# Övervakningskamera
# Digitala projekt 2006

Tove Henriksson E-02
Fredrik Lindell E-02

Handledare: Bertil Lindvall

24 maj 2006


LUNDS UNIVERSITET
Lunds Tekniska Högskola

# Abstract

Camera supervision is especially important in the society of today when assault and robbery are crimes constantly increasing. Nowadays even public places like buses are sometimes supervised to make the inhabitants feel safe. There are different needs and requirements on the systems depending on their purpose.

We have built a camera that can be used to supervise a corridor or room. It is a black and white camera, which takes a picture of the room and uses it as a reference picture. After that it continuously takes pictures and compare them to the reference. If there is a change in the picture it means someone has gone in to the room and a warning signal is generated.

We have used a CCD camera connected to a digital construction and a diode, which is lit when a change in the picture is discovered.

# Innehåll

# 1 Inledning

I dagens samhälle där våld, överfall och rån alltför ofta inträffar är behovet av olika övervakningssystem stort. Banker, affärer och liknande har sedan länge haft övervakningskameror, men på senare år har även offentliga platser och t.o.m. bussar börjat övervakas för att invånarna ska känna sig trygga. Behoven och kraven på systemen är väldigt olika beroende på vad de ska användas till.

Syftet med detta projekt är att bygga en övervakningskamera som ska kunna användas för att övervaka en korridor. Till detta används en svart-vit CCD-kamera. Den ska ta en bild och använda som referensbild och sedan kontinuerligt ta nya bilder och jämföra med referensbilden. Om förändring skett i bilden, d.v.s. någon tagit sig in i korridoren, ska en larmsignal genereras. Som larmsignal har vi för enkelhetens skull valt att tända en lysdiod.

Kapitel 2, Komponenter, beskriver vilka komponenter som valts samt deras funktioner. I kapitel 3, Metod, beskrivs sedan arbetsgången, allt från ritande av kretsschema till programmering. Sedan följer Resultat i kapitel 4 och en avslutande diskussion i kapitel 5. Kretsscheman, programkod och bilder går att finna i appendix A-C.

# 2 Komponenter

Komponenterna valdes utifrån olika prestandakrav; snabbhet, storlek på minne o.s.v. Efterhand som bygget pågick upptäcktes behov av nya komponenter, och även överflödiga komponenter som plockades bort. Konstruktionen delades upp i kontrolldel, kameradel och logik för att göra kretsschemat mer överskådligt.

## 2.1 Kontrolldel

Kontrolldelen fungerar i princip som en dator och är uppbyggd kring processorn. Kontroll- och kameradelen är åtskilda med buffertar, dessa behandlas dock under kameradelen för att följa kretsschemat i Powerlogic.

### 2.1.1 Processor

Till processor valdes Motorola 68008 med 48 pinnar. Denna processor har inget inbyggt minne utan externt program- och ramminne krävs. Adressbussen har 20 bitar och databussen 8 bitar, vilket medför att det tillgängliga adressutrymmet är 1 Mbyte. Enheterna vi använder måste separeras av buffertar eller kunna gå i 3-state för att undvika kollision på databussen. 3-state innebär att enheterna blir högimpediva (osynliga), och det får samma effekt som att skilja dem åt med buffertar.

### 2.1.2 Systemklocka

Systemet måste naturligtvis kunna klockas. Till detta ändamål valdes en färdig kristalloscillator, EXO-3 med grundfrekvens 20 MHz. Processorn jobbar med en frekvens på 10 MHz, därför delades grundfrekvensen ner till hälften innan den skickades till processorn. Datablad finns i ELFA-katalogen eller på www.elfa.se.

- EXO-3 20 MHz kristalloscillator

### 2.1.3 Minne

Processorn behöver ett minne att lagra programmet på, till detta är det lämpligt att använda ett EPROM på 8 eller 16 kbyte. För att ha lite marginaler till programmet valdes ett på 16 kbyte. EPROM-et går endast att läsa ifrån, ej skriva till. Variabler och stack lagras på ett RAM-minne. Här valdes ett SRAM på 8 kbyte, eftersom inga särskilda beräkningar kommer göras, utan endast jämförelse av två bilder.

- 27C128 16 kbyte x 8-bit parallell EPROM
- 6264 8 kbyte x 8-bit parallell CMOS SRAM

### 2.1.4 Parallellport

Parallellporten (eller D-latchen) har databussen som insignal och 8 utsignaler. En utsignal används för att tända dioden då förändring i bild upptäckts och en går till logik-kretsen och ger klartecken till kameradelen att ta en ny bild.

- 74HC373 Octal 3-State Non-Inverting D-Latch

### 2.1.5 Reset-knapp

En reset-knapp krävs för att kunna nollställa hela konstruktionen, t.ex. om fel uppstår eller om man vill ta en ny referensbild. Det behövs ingen knapp för att ta en bild eftersom detta görs kontinuerligt i programmet.

## 2.2 Kameradel

### 2.2.1 CCD-kamera

I konstruktionen används en svart-vit CCD-kamera. Eftersom utsignalen från kameran är analog krävs en A/D-omvandlare. Bild på kameran finns i appendix C, bild C.1.

### 2.2.2 A/D-omvandlare

A/D-omvandlaren gör om den analoga signalen till en digital så att den kan sparas i bildminnet och sedan behandlas av processorn. Bilden är väldigt stor och det krävs därför en snabb A/D-omvandlare för att det inte ska ta lång tid.

- TDA8703 8-Bit High Speed A/D Converter

### 2.2.3 Video Sync Separator

Bilderna från kameran är väldigt stora, och det är i denna konstruktion omöjligt att jämföra varje pixel i de två bilderna med varandra. Därför plockas endast några punkter per bild. Det är viktigt att punkterna tas på samma ställe i de två bilderna, annars blir jämförelsen helt felaktig. Separatorn ger ut två signaler, hsync som meddelar att en horisontell pixelrad är klar, och vsync som meddelar att den vertikala delen är klar, d.v.s. att alla rader är klara. Eftersom samma klocka används då bilden tas kommer de att tas från samma ställe såvida man startar samtidigt, och därför används här endast vsync-signalen. Då vsync-signalen är hög håller en bild på att tas. När bilden är färdigtagen går vsync låg, och när den går hög igen börjar nästa bild tas.

### 2.2.4 Bildminne

De A/D-omvandlade bilderna sparas i ett separat bildminne varifrån de kan läsas av processorn och jämföras med varandra. Minnet bör vara snabbt, och tillräckligt stort för att minst två bilder ska få plats. Storleken var svår att uppskatta, för säkerhets skull valde vi ett minne på 512 kbyte. Under arbetets gång när närmare beräkningar gjorts upptäckte vi dock att endast ca 64 kbyte kommer att användas.

- IDT7MP4045 512 kbyte x 8 –bit parallell CMOS SRAM

### 2.2.5 Adressräknare

Datan från A/D-omvandlaren ska sparas på bildminnet. Varje pixel behöver en egen adress och för att de ska hamna på rätt plats behövs räknare som styrs av logikkretsen.

- 74HC590 8-Bit Binary Counter 3-State OUT

### 2.2.6 Oscillator

A/D-omvandlaren behöver en klocka för att sampla bilden med en lämplig frekvens. Vi har räknat fram att bilden blir lagom stor då frekvensen är 1 MHz och valde därför en 16 MHz EXO-3 oscillator, där grundfrekvensen sedan delas ner 16 ggr till 1 MHz.

- EXO-3 16 MHz kristalloscillator

### 2.2.7 Buffertar

Som tidigare nämnts behövs buffertar för att skilja data- och adressbussarna åt mellan kontroll- och kameradel. De ska bara ha kontakt med bildminnet då processorn ska läsa en bild därifrån. All övrig tid ska de vara bortkopplade så att A/D-omvandling kan ske utan kollisioner och störningar. Som databuffert mellan kontrolldelen och bildminnet används en dubbelriktad buffert eftersom det ska gå att skicka data både från bildminnet till processorn och tvärtom. Eftersom 16 adressbitar används behövs två adressbuffertar som dock är enkelriktade

- 74HC245 Octal 3-State Non-Inverting Tranceiver
- 74HC541 Octal 3-State Non-Inverting Buffer

## 2.3 Logik

Alla våra periferikretsar ska kunna gå i 3-state och måste aktiveras av en chip select-signal. Till detta behövs en logikkrets som kan tolka vilken krets det är processorn vill ha kontakt med. Dessutom måste avbrott kunna genereras, t.ex. då en bild är färdigbehandlad, och A/D-omvandlingen ska styras. Det hade blivit väldigt svårt att själv bygga en krets som kan kontrollera alla dessa funktioner. Eftersom konstruktionen innehåller många delar som ska styras valdes en Lattice-krets med 64 pinnar, för att vara säkra på att de skulle räcka till.

- Lattice 1032E High-Density Programmable Logic

# 3 Metod

## 3.1 Hårdvarukonstruktion

### 3.1.1 Kretsschema

Då alla komponenter var valda var det dags att rita kretsschema vilket gjordes i programmet Powerlogic. Databladen till komponenterna studerades noga för att ta reda på hur allt skulle kopplas. Det tog lite tid att lära sig programmet som dessutom visade sig göra som det ville ibland... Det blev dock färdigt till sist, även om det under arbetets gång har upptäckts fel och ändringar som behövde göras, så kretsschemat har kontinuerligt uppdaterats, och kändes utan tvekan som en nödvändighet att ha under bygget. Våra kretsscheman över kontroll-, kamera- och logikdelen finns i appendix A.

### 3.1.2 Bygge

Konstruktionen byggdes på ett laborationskort som var förborrat och företsat. Som ovana hårdvarukonstruktörer var det svårt att veta var komponenterna bäst skulle placeras. Vi utgick från det grundläggande kravet att den digitala och analoga delen ska placeras på varsin sida av kortet för att undvika störningar, och försökte sedan placera komponenter som skulle kopplas ihop så nära varandra som möjligt. Avkopplingar med hjälp av kondensatorer gjordes till varje komponent för att minska risken för oönskade strömmar i jordplanet. Först lödades alla jord- och spänningspunkter fast, och sedan började det omfattande virningsarbetet. Det tog mycket tid att vira alla signaler, särskilt adress- och databuffert, och sedan när vi kom vidare i arbetet var det Lattice-signalerna som var tidsödande eftersom det var lite klurigt att hitta rätt ben. Flera fel samt saker vi inte tänkt på upptäcktes under virningsprocessen vilket också gjorde att det drog ut på tiden. Vi upptäckte också att virning kunde använts i ännu större utsträckning, vilket hade underlättat då fel skulle rättas till eller då ändringar skulle göras.

Bilder på kretskortet finns i appendix C, bild C.2 och C.3.

### 3.1.3 Testning

Periferienheterna testades med hjälp av ett utvecklingssystem för vår processor, 68008, som tagits fram av it-institutionen på LTH, se bild C.4. En komponent i taget testades och fel rättades till. Ett par fel som varken vi eller handledaren kunde förstå tog lång tid att lokalisera och rätta till vilket försenade oss mycket.

## 3.2 Logik

### 3.2.1 Adressering

Då processorn ska få kontakt med en periferienhet måste den adressera rätt enhet, som då ska få en chip select-signal och bli aktiv. Detta sköts av Lattice, en programmerbar logikkrets.

Programmet lagras på EPROM-et, vilket därför läggs på de första minnesplatserna. De andra enheterna processorn behöver komma åt är RAM-minnet, D-latchen och bildminnet. Bildminnet är på 512 kbyte och läggs på den sista halvan av processorns minnesplatser. EPROM-et behöver 16 kbyte och RAM-minnet och D-latchen behöver 8 kbyte vardera. Mellan adressplatserna för RAM-minnet och D-Latchen lämnas ett mellanrum på 8 kbyte för att ha marginaler ifall konstruktionen skulle ändras och mer minne behövas. Storleken på EPROM-et är redan väl tilltagen, och mellan D-Latchen och bildminnet är det tomma utrymmet redan stort så därför lämnades inget ytterligare utrymme till dem. I tabell 1 nedan visas hur adresspinnarna ska vara ställda för att processorn ska komma åt den aktuella enheten.

| Enhet | A19 | A18…A16 | A15 | A14 | A13 | A12…A0 | Adress |
|---|---|---|---|---|---|---|---|
| EPROM | 0 | 0 | 0 | 0 | 0 | - | 0x00000 |
| RAM | 0 | 0 | 0 | 1 | - | - | 0x04000 |
| D-Latch | 0 | 0 | 1 | - | - | - | 0x08000 |
| Bildminne | 1 | - | - | - | - | - | 0x80000 |

**Tabell 1.** För att processorn ska få kontakt med en periferienhet måste den adresseras. Tabellen visar hur processorns adresspinnar ska vara ställda för att få kontakt med de olika enheterna.
”-” = don´t care

### 3.2.2 Avbrott

Processorn kan generera avbrott med tre olika nivåer, nivå 7, 5 och 2. 7 är den högsta avbrottsnivån, den tar längre tid att behandla än de andra och går inte att stänga av. Vi kommer därför använda avbrottsnivå 5. I vår konstruktion kommer avbrott bara genereras vid ett tillfälle, nämligen då en bild är färdigbehandlad och nästa ska tas. Då avbrott genereras vid flera tillfällen är det bra att använda olika nivåer beroende på vilket avbrott som ska prioriteras högst. När ett avbrott genereras skickas två signaler, IPL0/2 och IPL1 som kodar avbrottsnivån.

Det finns två typer av avbrott, vektoriserade avbrott och autovektor avbrott. I denna konstruktion används autovektor, eftersom det finns periferienheter som inte kan svara med en adress till den vektor där avbrottsprogrammet finns. Med autovektor bestäms istället i mjukvaran vart i koden man ska hoppa då ett avbrott med en viss nivå inträffar. En signal, VPA, måste då skickas till processorn för att tala om att det är ett autovektor avbrott.

Processorn har en utgång med tre bitar, FC0, FC1 och FC2, för att berätta vilket tillstånd den befinner sig i. Då alla dessa är höga befinner sig processorn i interrupt acknowledge, vilket innebär att den accepterar att avbrott inträffar. Denna information ger möjlighet att skapa logiska uttryck som genererar rätt avbrott.

### 3.2.3 Tillståndsmaskin

När en bild tas ska den A/D-omvandlas och sen läggas in i minnet, pixel för pixel. Dock inte alla pixlar, bilden samplas med 1 MHz från oscillatorn. Pixlarna ska in på rätt plats i minnet, på den adress som adressräknarna räknar fram. När bilden är färdigbehandlad ska ett avbrott genereras. För att allt detta ska fungera är det många kontrollsignaler som ska skickas vid rätt tidpunkt, och för att få detta att fungera krävs en tillståndsmaskin, vilken implementerades i Lattice/ABEL, se appendix B.1. I bild 1 nedan syns vår tillståndsmaskin och sedan följer en förklaring av vad som händer i de olika tillstånden.

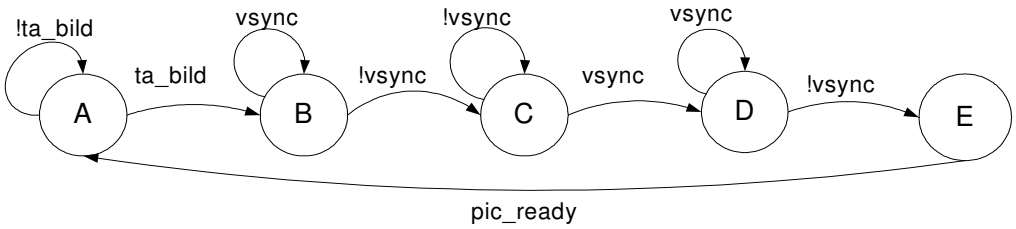



**Bild 1**. Tillståndsmaskinen

**Tillstånd A:**
Står och väntar i detta tillstånd tills processorn ger klartecken att ta bild genom signalen *ta_bild*.

**Tillstånd B:**
Väntar på signalen *vsync* ska gå låg, för att undvika att börja samplingen mitt i en bild.

**Tillstånd C:**
Vet att *vsync* är låg och stannar i detta tillstånd tills den går hög, vilket innebär att en ny bild börjar.

**Tillstånd D:**
Bildens pixlar läggs in i bildminnet.

**Tillstånd E:**
En bild är färdigbehandlad. Signalen *pic_ready* går hög, vilket genererar ett avbrott.

## 3.3 Mjukvarukonstruktion

Vår mjukvara flyttar den första bilden som tas, referensbilden, och lägger den sist i minnet. När den andra bilden är tagen jämför den de två bilderna med varandra, och om förändring har skett skickas en signal till parallellporten.

På grund av okänt hårdvarufel som ej lyckats lokaliseras, kan bilden ej läsas in till minnet. Därför fick ett program konstrueras som simulerar fel i bilden.

# 4 Resultat

Kretskortet är färdigmonterat och alla komponenter på plats utom EPROM-et och processorn. Detta på grund av ett okänt hårdvarufel som gör att bilden inte kan läsas in till minnet, vilket nämnts i kap 3.3. Testningen kunde därför inte slutföras och det är anledningen till att processorn och EPROM-et ej kopplats in.

För att komma vidare fick vi till sist ignorera felet och gå vidare och skriva programkoden, se appendix B.2. Programmet består av två delar, en som kan flytta bilden och en som jämför de två bilderna med varandra och tänder dioden om förändring upptäckts. Programmen fungerar som de ska, men eftersom vi inte kan läsa in en bild i minnet, får jämförelsen simuleras med konstruerade bilder med påhittade skillnader.

På grund av våra många problem räckte inte tiden till för att implementera avbrottsfunktionen. Den skulle dock fungerat som beskrivs i kapitel 3.2.2.

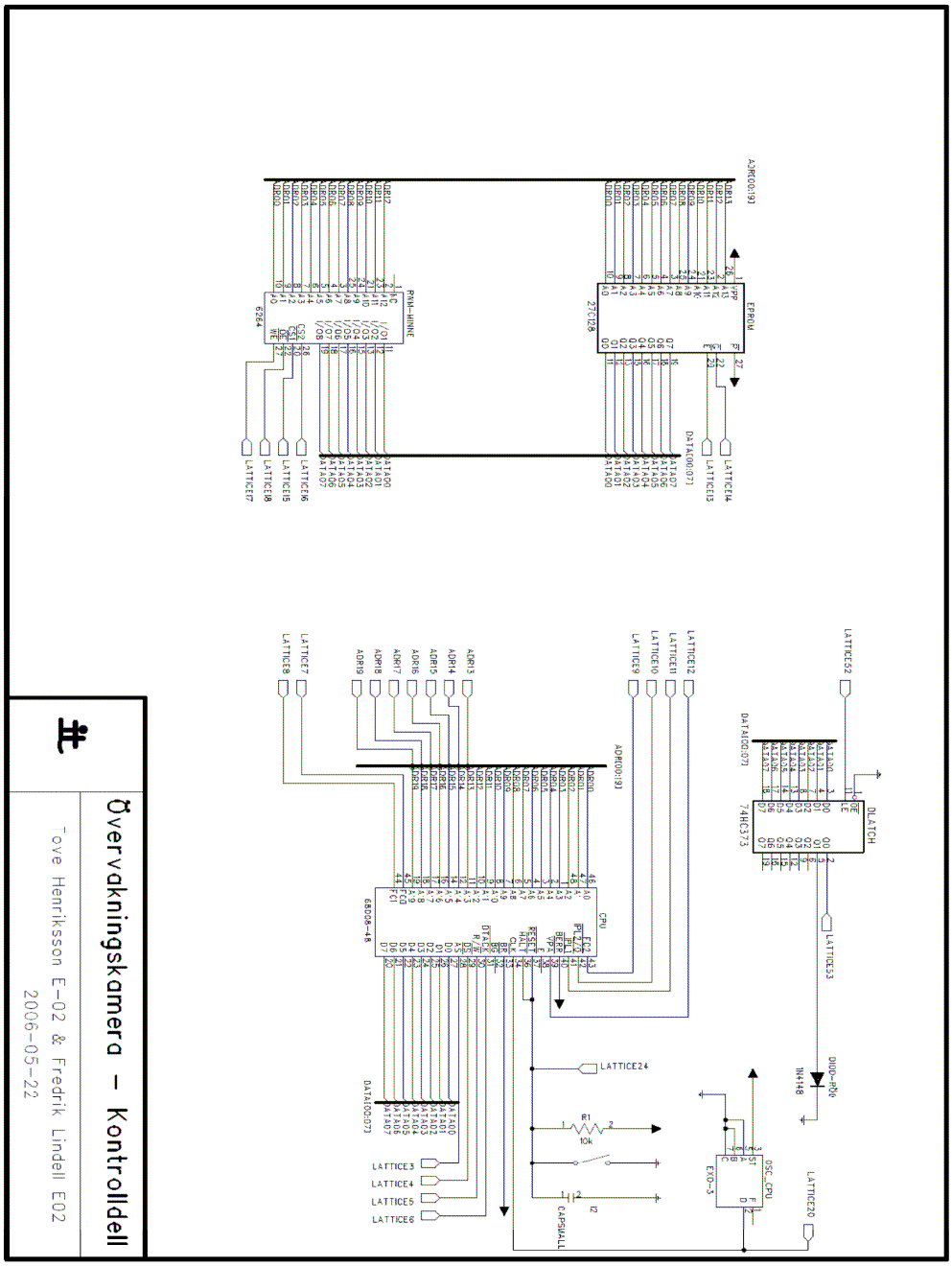
# 5 Avslutning

Digitala projekt är en nyttig kurs, och den har varit intressant att läsa. Dock har vi hela tiden känt att vi saknat tillräckliga förkunskaper, eftersom ingen av oss någonsin gjort något liknande tidigare, och det verkar som om flera av de andra kursdeltagarna sysslat en del med hemmabyggen på fritiden. Efter varje avslutad del har vi i princip fått börja om från noll, och känt oss som levande frågetecken igen. Men å andra sidan så är det precis så det kommer vara och kännas när vi så småningom gör examensarbete, och även senare i arbetslivet. Vi har lärt oss väldigt mycket, både om digital konstruktion och om problemlösning, som vi garanterat kommer ha glädje av i framtiden.

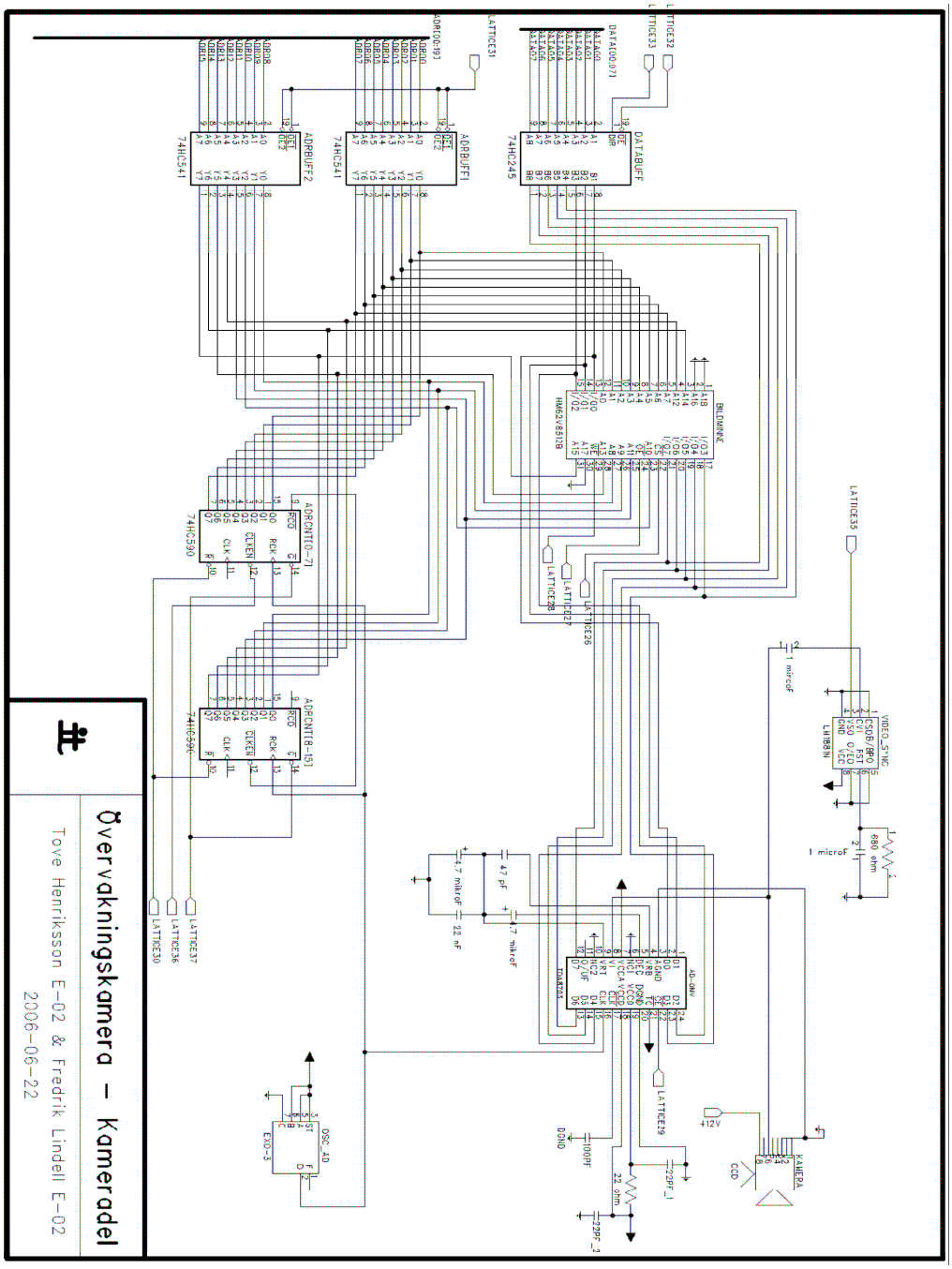
# Appendix

# A Kretsschema

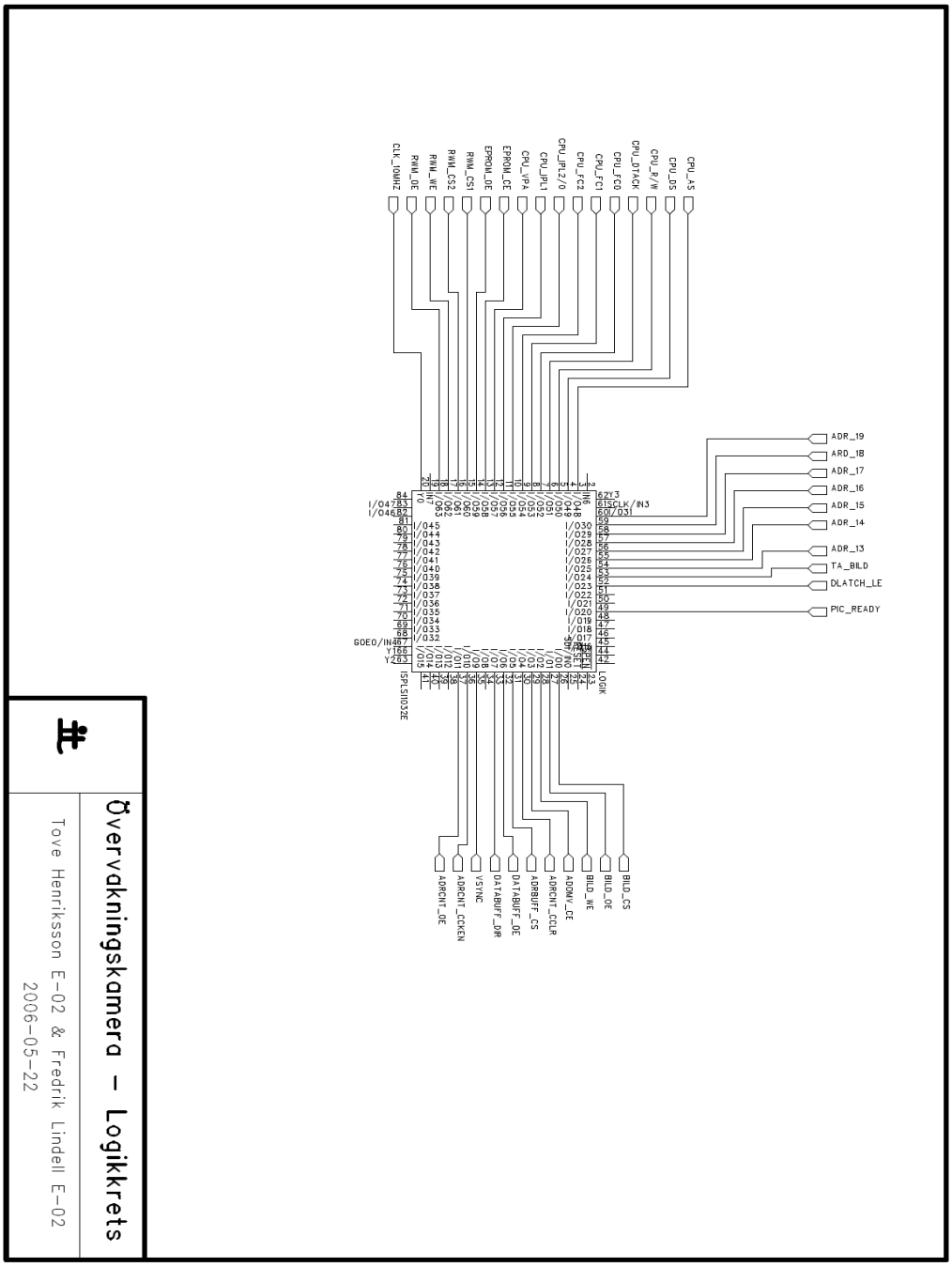
## A.1 Kontrolldel



# A.2 Kameradel



# A.3 Logik


Övervakningskamera – Logikkrets
Tove Henriksson E-02 & Fredrik Lindell E-02
2006-05-22
ISPLSI1032E
LOGIK
CPU_A5
CPU_DS
CPU_R/W
CPU_DTACK
CPU_FC0
CPU_FC1
CPU_FC2
CPU_IPL2/0
CPU_IPL1
CPU_VPA
EPROM_OE
RAM_CS1
RAM_CS2
RAM_WE
RAM_OE
CLK_10MHZ
ADR_19
ARD_18
ADR_17
ADR_16
ADR_15
ADR_14
ADR_13
TA_BILD
DLATCH_LE
PIC_READY
ADRCNT_OE
ADRCNT_CCKEN
VSYNC
DATABUFF_DIR
DATABUFF_OE
ADRBUFF_CS
ADRCNT_CCLR
ADDWV_OE
BILD_WE
BILD_OE
BILD_CS

# B Programkod

## B.1 Lattice-kod

```
MODULE decode

title 'memory decode'

declarations

        A19             pin     60;
        A18             pin     59;
        A17             pin     58;
        A16             pin     57;
        A15             pin     56;
        A14             pin     55;
        A13             pin     54;

        clock           pin     20;
        vsync           pin     35;
        PIC_READY       pin     49;
        CS_BILD_LA      node;
        WE_BILD_LA      node;

        H,L = 1,0;

        CS_EPROM        pin     13;
        OE_EPROM        pin     14;

        CS1_RWM         pin     15;
        CS2_RWM         pin     16;
        WE_RWM          pin     17;
        OE_RWM          pin     18;

        FC0             pin     7;
        FC1             pin     8;
        FC2             pin     9;
        IPL1            pin     10;
        IPL20           pin     11;
        VPA             pin     12;
        AS              pin     3;
        DS              pin     4;
        RW              pin     5;
        DTACK           pin     6;

        LE_DLATCH       pin     52;
        TA_BILD         pin     53; "Signal via D-Lacthen av processorn

        CS_BILD         pin     26;
        OE_BILD         pin     27;
        WE_BILD         pin     28;
```

```
CS_ADRBUFF          pin     31;

CS_DATABUFF         pin     32;
DIR_DATABUFF        pin     33;

CS_ADOMV            pin     29;
OE_ADRCNT           pin     37;
CCLR_ADRCNT         pin     30;
CCKEN_ADRCNT        pin     36;

Address = [A19,A18,A17,A16, A15,A14,A13,.X., .X.,.X.,.X.,.X., .X.,.X.,.X.,.X.,
.X.,.X.,.X.,.X.];


"---------------------- Variabler Tillståndsmaskin -------------------------

q2 node         istype 'reg_d,buffer,keep';
q1 node         istype 'reg_d,buffer,keep';
q0 node         istype 'reg_d,buffer,keep';

statereg = [q2,q1,q0];

"Värden på tillstånd
A=0;
B=1;
C=2;
D=3;
E=4;

"-----------------------------------------------------------------------

equations

"----------------------- Chipselect och Avbrott --------------------------

"Eprom
!CS_EPROM   = A19& !A18 & !A17 & !A16 & !A15 & !A14 & !A13 & !AS;
!OE_EPROM   = !DS & RW;

"Rwm
!CS1_RWM    = !A19 & !A18 & !A17 & !A16 & !A15 & A14 & !A13 & !AS;
CS2_RWM     = H;
!OE_RWM     = !DS & RW;
!WE_RWM     = !DS & !RW;

"Processor
!DTACK              = !CS1_RWM # !CS_EPROM # !CS_BILD #
!CS_ADRBUFF#LE_DLATCH;
!VPA            = FC0 & FC1 & FC2 & !AS;
!IPL1           = L;
!IPL20          = PIC_READY;

"D-Latch
LE_DLATCH   = !A19 & !A18 & !A17 & !A16 & A15 & !DS & !RW;
```

```
"Bildminne
!CS_BILD        = (A19 & !AS) # !CS_BILD_LA;
!OE_BILD        = !DS & RW;
!WE_BILD        = (!DS & !RW) # !WE_BILD_LA;

"Adressbuffert
!CS_ADRBUFF = A19 & !AS;


"Databuffert
!CS_DATABUFF            = A19 & !AS;
DIR_DATABUFF            = !WE_BILD;

statereg.clk = clock;

"------------------------- Tillståndsmaskinen ----------------------------

state_diagram statereg;

State A:

        CS_ADOMV=1;
        OE_ADRCNT=1;
        CS_BILD_LA=1;
        WE_BILD_LA=1;


        if(TA_BILD) then B
        else A;

State B:

        if(!vsync) then C
        else B;

State C:

        CCLR_ADRCNT=0;

        if(vsync) then D
        else C;

State D:

        CCKEN_ADRCNT=0;
        OE_ADRCNT=0;
        CS_ADOMV=0;
        CS_BILD_LA=0;
        WE_BILD_LA=0;
        CCLR_ADRCNT=1;

        if(vsync) then D

        else E;
```

```
State E:
                PIC_READY=1;
                goto A;
        "-------------------------------------------------------------------------------
END
```

## B.2 Mjukvarukod

```
/*************************************************************************
                    test.c
                    ------
  Detta är huvudprogrammet som är skrivet i ANSI C. Exekveringen av hela
  programpaketet börjar i pmain.68k (lage __main).
*************************************************************************/
unsigned short int *par_port;     /* Parallellport (D-Latch) */
unsigned short int *rwm;          /* Ramminne */
unsigned short int *bildminne;    /* Bildminne */

/* main: Huvudprogram */

main()
{
        par_port = (unsigned short int *) 0x8000;
        rwm = (unsigned short int *) 0x4000;
        bildminne = (unsigned short int *) 0x80000;

        memreset();
        genfel();
        compare();                /* Jämför de två bilderna */

        bryt();

        loop: goto loop;
}

/* compare: Jämför två bilder i bildminnet */

compare(){

        unsigned short int *pek1,*pek2;
        unsigned short int sum;

        pek1=0x80000;
        pek2=0x88000;
        sum=0;

        while(pek1<0x88000){
                if(*pek1!=*pek2){
                        sum++;
                }
                pek1++;
                pek2++;
        }

        if(sum!=0){
                *par_port=0x02;          /* Fel detekterat, tänd diod. */
        }
        return(0);
}
```

```
/* movepic: Flyttar referensbilden till nedre delen av bildminnet */

movepic(){
	unsigned short int *pek1,*pek2;

	pek1=0x80000;
	pek2=0x88000;

	while(pek1<0x88000){
		*pek2=*pek1;
		pek1++;
		pek2++;
	}
	return(0);
}

/* memreset: Nollställer minnet samt släcker dioden */

memreset(){
	unsigned short int *pek1,*pek2;

	pek1=0x80000;
	pek2=0x88000;

	while(pek1<0x88000){
		*pek1=0x00;
		*pek2=0x00;
		pek1++;
		pek2++;
	}
	*par_port=0x00;
}

/* genfel: Simulering, genererar ett fel i bilden */

genfel(){
	unsigned short int *pek1,*pek2;

	pek1=0x80000;
	pek2=0x88000;

	*pek1=0x12;
	*pek2=0x67;
}

/* bryt: Brytoperation */

bryt(){
	/* Gör inget, lägger in en brytpunkt på it68-systemet. */
}
```

# C Bilder

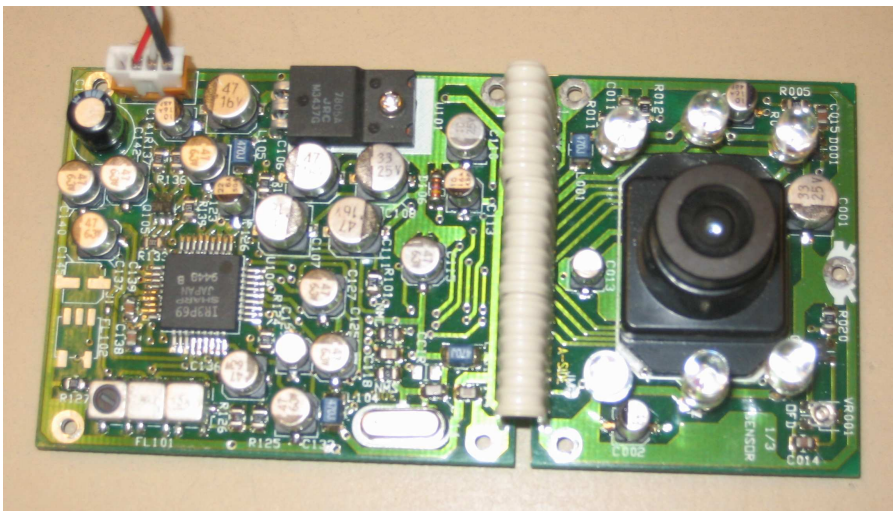


**Bild C.1.** CCD-kameran

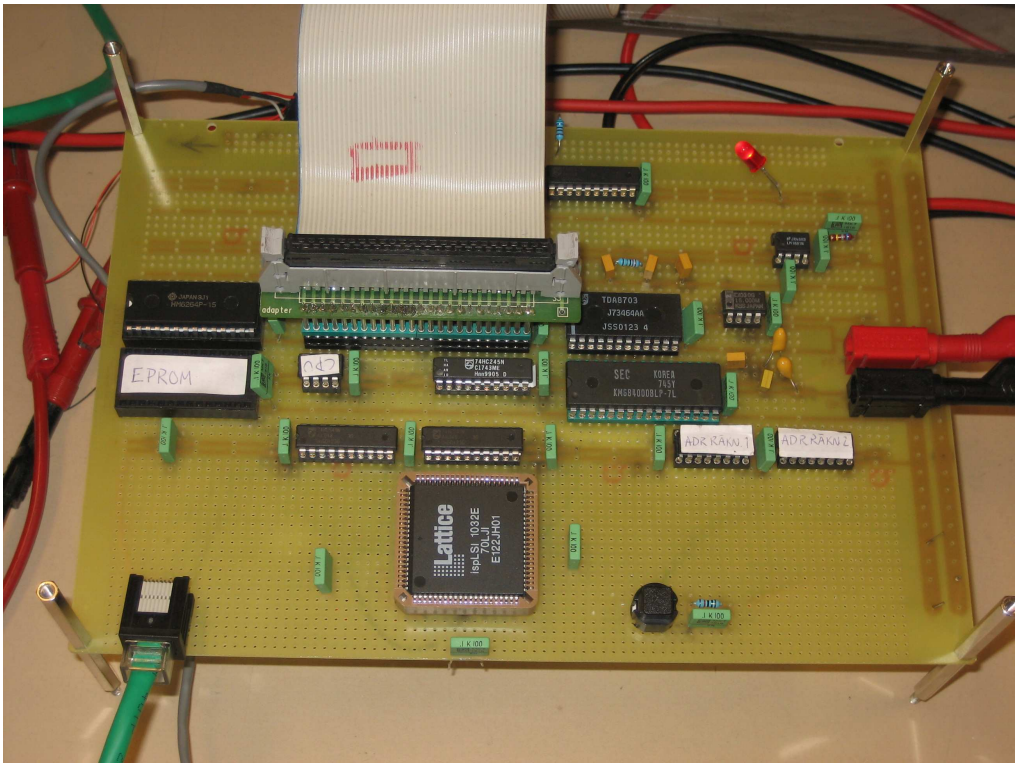



**Bild C.2.** Kretskortet med komponenter

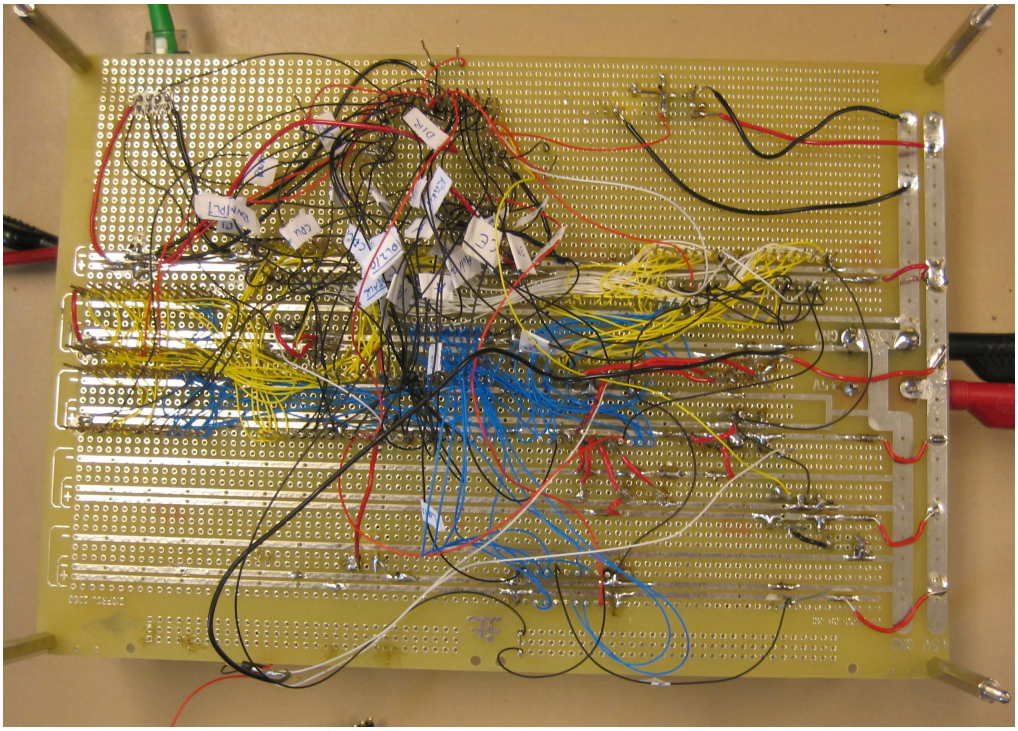


**Bild C.3.** Kretskortet underifrån.

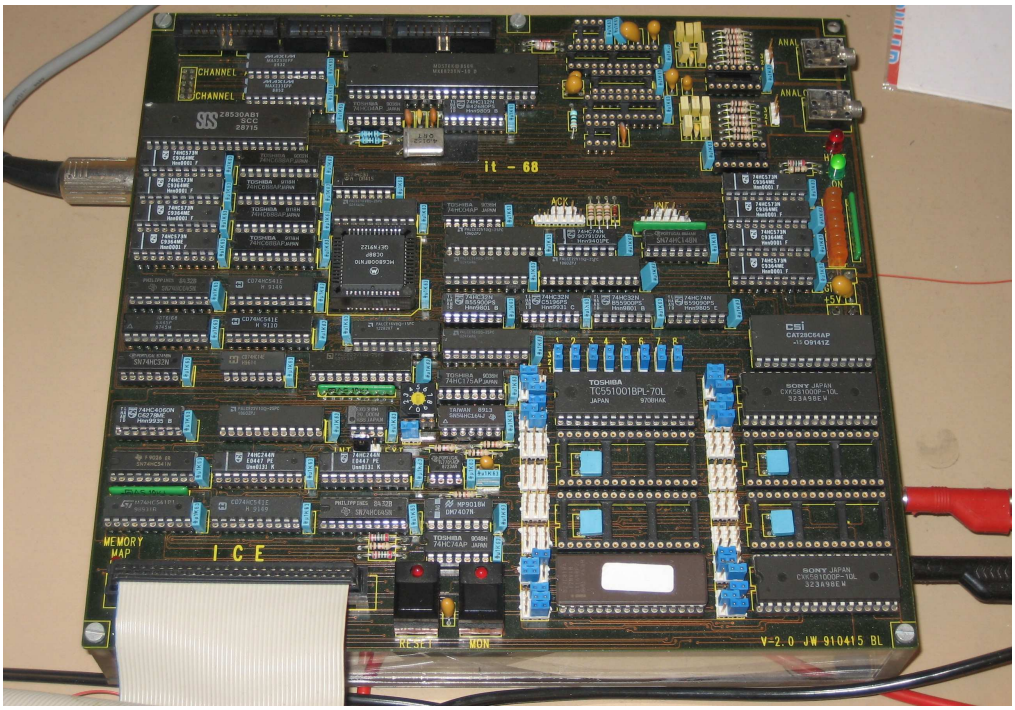



**Bild C.4.** It-68 testsystem.

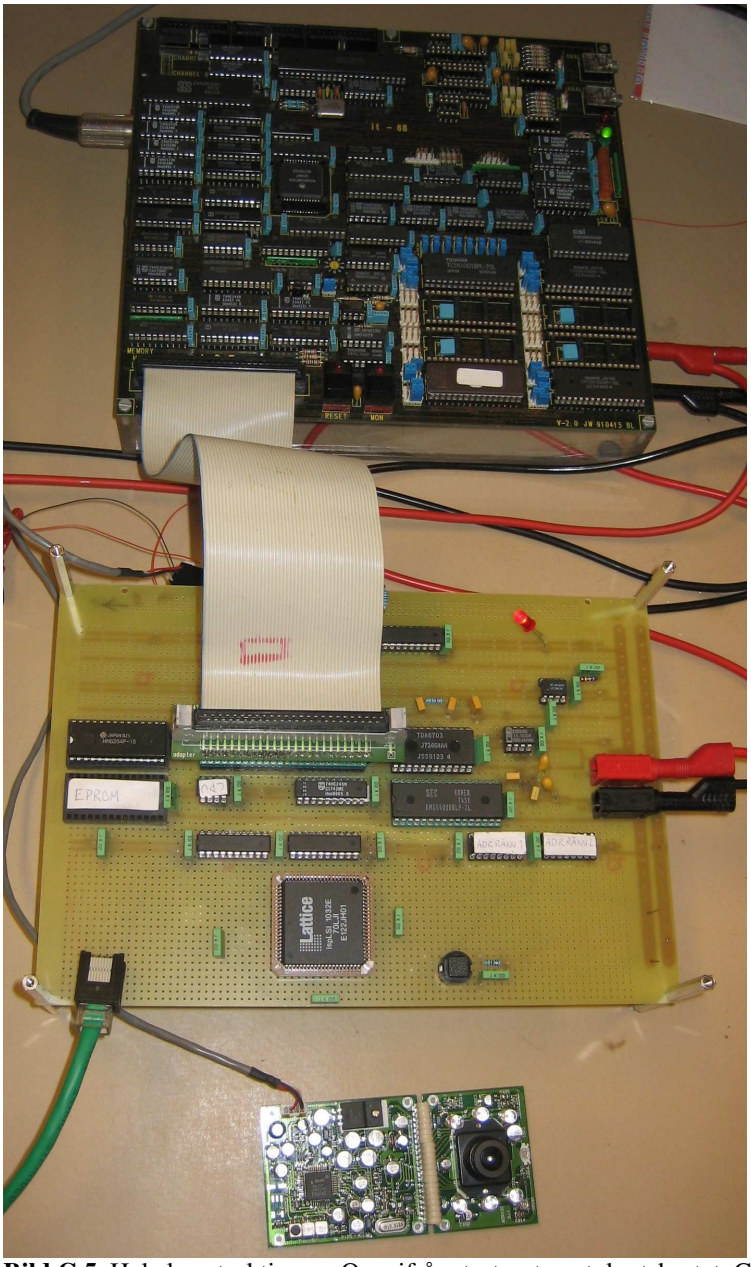



**Bild C.5.** Hela konstruktionen. Ovanifrån: testsystemet, kretskortet, CCD-kameran.