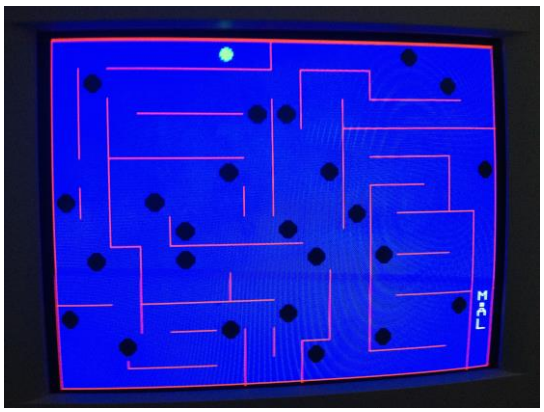
AGSTU
Arbete Genom STUdier
Utbildning

Digitalt Labyrint spel

Ingenjörsjobb 2



Sammanfattning

Spelet *Digitalt Labyrint spel* är skrivet i programmeringsspråket C För testkortet DE2-115 med en Cyclone IV FGPA. Till FPGA:n används en hårdvara tillhandahållen av Kunden. Som spelskärm ansluts en VGA-skärm till DE2-115 kortet, och med de fyra tryckknapparna styr spelaren kulan i spelet.

Själva utmaningen i spelet är att med hjälp av tryckknapparna få kulan att rulla genom labyrinten utan att rulla ner i något av de svarta hålen på vägen. Rullar kulan ner i ett hål stannar spelet och måste startas om med reset switchen.

Inlämningsuppgift inom: C

Författare: Patric Sjöberg

E-post: patric.sjoberg@home.se

Sommen Januari 2013

Innehåll

1 Inledning

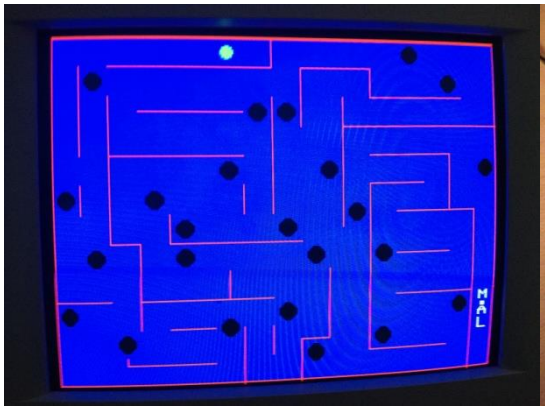
Företaget BRIO presenterade redan år 1946 det första exemplaret av spelet Labyrint. De har sedan dess sålt över 3 miljoner exemplar. Från detta spel hämtas inspirationen till det digitala labyrint spelet. Tanken är att spelaren ska få en kula att rulla igenom en labyrint med hjälp av knappar utan att kulan rullar i hål på vägen.

2 Kravspecifikation

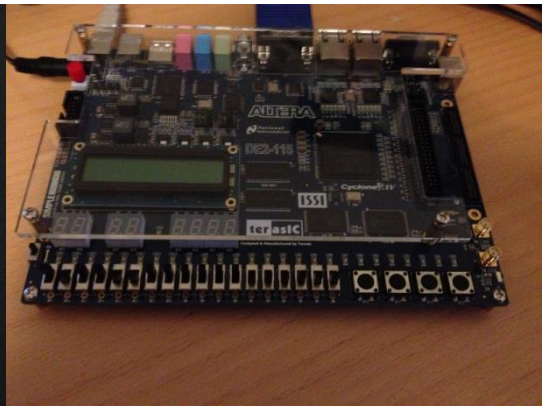
Bygg en spelprototyp på DE2-115 testkortet med en hårdvara tillhandahållen av kund. Spelet ska använda sig av en tidigare utvecklad VGA-komponent, men behöver inte vara helt fungerande. Konstruktionen ska vara väldokumenterad och även innehålla en spelinstruktion.

3 Konstruktionsbeskrivning

Den gröna kulan (se Figur 1)sätts i rullning när en eller två knappar på DE2-115 kortet (se Figur 2) trycks ner. Håller spelaren ner knapparna för ner och vänster samtidigt, rullar kulan ner åt vänster. Skulle ett hinder komma i vägen i en av riktningarna rullar den enbart åt den andra riktningen.



Figur 1



Figur 2

Spelet är programmerat så att kulan aldrig kan rulla över röd bakgrund, och att om den rullar över svart bakgrund så försvinner kulan och spelet är över.

3.1 Hårdvara

Hårdvaran som har tillhandahållits av kunden, består av följande komponenter:

CPU
RAM (327680 platser x 3-bitar brett),
VGA styrenhet med tre bitars färg och upplösning på 320x239 pixlar.
JTAG för datorkommunikation
4 tryckknappar
16 led dioder
LCD

IrDA
PGIO
Timers

I Tabell 1 nedan redogörs de olika funktionerna och vilka adresser de har.

Tabell 1 Nios II minnesadress karta

Slave Descriptor	Address Range	Size	Attributes/PIN
Terasic_IrDA_Rec_0	0x001010E0 - 0x001010E3	4	IR enheten
sysid	0x001010D8 - 0x001010DF	8	-
jtag_uart_0	0x001010D0 - 0x001010D7	8	Printf, scanf
Expansion_in	0x001010C0 - 0x001010CF	16	GPIO[7..0]
Expansion_out	0x001010B0 - 0x001010BF	16	GPIO[17..10]
lcd_0	0x001010A0 - 0x001010AF	16	LCD display
VGA_data_r	0x00101090 - 0x0010109F	16	VGA
VGA_Control_w	0x00101080 - 0x0010108F	16	VGA
VGA_data_w	0x00101070 - 0x0010107F	16	VGA
VGA_adress_w	0x00101060 - 0x0010106F	16	VGA
PERIPH_SYS_BUTTON_PIO	0x00101050 - 0x0010105F	16	Button 0-3
PERIPH_SYS_LEDS_PIO	0x00101040 - 0x0010104F	16	De längst till höger
PERIPH_HIGH_RES_TIMER	0x00101020 - 0x0010103F	32	timer
PERIPH_SYS_CLK_TIMER	0x00101000 - 0x0010101F	32	timer
onchip_memory2_0	0x00080000 - 0x000CFFFF	327680	memory

Det hårdvaruinterface som kunden har utvecklat i det här fallet använder sig av ett minne som speglar skärmen med 320x239 pixlar. För att läsa och skriva till minnet används några makron, vilka fungerar enligt följande specifikation.

Skrivning till pixel minnet (följande sekvens):

1. IOWR_ALTERA_AVALON_PIO_DATA(VGA_DATA_W_BASE,farg);

// tre bitar som bestämmer färger farg(2)= R farg(1) = G farg(0) = B

2. IOWR_ALTERA_AVALON_PIO_DATA(VGA_ADRESS_W_BASE,Pix_nr);

3. //Pix_nr = X + Y*320; //X = 1 – 320 Y = 1-239

4. IOWR_ALTERA_AVALON_PIO_DATA(VGA_CONTROL_W_BASE,1);

// skriver in i bildminnet

5. IOWR_ALTERA_AVALON_PIO_DATA(VGA_CONTROL_W_BASE,0);

// avsluta skrivningen, ska alltid vara noll när det inte skrivs.

Läsning från pixelminnet (Följande sekvens):

1. IOWR_ALTERA_AVALON_PIO_DATA(VGA_ADRESS_W_BASE,Pix_nr);

// Pix_nr = X + Y*320;

2. farg = IORD_ALTERA_AVALON_PIO_DATA(VGA_DATA_R_BASE);

// färger farg(2)= R farg(1) = G farg(0) = B

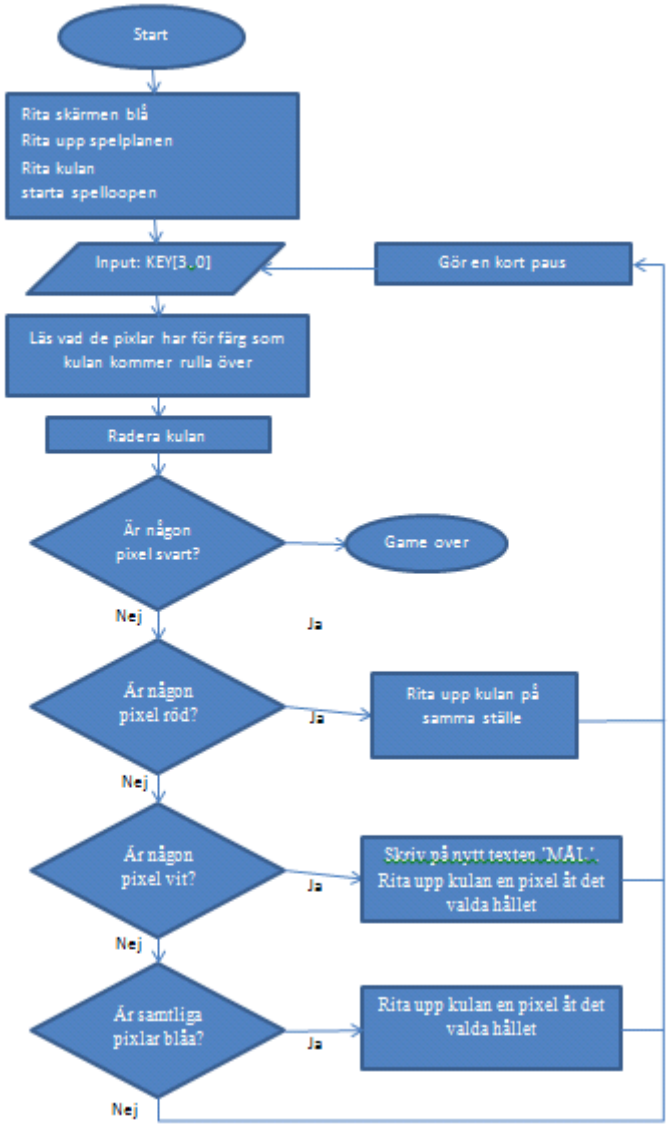
3.2 Mjukvara

Mjukvaran är uppbyggd i tre filer: 'C_Ingenjorsjobb_2.c' är huvudfilen där funktionen main() kör själva spelet. I filen Drawings.c finns de funktioner som main() använder, och Drawings.h som är header-filen till Drawings.c

Efter att ha ritat upp spelplanen så körs en While loop, som i varje varv känner av om någon eller några styrknappar är ned tryckta. I alla giltiga kombinationer av knapptryckningar så tar programmet först reda på hur spelplanen ser ut i den tänkta färdriktningen. Därefter flyttas om möjligt kulan åt det hållet. Efter att ha gjort en liten paus så kör loopen på nytt.

Figur 3 illustrerar lite förenklat hur spelet fungerar.

I följande avsitt redogörs de olika funktionerna.



Figur 3 Ett flödesdiagram över spelet

3.2.1 Print_pix

```
Print_pix (alt_u16 x, alt_u16 y, alt_u8 rgb)
```

Funktionen skriver en pixel i färgen rgb på positionen x och y. Alla funktioner som skriver till skärmen använder denna funktion. För att skriva ut en röd pixel på en position där x=50 och där y= 70 skrivs följande: **`Print_pix(50, 70, red)`**

3.2.2 print_Hline

```
print_Hline(alt_u16 x_start,alt_u16 y_start,alt_u16 len,alt_u8
rgb)
```

Denna funktion gör en horisontell linje på höjden y_start. Linjen har längden len, startpositionen x_start och färgen rgb. Print_pix() används för varje punkt i linjen. För att göra en blå horisontell linje som är hundra pixlar lång och där höjden y=50 och vänster kanten börjar där x=20 skrivs följande anrop: **`print_Hline(20, 50, 100, blue)`**

3.2.3 print_Vline

`print_Vline``(alt_u16 x_start,alt_u16 y_start,alt_u16 len,alt_u8 rgb)`
Funktionen skriver en vertikal linje med längden len med överkantens början på punkten x_start och y_start. För att rita en grön vertikal linje 50 pixlar in från vänster, med överkants början där y=30 och 100 pixlar lång, skrivs följande:

```
print_Vline(50,30,100,green)
```

3.2.4 print_char

`print_char``(alt_u16 x,alt_u16 y,alt_u8 rgb,alt_u8 BG_RGB,`**`char`**`Character)`
print_char ritar upp en bokstav med storleken 8.8 pixlar. Dessa pixlar finns definierade i en array med ett tecken för varje pixel. För varje nolla skirvs bakgrundsfärgen BG_RGB ut, och för varje etta skrivs teckenfärgen RGB ut. Tecknet skrivs ut med början på position x och y. För att skriva ett gult 'A' på blå bakgrund med positionen där x=58 och y=10, skrivs följande anrop:

```
print_char(58,10,yellow,blue,'A')
```

3.2.5 read_pixel_ram_int

`read_pixel_ram_int``(alt_u16 x_start, alt_u16 y_start)`
read_pixel_ram_int går en och läser av i minnet vilken färg en viss punkt (x_start och y_start) har. Färgen returneras sedan tillbaka som ett tal mellan 0-7. Vill man veta vilken färg pixeln på positionen där x=50 och y=70 är, så skriver man följande anrop:

```
read_pixel_ram_int(100,55)
```

3.2.6 clean_screen

`clean_screen``(alt_u8 rgb)`
Clean_screen over hela skärmen med den förvalda bakgrundsfärgen rgb.
För att rensa skärmen med blå bakgrund skrivs följande anrop:

```
clean_screen(blue)
```

3.2.7 print_circle

```
print_circle(unsigned int radie, unsigned int x_centrum,
 unsigned int y_centrum, int rgb)
```

Denna funktion ritar upp en fylld cirkel med radien 'radie' på position x_centrum och y_centrum, med färgen rgb. Print_circle används både till att rita upp kulan och hålen i labyrinten. Följande exempelkod ritar ett svart hål med radien 6 pixlar på position 30,30.

```
print_circle(6,30,30,black);
```

3.2.8 Next_step

```
Next_step(unsigned int radie, unsigned int x_centrum,
unsigned int y_centrum)
```

Next_step använder funktionen read_pixel_ram_int till att läsa av färgen på de pixlar som kulan kommer flyttas till i den valda färdriktningen. Alla de färger som har påträffats returneras därefter tillbaka (till main()).

3.2.9 DrawBoard

DrawBoard() använder sig av `print_Hline() och print_Vline()` till att rita upp labyrinten i röd färg. Print_circle ritar upp ett antal svarta hål och funktionen print_char skriver bokstäverna M, Å och L.

4 Verifiering och Validering

Varje delsystem har testats allt eftersom den har konstruerats. För att visuellt kunna se hur den tänkta spelplanen skulle se ut validerades systemet efter varje tillförd linje eller hål. Tester på att rulla kulan i alla tänkbara riktningar har gjorts över samtliga bakgrundsfärger. Slutligen har spelet spelats i sin helhet flera gånger. Se Figur 1 på sidan 1.

5 Slutsats och förbättringar

Att göra detta spel har varit både utmanande och lärorikt. Mjukvaran är begränsad till en hårdvara med enkel grafik och skulle kunna i en kommande version uppgraderas med snyggare grafik. Delen där kulan rullar ner i ett hål skulle kunna göras snyggare, men kräver att man vet vilket hål man närmar sig.

6 Bilagor

a) C_Ingenjorsjobb_2.c
b) Drawings.c
c) Drawings.h

```
/*
-- $Workfile : Ingenjosjobb_2.c
--
-- Programmer(s) : Patric Sjöberg
-- Date Created :  22 dec 2012
--
-- Description :Ett labyrintspel där man styr en kula med fyra knap-
par.
-- In_signals : KEY[3..0],X och Y koordinanter, rgb i heltal
-- Out_signals : pix_nr, rgb
 */

#include <stdio.h>
#include <system.h>
#include "altera_avalon_pio_regs.h"
#include "Drawings.h"


int main()
{
// deklarerar variabler
alt_u16 KEYS;
alt_u16 GameRun;
alt_u8 TheStep =0;
unsigned int x;
unsigned int y;

// initierar variabler
GameRun=1;
x = 120;
y = 10;
TheDirection = 0;

clean_screen(blue);           // skriver över hela skärmen i vald färg
      DrawBoard();                        // Rita upp Labyrinten
      print_circle(4,x,y,green);    // Rita upp Kulan
      printf ("klart");

      while(GameRun)
      {
            // Läs av KEY[3..0] och tilldela värdet till KEYS
KEYS = 0xF & IORD_ALTERA_AVALON_PIO_DATA(PERIPH_SYS_BUTTON_PIO_BASE);

            if(KEYS != 15)                        //Någon eller några
knappar är nedtryckta
            {
                  if(KEYS == 7)                 // Down
                  {

                        TheDirection = 1371;
                        TheStep = Next_step(4,x,y+1);
                        TheDirection = 0;
                        print_circle(4,x,y,blue);
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
```

```
                        {
                              y++;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {
                        y++;
                        print_circle(4,x,y,green);
                        }
                        //printf("%u \n", TheStep);
                        //printf("x:%u och y:%u\n", x, y);
                        TheStep = 0;
                  }
                  else if (KEYS == 11)    // Up
                  {
                        TheDirection = 2727;
                        TheStep = Next_step(4,x,y-1);
                        TheDirection = 0;
                        print_circle(4,x,y,blue);
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
                        {     y--;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {     y--;
                              print_circle(4,x,y,green);
                        }
                        //printf("%u \n", TheStep);
                        TheStep = 0;
                  }
                  else if (KEYS == 13)    // Left
                  {
                        TheDirection = 253;
                        TheStep = Next_step(4,x-1,y);
                        TheDirection = 0;
                        print_circle(4,x,y,blue);
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
                        {     x--;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {     x--;
                              print_circle(4,x,y,green);
                        }
```

```
                        //printf("%u \n", TheStep);
                        TheStep = 0;
                  }
                  else if (KEYS == 14)    // Right
                  {

                        TheDirection = 3854;
                        TheStep = Next_step(4,x+1,y);
                        TheDirection = 0;
                        print_circle(4,x,y,blue);
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
                        {     x++;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {
                              x++;
                              print_circle(4,x,y,green);
                        }
                        //printf("%u \n", TheStep);
                        TheStep = 0;
                  }
                  else if (KEYS == 5)     // DownLeft
                  { // down
                        TheDirection = 1371;
                        TheStep = Next_step(4,x,y+1);
                        TheDirection = 0;
                        TheDirection = 253;
                        print_circle(4,x,y,blue);
                        //Now left
                        if (TheStep & 0x2)TheStep = Next_step(4,x-1,y);
                        else
                        {y++;
                              TheStep = Next_step(4,x-1,y);
                        }
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
                        {
                              x--;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {
                              x--;
                              print_circle(4,x,y,green);
                        }
                        TheDirection = 0;
                        TheStep = 0;
```

```
                    }
                    else if (KEYS == 6)     // DownRight
                    { //Down
                          TheDirection = 1371;
                          TheStep = Next_step(4,x,y+1);
                          TheDirection = 0;
                          TheDirection = 3854;
                          print_circle(4,x,y,blue);
                          //Now right
                          if (TheStep & 0x2)TheStep = Next_step(4,x+1,y);
                          else
                          {y++;
                                TheStep = Next_step(4,x+1,y);
                          }
                          if (TheStep & 0x1)GameRun = 0;
                          else if (TheStep &
0x2)print_circle(4,x,y,green);
                          else if (TheStep & 0x4)
                          {
                                x++;
                                print_char(306,180,white,blue,'M');
                                print_char(306,190,white,blue,'Å');
                                print_char(306,200,white,blue,'L');
                                print_circle(4,x,y,green);
                          }
                          else if (TheStep & 0x8)
                          {
                                x++;
                                print_circle(4,x,y,green);
                          }
                          TheDirection = 0;
                          TheStep = 0;
                    }
                    else if (KEYS == 9)     // UpLeft
                    {     //Up
                          TheDirection = 2727;
                          TheStep = Next_step(4,x,y-1);
                          TheDirection = 0;
                          TheDirection = 253;
                          print_circle(4,x,y,blue);
                          //Now left
                          if (TheStep & 0x2)TheStep = Next_step(4,x-1,y);
                          else
                          {y--;
                                TheStep = Next_step(4,x-1,y);
                          }
                          if (TheStep & 0x1)GameRun = 0;
                          else if (TheStep &
0x2)print_circle(4,x,y,green);
                          else if (TheStep & 0x4)
                          {
                                x--;
                                print_char(306,180,white,blue,'M');
                                print_char(306,190,white,blue,'Å');
                                print_char(306,200,white,blue,'L');
                                print_circle(4,x,y,green);
                          }
                          else if (TheStep & 0x8)
                          {
```

```
                              x--;
                              print_circle(4,x,y,green);
                        }
                        TheDirection = 0;
                        TheStep = 0;
                  }
                  else if (KEYS == 10)    // UpRight
                  {     //Up
                        TheDirection = 2727;
                        TheStep = Next_step(4,x,y-1);
                        TheDirection = 0;
                        TheDirection = 3854;
                        print_circle(4,x,y,blue);
                        //Now right
                        if (TheStep & 0x2)TheStep = Next_step(4,x+1,y);
                        else
                        {y--;
                              TheStep = Next_step(4,x+1,y);
                        }
                        if (TheStep & 0x1)GameRun = 0;
                        else if (TheStep &
0x2)print_circle(4,x,y,green);
                        else if (TheStep & 0x4)
                        {
                              x++;
                              print_char(306,180,white,blue,'M');
                              print_char(306,190,white,blue,'Å');
                              print_char(306,200,white,blue,'L');
                              print_circle(4,x,y,green);
                        }
                        else if (TheStep & 0x8)
                        {
                              x++;
                              print_circle(4,x,y,green);
                        }
                        TheDirection = 0;
                        TheStep = 0;
                  }

            }
            //printf("%u \n", TheStep);
            //fördröjning med 5000 ms
            usleep(15000);
      }

  return 0;
}
```

```
/*
-- $Workfile : Drawings.c
--
-- Programmer(s) : Patric Sjöberg
-- Date Created :  22 dec 2012
--
-- Description :Funktioner till labyrintspelet där man styr en kula
med fyra knappar.
-- In_signals : X och Y koordinanter, rgb i heltal
-- Out_signals : pix_nr, rgb
 */


#include <stdio.h>
#include <system.h>
#include "altera_avalon_pio_regs.h"
#include "sys/alt_stdio.h"
#include "Drawings.h"

// print_pix skriver ut en pixel på position x och y med färgen rgb.
void print_pix
      (
            alt_u16 x,
            alt_u16 y,
            alt_u8 rgb
      )
      {
                  IOWR(VGA_DATA_W_BASE, 0, rgb); // The value (color)
that we want to write to our address.
                  IOWR(VGA_ADRESS_W_BASE, 0, x + y * 320); // The ad-
dress that we want to write our value on.
                  IOWR(VGA_CONTROL_W_BASE, 0, 1); // Start writing.
                  IOWR(VGA_CONTROL_W_BASE, 0, 0); // When written, stop
writing.
      }
// print_vline skirver ut en horisontell linje med start på x_start
och y_start
// som har längden len och färgen rgb.
void print_Hline
      (
            alt_u16 x_start,
            alt_u16 y_start,
            alt_u16 len,
            alt_u8 rgb
      )
      {
            alt_u16 f;
            for(f = x_start; f <= (x_start + len); f++)
            {
                  print_pix(f, y_start, rgb);
            }

      }
// print_vline skirver ut en vertikal linje med start på x_start och
y_start
// som har längden len och färgen rgb.
void print_Vline
      (
            alt_u16 x_start,
```

```
            alt_u16 y_start,
            alt_u16 len,
            alt_u8 rgb
      )
      {
            alt_u16 f;
            for(f = y_start; f <= (y_start + len); f++)
            {
                  print_pix(x_start,f , rgb);
            }

      }
//print_char skriver ut ett tecken med start position på x och y.
//Tecknen kan antingen vara M,Å eller L.
//Teckenfärg är rgb och bakgrundsfärg är BG_RGB.
void print_char(alt_u16 x,alt_u16 y,
            alt_u8 rgb,alt_u8 BG_RGB,char Character)
{
      alt_u8 index;
      alt_u8 i,j;


      alt_u16 array[30][8]={{0,0,0,0,0,0,0,0}, //teckentabell för A
                                    {0,1,0,0,0,0,0,1},
                                    {0,1,1,0,0,0,1,1},
                                    {0,1,0,1,0,1,0,1},
                                    {0,1,0,0,1,0,0,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,0,0,0,0,0,0,0},
                                    {0,0,0,0,0,0,0,0},
                                    {0,0,0,1,1,1,0,0}, //teckentabell
för B
                                    {0,0,0,1,0,1,0,0},
                                    {0,0,0,1,1,1,0,0},
                                    {0,0,0,0,0,0,0,0},
                                    {0,0,0,1,1,1,0,0},
                                    {0,0,1,0,0,0,1,0},
                                    {0,1,1,1,1,1,1,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,1,0,0,0,0,0,1},
                                    {0,0,0,0,0,0,0,0}, //teckentabell
för C
                                    {0,0,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,0,0,0,0,0,0},
                                    {0,1,1,1,1,1,1,1}};


      if(Character=='M')
            index=0;
      else if(Character=='Å')
```

```
            index=10;
      else if(Character=='L')
            index=20;
      else
      {
            index=30;
            alt_printf("Funktionen print_char fungerar bara med bokstä-
verna M,Å eller L\n");
      }
      if((x<1) | (x> 320) | (y < 1) | (y > 239))
                  alt_printf("Fel angivna intervall för x eller y
\n");
      else if (index<30)
              {

            for(i=0;i<=7;i++)
            {
                  for(j=index;j<=(index+9);j++)
                  {
                        if(array[j][i]==1)
                        {
                              print_pix((i+x), (j-index+y),rgb);

                        }

                        else

                     print_pix((i+x), (j-index+y),BG_RGB);
                  }
            }

              }

}
// Hämta rgb färgen från minnet på position x_start och y_start
alt_u8 read_pixel_ram_int(alt_u16 x_start, alt_u16 y_start)
{
      alt_u32 pix_nr;
      alt_u8 pixel_data=0;

      pix_nr=x_start+320*y_start;

      if((x_start<1) | (x_start> 320) | (y_start < 1) | (y_start >
239))
            alt_printf("Fel angivna intervall för x eller y \n");

      else
       {
        IOWR_ALTERA_AVALON_PIO_DATA(VGA_ADRESS_W_BASE,pix_nr);
        IOWR_ALTERA_AVALON_PIO_DATA(VGA_CONTROL_W_BASE,0);
        pixel_data = IORD_ALTERA_AVALON_PIO_DATA(VGA_DATA_R_BASE);
       }

      return pixel_data;
}
void clean_screen(alt_u8 rgb) //rensa skärmen
      {
      for(y = 1; y <= 239; y++)
      {
```

```
            for(x = 1; x <= 320; x++)
            print_pix(x, y, rgb);
      }
      }
// ritar en fylld cirkel med centrum i punkten x_centrum och y_centrum
// med färgen rgb och radien radie
void print_circle
            (
            unsigned int radie,
            unsigned int x_centrum,
            unsigned int y_centrum,
            unsigned int rgb
            )
//(int x0, int y0, int radius, alt_u8 rgb);

{
  int f = 1 - radie;
  int ddF_x = 1;
  int ddF_y = -2 * radie;
  int x = 0;
  int y = radie;

  print_Hline(x_centrum-radie, y_centrum, (2 * radie), rgb);

  while(x < y)
  {
    // ddF_x == 2 * x + 1;
    // ddF_y == -2 * y;
    // f == x*x + y*y - radie*radie + 2*x - y + 1;
    if(f >= 0)
    {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;

    print_Hline((x_centrum - x),(y_centrum + y),(2 * x), rgb);
    print_Hline((x_centrum - x),(y_centrum - y),(2 * x), rgb);
    print_Hline((x_centrum - y),(y_centrum + x),(2 * y), rgb);
    print_Hline((x_centrum - y),(y_centrum - x),(2 * y), rgb);

  }
}
alt_u8 Next_step
            (
            unsigned int radie,
            unsigned int x_centrum,
            unsigned int y_centrum
            )
//(int x0, int y0, int radius, alt_u8 rgb);

{
  int f = 1 - radie;
  int ddF_x = 1;
  int ddF_y = -2 * radie;
  int x = 0;
```

```
  int y = radie;
  alt_u8 ThePix;
  alt_u8 pix_data =0;//sätt värdet till '1' om det är ett hål, vägg,
eller text.


  if(TheDirection & 0x1)      // Väst     1
        { ThePix = read_pixel_ram_int((x_centrum-radie),y_centrum);
             if (ThePix == black)pix_data = pix_data |= 1 ;
              if (ThePix == red)pix_data = pix_data |= 2 ;
              if (ThePix == white)pix_data = pix_data |= 4 ;
              if (ThePix == blue)pix_data = pix_data |= 8 ;
        }
  if(TheDirection & 0x2)      // öst      2
        {ThePix = read_pixel_ram_int((x_centrum+radie),y_centrum);
             if (ThePix == black)pix_data = pix_data |= 1 ;
              if (ThePix == red)pix_data = pix_data |= 2 ;
              if (ThePix == white)pix_data = pix_data |= 4 ;
              if (ThePix == blue)pix_data = pix_data |= 8 ;}
  if(TheDirection & 0x4)      // Nord     3
  {ThePix = read_pixel_ram_int((x_centrum),(y_centrum-radie));
             if (ThePix == black)pix_data = pix_data |= 1 ;
              if (ThePix == red)pix_data = pix_data |= 2 ;
              if (ThePix == white)pix_data = pix_data |= 4 ;
              if (ThePix == blue)pix_data = pix_data |= 8 ;}
  if(TheDirection & 0x8)      // Syd      4
  {ThePix = read_pixel_ram_int((x_centrum),(y_centrum+radie));
             if (ThePix == black)pix_data = pix_data |= 1 ;
              if (ThePix == red)pix_data = pix_data |= 2 ;
              if (ThePix == white)pix_data = pix_data |= 4 ;
              if (ThePix == blue)pix_data = pix_data |= 8 ;}
  while(x < y)
  {
    // ddF_x == 2 * x + 1;
    // ddF_y == -2 * y;
    // f == x*x + y*y - radie*radie + 2*x - y + 1;
    if(f >= 0)
    {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;

    if(TheDirection & 0x10)   //syd-sydväst           5
    {ThePix = read_pixel_ram_int((x_centrum - x),(y_centrum + y));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x20)   // Nord-NordVäst  6
    {ThePix = read_pixel_ram_int((x_centrum - x),(y_centrum - y));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x40)   //Väst-sydväst          7
```

```
    {ThePix = read_pixel_ram_int((x_centrum - y),(y_centrum + x));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x80)   // Väst-Nordväst  8
    {ThePix = read_pixel_ram_int((x_centrum - y),(y_centrum - x));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x100)  // Syd-sydost           9
    {ThePix = read_pixel_ram_int((x_centrum + x),(y_centrum + y));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x200)  // Nord-NordOst         10
    {ThePix = read_pixel_ram_int((x_centrum + x),(y_centrum - y));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x400)  // Ost-SydOst           11
    {ThePix = read_pixel_ram_int((x_centrum + y),(y_centrum + x));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
    if(TheDirection & 0x800)  // Ost-NordOst          12
    {ThePix = read_pixel_ram_int((x_centrum + y),(y_centrum - x));
                  if (ThePix == black)pix_data = pix_data |= 1 ;
                   if (ThePix == red)pix_data = pix_data |= 2 ;
                   if (ThePix == white)pix_data = pix_data |= 4 ;
                   if (ThePix == blue)pix_data = pix_data |= 8 ;}
  }
  return pix_data;
}
//rita upp labyrinten
void DrawBoard()
{
      print_Hline(0,0,319,red);
      print_Hline(0,1,319,red);
      print_Hline(0,239,319,red);
      print_Vline(1,0,239,red);
      print_Vline(319,0,239,red);
      print_Vline(150,0,20,red);
      print_circle(6,30,30,black);
      print_Hline(40,20,110,red);
      print_Vline(20,20,60,red);
      print_Vline(20,100,20,red);
      print_circle(6,10,110,black);
      print_Vline(40,40,100,red);
      print_circle(6,30,150,black);
      print_Hline(0,180,40,red);
      print_Hline(40,140,20,red);
      print_Vline(60,140,60,red);
      print_circle(6,50,210,black);
      print_circle(6,10,190,black);
```

```
        print_Vline(50,220,5,red);
        print_Hline(50,225,60,red);
        print_Vline(130,200,40,red);
        print_circle(6,120,200,black);
        print_Hline(80,200,30,red);
        print_Hline(60,180,90,red);
        print_circle(6,160,190,black);
        print_Vline(150,200,20,red);
        print_Vline(170,210,30,red);
        print_Hline(170,210,20,red);
        print_Vline(190,160,50,red);
        print_circle(6,180,150,black);
        print_Hline(80,140,90,red);
        print_Vline(120,160,20,red);
        print_circle(6,90,150,black);
        print_Vline(80,120,20,red);
        print_circle(6,70,110,black);
        print_Hline(40,80,90,red);
        print_circle(6,120,90,black);
        print_Vline(130,100,20,red);
        print_circle(6,90,130,black);
        print_circle(6,160,130,black);
        print_Vline(150,40,80,red);
        print_circle(6,140,50,black);
        print_Hline(60,50,70,red);
        print_circle(6,160,50,black);
        print_Vline(170,20,40,red);
        print_Hline(170,20,50,red);
        print_circle(6,250,10,black);
        print_Vline(220,20,20,red);
        print_Hline(220,40,70,red);
        print_circle(6,280,30,black);
        print_Hline(200,60,120,red);
        print_Vline(200,40,90,red);
        print_circle(6,190,90,black);
        print_circle(6,210,120,black);
        print_Vline(220,100,120,red);
        print_circle(6,180,220,black);
        print_Hline(220,220,60,red);
        print_circle(6,290,190,black);
        print_Vline(300,120,120,red);
        print_Hline(240,180,60,red);
        print_circle(6,230,210,black);
        print_circle(6,230,150,black);
        print_Hline(240,150,40,red);
        print_Hline(240,120,60,red);
        print_Hline(220,100,40,red);
        print_Vline(280,80,40,red);
        print_Hline(220,80,60,red);
        print_circle(6,310,90,black);
        print_char(306,180,white,blue,'M');
        print_char(306,190,white,blue,'Å');
        print_char(306,200,white,blue,'L');
}
```

```
/*
 * Drawings.h
 *
 *  Created on: 28 dec 2012
 *      Author: Psj
 */

#ifndef DRAWINGS_H_
#define DRAWINGS_H_

#define black 0
#define blue 1
#define green 2
#define cyan 3
#define red 4
#define magenta 5
#define yellow 6
#define white 7

unsigned int x;
unsigned int y;
unsigned int rgb;
alt_u16 TheDirection;
unsigned int BG_RGB;
unsigned int pixel_data;

void print_pix(alt_u16 x, alt_u16 y, alt_u8 rgb);
void print_Hline(alt_u16 x_start, alt_u16 y_start, alt_u16 len, alt_u8
rgb);
void print_Vline(alt_u16 x_start, alt_u16 y_start, alt_u16 len, alt_u8
rgb);
alt_u8 read_pixel_ram_int(alt_u16 x_start, alt_u16 y_start);
void print_char(alt_u16 x, alt_u16 y, alt_u8 rgb,alt_u8 BG_RGB,char
Character);
void clean_screen(alt_u8 rgb);
void print_circle(unsigned int radie, unsigned int x_centrum, unsigned
int y_centrum, unsigned int rgb);
alt_u8 Next_step(unsigned int radie,unsigned int x_centrum,unsigned
int y_centrum);
void DrawBoard();
#endif /* DRAWINGS_H_ */
```