NTNU
Det skapende universitet

Implementering av robust kameramodul for lavbane, lavkost satellitt

Lasse Tolly Borja

Master i kommunikasjonsteknologi
Oppgaven levert: Juni 2006
Hovedveileder: Lars Magne Lundheim, IET

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst

Oppgaven er en videreføring av prosjektoppgave, og tar utgangspunkt i konklusjoner fra denne.

I den senere tid er en rekke transport-systemer til lav jordbane blitt fristilt til det kommersielle markedet. Disse innebærer en drastisk reduksjon i kilopris til jordbane. Dette, sammen med ny mikroteknologi gir muligheter for nytenkning når det gjelder implementering av satellitt-plattform og nyttelast.

Oppgaven går ut på å konstruere og implementere en kameramodul-prototyp med tanke på bruk i en liten lavkost-satellitt. Det skal legges vekt på enkel og robust konstruksjon, liten masse og størrelse, samt lavt effektforbruk. Det skal benyttes kommersielt tilgjengelige komponenter. Endelig satellittplattform er ikke kjent, og løsningen må ta hensyn til dette ved valg av grensesnitt.

Oppgaven gitt: 16. januar 2006
Hovedveileder: Lars Magne Lundheim, IET

Verdensrommet er svart. Kullsvart.
Og kometen er rød og den har halen bak seg.
Og den har tenkt å TANGERE jorden den sjuende august klokken 8,42 om kvelden.
Muligens 4 sekunder senere.
Professoren og jeg har regnet det ut.

Mummitrollets venn Sniff - etter å ha sett i verdens største stjernekikkert.

Forord

Denne oppgaven er, som den forutgående prosjektoppgaven, basert på et praktisk prosjekt. Det har derfor vært mange forsiktighetshensyn å ta underveis, og svært mye tid har blitt brukt til å skaffe tilveie kritiske komponenter, samt å sikre kvaliteten på arbeidet. Dette gjelder særlig teknologi som ikke ennå er satt i ordinær produksjon, såkalte *Engineering Samples*.

Betydelig arbeid er lagt ned i detaljer. Et hundretalls datablad er gjennomgått for å finne de best egnede enkeltkomponetene til prototypen. Enkeltkomponenter er grundig testet, og måleresultater sammenholdt med de oppgitte spesifikasjoner. Det er også foretatt sammenlignede tester, der det viser seg at kretser med de beste oppgitte spesifikasjoner ikke nødvendigvis har de beste målte ytelser.

Da kretsen skal fungere under forhold med parametre som delvis ikke lar seg predikere, og forhold det ikke er realistisk og gjøre langtidstester for, for eksempel termisk-vakuum test, må det være betydelig spillerom for parametre som driftstemperatur og forsyningspenning uten at kameramodulens stabilitet påvirkes.

Det har vist seg vanskelig og tidkrevende å skaffe tilveie presis informasjon om riktig anvendelse av teknologi og kretskomponenter. Kretskortutlegg, kondensatorteknologi og lavspenningsdrift av mikrokontrollere har vist seg å være det vanskeligste å innhente presis, og riktig, informasjon om. Det kan synes som det eksisterer et gap, mellom forståelse av teknologi i fagmiljøer som håndterer teoretisk aspekter, og praktisk ingeniørarbeid gjerne kalt ”god ingeniørpraksis”. Mange kretskonstruksjoner virker *på tross av* dårlig implementasjon og misforstått anvendelse av teknologi, og har svært små marginer mot ustabilitet.

Det har også tatt tid å sette seg inn i verktøy for kretskortutlegg, emulator for mikrokontroller og IDE (Integrated Development Environment) for assembly og ”C”.

Takk til førsteamanuensis Lars Lundheim for vilje til å påta seg veilederansvar for en så omfattende oppgave.

Takk også til Rein Anders Apeland og Hallvard Kringstad for hjelp med kildekode.

Lasse Tolly Borja

Sammendrag

Formålet med denne oppgaven er å videreføre arbeidet fra forutgående prosjektoppgave der muligheten for å implementere kameramodul i en studentsatellitt ble utredet.

Valg av billedsensorteknologi er gjort i prosjektoppgaven.

Arbeidsbetingelser relatert til rommiljø og belastninger under oppskyting av satellitten med henblikk på romkvalifisering av kameramodulen er også utredet i prosjektoppgaven.

Sammendrag av prosjektoppgaven er tatt med i Appendiks A.

Innledningsvis er det gjort rede for arbeidsbetingelser knyttet til kameraprosjektet, og oppgavens omfang er avgrenset. Den ferdige kameramodulen skal brukes i en studentsatellitt, bilder fra jordbane er tenkt brukt i arbeide med å rekruttere skoleungdom til realfag. Funksjonsannsynlighet og naturtro bildegjengivelse er derfor viktigere enn instrumentell nøyaktighet. Det overordnede mål er å utvikle en kameramodul med størst mulig funksjon-sannsynlighet, og derfor så få ,og robuste komponenter som mulig.

En kortfattet systembeskrivelse konkluderer med at kameramodulen skal kunne eksponere og lagre ett enkelt bilde, for senere overføring til bakkestasjon via satellittplattformens telemetrisamband. Overføring av datastrøm fra bildebrikke til minnekrets skal håndteres av en kommersielt tilgjengelig mikrokontroller med 8 bits arkitektur. Mikrokontrolleren AVR Mega1281 fra Atmel er valgt.

Prototyp av flightmodel består av to kort. SC-1A er et test- og tilkoblingskort for bildebrikken MT9V111, og ble utviklet under arbeidet med den forutgående prosjektoppgave. SC-1B er et kontrollkort med mikrokontroller, minnebrikke og noen enkle støttekretser.

Den fysiske implementeringen av kameraprototypen er beskrevet. Spesielt krevende er sanntid overføring av datastrøm fra bildebrikke til eksternt dataminne, gitt begrensninger i mikrokontrollerens klokkefrekvens. Overføring av data fra bildebrikken er avbruddstyrt med avbruddsignaler fra bildebrikken.

Kjernen i programvaren er et 2 instruksjoners assembly-segment, som med støtte i mikrokontrollerens maskinvare, overfører et 8 bits dataord fra bildebrikken til en ekstern RAM-brikke. Hele operasjonen utføres i løpet av 4 klokketikk, inklusive fortløpende inkrementering av eksternminnets adressebuss. Det er vist hvordan oppdeling av minneadressebussen, i én statisk og én dynamisk del, kan brukes til å omgå begrensninger i mikrokontrollerens bussbredde.

Prototypemodulens kretskomponenter er delt inn i inn i primær- og sekundærkomponenter, der de primære er del av den endelige kameramodulen som skal opp i rommet, mens de sekundære er støttekomponenter skal forenkle implementering av prototyp, og muliggjøre testing direkte på modulen mens den er i drift.

I gruppen primærkomponenter finner vi bildesensor, mikrokontroller, minnebrikke, adressebuss-lås og avkoblingskondensatorer. Disse kretskomponentene er gjennomgått i detalj, med spesiell vekt på egenskaper av betydning for kameramodulen.

Av sekundærkomponentene er det bare klokkekilden som er behandlet. For å begrense kompleksitet og antall frihetsgrader i eventuell feilsøking er det i prototypemodulen brukt en ekstern krystalloscillator med utgangsdrivere og programmerbar frekvensdeler, slik at klokkesignal med frekvensen $F_0/2$ kan tappes ut i tillegg til masterklokke med frekvensen F_0. $F_0/2$ trengs i denne implementering som masterklokke til bildebrikke.

I flightmodel SC-2 vil et krystall bli brukt som kilde for masterklokke. Mikrokontrolleren har krystall-inngang med inverterende forsterker. Klokkesignal til bildebrikken vil bli generert av mikrokontrolleren.

3.0V er valgt som et kompromiss mellom de ulike kretskomponentens krav til forsyningspenning. Med samme driftspenning for alle kretskomponenter kan databusser kobles direkte sammen, uten konvertering av logikknivåer.

Kretsskjema er utarbeidet i et CAD-basert kretskortverktøy. Kretskort for prototypemodulens kontrollkort skal produseres i 4-lags teknikk med forsyningspenning og jord distribuert på separate plan for å unngå serieinduktans i strømforsyning som kan resultere i at logiske desisjonsnivåer har distribuerte, instantane nivåforskjeller for de ulike kretskomponentene.

Fullstendig programkode i C er tatt med i Appendiks, sammen med MATLAB-skript for generering av bildefil.

Arbeidet med kameraprototypen viser at det er fullt mulig å realisere en robust kameramodul med dataoverføring styrt av en mikrokontroller.

1 Innledning

Formålet med denne oppgaven er å videreføre arbeidet fra forutgående prosjektoppgave, der muligheten for å implementere kameramodul i en studentsatellitt ble utredet [1]. Sammendrag av prosjektoppgaven finnes i Appendiks A.

I rom-sammenheng benyttes det i stor grad gjennomprøvd teknologi. I dag sendes det opp teknologi som ble utviklet på 60 og 70-tallet. Dette er teknologi med høy masse og høy pris. Det bør derfor legges resurser i å utrede om det vil gi bedre kost/nytte-forhold om man i fremtiden velger nyere teknologi, med lav masse, selv om denne er mindre gjennomprøvd. Besparelser i oppskytningskostnader kan vise seg å forsvare en moderat økning i prosjektrisiko.

Et godt eksempel er den europeiske landingsmodulen Huygens til en pris av 340 millioner Euro, som grunnet stor kompleksitet, svært lang utviklingstid og uhyre konservative teknologivalg, er utstyrt med et 128x128 piksel kamera [2]. Med så lav oppløsning er nytteverdien til kameraet redusert. Uten godt billedmateriell for publisering kan også det brede publikums støtte til tilsvarende prosjekter i fremtiden være skadelidende.

Å sende nyttelast til verdensrommet er ikke trivielt – det stilles krav til vekt, robusthet mot vibrasjon under oppskyting, evne til å tåle store temperatursvingninger, og strålingsresistens. Strømforbruket må minimeres for hver enkelt modul – alle enheter må kunne avgi overskuddsvarme i vakuum. For å begrense satellittens totale effektforbruk må alle nyttelastfunksjoner ha lavest mulig datarate over telemetrisambandet med jordstasjon.

Inspirert av den norske studentsatellitten nCube-1, som i hovedsak ble finansiert av Norsk Romsenter (NRS) og Andøya Rakettskytefelt (ARS), foreslo jeg vinteren 2004 å utrede muligheten for å sende opp en miniatyr kameramodul med en eventuell, fremtidig studentsatellitt.

Målsetningen for dette kamera-prosjektet er ikke først og fremst av vitenskapelig karakter. Bilder fra en fremtidig studentsatellitt er tenkt brukt ved rekruttering av realfagstudenter, og i tillegg å skape entusiasme for realfag blant norske skoleelever, og i befolkningen forøvrig. Ifølge Knut Jørgen Røed Ødegaard ved Universitetet i Oslo står interesse for romforskning og astrofysikk for en svært betydelig del av rekrutteringen til realfag.

NTNU ønsker å legge grunnlaget for en rekrutteringsatsning som engasjerer barn og unge gjennom direkte og indirekte deltagelse i prosjektet. Medlemmer av målgruppen kan f.eks. være opptatt av miljøvern og miljøovervåkning - men tørre tall og fakta er ikke alltid like spennende. Derimot vil et overvåkningsbilde fra jordbane - gjerne tatt på oppdrag fra, eller i samarbeid med ungdom være med på å skape eierskap til et student-forskningsprosjekt.

Målet er derfor ikke å utvikle et vitenskapelig måleinstrument, men et kamera som i størst mulig grad gir studentene opplevelse av å være tilstede i verdensrommet. Vi er ikke opptatt av instrumentell nøyaktighet men visuell opplevelse, altså levende og mest mulig naturtro bilder, innenfor rammen av de tekniske begrensninger som ligger i en lavbudsjett studentsatellitt. Farvesyn og sensorteknologi er behandlet i prosjektoppgavens kapittel 3 [1].

Et annet mål for kameramodulen er å demonstrere enkel og robust kamerateknologi i verdensrommet basert på kommersielt tilgjengelige lavkost-komponenter.

Det optimaliseres derfor med hensyn på funksjonsannsynlighet, og dermed implisitt på mekanisk og kretsteknisk robusthet - herunder også moderat strålingsresistens, sistnevnte begrenset av den overordnede plattformens levetid [1].

Ved valg av sensorteknologi ble det tatt utgangspunkt i hensikten med, og anvendelsen av, kameramodulen. Valg av bildesensor er gjort i prosjektoppgave [1].

Arbeidsbetingelser relatert til rommiljøet er i hovedsak knyttet til termiske problemstillinger og partikkelstråling. I tillegg vil kameramodulen utsettes for vibrasjoner og andre mekaniske belastninger under oppskyting. For detaljert behandling rommiljø med henblikk på rom-kvalifisering av kameramodulen, herunder strålingsresistens, henvises til prosjektoppgavens kapittel 2 og 5 [1].

Tolkning og fremvisning av det ferdige bildet ansees å høre inn under bakkesegmentet. Det er derfor, innenfor rammen av denne oppgave, tilstrekkelig å kunne lagre, og senere overføre ubehandlet datastrøm fra bildebrikken til satellittplatformens interne telemetribuss. Videre ansees jordkloden som tilstrekkelig jevnt belyst til at vi kan se bort fra problemstillinger knyttet til eksponering [1].

Optikk er heller ikke behandlet. Dette fordi den mekaniske konstruksjonen til den endelige satellittplattformen ikke er kjent. Linsens fysiske utforming avhenger bl.a. av bildeutsnitt, som igjen påvirkes av retningskontrollens pekenøyaktighet og plattformens stabilitet. Det henvises til prosjektoppgavens kapittel 4 - Optikk [1].

2 Systembeskrivelse

Målet med denne oppgaven er å konstruere et prototypeoppsett for satellitt-kameramodulen SC-2. SC-2 er betegnelsen for den endelige *flightmodel* av kameraet, d.v.s. modulen som skal sendes opp i rommet.

Prototypemodulen består av 2 kretskort. SC-1A er et test- og tilkoblingskort for bildebrikken MT9V111, og ble utviklet under arbeidet med den forutgående prosjektoppgave. Denne oppgave beskriver kontrollerkortet SC-1B. De sentrale deler av kretskonstruksjonen for SC-1A/B er identisk med det som skal anvendes i SC-2. Utover dette har SC-1B støttekretser og tilkoblingspunkter for kommunikasjon med, og overvåkning av modulen via en ordinær PC.

Prototypeoppsettet SC-1A/B skal fungere som et "proof of concept", og dessuten være et utgangspunkt for videre utvikling av den endelige kameramodulen - så snart målplattform er kjent.

Kameraet skal integreres i et system som på forhånd har et stramt effekt-budsjett. Strømforbruket må derfor være begrenset. Kameramodulen er riktignok i drift i svært korte tidsintervaller, men må allikevel ikke belaste systemet med kortvarige store strømmer av hensyn til strømforsyningens dimensjonering. Det er en fordel hvis billedbrikkens hvilestrøm er liten slik at det ikke er nødvendig å skru brikken helt av mellom hver eksponering – dette for å beholde relevante kontrollparametre i brikkens interne registre, og dermed unngå full reset-prosedyre mellom hver eksponering.

Satellittplattformen vil antagelig gå i en tilnærmet middag/midnatt solsynkron bane med omløpstid ca. 100 minutter. Satellitten vil altså passer gjennom jordskyggen med 100 minutters mellomrom, og oppholde seg der i noe under halve omløpstiden. Satellitten ytre struktur utsettes for enorme ytre temperatur-svingninger i løpet av en slik syklus. Alle moduler og enkeltkomponenter ombord må kunne avgi overskuddsvarme ved varmestråling i vakuum uten gravitasjon, og dermed uten atmosfæriske konveksjonsstrømmer. Fra tidligere Cubesat-prosjekter er det kjent at arbeidstemperatur i kjernen av en CubeSat vil holde seg i området omkring 20–30 grader Celsius [3]. For andre plattformer gjelder andre data.

Det vil nødvendigvis være forskjeller mellom prototypen SC-1A/B og endelig flightmodel SC-2. For det første kjenner vi ikke den endelige plattformens fysiske utforming og mål - ei heller plassering av kameramodul i forhold til f.eks. støykilder som radiomodul, antenne, switch-regulator i strømforsyning etc. Dernest kjenner vi ikke forsyningsspenning ombord - denne kan selvfølgelig konverteres - men det kreves også samsvar i logikknivåer - der kameramodul og satellittens øvrige elektronikk, f.eks. OBC (On Board Computer), er koblet sammen.

Formatet på kommandobuss og databuss er heller ikke kjent, og kan være alt fra USB, Ethernet eller ISP, til det proprietære I2C utviklet av Phillips, og som ble brukt i nCube, eller CAN-bus, sistnevnte en proprietær buss utviklet av Robert Bosch GmbH for bruk i kjøretøy, og ofte anvendt i lavkostsatellitter.

På kontrollerkortet SC-1B er det i stedet implementert UART, med en ekstern RS232-linjedriver, for å kunne kommunisere med en ordinær PC. I tillegg er det tilkobling for *JTAG-ICE* (In Circuit Emulator for mikrokontroller - basert på *Border-Scan*) og ISP (In System Programing).

Kommunikasjon mellom mikrokontroller og billedbrikke går over TWI (Two Wire Interface) som for alle praktiske formål er likt det proprietære formatet I2C fra Phillips.
For prototype-oppsettet SC-1A/B er forsyningspenning V_{cc} = 3V valgt. Dette valget vil bli motivert i kapittel 4. Bildesensor er spesifisert med analog og digital forsyningspenning i intervallet [2.55; 3.05] volt, med optimalpunkt 2.8V. Hvis det brukes annen sentral forsyningspenning enn 3V i den endelige satellitten kan det være aktuelt å forsyne mikrokontroller med V_{dd} > 3.0V for å kunne øke klokkefrekvensen. Logikknivå-konvertering må da legges inn i konstruksjonen, i form av såkalte *level-translators*, på databuss, og styrings- og synkroniseringsbuss mellom mikrokontroller og bildebrikke. Spenningsregulatorer må også legges inn i kretsen; eventuelt også DC-DC konvertering av forsyningsspenning.

Bildebrikken er levert av Micron og har betegnelsen MT9V111. Brikken har et sensorareal i VGA-format, med 640x480 bildeelementer, også kalt *piksler* (engelsk: *pixel* - forkortelse for Picture Element). Farveseparasjon foregår ved at bildeelementene er dekket av optiske farvefiltere i primærfarvene rødt, grønt og blått. Brikken er et såkalt SOC (System on Chip) bestående av en sensorkjerne som leverer en uprosessert datastrøm i 10 bit RAW-format, og en *Image Flow Prosessor* (IFP) med automatisk eksponeringskontroll og mulighet for å levere ferdig farveinterpolerte bilder i formatene ITU_R BT.656 (YCbCr), YUV, 565RGB, 555RGB, og 444RGB. IFP arbeider i sanntid. Kameraprototypen SC-1A/B benytter eksponeringskontroll og automatisk hvitbalanse for å sikre at det ønskede intervallet av motivets kontrastomfang er representert i høyest mulig tonal oppløsning, m.a.o. med flest mulige informasjonsbit pr. bildeelement. Farveinterpolasjon i IFP vil ikke bli brukt. Farveinterpolasjon øker datamengden fra 10 bit til to byte pr. bildeelement ved at hvert interpolert bildeelement er representert med 16 bit [4]. Det er viktig å merke seg at farveinterpolasjon *ikke* øker informasjonsinnholdet i bildet. Det er dessuten fordelaktig å foreta bildebehandling i bakkesegmentet slik at forskjellige algoritmer kan anvendes, og resultatene sammenlignes. Se for øvrig: Appendiks B - Farveseparasjon og Bayer-filter.

Forventet effektiv datarate for satellittens nedlink er ca. 1 kB/s [1]. Filstørrelsen for et ukomprimert bilde i VGA-oppløsning og 8bit/piksel-Bayer-format er 640x480 = 307.200 byte, som tilsvarer 300KB (2^{10} = 1024 bit pr. KB). Med et maksimalt nedlastningsvindu på 500KB er det mulig å laste ned et ukomprimert bilde i løpet av én passering av jordstasjon [1]. Kompresjon av datastrømmen stiller strenge krav til feilbeskyttelse i kilde- og kanalkoding. Det er vesentlig enklere å rekonstruere en ufullstendig datastrøm hvis bildet er ukomprimert. Med tanke på ikke-forutsett etterbruk av bildene er det vanskelig å velge en kompresjons-algoritme som ikke endrer bildene på en uønsket måte.

Det er besluttet å overføre en ukomprimert datastrøm i ubehandlet RAW-format.

Den største utfordringen i SC-1A/B er å overføre bildeinnformasjon, i form av en datastrøm, fra bildebrikken til et eksternt minne. Fordi billedbrikken ikke har buffer for mellomlagring av bildedata må overføring av datastrøm gjøres i sanntid. Vanligvis håndteres datastrømmen fra en bildebrikke v.h.a. en dedikert kontroll-krets, gjerne med bildekompresjon implementert i masinvare. Kontrollkrets tilpasset MT9V111 foreligger ikke. For ordens skyld nevnes allikevel noen ulemper med dedikert kontroller:

- kan ikke tilpasses ulike bussformater (endelig satellittplattform ikke kjent)
- vanligvis ingen mulighet for å styre øvrige kamerafunksjoner gjennom GPIO[1)]
- kan ikke utføre bildebehandling utover det som er implementert i brikken

1) General Purpose In/Out, inn- og utganger til generell bruk

Et annet alternativ er å implementere utlesnings- og overføringslogikk i en *FPGA*-krets (Field-Programmable Gate Array). En slik krets vil være mer sårbar enn en fastkoblet (hardwired) krets fordi logikken må settes opp ved hvert strømpåslag.

Det er selvfølgelig mulig å fremstille en fastkoblet krets til flightmodel basert på maskinvare-beskrivende kode, for eksempel VHDL, brukt for å sette opp FPGA-krets på prototypekortet. Kostnaden ved å fremstille en slik krets er svært høy. Viser det seg å være feil ved kretsen, eller det er behov for endringer i funksjonalitet under utvikling, må kretsen fremstilles på nytt basert, på ny VHDL-kode.

Kameramodulen anvender i stedet en mikrokontroller for utlesning av data fra bildebrikke til lokalt minne. Mikrokontrolleren vil også styre overføring av data fra minnet til satellittens telemetribuss, ved radiotransmisjon av bildedata.

Frem til nå har problemet ved bruk av mikrokontroller i små kameramoduler vært at maksimal klokkefrekvens ikke har vært tilstrekklig høy til å kunne holde følge med en sanntid datastrøm fra en bildebrikke. MT9V111 leverer en datastrøm på 3.5–13.5MB/s, avhengig av kameramodulens masterklokke.

Følgende tre momenter gjør det mulig å håndtere denne datastrømmen v.h.a. en mikrokontroller:

- Ny AVR-mikrokontroller fra Atmel som opererer ved relativt høye klokkefrekvenser ved svært lav spenning.

- Effektiv kode med utgangspunkt i enkle men kraftige instruksjoner, der deler av minnestyringen er implementert direkte i mikrokontrollerens maskinvare.

- Eksekvering av kode i mikrokontroller skjer strengt synkront med klokkesignalet til bildebrikken, og dermed billedbrikkens utgående databuss.

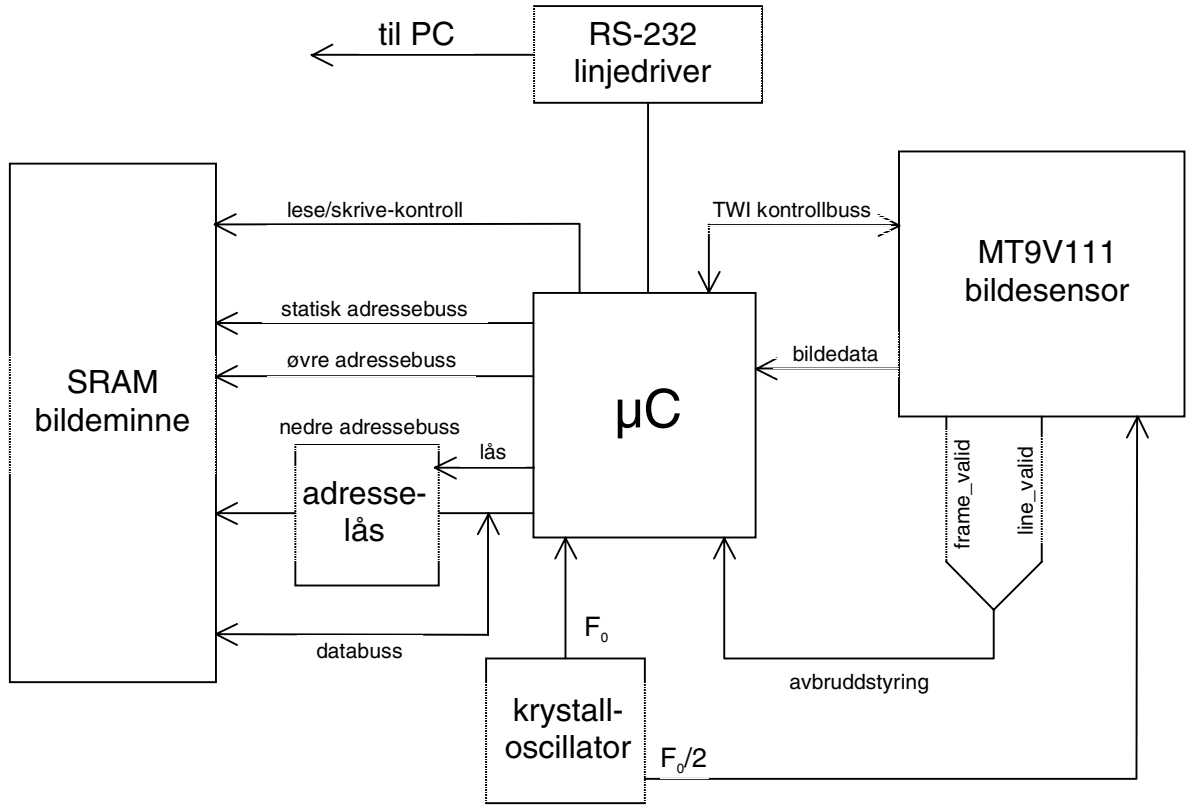
Mikrokontrolleren har noe lenger driftsintervaller enn bildebrikken, og dermed noe større behov for kjøling.

Minnet må ha lavt strømforbruk ved lese/skriveoperasjoner og bør ha minimal hvilestrøm. Den høye dataraten fra bildebrikken stiller også krav til minnets aksesstid. For å redusere kompleksitet, og øke funksjon-sannsynlighet, er minnestørrelsen begrenset til å romme ett enkelt bilde. Dette har ikke praktisk betydning da det som nevnt bare er mulig å laste ned ett enkelt bilde ved hver passering av jordstasjon.

Det overordnede mål er å utvikle en kameramodul med størst mulig funksjon-sannsynlighet, og derfor så få og robuste komponenter som mulig.

2.1 Fysisk krets

Den fysiske implementeringen av prototypemodulen SC-1A/B består av en bildesensor, en SRAM minnebrikke, en mikrokontroller, samt noen enkle støttekretser. Figur 2.1.1 viser et forenklet kretsdiagram.



Figur 2.1. 1 Forenklet kretsdiagram for kameramodul

Modulen er styrt av en krystall-oscillator med to utganger. Oscillatoren er utstyrt med en programmerbar frekvensdeler. Bildebrikken leverer et nytt bildeelement på bildedata-buss annethvert klokketikk. Mikrokontrolleren trenger 4 klokketikk for å overføre et bildeelement fra bildedatabuss til SRAM-bildeminne, ett tikk for å lese bildeelementet fra bildedata-buss til internt arbeidsregister, og ytterligere 3 tikk for å skrive data videre til eksternt minne. For å oppnå synkron datastrøm mellom bildebrikke og mikrokontroller må bildebrikken derfor påtrykkes klokkesignal med frekvensen $F_0/2$ – der F_0 er frekvensen til masterklokke påtrykt mikrokontroller.

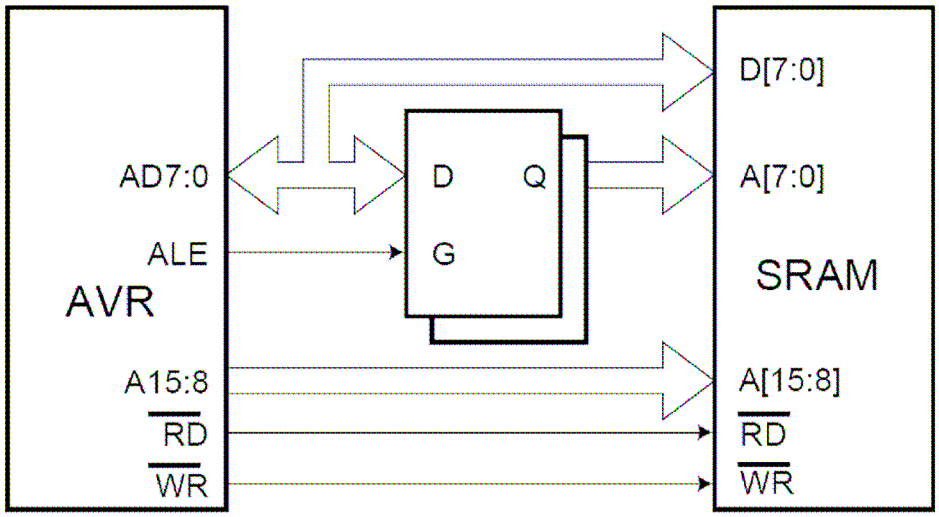
Prototypemodulen SC-1A/B overfører bildedata fra bildeminnet til en ordinær PC via en RS-232 seriebuss. Mikrokontrolleren har UART implementert i masinvare. Dataord som skal overføres skrives og leses direkte til/fra et dedikert register i kontrolleren. Maskinvaren tar seg av overføring, og setter interuptflag når UART-periferenheten er klar til å motta/levere et nytt dataord. En ekstern RS-232-linjedriver konverterer logisk 0 og 1 til/fra mikrokontrollerens UART-port (henholdsvis 0 og +2.4V) fra/til RS-232-nivåene +/-5V. Merk at logisk 1 er representert ved -5V. RS-232-signalet er altså invertert i forhold til ordinære logikknivåer.

Overføring av data fra bildebrikken til bildeminnet er avbruddstyrt. Bildebrikken setter porten *"frame_valid"* høy når utlesning av et nytt bilde starter. *"line_valid"* settes høy ved begynnelsen av hver ny linje, og returnerer til lav når overføring av linjen avsluttes. "line_valid" og "frame_valid" er koblet til hver sin *interupt*-inngang på mikrokontrolleren, med uavhengige *interupt-vektorer*. Flytdiagram for overføringsekvens er vist, og beskrevet, i kapittel 2.2 Programvare.

Mikrokontrolleren AVR Mega2561 kan skrive til, og lese fra, en ekstern SRAM-brikke [5]. Mega2561 er basert på en 8 bits kjerne, og har 8 bits intern databuss, men kan, med bruk av to sammenkjedede internregistre, styre en 16 bits ekstern adressebuss. Dataoverføring er begrenset til 8 bits dataord.

For å kommunisere med minnet bruker mikrokontrolleren to 8 bits GPIO-porter (General Purpose In/Out, inn- og utganger til generell bruk), i tillegg til to dedikerte linjer for lese/skrivekontroll av minnebrikken, og en linje for styring av en ekstern adresselås. Adresselåsen er en oktal, transparent D-lås som kan låse utgangen til logiske nivåer påtrykt inngangen.

Figur 2.1.2 viser hvordan mikrokontrolleren er koblet via adresselås til minnebrikken.

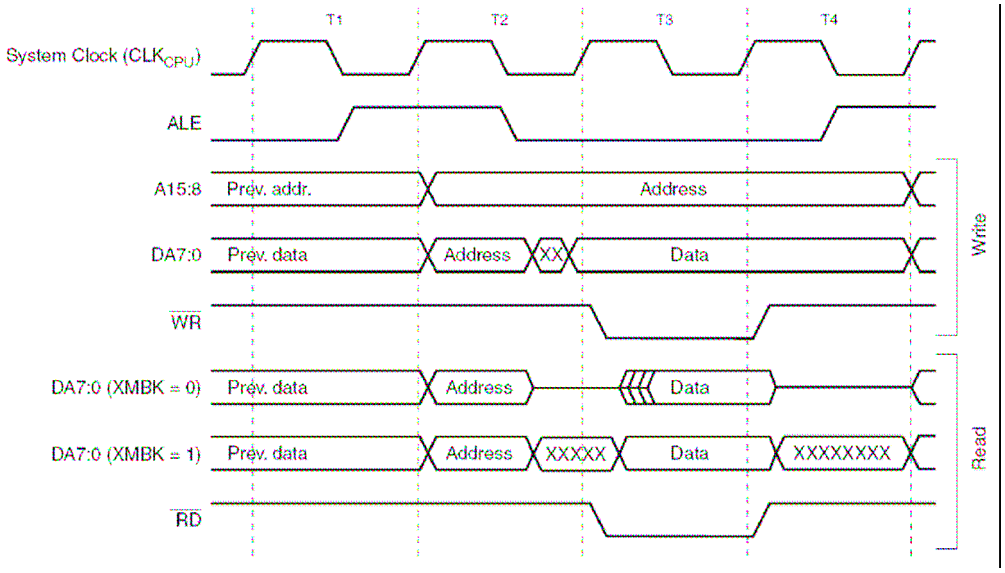


Figur 2.1. 2 Sammenkobling av mikrokontroller og minnebrikke via adresselås for nedre halvdel av adressebuss. Figuren er hentet fra [6]

Den eksterne adressebussens åtte minst signifikante bit (8 LSB) er, sammen med ekstern databuss, koblet til mikrokontrollerens PORT-A. Adressebussens åtte mest signifikante bit (8MSB) er koblet til PORT-B. De øvrige kontrollinjene er koblet til PORT-G.

Styring av minnebrikke, adresselås og adressebuss er direkte implementert i mikrokontrollerens maskinvare. En lese- eller skriveoperasjon fra/til eksternt minne aktiveres med én enkelt assembly-instruksjon, og eksekveres i løpet av tre klokketikk, inkludert inkrementering av registeret som lagrer verdien for den eksterne adressebussen [6] [7].

Figur 2.1.3 viser tidsdiagram for overføring av et dataord til og fra eksternt minne.



Figur 2.1. 3 Tidsdiagram for overføring av data mellom mikrokontroller og ekstern SRAM. [6]

Overføring av et dataord *fra* mikrokontrollerens internregister til en gitt minneposisjon i ekstern SRAM foregår på følgende måte:

Først skrives 8 MSB av adressebussen til mikrokontrollerens PORT-A, og dermed også til adresselåsens inngang. Deretter låses adresselåsens utgang ved å legge ALE (Adress Latch Enable) lav. Adressebussens 8 MSB er samtidig skrevet til PORT-C. Dataordet som skal skrives til SRAM kan nå overføres fra et av mikrokontrollerens internregistre til PORT-A, uten å påvirke adresseinngangen til minnebrikken. Tilslutt trekkes minnebrikkens WE-port (Write Enable) lav og dataordet overføres til minneposisjon med adresse tilsvarende verdien til adressebussen.

Figur 2.1.3 viser også tilsvarende sekvens for overføring av dataord fra eksternt minne *til* mikrokontrollerens intern-registerbank.

I kameramodulen må data fra en hel bildelinje d.v.s. 640 bildeelementer á 1 byte, overføres til bildeminnet i en uavbrutt sekvens, med likt antall eksekverte instruksjoner pr. bildeelement, for ikke å miste synkronitet. Dette krever at en 10 bits adressebuss inkrementeres fortløpende under utlesing av bildedata fra bildebrikken. Den samlede datamengden fra et bilde er 300kB som krever en adressebuss på 19 bit. Maksimal adressebuss-bredde styrt direkte fra mikrokontrollernes maskinvare er 16 bit. Adressebussen i kameramodulen er derfor delt opp i en dynamisk og en statisk del.

Nedre adressebuss og de nederste linjene (2 LSB) fra øverste adressebuss i figur 2.1.1 utgjør den dynamiske delen av minnestyringen, til sammen 10 bit. Den dynamiske adressebussen inkrementerer adresseverdi fortløpende under sanntid utlesing av bildedata fra bildebrikken.

De fem påfølgende bit i øvre adressebuss utgjør, sammen med fire bit fra mikrokontrollerens PORT-D, den statiske adressebussen, og styrer valg av virtuell minnebank for lagring av én enkelt bildelinje. For å spare klokketikk under overføring av datastrøm i sanntid oppdateres den statiske adressebussen i bildebrikkens horisontale *blanking-intervall*. Dette er nærmere beskrevet i kapittel 2.2 Programvare.

Data fra bildebrikken leses ut linje for linje, der en bildelinje utgjør 640 byte, og lagres i en virtuell minnebank. For å forenkle implementering er en virtuell minnebank satt til 1024 byte. Gitt standardstørrelser på minnebrikker er det ingen besparelser i å sette virtuell minnebank til 640 byte; men oppdatering av adressebuss vil bli betydelig mer komplisert, og ikke minst mindre oversiktlig, fordi 9 bit bare gir 512 minneposisjoner. Minnebrikker leveres i standardstørrelser på 256 og 512kB. Et bilde i VGA-oppløsning har 640x480 bildeelementer, som med 8 bit pr. bildeelement utgjør 300kB. Det er derfor ikke plass til et VGA-bilde på en 256kB brikke. Ved å lagre hver 640 byte bildelinje i en virtuell 1kB minnebank vil et VGA-bilde, som har 480 linjer, kreve 480kB minne, og få plass på en 512kB minnebrikke.

Merk: Det øverste bit i mikrokontrollerens øvre adressebuss (PORT-C) er av praktisk årsaker ikke i bruk. Fordi mikrokontrollerens I/O-registre og interne SRAM har adresser i samme adresserom som ekstern RAM, se Figur 3.1.2.1, og derfor deler en 16 bits adressebuss med maksimalt 64kB adresserom, må det eksterne minnet deles i 32kB hovedbanker for å kunne utnytte hele minnebrikken. Dette løses enklest ved å fristille øverste pinne (MSB) i mikrokontrollerens eksterne adressebuss [6]. Den fristilte pinnen kan selvfølgelig brukes i den statiske adressebussen, men det viser seg å gi mer oversiktlig kode hvis den statiske adressebussens 4 øverste linjer holdes samlet på mikrokontrollerens PORT-D.

For detaljert kretsskjema henvises til Appendiks C

2.2 Programvare

Programvarens hovedoppgave er å overføre 640 bildeelementer fra bildebrikken til kameraminnet i en uavbrutt sekvens, med likt antall eksekverte instruksjoner pr. bildeelement for ikke å miste synkronitet. Dette krever bl.a.at en 10 bits adressebuss inkrementeres fortløpende under utlesing av bildedata fra bildebrikken. Programvaren skal også utføre innledende oppsett av bildebrikkens interne kontrollregistre, samt overføre det eksponerte bildet til en PC via et UART seriegrensesnitt.

Kildekode for styring av *prototypemodulens* eksterne in/ut-enhet, og kodesegment for innledende oppsett av bildebrikkens kontrollregistre, er skrevet i programmering-språket C. Disse kodesegmentene vil ikke bli implementert i den ferdige flightmodel SC-2. All kode i flightmodel vil være skrevet i *assembly* (maskinkode) med unntak av styring av TWI (Two Wire Interface) som brukes til oppsett av bildebrikkens internregistre. Dette kodesegmentet vil være basert på de-assemblet C-kode som blir kvalitetssikret linje for linje, og om nødvendig skrevet om.

Mikrokontrolleren har 4kB internt statisk minne (SRAM). Et VGA-bilde i 8 bit/piksel oppløsning har en størrelse på 640x480 = 300kB. Det eksponerte bildet må derfor lagres i ett eksternt minne.

Kjernen i kameramodulens programvare styrer overføring av dataord fra billedbrikken til mikrokontrollerens eksterne statiske minne (XRAM). Dette kodesegmentet må skrives direkte i assembly for full kontroll av eksekveringsrekkefølge, og antall klokketikk pr. overført dataord. En kraftig instruksjon, med støtte for minnestyring implementert direkte i mikrokontrollerens maskinvare, overfører et dataord fra et vilkårlig internt arbeidsregister til en minnelokasjon i ekstern SRAM. Instruksjonen bruker 3 klokketikk på hele operasjonen, inklusive inkrementering av den eksterne adressebussen.

Figur 2.2.1 viser et forenklet blokkdiagram for programflyten i kameramodulen. Diagrammet er ikke et strengt formelt flytdiagram.

Programstart er lagt til mikrokontrollerens avbruddsvektor for strømpåslag. Denne vektoren ligger øverst i programmminnet og har høyest prioritet av avbruddsvektorene. Den fungerer også som programstartadresse ved fysisk reset av brikken, for eksempel ved ekstern reset aktivert over satellittens telemetrisamband.

Programmet starter med å sette opp bildebrikkens interne kontrollregistre ved å skrive de nye registerverdiene over et TWI seriegrensesnitt. Denne operasjonen er delvis støttet i mikrokontrollerens maskinvare og foregår ved at bildebrikkens kontrollregisteradresse og tilhørende registerverdi skrives til et dedikert register internt i mikrokontrolleren. Det er bare et fåtall kontrollregistere i bildebrikken som må endres i forhold til startverdi ved strømpåslag. Viktigst er endring av bildebrikkens utgangsignal, fra farveinterpolert RGB til ubehandlet RAW-datastrøm. Dessuten utføres en såkalt *soft-reset* av bildebrikkens kjerne og IFP ved å skrive 1, og deretter 0, til kjernens register 0x0D og IFPs register 0x07. dette gjøres for å sikre at bildebrikken er logisk stabil. TWI er for alle praktiske formål likt det proprietære formatet I2C fra Phillips. For detaljert beskrivelse av mikrokontrollerens dedikerte rutiner for kommunikasjon over TWI hevises til [6]. C-objektet ”i2c.c” finnes i Appendiks E.

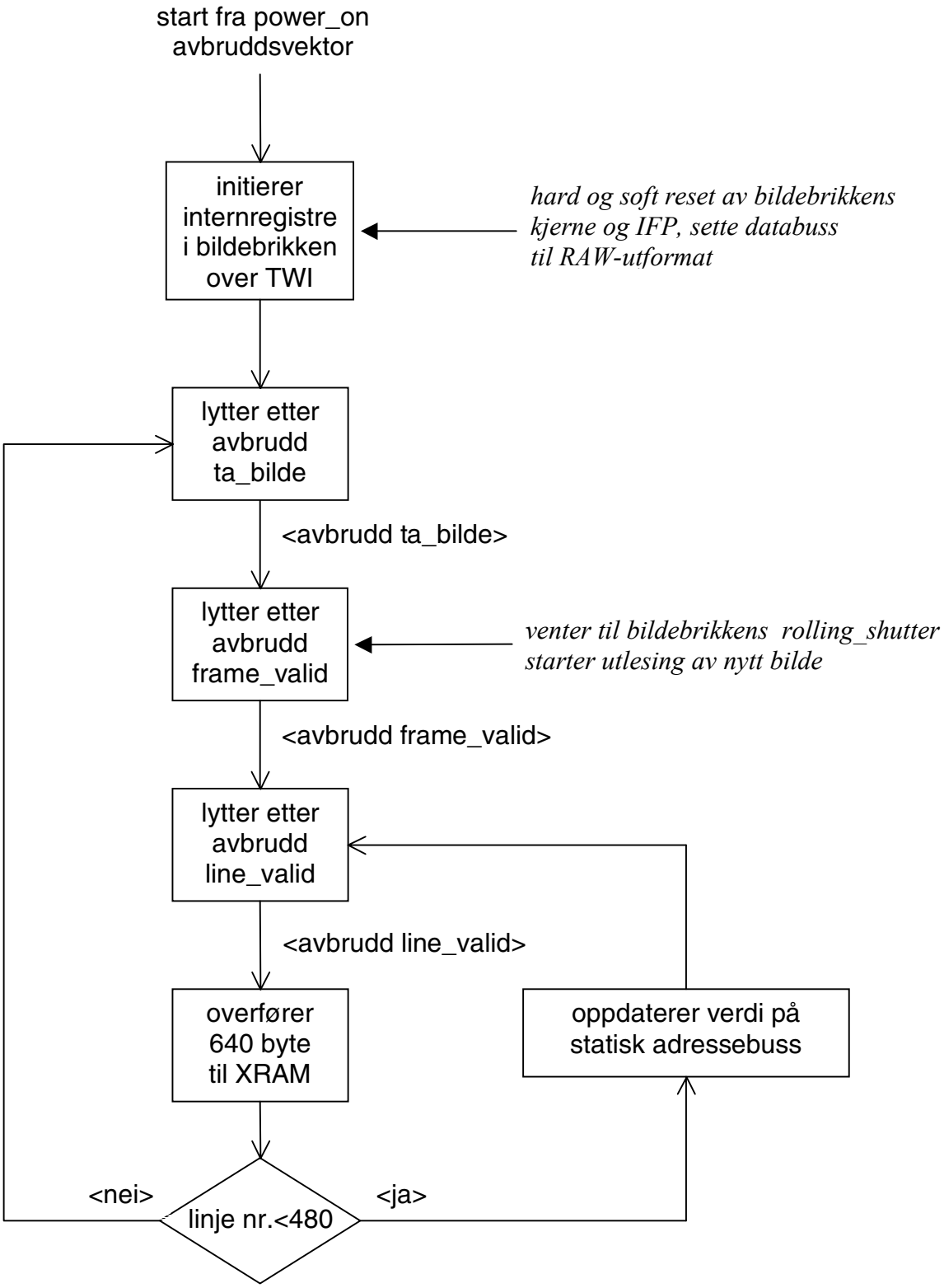
Etter at bildebrikken er satt opp går mikrokontrolleren over i ventetilstand i form av en løkke som lytter til porten for avbrudd <ta_bilde>. Alle andre avbruddsporter er deaktivert i denne tilstanden. Avbruddet <ta_bilde> aktiveres i prototypemodulen med en ekstern trykknapp. I flightmodel SC-2 kan avbrudd enten aktiveres eksternt via telemetri-komandobuss eller med en timer internt i mikrokontrolleren.

Når avbruddet <ta_bilde> er aktivert må programvaren finne begynnelsen av et bilde. Bildebrikkens sensorkjernen har progresiv utlesning som medfører at bildedata avleses, og overføres, linje for linje. Utlesning av bilder foregår kontinuerlig, med 10-30 bilder/sek avhengig av masterklokke og integreringstid (eksponeringstid). Utgangen *frame_valid* på bildebrikken er koblet til en avbruddsport på mikrokontrolleren og settes høy når utlesning av nytt bilde påbegynnes.

Når avbruddet <frame_valid> er aktivert venter kontrolleren på det første <line_valid> avbruddet. Billedbrikken setter *line_valid*-port høy når gyldig billeddata befinner seg på utgående databuss. Denne porten brukes til å styre mikrokontrollerens avbruddsrutine som overfører en enkelt linje med billedata fra billedbrikke til ekstern minnebrikke. Overføring av data fra ett enkelt bildeelement utføres med assembly-koden:

```
IN temp, PORTD    ; dataord hentes fra port D til mikrokontrollerens arbeidsregister
ST Z+, temp       ; dataord skrives til ekstern SRAM med adresse fra register Z,
                  ; ”+” inkrementerer adresseregister fortløpende
```

En full linje med bildeinformasjon, tilsvarende 640 dataord, må overføres i et strekk, uten klokketikk-krevende hoppinstruksjoner i koden. For å ungå hoppinstruksjon gjentaes ovenstående kodesegmentet 640 ganger. Instruksjonen IN opptar en programminne-adresse mens ST opptar to. Overføring av en bildelinje krever altså 640x(2+1) = 1920 minnelokasjoner. Siden flashminnets størrelse er oppgitt i antall byte, ikke antall minnelokasjoner krever denne rutinen 3840 byte + noe administrasjonskode – til sammen ca. 4MB programminne. Dette er betydelig mindre enn minnestørrelsen i mikrokontrollerens



Figur 2.2. 1 Programflyt i kameramodulen

minste konfigurasjon, Mega640, som har 64MB flash-minne. Utgaven tilgjengelig som vareprøve for innledende tester, Mega2561, har 256MB flash-minne.

For å slippe å skrive inn ovenstående kodesegment 640 ganger brukes en såkalt *makro* støttet i *IDE* [2)] der kodesegmentet kan kalles gjentatte ganger.

Etter at en bildelinje er overført inkrementeres en intern linjeteller. Dette skjer i bildebrikkens såkalte *blanking*-intervall, vist i figur 3.1.1.6 og beskrevet i kapittel 3.1.1. Hvis linjenummeret etter inkremetering er <480 oppdateres verdien på den statiske delen av den ekstern adressebussen vist i figur 2.1.1, og beskrevet i kapittel 2.1. Deretter venter programmet på det neste <line_valid> avbruddet.

Hvis linjenummeret etter inkrementering av linjeteller er ≥480 går programmet tilbake til tilstanden ”lytt etter avbrudd <ta_bilde>.”

Utlesning av det eksponerte bildet lagret i eksternt minne er med hensikt ikke tatt med i figur 2.2.1. Denne operasjonen vil utføres på ulike måter, avhengig av om bildet skal overføres fra prototype-modulen til en ordinær PC, eller fra flightmodel til en ennå ikke kjent satellittplattforms interne telemetribuss.

2) *IDE* - Integrated Development Environment – brukergrensesnittet i det dedikerte programmeringsverktøyet AVR-Studio

3 Kretskomponenter

Kretskomponenter til kameraprototypen kan naturlig deles inn i primær- og sekundærkomponenter, der de primære er del av den endelige kameramodulen som skal opp i rommet, mens de sekundære er støttekomponenter skal forenkle implementering av prototyp, og muliggjøre testing direkte på modulen mens den er i drift.

3.1 Primærkomponenter

I gruppen *primærkomponenter* finner vi bildesensor, mikrokontroller, minnebrikke, adressebuss-lås og avkoblingskondensatorer.

3.1.1 Bildesensor

En bildebrikkes sensorareal består i hovedsak av et rutenett av fotodioder som konverterer innfallende fotoner til elektrisk ladning, som igjen akkumuleres i en lokal kondensator. Ladning fra hvert bildeelement konverteres til spenning, forsterkes, kvantiseres og kan senere vises som et bilde på en skjerm, der hvert punkt har luminans og krominans proporsjonal med den utleste ladning eller spenning for tilsvarende koordinat i sensorens rutenett.

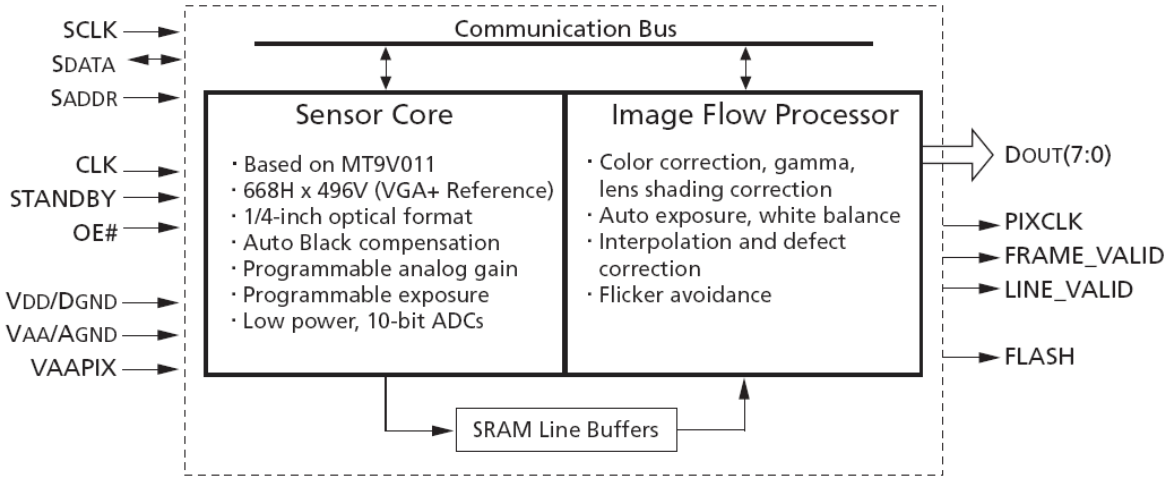
I motsetning til CCD-teknologi har CMOS-bildebrikken kvantisering av bildesignalet implementert på substratet, og har derfor digital utgang. Øvrig signalbehandling kan også integreres på substratet. Eksempler på slik signalbehandling er automatisk hvitbalanse, eksponeringskontroll, vignetterings-kompensasjon og reduksjon av Fixed Pattern Noise (FPN). De fleste CMOS-teknologier har, i tillegg til en fotodiode, minst tre transistorer integrert i hvert billedelement. Rimelige brikker med dårlig arealeffektivitet har gjerne en mikrolinse plassert over hver piksel, som samler lyset på den lysfølsomme fotodioden. Høykvalitetsensorer har 4 transistorer i hvert bildeelement. Dette muliggjør *Korrelert dobbeltsampling* (CDS) der bildeelementet punktprøves rett før eksponering for å kartlegge den instantane støyen. Etter eksponering punktprøves billedelementet på nytt og verdien fra første punktprøving blir trukket fra. CDS reduserer bl.a. FPN og reset-støy fra fotodiodens kapasitans [8].

Til SC-1A/B og SC-2, er bildebrikken MT9V111, levert av Micron, valgt. Valg av sensorteknologi og leverandør er motivert i prosjektoppgaven [1].

MT9V111s sensorkjerne (*sensor core*) er hentet fra Micron MT9V011, som er en ren bildesensor uten eksponeringskontroll og bildeprosessering [5]. MT9V011 leverer et RAW Bayer-signal på en ekstern 10 bits parallellbuss. Nytt bildeelement presenteres på utgangen *annethvert* masterklokketikk.

Figur 3.1.1.1 er hentet fra [4] og viser oppbygningen av MT9V111. Brikken er et såkalt SOC (System on Chip) bestående av en sensorkjerne som leverer en uprosessert intern datastrøm i 10 bit RAW-format, og en *Image Flow Prosessor* (IFP - bildestrømprosessor) med automatisk eksponeringskontroll og mulighet for å levere ferdig farveinterpolerte bilder i formatene ITU_R BT.656 (YCbCr), YUV, 565RGB, 555RGB, og 444RGB. IFP arbeider i sanntid. Et interpolert bildeelement er representert med 16 bit delt i to byte. Én byte leveres på en 8 bits parallellbuss hvert masterklokketikk. IFP er vist til høyre i figur 3.1.1.1.

Et blokkdiagram av IFP er vist i figur 3.1.1.2. Automatisk eksponeringskontroll (AE) og hvitbalanse (AWB) foregår ved at IFP beregner riktige verdier for individuell forsterkning av hver primærfarvekanal, samt bildeelementenes integreringstid, på grunnlag av akkumulerte statistiske data fra det ferdig farveinterpolerte bildet. Sensorkjernen mottar registersettinger fra IFP gjennom den interne kommunikasjonsbussen vist i figur 3.1.1.1.



Figur 3.1.1. 1 Micron MT9V111 SOC (System on Chip) med kjerne basert på MT9V011 [4]

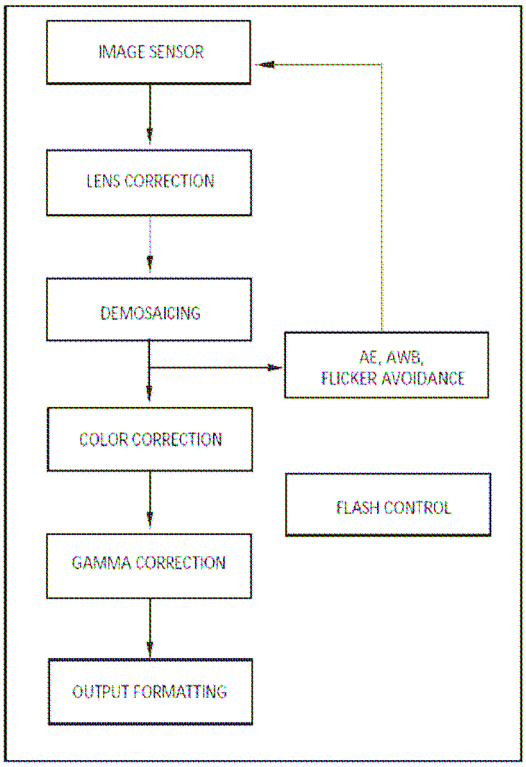
Selv om automatisk eksponeringskontroll- og hvitbalanse henter data fra farveinterpolert bilde, viser tester at RAW-data allikevel kan leses ut med AE og AWB aktivert, gitt riktig verdi i IFP-register 0x08. Dette er av stor betydning for implementering. Kameraprototypen SC-1A/B benytter eksponeringskontroll og automatisk hvitbalanse for å sikre at det ønskede intervallet av motivets kontrastomfang er representert i høyest mulig tonal oppløsning, m.a.o. med flest mulige informasjonsbit pr. bildeelement. Farveinterpolasjon øker datamengden fra 10 bit til to byte pr. bildeelement ved at hvert interpolert bildeelement er representert med 16 bit, uten at informasjonsinnholdet økes, og er derfor ikke egnet for overføring over satellittplattformens telemetrisamband [4]. AE og AWB må altså brukes samtidig med RAW-format på utgående databuss.

MT9V111 er i utgangspunktet ikke beregnet på å levere RAW-data og har derfor en 8 bits parallell-buss som utgang. MT9V111 kan settes opp til å levere en datastrøm i 10 bit RAW-format, men deler da hvert bildeelement opp i to segmenter som klokkes ut hver for seg på annenhver masterklokke. På første klokketikk presenteres 8MSB, på neste klokketikk presenteres 2LSB som én byte, med 6 logiske nuller som fyller de tomme plassene i dataordet. Kameramodulen bruker kun 8MSB og leser derfor bare annethvert dataord fra MT9V111s utgang.

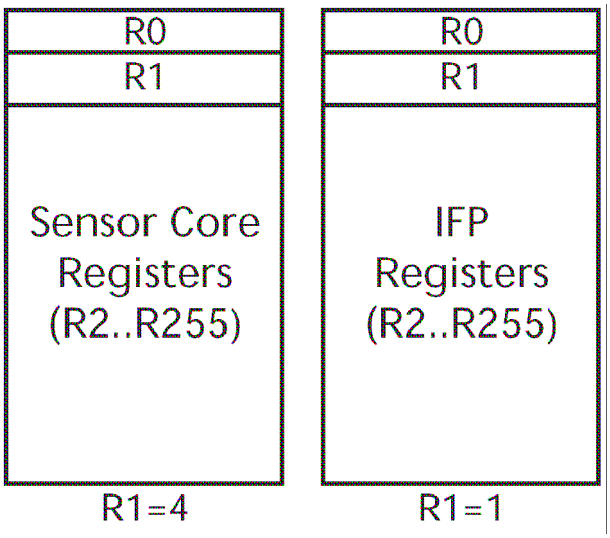
MERK: Det foreligger svært god parametrisering av jordens luminans og kontrastomfang - eksponeringsparametrene kan derfor forhåndsberegnes svært nøyaktig. Vi kan altså plassere den interessante delen av motivets kontrastomfang nøyaktig innenfor kamerasystemets (inkl. optikk) eksponeringsomfang. En tonal oppløsning på 8 bit er derfor tilstrekkelig – spesielt sett i lys av kameramodulens forutsetninger – som er å levere et vilkårlig bilde fra jordbane for å demonstrere at teknologien virker.

Sensorkjernen og bildestrømprosessoren (IFP) er styrt av registre, delt inn i to separate adresseområder, vist i figur 3.1.1.3.

Registrene kan leses fra, og skrives til, via et eksternt seriegrensesnitt i TWI-format (Two Wire Interface). Valg av adresserom gjøres ved å skrive verdien 0x001 eller 0x004 til register R1 (Reg0x01), som er representert i begge adresseområder.



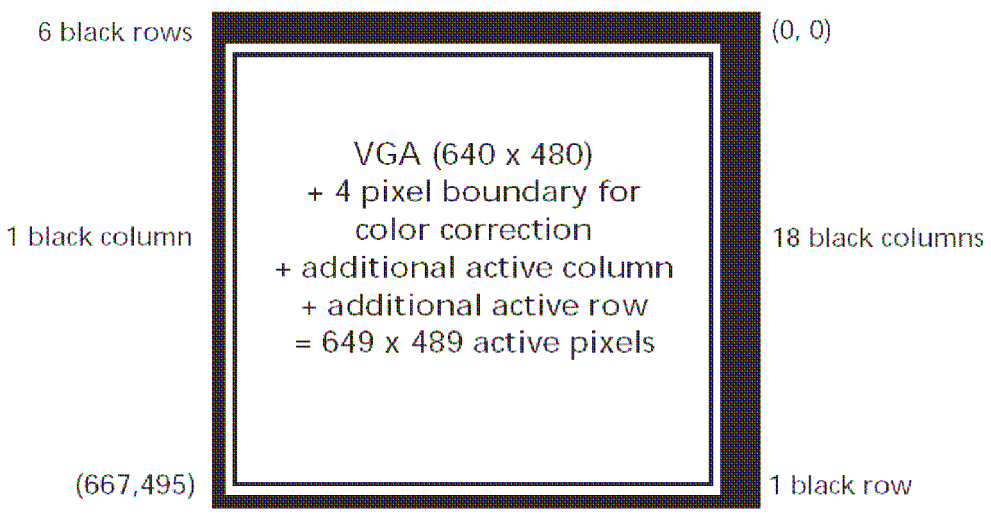
Figur 3.1.1. 2 Blokkdiagram for *Image Flow Prosessor* (IFP) i bildebrikken MT9V111 [4]



Figur 3.1.1. 3 Oppdeling av interne kontrollregistre for sensorkjerne og IFP [4]

Sensorkjernen består av en pikselmatrise, en analog utlesningsmodul og en 10 bits ADC med programmerbar forsterkning og sortnivå-korrigering. Figur 3.1.1.4 er hentet fra MT9V111s datablad og viser inndelingen av billedbrikkens sensorareal [4]. Bildeelementene er ordnet i 668 kolonner og 496 rader. De første 18 kolonner og de første 6 rader er dekket av et optisk

sort belegg, og blir brukt internt i brikken til korrigering av bildesignalets sortnivå.. Den siste raden samt den siste kolonnen er også dekket, og dermed også optisk sort.



Figur 3.1.1. 4 Inndelingen av billedbrikkens sensorareal [4]

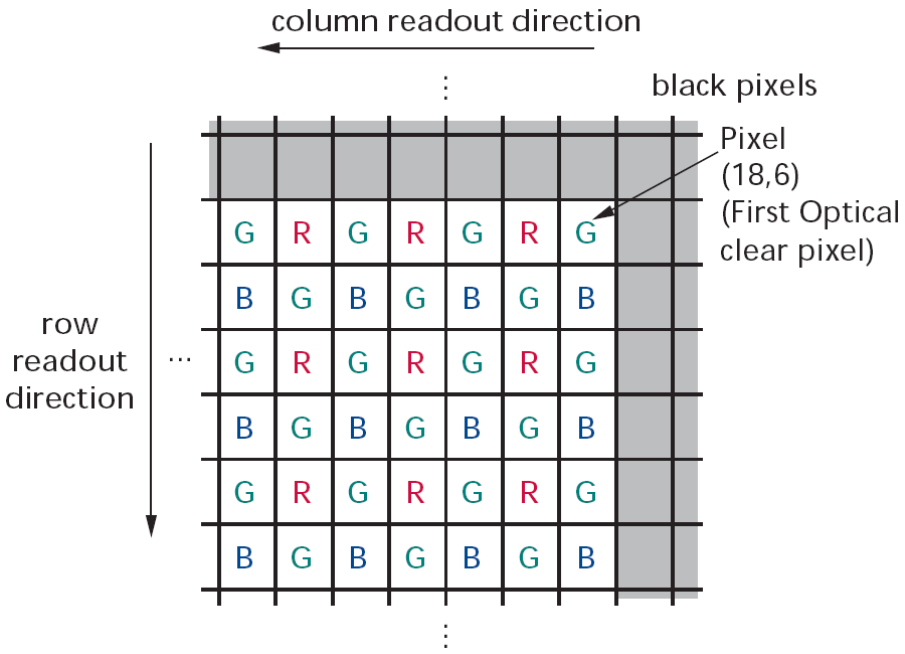
Sensorarealet har 649 rader og 489 kolonner med optisk aktive bildeelementer. Dette danner en ramme, 4 bildeelementer bred, rundt et 640x480 piksler stort område som tilsvarer VGA-oppløsning. Rammen brukes for å unngå kanteffekter ved interpolasjon av farveinformasjon for hvert enkelt billedelement, som kun leverer farveinformasjon i én av de tre farvekanalene rød, grønn og blå. Den manglende farveinformasjonen beregnes ved å sammenligne bildeelementets luminansverdi med luminans fra omliggende bildeelementer med ønsket farve, og vekte beregningen etter en valgt algoritme.[3)] Hvis de ytterste aktive bildeelementene henter informasjon fra de blendede bildeelementene i ytterkant vil bildet bli skjemmet av en mørk rand.

Sensorkjernens billedelementer er dekket av farvefiltre ordnet i et såkalt *Bayer-mønster* – se fig. 3.1.1.5. Aktiv rad nr. 0 og påfølgende partallsnummererte rader har røde og grønne bildeelementer, aktiv rad nr. 1 og påfølgende oddetallsnummererte rader har blå og grønne bildeelementer. Tilsvarende har partallsnummererte kolonner blå og grønne bildeelementer mens oddetallsnummererte kolonner har røde og grønne elementer. Bildeelementet øverst til høyre på en bildebrikke har ved konvensjon posisjonsnummer (0,0). Sensormatrisen avleses progressivt linje for linje, såkalt *progressive scan*. En ekstra aktiv rad, og kolonne, muliggjør utlesning av bildet fra motsatt kant, eller fra bunnen, uten at utlesningen starter med feil farvekombinasjon i kolonne og rad. Bildet kan altså speiles langs x- og y-aksen uten å endre farveinterpolasjonsalgoritmen.

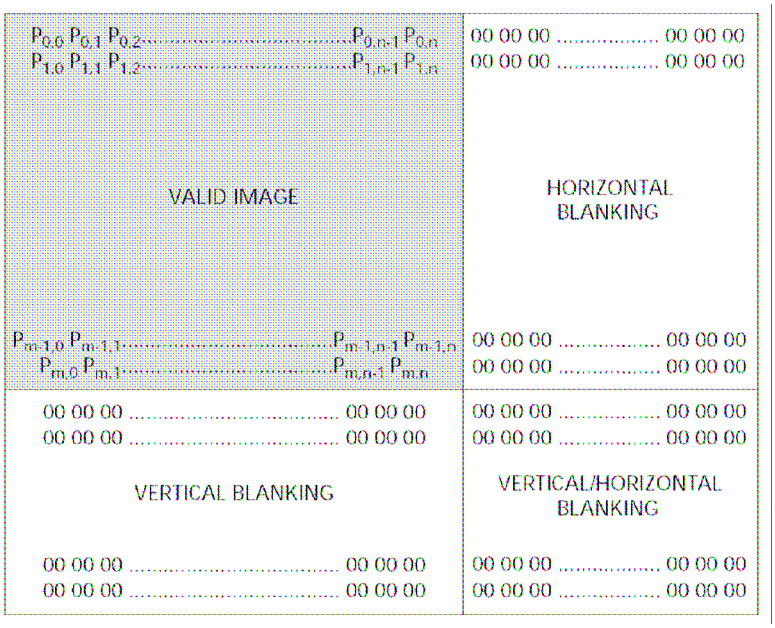
Sensorkjernen har såkalt *Electronic Rolling Shutter* (ERS) med *Progressive Scan* utlesning, som medfører at bildedata avleses, og overføres, linje for linje.[4)] Gyldige bildedata er omgitt av horisontal og vertikal *blanking* som i praksis er masterklokketikk uten gyldige data på brikkens databuss, se figur 3.1.1.6. Blanking er altså ikke et fysisk område på billedbrikken, men et tomgangsignal som sendes mellom hver bildelinje (horisontal blanking) og mellom hver bilderute (vertikal blanking).

3) Farveinterpolasjon av bilder i Bayer-mønster er beskrevet i Appendiks B

4) For nærmere beskrivelse av ERS, Rolling Shutter og beregning av integreringstid for piksler henvises til [4]



Figur 3.1.1. 5 Utsnitt av bildebrikke med farvefiltre ordnet i Bayer-mønster [4]



Figur 3.1.1. 6 Intervaller for vertikal og horisontal *blanking* – gyldige bildedata er indikert med skyggelagt område [4]

Blankingsignalet har sine røtter i den gamle analoge fjernsynsteknologien der elektronkanonen i et billedrør trenger tid til å returnere til vertikalt utgangspunkt mellom hver skannet linje, og tid til å returnere til bildets øvre høyre hjørne før avtasting av et nytt delbilde.

Lengden på vertikalt og horisontalt blankingintervall bestemmes av verdien til sensorkjernens registre Reg0x05 og Reg0x06. De endelige verdien for register Reg0x05 er en avveining mellom høyest mulig linjerate, for reduksjon av eventuell bevegelses-uskarphet i bildet, og tilstrekkelig tid til å oppdatere statisk adressebuss før overføring av påfølgende bildelinje til eksternt minne. Reg0x06 påvirker bilderate i *video-modus* og er uten betydning for vår anvendelse.

Billedbrikken setter LINE_VALID-port høy når gyldig billeddata befinner seg på utgående databuss, vist som skyggelagt område i figur 3.1.1.6. Denne porten brukes til å styre mikrokontrollerens avbruddsrutine som overfører en enkelt linje med billedata fra billedbrikke til ekstern minnebrikke. Utgangen FRAME_VALID brukes til å finne startpunktet for et bilde, og ligger lav i bildebrikkens vertikale blanking-intervall.

Se forøvrig kapittel 2.2 Programvare.

Tabell 3.1 viser et utdrag fra de grunnleggende tekniske spesifikasjonene for MT9V111. En *Automotive*-versjon av brikken er tilgjengelig, med temperaturområde -40 til +125 °C.

PARAMETER	VERDI
Optisk format	1/4-tomme (4:3)
Aktivt billedareal	3.58mm(H) x 2.69mm(V) 4.48mm (Diagonal)
Antall aktive billedelementer	640H x 480V (VGA)
Piksel-størrelse	5.6µm x 5.6µm
Optisk farvefilter	RGB Bayer-mønster
ADC oppløsning	10 bit, on-chip
Følsomhet	1.9 V/lux-sek (550nm)
Dynamisk område	60dB
SNR_{maks}	45dB
Forsyningspenning	2.8V +0.25V
Effektforbruk	<80mW ved 2.8V og 12MHz klokkefrekvens
Temperaturområde	– 20 °C til +60 °C
Kapsel	44-Ball ICSP (BGA)

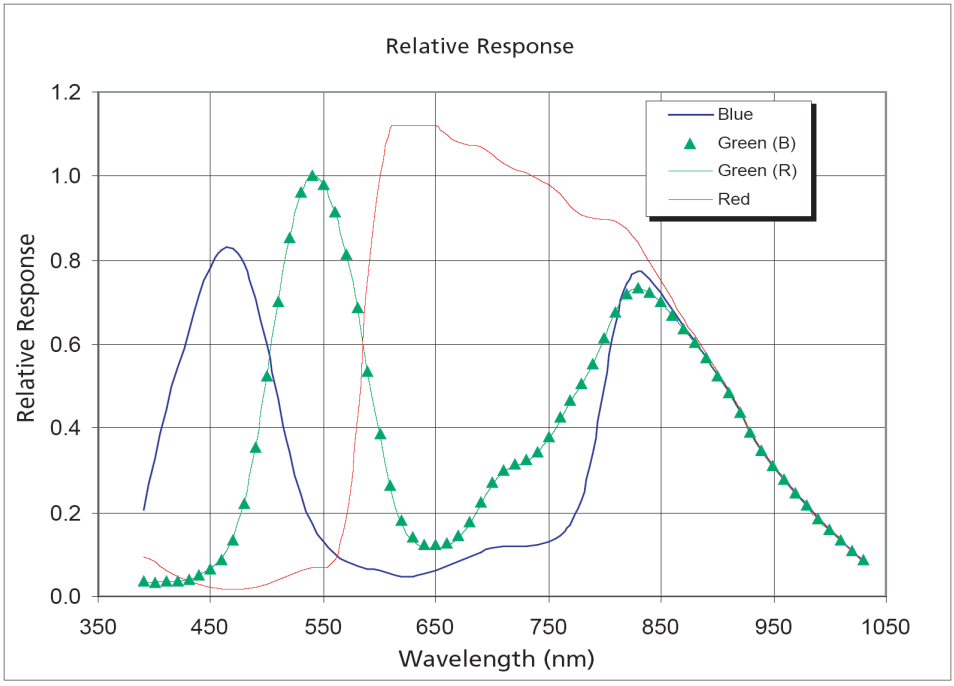
Tabell 3. 1 Grunnleggende tekniske spesifikasjoner for MT9V111 [4]

Figur 3.1.1.7 viser MT9V111s spektralfølsomhet.[5] Merk at vi må filtrere vekk infrarødt, fordi alle pikselgruppene er følsomme for NIR (*Near InfraRed*). Med et "IR-pass/visuell-blokk" filter er MT9V111 godt egnet for akromatisk NIR-foto, som gir bedre gjennomtrengning under disige forhold [1].

Micron oppgir at den termiske mørkestrøm-støyen avtar med 3dB/8Kelvin. Dette tilsvarer en støyreduksjon på 30dB hvis arbeidstemperaturen for brikken senkes med 80 Kelvin, eller 80 grader Celsius, f.eks. fra +60 til -20 grader Celsius. Termisk kobling mellom billedbrikken og

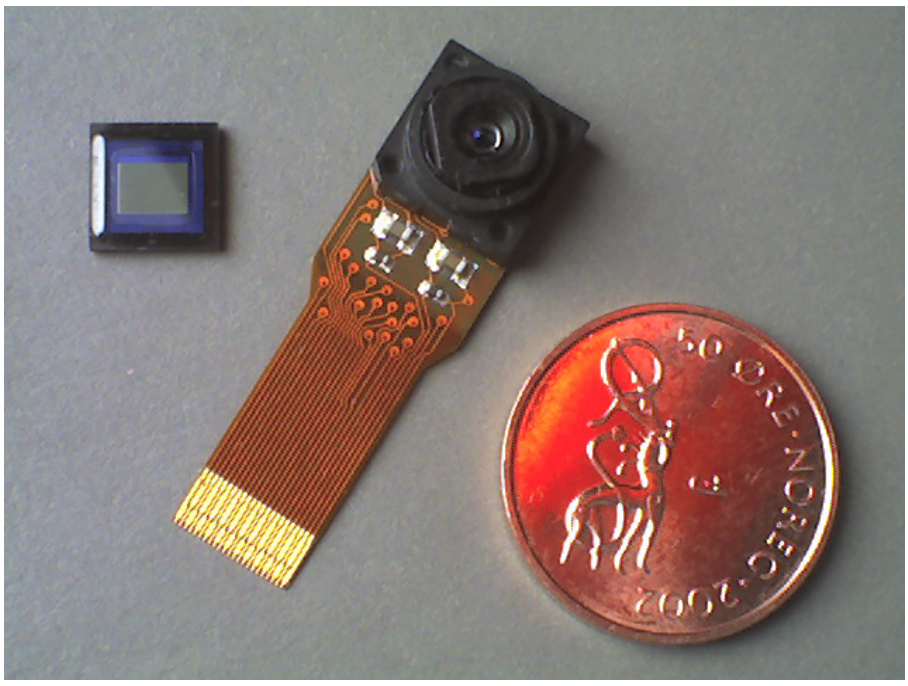
5) Farvesyn og sensorteknologi er behandlet i prosjektoppgavens kapittel 3 [1].

satellittplattformens ytre struktur bør derfor utredes. Ved fotografering av svært lyssvake objekter kan det da dras nytte av nedkjøling i den delen av baneløpet som er i jordskyggen.



Figur 3.1.1. 7 Respons-funksjoner for MT9V111s røde, grønne og blå pikselgrupper [4]

Micron MT9V111 blir levert i to utførelser, henholdsvis BGA (Ball Grid Array) og *bonded-DIE* på fleksiprint med integrert optikk, se figur3.1.1.8.

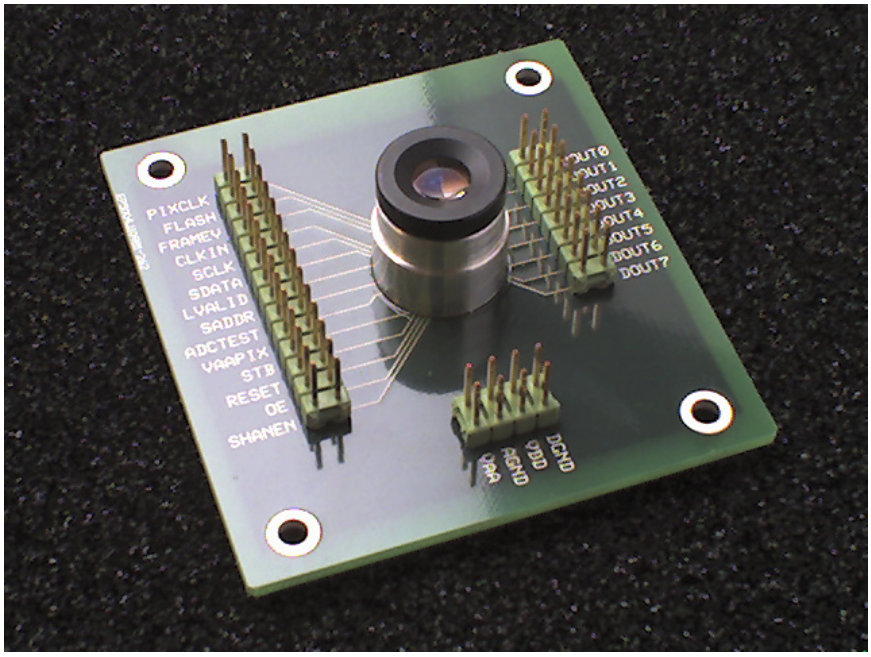


Figur 3.1.1. 8 Micron MT9V111 som henholdsvis BGA og bonded-DIE på fleksiprint

I vakuum vil de fleste materialer i større eller mindre grad avgi gass eller ioner ved oppvarming. Dette fenomenet har fått samlebetegnelsen *utgassing*. Et eksempel er PVC (Polyvinylklorid) som avgir svært aggressive klor-ioner. Innledende tester indikerer at *Bonded-DIE*-versjonen av MT9V111 har optikk-fatning av PVC. (Tekniske data om fatningsmaterialet er ikke tilgjengelig da Bonded-DIE-modulen er levert av hemmelig tredjepart) BGA-versjonen har optisk vindu av borsilikat som ikke gir nevneverdig utgassing i det temperaturområdet brikken opererer, og er forøvrig fremstilt av samme materialer som andre, romkvalifiserte kretskomponenter.

Alle enheter ombord i en satellitt må kunne avgi overskuddsvarme i vakuum uten gravitasjon, og dermed atmosfæriske konveksjonsstrømmer. BGA-montasje gir svært god termisk kobling til kretskortet.

BGA-kapslet utgave av Micron MT9V111 billedsensor er valgt for implementering i kameramodulen.[6)]



Figur 3.1.1. 9 Tilkoblingskort SC-1A med terminaler for alle kontaktpunkter på MT9V111-BGA

Figur 3.1.1.9 viser SC-1A, et kort med tilkoblingsterminaler for alle kontaktpunkter på MT9V111-BGA. SC-1A er tilpasset sammenkobling med STK500 utviklerverktøy fra Atmel.

6) Problemstillinger knyttet til montering og romkvalifisering av BGA-kapsler er behandlet i prosjektoppgavens kapittel 5.1.2 [1]

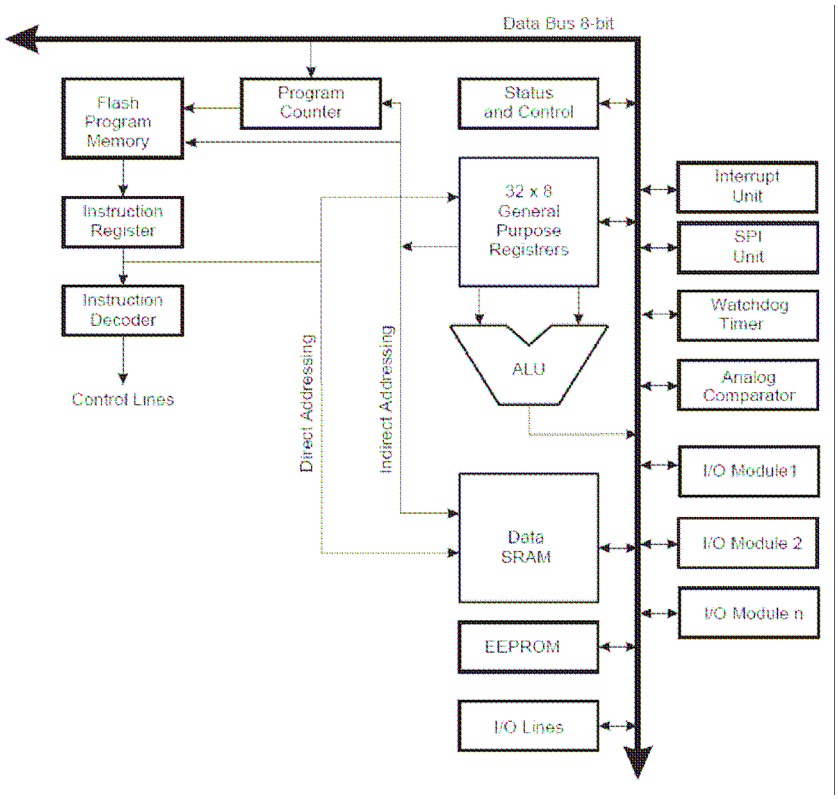
3.1.2 Mikrokontroller

En mikrokontroller skiller seg fra en mikroprosessor ved at førstnevnte har såkalte *perifer-enheter* implementert direkte på brikken. Der en mikroprosessor trenger eksterne støttekretser som linjedrivere, minnekontroll og ADC, er disse en integrert del av silisiumbrikken til en mikrokontroller. Av historiske årsaker omtales disse enhetene fortsatt som perifer-enheter selv om de i en mikrokontroller altså befinner seg internt i kretskomponenten.

Programminne i form av EEPROM og/eller en flash -partisjon kan også være implementert på brikken. Noen brikker har statisk skrive/lese-minne (SRAM) som gjør mikrokontrolleren i stand til å kjøre maskinkode kompilert fra programmeringsspråkene C og C++.

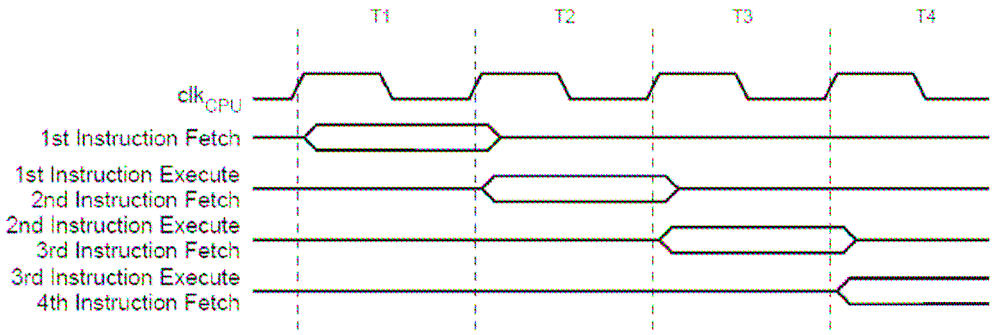
Perifer-enhetene styres direkte med egne instruksjoner, eller aktiveres ved å skrive til dedikerte registre. Et eksempel er seriell kommunikasjon via UART der dataordet som skal overføres skrives til et bestemt register. Maskinvaren tar seg av overføring av alle informasjons- og protokollbit uten å legge beslag på klokkesykler i prosessorkjernen. Dette reduserer kildekodeomfanget betraktelig sammenlignet med kode uten støtte i maskinvare. Enkelte operasjoner, som ekstern minneaksess, utføres også raskere med maskinvarestøtte, enn når rutinen er implementert som ren programkode.

Alle mikrokontrollere i Atmels 8 bits AVR-familie er basert på samme kjerne. De deler også samme grunnsett av instruksjoner. Avhengig av tilleggsfunksjoner har noen av versjonene utvidede instruksjonsett. Et eksempel er multiplikasjon utført på to klokketikk med maskinvarestøtte i ALU (Aritmetic Logic Unit – del av prosessorkjernen der aritmetiske og logiske operasjoner utføres). Figur 3.1.2.1 viser et forenklet blokkdiagram av en AVR-kjerne [6].



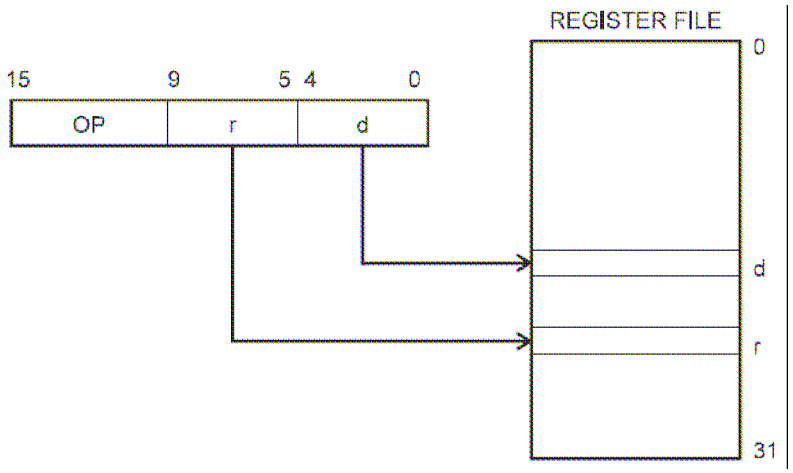
Figur 3.1.2. 1 Blokkdiagram for AVR-arkitekturen [6]

AVR-kjernen har Harvard-arkitektur med atskilt program- og dataminne. Fordi instruksjons-bussen er uavhengig av databussen kan programinstruksjoner eksekveres i en såkalt *pipeline* der neste instruksjon hentes fra programminnet samtidig med eksekvering av instruksjonen som befinner seg i instruksjonsdekoderen. Dermed kan en instruksjon utføres *hvert* klokketikk, som vist i figur 3.1.2.2 [6]. Dette kalles gjerne en *ett-nivå pipeline*.



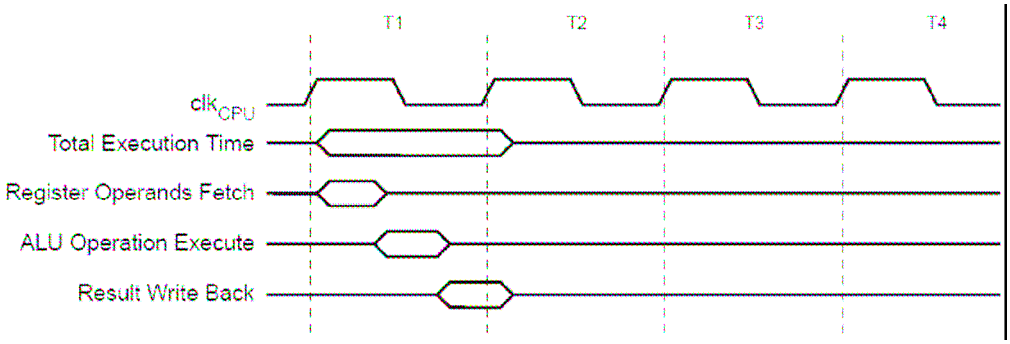
Figur 3.1.2. 2 Tidsdiagram for parallell eksekvering, og opphent av ny, instruksjon [6]

Separat databuss og instruksjonsbuss gjør det også mulig å ha ulik ordlengde i programminne og dataminne. AVR har 8 bits arbeidsregistre og en 8 bits intern databuss. Arbeidsregisterfilen har 32 minneadresser, som i et instruksjonsord tildeles et 5 bits adresseord. Data i arbeidsregisteret kan aksesseres direkte fra ALU. Instruksjonsbuss og programminne har 16 bits ordlengde som i *direkte register-adressering* modus kan romme operand- og destinasjonsadresse, sammen med operasjonskode, i ett enkelt instruksjonsord. Figur 3.1.2.3 viser et eksempel på et instruksjonsord med to registeradresser og operasjonskode. Begge operander kan delta i en aritmetisk-logisk operasjon i ALU. Resultatet skrives til destinasjonsregisteret. Hele operasjonen utføres i løpet av en enkelt klokkesyklus, som vist i figur 3.1.2.4.



Figur 3.1.2.3 Eksempel på 16 bits instruksjonsord med operasjonskode og operandadresser [7]

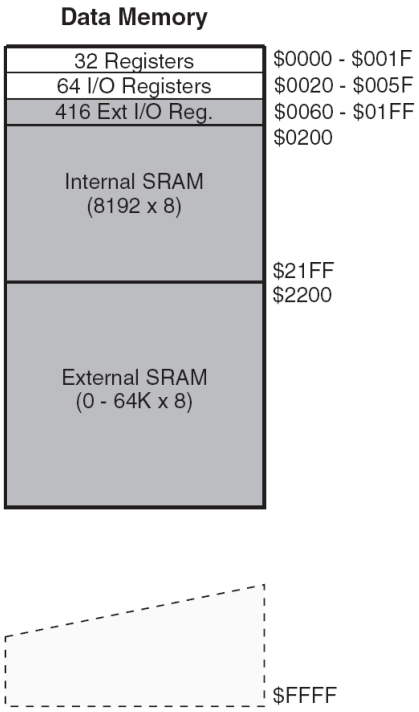
Avhengig av adresseringsmodus krever noen instruksjoner mer enn ett instruksjonsord, og tar derfor mer enn en klokkesyklus å utføre, bl.a. fordi programminnebussen bare kan overføre ett enkelt 16 bits ord av gangen [9]. For detaljert beskrivelse av de ulike adresseringsmodi med tilhørende instruksjonsformat henvises til ”AVR Instruction Set Nomenclature” [7].



Figur 3.1.2. 4 Tidsdiagram for en-klokkesykel eksekvering av aritmetisk-logisk instruksjon [6]

I datablad og annen dokumentasjon for AVR-kretsene er programminnets størrelse oppgitt i antall byte. En minneadresse i instruksjonsminnet peker på en 16 bits minnelokasjon fordi instruksjonsbuss og programminne har 16 bits ordlengde. Antall instruksjoner det er plass til i programminnet er derfor maksimalt halvparten av programminnestørrelsen oppgitt byte. 128kB minne rommer altså 64k instruksjoner, eller færre hvis instruksjoner som opptar to minnelokasjoner blir brukt. Det er viktig å ta hensyn til dette ved beregning av nødvendig størrelse på programminnet.

ALU kan ikke skrive og lese data direkte til og fra GPIO (General Purpose In/Out, inn- og utganger til generell bruk) og andre registre som ligger i det såkalte I/O-register området. Dataord må i stedet mellomlagres i ett av de 32 arbeidsregistrene og overføres til og fra I/O-registeret med instruksjonene IN og OUT. I/O-registerområdet har 64 adresseposisjoner representert med en 6 bits adresse i instruksjonsordene til IN og OUT. Instruksjonene IN og OUT kan derfor utføres i løpet av en enkelt klokkesyklus. I kameramodulen er mellomlagring av dataord i arbeidsregister brukt ved overføring av datastrøm fra bildebrikken til bildeminnet. Data blir ikke behandlet i ALU, som av den grunn ikke er direkte involvert i overføringen av bildedata.



Figur 3.1.2. 5 Adresserom for dataminne, med adresser for minnepartisjonenes grenser [6]

De 32 arbeidsregistrene og I/O-registerfilen kan også nåes med instruksjoner for generell dataminneaksess. Figur 3.1.2.5 viser hele adresserommet for dataminnet, med adresser for bruk i direkte og indirekte *data-adresseringsmodus*. Merk at I/O-registrenes adresser er forskjøvet med 0x20 i forhold til adresser brukt med de I/O-spesifikke instruksjonene IN og OUT. Arbeidsregistrene er tildelt data-adressene 0x0000 – 0x001F. Adresserommet for de 416 *eksterne* I/O-registrene, som ikke kan aksesseres med IN og OUT, inneholder bl.a. dedikerte registre for periferenheter.

De seks øverste arbeidsregistrene, R26-R31, kan sammenkjedes til tre 16 bits registre, kalt X, Y og Z. Disse registrene brukes bl.a. ved skrive- og leseoperasjoner i eksternt tilkoblet RAM. Fysisk implementering av ekstern RAM-aksess er beskrevet i kapittel 2.1 Fysisk krets. Instruksjoner og programkode er beskrevet i kapittel 2.2 Programvare. Forøvrig vises til [6].

AVR-mikrokontrolleren er implementert på et *monolittisk substrat* der flashminne er integrert på samme silisium-brikke som resten av kontrolleren. Dette gir svært billige og robuste kretser. Flashminne stiller andre krav til silisiumbrikken enn logikkretser brukt i ALU og periferenheter og det må inngåes kompromisser i substratmaterialet. Resultatet er generell begrensning i klokkehastighet for alle AVR-brikker.

Til flightmodel SC-2 er AVR Mega1281 valgt. Dette er en er en ny versjon av mikrokontrolleren Mega128, som ble brukt i den norske studentsatellitten nCube.

AVR og Mega-serien står forøvrig på anbefalt-listen til Cubesat-prosjektet [3].

Mega1281 tilhører siste generasjon av AVRs Mega-familie, og har utvidet område for driftspenning samt flere perifer-enheter enn forgjengerne. Kretsene leveres i 64 og 100 pinners utgave sistnevnte med flere GPIO, USART, interupt-innganger og timere. Standardutgaven aksepterer driftspenning fra 2.7- 5.5V. Den såkalte ”V”-serien har utvidet spenningsområde fra 1.8 – 5.5V, men med betydelig redusert maksimal klokkefrekvens i forhold til standardutgave.

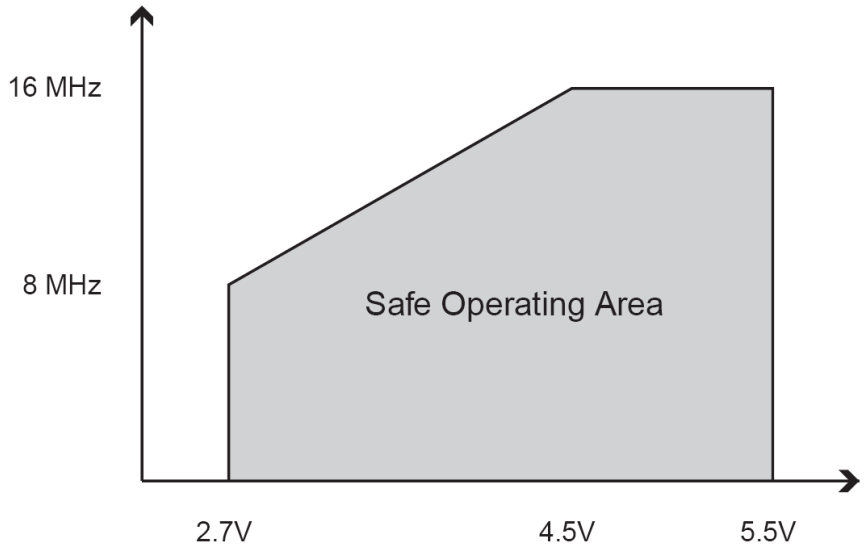
Mega1281 leveres i 64 pinners kapsel og har 128kB flashminne. Kretsen kan også leveres med 64kB og 256kB flashminne. 64k-utgave leveres kun i 100 pinners kapsel. Alle kretsene er i binærkompatible. Kode kan flyttes mellom kretsene så lenge den ikke overskrider minnestørrelse.

For detaljert beskrivelse av grensesnitt, porter og periferenheter med støtte i maskinvare, samt oversikt over produktnavn og versjoner, henvises til datablad [6].

Under arbeidet med SC-1B var 2 eksemplarer av Mega2561 i *engineering sample*.[7)] tilgjengelig. Mega2561 har 256kB flashminne, leveres i 64 pinners kapsel og er for øvrig binærkompatibel med Mega1281.

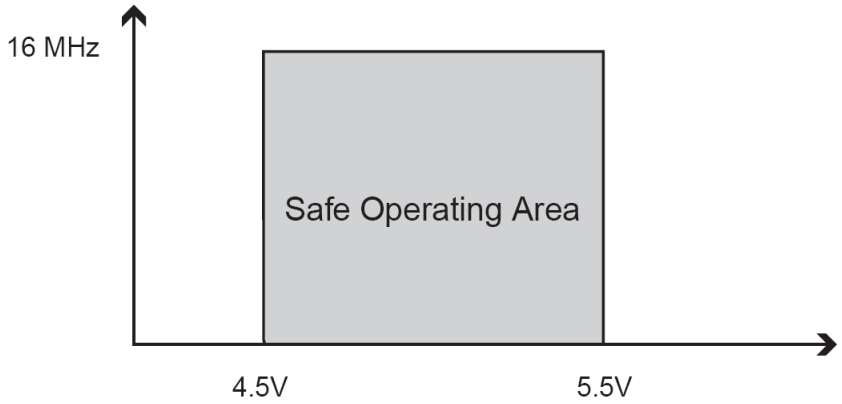
Tidligere revisjoner av datablad har indikert at Mega2561 vil fungere med driftspenning ned til 2.7V – da med 8MHz som maksimal klokkefrekvens, altså samme spesifikasjon som Mega1281, med *”Safe Operating Area”* som vist i figur 3.1.2.6.

7) Engineering sample er en krets fremstilt i et mindre prøveopplag for testing og feilsøking.



Figur 3.1.2. 6 Område for stabil drift i henhold til spesifikasjon for AVR Mega1281 [6]

Ifølge siste revisjon av datablad kan Mega2561 ikke drives med lavere forsyningsspenning enn 4.5V. Figur 3.1.2.7 viser revidert utgave av såkalt *"Safe Operating Area"* for Mega2561.

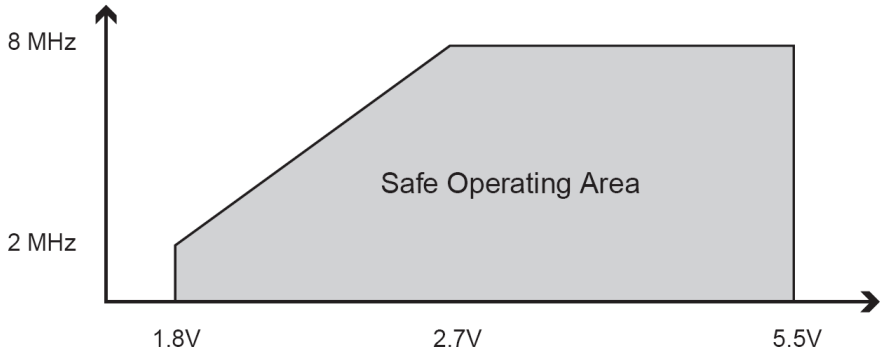


Figur 3.1.2. 7 Område for stabil drift i henhold til revidert spesifikasjon for AVR Mega2561 [6] Rev. E

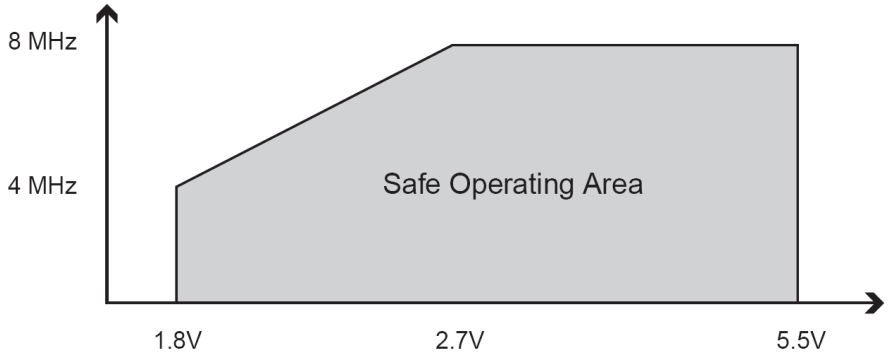
Figur 3.1.2.8 viser at lavvoltutgaven Mega2561V, med 256kB flashminne, har vesentlig lavere maksimal klokkefrekvens ved lave driftspenninger enn tilsvarende for Mega1281V med 128kB flashminne vist i figur 3.1.2.9, henholdsvis 2MHz og 4MHz ved 1.8V.

Selv om dette siste eksempelet gjelder lavvoltutgaven av kretsene skal det i følge representanter for Atmel generelt være problematisk å oppnå fullt klokkesving på databussen til flashminnet ved stor last som følge av utvidet programminne. Siden dette er problemstillinger knyttet til Atmels løpende utviklings- og revisjonsarbeid for kretsen foreligger det ingen skriftlige referanser. De to tilgjengelige eksemplarene av Mega2561 er testet for lavspenningsoperasjon v.h.a. et enkelt program som aktiverer store deler av prosessorkjernen ved å telle opp et register. For kontroll ble resultatet vist ved å tenne og slukke lysdioder på et STK500-kort.[8)]

8) For beskrivelse av utviklerverktøyet STK500 fra ATMEL henvises til [27]



Figur 3.1.2. 8 Område for stabil drift i henhold til spesifikasjon for AVR Mega2561V (lavvoltsutgave) [6]



Figur 3.1.2.9 Område for stabil drift i henhold til spesifikasjon for AVR Mega1281V (lavvoltsutgave) [6]

Begge brikker fungerte stabilt ned til 1.7V ved 24MHz – noe som er en betydelig overklokking av kontrolleren i forhold til oppgitte spesifikasjoner, selv for Mega1281. Kontrollerne stoppet opp ved 1.6V – og trengte da 1.8-1.9V for å starte igjen uten fysisk reset. Med fysisk reset startet brikkene ved 1.7V. Dette indikerer egenskaper langt utenfor oppgitt spesifikasjon. To mulige forklaringer kan være:

1) For å redusere andel underkjente kretser fra en produksjonserie, og dermed øke lønnsomheten, oppgis vesentlig dårligere garanterte spesifikasjoner enn faktisk ytelse for den gjennomsnittlige krets. Siden det for kameraets del ikke dreier seg om volumproduksjon kan flere kretser fra ulike produksjonserier kjøpes inn og testes, og enkeltkretser med egenskaper bedre enn de oppgitte spesifikasjoner kan velges ut. Dette er vanlig praksis ved lavvolum-produksjon av høykvalitets teknologi.

2) Lavspenningoperasjon på høy klokkefrekvens er for Mega1281/2561 først og fremst begrenset av nøyaktigheten til ADC-innganger, samt drift av den interne spenningsomformeren som leverer oppkonvertert spenning til selvprogrammering av flashminnet [10]. Kameramodulen benytter seg ikke av mikrokontrollerens ADC da all konvertering av analoge bildedata til digital datastrøm foregår internt i bildebrikken. Selvprogrammering av programminnet er en teknikk som brukes ved lokal oppdatering av programkode. Et eksempel er oppdatering av styringsenheten til en brusautomat ved endring av priser, sortiment eller myntenheter. Kameramodulens programminne vil av pålitelighetshensyn være låst for selvprogrammering. Det vil ikke bli foretatt oppdateringer av flashminnet i jordbane. Det er altså ikke aktuelt å bruke selvprogrammering av flashminnet.

Innledende tester av kameraprototypen er gjort med prøve-eksemplarer av Mega2561. Mega1281 forventes å tåle overklokking like godt, eller bedre enn Mega2561, fordi den har et mindre flashminne.

På bakgrunn av ovenstående ansees Mega1281 som egnet for moderat overklokking.

Skulle det vise seg i senere pålitelighetstester av ferdig kameramodul at overklokking reduserer mikrokontrollerens stabilitet kan kameramodulens masterklokkefrekvens reduseres slik at den faller innenfor kontrollerens spesifikasjon for gitt driftspenning. Dette kan gjøres uten andre konsekvenser enn moderat økning av eksponeringstid som gjør kameraet noe mer følsomt for bevegelsesuskarphet.

Mega640 har 64kB flashminne og er i teorien derfor bedre egnet for overklokking, men kretsen leveres bare i en lite robust 100 pinners kapsel. Mega640 er derfor inntil videre ikke en aktuell kandidat for bruk i SC-2.

Motivasjon for valg av Atmels AVR mikrokontroller kan oppsummeres slik:

- enkel, stabil og robust teknologi med 8 bits arkitektur egnet for kameramodulens 8 bits dataordlengde

- effektiv minnekontroll med 16 bits adressebuss implementert i maskinvare – tilstrekkelig adressebussbredde for ubrutt utlesing av hel bildelinje fra bildebrikke

- stor bredde i utvalg og kompleksitet - vi kan velge en tilnærmet skreddersydd kontroller for ulike formål direkte fra Atmels standardsortiment

- ”System on Chip” filosofi der alle nødvendige eksterne grensesnitt er implementert i mikrokontrolleren, med støtte i maskinvare

- de nyeste brikkene fra AVR har svært bredt driftspenningsområde og kan kjøres på relativt høye klokkefrekvenser ved lave spenninger

- svært godt utviklerverktøy der store deler av utviklermiljøet er gratis eller svært rimelig

- brukt i studentsatellitten nCube - sannsynlig at AVR vil være gjennomgående mikrokontroller-teknologi i det neste Cubesat-prosjektet til strømforsyning, initiering av radiokretser o.l.

- Atmels AVR-avdeling holder til i Trondheim og har god kontakt med fagmiljøet IET-NTNU - teknologi og brukerstøtte er lett tilgjengelig

- det er naturlig å anvende norsk teknologi i et rekrutteringsprosjekt for norske universiteter

3.1.3 Avkoblings-kondensatorer

Serieinduktans i kretskortet kan føre til kortvarige lokale spenningsfall i forsyningspenning, som følge av varierende last når digitale porter i de enkelte kretskomponenter skifter logisk nivå. Dette kan forskyve komponentenes logiske desisjonsnivåer, og i verste fall medføre logiske låsninger med oppbrente kretser som resultat.

For avkobling av støy i kretsen står valget i praksis mellom keramiske- og tantalkondesatorer. Elektrolyttkondensatorer er uegnet grunnet utgassing i vakuum fra den flytende elektrolytten. For de keramiske kondesatorene er det dessuten et valg mellom ulike dielektrika, med aveiing mellom høy kapasitans i lite volum og stabil kapasintans ved temperatursvingninger [11]

Tantalkondensatorer har betydelig serielekasjestrøm, typisk i uA-området for de kapasitanser og kapselstørrelser det er aktuelt å anvende i kameramodulen. Lekasjestrømmen øker med økende temperatur og er typisk 10x større ved +85 °C. enn referanse verdi oppgitt i datablad, og målt ved 25 °C. Ved 125 °C er lekasjestrømmen 12x større. Typisk lekasjeresistans for en tantalkondensator er i størrelsesorden 50MΩ, for en keramisk kondensator kan lekasjere-sistansen være større enn $10^{12}\Omega$ [11]

Tantalkondensatorer har risiko for såkalt "katastrofal feil" der kondesatoren kortslutter helt. Dette vil i mange tilfeller bety at også kretsens strømforsyning kortsluttes fullstendig, og blir ødelagt. Den ødelagte kondensatoren vil ha en seriemotstand typisk 10^{1}-$10^{4}\Omega$, og kan dermed overopphetes og ødelegge kort og omliggende komponenter. Det siste er spesielt kritisk i romsammenheng der det ikke finnes atmosfære og konveksjonstrømmer. All kjøling stammer fra dispersjon og varmeledning til andre deler av satellitten, og det termisk regnskapet for hver enkelt modul må overholdes strengt.

Tantalkondensatorer med innebygget kortslutningsbeskyttelse i form av en sikring er kommersielt tilgjengelige, men en slik utkobling medfører samtidig at kretsen mister støyavkobling i dette kretspunktet.

Tantalkondensatoren er en såkalt polarisert komponent og tåler ikke høye spenninger i reversert retning.Ved 25 °C tåler tantalkondensatoren 15% av nominell spenning (merke-spenningen) i bakoverretning, altså -1.5V for en 10V kondensator. Ved 85 °C er tillatt bakoverspenning sunket til 5%, og ved 125 °C 1%. Bakoverpenning kan f.eks. oppstå ved ringeeffekter (resonans) som følge av induktanser i kretskort, og andre i komponenters tilførselsledere. Statiske utladninger i forbindelse med håndtering, og under transport opp i rommet, kan også påføre kondensatoren bakoverspenning.

Tradisjonelt har tantalkondensatorer blitt brukt der kapasitanser større enn 1µF er nødvendig. Med utvikling av nye dielktrika, og forbedrede produksjonsmetoder, er keramiske kondensatorer med kapasitans >10µF kommersielt tilgjengelig. Disse kondesatorene har forhold mellom volum og kapasitans i samme størrelsesorden som tantalkondensatorer med samme nominelle spenning [11].

Både tantal og keramiske kondensatorer fra ulike produsenter er tatt med i "European Preferred Parts List" (EPPL) som utarbeides av ESCC (European Space Components Coordination) [13]. Det er altså ingen fundamentale utgassings-problemer knyttet til bruk av tantalkondensatorer i rom-sammenheng.

Kraftige, lavfrekvente fluktuasjoner i last er ikke ventet, da kameramodulen ikke har strømkrevende moduler som blir skrudd av og på. Høye kapasitanser nær tilførselspinner for den enkelte kretskomponent i kameramodulen er derfor ikke nødvendig.

Med bakgrunn i ovenstående er keramiske kondensatorer valgt, også om større kapasitanser (>1uF) viser seg å være påkrevd.

Fysisk realiserte kondensatorer er ikke konstante entiteter med lineære egenskaper. Kapasitans og frekvensavhengig impedans er resultatet av samspillet mellom kondensatorens grunnkapasitans, induktive egenskaper i tilkoblingsledere og kapsel, ulinearitet og tap i dielektrikum, samt kapasitive og induktive egenskaper i kretskort [11] [12] [14] [15] [16]. Dette samspillet er det svært krevende å simulere, og vanskelig å beregne med enkle verktøy. En praktisk infallsvinkel er å gjøre visse forutgående antagelser basert på diagrammer for enkeltkomponeters egenskaper og deretter utføre omfattende målinger på ferdig implemetert kretskort for å sikre seg mot fatale, ikke-intuitive avvik.

For avkobling av støy i forsyningsspenning er det vanlig ingeniørpraksis å koble to kondensatorer, en med relativt stor kapasitans og en med 1-3 dkader mindre kapasitans, i parallell. En vanlig misforståelse er at kondensatorens avkoblingsevne i den høfrekvente delen av spekteret utelukkende er avhengig av, og omvendt proporsjonal med kapasitans. En kondesator med relativt liten kapasitans er altså antatt å være bedre egnet til avkobling av høyfrekvent støy enn en tilsvarende kondensator med høyere kapsitans. I virkeligheten er kondensatorens praktiske egenskaper, spesielt ved frekvenser høyere enn det såkalte *resonanspunktet*, avhengig av kapselens størrelse, fysisk utforming og valgt dielektrikum. Resonanspunktet er frekvensen der kondesatorens reaktans endrer seg fra kapasitiv til induktiv, og impedansen begynner å *øke* med frekvensen.

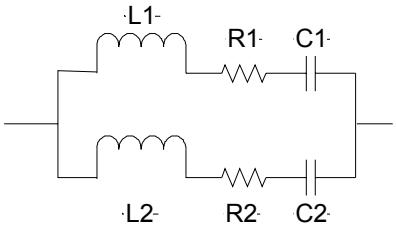
En forenklet kretsekvivalent til kondesatoren er vist i figur 3.1.3.1.

Figur 3.1.3. 1 Forenklet kretsekvivalent for kondensator, ESL er ekvivalent serieinduktans og ESR er ekvivalent serieresistans [12]

En kondesator kan, svært forenklet, beskrives som en kapasitans i serie med en induktans (ESL) og en ekvivalent serieresistans (ESR). ESR er et utrykk for tap i dielektrikum og ledere for tilkobling [15] [16]. Kondesatorens induktive egenskaper er bl.a. knyttet til kapsel og tilførselsledere. Mindre kapsel gir vanligvis mindre serieinduktans og dermed høyere frekvens for resonanspunktet [12]. Ved frekvenser høyere en kondesatorens resonansfrekvens er det de induktive egenskapene som dominerer, og impedansen *øker* med økende frekvens.

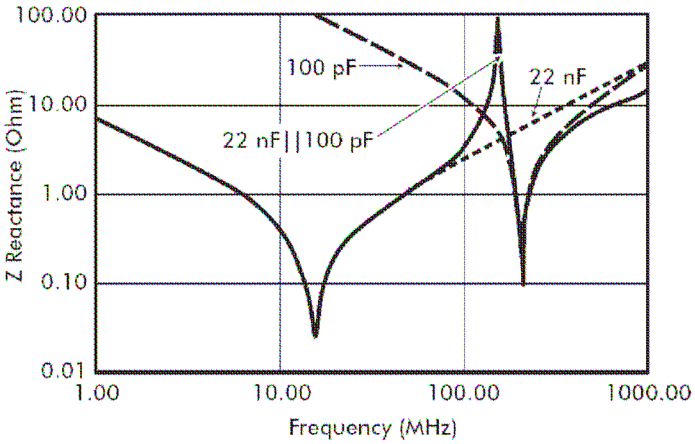
Det optimale er altså en kondesator med størst mulig kapasitans, i minst mulig kapsel med kortest mulig tilførselsledere, og som samtidig tåler DC-komponenten av den påtrykte spenningen.

Hvis to *ulike* kondesatorer kobles i parallell kan det oppstå uønsket resonans. Figur 3.1.3.2 viser en forenklet kretsekvivalent for to parallellkoblede kondesatorer.



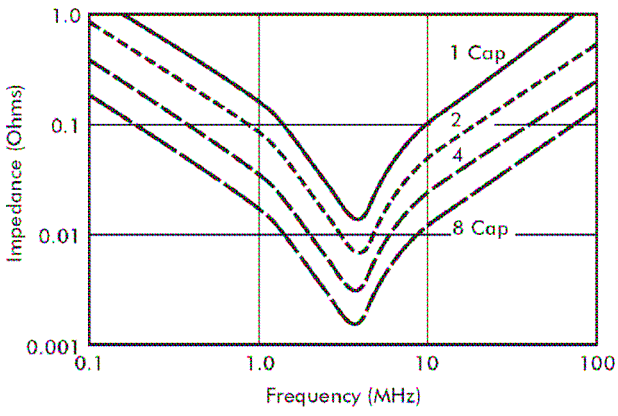
Figur 3.1.3. 2 Forenklet kretsekvivalent for to parallellkoblede kondesatorer

C1 er gjennom R1 og R2 koblet i parallell med L2; tilsvarende gjelder for C2 og L1. Figur 3.1.3.3 viser frekvensavhengig impedans for en 100pF kondensator, en 22nF kondensator og resulterende impedans ved parallellkobling av disse. Det oppstår et nullpunkt i nærheten av 200MHz hvis de to ulike kondensatorene kobles i parallell. Resultatet er at støy i dette frekvensområdet ikke blir tilstrekkelig avkoblet.



Figur 3.1.3. 3 Frekvensavhengig impedans ved parallellkobling av 100pF og 22nF kondensator [12]

Løsningen er å koble to eller flere *like* kondensatorer i parallell. Da vil samtidig den samlede serieinduktansen, gjennom parallelkobing av enkeltkomponentenes serieinduktanser, reduseres, og høyfrekvensegenskapene forbedres ytterligere. Figur 3.1.3.4 viser frekvensavhengig impedans for én enkelt kondesator samt parallellkobling av 2, 4 og 8 identiske kondensatorer [12].



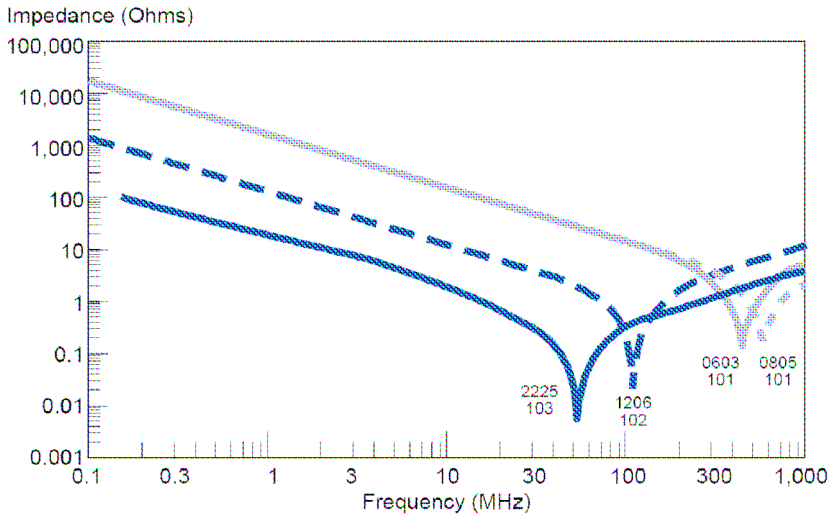
Figur 3.1.3. 4 Frekvensavhengig impedans ved parallellkobling av identiske kondensatorer [12]

Riktig praksis er derfor å velge en kondesator med tilstrekkelig avkobling ved høye frekvenser, og deretter koble identiske kondesatorer i parallell til nødvendig lavfrekvent avkobling er oppnådd. Påfølgende kontrollmålinger direkte på det ferdige kretskortet er nødvendig da kortets kapasitive (mellom lagene) og induktive (i kretsbanene) egenskaper vil virke inn [16]. Fordi parallellkobling av kondesatorer reduserer den samlede serieinduktansen kan en kondensator med litt for høy serieinduktans velges som utgangspunkt.

På selve kretskortet må kondesatoren plasseres nærmest mulig forsyningspinnene til de aktive kretskomponentene for å ungå at induktans i kretskortets baner skal begrense avkoblingen i de høyere frekvensområder. Overflatemonterte kapsler (SMD - Surface Mounted Device) er best egnet [12].

Kapselstørrelse for overflatemonterte keramiske kapsler oppgis vanligvis som et 4-sifret nummer, der de to første sifrene angir kapselens lengde i 1/100 tommer (0.254mm) og de to siste sifrene angir kapselens bredde på tilsvarende måte. En 1206-kapsel har altså et fotavtykk på kretskortet på omtrent 3.2x1.6mm. Høyden kan variere noe for de forkjellige modellene.

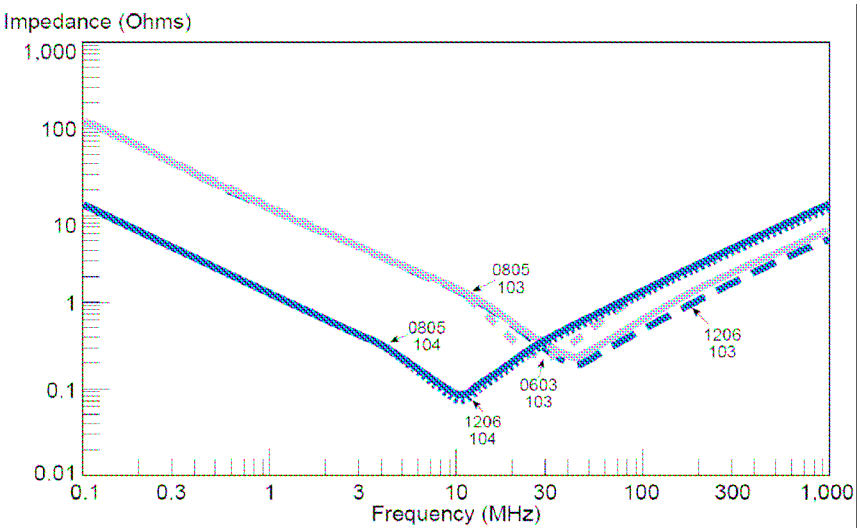
Figur 3.1.3.5 viser eksempler på frekvensavhengig impedans for keramiske kondesatorer med forskjellige kapasitans, og i ulike kapsler.



Figur 3.1.3. 5 Frekvensavhengig impedans for kondesatorer i ulike kapsler: 101=100pF; 102=1nF; 103=10nF, dielektrikum er C0G [11]

Figur 3.1.3.6 viser at en 100nF kondensator i 1206-kapsel har *lavere* impedans ved mikro-kontrollerens klokkefrekvens, (≈14MHz) som utgjør den grunnharmoniske komponenten i klokkestøyen, enn en 10nF kondesator i lik kapsel og med samme dielektrikum, og bare ubetydelig høyere impedans ved frekvenser for høyere ordens klokkestøy.

Ved instantane endringer av logisk nivå på port-drivere er det gunstig med høy kapasitans i kretskomponentens avkobling. En eller flere 100nF kondensatorer ved hver forsyningspinne synes derfor å være et godt utgangspunkt for innledende tester og målinger.

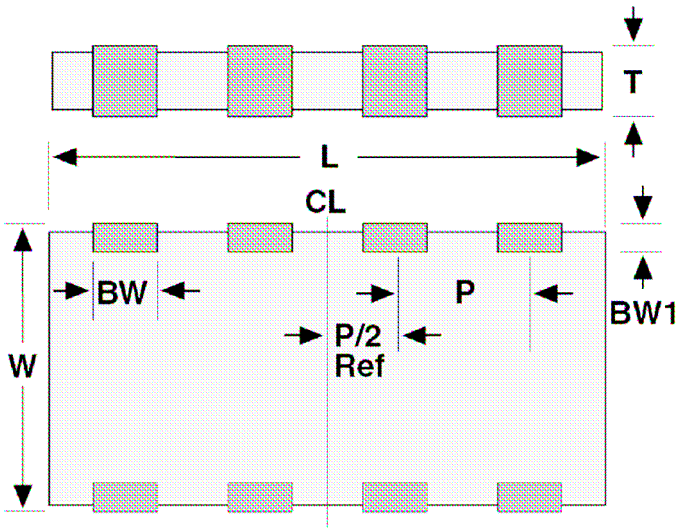


Figur 3.1.3. 6 Frekvensavhengig impedans for kondesatorer i ulike kapsler: 103=10nF, 104=100nF; dielektrikum er X7R [11]

På prototypekortet SC1-B er det valgt å lage plass til 2 stk. SMD kondensatorer ved hvert enkelt par av forsyningspinner. Målinger vil bli gjort med en og to kondensatorer. Loddepunkter på kortet, såkalte *pads*, er tilpasset kapsler i 1206-format, men det vil også være mulig å montere komponenter i 0805-størrelse.

Som utgangspunkt for målinger er det valg å avkoble hver forsyningspinne med en eller to 100nF kondensatorer. 1206 kapsler er valgt fordi disse er enklere å montere for hånd.

På flightmodel SC-2 vil det bli benyttet overflatemonterte kondesator-rekker med fire identiske kondensatorer i en enkelt 1206-kapsel, som vist i figur 3.1.3.7. Dette tilsvarer fire parallellkoblede kondesatorer i 0603 kapsel. Høyere kapasitans og mindre kapsel enn det som er anvendt i prototypen SC-1A/B gir ekstra margin for stabilitet.



Figur 3.1.3. 7 Fire identiske keramiske kondesatorer i en enkelt 1206-kapsel [11]

For detaljert beskrivelse av keramiske kondesatorers mekaniske oppbygging, og egenskapene til de ulike dielektrika henvises til [11].

3.1.4 Statisk minne – SRAM

Til eksternt bildeminne stod valget mellom BS62LV4006 fra BSI, og AS7C34096 fra Alliance [17] [18].

BS62LV4006 har spesifisert forsyningspenning 2.4V-5.5V, og nominell aksesstid 55nsek, men aksesstiden er redusert til 70 nSek i området 2.7-3.0V.

AS7C34096 som riktignok er implementert i 3.3V-teknologi, har spesifisert forsyningspenning 3.0V-3.6V og leveres i utgaver med12/15/20 nsek aksesstid. Kretsen kan også leveres i en 10nsek utgave, men krever da 3.15V forsyningspenning og kan derfor ikke brukes i kameramodulen.[9)]

Alliancekretsen er vesentlig raskere en kretsen fra BSI og gir større spillerom for propagasjonsforsinkelse i minnelås, samt toleranse for betydelig kortere intervall for gyldig data på mikrokontrollerens utgang ved planlagt overklokking som beskrevet i kapittel 3.1.2 [6].

Da billedbrikken har spesifisert forsyningspenning 2.8 +/-0.25V og dermed problemfritt kan forsynes med 3.0V er Alliance-kretsen valg gitt overlegne timing-egenskaper.[9)]

AS7C34096 har pinner for forsyningspenning sentrert på brikken (pinne 11, 12, 33 og 34) og dermed minimal banelengde for avkoblingskondensatorene på kretskortet.

Teknologien iAlliance AS7C34096 er forhåndstestet ved at AS7C31025B, en 128k byte (1024kbit) versjon av AS7C34096, har blitt montert på et Atmel STK501 testkort for Mega2561, som minne i testoppsettet før implementering av SC-1B kortet.

For detaljert beskrivelse og spesifikasjoner for AS7C34096 henvises til [18].

9) Porblemstillinger knyttet til valg av forsyningspenning er behandlet i Kapittel 4 Valg av forsyningsspenning

3.1.5 Ekstern adresselås (Latch)

Adresselås er valgt i henhold til anbefalinger i mikrokontrollerens datablad [6].
Valgt krets er SN74AHC573 fra Texas Instruments [19]. Kretsen har nominell forsyningspenning 2.0V – 5.5V (AHC-logikk) og en ende-til-ende propagasjonsforsinkelse mindre enn 10nsek. Kretsen leveres også i utgave med militær spesifikasjon med utvidet temperaturområde, da med typebetegnelsen SN54AHC573, men denne kretsen er vanskelig tilgjengelig grunnet amerikanske eksportrestriksjoner.

Kretsen SN74AHC373 har noe annerledes pinne-konfigurasjon men er for øvrig lik SN74AHC573 [20]. SN74AHC373 kan vise seg bedre egnet for det endelige kretskort-utlegget.

For detaljert beskrivelse av SN74AHC373 og SN74AHC573 henvises til [19] [20].

3.2 Sekundærkomponenter

Sekundærkomponenter er kretskomponenter som ikke skal være med i den endelige *flightmodel* av kameramodulen men som er brukt i protoypemodulen som støtteteknologi, for eksempel strømforsyning og driver for eksternt seriegrensesnitt. Disse komponentene er ikke kritiske med henblikk på temperaturområde og overlevelsesevne i jordbane. Strømforbruk og varmeutvikling er også uten praktisk betydning. Filosofi for valg av sekundære kretskomponenter er å overdimensjonere for å oppnå maksimal stabilitet for støttefunksjonene, slik at unødig feilsøking unngåes.

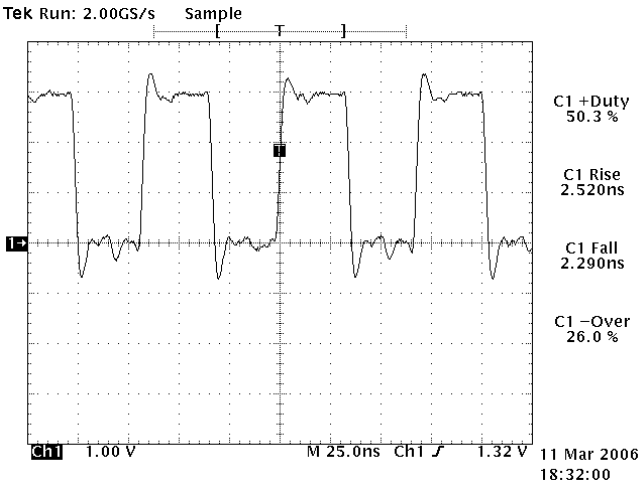
3.2.1 Krystall-oscillator til prototyp

For å begrense kompleksitet og antall frihetsgrader i eventuell feilsøking er det i prototypemodulen brukt en ekstern krystalloscillator med utgangsdrivere. Oscillatoren har også en programmerbar frekvensdeler slik at klokkesignal med frekvensen $F_0/2^n$ (der n et heltall) kan tappes ut i tillegg til masterklokke med frekvensen F_0. Signalet $F_0/2$ trengs i denne implementering som masterklokke for MT9V111 bildebrikke.

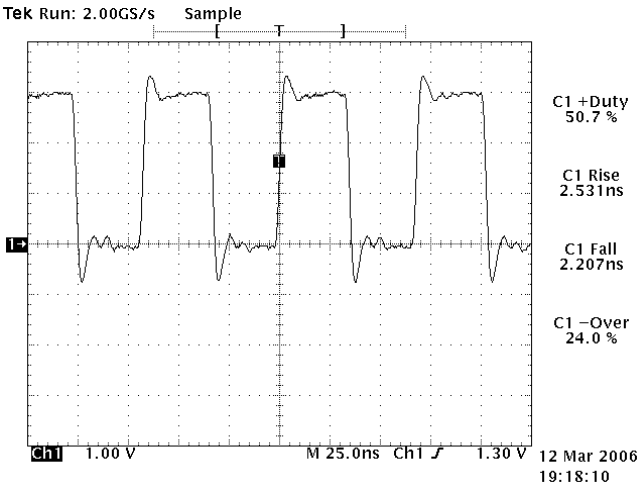
Krystalloscillatoren EXO-3 er en standardkomponent som leveres av flere ulike produsenter. Stige- og fall-tid er i datablad konservativt oppgitt til 15nsek, mens såkalt *Duty Cycle*, som er innbyrdes forhold mellom lengden på høyt og lavt intervall av klokkepuls, er oppgitt til 40/60%.[21]. Bildebrikken MT9V111 krever i henhold til datablad klokkesignal med stige- og falltid på 2 nsek og 50/50% Duty Cycle, med maksimal asymetri 45/55% ved lave til moderate masterklokkefrekvens [4].

EXO-3 har oppgitt nominell driftspenning 3V-6V [21]. Andre produsenter av samme krets oppgir forsyningspenning 4V-6V [22]. Kretsen er testet og viser seg å fungere tilfredstillende med driftspenning ned til 2.4Volt, som gir et spillerom på 0.6 Volt i forhold til prototypemodulens forsyningspenning på 3.0V +/-0.05V [23].

To kretser fra ulike produksjonserier og leverandører er testet. Figur 3.2.1.1 viser målinger av signal fra F_0-port til EXO-3, 14.7456MHz produsert av KSS Kinseki, og levert av komponenthuset Farnel. Figur 3.2.1.2 viser tilsvarende måling på EXO-3, 14.7456MHz produsert av IQD, og levert av ELFA.

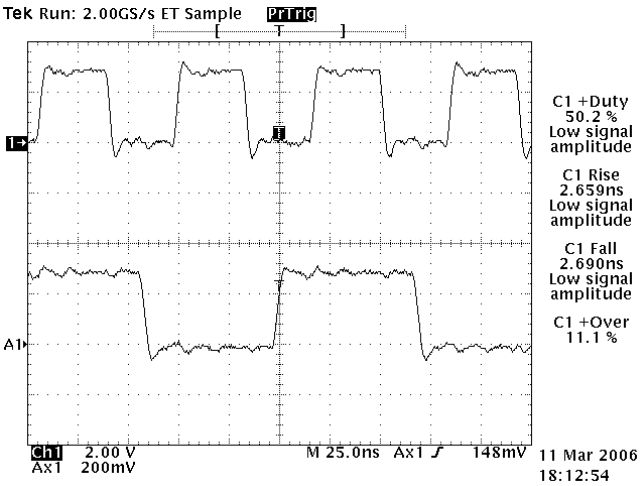


Figur 3.2.1. 1 Signal fra F_0-port - EXO-3, 14.7456MHz produsert av KSS Kinseki



Figur 3.2.1. 2 Signal fra F_0-port - EXO-3, 14.7456MHz produsert av IQD

Figur 3.2.1.3 viser inbyrdes faser for F_0 og $F_0/2$ for kretsen fra KSS Kinseki.



Figur 3.2.1. 3 Innbyrdes faser for F_0 og $F_0/2$ for EXO-3 produsert av KSS Kinseki

Stige- og fall-tid er for EXO-3 er målt til ≈2.5nSek med 3.0V forsyningspenning. Målingen er foretatt med en 300MHz probe med last-kapasitans 10 pF og oscilloscop i henhold til instrumentliste i Appendiks G. Gitt et minimumskrav [12] til målesytemets båndbredde på 10x signalfrekvensen for firkantpulser for å oppnå rimelig feiltoleranse på stigetid og kurveform, er målingen signfikant for denne oscillator som har F_0 = 14.7456MHz.

Figurene 3.2.1.1 og Figur 3.2.1.2 viser at krystallgenratoren har betydelig *overshot og undershot*, som er signal nivåer høyere enn forsyningspenning, og lavere enn jordplan. Årsaken er induktivitet i kretsens tilførselsledere, og i gulltrådsbonding fra silisiumbrikke til kapsel-pinne. Overshot vil under last kompenseres noe av kapasintans i last og kretskort. Over/undershot ut fra *ulastet* krets er således en fordel for å bevare høy stige/fallrate med kapasitiv last.

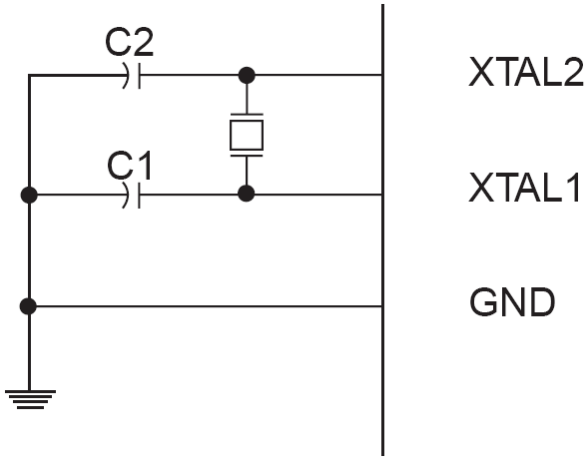
Oscilloscopet har i seg selv en stigetid på ca. 0.8 nsek [24].Sammen med en dempning i måleprobe på 3dB ved 300MHz, som demper høyfrekvenskomponenten i firkantsignalet, og dermed øker stigetid, er målt stigetid et verste tilfelle estimat for oscillator uten last. Med

kapasitiv last på oscillatorens utgang vil stigetid på klokkesignalet øke. Slik kapasitans finnes i kretskortets transmisjonslinje og i mikrokontrollerens, og billedbrikkens, MOSFET-inngang som pr. definisjon er en kondensator. Stigetid for masterklokke vil derfor ikke være kjent før målinger er på det ferdige kortet er foretatt.

Alle sammenlignede målinger med to oscilloscop-kanaler i bruk er gjort med 150MHz 17pF prober. Dette fordi NTNU kun disponenere en enkelt probe av 300MHz-typen til anvendte oscilloscop. Sammenlignende målinger av to samtidige kilder er således kun av aproksimativ karakter og er egnet til f.eks. vurdering av faseforsinkelser mellom to signaler.

3.2.2 Oscillator Flightmodell SC-2

I flightmodel SC-2 vil et krystall bli brukt som kilde for masterklokke. Mega1281 har krystall-inngang med inverterende forsterker [6]. Figur 3.2.2.1 viser hvordan krystallet kobles til mikrokontrolleren sammen med to kondensatorer.



Figur 3.2.2. 1 Tilkobling av eksternt krystall til mikrokontrollerens interne klokkegenerator

Valg av resonatorkondensatorer vil p.g.a. parasitiske kapasitanser bl.a. på kretskort være kritisk, og for flightmodel gjenstand for grundig testing [25].

Klokkesignal til bildebrikke med frekvensen $F_0/2$ vil i flightmodell bli generert av PWM-utgang (Pulse Width Modulation) på AVR Mega1281. Porten PE3 på mikrokontrolleren er fristilt for dette formål.

For detaljer om spenningsregulator og RS-232 linjedriver henvises til datablad [23] [26].

4 Valg av forsyningspenning

3.0V forsyningspenning er valgt som et kompromiss mellom mikrokontroller og billedbrikken MT9V111 som har spesifisert analog og digital driftspenning 2.8V +/- 0.25V, altså maksimum 3.05V.

Med samme driftspenning for begge kretser kan databusser kobles direkte sammen uten konvertering av logikknivåer v.h.a. en såkalt *leveltranslator*, som er strømkrevende og introduserer propagasjonsforsinkelse og dermed reduserer tidsvinduer for gyldig-data. Det er også et mål å redusere antall kretser på kortet for å minske antall mulige feilkilder.

Ved øke bildebrikkens driftspenning fra 2.8V til 3.0V vil også effekt, og dermed kjølebehov øke:

$$P_{ink} = \left(\frac{3.0}{2.8}\right)^2 \approx 1.15 \tag{4.1}$$

Kjølebehovet øker altså med ca. 15 prosent. Dette er uten betydning for kontrollerkortets varmebudsjett fordi bildebrikken kun er aktiv en brøkdel av et sekund, og bare når et bilde eksponeres, noe som normalt skjer en gang hvert jordomløp. Radiogrensesnittets begrensede datarate tillater bare nedlasting av ett enkeltbilde ved hver passering av jordstasjonen. [1]

Begrensninger i spesifikasjoner er vanligvis gjort for å redusere andel underkjente kretser fra en produksjonserie. Vi har fordelen av at kameramodulen ikke skal serieproduseres, og kan som med mikrokontrolleren, derfor teste flere brikker fra ulike produksjonserier og luke ut brikker med spesifikasjoner som grenser mot nedre del av intervall for oppgitte spesifikasjoner.

MT9V111 skal i sin tiltenkte anvendelse tåle kontinuerlig drift med dårlig kjøling, for eksempel et påslått kamera gjenglemt i hanskerommet på en bil en varm sommerdag. Kontinuerlig drift med for høy spenning har også betydning for den termiske mørkestrøm-støyen som øker med 3dB/8Kelvin. Brikken er spesifisert til alikevel å være i stand til å ta gode bilder under dårlige lysforhold, med høy signalforsterkning, for eksempel innendørs.

Med god termisk kontroll og under ett sekund driftstid er 3.0V forsyningspenning derfor å anse som uproblematisk for bildebrikkes del.

5 Kretskort, og fysisk implementering av kontrollerkort SC-1B

Formålet med SC-1A/B er å disponere en prototypemodul der alle kretser kan testes – med måle-tilkobling for alle viktige busser. Modulen skal brukes til å evaluere kurveformer, signalintegritet, spenningsnivåer, støykomponenter, overhøring mellom linjer, faser og logisk timing m.v.

Prototypemodulen bygger på SC-1A, et kort med tilkoblingsterminaler for alle kontaktpunkter på bildebrikken MT9V111-BGA. SC-1A ble utviklet under arbeidet med forutgående prosjektoppgave og er tilpasset sammenkobling med STK500 utviklerverktøy fra Atmel [27].

Prototypemodulens kontrollerkort, SC-1B, må være av høy kvalitet, uten forenklinger og besparelser av noe slag, for å kunne sansynliggjøre at det er kretsens egenskaper som måles – ikke kortets svakheter. Dette innebærer bl.a. at forsyningspenning og jord må distribueres på separate plan for å sikre ren stjernekobling uten jordsløyfer, og dessuten ungå serieinduktans i strømforsyning som kan resultere i at logiske desisjonsnivåer har distribuerte, instantane nivåforskjeller for de ulike kretskomponentene. Alle signaler mellom kretskomponenter må, så langt det lar seg gjøre, holdes på et plan uten gjennomføringer (vias) som introduserer serieresistans så vel som serieinduktans, og som dermed påvirker signalintegritet.

Tilgjengelig lisens for kretskortverktøy, CadSoft Eagle, støtter dessverre ikke produksjon av 4-lags kretskort [28]. Ferdigstilling av kontrollerkortet SC-1B må derfor utsettes til etter innlevering av denne oppgave. Formgivning av kretskortet er ferdigstillt med unntak av lag for forsyningspenning og jord. Disse lagene generes automatisk ved overflytting av kretskortfiler til CAD-program med oppgradert lisens for 4-lags kretskort.

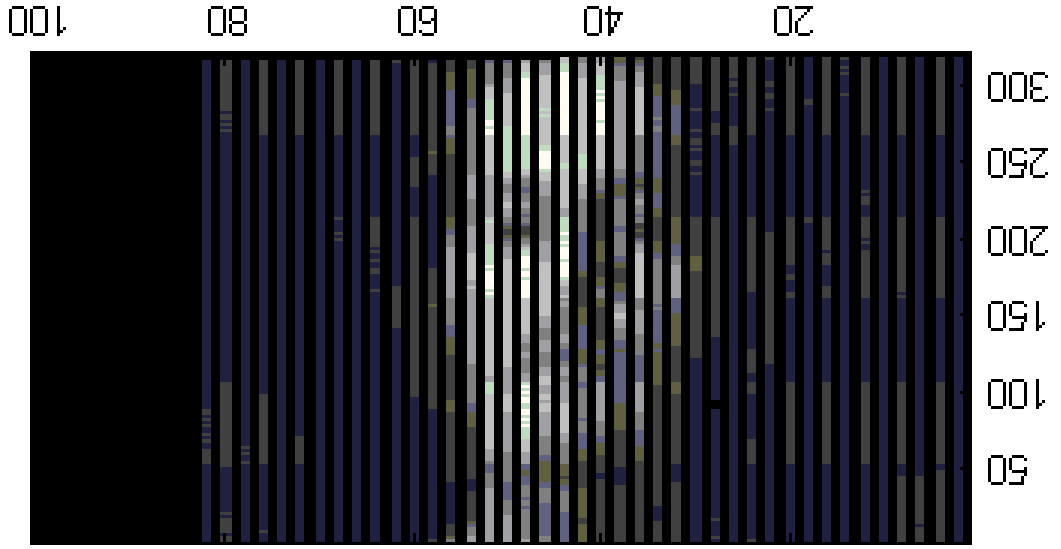
Kretsskjema for SC-1B, generert av kretskortverktøy, er vist i Appendiks C.

Tilgjengelig lisens for kretskortverktøy støtter heller ikke fordeling av kretskjema på mer enn ett ark. Kretsskjema i Appendiks er derfor noe uoversiktelig grunnet fortetting.

SC-1A er basert på et 4-lags kort med alle signaler samlet på øverste lag. Lag 2 er plan for digital jord, og lag 3 plan for digital forsyningsspenning. Analog forsyningspenning og analog jord er lagt til lag 4, sammen med avkoblingskondesatorer. SC-1A er vist i Appendiks D.

6 Konklusjon – veien videre

Arbeidet med kameraprototypen har vist at det er fullt mulig å realisere en robust kameramodul med dataoverføring styrt av en mikrokontroller. Figur 6.1 viser et bilde tatt med sensormodulen SC1-A koblet sammen med et Atmel AVR STK500 utviklerkort [27]. Selve bildefilen er generert i MATLAB med kode som vist i Appendiks F. Mikrokontrollerens kodesegmentet for dataoverføring er for denne testen skrevet i C, og ikke optimalisert. Bildefilen inneholder derfor ikke alle bildeelementer levert fra bildebrikken. De mørke kolonnene oppstår der mikrokontrolleren, p.g.a. manglede synkronitet, leser ut dataord med 2MSB som beskrevet i kapittel 3.1.1. Etter impementering av assembly-kode (vist i kapittel 2.2) for datautlesning vil bildefilen selvfølgelig ikke mangle bildeelementer. Hvilke piksel-posisjoner som er lest ut i figur 6.1 er ikke kjent. Farveinterpolasjon ikke er mulig, og bildet er derfor presentert monokromt. Det unaturlige tonale omfanget skyldes at alle bildebrikkens piksler er følsomme for NIR (*Near InfraRed*), uansett farvekanal. Prøvebildet i figur 6.1 er eksponert uten å bruke IR-blokk filter.



Figur 6. 1 Monokromt bilde fra prøveprogram skrevet i C – bildet generert i MATLAB

Neste steg på veien er å implementere flightmodel SC-2 med utgangspunkt i kretsskjema for SC-1A/B. Dette arbeidet kan påbegynnes så snart mål-plattform er kjent.

Det er også aktuelt å lage en versjon utvidet med eksternt flashminne slik at eksponert bilde ikke slettes selv om kameramodulens fosyningspenning frakobles. Denne versjenen vil få betegnelsen SC-3.

En modul, med betegnelsen SC-4, basert på helt ny brikke fra Micron, er under utvikling med assistanse fra selskapet Norspace i Horten. Den nye bildesensoren har JPEG-kompresjon implementert i maskinvare, med valgfrie parametre for Huffman-tabell og kvantiserings-matrise. Dette gir mulighet for spesialtilpassing av kompresjon til motiv, og moderat eller lite tap, fordi jorden sett fra rommet har karakteristiske farver, og kjent fordeling av frekvens-komponenter i bildeplanet. SC-4 har 2 megapiksler oppløsning, som tilsvarer 7 x bilde-oppløsningen til SC-2. Hvert enkelt bildeelement har ¼ av arealet til MT9V111s piksler. SC-4 vil, med samme optikk, ha dobbelt så høy bakkeoppløsning som SC-2. Se for øvrig prosjektoppgavens kapittel 3.3. for nærmere omtale av den såkalte *formfaktor-relasjonen*.

7 Litteraturreferanser

[1] Borja, L. T, Kameramodul til nCube-2, Prosjektoppgave, Institutt for elektronikk og telekommunikasjon, Norges teknisk-naturviteskapelige universitet, juni 2005.

[2] Marcus, S. J, "Grand Illusion: the Risk of Cassini", IEEE Spectrum, Vol. 35, No.3, pp.58-65, March 1998.

[3] Cubesat Documents, California Polytechnic StateUniversity, http://cubesat.calpoly.edu/_new/documents/index.html

[4] Micron 1/4-Inch SOC VGA CMOS Active-Pixel Digital Image Sensor MT9V111, Rev. G - Production, January 2005, http://download.micron.com/pdf/datasheets/imaging/MT9V111.pdf

[5] Micron 1/4-Inch VGA CMOS Digital Image Sensor MT9V011, Rev. B, January 2005, http://download.micron.com/pdf/datasheets/imaging/MT9V011.pdf

[6] ATMEL AVR ATmega640/1280/1281/2560/2561 8-bit RISC, Preliminary, Revision F, updated 04/06, http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf

[7] ATMEL 8-bit AVR Instruction Set, User Guide, updated 11/05, http://www.atmel.com/dyn/resources/prod_documents/doc0856.pdf

[8] Cypress Semiconductor – FillFactory, About Correlated Double Sampling, 2005, http://www.fillfactory.com/htm/technology/htm/cds.htm

[9] Mano, M.M. and Kime, C. R, Logic and Computer Design Fundamentals, Second Edition, Prentice Hall, 2001.

[10] Skriftlig medelelse, Morten Larsen, Field Applications Engineer Microcontroller applications, datacom, networking, ACTE AS, Kjeller Norway.

[11] KEMET, Surface Mount Capasitors, Datasheets, F3102K 8/05, http://www.kemet.com/kemet/web/homepage/kechome.nsf/weben/039A1B3C5D08147ACA2570A5001283B2/$file/F3102.pdf

[12] Cypress Semiconductor, Buterbaugh, E. with Contributers, Perfect Timing, A Design Guide for Clock Generation and Distribution, Second Edition, Cypress Semiconductor Corporation, 2002, http://www.cypress.com/publishedcontent/publish/design_resources/application_notes/contents/perfect_timing_ii_12.pdf

[13] European Space Components Information Exchange System (ESCIES), escies.org

[14] Svaasand, L. O, Elektrisitet og magnetisme, del 1 – elektrostatikk, 8. opplag, Tapir forlag, 1999.

[15] Cheng, D. K, Field and Wave Electromagnetics, Second Edition, Addison-Wesley Publishing Company, 1989.

[16] Pozar, D. M, Microwave and RF Wireless Systems, John Wiley & Sons, 2001.

[17] BSI BS62LV4006, datablad, http://www.bsi.com.tw/product/BS62LV4006.pdf

[18] Alliance AS7C34096, datablad, http://www.alliancememory.com/pdf/sram/fa/as7c34096a_v2.1.pdf

[19] Texas Instruments SN74AHC573 Octal Transparent D-Type Latches With 3-State Outputs, datablad, http://focus.ti.com/lit/ds/symlink/sn74ahc573.pdf

[20] Texas Instruments SN74AHC373 Octal Transparent D-Type Latches With 3-State Outputs, datablad, http://focus.ti.com/lit/ds/symlink/sn74ahc373.pdf

[21] KSS Kinseki EXO-3 CMOS Programmable Oscillator, datablad, http://www.farnell.com/datasheets/74178.pdf

[22] IQD IQEXO-3 Oscillator, datablad, http://www.elfa.se/pdf/74/07454002.pdf

[23] Texas Instruments TPS79630, Ultralow-Noise, High PSRR, Fast, RF, 3V 1A, Low-Dropout Linear Regulator, http://focus.ti.com/lit/ds/symlink/tps79630.pdf

[24] Tektronix TDS 500D, TDS 600B & TDS 700D Digitizing Oscilloscopes Performance Verification and Specifications, Tektronix, 1998.

[25] O'Flynn, C, Why You Need a Clock Source, www.avrfreaks.net/index.php?module=dpDocs&func=index&cid=9

[26] Maxim MAX3233E, ±15kV ESD-Protected, 1µA, 250kbps, 3.3V/5V, Dual RS-232 Transceivers with Internal Capacitors, http://pdfserv.maxim-ic.com/en/ds/MAX3233E-MAX3235E.pdf

[27] ATMEL AVR STK500 User Guide, http://www.atmel.com/dyn/resources/prod_documents/doc1925.pdf

[28] CadSoft EAGLE ver. 4.16, Light Edition (Freeware), http://www.cadsoft.de/cgi-bin/download.pl?page=/home/cadsoft/html_public/download.htm.en&dir=eagle/program/4.1

[B-1] Cypress Semiconductor – FillFactory, The Color Filter Array FAQ – Introduction, 2005, http://www.fillfactory.com/htm/technology/htm/rgbfaq.htm

[B-2] Gonzalez, R. Woods, R. and Eddins, S. Digital Image Processing Using MATLAB, Prentice Hall, 2003.

[B-3] Ting Chen, A Study of Spatial Color Interpolation Algorithms for Single-Detector Digital Cameras, Stanford University, 1999, http://www-ise.stanford.edu/~tingchen/main.htm

Alle internett-referanser er kontrollert og funnet gyldige ved innleveringsdato for denne oppgave. Utskrift av nettsteder er arkivert.

Appendiks A Sammendrag av forutgående prosjektoppgave

Formålet med denne oppgaven er å utrede en kameramodul for mulig implementering i studentsatellitten nCube-2, som har planlagt oppskyting august 2005.

Først er grunnleggende forutsetninger og arbeidsbetingelser analysert, herunder satelittens baneparametre, radiogrensesnitt, mekaniske struktur og strømforsyning.

Ved valg av billedsensorteknologi er det tatt utgangspunkt i analyse og matematisk beskrivelse av det menneskelige farvesyn. For å etterligne synet trenger vi bare 3 sett sensorer, men med spektral følsomhetsfordeling noenlunde lik øyets tre pigmenter. Det siste viser seg å være krevende. Det finnes heller ikke tre monokromatiske farver som kan gjenskape synsintrykket for alle fysisk realiserbare farver. Gode aproksimasjoner er alikevel mulig.

Forskjellige teknikker for optisk farveseparasjon er beskrevet og evaluert. Valg av teknologi avhenger av anvendelse. Viktigst er avveining mellom billedkvalitet og kamerastørrelse. Optisk farvefiltrering med Bayer-mønster og én enkelt billedbrikke viser seg å være den beste løsningen for små og robuste kameramoduler.

Det er påvist følgende generelle sammenheng mellom kamerakonstruksjonens volum og brikkediagonal:

$$V = O\left(D^2 \cdot L\right) = O\left(d^3\right)$$

Denne relasjonen har vi valgt å kalle *formfaktor-relasjonen.* Den praktiske konsekvensen av denne relasjonen er at en halvering av billedbrikkens diagonale størrelse reduserer kameraets volum og masse til 1/8.

To typer sensorteknologi er vurdert, CCD (Charge Coupled Device) og CMOS (Complimentary Metal Oxide Semiconductors). CCD-teknologi har til nå vært enerådende i videnskapelig sammenheng. CCD-sensorer har svakheter, de viktigste er høyt effektforbruk, sårbarhet for høyenergi partikkelstråling og ladningslekasje mellom billedelementene. Innledende målinger viser at ny, støysvak, CMOS-teknologi med fordel kan anvendes i et satellitt-kamera.

Ved valg av sensorteknologi er strømforbruk, formfaktor, antall nødvendige støttekretser samt tidsbudsjett for utvikling og integrering vurdert. CMOS-teknologien viser seg best egnet for vårt formål.

Sammenhengen mellom satellittens banehøyde, bildevinkel og bildeutsnitt er vist.
Gitt begrensninger i nCube-2s retningskontroll vil satellitten pendle +/-20grader.
For å sikre at bildeutsnittet inkluderer motivet som er utpekt velges en horisontal billedvinkel på minimum 50 grader. En enkel metode for aproksimativ beregning av optikkens brennvidde basert geometriske betrakninger er også vist, sammen med et enkelt verktøy for vurdering av optikkens kvalitet.

Arbeidsbetingelser relatert til rommiljø og belastninger under oppskyting av satellitten er diskutert med henblikk på rom-kvalifisering av kameramodulen.

Elektrisk ladede partikler med høy bevegelsesenergi er fanget i jordens magnetfelt. Konsentrasjonen varierer med høyde over jorden og er ekstra stor i de såkalte Van Allen-beltene. nCube-2 vil befinne seg under Van Allen-beltene og vil i sin forventede levetid avsorbere en strålingsdose på 0.5krad(Si). Ordinære kommersielle kretser tåler betydlig høyere stråledoser. Det viser seg at CMOS-billedbrikker grunnet spesielle krav til produksjonsparamtre er særlig robuste mot partikkelstråling. Gitt nCube-2s baneparametre, og korte forventede levetid, er det ikke nødvendig å anvende såkalte rad-hard kretser, kretser som er fremstilt for å kunne tåle høy strålingsdose.

Optisk glass som utsettes for ioniserende stråling endrer etterhvert egenskaper – denne prossesen kalles *browning* og kan observeres som mørkning av glasset. Et Ce-dopet strålingsresistent planart frontglass er vist å være tilstrekkelig beskyttelse mot denne strålingen.

Raporten konkluderer med at det er fullt mulig å oppdatere nCube-2 med en kameramodul uten nevneverdige inngrep i struktur og elektronikk.

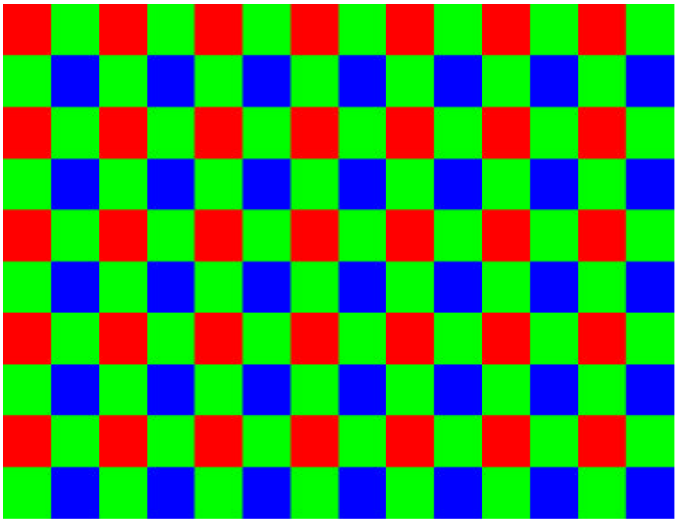
Appendiks B Farveseparasjon og Bayerfilter

Dette avsnittet er hentet fra Prosjektoppgave [1]. Teksten er noe omarbeidet.

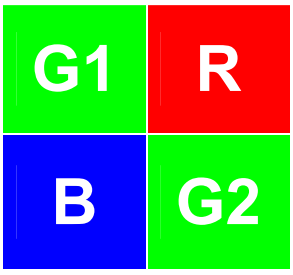
For å kunne registrere, og senere gjengi, et farvebilde basert på den menneskelige oppfatning av farver må de tre såkalte *primærfargene* rødt, grønt og blått skilles fra hverandre. [10)]
En metode for å separere farver er å legge en mosaikk av farvefiltre direkte på bildebrikken. Dette forutsetter at sensorarealet er oppdelt i adresserbare felter slik at det er mulig å håndtere farveinformasjonen når bildet skal rekonstrueres. Metoden er altså ikke egnet for f.eks. rørkameraer.

Bildebrikken MT9V111s sensorareal er delt opp i bildeelementer, *piksler* (engelsk: *pixel* - forkortelse for Picture Element) som er ordnet i rader og kolonner. Hvert enkelt bildeelement kan adresseres og avleses av en ekstern styringslogikk.

Figur B.1 viser et *Bayer-mønster* for pikselfiltere [B-1]. Dette består av rader og kolonner med alternerende røde, grønne og blå farvefiltre. Kolonner med like nummer og rader med ulike nummer har blå og grønne filtre, kolonner med ulike nummer og rader med like nummer har røde og grønne filtre. Merk at bildeelement øverst til høyre på en bildebrikke ved konvensjon har posisjonsnummer (0,0).
50% av pikslene er altså grønne, mens røde og blå piksler utgjør 25% hver. Dette svarer til synets noe høyere oppløsning i den grønne delen av spekteret. Figur B.2 viser vekting av rødt, grønt og blått.



Figur B.1 Bayer-mønster for pikselfiltre



Figur B.2 Farvevekting

I hver enkelt pikselposisjon (m,n) har vi bare informasjon om én av farvekomponentene. De resterende to farvekomponenter rekonstrueres ved å sammenholde luminasverdi i posisjon (m,n) med luminansverdi i omkringliggende piksler med riktig farve. De to manglende farveverdier for piksel (m,n) beregnes så gjennom en vektet algoritme.

Det finnes en rekke prinsipper og algoritmer for rekonstruksjon av et bilde kodet i Bayer-mønster. Disse deles gjerne inn i *adaptive* og *ikke-adaptive* algoritmer [B-2] [B-3].

10) For nærmere beskrivelse av farvesyn , farveseparasjon eksponering og representasjon, henvises til prosjektoppgave [1]

En ikke-adaptiv algoritme er uavhengig av luminansverdier i piksel-gruppen som er med i beregningen. En adaptiv algoritme tilpasser interpolasjonsalgoritmen avhengig av verdi på omliggende piksler.

Et eksempel på en enkel, ikke-adaptiv algoritme er *bilineær interpolasjon*. Figur B.3 viser et nummerert rutenett av piksler med farvefiltre. Luminansverdien til grønt i en ikke-grønn pikselposisjon beregnes som gjennomsnittet av de fire nærmest tilliggende grønne piksler – grønn luminansverdi i posisjon B_8 er da gitt ved [B-2] [B-3]:

$$G_8 = \frac{G_3 + G_7 + G_9 + G_{13}}{4} \qquad \text{(b-1)}$$

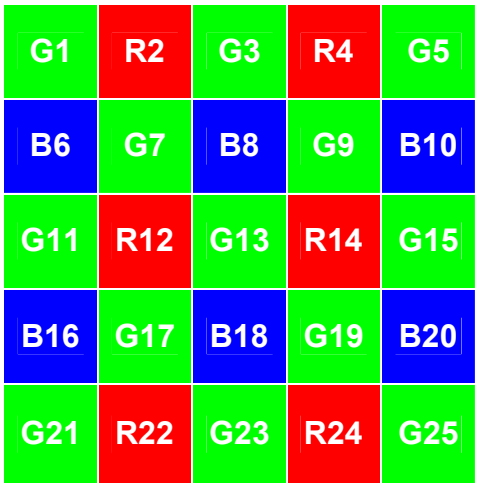
Luminansverdien for rødt eller blått i en grønn pikselposisjon beregnes som snittet av de *to* nærmest tilliggende piksler av samme farve, altså luminansverdi enten over og under, eller til høyre og venstre for den grønne pikselposisjonen – rød og blå luminansverdi i posisjon G_7 er gitt ved:

$$B_7 = \frac{B_6 + B_8}{2} \qquad R_7 = \frac{R_2 + R_{12}}{2} \qquad \text{(b-2)}$$

Luminansverdi for henholdsvis rødt i en blå pikselposisjon og blått i en rød pikselposisjon er snittet av luminansverdien til de fire diagonalt tilliggende pikselposisjoner:

$$R_8 = \frac{R_2 + R_4 + R_{12} + R_{14}}{4} \qquad B_{12} = \frac{B_6 + B_8 + B_{16} + B_{18}}{4} \qquad \text{(b-3)}$$

Merk at *bilineær interpolasjon* ikke tar hensyn til luminansverdien i pikselposisjonen det interpoleres for. Algoritmer som f.eks. *Smooth Hue Transition Interpolation* vil derimot vekte interpolerte luminansverdier mot luminansverdi i gjeldende pikselposisjon [B-3].



Figur B.3 Nummerert rutenett av piksler med farvefiltre

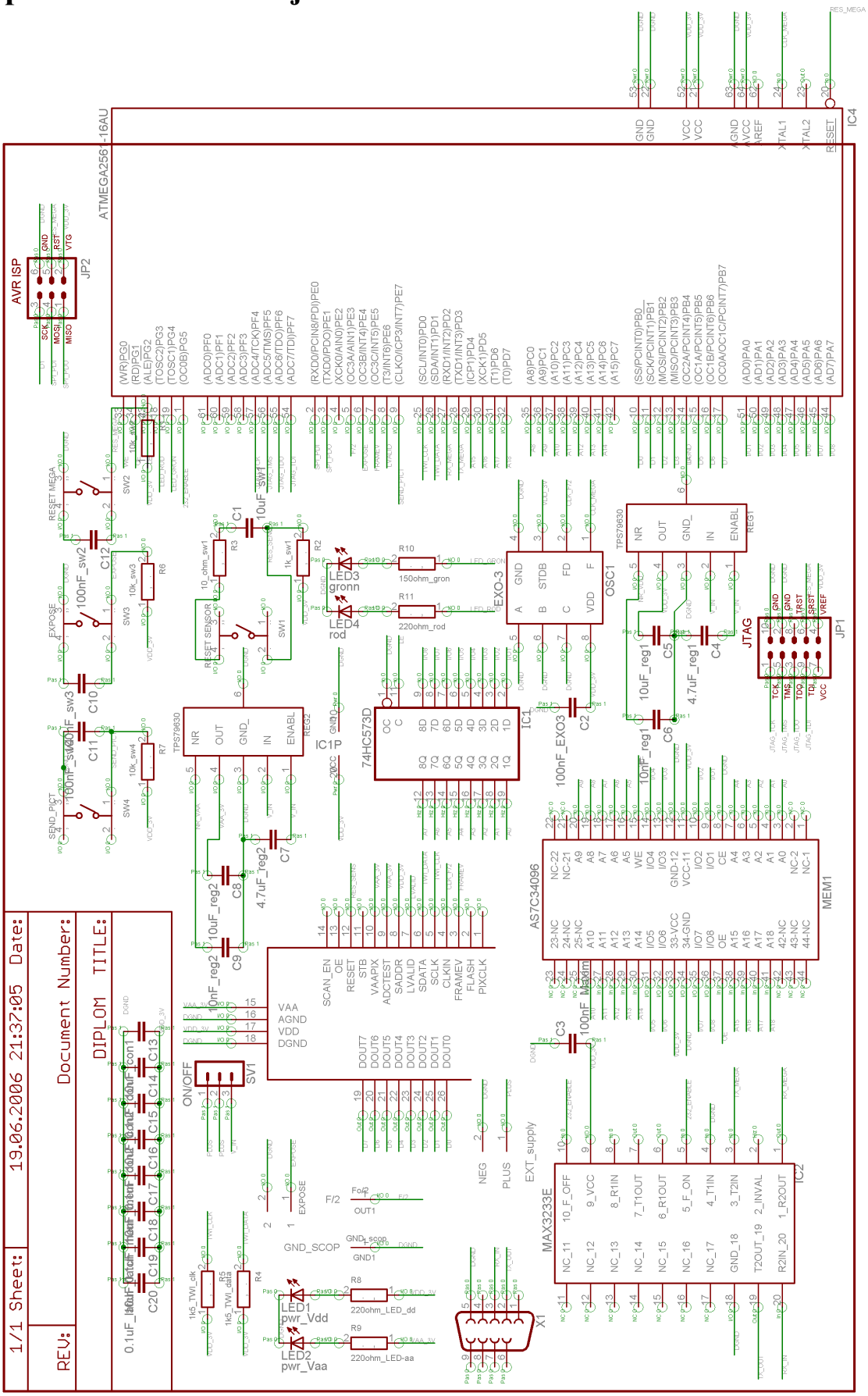
Adaptive interpolasjonsalgoritmer er vesentlig mer komplekse enn tilsvarende ikke-adaptive algoritmer. Rekonstruksjon av bildet hører inn under bakkesegmentet og en detaljbeskrivelse av de adaptive algoritmene faller utenfor rammen av denne oppgave. Vi nevner derfor bare at en adaptiv algoritme tilpasser interpolasjonsalgoritmen dynamisk, avhengig av kontrast i

luminans mellom interpolasjons-koordinat og omliggende piksler. En adaptiv algoritme gir vanligvis bedre resultat en en ikke-adaptiv algoritme, men er også mer beregningsintensiv. Fordi rekonstruksjon av bildet foregår i bakkesegmentet, og dessuten ikke skal utføres i sanntid vil det for et satellittkamera være naturlig å velge den algoritmen som gir best resultat for det faktiske bildematerialet, uten hensyn til resurskrav. Bildet kan behandles med forskjellige interpolasjonsalgoritmer, og de ulike resultater kan sammenlignes for å ekstrahere ytterligere visuell informasjon.

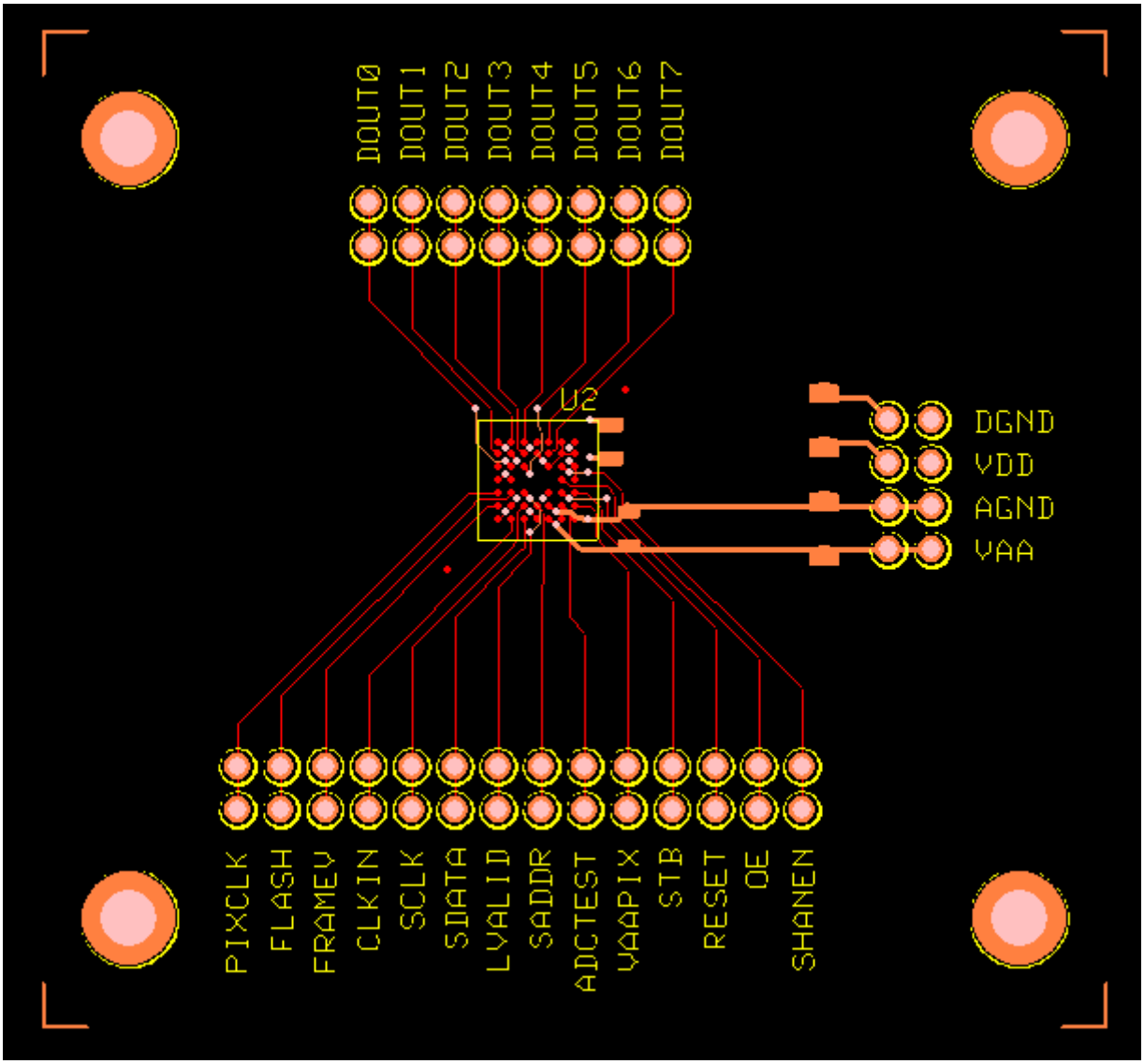
Det finnes også andre måter å ordne filtermosaikken, med ulik vekting av farvekomponentene. Rekonstruksjonsalgorimen må selvfølgelig tilpasses det anvendte farvemønster.

Det er viktig å merke seg at farveinterpolasjon *ikke* øker informasjonsinnholdet i bildet. Alle algoritmer for farveinterpolasjon baserer seg på kvalifiserte gjetninger om luminansverdier for de manglende farvekomponeter i hvert bildeelement. Den reelle verdien kan være en helt annen. Farveinterpolasjon er kun et hjelpemiddel for visuell tolkning av bildet.

Appendiks C Kretsskjema SC-1B



Appendiks D Kretskortutlegg SC-1A



Figur D.1 Kretskortutlegg prototype SC-1A; Signalførende lag (lag 1) samt analog forsyningspenning (lag 4). Lag 2 - digital jord DNGD og lag 3 - plan for digital forsyningsspenning VDD er *ikke* tatt med.

MERK: Figuren er i målestokk 3:1

Appendiks E Kildekode i C

Under følger utskrift av arbeidskode brukt under det fortløpende utviklingsarbeidet med kameraprototypen. Koden er ikke ferdig og er kun tatt med for referranse.
Midlertidig deaktiverte kodesegmenter er enten utkommentert som linje med ”/”, som segment med ”/* */” eller med betingelsen: #if 0 ”deaktivert_kode” endif.

Koden er skrevet av Rein Anders Apeland, i sammarbeid med oppgaveskriver.

```
/*! \file
 *
 * \brief  Main source file for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: main.c,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#include "config.h"
#include "uart.h"
#include "i2c.h"



// --- Local prototypes ---

//! Configure camera IO.
void camio_Init( void );

//! Use I2C driver to configure camera.
int8_t ConfigureCamera( void );

int8_t cam_WriteVerifyIFP( uint8_t reg, uint16_t value );
int8_t cam_WriteVerifyCore( uint8_t reg, uint16_t value );
int8_t cam_SoftResetCamera( void );

//! Setup XRAM interface.
void xram_Init( void );
//! Select 32k XRAM bank.
void xram_SetBank( uint8_t bank );
//! XRAM 32k bank starting pointer.
uint8_t * const xmem = (void *) 0x2200;
```

```
//! Dump pixel data to UART.
int8_t DumpPixelData( void );



void errorLoop( int8_t errorCode )
{
	//! \todo  Do something with this error code.
	PORTB = errorCode;
	for(;;);
}


// --- START ---
void main( void )
{
	int8_t ret;
	xram_Init();
	camio_Init();
	i2c_Init();
	ret = ConfigureCamera();
	if( ret != 0 ) errorLoop( ret );
	uart_Init();

	uint8_t buf[ 10 ];
	for( int8_t adr = 0; adr < 10; ++adr ) buf[ adr ] = adr + 50;
	for( int8_t adr = 0; adr < 10; ++adr ) xmem[ adr ] = adr + 20;
	for( int8_t adr = 0; adr < 10; ++adr ) buf[ adr ] = xmem[ adr ];
	for( int8_t adr = 0; adr < 10; ++adr ) xmem[ adr ] = adr + 30;
	for( int8_t adr = 0; adr < 10; ++adr ) buf[ adr ] = xmem[ adr ];
	for( int8_t adr = 0; adr < 10; ++adr ) xmem[ adr ] = buf[ adr ];

	// Enable interrupts and let the state maching run forever.
	__enable_interrupt();

	for(;;);
}



void xram_Init( void )
{
	XMCRA = (1<<SRE) | (0<<SRW11) | (0<<SRW10);
	XMCRB = (0<<XMBK) | (1<<XMM0);
}



void xram_SetBank( uint8_t bank )
{
```

```
	if( bank == 0 ) {
//		PORTC = (0<<7);
	} else if( bank == 1 ) {
//		PORTC = (1<<7);
	}
}



int8_t cam_WriteVerifyIFP( uint8_t reg, uint16_t value )
{
	int8_t ret;
	ret = i2c_WriteVerifyRegister16( 0x01, 0x0001 );
	if( ret != 0 ) return ret;
	ret = i2c_WriteVerifyRegister16( reg, value );
	if( ret != 0 ) return ret;

	return 0;
}



int8_t cam_WriteVerifyCore( uint8_t reg, uint16_t value )
{
	int8_t ret;
	ret = i2c_WriteVerifyRegister16( 0x01, 0x0004 );
	if( ret != 0 ) return ret;
	ret = i2c_WriteVerifyRegister16( reg, value );
	if( ret != 0 ) return ret;

	return 0;
}



int8_t cam_SoftReset( void )
{
	int8_t ret;

	i2c_Release();
	ret = cam_WriteVerifyCore( 0x0D, 0x0001 );
	if( ret != 0 ) return ret;
	ret = cam_WriteVerifyCore( 0x0D, 0x0000 );
	if( ret != 0 ) return ret;
	ret = cam_WriteVerifyIFP( 0x07, 0x0001 );
	if( ret != 0 ) return ret;
	ret = cam_WriteVerifyIFP( 0x07, 0x0000 );
	if( ret != 0 ) return ret;

	return 0;
```

```
}


int8_t ConfigureCamera( void )
{
	int8_t ret;

	ret = cam_SoftReset();
	if( ret != 0 ) { i2c_Release(); return -1; }

	ret = cam_WriteVerifyIFP( 0x08, (1<<7)|(1<<11)|(1<<14)|(1<<15) );
	if( ret != 0 ) { i2c_Release(); return -1; }

	return 0;
}


void camio_Init( void )
{
	// Enable pullup on button input.
	BUTTON_PORT |= (1<<BUTTON_BIT);
	// Button interrupt fires on falling edge.
	EICR_BUTTON |= (1<<ISC1_BUTTON) | (0<<ISC0_BUTTON);
	// Frame interrupt fires on rising edge.
	EICR_FRAME |= (1<<ISC1_FRAME) | (1<<ISC0_FRAME);
	// Line interrupt fires on rising edge.
	EICR_LINE |= (1<<ISC1_LINE) | (1<<ISC0_LINE);

	// Clear pending and enable button interrupt.
	EIFR |= (1<<INT_BUTTON);
	EIMSK |= (1<<INT_BUTTON);
}


//! Countdown for start vertical blanking.
uint16_t blankLineCountdown;
//! Countdown for active lines.
uint16_t activeLineCountdown;
//! Destination XRAM pointer for pixel data.
uint8_t * pixelData;


#pragma vector = BUTTON_INTERRUPT
__interrupt void ButtonISR( void )
{
	// Disable button interrupt.
```

```
        EIMSK &= ~(1<<INT_BUTTON);

        // Clear pending and enable frame interrupt.
        EIFR |= (1<<INT_FRAME);
        EIMSK |= (1<<INT_FRAME);
}



#pragma vector = FRAME_INTERRUPT
__interrupt void FrameISR( void )
{
        // Init line interrupt control variables.
        blankLineCountdown = VERT_BLANK_LINES;
        activeLineCountdown = VERT_ACTIVE_LINES;
        pixelData = xmem;

        // Disable frame interrupt.
        EIMSK &= ~(1<<INT_FRAME);

        // Clear pending and enable line interrupt.
        EIFR |= (1<<INT_LINE);
        EIMSK |= (1<<INT_LINE);
}



#pragma vector = LINE_INTERRUPT
__interrupt void LineISR( void )
{
        if( blankLineCountdown > 0 ) {
                --blankLineCountdown;
        } else {
                // Wait for horizontal blanking.
                __delay_cycles( HORIZ_BLANK_CYCLES );

                // Read pixel data into SRAM.
                uint16_t pixelCountdown = HORIZ_ACTIVE_PIXELS;
                do {
                        *pixelData = CAMERA_DATA;
                        ++pixelData;
                } while( --pixelCountdown > 0 );

                // Wait for Line Valid to go low.
                __delay_cycles( HORIZ_END_WAIT_CYCLES );

                // Last line?
                if( --activeLineCountdown == 0 ) {
                        // Disable line interrupt.
                        EIMSK &= ~(1<<INT_LINE);
```

```
                        // Output pixel data from XRAM.
                        DumpPixelData();

                        // Clear pending and enable button interrupt.
                        EIFR |= (1<<INT_BUTTON);
                        EIMSK |= (1<<INT_BUTTON);
                }
        }
}



int8_t DumpPixelData( void )
{
        /*
        for( uint16_t y = 0; y < 200; ++y ) {
                for( uint16_t x = 0; x < 5; ++x ) {
                        for( uint8_t sub = 0; sub < 20; ++sub ) {
                                uart_SendChar( 0xFF );
                        }
                        for( uint8_t sub = 0; sub < 20; ++sub ) {
                                uart_SendChar( 0x80 );
                        }
                        for( uint8_t sub = 0; sub < 20; ++sub ) {
                                uart_SendChar( 0x00 );
                        }
                }
        }
        */

        pixelData = xmem;
        for( uint16_t line = 0; line < VERT_ACTIVE_LINES; ++line ) {
                for( uint16_t pixel = 0; pixel < HORIZ_ACTIVE_PIXELS; ++pixel ) {
                        uart_SendChar( *pixelData );
                        ++pixelData;
                }
        }

        return 0;
}
```

```
/*! \file
 *
 * \brief  IO and resource configuration for for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: config.h,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#ifndef CONFIG_H
#define CONFIG_H

#include <ioavr.h>
#include <inavr.h>



//! Camera I2C write address.
#define CAM_WR_ADDR 0xB8
//! Camera I2C read address.
#define CAM_RD_ADDR 0xB9



//! CPU speed in kHz.
//#define CPU_F 11059.2
#define CPU_F 14745.600
//! Desired baudrate (kbps).
#define BAUDRATE 115.200
//! UART Baud Rate Register value.
#define UBRR_VALUE (CPU_F / (16 * BAUDRATE) - 1)



//! Handy typedef for 8-bit unsigned.
typedef unsigned char uint8_t;
//! Handy typedef for 8-bit signed.
typedef signed char int8_t;
//! Handy typedef for 16-bit unsigned.
typedef unsigned short int uint16_t;
//! Handy typedef for 32-bit unsigned.
typedef unsigned long int uint32_t;



//! Camera horizontal blanking CPU cycles.
```

```
#define HORIZ_BLANK_CYCLES 50
//! Camera horizontal active lines.
#define HORIZ_ACTIVE_PIXELS 100
//! Camera vertical blanking lines.
#define VERT_BLANK_LINES 10
//! Camera vertical active lines.
#define VERT_ACTIVE_LINES 320
//! Camera horizontal end blanking CPU cycles.
#define HORIZ_END_WAIT_CYCLES 10



//! Vector for active-low external button interrupt (Take picture).
#define BUTTON_INTERRUPT INT4_vect
//! External Interrupt Control Register for button.
#define EICR_BUTTON EICRB
//! Interrupt Sense Control Bit 0 for button.
#define ISC0_BUTTON ISC40
//! Interrupt Sense Control Bit 1 for button.
#define ISC1_BUTTON ISC41
//! Interrupt Mask Bit for button.
#define INT_BUTTON INT4
//! Button PORT register.
#define BUTTON_PORT PORTE
//! Button pin bit.
#define BUTTON_BIT PE4



//! Vector for rising-edge Frame Start signal interrupt.
#define FRAME_INTERRUPT INT5_vect
//! External Interrupt Control Register for Frame Start.
#define EICR_FRAME EICRB
//! Interrupt Sense Control Bit 0 for Frame Start.
#define ISC0_FRAME ISC50
//! Interrupt Sense Control Bit 1 for Frame Start.
#define ISC1_FRAME ISC51
//! Interrupt Mask Bit for Frame Start.
#define INT_FRAME INT5



//! Vector for rising-edge Line Start signal interrupt.
#define LINE_INTERRUPT INT6_vect
//! External Interrupt Control Register for Line Start.
#define EICR_LINE EICRB
//! Interrupt Sense Control Bit 0 for Line Start.
#define ISC0_LINE ISC60
//! Interrupt Sense Control Bit 1 for Line Start.
#define ISC1_LINE ISC61
```

```
//! Interrupt Mask Bit for Line Start.
#define INT_LINE INT6



//! Camera data input.
#define CAMERA_DATA PINB



//! UART Baud Rate Register.
#define UBRR UBRR1
//! UART Data Register.
#define UDR UDR1
//! UART Control and Status Register A.
#define UCSRA UCSR1A
//! UART Data Register Empty bit.
#define UDRE UDRE1
//! UART Control and Status Register B.
#define UCSRB UCSR1B
//! Transmitter Enable bit.
#define TXEN TXEN1

#endif
```

```
/*! \file
 *
 * \brief  Polled I2C communication driver for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: i2c.h,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#ifndef I2C_H
#define I2C_H

#include "config.h"



//! Setup bus.
void i2c_Init( void );
//! Release bus.
void i2c_Release( void );
//! Writes 'value' to register address 'reg'. Returns 0 if ok, -1/err if not.
int8_t i2c_WriteRegister16( uint8_t reg, uint16_t value );
//! Reads from register address 'reg' to '*value'. Returns 0 if ok, -1/err if not.
int8_t i2c_ReadRegister16( uint8_t reg, uint16_t * value );
//! Writes and verifies value. Returns 0 if ok, -1/err if not.
int8_t i2c_WriteVerifyRegister16( uint8_t reg, uint16_t value );
//! Verifies register value. Returns 0 if ok, -1/err if not.
int8_t i2c_VerifyRegister16( uint8_t reg, uint16_t value );



#endif
```

```
/*! \file
 *
 * \brief  Polled I2C communication driver for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: i2c.c,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#include "i2c.h"



#define TWSRMASK (128+64+32+16+8)



void i2c_Init( void )
{
	// Bit rate = CPU_F / (16 + 2*TWBR*4^TWPS).
	TWSR |= (1<<TWPS1) | (1<<TWPS0); // Set TWI bit rate prescaler.
	TWBR = 2; // Set TWI bit rate.
}



void i2c_Release( void )
{
	// Send STOP condition and exit.
	TWCR = (1<<TWINT) | (1<<TWSTO);
}



int8_t i2c_WriteRegister16( uint8_t reg, uint16_t value )
{
	uint8_t status;

	// Send START condition and check status code.
	TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x08 ) return -1;

	// Send SLA+W and check status code.
	TWDR = CAM_WR_ADDR;
```

```
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x18 ) return -2;

	// Send register address byte and check status code.
	TWDR = reg;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x28 ) return -3;

	// Send 1st data byte and check status code.
	TWDR = (value >> 8) & 0xff;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x28 ) return -4;

	// Send 2nd data byte and check status code.
	TWDR = value & 0xff;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x28 ) return -5;

	// Send STOP condition and exit.
	TWCR = (1<<TWINT) | (1<<TWSTO);
	return 0;
}



int8_t i2c_ReadRegister16( uint8_t reg, uint16_t * value )
{
	uint8_t hibyte;
	uint8_t lobyte;
	uint8_t status;

	// Send START condition and check status code.
	TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x08 ) return -11;

	// Send SLA+W and check status code.
	TWDR = CAM_WR_ADDR;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
```

```
	if( status != 0x18 ) return -12;

	// Send register address byte and check status code.
	TWDR = reg;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x28 ) return -13;

	// Send REPEATED START condition and check status code.
	TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x10 ) return -14;

	// Send SLA+R and check status code.
	TWDR = CAM_RD_ADDR;
	TWCR = (1<<TWINT) | (1<<TWEN);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x40 ) return -15;

	// Receive 1st value byte.
	TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x50 ) return -16;
	hibyte = TWDR;

	// Receive 2nd value byte.
	TWCR = (1<<TWINT) | (1<<TWEN) | (0<<TWEA);
	do {} while( (TWCR & (1<<TWINT)) == 0 );
	status = TWSR & TWSRMASK;
	if( status != 0x58 ) return -17;
	lobyte = TWDR;

	// Send STOP condition and exit.
	TWCR = (1<<TWINT) | (1<<TWSTO);
	*value = (hibyte << 8) | lobyte;
	return 0;
}



int8_t i2c_WriteVerifyRegister16( uint8_t reg, uint16_t value )
{
	uint16_t readout;
	int8_t retval;

	// Write value first.
```

```
	retval = i2c_WriteRegister16( reg, value );
	if( retval != 0 ) return retval;

	// Read back register.
	retval = i2c_ReadRegister16( reg, &readout );
	if( retval != 0 ) return retval;

	// Compare.
	if( readout != value ) return -21;

	return 0;
}



int8_t i2c_VerifyRegister16( uint8_t reg, uint16_t value )
{
	uint16_t readout;
	int8_t retval;

	// Read register.
	retval = i2c_ReadRegister16( reg, &readout );
	if( retval != 0 ) return retval;

	// Compare.
	if( readout != value ) return -21;

	return 0;
}
```

```
/*! \file
 *
 * \brief  Polled UART communication driver for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: uart.h,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#ifndef UART_H
#define UART_H

#include "config.h"



//! Setup bus.
void uart_Init( void );
//! Send one character.
void uart_SendChar( char ch );
//! Send 0-terminated SRAM string.
void uart_SendStringSRAM( char * str );
//! Send 0-terminated Flash string.
void uart_SendStringFlash( char __flash * str );
//! Convert unsigned byte and send as decimal.
void uart_SendByte( uint8_t value );
//! Convert unsigned word and send as decimal.
void uart_SendWord( uint16_t value );



#endif
```

```
/*! \file
 *
 * \brief  Polled UART communication driver for CamSat demo application.
 *
 * Device: ATmega640/128x/256x
 * CPU speed: As fast as possible
 * CPU Voltage: 3V
 *
 * $Id: uart.c,v 1.1 2006/02/27 16:54:34 apeland Exp $
 *
 * Author: Rein Anders Apeland, 2006
 * Email: apeland@mivu.no
 */
#include "uart.h"



void uart_Init( void )
{
	UBRR = UBRR_VALUE; // Set baud rate.
	UCSRB |= (1<<TXEN); // Enable transmitter module.
}



void uart_SendChar( char ch )
{
	do {} while( (UCSRA & (1<<UDRE)) == 0 ); // Wait for data register empty.
	UDR = ch; // Start transmission.
}



void uart_SendStringSRAM( char * str )
{
	while( *str != '\0' ) {
		uart_SendChar( *str++ );
	}
}



void uart_SendStringFlash( char __flash * str )
{
	while( *str != '\0' ) {
		uart_SendChar( *str++ );
	}
}
```

```
void uart_SendByte( uint8_t value )
{
	char digit0;
	char digit1;
	char digit2;

	digit0 = '0' + (value % 10);
	value /= 10;
	digit1 = '0' + (value % 10);
	value /= 10;
	digit2 = '0' + value;

	if( digit2 != '0' ) {
		uart_SendChar( digit2 );
		uart_SendChar( digit1 );
	} else if( digit1 != '0' ) {
		uart_SendChar( digit1 );
	}
	uart_SendChar( digit0 );
}



void uart_SendWord( uint16_t value )
{
	char digit0;
	char digit1;
	char digit2;
	char digit3;
	char digit4;

	digit0 = '0' + (value % 10);
	value /= 10;
	digit1 = '0' + (value % 10);
	value /= 10;
	digit2 = '0' + value;
	value /= 10;
	digit3 = '0' + (value % 10);
	value /= 10;
	digit4 = '0' + value;

	if( digit4 != '0' ) {
		uart_SendChar( digit4 );
		uart_SendChar( digit3 );
		uart_SendChar( digit2 );
		uart_SendChar( digit1 );
	} else if( digit3 != '0' ) {
		uart_SendChar( digit3 );
		uart_SendChar( digit2 );
```

```
                uart_SendChar( digit1 );
        } else if( digit2 != '0' ) {
                uart_SendChar( digit2 );
                uart_SendChar( digit1 );
        } else if( digit1 != '0' ) {
                uart_SendChar( digit1 );
        }
        uart_SendChar( digit0 );
}
```

Appendiks F Matlab-kode

Matlabkode er brukt for å motta bildedata fra kameramodulen over RS-232 (serieport) på en ordinær PC, samt for fremvisning av ukorrigert prøvebilde. Koden er skrevet, og stilt til disposisjon for prosjektet, av Stud. Tech. Halvard Kringstad - NTNU.

mottaData.m

```
%MOTTADATA
%Stiller inn matlab til å motta data på en av com-portene.
%
%Syntax: integer_vektor = mottaData(antallByte, com_port_instans)
%integer_vektor er mottatt data på seriellporten. AntallByte er hvor mange
%byte som skal tas imot og com_port_instans er en opprette instans av en
%com-port.
%
%
%*****************************************************************************
%*Av: Hallvard Kringstad, april 2006
%*Revisjon av kommentarer, Lasse Borja, mai 2006
%*****************************************************************************
function[ut] = mottaData(antallByte,comPort)
fopen(comPort);
ut(1:antallByte) = [0]; %opprettelse av tom vektor for mottatte data.
antallIterasjoner=floor(antallByte/512); %antall ganger mottaksbufferet skal legges inn i mottatte data.
rest = antallByte - antallIterasjoner*512; %Resterende antall byte.
disp('Venter på data');
for i = 1:antallIterasjoner %mottaksbufferet legges inn i mottatte data.
   ut((i-1)*512+1:i*512) = uint8(fread(comPort,512,'uint8'));
end
if rest >= 1 %hvis antallbyte ikke går opp i 512, hentes resterende byte ut til slutt.
   ut(end-rest+1:end) = uint8(fread(comPort,rest,'uint8'));
end
fclose(comPort);
```

settKomPorten.m

```
s = serial('COM2','Baudrate', 115200, 'Parity', 'none', 'StopBits', 2,'InputBufferSize', 512);
```

taBilde.m

```
%TABILDE
%Lytter på spesifisert com-port etter datastrøm. Ut fra konstantene
%bildeBredde og antallByte konverteres denne datastrømmen til en matrise
%som kan skrives ut som et bilde.
%Syntaks: tabilde(comport,valg)
%comport er objekt av type com-port og må være med som inngangsparameter.
%Ferdig initiert har navnet "s" som dermed også er innparameter: s
%Antall valg er for øyeblikket begrenset til ett valg og har defaultverdi 1.
%OBS: Skulle funksjon returnere pga time-out på serieporten må denne lukkes
%manuelt:  fclose(comPort)
%*************************************************************************
%*Av: Hallvard Kringstad, april 2006
%*Revisjon av kommentarer, Lasse Borja, mai 2006
%*************************************************************************
function[] = tabilde(comport)
antallByte = 60e3;
bildeBredde = 300;
if nargin == 0
   disp('Ett com-port-objekt må være med som inngangsvektor');
   disp('se skriptet til settKomPorten')
   return;
end

bayerMatrise = uint8(vektor2matrise(mottaData(antallByte,comport),bildeBredde));
bildeMatrise(:,:,1) = bayerMatrise;
bildeMatrise(:,:,2) = bayerMatrise;
bildeMatrise(:,:,3) = bayerMatrise;
image(bildeMatrise);
```

vektor2matrise.m

```
%VEKTOR2MATRISE:
%Konverterer en vektor til en matrise ved å dele vektoren inn i deler med
%en gitt lengde. Hver del legges inn som en rad i utgangsmatrisen.
%
%Syntax:
%matrise = vektor2matrise(vektor,bredde);
%
%*************************************************************************
%*Av: Hallvard Kringstad, april 2006
%*Revisjon av kommentarer, Lasse Borja, mai 2006
%*************************************************************************
function[ut] = vektor2matrise(inn,bredde)
%sjekk for å se om "lengde" går opp i lengden til inn.
rest = mod(length(inn),bredde) %finner rest
if rest ~= 0 %hvis det er noen rest så legges det til 'rest' antall nullsamplinger på enden av
innvektoren.
   disp('Bredde er ingen faktor i inn. Nullpadder for å kompansere');
   inn(end+1:end+bredde-rest) = [0];
end
%selve oppdelingen.
hoeyde = length(inn)/bredde; %finner høyden
ut(1:hoeyde,1:bredde) = [0]; %oppretter en tom matrise
for i = 1:hoeyde
   ut(i,1:bredde) = inn(bredde*(i-1)+1:bredde*i); %og fyller den rad for rad med data fra
inngangsvektoren.
end
```

Appendiks G Instrumentliste

Kenwood DLE1041 True RMS Programmable Multimeter, reg.nr. GR4129, serie nr. 005222
Sist kalibrert: ikke kjent.

Hewlett Packard 33120A Arbritrary Waveform Generator, reg.nr. AC4133,
serie nr.US36026873, sist kalibrert: ikke kjent.

Tektronix 724D Two Channel Digital Phosphor Osilloscope 500MHz 2GS/s,
reg. nr.KA4254, serie nr. B031435, sist kalibrert: ikke kjent.

Probe: Tek P6131 10MΩ, 10pF, 300MHz.

Strømforsyning: Harrison 6200B, reg. nr. FC4096, serie nr. 737-00194, sist kalibrert 1985

Ved oscilloscop-målinger der to like og sammenlignbare kanaler må brukes samtidig er Instek GTP-150-2, 150MHz prober brukt. Disse probene er av langt dårligere kvalitet enn oscilloscopet, men andre prober var ikke tilgjengelig fra NTNUs instrumenttjenese.