FFI RAPPORT

MULTISPEKTRAL KVADRATUR DOPPLERGENERATOR

IVERSEN Alexander

FFI/RAPPORT-2002/03854

FFIE/838/113

Godkjent
Kjeller 13. september 2002

Torleiv Maseng
Forskningssjef

MULTISPEKTRAL KVADRATUR DOPPLERGENERATOR

IVERSEN Alexander

FFI/RAPPORT-2002/03854

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
Postboks 25, 2027 Kjeller, Norge

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO-2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER	2) SECURITY CLASSIFICATION	3) NUMBER OF PAGES
FFI/RAPPORT-2002/03854	UNCLASSIFIED	85
1a) PROJECT REFERENCE	**2a) DECLASSIFICATION/DOWNGRADING SCHEDULE**	
FFIE/838/113	-	

4) TITLE

MULTISPEKTRAL KVADRATUR DOPPLERGENERATOR

MULTISPECTRAL QUADRATURE DOPPLER GENERATOR

5) NAMES OF AUTHOR(S) IN FULL (surname first)

IVERSEN Alexander

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Doppler	a) Doppler
b) Generator	b) Generator
c) User Interface	c) Brukergrensesnitt
d) Programming	d) Programmering
e)	e)

THESAURUS REFERENCE:

8) ABSTRACT

Multispectral Quadrature Doppler Generator is a general purpose tool to create multiple Doppler signals. The tool is developed in Microsoft Visual C++ and includes a graphical user interface enabling the user to manually input ten signals with different frequency, amplitude, phase and noise. The tool also have a file input option for a maximum of one hundred signals. These signals are decomposed in I/Q-components and added before creating an output on a D/A-card.

9) DATE	AUTHORIZED BY	POSITION
	This page only	
13. September 2002	Torleiv Maseng	Director of Research

ISBN-82-464-0649-3

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

INNHOLD

Side

MULTISPEKTRAL KVADRATUR DOPPLERGENERATOR

1 INNLEDNING

Denne rapporten er sluttføringen av sommerjobboppgaven *Multispektral Kvadratur Dopplergenerator*, som er utført ved Forsvarets forskningsinstitutt fra 24 juni til 20 september 2002. Oppgaven faller under prosjekt 838 – EK for Luftforsvaret II. Veileder for oppgaven har vært Øyvind Thingsrud.

Multispektral Kvadratur Dopplergenerator er et generelt verktøy for summering og generering av dopplersignaler. Verktøyet er utviklet i *Microsoft Visual C++* og består av et grafisk brukergrensesnitt hvor brukeren kan taste inn opptil ti signaler med forskjellig frekvens, amplitude, fase og støy, eller hente opp til hundre tilsvarende signaler fra fil. Disse signalene blir dekomponert i I/Q komponenter, og summert før de sendes ut via et D/A-kort.

1.1 Oppgavespesifikasjon

Bakgrunnen for oppgaven var at prosjektet ønsket en generell PC-basert kompleks dopplergenerator for å modellere falske mål ved jamming av blant annet SAR-radarer. Her var kravet at dopplergeneratoren skulle kunne modellere et stort antall samtidige dopplerfrekvenser som måtte kunne varieres over tid.

Følgende spesifikasjoner ble gitt som grunnlag for oppgaven:

- Utgangsnivå / oppløsning:
  +10 dBm til –10 dBm / 1dB (gjerne valgbart dBm/V)

- Faseforskyvning I og Q-utgangene:
  +90° for positive dopplerfrekvenser, -90° for negative dopplerfrekvenser

- Frekvensområde / oppløsning:
  Minimum ±0.1 Hz til ±100.0 kHz / 0.1Hz.
  Ønskelig ±0.01 Hz til ±500.0 kHz / 0.01 Hz.
  Eventuelt dele området i to : 0.1 Hz til 1 kHz / 0.1 Hz
  10 Hz til 100 kHz / 10 Hz

- Antall samtidige frekvenser:
  Mer enn 5, helst 20, gjerne 100

- Modulasjonsområde / oppløsning:
  Amplitude 0 dB til –20 dB / 1 dB
  Fase ±180° / 1°

- Støy:
  Det bør være mulighet til å legge på fase - og amplitudestøy

- Brukergrensesnitt:
  Mulighet for oppsett av minst 5 frekvenser med modulasjon

- Import av et stort antall frekvenser fra fil med ASCII-format:
  Filformat 1 : {(t1),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)
  (t2),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)
  …
  (tn),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)}
  Hvor t = tid[ms], f = frekvens [Hz], a = amplitude [dB] og p = fase []

  Filformat 2 : {(I1,Q1),(I2,Q2),…(In,Qm)}
  Hvor I og Q er rådata på kompleks form

- Synkronisering med omverdenen:
  Synk-inngang: Sørger for at tiden settes til t = 0.
  Synk-utgang: Signal som forteller at tiden t = 0.

2 TEORETISK GRUNNLAG

2.1 Dopplergenerator

Dopplereffekten benyttes i radarer for å måle hastigheten til mål. I tillegg er det en effektiv måte å skille ut virkelige mål fra f.eks. termisk støy i luften (1). I en SAR-radar, brukes blant annet dopplerskiftene for å prosessere landbilder.

Jammere som bruker digitalt radiofrekvensminne kan lagre et mottatt radarsignal digitalt, manipulere det, og sende det tilbake som et troverdig ekko til radaren. Manipuleringen kan bestå av å legge til tidsforsinkelser og kunstige dopplerskift for å narre radaren med avstand og hastighet til målene (1). Ved jamming av SAR-radarer, som benytter ekkoets tidsforskyvning og dopplerskift for å prosesserer bildene, kan en kompleks dopplergenerator være et verktøy for å fremstille falske elementer i terrenget.

2.2 Microsoft Visual C++

I henhold til oppgavens spesifikasjoner er programvaren utviklet i Microsoft Visual C++. Dette er blant annet på grunn av at D/A-kortets rutiner støtter utvikling i nettopp dette miljøet. Microsoft Foundation Classes (MFC), som er et klassebibliotek til bruk for objektorientert utvikling av *Windows*-programmer, er brukt som utviklingsmetode.

Windows er et meldingsstyrt operativsystem, noe som betyr at kommunikasjon mellom operativsystem og applikasjoner samt applikasjoner seg imellom, foregår ved hjelp av meldinger bestående av små mengder data. Vanlige eksempler på slike meldinger kan være når man trykker på tastaturet, beveger musa, mottar data på en port og så videre. Meldingene går via en meldingskø, og blir fordelt av *Windows*-operativsystemet til de ulike applikasjonene (3). En viktig del av et *Windows*-program er å fange opp og behandle slike meldinger. Utvikling med *Software Development Kit* (SDK) (standard Win32 applikasjoner) i Visual C++ ligger nært til disse grunnleggende meldingsrutinene, og er den mest direkte måten å utvikle *Windows*-programmer på.

MFC har disse grunnleggende rutinene i SDK som underliggende rutiner, men de er mer skjult for programmereren. Dette, sammen med en rekke kodegenereringsverktøy, gjør at MFC applikasjoner er enklere og raskere å utvikle, samtidig som de har en sikrere, men litt tyngre struktur.

2.2.1 Verktøyet *MFC Application Wizard*

Visual C++ bruker *MFC Application Wizard* for å generere den grunnleggende funksjonaliteten til en applikasjon, og det finnes tre hovedtyper applikasjoner. Én type er singeldokument-applikasjoner som for eksempel det enkle tekstbehandlingsprogrammet *Notepad*. En annen type er multidokumentapplikasjoner, som eksempelvis *Word* og *Excel*. En tredje type er dialogbaserte applikasjoner, som for eksempel kalkulatorapplikasjonen i *Windows*. *Multispektral Kvadratur Dopplergenerator* er en dialogbasert applikasjon.

2.2.2 Verktøyet *MFC Class Wizard*

Verktøyet *MFC Class Wizard* er et verktøy for å behandle klassene med deres medlemsfunksjoner og medlemsvariabler.

I motsetning til SDK hvor hver *Windows*-melding som eksempelvis *WM_KEYDOWN* og *WM_KEYUP* blir behandlet i en *switch-case* struktur, kan du i MFC bruke verktøyet *Class Wizard* til å knytte slike meldinger (*mapping*) til medlemsfunksjoner i klassen. I eksempelet over vil meldingen *WM_KEYDOWN* bli knyttet til medlemsfunksjonen *OnKeyDown()*. Når *Windows* mottar og fordeler denne meldingen, vil det føre til at funksjonen *OnKeyDown()* blir kalt.

Class Wizard behandler medlemsvariablene som tilhører hvert kontrollelement i applikasjonen. Et kontrollelement kan være et menyvalg, en knapp, et inputfelt osv. Det finnes to typer medlemsvariabler; kontrollvariabler og verdivariabler. En kontrollvariabel er en forekomst av kontrollelementet i programkoden, som brukes til å kontrollere oppførselen til elementet. Det kan for eksempel være å aktivere eller deaktivere en knapp. En verdivariabel vil være knyttet til datainnholdet i et element, for eksempel innholdet i et inputfelt.

All kode som genereres i *ClassWizard* er merket grå i deklarasjons- og implementasjonsfilene, og skal ikke endres manuelt.

2.2.3 Verktøyet *MFC Resource Editor*

Visual C++ har også en ressurseditor, hvor brukeren blant annet kan bygge opp dialogboksene, tegne applikasjonens ikoner og kontrollere hurtigtastfunksjonalitet. Editoren tilbyr *drag and drop*- støtte av kontrollelementene som man vil ha i applikasjonen.

2.2.4 Visual C++ Prosjektfiler

Ved bruk av *MFC Application Wizard* blir det generert en rekke filer:

- Prosjektfil (*.dsp*) : Inneholder informasjon om Visual C++ prosjektet som applikasjonen består av.
- Arbeidsområdefil (*.dsw*): Inneholder informasjon om prosjektfilene og det øvrige miljøet (*workspace*) som er satt for det tilhørende prosjektet.
- Deklarasjonsfiler (*.h*) : Standard C++ header filer og klassedeklarasjoner
- Implementasjonsfiler (*.cpp*): Standard C++ kildefiler
- Ressurseditorfil (*.rc*): Inneholder en listing av alle *Microsoft Windows* ressurser som applikasjonen bruker. Denne filen editeres ved hjelp av ressurseditoren.
- Class Wizard fil (*.clw*): Inneholder informasjon om klassene som brukes av *Class Wizard*- verktøyet.
- Ikonfil (*.ico*): Ligger under */res* katalogen, og inneholder applikasjonens ikon. Ikonet kan endres i ressurseditoren.
- Ressursfil (*.rc2*): Ligger under */res* katalogen. Her kan brukeren legge alle ressursene som ikke skal kunne editeres ved hjelp av ressurseditoren.

2.3 National Instruments *NI-6731*

D/A-kortet som er brukt i denne oppgaven er et *plug & play*, analogt output, digital- og timing I/O-kort for PCI bus. Noen viktige spesifikasjoner er:

- 4 kanaler for analogt output
- 16 bit digital til analog konverterer pr kanal
- Oppdateringsrate på maksimalt 1,000,000 sampler pr sekund pr kanal
- Innebygd timer kontroller
- Innebygd FIFO buffer på 8,192 sampler à 16 bit
- 8 digitale I/O kanaler for generell bruk

Kortet gir også mulighet til å sette opp ekstern referanse pr kanal for å justere spenningssvinget samtidig som man beholder spennet på 16 bit. Den eksterne spenningsreferansen kjøres inn på den aktuelle kanalen på kortet, og kan enten komme fra en ekstern kilde, eller direkte fra en av de analoge utkanalene på kortet. Uttrykk 2.1 og 2.2 viser eksempler på minste oppløsning med ekstern referanse på henholdsvis 10 volt og 1 volt . Fordi spenningen er bipolar vil ekstern referansespenning på for eksempel 10 volt ha et spenn på 20 volt. (-10 volt til +10 volt).

$$\textit{Minste oppløsning med ekstern referanse på 10 V}: \frac{20V}{65{,}536} = 305.1\mu V \tag{2.1}$$

$$\textit{Minste oppløsning med ekstern referanse på 1 V}: \frac{2V}{65{,}536} = 30.51\mu V \tag{2.2}$$

Med kortet følger drivere og funksjoner som støtter utvikling i *Visual C++*.

3 METODE

3.1 Administrasjon

3.1.1 Organisering

Oppgaven har vært atskilt fra resten av prosjektutviklingen, i den hensikt at den kunne utføres selvstendig selv om flere av prosjektmedarbeiderne var borte i forbindelse med ferieavvikling.

3.1.2 Endring og avgrensing

Endringer i spesifikasjonene og forslag til endringer/utvidelser i funksjonalitet til dopplergeneratoren har blitt godtatt og forsøkt gjennomført i den grad tiden har strekt til.

3.1.3 Møteaktivitet

Jeg har deltatt på to prosjektmøter. Det første var 27 juni 2002, hvor forslag til grafisk brukergrensesnitt og generell funksjonalitet ble presentert. Det andre møtet var den 5 september 2002, hvor resultatene så langt ble presentert. I tillegg har det blitt ført en jevnlig dialog med veileder.

3.1.4 Overlevering

Rapporten blir distribuert som ugradert intern FFI rapport. To CD-er med eksekverbar applikasjon, kildekode, dokumenter og drivere for D/A-kortet blir overlevert til prosjektet. På CD-ene finnes også en testversjon av dopplergeneratoren som ikke krever at D/A-kortet og driverne er installert på maskinen.

3.2 Teknisk

3.2.1 Ressurser

Følgende maskinvare og programvare har vært benyttet:

- Maskinvare:
  PC koblet til FFIs utenett med tilgang til Internett
  PC koblet til FFIs innenett
  National Instruments NI-6731 D/A PCI innstikkskort
  Oscilloskop

- Operativsystem:
  Microsoft Windows XP

- Programvare:
  Microsoft Visual C++
  Microsoft Office 2000
  National Instruments NI-6731 drivere og rutiner for *Visual C++*.

3.2.2 Fremgangsmåte

For å få en felles forståelse for hvordan dopplergeneratoren skulle realiseres utover de skriftlige spesifikasjonene, ble et forslag til grafisk brukergrensesnitt utviklet. Det ble fremvist på et prosjektmøte tidlig i fasen for å få innspill og ideer fra veileder og de potensielle brukerne. Deretter har en evolusjonær utviklingsmetode blitt brukt. Det vil si at brukergrensesnittet gradvis har blitt tilført funksjonalitet.

3.2.3 Språkkonvensjon

På starten av alle filer finnes en seksjon med kommentarer som oppsummerer innholdet i filen, og eventuelt informasjon om klasser som er opprettet i filen. Hver medlemsvariabel og funksjonsvariabel har en tilhørende kommentar som beskriver hensikten med variabelen. Hver funksjon er kommentert med inn parametere og returverdi, samt en beskrivelse av hensikten til funksjonen. Kommentarer er også lagt til der virkemåten/hensikten ikke er åpenbar.

Navn på medlemsvariabler, det vil si variabler som tilhører klassen, starter alle med *m_...* for å angi at de er medlemsvariabler, og deretter *str...*, *srt...*, *int...*, *dbl…*, *flt...*, osv for å angi datatype. Navn på medlemsvariabler som er pekere til datatyper begynner med *m_p...* og deretter navnet på datatypen. Eksempler på medlemsvariabler av typen *float* vil derfor bli *m_fltVariable1*, *m_fltVariable2* osv., og medlemsvariabler som er pekere til en variabel av typen *float* blir dermed *m_pfltVariable1*, *m_pfltVariable2* osv.

Navn på Lokale variabler, det vil si variabler som er opprettet i funksjoner, følger samme konvensjon, men starter ikke med *m_...*.

Alle medlemsfunksjoner starter med stor bokstav, og har en stor bokstav for alle naturlige delenavn som funksjonen består av, eksempelvis *UpdateStatusWindow(...)*.

Alle klasser starter med stor C for *Class*, og følger for øvrig konvensjonen for funksjonsnavn. Navnet på egendefinerte klasser som arver fra standard MFC klasser som *CDialog*, *CEdit*, *CComboBox*, osv., slutter med navnet på morklassen. Eksempler på dette er *CSysVarDlg*, *CFloatEdit* og *CFloatLimitComboBox*.

All kode og alle kommentarer er på engelsk.

3.2.4 Sikkerhetskopiering

Sikkerhetskopier har blitt tatt minst daglig, og gjerne flere ganger om dagen i de tilfeller hvor betydelige endringer har blitt gjort.

Hver sikkerhetskopi har inneholdt alle prosjektfilene, bortsett fra de *debug*-filene og den eksekverbare applikasjonen som *Visual C++* kan generere ut fra prosjektfilene. Hver sikkerhetskopi har blitt merket med dato og blitt lagret på brukerkontoen på FFIs filserver.

4 RESULTATER

Applikasjonens klassestruktur og den teoretiske bakgrunnen for programløsningene blir beskrevet i dette kapitlet. I tillegg gis en veiledning og forklaring på både installasjon og bruk av dopplergeneratoren.

4.1 Installasjon

4.1.1 Operativsystem

Driverne for *NI-6731* er for *Windows 2000/NT/XP/Me/9x*. Dopplergeneratoren skal kunne kjøre på alle 32 bits *Windows*-operativsystem.

Forøvrig har det oppstått problemer ved installasjon av D/A-kortet på operativsystem som er installert med *image*- disker fra FFIs IT-tjeneste *Munin*. For at kortet skal fungere, bør derfor operativsystemet bli installert fra originale *Windows*-CD-er.

4.1.2 Installasjon av *NI-6731*

Driverne til kortet, *NI-DAQ 6.9.2 Device Drivers*, finnes på CD-en som fulgte kortet. For å bruke dopplergeneratoren er det tiltrekkelig kun å installere *NI-DAQ 6.9.2 Device Drivers*. Ved videreutvikling/endring av dopplergeneratoren, bør også *Microsoft Visual C/C++ Support* installeres.

Etter driverinstallasjon skal ikke maskinen restartes, men slås av for innsetting av D/A-kortet. Kortet er et *plug & play* PCI-kort, og plasseres i en ledig PCI slot på maskinen. Etter å ha satt i kortet, startes maskinen, og installasjonen vil fortsette. Etter at installasjon er fullført vil *Measurement & Automation Explorer* (MAX) startes, hvor kortets innstillinger kan endres. MAX har også et testprogram som bør kjøres for å forsikre om at kortet fungerer.

Utgangene som brukes på *NI-6731* er:

- *DAC0OUT*: Analog utgang for signalets I komponent.
- *DAC1OUT*: Analog utgang for signalets Q komponent
- *DAC2OUT*: Analog utgang for styring av ekstern referansespenning
- *EXT_REF*: Analog inngang for ekstern referansespenning

For å styre referansespenningen i dopplergeneratoren må *DAC2OUT* utgangen kobles direkte til *EXT_REF* inngangen.

4.1.3 Installasjon av dopplergeneratoren

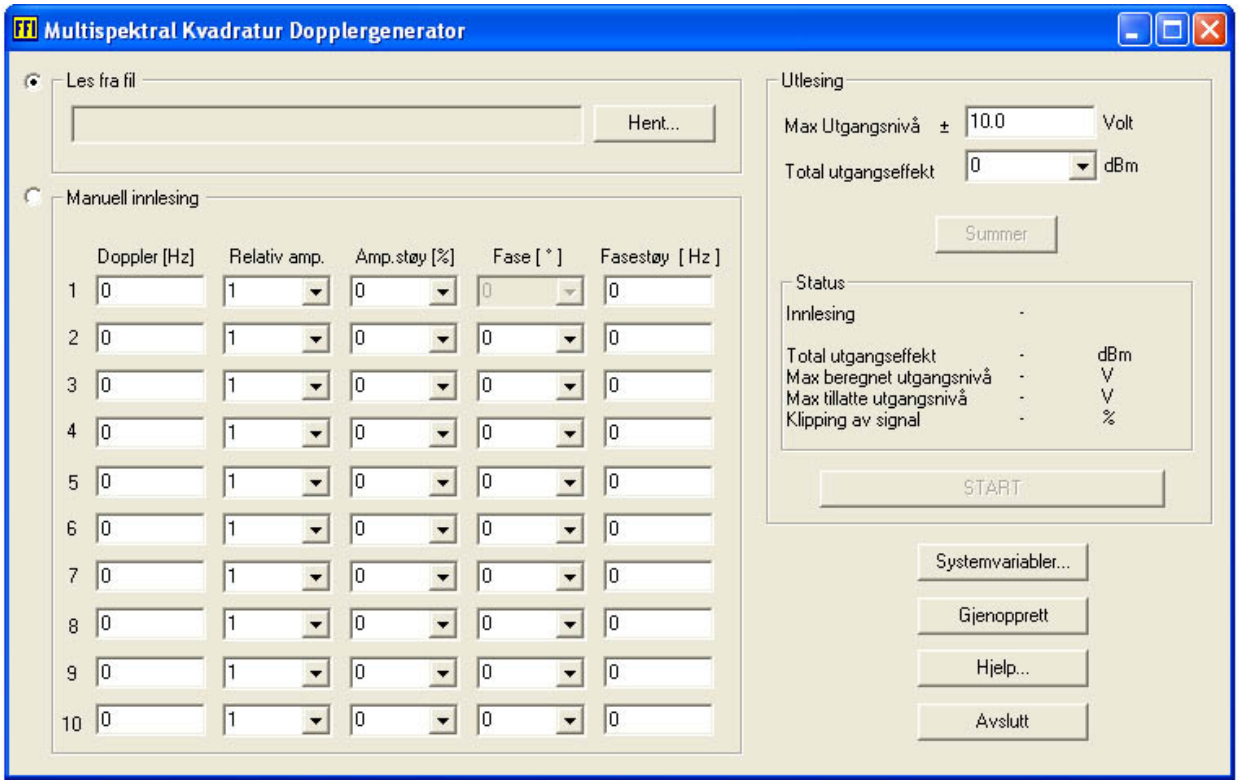
Multispektral Kvadratur Dopplergenerator bruker statisk linket MFC bibliotek, noe som betyr at applikasjonen kan kjøre på maskiner som ikke har installert *Microsoft Visual C++* og dermed ikke har MFC biblioteket. Derimot må *NI-DAQ 6.9.2 Device Drivers* være installert på maskinen, for at applikasjonen skal kunne starte opp. Dopplergeneratoren består av én eksekverbar fil; *MKDopGen.exe*.

Alle *Visual C++* prosjektfiler og kildekode er også tilgjengelig. Ved videreutvikling/ endring og rekompilering av dopplergeneratoren, kan hele prosjektarbeidsområdet åpnes med filen *MKDopGen.dsw*.

4.2 Brukergrensesnittet

Dopplergeneratoren er en dialogbasert applikasjon, og består hovedsakelig av to dialogbokser; hoveddialogboksen og systemvariabler. Virkemåten til applikasjonen blir beskrevet her.

4.2.1 Hoveddialogboksen



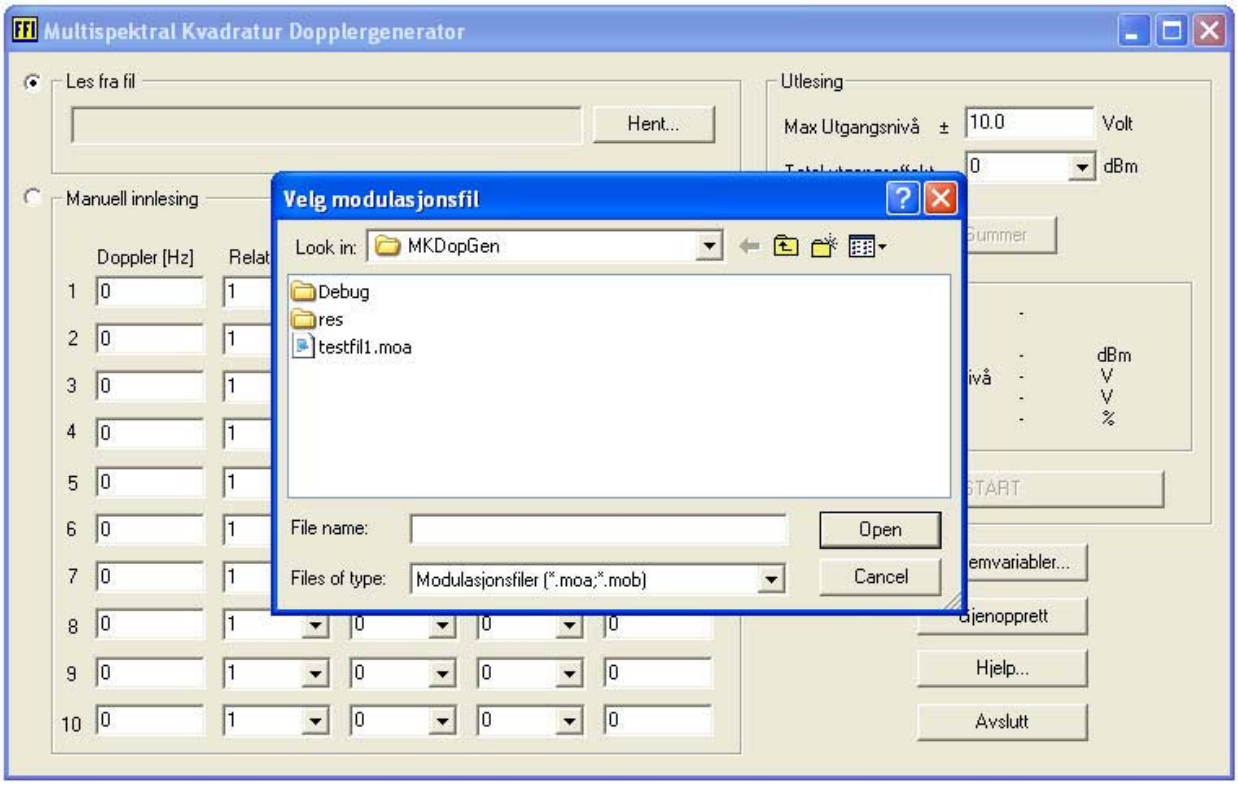
Figur 4.1 Hoveddialogboksen ved oppstart

Når applikasjonen starter vil hoveddialogboksen komme opp (figur 4.1). Den er visuelt inndelt i fire seksjoner. Seksjonen øverst til venstre brukes til filinnlesing av dopplersignaler. Under denne er det inputfelt for manuell innlesing av dopplersignaler. Øvre høyre del av vinduet er for utlesing. Her settes noen utgangsparametere, og her er knappene for summering av signalene, starting og stopping av signalgenereringen. I tillegg finnes et statusvindu som gir brukeren informasjon om sine valg. Nederst til høyre finnes programknapper for å sette systemvariabler, gjenopprette initialverdiene, åpne hjelpedialogboks, samt avslutte applikasjonen.

Inputfelt der brukeren kan taste inn verdier, er hvite. Noen felt har rullemeny for valg av verdier, og i disse feltene kan brukeren enten velge en verdi fra rullemenyen eller skrive de inn manuelt. Alle rullemenyfelt vil kun ta imot heltall, de andre kan ta imot desimaltall med en desimal.

4.2.1.1 Filinnlesing

For å lese inn signaldata fra fil, må man trykke på ”Hent…”- knappen. Da åpnes en standard *Windows*-dialogboks for filåpning, som vist på figur 4.2.

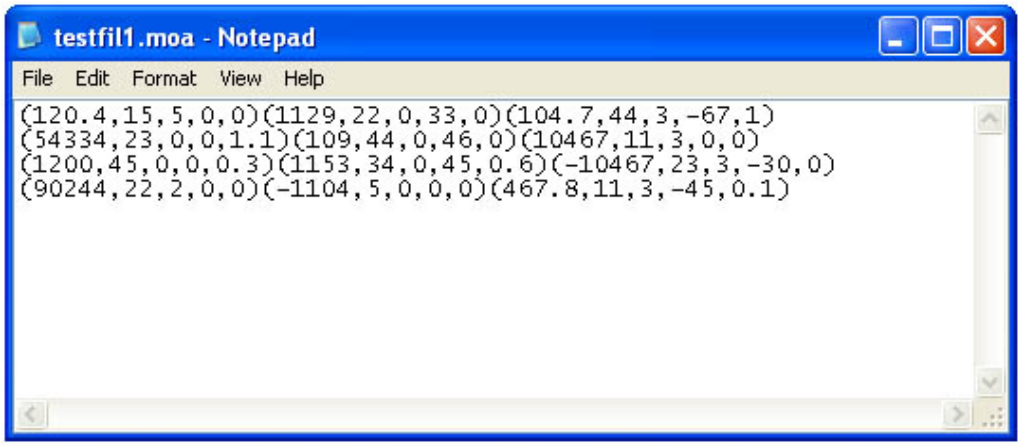


Figur 4.2 Henting av fil

Det er bare mulighet for å hente filer av typen *.moa* og *.mob*. Disse skal representere modulasjonsfil type A, og modulasjonsfil type B. Foreløpig er applikasjonen bare satt opp for å lese én type filformat, som er lik *filformat 1* angitt i oppgavens spesifikasjoner, men uten tidsvariabelen. Dette filformatet må derfor være på følgende form:

(f1,a1,an1,p1,pn1)(f2,a2,an2,p2,pn2)…(f100,a100,an100,p100,pn100)

Hvor *f* er dopplerfrekvens i Hz, *a* er relativ amplitude, *an* er amplitudestøy i prosent, *p* er fase i grader og *pn* er fasestøy i Hz. Desimaltall oppgis med punktum. Det kan godt være mellomrom eller linjeskift mellom tegnene. Applikasjonen gir ingen feilmelding om filformatet er feil. Figur 4.3 viser et eksempel på filformatet som applikasjonen godkjenner.

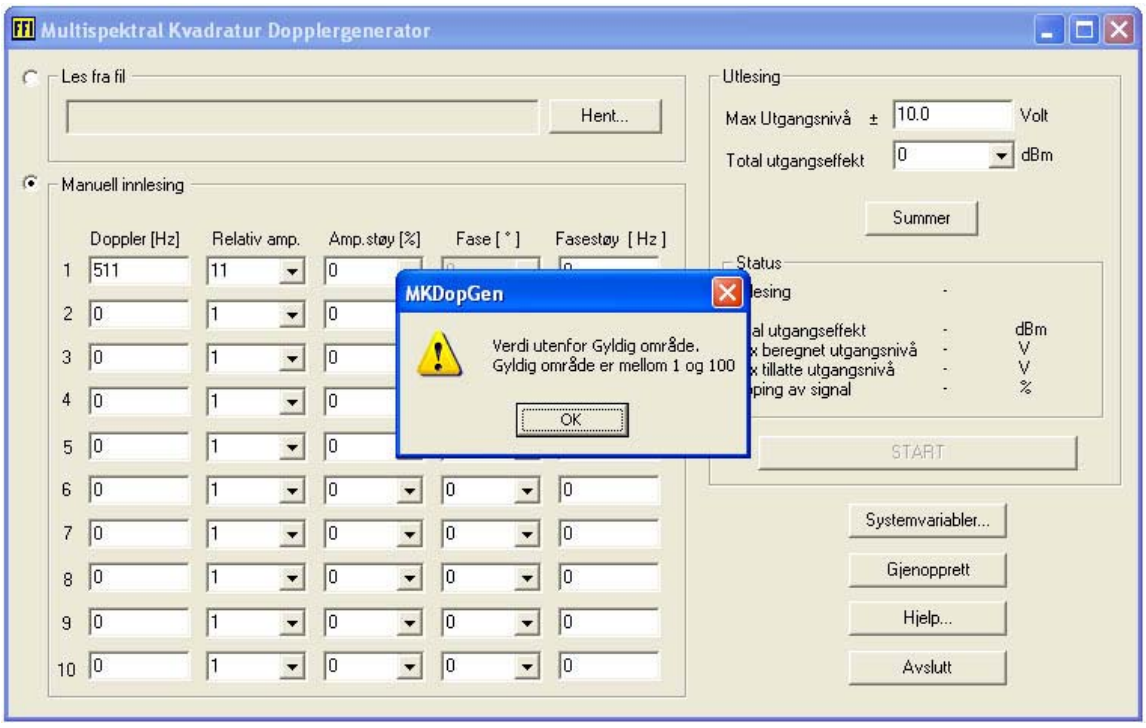


Figur 4.3 Eksempel på et modulasjonsdata fra fil

En fil velges ved å dobbelklikke på filen, eller ved å merke den og trykke på ”OK” i fildialogboksen. Da lukkes fildialogboksen, og hoveddialogboksen vil vise filnavnet i det grå inputfeltet oppe til høyre. Dette inputfeltet er grått fordi det ikke er editerbart. Radioknappen som bestemmer om det leses fra fil eller manuelt vil automatisk bli valgt til filinnlesing. Radioknappen for filinnlesing kan ikke velges hvis det ikke er valgt noen fil, eller hvis det er valgt en tom fil. Når fil velges vil summeringsknappen bli aktivert slik at brukeren kan summere innholdet i filen.

4.2.1.2 Manuell innlesing

Manuell innlesing skjer ved at brukeren taster inn opptil ti signaler med dopplerfrekvens, amplitude, eventuell amplitudestøy, fase og eventuell fasestøy. Signaler med dopplerfrekvens på null Hz vil ikke bli medregnet. En negativ dopplerfrekvens betyr et negativt dopplerskift. Hvis brukeren taster inn verdier utenfor de grenseverdiene satt i systemvariablene, vil en feilmelding komme opp, som vist i figur 4.4.



Figur 4.4 Feilmelding ved inntasting av verdi utenfor gyldig område

Brukeren kan når som helst skifte mellom manuell innlesing og filinnlesing, ved å trykke på de tilhørende radioknappene. Summeringsfunksjonen vil hele tiden summere signalene tilhørende radioknappens posisjon.

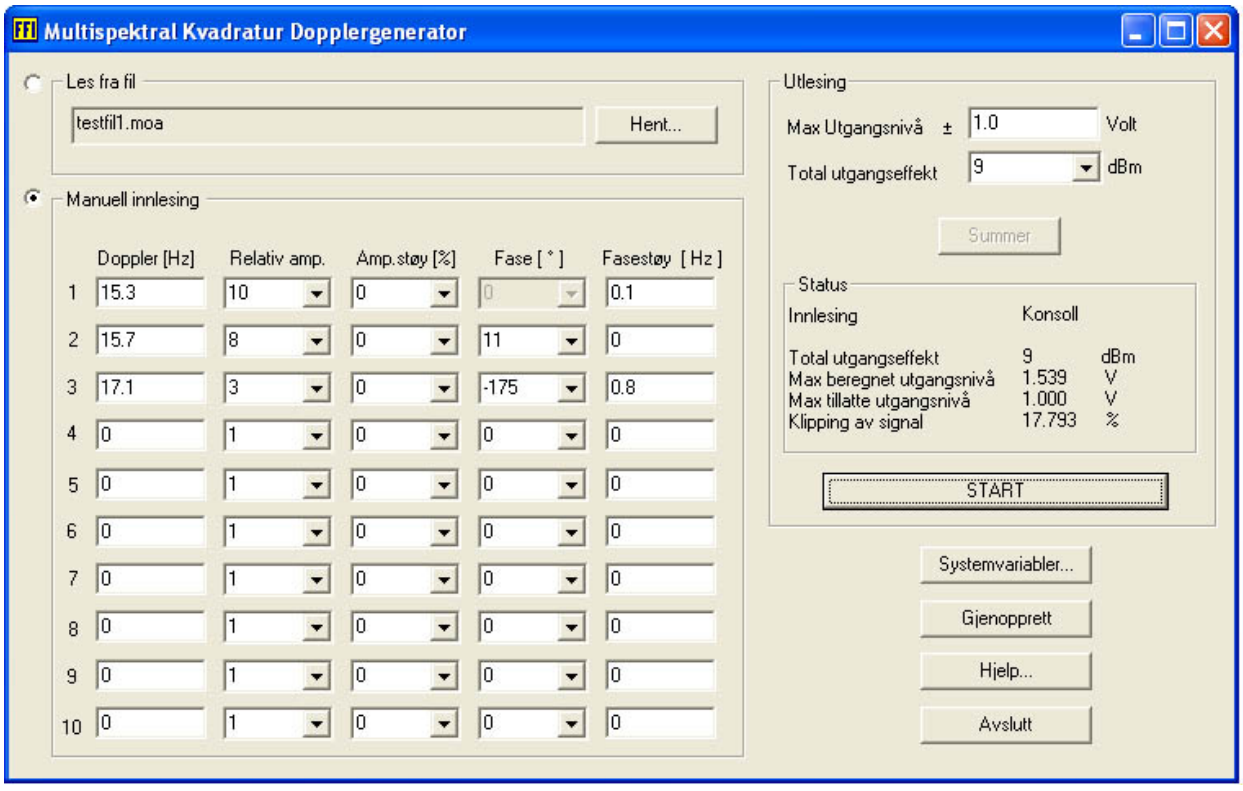
4.2.1.3 Utlesing

Max utgangsnivå bestemmer maksimalt tillatte voltamplitudeverdi. Utgangssignalet vil bli klippet i de tilfeller hvor det summerte signalet overstiger denne verdien. *Total utgangseffekt* er den effekten som det komplekse signalet vil ha, uavhengig av hvor mange delsignaler det består av.

Summeringsknappen vil være deaktivert i de tilfeller hvor det ikke er noe å summere, og ingen endringer er gjort. Da er verdiene i inputfeltene summert. Når summeringsknappen er aktivert, betyr det altså at endringer av delsignalene er blitt foretatt. Når summeringen pågår vil musepekeren være et timeglassymbol, og input i applikasjonen er da ikke mulig.

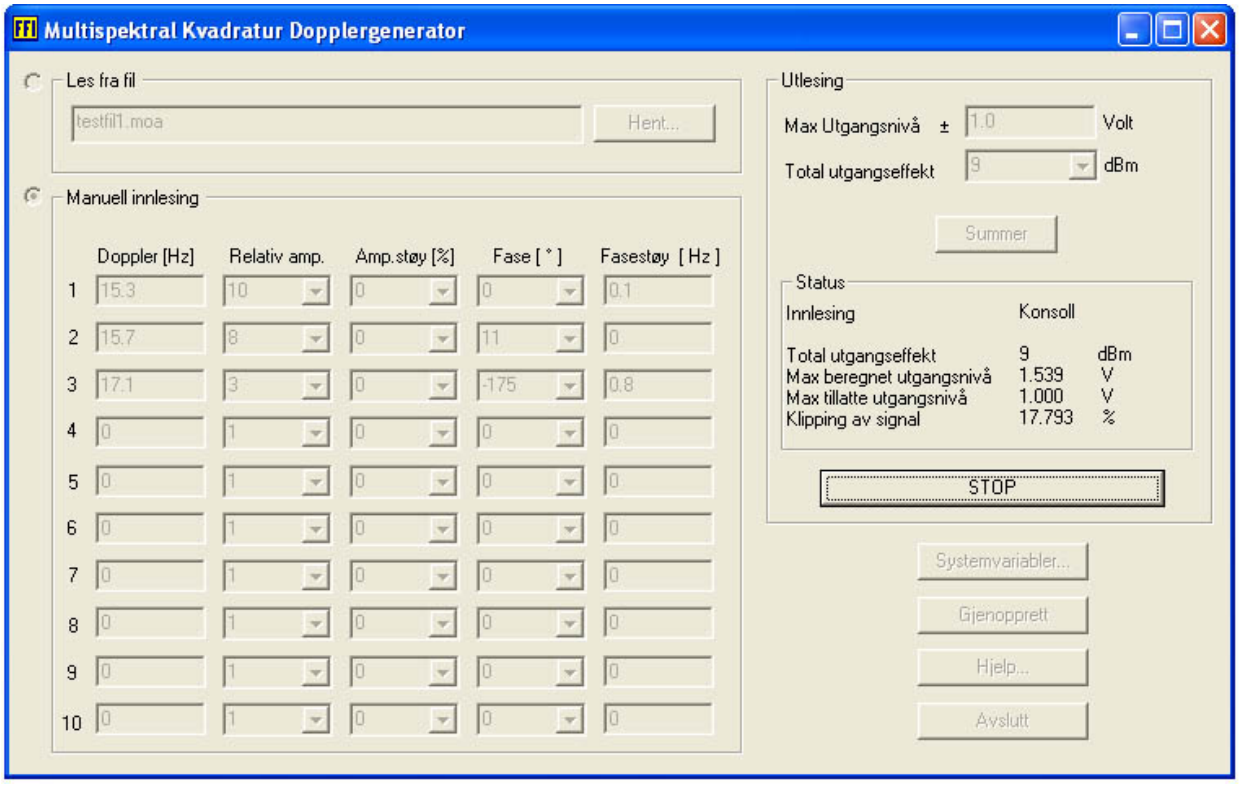
Resultatet av summeringen blir skrevet ut i statusvinduet som vist i figur 4.5, og gir følgende informasjon:

- Om signalene er lest fra fil eller fra konsollen, og eventuelt filnavnet
- Den totale utgangseffekten i dBm
- Det maksimale beregnede utgangsnivået i volt
- Det maksimale tillatte utgangsnivået i volt
- Klipping av signalet i prosent. Dette forteller hvor mye av det summerte signalet som har blitt klippet av på det maksimalt/minimalt tillatte utgangsnivået.



Figur 4.5 Statusvinduet etter summering

Når summering er fullført, blir summeringsknappen deaktivert, samtidig som at startknappen blir aktivert. Når startknappen trykkes inn, kjøres I og Q komponentene av signalet ut på henholdsvis kanal *DAC0OUT* og *DAC1OUT* på D/A-kortet inntil knappen trykkes inn igjen (knappen skifter navn til "Stopp" når genereringen pågår). Når genereringen pågår vil alt annet en stoppknappen være deaktivert. (figur 4.6).



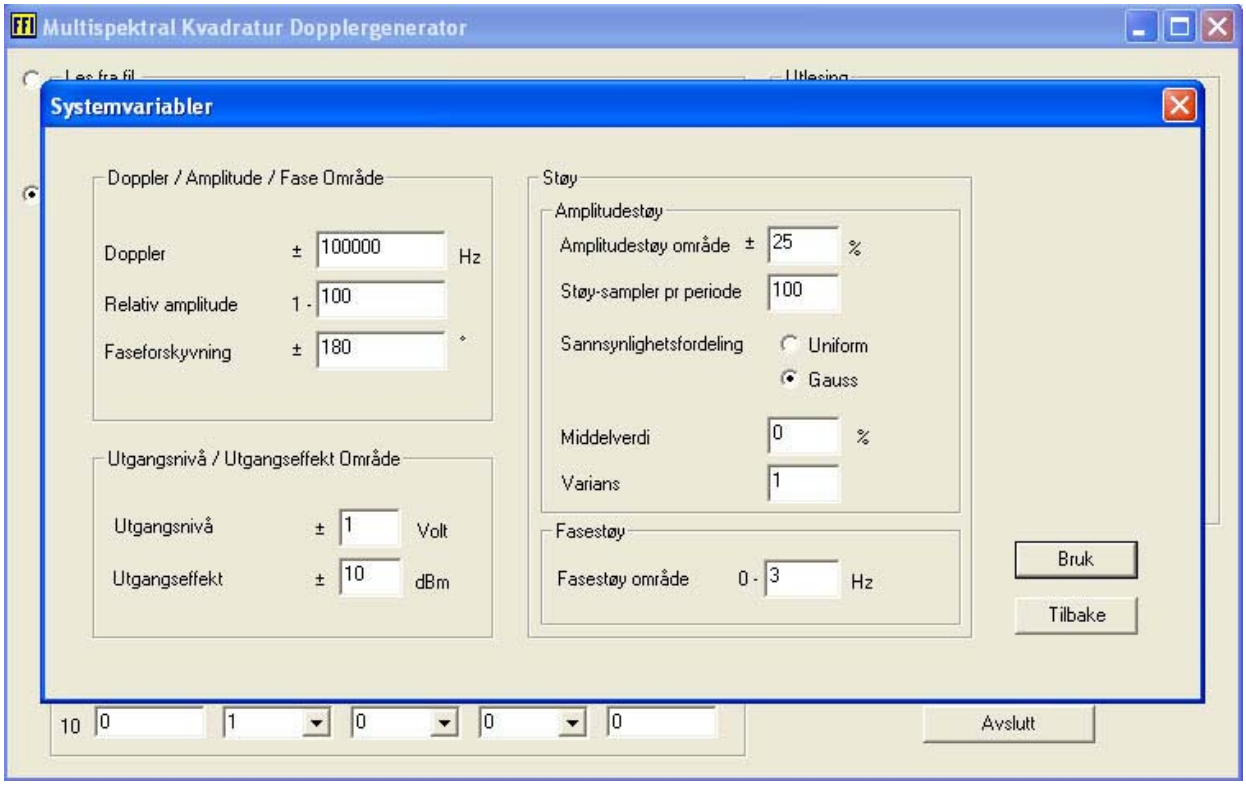
Figur 4.6 Generering pågår. Det komplekse signalet kjøres ut på D/A-kortet

4.2.1.4 Øvrige programknapper

De resterende knappene i hoveddialogvinduet har følgende funksjoner:

- "Systemvariabler…" åpner systemvariabeldialogboksen.
- "Gjenopprett" gjenoppretter hoveddialogboksen med initialverdiene.
- "Hjelp" åpner hjelpedialogboksen.
- "Avslutt" lukker hoveddialogboksen og avslutter applikasjonen.

4.2.2 Systemvariabler



Figur 4.7 Systemvariabler

Systemvariablene (figur 4.7) er visuelt delt inn i tre seksjoner. Øverst til venstre settes det lovlige området for dopplerfrekvens, relativ amplitude og fase opp. I seksjonen under, settes det lovlige området for utgangsnivå og utgangseffekt opp. Seksjonen for støy er delt i amplitudestøy og fasestøy.

Amplitudestøyområdet bestemmer det lovlige området for amplitudestøy. Støysampleoppløsningen bestemmer hvor mange ganger et støysignal kan oppstå pr periode. Sannsynlighetsfordelingen av de tilfeldig valgte støyprosentverdiene kan være enten uniform eller normal (gaussisk). Hvis uniform fordeling blir valgt, vil inputfeltene for middelverdi og varians bli deaktivert. Hvis normal fordeling blir valgt, får brukeren mulighet til å sette opp middelverdien og variansen til støyfordelingen. Fasestøyområdet bestemmer det lovlige området for fasestøy.

Nye verdier lagres ved å trykke på ”Bruk”- knappen. Da lukkes dialogvinduet, og hoveddialogvinduet gjenopprettes med de nye verdiene. Det er derfor aldri mulig å taste inn verdier i hoveddialogboksen som er utenfor de områdene definert i systemvariablene. Ved å trykke på ”Tilbake” vil man lukke dialogboksen uten å lagre nye verdier.

Initialverdiene ved oppstart av applikasjonen er definert i funksjonen *OnInitDialog()* i klassen C*MKDopGenDlg* (*MKDopGenDlg.cpp*). For å endre disse initialverdiene må applikasjonen rekompileres. Altså, alle endringer som gjøres i systemvariabeldialogboksen vil kun være gjeldende ved kjøring av applikasjonen.

Alle inputfelt består av heltallsverdier, bortsett fra middelverdi og varians, som godtar flyttall. Hvis det tastes inn et flyttall i et felt som bare godtar heltall, vil en feilmelding komme idet ”Bruk”- knappen blir trykt, og det aktuelle inputfeltet vil bli markert.

4.2.3 Hjelpedialogboksen

Hjelpedialogboksen er ikke fullført. Hvis brukeren trykker på ”Hjelp” i hoveddialogboksen, vil en meldingsboks fortelle dette.

4.2.4 Manøvrering

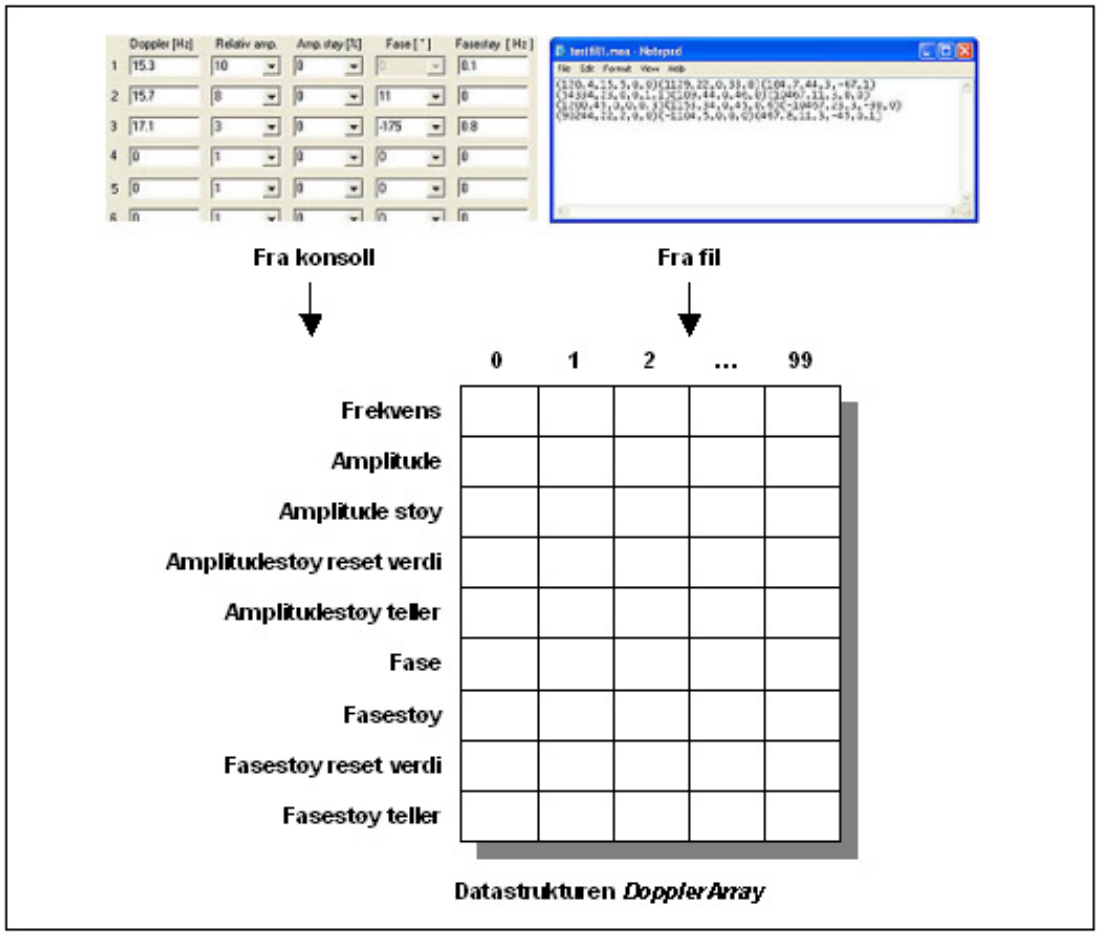
Manøvrering i brukergrensesnittet kan gjøres med musepekeren eller ved hjelp av tabulatortasten. Ved å trykke på tabulatorknappen forflyttes skrivemarkøren / merkingen mot venstre eller nedover innen de forskjellige seksjonene i dialogboksene. Ved å holde inne *SHIFT*-knappen og deretter trykker på tabulatorknappen forflyttes skrivemarkøren / merkingen mot høyre eller oppover. Trykk på *SPACE*- tasten eller *ENTER*- tasten når en knapp er merket, har samme funksjon som et museklikk på knappen. For å skifte mellom tilhørende radioknapper kan også piltastene brukes.

4.3 Programmeringsløsninger

Dette kapittelet gir en skjematisk og matematisk oversikt over hvordan summeringen av signalene med den eventuelt tillagte støy foregår. I tillegg gis en beskrivelse av hvordan programkoden er forsøkt optimalisert, for å få summeringen til å gå raskest mulig.

4.3.1 Summering av delsignaler

Signalene som summeres blir enten skrevet inn i hoveddialogboksen (konsollen) eller lest fra fil. (figur 4.8). Når brukeren trykker på ”Summer”- knappen, blir dopplerfrekvensen, amplituden, fasen og eventuell støy først lagret i det todimensjonale *array*et *DopplerArray*. *Amplitudestøy reset verdi* og *fasestøy reset verdi* blir regnet ut på grunnlag av hvor mange sampler pr periode som skal inneholde støy. Denne verdien kan endres i dialogboksen for systemvariabler. *Amplitudestøy teller* og *fasestøy teller* er tellere som for hver sample skal inkrementere og resettes når de når sin respektive *reset* verdi.

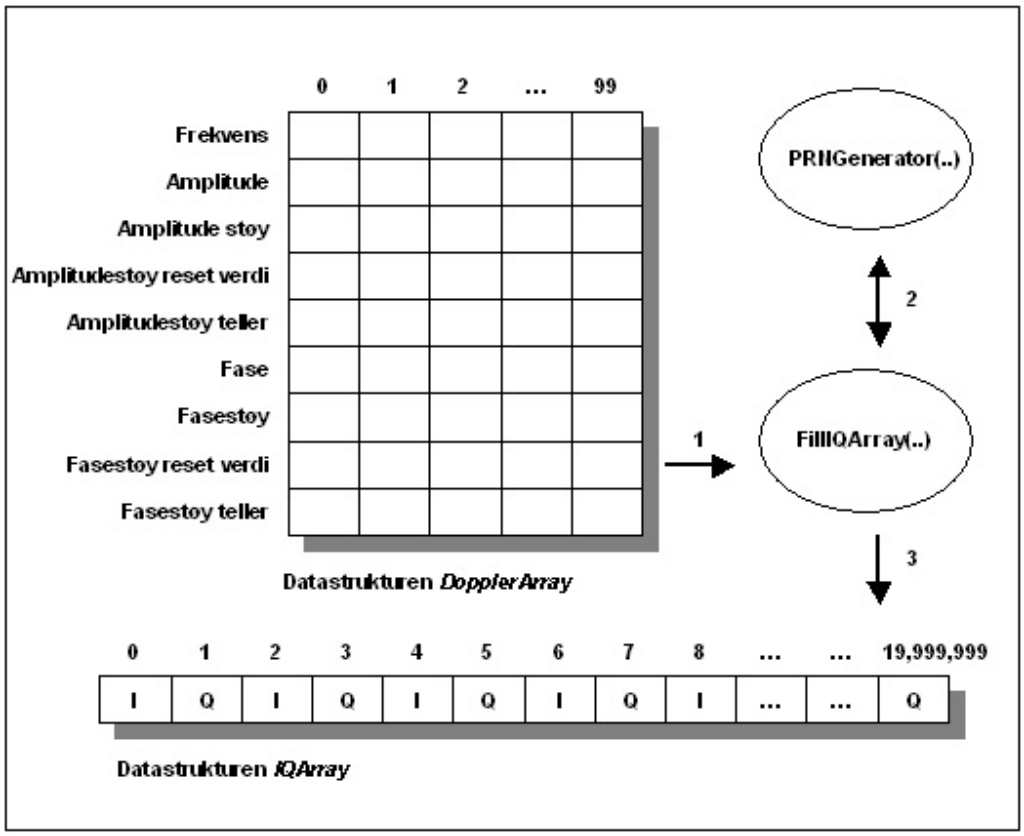


Figur 4.8 Data flyttes fra konsoll eller fil til et todimensjonalt array

Etter at alle verdier er satt i *DopplerArray*, sendes det til summeringsfunksjonen *FillIQArray(...)*. Summeringsfunksjonen vil så i en løkke iterere like mange ganger som det er samples i *IQArray* (figur 4.9). Dette består av 20,000,000 sampler på 16 bit, og I og Q signalene bruker annenhver sample. Størrelsen er bestemt ut fra kravet om et frekvensområde fra ± 0.1 Hz til ± 100 kHz med en oppløsning på 0.1 Hz. Når utlesingshastigheten på D/A kortet er 1,000,000 sampler pr sekund pr kanal kan *IQArray* derfor sees på som en tidsperiode på 10 sekunder, hvor sample 0 og 1 tilsvarer tiden t = 0 ms og sample 2 og 3 tilsvarer tiden t = 0.0001 ms (10,000 ms / 10,000,000 sampler), osv.

For hver iterasjon vil alle signalene dekomponeres i I og Q, summeres og lagres i *IQArray*. I-signalet blir lagret på plass i, Q-signalet på plass i + 1, hvor i går fra 0 til 19,999,998 . I tillegg vil amplitude- og fasestøy tellerne telle opp til de blir like henholdsvis *amplitudestøy reset verdi* og *fasestøy reset verdi*. Det er støyoppløsningen i systemvariabeldialogboksen som bestemmer *reset* verdiene. Når de blir like skal et støysignal (amplitude eller fase) legges til og da regnes dette ut ved hjelp av støygeneratorfunksjonen *PRNGenerator(...)*. Deretter resettes tellerne.

I applikasjonen setter brukeren opp den totale utgangseffekten til det komplekse signalet uavhengig av antall delsignaler. Hvert delsignal blir så tilpasset slik at summen av effektene til hvert delsignal blir lik den oppgitte totale utgangseffekten. Forholdet mellom delsignalene bestemmes av den relative amplituden, og forholdet er lineært. Hvis et komplekst signal for eksempel består av to delsignaler med relativ amplitude på henholdsvis 2 og 1 så er det første delsignalets voltamplitudeverdi dobbel så stor som den andre.



Figur 4.9 Summering av DopplerArray, resultat lagres i IQBuffer

Hvert delsignals andel, α_i, av den totale voltamplitudeverdien A, hvor N er antall delsignaler er beskrevet i uttrykk 4.1 :

$$\alpha_i = \frac{A_i}{\sum_{n=1}^{N} A_n} \tag{4.1}$$

Voltamplitudeverdien $A = \sqrt{(2)} \times A_{RMS}$, hvor $A_{RMS} = \sqrt{(R \times P)}$ og hvor R er resistans på 50Ω og P er effekten i Watt. Den totale utgangseffekten i applikasjonen, P', er oppgitt i dBm. Omgjort i Watt blir det som i uttrykk 4.2 :

$$P = 10^{\frac{P'}{10}-3} W \tag{4.2}$$

Uttrykk 4.3 viser da voltamplitudeverdien til ett signal:

$$A = \sqrt{2 \cdot R \cdot P}\ V = \sqrt{100 \cdot P}\ V = \sqrt{100 \cdot 10^{\frac{P'}{10}-3}}\ V = \sqrt{\frac{10^{\frac{P'}{10}}}{10}}\ V \tag{4.3}$$

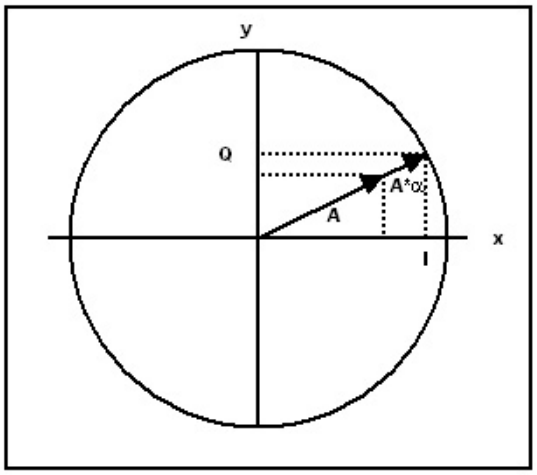
Dermed viser uttrykk 4.4 den absolutte andelen, A_{abs} , til hver av de N delsignalene med relativ amplitude A_i ,og med total utgangseffekt på P' :

$$A_{abs} = \sqrt{\frac{N \cdot 10^{\frac{P'}{10}}}{10}} \cdot \frac{A_i}{\sum_{n=1}^{N} A_n} \quad V \tag{4.4}$$

4.3.2 Støygenerering

Fasestøyen blir i applikasjonen oppgitt i Hz, og støyen består av et tilfeldig, uniformt fordelt faseskift mellom 0 og 2π, generert av *PRNGenerator(...)* ved tiden 1/*fasestøy*.

Amplitudestøyen, α, blir i applikasjonen oppgitt i prosent. Dette betyr at amplituden i et punkt, A, blir tillagt en tilfeldig amplitude mellom *–amplitudestøy* og *+amplitudestøy* prosent av den opprinnelige amplituden i det punktet. Amplitudestøyverdien blir fordelt på I og Q komponentene som vist på figur 4.10.



Figur 4.10 Beregning av støy til I og Q komponentene

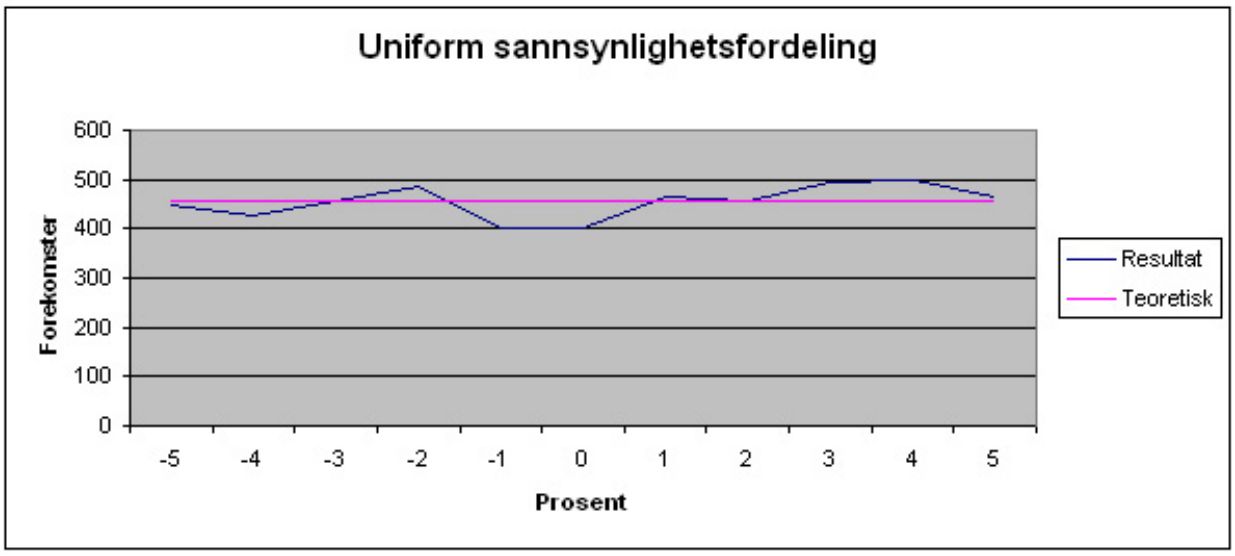
Brukeren kan i applikasjonen velge uniform amplitudestøyfordeling eller normal (gaussisk) amplitudestøyfordeling, og i sistnevnte fall sette opp middelverdi og varians. Ved normalfordelt amplitudestøyfordeling blir verdien blir regnet ut ved hjelp av *Box-Muller*- transformasjon (6). Det vil si at en uniformfordelt *random*- verdi går inn, og en normalfordelt *random*- verdi kommer ut.

I de tilfeller hvor den beregnede støyverdien α kommer utenfor *–amplitudestøy* og *+amplitudestøy*, vil α bli regnet ut på nytt med uniform fordeling. På denne måten vil støyverdiens normalfordelte kurve være lik den teoretiske normalfordelte kurven innenfor *–amplitudestøy* og *+amplitudestøy*.

Siden *IQArray* er 10 sekunder langt, vil de tilfeldige støyverdiene gjenta seg etter 10 sekunder.

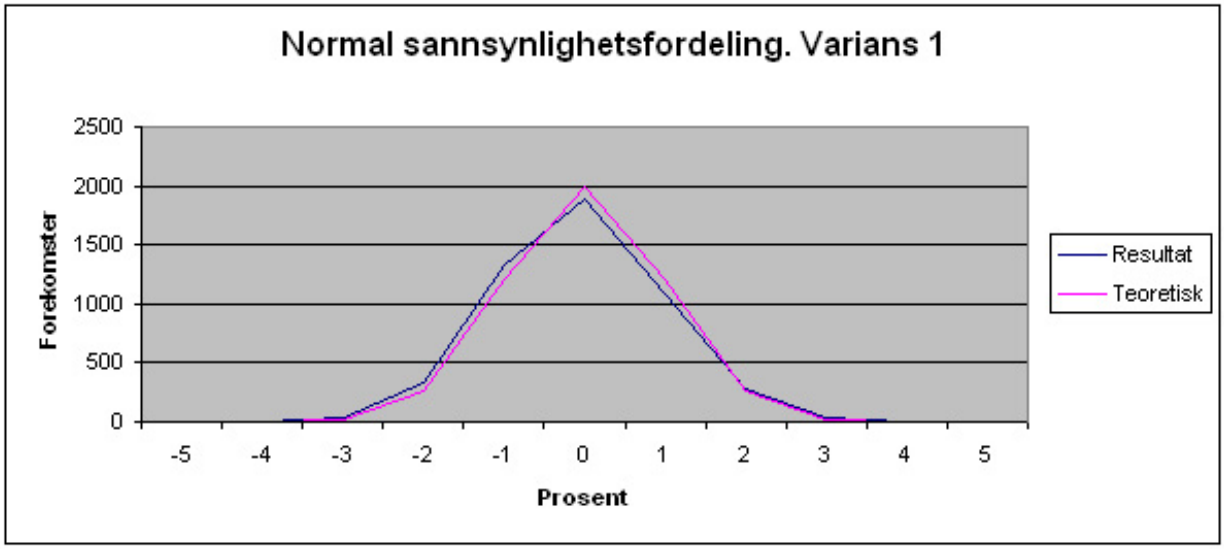
4.3.2.1 Sannsynlighetsfordeling

Et testprogram ble utviklet for å teste den reelle sannsynlighetsfordelingen opp mot den teoretiske ved generering av tilfeldige støyverdier. Testprogrammet genererte i alle forsøkene 5000 verdier. Figur 4.11 viser resultatet på støyverdier mellom –5 og 5 % med en uniform sannsynlighetsfordeling.

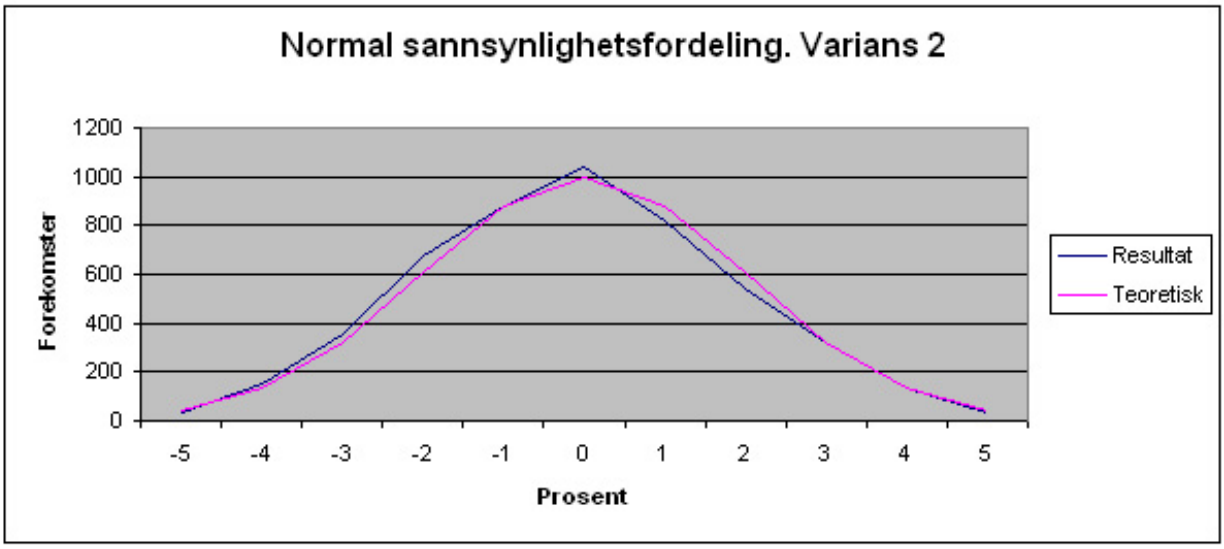


Figur 4.11 Uniform fordeling av støyverdier

Figur 4.12 og 4.13 viser resultatet på støyverdier mellom –5 og 5 % med en normal sannsynlighetsfordeling med varians på henholdsvis 1 og 2.

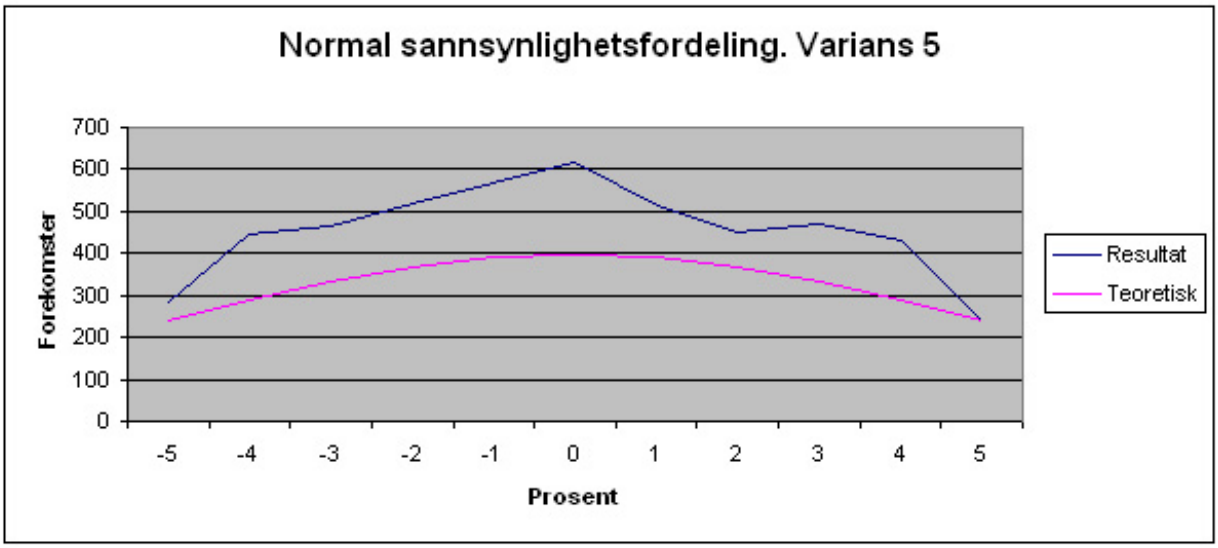


Figur 4.12 normal fordeling av støyverdier med varians 1



Figur 4.13 Normal fordeling av støyverdier med varians 2

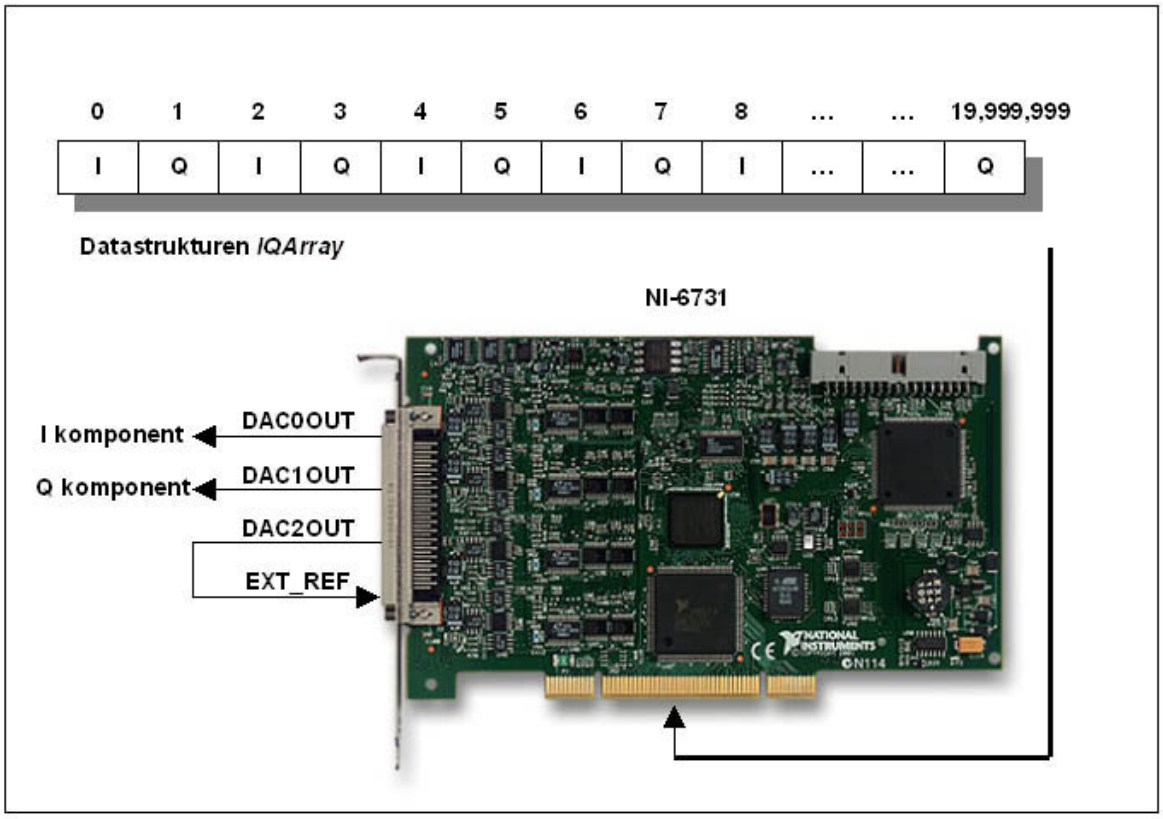
Figur 4.14 viser det samme som de to foregående men med en varians på 5. En så stor varians innebærer at sannsynligheten er rimelig stor for at verdier er mindre enn –5 og større enn 5 %. Her blir da verdier som faller utenfor ±5 % fordelt på nytt med uniform sannsynlighetsfordeling, og resultatet blir da at resultatkurven i 4.14 blir høyere enn den teoretiske fordelingen, men beholder den samme variansen.



Figur 4.14 Normal/uniform fordeling av støyverdier med varians 5

4.3.3 Analogt output

Når brukeren trykker på startknappen, vil *IQArray* bli lest med en hastighet på 1,000,000 samples pr sekund pr kanal ut på *DAC0OUT* og *DAC1OUT* på D/A kortet. Innholdet i *IQArray* blir demultiplexet på kanalene slik at I-komponenten av signalet skrives ut på *DAC0OUT* og Q-komponenten skrives ut på *DAC1OUT*. Lesingen gjentas inntil brukeren trykker på stoppknappen. Da vil 0 volt bli kjørt ut på begge kanalene.



Figur 4.15 Data fra IQArray kjøres ut på utgangene som analoge I og Q signaler

DAC0OUT og *DAC1OUT* er satt opp med ekstern referanse som brukeren selv kan styre. Den valgte maksimale utgangsspenningen som velges i hoveddialogboksen blir kjørt ut på kanalen *DAC2OUT* og inn på *EXT_REF* på kortet. (figur 4.15). *DAC2OUT* kjøres alltid med intern referanse som er 10 volt.

Signalene som kjøres ut vil ha dårligere oppløsning jo større frekvensen er. Eksempelvis vil et signal på 0.1 Hz ha 20,000,000 sampler pr periode, og et signal på 100 kHz vil ha kun 10 sampler pr periode.

4.3.4 Kodeoptimalisering

Den største flaskehalsen i applikasjonen er summeringen av signalene. Summeringsfunksjonen inneholder en løkke som itererer 20,000,000 ganger. På grunn av dette vil kun de ytterst nødvendige utregningene foregå i denne løkken. Det som må regnes ut er andelen av hvert delsignal, tillegging av eventuell støy, og dekomponeringen i I og Q signaler.

Andelen av hvert delsignal som er beskrevet i uttrykk 4.4 består av en konstant del og en del som varierer for hvert enkelt delsignal. I programkoden er derfor den konstante delen K (uttrykk 4.5) skilt ut fra uttrykk 4.4. og regnet ut på forhånd av funksjonen *ComputeConstant(...)* for å minimere utregningene i summeringsløkken.

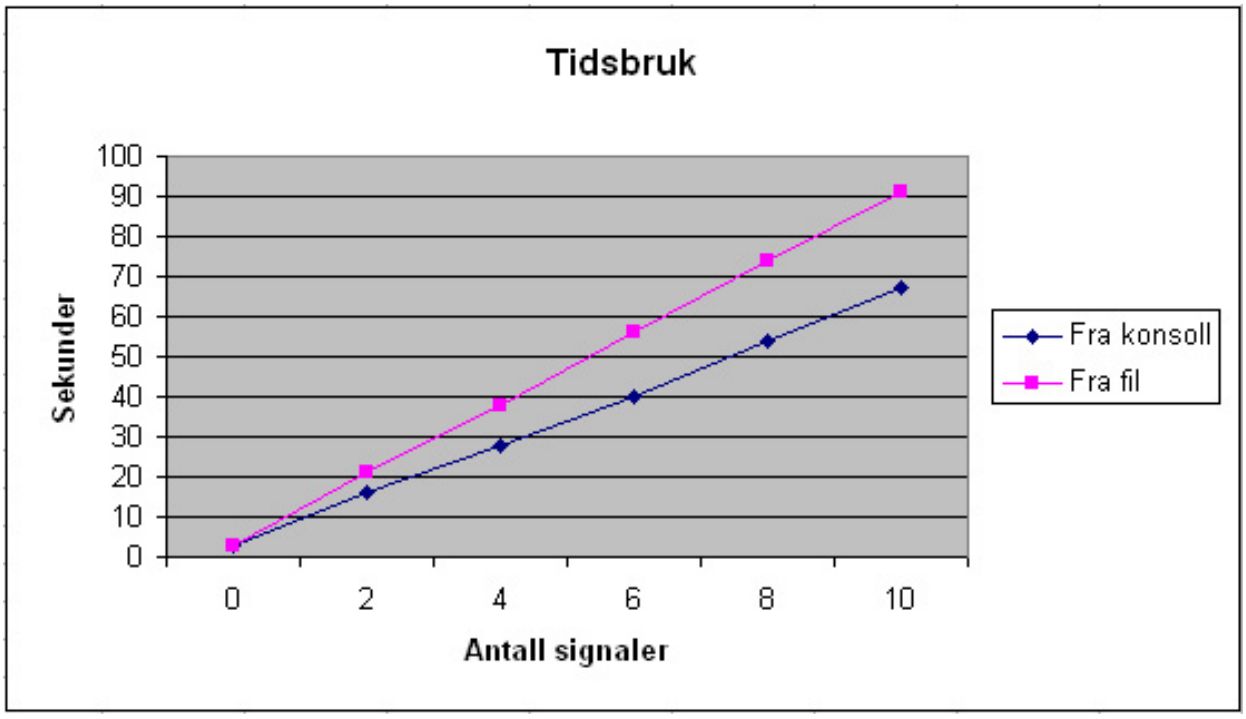
$$K = \frac{\sqrt{\frac{N \times 10^{\frac{P'}{10}}}{10}}}{\sum_{n=1}^{N} A_n} \qquad (4.5)$$

Genereringen av tilfeldige verdier som brukes til støygenerering gjøres lokalt. Dette går raskere enn hvis stadige kall skulle gjøres til standardbibliotekets *random*- funksjoner. I tillegg brukes en polar form av *Box-Muller*- transformasjonen ved generering av normalfordelte støyverdier, som gjør at man slipper kall til matematikkbiblioteket (6).

4.4 Ressursbruk

Den største minneforbrukeren i applikasjonen er datastrukturen *IQArray* som inneholder de summerte I og Q signalene. Med 20,000,000 sampler og 16 bit pr sample, vil *IQArray* konsumere 40 MB RAM. Applikasjonen for øvrig tar beslag på om lag 10 MB RAM, slik at den totale minnebruken blir ca. 50 MB.

Hastigheten på summeringen er vist i figur 4.16. Målingene er foretatt på en P-III 1000 MHz maskin med 256 MB RAM. Det er foretatt 6 målinger fra fil, og tilsvarende fem fra konsollen. Hvert delsignal er gitt både amplitudestøy, med støyoppløsning på 100 sampler pr periode, og fasestøy.



Figur 4.16 Tidsbruk ved summering av signaler

Disse målingene viser at tidsbruken, *t,* av summeringen som funksjon av antall signaler, *s,* fra konsoll $t_{konsoll}(s)$ og fra fil $t_{fil}(s)$ kan bli beskrevet i uttrykk 4.6 og 4.7

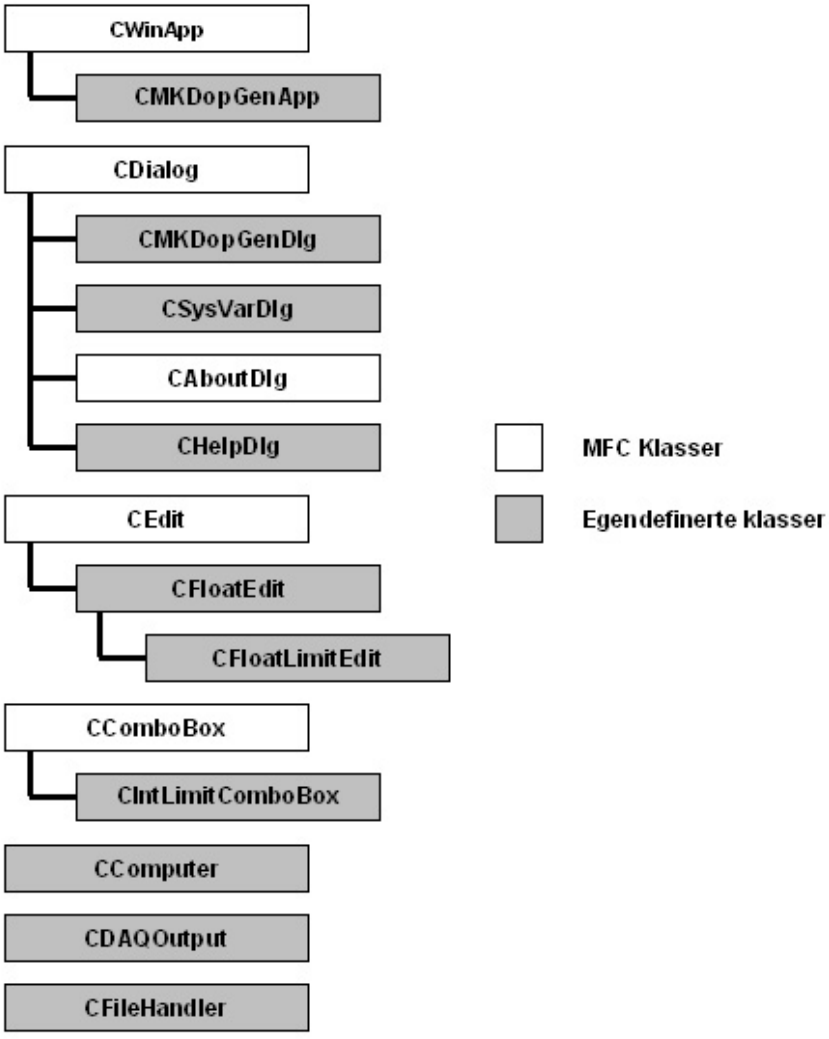
$$t_{konsoll}(s) = 6.4s + 3 \tag{4.6}$$

$$t_{fil}(s) = 8.8s + 3 \tag{4.7}$$

4.5 Klassestruktur

4.5.1 Klassediagram

Figur 4.17 viser dopplergeneratorens klassediagram. De hvite boksene representerer standard *Windows*-klasser, mens de grå boksene representerer klasser som er spesielt utviklet for denne applikasjonen.



Figur 4.17 Klassediagram

4.5.2 Klassebeskrivelse

Alle klassenes funksjon og virkemåte blir beskrevet i dette kapittelet. I tillegg brukes en rekke standard Windowklasser som for eksempel *CString*, *CButton*, *CComboBox* osv. De blir ikke beskrevet her. Appendiks C inneholder kildekoden med detaljerte kommentarer.

4.5.2.1 Klassen *CWinApp*

CWinApp er baseklassen for applikasjonen, og inneholder alle grunnleggende egenskaper, og sørger for standardoppførselen til en *Windows*- applikasjon. Blant annet tar den i mot og fordeler alle meldinger som blir sendt.

4.5.2.2 Klassen *CMKDopGenApp*

Dette er startklassen som oppretter en forekomst av hoveddialogboksen i applikasjonen. I initieringen av dette objektet opprettes en forekomst av klassen *CMKDopGenDlg*, som er hoveddialogboksen i denne applikasjonen. Når hoveddialogboksen avsluttes, avsluttes også applikasjonen.

4.5.2.3 Klassen *CDialog*

CDialog er morklassen for alle dialogvindu. Det finnes to typer dialogbokser, den ene kalles *modal* som betyr at dialogboksen må lukkes for at applikasjonen kan fortsette. Den andre er *modeless*, og den innebærer at flere slike dialogbokser kan være åpne samtidig. En *modal* dialogboks opprettes ved å bruke objektets *DoModal()* funksjon. En *modeless* dialogboks opprettes ved å bruke objektets *Create()* funksjon. *Multispektral Kvadratur Dopplergenerator* benytter seg kun av *modale* dialogbokser.

4.5.2.4 Klassen *CMKDopGenDlg*

Denne klassen har hovedstyringen på applikasjonen. Dette er hoveddialogvinduet hvor brukeren enten taster inn signalene som skal summeres eller henter disse fra fil, og summerer og starter genereringen. Det finnes i tillegg knapper for å avslutte applikasjonen, åpne systemvariabeldialogboksen, åpne hjelpdialogboksen og gjenopprette vinduet med de opprinnelige verdiene.

4.5.2.5 Klassen *CSysVarDlg*

Systemvariablene settes i *CSysVar* klassen. Systemvariabler er grenseverdier for blant annet dopplerfrekvens, amplitude, fase, støy og utgangseffekt. Variablene har en initialverdi som er satt i programkoden, men kan under kjøring av programmet endres. Verdien på variablene i denne klassen gir begrensingene for hva som kan tastes inn i hoveddialogboksen. Brukeren kan derfor ikke i noen tilfeller overstige verdiene som er satt i *CSysVar* dialogboksen.

4.5.2.6 Klassen *CAboutDlg*

En *About*- boks er standard for alle *Windows*-programmer, og kan velges fra *Windows*-menyen som kommer opp når man klikker på ikonet i øverste venstre hjørne eller høyreklikker på tittellinjen i applikasjonen.

4.5.2.7 Klassen *CHelpDlg*

Hjelpdialogboksen skal inneholde hjelpetekst og brukerveiledning for applikasjonen, men klassen er ikke ferdig utviklet.

4.5.2.8 Klassen *CEdit*

CEdit er baseklasse og gir standardoppførsel for alle *edit*- kontrollere. En *edit*- kontroll gir brukeren mulighet til å skrive inn tekst.

4.5.2.9 Klassen *CFloatEdit*

Dette er en spesialisert klasse som arver all funksjonalitet fra morklassen *CEdit*, men hvor brukeren bare tillattes å taste inn heltall og flyttall. Disse *edit*- kontrollene brukes i systemvariabeldialogboksen.

4.5.2.10 Klassen *CFloatLimitEdit*

Dette er nok en spesialisert klasse, som arver funksjonaliteten fra morklassen *CFloatEdit*, men som i tillegg bare tillatter tallverdier innen et visst område. Det lovlige området blir satt av grenseverdiene i *CMKDopGenDlg* klassen. Disse *edit*- kontrollene blir brukt i hoveddialogboksen.

4.5.2.11 Klassen *CComboBox*

CComboBox er baseklasse og gir standardoppførsel for alle *combo*- bokser. En *combo*- boks er en kombinert listeboks og *edit*- kontroll. Den gir brukeren mulighet til både å velge verdier fra en liste, og å skrive inn tekst.

4.5.2.12 Klassen *CIntLimitComboBox*

Dette er en spesialisert klasse, som arver funksjonaliteten fra morklassen *CComboBox*, men hvor det bare tillates å taste in heltall innen et visst område. Det lovlige området blir satt av grenseverdiene i *CMKDopGenDlg* klassen. Disse *combo*- boksene blir brukt i hoveddialogboksen.

4.5.2.13 Klassen *CComputer*

Denne klassen samler en rekke funksjoner for matematiske utregninger. Her er funksjonene for summeringen av signalene og generering av støy.

4.5.2.14 Klassen *CDACOutput*

CDAQOutput samler alle funksjoner som har med I/O til D/A-kortet å gjøre.

4.5.2.15 Klassen *CFileHandler*

Klassen behandler alt som har med fil I/O å gjøre.

4.6 Resultat i forhold til opprinnelig oppgavespesifikasjon

Her sammenliknes resultatet av oppgaven med de opprinnelige oppgavespesifikasjonene:

- Utgangsnivå / oppløsning:
  Spesifikasjon: +10 dBm til –10 dBm / 1dB (gjerne valgbart dBm/V)
  Resultat: Valgbart i systemvariableldialogboksen / 1dB (ikke valgbart dBm/V)

- Faseforskyvning I og Q-utgangene:
  Spesifikasjon: +90° for positive dopplerfrekvenser, -90° for negative dopplerfrekvenser
  Resultat: +90° for positive dopplerfrekvenser, -90° for negative dopplerfrekvenser

- Frekvensområde / oppløsning:
  Spesifikasjon: Minimum ±0.1 Hz til ±100.0 kHz / 0.1Hz.
  Ønskelig ±0.01 Hz til ±500.0 kHz / 0.01 Hz.
  Eventuelt dele området i to : 0.1 Hz til 1 kHz / 0.1 Hz
  10 Hz til 100 kHz / 10 Hz
  Resultat: ±0.1 Hz til ±100.0 kHz / 0.1Hz

- Antall samtidige frekvenser:
  Spesifikasjon: Mer enn 5, helst 20, gjerne 100
  Resultat: 10 fra konsoll, 100 fra fil.

- Modulasjonsområde / oppløsning:
  Spesifikasjon: Amplitude 0 dB til –20 dB / 1 dB
  Fase ±180° / 1°
  Resultat: Amplitude valgbart / relativ amplitude (lineær voltamplitude)

- Støy:
  Spesifikasjon: Det bør være mulighet til å legge på fase - og amplitudestøy
  Resultat: Det er mulighet for å legge på fase – og amplitudestøy
  Amplitudestøy er valgbart i prosent, fasestøy er valgbart i Hz

- Brukergrensesnitt:
  Spesifikasjon: Mulighet for oppsett av minst 5 frekvenser med modulasjon
  Resultat: Oppsett for opptil 10 frekvenser med modulasjon

- Import av et stort antall frekvenser fra fil med ASCII-format:
  Spesifikasjon: Filformat 1 : {(t1),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)
  (t2),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)
  …
  (tn),(f1,a1,p1),(f2,a2,p2)…(fn,an,pn)}
  Hvor t = tid[ms], f = frekvens [Hz], a = amplitude [dB] og p = fase [°]

  Filformat 2 : {(I1,Q1),(I2,Q2),…(In,Qm)}
  Hvor I og Q er rådata på kompleks form

  Resultat: Filformat : {(f1,a1,an1,p1,pn1)(f2,a2,an2,p2,pn2)…(fn,an,ann,pn,pnn)}
  Hvor f = frekvens [Hz], a = relativ amplitude, an = amplitudestøy [%],
  p = fase [°] og pn = fasestøy [Hz]

- Synkronisering med omverdenen:
  Spesifikasjon: Synk-inngang: Sørger for at tiden settes til t = 0.
  Synk-utgang: Signal som forteller at tiden t = 0.
  Resultat: Ikke utført

5 DRØFTING

De opprinnelige spesifikasjonene har ikke vært absolutte. Veileder og potensielle brukere av dopplergeneratoren har i løpet av utviklingsperioden innsett at noen endringer på spesifikasjonene har vært fordelaktig. Disse endringene har blitt imøtekommet.

De fleste spesifikasjonene har dermed blitt oppfylt. De mest kritiske spesifikasjonene som ikke er realisert er *filformat 2* og *filformat 1* der signalene kunne endre etter visse tidsperioder. I tillegg er det ikke gjennomført noen synkronisering med omverdenen. Bakgrunnen for dette er at summeringen er forholdsvis tidkrevende, og at maskinens ressurser dermed ikke strekker til. Dopplergeneratoren må kunne summere signaler som spenner fra 0.1 Hz til 100 kHz med en oppløsning på 0.1 Hz. Datastrukturen *IQArray* er minste felles multiplum for signalene, og krever stor plass.

Det finnes flere mulige løsninger på dette problemet. En måte er å dele frekvensområdene inn i flere områder. Ved lave frekvenser kreves det forholdsvis høy oppløsning, for høyere frekvenser vil det være tilstrekkelig med en mye lavere oppløsning. En annen, og mer komplisert løsning, kan være å dynamisk beregne *IQArray* på grunnlag av de signalene som skal summeres.

Det finnes ulike måter å forbedre og videreutvikle dopplergeneratoren på. Dette har også vært diskutert som en mulig hovedoppgave for meg i siste året på *BSc(Hons) Computer Science* ved *Heriot-Watt University*. Noen punkter som har kommet opp under utviklingsfasen er :

- Hensiktsmessig og optimalisert oppdeling av frekvensområdet i forhold til bruksområde.
- Jevne ut signaler med høye frekvenser og lav oppløsning ved hjelp av filter.
- Summering og generering av ulike signaler i korte tidsintervaller. Gjerne forhåndslagring av ferdigsummerte dopplersignaler på fil.
- Synkronisering med omverdenen, starte generering ved ekstern *trigging*.
- Grafisk fremstilling av det komplekse dopplersignalet etter summering, slik at endring og justering kan gjøres raskt.

6 KONKLUSJON

Multispektral Kvadratur Dopplergenerator er en applikasjon som fungerer etter spesifikasjonene som har blitt gitt, og etter endringer av disse i henhold til ønske fra oppdragsgiver. Unntakene er synkronisering med omverdenen, og endring av det komplekse summerte signalet i gitte tidsintervaller.

Litteratur

(1) George W. Stimson (1998): Introduction to Airborne Radar Second Edition, SciTech Publishing Inc., USA, 189, 446-447.

(2) National Instruments (2001): DAQ NI 671X/673X User Manual, December 2001 Edition, USA

(3) Windowsprogrammering med MFC : http://olga.hials.no/mfc/.

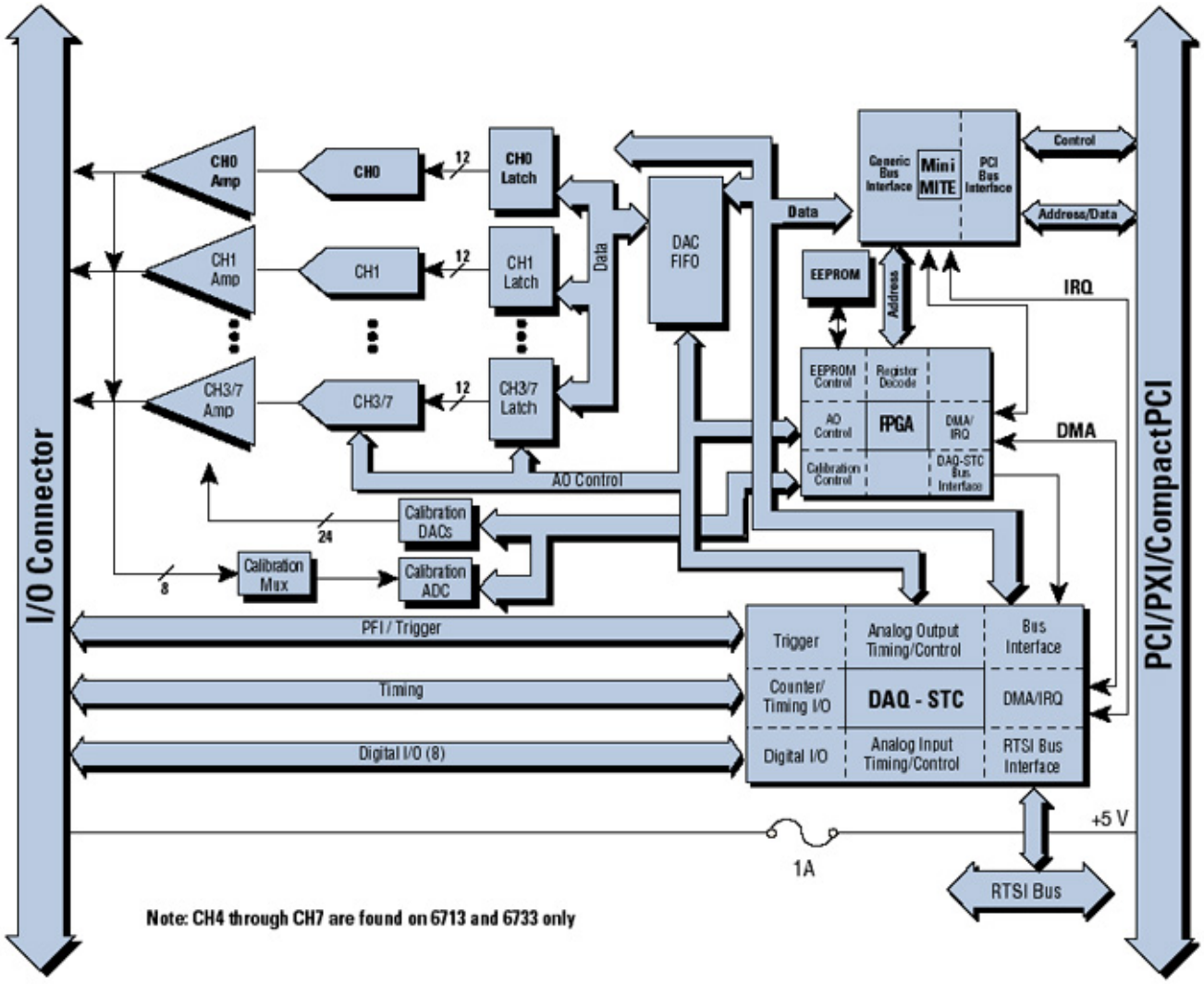
(4) The Code Project : http://www.codeproject.com.

(5) The C++ Resources Network : http://www.cplusplus.com.

(6) Generating Gaussian Random Numbers : http://www.taygeta.com/random/gaussian.html.

APPENDIKS

A NI-6731 HARDWARE BLOCK DIAGRAM



Figur A.1 Hardware Block Diagram

B NI-6731 PIN CONNECTOR DIAGRAM

AOGND	34	68	NC
NC	33	67	AOGND
AOGND	32	66	AOGND
AOGND	31	65	DAC7OUT[1]
DAC6OUT[1]	30	64	AOGND
AOGND	29	63	AOGND
DAC5OUT[1]	28	62	NC
AOGND	27	61	AOGND
AOGND	26	60	DAC4OUT[1]
DAC3OUT	25	59	AOGND
AOGND	24	58	AOGND
AOGND	23	57	DAC2OUT
DAC0OUT	22	56	AOGND
DAC1OUT	21	55	AOGND
EXTREF	20	54	AOGND
DIO4	19	53	DGND
DGND	18	52	DIO0
DIO1	17	51	DIO5
DIO6	16	50	DGND
DGND	15	49	DIO2
+5 V	14	48	DIO7
DGND	13	47	DIO3
DGND	12	46	NC
PFI0	11	45	EXTSTROBE
PFI1	10	44	DGND
DGND	9	43	PFI2
+5 V	8	42	PFI3/GPCTR1_SOURCE
DGND	7	41	PFI4/GPCTR1_GATE
PFI5/UPDATE	6	40	GPCTR1_OUT
PFI6/WFTRIG	5	39	DGND
DGND	4	38	PFI7
PFI9/GPCTR0_GATE	3	37	PFI8/GPCTR0_SOURCE
GPCTR0_OUT	2	36	DGND
FREQ_OUT	1	35	DGND

[1] No Connect on 6711 or 6731

Figur B.1 Pin Connector Diagram

C KILDEKODE FOR PROGRAMSPESIFIKKE KLASSER

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            -                                                   **/
/** Filename:         MKDopGen.h                                          **/
/** Filetype:         Declaration file for the MKDOPGEN application       **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/

#if !defined(AFX_MKDOPGEN_H__73E5F603_4418_49DD_AC06_56F86321318F__INCLUDED_)
#define AFX_MKDOPGEN_H__73E5F603_4418_49DD_AC06_56F86321318F__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
  #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

////////////////////////////////////////////////////////////////////////////
// CMKDopGenApp:
// See MKDopGen.cpp for the implementation of this class
//

class CMKDopGenApp : public CWinApp
{
public:
  CMKDopGenApp();

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CMKDopGenApp)
  public:
  virtual BOOL InitInstance();
  //}}AFX_VIRTUAL

// Implementation

  //{{AFX_MSG(CMKDopGenApp)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};


////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif // !defined(AFX_MKDOPGEN_H__73E5F603_4418_49DD_AC06_56F86321318F__INCLUDED_)
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            -                                                   **/
/** Filename:         MKDopGen.h                                          **/
/** Filetype:         Implementation file for the MKDOPGEN application    **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/

#include "stdafx.h"
#include "MKDopGen.h"
#include "MKDopGenDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMKDopGenApp

BEGIN_MESSAGE_MAP(CMKDopGenApp, CWinApp)
  //{{AFX_MSG_MAP(CMKDopGenApp)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
  //}}AFX_MSG
  ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMKDopGenApp construction

CMKDopGenApp::CMKDopGenApp()
{
  // TODO: add construction code here,
  // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CMKDopGenApp object

CMKDopGenApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CMKDopGenApp initialization

BOOL CMKDopGenApp::InitInstance()
{
  AfxEnableControlContainer();

  // Standard initialization
  // If you are not using these features and wish to reduce the size
  //  of your final executable, you should remove from the following
  //  the specific initialization routines you do not need.

#ifdef _AFXDLL
  Enable3dControls();              // Call this when using MFC in a shared DLL
#else
  Enable3dControlsStatic();  // Call this when linking to MFC statically
#endif

  CMKDopGenDlg dlg;
  m_pMainWnd = &dlg;
  int nResponse = dlg.DoModal();



  // Since the dialog has been closed, return FALSE so that we exit the
  //  application, rather than start the application's message pump.
  return FALSE;
}
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator               **/
/** Class:           CMKDopGenDlg                                             **/
/** Filename:        MKDopGenDlg.h                                            **/
/** Filetype:        Declaration file                                         **/
/** Author/year:     Alexander Iversen / 2002                                 **/
/*****************************************************************************/

#include "FloatLimitEdit.h"
#include "IntLimitComboBox.h"
#include "afxtempl.h"
#include "SysVarDlg.h"
#include "HelpDlg.h"
#include "DACOutput.h"
#include "Computer.h"
#include "FileHandler.h"

#if !defined(AFX_MKDOPGENDLG_H__2EDDF221_6774_4D49_83A7_72E7A86B8815__INCLUDED_)
#define AFX_MKDOPGENDLG_H__2EDDF221_6774_4D49_83A7_72E7A86B8815__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


/////////////////////////////////////////////////////////////////////////////
// CMKDopGenDlg dialog

class CMKDopGenDlg : public CDialog
{
// Construction
public:
  CMKDopGenDlg(CWnd* pParent = NULL);     // standard constructor

// Dialog Data
  //{{AFX_DATA(CMKDopGenDlg)
  enum { IDD = IDD_MKDOPGEN_DIALOG };
  CFloatLimitEdit     m_edit9PN;
  CFloatLimitEdit     m_edit8PN;
  CFloatLimitEdit     m_edit7PN;
  CFloatLimitEdit     m_edit6PN;
  CFloatLimitEdit     m_edit5PN;
  CFloatLimitEdit     m_edit4PN;
  CFloatLimitEdit     m_edit3PN;
  CFloatLimitEdit     m_edit2PN;
  CFloatLimitEdit     m_edit1PN;
  CFloatLimitEdit     m_edit10PN;
  CFloatLimitEdit     m_editVOut;
  CButton       m_butSum;
  CIntLimitComboBox   m_combodBmOut;
  CButton       m_butHelp;
  CButton       m_butSysVar;
  CButton       m_butStart;
  CButton       m_butGet;
  CButton       m_butExit;
  CButton       m_butClear;
  CEdit  m_editStatus;
  CButton       m_radioFile;
  CButton m_radioMan;
  CEdit  m_editFile;
  CIntLimitComboBox   m_combo9P;
  CIntLimitComboBox   m_combo9AN;
  CIntLimitComboBox   m_combo8P;
  CIntLimitComboBox   m_combo8AN;
  CIntLimitComboBox   m_combo7P;
  CIntLimitComboBox   m_combo7AN;
  CIntLimitComboBox   m_combo6P;
  CIntLimitComboBox   m_combo6AN;
  CIntLimitComboBox   m_combo5P;
  CIntLimitComboBox   m_combo5AN;
  CIntLimitComboBox   m_combo4P;
  CIntLimitComboBox   m_combo4AN;
  CIntLimitComboBox   m_combo3P;
  CIntLimitComboBox   m_combo3AN;
  CIntLimitComboBox   m_combo2P;
  CIntLimitComboBox   m_combo2AN;
  CIntLimitComboBox   m_combo1P;
  CIntLimitComboBox   m_combo1AN;
  CIntLimitComboBox   m_combo10P;
  CIntLimitComboBox   m_combo10AN;
  CIntLimitComboBox   m_combo9A;
  CIntLimitComboBox   m_combo8A;
  CIntLimitComboBox   m_combo7A;
```

```
CIntLimitComboBox   m_combo6A;
CIntLimitComboBox   m_combo5A;
CIntLimitComboBox   m_combo4A;
CIntLimitComboBox   m_combo3A;
CIntLimitComboBox   m_combo2A;
CIntLimitComboBox   m_combo1A;
CIntLimitComboBox   m_combo10A;
CFloatLimitEdit     m_edit9F;
CFloatLimitEdit     m_edit8F;
CFloatLimitEdit     m_edit7F;
CFloatLimitEdit     m_edit6F;
CFloatLimitEdit     m_edit5F;
CFloatLimitEdit     m_edit4F;
CFloatLimitEdit     m_edit3F;
CFloatLimitEdit     m_edit2F;
CFloatLimitEdit     m_edit10F;
CFloatLimitEdit     m_edit1F;
CString      m_str10A;
CString      m_str10AN;
CString      m_str10P;
CString      m_str1A;
CString      m_str1AN;
CString      m_str1P;
CString      m_str2A;
CString      m_str2AN;
CString      m_str2P;
CString      m_str3A;
CString      m_str3AN;
CString      m_str3P;
CString      m_str4A;
CString      m_str4AN;
CString      m_str4P;
CString      m_str5A;
CString      m_str5AN;
CString      m_str5P;
CString      m_str6A;
CString      m_str6AN;
CString      m_str6P;
CString      m_str7A;
CString      m_str7AN;
CString      m_str7P;
CString      m_str8A;
CString      m_str8AN;
CString      m_str8P;
CString      m_str9A;
CString      m_str9AN;
CString      m_str9P;
CString      m_strFile;
CString      m_strdBmOut;
float  m_flt10F;
float  m_flt1F;
float  m_flt3F;
float  m_flt4F;
float  m_flt5F;
float  m_flt6F;
float  m_flt7F;
float  m_flt8F;
float  m_flt9F;
float  m_flt2F;
float  m_fltOutV;
float  m_flt10PN;
float  m_flt1PN;
float  m_flt2PN;
float  m_flt3PN;
float  m_flt4PN;
float  m_flt5PN;
float  m_flt6PN;
float  m_flt7PN;
float  m_flt8PN;
float  m_flt9PN;
//}}AFX_DATA

int   m_intRangeODBM;      // total effect out
int   m_intRangeOV;        // extern reference voltage
int   m_intRangeP;         // phase range
int   m_intRangePN;        // phase noise range
int   m_intRangeA;         // amplitude range
int   m_intRangeAN;        // amplitude noise range
float m_fltRangeANM;       // amplitude noise mean
float m_fltRangeANV;       // amplitude noise variance
int   m_intRangeF;         // frequency range
int   m_intNoiseRes;       // number of noise samples pr period
float m_fltA;              // phase noise constant
```

```
  CString m_strViewVeff;     // volt effect equivalent
  CString m_strViewVamp;     // volt amplitude equivalent
  CString m_strPathname;     // modulation-file path name
  CString m_strFContents; // modulation-file contents
  CString m_strStatus;// status field contents
  bool m_blTgl;              // toggling start/stop button
  bool m_blGaussA;           // amplitude noise normal/gauss

     short * m_psrtIQArray;  // pointer to array containing 16 bit values
  float (* m_pfltDopplerArray)[100];//pointer to 2dim array containing signal info
  CDACOutput * m_pDac;// pointer to CDACOutput object
  CFileHandler * m_pFile;    // pointer to CFileHandler object

  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CMKDopGenDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);     // DDX/DDV support
  //}}AFX_VIRTUAL

// Implementation
protected:
  HICON m_hIcon;

  // Generated message map functions
  //{{AFX_MSG(CMKDopGenDlg)
  virtual BOOL OnInitDialog();
  afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
  afx_msg void OnPaint();
  afx_msg HCURSOR OnQueryDragIcon();
  afx_msg void OnExit();
  afx_msg void OnClear();
  afx_msg void OnSysvar();
  afx_msg void OnGet();
  afx_msg void OnStart();
  afx_msg void OnHelp();
  afx_msg void OnSum();
  afx_msg void OnSelchangeCombo10a();
  afx_msg void OnSelchangeCombo10an();
  afx_msg void OnSelchangeCombo10p();
  afx_msg void OnSelchangeCombo1a();
  afx_msg void OnSelchangeCombo1an();
  afx_msg void OnSelchangeCombo1p();
  afx_msg void OnSelchangeCombo2a();
  afx_msg void OnSelchangeCombo2an();
  afx_msg void OnSelchangeCombo2p();
  afx_msg void OnSelchangeCombo3a();
  afx_msg void OnSelchangeCombo3an();
  afx_msg void OnSelchangeCombo3p();
  afx_msg void OnSelchangeCombo4a();
  afx_msg void OnSelchangeCombo4an();
  afx_msg void OnSelchangeCombo4p();
  afx_msg void OnSelchangeCombo5a();
  afx_msg void OnSelchangeCombo5an();
  afx_msg void OnSelchangeCombo5p();
  afx_msg void OnSelchangeCombo6a();
  afx_msg void OnSelchangeCombo6an();
  afx_msg void OnSelchangeCombo6p();
  afx_msg void OnSelchangeCombo7a();
  afx_msg void OnSelchangeCombo7an();
  afx_msg void OnSelchangeCombo7p();
  afx_msg void OnSelchangeCombo8a();
  afx_msg void OnSelchangeCombo8an();
  afx_msg void OnSelchangeCombo8p();
  afx_msg void OnSelchangeCombo9a();
  afx_msg void OnSelchangeCombo9an();
  afx_msg void OnSelchangeCombo9p();
  afx_msg void OnSelchangeComboOutDbm();
  afx_msg void OnUpdateEdit10f();
  afx_msg void OnUpdateEdit1f();
  afx_msg void OnUpdateEdit2f();
  afx_msg void OnUpdateEdit3f();
  afx_msg void OnUpdateEdit4f();
  afx_msg void OnUpdateEdit5f();
  afx_msg void OnUpdateEdit6f();
  afx_msg void OnUpdateEdit7f();
  afx_msg void OnUpdateEdit8f();
  afx_msg void OnUpdateEdit9f();
  afx_msg void OnUpdateEditFile();
  afx_msg void OnClose();
  afx_msg void OnUpdateEditOutV();
  afx_msg void OnUpdateEdit10pn();
  afx_msg void OnUpdateEdit1pn();
  afx_msg void OnUpdateEdit2pn();
```

```
  afx_msg void OnUpdateEdit3pn();
  afx_msg void OnUpdateEdit4pn();
  afx_msg void OnUpdateEdit5pn();
  afx_msg void OnUpdateEdit6pn();
  afx_msg void OnUpdateEdit7pn();
  afx_msg void OnUpdateEdit8pn();
  afx_msg void OnUpdateEdit9pn();
  afx_msg void OnEditupdateComboOutDbm();
  afx_msg void OnEditupdateCombo10a();
  afx_msg void OnEditupdateCombo10an();
  afx_msg void OnEditupdateCombo10p();
  afx_msg void OnEditupdateCombo1a();
  afx_msg void OnEditupdateCombo1an();
  afx_msg void OnEditupdateCombo1p();
  afx_msg void OnEditupdateCombo2a();
  afx_msg void OnEditupdateCombo2an();
  afx_msg void OnEditupdateCombo2p();
  afx_msg void OnEditupdateCombo3a();
  afx_msg void OnEditupdateCombo3an();
  afx_msg void OnEditupdateCombo3p();
  afx_msg void OnEditupdateCombo4a();
  afx_msg void OnEditupdateCombo4an();
  afx_msg void OnEditupdateCombo4p();
  afx_msg void OnEditupdateCombo5a();
  afx_msg void OnEditupdateCombo5an();
  afx_msg void OnEditupdateCombo5p();
  afx_msg void OnEditupdateCombo6a();
  afx_msg void OnEditupdateCombo6an();
  afx_msg void OnEditupdateCombo6p();
  afx_msg void OnEditupdateCombo7a();
  afx_msg void OnEditupdateCombo7an();
  afx_msg void OnEditupdateCombo7p();
  afx_msg void OnEditupdateCombo8a();
  afx_msg void OnEditupdateCombo8an();
  afx_msg void OnEditupdateCombo8p();
  afx_msg void OnEditupdateCombo9a();
  afx_msg void OnEditupdateCombo9an();
  afx_msg void OnEditupdateCombo9p();
  afx_msg void OnRadioFile();
  afx_msg void OnRadioMan();
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()

public: // Operations
  // Reset values in all fields
  void ResetValues();
  // Fill combobox with values ranging from min to max
  void PopulateComboBox(CIntLimitComboBox* pComboBox,
                                    const int& intMin, int& intMax);
  // Update status window
  void UpdateStatusWindow(CString &strMax, int &intClipping);
  // Update status on input (control) fields (generator running/stopped)
  void UpdateControlStatus(bool& blEnableControls);
  // Update status on buttons (generator running/stopped)
  void UpdateButtons(const short &srtRadioButton);
  // Obtain numerical values from combo boxes
  float GetFAP(const short &srtType, short& intI);
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif // !defined(AFX_MKDOPGENDLG_H__2EDDF221_6774_4D49_83A7_72E7A86B8815__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator               **/
/** Class:           CMKDopGenDlg                                             **/
/** Filename:        MKDopGenDlg.cpp                                          **/
/** Filetype:        Implementation file                                      **/
/** Author/year:     Alexander Iversen / 2002                                 **/
/*****************************************************************************/

#include "stdafx.h"
#include "MKDopGen.h"
#include "MKDopGenDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
  CAboutDlg();

// Dialog Data
  //{{AFX_DATA(CAboutDlg)
  enum { IDD = IDD_ABOUTBOX };
  //}}AFX_DATA

  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CAboutDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
  //}}AFX_VIRTUAL

// Implementation
protected:
  //{{AFX_MSG(CAboutDlg)
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
  //{{AFX_DATA_INIT(CAboutDlg)
  //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CAboutDlg)
  //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
  //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMKDopGenDlg dialog

CMKDopGenDlg::CMKDopGenDlg(CWnd* pParent /*=NULL*/)
  : CDialog(CMKDopGenDlg::IDD, pParent)
{
  //{{AFX_DATA_INIT(CMKDopGenDlg)
  m_str10A = _T("");
  m_str10AN = _T("");
  m_str10P = _T("");
  m_str1P = _T("");
  m_str2A = _T("");
  m_str2AN = _T("");
  m_str2P = _T("");
  m_str3A = _T("");
  m_str3AN = _T("");
  m_str3P = _T("");
  m_str4P = _T("");
  m_str5A = _T("");
  m_str5AN = _T("");
  m_str5P = _T("");
```

```
  m_str6A = _T("");
  m_str6AN = _T("");
  m_str6P = _T("");
  m_str7P = _T("");
  m_str8A = _T("");
  m_str8AN = _T("");
  m_str8P = _T("");
  m_str9A = _T("");
  m_str9AN = _T("");
  m_str9P = _T("");
  m_strFile = _T("");
  m_strdBmOut = _T("");
  m_flt10F = 0.0f;
  m_flt1F = 0.0f;
  m_flt3F = 0.0f;
  m_flt4F = 0.0f;
  m_flt5F = 0.0f;
  m_flt6F = 0.0f;
  m_flt7F = 0.0f;
  m_flt8F = 0.0f;
  m_flt9F = 0.0f;
  m_flt2F = 0.0f;
  m_fltOutV = 0.0f;
  m_flt10PN = 0.0f;
  m_flt1PN = 0.0f;
  m_flt2PN = 0.0f;
  m_flt3PN = 0.0f;
  m_flt4PN = 0.0f;
  m_flt5PN = 0.0f;
  m_flt6PN = 0.0f;
  m_flt7PN = 0.0f;
  m_flt8PN = 0.0f;
  m_flt9PN = 0.0f;
  //}}AFX_DATA_INIT
  // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
  m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMKDopGenDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CMKDopGenDlg)
  DDX_Control(pDX, IDC_EDIT_9PN, m_edit9PN);
  DDX_Control(pDX, IDC_EDIT_8PN, m_edit8PN);
  DDX_Control(pDX, IDC_EDIT_7PN, m_edit7PN);
  DDX_Control(pDX, IDC_EDIT_6PN, m_edit6PN);
  DDX_Control(pDX, IDC_EDIT_5PN, m_edit5PN);
  DDX_Control(pDX, IDC_EDIT_4PN, m_edit4PN);
  DDX_Control(pDX, IDC_EDIT_3PN, m_edit3PN);
  DDX_Control(pDX, IDC_EDIT_2PN, m_edit2PN);
  DDX_Control(pDX, IDC_EDIT_1PN, m_edit1PN);
  DDX_Control(pDX, IDC_EDIT_10PN, m_edit10PN);
  DDX_Control(pDX, IDC_EDIT_OUT_V, m_editVOut);
  DDX_Control(pDX, IDSUM, m_butSum);
  DDX_Control(pDX, IDC_COMBO_OUT_DBM, m_combodBmOut);
  DDX_Control(pDX, IDHELP, m_butHelp);
  DDX_Control(pDX, IDSYSVAR, m_butSysVar);
  DDX_Control(pDX, IDSTART, m_butStart);
  DDX_Control(pDX, IDGET, m_butGet);
  DDX_Control(pDX, IDEXIT, m_butExit);
  DDX_Control(pDX, IDCLEAR, m_butClear);
  DDX_Control(pDX, IDC_EDIT_STATUS, m_editStatus);
  DDX_Control(pDX, IDC_RADIO_FILE, m_radioFile);
  DDX_Control(pDX, IDC_RADIO_MAN, m_radioMan);
  DDX_Control(pDX, IDC_EDIT_FILE, m_editFile);
  DDX_Control(pDX, IDC_COMBO_9P, m_combo9P);
  DDX_Control(pDX, IDC_COMBO_9AN, m_combo9AN);
  DDX_Control(pDX, IDC_COMBO_8P, m_combo8P);
  DDX_Control(pDX, IDC_COMBO_8AN, m_combo8AN);
  DDX_Control(pDX, IDC_COMBO_7P, m_combo7P);
  DDX_Control(pDX, IDC_COMBO_7AN, m_combo7AN);
  DDX_Control(pDX, IDC_COMBO_6P, m_combo6P);
  DDX_Control(pDX, IDC_COMBO_6AN, m_combo6AN);
  DDX_Control(pDX, IDC_COMBO_5P, m_combo5P);
  DDX_Control(pDX, IDC_COMBO_5AN, m_combo5AN);
  DDX_Control(pDX, IDC_COMBO_4P, m_combo4P);
  DDX_Control(pDX, IDC_COMBO_4AN, m_combo4AN);
  DDX_Control(pDX, IDC_COMBO_3P, m_combo3P);
  DDX_Control(pDX, IDC_COMBO_3AN, m_combo3AN);
  DDX_Control(pDX, IDC_COMBO_2P, m_combo2P);
  DDX_Control(pDX, IDC_COMBO_2AN, m_combo2AN);
  DDX_Control(pDX, IDC_COMBO_1P, m_combo1P);
  DDX_Control(pDX, IDC_COMBO_1AN, m_combo1AN);
```

```
  DDX_Control(pDX, IDC_COMBO_10P, m_combo10P);
  DDX_Control(pDX, IDC_COMBO_10AN, m_combo10AN);
  DDX_Control(pDX, IDC_COMBO_9A, m_combo9A);
  DDX_Control(pDX, IDC_COMBO_8A, m_combo8A);
  DDX_Control(pDX, IDC_COMBO_7A, m_combo7A);
  DDX_Control(pDX, IDC_COMBO_6A, m_combo6A);
  DDX_Control(pDX, IDC_COMBO_5A, m_combo5A);
  DDX_Control(pDX, IDC_COMBO_4A, m_combo4A);
  DDX_Control(pDX, IDC_COMBO_3A, m_combo3A);
  DDX_Control(pDX, IDC_COMBO_2A, m_combo2A);
  DDX_Control(pDX, IDC_COMBO_1A, m_combo1A);
  DDX_Control(pDX, IDC_COMBO_10A, m_combo10A);
  DDX_Control(pDX, IDC_EDIT_9F, m_edit9F);
  DDX_Control(pDX, IDC_EDIT_8F, m_edit8F);
  DDX_Control(pDX, IDC_EDIT_7F, m_edit7F);
  DDX_Control(pDX, IDC_EDIT_6F, m_edit6F);
  DDX_Control(pDX, IDC_EDIT_5F, m_edit5F);
  DDX_Control(pDX, IDC_EDIT_4F, m_edit4F);
  DDX_Control(pDX, IDC_EDIT_3F, m_edit3F);
  DDX_Control(pDX, IDC_EDIT_2F, m_edit2F);
  DDX_Control(pDX, IDC_EDIT_10F, m_edit10F);
  DDX_Control(pDX, IDC_EDIT_1F, m_edit1F);
  DDX_CBString(pDX, IDC_COMBO_10A, m_str10A);
  DDX_CBString(pDX, IDC_COMBO_10AN, m_str10AN);
  DDX_CBString(pDX, IDC_COMBO_10P, m_str10P);
  DDX_CBString(pDX, IDC_COMBO_1A, m_str1A);
  DDX_CBString(pDX, IDC_COMBO_1AN, m_str1AN);
  DDX_CBString(pDX, IDC_COMBO_1P, m_str1P);
  DDX_CBString(pDX, IDC_COMBO_2A, m_str2A);
  DDX_CBString(pDX, IDC_COMBO_2AN, m_str2AN);
  DDX_CBString(pDX, IDC_COMBO_2P, m_str2P);
  DDX_CBString(pDX, IDC_COMBO_3A, m_str3A);
  DDX_CBString(pDX, IDC_COMBO_3AN, m_str3AN);
  DDX_CBString(pDX, IDC_COMBO_3P, m_str3P);
  DDX_CBString(pDX, IDC_COMBO_4A, m_str4A);
  DDX_CBString(pDX, IDC_COMBO_4AN, m_str4AN);
  DDX_CBString(pDX, IDC_COMBO_4P, m_str4P);
  DDX_CBString(pDX, IDC_COMBO_5A, m_str5A);
  DDX_CBString(pDX, IDC_COMBO_5AN, m_str5AN);
  DDX_CBString(pDX, IDC_COMBO_5P, m_str5P);
  DDX_CBString(pDX, IDC_COMBO_6A, m_str6A);
  DDX_CBString(pDX, IDC_COMBO_6AN, m_str6AN);
  DDX_CBString(pDX, IDC_COMBO_6P, m_str6P);
  DDX_CBString(pDX, IDC_COMBO_7A, m_str7A);
  DDX_CBString(pDX, IDC_COMBO_7AN, m_str7AN);
  DDX_CBString(pDX, IDC_COMBO_7P, m_str7P);
  DDX_CBString(pDX, IDC_COMBO_8A, m_str8A);
  DDX_CBString(pDX, IDC_COMBO_8AN, m_str8AN);
  DDX_CBString(pDX, IDC_COMBO_8P, m_str8P);
  DDX_CBString(pDX, IDC_COMBO_9A, m_str9A);
  DDX_CBString(pDX, IDC_COMBO_9AN, m_str9AN);
  DDX_CBString(pDX, IDC_COMBO_9P, m_str9P);
  DDX_Text(pDX, IDC_EDIT_FILE, m_strFile);
  DDX_CBString(pDX, IDC_COMBO_OUT_DBM, m_strdBmOut);
  DDX_Text(pDX, IDC_EDIT_10F, m_flt10F);
  DDX_Text(pDX, IDC_EDIT_1F, m_flt1F);
  DDX_Text(pDX, IDC_EDIT_3F, m_flt3F);
  DDX_Text(pDX, IDC_EDIT_4F, m_flt4F);
  DDX_Text(pDX, IDC_EDIT_5F, m_flt5F);
  DDX_Text(pDX, IDC_EDIT_6F, m_flt6F);
  DDX_Text(pDX, IDC_EDIT_7F, m_flt7F);
  DDX_Text(pDX, IDC_EDIT_8F, m_flt8F);
  DDX_Text(pDX, IDC_EDIT_9F, m_flt9F);
  DDX_Text(pDX, IDC_EDIT_2F, m_flt2F);
  DDX_Text(pDX, IDC_EDIT_OUT_V, m_fltOutV);
  DDX_Text(pDX, IDC_EDIT_10PN, m_flt10PN);
  DDX_Text(pDX, IDC_EDIT_1PN, m_flt1PN);
  DDX_Text(pDX, IDC_EDIT_2PN, m_flt2PN);
  DDX_Text(pDX, IDC_EDIT_3PN, m_flt3PN);
  DDX_Text(pDX, IDC_EDIT_4PN, m_flt4PN);
  DDX_Text(pDX, IDC_EDIT_5PN, m_flt5PN);
  DDX_Text(pDX, IDC_EDIT_6PN, m_flt6PN);
  DDX_Text(pDX, IDC_EDIT_7PN, m_flt7PN);
  DDX_Text(pDX, IDC_EDIT_8PN, m_flt8PN);
  DDX_Text(pDX, IDC_EDIT_9PN, m_flt9PN);
  //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMKDopGenDlg, CDialog)
  //{{AFX_MSG_MAP(CMKDopGenDlg)
  ON_WM_SYSCOMMAND()
  ON_WM_PAINT()
  ON_WM_QUERYDRAGICON()
```

```
ON_BN_CLICKED(IDEXIT, OnExit)
ON_BN_CLICKED(IDCLEAR, OnClear)
ON_BN_CLICKED(IDSYSVAR, OnSysvar)
ON_BN_CLICKED(IDGET, OnGet)
ON_BN_CLICKED(IDSTART, OnStart)
ON_BN_CLICKED(IDHELP, OnHelp)
ON_BN_CLICKED(IDSUM, OnSum)
ON_CBN_SELCHANGE(IDC_COMBO_10A, OnSelchangeCombo10a)
ON_CBN_SELCHANGE(IDC_COMBO_10AN, OnSelchangeCombo10an)
ON_CBN_SELCHANGE(IDC_COMBO_10P, OnSelchangeCombo10p)
ON_CBN_SELCHANGE(IDC_COMBO_1A, OnSelchangeCombo1a)
ON_CBN_SELCHANGE(IDC_COMBO_1AN, OnSelchangeCombo1an)
ON_CBN_SELCHANGE(IDC_COMBO_1P, OnSelchangeCombo1p)
ON_CBN_SELCHANGE(IDC_COMBO_2A, OnSelchangeCombo2a)
ON_CBN_SELCHANGE(IDC_COMBO_2AN, OnSelchangeCombo2an)
ON_CBN_SELCHANGE(IDC_COMBO_2P, OnSelchangeCombo2p)
ON_CBN_SELCHANGE(IDC_COMBO_3A, OnSelchangeCombo3a)
ON_CBN_SELCHANGE(IDC_COMBO_3AN, OnSelchangeCombo3an)
ON_CBN_SELCHANGE(IDC_COMBO_3P, OnSelchangeCombo3p)
ON_CBN_SELCHANGE(IDC_COMBO_4A, OnSelchangeCombo4a)
ON_CBN_SELCHANGE(IDC_COMBO_4AN, OnSelchangeCombo4an)
ON_CBN_SELCHANGE(IDC_COMBO_4P, OnSelchangeCombo4p)
ON_CBN_SELCHANGE(IDC_COMBO_5A, OnSelchangeCombo5a)
ON_CBN_SELCHANGE(IDC_COMBO_5AN, OnSelchangeCombo5an)
ON_CBN_SELCHANGE(IDC_COMBO_5P, OnSelchangeCombo5p)
ON_CBN_SELCHANGE(IDC_COMBO_6A, OnSelchangeCombo6a)
ON_CBN_SELCHANGE(IDC_COMBO_6AN, OnSelchangeCombo6an)
ON_CBN_SELCHANGE(IDC_COMBO_6P, OnSelchangeCombo6p)
ON_CBN_SELCHANGE(IDC_COMBO_7A, OnSelchangeCombo7a)
ON_CBN_SELCHANGE(IDC_COMBO_7AN, OnSelchangeCombo7an)
ON_CBN_SELCHANGE(IDC_COMBO_7P, OnSelchangeCombo7p)
ON_CBN_SELCHANGE(IDC_COMBO_8A, OnSelchangeCombo8a)
ON_CBN_SELCHANGE(IDC_COMBO_8AN, OnSelchangeCombo8an)
ON_CBN_SELCHANGE(IDC_COMBO_8P, OnSelchangeCombo8p)
ON_CBN_SELCHANGE(IDC_COMBO_9A, OnSelchangeCombo9a)
ON_CBN_SELCHANGE(IDC_COMBO_9AN, OnSelchangeCombo9an)
ON_CBN_SELCHANGE(IDC_COMBO_9P, OnSelchangeCombo9p)
ON_CBN_SELCHANGE(IDC_COMBO_OUT_DBM, OnSelchangeComboOutDbm)
ON_EN_UPDATE(IDC_EDIT_10F, OnUpdateEdit10f)
ON_EN_UPDATE(IDC_EDIT_1F, OnUpdateEdit1f)
ON_EN_UPDATE(IDC_EDIT_2F, OnUpdateEdit2f)
ON_EN_UPDATE(IDC_EDIT_3F, OnUpdateEdit3f)
ON_EN_UPDATE(IDC_EDIT_4F, OnUpdateEdit4f)
ON_EN_UPDATE(IDC_EDIT_5F, OnUpdateEdit5f)
ON_EN_UPDATE(IDC_EDIT_6F, OnUpdateEdit6f)
ON_EN_UPDATE(IDC_EDIT_7F, OnUpdateEdit7f)
ON_EN_UPDATE(IDC_EDIT_8F, OnUpdateEdit8f)
ON_EN_UPDATE(IDC_EDIT_9F, OnUpdateEdit9f)
ON_EN_UPDATE(IDC_EDIT_FILE, OnUpdateEditFile)
ON_WM_CLOSE()
ON_EN_UPDATE(IDC_EDIT_OUT_V, OnUpdateEditOutV)
ON_EN_UPDATE(IDC_EDIT_10PN, OnUpdateEdit10pn)
ON_EN_UPDATE(IDC_EDIT_1PN, OnUpdateEdit1pn)
ON_EN_UPDATE(IDC_EDIT_2PN, OnUpdateEdit2pn)
ON_EN_UPDATE(IDC_EDIT_3PN, OnUpdateEdit3pn)
ON_EN_UPDATE(IDC_EDIT_4PN, OnUpdateEdit4pn)
ON_EN_UPDATE(IDC_EDIT_5PN, OnUpdateEdit5pn)
ON_EN_UPDATE(IDC_EDIT_6PN, OnUpdateEdit6pn)
ON_EN_UPDATE(IDC_EDIT_7PN, OnUpdateEdit7pn)
ON_EN_UPDATE(IDC_EDIT_8PN, OnUpdateEdit8pn)
ON_EN_UPDATE(IDC_EDIT_9PN, OnUpdateEdit9pn)
ON_CBN_EDITUPDATE(IDC_COMBO_OUT_DBM, OnEditupdateComboOutDbm)
ON_CBN_EDITUPDATE(IDC_COMBO_10A, OnEditupdateCombo10a)
ON_CBN_EDITUPDATE(IDC_COMBO_10AN, OnEditupdateCombo10an)
ON_CBN_EDITUPDATE(IDC_COMBO_10P, OnEditupdateCombo10p)
ON_CBN_EDITUPDATE(IDC_COMBO_1A, OnEditupdateCombo1a)
ON_CBN_EDITUPDATE(IDC_COMBO_1AN, OnEditupdateCombo1an)
ON_CBN_EDITUPDATE(IDC_COMBO_1P, OnEditupdateCombo1p)
ON_CBN_EDITUPDATE(IDC_COMBO_2A, OnEditupdateCombo2a)
ON_CBN_EDITUPDATE(IDC_COMBO_2AN, OnEditupdateCombo2an)
ON_CBN_EDITUPDATE(IDC_COMBO_2P, OnEditupdateCombo2p)
ON_CBN_EDITUPDATE(IDC_COMBO_3A, OnEditupdateCombo3a)
ON_CBN_EDITUPDATE(IDC_COMBO_3AN, OnEditupdateCombo3an)
ON_CBN_EDITUPDATE(IDC_COMBO_3P, OnEditupdateCombo3p)
ON_CBN_EDITUPDATE(IDC_COMBO_4A, OnEditupdateCombo4a)
ON_CBN_EDITUPDATE(IDC_COMBO_4AN, OnEditupdateCombo4an)
ON_CBN_EDITUPDATE(IDC_COMBO_4P, OnEditupdateCombo4p)
ON_CBN_EDITUPDATE(IDC_COMBO_5A, OnEditupdateCombo5a)
ON_CBN_EDITUPDATE(IDC_COMBO_5AN, OnEditupdateCombo5an)
ON_CBN_EDITUPDATE(IDC_COMBO_5P, OnEditupdateCombo5p)
ON_CBN_EDITUPDATE(IDC_COMBO_6A, OnEditupdateCombo6a)
ON_CBN_EDITUPDATE(IDC_COMBO_6AN, OnEditupdateCombo6an)
ON_CBN_EDITUPDATE(IDC_COMBO_6P, OnEditupdateCombo6p)
```

```
  ON_CBN_EDITUPDATE(IDC_COMBO_7A, OnEditupdateCombo7a)
  ON_CBN_EDITUPDATE(IDC_COMBO_7AN, OnEditupdateCombo7an)
  ON_CBN_EDITUPDATE(IDC_COMBO_7P, OnEditupdateCombo7p)
  ON_CBN_EDITUPDATE(IDC_COMBO_8A, OnEditupdateCombo8a)
  ON_CBN_EDITUPDATE(IDC_COMBO_8AN, OnEditupdateCombo8an)
  ON_CBN_EDITUPDATE(IDC_COMBO_8P, OnEditupdateCombo8p)
  ON_CBN_EDITUPDATE(IDC_COMBO_9A, OnEditupdateCombo9a)
  ON_CBN_EDITUPDATE(IDC_COMBO_9AN, OnEditupdateCombo9an)
  ON_CBN_EDITUPDATE(IDC_COMBO_9P, OnEditupdateCombo9p)
  ON_BN_CLICKED(IDC_RADIO_FILE, OnRadioFile)
  ON_BN_CLICKED(IDC_RADIO_MAN, OnRadioMan)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMKDopGenDlg message handlers

BOOL CMKDopGenDlg::OnInitDialog()
{
  CDialog::OnInitDialog();

  // Add "About..." menu item to system menu.

  // IDM_ABOUTBOX must be in the system command range.
  ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
  ASSERT(IDM_ABOUTBOX < 0xF000);

  CMenu* pSysMenu = GetSystemMenu(FALSE);
  if (pSysMenu != NULL)
  {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
              pSysMenu->AppendMenu(MF_SEPARATOR);
              pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
  }

  // Set the icon for this dialog.  The framework does this automatically
  //  when the application's main window is not a dialog
  SetIcon(m_hIcon, TRUE);                // Set big icon
  SetIcon(m_hIcon, FALSE);         // Set small icon

  // TODO: Add extra initialization here

  //Initial default values

  m_intRangeP    = 180;      // Phase range [deg]
  m_intRangePN   = 5;        // Phase noise range [Hz]
  m_intRangeA    = 100;      // Relative amplitude range
  m_intRangeAN   = 50;       // Amplitude noise range [%]
  m_fltRangeANM  = 0;        // Gaussian distribution mean value
  m_fltRangeANV  = 1;        // Gaussian distribution variance value
  m_intRangeF    = 100000;   // Frequency range [Hz]
  m_intRangeODBM = 10;       // Output effect range [dBm]
  m_intRangeOV   = 10;       // Ouput voltage range [V]
  m_intNoiseRes  = 100;      // Amplitude noise resolution pr period
  m_fltA        =  1;        // Phase noise constant A

  // Clear status fields
  m_strStatus = "-\r\n\r\n- \tdBm\r\n- \t V \r\n- \t V \r\n- \t %";

  m_blTgl = true;            // Generator running/stopped
  m_blGaussA = true;  // Gaussian/uniform distribution


  m_pDac = new CDACOutput();        // Assign pointer to CDACOutput object
  m_pFile = new CFileHandler();     // Assign pointer to CFileHandler object

  // Assign pointer to doppler buffer
  m_psrtIQArray = new short[BUF];
  // Assign pointer to array containing signal information
  m_pfltDopplerArray = new float[9][100];

  // Set initial values
  ResetValues();

  return TRUE;  // return TRUE  unless you set the focus to a control
}


// Input: [CIntLimitComboBox*] pComboBox, [const int&] intMin, [int&] intMax
```

```
// Populates combo box pointed to by p with values ranging from intMin to intMax
// and sets the cursor on the first or the middle value
void CMKDopGenDlg::PopulateComboBox(CIntLimitComboBox* pComboBox,
                                                        const int& intMin, int& intMax)
{
  char strTemp[5];    // temporary storage for int to ascii conversion
  int intInc=0;       // number of elements in combobox

  pComboBox->ResetContent();
  pComboBox->SetLimit(intMin,intMax);

  for(int intI=intMin; intI<=intMax; intI++)
  {
        _itoa(intI,strTemp,10);
        pComboBox->AddString(strTemp);
        intInc++;
  }

  // set selected: minimum value if min is zero or one
  if(intMin==0 || intMin==1) pComboBox->SetCurSel(0);
  // set selected: center value if range is -x to x
  else pComboBox->SetCurSel(intInc/2);
}




// The system calls this to obtain the cursor to display while the user drags
//  the m_intMinimized window.
HCURSOR CMKDopGenDlg::OnQueryDragIcon()
{
  return (HCURSOR) m_hIcon;
}

// Called when pressing "Avslutt" button
// Free memory allocated buffers before exiting
void CMKDopGenDlg::OnExit()
{
  delete m_psrtIQArray;
  delete[] m_pfltDopplerArray;
  delete m_pDac;
  delete m_pFile;
  CMKDopGenDlg::DestroyWindow();
}

// Called when ending application using standard window options
// Free memory allocated buffers before exiting
void CMKDopGenDlg::OnClose()
{
  delete m_psrtIQArray;
  delete[] m_pfltDopplerArray;
  delete m_pDac;
  delete m_pFile;
  CDialog::OnClose();
}

// Called when pressing "Gjenopprett" button
// Reset all values
void CMKDopGenDlg::OnClear()
{
  ResetValues();
}

// Called when pressing "Systemvariabler" button
// First pass all initial values to member variables in
// CSysVarDlg object. Then pass new values back when pressing OK
void CMKDopGenDlg::OnSysvar()
{
  CSysVarDlg sysVarDlg; // CSysVarDlg (system variables) object

  // copy all default values to the sysvar variables
  sysVarDlg.m_intSetF = m_intRangeF;
  sysVarDlg.m_intSetA = m_intRangeA;
  sysVarDlg.m_intSetAN = m_intRangeAN;
  sysVarDlg.m_fltSetANM = m_fltRangeANM;
  sysVarDlg.m_fltSetANV = m_fltRangeANV;
  sysVarDlg.m_intSetP = m_intRangeP;
  sysVarDlg.m_intSetPN = m_intRangePN;
  sysVarDlg.m_intSetODBM = m_intRangeODBM;
  sysVarDlg.m_intSetOV = m_intRangeOV;
  sysVarDlg.m_blGaussA = m_blGaussA;
  sysVarDlg.m_intSetANSMP = m_intNoiseRes;
```

```
  // DoModal makes the CSysVarDlg dialog persistant until the
  // OK or CANCEL button is pressed. Function returns IDOK
  // When OK button is pressed, and gives the control back
  // to the main window. Values are passed.
  if(sysVarDlg.DoModal() == IDOK)
  {
        // copy all new set sysvar values to corresponding
        // values in this object
        m_intRangeF = sysVarDlg.m_intSetF;
        m_intRangeA = sysVarDlg.m_intSetA;
        m_intRangeAN = sysVarDlg.m_intSetAN;
        m_fltRangeANM = sysVarDlg.m_fltSetANM;
        m_fltRangeANV = sysVarDlg.m_fltSetANV;
        m_intRangeP = sysVarDlg.m_intSetP;
        m_intRangePN = sysVarDlg.m_intSetPN;
        m_intRangeODBM = sysVarDlg.m_intSetODBM;
        m_intRangeOV = sysVarDlg.m_intSetOV;
        m_blGaussA = sysVarDlg.m_blGaussA;
        m_intNoiseRes = sysVarDlg.m_intSetANSMP;
        ResetValues();
  }
}

// Called when pressing the "Hent..." button
// Opens a standard windows file dialog box. When OK is pressed
// it copies the filecontents into the m_fltDoppler array
void CMKDopGenDlg::OnGet()
{
  // Open file dialog and get pathname, filename and contents
  m_pFile->OpenFileDialog(this,m_strPathname, m_strFile, m_strFContents);

  // Output filename in edit control
  m_editFile.SetWindowText(m_strFile);

  // Update radio buttons if file is chosen
  if(m_strFContents.GetLength() > 0)
  {
        m_radioFile.SetCheck(BST_CHECKED);
        m_radioMan.SetCheck(BST_UNCHECKED);
  }
  else
  {
        m_radioFile.SetCheck(BST_UNCHECKED);
        m_radioMan.SetCheck(BST_CHECKED);
  }

  // Enable "summer" button
  m_butSum.EnableWindow(true);

  // Fill the doppler array with the signal information
  // contained in the m_strFContents object.
  m_pFile->FillDopplerArray(m_strFContents,m_pfltDopplerArray);
}


// Called when pressing the "Start" button
// Disables all fields but the "Stop" button when started
// Enables all fields when stopped. Calls the CDACOutput object
// to start and stop the signal output
void CMKDopGenDlg::OnStart()
{
  // output voltage 0 when signal generation is not running
  float fltVoltage = 0.0;

  // If started, button is renamed
  // output waveform
  if(m_blTgl)
  {
        m_blTgl=false;
        m_butStart.SetWindowText("STOP");
        m_pDac->OutputWaveform(m_psrtIQArray);
  }

  // If stopped, button is renamed
  // output 0 volts on the channels
  else
  {
        m_blTgl=true;
        m_butStart.SetWindowText("START");
        m_pDac->OutputVoltage(fltVoltage,0);
        m_pDac->OutputVoltage(fltVoltage,1);
  }
```

```
  // Update all controls and buttons in the application
  UpdateControlStatus(m_blTgl);
}

// Called when pressing the "summer" button
// Gets the signal information either from file or from the consol, and
// passes this information to the CComputer object. The CComputer object
// performs the mathematical computations and stores the result in the
// memory allocation pointed to by m_psrtBuffer
void CMKDopGenDlg::OnSum()
{
  double dblFrequency;// frequecny
  int intSignals = 0;        // counter variable
  int intClipping = 0;// number of clipped samples
  float fltPeak = 0;         // value of highest amplitude
  short srtI,srtJ;           // counter variables
  double dblSum = 0.0;// amplitude sum
  CString strMaxVoltOut;     // string equivalent to peak
  CComputer compObj;         // CComputer object

  // Creates a hour-glass cursor. Object destructor is called
  // automatically when exiting function
  CWaitCursor waitCursorWhileInFunction;

  // Copy data in control fields to member variables
  UpdateData(true);

  // Output external reference voltage on channel 2
  m_pDac->OutputVoltage(m_fltOutV,2);


  // If manual input is selected
  if(m_radioMan.GetCheck() == 1 && m_radioFile.GetCheck() == 0)
  {
        //Clear buffer
        for(srtI = 0; srtI < 9; srtI++)
              for(srtJ = 0; srtJ < 100; srtJ++)
                    m_pfltDopplerArray[srtI][srtJ] = 0;

        //For each field on the consol
        for(srtI=1; srtI<=10; srtI++)
        {
              // Get frequency from consol
              dblFrequency = GetFAP(FREQUENCY,srtI);
              // If frequency is != 0
              if(dblFrequency >= -m_intRangeF &&
                    dblFrequency <= m_intRangeF &&
                    dblFrequency != 0.0)
              {
                    // Store frequency multiplied with 2*PI in array
                    m_pfltDopplerArray[FREQUENCY][intSignals] =
                          (float)(dblFrequency*PI*2);
                    // Store amplitude noise reset value corresponding
                    // to the amplitude noise resolution
                    m_pfltDopplerArray[AMPLRESET][intSignals] =
                          (float)floor((m_pDac->GetUpdateRate() /
                                              dblFrequency / m_intNoiseRes)+0.5);
                    // Store phase noise reset value corresponding
                    // to the phase noise frequency
                    if(GetFAP(PHASNOISE,srtI) != 0)
                          m_pfltDopplerArray[PHASRESET][intSignals] =
                                (float)floor((m_pDac->GetUpdateRate() /
                                (GetFAP(PHASNOISE,srtI))*m_fltA));
                    // Store the amplitude noise value
                    m_pfltDopplerArray[AMPLNOISE][intSignals] = GetFAP(AMPLNOISE,srtI);
                    // Store the phase noise frequency
                    m_pfltDopplerArray[PHASNOISE][intSignals] = GetFAP(PHASNOISE,srtI);
                    // Store the relative amplitude value
                    m_pfltDopplerArray[AMPLITUDE][intSignals] = GetFAP(AMPLITUDE,srtI);
                    // Add all amplitude values
                    dblSum += m_pfltDopplerArray[AMPLITUDE][intSignals];
                    // Store the phase in radians
                    m_pfltDopplerArray[PHASESHFT][intSignals] =
                          compObj.DegToRad(GetFAP(PHASESHFT,srtI));
                    // Keeps track on the number of signals
                    intSignals++;
              }
        }
  }

  // If a modulation file is selected
  else if(m_radioFile.GetCheck() == 1 && m_radioMan.GetCheck() == 0)
```

```
  {
        // For each possible number of signals
        for(srtI=0; srtI<100; srtI++)
        {
              // If there is a signal in the array
              if(m_pfltDopplerArray[FREQUENCY][srtI] != 0)
              {
                    // Store amplitude noise reset value corresponding
                    // to the amplitude noise resolution
                    m_pfltDopplerArray[AMPLRESET][srtI] =
                          (float)floor((m_pDac->GetUpdateRate() /
                          m_pfltDopplerArray[FREQUENCY][srtI] / m_intNoiseRes)+0.5);
                    // Store phase noise reset value corresponding
                    // to the phase noise frequency
                    if(m_pfltDopplerArray[PHASNOISE][srtI] != 0)
                          m_pfltDopplerArray[PHASRESET][srtI] =
                          (float)floor((m_pDac->GetUpdateRate() /
                                m_pfltDopplerArray[PHASNOISE][srtI])) * m_fltA;
                    // Add all amplitude values
                    dblSum += m_pfltDopplerArray[AMPLITUDE][intSignals];
                    // Keeps track on the number of signals
                    intSignals++;
              }
        }
  }

  // Calculate constant. This value is calculated in advance, to minimize
  // the computational load when adding the signals
  compObj.ComputeConstant(dblSum, intSignals, atoi(m_strdBmOut));

  // Calculate and fill buffer with amplitude values.
  // The function will in addition give value to the number of clipped amplitudes
  // and the highest amplitude peak
  compObj.FillIQArray(m_psrtIQArray, m_pfltDopplerArray, m_fltOutV, m_blGaussA,
                          m_fltRangeANM, m_fltRangeANV, intSignals, intClipping,
fltPeak);

  strMaxVoltOut.Format("%.3f",fltPeak);

  // Disable the "summer" button and enable the "Start" button.
  m_butSum.EnableWindow(false);
  m_butStart.EnableWindow(true);

  // Output statistical results in status window
  UpdateStatusWindow(strMaxVoltOut,intClipping);
}


// Input:      [CString&] strMax, [int&] intClipping
// Outputs the following statistical information in the status window:
// output effect , max amplitude, clipping percentage and voltage limits
void CMKDopGenDlg::UpdateStatusWindow(CString &strMax, int &intClipping)
{
  CString strStatus;               // status string
  CString strMaxVoltOut;           // max output voltage
  CString strClipping;       // string equivalent to clip_percent
  double dblClipPercentage;  // percentage of signal clipping

  dblClipPercentage = (double)intClipping/BUF*100.0;

  strMaxVoltOut.Format("%.3f",m_fltOutV);
  strClipping.Format("%.3f",dblClipPercentage);

  if(m_radioFile.GetCheck()==BST_CHECKED)
        strStatus = m_strFile + "\r\n\r\n";
  else
        strStatus = "Konsoll\r\n\r\n";

  strStatus += m_strdBmOut   + "\tdBm \r\n";
  strStatus += strMax        + "\t V  \r\n";
  strStatus += strMaxVoltOut + "\t V  \r\n";
  strStatus += strClipping   + "\t %  \r\n";

  m_editStatus.SetWindowText(strStatus);
}

// Called when "hjelp" button is pressed
// Opens the CHelpDlg dialog
void CMKDopGenDlg::OnHelp()
{

  AfxMessageBox("Hjelpefunksjon er ikke implementert");
```

```
  // Not activated:

  //CHelpDlg hlp; // CHelpDlg object
  //hlp.DoModal(); // Open "modal" dialog box
}

// Resets contents in the application, based on the current
// default values. Called whenever values are changed in the
// system variable dialog, or when pressing "Gjenopprett"
void CMKDopGenDlg::ResetValues()
{
  // Convert output voltage range to string
   char strRangeOV[10];      // output voltage range
  itoa(m_intRangeOV,strRangeOV,10);

  // Clear CString objects
  m_strFContents = "";
  m_strPathname = "";
  m_strFile = "";

  //Reset edit boxes
  //Set limit as m_intRangeF
  m_edit1F.SetWindowText("0");
  m_edit1F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit2F.SetWindowText("0");
  m_edit2F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit3F.SetWindowText("0");
  m_edit3F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit4F.SetWindowText("0");
  m_edit4F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit5F.SetWindowText("0");
  m_edit5F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit6F.SetWindowText("0");
  m_edit6F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit7F.SetWindowText("0");
  m_edit7F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit8F.SetWindowText("0");
  m_edit8F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit9F.SetWindowText("0");
  m_edit9F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit10F.SetWindowText("0");
  m_edit10F.SetLimit(-m_intRangeF,m_intRangeF);
  m_edit1PN.SetWindowText("0");
  m_edit1PN.SetLimit(0,m_intRangePN);
  m_edit2PN.SetWindowText("0");
  m_edit2PN.SetLimit(0,m_intRangePN);
  m_edit3PN.SetWindowText("0");
  m_edit3PN.SetLimit(0,m_intRangePN);
  m_edit4PN.SetWindowText("0");
  m_edit4PN.SetLimit(0,m_intRangePN);
  m_edit5PN.SetWindowText("0");
  m_edit5PN.SetLimit(0,m_intRangePN);
  m_edit6PN.SetWindowText("0");
  m_edit6PN.SetLimit(0,m_intRangePN);
  m_edit7PN.SetWindowText("0");
  m_edit7PN.SetLimit(0,m_intRangePN);
  m_edit8PN.SetWindowText("0");
  m_edit8PN.SetLimit(0,m_intRangePN);
  m_edit9PN.SetWindowText("0");
  m_edit9PN.SetLimit(0,m_intRangePN);
  m_edit10PN.SetWindowText("0");
  m_edit10PN.SetLimit(0,m_intRangePN);
  m_editVOut.SetWindowText(strRangeOV);
  m_editVOut.SetLimit(0,m_intRangeOV);
  m_editFile.SetWindowText("");
  m_editStatus.SetWindowText(m_strStatus);

  // Populate output combobox
  PopulateComboBox(&m_combodBmOut,-m_intRangeODBM,m_intRangeODBM);

  // Populate amplitude comboboxes
  PopulateComboBox(&m_combo1A,1,m_intRangeA);
  PopulateComboBox(&m_combo2A,1,m_intRangeA);
  PopulateComboBox(&m_combo3A,1,m_intRangeA);
  PopulateComboBox(&m_combo4A,1,m_intRangeA);
  PopulateComboBox(&m_combo5A,1,m_intRangeA);
  PopulateComboBox(&m_combo6A,1,m_intRangeA);
  PopulateComboBox(&m_combo7A,1,m_intRangeA);
  PopulateComboBox(&m_combo8A,1,m_intRangeA);
  PopulateComboBox(&m_combo9A,1,m_intRangeA);
  PopulateComboBox(&m_combo10A,1,m_intRangeA);

  // Populate amplitude noise comboboxes
```

```
  PopulateComboBox(&m_combo1AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo2AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo3AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo4AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo5AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo6AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo7AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo8AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo9AN,0,m_intRangeAN);
  PopulateComboBox(&m_combo10AN,0,m_intRangeAN);

  // Populate phase combo boxes
  PopulateComboBox(&m_combo1P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo2P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo3P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo4P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo5P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo6P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo7P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo8P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo9P,-m_intRangeP,m_intRangeP);
  PopulateComboBox(&m_combo10P,-m_intRangeP,m_intRangeP);

  // Disable button
  m_butSum.EnableWindow(false);

  //Initialize radio buttons
  m_radioMan.SetCheck(BST_CHECKED);
  m_radioFile.SetCheck(BST_UNCHECKED);
}

// Input:      [bool&] blEnableControls
// Enables/disables all fields in the application
void CMKDopGenDlg::UpdateControlStatus(bool& blEnableControls)
{
  m_combodBmOut.EnableWindow(blEnableControls);
  m_editVOut.EnableWindow(blEnableControls);
  m_radioFile.EnableWindow(blEnableControls);
  m_radioMan.EnableWindow(blEnableControls);
  m_editFile.EnableWindow(blEnableControls);
  m_edit9PN.EnableWindow(blEnableControls);
  m_combo9P.EnableWindow(blEnableControls);;
  m_combo9AN.EnableWindow(blEnableControls);
  m_edit8PN.EnableWindow(blEnableControls);
  m_combo8P.EnableWindow(blEnableControls);
  m_combo8AN.EnableWindow(blEnableControls);
  m_edit7PN.EnableWindow(blEnableControls);
  m_combo7P.EnableWindow(blEnableControls);
  m_combo7AN.EnableWindow(blEnableControls);
  m_edit6PN.EnableWindow(blEnableControls);
  m_combo6P.EnableWindow(blEnableControls);
  m_combo6AN.EnableWindow(blEnableControls);
  m_edit5PN.EnableWindow(blEnableControls);
  m_combo5P.EnableWindow(blEnableControls);
  m_combo5AN.EnableWindow(blEnableControls);
  m_edit4PN.EnableWindow(blEnableControls);
  m_combo4P.EnableWindow(blEnableControls);
  m_combo4AN.EnableWindow(blEnableControls);
  m_edit3PN.EnableWindow(blEnableControls);
  m_combo3P.EnableWindow(blEnableControls);
  m_combo3AN.EnableWindow(blEnableControls);
  m_edit2PN.EnableWindow(blEnableControls);
  m_combo2P.EnableWindow(blEnableControls);
  m_combo2AN.EnableWindow(blEnableControls);
  m_edit1PN.EnableWindow(blEnableControls);
  m_combo1AN.EnableWindow(blEnableControls);
  m_edit10PN.EnableWindow(blEnableControls);
  m_combo10P.EnableWindow(blEnableControls);
  m_combo10AN.EnableWindow(blEnableControls);
  m_combo9A.EnableWindow(blEnableControls);
  m_combo8A.EnableWindow(blEnableControls);
  m_combo7A.EnableWindow(blEnableControls);
  m_combo6A.EnableWindow(blEnableControls);
  m_combo5A.EnableWindow(blEnableControls);
  m_combo4A.EnableWindow(blEnableControls);
  m_combo3A.EnableWindow(blEnableControls);
  m_combo2A.EnableWindow(blEnableControls);
  m_combo1A.EnableWindow(blEnableControls);
  m_combo10A.EnableWindow(blEnableControls);
  m_edit9F.EnableWindow(blEnableControls);
  m_edit8F.EnableWindow(blEnableControls);
  m_edit7F.EnableWindow(blEnableControls);
  m_edit6F.EnableWindow(blEnableControls);
```

```
  m_edit5F.EnableWindow(blEnableControls);
  m_edit4F.EnableWindow(blEnableControls);
  m_edit3F.EnableWindow(blEnableControls);
  m_edit2F.EnableWindow(blEnableControls);
  m_edit10F.EnableWindow(blEnableControls);
  m_edit1F.EnableWindow(blEnableControls);
  m_butSysVar.EnableWindow(blEnableControls);
  m_butHelp.EnableWindow(blEnableControls);
  m_butGet.EnableWindow(blEnableControls);
  m_butExit.EnableWindow(blEnableControls);
  m_butClear.EnableWindow(blEnableControls);
}

// Input:      [const short&] srtType, [short&] srtI
// Return:     [float] control field value
// Function returns the numerical value corresponding to
// the row and coloumn of the consol
float CMKDopGenDlg::GetFAP(const short &srtType, short& srtI)
{
  double dblResult; // return value

  switch(srtType)
  {
        case FREQUENCY: // Get Frequency
              switch(srtI)
              {
                    case 1:  dblResult= m_flt1F; break;
                    case 2:  dblResult= m_flt2F; break;
                    case 3:  dblResult= m_flt3F; break;
                    case 4:  dblResult= m_flt4F; break;
                    case 5:  dblResult= m_flt5F; break;
                    case 6:  dblResult= m_flt6F; break;
                    case 7:  dblResult= m_flt7F; break;
                    case 8:  dblResult= m_flt8F; break;
                    case 9:  dblResult= m_flt9F; break;
                    case 10: dblResult= m_flt10F; break;
                    default: dblResult=-1; break;
              }
              break;

        case AMPLITUDE: // Get Amplitude
              switch(srtI)
              {
                    case 1: dblResult= atof(m_str1A); break;
                    case 2: dblResult= atof(m_str2A); break;
                    case 3: dblResult= atof(m_str3A); break;
                    case 4: dblResult= atof(m_str4A); break;
                    case 5: dblResult= atof(m_str5A); break;
                    case 6: dblResult= atof(m_str6A); break;
                    case 7: dblResult= atof(m_str7A); break;
                    case 8: dblResult= atof(m_str8A); break;
                    case 9: dblResult= atof(m_str9A); break;
                    case 10: dblResult= atof(m_str10A); break;
                    default: dblResult=-1; break;
              }
              break;

        case AMPLNOISE: // Get Amplitude noise
              switch(srtI)
              {
                    case 1: dblResult= atof(m_str1AN); break;
                    case 2: dblResult= atof(m_str2AN); break;
                    case 3: dblResult= atof(m_str3AN); break;
                    case 4: dblResult= atof(m_str4AN); break;
                    case 5: dblResult= atof(m_str5AN); break;
                    case 6: dblResult= atof(m_str6AN); break;
                    case 7: dblResult= atof(m_str7AN); break;
                    case 8: dblResult= atof(m_str8AN); break;
                    case 9: dblResult= atof(m_str9AN); break;
                    case 10: dblResult= atof(m_str10AN); break;
                    default: dblResult=-1; break;
              }
              break;

        case PHASESHFT: // Get Phase
              switch(srtI)
              {
                    case 1: dblResult= atof(m_str1P); break;
                    case 2: dblResult= atof(m_str2P); break;
                    case 3: dblResult= atof(m_str3P); break;
                    case 4: dblResult= atof(m_str4P); break;
                    case 5: dblResult= atof(m_str5P); break;
                    case 6: dblResult= atof(m_str6P); break;
```

```
                        case 7: dblResult= atof(m_str7P); break;
                        case 8: dblResult= atof(m_str8P); break;
                        case 9: dblResult= atof(m_str9P); break;
                        case 10: dblResult= atof(m_str10P); break;
                        default: dblResult=-1; break;
                }
                break;

        case PHASNOISE: // Get Phase Noise
                switch(srtI)
                {
                        case 1: dblResult= m_flt1PN; break;
                        case 2: dblResult= m_flt2PN; break;
                        case 3: dblResult= m_flt3PN; break;
                        case 4: dblResult= m_flt4PN; break;
                        case 5: dblResult= m_flt5PN; break;
                        case 6: dblResult= m_flt6PN; break;
                        case 7: dblResult= m_flt7PN; break;
                        case 8: dblResult= m_flt8PN; break;
                        case 9: dblResult= m_flt9PN; break;
                        case 10: dblResult= m_flt10PN; break;
                        default: dblResult=-1; break;
                }
                break;

        default: dblResult=-1; break;
  }
  return (float)dblResult;
}


// Called when file radio button is pressed
void CMKDopGenDlg::OnRadioFile()
{
  // Do not allow to select from file if no file selected.
  if(m_strFContents.IsEmpty())
        UpdateButtons(1);

  // If file is present,
  else
  {
        m_pFile->FillDopplerArray(m_strFContents,m_pfltDopplerArray);
        UpdateButtons(0);
  }

}

// Called when manual input radio button is pressed
// Enables the "Summer" button
void CMKDopGenDlg::OnRadioMan()
{
  UpdateButtons(0);
}



// Input:      [const short&] srtRadioButton
// Updates the "Summer", "Start" and file/manual radio buttons
// Function is used whenever an update of the values has occured
void CMKDopGenDlg::UpdateButtons(const short &srtRadioButton)
{
  //Enable "Summer" and disable "Start"
  m_butSum.EnableWindow(true);
  m_butStart.EnableWindow(false);

  switch(srtRadioButton)
  {
  case 1: // Change in one of the 10 manual input signal fields
        m_radioFile.SetCheck(BST_UNCHECKED);
        m_radioMan.SetCheck(BST_CHECKED);
        break;
  case 2: // Change of file input
        m_radioFile.SetCheck(BST_CHECKED);
        m_radioMan.SetCheck(BST_UNCHECKED);
        break;
  case 0: default: // Change of output values.
                        // Does nothing with the radio buttons
        break;
  }
}


// Function is called when selecting the standard windows
```

```
// menu in the upper left corner of the application
void CMKDopGenDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
  if ((nID & 0xFFF0) == IDM_ABOUTBOX)
  {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
  }
  else
  {
        CDialog::OnSysCommand(nID, lParam);
  }
}

// If you add a m_intMinimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
void CMKDopGenDlg::OnPaint()
{
  if (IsIconic())
  {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
  }
  else
  {
        CDialog::OnPaint();
  }
}

// Update buttons when editing frequency
void CMKDopGenDlg::OnUpdateEdit1f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit2f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit3f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit4f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit5f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit6f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit7f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit8f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit9f()             {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit10f()            {UpdateButtons(1);}

// Update buttons when editing or selecting amplitude
void CMKDopGenDlg::OnSelchangeCombo1a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo2a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo3a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo4a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo5a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo6a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo7a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo8a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo9a()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo10a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo1a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo2a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo3a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo4a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo5a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo6a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo7a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo8a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo9a()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo10a()   {UpdateButtons(1);}

// Update buttons when editing or selecting amplitude noise
void CMKDopGenDlg::OnSelchangeCombo1an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo2an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo3an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo4an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo5an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo6an()  {UpdateButtons(1);}
```

```
void CMKDopGenDlg::OnSelchangeCombo7an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo8an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo9an()  {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo10an() {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo1an() {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo2an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo3an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo4an() {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo5an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo6an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo7an() {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo8an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo9an()   {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo10an()  {UpdateButtons(1);}

// Update buttons when editing or selecting phase
void CMKDopGenDlg::OnSelchangeCombo1p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo2p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo3p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo4p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo5p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo6p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo7p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo8p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo9p()         {UpdateButtons(1);}
void CMKDopGenDlg::OnSelchangeCombo10p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo1p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo2p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo3p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo4p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo5p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo6p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo7p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo8p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo9p()  {UpdateButtons(1);}
void CMKDopGenDlg::OnEditupdateCombo10p() {UpdateButtons(1);}

// Update buttons when editing phase noise
void CMKDopGenDlg::OnUpdateEdit1pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit2pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit3pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit4pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit5pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit6pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit7pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit8pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit9pn()            {UpdateButtons(1);}
void CMKDopGenDlg::OnUpdateEdit10pn()           {UpdateButtons(1);}

// Update buttons when editing file
void CMKDopGenDlg::OnUpdateEditFile()           {UpdateButtons(2);}

// Update buttons when editing max output voltage
void CMKDopGenDlg::OnUpdateEditOutV()           {UpdateButtons(0);}

// Update buttons when editing or selecting output effect
void CMKDopGenDlg::OnSelchangeComboOutDbm()     {UpdateButtons(0);}
void CMKDopGenDlg::OnEditupdateComboOutDbm(){UpdateButtons(0);}
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            CSysVarDlg                                          **/
/** Filename:         SysVarDlg.h                                         **/
/** Filetype:         Declaration file                                    **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/


#if !defined(AFX_CSysVarDlg_H__A10B110F_4185_4849_B2D2_94354DD7E5AC__INCLUDED_)
#define AFX_CSysVarDlg_H__A10B110F_4185_4849_B2D2_94354DD7E5AC__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SysVar.h : header file
//

#include "FloatEdit.h"


/////////////////////////////////////////////////////////////////////////////
// CSysVarDlg dialog

class CSysVarDlg : public CDialog
{
// Construction
public:
  CSysVarDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
  //{{AFX_DATA(CSysVarDlg)
  enum { IDD = IDD_SYSTEMVAR };
  CFloatEdit    m_editSetPN;
  CFloatEdit    m_edit_setANSMP;
  CFloatEdit    m_edit_setOV;
  CFloatEdit    m_edit_setODBM;
  CButton       m_radio_setAU;
  CButton       m_radio_setAG;
  CFloatEdit    m_edit_setANV;
  CFloatEdit    m_edit_setANM;
  CFloatEdit    m_edit_setP;
  CFloatEdit    m_edit_setF;
  CFloatEdit    m_edit_setAN;
  CFloatEdit    m_edit_setA;
  float  m_fltSetANM;
  float  m_fltSetANV;
  int           m_intSetA;
  int           m_intSetAN;
  int           m_intSetF;
  int           m_intSetP;
  int           m_intSetODBM;
  int           m_intSetOV;
  int           m_intSetANSMP;
  int           m_intSetPN;
  //}}AFX_DATA

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CSysVarDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
  //}}AFX_VIRTUAL

public: // Member variables
  bool m_blGaussA;    // amp noise uniform/gauss
  bool m_blODBM;             // output dBm/Volts

private: // Operations

  // Enables/Disables gauss edit controls according to radio button choices
  void UpdateSysVar();



// Implementation
protected:

  // Generated message map functions
  //{{AFX_MSG(CSysVarDlg)
  virtual BOOL OnInitDialog();
  afx_msg void OnRadioSetAGau();
```

```
  afx_msg void OnRadioSetAUnf();
  //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif // !defined(AFX_CSysVarDlg_H__A10B110F_4185_4849_B2D2_94354DD7E5AC__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator                **/
/** Class:           CsysVarDlg                                                **/
/** Filename:        SysVarDlg.cpp                                             **/
/** Filetype:        Implementation file                                       **/
/** Author/year:     Alexander Iversen / 2002                                  **/
/*****************************************************************************/

#include "stdafx.h"
#include "MKDopGen.h"
#include "SysVarDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSysVarDlg dialog


CSysVarDlg::CSysVarDlg(CWnd* pParent /*=NULL*/)
  : CDialog(CSysVarDlg::IDD, pParent)
{
  //{{AFX_DATA_INIT(CSysVarDlg)
  m_fltSetANM = 0.0f;
  m_fltSetANV = 0.0f;
  m_intSetA = 0;
  m_intSetAN = 0;
  m_intSetF = 0;
  m_intSetP = 0;
  m_intSetODBM = 0;
  m_intSetOV = 0;
  m_intSetANSMP = 0;
  m_intSetPN = 0;
  //}}AFX_DATA_INIT
}


void CSysVarDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CSysVarDlg)
  DDX_Control(pDX, IDC_EDIT_SET_PN, m_editSetPN);
  DDX_Control(pDX, IDC_EDIT_SET_AN_SMP, m_edit_setANSMP);
  DDX_Control(pDX, IDC_EDIT_SET_O_V, m_edit_setOV);
  DDX_Control(pDX, IDC_EDIT_SET_O_DBM, m_edit_setODBM);
  DDX_Control(pDX, IDC_RADIO_SET_A_UNF, m_radio_setAU);
  DDX_Control(pDX, IDC_RADIO_SET_A_GAU, m_radio_setAG);
  DDX_Control(pDX, IDC_EDIT_SET_AN_V, m_edit_setANV);
  DDX_Control(pDX, IDC_EDIT_SET_AN_M, m_edit_setANM);
  DDX_Control(pDX, IDC_EDIT_SET_P, m_edit_setP);
  DDX_Control(pDX, IDC_EDIT_SET_F, m_edit_setF);
  DDX_Control(pDX, IDC_EDIT_SET_AN, m_edit_setAN);
  DDX_Control(pDX, IDC_EDIT_SET_A, m_edit_setA);
  DDX_Text(pDX, IDC_EDIT_SET_AN_M, m_fltSetANM);
  DDX_Text(pDX, IDC_EDIT_SET_AN_V, m_fltSetANV);
  DDX_Text(pDX, IDC_EDIT_SET_A, m_intSetA);
  DDX_Text(pDX, IDC_EDIT_SET_AN, m_intSetAN);
  DDX_Text(pDX, IDC_EDIT_SET_F, m_intSetF);
  DDX_Text(pDX, IDC_EDIT_SET_P, m_intSetP);
  DDX_Text(pDX, IDC_EDIT_SET_O_DBM, m_intSetODBM);
  DDX_Text(pDX, IDC_EDIT_SET_O_V, m_intSetOV);
  DDX_Text(pDX, IDC_EDIT_SET_AN_SMP, m_intSetANSMP);
  DDX_Text(pDX, IDC_EDIT_SET_PN, m_intSetPN);
  //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CSysVarDlg, CDialog)
  //{{AFX_MSG_MAP(CSysVarDlg)
  ON_BN_CLICKED(IDC_RADIO_SET_A_GAU, OnRadioSetAGau)
  ON_BN_CLICKED(IDC_RADIO_SET_A_UNF, OnRadioSetAUnf)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSysVarDlg message handlers

BOOL CSysVarDlg::OnInitDialog()
{
  CDialog::OnInitDialog();
```

```
  // Initialise radio buttons
  if(m_blGaussA)
  {
        m_radio_setAU.SetCheck(BST_UNCHECKED);
        m_radio_setAG.SetCheck(BST_CHECKED);
  }
  else
  {
        m_radio_setAU.SetCheck(BST_CHECKED);
        m_radio_setAG.SetCheck(BST_UNCHECKED);
  }

  UpdateSysVar();

  return TRUE;  // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

// Enables/Disables gauss edit controls
// according to radio button choices
void CSysVarDlg::UpdateSysVar()
{
        m_edit_setANM.EnableWindow(m_blGaussA);
        m_edit_setANV.EnableWindow(m_blGaussA);
}

// Amplitude noise with gaussian distribution
// Updates edit controls
void CSysVarDlg::OnRadioSetAGau()
{
  m_blGaussA=true;
  UpdateSysVar();
}

// Amplitude noise with uniform distribution
// Updates edit controls
void CSysVarDlg::OnRadioSetAUnf()
{
  m_blGaussA=false;
  UpdateSysVar();
}
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            ChelpDlg                                            **/
/** Filename:         HelpDlg.h                                           **/
/** Filetype:         Declaration file                                    **/
/** Author/year:      Alexander Iversen / 2002                             **/
/*************************************************************************/

#if !defined(AFX_HELP_H__52FA15E0_4905_469B_82EA_4FAA2E45C419__INCLUDED_)
#define AFX_HELP_H__52FA15E0_4905_469B_82EA_4FAA2E45C419__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Help.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CHelpDlg dialog

class CHelpDlg : public CDialog
{
// Construction
public:
  CHelpDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
  //{{AFX_DATA(CHelpDlg)
  enum { IDD = IDD_HELP };
        // NOTE: the ClassWizard will add data members here
  //}}AFX_DATA


// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CHelpDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
  //}}AFX_VIRTUAL

// Implementation
protected:

  // Generated message map functions
  //{{AFX_MSG(CHelpDlg)
        // NOTE: the ClassWizard will add member functions here
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif // !defined(AFX_HELP_H__52FA15E0_4905_469B_82EA_4FAA2E45C419__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator              **/
/** Class:            ChelpDlg                                                **/
/** Filename:         HelpDlg.cpp                                             **/
/** Filetype:         Implementation file                                     **/
/** Author/year:      Alexander Iversen / 2002                                **/
/*****************************************************************************/

#include "stdafx.h"
#include "MKDopGen.h"
#include "HelpDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CHelpDlg dialog


CHelpDlg::CHelpDlg(CWnd* pParent /*=NULL*/)
  : CDialog(CHelpDlg::IDD, pParent)
{
  //{{AFX_DATA_INIT(CHelpDlg)
        // NOTE: the ClassWizard will add member initialization here
  //}}AFX_DATA_INIT
}


void CHelpDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CHelpDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
  //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CHelpDlg, CDialog)
  //{{AFX_MSG_MAP(CHelpDlg)
        // NOTE: the ClassWizard will add message map macros here
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CHelpDlg message handlers
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            CfloatEdit                                          **/
/** Filename:         FloatEdit.h                                         **/
/** Description:      Declaration file                                    **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/


#if !defined(AFX_FLOATEDIT_H__ADA38395_C216_4439_AC1F_DF5D37EB3E43__INCLUDED_)
#define AFX_FLOATEDIT_H__ADA38395_C216_4439_AC1F_DF5D37EB3E43__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// FloatEdit.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CFloatEdit window

class CFloatEdit : public CEdit
{
// Construction
public:
  CFloatEdit();

// Attributes
public:

// Operations
public:

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CFloatEdit)
  //}}AFX_VIRTUAL

// Implementation
public:
  virtual ~CFloatEdit();

  // Generated message map functions
protected:
  //{{AFX_MSG(CFloatEdit)
  afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
  afx_msg void OnKillFocus(CWnd* pNewWnd);
  //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif // !defined(AFX_FLOATEDIT_H__ADA38395_C216_4439_AC1F_DF5D37EB3E43__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator              **/
/** Class:            CfloatEdit                                              **/
/** Filename:         FloatEdit.cpp                                           **/
/** Description:      Implementation file                                     **/
/** Author/year:      Alexander Iversen / 2002                                **/
/*****************************************************************************/


#include "stdafx.h"
#include "MKDopGen.h"
#include "FloatEdit.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFloatEdit

CFloatEdit::CFloatEdit()
{
}

CFloatEdit::~CFloatEdit()
{
}


BEGIN_MESSAGE_MAP(CFloatEdit, CEdit)
  //{{AFX_MSG_MAP(CFloatEdit)
  ON_WM_CHAR()
  ON_WM_KILLFOCUS()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFloatEdit message handlers

// Function is called when entering a value in the edit box
// Will only accept numbers, and discard all other input
void CFloatEdit::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
  if(nChar=='-' || nChar=='.' || ((nChar>='0')&&(nChar<='9')) || nChar==8)
        CEdit::OnChar(nChar, nRepCnt, nFlags);
}

// Function is called on edit box kill focus
// Sets contents "0" if empty
void CFloatEdit::OnKillFocus(CWnd* pNewWnd)
{
  CEdit::OnKillFocus(pNewWnd);

  CString strContent; // string containing edit box contents

  //Get text in edit box
  GetWindowText(strContent);

  //If box is empty, set value=0
  if(strContent.GetLength() == 0)
  {
        SetWindowText("0");
  }
}
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            CfloatLimitEdit                                     **/
/** Filename:         FloatLimitEdit.h                                    **/
/** Filetype:         Declaration file                                    **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/

#if !defined(AFX_FloatLimitEdit_H__498143FD_EA64_4170_92BD_1409A69222E3__INCLUDED_)
#define AFX_FloatLimitEdit_H__498143FD_EA64_4170_92BD_1409A69222E3__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// FloatLimitEdit.h : header file
//
#include "FloatEdit.h"

/////////////////////////////////////////////////////////////////////////////
// CFloatLimitEdit window

class CFloatLimitEdit : public CFloatEdit
{
// Construction
public:
  CFloatLimitEdit();

// Attributes
private:
  double m_dbMin;
  double m_dbMax;
// Operations
public:
  void SetLimit(const int &intMin, int &intMax);

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CFloatLimitEdit)
  //}}AFX_VIRTUAL

// Implementation
public:
  virtual ~CFloatLimitEdit();

  // Generated message map functions
protected:
  //{{AFX_MSG(CFloatLimitEdit)
  afx_msg void OnKillFocus(CWnd* pNewWnd);
  afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
  //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif // !defined(AFX_FloatLimitEdit_H__498143FD_EA64_4170_92BD_1409A69222E3__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator               **/
/** Class:           CfloatLimitEdit                                          **/
/** Filename:        FloatLimitEdit.cpp                                       **/
/** Filetype:        Implementation file                                      **/
/** Author/year:     Alexander Iversen / 2002                                 **/
/*****************************************************************************/


#include "stdafx.h"
#include "MKDopGen.h"
#include "FloatLimitEdit.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFloatLimitEdit

CFloatLimitEdit::CFloatLimitEdit()
{
}

CFloatLimitEdit::~CFloatLimitEdit()
{
}


BEGIN_MESSAGE_MAP(CFloatLimitEdit, CEdit)
  //{{AFX_MSG_MAP(CFloatLimitEdit)
  ON_WM_KILLFOCUS()
  ON_WM_CHAR()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFloatLimitEdit message handlers


// Function is called when entering a value in the edit box
// Inherits the behaviour of CFloatEdit
void CFloatLimitEdit::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        CFloatEdit::OnChar(nChar, nRepCnt, nFlags);
}

// Function is called before on edit box kill focus
// Function controls the input, and only allows values within the
// limits set by the SetLimit function
void CFloatLimitEdit::OnKillFocus(CWnd* pNewWnd)
{
  CEdit::OnKillFocus(pNewWnd);

  CString strContent;        // contents of edit box
  CString strMin,strMax;     // min and max values in string format
  double dblValue;           // value of edit box input
  int intLength;                   // string length of edit box input

  strMin.Format("%.1f",m_dbMin);
  strMax.Format("%.1f",m_dbMax);

  //Get text in edit box
  GetWindowText(strContent);

  dblValue = atof(strContent);
  intLength = strlen(strContent);

  //If box is empty, set value=0
  if(intLength == 0)
  {
        SetWindowText("0");
  }
  //If value exceeds limits, produce an error message
  if(dblValue<m_dbMin || dblValue > m_dbMax)
  {
        AfxMessageBox("Verdi utenfor gyldig område.\nGyldig område er mellom "
                        + strMin + " og " + strMax);
        SetWindowText("0");
        SetFocus();
  }
```

```
}

// Input:      [const int&] intMin, [int&] intMax
// Sets the limits of the edit box
void CFloatLimitEdit::SetLimit(const int &intMin, int &intMax)
  {
        m_dbMin = intMin;
        m_dbMax = intMax;
  }
```

```
/*************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator          **/
/** Class:            CintLimitComboBox                                   **/
/** Filename:         IntLimitComboBox.h                                  **/
/** Filetype:         Declaration file                                    **/
/** Author/year:      Alexander Iversen / 2002                            **/
/*************************************************************************/

#if !defined(AFX_IntLimitComboBox_H__16539923_ADF4_498C_B1F0_C4B0D2B88C5D__INCLUDED_)
#define AFX_IntLimitComboBox_H__16539923_ADF4_498C_B1F0_C4B0D2B88C5D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// IntLimitComboBox.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CIntLimitComboBox window

class CIntLimitComboBox : public CComboBox
{
// Construction
public:
  CIntLimitComboBox();

// Attributes
private:
  double m_dbMin;            // minimum legal value
  double m_dbMax;            // maximum legal value
// Operations
public:

  // Set input limits on CIntLimitComboBoxes
  void SetLimit(const double &mn, double &mx);
  void SetLimit(const int &mn, int &mx);

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CIntLimitComboBox)
  //}}AFX_VIRTUAL

// Implementation
public:
  virtual ~CIntLimitComboBox();

  // Generated message map functions
protected:
  //{{AFX_MSG(CIntLimitComboBox)
  afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor);
  //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif //
!defined(AFX_IntLimitComboBox_H__16539923_ADF4_498C_B1F0_C4B0D2B88C5D__INCLUDED_)
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator              **/
/** Class:            CintLimitComboBox                                       **/
/** Filename:         IntLimitComboBox.cpp                                    **/
/** Filetype:         Implementation file                                     **/
/** Author/year:      Alexander Iversen / 2002                                **/
/*****************************************************************************/

#include "stdafx.h"
#include "MKDopGen.h"
#include "IntLimitComboBox.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CIntLimitComboBox

CIntLimitComboBox::CIntLimitComboBox()
{
}

CIntLimitComboBox::~CIntLimitComboBox()
{
}


BEGIN_MESSAGE_MAP(CIntLimitComboBox, CComboBox)
  //{{AFX_MSG_MAP(CIntLimitComboBox)
  ON_WM_CTLCOLOR()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIntLimitComboBox message handlers



// Function is called before drawing the contents of the combo box
// Function controls the input, and only allows values within the
// limits set by the SetLimit function
HBRUSH CIntLimitComboBox::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
  HBRUSH hbr = CComboBox::OnCtlColor(pDC, pWnd, nCtlColor);

  // strA holds contents of combo box
  // strB holds the CString equivalent of the value of the content
  CString strA,strB;
  // minimum and maximum values
  CString      strMin,strMax;
  // intALength holds length of strA
  // intBLength holds length of strB
  int intALength, intBLength;
  // intSetValue is the value to be set in combobox in case of wrong input
  int intSetValue;
  // buf is used in int to ascii conversion
  char strBuf[10];
  // integer value of contents in combobox
  int intValue;
  // boolean set when distinguishing digits from non-digits
  bool blDigit=true;

  // min and max in string format
  strMin.Format("%.f",m_dbMin);
  strMax.Format("%.f",m_dbMax);

  //Get text in edit box
  GetWindowText(strA);
  intValue = atoi(strA); //returns 0 if strA is "0" or a character
  intALength = strA.GetLength();

  // gets the length of the value
  _itoa(intValue,strBuf,10);
  strB = strBuf;
  intBLength = strB.GetLength();

  // string contents of combobox can be e.g. 123abc. When converting this to
  // int, value is 123. this check determines if any characters are present.
  if(intBLength != intALength)
        blDigit=false;
```

```
  // If the value is 0, strA can be both the number 0 and a character
  if(intValue==0)
  {
        // strA is "0" or "-" implies that the input is legal
        if(strA=="0" || strA=="-")
              blDigit=true;
        // Else, when strA is non of the above, it implies that the input
        // is an illegal character
        else
              blDigit=false;
  }



  //If outside legal values, output error message, and reset combo box
  if(intALength == 0 || intValue<m_dbMin || intValue>m_dbMax || !blDigit)
  {
        if((intValue < m_dbMin || intValue > m_dbMax) && blDigit)
              AfxMessageBox("Verdi utenfor Gyldig område.\nGyldig område er mellom "
                                + strMin + " og " + strMax);
        if(intBLength != intALength)
              intSetValue = intValue;
        else if(m_dbMin<0)
              intSetValue = (int)(m_dbMax + m_dbMin);
        else
              intSetValue = (int)m_dbMin;
        _itoa(intSetValue,strBuf,10);
        SetWindowText(strBuf);
        SetFocus();
  }

  return hbr;
}

// Input:      [const double&] mn, [double&] mx
// Sets the limits of the combobox
void CIntLimitComboBox::SetLimit(const double& dblMin, double& dblMax)
{
  m_dbMin = dblMin;
  m_dbMax = dblMax;
}

// Input:      [const int&] mn, [int&] mx
// Sets the limits of the combobox
void CIntLimitComboBox::SetLimit(const int& intMin, int& intMax)
{
  m_dbMin = intMin;
  m_dbMax = intMax;
}
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator                **/
/** Class:            CComputer                                                 **/
/** Filename:  Computer.h                                                  **/
/** Description:      Declaration file                                          **/
/** Author/year:      Alexander Iversen / 2002                                  **/
/*****************************************************************************/

#if !defined( COMPUTER_H )
#define COMPUTER_H


#include <nidaq.h>
#include <math.h>
#include <time.h>
#include "DACOutput.h"


#define PI                  3.14159265358979
#define PI2                 6.28318530717958
#define MAX                 32767
#define MIN                 -32768

#define FREQUENCY     0
#define AMPLITUDE   1
#define AMPLNOISE   2
#define AMPLRESET   3
#define AMPLCOUNT   4
#define PHASESHFT   5
#define PHASNOISE     6
#define PHASRESET   7
#define PHASCOUNT   8

class CComputer
{

private: // Variables

  unsigned int m_uintSeed;   // Random number seed
  double m_dblRA;                         // Variable (A)
  double m_dblRB;                         // Variable (B)
  double m_dblRC;                         // Variable (C)

  double m_dblConst;               // A frequently used constant
  double m_dblConstN;              // The negative equivalent of m_dblConst


private: // Operations

  // Generates random number between zero and one
  float RandomUniformZtoO();

  // Calculates I component at time dblTime
  double ISignal(float &fltF, float &fltA, float &fltP, double &dblTime);

  // Calculates Q component at time dblTime
  double QSignal(float &fltF, float &fltA, float &fltP, double &dblTime);


public: // Operations

  // Constructor
  CComputer();

  // Random Number generator
  double PRNGenerator(bool &blGaussDist, float &fltMean,
                                  float &fltVariance, float &fltPct);

  // Overloaded functions
  // converts dBm to volt
  void dBmToVolt(int &intDbm, CString &strVolt, const bool &blVEff);
  void dBmToVolt(int &intDbm, double &dblVolt, const bool &blVEff);

  // Convert degrees to radians
  float DegToRad(const float &fltDeg);

  // Calculates a constant used in the signal function
  double ComputeConstant(double &dblSum, int &intSig, const int &intEff_out);

  // Calculates the sum of the signals
  void FillIQArray(short *psrtIQArray,
                           float (*pfltDopplerArray)[100],
                           float &fltExtRef,
```

```
                                bool &blDistribution,
                                float &fltMean,
                                float &fltVariance,
                                int &intNumberOfSignals,
                                int &intClipping,
                                float &fltPeak);
};

#endif //!defined ( COMPUTER_H )
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator                **/
/** Class:            Ccomputer                                                 **/
/** Filename:  Computer.cpp                                                  **/
/** Description:      Implementation file                                       **/
/** Author/year:      Alexander Iversen / 2002                                  **/
/*****************************************************************************/

#include "StdAfx.h"
#include "Computer.h"

// Constructor
CComputer::CComputer()
{
  // Seed is different for each occurance
  // of the CComputer object
  m_uintSeed = (unsigned)time(NULL);
}

// Input: [int&] intDbm, [CString&] strVolt, [const bool&] blVEff
// Saves volt value text corresponding to dBm
void CComputer::dBmToVolt(int& intDbm, CString& strVolt, const bool &blVEff)
{
  if(blVEff) // If V eff
        strVolt.Format("%.3f",sqrt(pow(10.0,intDbm/10.0)/10.0)/sqrt(2));
  else    // If V amp
        strVolt.Format("%.3f",sqrt(pow(10.0,intDbm/10.0)/10.0));
}

// Input: [int&] intDbm, [double&] dblVolt, [const bool&] blVEff
// Saves volt value corresponding to dBm
void CComputer::dBmToVolt(int& intDbm, double& dblVolt, const bool &blVEff)
{
  if(blVEff) // If V eff
        dblVolt = sqrt(pow(10.0,intDbm/10.0)/10.0)/sqrt(2.0);
  else // If V amp
        dblVolt = sqrt(pow(10.0,intDbm/10.0)/10.0);
}

// Input:      [int&] fltDeg
// Output:     [double] corresponding radian value
float CComputer::DegToRad(const float &fltDeg)
{
  return (float)(fltDeg*(PI/180.0));
}


// Input:      [float&] fltF, [float&] fltA, [float&] fltP, [double&] dblTime
// Output:     [double] volt value. This function is called to calculate
//                    the volt value of the I signal at the current time
//                    The function is inline
double CComputer::ISignal(float &fltF, float &fltA, float &fltP, double &dblTime)
  {
        if (fltF > 0)
              return m_dblConst * fltA * cos(fltF * dblTime + fltP);
        else
              return m_dblConst * fltA * cos(-fltF * dblTime + fltP);
  }

// Input:      [float&] fltF, [float&] fltA, [float&] fltP, [double&] dblTime
// Output:     [double] volt value. This function is called to calculate
//                    the volt value of the Q signal at the current time
//                    The function is inline
double CComputer::QSignal(float &fltF, float &fltA, float &fltP, double &dblTime)
  {
        if (fltF > 0)
              return m_dblConst * fltA * sin(fltF * dblTime + fltP);
        else
              return m_dblConstN * fltA * sin(-fltF * dblTime + fltP);
  }



// Returns random number between zero and one
float CComputer::RandomUniformZtoO()
{
  m_uintSeed = m_uintSeed * 1103515245 + 12345;
  return (float)((unsigned int)(m_uintSeed / 65536) % 10000)/10000;
}


// Input:      [bool&] blGaussDist, [float&] fltMean, [float&] fltVariance,
```

```
//                    [float&] fltPct
// Return:     [double] pseudo random number within the range -pct/100 and +pct/100,
//                    uniformly or normally distributed according to the gauss_dist value.
double CComputer::PRNGenerator(bool &blGaussDist, float &fltMean,
                                                  float &fltVariance, float &fltPct)
{
  switch(blGaussDist)
  {
        case true: // Gaussian (normal) distribution
                do
                {
                        m_dblRA = 2.0 * RandomUniformZtoO() - 1.0;
                        m_dblRB = 2.0 * RandomUniformZtoO() - 1.0;
                        m_dblRC = m_dblRA * m_dblRA + m_dblRB * m_dblRB;
                }while(m_dblRC >= 1.0);

                m_dblRC = m_dblRA * sqrt((-2.0*log(m_dblRC))/m_dblRC);
                m_dblRC = m_dblRC * fltVariance + fltMean;

                // If normally distributed random number falls within the
                // legal range -pct to +pct
                if(m_dblRC > -fltPct && m_dblRC < fltPct)
                {
                        m_dblRC /= 100.0;
                        break;
                }

                // If not, compute uniformly distributed random number

        case false: // Uniform distribution
                        m_uintSeed = m_uintSeed * 1103515245 + 12345;
                        m_dblRC = ((unsigned)(m_uintSeed / 65536) % (int)(2*fltPct+1));
                        m_dblRC = (m_dblRC - fltPct)/100.0;
                        break;
  }
  return m_dblRC;
}

// Input:      [double&] dblSum, [int&] intSig, [const int&] intEffOut
// Return:     [double] m_dblConst.
// This function computes a value, which is a constant value used in
// the ISignal and QSignal functions. This is to avoid
// recalculating the value each time ISignal or QSignal are called
double CComputer::ComputeConstant(double &dblSum, int &intSig, const int &intEffOut)
{
  m_dblConst = sqrt(intSig * pow(10.0,intEffOut/10.0) / 10.0)/dblSum;
  m_dblConstN = -m_dblConst;
  return m_dblConst;
}

// Input:      [short*] psrtIQArray, [float*] pfltDopplerArray, [float&] fltExtRef,
//                    [bool&] blGaussDist, [float&] fltMean, [float&] fltVariance,
//                    [int&] intNumberOfSignals, [int&] intClipping, [float&] fltPeak
// This function calculates the sum of the signals stored in the array pointed to by
// *psrtDopplerArray, and saves it in the array pointed to by *psrtIQArray.
void CComputer::FillIQArray(short *psrtIQArray,  // pointer to resultant IQ array
                                  float (*pfltDopplerArray)[100],  // pointer to array .
                                  float &fltExtRef,                // external reference
                                  bool &blGaussDist,         // gaussian or uniform
distribution
                                  float &fltMean,                  // mean of gaussian random
number
                                  float &fltVariance,        // variance of normal random
number
                                  int &intNumberOfSignals,// number of signals to be added
                                  int &intClipping,   // returning number of clipped samples
                                  float &fltPeak)     // returning highest theoretical amp.
peak
{

  // CDACOutput object
  CDACOutput dacOut;
  // variable to reflect the time in the doppler array
  double dblTime = 0.0;
  // total sum of I signals at each time interval
  double dblSumI;
  // total sum of Q signals at each time interval
  double dblSumQ;
  // temporary I and Q signal variables used when adding amplitude noise
  double dblTmpI,dblTmpQ;
  // the random noise factor is multiplied with the amplitude
  double dblRandomFactor;
  // time variable is incremented with this inc value (time respolution)
```

```
const double dblIncrement = 1.0/dacOut.GetUpdateRate();
// factor used to convert total signal to 16bit value
const double dblFactor = 32768.0/fltExtRef;
// increment variables
int intI,intJ,intK,intCounter=0;


// Stepping through every second array position
// (I and Q signals are interleaved in the array)
for(intJ=0;intJ<BUF;intJ+=2)
{
       dblTime += dblIncrement; // time is incremented according to the update rate
       intK = intJ+1;     // j is for the I signal, k is for the Q signal
       dblSumI = 0.0;       // reset sumI and sumQ for every step through buffer
       dblSumQ = 0.0;

       // For every array position, step through
       // the number of signals to be added
       for(intI=0;intI<intNumberOfSignals;intI++)
       {
              // If the signal has any phase noise
              if(pfltDopplerArray[PHASNOISE][intI] > 0)
              {
                     // If phase noise is not ready to be added,
                     // increment the counter
                     if(pfltDopplerArray[PHASCOUNT][intI] <
                            pfltDopplerArray[PHASRESET][intI])
                            pfltDopplerArray[PHASCOUNT][intI]++;

                     // If phasenoise ready to be added, reset counter
                     // and add random phase
                     else
                     {
                            pfltDopplerArray[PHASCOUNT][intI] = 0;
                            pfltDopplerArray[PHASESHFT][intI] =
                                   (float)(PI2 * RandomUniformZtoO());
                     }
              }

              // If the signal has no amplitude noise, compute new signal
              // at current time, and add it to sumI and sumQ
              if(pfltDopplerArray[AMPLNOISE][intI] <= 0)
              {
                     dblSumI += ISignal(pfltDopplerArray[FREQUENCY][intI],
                            pfltDopplerArray[AMPLITUDE][intI],
                            pfltDopplerArray[PHASESHFT][intI],
                            dblTime);
                     dblSumQ += QSignal(pfltDopplerArray[FREQUENCY][intI],
                            pfltDopplerArray[AMPLITUDE][intI],
                            pfltDopplerArray[PHASESHFT][intI],
                            dblTime);
              }

              // If the signal has amplitude noise
              else
              {
                     // If amplitude noise is not ready to be added,
                     // increment the counter, add signals
                     if(pfltDopplerArray[AMPLCOUNT][intI] <
                            pfltDopplerArray[AMPLRESET][intI])
                     {
                            pfltDopplerArray[AMPLCOUNT][intI]++;
                            dblSumI += ISignal(pfltDopplerArray[FREQUENCY][intI],
                                   pfltDopplerArray[AMPLITUDE][intI],
                                   pfltDopplerArray[PHASESHFT][intI],
                                   dblTime);
                            dblSumQ += QSignal(pfltDopplerArray[FREQUENCY][intI],
                                   pfltDopplerArray[AMPLITUDE][intI],
                                   pfltDopplerArray[PHASESHFT][intI],
                                   dblTime);
                     }


                     // If amplitude noise is ready to be added, reset counter
                     // and add random noise to sumI and sumQ
                     else
                     {
                            pfltDopplerArray[AMPLCOUNT][intI] = 0;
                            dblRandomFactor = PRNGenerator(blGaussDist,fltMean,fltVariance,
                                   pfltDopplerArray[AMPLNOISE][intI]);

                            dblTmpI = ISignal(pfltDopplerArray[FREQUENCY][intI],
                            pfltDopplerArray[AMPLITUDE][intI],
```

```
                                pfltDopplerArray[PHASESHFT][intI],
                                dblTime);
                        dblTmpQ = QSignal(pfltDopplerArray[FREQUENCY][intI],
                                pfltDopplerArray[AMPLITUDE][intI],
                                pfltDopplerArray[PHASESHFT][intI],
                                dblTime);

                        dblSumI = dblSumI + dblTmpI + dblTmpI * dblRandomFactor;
                        dblSumQ = dblSumQ + dblTmpQ + dblTmpQ * dblRandomFactor;
                    }
                }


            }

            // peak will hold the largest amplitude
            if (dblSumI > fltPeak)
                fltPeak = (float)dblSumI;
            if (dblSumQ > fltPeak)
                fltPeak = (float)dblSumQ;


            // rescale I and Q to values within
            // 16 bit legal range (0 - 65536)
            dblSumI *= dblFactor;
            dblSumQ *= dblFactor;


            // Signals outside legal range will be clipped
            // Integer clipping holds the number of clipped
            // values to calculate the clipping percentage

            // I and Q signals are interleaved in the doppler array:
            // [I1,Q1,I2,Q2,I3,Q3,...In,Qn]

            if(dblSumI > MAX)
            {
                psrtIQArray[intJ] = MAX;
                intClipping++;
            }
            else if(dblSumI < MIN)
            {
                psrtIQArray[intJ] = MIN;
                intClipping++;
            }
            else
                psrtIQArray[intJ] = (short)(dblSumI + 0.5);

            if(dblSumQ > MAX)
            {
                psrtIQArray[intK] = MAX;
                intClipping++;
            }
            else if(dblSumQ < MIN)
            {
                psrtIQArray[intK] = MIN;
                intClipping++;
            }
            else
                psrtIQArray[intK] = (short)(dblSumQ + 0.5);
        }
    }
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator              **/
/** Class:            CDACOutput                                              **/
/** Filename:         DACOutput.h                                             **/
/** Description:      Declaration file                                        **/
/** Author/year:      Alexander Iversen / 2002                                **/
/*****************************************************************************/

#if !defined( DACOUTPUT_H )
#define DACOUTPUT_H

#include "nidaq.h"

#define BUF           20000000


class CDACOutput
{
private: // NI6731 specific variables

  // the special data types are specified in nidaq.h
  i16 m_pi16ChanVect[2];     // channel vector
  i16 m_i16Device;           // device number
  i16 m_i16Group;                  // group number
    i16 m_i16NumChans;             // number of channels
    i16 m_i16Chan;                 // channel number
  u32 m_u32Count;                  // buffer size
    u32 m_u32Iterations;     // number of iterations 0=infinitely
  i16 m_i16Polarity;         // analog output channel polarity
  i16 m_i16Ref;              // reference source internal or external
  f64 m_f64RefVolt;          // voltage reference value
  f64 m_f64Voltage;          // voltage
  i16 m_i16UpdateMode;// when to update the DACs
  f64 m_f64UpdateRate;// update rate, max is 1M



public: // Operations

  // Constructor
  CDACOutput();

  // Outputs a continous waveform on channel 0 and 1
  void OutputWaveform(i16 * pi16IQArray);

  // Outputs a voltage on a channel
  void OutputVoltage(f32 &f32Voltage, const i16 &i16Channel);

  // Returns the update rate
  double GetUpdateRate()
  {
        return m_f64UpdateRate;
  }
};

#endif // !defined( DACOUTPUT_H )
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator               **/
/** Class:           CDACOutput                                               **/
/** Filename:        DACOutput.cpp                                            **/
/** Description:     Implementation file                                      **/
/** Author/year:     Alexander Iversen / 2002                                 **/
/*****************************************************************************/

#include "StdAfx.h"
#include "DACOutput.h"

// Constructor
CDACOutput::CDACOutput()
{
  m_pi16ChanVect[0] = 0;           // DAC0OUT
  m_pi16ChanVect[1] = 1;           // DAC1OUT

  m_i16Device = 1;                 // Device number (standard)
  m_i16Group = 1;                        // Group number (standard)
    m_i16NumChans = 2;                   // Number of channels (size of m_pi16ChanVect)
  m_u32Count = BUF;                // Size of waveform array
    m_u32Iterations = 0;           // Number of iterations of waveform (0 = infinite)
    m_f64UpdateRate = 1000000;     // Updaterate

  m_i16Polarity = 0;               // Polarity (0 = bipolar, 1 = unipolar)
  m_f64RefVolt = 10.0;       // Reference voltage
  m_i16UpdateMode = 0;       // when to update analog channel
                                          // (0 = when written to)

}

// Input:      [f32&] f32Voltage, [i16&] i16Channel
// This function outputs the voltage on the specific channel on the NI-6731 card
// The channel uses internal reference (10.0 V)
void CDACOutput::OutputVoltage(f32 &f32Voltage, const i16 &i16Channel)
{
  m_i16Chan = i16Channel; // channel number
  m_f64Voltage = (f64)f32Voltage; // output voltage
  m_i16Ref = 0;// Internal reference

  // Configure channel to internal reference (always 10V) and
  // bipolar mode (-10V to +10V)
  AO_Configure(m_i16Device, m_i16Chan, m_i16Polarity,
                       m_i16Ref, m_f64RefVolt, m_i16UpdateMode);

  // Terminate any waveform
  WFM_Group_Control(m_i16Device,m_i16Group, 0);

  // Output voltage on channel
  AO_VWrite(m_i16Device, m_i16Chan, m_f64Voltage);
}

// Input:      [i16*] pi16IQBuffer
// This function outputs the waveform in volt_buffer. The buffer contains
// two waveforms which are interleaved, and the NI-6731 multiplexes the content
// on channels 0 (I) and 1 (Q). The channels use external reference
void CDACOutput::OutputWaveform(i16 * pi16IQArray)
{
  m_i16Ref = 1;// External reference

  m_i16Chan = 0; // Channel DAC0OUT
  // Configure channel 0 to external reference and bipolar mode
  AO_Configure(m_i16Device, m_i16Chan, m_i16Polarity,
                       m_i16Ref, m_f64RefVolt, m_i16UpdateMode);
  m_i16Chan = 1;      // Channel DAC1OUT

  // Configure channel 1 to external reference and bipolar mode
  AO_Configure(m_i16Device, m_i16Chan, m_i16Polarity,
                       m_i16Ref, m_f64RefVolt, m_i16UpdateMode);

  // Output waveform pointed to by pi16IQArray on channels in m_pi16ChanVect,
  // with selected updaterate.
  WFM_Op(m_i16Device, m_i16NumChans, m_pi16ChanVect, pi16IQArray, m_u32Count,
            m_u32Iterations, m_f64UpdateRate);
}
```

```
/*****************************************************************************/
/** Application:      Multispectral Quadrature Doppler Generator               **/
/** Class:            CFileHandler                                             **/
/** Filename:         FileHandler.h                                            **/
/** Description:      Declaration file                                         **/
/** Author/year:      Alexander Iversen / 2002                                 **/
/*****************************************************************************/

#if !defined ( FILEHANDLER_H )
#define FILEHANDLER_H

#include "Computer.h"


class CFileHandler
{
private: // Member variables
  CString m_strIllegalCharacters;

public: // Operations

  // Constructor
  CFileHandler();

  // Opens a standard file dialog
  void OpenFileDialog(CWnd *pwndParent,CString &strPathname,
                                  CString &strFilename, CString &strResult);

  // Organizes string contents into an array
  void FillDopplerArray(CString &strFileContent, float (*pfltDopplerArray)[100]);

  // Returns the value of m_strIllegalCharacters
  CString GetIllegalCharacters()
  {
        return m_strIllegalCharacters;
  }
};

#endif //!defined( FILEHANDLER_H )
```

```
/*****************************************************************************/
/** Application:     Multispectral Quadrature Doppler Generator               **/
/** Class:           CFileHandler                                             **/
/** Filename:        FileHandler.cpp                                          **/
/** Description:     Implementation file                                      **/
/** Author/year:     Alexander Iversen / 2002                                 **/
/*****************************************************************************/

#include "StdAfx.h"
#include "FileHandler.h"


// Constructor
CFileHandler::CFileHandler()
{
  m_strIllegalCharacters = "";
}

// Input:      [CWnd*] pwndParent, [CString&] strPathname, [CString&] strFilename,
//                     [CString&] strResult
// The function opens a standard file dialog window. The selected filename and
// pathname is returned. So is the contents of the file in the CString pointed
// referenced to by strResult.
void CFileHandler::OpenFileDialog(CWnd *pwndParent, CString &strPathname,
                                                    CString &strFilename, CString
&strResult)
{
  CStdioFile file; // Standard IO file object
  CString strBuffer="";       // buffer used when reading file contents to string
  CFileDialog fileDlg(TRUE, //true=open file / false=save file
                                    NULL, //file extension appended
                                    NULL, //initial file name
                                    OFN_HIDEREADONLY | OFN_FILEMUSTEXIST,//flags
                                    "Modulasjonsfiler
(*.moa;*.mob)|*.moa;*.mob||",//filter
                                    pwndParent); //parent window

  fileDlg.m_ofn.lpstrTitle = "Velg modulasjonsfil"; // Dialog title

  //Write filename in edit box if OK pressed
  if ( fileDlg.DoModal() == IDOK)
  {
        strPathname = fileDlg.GetPathName();
        strFilename = fileDlg.GetFileName();

        // Read contents of file to [CString] m_strFContents
        file.Open(strPathname,CStdioFile::modeRead,NULL);
        strResult = "";
        while(file.ReadString(strBuffer)!=0)
        {
              strResult += '\n'+strBuffer;
        }
        file.Close();
  }
}

// Input:      [CString&] strFileContents, [float*] pfltDopplerArray
// strFileContents should be of the format:
// (f1,a1,an1,p1,pn1)(f2,a2,an2,p2,pn2)...(f100,a100,an100,p100,pn100)
// where f=frequency, a=amplitude,an=amplitude noise, p=phase, pn=phase noise
// This function assembles the values in the array pointed to by pfltDopplerArray
void CFileHandler::FillDopplerArray(CString &strFileContent,
                                                        float (*pfltDopplerArray)[100])
{
  // counter variables
  int intI,intJ,intCounter=0;
  // type value represent next expected number, initially not defined (-1)
  int intType = -1;
  // each character in file is temporarily stored in tchar
  TCHAR strCharacter;
  // each occurence of a collection of digits are stored in buf
  CString strBuffer = "";
  // variable to store all characters not recognized as valid
  m_strIllegalCharacters = "";
  // CComputer object
  CComputer compObj;

  //Clear array
  for(intI = 0; intI < 9; intI++)
        for(intJ = 0; intJ < 100; intJ++)
              pfltDopplerArray[intI][intJ] = 0;

  // For each character in str
```

```
  for(intI=0;intI<strFileContent.GetLength();intI++)
  {
        strCharacter = strFileContent.GetAt(intI);

        switch(strCharacter)
        {

        // Assemble all characters creating an integer or floating
        // point number.
        case '0':case '1':case '2':case '3':case '4':case '5':
        case '6':case '7':case '8':case '9':case '.':case '-':
               strBuffer += strCharacter;
               break;

        // If '(', next expected number is frequency
        case '(':
                     intType = FREQUENCY;
               break;
        // On ',' check what type is expected
        case ',':
               switch(intType)
               {

               // If FREQUENCY, save buf as frequency,
               // next expected number is amplitude
               case FREQUENCY:
                     pfltDopplerArray[intType][intCounter] =
                            (float)(atof(strBuffer)*PI2);
                     intType = AMPLITUDE;
                     strBuffer = "";
                     break;

               // If AMPLITUDE, save buf as amplitude,
               // next expected number is amplitude noise
               case AMPLITUDE:
                     pfltDopplerArray[intType][intCounter] = (float)atof(strBuffer);
                     intType = AMPLNOISE;
                     strBuffer = "";
                     break;

               // If AMPLNOISE, save buf as amplitude noise,
               // next expected number is phase
               case AMPLNOISE:
                     pfltDopplerArray[intType][intCounter] = (float)atof(strBuffer);
                     intType = PHASESHFT;
                     strBuffer = "";
                     break;

               // If PHASESHFT, save buf as phase,
               // next expected number is phase noise
               case PHASESHFT:
                     pfltDopplerArray[intType][intCounter] =
                            compObj.DegToRad((float)atof(strBuffer));
                     intType = PHASNOISE;
                     strBuffer = "";
                     break;
               }
               break;

        // If ')', save buf as phase noise
        case ')':
               pfltDopplerArray[intType][intCounter] = (float)atof(strBuffer);
               strBuffer = "";
               intCounter++;
               intType = -1;
               break;

        default:
               m_strIllegalCharacters += strBuffer;
               break;
        }
  }
}
```

FORDELINGSLISTE

FFIE **Dato:** 13. september 2002

RAPPORTTYPE (KRYSS AV)	RAPPORT NR.	REFERANSE	RAPPORTENS DATO
X RAPP ☐ NOTAT ☐ RR	2002/03854	FFIE/838/113	13. september 2002
RAPPORTENS BESKYTTELSESGRAD		ANTALL EKS UTSTEDT	ANTALL SIDER
UGRADERT		23	85
RAPPORTENS TITTEL		FORFATTER(E)	
MULTISPEKTRAL KVADRATUR DOPPLERGENERATOR		IVERSEN Alexander	
FORDELING GODKJENT AV FORSKNINGSSJEF		FORDELING GODKJENT AV AVDELINGSSJEF:	
Torleiv Maseng		Johnny Bardal	

EKSTERN FORDELING

ANTALL	EKS NR	TIL
1		Alexander Iversen 5 Roseneath Street 1F1 Marchmont EH9-1JH Edinburgh Skottland

INTERN FORDELING

ANTALL	EKS NR	TIL
14		FFI-Bibl
1		Adm direktør/stabssjef
1		FFIE
1		FFISYS
1		FFIBM
1		FFIN
3		Restopplag til Bibliotek
		ELEKTRONISK FORDELING:
		Torleiv Maseng, FFIE
		Arne Petter Bartholsen, FFIE
		Tor-Odd Høydal, FFIE
		Øyvind Thingsrud, FFIE
		Tor Holmboe, FFIE
		Trond Hellum, FFIE
		Stein Kristoffersen, FFIE
		Haakon Vinge, FFIE
		Harald Hovland, FFIE
		Jonas Moen, FFIE
		Eivind Engebretsen, FFIE
		Hans Øhra, FFIE
		Reidar Haugen, FFIE
		Tore Smestad, FFIE
		Svein-Erik Hamran, FFIE
		Halvor Bjordal, FFIE
		Ivar Tansem, FFIE
		Rune Gundersen, FFIE
		Steinar Johnsrud, FFIE
		Per Sørnes, FFIE
		FFI-veven

FFI-K1 Retningslinjer for fordeling og forsendelse er gitt i Oraklet, Bind I, Bestemmelser om publikasjoner for Forsvarets forskningsinstitutt, pkt 2 og 5. Benytt ny side om nødvendig.