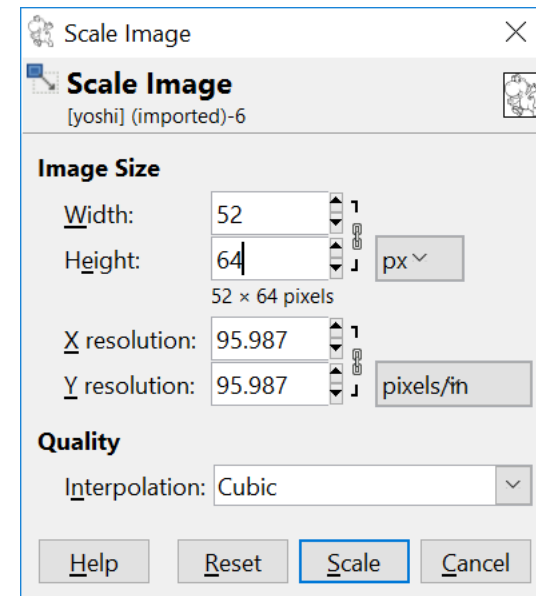
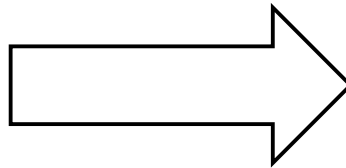
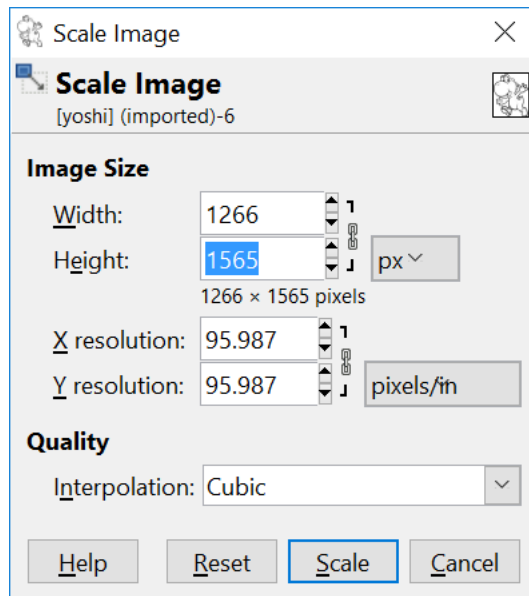
Exempel: Yoshi.



Design av grafik

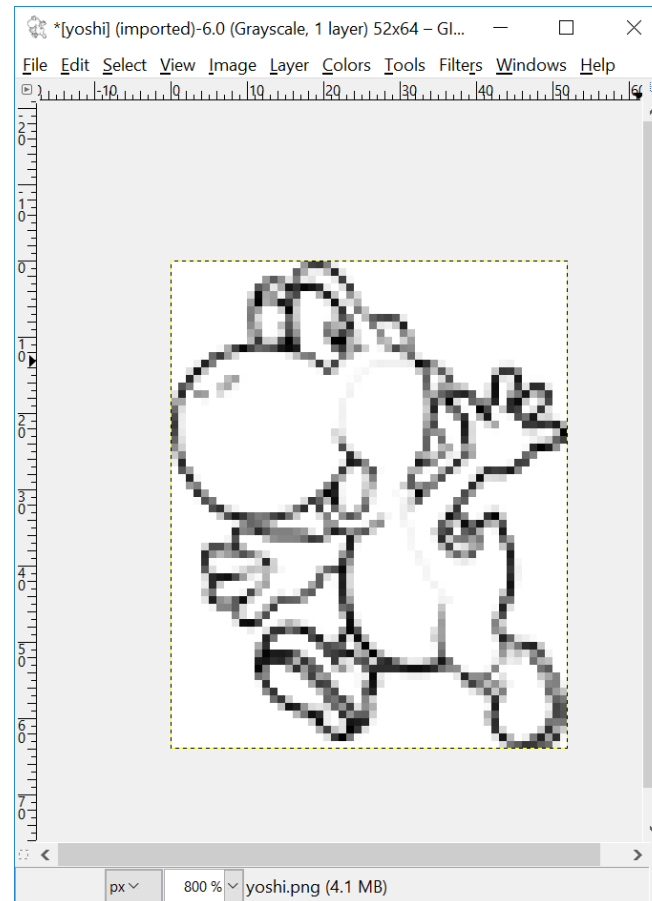
Välj Image->Scale Image...

Skala om till önskad storlek.



Design av grafik

Resultat
(efter rejäl inzoomning)

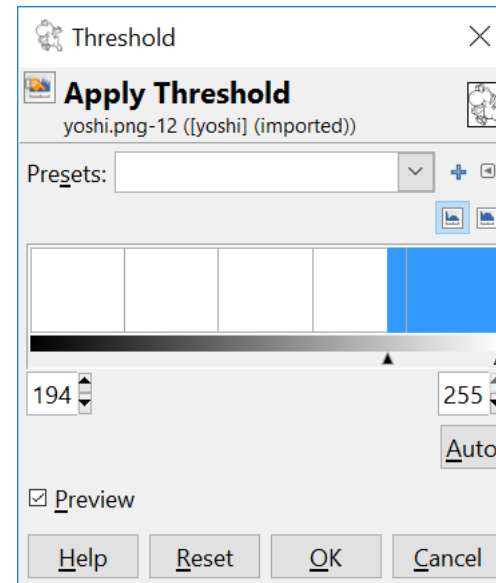


Design av grafik

Trunkera till 1-bits färg.

Välj Colors->Threshold...

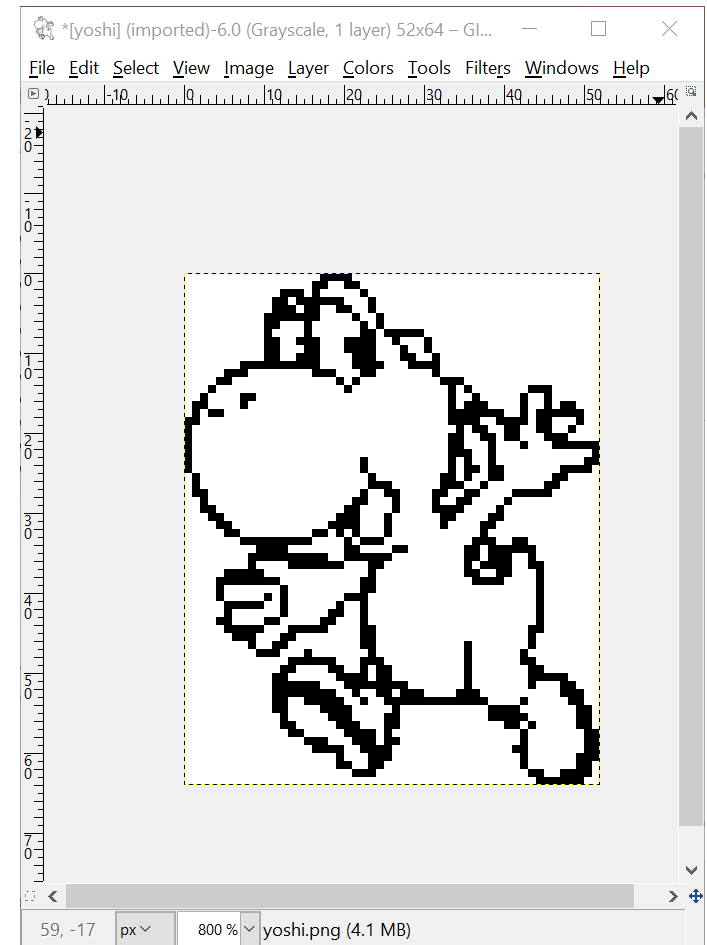
Dra i slidern tills det ser bra ut.



Design av grafik

Nu är sprite:n färdig att använda i programmet.

Notera: utgick från en svartvit bild. Kan även ta färgbild och konvertera till svartvit, men kan vara svårt att få bra resultat.



Design av grafik

Ni vill vi designa en funktion för att rita ut sprites på X Bitmap-format. Givet struct och hjälpfunktion nedan.

```
typedef struct
```

```
{
```

```
    unsigned char width;
```

```
    unsigned char height;
```

```
    unsigned char* data;
```

```
} sprite;
```

```
static void load_sprite(sprite* s, unsigned char* data, int width, int height)
```

```
{
```

```
    s->width = width;
```

```
    s->height = height;
```

```
    s->data = data;
```

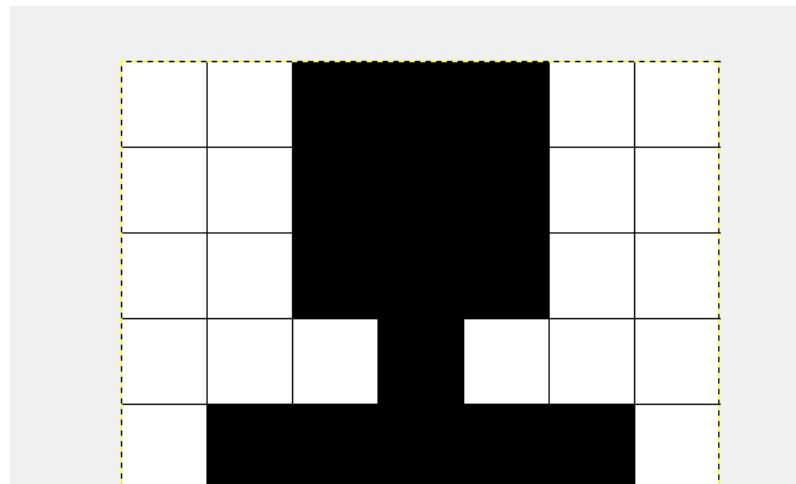
```
}
```

Design av grafik

```
person.xbm X
1  #define person_width 7
2  #define person_height 10
3  static unsigned char person_bits[] = {
4      0x1c, 0x1c, 0x1c, 0x08, 0x3e, 0x6b, 0x49, 0x1c, 0x14, 0x36 };
5
```

bit

1 2 3 4 5 6 7 8



Design av grafik

```
void draw_sprite(sprite* s, int x, int y, int set) {
    int i,j,k, width_in_bytes;

    if (s->width % 8 == 0)
        width_in_bytes = s->width / 8;
    else
        width_in_bytes = s->width / 8 + 1;

    for (i = 0; i < s->height; i++)
        for (j = 0; j < width_in_bytes; j++) {
            unsigned char byte = s->data[i * width_in_bytes + j];
            for (k = 0; k < 8; k++) {
                if (byte & (1 << k))
                    pixel(8 * j + k + x + 1, i + y + 1, set);
            }
        }
}
```