AGSTU TEIS 2014/2015

SLUTRAPPORT
SPELET BLACK JACK I C

Magnus Mårtensson, magnus.maartensson@gmail.com
Magnus Flysjö, magnus@flysjo.com

2015-01-03

Slutrapport - konstruktion av kortspelet Black Jack i C.

INNEHÅLL

1 - KRAVSPECIFIKATION SLUTLEVERANS

Följande ska genomföras i slutleveransen:

Krav	Beskrivning
Förstudie	
Krav_001	Krav på konstruktionskoden: 1) ”filhuvud” i alla C-filer, 2) Namn på variabler m.m. ska vara tydliga, 3) Kontroll av gränsvärden ska ske, t.ex. att värdet inte ligger utanför skärmen, då ska ett felmeddelande skrivas ut, 4) Koden ska vara rikligt kommenterad. 5) Alla drivrutiner ska finnas i BSP 6) Strukturerad kod, inte bara en applikationsfil för hela spelet
Krav_002	Spel specifikation, testprotokoll, SW arkitekturen, hierarkin, minneskarta, alla I/O enheter och de viktiga funktionerna tydligt i ett kapitel (mycket från start rapporten). Testprotokollet ska minst testa ett fall från varje krav.
Krav_003	Tidsanalys av den färdiga koden. Svara på frågan om hur mycket ledig tid som finns i systemet. Frekvensen för NIOS är 50 MHz (Ändrad 20141205).
Krav_004	Alla drivrutinerna ska finnas i komponenterna under HAL/src och inc. ”tcl” filerna ska definiera alla drivrutinerna för respektive komponent, så att de kommer med i BSP foldern (inte under applikations folder).
Krav_005	**Följande drivrutiner är beställda och ska vara med i BSP (ska vara under VGA/HDL/src och inc, laddas in i BSP)**: En kort SW manual för VGA drivrutinerna (exempel se Sierra SW manual). Även exekveringstiden ska vara defilerad för varje funktion. Koden ska vara robust skriven. **Funktion:** print_pix(unsigned int x,unsigned int y,unsigned int rgb); **Funktionsbeskrivning:** Skriver en pixel med färgen *rgb* på koordinaten (x, y).

	Funktion: print_hline(unsigned int x_start, unsigned int y_start, unsigned int len, unsigned int RGB); **Funktionsbeskrivning:** Skriver en horisontell linje med färgen *rgb* och med längden *len* vilken startar på koordinaten (x_start, y_start). **Funktion:** print_vline(unsigned int x_start, unsigned int y_start, unsigned int len, unsigned int RGB); **Funktionsbeskrivning:** Skriver en vertikal linje med färgen *rgb* och med längden *len* vilken startar på koordinaten (x_start, y_start). **Funktion:** print_char(unsigned int x,unsigned int y,unsigned int rgb,unsigned int BG_RGB,char Character); **Funktionsbeskrivning:** Skriver tecknet "Character" med färgen *rgb* och med bakgrundsfärgen "BG_RGB" på koordinaten (x, y). **NOTERA!** En del tecken fungerar inte med den här funktionen, (lägg till de ASCI som behövs för att skriva det som krävs). Det går att bestämma hur tecknen ska visas på skärmen, se bilaga, äldre projekt och på webben. **Funktion:** read_pixel_ram_int(unsigned int x_start, unsigned int y_start); return: unsigned pixel_data (obs! enbart de sista tre bitarna är intressanta). **Funktionsbeskrivning:** Läser pixel_data från pixel RAM (3 bitar, RGB) från adress x och y (beräknat från x och y). **Funktion:** print_circle(unsigned int radie, unsigned int x_centrum, unsigned int y_centrum, unsigned int rgb); **Funktionsbeskrivning:** Skriver en cirkel med radien "radie" och färgen" rgb" på mittkoordinaten (x, y). Cirkeln ska fyllas med samma färg. **Funktion**: clear_screen(int rgb); **Funktionsbeskrivning**: Denna funktion rensar skärmen genom att RGB-värdet skrivs till alla pixlar på skärmen.
Krav_006	Beskriv hur koden har optimerats för hastighet och storlek med inställningar i verktygen och i koden. Beskriv minnets arkitektur och motivera varför vissa delar av koden/data har länkats till olika minnen.

Krav_007	Spela in en kort film på Youtube, som valideringsbevis att spelet fungerar och lägg in ”AGSTUTEIS” som ett av sökorden. För inspiration titta på äldre projekt (TEIS, AGSTU och FPGA). Filmen ska börja med en kort förklaring av spelet, därefter en översiktsbild av SW konstruktionen och till sist ett smakprov på spelet. Gör filmen lik en presentation av hela projektet och det ska hållas kort (max 3 minuter).
Dokumentationskrav	
Krav_008	Leveransen ska innehålla tre foldrar: • **Konstruktionsbeskrivning** o Konstruktionsrapport på ett standardiserat sätt, (Krav 3, 6) o En kort SW manual för VGA drivrutinerna (se krav 5) o SW filerna i en mapp, (Krav 1) o Alla IP komponenter, Timer, VGA IP med dess drivrutiner. (Krav 4) • **Spel** o En enkel spelmanual (beskriv hur användaren ska spela spelet), (Krav 2) • **Diverse** o Start rapporten
Leveranskrav	
Krav_009	Leveransen ska ske till plattformen Itslearning. Namnet på filen ska vara ”förnamn_efternamn_C_ingenjorsjobb_b.zip”.

2 - BESKRIVNING AV PROJEKT

2.1 - SPELET BLACK JACK

Black Jack är ett kortspel där det gäller att komma så nära talet 21 som möjligt utan att bli ”tjock”. Tjock blir man ifall summan på korten överstiger 21. Varje kort är värt ett visst antal poäng och det gäller att vara den spelare som kommer närmast 21. Klädda kort är värda 10 poäng, ett ess antingen 1 eller 11. Övriga kort är värda så mycket som de representerar.

Spelet börjar med att blanda 2 kortlekar, d.v.s. 104 kort totalt. Spelare och givare får sedan två kort var från kortleken. Spelaren anger sedan om fler kort skall dras från kortleken. Det gäller då att det dragna kortet inte ger en summa som överstiger 21. Så länge spelarens kortsumma inte är 21 eller mer kan fler kort dras från kortleken.

Spelet stödjer endast en spelare, denne spelar alltid mot givaren. Givaren måste alltid dra ett nytt kort om summan på givarens kort understiger 17 och alltid stanna om summan är 17 eller högre.

Om både spelare och givare får samma summa på korten blir spelet oavgjort.

2.2 - KRAVSPECIFIKATION

Spelet skall exekveras av erhållen hårdvara från kund på DE2-115 kortet med en ansluten VGA skärm. Hela spelet är baserat på JA/NEJ frågor från givaren. Genom att trycka på ”KEY0” på DE2-115 kortet svarar man ”NEJ” och genom att trycka på ”KEY1” svarar man ”JA”.

Vid uppstart fylls skärmen med en grön bakgrund och texten ”BLACK JACK” på en rad, samt en text som säger ”STARTA SPEL?”. Genom att trycka på ”KEY1” startar spelet.

Kortleken blandas vid uppstart men inte mellan varje spel. Använda kort från kortleken är således förbrukade för nästa spelomgång tills hela kortleken använts. När alla korten är förbrukade blandas en ny kortlek med 104 kort och spelet fortsätter automatiskt.

Skapa en kortlek som innehåller 104 kort (2x52). Varje kort skall innehålla färg (hjärter, spader, ruter eller klöver) samt kortets värde (1=Ess, 11=Knäckt, 12=Dam, 13=Kung, osv).

Blandningsrutinen skall använda ”rand” för att slumpa fram en färg (0...3) samt ett värde (1...13). Finns den slumpade kombinationen redan i kortleken två gånger skall en ny kombination slumpas fram till dess att ett icke existerande kort erhålls. Detta upprepas för att generera en blandad kortlek på 104 kort.

Vid blandning av kortleken skall texten ”BLANDAR KORT” visas på skärmen.
Spelarens kort visas överst på skärmen. Givarens kort visas nederst på skärmen. Kortens symboler/färg (ruter, klöver, hjärter eller spader) samt värde ska visas grafiskt på korten på

skärmen. Vid klädda kort räcker det att representera dess värde med en bokstav. Knekt = Kn, Dam = D , Kung = K. Ett ess får även representeras av bokstaven ”E”.

När givare och spelare fått två kort tilldelat skall frågan ”VILL DU HA ETT TILL KORT?” visas på skärmen. Frågan upprepas så länge spelaren väljer att få ett till kort och så länge spelaren inte blivit tjock.

När spelaren inte längre vill ha ett nytt kort så är det givarens tur att dra kort. Givaren måste dra ett till kort om summan understiger 17, och stanna ifall summan är 17 eller mer.

När givaren är färdig och ingen spelare blivit tjock så jämförs summorna på korten. Den som är närmast 21 vinner.

- Vid vinst ska texten ”DU VANN” visas i 2 sekunder.
- Vid summa över 21 ska texten ”TJOCK” visas i 2 sekunder.
- Vid förlust ska texten ”DU FÖRLORADE” visas i 2 sekunder.
- Vid samma summa mellan givare och spelare ska texten ”OAVGJORT” visas i 2 sekunder.

När en spelomgång är över ska texten ”Vill du spela igen?” visas.

Genom att trycka på ”KEY4” på DE2-115 kortet ska det gå att se hela kortleken på VGA skärmen. Även vilken position som är nästa kort ur leken ska visas. Detta används för att testa att blandningen av kortleken har skett tillfredsställande samt för att fuska. Genom att trycka på ”KEY3” på DE2-115 i detta läge ska man också kunna blanda om kortleken.

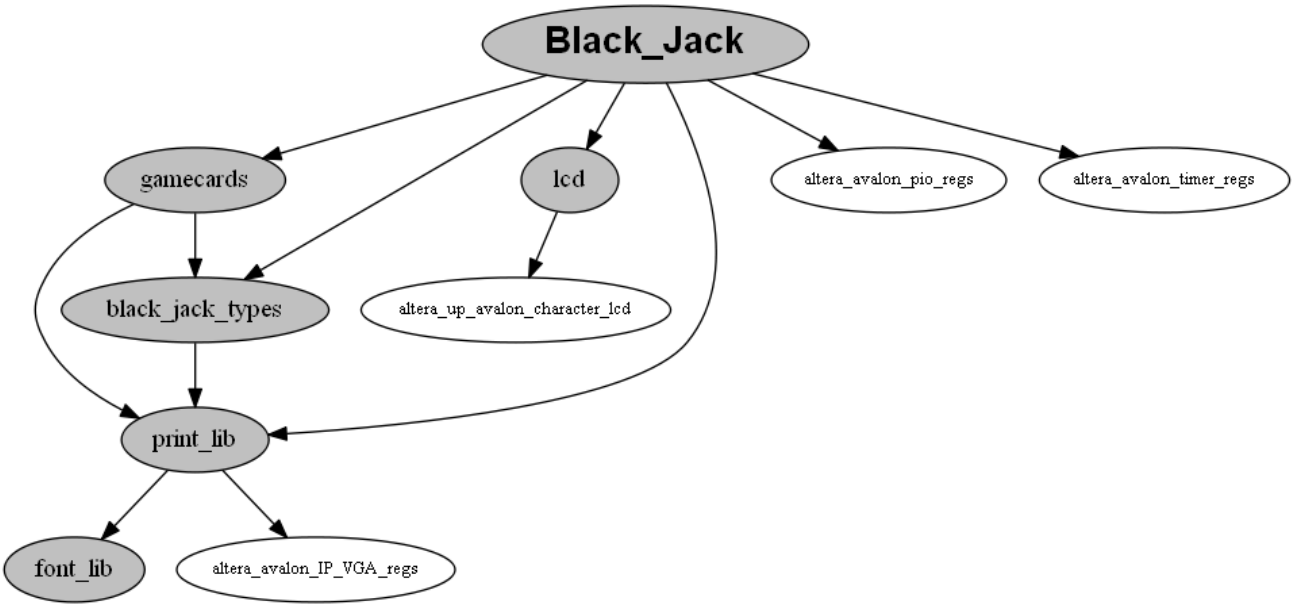
2.3 - TESTPROTOKOLL

Testfall	Beskrivning	Ok
1	Två kortlekar skall blandas.	Ett specifikt kort får endast finnas två gånger i kortleken.
2	Vid KEY4 nedtryckt skall kortleken visas.	En lista på alla 104 kort visas på skärmen.
3	Vid spelstart ska givare och spelare få kort.	Två kort för spelaren samt ett kort för givaren ska visas direkt vid spelstart.
4	Begäran om fler kort från spelare ska fungera.	Om spelaren trycker på KEY1 skall fler kort delas ut under förutsättning att spelaren inte har fått över 21. Om spelaren trycker på KEY0 går turen över till givaren.
5	Givaren måste dra kort på rätt sätt.	Om givaren har en kortsumma under 17 måste ett till kort dras. Vid en summa av 17 eller mer måste givaren stanna och spelomgången avslutas.
6	Spelomgången ska avslutas korrekt.	Om spelaren och givare fått samma summa blir det oavgjort. Om spelaren får över 21 (blir tjock) förlorar spelaren. Om givaren får över 21 vinner spelaren. I övriga fall vinner den som ligger närmst 21.
7	Blandning av kortlek ska fungera.	Om spelaren trycker på KEY3 när hela kortleken visas (se punkt 2) skall hela kortleken blandas om. Om spelaren trycker på KEY3 i annat läge ska kortleken inte blandas om

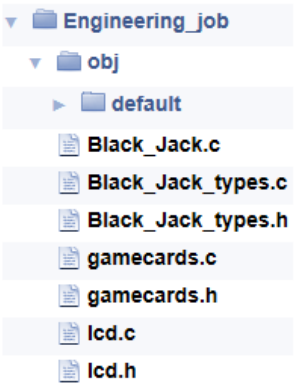
3 - MJUKVARA

3.1 – HIERARKI OCH SW ARKITEKTUREN

SW arkitekturen för Black Jack ser ut enligt nedan, se figurerna 1,2 och 3:



Figur 1: Sw arkitekturen för spelet Black Jack

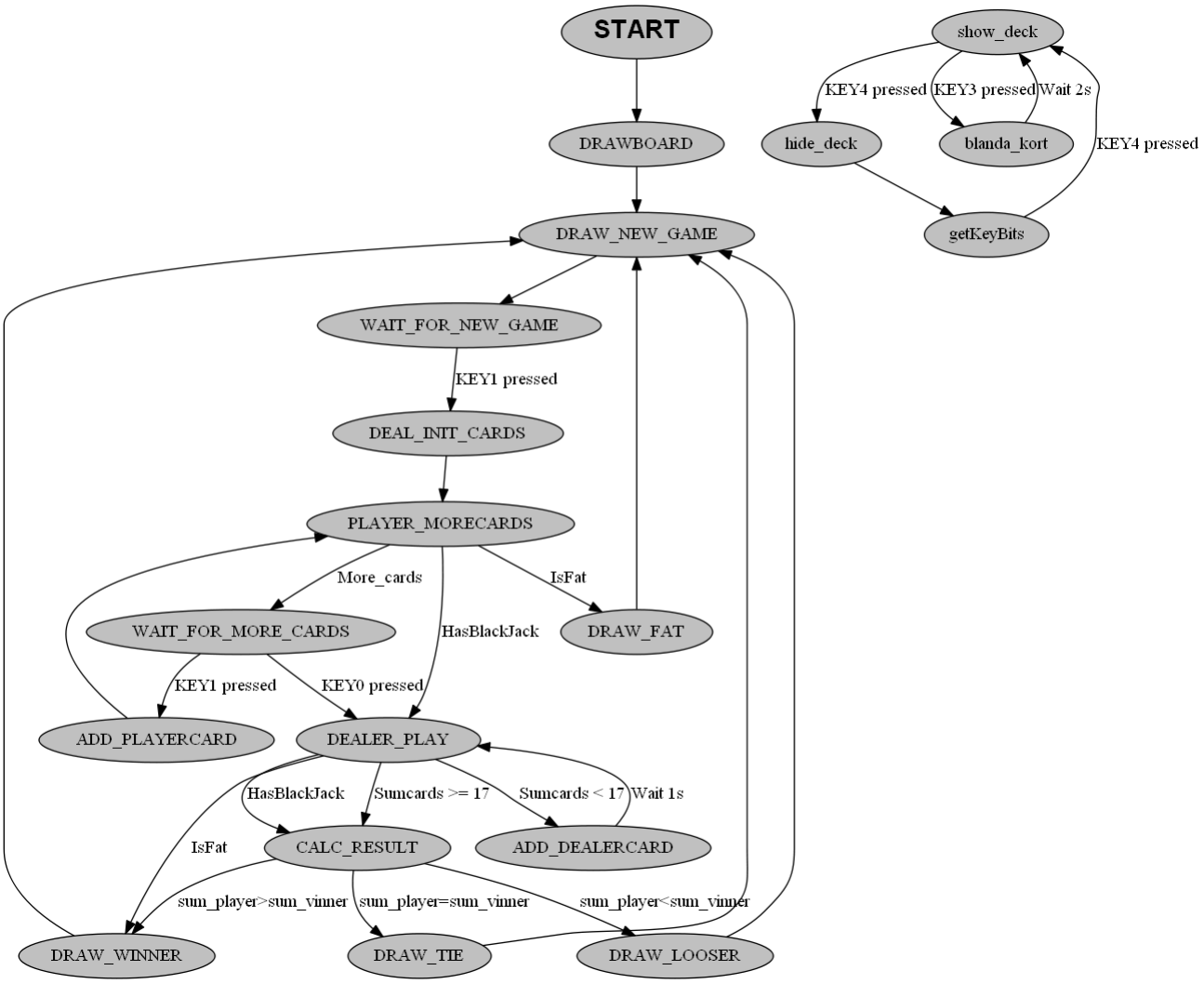


Figur 2: Sw arkitekturen

Figur 3: Sw arkitekturen BSP

3.2 – STATE-MASKINEN

State maskinen för Black jack ser ut enligt nedan:



Figur 4: State maskinen i Black_Jack.c

3.3 – FUNKTIONER

Nedan beskrivs de funktioner som används i respektive mjukvarufil.

3.3.1 – GAMECARDS.H /GAMECARDS.C

Globala funktioner tillgängliga i Gamecards:

```
void init_new_game(void);
Nollställer och förbereder "händerna" på spelaren och givaren inför nytt spel.
Ser till så att spelaren och givaren inte har några gamla kort.

void shuffle_deck (void);
Blandar slumpmässigt korten i kortlekarna.

void erase_cards (void);
Raderar alla kort som ligger på bordet.
Endast pixels som används av korten skrivs över/raderas med bakgrundsfärgen.

void AddCardToPlayer(void);
Funktion för att tilldela och rita ut nytt kort till spelare, men innan dess
kontrolleras att det finns fler kort i leken.
Uppdaterar också aktuell summa för spelarens kort.

void AddCardToDealer(void);
Funktion för att tilldela och rita ut nytt kort till givaren, men innan dess
kontrolleras att det finns fler kort i leken.
Uppdaterar också aktuell summa för givarens kort.
Anropar give_new_card_to_dealer();

void give_new_card_to_player(void);
Tar ett kort från kortleken och ger till spelaren.
Kortindex räknas upp för både kortleken och spelaren.

void give_new_card_to_dealer(void);
Tar ett kort från kortleken och ger till givaren.
Kortindex räknas upp för både kortleken och givaren.

alt_u8 IsCardsFat(TCards *cards);
Funktionen kontrollerar om spelare eller givare har blivit fet, fått resultat över 21.

alt_u8 HasBlackJack(TCards *cards);
Funktionen kontrollerar om spelare eller givare har fått Black Jack, dvs ett resultat
lika med 21.

alt_u8 sum_cards(TCards *cards);
Funktionen kontrollerar om spelare eller givare har fått Black Jack, dvs ett resultat
lika med 21.
```

```
void draw_board(alt_u8 color);
```
Ritar upp ramen runt spelplanen.

```
void draw_entire_deck(DrawType draw, TCards *cards);
```
Ritar upp samtliga kort i kortleken på höger sida av spelplanen, som "spelhjälp".

```
void draw_nextcard_marker(alt_u8 aIdx, alt_u8 color);
```
Ritar upp markören framför nästa lediga kort i "spelhjälpen".

```
void hide_nextcard_marker(void);
```
Döljer markören framför nästa lediga kort i "spelhjälpen".
Används för få markören att "blinka".

```
void init_score(deal player, alt_u8 value, alt_u8 color);
```
Initierar och skriver ut text för antal vinster spelare och givare.

```
void update_score(deal player, alt_u8 value, alt_u8 color);
```
Tar bort gammalt värde och uppdaterar med ny text för antal vinster spelare och givare.

```
void update_card_result(deal player, TCards *cards, alt_u8 color);
```
Skriver ut aktuella poäng för spelarens eller givarens hand.

Lokala funktioner tillgängliga i Gamecards:

```
void swap_cards(struct TCard *a, struct TCard *b);
```
Skiftar plats på två kort. Används för att blanda kortleken.

```
void shuffle_if_needed(void);
```
Om alla 104 kort är använda skrivs en text ut och kortleken blandas.
Används under pågående spel.

```
void draw_card(DrawType draw, deal player, TCards *cards);
```
Ritar upp ett kort för spelare eller givare med värde, symboler och eventuella figurer.
Använder funktionerna draw_blank_card och draw_card_symbols.

```
void draw_blank_card(alt_u16 xPos, alt_u16 yPos,bool border);
```
Ritar upp ett blankt spelkort på positionen xPos samt YPos med skiljelinje från föregående kort.

```
void draw_card_symbols(DrawType draw, alt_u16 xPos, alt_u16 yPos, deck symbol, alt_u8 value);
```
Ritar ut symboler och siffror på ett aktuellt kort. På klädda kort ritas också ramen ut.
Använder funktionerna draw_char, draw_symbol, draw_dressed_cards samt draw_card_frame.

```
void draw_card_frame (alt_u16 xPos, alt_u16 yPos, alt_u8 color);
```
Ritar ut symboler och siffror på ett aktuellt kort. På klädda kort ritas också ramen ut.
Använder funktionerna draw_char, draw_symbol, draw_dressed_cards samt draw_card_frame.

```
char* IntToStr(int i, char txt[]);
```
Funktion för att konvertera en integer till en textsträng.

```
void printArray (void);
```
Skriver ut innehållet i den blandade kortleken i NIOS II konsol.
För debugging.

3.3.2 – LCD.H/LCD.C

Globala funktioner tillgängliga i Gamecards:

```
void init_lcd(void);
Funktion för att sätta upp/initiera lcd:n.

void write_lcd_welcometext(void);
Funktion för att skriva ut en fast välkomsttext på lcd:n.
```

3.3.3 – BLACK_JACK_TYPES.H/ BLACK_JACK_TYPES.

Inga globala eller lokala funktioner finns tillgängliga i Black_Jack_types.

3.3.4 – PRINT_LIB.H / PRINT_LIB.C

Globala funktioner tillgängliga i print_lib:

```
void clear_screen(alt_u8 rgb);
Tömmer skärmen med färgen rgb

void print_pix(register alt_u16 xPos, register alt_u16 yPos,register alt_u8 rgb);
Skriver en pixel i positionen för X samt Y med färgen rgb

alt_u8 read_pixel_ram(alt_u16 xPos, alt_u16 yPos);
Läser och returnerar färgen rgb för en specific pixel i positionen X samt Y.

void draw_hline(alt_u16 x_start, alt_u16 y_start, alt_u16 length, alt_u8 rgb);
Ritar en horisontell linje med startpunkter xPos samt yPos, längden length och färgen
rgb.

void draw_vline(alt_u16 x_start, alt_u16 y_start, alt_u16 length, alt_u8 rgb);
Ritar en vertikal linje med startpunkter xPos samt yPos, längden length och färgen rgb.

void draw_char(alt_u8 ch, alt_u16 x_start, alt_u16 y_start, alt_u8 rgb, alt_u8
upsideDown);
Skriver ut en bokstav på positionen xPos samt YPos färgen rgb. Skriver även ut bokstaven
upp och ner om så önskas. Använder fonttabellen för "fontdata_16x20_1252" i font_lib.h

void draw_circle(alt_u8 radius, alt_u16 xPos, alt_u16 yPos, alt_u8 rgb);
Ritar ut en fylld cirkel på positionen xPos samt YPos, önskad radie och med färgen rgb.

void draw_text(char *str, alt_u16 xPos, alt_u16 yPos, alt_u8 rgb, Alignment alignment);
Skriv ut en text i xPos samt yPos, med färgen rgb och önskad alignment (vänster eller
centrerad). Använder fonttabellen för "fontdata_16x20_1252" i font_lib.h

alt_u16 calc_text_width(char *str);
Beräkna bredden på en sträng i pixels.
```

```
void draw_symbol(DrawType draw, alt_u16 xPos, alt_u16 yPos, alt_u8 symIdx, alt_u8 rgb,
alt_u8 upsideDown);
```
Ritar upp en symbol på ett kort. Ritar också symbolen upp och ner om så önskas. Använder fonttabellen för "cardsymbols" i font_lib.h.

```
void draw_dressed_cards(DrawType draw, alt_u16 xPos, alt_u16 yPos, alt_u8 symIdx,
alt_u8 rgb, alt_u8 upsideDown);
```
Ritar upp en figuren för dom klädda korten. Ritar också figuren upp och ner. När man använder upside down byggs dom båda halvorna ihop till en helfigur. För att skapa helfiguren räcker det dock inte att spegla symbolen utan den måste roteras 180grader. Använder fonttabellen för "dressedcards" i font_lib.h.

```
void draw_small_card_value(DrawType draw, alt_u16 xPos, alt_u16 yPos, deck symbol,
alt_u8 value);
```
Ritar ut små symboler och siffror med rätt förg och värde i xPos samt yPos för "spelhjälpen”.

3.3.5 – FONT_LIB.H

Font_lib.h innehåller inga egna funktioner.

Den används av print_lib.c och innehåller konstant data tabeller över fonter och symboler.

Fonterna kan ritas manuellt alternativ genereras automatiskt genom import från Windows med hjälp av pc verktyget FontSymEdit.exe.

Fonter och symboler kan genereras i godtycklig storlek.

“FontSymEdit.exe” stöder Windows Codepage SBCS (Single byte character set) samt MBCS (Multi byte character set), se http://msdn.microsoft.com/en-us/goglobal/bb964654

3.4 – OPTIMERINGAR

- C++ är deaktiverat i BSP.
- Vid leverans är debug avstängt.
- För att öka hastigheten för vissa funktioner som används mest frekvent, har ”register” använts. Ett exempel är för funktionen print_pix(), ett annat Clear_screen() där det sker väldigt många beräkningar av pixel positionen.
- Vid radering gör man inte clean_screen eller liknande utan man skriver symbolen, fonten eller korten, pixel för pixel med bakgrundsfärgen för att minimera skrivningen av antalet pixel med print_pixel().
- Ramar runt klädda kort ingår inte i symbolen utan dom skrivs ut separat.
- Makros har använts för konstanter.
- Samma funktioner för att skriva symboler/fonter upp och ner på korten.

3.5 – TIDSANALYS

BLACK JACK är inte speciellt tidskritiskt.
De fördröjningar som blir när man när man ritar upp och raderar kort upplevs närmast realistiska, ungefär som vid en giv med riktiga kort. Vid utskrift av vissa texter har vi använt usleep() ibland under flera sekunder, för att man skall hinna uppfatta och läsa texterna.

Att rita upp ett nytt kort tar ungefär 0.5 Sekunder, lite beroende på valören på kortet.
Radera korten tar något längre, också beroende på hur många kort som skall raderas.

Rita upp eller radera spelhjälpen tar ca 1.3 s vilket är ganska tidskrävande. Spelhjälpen tillsammans med Clear_screen är de mest tidskrävande funktionerna. Clear_screen utförs dock bara vid uppstart och reset, spelhjälpen används normalt inte vid ett spel.

Hur mycket tid som finns över i systemet avgörs av hur frekvent man trycker fram nya kort/startar nytt spel.

4 – HÅRDVARA

De olika komponenterna och deras funktion är beskrivna enligt nedan:

4.1 - ALTERA_UP_AVALON_CHARACTER_LCD

LCD-komponenten låter dig skriva ut text till LCD-enheten på DE2-115 kortet.
För att kunna använda LCD komponenten måste följande rad läggas till i C koden:

```
#include <altera_up_avalon_character_lcd.h>
```

Filen innehåller bland annat följande funktioner:

- **alt_up_character_lcd_init**
- **alt_up_character_lcd_write**
- **alt_up_character_lcd_set_cursor_pos**
- **alt_up_character_lcd_erase_pos**

4.2 - IP_VGA

IP_VGA-komponenten används för att kunna använda en VGA-skärm.
IP_VGA-komponenten följer VGA protokollet med en upplösning 640x480 pixlar, 8 färger.
Ett dubbelportsminne används för att skriva och läsa pixel data till skärmen.

För att kunna använda IP_VGA-komponenten måste följande rad läggas till i C koden:

```
#include <altera_avalon_IP_VGA_regs.h>
```

C-koden kommunicerar med VGA minnet med hjälp av två HAL kommando:

- VGA_WRITE(address, data) IOWR_32DIRECT(IP_VGA_BASE, address * 4,data);
- VGA_READ(address) IORD_32DIRECT(IP_VGA_BASE, address * 4);

4.3 - TERASIC_SRAM

SRAM används för att ge konstruktionen tillgång till mer minne.

Hur minnet skall användas konfigureras i projektets BSP (Board Support Package).

Adressen till SRAM är 0x0200000. Minnet är 1MB stort.

All kod inklusive konstant data sparas i SRAM.

4.4 - TIMER

Används för att kunna mäta tid i konstruktionen
För att kunna använda timern måste följande rad läggas till i C koden:

```
#include <altera_avalon_timer_regs.h>
```

C-koden kommunicerar med timern med hjälp av två HAL kommando:

- TIMER_RESET
- TIMER_START
- TIMER_READ

4.5 - PIO

Periphial In/Out används för alla tryckknappar, switchar och lysdioder.
Den används också för VGA_SYNC_N, d.v.s. signalen som säger när man skall skriva till VGA minnet.

För att kunna använda PIO måste följande rader läggas till i C koden:

```
#include <io.h>
#include <system.h>
#include <altera_avalon_pio_regs.h>
```

C-koden kommunicerar med timern med hjälp av HAL kommando, t.ex:

- IOWR_8DIRECT
- IORD_8DIRECT
- IOWR_32DIRECT
- IORD_32DIRECT
- IORD_ALTERA_AVALON_PIO_DATA

och makros för de olika adresserna, t.ex:

- PIO_OUT_LED_BASE
- PIO_IN_KEY_BASE

4.6 - MINNESMAPPNING

Adresserna till de olika komponenterna för Black Jack kan ses i tabell 1 nedan.

Tabell 1 - Minnesmappning

Slave Descriptor	Address Range	Size	Attributes
character_lcd	0x00381054-0x00381055	2	
terasic_IrDA_Rec_0	0x00381050-0x00381053	4	
jtag_uart	0x00381048-0x0038104F	8	Printable
Sysid	0x00381040-0x00381047	8	
pio_in_key	0x00381030-0x0038103F	16	
pio_out_led	0x00381020-0x0038102F	16	
MY_TIMER	0x00381010-0x0038101F	16	
pio_0	0x00381000-0x0038100F	16	
onchip_ram	0x00340000 -0x0036FFFF	196608	Memory
TERASIC_SRAM_0	0x00200000 -0x002FFFFF	1048576	Memory
MY_VGA	0x00000000 -0x001FFFFF	2097152	

5 – VERIFIERING OCH VALIDERING

5.1 - VERIFIERING

Testfallen har verifierats enligt nedan:

Testfall	Beskrivning	Ok	Resultat
1	Två kortlekar skall blandas.	Ett specifikt kort får endast finnas två gånger i kortleken.	OK
2	Vid KEY4 nedtryckt skall kortleken visas.	En lista på alla 104 kort visas på skärmen.	OK
3	Vid spelstart ska givare och spelare få kort.	Två kort för spelaren samt ett kort för givaren ska visas direkt vid spelstart.	
4	Begäran om fler kort från spelare ska fungera.	Om spelaren trycker på KEY1 skall fler kort delas ut under förutsättning att spelaren inte har fått över 21. Om spelaren trycker på KEY0 går turen över till givaren.	OK
5	Givaren måste dra kort på rätt sätt.	Om givaren har en kortsumma under 17 måste ett till kort dras. Vid en summa av 17 eller mer måste givaren stanna och spelomgången avslutas.	OK

6	Spelomgången ska avslutas korrekt.	Om spelaren och givare fått samma summa blir det oavgjort. Om spelaren får över 21 (blir tjock) förlorar spelaren. Om givaren får över 21 vinner spelaren. I övriga fall vinner den som ligger närmst 21.	OK
7	Blandning av kortlek ska fungera.	Om spelaren trycker på KEY3 när hela kortleken visas (se punkt 2) skall hela kortleken blandas om. Om spelaren trycker på KEY3 i annat läge ska kortleken inte blandas om	OK

5.2 – VALIDERING

En film som ger en förklaring av spelet, en översiktsbild av mjukvaru konstruktionen, och ett smakprov av spelet finns tillgängligt på nedanstående länk:

https://www.youtube.com/watch?v=0PP2qzwRzzw

6 - VERKTYG

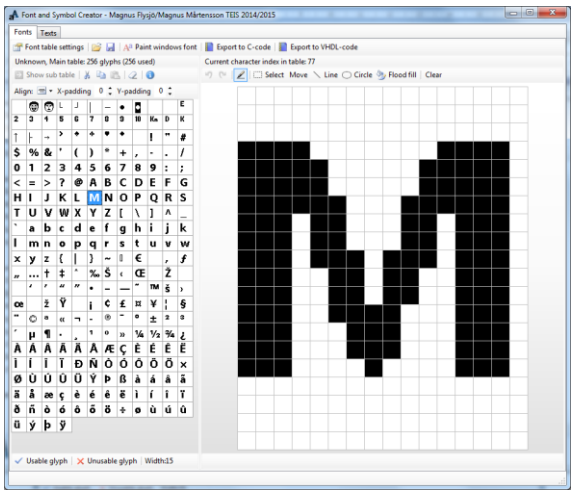
Vi insåg ganska fort att symboler, fonter och klädda kort skulle bli en utmaning att generera och rita upp. Handlar det bara om några få fonter kan man rita och avkoda dessa manuellt, men för detta projekt krävdes något bättre.
Som en del av detta har därför har ett PC verktyg (FontSymEdit.exe) tagits fram för att på ett enklare sätt generera önskade fonter och symboler både för C projektet och VHDL projektet.
I konstruktionen finns dessa sparade i tre olika filer:

- Font.fnt (innehåller samliga 256 ASCII för codepage 1252, men där några av de lägre systemtecknen blivit ersatta av kortsymboler och mindre storlek på siffrorna (se figur 5).
- KortSymboler.fnt Innehåller de fyra symbolerna för klöver (se figur 6).
- DressedCards.fnt Innehåller halvfigurer för de klädda korten (se figur 7).

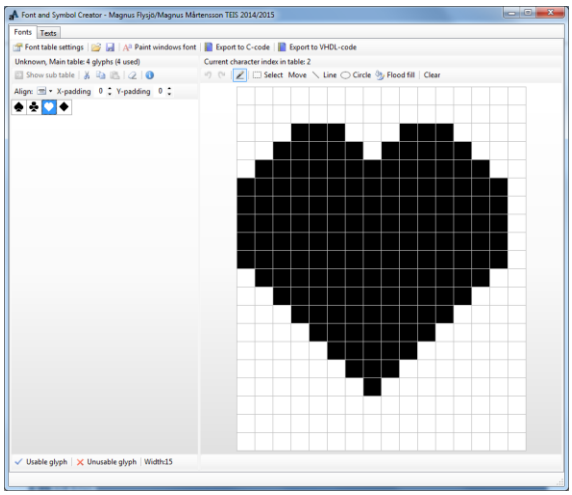
Ovanstående ger oss också möjligheten att skriva våra kära Å,Ä och Ö, men också att möjligheten att internationalisera applikationen om detta blir intressant.

Innehållet i filerna genereras sedan ut som hexdata och är inkluderade i filen font_lib.h.

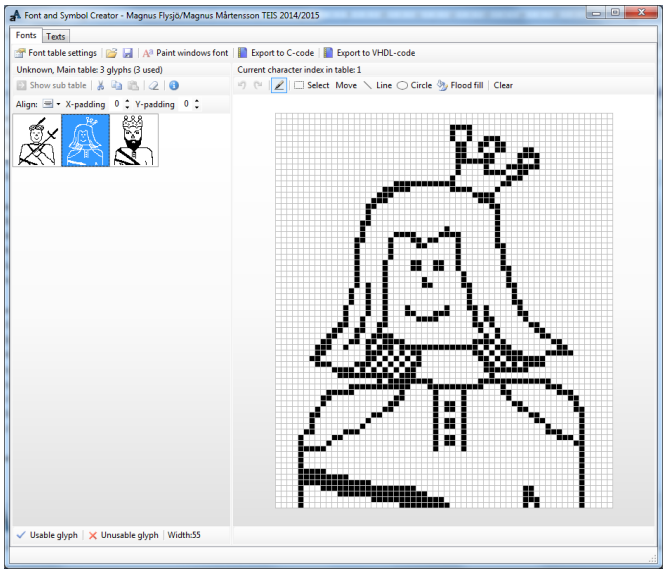
I mjukvaran krävs sedan funktionalitet för att skriva ut, spegla och i fallet med klädda kort, rotera och sätta ihop två halvfigurer till en helfigur.



Figur 5: Exempel från Font.fnt



Figur 6: Exempel från KortSymboler.fnt



Figur 7: Exempel från DressedCards.fnt

7 - SLUTSATSER OCH FRAMTIDA FÖRBÄTTRINGAR

Funktionen clear_screen tar lång tid. Istället för att anropa print_pix() funktionen för varje pixel borde man skriva till VGA_WRITE() direkt.

LCD displayen skriver inte ut alla tecken eftersom man loopar igenom strängen fortare än vad hårdvaran klarar av. En kort fördröjning (1ms) har lagts till i funktionen alt_up_character_lcd_string i filen altera_up_avalon_character_lcd.c för att få IP komponenten för altera_up_avalon_character_lcd att fungera och skriva ut alla tecken.

Symbolerna för klädda kort skulle kunna förbättras i projektet.

Intressant att jobba i team, särskilt då man sitter på olika orter och kanske inte har tillgång till en riktig fildelningsserver. Svårigheter när man behöver åtkomst till samma filer, samtidigt en stor fördel om man kör fast. Vi har valt att fildela via dropbox och även om det är långt ifrån optimalt, har det fungerat hyfsat bra. Kräver dock lite disciplin eftersom man lätt kan skriva sönder något för varandra.

8 - REFERENSER

Material har hämtats från:

- Low level C- Programming for Microcontrollers – Lennart Lindh
- DE2-115_User_manual.pdf, Altera

9 - BILAGOR

Samtliga bilagor och konstruktions kod nedan bifogas inkluderade i samma zip-fil: ”Magnus_Mårtensson_Magnus_Flysjö_C_ingenjorsjobb_b.zip”.

Följande filer har inkluderats:

- Slutrapporten ”Magnus_Mårtensson_Magnus_Flysjö_C_ingenjorsjobb_b.pdf ”
- I mappen diverse:
  Startrapporten ”Magnus_Mårtensson_Magnus_Flysjö_C_ingenjorsjobb_a.pdf ”
- I mappen konstruktionsfiler, samtliga SW filer:
  Black_Jack.c
  Black_Jack_types.h
  Black_Jack_types.c
  gamecards.h
  gamecards.c
  lcd.h
  lcd.c
  DE2_HW_engjob med nya och uppdaterade filer:
  \IP\IP_VGA\HAL\inc\font_lib.h
  \IP\IP_VGA\HAL\inc\print_lib.h
  \IP\IP_VGA\HAL\src\inc\ print_lib.c
  \IP\IP_VGA\HAL\IP_VGA_sw.tcl (utökad)
  \IP\IP_VGA\doc\Magnus_Flysjö_Magnus_Mårtensson__IP_VGA.pdf
- I mappen dokumentation:
  Spelmanual Black Jack.pdf
  Magnus_Flysjö_Magnus_Mårtensson__IP_VGA.pdf
- I mappen Extra – Fonter:
  FontSymEdit.exe (pc verktyg)
  Font.fnt
  KortSymboler.fnt
  DressedCards.fnt