Hårdvarulaboration II

Mål

- Ge en allmän förståelse för busshantering och adressavkodning
- Fördjupad förståelse för ARM:s adresshantering
- Ge träning i virteknik för prototypbyggen
- Ge träning i assemberprogrammering (ARM) med utvecklingsverktyget IAR Embedded Workbench v3.21
- Laborationen ligger till grund för hårdvaruprojekt

Förkunskaper

- Utförd hårdvarulaboration I

Uppläggning. Arbetsform. Examination

Laborationerna sker i grupper om två, vissa fall tre.

F-uppgifterna utgör förberedande uppgifter och ska utföras före laborationstillfället. Laborations-assistenten har rätt att kontrollera dig och kan vid eventuellt missnöje med din förberedelse skicka hem din grupp.

L-uppgifter är laborationsuppgifter som ska utföras under labassistentens handledning.

Examinationen sköts fortlöpande tillsammans med labassistenten.

Uppgift

Laborationen går ut på att koppla till en grafisk display till högskolans ARM-baserade labplattform. Adressavkodning ska ske med dekoderkretsen 74HC138. Laborationen ligger till grund för det hårdvaruprojekt som avslutar första delen av kursen datorsystemteknik.

Teorin till dom olika uppgifterna ligger i bilagor. Vilka bilagor som är lämpliga att läsa står i de olika deluppgifterna.

Här följer en kort beskrivning av bilagorna

Bilaga 1	Beskrivning av ARM:s adressbanker och adresskarta
Bilaga 2	Kopplingsschema för Lab2

Förberedelsuppgifter

F1 Adresskartan: Analys av kretsschema och adresskarta. Bilaga 1 och 2

I schemat i bilaga 2 syns att det är signalen nCS3 som slår av/på demuxen 74HC138. Det betyder att det är genom att skriva/läsa till minnesbank3 man når både knappar och display. Analysera makrofilen `config_SDRAM.mac` och beräkna med hjälp av denna och processormanual (finns på hemsidan) fram vad som gäller för konfigurering av bank3. Det betyder att man ska analysera raderna som berör:

```
1. Chip select to bank 3
2. Turn off bank0 and Initiera bank 3
```

Genom detta får man fram adresser för bank3 som behövs för förberedelseuppgift 4!!

Exempel på tolkning av makrofil)

```
__message "Watchdog-timer off";
__writeMemory16(0xa500,0x07ffa002,"Memory");
```

Dessa två rader ska tolkas på följande sätt:

Första raden är bara ett meddelande som skrivs ut i messages fönstret vid nerladdning. Andra raden betyder att man skriver 16-bitar till en viss adress. I exemplet skriver man talet `0xA500` till adressen `0x07FFA002`. Tittar man i manualen (sid 1.14-1.18) och utgår från att basadressen är `0x07FF0000` så skrivs till BTCON-registret (`0xA002`). Läser man om BTCON-registret på sidan 13.5 att skrivning av talet 0xA5 till bit [15:8] betyder watchdog off!

Syftet med uppgiften är att få ingående förståelse hur minnet konfigureras upp.

F3 labkort: Vira upp elektronik

Vira upp enligt kopplingsschemat i bilaga 2. Tänk på att placera komponenterna på ett strategiska ställen så att display och framtida telefontangentbord får plats. Tangentbordet kopplas ej in nu!

Displayen ska (om den inte redan är monterad) fästas i ett underliggande kretskort. För dom med lös display gäller att displayen ska monteras i stickan på det lilla kretskortet så att man ser det underliggande kortet. *INTE åt andra hållet!*

Denna gång ska du vira IC-kretsar vilket skiljer sig lite:

- Placera ut virsocklarna. Använd gärna lite för stora virsocklar. Då kan du sticka ner avkopplingskondensatorn bredvid själva IC-kretsen. Fäst virsocklar med speciella märklappar (går att göra själv) för virsocklar eller fäst sockel med lite virtråd.
- OBS! Tänk på att matningsspänning inte är utsatt på logikkretsarna. För bennummer, se datablad på hemsidan!

Vira inte allt utan vira och kontrollera i steg enligt följande lista:

- Börja vira avkopplingskondensatorerna direkt till aktuell IC-kapsel. Dessa ska vara kopplade mellan VCC och GND på kretsen med *korta* sladdar. Vira matningsspänning VCC med röd vitråd och "jord" (GND) med svart. Pricka av i schemat att ledningsdragningen är utförd.
- Fortsätt med matningen. Tag en IC-kapsel i taget. Varje IC-kapsel ska helst ha en egen förbindelse direkt till listen med VCC och GND. Vira matningsspänning VCC med röd vitråd och "jord" (GND) med svart. Pricka av i schemat att ledningsdragningen är utförd. Kontrollera och pricka av.
- Fortsätt med styrsignalerna, dvs alla former av kontroll och chip-selectsignaler tex nCS3, nOE, nCS_DISP osv. Tag en IC-kapsel i taget. Använd gul tråd till styrsignaler.
- Behöver några av signalerna på en IC på kapsel gå till VCC eller GND så dra dem till GND resp VCC på aktuell IC. På så sätt minskas antal kablar till matningslisten på labkortet.
- Avsluta med att dra datasignalerna (D0-D7) och använd vit virtråd.

Kontrollmätning!

- Kontrollera virningarna m h a ohmmätning. Ohmmätning utförs från ovansidan av kortet. Pricka av kontinuerligt!
- Kontrollera så att inga närliggande ben kommer i kontakt med de aktuella virtrådarna.
- Gör kontaktmätning mellan VCC och GND (listen på kortet). Är det kontakt där emellan har du kortslutning någonstans. Då går säkringen så fort du kopplar in matning.

F4 Testprogram: inhantering/uthantering. Bilaga 2

Nedanstående kod har som syfte att excitera de signaler som krävs för att kunna göra mätningar på kortet.

```
#include "s3c3410x_asm.h"

RESET_VEC             EQU        0x00000000
START                 EQU        0x00001000
STACK_START           EQU        0x00003000
BUTTON_ADR            EQU        <fyll i själv, se förb.uppg1 och Föreläsn 3>
DISPLAY_CMD           EQU        <fyll i själv>
DISPLAY_DAT           EQU        <fyll i själv>
; ---------------------------------------------------------------------
                      CODE32

                      ORG     RESET_VEC
                      B       MAIN

                      ORG     START
MAIN:
; ----------------------- initiering -------------------
                      LDR     SP, =STACK_START
; ------------------------------------------------------

; --------------------- Skrivloop ----------------------
                      MOV     R3, #0x0A
                      MOV     R4, #0x05
loop_w
                      LDR     R0, =DISPLAY_CMD
                      LDR     R1, =DISPLAY_DAT
                      STRb    R3, [R0]
                      STRb    R4, [R1]
                      B       loop_w

; --------------------- Läsloop ----------------------
                      MOV     R2, #0xFF
loop_r
                      LDR     R0, =BUTTON_ADR
                      LDRb    R1, [R0]      ; läs knappvärde
                      STRb    R2, [R0]      ; skriv knappvärde
                      B       loop_r
```

Kommentar skrivloop (`loop_w`):

Genom att först skriva 0x0A och sedan 0x05 fås en toggling på databitar D0-D3. Vid skrivning av ett displaykommando ska data vara 0x0A, dvs 1010b och vid displaydata 0x05, dvs 0101b. De styrsignaler som genereras mot displaykretsen 1560 mäts sedan upp med oscilloskop.

Kommentar läsloop (`loop_r`):

Först läses knapparna av och sedan skrivs avläst knappvärde tillbaka. Då knapparna är aktivt låga kommer man se låga pulser på databussen vid avläsningen. Eftersom kretsen 74HC541 är kopplad att endast släppa igenom knapparna vid läsning testas sedan att skriva till knapparna. Fungerar allt som det ska bör inte knapparnas låga värde komma igenom till databussen vid detta tillfälle, utan endast ettor (därav talet 0xFF).

Koden innehåller två eviga loopar. Dessa två ska köras var för sig och du kommer mäta upp och verifiera funktionen hos din koppling med oscilloscop. Detta görs senare på laborationstillfället. Innan dess skall du studera manualen till displayen samt kopplingen för att kunna avgöra lämpliga adresser för knappar och displayer!

Gör ett nytt projekt (i IAR) för ovanstående kod. Du kan utgå från ett tomt projekt som heter `blank_proj.zip`.

Branch-instruktionen på adress `0x00000000` (RESET_VEC) är där för att PC sätts dit automatiskt av utvecklingsverktyget då man laddat ner koden med. Genom detta trick kan man börja stega direkt och slipper sätta programräknaren manuellt.

Titta på kopplingen och fundera igenom vilka signaler som behövs sättas vid skrivning respektive läsning. Fyll i tabellen nedan

	A0	A1	nWBE0	nOE
Skriver kommando till Displayen				
Skriver data till Displayen				
Läsa från knapparna				

F5 Testprogram: initiering av display. Datablad på hemsida

Displaydrivarkretsen är SED1560 och är inbyggd i displayen. För att initiera displayen behövs följande kommandon utföras.

1. Reset
2. Output status register set: Case2. D3=0
3. Duty select: 1/64
4. Built-in power supply ON/OFF: On
5. Power-on completion
6. Display START Line set: Adress 0
7. Electronik control register set: Max contrast
8. ADC select: Normal
9. Normal/reverse display: Normal
10. Display ON/OFF: On

Genom att skicka ett kommando i taget till displayen kan detta utföras. Följ beskrivningen på kommandon i manualen för SED1560 sid 7-38 och framåt och skriv kod för att skriva till displayen.

ex) kod för att skicka reset skickat till displayen. Förutsätter givetvis DISPLAY_CMD är initerad rätt!

```
LDR     R0, =DISPLAY_CMD
MOV     R1, #0xE2       // kommando för reset. Se manual!
STRb    R1,[R0]
```

Studera databladen och bekanta dig med displayen!

Laborationsuppgifter

L1 Databussen: Oscilloskopmätning vid läsning

I denna uppgiften ska de signaler som styr de olika kretsarna mätas upp. Genom att göra den här mätningen utför man mycket värdefull felsökning/verifiering att kopplingen fungerar!

1. Börja med att ansluta CH2 till nCS3. Välj CH2 som triggkälla (Görs genom knappen TRIG MENU). Det betyder att det är nCS3-pulsen som bestämmer när oscilloscopskärmen kommer ritas ut.

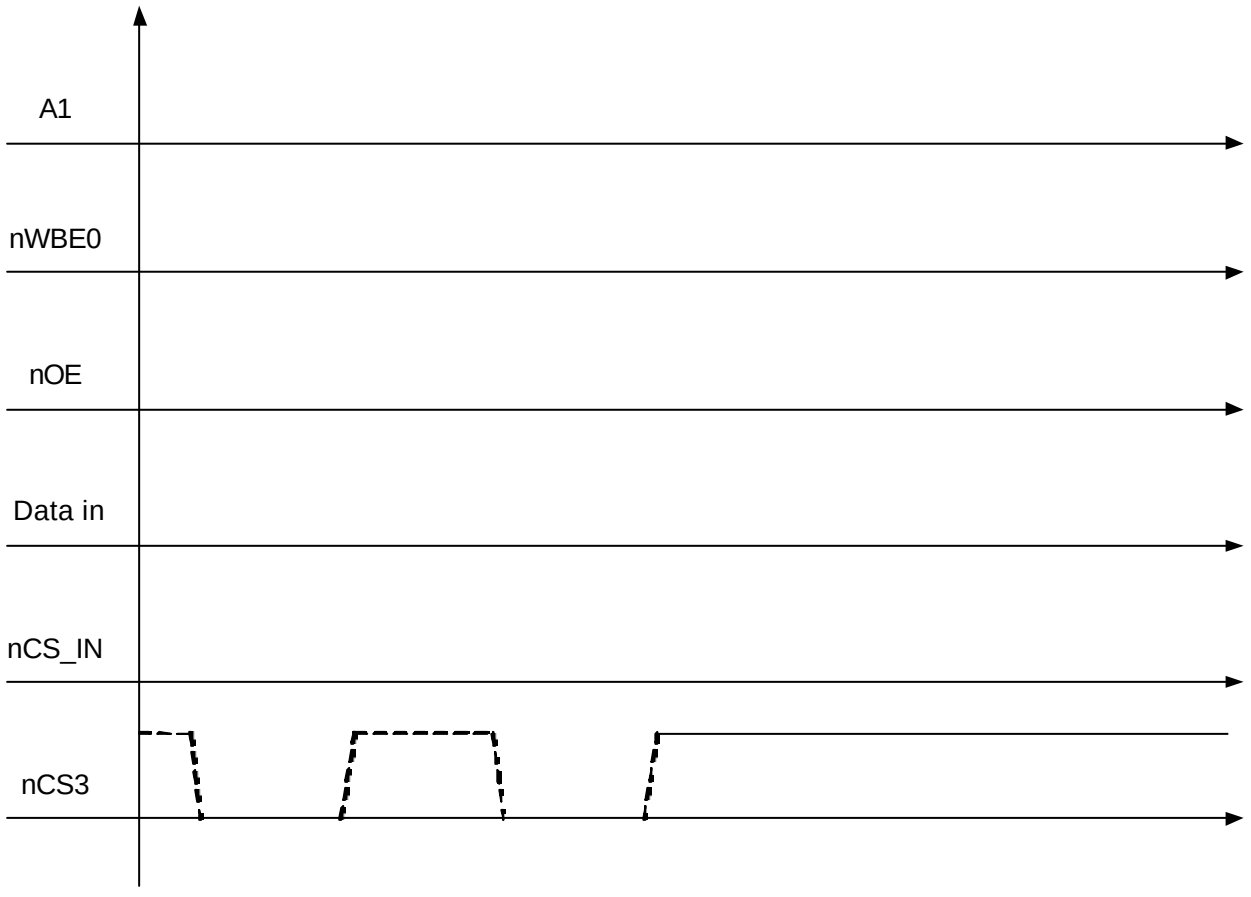
2. Ställ in CH2 så att nCS3 ser ut ungefär som i grafen nedan. Man kan flytta pulsen så med HORIZONTAL POSITION. Med denna pulsen som referens kan man nu mäta resterande signaler

Låt oscilloscopets inställningar vara som dom är och kör koden `loop_r` från förberedelseuppgift F4. Mät med CH1 upp och rita in signalerna som anges i grafen INLÄSNING. Det kan vara lurigt att få till triggningen ibland. En variant är att ta enkla pulssekvenser. Gör detta med knappen SINGEL SEQ. Tänk dock på att det då blir stillbilder. För ny mätning tryck åter igen på knappen

Försök göra detta så skalenligt som möjligt. Tiden för en ruta står på skärmen. De intressanta signalerna är då nCS3 är låg!

Data in fås genom att hålla in en knapp och mäta på motsvarande datasignal. Tänk på att knapparna är aktivt låga.

INLÄSNING till processor (från knappar)



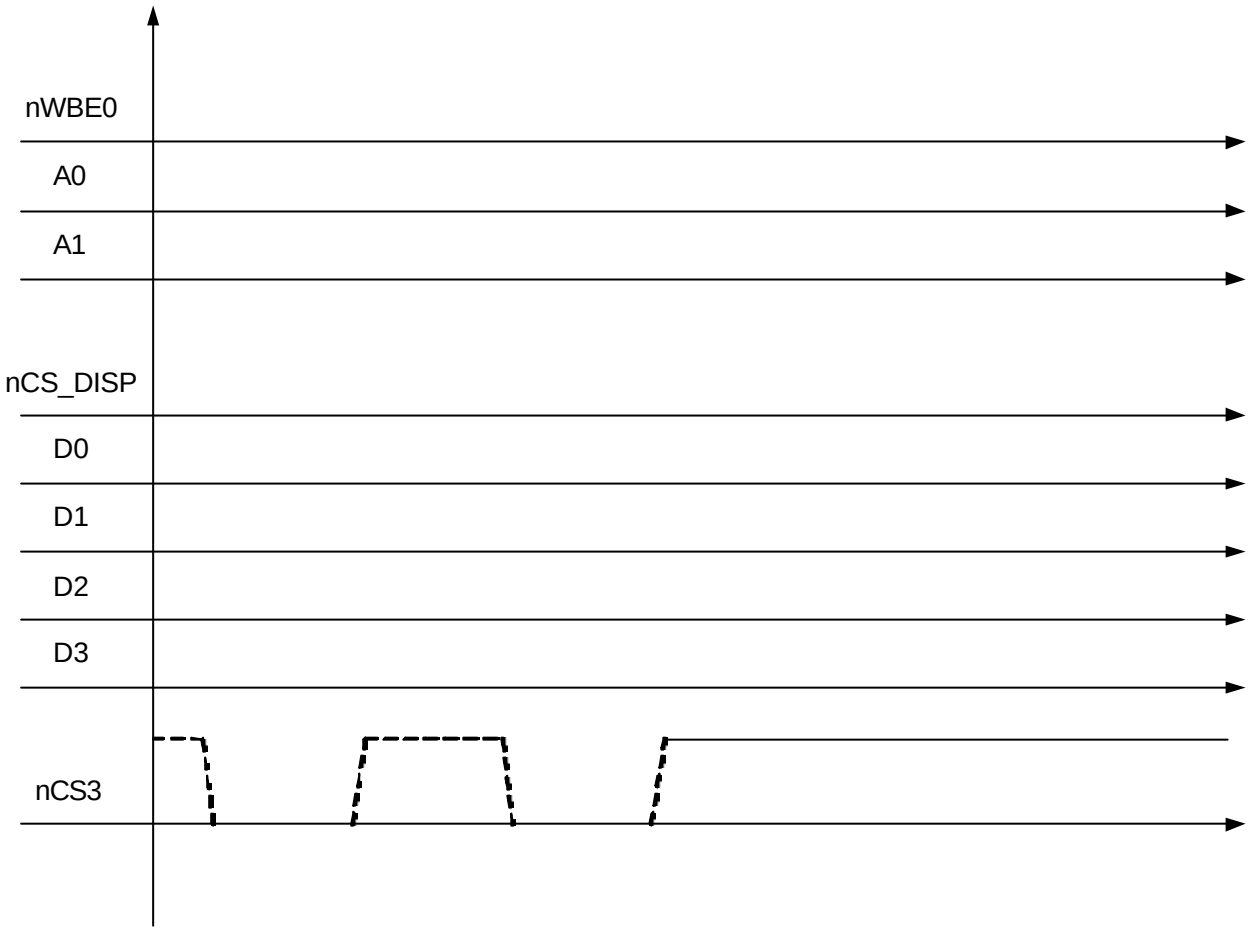
Den första låga pulsen på nCS3 är då man läser, den andra då man skriver.

Det intressanta att se är när data kommer från knapparna till processorn. Vid vilka tillfällen kommer data från knapparna in till processorn? Stämmer detta med teorin?

L2 Databussen: Oscilloskopmätning vid skrivning

Gör motsvarande som i L1 fast för koden `loop_w` och fyll i SKRIVNING-grafen!

SKRIVNING från processor (till display)



Notera vad som skiljer när man skriver data respektive kommando till displayen! Tänk efter vilket data i koden som skrivs som kommando respektive data

Slutsatser:

Läs ut ur grafen vid skrivning resp läsning och läs av vad följande signaler har för värden då man...

	A0	A1	nWBE0	nOE
skriver kommando till Displayen.				
skriver data till Displayen.				
läser från knapparna.				

Jämför detta med förberedelseuppgift 4!

L3 Display:

Displayen är uppdelad i ett antal sidor (pages). Då man vill skriva till displayen måste man först sätta vilken sida (y-led) , sen vilken kolumn (x-led). Man skriver högbyte och lågbyte för sig med olika kommandon. Efter detta skriver man data till displayen. Man måste då skriva en hel byte, dvs 8 rader i en kolumn på en gång. Varje gång man skriver till skärmen uppdateras kolumnräknaren, men man kan givetvis sätta kolumnen manuellt om man inte vill skriva på nästa kolumn.

Varje kolumn har en offset på 0x21, dvs kolumn 0x21 är den första (vänstra hörnet längst upp) om man tänker sig kontakten på ovansidan av displayen..

Displayen har 8 sidor och 100 kolumner. Eftersom varje sida innehåller 8 rader blir totala storleken på displayen 64 rader x 100 kolumner.

a) Skriv ett program som fyller var fjärde rad med linjer.

b) Skriv sedan ett program som rensar skärmen.

Sätt samman alltihop till ett program med följande funktion

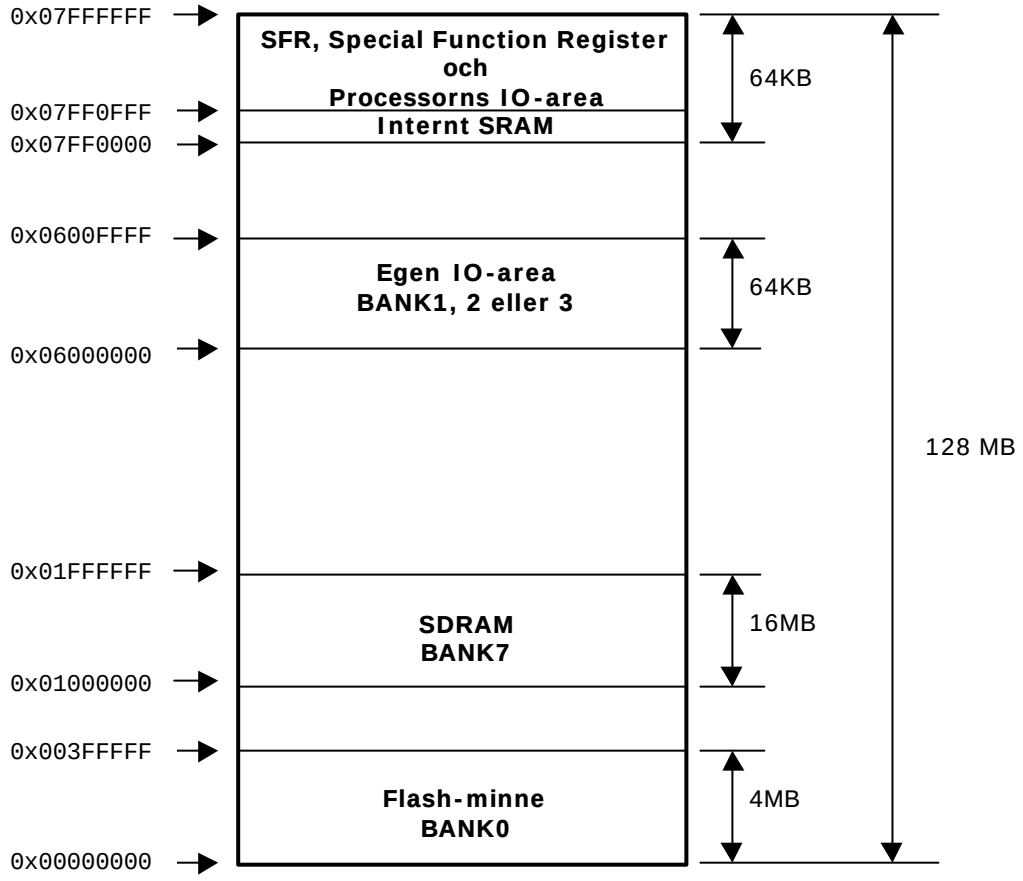
Programflöde

1. Initiering av processor och display
2. Vänta på att höger knapp trycks
3. Rita linjer på skärmen
4. Vänta på att vänster knapp trycks
5. Rensa skärmen
6. Avsluta (lämpligt med evig loop)

BILAGA 1: Adresskarta på ARM

Möjlig adresskarta

Med tanke på de resurser som finns på processormodulen är en adresskarta av följande utseende möjlig.



Adresskartan på ARM kan ändras – ovanstående adressområden är ett förslag. ARM-processorn har ett mycket flexibelt sätt att hantera disponibelt minnesutrymme.

Konvention

I datortekniska sammanhang används ibland förkortningen 1 M för värdet 1024 × 1024 (=1048576). Detta är vilseledande eftersom 1 M vanligtvis betyder en miljon. I denna framställning betyder 1 M (ett mega) 1024 × 1024. När det gäller värdet 1024 finns ett gammalt prefix – nämligen 1 K. Stort K betyder sålunda 1024.

Adressrymd på ARM

Processorn har totalt 24 adressledningar, vilket medför att adresseringskapaciteten är lika med 16 megaord(2^{24} = 16777216 = 16777216/(1024 × 1024) = 16 M). Hur kan då 128 MB adresseras? Jo, det finns också 8 väljarsignaler (chipselect-signaler nCS0 – nCS7). Detta medför att minnet kan delas upp i 8 separata områden – s k banker. På så sätt blir den totala adresseringskapaciteten lika med 8 × 16 megaord, dvs 128 megaord.

Om databredden på varje bank är lika med 8 bitar blir processorns totala kapacitet lika med 128 MB. Vid 16 bitars databredd blir kapaciteten för varje bank 32 MB, dvs totalt 256 MB. I fortsättningen förutsätts att en 8 bitars databuss används.

Minnesbanker

De åtta bankerna är numrerade från 0 (BANK0) till 7 (BANK7). Bank 0 används till flash-minnet. Detta är grundinställningen. Minnen av DRAM-typ måste ligga på bank 6 eller 7. Labsystemets SDRAM utnyttjar bank 7. Labsystemet tillåter användaren att experimentera med tre banker (bank 1, 2 och 3). Varje bank måste ha ett unikt adressområde men områden kan variera i storlek och de behöver inte komma i någon speciell ordning. Minsta storlek är 64 KB och största är 16 MB.

Varje bank har en unik chipselect-signal (nCS0-nCS7) kopplad till sig. Då man konfigurerat bankerna kommer automatiskt rätt chipselectsignal att gå låg, beroende på vilken area man skriver/läser i.

Skapa adresskartan

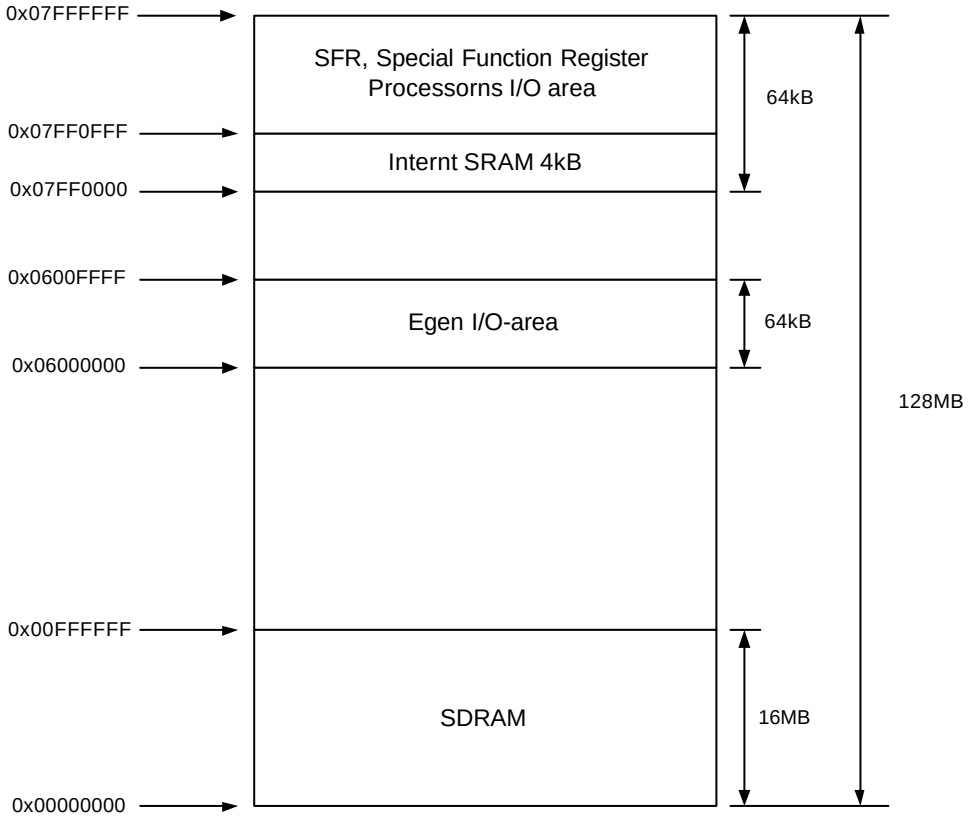
Adresskartan skapas genom att konfigurera ett antal register hos processorn. Konfigureringen beskrivs i processormanualen Kapitel 4. En kort sammanfattning av de register som används följer här:

SYSCFG	Bestämmer i huvdsak om interna SRAM ska användas, startadress för SFR samt minnestyperna för bank 6 och 7.
BANKCON0-5	Konfigurering av minnesbankerna 0-5. Bestämmer bla. bussbredd, timing samt vilka adresser som ska ingå i bankerna.
BANKCON6-7	Speciella kontrollregister för bank 6-7. Handlar om specialinställningar för bla DRAM
PCON1	Högsta bitarna [15:8] styr om adresssignalerna A16-A23 ska användas.
PCON2	Här konfigureras om chipselect-signalerna till dom olika bankerna ska vara på eller inte.
PCON3	Slår av och på styrsignaler till minnet. Bla. nWBE0-1

Detta skulle kunna göras i början av koden, men då processorn vid RESET inte har några minnesbanker konfigurerade hamnar man i ett moment22 Koden kan ju inte kan laddas ner till något minne för att köras om inget minne är konfigurerat! Vid utveckling löser man det genom att först ladda ner data rakt ner i minnet *innan den vanliga koden laddas ner!* Man skriver till dom minnesmappade specialregistren och konfigurerar upp minnet innan koden laddas ner. På IAR Embedded Workbench sker detta genom en sk makrofil

Makrofil

Makrofilen laddas automatiskt ner av då du laddar ner din kod. I den färdiga filen `config_SDRAM.mac` som är inkluderad i projektet finns inställningar för en adresskarta enligt nedan.



Makrofilen `config_SDRAM.mac` som initierar minnet:

```
// ISRAM
execUserPreload()
{
  __message "Watchdog-timer off";
  __writeMemory16(0xa500,0x07ffa002,"Memory");
  __message "Initiate clock";
  __writeMemory8(0x18,0x07ffd003,"Memory");
  __message "Cache OFF and set SFR 7ff";
  __writeMemory32(0x00317ff0,0x07ff1000,"Memory");
  __message "Chip select to bank 3";
  __writeMemory16(0xaaaa,0x07ffb014,"Memory");
  __message "Fixa skrivsignal: WBE0 ";
  __writeMemory16(0x2aaa,0x07ffb016,"Memory");
  __message "Initiate A16-A23 ";
  __writeMemory16(0xff00,0x07ffb012,"Memory");
  __message "Turn off bank0 and Initiera bank 3";
  __writeMemory32(0x0,0x07ff2000,"Memory");
  __writeMemory32(0xE0180070,0x07ff200c,"Memory");
  __message "Dummyskriving till bank6";
  __writeMemory32(0x0,0x07ff2018,"Memory");
  __message "Initiera 16 MB SDRAM på bank 7 adr 0";
  __writeMemory32(0x20000183,0x07ff201c,"Memory");
  __writeMemory32(0x00003a69,0x07ff2020,"Memory");
  __message "och kvittera  av minneskarta";
}

execUserReset()
{
 __message "Watchdog-timer off";
  __writeMemory16(0xa500,0x07ffa002,"Memory");
  __message "Initiate clock";
  __writeMemory8(0x18,0x07ffd003,"Memory");
  __message "Cache OFF and set SFR 7ff";
  __writeMemory32(0x00317ff0,0x07ff1000,"Memory");
  __message "Chip select to bank 3";
  __writeMemory16(0xaaaa,0x07ffb014,"Memory");
  __message "Fixa skrivsignal: WBE0 ";
  __writeMemory16(0x2aaa,0x07ffb016,"Memory");
  __message "Initiate A16-A23 ";
  __writeMemory16(0xff00,0x07ffb012,"Memory");
  __message "Turn off bank0 and Initiera bank 3";
  __writeMemory32(0x0,0x07ff2000,"Memory");
  __writeMemory32(0xE0180070,0x07ff200c,"Memory");
  __message "Dummyskriving till bank6";
  __writeMemory32(0x0,0x07ff2018,"Memory");
  __message "Initiera 16 MB SDRAM på bank 7 adr 0";
  __writeMemory32(0x20000183,0x07ff201c,"Memory");
  __writeMemory32(0x00003a69,0x07ff2020,"Memory");
  __message "och kvittera  av minneskarta";
}
```

Syntax

Funktionen `execUserPreload()` anropas då man laddar ner koden och `execUserReset()` då man gör reset i debugläget. Syntaxen är följande:

```
ex)
  __writeMemory32(0x00003a69,0x07ff2020,"Memory");
```

Skriver 32-bitar till en minnesadress. I detta fall talet `0x00003a69` till minnesplats `0x07ff2020`

```
  __writeMemory16(0x2aaa,0x07ffb016,"Memory");
```

Skriver 16-bitar till en minnesadress. I detta fall talet `0x2aaa` till minnesplats `0x07ffb016`

```
  __writeMemory8(0x18,0x07ffd003,"Memory");
```

Skriver 8-bitar till en minnesadress. I detta fall talet `0x18` till minnesplats `0x07ffd003`

```
  __message "och kvittera  av minneskarta";
```

Skriver ut meddelandet `och kvittera  av minneskarta` till logfönstret

BILAGA 3: Kopplingsschema - lab 2

Schemat finns att hämta från kursens hemsida och öppnas med schemaritningsprogrammet OrCAD Capture. Där kan schemat även editeras om man så önskar. Capture är en del av ett komplett programpaket för elektronikkonstruktion med möjlighet till simulering, schemaritning och kretskortsframtagning.