TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Testprogram till hårdvara på en viktindikator

Test program for the hardware on a weight indicator

Henrik Dahl

EXAMENSARBETE 2011

Datateknik

Postadress:
Box 1026
551 11 Jönköping

Besöksadress:
Gjuterigatan 5

Telefon:
036-10 10 00 (vx)

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom ämnesområdet Datateknik med inriktning inbyggda system. Arbetet är ett led i det treåriga högskoleingenjörsprogrammet

Examinator: Anders Arvidsson

Handledare: Rickard Holsmark
Företagets handledare: Tony Kubek

Omfattning: 15 hp (grundnivå)

Datum: 2012-09-15

Postadress:	*Besöksadress:*	*Telefon:*
Box 1026	*Gjuterigatan 5*	*036-10 10 00 (vx)*
551 11 Jönköping		

Abstract

This is a thesis made in cooperation with Flintab AB in Jönköping. The work revolves around a software, which is programmed as a tool to one of Flintabs weight indicators, weight indicator 4720.

The software has the main requirement to perform an operational test on parts of the weight indicators hardware.

The weight indicator could be compared to a normal computer in a minimized format since it has almost the same hardware. The weight indicator has hardware like processor, primary and secondary memory and touch screen etc.

The developed software is separated from the weight indicators existing software and is going to work as a plug-in module to the weight indicators software.

The software is created with the purpose to help the software development team on Flintab AB to test the hardware on the weight indicator. The software will save time and effort for the department on occasions when the weight indicators hardware needs to be tested.

Issues I predicted before the programming was the following:

How am I going to implement design and programming in terms of efficiency without contradicting other issues?

What needs to be tested on respective hardware to show full functionality?

The project is practical and mainly contains programming, the emphasis on the report is therefore mostly going to be on how and what has been programmed.

The end result of the software contains tests of the most relevant parts of the weight indicators hardware.

Sammanfattning

Det här är en examensrapport gjord i samarbete med Flintab AB i Jönköping. Arbetet kretsar kring en programvara som programmeras som redskap till en av Flintabs produkter, viktindikator 4720.

Programvaran har som huvudkrav att kunna genomföra ett funktionsdugligt hårdvarutest på delar av viktindikatorns hårdvara.

Viktindikatorn skulle kunna liknas vid en dator i miniformat då den består av ungefär samma hårdvara som en vanlig dator. Viktindikatorn har hårdvara såsom processor, sekundär och primärminne, touchskärm etc.

Den programmerade mjukvaran är separerad från viktindikatorns existerande mjukvara och kommer att fungera som en insticksmodul till viktindikatorns mjukvara.

Programvaran skapas med syftet att hjälpa programvaruavdelningen på Flintab att testa hårdvaran på viktindikatorn. Programvaran kommer att spara tid och möda för avdelningen vid tillfällen då viktindikatorns hårdvara behöver testas.

Frågeställningar innan programmering är följande:

Hur ska jag lägga upp design och programmering för att skapa ett så snabbt program som möjligt utan att gå emot andra frågeställningar.

Vad behövs testas på respektive hårdvara för att påvisa full funktionsduglighet?

Examensarbetet är i stort sätt praktiskt och innehåller till stor del programmering, tonvikten på rapporten kommer därför ligga mycket på just hur och vad som har programmerats.

Slutresultatet av programvaran innehåller tester av de väsentligaste delarna av viktindikatorns hårdvara.

Nyckelord

Inbyggda system, programvara, viktindikator, ad-omvandlare

Förord

Jag skulle vilja tacka Tony Kubek, min handledare på Flintab AB för all hjälp och handledning jag fått genom projektet, Rickard Holsmark, min handledare på högskolan, för all hjälp jag fått med rapportskrivandet.

Innehållsförteckning

1 Inledning

Det här är en examensrapport baserad på ett praktiskt projekt skrivet på företaget Flintab AB. Examensarbetet har genomförts som en avslutande del i min treåriga ingenjörsutbildning på Tekniska Högskolan i Jönköping.

Flintab AB är ett företag som ligger på Ljungarums industriområde i Jönköping. De tillhandahåller industrivågar med tillbehör och anses vara ledande inom sitt område i Sverige.

Projektet och rapporten kretsar kring en produkt från Flintab, en viktindikator som är en dataenhet med Windows CE som operativsystem. Viktindikatorn har en rad olika hårdvaror som är svåra att testa. Det finns dock en möjlighet att testa den genom mjukvaran som kommer med viktindikatorn. Nackdelen är att det är tidskrävande att gå igenom alla olika menyer för att komma åt respektive delar i hårdvaran.

För att underlätta testningen skapas alltså en mer praktiskt lösning genom ett separat testprogram för hådvaran.

Projektet är till största delen ett programmeringsprojekt och all kod är skriven i Microsofts programmeringsspråk C#.

1.1 Syfte och mål

Syftet med projektet är att utveckla ett program i C# som testar relevanta delar av hårdvaran i den Windows CE-baserade viktindikatorn. Detta för att kunna testa hårdvaran i viktindikatorn innan produkten skickas ut till kunden.

Målet med projektet är att utveckla ett program som är anpassat till personalen på mjukvaruavdelningen på flintab. Mjukvaran ska kunna testa hårdvaran snabbt och effektivt på ett sätt som slår de tidigare tidskrävande metoderna med övertygande resultat. Allting ska vara optimerat för den lilla touch-skärmen och det får därmed inte vara för mycket saker på skärmen samtidigt. Det får heller inte vara för små knappar så att det blir svårt att navigera med pekfingret.

1.2 Avgränsningar

Viss hårdvara är svårt att göra ett riktigt test på. Detta leder till att all hårdvara inte kommer att kunna testas. Dock så kommer ändå den mest relevanta hårdvaran att vara med i programvarutestet. Exempel på sådant som inte kommer att vara med är TCP-kommunikationen mellan microprocessorn och periferienhet.

1.3 Disposition

Eftersom rapporten är centrerad kring Flintabs viktindikator så kommer först en kort genomgång av viktindikatorns funktion. Sedan beskrivs viktindikatorns operativsystem samt programmeringsspråket C# som viktindikatorns programvara och mitt testprogram är skrivet i.

Efter teoretisk bakgrund kommer vi till kapitel 3 och metod och genomförande. Här gås metodiken igenom, alltså hur jag går tillväga för att uppnå önskat resultat (vilka verktyg som används m.m.).

Kapitel 4 är den lite mer intressanta delen av rapporten, här redovisas resultatet av all programmering.

Slutligen i kapitel 5 kommer jag att diskutera och utvärdera mina resultat i kapitel 4. Diskussionen kommer att föras med hänsyn till mitt mål och syfte.

2 Teoretisk bakgrund

2.1 Flintab AB

Flintab AB är ett företag som är lokaliserat på Ljungarums industriområde i Jönköping. Företaget tillverkar, servar och säljer industrivågsystem. Flintab tillhandahåller allt från mindre gramvågar till större lastbilsvågar.

Figur 2-1 Flintabs Huvudkontor

2.2 En av Flintabs viktindikatorer

Huvudsakligen består ett vågsystem av en våg och en viktindikator. Viktindikatorn (se figur 2-2) är det system som har i huvuduppgift att visa vikten på föremålet som mäts med hjälp av vågen.

Figur 2-2 Viktindikator 47-20

Flintab Weight indicator 47-20:s programvara möjliggör enklare funktioner såsom att tarrera vågen (nollställa vikten då en behållare eller container samtidigt är placerad på vågen) eller nolla vågen (sätta vikten till noll då nollvikten avviker något från 0.000).

Programvaran har också mer avancerade funktioner som till exempel DSD (Data Storage Device).

Varje vikttransaktion som sker sparas i DSD:n för att sedan skickas vidare till en peiferienhet (till exempel dator) för att utföra sin uppgift hos kunden. I DSD:n är det möjligt att kolla upp information om varje transaktion.

2.3 Teorin bakom vägning

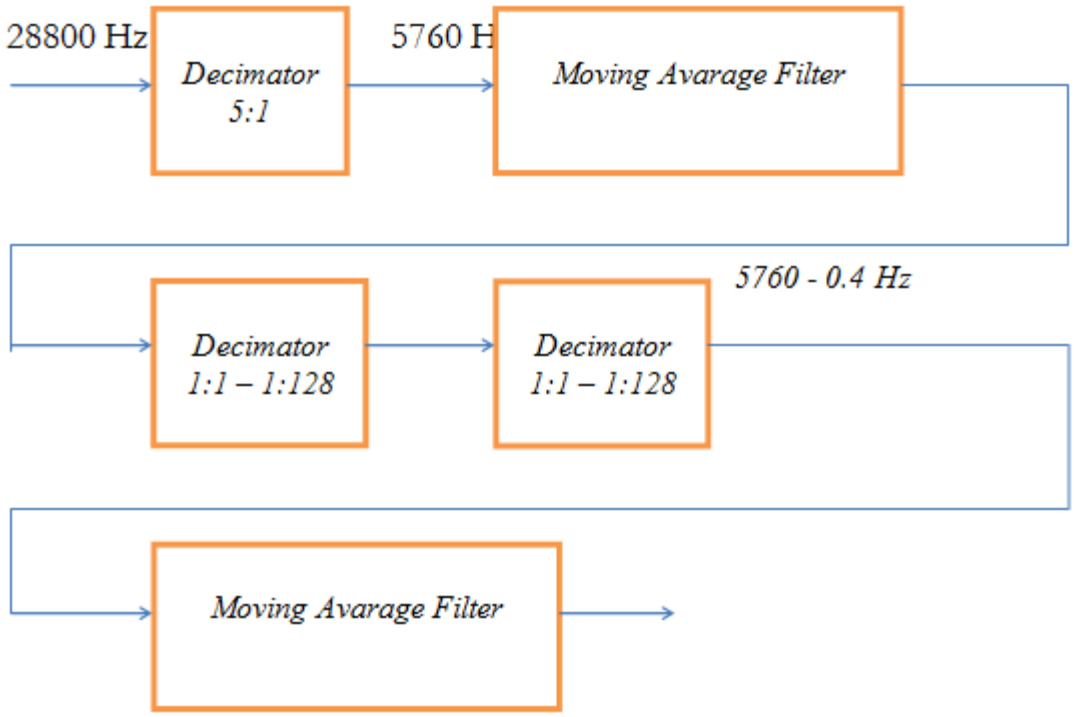
En viktindikator från Flintab har möjligheten att ta emot signaler från fyra olika lastceller. En typisk lastbilsvåg t. ex. består utav fyra lastceller med en lastcell lokaliserad i varje hörn på vågen. De multipla lastcellerna möjliggör mer komplexa beräkningar m.h.a. viktindikatorn. Ett exempel är beräkning av objektlokalisering på vågen.

2.3.1 Signal från våg till ADC

AD-omvandlaren tar emot fyra signaler från vågen. Innan signalerna når AD-omvandlaren går de igenom en OP-förstärkare. Signalerna är i behov av förstärkning eftersom AD-omvandlaren jobbar med mycket högre spänningar än de signaler som kommer från vågen och därmed kan högre upplösning på signalen uppnås.

2.3.2 Signal från ADC till drivrutiner

Viktindikatorn har drivrutiner för att skala ner antal värden /s som kommer in till viktindikatorn samt för att jämna ut signalen. Själva nedskalningen görs m.h.a. decimatorer och olika filter. Se figur 2-3 nedanför.

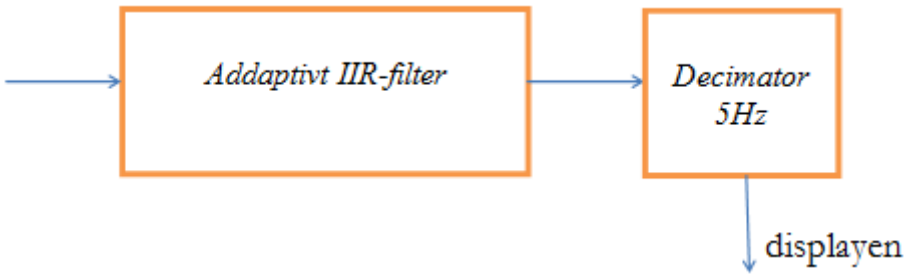


Figur 2-3 Nedskalning av signal från ADC

Den första decimatorn skalar ner singalen 5 gånger och slänger alltså 4 samples av 5. Den andra och tredje decimatorn kan slänga allt från 1 till 128 samples vardera, beroende på vad som är inställt i mjukvarans inställningar. Moving average-filtret finjusterar signalen genom att skapa ett glidande medelvärde på signalen.

2.3.3 Signal från drivrutiner till viktindikatorns mjukvara

Signalen har möjligheten att gå igenom ytterligare ett filter innan den skrivs ut på displayen. Filtret är ett adaptivt IIR-filter och används för att värdet på displayen inte ska svänga hela tiden och genom det göra det behagligare att titta på displayen. Resultatet efter decimatorn gör alltså att frekvensen alltid blir 5Hz vilket är en frekvens som är behagligt att se på för ögonen. Observera att decimatorerna i figur 2-3 inte ska gå under 5 Hz om detta filtret ska användas.



Figur 2-4 Signal skalas ner ytterligare en gång för att kunnas skrivas ut på displayen

Programvaran tar via drivrutiner emot värden från AD-omvandlaren från fyra olika kanaler samtidigt. Efter att värdena justerats för aktuell förstärkning visas lastcellernas utsignal på displayen med enheten mV/V (mV/V används ofta inom vägning och relaterar lastcellens utsignal till dess matningsspänning [1]).

Vikten visas alltså i mV/V när det är simulerade värden i testprogrammet och skrivs i kilogram i viktindikatorns programvara som är avsedd för kunderna.

2.4 A/D Omvandlare

En A/D-omvandlare är en hårdvaru-komponent som har till uppgift att omvandla en kontinuerlig analog signal till en digital signal [2]. Den analoga signalen kan vara antingen en ström eller en spänning och signalen är proportionell till det omvandlade digitala värdet.

AD-omvandlaren tar med jämna mellanrum samples från den analoga signalen innan själva omvandlingen sker. Antalet samples som tas per sekund kallas samplingfrekvens. Om t.ex. 20 000 samples tas på en sekund så är samplingsfrekvensen 20 kHz.

Är samplingshastigheten för låg så kommer den diskreta signalen skilja sig mycket från den analoga ursprungssignalen, och är samplingshastigheten för hög så kommer den diskreta signalen att ta väldigt stort utrymme när den skall sparas.

Dilemmat ligger alltså i att bestämma vilken samplingshastighet man ska välja för att förlora så lite kvalitét som möjligt och samtidigt ta upp så lite lagringskapacitet som möjligt. En vägledning på problemet får man med nyquistfrekvensen [3]. Det här innebär t.ex. att samplingshastigheten för ett telefonsamtal är 8 kHz eftersom högsta frekvensen för tal ligger på omkring 4 kHz.

En AD-omvandlares upplösning anger hur många värden den analoga signalen delas upp i. Upplösningen är alltid en tvåpotens och skulle upplösningen t.ex. vara 256 så skulle varje digital bit motsvara 6/256 = 0.0234V. Desto högre upplösning AD-omvandlaren har desto högre precision får den. Här gäller samma sak som för samplingsfrekvensen, används en högre upplösning så används också större del av hårdiskens lagringskapacitet.

Beräkningen för hur mycket lagringskapacitet (LK) som som går åt per sekund räknas ut med den enkla formeln:

$$LK = f \times b$$

Där *f* representerar frekvensen och *b* antalet bitar i upplösningen. Lagringskapaciteten för t.ex. en musikfil med 44 kHz samplingsfrekvens och 16bitars upplösning blir där med 44000 * 16 = 704 kbit/s = 88 Bytes/s.

2.5 Windows CE

Windows CE (Windows Embedded Compact) [4] är ett komplett operativsystem med inkluderad kernel. Operativsystemet stödjer processorer med Intel x86, ARM och MIPS-arkitektur.

MIPS (Microprocessor without Interlocked Pipeline Stages) [5] är en arkitektur som primärt används till inbyggda system. Spelkonsoller och routrar är vanliga exempel på inbyggda system där MIPS-processorer används i.

ARM-arkitekturen [6] var från början utformad till PC:s men den marknaden har nu istället x86-arkitekturen tagit över. ARM-processorerna används nu istället till inbyggda system eftersom de bl. a. är billiga och energisnåla. X86-arkitekturen är anpassade både till inbyggda system och PC:s.

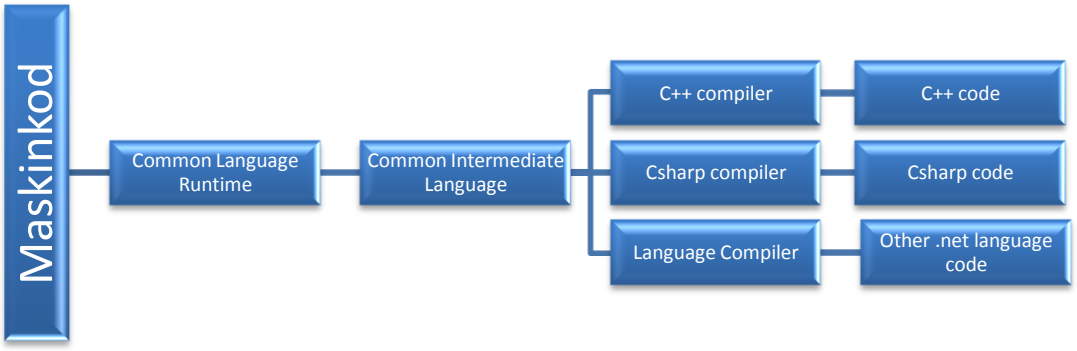
Windows CE är optimerat att köra på system med lite eller ingen lagringskapacitet (möjligt att köras på ett system med mindre än 1 MB lagringskapacitet.).

Första versionen gick under kodnamnet ”Pegasus” och hade ett windowsliknande grafiskt gränssnitt med de vanligaste windows applikationerna implementerade. Efter några års vidareutvecklande blev operativsystemet vanligare på allt fler plattformar, t. ex. i bankomater, mobiler (genom operativsystemet windows mobile) och kassaapparater.

2.6 Csharp

C# [7] utvecklas av Microsoft och första versionen släpptes i början av 2002. C# är ett komponent och objektorienterat programmeringsspråk som liknar Java, C++ och Visual Basic. Språket programmeras med hjälp av Microsofts programutvecklingsmiljö Microsoft Visual Studio. Utvecklingsteamet leds av dansken Anders Hejlsberg.

C# är en del av Microsofts ramverk dotnet (.net) vilket möjliggör många funktioner till programutvecklandet. Dotnet supportar flera programeringsspråk och tillåter språken att använda kod från andra språk inom dotnet (interoperabilitet). Alla programeringsspråk inom dotnet kompilerar sin kod till ett språk kallat Common Language Runtime som vidarekompilerar koden till maskinkod under exekvering. Se figur 2-5.



Figur 2-5 Programkompilering i dotnet.

Utvecklingsmiljön är ett så kallat drag and drop-program vilket innebär att du drar ut windowskomponenter du vill använda för att sedan implementera kod som tillhör vald komponent. Detta leder till att programet blir lätthanterligt.

Syntaxen för Csharp är liknande den för C och C++.

2.7 Seriell kommunikation genom com-port

En com-port (också kallad seriell port) är en port som överför data, bit för bit, seriellt. Förr i tiden användes com-porten till att ansluta skrivare, datormöss, kabelmodem m.m. till datorn. Numera har com-porten ersatts av USB i alla nämnda fall ovan. Com-porten används nu för tiden bl.a. till GPS-mottagare, streckkod-scanners och kommunikation mellan Microchips PIC-utvecklingskort och datorn.

Antal pinnar på en com-port kan variera, men en vanlig uppsättning är 9 stycken pinnar. De olika pinnarna kan beskådas i tabellen nedan:

Pinne	Uppgift	Signalnamn
Pin1	Olika uppgifter beroende på instrument. Används ofta till att kolla status på uppkopplingen, på/av.	DCD
Pin2	Data som tas emot.	RxData
Pin3	Data som skickas.	TxData
Pin4	Olika uppgifter beroende på instrument. Kan användas till flödeskontroll.	DTR
Pin5	Jord	Gnd
Pin6	Redo att ta emot data.	DSR
Pin7	Flow control, anger att data är klart för överföring.	RTS
Pin8	Flow control, anger att data kan föras över.	CTS
Pin9	Används till modem för att visa att telefonlinjen ringer.	RI

En seriell port kräver en del inställningar för att kommunikationen skall fungera korrekt. Inställningarna kodas i mjukvaran med hjälp av t.ex. C-sharp. De olika inställningarna är:

- Baudrate
- Databitar
- Paritetsbit
- Handskakning
- Stopbit

Baudraten bestämmer anslutningens hastighet i bitar/sekund. Vanliga hastigheter ligger mellan 1200 och 115200 bitar/sekund.

Databitar anger hur många databitar som behövs för att definiera ett tecken (för att kunna använda hela ASCII-tabellen t.ex. krävs minst 7 bitar).

Paritetsbit används till att upptäcka fel i överföringen. Paritetsbiten läggs till i datasekvensen för att t. ex. skapa ojämnt antal bitar . Kommer datasekvensen då fram med jämnt antal bitar så stämmer datasekvensen som skickats inte överens med den som mottagits.

Handskakning är en form av flödeskontroll (reglerar när data kan skickas och tas emot). Exempel på flow control är RTS/CTS som tidigare beskrivits i samband med com-porten.

Stopbiten används för att underlätta synkronisering men främst för att det skall vara möjligt att upptäcka var slutet på datasekvensen (tecknet) är.

2.8 Scrum

Scrum [8] är en inkrementell utveklingsmetod som används till att hantera projekt inom mjukvaru-utveckling. Metoden används bland annat till projekt där det är svårt att i detalj planera projektet från början till slut, vilket gör att projektet kan utvecklas under tidens gång.

Projektet delas in i så kallade sprints, där varje sprint är någonstans mellan en vecka och en månad. I början av varje sprint har utvecklingsteamet ett möte där teamet identifierar vad som ska uppnås och hur. I slutet av varje sprint-intervall går teamet igenom vad som hittills har utvecklats samt drar lärdomar om problem som har uppstått. Innan sprinten är slut görs en demonstration för kunden för att utröna hurvida mjukvaran möter de krav kunden har och för att eventuellt justera mjukvaran till kundens önskemål. Sedan går utvecklingsteamet vidare till nästa sprintsession ända tills produkten är färdig.

Ett Scrum-utvecklingsteam har de tre rollerna: Product Owner, Scrum Master, och Team Member. Product Owner ansvarar för värdet och produkten och vad som ska utvecklas. Alla Team Members bestämmer hur man ska utveckla det som produktägaren har bestämt ska utvecklas. Medan Scum Master försöker hålla teamet på rätt bana hela tiden så att gruppen kan fokusera på sina sprintmål. Rollen innebär också att bestämma regler så att teamet fokuserar på sina resultat etc. och ser till att reglerna hålls. I stort sett har man som Scrum Master som uppgift att se till att alla gör det dom ska och att alla gruppmedlemmar kan jobba så fokuserat som möjligt.

3 Metod och genomförande

För att lösa programmeringen användes ett genomtänkt och strukturerat arbetssätt, annars skulle onödiga problem kunna uppstå med risk att projektet tagit längre tid än nödvändigt. Jag använde en variant av Scrum [8] för att utvecklandet skulle bli så effektivt som möjligt.

Vanligt vis används Scrum då utvecklingsteamet är flera personer som jobbar på ett och samma projekt. I mitt fall där jag bara är en person blir det lite annorlunda. Jag får axla de tre rollerna som Product Owner, Scrum Master, och Team Member som normalt sett hade delats ut på minst 3 gruppmedlemmar. Den här varianten av Scrum kallas Personal Scrum

Mitt sätt att implementera det här arbetssättet med sprints har kommit att visa sig att varje enskilt sprint ofta har motsvarat ett test av en enskild hårdvara.

Innan varje sprint har jag har jag först tagit på mig rollen som Product Owner och identifierat vad det egentligen är som ska göras inom intervallet på 2 veckor.

Då det är konstaterat vad som ska göras börjar alltså nästa steg som är att identifiera hur allting ska lösas och för att sedan övergå till att börja själva utvecklingen. När all design och programkod är färdig, testar jag igenom allt som har gjorts hittils, för att säkerställa att allting fungerar och tillfredsställer alla krav på vad som ska göras.

3.1 Scrum sprint-plannering

3.1.1 Sprint 1 v1-2

Frågeställningar:

- Vilket är egentligen den bästa lösningen på hur jag ska testa skärmen?
- Hur testar jag kalibreringsbehovet på touch-screenen på ett snabbt och smidigt sätt?
- Hur användarvänligt behöver egentligen gränssnittet vara?

Uppgifter:

- Designa gänssnitt för att kunna rita ut figurer på skärmen.
- Designa gränssnitt till att visa manövreringsinfo.
- Koda programfunktioner som möjliggör figurritning.

3.1.2 Sprint 2 v3-4

Frågeställningar:

- Vad behöver testas för att säkerställa AD:ns funktionsduglighet?

- Finns det ytterligare information om AD-omvandlaren som kan vara intressant att veta?

Uppgifter:

- Designa gränssnitt för att visa AD-värden, brus på signal, och temperatur.
- Koda funktioner som räknar ut och/eller visar AD-värden, brus på signal, och temperatur.

3.1.3 Sprint 3 v 5-6

Frågeställningar:

- Vad behöver jag veta för att ta reda på om klockan fungerar helt?

Uppgifter:

- Designa gränssnitt för att visa tiden.
- Designa gränssnittet för att visa manövreringsinformation.
- Koda funktioner som gör det möjligt att ändra datum och tid.

3.1.4 Sprint 4 v. 7-8

Frågeställningar:

- Hur gör jag för att redovisa status av IOs tillräckligt tydlig, och hur gör jag en bra lösning för att visa att IOsen verkligen fungerar?

Uppgifterr:

- Designa gränssnitt för att visa IOs.
- Koda funktionalitet som visar status på IOs (knapp intryckt eller inte)

3.1.5 Sprint 5 v. 9-10

Frågeställnigar:

- Vad behöver jag ha reda på för att veta hurvida com-porten fungerar eller ej?

Uppgifter:

- Designa gränssnitt för att kunna: kofigurera konfigurationsbitarna, mata in data, visa data, välja dataformat, och skicka data.
- Skapa kod som implementerar designen.

3.2 Viktindikatorns hårdvara och anslutningar

Viktindikatorns baskretskort består av datormodulen PXA320 [9] från Toradex samt diverse anslutningar och periferienheter. Mer detaljerad beskrivning kommer nedan.

3.2.1 CPU

PXA320-modulen använder sig av processorn med samma namn, PXA320 från Maxwell. Processorns klockfrekvens når upp till 806 mHz.

3.2.2 SD-kort och minnen

Datormodulen har tillgång till 1GB NAND minne avsett för operativsystemet och dess applikationer samt 128MB RAM-minne.

Modulen har en SD-kort-läsare som kan ta emot alla SD-kort upp till 32GB. Detta ifall modulens något begränsade basminne på 1GB inte är tillräckligt.

3.2.3 Ethernet, USB och COM-portar

En vanlig ethernet-anslutning finns tillgänglig för att göra det möjligt att ansluta viktindikatorn till nätverk. Datahastigheten är 10/100 Mbit/s.

Det finns en USB-Host-anslutning till att koppla in periferienheter som tangentbord etc. och en USB-Client-anslutning till att koppla in en host-dator till modulen för att styra den.

Com1 och Com3-portarna tillhandahåller asynkrona signaler för seriell kommunikation. Com1-porten har möjligheten att kommunicera med både RS-232 och RS-485 medan Com3 endast kan kommunicera med RS-232.

3.2.4 LCD display

För att navigera viktindikatorn används en touch screen. Displayen är 4,3” stor och använder LED-backlight teknik.

3.2.5 Digital IO och expansionkort

Det finns möjlighet att koppla in 7 st IOs (4 outputs och 3 inputs) genom en 9-polig dsub kontakt. Det finns också möjlighet till att koppla in ett expansionkort till baskretskortet så att ytterligare 14 IOs (7 inputs och 7 outputs) kan nyttjas.

3.3 Viktindikatorns mjukvara

Viktindikatorn har inte bara till uppgift att visa vikten på ett visst föremål utan den skall också kunna göra vissa andra mer eller mindre avancerade funktioner. För att kunna utföra dessa funktioner behövs så klart en mjukvara.

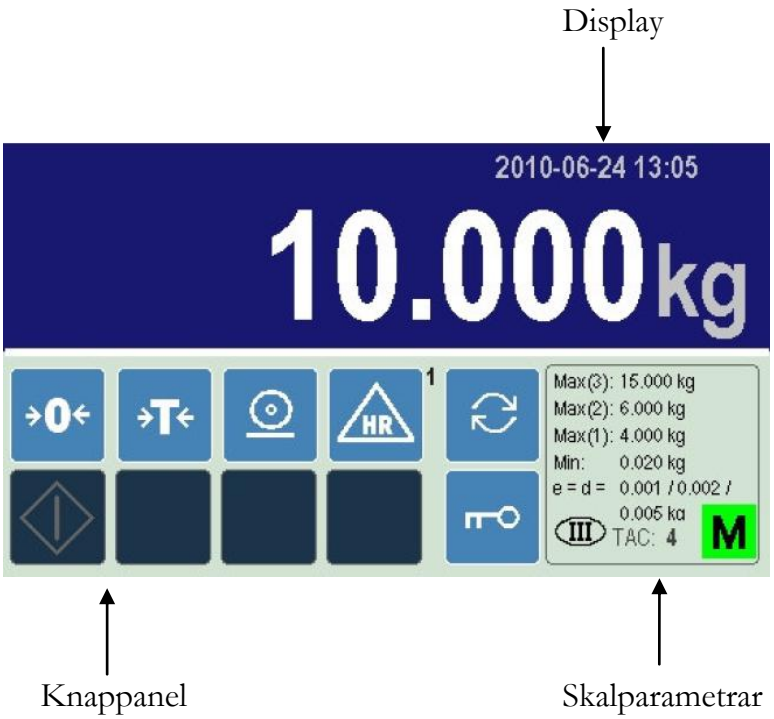
Mjukvaran kan delas in i tre delar på skärmen. Den första delen är displayytan som upptar hela den övre delen av skärmen. Här visas vikten och alla funktioner som ingår i mjukvaran. Se figur 3-1.

Den andra delen ligger ner till höger på skärmen och visar olika skalparametrar.

Den tredje delen är knappanelen som upptar resten av ytan, här finns alla funktioner till viktindikatorn. Genom att trycka på knappen längst upp till höger kommer man vidare till olika sektioner med funktioner.

3.3.1 Standard-vågfunktioner

Det finns fyra standard vågfunktioner. De är nollning, tarrering, summering och högupplöst läge.



Figur 3-1 Standard vågfunktioner

När vågen visar ett annat värde än noll av andra anledningar än att ett objekt vägs, så kan funktionen nollning användas. Vågen ställer då in sig på 0.000kg. Den här funktionen används innan ett objekt läggs på vågen.

Tarrering kan användas när nät-vikten av ett visst föremål skall visas. Det vill säga att om du väger ett flertal föremål så visas endast vikten på det föremål du sist la på vågen.

Summering gör precis som namnet på funktionen säger, den summerar vikterna på en serie vägningar.

I högupplöst läge så visas vikten med en 10x högre upplösning. Den här funktionen är aktiv i 3 sekunder innan den återgår till ursprungsupplösningen.

3.3.2 Avancerade vågfunktioner

Användaren har tillgång till två stycken avancerade vågfunktioner. Se figur 3-2. Den ena är dosering och den andra är set point quick access.

Figur 3-2 Knappar för avancerade funktioner

Set point quick access har i korta drag till uppgift att jämföra ett föremåls vikt med en målvikt för föremålet. Det finns möjlighet att ställa in fyra olika målvikter med hjälp av denna funktion.

Doseringsfunktionen används för att dosera en vikt till eller från en behållare. Den använder avancerade metoder för att kompensera för variationer i flödet på signalen etc.

3.3.3 Nyttofunktioner

Mjukvaran har också tillgång till en rad nyttofunktioner bland andra alibi memory, weight data recorder och status information. Se figur 3-3

Figur 3-3 Nyttofunktioner

Alibi memory eller DSD som det också kallas används till att spara vikttransaktioner som skickas till perifera enheter. Varje gång en vikttransaktion görs så sparas vikten undan tillsammans med ett sekvensnummer. Genom att använda sekvensnumret så kan man senare kolla upp all information i varje vikttransaktion.

Genom weight data recorder så kan viktvärden loggas vid specifika tidpunkter och intervall.

Genom funktionen status information så updateras en informationslogg varje gång en särskild händelse sker. En händelse skulle t.ex. kunna vara en varning eller ett fel.

3.4 Hjälpmedel och dess funktion

3.4.1 Simulator för ADC:n

För att testa ADC:n används en spänningsregulator som genererar en spänning. Spänningen skickas in i en kopplingslåda där den dupliceras och bildar fyra identiska signaler. Den delas upp i fyra signaler så att alla fyra kanaler på viktindikatorn kan testas samtidigt. Det kan tyckas vara konstigt att signalerna är identiska men syftet som är att testa ADC:n på viktindikatorn uppfylls oavsätt om signalerna är identiska eller inte.

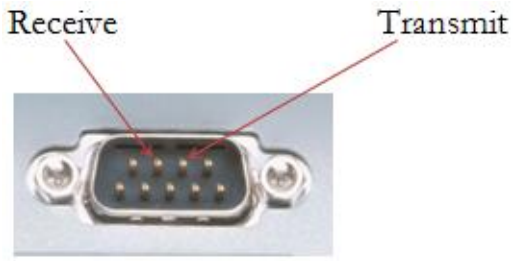
3.4.2 Knappdosa för simulering av IOs

För att kunna simulera in och utsignaler till testprogramet används en liten dosa som har knappar och en lampa. Knapparna kan simulera ettor till ingångarna på microprocessorn och en av knapparna kan även sätta en etta på en utgång på processorn så att lampan på knappdosan lyser.

3.4.3 Modifierad RS-232-kontakt

För att kunna testa com-porten används en modifierad seriell kontakt.

Kommunikation mellan com-portar sker genom att skicka data från sändarens transmit-pinne till mottagarens receive-pinne. Sätts en ledare mellan transmit och receive-pinnen i samma kontakt (se figur 3-4) så kommer all data som skickas från en enhet tas emot av samma enhet. Detta gör att testet kan utföras med endast en enskild enhet.



Figur 3-4 Transmitt kopplas mot receive för test av comport.

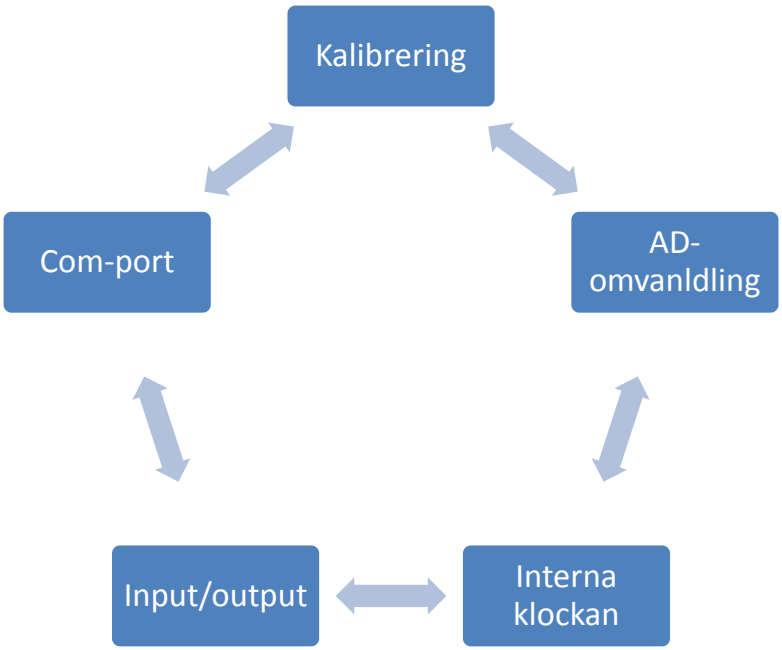
4 Resultat och analys

Programvaran som utvecklats är separerad från den existerande mjukvaran och körs som ett vanligt program. Programmet navigeras genom en meny som är placerad längst ner på displayen.

Menyn består av en framåtknapp, bakåtknapp och close-knapp vilket möjliggör smidig navigation av programmet. Test kan därmed genomföras på enskilda komponenter eller på större delar av systemet, anpassat efter behov.

4.1 Programvarans funktion och struktur

Mjukvaran för testprogramet är uppdelad i fem sektioner, strukturen på programet är cyklist på så sätt att för att få åtkomst till en viss sektion behöver användaren navigera igenom i mellanliggande sektioner (behöver dock inte genomgå något test). Programmet kan som tidigare beskrivit navigeras framåt och bakåt. Se figur 4-1. All åtkomst till viktindikatorns hårdvara görs via windows drivrutiner.



Figur 4-1 Programets funktion stegas cyklist

4.2 Programdelarnas konstruktion

Viktindikatorn innehar flera olika typer av hårdvara och för att testa dessa är det viktigt att testet utförs med hänsyn till varje specifik hårdvara. Alltså all hårdvara har olika frågeställningar och därmed behövs olika metoder för att lösa dem och skapa ett bra hårdvarutest.

4.2.1 Kalibrering av touch-screen

Innan jag programmerade touch-screen applikationen hade jag följande frågeställningar:

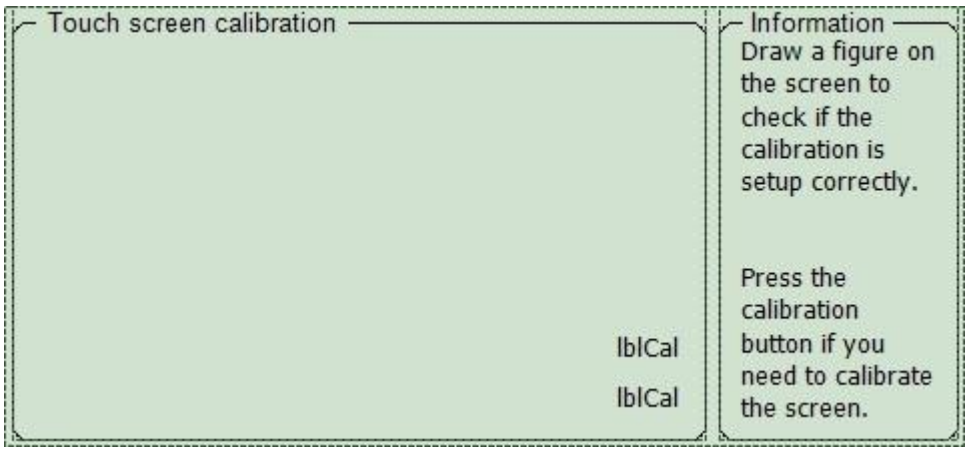
- Vilket är egentligen den bästa lösningen på hur jag ska testa skärmen?
- Hur testar jag kalibreringsbehovet på touch-screenen på ett snabbt och smidigt sätt?
- Hur användarvänligt behöver egentligen gränssnittet vara?

Det viktigaste syftet med kalibreringstestet är att veta hurvida touch-screenen fungerar eller ej. Detta gör att det finns många möjligheter på lösningar som alla uppfyller syftet någorlunda bra. Genom att rita ut något på skärmen uppfylls huvudsyftet.

Något annat som är viktigt är att få igenom testet så snabbt som möjligt. Så fort en figur har försökts ritas ut så får användaren direkt reda på ifall skärmen behöver kalibreras. Det enda fallet då det skulle ta lite längre tid är då skärmen är i behov av kalibrering. Programmet använder sig av Windows egna kalibreringsprogram för att kalibrera skärmen. Frågeställningen om hurvida testet kalibreras tillräckligt snabbt är därför både ja och nej, å andra sidan finns det inte något annat enkelt alternativ eftersom Windows egna kalibrering är det enda sättet att kalibrera skärmen.

De som skall använda programet är personerna på mjukvaruavdelningen på Flintab, programet kan därför läggas på en ganska avancerad nivå och den mest oerfarna personen kan förstå den här kalibreringsapplikationen.

Den första hårdvaran som testas är kalibreringsbehovet av touch-screenen. Touch screen-sliden är indelad i två group boxes, en för att visa information om hur testet skall utföras och en för att utföra själva testet, se figur 4-1.

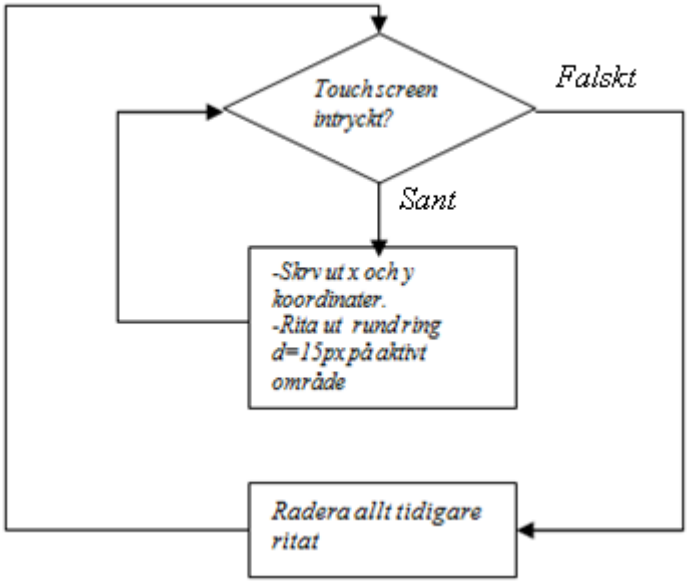


Figur 4-1 Kalibrering av touch screen

'

För att testa hurvida touch-screenen behöver kalibreras, ritar man ut en figur i den stora groupboxen. Ritas figuren ut på annat ställe än där den faktiskt ritas ut så är touch-screenen i behov av kalibrering. En beskrivning till hur ritfunktionen är kodad finns i figur 4-2. Kalibrering sker genom att trycka in kalibreringsknappen som är lokaliserad på bakdelen av viktindikatorn.

För att ytterligare beskriva var på skärmen ritpenseln är så finns där så kallade labels för både x och y-axeln ner till höger i samma group box. En ytterligare beskrivning till hur programmet är kodat kan ses på figur 4-2.



Figur 4-2 Flödesschema för test av touchscreen

4.2.2 AD-omvandlare

Frågeställningar innan programmering av AD-omvandlartestet var följande:

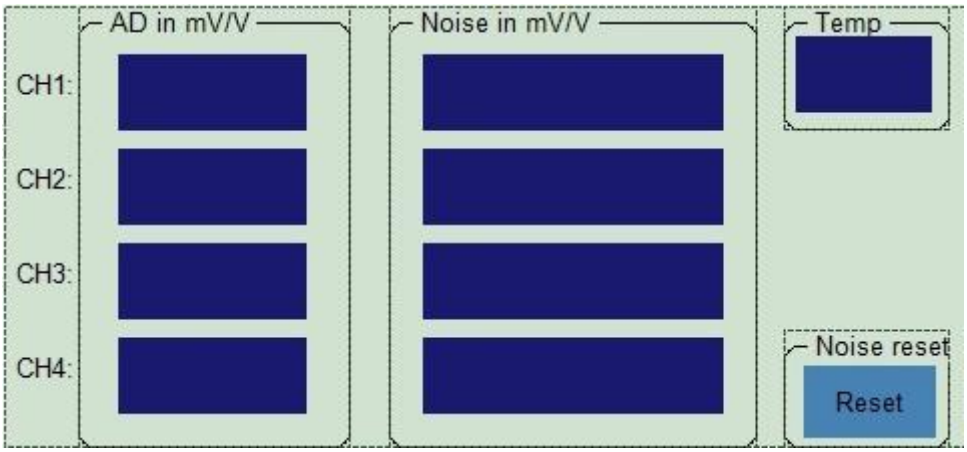
- Vad behöver testas för att säkerställa AD:ns funktionsduglighet?
- Finns det ytterligare information om AD-omvandlaren som kan vara intressant att veta?

För att veta hurvida AD-omvandlaren fungerar behöver man ha reda på vilket digitalt värde AD-omvandlaren skapar. Värdet behöver konverteras till ett värde i mV/V för att sedan kunna jämföras med det värde som simuleras in med hjälp av ADC-simulatorn.

Något som kan vara intressant att veta om AD-omvandlaren är hur mycket värdet från A/D varierar även om insignalen är konstant. Vid följande test av A/D benämns denna variation som brus.

Efter att ha testat de här två sakerna så kan AD-omvandlarens funktionsduglighet säkerhetsställas.

En bild på hur AD-sliden ser ut kan beskådas i figur 4-4. För att räkna ut bruset på signalen jämförs inkommande värde med tidigare min och max värde. Är värdet mindre än det minsta värdet eller större än det största värdet så sparas värdet undan. Bruset beräknas genom att subtrahera min-värdet från max-värdet.

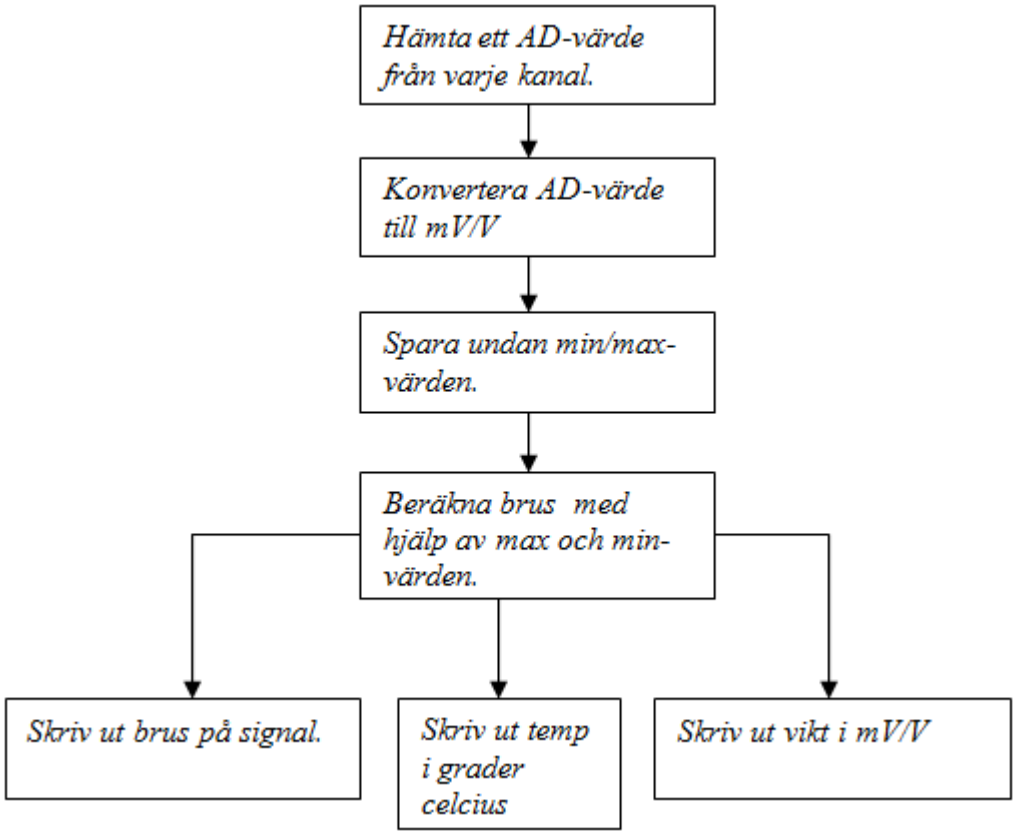


Figur 4-4 Display för AD signaler

De fyra olika kanalernas värden i mV/V printas ut i de blåa rutorna till vänster och varje kanals brus printas ut i de blåa rutorna till höger med en nogrannhet på 8 decimaler. För att nollställa bruset är det möjligt att trycka på reset-knappen varvid max och min-värdena nollställs.

En temperaturgivare finns tillgänglig innanför viktindikatorns skal och dess enda egentliga syfte för att mäta temperaturen i hårdvaran för att se så att inget blir överhettat. Temperaturen printas ut i den blåa rutan upp till höger.

Kodens händelseförlopp kan beskådas i figur 4-5.



Figur 4-5 Flödesschema för test av A/D

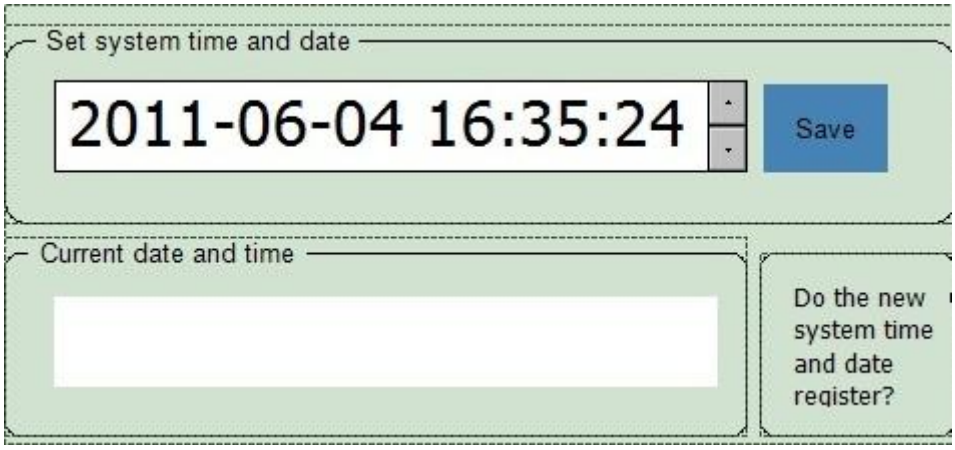
4.2.3 Intern klocka

Frågeställning:

- Vad behöver jag veta för att ta reda på om klockan fungerar helt?

Att kolla ifall klockan fungerar kan tyckas verka enkelt men hur enkelt är det egentligen? Batteriet i viktindikatorn kanske är kasst? I så fall skulle tiden nollställas varje gång viktindikatorn stängs av. För att säkerställa att klockan fungerar korrekt behöver man alltså dels kunna justera datum och tid men också på något sätt testa att klockan verkligen fungerar vid omstart av viktindikator.

Sliden för att testa den interna klockan i viktindikatorn är uppdelad i två stycken group boxes. En för att sätta en ny systemtid till viktindikatorn och en för att visa den nuvarande systemtiden. Där finns också en groupbox som visar information om hur testet skall utföras. Se figur 4-6.



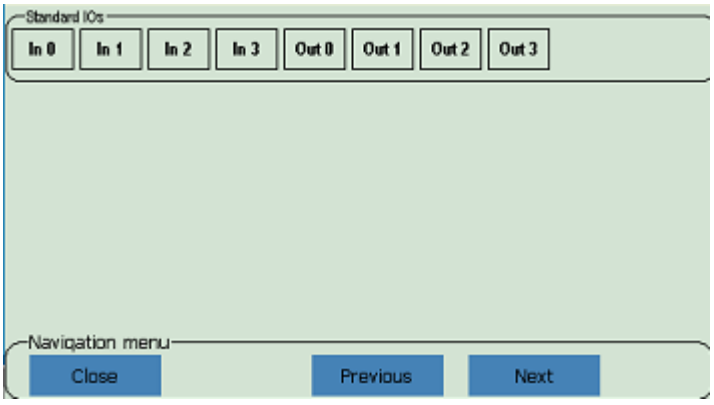
Figur 4-6 Display för test av klockan

4.2.4 Input/Output

Sliden för att testa input och outputs ser ut som bilden nedan. I groupboxen som är namngiven Standard IOs finns alla IOs som är inkopplade till viktindikatorn. IOs som kan vara inkopplade är sådana som kommer från externa tillkopplingskort, eller andra perifirienheter. För att simulera några in och utgångar används som tidigare beskrivits en knappdosa.

Knappdosan är mycket enkel och består egentligen bara av några knappar och en lampa. Ingång 3 i programmet är kopplad till utgång 0 vilket betyder att om ingång 3 trycks in så kommer utgång 0 lysa, det här kommer också visas i programmet.

Då en ingång simuleras på knappdosan lyser motsvarande ruta grönt i programmet och då en utgång triggas lyser det rött i motsvarande ruta för utgångar.



Figur 4-7 Display över Input/Outputs

Frågeställning:

- Är ovanstående lösning på att redovisa status av IOs tillräckligt tydlig och är det en bra lösning för att visa att IOsen verkligen fungerar?

Statusen på IOsen visas genom att rutorna ändrar färg till grön, röd eller ingen färg alls. Programmet ger därmed en tydlig indikation på hurvida programmet fungerar eller ej.

Viktindikatorn har möjligheten att koppla in många fler IOs än de sju IOs som visas på bild 4-7. Testet är en bra och tydlig lösning men det kan vara värt att nämna att det kommer ta någon minut extra då alla IOs (närmare en hel sida på bild 4-7) är inkopplade.

4.2.5 Terminalprogram för test av com-port

Mina frågeställningar innan jag programmerade com-porten var följande:

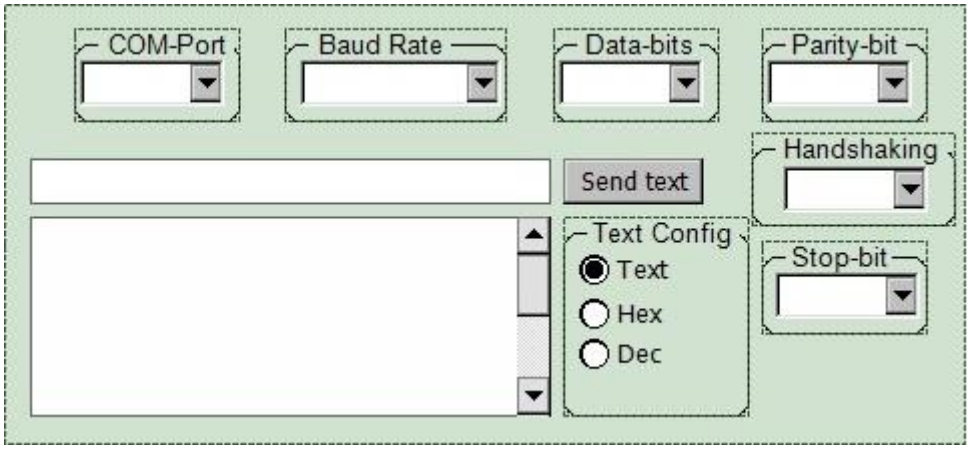
- Vad behöver jag ha reda på för att veta hurvida com-porten fungerar eller ej?

För att veta om com-porten fungerar behöver någon slags dataöverföring göras på com-porten. Dataöverföringen som görs skall kunna bli verifierad genom att jämföra mottagen data med skickad data. Det är upp till användaren hur utförligt com-porten ska testas och för att göra ett värsta fallet-scenario kan hela ascii-tabellen skickas över (0x00-0x7F)

Det vanligaste problemet en com-port har är att den inte kan överföra någon data. Det finns dock fler problem. En com-port har som tidigare beskrivit ett flertal inställningar, såkallade konfigurationsbitar. De här bitarna är bra att kunna manipulera för att möjliggöra ett fullständigt test på com-porten.

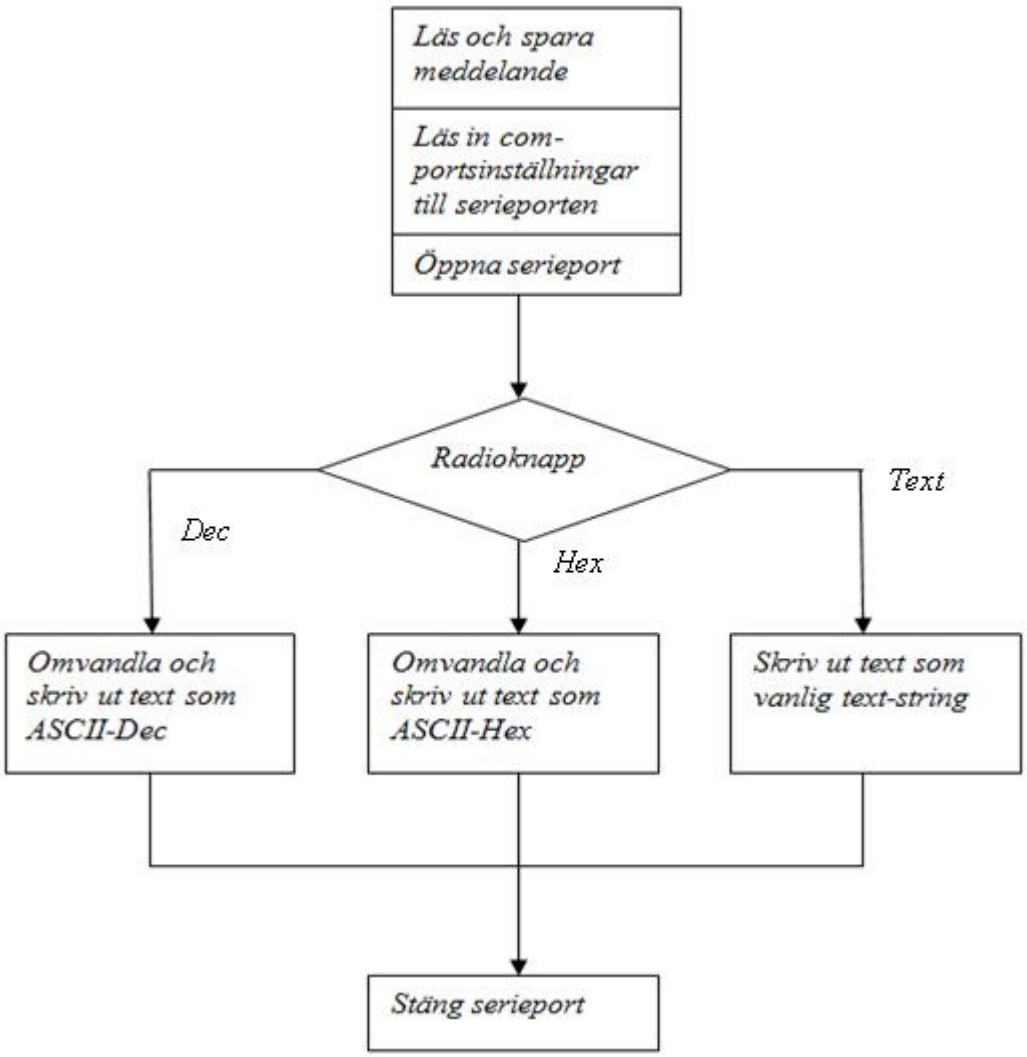
För att testa com-portens funktionsduglighet, har ett terminalprogram skapats. För att testprogramet skall fungera är det förutsatt att den modifierade RS-232-kontakten är isatt.

Programet kan delas upp i fyra delar: inställningar, skickafönster, mottagarfönster och textkonfigurering. Under inställningar ställs com-port, baud rate, data-bitar, paritets-bit, handskakning och stop-bit in. I skicka-fönstret skrivs ett meddelande in som sedan skickas vidare(till samma com-port som beskrivet i kapitel 3) genom att trycka på send-knappen. Terminalprogrammets utseende kan ses i figur 4-8 nedanför.



Figur 4-8 Display över terminalprogram

Beroende på vilken knapp som är intryckt under textkonfigureringen kommer mottagarfönstret ta emot en text i antingen ascii-hex, ascii-decimal eller vanligt string format. För ytterligare förståelse för hur den vitala delen av terminalprogramet är kodat se Figur 4-9.



Figur 4-9 Flödesschema för terminalprogram

5 Diskussion och slutsatser

5.1 Resultatdiskussion

5.1.1 Programets struktur

Jag tycker att uppdelningen av programet i 5 sektioner gör programet överskådligt och snabbt att använda. Det gör det också lätt att testa de delar av hårdvaran som för varje specifikt tillfälle är relevant att testa.

Något som skulle kunna vara lockande att göra, design och strukturmässigt, är att lägga in så många test som möjligt på en och samma sida men eftersom ytan på touch-screenen är så begränsad blir det inte användarvänligt.

Ett annat alternativ hade varit att lägga in flikar istället för knappnavigeringen men återigen så sätter ytan begränsningar. Det hade inte fått plats med många flikar eftersom touchscreenens bredd är så begränsad. Vad man i så fall hade fått göra hade varit att använda flikar tillsammans med någon slags knappnavigering för att få plats med all information.

5.1.2 Kalibrering

Frågor jag ställde mig i resultatdisskussionen:

Bästa lösningen?

Testtid?

Användarvänlighet?

Det finns flera olika sätt att testa touch-screenen på och jag sökte ett sätt som också skulle uppnå ovanstående krav.

Testet är bra därför att användaren ser om touch-screenen fungerar. Användaren behöver bara rita ut en figur på touch screenen vilket kan göras på några få sekunder.

Testet skall utföras av personerna på utvecklingsavdelningen på Flintab och ett av målen var användarvänlighet. Jag tycker att programmet är designat på så sätt att de flesta skulle kunna förstå det, inte minst en systemutvecklare. Textboxarna till höger i programmet kan klara ut de tvetydigheter som skulle kunna finnas.

Vad som möjligen skulle kunna göras på ett annat sätt är att kunna kalibrera skärmen i programmet också. Som det ser ut nu kalibreras skärmen via windows egna kalibreringsprogram, detta gör att man måste starta om testprogramet och göra om allt från början. Någonting som man i och för sig inte lägger på mycket tid eftersom kalibreringen är det första som testas men det är ändå något som skulle kunna förbättras.

5.1.3 AD-omvandlare

AD-omvandlaren är en av hårdvarubitarna som jag har lagt mest tid på eftersom det är den viktigaste delen i hela systemet. Det är också en del som till skillnad från kalibreringsdelen inte kan göras på så många sätt funktionsmässigt.

Jag väljer att rada upp alla fyra kanaler i textboxar i samma kolonn för att på så sätt möjliggöra en jämförelse med värdena på instrumentets display. Detta går att göra på flera olika sätt men ska en jämförelse göras med värden på instrumentet är det bra om de är radade på lika sätt.

Testet kan verka lite krångligt att göra men egentligen behöver man bara koppla in spänningsregulatorn till viktindikatorn, välja en spänning och så sköter programmet resten.

I testet räknar jag också ut signalens avvikelse, brus. Det här är någonting som kanske egentligen inte var helt nödvändigt men jag tycker att det är en intressant bit att veta för en AD-omvandlare. Det finns ju också en möjlighet att signalen har en stor avvikelse samtidigt som medelvärdet på signalen stämmer någorlunda, även om sannorlikheten kan tyckas vara liten.

5.1.4 Intern klocka

Det här testet kan verka ganska enkelt, det enda som behövs göras är att skriva ut testet och jämföra med rätt tid. Men riktigt så lätt var det faktiskt inte. Viktindikatorn som jag hade var faktiskt defekt även om den till synes verkade korrekt. När jag testade ifall tiden stämde så visade den rätt varje gång, men så fort jag startade om viktindikatorn så återställdes tiden till något helt fel.

Det skulle ha kunnat vara vad som helst egentligen som var fel på den så jag valde att inte göra applikationen allt för avancerad eftersom det skulle vara svårt att täcka de olika felen. Istället valde jag att göra en enkel klocka som gick att justera och helt enkelt uppmana användaren till att starta om viktindikatorn för att på så sätt få reda på om något var fel med den.

5.1.5 Input/Output

Alla IOs som registreras i programmet kommer från antingen externa tillkopplingskort eller från någon periferienhet. Antalet IOs skulle därför kunna variera från några få till väldigt många.

Jag kan tycka att det blir lite kladdigt vid de fall då det är så många IOs inkopplade att fler än en sida behövs för att visa dem. Hålls det däremot till mindre än en sida blir det ganska översiktligt att se över vilka IOs som är inkopplade.

Knappdosan är egentligen begränsad till några få ingångar och en utgång så finessen att manipulera in och utgångar går inte att använda på dom alla. Det är egentligen lite synd. För att kunna göra det skulle det krävas att en ny dosa tas fram med lika många knappar som ett tangentbord. Någonting som är långt ifrån omöjligt men kanske en aning tidskrävande.

Genom att bara ha en liten knappdosa kan endast ett fåtal in-utgångar testas men det räcker ganska långt också. Vad som också testas är att alla de externa kort registreras som hårdvara samt att samtliga ingångar registreras även om de inte kan statustestas.

Statusvisningen av IOsen görs m.h.a. att visa en grön, röd eller ofärgad ruta. Jag tycker att det här är ett väldigt tydligt sätt att visa statusen på. Det går inte att läsa fel eller på något annat sätt misstolka vilken status en viss IO har.

5.1.6 Terminalprogram

Denna del av programmet har krävt den största arbetsinsatsen programmeringsmässtigt. Mina frågeställningar innan jag skrev det var egentligen bara hur jag skulle bekräfta att com-porten fungerade.

I normalfallet behöver jag egentligen bara veta att data kan skickas över på com-porten eftersom om något fel har uppstått med kommunikationen så är det i de flesta fall inte något fel på datan utan snarare att överföringen inte fungerar över huvud taget.

Jag valde dock att göra programmet mer avancerat för att kunna fånga upp så många olika fel som möjligt.

Rent designmässigt är det enkelt att jämföra datan eftersom båda textboxarna ligger i kant med varandra. Jag gjorde också så att man skulle kunna konvertera om texten till olika textformat för att förenkla debugging. I hexadecimalt läge kan du se saker som du inte kan i vanligt text-läge. Olika tangenttryck är exempel på det (returtryck etc).

5.2 Slutsatser och rekommendationer

Den här uppsatsen har dokumenterat utvecklingen av ett testprogram för viktindikator 47-20. Den har även gått igenom teorin bakom hur vågar och dess viktindikator fungerar i teorin och framförallt hur en vågs centrala del, AD-omvandlaren fungerar. Den har också beskrivit Flintabs egna viktindikator och dess hårdvara. Framförallt har rapporten handlat om en mjukvara som testar hårdvaran i viktindikatorn. Hårdvaran som testades var touch-screen, AD-omvandlare, klocka, com-port och input/output-knappar. Rapporten har i en lagom nivå av detalj gått igenom programets uppbyggnad m.h.a. flödesdiagram och text. Rapporten avslutas med en diskussion där varje programdel har en egen diskussions-sektion där saker kring programdelen diskuteras samt för/nackdelar.

Sammantaget kan det konstateras att programvaran upfyller sitt syfte: att erbjuda korrekt och användarvänlig testav viktindikatorns hårdvara.

Programmet är programmerat på ett modulärt sätt för att underlätta vidareutveckling. Det finns mer hårdvara att testa i viktindikatorn så som nätverksanslutningar etc. som på ett enkelt sätt skulle kunna implementeras om tid finns tillgänglig.

6 Referenser

1. Norden, K. Elis. Weighting, Handbook of electronic. 1998.
2. Danielsson, Per-Erik. Digital teknik . Lund : Studentlitteratur AB, 1996. 9789144001104.
3. Nyquestfrekvensen. [Online] http://www.ni.com/white-paper/4653/en.
4. Microsoft. Windows Embedded. [Online] http://www.microsoft.com/windowsembedded/en-us/evaluate/windows-embedded-compact-7.aspx.
5. MIPS. [Online] http://www.mips.com/products/architectures/.
6. ARM. [Online] http://www.arm.com/products/processors/index.php.
7. MSDN. Microsoft. [Online] http://msdn.microsoft.com/library/z1zx9t92.
8. SCRUM. Scrumalliance. [Online] http://www.scrumalliance.org.
9. Toradex. [Online] http://www.toradex.com/Products/Colibri_Modules/Colibri_PXA320_2.0b.