30010 - Programmeringsprojekt

Reflexball

Gruppe 3

Martin Boye Brunsgaard, s144012(1)
Tore Gederaas Kanstad, s144021(2)
Peter Asbjørn Leer Bysted, s144045(3)

1 2 3

Alle medlemmer har været tilstede under øvelserne, og deltaget i udarbejdelse af journalerne. Ydermere har arbejdet været fordelt ligeligt over gruppemedlemerne, og løst i fællesskab. Rapporten er blevet udarbejdet og gennemlæst i kollektiv.

Tecnical University of Denmark DTU
National Space Institute

30010 - Programming Project
25.06.2015

Abstract

This report is a mandatory part of the B.Sc. EE course 30010, Programming Project.
The report documents the entirety of this course, including the exercise journals and the final product, Reflexball, a program written in C and implemented on a microprocessor.
The first part of the course was learning how to use the ZDS II - Z8Encore! 4.9.3 tools, how to access the timers, the LED's, the buttons and how to use the PuTTY for displaying graphic. The code from these exercises were used to implement the HAL and some of the API for the project. The program was designed using a flow chart for the main function and a block diagram for the different program layers. The project was successfully implemented on a Z8 Encore Evaluation Board.

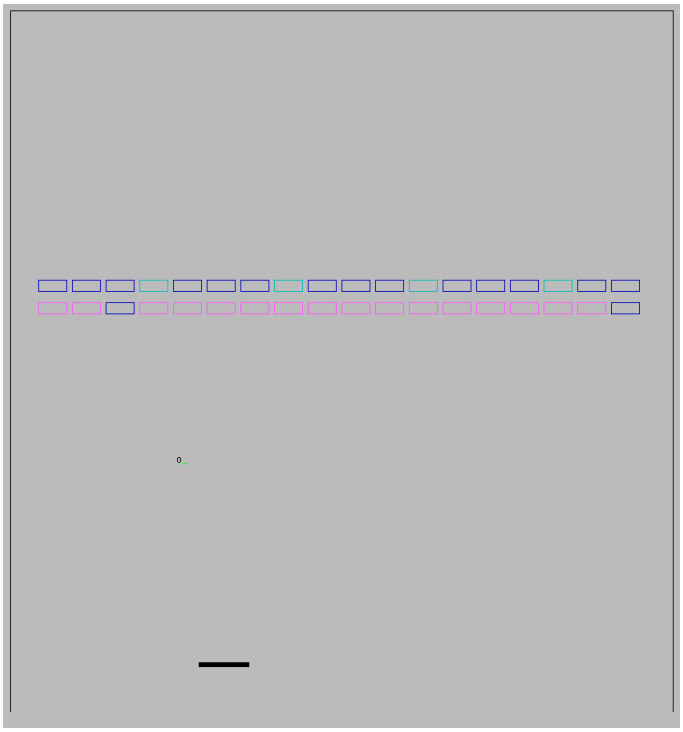
Resume

Denne rapport er en obligatorisk del af 30010, Programmeringsprojektet, som er en af de teknologiske linjefag på B.sc, EE.
Denne rapport dokumenterer helheden af dette kursus, inklusive journalerne og det endelige produkt, Reflexball, der er et program skrevet i C og implementeret på en mikroprocessor. I de første fire dage lærte forfatterne at bruge ZDS II - Z8Encore! 4.9.3 værktøjskassen, at konfigurere timerne, at vise strenge på LED'erne, at læse inputs fra knapperne og at bruge PuTTY til at vise grafik. Koden fra øvelserne blev brugt til at lave et HAL og en stor del af programmets API. Programmet blev designet ved hjælp af bla. flow charts og block diagrammer til at repræsentere programmets lag. Programmet blev med success implenteret på et Z8 Encore Evaluation Board.

Indhold

1 Introduktion

Målet med dette projekt er at designe og implementere et program. Programmet skal skrives i C og det skal implementeres på en Zilog Z8 encore microprocessor vha. ZDS II - Z8Encore! 4.9.3 værktøjer. Programmet skal dokumenteres vha. flowcharts, grafer og beskrivelser af de enkelte funktioner.



Figur 1: Reflexball vist i PuTTY.

Programmet skal være et spil, Reflexball. Spilleren styrer en striker, som skal bruges til at reflektere en bold, således den kan bevæge sig rundt på banen. Hvis bolden rammer en af kanterne, skal bolden ligeledes også reflekteres. Hvis spilleren ikke rammer bolden ryger bolden ud af banen, og spilleren fratrækkes et liv. Såfremt spilleren ikke har flere liv tilbage, afsluttes spillet. Desuden indføres der nogle bokse i spillet, som spilleren skal ødelægge. Når spilleren har ødelagt alle disse bokse går spilleren videre til næste bane, eller vinder såfremt han er på sidste bane. Den grafiske flade bliver implementeret ved at skrive til en terminal. Ydermere får brugeren fremvist informationer fra spillet på LED'erne på boardet.

2 Teori

Vi vil i dette afsnit gennemgå den basale teori bag binære tal og slutteligt indføre læseren i de forskellige formater, deriblandt fixed-point format, og hvorfor det er interessant at bruge denne repræsentation i vores projekt.

2.1 Binære tal

Et binært tal er et tal der kan udtrykkes i det binære talsystem/base-2, hvor grundtallet er 2. Binære tal er meget lette at implementere i digital logik, og det binære talsystem bruges derfor internt i computere verden over.
Et binært tal består af bits, som svarer til et ciffer. Et bit kan have en af to tilstande: logisk højt eller logisk lavt. Dette medfører da hvis vi har n bits har vi 2^n forskellige tilstande. Disse forskellige tilstande kan fortolkes på forskellige måder, og vi vil i de næste afsnit gennemgå nogle af de forskellige representationer.

2.2 Unsigned repræsentation

I det binære talsystem er grundtallet vanligvis 2(det kunne potentielt også være -2). Det betyder således at i en n-bit streng, vil bittet yderst til højre være vægtet med 2^0, det næste med 2^1 op til 2^{n-1} gående mod venstre. Tallet 5(base-10) kan da skrives som i ligning 1. Ydermere tæller vi også fra højre mod venstre, og første bit står således også på 0 plads. Dette bit kaldes oftest LSB(least significant bit), hvorimod det bit der står helt til venstre oftest kaldes MSB(most significant bit) [2, s. 18].

$$5_{10} = 101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \tag{1}$$

Med de indførte definitioner har vi kun mulighed for at repræsentere positive heltal. Vi ønsker også at kunne skrive kommatal og negative tal.

2.3 Fixed point kommatal

Kommatal kan indføres på en simpel måde, ved blot at vægte i omvendt retning når man går mod højre, således at bittet til højre for kommaet har vægtningen 2^{-1}, bittet 2 til højre for kommat vægtningen 2^{-2} osv. Hvis man har en n-bit streng med b tal til højre for kommaet, har man da muligheden for at skrive tal mellem 0 og $\frac{2^n-1}{2^b}$ [1, s. 4]

Tallet 13.625 kan f.eks skrives som

$$13.625_{10} = 1101.101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-3} \quad (2)$$

2.4 Repræsentation af negative tal

Hvis vi ønsker at repræsentere negative tal, gøres det oftest på 3 forskellige måder: Signed magnitude, 1's kompliment eller 2's kompliment. Vi vil her gennemgå signed magnitude og 2's komplement.

2.4.1 Signed magnitude

En af måderne at repræsentere fortegnet på bit-strengen, er ved at lade det mest signifikante bit(MSB: længst til venstre) bestemme fortegnet, hvor 0 indikerer et positivt tal og 1 indikerer et negativt tal. F.eks. kan tallet -37 i signed magnitude repræsentation skrives således:

$$-37_{10} = 1100101_2 \quad (3)$$

Signed magnitude repræsentation, har dog den ulempe, at man spilder et bit, f.eks. hvis man har en 4-bit streng gælder der at $1000 = 0000$, så istedet for at have 2^4 tilstande har man blot $2^4 - 1$. Desuden er signed magnitude ikke så velegnet til brug i digitale systemer.[2, s. 260]

2.4.2 2's komplement

En anden måde at repræsentere negative tal, kan gøres vha. 2's komplement. 2's komplement findes ved at invertere et unsigned tal og derefter lægge 1 til. 2's komplement har to fordele: Der er kun et 0, og subtraktion kan gøres på samme måde som addition, så hvis vi ønsker at subtrahere 3 fra 5, skal vi blot finde 2's kompliment af 3 og lægge det til 5, som i ligning 4. Disse fordele gør 2's komplement en god metode til at repræsentere tal i digitale systemer.

$$5 - 3 = 5 + (-3) \quad (4)$$

Ydermere har 2's komplement en cyklisk natur, der gør den smart at bruge sammen med triogonmetriske funktioner, såsom cos og sin. Den cykliske natur gør os istand til at gå begge veje rundt i enhedscirklen Dette brugte vi i øvelse 4, hvor vi sørgede for kun at se på bit 0-8.

2.5 Fixed point vs floating

I dette projekt brugte vi ikke floating-point typer. Dette skyldes at Z8 serien er en 8-bit processor, og disse bruger oftest ALU's(Arithmetic Logic Unit) designet til fixed point aritmetik.[3, s. 5]. Det er muligt at få processorer med indbyggede FPU(floating-point units), der er optimerede til at arbejde med floating-point typer, eller coprocessorer til at supplere CPU'en. Hvis vi insisterede på at arbejde i floating-point format, ville vi være nødt til at bruge Zilog's bibliotek til floating-point, hvilket ville være for langsomt til vores behov[4][5]. Vi brugte derfor kun integers, og omdannede dem til fixed point.

3 Design af Reflexball

I udarbejdelsen af dette program havde vi nogle forskellige tekniske krav og nogle designmål, som vi ønskede at designe programmet efter.

3.1 Tekniske mål

Vi lavede en liste af krav til programmets tekniske opbygning som vi i så høj grad som muligt ønskede at overholde.

1. Vi ønsker en veldefineret struktur. Vi vil derfor undgå globale variable i så høj grad som muligt, derfor skal vi lave funktioner som tager pegere til strukturer eller variable som inputs, frem for at tilgå globale variable. Få undtagelser findes dog til dette, f.eks. i modulet der tilgår timeren og LED skærmene.

2. Vi vil udvikle nogle moduler der er uafhængige af hinanden, således at vores grafik i mindst mulig grad kommunikerer med vores modul indeholdende spillets regler(refball.h). Denne kommunikation skal foregå igennem main-metoden, således man let kan få et overblik ved at se på main-metoden.

3. Vi vil bruge timerne på boardet til at styre tidsaspekter i spillet.

3.2 Krav til spillet

3.2.1 Overordnede krav til spillet

1. Spillet er et arkanoid spil, bestående af 5 levels. Banerne skal være i stigende sværhedsgrad. Dette gøres ved at boksene gøres stærkere, således de skal rammes flere gange for at ødelægges, og også tilføje flere kasser.

2. Der skal være mulighed for at vælge sværhedsgrad, hvilket afgør hvor mange liv spilleren har, og hvor hurtigt bolden bevæger sig.

3. Hvis spilleren ikke har flere liv tilbage, afsluttes spillet og der vises game over på skærmen. Efter et par sekunder går spillet automatisk tilbage til menuen.

4. Hvis spilleren vinder spillet vises et victory-screen og efter et par sekunder går spillet automatisk tilbage til menuen.

5. Når banen begynder, eller hvis spilleren mister et liv, placeres bolden over strikeren, og spilleren kan frit bevæge strikeren, hvor bolden følger efter. Hvis spilleren trykker på den givne knap, affyres bolden.

6. Spillerens liv og power skal skrives på LED-skærmen når spillet er igang

7. Spilleren samler power hver gang han ødelægger en kasse. Hvis brugeren trykker på venstre og højre-tasten på en gang bruger han sit power og akitverer high power. Når high power er aktiveret ødelægges kasser når de rammes, uafhængigt af deres liv, og bolden reflekteres ikke, men fortsætter gennem kassen. Power fratrækkes 1 hver gang den ødelægger en kasse.

8. Når spilleren bruger high power, vinder en bane, vinder spillet eller dør skal der rulles en tekst over LED-skærmene, der passer til situationen. Efter teksten er rullet over skal livene og power igen vises på skærmen efter teksten er rullet over.

9. Hver gang spilleren går videre til næste level, får spilleren fuldt liv og spillerens power sættes til 0.

10. Spillere kan pause spillet, ved at trykke på en af knapperne.

11. Ved at trykke på alle knapper samtidigt, aktiverer brugeren chef-mode, som giver en blank skærm.

12. Der er ikke noget point-system i spillet, da vi vælger at lægge fokus andre steder.

13. De forskellige levels skal have forskellige baggrundsbilleder(dette nåede vi ikke).

3.2.2 Krav til strikeren

1. Strikeren skal maskimalt fylde 10% af skærmen på x-aksen.

2. Strikeren skal være delt ind i 3 forskellige områder. Disse 3 områder skal reflektere bolden på forskellig vis afhængig af indgangsvinklen og hvilken del af strikeren den rammer. Reflektionen skal findes igennem trial and error, og vurderes hvad der virker mest naturligt. I oplægget var der lagt op til at strikeren skulle have 5 områder, men vi syntes

ikke det fungerede særligt godt, da det var vanskeligt for brugeren at forholde sig til. Vi har derfor valgt 3 områder.

3. Brugeren skal kunne styre strikeren, vha. knapperne på boardet.

3.2.3 Krav til bolden

1. Bolden skal have et x- og y-koordinat og en retningsvektor, begge i 18.14 fixed-point format. Bolden har desuden nogle variable med info om spillerens power, om bolden er ude og om spilleren har aktiveret power.

2. Boldens retningsvektor skal altid have længden 1, da dette gør kollisionstest let.

3.2.4 Krav til boksene

1. Alle bokse skal have de samme dimensioner, vi valgte 2x6 pixels.

2. Boksene skal kunne have forskellig styrke, således at nogle kasser skal rammes flere gange før de går i stykker. Kassens styrke skal således repræsenteres ved en farve, og farven ændrer sig således også når man rammer en kasse uden at ødelægge den.

3. Hvis man rammer boksen på den horizontale side, skal y-elementet af retningsvektoren inverteres.

4. Hvis man rammer boksen på den vertikale side, skal x-elementet af retningsvektoren inverteres.

5. Hvis man rammer et hjørne, skal både x- og y-elementet inverteres.

6. Når en kasse bliver ødelagt slettes den fra banen

3.3 Timere

På Z8 Encore Evauluation Boardet er der 4 forskellige timere, timer0 til timer3. Disse timere kan konfigureres efter brugerens behov. I vores projekt har vi brugt 2 timere, en til at styre spillets tid, og en anden til at styre LED skærmene. Disse 2 timere er hhv. timer0 og timer1.

3.3.1 Timer0

Timer0 er en timer der sender et tick hvert millisekund. Timeren bliver brugt i main-funktionen og til debouncing af knapperne. Timeren er sat i continous mode, da vi ønsker at den blot skal fortsætte ubetinget, og der foretages ingen clock division af tælleren. Reload værdien fandtes ved udregningen i ligning 5. Interrupt Prioriteten sættes til høj ved at skrive $0x20$ til både IRQ0ENH og IRQ0ENL.

$$Reloadvalue = 0.001\text{s} \cdot 18.432.000\text{s}^{-1} = 18432_{10} = 4800_{16} \tag{5}$$

3.3.2 Timer1

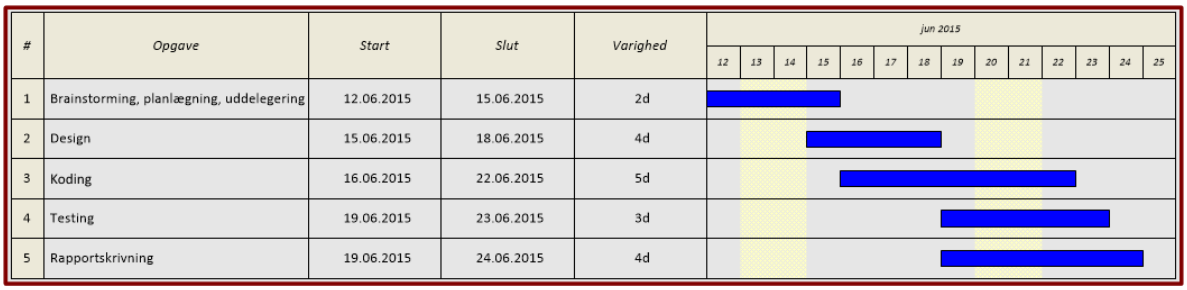
Timer1 er en timer der sender et tick hvert 500 μs. Timeren bliver kun brugt i **led.h**. Denne timer er også sat i continous mode, og der bliver heller ikke her foretaget clock division. Reload værdien fandtes ved udregningen i ligning 6. Interrupt prioriten sættes til lav ved kun at skrive til IRQ0ENL, eftersom at opdateringen af LED skærmene ikke er vigtigere end spillets tid.

$$Reloadvalue = 0.0005\text{s} \cdot 18.432.000\text{s}^{-1} = 9216_{10} = 2400_{16} \tag{6}$$

4 Planlægning og test af programmet

4.1 Plan og Gantt chart

Planen som blev lagt de første dage i projektet blev egentlig fulgt ganske godt igennem projektet. De første dage blev brugt på brainstorming og planlægning af, hvordan vi ønskede at vores spil skulle se ud, og hvilke features som skulle inkluderes. De tekniske specifikationer er et resultat deraf. I overlappet på design og brainstorm delen blev forskellige dele af projektet delt ud, sådan at hver enkel gruppemedlem kom med et udkast til design af forskellige dele af programmet. Igennem designfasen blev der kigget på flowcharts, samt skrevet pseudokode. Hen mod slutningen af designfasen blev der skrevet mere og mere reel kode. Kodeskrivningen og testing delen følger hinanden. Sidst i kodefasen blev vi nød til at droppe et af de mål vi havde sat os for at overholde planen, som vi havde lagt og blive færdig med projektet. De sidste dage op mod deadline blev der skrevet rapport.



#	Opgave	Start	Slut	Varighed
1	Brainstorming, planlægning, uddelegering	12.06.2015	15.06.2015	2d
2	Design	15.06.2015	18.06.2015	4d
3	Koding	16.06.2015	22.06.2015	5d
4	Testing	19.06.2015	23.06.2015	3d
5	Rapportskrivning	19.06.2015	24.06.2015	4d

Figur 2: Gantt chart af vores plan

4.2 Problemer

Under udarbejdelsen af programmet havde vi problemer, som vi ikke havde forudset under design-fasen, vi vil her gennemgå nogle af dem der voldte os mest besvær, at debugge.

4.2.1 Problemer med realloc

I designfasen havde vi forestillet os at vores boxstruct blot skulle være skrevet som i **newBoxStack()**, hvor vi dog kun allokerede plads til et enkelt element i hvert array. Ydermere skulle der være en variabel kaldet capacity, der betegnede hvor stort stacket var. Vi ville derefter i **createBoxes()** undersøge om **(box).capacity == *(box).size* , og hvis det var sandt allokere yderlige 10 pladser med realloc. Dette fungerede dog ikke, og boksene fik tilfældige lokationer på banen. Vi mistænker at der ikke var plads til at dynamisk allokere plads på boardet og finde sammenhængende plads i rammene, og det derfor gik galt. Hvis vi i stedet startede med at allokere plads, gav det os ikke problemer.

4.2.2 Problemer med knapperne

Vi havde problemer med knapperne på boardet: Vi var i tvivl om vores kode var dårlig, eller om det var fordi knapperne var slidte og ødelagte. Vi lavede en debouncer, og det hjalp lidt på nogle af knapperne, men vi havde stadig problemer, og nåede aldrig at komme til bunds i problemet. Vi er dog ret overbevist om at problemerne stammer fra de slidte knapper.

4.3 Test af programmet

Programmet blev testet op mod vores designkrav og de tekniske specifikationer. Først blev grundelementerne kodet og testet, og når det levede op til specifikationerne blev programmet udvidet med nye funktioner. Koden blev altså skrevet og testet med en buttom up metode, hvor det første element var selve banen. Derefter fulgte strikeren, samt det at kunne få strikeren til at flytte sig. Næste skridt var boldens bevægelse, samt den simple refleksion på kanterne og strikeren, hvor indgangsvinkel var lig udgangsvinkel. Så fulgte kasserne med alle deres egenskaber, og boldens refleksion på kasserne. Endeligt blev strikerens udgangsvinkel ændret alt efter indgangsvinkel. En menu blev lavet sideløbende med spillet. Al vores testing blev udført i terminalen, hvor programmet var sat op til at teste, i den forstand, at programmet printede informationer som vi havde brug for. Dette gjorde vi f.eks. i forbindelse med testing af strikeren, hvor vi printede hvor bolden blev detekteret på strikeren, hvad ind- og udgangsvinklen var. Dette gjorde vi for at checke om vores program opførte sig som forventet.

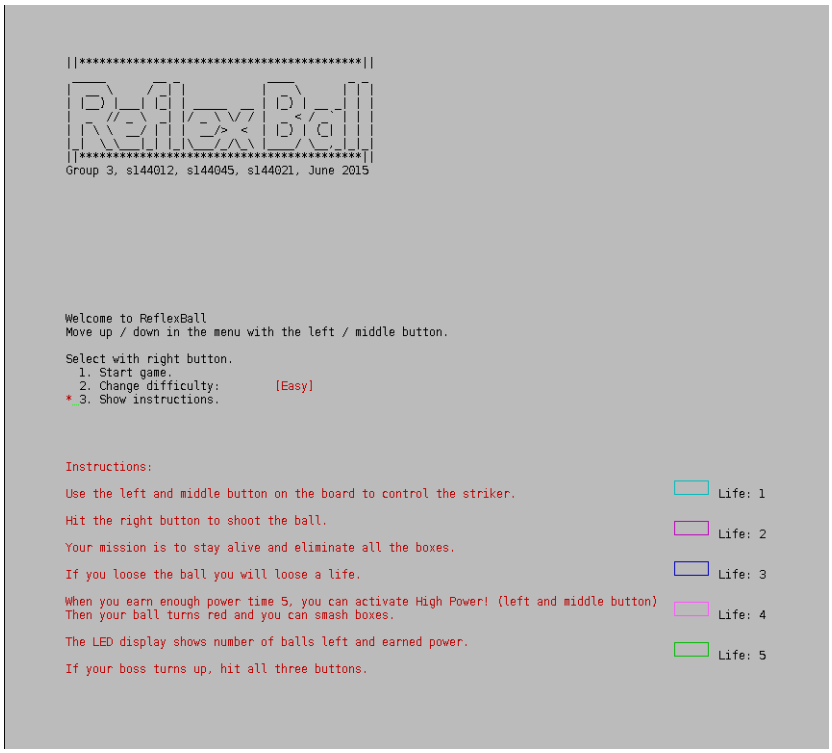
5 Implementation

Vi vil i denne sektion gennemgå implementationen af spillet. Vi har valgt blot at gennemgå main-metoden, og flowet af denne. Info om de andre moduler kan findes i dokumentationen.

5.1 main.c

Det her spil styres af en main.c fil med en main funktion. For at forbedre strukturen og øge læsbarheden er selve gameplayet håndteret af en funktion der hedder **Game**. Denne funktion bliver kaldet af main når brugeren vælger start game og returnerer antal liv der er tilbage når spillet afsluttes. På denne måde registeres sejr / tab.
Main funktionen starter med at tegne menuen, sende teksten "Welcome" til LED-displayet og går derefter ind i en uendelig løkke, hvor den venter på at brugeren trykker på en knap. Når brugeren trykker på den venstre eller midterste knap bladrer man igennem menuen ved at øge eller formindske *selectedOption*, der holder styr på hvor man befinder sig i menuen. Når brugeren trykker på højre knap undersøger programmet *selectedOptions* værdi og foretager en handling baseret på dens værdi. Det kan være at starte et nyt spil, ændre sværhedssgrad eller vise hjælp.

Når et nyt spil begynder, starter funktionen Game med at positionere stri-

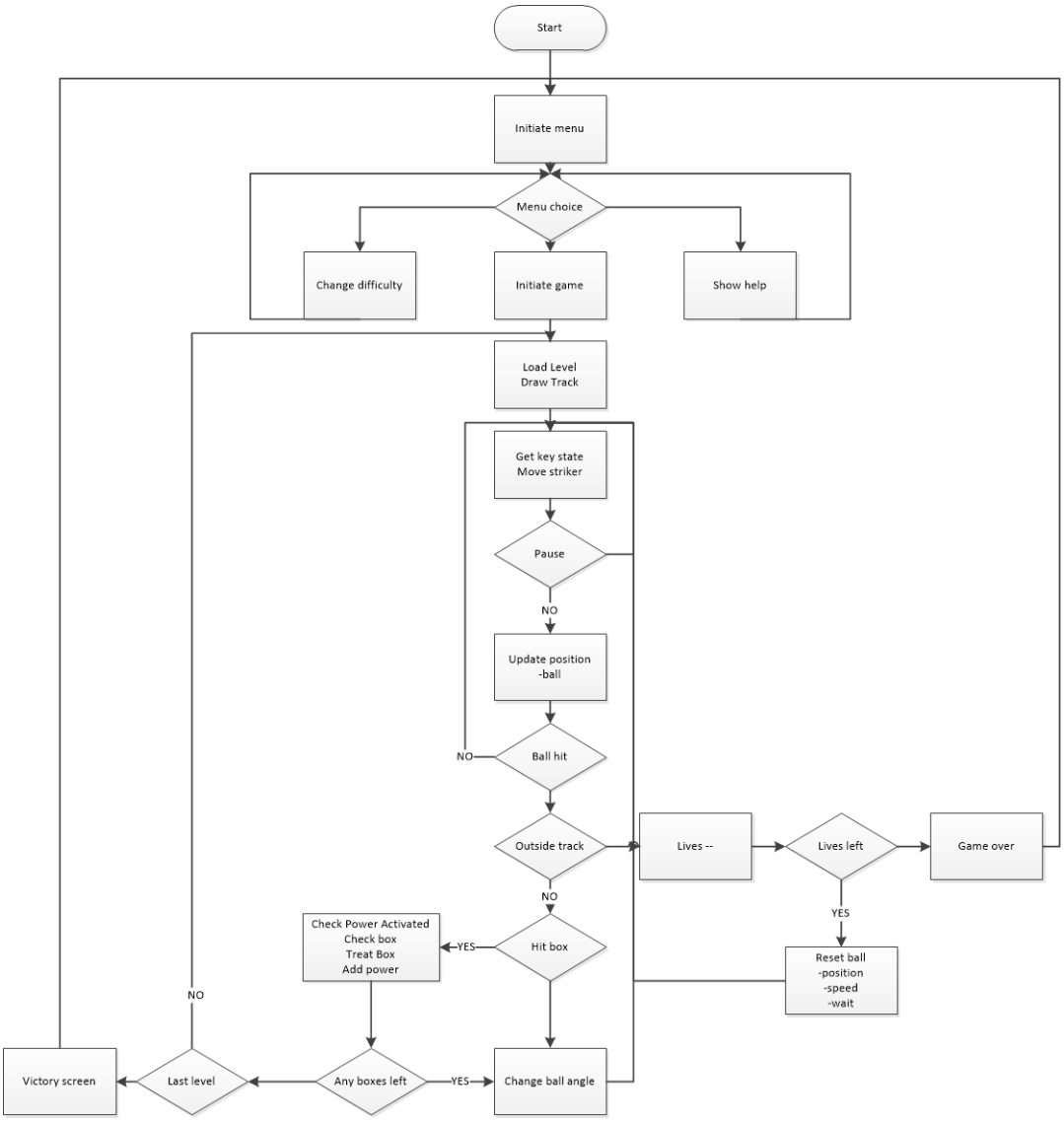


Figur 3: Her ses menuen med show instructions aktiveret

keren, vælge hvor mange bolde brugeren skal have hver level, hvor hurtigt bolden skal køre og tegner tilsidst banen.
Derefter går man ind i en løkke der kører en gang per level helt til max level er nået eller til brugeren ikke har flere liv. Hver gang en ny level starter får brugeren fuldt liv og bolden bliver placeret over strikeren. LED-displayet viser også hvilken level man er nået til. For at skrive tal fra variable på LED-displayet type-castes de til den tilhørende char-værdi og lægges ind i et char array der bliver konkateneret med den resterende streng. Denne streng sendes til LED-displayet med funktionen **LEDSetString**. Funktionen **setLedMode** benyttes for at rigtig visning bliver brugt.
Bolden tegnes med funktionen **drawChar**, hvor det 3. argument er typen character der skal tegnes. Oftest er dette et "o", men hvis bolden rammer strikeren eller kanterne printes der en char der gør at disse bliver grafisk bevaret. Hvilken character der skal styres af funktionen **checkBall**. Envidere dannes og tegnes bokserne med farver der svarer til styrken.
Når initialiseringen for en level er færdig går man ind i en ny løkke, som kører

så længe man har liv og bokse tilbage. Til at begynde med registrerer programmet hvilke knapper der et trykket ind. Når brugeren trykker højre knap skydes bolden. Hvis brugeren trykker på den højre tast igen pauses spillet. når brugeren trykker alle tre knapper bliver skærmen renset (chef-mode).
Hvis spillet ikke er pauset har brugeren mulighed for at flytte strikeren med venstre og midterste knap, og aktivere High Power med begge knapper. Når High Power aktiveres ruller teksten "Power!"over LED-displayet og BEL-characteren printes(siger en lyd, hvis PuTTY er indstillet korrekt)

Det benyttes en tæller til at regulere den frekvens boldens position opdateres ved. Hvis bolden er skudt ud og aktiv, checkes bolden for om den kommer til at ramme en kant, en boks eller strikeren. Hvis den kommer til at gøre det endres boldens retning. Envidere males bolden over.
Der testes for om bolden er ude af banen. Hvis det er tilfældet bliver bolden sat over strikeren igen og brugeren har nu en mindre bold tilbage. LED-displayet opdateres med rigtige antal bolde og mængde *Power*. Til sidst flyttes og tegnes bolden. Bolden tegnes med farven rød hvis High Power er aktiveret.
Når brugeren ikke har flere bolde tilbage eller gennemført spillet kaldes funktionen **drawGameOver** eller **drawVictory**. De funktioner bruger lang tid på at køre, hvilket giver brugeren tid til at se hvad der står.



Figur 4: Flowchart over main

6 Diskussion

Et af de krav der blev stilt var at der skulle bruges så få globale variable som muligt. I vores program bruges der en global variabel til vores timer for at tælle antal millisekunder og nogle til vores LED, der skulle køre selvstendig. Selv om der altså bruges nogle blev det kun blev gjort i de tilfælde hvor det var hensiktsmæssigt og ikke hindrede abstraktionen, strukturen og modulariteten i vores projekt. Under planlægningsfasen blev der planlagt at de forskellige levels skulle have forskellige baggrunde. Dette nåede vi ikke at implementere, men det er dog ikke væsentligt, da programmets kernefunktionalitet er fungerende.
Grundig brugertesting af programmet hvor brugerne ikke fik nogle instrukser blev gennemført. Resultatet af testene, er at spillet er brugervenligt, men at det kan være vanskeligt for brugere at finde ud af at bruge knapperne på boardet. Det sidstnævnte omhandler ikke software og er sådan set ikke vores fokus i dette projekt.
Vi har så længe det gav mening, forsøgt at undgå globale variable. Vi har brugt mange globale konstanter, da vi mener det øger læsbarheden og gør ændringer i koden hurtigere og simplere. Vi har ydermere forsøgt at adskille de forskellige moduler, så de kun kommunikerer indirekte igennem main metoden.

7 Konklusion

I dette kursus har vi fået et kendskab til mikroprocessorere, programarkitektur og planlægning af design af et medium størrelse program. Vi har i kursusforløbet med success lavet et program, Reflexball, som levede op til de designmæssige og tekniske krav stillet af Lektor José M.G Merayo(dog med en lille ændrigt mht. striker-zoner), samt vores egne krav. Der blev lavet en plan for udarbejdelsen af projektet, som blev omsat til en Gantt chart. Planen blev i høj grad fulgt, og vi nåede næsten alle vores mål. Slutteligt fik vi med success implementeret programmet på et Zilog Z8 Encore Evaluation Board.

8 Kildeliste

Litteratur

[1] Randy Yates, *Fixed-Point Arithmetic: An Introduction*, Digital Signal Labs 23. August 2007

[2] Stephen Brown & Zvonko Vranesic *Fundamentals of Digital Logic with VHDL Design*, McGraw Hill International Edition , Third Edition, 2009

[3] Zilog, *Using the ZiLOG Xtools Z8 Encore! C Compiler*,
http://goo.gl/kUPqL7 ,
link sidst checket 23-06. Kan ellers findes ved google søgning eller på Zilog's hjemmeside

[4] Zilog, *Technical Note Floating Point Routines*,
http://goo.gl/eRBEn0 ,
link sidst checket 23-06. Kan ellers findes ved google søgning eller på Zilog's hjemmeside

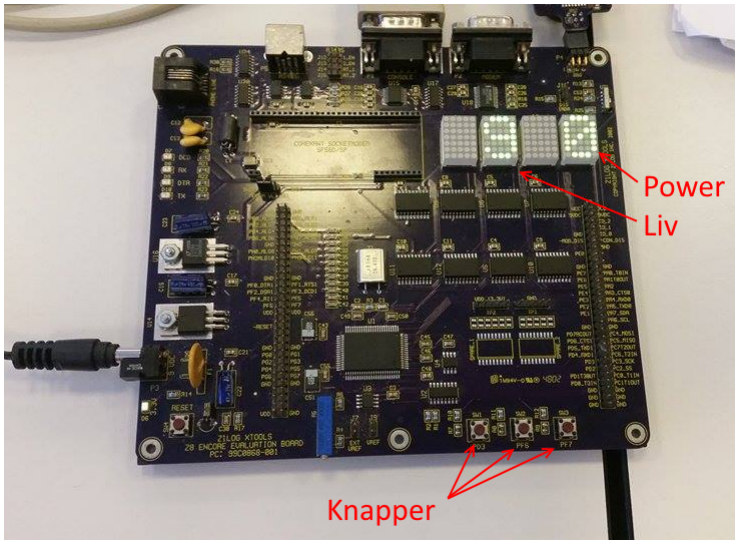
[5] Zilog, *Technical Note Floating Point Multiplication*,
http://goo.gl/vUdPW3,
link sidst checket 24-06.

9 Brugervejledning til ReflexBall

Spillet Reflexball er et arkanoid spil, som handler om at ramme kasser med en bold. Bolden skal holdes i live med en striker, som reflekterer bolden op mod kasserne. Hvis bolden ikke rammer strikeren, og dermed ryger ud af banen, mister man et liv. Kasserne har forskellig styrke alt efter hvilken farve de er. Styrken svarer til det antal gange kassen skal rammes, før den bliver ødelagt. Her er en liste, hvor styrken står til venstre og farven til højre.

1. Lyseblå
2. Lilla
3. Blå
4. Pink
5. Grøn

Når brugeren starter spillet vises en menu. Her kan brugeren styre markøren med knapperne til venstre og i midten. Brugeren vælger den mulighed som markøren står foran ved brug af den højre knap. I menuen kan brugeren se instruktioner, og vælge sværhedsgrad. Når brugeren vælger at starte spillet, vil spillet loade, og bolden sættes over strikeren. Brugeren flytter strikeren ved brug af den venstre og midterste knap. Når brugeren ønsker at sætte bolden i gang trykkes på knappen til højre. Når spillet er i gang kan spillet sættes på pause ved at trykke på den højre knap, og spillet startes da igen ved et tryk på højre knap. Hvis brugeren mister alle sine liv, afsluttes spillet og vender tilbage til menuen. Hvis spilleren derimod får ødelagt alle kasserne vil næste level loade. Ved det sidste level vil en victory screen loade, og vende tilbage til menuen.
Spillets sværhedsgrad har indflydelse på, hvor hurtigt bolden bevæger sig, og hvor mange liv man har. Når man spillet på easy har brugeren 9 liv, og bolden bevæger sig forholdsvis langsomt. På medium er antallet af liv 5, og farten er sat lidt op. På hard har brugeren 3 liv og farten er høj.
Bolden har også egenskaben High Power. Egenskaben aktiveres ved at trykke på den venstre og den midterste knap samtidigt. For hver kasse som brugeren ødelægger, bliver power 1 større. Power skal være 5, før brugeren kan aktivere High Power. Når High Power er aktiveret ødelægger bolden kasserne uanset hvilken holdbarhed de har. Dog mister man 1 power for hver kasse man ødelægger. High Power bliver deaktiveret igen når power er 0. Power bliver nulstillet hvis man mister et liv. Man kan maksimalt have et Power



niveau på 9, derefter tæller den ikke yderligere op.

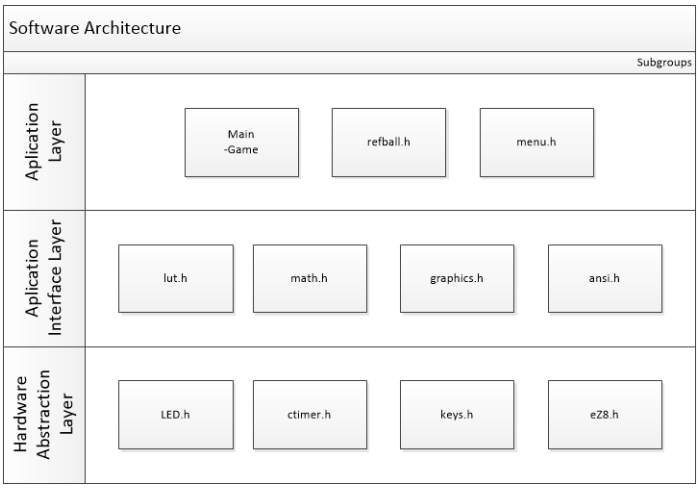
Der findes en hemmelig feature: En boss key er implementeret, som sletter alt hvad der er på skærmen. Denne tilstand aktiveres ved at trykke på alle tre knapper på samme tid. Tilstanden er dog permanent, og spillet skal genstartes, hvis man fortsat vil spille, når chefen er gået.

9.1 Opsætning af PuTTY

Putty-guide: Under *Bell* skal der sættes flueben ved *Play a custom sound file*. Filen *HighPower.wav* skal vælges. Under *Window* skal der være 2000 lines of scrollback, 274 kolonner og 71 rækker, og der skal sættes flueben ved *Change the number of rows and columns* og *Reset scrollback on display activity*. *Cursor appearance* sættes til *Underline*, skriften skal være af typen DejaVu Sans Mono, 7-point, Non-Antialiased. Remote character set er *CP850* og Unicode line drawing code points bruges. Der skal også sættes flueben ved *Allow terminal to specify ANSI colors*. Under *Connection/Serial* skal der vælges en baudrate på 57600, 8 data bits, 1 stop bit og ingen paritet.

10 Dokumentation

I dette afsnitt er projektets filer, funktioner, moduler og makroer dokumenteret og beskrevet. Fokuset er på funktionernes vigtigste egenskaber og virkemåder. Filerne med tilhørende funktioner er delt op i tre forskellige lag: Application, Application Interface og Hardware Abstraction Layer. Se figur 10. I Application Layer ligger de applikationspecifikke filer, disse filer kan bruges på anden hardware uden at modificere dem. I Hardware Abstraction Layer ligger de hardwarespecifikke filer. Disse filer muligør kommunikation mellem hardware og vores software. Disse filer kan benyttes i helt andre programmer såfremt det er samme type hardware der benyttes.



Figur 5: Block diagram over de forskellige lag

Vi har udviklet alle moduler i fællesskab, og det giver ikke mening at tilskrive nogle personer en særlig del af koden, da meget lidt kode er skrevet af udelukkende en person.

10.1 Application layer

I dette lag findes refball modulet, menu modulet og main modulet.

10.1.1 main.c

ReflexBall styres af **main.c** som består af to funktioner. **Main** styrer den overordnede kørsel av programmet og menuen, og **Game** kontrollerer de forskellige levels og selve gameplayet. Se nærmere beskrivelse af **main.c** under implementation.

10.1.2 refball.h

refball.h er et modul der indeholder grundlæggende regler om spillet: kollision, hvordan bolden skal bevæge sig og hvorledes strikeren skal opføre sig. Desuden indeholder den strukturerne *Ball* og *Box*. Ydermere indeholder dette modul også særligt mange konstanter.

Strukturen Ball

Ball er en stuktur som har variablene opgivet i 3.2.3. *outOfBounds*, der fortæller om spilleren er inde for banen, og *powerActivated*, der fortæller om high power er aktiveret, er implementeret som unsigned char's. I *power*, gemmes hvor meget power spilleren har opladet.

Strukturen Box

Box er en struktur der indeholder samtlige bokse i spillet. Den indeholder 3 pointere til char-arrays: et koordinatsæt og et tilhørende array med boksenes styrke. Desuden er der to variable der fortæller antallet af bokse der er fyldt i stacket og hvor mange der ikke er ødelagte. I designfasen forestillede vi os at den skulle implementeres som et stack, således arrayernes størrelse var variable, men hvorfor vi ikke gjorde det står i afsnit 4.2.1.

void moveBall(Ball * ball)

Denne funktion flytter bolden ved at tage en peger til en *Ball* som argument og lægger retningsvektoren til x og y koordinaterne.

void moveStriker(long * x,char direction)

Denne funktion tager to variable som argumenter, en pointer til strikerens x-lokation, og strikerens retning. Hvis *direction* er 1 bevæger strikeren sig mod positiv x-retning og STRIKER_SPEED lægges til, ellers bevæger strikeren sig mod negativ x-retning og STRIKER_SPEED fratrækkes strikerens x-lokation.

unsigned char checkBall(Ball * ball,Box * box, int x)

checkBall() er en funktion som kontrollerer og bestemmer boldens bevægelse. checkBall tager bolden, kasserne og strikerens position som argumenter. I funktionen gennemgåes de forskellige scenarier, hvor bolden kan ramme. Først kontrolleres om strikeren er ramt, derefter om kanterne er ramt og til sidst gennemgås alle kasserne og kontrolleres for om de er ramt. Hvis kassen bliver ramt ændres der i boldens retningsvektor, alt afhængigt af hvordan kassen rammes. Endeligt returnerer funktionen et tegn, svarerende til det, bolden har ramt. Hvis bolden intet har ramt foretages ingen ændring på retningsvektorerne, og et blankt tegn sendes tilbage.

long toTerminalCoordinates(long x)

Denne funktion omdanner tal i 2.14 eller 18.14 til heltal man kan bruge i terminalen. Afrunding laves som vanligvis, ved at afrunde til nærmeste heltal.

void setBallOverStriker(Ball * ball, long st)

Denne funktion sætter bolden over strikeren. Funktionen omdanner st til 18.14 format og sætter boldens x-koordinat til dens værdi. Boldens y-koordinat sættes over strikeren, vha. konstanterne STRIKER_Y og OVER_STRIKER, også i 18.14. Boldens retningsvektor sættes derefter til at gå lodret op, og roteres derefter 40 grader mod venstre.

Box * newBoxStack()

Denne funktion bliver brugt til at lave et nyt *Box*-stack. Der bliver allokeret plads, så der er plads til antallet af bokse givet ved konstanten MAX_BOXES. Antallet af elementer, *size*, i stacket sættes til 0 og pegeren til *Box*-stacket.

void createBoxes(Box * box,char level)

Denne funktion tager en peger til Box-stacket og en character der repræsenterer level som argumenter. Afhængigt af levels værdi, bliver Box-stacket fyldt på en speciel måde, således hvert level er unikt.

10.1.3 menu

Dette modul indeholder funktioner til at tegne og vise grafik når man bevæger sig rundt i menuen.

void initiateMenu()

Denne funktion renser først skærmen og printer derefter menuen. Slutteligt sættes markøren på Start Game.

void moveMarker(int selectedOption)

Denne funktion sætter markøren alt afhængigt af inputtet.

void printDifficulty(short diff)

Denne funktion har til formål at printe sværhedsgraden når brugeren vælger:

1. Hvis *diff* er 1, skrives der "Easy"
2. Hvis *diff* er 2, skrives der "Normal"
3. Hvis *diff* er 3, skrives der "Hard"

void printHelp()

Denne funktion printer hjælpe-teksten. Startstedet for teksternes x-koordinat bestemmes af konstanten LEFT_BORDER

10.2 Application Interface Layer

10.2.1 graphics.h

Dette modul indeholder grafiske elementer til brug i terminalen. Nogle af funktionerne er særligt udviklet til dette spil, men det er muligt de også ville

kunne bruges i andre sammenhæng. Det kan derfor diskuteres om funktionen strengt taget liger i API-laget. Måske den kan siges at ligge i grænsefeltet.

void drawBox(unsigned char x, unsigned char y,unsigned char color)

Denne funktion tegner en boks med bredden givet ved argumentet og højden 2. Koordinaterne til det øverste venstre hjørne gives som argumenter, sammen med kassens farve, hvor farveskemaet i fgcolor bruges.

void drawChar(unsigned char x, unsigned char y,char tegn)

Denne funktion tager et koordinatsæt og et tegn som argumenter. Tegnet bliver skrevet på det givne koordinatsæt.

void moveDrawStriker(unsigned char x, unsigned char direction)

heeej

void drawBounds(int x1,int y1, int x2, int y2,unsigned char color)

Denne funktion tegner banens kanter. Den tager 2 koordinatsæt som input, x1 og y1 svarende til det øverste venstre hjørne og x2 og y2 svarende til det nederste højre hjørne. Variablen color bruges til at bestemme farven på kanterne.

void drawLogo()

Denne funktion tegner spillets logo. Den bruger konstanten LEFT_BORDER til at bestemme på hvilket x-koordinat den skal begynde at skrive fra, således det bliver logoets venstre kant.

drawBall(unsigned char x, unsigned char y, unsigned char color)

Denne funktion tager et koordinatsæt og printer et o i farven bestemt af det 3. parameter.

void drawStriker(unsigned char x, unsigned char color, char strikerWidth, char strikerY)

Denne funktion tegner blot en striker centrummet for strikeren er x, color er farven, strikerWidth er bredden på hver side, og strikerY er Y-koordinatet.

void drawGameOver()

Denne funktion tegner ASCII-kunst af /textitGAME OVER og scroller den ud af skærmen.

void drawVictory()

Denne funktion tegner ASCII-kunst af Arnold Schwarzenegger og scroller den ud af skærmen.

void scrollText(char y, char delay)

Denne funktion printer mange linjer med mellemrum sådan at det der tidligere er skrevet bliver scrollet væk.

void printExampleBoxes(char x, char y, char boxSize)

Denne funktion printer de 5 typer bokser der bruges i spillet sådan at brugeren kan se hvor meget de forskellige boksr tåler.

10.2.2 lut.h

Dette modul indeholder en konstant tabel med sinus værdier for en cirkel delt i 512 stykker. Hvis x er vinklen i radian indsættes da blot $\frac{x \cdot \pi}{256}$ i tabellen.

10.2.3 math.h

Dette modul indeholder nogle generelle matematiske funktioner, heriblandt sin og cosinus, og to makroer til at regne i 2.14 eller 18.14.

Makroer

Modulet indeholder to makroer, en til at multiplicere to tal i .14 format, og en til at dividere to tal i .14 format, hhv. FIX14_MULT(a, b) og FIX14_div(a,b)

long sin(int x)

Denne funktion tager en int som argument. Vinklen skal ikke være i radian, men skal bruge opdelingen af cirklen beskrevet i afsnit 10.2.2. Der returneres sinus til den givne vinkel.

long cos(int x)

Denne funktion tager en int som argument. Vinklen skal ikke være i radian, men skal bruge opdelingen af cirklen beskrevet i afsnit 10.2.2. Der returneres cosinus til den givne vinkel.

10.3 Hardware Abstraction Layer

10.3.1 keys.h

Dette modul får inputs fra knapperne, og kan debounce ved hjælp af timer.h

char readKey()

Denne funktion læser fra knapperne, og returnerer en bit streng, hvor de tre knapper er på hver deres plads i strengen. Hvis pladsen tilhørende knappen er 1, betyder det at knappen bliver trykket. Denne funktion kan godt detektere hvis brugeren trykker flere knapper ind samtidigt. Pladserne er konfigureret således:

1. Knappen til højre er på LSB(least significant bit)
2. Den midterste knap er på 1. plads i bit-strengen.
3. Knappen til venstre er på 2. plads i bit-strengen.

char getKey()

Denne funktion bruges hvis man ønsker debouncing. Den læser vha. readKey() og checker derefter om værdien er det samme efter 10 ms og returner dette.

10.3.2 ctimer.h

Dette modul har med vores primære timer at gøre. Den har 2 globale variable: *time* og *timeWait*. Time tæller hvor lang tid timeren har været tændt. Grunden til at vi har globable variable her, er fordi timeren skal være uafhængig af main og køre så hurtigt som muligt. Main funktionen kan få adgang til variablene ved nogle setter- og getter-funktioner

void setTimer()

Denne funktion sætter vores timer til prescaling 0, continous mode og høj prioritet for interrupt funktionen. Denne timer er sat til at køre hvert ms.

void resetTimer()

Denne funktion sætter de til modulet tilhørende globale variable, *time* og *timeWait*, til 0.

void timer0int

Dette er interruptfunktionen tilhørende timeren. Den lægger 1 til *time* og trækker 1 fra *timeWait*.

void SetDelay(int input)

Denne funktion sætter timeWait til værdien givet i argumentet. Meningen er at bruge *timeWait* som en slags delay, man kan checke værdien på

int getDelay

Denne funktion er blot en getter, der returnerer *timeWait*

unsigned long getCentis()

Denne funktion er blot en getter, der returnerer *time*

10.3.3 LED.h

I dette modul bruges der nogle globale variable. Dette gøres for at f.eks. kunne holde styr på hvilken kolonne og LED-enhet der skal lyse. Det hadde været muligt at undgått brugen af globale variable, men da ville man være nødt til

at sende mange flere variable som parametre fra main. Dette ville ikke øge effektiviteten af programmet, men snarere gjort det hele mere kompliceret eftersom dette modul kører på sin egen frekvens. Og siden variablene kun er relevante for modulet er det ikke nogen idé i at de ligger i main.

void LEDInit()

Denne funktion indstiller TIMER1 til at give et interrupt hvert 0.5 ms. Envidere initialiseres de globale variabler i LED.C.

void timer1int()

Denne funktion kaldes af interrupts fra TIMER1 på boardet. Denne funktion kalder funktionen LEDUpdate().

void setLedMode(char valueIn)

Denne funktion er blot en setter der kontrollerer hvilken funktion der kaldes af LEDUpdate.

void LEDUpdate()

Denne funktion kalder en af to funktioner, LEDUpdateOnce() eller LEDUpdatePrint().

void LEDSetString(char *string)

Denne funktion læser en streng og kopierer den over til den globale variabel /textitstring. Envidere nulstiller den de globale variable.

void LEDLoadBuffer()

Denne funktion indlæser bufferen fra et karakterset på den måde, at når funktionen LEDUpdatePrint() kaldes er teksten umiddelbart på displayet. Med andre ord; denne funktion gør at teksten ikke ruller ind og er derfor nyttig når man skal opdatere displayet hurtigt.

void LEDUpdatePrint()

Denne funktion bruger tids-multiplexing til at belyse alle kolonnene. For at få dette til at fungere bruges to variable der holder styr på hvilken LED-enhed og kolonne der skal lyse.

void LEDUpdateOnce()

Denne funktion ruller en tekststreng over displayet og holder på de sidste fire tegn. Displayet multiplexes på samme måde som i LEDUpdatePrint(), men strengen bliver også rullet. For at kontrollere hastigheden bruges en tæller.

11 Appendix A

Dette appendix indeholder alt kode til programmet. Koden gik ud over siden, så vi har formatteret det, så det kan være på siden, om end det ser lidt mærkeligt ud.

11.1 Kode til refball

main.c

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include "refball.h"
3 #include "graphics.h"
4 #include "ctimer.h"
5 #include "keys.h"
6 #include <sio.h>
7 #include "ansi.h"
8 #include "math.h"
9 #include "menu.h"
10 #include "LED.h"
11 #include <string.h>
12
13 int Game(int difficulty);
14
15 void main(){
16   int selectedOption;
17   char key, lastKey, lastKey2;
18   int difficulty,victory;
19
20
21   difficulty = 1;
22   selectedOption = 1;
23   victory = 0;
24
25   init_uart(_UART0,_DEFFREQ,_DEFBAUD);
26   setTimer();
27   initiateMenu();
28   printDifficulty(difficulty);
```

```
29
30  //iniKeys();
31  LEDInit();
32  LEDSetString("Welcome      ");
33  setLedMode(2);
34  //LEDLoadBuffer();
35
36
37  for(;;){
38    key = getKey();
39
40    //Navigates the menu
41    while(key != KEY_RIGHT){
42      //Move up
43      if(key == KEY_MIDDLE){
44        selectedOption++;
45      //Move down
46      }else if(key == KEY_LEFT){
47        selectedOption--;
48      }
49      if(selectedOption >= 4){
50        selectedOption = 1;
51      }else if(selectedOption <= 0){
52        selectedOption = 3;
53      }
54      moveMarker(selectedOption);
55
56      lastKey = key;
57
58      while(key == lastKey){
59        key = getKey();
60      }
61
62    }
63
64    if(selectedOption == 1){
65      victory = Game(difficulty);
66      if(victory >= 0){
67        LEDSetString("Victory      ");
68        setLedMode(2);
69        drawVictory();
```

```
70        }else{
71          LEDSetString("Game over. Try again.     ");
72          setLedMode(2);
73          drawGameOver();
74        }
75
76        initiateMenu();
77        printDifficulty(difficulty);
78      }else if(selectedOption == 2){
79        difficulty ++;
80        if(difficulty >= 4){
81          difficulty = 1;
82        }
83        printDifficulty(difficulty);
84      }
85      else if(selectedOption == 3){
86        printHelp();
87        printExampleBoxes(100,36,BOXSIZE);
88      }
89
90      lastKey2 = key;
91      while(key == lastKey2){
92        key = getKey();
93      }
94    }
95
96 }
97
98
99 int Game(int difficulty){
100     int i, livesPerLevel;
101         Ball ball;
102     Box * box = newBoxStack();
103         long strikerx;
104     char key, lives, level, pause;
105     char waitStart, gameDelay;
106         unsigned long refreshTime;
107       char str[40];
108     char temp[3];
109
110       clrscr();
```

```
111
112         //Initialize
113         level = 1;
114         lives = 0;
115         strikerx = 30;
116         refreshTime = 100;
117         ball.x = 5 << FIX14_SHIFT;
118         ball.y = 5 << FIX14_SHIFT;
119         ball.outOfBounds = 0;
120         ball.power = 0;
121         pause = 0;
122
123         temp[2] = '\0';
124
125       if(difficulty == 1){
126         livesPerLevel = 9;
127         gameDelay = 50;
128       }else if(difficulty == 2){
129         livesPerLevel = 5;
130         gameDelay = 35;
131       }else{
132         livesPerLevel = 3;
133         gameDelay = 20;
134       }
135
136
137       drawBounds(L_EDGE_COORD,TOP_EDGE_COORD,R_EDGE_COORD
              ,OUT_OF_BOUNDS,0);
138
139
140       drawStriker(strikerx,0,STRIKER_WIDTH, STRIKER_Y);
141
142
143
144
145       //Initialization for each level
146       while(level <= MAX_LEVEL && lives >= 0){
147
148
149         lives = livesPerLevel;
150         waitStart = 1;
```

```
151         ball.powerActivated = 0;
152         ball.power = 0;
153
154         //Sends info to LED display
155         temp[1]=(char)(CHARSET_START + level);
156         temp[0] = ' ';
157         str[0] = '\0';
158         strcat(str, "Level");
159         strcat(str, temp);
160         temp[1] = (char)(CHARSET_START + lives);
161         strcat(str,"     ");
162         strcat(str,temp);
163         temp[1] = (char)(CHARSET_START + ball.power);
164         strcat(str, temp);
165         LEDSetString(str);
166         setLedMode(2);
167
168         drawChar(toTerminalCoordinates(ball.x),
               toTerminalCoordinates(ball.y), checkBall(&ball
               ,box,strikerx));
169         setBallOverStriker(&ball, strikerx);
170         drawBall(toTerminalCoordinates(ball.x),
               toTerminalCoordinates(ball.y),0);
171
172         createBoxes(box,level);
173         for(i = 0; i < box->size; i++){
174           drawBox(box->x[i],box->y[i],7 - box->durability
                 [i],BOXSIZE);
175         }
176
177
178
179
180         while(lives >= 0 && box->boxesLeft > 0){
181
182               key = getKey();
183
184               if(key == 1){
185                 if(!waitStart && !pause){
186                   pause = 1;
187                 }else if(pause){
```

```
188                 pause = 0;
189               }
190               waitStart = 0;
191             }
192
193             if(!pause){
194
195               if(key == 2){
196                 moveStriker(&strikerx, 1);
197                 moveDrawStriker(strikerx,1,
                      STRIKER_WIDTH,STRIKER_Y);
198               }else if(key == 4){
199                   moveStriker(&strikerx,0);
200                   moveDrawStriker(strikerx,0,
                        STRIKER_WIDTH,STRIKER_Y);
201                }else if(key == 6){
202                   if(ball.power >= POWER_LIMIT){
203                     ball.powerActivated = 1;
204                     str[0] = '\0';
205                     strcat(str,"Power!");
206                     temp[1] = (char)(lives +
                          CHARSET_START);
207                     strcat(str,temp);
208                     temp[1] = (char)(ball.power +
                          CHARSET_START);
209                     strcat(str,temp);
210                     strcat(str," ");
211                     LEDSetString(str);
212                     setLedMode(2);
213                     printf("%c",7);
214                   }
215                }else if(key == 7){
216                  pause = 1;
217                  clrscr();
218                }
219
220
221               if(getCentis() - gameDelay > refreshTime)
                    {
222
223                 if(!waitStart){
```

```
224                 refreshTime = getCentis();
225                 drawChar(toTerminalCoordinates(ball.x
                        ),toTerminalCoordinates(ball.y),
                        checkBall(&ball,box,strikerx));
226
227                 if(ball.outOfBounds){
228                   ball.outOfBounds = 0;
229                   lives--;
230                   waitStart = 1;
231                   ball.power = 0;
232                   ball.powerActivated = 0;
233                 }
234
235                 if(!ball.powerActivated || ball.power
                        < POWER_LIMIT){
236                   str[0] = '\0';
237                   temp[1] = (char)(CHARSET_START +
                          lives);
238                   strcat(str,temp);
239                   temp[1] = (char)(CHARSET_START +
                          ball.power);
240                   strcat(str,temp);
241                   strcat(str," ");
242                   setLedMode(3);
243                   LEDSetString(str);
244                   LEDLoadBuffer();
245                 }
246                 moveBall(&ball);
247                 drawBall(toTerminalCoordinates(ball.x
                        ),toTerminalCoordinates(ball.y),
                        ball.powerActivated);
248
249               }else{
250               drawBall(toTerminalCoordinates(ball.x),
                    toTerminalCoordinates(ball.y),7);
251               setBallOverStriker(&ball, strikerx);
252               drawBall(toTerminalCoordinates(ball.x),
                    toTerminalCoordinates(ball.y),0);
253               }
254
255
```

```
256                 }
257               }//!pause
258         }//while
259         level ++;
260       }//while - level
261
262   return lives;
263 }
```

../main.c

refball.c

```
1 #include <eZ8.h>
2 #include <sio.h>
3 #include <stdlib.h>
4 #include "refball.h"
5 #include "math.h"
6 #include "graphics.h"
7 #include "ansi.h"
8
9 void moveBall(Ball * ball){
10   ball->x += (ball->xdir);
11   ball->y += (ball->ydir);
12 }
13
14 void moveStriker(long * x,char direction){
15     if(direction && ((*x + STRIKER_WIDTH + 2) <
         R_EDGE_COORD))
16         * x += STRIKER_SPEED;
17
18     else if(!direction && ((*x - STRIKER_WIDTH - 2) > (
         L_EDGE_COORD)))
19         * x -= STRIKER_SPEED;
20 }
21
22 unsigned char checkBall(Ball * ball,Box * box,  int x){
23   char right;
24   int angle;
25   char nextPosX, nextPosY;
26   unsigned char j;
27   unsigned char xt = toTerminalCoordinates(ball->x); //
       Defineres så vi undgår at kalde funktionen flere
       gange
28   unsigned char yt = toTerminalCoordinates(ball->y); //
       Defineres så vi undgår at kalde funktionen den
       flere gange
29   nextPosX = toTerminalCoordinates(ball->x + ball->xdir
       );
30   nextPosY = toTerminalCoordinates(ball->y + ball->ydir
       );
```

```
31      if((nextPosY == STRIKER_Y) && (nextPosX >= (x -
            STRIKER_WIDTH)) && nextPosX <= (x +
            STRIKER_WIDTH)){
32
33
34        if(ball->xdir > 0){
35          right = 1;
36        }else{
37          right = 0;
38        }
39
40
41
42        //Left part of striker
43      if(nextPosX >= x - STRIKER_WIDTH && nextPosX < (x -
            1)){
44            if(right){
45        if(ball->ydir > (7 << 11)){//0.875
46          angle = -20;
47        }else if ( ball->ydir > (1 << 13)){//0.5
48          angle = -30;
49        }else{
50          angle = -80;
51        }
52      }else{
53        if(ball->ydir > (7 << 11)){
54          angle = -20;
55        }else if ( ball->ydir > (1 << 13)){
56          angle = -30;
57        }else{
58          angle = 80;
59        }
60      }
61      rotate(ball,angle);
62
63        //Middle part of striker
64        }else if (nextPosX <= x -1 || nextPosX >= x +
              1){
65
66        //Right part of striker
67        }else{
```

```
68            if(right){
69          if(ball->ydir > (7 << 11)){
70            angle = -20;
71          }else if ( ball->ydir > (1 << 13)){
72            angle = -30;
73          }else{
74            angle = 80;
75          }
76        }else{
77          if(ball->ydir > (7 << 11)){
78            angle = -20;
79          }else if ( ball->ydir > (1 << 13)){
80            angle = -30;
81          }else{
82            angle = -80;
83          }
84        }
85        rotate(ball,angle);
86      }
87
88      ball->ydir *= -1;
89
90
91
92      }
93      else if(nextPosX >= R_EDGE_COORD || nextPosX <=
           L_EDGE_COORD){
94        ball->xdir *= -1;
95      }
96      else if(nextPosY <= TOP_EDGE_COORD){
97        ball->ydir *= -1;
98        }
99      else if(nextPosY >= OUT_OF_BOUNDS){
100     ball->outOfBounds = 1;
101     }
102
103     else{
104
105         for(j=0; j < box->size; j++){
106
```

```
107             if((box->durability[j] > 0) && (nextPosX >=
                    box->x[j] && nextPosX < box->x[j]+BOXSIZE)
                     && (box->y[j] == nextPosY || box->y[j]+1
                    == nextPosY))// Boksene har en bredde på
                     3, vi tester alle koordinater
108                  {
109
110             if(!(ball->powerActivated)){
111                     if((xt >= box->x[j]) && (xt < box->x[
                            j]+BOXSIZE))
112                     ball->ydir *= -1;
113
114                     else if(yt == box->y[j] || yt == box
                            ->y[j]+1)
115                     ball->xdir *= -1;
116
117                     else{
118                       ball->xdir *= -1;
119                       ball->ydir *= -1;
120                     }
121         }
122
123         //Kills the box instantly when high power
124         if(ball->powerActivated && ball->power){
125           box->boxesLeft--;
126           box->durability[j] = 0;
127           ball->power --;
128           if(ball->power <= 0){ ball->powerActivated =
                0;}
129
130         }else{
131           if(--box->durability[j] == 0){
132             box->boxesLeft--;
133             ball->power ++;
134             if(ball->power > 9) ball->power = 9;
135           }
136
137         }
138         drawBox(box->x[j],box->y[j],7-box->durability[j
              ],BOXSIZE);
139
```

```
140
141                 }
142             }
143
144         }
145       if (xt == L_EDGE_COORD || xt == R_EDGE_COORD){
146         return EDGE;
147       }
148       else if(yt == TOP_EDGE_COORD){
149         return TOP_EDGE;
150       }
151       else if(xt == x && yt ==STRIKER_Y){
152         return STRIKER;
153       }
154       else
155         return BLANK;
156
157 }
158   long toTerminalCoordinates(long x){
159     long o = x >> FIX14_SHIFT;
160   o += (x >> (FIX14_SHIFT-1)) & 0x1;
161   return o;
162
163 }
164
165 void setBallOverStriker( Ball * ball, long st){
166   ball->x = (st << FIX14_SHIFT);
167   ball->y = ((STRIKER_Y-OVER_STRIKER) << FIX14_SHIFT);
168
169   ball->xdir = 0;
170   ball->ydir = (-1) << FIX14_SHIFT;
171   rotate(ball, -(int) 40);
172 /*
173   ball->xdir = (11 << (FIX14_SHIFT - 4));
174   ball->ydir = (-11 << (FIX14_SHIFT - 4));
175 */
176
177 }
178
179 Box * newBoxStack() {
180     Box * stackContents;
```

```
181     stackContents = malloc(sizeof(Box));
182     stackContents-> size = 0;
183     stackContents->x= malloc(sizeof(char)*MAX_BOXES);
184     stackContents->y = malloc(sizeof(char)*MAX_BOXES);
185     stackContents->durability = malloc(sizeof(char)*
          MAX_BOXES);
186     return stackContents;
187 }
188
189 void createBoxes( Box * box,char level){ //Creates and
      draws boxes
190   unsigned char j,i;
191   unsigned char * xtemp, * ytemp, * dtemp;
192         box->size = 0;
193         if(level == 1){
194             for(j=0;j<1;j++){
195                 for(i = L_EDGE_COORD + 5; i < (
                      R_EDGE_COORD-5); i+=BOXSIZE){
196
197                     box->x[box->size] = i;
198                     box->y[box->size] =
                        TOP_EDGE_COORD+20+j*2;
199                     box->durability[box->size] =
                        1;
200                     box->size++;
201
202                 }
203             }
204         }else if(level == 2){
205             for(j=0;j<2;j++){
206                 for(i = L_EDGE_COORD + 5; i < (
                      R_EDGE_COORD-5); i+=BOXSIZE){
207
208                     box->x[box->size] = i;
209                     box->y[box->size] =
                        TOP_EDGE_COORD+20+j*2;
210                     box->durability[box->size]
                        = j+1;
211                     box->size++;
212
213                 }
```

```
214                 }
215
216             }else if(level == 3){
217               /*for(i = 0; i < 16; i++){
218                 box->x[box->size] = 20 + 4*i;
219                 if(i < 8) box->y[box->size] = 5 + i;
220                 else box->y[box->size] = 19 - (i-8);
221
222                 box->durability[box->size] = 3;
223                 box->size++;
224               }
225
226               */
227
228                   for(j=0;j<3;j++){
229                         for(i = L_EDGE_COORD + 5; i < (
                              R_EDGE_COORD-5); i+=BOXSIZE){
230
231                                 box->x[box->size] = i;
232                                 box->y[box->size] =
                                      TOP_EDGE_COORD+20+j*2;
233                                 box->durability[box->size]
                                      = j+1;
234                                 box->size++;
235
236                             }
237                   }
238
239
240             }else if(level == 4){
241                   for(j=2;j<4;j++){
242                         for(i = L_EDGE_COORD + 5; i < (
                              R_EDGE_COORD-5); i+=BOXSIZE){
243
244                                 box->x[box->size] = i;
245                                 box->y[box->size] =
                                      TOP_EDGE_COORD+20+j*2;
246                                 box->durability[box->size]
                                      = j+1;
247                                 box->size++;
248
```

```
249                             }
250                     }
251
252             }else if(level == 5){
253                     for(j=3;j<5;j++){
254                             for(i = L_EDGE_COORD + 5; i < (
                                    R_EDGE_COORD-5); i+=BOXSIZE){
255
256                                     box->x[box->size] = i;
257                                     box->y[box->size] =
                                            TOP_EDGE_COORD+25+j*2;
258                                     box->durability[box->size]
                                            = j+1;
259                                     box->size++;
260
261                             }
262                     }
263
264             }
265             box->boxesLeft = box->size;
266 }
```

../refball.c

refball.h

```
 1 #ifndef _REFBALL_H_
 2 #define _REFBALL_H_
 3 #define STRIKER_SPEED 2
 4 #define L_EDGE_COORD 2
 5 #define R_EDGE_COORD 120
 6 #define STRIKER_WIDTH 4
 7 #define OUT_OF_BOUNDS 65
 8 #define GAMESPEED 5
 9 #define OVER_STRIKER 2
10 #define STRIKER_Y  60
11 #define STRIKER_START  20
12 #define TOP_EDGE_COORD 2
13 #define BLANK 32
14 #define EDGE 180
15 #define TOP_EDGE 196
16 #define STRIKER 220
17 #define BOXSIZE 6
18 #define MAX_BOXES 50
19 #define MAX_LEVEL 5
20 #define POWER_LIMIT 5
21
22
23
24   typedef struct{
25   long x,y,xdir, ydir;
26   unsigned char outOfBounds, powerActivated;
27   int power;
28   } Ball;
29
30   typedef struct{
31     unsigned char *x, *y;
32   unsigned char * durability;
33     unsigned char size,boxesLeft;
34   } Box;
35
36   void moveBall(Ball * ball);
37   void moveStriker(long * x, char direction);
```

```
38   unsigned char checkBall(Ball * ball,Box * box, int x)
       ;
39   long toTerminalCoordinates(long x);
40   void setBallOverStriker( Ball * ball,  long st);
41
42   Box * newBoxStack(void);
43   void createBoxes( Box * box,char level);
44 #endif
```

../refball.h

menu.c

```
 1 #include <eZ8.h>
 2 #include <sio.h>
 3 #include "ansi.h"
 4 #include "menu.h"
 5 #include "graphics.h"
 6
 7
 8
 9 void initiateMenu(){
10   clrscr();
11   fgcolor(0);
12   bgcolor(7);
13   drawLogo();
14   gotoxy(LEFT_BORDER,24);
15   printf("Welcome to ReflexBall");
16   gotoxy(LEFT_BORDER,25);
17   printf("Move up / down in the menu with the left /
       middle button.");
18   gotoxy(LEFT_BORDER,27);
19   printf("Select with right button.");
20   gotoxy(LEFT_BORDER,28);
21   printf("  1. Start game.\n");
22   gotoxy(LEFT_BORDER,29);
23   printf("  2. Change difficulty:\n");
24   gotoxy(LEFT_BORDER,30);
25   printf("  3. Show instructions.\n");
26   gotoxy(LEFT_BORDER,31);
27
28   //Prints the menu-select marker
29   moveMarker(1);
30
31 }
32
33 void moveMarker (int selectedOption){
34   int i;
35   fgcolor(1);
36   bgcolor(7);
37   //Clears the first column
```

```
38    for(i = 0;i < 3; i++){
39      gotoxy(LEFT_BORDER,LINE_NUMBER + i);
40      printf(" ");
41    }
42    gotoxy(LEFT_BORDER, LINE_NUMBER + selectedOption - 1)
        ;
43    printf("*");
44
45 }
46
47
48 void printDifficulty(short diff){
49    fgcolor(1);
50    bgcolor(7);
51
52    gotoxy(40,LINE_NUMBER + 1);
53
54    if(diff == 1){
55      printf(" [Easy]   ");
56    }else if(diff == 2){
57      printf(" [Medium] ");
58    }else if(diff == 3){
59      printf(" [Hard]   ");
60    }else{
61      printf(" [Error]  ");
62    }
63 }
64
65 void printHelp(){
66    gotoxy(LEFT_BORDER,35);
67    printf("Instructions:");
68    gotoxy(LEFT_BORDER,37);
69    printf("Use the left and middle button on the board
        to control the striker.\n");
70    gotoxy(LEFT_BORDER,39);
71    printf("Hit the right button to shoot the ball.");
72    gotoxy(LEFT_BORDER,41);
73    printf("Your mission is to stay alive and eliminate
        all the boxes.\n");
74    gotoxy(LEFT_BORDER,43);
```

```
75    printf("If you loose the ball you will loose a life.\
        n");
76    gotoxy(LEFT_BORDER,45);
77    printf("When you earn enough power time (5), you can
        activate High Power! (left and middle button)\n");
78    gotoxy(LEFT_BORDER,46);
79    printf("Then your ball turns red and you can smash
        boxes.");
80    gotoxy(LEFT_BORDER,48);
81    printf("The LED display shows number of balls left
        and earned power.");
82    gotoxy(LEFT_BORDER,50);
83    printf("If your boss turns up, hit all three buttons.
        ");
84
85 }
```

../menu.c

menu.h

```
1 #ifndef _MENU_H_
2 #define _MENU_H_
3
4
5 void initiateMenu();
6 void moveMarker(int selectedOption);
7 char getChoice();
8 void printDifficulty(short diff);
9 void printHelp();
10 #endif
```

../menu.h

lut.h

```
1 //
    ================================================================================


2 //
3 //  Exported by Cearn's excellut v1.0
4 //  (comments, kudos, flames to daytshen@hotmail.com)
5 //
6 //
    ================================================================================


7
8 #ifndef LUT_H
9 #define LUT_H
10
11 // === LUT SIZES ===
12 #define SIN_SIZE 512
13
14 // === LUT DECLARATIONS ===
15 extern const signed short SIN[512];
16 #endif  // LUT_H
```

../lut.h

math.c

```
1 #include "lut.h"
2 #include "math.h"
3 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
4 #include <sio.h>
5 #include "refball.h"
6
7 long sin(int x){
8   return SIN[0x01FF & x];
9 }
10
11 long cos(int x){
12   return sin(x+128);
13 }
14
15 void rotate(Ball * ball , int ang){
16   long sinA = sin(ang);
17   long cosA = cos(ang);
18   long tempX = ball->xdir;
19   ball->xdir = (FIX14_MULT(tempX,cosA) - FIX14_MULT(
        ball->ydir,sinA));
20   ball->ydir = (FIX14_MULT(tempX,sinA) + FIX14_MULT(
        ball->ydir,cosA));
21 }
```

../math.c

math.h

```
1 #include "refball.h"
2 #ifndef _MATH_H_
3 #define _MATH_H_
4 #define FIX14_SHIFT 14
5 #define FIX14_MULT(a, b) ( (a) * (b) >> FIX14_SHIFT)
6 #define FIX14_div(a,b)   ( ((a) << FIX14_SHIFT / (b) ))
7
8
9
10 long sin(int x);
11 long cos(int x);
12 int arcsin(int x);
13 void rotate(Ball * ball, int ang);
14 long expand(long i);
15 void printFix(long i);
16 //long abs(long a);
17
18 #endif
```

../math.h

graphics.c

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>                    // special encore serial i
      /o routines
3 #include "ansi.h"
4 #include "graphics.h"
5
6
7 void drawBox(unsigned char x, unsigned char y,unsigned
      char color, char boxSize){
8   char j;
9
10  //Avoids the color yellow
11  if(color == 3) color = 13;
12
13  fgcolor(color);
14  drawBounds(x,y,x+(boxSize-1),y+1,color);
15  //draws last line
16  fgcolor(color);
17  gotoxy(x,y+1);
18  printf("%c",192);
19  for(j=0; j < (boxSize-2); j++){
20  printf("%c",196);
21  }
22  printf("%c",217);
23  fgcolor(0);
24 }
25 void drawChar(unsigned char x, unsigned char y,char
      tegn){
26  gotoxy(x,y);
27  printf("%c",tegn);
28 }
29 void drawBall(unsigned char x, unsigned char y,
      unsigned char color){
30  fgcolor(color);
31  gotoxy(x,y);
32  printf("%c", 111);
33  fgcolor(0);
```

```
34 }
35 void moveDrawStriker(unsigned char x, unsigned char
     direction, char strikerWidth, char strikerY){
36   fgcolor(0);
37   if(direction==1){
38     gotoxy(x - (strikerWidth +2),strikerY);
39     printf("  ");
40     gotoxy(x + strikerWidth - 2,strikerY);
41     printf("%c%c%c",220,220,220);
42   }else{
43     gotoxy(x + strikerWidth + 1,strikerY);
44     printf("  ");
45     gotoxy(x - strikerWidth,strikerY);
46     printf("%c%c%c",220,220,220);
47   }
48 }
49 void drawStriker(unsigned char x, unsigned char color,
     char strikerWidth, char strikerY){
50   unsigned char i;
51   fgcolor(color);
52   gotoxy(x-strikerWidth,strikerY);
53   for(i=0;i < 2*strikerWidth + 1;i++)
54     printf("%c",220);
55   fgcolor(0);
56 }
57 void drawBounds(int x1,int y1, int x2, int y2, unsigned
      char color){
58   int i,j;
59   char hs,vs,h1,h2,h3,h4;
60   int height = y2 - y1+1;
61   int width = x2 - x1+1;
62   fgcolor(color);
63   hs=196;
64   vs=179;
65   h1=218;
66   h2=191;
67   h3=217;
68   h4=192;
69   gotoxy(x1,y1);
70   printf("%c",h1);
71
```

```
72    for(i=1;i<=width-2;i++){
73       printf("%c",hs);
74    }
75
76    printf("%c",h2);
77
78    for(i=1;i<=height-2;i++){
79       gotoxy(x1,y1+i);
80       printf("%c",vs);
81       gotoxy(x2,y1+i);
82       printf("%c",vs);
83    }
84 fgcolor(0);
85 }
86
87 void drawLogo(){
88    gotoxy(LEFT_BORDER,5);
89    printf("
         ||****************************************||\n")
         ;
90    gotoxy(LEFT_BORDER,6);
91    printf(" _____          __ _                ____          _ _
         \n");
92    gotoxy(LEFT_BORDER,7);
93    printf("|  __ \      / _| |              |  _ \        | |
         |\n");
94    gotoxy(LEFT_BORDER,8);
95    printf("| |__) |___| |_| |  _____   __ | |_) | __ _| |
         |\n");
96    gotoxy(LEFT_BORDER,9);
97    printf("|  _  // _ \  _| |/ _ \ \/ / |  _ < / _` | |
         |\n");
98    gotoxy(LEFT_BORDER,10);
99    printf("| | \ \  __/ | | |  __/>  <  | |_) | (_| | |
         |\n");
100   gotoxy(LEFT_BORDER,11);
101   printf("|_|  \_\___|_| |_|\___/_/\_\ |____/ \__,_|_|_
         |\n");
102   gotoxy(LEFT_BORDER,12);
103   printf("
         ||****************************************||\n")
```

```
        ;
104     gotoxy(LEFT_BORDER,13);
105     printf("Group 3, s144012, s144045, s144021, June 2015
            ");
106     //printf("Ver. 0.2.0.3 Beta\n
                                        ");
107
108 }
109
110 void drawGameOver(){
111     int i;
112     clrscr();
113     for(i = 0; i < 4; i++){
114       gotoxy(LEFT_BORDER,5);
115       printf(" .----------------.  .----------------.
            .----------------.  .----------------.  \n");
116       gotoxy(LEFT_BORDER,6);
117       printf("| .--------------. || .--------------. ||
            .--------------. || .--------------. |\n");
118       gotoxy(LEFT_BORDER,7);
119       printf("| |     ______     | || |      __      | || |
             ____    ____ | || |  _________   | | \n");
120       gotoxy(LEFT_BORDER,8);
121       printf("| |   .' ___  |    | || |     /  \     | ||
            ||_   \  /   _|| || | |_   ___  |  | | \n");
122       gotoxy(LEFT_BORDER,9);
123       printf("| |  / .'   \_|    | || |    / /\ \    | || |
              |   \/   |  | || |   | |_  \_|  | | \n");
124       gotoxy(LEFT_BORDER,10);
125       printf("| |  | |     ____   | || |   / ____ \   | || |
              | |\  /| |  | || |   |  _|  _   | | \n");
126       gotoxy(LEFT_BORDER,11);
127       printf("| |  \ `.___]  _| | || | _/ /    \ \_ | || |
             _| |_\/_| |_ | || |  _| |___/ |  | |  \n");
128       gotoxy(LEFT_BORDER,12);
129       printf("| |   `._____.'    | || ||_____|  |_____|| ||
            || _____|| _____|| || | |_________|  | |\n");
130       gotoxy(LEFT_BORDER,13);
131       printf("| |                | || |              | || |
                          | || |              | | \n");
132       gotoxy(LEFT_BORDER,14);
```

```
133     printf("| '----------------' || '----------------' ||
          '----------------' || '----------------' |  \n");
134     gotoxy(LEFT_BORDER,15);
135     printf(" '------------------' '------------------'
          '------------------' '------------------'  \n");
136     gotoxy(LEFT_BORDER,16);
137     printf("

          \n");
138     gotoxy(LEFT_BORDER,17);
139     printf(" .------------------. .------------------.
          .------------------. .------------------.  \n");
140     gotoxy(LEFT_BORDER,18);
141     printf("| .----------------. || .----------------. ||
          .----------------. || .----------------. |\n");
142     gotoxy(LEFT_BORDER,19);
143     printf("| |     ____     | || |  ____  ____  | || |
            _________   | || |  _______     | |\n");
144     gotoxy(LEFT_BORDER,20);
145     printf("| |   .'    `.   | || ||_   _| |_  _| | || |
          |_   ___  |  | || | |_   __ \    | |\n");
146     gotoxy(LEFT_BORDER,21);
147     printf("| |  /  .--.  \  | || |  \ \  / /   | || |
            | |_  \_|  | || |    | |__) |   | |\n");
148     gotoxy(LEFT_BORDER,22);
149     printf("| |  | |    | |  | || |   \ \/ /    | || |
            |  _|  _   | || |    |  __ /    | |\n");
150     gotoxy(LEFT_BORDER,23);
151     printf("| |  \ `--' /  | || |    \ ' /     | || |
          _| |___/ |  | || |   _| |  \ \_  | |\n");
152     gotoxy(LEFT_BORDER,24);
153     printf("| |   `.____.'   | || |     \_/     | || |
          |_________|  | || | |____| |___| | | \n");
154     gotoxy(LEFT_BORDER,25);
155     printf("| |              | || |             | || |
                        | || |             | | \n");
156     gotoxy(LEFT_BORDER,26);
157     printf("| '----------------' || '----------------' ||
          '----------------' || '----------------' |\n");
158     gotoxy(LEFT_BORDER,27);
```

```
159       printf(" '------------------'  '------------------'
            '------------------'  '------------------' \n");
160     }
161
162
163     scrollText(28, 100);
164
165
166     printf("\n");
167 }
168
169 void drawVictory(){
170     int i;
171     clrscr();
172
173     for(i = 0; i < 4; i++){
174
175
176       gotoxy(LEFT_BORDER, 5);
177       printf("Victory!!!");
178       gotoxy(LEFT_BORDER, 6);
179       printf("                  `                `.-/---/+yyd:`
                    ::-:::-.```             `
                                              ");
180       gotoxy(LEFT_BORDER, 7);
181       printf("                  `              `.-/hNy+ommNNNs
              ````oNNddmmNNNmmy+:.`      ``
                                              ");
182       gotoxy(LEFT_BORDER, 8);
183       printf("                  `  `  `      .-/sdNNo:shmNNdo  `
                  .dydmmNmddNmNNNNy/`       ```.    `...::--..
                                  ");
184       gotoxy(LEFT_BORDER, 9);
185       printf("                  ```          .--/+ymNo``   sm+```
                  ``/::hNNNNNNNNNNNNd`       .    `  .:--::os/yho/
                                  ");
186       gotoxy(LEFT_BORDER, 10);
187       printf("                  `        `-.-/odNNh`  ``````
              ``  +so+/yNNNNNNNMNNMd`      `         :y:/o/yddmNdN/
                                  ");
188       gotoxy(LEFT_BORDER, 11);
```
```

```
189     printf("                ‘       ‘...-+ymNNN+                ‘
            ‘‘..:-/++dNNNdNNNNNNMy      ‘‘          -omhss+:mNNmd‘
                        ");
190     gotoxy(LEFT_BORDER, 12);
191     printf("                    ‘‘.--/ymmNNNN.     ‘
            ‘‘:+::+syhddsddNNMMMMMm     ‘          ---‘    ‘yNshmo
                        ");
192     gotoxy(LEFT_BORDER, 13);
193     printf("                    ‘‘‘-:/sdmmNNNhoyooo:
            .+//+ohmdysmNNMMNMNN:  .                .oo-/dN+
                        ");
194     gotoxy(LEFT_BORDER, 14);
195     printf("                    ‘.:oshmmNNNNm+-:/sms‘   ‘‘‘‘
            ‘://oydmmm/omNNNNNNNs.‘                ‘/--:smm-
                        ");
196     gotoxy(LEFT_BORDER, 15);
197     printf("                     -//+ydmNNNNs/+osymmh:--:+oso
            +:+yhdmmmo+dNNNNNNN+‘                .:--:odNy
                        ");
198     gotoxy(LEFT_BORDER, 16);
199     printf("                       /ymddmmmmmyyddmmmmmh-.-/
            oyhdds++sdNdodNmmddmmmy-.-::::::-‘    ....‘  .:.-:
            odmN‘           ");
200     gotoxy(LEFT_BORDER, 17);
201     printf("                    ‘   ‘‘/dmmmdss+oydmmmmmmm-.-:+
            ssso++ohdos/+////+++/::::/+oosso+.--:+shho:-:
            ohmmN.          ");
202     gotoxy(LEFT_BORDER, 18);
203     printf("                    ‘       ‘./ymdddddmmmmmNNNN
            /---/+++/::/++:---::/+++///+oyhdmh/----::/+sss+/
            ohddmNN/            ");
204     gotoxy(LEFT_BORDER, 19);
205     printf("                          ‘-+hmNNNNNNNNNNh
            /:--:::::/::-...-:/oosyyhddmmNNh:::/+osyys+---:+
            oydNNNs         ");
206     gotoxy(LEFT_BORDER, 20);
207     printf("                                ../oyyhddmmNms
            /:----:/-:-...-:+oydmmmmmNNNNNys+/+oooo+:---:/+
            sydNNNs‘            ");
208     gotoxy(LEFT_BORDER, 21);
```

```
209     printf("                `           ``      ```.../moo
            /:---::::-.-:/oydmNNNNNNNNNNNNNms+/:::--:/
            osyhhdmmdo.`            ");
210     gotoxy(LEFT_BORDER, 22);
211     printf("              .`          `                /+so+ooy
            +:::::ohhdmNNmmNNNNNNNNNNNNNmdysssssyyhmNNds/.`
            `           ");
212     gotoxy(LEFT_BORDER, 23);
213     printf("          ``  ```      `                ``+oshdh
            +/::/:+hddddmmNNNNNNNNNNNNNNNyydmmmmmmmdyo-`
            `           `");
214     gotoxy(LEFT_BORDER, 24);
215     printf("          `     ``   .                 `-oo
            +//:::--:/++oyhmmmNNNNNNNmmmmmmmh.``..--.```
                        `");
216     gotoxy(LEFT_BORDER, 25);
217     printf("          `       `                    `+++oso
            +/:-.-/shhhdmmmmmmmmmmmmmmmy.
                                `     ");
218     gotoxy(LEFT_BORDER, 26);
219     printf("            `                          `-+
            oyhsss/--/oshdddmddhdmmmmmmds:`
                                    ");
220     gotoxy(LEFT_BORDER, 27);
221     printf("            `                          `/oyyoo
            +o/+shddmmmmmmmmmmmmdo.`
                                      ");
222     gotoxy(LEFT_BORDER, 28);
223     printf("              .                        .oyhy
            +//+oshddmmmmmmmmmmmdo.
                                       ");
224     gotoxy(LEFT_BORDER, 29);
225     printf("              `                          ohho
            ////+oydmmNmmmmmmmms-`
                                       ");
226     gotoxy(LEFT_BORDER, 30);
227     printf("                ``
            /+/:///::/ohmmmmmmmmmh:`
                                        `");
228     gotoxy(LEFT_BORDER, 31);
```

```
229       printf("                    `  ```
              .+://:::-:+ymmmmmmmmmdo.          ```   ```   ```
              ``    ``    ``````");
230       gotoxy(LEFT_BORDER, 32);
231       printf("                    `    ````
              /+//:::-:+hmmmmmmmmd+.`````   ```
                                          ````    ");
232   gotoxy(LEFT_BORDER, 33);
233   printf("   `   ````
              -://::::/sdmNNNNNNs.
                                             ``   ");
234   gotoxy(LEFT_BORDER, 34);
235   printf("   ```   /y/:/+
              syhdNNNNNNNNy`
              ``   ");
236   gotoxy(LEFT_BORDER, 35);
237   printf("   `   ``_
              mNdmNMMMMMMMMMMMMMm.
                                          ``   ");
238   gotoxy(LEFT_BORDER, 36);
239   }
240
241
242
243
244
245
246   scrollText(37, 100);
247
248
249 }
250
251 void scrollText(char y, char delay){
252   //Makes the Game Over / Victory text stay on the
              screen for a little while
253
254   char i, j;
255   for(i = 0; i < delay; i++){
256   gotoxy(2,y + i);
257   printf("
```
```

```
            \n");
258
259     }//for
260
261 }
262
263 void printExampleBoxes(char x, char y,char boxSize){
264     int i;
265     for(i = 0; i < 5; i++){
266       drawBox(x,y + 3*i,7-(i+1),boxSize);
267       printf(" Life: %d", (i+1));
268     }
269
270 }
```

../graphics.c

graphics.h

```
1 #ifndef _GRAPHICS_H_
2 #define _GRAPHICS_H_
3 #define LEFT_BORDER 10
4 #define LINE_NUMBER 28
5 #define CHARSET_START 48
6
7
8 void drawBox(unsigned char x, unsigned char y,unsigned
      char color,char boxSize);
9 void drawBall(unsigned char x, unsigned char y,
      unsigned char color);
10 void drawChar(unsigned char x, unsigned char y,char
      tegn);
11 void drawStriker(unsigned char x,unsigned char color,
      char strikerWidth, char strikerY);
12 void drawBounds(int x1,int y1, int x2, int y2,unsigned
      char color);
13 void moveDrawStriker(unsigned char x, unsigned char
      direction, char strikerWidth, char strikerY);
14 void drawLogo();
15 void drawGameOver();
16 void drawVictory();
17 void scrollText(char y, char delay);
18 void printExampleBoxes(char x, char y,char boxSize);
19
20 #endif
```

../graphics.h

ansi.c

```
1 #include <eZ8.h>                    // special encore
    constants, macros and flash routines
2 #include <sio.h>                    // special encore serial i
    /o routines
3 #include "ansi.h"
4
5 void fgcolor(int foreground) {
6 /*   Value      foreground      Value      foreground
7      ------------------------------------------------
8        0        Black             8        Dark Gray
9        1        Red               9        Light Red
10       2        Green            10        Light Green
11       3        Brown            11        Yellow
12       4        Blue             12        Light Blue
13       5        Purple           13        Light Purple
14       6        Cyan             14        Light Cyan
15       7        Light Gray       15        White
16 */
17   int type = 22;                 // normal text
18   if (foreground > 7) {
19     type = 1;                    // bold text
20     foreground -= 8;
21   }
22   printf("%c[%d;%dm", ESC, type, foreground+30);
23 }
24
25 void bgcolor(int background) {
26 /* IMPORTANT:   When you first use this function you
      cannot get back to true white background in
      HyperTerminal.
27    Why is that? Because ANSI does not support true
        white background (ANSI white is gray to most
        human eyes).
28                  The designers of HyperTerminal, however
                      , preferred black text on white
                      background, which is why
29                  the colors are initially like that, but
                      when the background color is first
```

```
                    changed there is no
30                  way comming back.
31    Hint:         Use resetbgcolor(); clrscr(); to force
        HyperTerminal into gray text on black background.
32
33     Value        Color
34     ---------------------
35       0          Black
36       1          Red
37       2          Green
38       3          Brown
39       4          Blue
40       5          Purple
41       6          Cyan
42       7          Gray
43 */
44   printf("%c[%dm", ESC, background+40);
45 }
46
47 void color(int foreground, int background) {
48 // combination of fgcolor() and bgcolor() - uses less
     bandwidth
49   int type = 22;               // normal text
50   if (foreground > 7) {
51     type = 1;                  // bold text
52     foreground -= 8;
53   }
54   printf("%c[%d;%d;%dm", ESC, type, foreground+30,
       background+40);
55 }
56 void underline(char on){
57   int d;
58   if (on==1){
59
60     d=4;
61    }else{
62     d=24;
63
64    }
65   printf("%c[%dm", ESC, d);
66 }
```

```
67 void clrscr(){
68   bgcolor(7);
69   printf("%c[2J", ESC);
70 }
71 void clreol(){
72   printf("%c[K",ESC);
73 }
74 void gotoxy(int x,int y){
75   printf("%c[%d;%dH",ESC,x,y);
76 }
77 void blink(char on){
78   int d;
79   if(on==1)
80     d=5;
81   else
82     d=25;
83
84   printf("%c[%dm",ESC,d);
85 }
86 void reverse(char on){
87   int d;
88   if(on==1)
89     d=7;
90     else
91     d=27;
92
93   printf("%c[%dm",ESC,d);
94 }
95
96 //Tegner et vindu, style=1 enkel linje, style=2 dobbel
     linje
97 void window(int x1,int y1,int x2, int y2,int style,char
     *s){
98   int i,j;
99   char hs,vs,h1,h2,h3,h4,headerv,headerh;
100  int height = x2 - x1+1;
101  int width = y2 - y1+1;
102
103  //Velger enkel / dobbel linjebredde
104  if (style==1){
105    hs=196;
```

```
106       vs=179;
107       h1=218;
108       h2=191;
109       h3=217;
110       h4=192;
111       headerv=180;
112       headerh=195;
113    }else{
114       hs=205;
115       vs=186;
116       h1=201;
117       h2=187;
118       h3=188;
119       h4=200;
120       headerv=185;
121       headerh=204;
122    }
123    //Printer øverste linje
124    gotoxy(x1,y1);
125    //Første hjørne
126    printf("%c%c",h1,headerv);
127    reverse(1);
128    //Regner længden på strengen
129    for(i=0;i<width;i++){
130       if(s[i]=='\0')
131    break;
132    }
133    printf("%s",s);
134
135
136    for(j=0;j<(width-i-4);j++)
137       printf(" ");
138    reverse(0);
139    printf("%c%c ",headerh,h2);
140    //Printer de vertikale linjer
141    for(i=1;i<=height-2;i++){
142       gotoxy(x1+i,y1);
143       printf("%c",vs);
144       gotoxy(x1+i,y2);
145       printf("%c",vs);
146    }
```

```
147     //Printer bunden
148     gotoxy(x2,y1);
149     printf("%c",h4);
150
151     for(i=1;i<=width-2;i++)
152        printf("%c",hs);
153
154
155     printf("%c",h3);
156 }
157
158 //Flytter skrivehodet
159 void up(int x){
160     printf("%c[%dA",ESC,x);
161 }
162 void down(int x){
163     printf("%c[%dB",ESC,x);
164 }
165 void right(int x){
166     printf("%c[%dC",ESC,x);
167 }
168 void left(int x){
169     printf("%c[%dD",ESC,x);
170 }
```

../ansi.c

ansi.h

```
 1 #ifndef _ANSI_H_
 2 #define _ANSI_H_
 3 #define ESC 0x1B
 4
 5 void fgcolor(int foreground);
 6
 7 void bgcolor(int background);
 8
 9 void color(int foreground, int background) ;
10 void underline(char on);
11 void clrscr();
12 void clreol();
13 void gotoxy(int x,int y);
14 void blink(char on);
15 void reverse(char on);
16 void window(int x1,int y1,int x2, int y2,int style,char
       *s);
17 void up(int x);
18 void down(int x);
19 void right(int x);
20 void left(int x);
21
22 #endif
```

../ansi.h

LED.c

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>
3 #include "charset.h"
4 #include "LED.h"
5 char buffer[5][6];
6
7
8 char lastString[];
9
10 int unitnr = 0;
11 int kolonnenr = 0;
12
13 int counter = 0;
14 int shift = 0;
15 int numberShifts = 0;
16
17 char done = 0;
18 char mode = 1;
19
20 char string[] = "                                                  ";
21
22 #pragma interrupt
23 void timer1int(){
24   LEDUpdate();
25 }
26
27
28 void LEDInit(){
29   PEDD = 0x00;
30   PGDD = 0x00;
31   DI();
32   T1CTL = 0x01; // Prescale value er 1, ingen division
33   T1H = 0x00;
34   T1L = 0x01;
35   T1RH = 0x24; // Reload værdi er 9216  = 2400
36   T1RL = 0x00;
37   SET_VECTOR(TIMER1, timer1int);
```

```
38    IRQ0ENL |= 0x40; //Sætter priority lav
39    T1CTL |= 0x80; //Starter timeren
40    EI();
41    done = 0;
42    counter = 0;
43    numberShifts = 0;
44    shift = 0;
45    unitnr = 0;
46    kolonnenr = 0;
47 }
48
49 void setLedMode(char modeIn){
50    mode = modeIn;
51 }
52
53
54 void LEDUpdate(){
55    switch (mode){
56       case 2: LEDUpdateOnce(); break;//Scrolls text and
             stops at 4 last characters.
57       case 3: LEDUpdatePrint(); break;//Prints 4
             characters
58       default: ; break;
59    }
60 }
61
62
63
64 void LEDSetString(char *string1){
65    int i,j;
66    for(i = 0; string1[i] != '\0'; i++){
67       string[i] = string1[i];
68
69    }
70    string[i] = '\0';
71    done = 0;
72    counter = 0;
73    numberShifts = 0;
74    shift = 0;
75    kolonnenr = 0;
76    unitnr = 0;
```

```
77
78 }
79
80
81 //Trenger kun at kaldes inden LEDUpdatePrint.
82 void LEDLoadBuffer(){
83   int i;
84   int j;
85   //Læser hver karakter
86   for(i = 0; i<5; i++){
87
88     //Læser hver kolonne
89     for(j = 0; j < 5; j++){
90       buffer[i][j] = character_data[string[i]-0x20][j];
91     }
92     buffer[i][5] = 0x00;
93   }
94
95 }
96
97 //Printer 4 bogstaver på skærmen.
98 void LEDUpdatePrint(){
99
100
101   PGOUT = buffer[unitnr][kolonnenr] ;
102
103   switch(kolonnenr){
104     case 0: PEOUT = 0x0F; break;
105     case 1: PEOUT = 0x17; break;
106     case 2: PEOUT = 0x1B; break;
107     case 3: PEOUT = 0x1D; break;
108     case 4: PEOUT = 0x1E; break;
109     case 5: PEOUT = 0x1F; break;
110   }
111
112   //Tester for unit nr, hvilket display vi er på
113   switch (unitnr) {
114     case 0:
115       PEOUT |= 0x80;
116       PEOUT &= ~(1 << 7);
117       break;
```

```
118       case 1:
119         PGOUT |= (1 << 7);
120         PGOUT &= ~(1 << 7);
121         break;
122       case 2:
123         PEOUT |= 0x20;
124         PEOUT &= ~(1 << 5);
125         break;
126       case 3:
127         PEOUT |= 0x40;
128         PEOUT &= ~(1 << 6);
129         break;
130       default:
131
132         break;
133     }
134
135     if(unitnr++ == 4){
136       unitnr = 0;
137       if(kolonnenr++ == 6){
138         kolonnenr = 0;
139       }
140     }
141 }
142
143
144
145 void LEDUpdateOnce(){
146
147   int i;
148   int j;
149
150   if(!done){
151     //Flytter en kolonne
152     if (counter == 100){
153       counter = 0;
154       shift++;
155
156       //Flytter displays
157       if(shift == 6){
158         shift = 0;
```

```
159
160         //Shifter buffer
161         for(i = 0; i < 4; i++){
162           for(j = 0; j < 6; j++){
163             buffer[i][j] = buffer[i+1][j];
164           }
165         }
166         //Sjekker for enden a string
167         if(string[numberShifts] == '\0'){
168           numberShifts = 0;
169           done = 1;
170         }
171
172         //Henter inn siste verdi i buffer
173         for(j = 0; j < 5; j++){
174           if(done){
175             //buffer[4][j] = 0x00;
176
177           }else{
178             buffer[4][j] = character_data[string[
                  numberShifts]-0x20][j];
179           }
180
181         }
182
183         //Setter siste sæyle i buffer til 0
184         buffer[4][5] = 0x00;
185         numberShifts++;
186       }
187
188     }
189
190     counter++;
191   }
192
193
194   PGOUT = buffer[unitnr][kolonnenr+shift] ;
195   // PEOUT = 0x1F & ~(1 << (4-kolonnenr));
196
197   switch(kolonnenr){
198     case 0: PEOUT = 0x0F; break;
```

```
199       case 1: PEOUT = 0x17; break;
200       case 2: PEOUT = 0x1B; break;
201       case 3: PEOUT = 0x1D; break;
202       case 4: PEOUT = 0x1E; break;
203       case 5: PEOUT = 0x1F; break;
204     }
205
206     //Tester for unit nr, hvilket display vi er på
207     switch (unitnr) {
208       case 0:
209         PEOUT |= 0x80;
210         PEOUT &= ~(1 << 7);
211         break;
212       case 1:
213         PGOUT |= (1 << 7);
214         PGOUT &= ~(1 << 7);
215         break;
216       case 2:
217         PEOUT |= 0x20;
218         PEOUT &= ~(1 << 5);
219         break;
220       case 3:
221         PEOUT |= 0x40;
222         PEOUT &= ~(1 << 6);
223         break;
224       default:
225
226         break;
227     }
228
229     if(unitnr++ == 4){
230       unitnr = 0;
231       if(kolonnenr++ == 6){
232         kolonnenr = 0;
233       }
234     }
235
236 }
```

../led.c

LED.h

```
1 #ifndef _LED_H_
2 #define _LED_H_
3
4 void timer1int();
5
6 void LEDInit();
7
8 void setLedMode(char modeIn);
9
10 void LEDUpdate();
11
12 void LEDSetString(char *string1);
13
14 void LEDUpdateOnce();
15
16 void LEDLoadBuffer();
17
18 void LEDUpdatePrint();
19
20 #endif
```

../led.h

ctimer.c

```
 1 #include <ez8.h>
 2 #include "ctimer.h"
 3 unsigned long time;
 4 int timeWait;
 5 void resetTimer(){
 6 time = 0;
 7 timeWait = 0;
 8 }
 9 #pragma interrupt
10 void timer0int(){
11 time++;
12 timeWait--;
13 }
14
15 int getDelay(){
16 return timeWait;
17 }
18
19 void setDelay(int input){
20 timeWait = input;
21 }
22
23
24 void setTimer(){
25   DI();
26   T0CTL = 0x01;
27   T0H = 0x00;
28   T0L = 0x01;
29   T0RH = 0x48; //0x05
30   T0RL = 0x00; //0xA0
31   SET_VECTOR(TIMER0, timer0int);
32   IRQ0ENH |= 0x20;
33   IRQ0ENL |= 0x20;
34   T0CTL |= 0x80;
35
36   EI();
37 }
38
```

```
39 unsigned long getCentis(){
40   return time;
41 }
```

../ctimer.c

ctimer.h

```
1 #ifndef _CTIMER_H_
2 #define _CTIMER_H_
3
4 void resetTimer();
5 void timer0int();
6 int getDelay();
7 void setDelay(int input);
8 void setTimer();
9 unsigned long getCentis();
10 #endif
```

../ctimer.h

keys.c

```
1    #include <eZ8.h>              // special encore
     constants, macros and flash routines
2 #include <sio.h>
3 #include "keys.h"
4 #include "ctimer.h"
5
6 char readKey(){
7    char a,b=0;
8    a = PFIN;
9    if((a & 0x80) == 0) b = 1;
10   if((a & 0x40) == 0) b += 2;
11   a = PDIN;
12   if((a & 0x08) == 0) b += 4;
13   return b;
14   }
15 char getKey(){
16   char key;
17   key=readKey();
18   setDelay(10);
19   while(getDelay()>0){
20   }
21   return key &=readKey();
22 }
```

../keys.c

keys.h

```
1 #ifndef _KEYS_H_
2 #define _KEYS_H_
3 #define KEY_LEFT 4
4 #define KEY_MIDDLE 2
5 #define KEY_RIGHT 1
6
7 char readKey();
8 char getKey();
9
10 #endif
```

../keys.h

12 Appendix B

Dette appendix har med øvelserne at gøre

12.1 Journal

Journal Dag 1

Øvelse 1.1

Den første øvelse startede med at blive fortrolig med udviklingsværktøjet ZDS II. Vi lavede et nyt projekt, hvor indstillingerne blev sat til en CPU af typen Z8F6403 og et registerminde på 4KB. Vi skrevskrevet en simpel C-fil der udskrev "hello world". Pakkene ez8.h og sio.h blev også inkluderet i filen. Til sidst blev C-filen gemt, bygget og simuleret i HyperTerminal.

Øvelse 1.2-1.3

I denne øvelse blev der opgivet en C-fil der indeholdt fejl. Motivationen for dette, var at lære hvordan ZDS II's debugger fungerer. Debuggeren blev brugt til at manuelt køre linje for linje i C-programmet helt til alle fejl var rettet. Debuggerens "Go to Cursor" og "Step Over" blev flittigt brugt sammen med en oversigt over variablene i og r. En af de fejl der blev opdaget var at funktionen power multiplicerte tallet a en gang for meget. Dette medførte bit-overflow, og forkerte udregninger som følge. Denne fejl blev rettet i for løkken, og programmet kørte korrekt.

Øvelse 2

I denne øvelse blev HyperTerminal's output kontrolleret ved hjælp af nogle selvdefinerede funktioner. Vi lavede alle de funktioner der blev bedt om og supplerede også med de frivilligefunktioner(up, down, left, right). Funktionen clrscr() som renser skærmen. funktionen clreol() som renser resten af linjen som markøren står på. Funktionen gotoxy(unsigned char x,unsigned char y) som flyttede markøren til den valgte position. Funktionen underline(char on) som bruges til at vælge eller fravægle om teksten skal være understreget. Funktionen blink(char on) som gør, at den skrevne tekst blinker. Funktionen reverse(char on) som bruges til at bytte om på farvene på forgrund og baggrund.

Til sidst lavede vi en funktion der kunne lave et vindue i terminalvinduet. Funktionen accepterede to sæt med koordinater(for hhv. Nordvestlige og sydøstlige hjørner), stilform for vinduet og titeltekst. Vi besluttede at lave funktionen således den ikke skrev de blanke mellemrum inde i vinduet.
På denne måde kunne man omringe tekst der allerede var skrevet. Funktionen blev udstyret med 2 forskellige stilarter, som gav forskelligt udseende vinduer.

Journal Dag 2

Øvelse 3

I denne øvelse startede vi med at lave en filstruktur vi kunne bruge senere. En header-fil blev defineret med de funktioner der blev skrevet under sidste øvelse. For at systemet skulle køre effektivt blev denne header-fil bliver kun læst ind hvis den ikke var læst ind fra før. Et nyt projekt blev lavet og de gamle funktioner blev gemt i filen ansi.c. De funktioner som beskrev fixed-point aritmetik blev skrevet i filen math.c.

Øvelse 3.1

Denne øvelse, der indeholt spørgsmål omkring bit-manipulation blev ligeledes besvaret.

Øvelse 4.1

Instruktionerne blev blot fulgt og vi fik med succes skabt en LUT vha. excel-programmet, excellut.

Øvelse 4.2

Vi skulle i denne opgave lave en funktion der kunne finde sinus for enhver integer som input – den skulle således kunne finde for negative inputs og positive inputs. Vi kunne i denne sammenhæng blot bruge to complement's repetetive natur sammen med sinus repetetive natur, og derved indse at vi blot kunne fjerne de sidste 7 bit, og kun bruge de første 9. Dette blev gjort ved en simpel bit operation, hvor inputtet blev and'et med 0x01FF, hvilket gjorde, at vores input altid var mellem 0-512.

Cosinus funktionen implementeres også ved blot at bruge egenskaben cos(a)=sin(a+90)
Da vi testede den funktion fandt vi et problem med compileren, da den compilerede -128 som 128.
Følgende workaround blev brugt: -128 = -(int)128

Øvelse 4.3

I denne øvelse skulle vi lave en funktion der kunne rotere en vektor.
Vi lavede første en funktion, der kunne initialisere en vektors x og y værdier. Disse inputs var floating point format, og blev omformet til fixed-point format.
Rotationsfunktionen blev derefter implementeret, hvor multiplikation blev gjort med de opgivne makroer og de før kreerede cos- og sinusfunktioner.

Øvelse 4.4

Denne øvelse, der indeholt spørgsmål omkring bit-manipulation blev ligeledes besvaret.

Journal Dag 3

Øvelse 5.1

I denne øvelse skulle vi lave en funktion, readKey() der kunne læse brugerinput på de tre knapper. Vi skulle derefter teste funktionen med en counter. Vores funktion virkede ikke i første omgang, da vi havde læst kredsløbs-diagrammet forkert og troet at schmitt-triggeren på inputtet var af inverterende type. Vi regner derfor med at inputværdierne var høj når de blev trykket – det var selvfølgelig ikke tilfældet og det gav os problemer – det virkede dog da vi fik det rettet. Vi havde også nogle problemer med debouncing og dårlige knapper. Alt i alt blev resultatet acceptabelt.

Øvelse 5.2

I denne øvelse skulle vi lave et program der ville outputte værdien for vores counter fra øvelse 5.1 og vise dem på LED'erne på boardet. Dette blev gjort ved at sætte værdien af counteren inputtet til LED'erne. Dette program fungerede som forventet.

Øvelse 6

I denne øvelse skulle vi lære at skrive til en timer. Vi skulle i høj grad blot følge instruktionerne. Vi valgte en prescale værdi på 2^7, altså den største mulige for timerne. Reloadværdien blev da udregnet udfra dette. Vores nederste værdi blev bare sat til 1. Interrupt prioriteten blev sat til normal. Vi havde nu problemer med at forbinde til vores board, men simuleringen i programmet fungerede fint. Dagen efter fik vi vores board til at fungere igen, og så at vores program fungerede.

Øvelse 6.1

Vi brugte vores timer fra øvelse 6 til at lave et stopur, og det fungerede også som forventet. Vi brugte her knapperne på boardet til at styre vores stopur. Det var her vigtigt at gøre så lidt som muligt i vores interrupt-funktion, da vi ønsker at den skal være hurtig. Vores samlede stopurs løsning brugte funktionen windows(unsigned char x, unsigned char y, unsigned char x1, unsigned char y2) til at danne rammen. Derefter blev der lavet en funktion som returnerer tiden fra vores timer. Dette blev brugt til at skrive splittime 1 og 2. Derefter blev tiden skrevet sådan, at den ville opdaterer hvert sekund. Programmet blev lavet sådan, at det også var muligt at resette timeren.

Journal Dag 4

Øvelse 7

I denne øvelse skulle vi skrive en streng indeholdende 4 karakterer på vores LED-skærme. Dette gjordes ved at multiplexe signalerne ud. Først skulle vi have en ny clock med en periodetid på 0.5 ms. Vores design fungerede ikke helt i første implementation, da bogstaverne var vendt forkert. Dette skyldes at søjle 0 på LED'en er søjlen helt til højre(og 4 til venstre), og vi havde således vendt bogstaverne om. Dette blev fikset og derefter fungerede vores design efter hensigten.

Øvelse 8

I denne øvelse skulle vi have en streng til at rulle henover skærmen. Dette blev løst med en en buffer som indeholdte 5 bogstaver. Fire bogstaver som vises, og det femte som er klar til at blive hentet ind ligger i bufferen. Den måde rulle effekten blev implementeret var ved at skifte søjlerne en gang til venstre, og dermed give effekten af, at der rulles mod højre. Når der er blevet skiftet 6 gange, svarende til et helt tegn indlæses et nyt tegn i videobufferen, og det som tidligere stod først smides ud. Sådan fortsætter koden indtil den løber ind i enden på strengen. Denne øvelse blev også løst, og forskellige funktioner, som udbyggede funktionaliteten blev lavet, som blev brugt senere i projektet.

12.2 Kode fra øvelserne

12.2.1 Øvelse 2

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>                    // special encore serial i
      /o routines
3
4 #define ESC 0x1B
5
6 void fgcolor(int foreground) {
7   int type = 22;                    // normal text
8   if (foreground > 7) {
9     type = 1;                       // bold text
10    foreground -= 8;
11  }
12  printf("%c[%d;%dm", ESC, type, foreground+30);
13 }
14
15 void bgcolor(int background) {
16  printf("%c[%dm", ESC, background+40);
17 }
18
19 void color(int foreground, int background) {
20 // combination of fgcolor() and bgcolor() - uses less
      bandwidth
21  int type = 22;                    // normal text
22  if (foreground > 7) {
23    type = 1;                       // bold text
24    foreground -= 8;
25  }
26  printf("%c[%d;%d;%dm", ESC, type, foreground+30,
       background+40);
27 }
28 // Tnder/slukker for understregning
29 void underline(char on){
30  int d;
31  if (on==1){
32
```

```
33        d=4;
34      }else{
35        d=24;
36
37      }
38    printf("%c[%dm", ESC, d);
39 }
40 //Renser skrmen
41 void clrscr(){
42    bgcolor(7);
43    printf("%c[2J", ESC);
44 }
45 //Renser resten af linjen
46 void clreol(){
47    printf("%c[K",ESC);
48 }
49 //Flytter markren til x,y
50 void gotoxy(int x,int y){
51    printf("%c[%d;%dH",ESC,x,y);
52 }
53 //Bruges til attnde/slukke for funktionen blink
54 void blink(char on){
55    int d;
56    if(on==1)
57      d=5;
58    else
59      d=25;
60
61    printf("%c[%dm",ESC,d);
62 }
63 //Bruges til at tnde/slukke reverse funktionen
64 void reverse(char on){
65    int d;
66    if(on==1)
67      d=7;
68      else
69      d=27;
70
71    printf("%c[%dm",ESC,d);
72 }
73
```

```
74 //Tegner et vindue, style=1 enkel linje, style=2 dobbel
       linje
75 void window(int x1,int y1,int x2, int y2,int style,char
       *s){
76   int i,j;
77   char hs,vs,h1,h2,h3,h4,headerv,headerh;
78   int height = x2 - x1+1;
79   int width = y2 - y1+1;
80
81   //Velger enkel / dobbel linjebredde
82   if (style==1){
83     hs=196;
84     vs=179;
85     h1=218;
86     h2=191;
87     h3=217;
88     h4=192;
89     headerv=180;
90     headerh=195;
91   }else{
92     hs=205;
93     vs=186;
94     h1=201;
95     h2=187;
96     h3=188;
97     h4=200;
98     headerv=185;
99     headerh=204;
100  }
101  //Printer verste linje
102  gotoxy(x1,y1);
103  //Frste hjrne
104  printf("%c%c",h1,headerv);
105  reverse(1);
106  //Regner lngden p strengen
107  for(i=0;i<width;i++){
108    if(s[i]=='\0')
109  break;
110  }
111  printf("%s",s);
112
```

```
113
114     for(j=0;j<(width-i-4);j++)
115       printf(" ");
116     reverse(0);
117     printf("%c%c ",headerh,h2);
118     //Printer de vertikale linjer
119     for(i=1;i<=height-2;i++){
120       gotoxy(x1+i,y1);
121       printf("%c",vs);
122       gotoxy(x1+i,y2);
123       printf("%c",vs);
124     }
125     //Printer bunden
126     gotoxy(x2,y1);
127     printf("%c",h4);
128
129     for(i=1;i<=width-2;i++)
130       printf("%c",hs);
131
132
133     printf("%c",h3);
134 }
135
136 //Flytter skrivehodet
137 void up(x){
138     printf("%c[%dA",ESC,x);
139 }
140 void down(x){
141     printf("%c[%dB",ESC,x);
142 }
143 void right(x){
144     printf("%c[%dC",ESC,x);
145 }
146 void left(x){
147     printf("%c[%dD",ESC,x);
148 }
149
150 //Bruges til at teste funktionen
151 void main() {
152 init_uart (_UART0,_DEFFREQ, _DEFBAUD);
153 clrscr();
```

```
154 underline(0);
155 gotoxy(10,10);
156 printf("HEJ!!!!!!! \n hej \n \n tore");
157 window(6,9,25,25,2,"hello freinds");
158 do {} while (1!=2);
159 }
```

kode/1.c

12.2.2 Øvelse 3

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>
3 #include <stdlib.h>                  // special encore
      serial i/o routines
4 #include "ansi.h"
5 #include "math.h"
6 //Main funktion bruges kun til at teste funktionen af
      math.
7 void main(){
8 struct TVector vektoren;
9 init_uart(_UART0,_DEFFREQ,_DEFBAUD);
10
11 clrscr();
12 blink(0);
13 printf("feeuhst\n");
14 printf("sinus: 45, 90, -90? \n");
15 printFix(expand(sin(64)));
16 printf(", ");
17 printFix(expand(sin(128)));
18 printf(", ");
19 printFix(expand(sin(-(int)128)));
20 printf("\n");
21 printf("long: %d, %d.\n",sin(128),sin(-(int)128));
22 initVector(&vektoren,-4,2);
23 rotate(&vektoren,-50);
24 printFix(expand(vektoren.x));
25 printf("\n");
26 printFix(expand(vektoren.y));
27
28
29 do{} while(1!=2);
30 }
```

kode/2.c

12.2.3 Øvelse 4

```
1 #include "lut.h"
2 #include "math.h"
3 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
4 #include <sio.h>
5 //Finder en vrdi i vores sinus lut
6 long sin(int x){
7   return SIN[0x01FF & x];
8 }
9
10 long cos(int x){
11   return sin(x+128);
12 }
13
14 void printFix(long i){
15   if ((i & 0x80000000) !=0) {
16     printf("-");
17     i = ~i+1;
18   }
19   printf("%ld.%04ld",i>>16, 10000*(unsigned long) (i &
        0xffff)>>16);
20 }
21
22 long expand(long i){
23   return i<<2;
24 }
25
26 //vanlig kommatal ind, konstruerer en vektor i fix14
      format.
27 void initVector(struct TVector *v, long x, long y){
28 v->x = x << FIX14_SHIFT;
29 v->y = y << FIX14_SHIFT;
30 }
31 //Roterer en vektor med en valgt vinkel
32 void rotate(struct TVector *v , int ang){
33   int sinA = sin(ang);
34   int cosA = cos(ang);
35   long tempX = v->x;
```

```
36
37
38    v->x = FIX14_MULT(tempX, cosA) - FIX14_MULT(v->y, sinA)
          ;
39    v->y = FIX14_MULT(tempX, sinA) + FIX14_MULT(v->y, cosA)
          ;
40 }
```

kode/3.c

12.2.4 Øvelse 5

```
1 #include <eZ8.h>
2 #include <sio.h>
3 #include "ansi.h"
4
5
6
7
8 char readKey(){
9 char a,b=0;
10 PEDD = 0xFF;
11 PGDD = 0xFF;
12    a = PFIN; // Sætter a lig inputtet fra knapperne
13    //Adderer den værdi som knapperne repræsenterer i
          binær format.
14    if((a & 0x80) == 0) b = 1;
15    if((a & 0x40) == 0) b += 2;
16    a = PDIN;
17    if((a & 0x08) == 0) b += 4;
18    return b;
19 }
20 //Indlæser en valgt værdi til LED
21 void writeLed(unsigned char i){
22 PEDD = 0x00;
23 PGDD = 0x00;
24 PEOUT = 0x0F;
25 PGOUT = i & 0x7F;
26 PEOUT |= 0x80;
27
28 }
29 void main(){
30    char temp,key;
31    int i=0;
32    init_uart(_UART0,_DEFFREQ,_DEFBAUD);
33
34    while(1!=2){
35       key=readKey();
36       if(key != temp){
37          temp=key;
```

```
38          // T ller  op, hver gang der er en  dring  p  keys,
                som ikke er, at den  dre  sig til nul.
39          if(key != 0x00){
40            i += 1;
41            clrscr();
42            printf("%d",i);
43          }
44        }
45    writeLed(i);
46    }
47
48 }
```

kode/4.c

12.2.5 Øvelse 6

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>
3 #include "ansi.h"
4 #include "keys.h"
5 //Laver structure time
6 struct time {
7   unsigned char h,m,s,hs;
8
9 };
10 struct time tid;
11 //Opdaterer tiden.
12 #pragma interrupt
13 void timer0int(){
14   tid.hs++;
15   if(tid.hs == 100){
16     tid.s++;
17     tid.hs=0;
18
19
20     if(tid.s == 60){
21     tid.m++;
22     tid.s=0;
23     if(tid.m == 60){
24     tid.h++;
25     tid.m=0;
26   }}
27   //Skrevet indenfor iflkken som opdateres hvert
        sekund, sdan at tiden skrives hvert sekund.
28   gotoxy(2,2);
29   printf("Time: %02d:%02d:%02d",tid.h,tid.m,tid.s);
30   }
31
32 }
33
34 void setTimer(){
35   char preScale = 0x07<<3;
36   DI();
```

```
37    T0CTL = 0x01 | preScale;
38    T0H = 0x00;
39    T0L = 0x01;
40    T0RH = 0x05; // Sætter reload så dan, at den tæller
          hvert 100 del af et sekund.
41    T0RL = 0xA0;
42    SET_VECTOR(TIMER0, timer0int);
43    IRQ0ENH |= 0x20; //høj prioritet
44    IRQ0ENL |= 0x20;
45    T0CTL |= 0x80; //enable count
46
47    EI();
48 }
49 //Stopper timeren
50 void stop(){
51    DI();
52    T0CTL &= 0x7F;
53
54 }
55
56 void main(){
57 char key;
58 init_uart(_UART0,_DEFFREQ,_DEFBAUD);
59 tid.h=0;
60 tid.m,tid.s,tid.hs=0;
61 clrscr();
62 window(1,1,10,25,1,"Stopur");
63 setTimer();
64
65 while(1 != 2){
66 //Keys realiserer de forskellige funktioner i timeren
67 key=getKey();
68    if(key==6){
69    //resetter timeren
70       stop();
71       tid.h=0;
72       tid.m=0;
73       tid.s=0;
74       tid.hs=0;
75       gotoxy(2,2);
76       printf("Time: %02d:%02d:%02d",tid.h,tid.m,tid.s);
```

```
77 }
78   else if(key==4){
79   //Skriver splittime2
80     gotoxy(4,2);
81     printf("Splittime2: %02d:%02d:%02d",tid.h,tid.m,tid
         .s);
82   }
83   else if (key==2){
84   Skriver splittime1
85     gotoxy(3,2);
86     printf("Splittime1: %02d:%02d:%02d",tid.h,tid.m,tid
         .s);
87     }
88   else if (key==1){
89   //Starter/stopper uret
90     if((T0CTL & 0x80) == 0x80){
91       stop();
92     } else{
93       T0CTL|=0x80;
94       EI();
95     }
96   }}}
```

kode/5.c

12.2.6 Øvelse 7 og 8

```
1 #include <eZ8.h>                    // special encore
      constants, macros and flash routines
2 #include <sio.h>
3 #include "ansi.h"
4 #include "charset.h"
5 char buffer[5][6];
6 char column;
7 char display;
8 char index=0;
9 int strlength;
10 char *string;
11 long flag2;
12 //Krer vores interrupt
13 #pragma interrupt
14 void timer1int(){
15   LEDUpdate();:
16   flag2++;
17 }
18 //Indlser et nyt tegn p den sidste plads i buffer
19 void loadNew(){
20 char j,i;
21
22   for (i=0;i<5;i++){
23     for(j=0;j<6;j++){
24       if(i<4)
25       buffer[i][j]=buffer[i+1][j];
26       else
27       buffer[i][j]=character_data[*string-0x20][j];
28
29       }
30
31    }
32 string++;
33 buffer[4][5]=0;
34 }
35
36 void LEDInit(){
37   PEDD = 0x00;
```

```
38   PGDD = 0x00;
39   DI();
40   T1CTL = 0x01; // Prescale value er 1, ingen division
41   T1H = 0x00;
42   T1L = 0x01;
43   T1RH = 0x24; // Reload v rdi er 9216 = 2400
44   T1RL = 0x00;
45   SET_VECTOR(TIMER1, timer1int);
46   IRQ0ENL |= 0x40; // S tter priority lav
47   T1CTL |= 0x80; //Starter timeren
48   EI();
49 }
50
51 void LEDSetString(char *s){
52   char i,j;
53   strlength = 0;
54   string = s;
55   //Regner l ngden p strengen
56   while(* string != '\0'){
57   strlength++;
58   string++;
59   }
60   string -= strlength;// S tter pegeren tilbage til
        starten p strengen
61   for(i=0;i <5; i++) string[strlength+i]=' ';
62   strlength+=5;
63   string[strlength]='\0';
64   // L ser hver karakter
65   for(i = 0; i<5; i++){
66     // L ser hver kolonne
67     for(j = 0; j < 5; j++){
68       buffer[i][j] = character_data[*string-0x20][j];
69     }
70     buffer[i][5]=0;
71     string++;
72   }
73   display = 0;
74   column = 0;
75   index = 0;
76   printf("%d",strlength);
77 }
```

```
78
79 void LEDUpdate(){
80     //Opdaterer Vores LED.
81     PEOUT = 0x1F;
82     PEOUT ^= ( 1 << (4-column));
83
84     PGOUT = *(&buffer[0][0] + column + display*6+index)
           ;
85
86
87
88     if(display==0){
89       PEOUT |=0x80;//D1 clock
90       PEOUT &= ~(1 << 7);
91     }
92     else if(display==1){
93       PGOUT|=0x80; //D2 clock
94       PGOUT &= ~(1 << 7);
95     }
96     else if(display==2){
97       PEOUT|=0x20; //D3 clock
98       PEOUT &= ~(1 << 5);
99     }
100    else if(display==3){
101      PEOUT|=0x40; //D4 clock
102      PEOUT &= ~(1 << 6);
103    }
104
105  display++;
106  if(display>3){
107    display=0;
108    column++;
109  }
110  if(column>4){
111    column=0;
112  }
113  if(flag2 > 150 && strlength>9){
114    index++;
115    flag2 = 0;
116  }
117
```

```
118
119     if(index>5){
120       index=0;
121       loadNew();
122
123
124 }
125 main(){
126     char str2[] = "FEEL THE PUMP!";
127     char str3[5] = {'H','E','E','E','\0'};
128     init_uart(_UART0,_DEFFREQ,_DEFBAUD);
129     flag2=0;
130
131 clrscr();
132 printf("HALLO! ffffffff!");
133 LEDInit();
134 display2(str2,str3);
135
136 }
```

kode/6.c