Laddningsstation

Ett digitalt projekt

Våren 2003
av

Christel Balogh och Anders Kamf

Lunds Tekniska Högskola
Institutionen för informationsteknologi

Abstract

The students of electrical engineering in Lund have a couple of vending machines, serving coffee, food and candy around the clock. To limit the huge amount of coins, which demands lots of resources to handle, a card payment system was developed in 1999. This includes a database that takes care of customer accounts, magnetic card number, card balance etc.

This report describes the development of a charge station for magnetic cards in the system above. The charge station includes a magnetic card reader, a display for balance information, a banknote validator and connection to the database via RS232. The report includes choice of components, circuit board design and program code.

Innehållsförteckning

1. Inledning

Denna rapport beskriver genomförandet av ett digitalt projektet i kursen med samma namn. Projektet gick ut på att konstruera en laddningsstation och genomfördes våren 2003 av Christel Balogh, 02E95 (e95cb@efd.lth.se) och Anders Kamf, 02E96 (e96ak@efd.lth.se).

1.1. Disposition

Vi börjar med att presentera en bakgrund för att på så vis ge en bild av vad projektet går ut på och formulera själva problemet. Därefter formulerar vi kraven och går in på lösningen, först hårdvaran sedan mjukvaran. Sedan går vi in på det praktiska genomförandet och avslutar med resultat, slutsats och diskussion om fortsatt arbete.

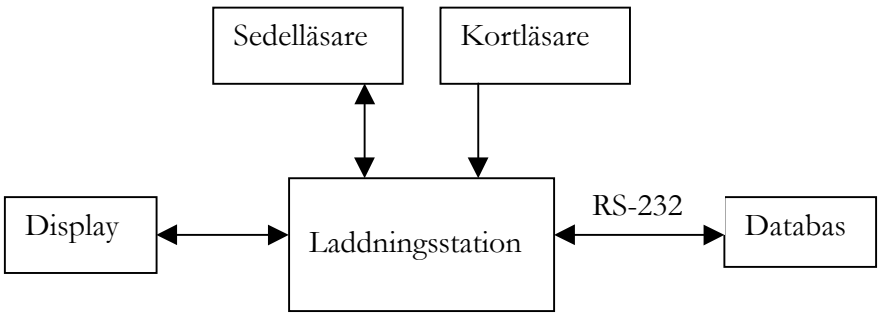
1.2. Bakgrund

I Edekvata, E-sektionens lokaler i E-husets källare, finns läsk- och godisautomater, s.k. mojter. I dessa betalar man idag med mynt. Då det handlas för ganska stora summor blir det mycket jobb med hantering av alla mynt. Under en längre tid har det funnits idéer om att skapa ett kortbetalningssystem. Tanken är att varje mojt ska ha en kortläsare, istället för att stoppa in mynt i mojten drar kunden sitt kort och handlar via detta.

Tanken är att varje kortkund registrerar sig och sitt kort i ett kortsystem. En databas håller reda på kunder, kort, saldo, mojter och varornas priser. Här registreras även alla transaktioner.

Ett första steg i implementationen togs i ett digitalt projekt 1999, där delen med databas och inköp i mojt behandlades [3]. En annan del i systemet är en laddningsstation för korten där kunderna själva drar sitt kort och matar in pengar varvid saldot (lagrat i databasen) uppdateras. Det är konstruktionen av denna laddningsstation som detta digitala projekt innefattar.

En enkel skiss över laddningsstationen finns i *Figur 1*.



Figur 1. *Blockschema över laddningsstationen*

2. Krav

Följande användarfall utgör en enkel kravspecifikation för laddningsstationen.

2.1. Användarfall - Standardladdning

En vanlig laddning går till enligt följande:

1. Laddningsstationen står i sitt viloläge och väntar på att ett kort ska dras.

2. Kunden drar sitt kort i kortläsaren.
   a) Processorn tar emot kortinformationen och sänder en saldoförfrågan till databasen.
   b) Databasen söker upp rätt konto, om det finns, och returnerar saldo eller meddelar att kortet inte hör till något konto.
   c) Processorn ser till att saldo eller felmeddelande visas på display.
   d) Om inga sedlar laddas in återställs stationen till viloläge ca 20 sekunder efter att kortet dragits.
3. Kunden väljer att sätta in pengar på sitt konto genom att stoppa in sedlar i sedelläsaren.
   a) Sedelläsaren skickar över sedelvalören till processorn, som i sin tur skickar vidare värdet till databasen. I databasen kontrolleras om sedeln ska accepteras. Varje kort får max innehålla en inställbar summa.
   b) Om denna summa ej överstigs med laddningen accepteras sedeln, databasen uppdateras och en ok-signal skickas till laddningsstationen. Displayen uppdateras med det nya saldot.
   c) Om summan överstigs skickas en icke-ok-signal till laddningsstationen som visar ett felmeddelande på displayen och därefter aktuella summa på kontot. Sedeln spottas ut av sedelläsaren.
   d) Displayen visar efter fullbordat laddning det nya saldot i ca 20 sekunder. Sedan återställs stationen till viloläge.

Om ett nytt kort dras ska det nya kortets laddningscykel påbörjas. Tanken är att endast 20-, 50- och 100-lappar ska kunna användas i laddningsstationen.

2.2. Grafiskt gränssnitt

Det grafiska gränssnittet utgörs av meddelanden på displayen. De meddelanden som kan förekomma återfinns i *Tabell 1*.

Tabell 1. *Möjliga meddelanden på display*

Meddelande	Förklaring
Dra kortet	Används då laddningsstationen står i viloläge
Konto finns ej	Används då det dragna kortet ej är registrerat i databasen
Felaktigt kort	Används då kortavläsningen blir felaktig
Databasfel	Laddningsstationen kan ej kommunicera med databasen
Sedel ej giltig	Felaktig sedel
FEL: Saldo > XXX kr	Används då en laddning med den inmatade sedeln skulle överstiga det maximala tillåtna beloppet på kontot.
Saldo XXX.YY kr	Saldot på det aktuella kortet

3. Hårdvarulösning

Till hårdvarulösningen hör val av komponenter och design av kretsschema. Det sistnämnda återfinns i Bilaga 1.

Den centralaste komponenten är processorn, utifrån den väljs resterande komponenter. Två processorer var aktuella för laddningsstationen, HC11 och MC68008 (just därför att de används på kursen i fråga). Valet föll på HC11 då den har mer inbyggt i form av minnen och kommunikationsenheter än 68008 och på så vis är enklare att arbeta. Den något högre prestanda som fås med 68008 är inget som behövs i detta projekt.

3.1. Periferienheter

Förutom processorns består konstruktionen av ett antal periferienheter, vilka presenteras nedan tillsammans med beskrivning över dess funktioner.

3.1.1. Multiplexer

För att kunna kommunicera med alla enheter behövdes en multiplexer. Vi valde en latch, 74HC373 från Texas Instrument, denne fanns redan i institutionens lager. Latchen fungerar som en multiplexer om man sätter `OE´` (Output Enable) låg och `LE` (Latch Enable) hög, vilket kan ses i *Tabell 2*. Se vidare [1] för mer information. `OE´` och `LE` är kopplade till processorns port B, D (databitarna) kommer från dels kortläsaren och dels från sedelläsaren. Utgångarna på kretsen går via databuss C till processorns port C.

Tabell 2. *Signaler till och från multiplexern*

Ingång			Utgång
OE´	LE	D	Q
L	H	H	H
L	H	L	L
L	L	X	Qo
H	X	X	Z

3.1.2. Kortläsare

Grundidén är en kortläsare för magnetkort. Dessa kort är vanliga och är mer eller mindre i var mans ägo. Tanken är att de som blir kunder själva väljer ett eget kort, ex. studentkort eller bankomatkort, som sedan registreras och används i mojtsystemet.

Det finns olika standarder för magnetkort. I en av de vanligare, som bl.a. bankkort använder, har korten tre spår. Det nummer som ofta syns skrivet på korten är spår två, detta vill vi använda.

Kravet på kortläsaren är därför att den måste klara av att läsa spår två. Vi har valt en läsare som klarar både spår två och tre. Se Bilaga 6. Denna kommunicerar med processorn via tre signaler, se *Tabell 3* där även signalen `CSV'` som vi inte använder finns med.

Tabell 3. *Signaler till och från kortläsaren.*

Namn		Typ	Funktion	Koppling
`CLS'`	`Card Loading Signal`	Ut	Hög vid normalläge, låg då kort dras genom läsaren.	Till avbrottspinne på processorn
`RCP'`	`Read Clock Pulse`	Ut	Klockpuls genereras då kort dras. Giltig data finns på `RDP'` vid negativ klockflank.	Till avbrottspinne på processorn.
`RDP'`	`Read Data Pulse`	Ut	Databit. H=’0’, L=’1’.	Till standardingång på processorn.
`CSV'`	`Current Saver`	In	Hög motsvarar arbetsläge, låg viloläge.	+5 V (vi använder oss inte av viloläget)

Dataströmmen

Tecknen från kortläsaren består av 5 bitar, `[P, b4, b3, b2, b1]` vilka sänds från kortläsaren i omvänd ordning (med början på `b1`), se Bilaga 5. Bitströmmen startar med ett starttecken och avslutas med antingen ett separator- eller stopptecken beroende på om där finns ett tredje spår eller ej på kortet.

3.1.3. Sedelläsare

För att sedelläsaren ska acceptera en viss sedel måste man först ”lära” den vilka sedlar som är korrekta. Då sedelläsaren känner igen en viss sedel sänder den en puls på den kanalen som är associerad med just den typen av sedel. Vi har valt att jobba med 20-, 50- och 100-lappar och dessa har vi placerat på kanal 1,2 och 3. För en översikt över signalerna som behövs vid kommunikation med sedeläsaren, se *Tabell 4* nedan. Då databasen ska ha möjlighet att välja om sedeln ska accepteras eller inte så behövs en signal för detta, denna kallas Escrow. Det fungerar som så att om Escrow är låg och en sedel matas in i läsaren, sänds en signal på den associerade kanalen till processorn, därefter har man 5 sekunder på sig att acceptera sedeln innan sedeln automatiskt returneras. Sedeln accepteras genom att sätta Escrow hög och som kvittens på att sedeln är accepterad sänder läsaren tillbaka en signal på den associerade kanalen.

Tabell 4. *Signaler till och från sedelläsaren*

Beteckning		Typ	Funktion	Koppling
NACHx´	Note Accepted Channel x X = [1..4]	Ut	En 100 ms lång låg puls ges då en sedel accepteras på kanal x.	Till processorns port C via multiplexern
ENCHx´	Enable Channel x X = [1..4]	In	0 V - sedel på kanal x ska accepteras	Från en switch på kretskortet
Escrow		In	0 V ger möjlighet att då en sedel kommer välja om den ska accepteras eller inte. 5 V då accepteras alla korrekta sedlar.	Från processorns port B
Alarm´		Ut	En 100 ms lång låg puls sänds om motorerna till inmatningen av sedel fastnar.	Till processorns port C via multiplexern
+v COM		In	Sätter den spänning med vilken sedelläsaren kommunicerar med omvärlden (5-12 V.)	Direkt satt till +5V

Sedelläsarens motorer kräver +12 V, det är enbart därför som 12 V krävs på kortet.

3.1.4. Display

Displayen används för att ge användaren kort och koncis återkoppling under bruk av laddningsstationen. Främst är denne intresserad av saldot och att det uppdateras korrekt då pengar matas in. Vi har valt en tvåradig punktmatrisdisplay vilken fyller dessa krav till fullo. Se Bilaga 8.

Displayen kommuniceras med via skrivning och läsning till dess register på 8 bitar. Den har en färdig teckenuppsättning inbyggd vilket gör hanteringen enkel, man skriver ascii-koden för tecknet till displayens dataregister. Den har även ett instruktionsregister, via skrivning till detta hanteras ex. rensning av displayen och flyttning av markör. Vissa skrivningar tar lång tid, särskilt till instruktionsregistret, genom läsning av displayens busy-bit vet man om det är OK att skriva eller inte.

I *Tabell 5* finns displayens signaler beskrivna.

Tabell 5. *Signaler till och från displayen*

Beteckning		Typ	Funktion	Koppling
R/W	Read/Write	In	Indikerar läsning från eller skrivning till displayen	Från processorns port B
E	Enable	In	Används vid läsning och skrivning	Från processorns port B
RS	Register Select	In	Väljer instruktions eller dataregistret	Från processorns port B
D[7-0]	Data[7-0]	I/U	Databitar	Till processorns port C via multiplexern

3.1.5. Databas

Databasen är redan implementerad i mSQL under ett tidigare projekt [3]. Kommunikationen från processorn till databasen sker i princip enligt standarden RS-232. Detta innebär att en sladd behövs till sändning och en till mottagning. De resterande bitarna av signalhantering är inte implementerat alls, eftersom de inte behövs. Själv kommunikationen sker genom ett utbyte av strängar, formen på strängen kan ses i *Tabell 6*. Operationsnumret är den variabel som talar om vad som ska göras med det aktuella kontot. En översikt över operationskoderna finns i *Tabell 7*. Operationkoderna ska skickas som ASCII-tecken enligt samma tabell. Alla operationskoderna används dock inte i vårt projekt.

Tabell 6. *Standardiserad dataöverföring, datapaket med längden 26 byte.*

Start flagga	Mojtnr	Kortnr	Operations-kod	Data	CRC	Skräp-byte	Stopp-byte
1 byte	1 byte	16 byte	1 byte	4 byte	1 byte	1 byte	1 byte

Tabell 7. *Operationskoder*

Operationskod	Skickad ASCII-kod	Uppgift
1	0x31	Begär saldo
2	0x32	Svar saldo
3	0x33	Köp
4	0x34	Svar köp
5	0x35	Ej tillräckligt saldo
6	0x36	Ogiltigt kort
10	0x40	Insättning
11	0x41	Svar saldot för stort
12	0x42	Misslyckad laddning

Spänningen ut från processorn ligger på TTL-nivåer, dvs +5 V. För att följa RS-232 standarden ska spänningen vara ±9 V ut på sladden. Detta klarades av genom att använda en extern drivkrets, MAX232 från MAXIM [2]. Denna matas med 5 V och klarar av att ge ±9 V.

För en översikt över signalerna till och från drivaren, se *Tabell 8*.

Tabell 8. *Signaler till och från drivare*

Beteckning		Funktion	Koppling
R1	Receive	Tar emot paket från omvärlden och skickar vidare till processorn	Till processorns port D
T1	Transmit	Skickar vidare paket från processorns ut till omvärlden	Från processorns port D

4. Mjukvarulösning

Mjukvarulösningen kan delas in i två delar. Dels innefattar den lågnivådesign för kommunikation med kortets periferienheter, alltså drivrutiner, och dels design av huvudprogrammet som handhar kortladdning, vi kallar det i texten för högnivådesign.

4.1. Lågnivådesign

Lågnivådesignen innebär drivrutiner för kommunikation med kortets periferienheter såsom display och sedelläsare. Genom att lägga denna kommunikation i drivrutiner förenklas högnivådesignen. I drivrutinerna ligger all maskinnära kod, dvs. registerinställningar, läsning och skrivning på portar m.m.

Alla perifiera enheter på kortet anropas genom drivrutiner, vilket förenklar högnivådesignen eftersom hänsyn inte behöver tas till detaljer på lågnivå.

Specifikationer

- Kortläsaren är kopplad till en avbrottspinne på processorn eftersom läsningsprocessen är tidskritisk, datan kommer samtidigt som ett kort dras och kan ske när som helst. När avbrott kommer lagras databitarna från kortläsaren.
- Displayen är kopplad till bussen på port C. Varje skrivning inleds med en läsning av busy-biten i displayen, är denna satt läses på nytt ända tills den inte längre är satt. Då sker skrivningen.
- Sedelläsaren avläses via polling. Detta går bra då vi vet när i programflödet användaren förväntas mata in en sedel. Skulle en sedel matas in vid annat tillfälle gör det inget, sedelläsaren sväljer nämligen inga sedlar om den inte får en ack-signal.
- Lågnivådesign för kommunikation med databasen är inte implementerad.

Lågnivådesignen återfinns i Bilaga 3.

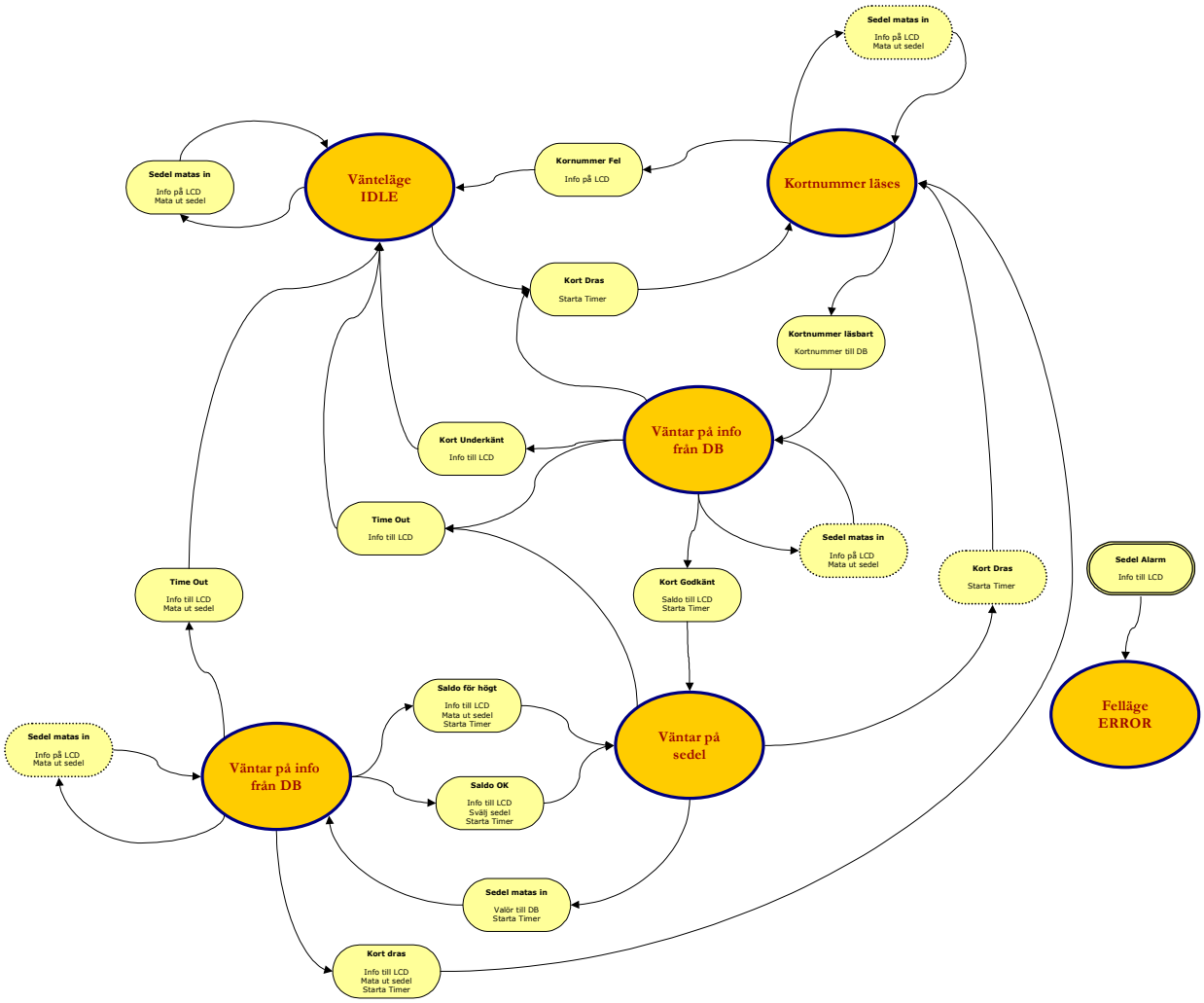
4.2. Högnivådesign

Högnivådesignen beskrivs enklast med ett flödeschema, se *Figur 2*. Tanken är att en oändlig loop ligger och kör i IDLE-läge fram tills att ett kort dras. Sedan drar laddningsprocessen igång. Jämför med användarfallet i kapitel 2.

Specifikationer

- Kort läses alltid då det dras, en flagga indikerar att ett nytt kort är inläst.
- Sedelläsaren har en alarmsignal, utlöses exempelvis då sedel fastnat, kopplad till avbrott. Detta försätter programmet i ett felläge och en ’RESET’ måste göras.
- I övrigt skall bl.a. utskrifter på display m.m. tas upp här. Då högnivådesignen ännu inte är implementerad är detta dock inte fastställt.

Listning av programkod finns i Bilaga 4.



Figur 2. *Flödesschema för huvudprogrammet*

5. Genomförande

Vi började med att skriva en kravspecifikation på vad vårt bygge skulle klara av. Därefter studerades vilka komponenter som skulle ingå. Både sedelläsaren och kortläsaren är komponenterna som E-sektionen har haft liggande som reservdelar. Databasen är redan implementerad.i ett tidigare projekt[3]. LCD-skärmen var typ som institutionen tillhandahöll och den passade projektet utmärkt.

Därefter bestämdes vilka ben på processorn som skulle utnyttjas till vad, samtidigt ritades ett kretsschema, se Bilaga 1. Vi placerade ut komponenterna på kortet och lödade respektive virade signalvägar.

Parallellt med hårdvarudesignen påbörjades tankarna kring mjukvarulösningen. Ett flödesschema för huvudprogrammet gjordes upp och testrutiner för de olika kretsarna påbörjades. När kortet var klart vidtog testning av pereferienheterna.

5.1. Inkoppling/test av pereferienheter

Nedan följer en beskrivning av inkopplingen av respektive periferienhet.

Kortläsare
Kortläsaren är kopplad till kretskortet via en sladd med kontakt, på så vis är det enkelt att ta loss och flytta på den. Implementationen var inte helt trivial, bl.a. var det svårt att med ett fett lödstift löda fast sladdar på kortläsarens kontaktyta, avsedd för ytmontering.
Till en början testade vi att polla signalen från kortläsaren vilket fungerade bra. Lösningen med avbrott valdes ändock till sist då denna metod kändes något säkrare.
Drivrutinen för kortläsaren fungerar bra.

Sedelläsare
Det visade sig att utgångarna från sedelläsaren, dvs. `NACHx´` och `Alarm´` hade en transistor på utgången. För att få den höga signalen till att vara hög fick vi sätta pull-up motstånd på dessa utgångar.
För att göra vår mjukvara så enkel som möjligt valde vi att sätta fyra switchar på kortet, dessa ställer vilka kanaler som ska vara aktiva i sedelläsaren, dvs. `ENCHx´`. Till switcharna är det kopplat pull-up motstånd.
Ett kort testprogram är skrivet som testar att kommunicera med sedelläsaren fungerar. Då man matar in en sedel i läsaren kommer en puls på den aktuella kanalen, då detekterar processorn att sedeln är på väg, processorn kvitterar med att sätta `Escrow` hög om sedeln ska accepteras. Läsaren sänder ytterligare en puls på aktuell kanal för att berätta att sedeln nu är accepterad.

Display
Displayen var den enklaste enheten att få att fungera. Det tar olika lång tid för displyen att hantera en skrivning beroende på om det är en skrivning till data- eller instruktionsregister samt beroende på vilken operation som skall utföras (själva skrivningen tar alltid lika lång tid). Till en början testades att i drivrutinen vänta den tid respektive operation skulle ta för att slippa läsa från den. Detta var dock omständligt och till sist valdes ändock att läsa displayens busy-bit vilket visade sig vara en enklare och generellare lösning som fungerar betydligt bättre.

Databas
Kontakten för databaskommunikation är färdig på kortet, sladden är byggd men tyvärr har vi inte hunnit koppla in databasen då detta skrivs. Inte heller är det skriven någon kod alls för kommunikationen med databasen.

5.2. Programutveckling
Direkt efter att periferenheterna befunnits fungera är det dags att implementera huvudprogrammet enligt det tänkta flödesschemat (*Figur 2*). Då alla periferienheter ännu ej testats har denna del inte påbörjats.

6. Resultat
Resultatet av projektet är ett fungerande kretskort med kortläsare, display och sedelläsare. Periferienheterna är testade och har befunnits fungera, så när som på databaskommunikationen som är inkopplad men otestad.

Huvudprogrammet är dock ännu ej implementerat.

7. Utvärderande diskussion
I detta kapitel diskuteras projektet, hur arbetet förflutit och vad som är kvar att göra.

7.1. Kvar att göra

För att färdigställa laddningsstationen krävs en del ytterliggare arbete, vilket utvecklarna ämnar göra. Det återstående arbetet kan spaltas upp enligt:

- Testning av databaskommunikation och färdigställande av dess drivrutin
- Implementation av högnivådesignen
- Testning
- Packetering i låda och infogning med övriga kortbetalningssytemet

7.2. Slutligen

Arbetet med projektet har flutit på bra. Dock uppstod vissa förseningar pga. av yttre omständigheter varför projektet vid denna rapports färdigställande som beskrivits ännu ej fungerar fullt ut.

Utvecklingsmiljön [5] har varit bra och fullt tillräcklig, dock sinkades arbetet av att funktionen 'CALLS', som har som bieffekt att processorn kör betydligt långsammare, var inkopplad.

Till sist kan nämnas att komponentvalet varit lyckat och processorn håller den prestanda som behövs.

Referenslista

[1] Datablad 74HC373, *Texas Instrument.* 1997.

[2] Datablad HC11, Digitala Projekt. *Institutionen för informationsteknologi, Lunds Tekniska Högskola.*

[3] M Andersson, H Ardö, E Lundgren. Kortbetalningssystem för Edekvatamojter. *Institutionen för informationsteknologi, Lunds Tekniska Högskola,* 1999.

[4] Datablad display SHARP Dot-Matrix LCD units:
`http://www.it.lth.se/datablad/display/LCD.pdf`

[5] C-spy. Utvecklings- och debuggningsmiljö för C.

Appendix

I detta appendix finns de bilagor som ingår i rapporten.

Bilaga 1. Kretsschema

Här finns det slutgiltiga kretsschemat.

Bilaga 2. Processorportarna – Översikt

Följande schema beskriver hur processorpinnarna är kopplade:

Pinne	Namn	In/Ut	Kommentar
1	Vss (gnd)	In	Jord
2	MODB	In	Enligt schema
3	MODA	In	Enligt schema
4	STRA/AS	In	Hög
5	E	Un	
6	STRB/R/W	Un	
7	EXTAL	In	Klocka in, enligt schema
8	XTAL	In	Klocka in. enligt schema
9	PC0	I/U	Riktning styrs via DDRC
10	PC1	I/U	"
11	PC2	I/U	"
12	PC3	I/U	"
13	PC4	I/U	"
14	PC5	I/U	"
15	PC6	I/U	"
16	PC7	I/U	"
17	RESET	In	Krets kopplad hit
18	XIRQ	In	Sätt hög
19	IRQ	In	Sätt hög
20	RXD	I/U	PD0
21	TXD	I/U	PD1
22	PD2	I/U	Styrs via DDRD
23	PD3	I/U	"
24	PD4	I/U	"
25	PD5	I/U	"'
26	Vdd	In	+5V
27	PA7	I/U	RDP'
28	PA6	Ut	
29	PA5	Ut	
30	PA4	Ut	
31	PA3	I/U	Vi använder som U, riktning via DDRA3
32	PA2/IC1	In	Hög
33	PA1/IC2	In	RCP
34	PA0/IC3	In	CLS
35	PB7	Ut	
36	PB6	Ut	
37	PB5	Ut	
38	PB4	Ut	
39	PB3	Ut	
40	PB2	Ut	
41	PB1	Ut	
42	PB2	Ut	
43	PE0	I/U	Hög, För AD-omvandling
44	PE4	I/U	Hög, För AD-omvandling
45	PE1	I/U	Hög, För AD-omvandling
46	PE5	I/U	Hög, För AD-omvandling
47	PE2	I/U	Hög, För AD-omvandling

48	PE6	I/U	Hög, För AD-omvandling
49	PE3	I/U	Hög, För AD-omvandling
50	PE7	I/U	Hög, För AD-omvandling
51	VRL	In	0V, för AD/DA-omvandling
52	VRH	In	3V högre än VRL, för AD/DA-omvandling. Här: 0V

Bilaga 3. Programlistning – lågnivådesign

Nedan följer programlistning av lågnivådesignen, alltså drivrutiner. De är representerade som en testfil, där kretskortets periferienheter testas.

```
//*********************************
//*    ALL                        *
//*********************************
/*

Testprogram för samtliga enheter på processorn:


* Konstanter:
PORTC  equ     $1003    Knapparnas adress.
PORTB  equ     $1004    Displayens adress.
PACTL  equ     $1026    Styr bl.a. riktningen hos PA3.

       org     $F800    Anger var programmet ska ligga.
* Initieringar:
Start  LDAA    #%00001000    Bit 3 i port A
       STAA    PACTL    ska vara utgång.
    */


//*********************************
//*     C-version                 *
//*********************************
/*  Konstantdeklarationer:        */
#include "io6811.h"
//#include "int6811.h"
#include "vectors.h"
#include "intr6811.h"

/*  Globala variabeldeklarationer här        */

char sek, csek, min;
int tstart, tstopp;
char valor;

//För kortläsning:
char ch,a,pos,g,bitpos,bytepos,p;
char kortnr[16], kortOK;

/*Initiering*/
void initL1(void)
{
 kortOK = 0;


 DDRC = 0xFF; /*PC7-PC4=ut, PC3-PC0=ut*/
 DDRD = 0xFF; /*PD7-PD0=ut*/
 PACTL= 0x08; //PA7 = Inport, PA3 = Utport, Sid 6-2.
 PORTB = PORTB & 0x7F; //Bit 7 låg, sedlar accepteras ej.

 //Avbrott
 TCTL2 = 0x0a; //Sid 9-5   -flank
 TMSK1 = 0x13; //Sid 9-10  -vilka avbrott
 TFLG1 = 0xff; //Sid 9-11  -kvitterar alla avbrott

 /*
 TCTL1 rätt från början Sid 9-10.

  sid 9-3 -> 1 KC = 500ns (16 bitarsregister ger 32ms = OF),  Vi vill ha 10ms, dvs 20000KC (a 500ns).
  Alltså, interrups var 20000 KC.
 */


 enable_interrupt();


}

/*Avbrottsrutiner:*/

//Tidsavbrott
interrupt void OC4_interrupt(void) {
  csek++;
  if (csek > 99 ){
    sek++;
    csek -= 100;
    if (sek > 59){
      min++;
      sek -= 60;
```

```
      if (min > 59) min -= 60;
    }
  }
  TOC4 += 20000; //avbrott var 10:e ms.
  TFLG1 = 0x10;
}

//RCP går låg, kort dras.
interrupt void IC3_interrupt(void) {
  //Kort börjar dras, nollställ alla kortvariabler (kortnr etc)
  int x;
  kortOK=0;
  g=0;
  for (x=0; x<17; x++ )  kortnr[x] = 0x00;  //Rensa kortvariabel
  sek = 0; //Nollställ timer

  /*Kortläsningsvariabler*/
  pos=0;
  ch=0; g=0;
  bytepos=0;
  bitpos=0;
  kortOK=0;

  TFLG1 = 0x01;
}

//Klockpuls från kort
interrupt void IC2_interrupt(void) {
  //Kortläsarens klockpuls går låg, lagra tecken
  static char CRC;
  a = !(PORTA&0x80);  //a = RDP
  ch = ((ch & 0x0F) << 1) | a;
  if (g) {
    p += a;
    bitpos++;
    if (bitpos == 5) {
      if (ch == 0x1F)
        kortOK = 1;   //Stop char
      if (ch == 0x16)
        kortOK = 1;   //Separator char
      if ((p&0x01) == 0)
        CRC = 1;   //Bad CRC

      p=0;
      if (bytepos > 15) bytepos = 0;

      if (!kortOK & !CRC){
        kortnr[bytepos] = ( ((ch&0x02)<<2) | (ch&0x04) |
                            ((ch&0x08)>>2) | ((ch&0x10)>>4) )+0x30;
      }
      bytepos++;
      bitpos = 0;
     }
  } else {
     if (ch == 0x1A){ /* Start char */
        g = 1;
        CRC = 0;
     }
  }

  //När kortet är färdigläst skall 'rätt' rutin anropas.
  //if (ch == 0x1F || ch == 0x16) return 1; else return 0;

  TFLG1 = 0x02;
}



/*Lågnivårutin för skrivning på LCD
   -Write LCD
 */
char wlcd (char sign, int RS)
{
  char busy;
  busy = 1;

  // 1) Se till att 74HC373 inte driver C-bussen. (OE' = H)

  PORTB |= 0x20;

  // 2) Läs busy-biten från LCD tills dess att den är OK.

  DDRC = 0x00;  //Läsning på C-porten
  while(busy){
     //R/W=1, E=1, RS=0. ---1 10--
     PORTB |= 0x18;
     PORTB &= 0xfb;
```

```
     //Då E varit hög ett tag har vi giltig data på C-porten, vi läser busy-biten (nr 7)
     busy = (PORTC & 0x80);
     PORTB &=0xef;  //E låg:  ---0 ----
  }

  //Skrivning till C-porten:
  DDRC = 0xFF; /*PC6-PC0=ut (11111111) */

  /*Styrsignaler:
    OE (PB5)=1, E(PB4)=0, R/W(PB3)=0
    1) RS(PB2)=0: PB=XX1000--, |=00100000, &= 11100011
    2) RS(PB2)=1: PB=XX1001--, |=00100100, &= 11100111
  */
  if (!RS){
    PORTB &= 0xE3;
    PORTB |= 0x20; //140ns
  }else{
    PORTB &= 0xE7;
    PORTB |= 0x24; //140ns
  }

  /*Sätt E(PB4) hög
    PB4 |= 00010000
  */
  PORTB |= 0x10; //25ns

  /*Data på buss C*/
  PORTC = sign; //450ns

  /*Sätt E(PB4) låg
    PB4 &= 11101111
  */
  PORTB &= 0xEF; //10ns

  /*Nu skall tecken finnas på display, måste gå tid innan E kan gå hög igen.*/

  //450ns

  return 1;
}

/* Print LCD*/
void prlcd(char string []){
 int x = 0;

 while (string[x] != 0){

   wlcd(string[x],1);
   x++;
 }

}


void initLCD(void){
  wlcd(0x38,0); //Function set, C=001110XX
  wlcd(0x0F,0); //On/Off-control, C=00001111

  wlcd(0x01,0); //Rensa display
  wlcd(0x02,0); //Flytta markör hem
}

/*Acceptera sedel
  Lägger retursignal (Escrow) hög till sedelläsaren, väntar
  tills den bekräftat.
*/
void accept(void){

  PORTB |= 0x80;
  while (PORTC & 0x1E == 0x1E);

  PORTB &= 0x7F;
}

char sedelpoll(void){
  char sedel;
  char PC;

  DDRC = 0x80;          /*PC6-PC5=in, PC7, PC4-PC0=in (1000 0000)   */
  /*Sätt riktningen på 74HC373:
     1) Se till att LCD:n INTE driver
        Bit 3 & 4 låga: 1110 0111 = 0xE7 (E = L, R/W = L)
 */

  PORTB = PORTB & 0xE7;
  PORTB = PORTB & 0xDF;
  PORTB = PORTB | 0x40;
  PORTB = PORTB & 0x7F; //Bit 7 låg, sedlar accepteras ej.
```

```
  sedel = 0;
  sek = 0;

  while (sek<30){
    PC = PORTC;
    if (!((PC&0x1E) == 0x1E)){ //Alla är inte höga, någon är låg!
      wlcd(0x01,0); //Rensa display
      wlcd(0x02,0); //Flytta markör hem
      //prlcd(&PC);
      wlcd(PC+0x30,1); //Flytta markör hem
      sek=0; while(sek<2); //Vänta 2 sek
      switch ((~PC & 0x1E)){

        case 0x02: { //Kanal 1
          sedel=1;
          break;
        }
        case 0x04: { //Kanal 2
          sedel=2;
          break;
        }
        case 0x08: { //Kanal 3
          sedel=3;
          break;
        }
        case 0x10: { //Kanal 4
          sedel=4;
          break;
        }
      }//switch
      break;
    }//if
  }//while
  return sedel;
}

main(void){
/*  Lokala variabeldeklarationer här        */
 int x;
// char sedel;

//Initiering
  initL1();
  initLCD(); /*LCD-init*/

/*---------------------------------------------
     LCD-test
  ---------------------------------------------*/

  wlcd(0x43,1); //'C'
  wlcd(0x26,1); //'&'
  wlcd(0x41,1); //'A'
  wlcd('f',1);  //'f'

  wlcd(0x01,0); //Rensa display
  wlcd(0x02,0); //Flytta markör hem

  prlcd("LCD-test OK");
  wlcd(0x1C,0); //Display shift, right

  sek=0; while(sek<5); //Vänta 4 sek

/*---------------------------------------------
     Kortläsare
  ---------------------------------------------*/

  wlcd(0x01,0); //Rensa display
  wlcd(0x02,0); //Flytta markör hem
  prlcd("Dra kort!");

 while (1){
     if (kortOK){
       wlcd(0x01,0); //Rensa display
       wlcd(0x02,0); //Flytta markör hem
       for(x=0; x<16; x++) wlcd(kortnr[x],1); //prlcd(kortnr[x]);
       prlcd("Kort OK");
       kortOK=0;
       sek=0; while(sek<2); //Vänta 2 sek


     /*---------------------------------------------
          Sedelläsare
       ---------------------------------------------*/

       wlcd(0x01,0); //Rensa display
       wlcd(0x02,0); //Flytta markör hem
       prlcd("Mata in sedel!");
```

```
        valor = sedelpoll();
        if (valor == 2){
          accept();


          wlcd(0x01,0); //Rensa display
          wlcd(0x02,0); //Flytta markör hem
          prlcd("20 kronor, -tackar!");
          sek=0; while(sek<2); //Vänta 2 sek

          //sek = 0;
          //skicka valör till DB
        }else{
          wlcd(0x01,0); //Rensa display
          wlcd(0x02,0); //Flytta markör hem
          prlcd("Ingen sedel acc.");
          sek=0; while(sek<2); //Vänta 2 sek
        }
        wlcd(0x01,0); //Rensa display
        wlcd(0x02,0); //Flytta markör hem
        prlcd("Dra kort");
      }
    }
}
```

Bilaga 4. Programlistning – högnivådesign

Då högnivådesignen inte är implementerad lämnas denna bilaga tom.

Bilaga 5. Magnetkortsstandard

Bilaga 6. Datablad kortläsare

Bilaga 7. Datablad sedelläsare

Bilaga 8. Datablad display

Nedan följer ett utdrag ur databladet för displayen, för fullständigt datablad se [4].