Zajemanje in generiranje analognih signalov

Okolje Labviev

Zgledi programov za zajemanje in generiranje ter skelet programa za sprotno obdelavo podatkov

November 2012
in dopolnjeno
Januar 2026

Dušan Ponikvar

Fakulteta za Matematiko in Fiziko, Oddelek za fiziko
Jadranska 19, Ljubljana

Zajemanje analognih signalov z vmesnikom NI6211 v okolju Labview

Računalniški vmesnik NI6211 (National Instruments) med drugim vsebuje tudi osemkanalni (šestnajstkanalni) analogno-digitalni pretvornik in vsa potrebna logična vezja za periodično zajemanje signala. V vezjih vmesnika in gonilnikih osebnega računalnika so vzpostavljeni mehanizmi za sprotno shranjevanje zajetih vrednosti v spomin vmesnika in prenašanje paketov izmerkov iz spomina vmesnika v spomin osebnega računalnika ter naprej v uporabniške programe, ki tečejo na osebnem računalniku. Ti mehanizmi potrebujejo za delovanje nekaj parametrov, ki jih posreduje uporabnik preko klicev programskih modulov iz priloženih knjižnic.

- **Posamično merjenje velikosti signala**
  Pri tem načinu ob ukazu vmesnik pomeri trenutno vrednost vhodnega signala in rezultat sporoči programu, ki teče na osebnem računalniku.
- **Enkratno vzorčenje signala**
  Pri enkratnem vzorčenju signala vmesnik periodično meri vrednost vhodnega signala. Časovni interval med sosednjima izmerkoma in skupno število izmerkov v nizu v naprej določi uporabnik. Ko vmesnik opravi nalogo, posreduje celoten niz izmerkov v program, ki teče na osebnem računalniku in preneha meriti.
- **Neprekinjeno vzorčenje signala**
  Pri neprekinjenem vzorčenju signala vmesnik periodično meri vrednost vhodnega signala. Časovni interval med sosednjima izmerkoma v naprej določi uporabnik. Ko se v spominu vmesnika nabere zadostno, v naprej izbrano, število izmerkov, vmesnik nabrani niz posreduje v program, ki teče na osebnem računalniku. Med prenašanjem niza izmerkov še naprej z enako periodo meri vhodni signal in izmerke sproti shranjuje v svoj pomnilnik tako, da se nič izmerkov ne izgubi. Vmesnik preneha zajemati vhodni signal takrat, ko uporabnik pošlje vmesniku ustrezen ukaz.

Pri obeh načinih zajemanja mora uporabniški program opraviti pet korakov:

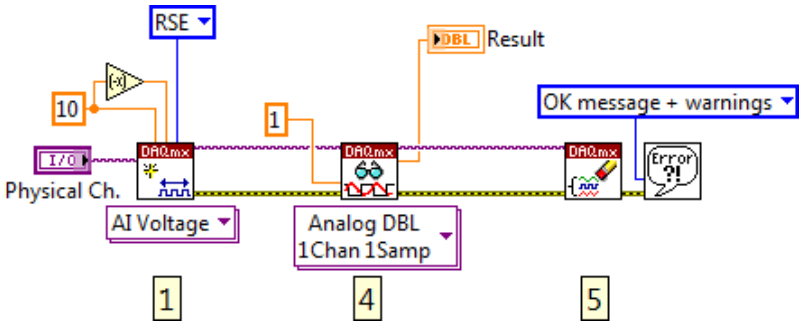
1. Vzpostavitev programske povezave med vmesnikovim gonilnikom in uporabniškim programom ter posredovanje parametrov delovanja za analogno-digitalni pretvornik
2. Posredovanje parametrov delovanja enoti, ki skrbi za časovne intervale med izmerki (ta korak lahko v nekaterih primerih opustimo)
3. Start zajemanja (ta korak lahko v nekaterih primerih opustimo)
4. Branje (niza) izmerkov iz vmesnika v uporabniški program
5. Prekinitev programske povezave med vmesnikovim gonilnikom in uporabniškim programom

Vsakemu koraku ustreza en klic v knjižnico, vsakokrat s pravimi parametri.

Posamično merjenje velikosti signala

Na sliki 1 je podan zgled programa za posamično merjenje velikosti signala. Vmesnik opravi eno samo meritev, rezultat izpiše v okence »Result«. Prvi korak v programu predstavlja levi blok, ki vzpostavi povezavo med našim programom in gonilnikom vmesnika.

- Bloku definiramo funkcijo: z njim želimo inicializirati analogno zajemanje signala, zato v meniju tik pod blokom izberemo opcijo »AI Voltage« (»Analog Input Voltage«).
- Navedemo oznako NI vmesnika, ki ga bomo uporabili za merjenje. Tega uporabnik navadno izbere v meniju (v zgledu: »drop-down box« z imenom »Physical Channel«), tipično ime vmesnika je »Dev1«, odvisno od števila in zaporedja priključenih vmesnikov. Ime vmesniku dodeli gonilnik ob prvi priključitvi vmesnika na računalnik, lista imen vseh vmesnikov je vidna z orodjem »Measurement & Automation«, ki je del Labview paketa. Imenu uporabniškega vmesnika sledi oznaka kanala, kamor je merjeni signal priključen. Če je signal priključen med sponki »ai0« (»analog input 0«) in »AGND« (»Analog GrouND«) vmesnika, se celotno ime v omenjenem meniju glasi »Dev1/ai0«.
- Navedemo območje merjenja. Za dani vmesnik lahko izbiramo med ±10V, ±5V, ±1V in ±0,2V, za zgled je izbrano območje ±10V. Pri tem je spodnja meja območja izračuna iz zgornje.
- Zadnja je na vrsti razporeditev vhodnih priključkov in referenčna vrednost za merjenje. Izbiramo lahko med načini RSE (»Referenced Signgle Ended«, signal proti AGND), NRSE (»NonReferenced Single Ended«, signal proti skupni točki »AI Sense«) in Differential (signal je pripet med dve vhodni sponki vmesnika, na primer med »ai0« in »ai8«, vendar mora biti potencial vsakega od priključkov v izbranem območju delovanja ADC). V zgledu je izbran način RSE, ki je najbolj običajen.



Slika 1: Program za posamično merjenje velikosti signala

Rezultat izvrševanja tega bloka je inicializiran analogno-digitalni pretvornik in identifikacijska koda enote, ki jo bomo uporabljali pri klicih vseh blokov, povezanih s to inicializacijo. Identifikacijska koda je na razpolago na zgornjem robu prvega bloka. Če je pri izvajanju tega bloka prišlo do težav, je sporočilo o težavah posredovano preko priključka na spodnjem robu tega bloka. Sporočilo posredujemo naslednjemu bloku v verigi, ki se dejansko izvede le, če je prvi blok zaključen brez napake. V nasprotnem primeru blok le posreduje sporočilo o napaki naprej naslednjemu bloku in tako do konca verige, kjer sporočilo prestreže podprogram za izpis sporočila na zaslonu računalnika.

Vmesnik je sedaj pripravljen in zato lahko od njega zahtevamo rezultat meritve. To storimo s klicem naslednjega bloka z oznako 4. Blok za delovanje potrebuje identifikacijsko kodo vmesnika in sporočilo o morebitni napaki, oboje povežemo na njegova priključka na levem robu bloka.

- Blok je namenjen branju različnih vrst podatkov iz vmesnika, zato najprej definiramo količino in format prebranih podatkov z izbiro prave opcije iz menija tik pod blokom. Prenesti želimo rezultat meritve analognega signala, zato izberemo opcijo »Analog« , kar odpre pod-meni. Prenašati želimo rezultat meritve na enem samem kanalu, zato tu izberemo opcijo »Single Channel«, kar odpre pod-pod-meni. Prenesti želimo rezultat ene same meritve, zato tu izberemo opcijo »Single Sample«, kar odpre zadnji pod-meni, v tem izberemo format, ki naj bo realno število v dvojni preciznosti.
- Na rezultat meritve bo treba malo počakati. Z obilo dobre mere smo pripravljeni čakati največ eno sekundo, to vrednost posredujemo bloku preko srednjega priključka na levi strani.

Ko se ta blok izvede dobimo rezultat meritve na srednjem priključku na desni strani modula, ta rezultat posredujemo v okence »Result«. Izhoda iz bloka sta še dva, to sta identifikacijska koda uporabljenega vmesnika in koda morebitne napake, oboje posredujemo zadnjemu bloku v verigi, ki prekine povezavo med našim programom in gonilnikom vmesnika. Če je pri delovanju prišlo do napake, se sporočilo o napaki izpiše na zaslonu.

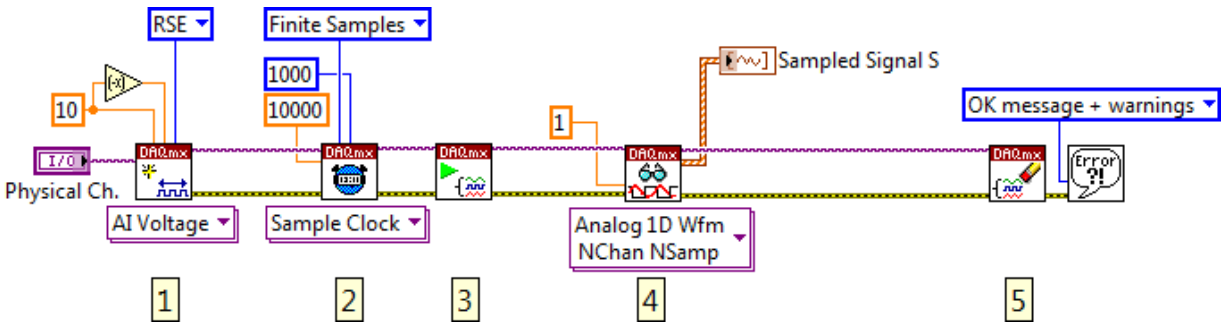
V zgledu smo posamično meritev opravili na enem samem signalu. Kadar želimo sočasno pomeriti več signalov, jih priključimo na več vhodnih sponk vmesnika. Uporabljene vhodne sponke vmesnika navedemo v listi »Physical Channel« kot »Dev1/ai0, Dev1/ai1, Dev1/ai2« če na primer želimo pomeriti tri signale hkrati. Seveda to pomeni, da bomo morali v naslednjem koraku tudi prebrati tri rezultate, kar bloku z oznako 4 dopovemo s pravilno izbiro v meniju pod blokom. Tokrat moramo v pod-meniju izbrati opcijo »Multiple Channel«, ostalo pa ostane enako. Za rezultat tokrat dobimo polje s tremi rezultati, zato bi morali tudi izpis rezultatov na ekran primerno preurediti, zadnji korak v verigi pa ostane nespremenjen.

Enkratno vzorčenje signala

Na sliki 2 je podan zgled programa za enkratno vzorčenje signala. Tokrat naj vmesnik vhodni signal meri v enakovrednih časovnih intervalih. V naprej se odločimo, da želimo opraviti 1000 meritev na enem kanalu analogno-digitalnega pretvornika, med sosednjima meritvama pa naj mine 100µs. Rezultate (niz izmerkov) bomo izrisali v diagram na zaslonu računalnika.

Začnemo enako, kot smo to storili v prejšnjem primeru. Levi blok vzpostavi programsko povezavo med našim programom in gonilnikom vmesnika. Naslednji blok v verigi je zadolžen za izbiro in inicializacijo enote, ki skrbi za časovne intervale med zaporednimi izmerki. Na njegova vhoda sta povezani identifikacijska koda vmesnika iz prvega koraka in koda sporočilo o morebitni napaki.

- Bloku definiramo funkcijo: zajemati želimo priodično z intervali, ki jih določa notranja ura vmesnika, zato v meniju tik pod blokom izberemo opcijo »Sample Clock«. Ker želimo uporabiti notranjo uro, nam je ni treba eksplicitno navajati in zato pustimo tretji priključek od spodnjega roba na levi strani modula prost.
- Zajeti želimo končno število vzorcev, ta način delovanja sporočimo modulu preko priključka na zgornjem robu modula, kjer v meniju izberemo »Finite Samples«. Število izmerkov v sekundi definiramo na drugem priključku na levi strani modula šteto od zgoraj navzdol. V danem primeru želimo zajemati 10000 izmerkov v sekundi. Število izmerkov v nizu definiramo na



Slika 2: Program za enkratno vzorčenje signala

drugem priključku na zgornjem robu gledano od leve proti desni, zato tja povežemo število 1000.

Izhoda iz modula sta spet identifikacijska koda in sporočilo o morebitni napaki, ki ju posredujemo naslednjemu modulu v verigi. Tretji blok v verigi sproži zajemanje, njegova izhoda sta identifikacijska koda in koda sporočila o morebitni napaki, ki ju posredujemo četrtemu bloku v verigi.

Četrti blok je namenjen prenašanju zajetega niza izmerkov v naš program.

- Tokrat želimo prenesti iz vmesnika niz izmerkov iz enega kanala, zato moramo iz menija tik pod blokom izbrati pravo opcijo. Spet gre za meritve analognih signalov, zato izberemo »Analog«, kar odpre pod-meni za izbiro števila kanalov. Izberemo opcijo »Single Channel«, kar odpre naslednji pod-pod-meni. Vmesnik za nas zajema niz izmerkov, ki jih želimo prenesti v enem paketu, zato v tu izberemo opcijo »Multiple Samples«. Ta izbira odpre zadnji pod-pod-pod-meni, kjer se lahko odločimo o formatiranju prebranih podatkov. Ti so lahko prevedeni v enodimenzijsko polje realnih vrednosti (»1D DBL«) ali pa v grozd podatkov oblike »Waveform«, ki vsebuje poleg enodimenzijskega polja realnih vrednosti še časovni interval med izmerki in podatek o času začetka meritve. V zgledu je izbrano slednje.
- Na niz je treba malo počakati: po izdanem ukazu za začetek zajemanja potrebuje vmesnik nekaj časa za dejanski začetek, nato potrebuje vmesnik še čas za zajemanje celotnega niza, podanega s številom izmerkov v nizu in časovnim intervalom med sosednjima izmerkoma. V našem primeru bi zadoščala dobra desetinka sekunde. Za dobro mero dopustimo vmesniku eno sekundo, kar sporočimo modulu preko drugega priključka od spodaj navzgor na levi strani modula. Če vmesnik v eni sekundi ne posreduje vzorcev temu modulu, modul to razume za napako, prekine svoje izvajanje in pošlje kodo napake naslednjemu modulu v verigi.

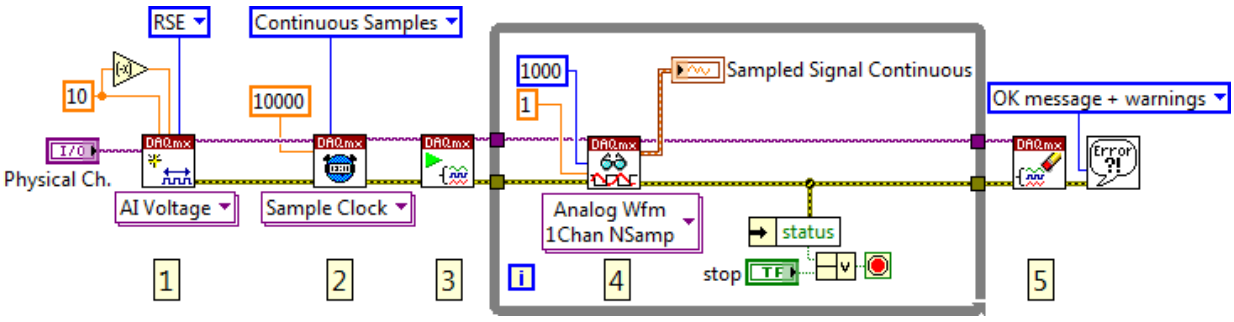
Prebrani rezultati so na razpolago na srednjem priključku modula na desni strani in so formatirani na izbran način. V zgledu jih posredujemo grafu v prikaz. Debela šrafirana črta med modulom in grafom nakazuje, da so podatki formatirani na način »Waveform«, zato graf »Sampled Signal S« na uporabniškem vmesniku lahko abscisno os označi s pravimi enotami.

Po končanem zajemanju je treba še pravilno prekiniti na začetku verige vzpostavljeno povezavo med našim programom in gonilnikom vmesnika. To naredi peti blok v verigi, ki samo sprejema identifikacijsko kodo in sporočilo o morebitni napaki, prekine povezavo in sporočilo o morebitni napaki izpiše na zaslon računalnika.

Neprekinjeno vzorčenje signala

Za neprekinjeno zajemanje signala ponavljamo akcijo iz prejšnjega zgleda tako, da se med ponovitvami nič vzorcev ne izgubi. To dosežemo z istimi bloki, le parametre delovanja je treba ponekod spremeniti, branje niza izmerkov pa ponavljati. Program, ki opravlja to nalogo, je na sliki 3. Razlike so:

- V drugem bloku je treba navesti, da hočemo neprekinjeno zajemanje, zato v meniju nad drugim modulom izberemo opcijo »Continous samples«.
- V drugem bloku ni treba več navesti števila vzorcev v nizu, saj bo vzorčenje stalno. Zato pustimo drugi priključek na zgornjem robu drugega modula prost.
- V četrtem koraku moramo tokrat navesti število izmerkov, ki jih naj blok med posameznim branjem prenese v naš program. To število navedemo na drugem priključku od zgoraj navzdol na levem robu bloka. Blok čaka, dokler ni navedeno število izmerkov na razpolago v spominu (vmesnika ali računalnika), potem jih kot paket posreduje našemu programu.
- Četrti korak lahko poljubno ponavljamo, vsakokrat bomo dobili nov niz izmerkov, ki se na prejšnjega navezuje brez presledka. Zato je četrti blok vstavljen v neskončno zanko, katere izvajanje lahko prekine uporabnik s pritiskom na tipko »STOP«. Izvajanje zanke prekine tudi morebitna napaka, do katere lahko pride pri zajemanju ali prenašanju izmerkov.



Slika 3: Program za kontinuirano vzorčenje signala

Pri neprekinjenem vzorčenju se izmerki nalagajo v vmesni pomnilnik, ki je lahko razdeljen med vmesnik in spomin osebnega računalnika, kjer gonilnik rezervira nekaj prostora po svoji presoji. Če z našim programom sproti prenašamo izmerke iz tega vmesnega pomnilnika v naš program, se vse odvija normalno. Če pa izmerkov ne prenešamo dovolj hitro, se le ti začnejo v vmesnem pomnilniku kopičiti in ga prej ali slej zapolnijo. Ko je vmesni pomnilnik zapolnjen, bi novi izmerki prekrili stare, še ne prenesene izmerke in tako pokvarili meritev. Zato gonilnik v takem primeru prekine zajemanje in javi napako. V našem programu moramo preprečiti tako situacijo, torej moramo prenašati podatke dovolj pogosto.

Pogostost prenašanja podatkov ovira trajanje izvajanja vsega, kar mora računalnik vsakokrat opraviti v zanki ob bloku za prenašanje podatkov. To je lahko obdelava izmerkov (filtriranje, računanje spektra, pisanje na disk, risanje ali pisanje po ekranu, ...) ali zasedenost računalnika z opravili operacijskega sistema. Seveda na dogajanje vpliva tudi zmogljivost računalnika...

Generiranje analognih signalov z vmesnikom NI6211 v okolju Labview

Računalniški vmesnik NI6211 vsebuje dvokanalni digitalno-analogni pretvornik, ki je namenjen spreminjanju digitalnih števil iz programa uporabnika v dejanske napetosti ustrezne velikosti. Poleg pretvornika so na razpolago tudi enote za samodejno generiranje spreminjajoče se napetosti, ki ustreza v polju zapisanem nizu števil in za shranjevenje elementov polja ustrezna količina pomnilnika. Za pravilno delovanje enot skrbi gonilnik, ki enotam posreduje parametre delovanja, rezervira primeren del računalnikovega spomina v podporo spominu v vmesniku ter za pravilen prenos niza števil med letemi. Gonilniku posredujemo parametre preko klicev v knjižnico, ki je del programskega paketa Labview. Na razpolago so tudi knjižnice za druga programska okolja.

Signale generiramo lahko na več načinov:

- **Konstantna napetost**
  Uporabniški program pošlje vmesniku število, vmesnik pa na analognem izhodu generira napetost, ki ustreza poslanemu številu. Generirana napetost je konstantna in se spremeni šele, ko uporabniški program pošlje novo število ali pa se njegovo izvajanje prekine.
- **Signal, ki se periodično ponavlja, njegove vrednosti v eni periodi pa izračunamo v naprej**
  Uporabniški program pošlje vmesniku niz števil in nekaj parametrov za delovanje, vmesnik pa na analognem izhodu zaporedoma generira napetosti, ki ustrezajo poslanim številom. Ko pride z generiranjem do konca niza, začne znova na začetku niza. Časovni interval med generiranjem dveh zaporednih napetosti je eden od parametrov.
- **Poljubni signal, katerega vrednosti sproti določamo**
  Uporabniški program pošlje vmesniku niz števil in nekaj parametrov za delovanje, vmesnik pa na analognem izhodu zaporedoma generira napetosti, ki ustrezajo poslanim številom. Časovni interval med generiranjem dveh zaporednih napetosti je eden od parametrov. Še preden pride vmesnik z generiranjem do konca niza, mora uporabniški program poslati vmesniku nov niz števil, drugače vmesniku zmanjka podatkov in vmesnikov gonilnik prekine generiranje ter javi napako.

Izhodni signal vmesnika odčitamo kot razliko proti skupni sponki »AGND«. V vsakem primeru moramo svoje želje posredovati gonilniku vmesnika v sedmih korakih, ki so navedeni spodaj.

1. Vzpostavitev programske povezave med vmesnikovim gonilnikom in uporabniškim programom ter posredovanje parametrov delovanja za izbrani digitalno-analogni pretvornik
2. Posredovanje parametrov delovanja enoti, ki skrbi za časovne intervale med generiranimi napetostmi
3. Pošiljanje (nizov) digitalnih števil iz uporabniškega programa v vmesnik
4. Start generiranja
5. Periodično preverjanje statusa vmesnika (ta korak lahko tudi izpustimo, če nas status ne zanima)
6. Stop generiranja (ta korak lahko pogosto tudi izpustimo, saj korak 7 samodejno najprej ustavi generiranje signalov in šele nato prekine programsko povezavo)
7. Prekinitev programske povezave med vmesnikovim gonilnikom in uporabniškim programom

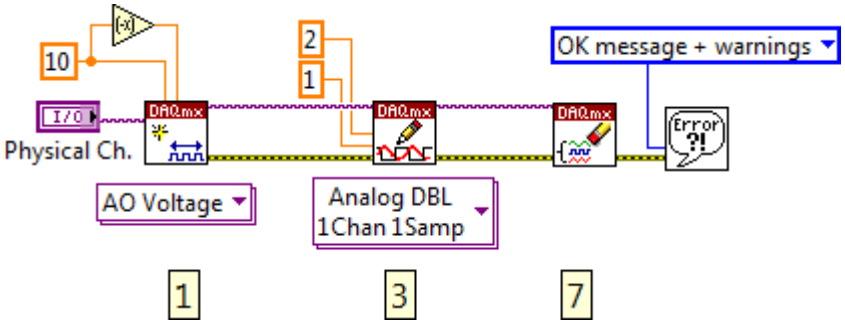
Vsakemu koraku ustreza en klic v knjižnico, vsakokrat s pravimi parametri.

Generiranje konstantne napetosti

Na sliki 4 je podan zgled programa za enkratno generiranje napetosti. Rezultat programa je stalna napetost na analognem izhodu vmesnika, velikost napetosti ustreza številu, ki ga v programu poda uporabnik.

Levi blok na sliki 4 opravi prvi korak v verigi, to je vzpostavljanje povezave med našim programom in vmesnikovim gonilnikom.

- Bloku definiramo funkcijo. Generirati želimo analogno napetost, zato v meniju tik pod blokom izberemo opcijo »AO Voltage« (»Analog Output Voltage«).
- Navedemo oznako vmesnika in ime izhodnega priključka digitalno-analognega pretvornika, s katerega gonilnikom želimo vzpostaviti povezavo. Oboje izbere uporabnik v uporabniškem oknu iz menija (»DropDown Box« z oznako »Physical Ch.«). Tipična oznaka vmesnika je »Dev1«, ime izhodnega priključka digitalno-analognega vmesnika pa »ao0« ali »ao1« (ali oba), zato je obje na primer sestavljeno v »Dev1/ao0«.
- Navedemo območje izhodnih napetosti vmesnika, dovoljena vrednost za vmesnik NI6211 je ±10V. Ker sta meji območja simetrični, negativno mejo izračunamo z negacijo pozitivne meje. Obe meji posredujemo bloku preko priključkov na zgornjem robu.



Slika 4: Program za generiranje konstantne napetosti

Rezultat tega izvajanja tega bloka je identifikacijska koda, ki jo dobimo na priključku bloka na desnem robu zgoraj ter koda morebitne napake, ki jo dobimo na desnem robu spodaj. Obe kodi, podobno kot pri zajemanju signalov, posredujemo naslednji enoti.

S srednjim blokom na sliki 4 pošljemo število, ki naj ga vmesnik pretvori v napetost.

- Bloku definiramo format, v katerem so podani parametri, v našem primeru je to le eno število, zato v meniju tik pod blokom izberemo opcijo »Analog DBL, 1Chan 1Samp«. Torej je število podano kot realno število dvojne natančnosti, blok pa pričakuje eno število, ki ga mora posredovati vmesniku na en kanal.
- Število navedemo na drugi priključni sponki od zgoraj na levem robu, na sliki 4 je to 2.
- Če je vmesnik iz kakršnegakoli razloga zaseden, naj blok zadrži izvajanje do takrat, ko bo vmesnik prost, a največ eno sekundo, kar mu določa priključena vrednost na drugi priključni sponki od spodaj na desnem robu.
- Blok potrebuje identifikacijsko kodo vzpostavljene povezave do vmesnika in kodo morebitne napake iz prvega bloka, obe sta povezani na vhoda bloka na levi strani.

- Če pri pošiljanju števila pride do napake, blok kodo napake posreduje preko svojega izhodnega priključka na desnem spodnjem robu naslednji enoti. Hkrati posreduje tudi identifikacijsko kodo enote, s katero je komuniciral.

Ko je izvajanje tega bloka končano, je na izhodu digitalno-analognega vmesnika že zahtevana napetost. Preostane nam le, da prekinemo povezavo z vmesnikom, kar storimo s tretjim blokom. Tudi ta potrebuje identifikacijsko kodo vmesnika in kodo morebitne napake, kar mu sporočimo preko priključkov na levem robu. Morebitno napako blok posreduje na zaslon računalnika.

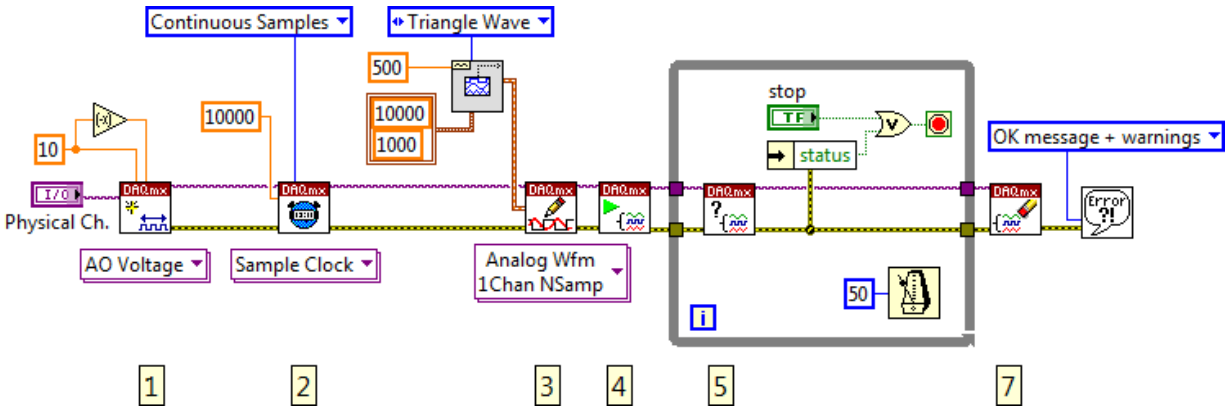
Če želimo generirati več zaporednih napetosti, lahko srednji blok s slike 4 vstavimo v zanko in v zanki vsakokrat sproti pošljemo drugo številko. Tak način generiranja signalov je neprimeren, ker je omejena hitrost komuniciranja z vmesnikom po USB vodilu. Ker je računalnik pogosto zaseden z izvajanjem drugih programov ali nalog operacijskega sistema, so časovni intervali med poslanimi števili neenaki. Kadar želimo generirati periodične signale raje uporabimo katero od spodnjih dveh metod.

Generiranje periodičnega signala s ponavljanjem

Na sliki 5 je podan zgled programa za generiranje periodičnega signala. Ker je signal v naprej znan, lahko v naprej izračunamo zaporedne vrednosti, ki naj jih signal zavzame v eni periodi in ustrezna števila v enem neprekinjenem nizu pošljemo vmesniku, ta jih shrani v lastnem pomnilniku. Definiramo še parametre delovanja vmesnika tako, da generiranje napetosti po poslanih številih periodično ponavlja ter sprožimo delovanje in dobimo periodičen signal po prejšnjem izračunu.

Program začne enako kot v prejšnjem primeru z vzpostavljanjem povezave med našim programom in vmesnikovim gonilnikom. V naslednjem koraku pošljemo parametre delovanja enoti, ki skrbi za časovne intervale med generiranimi vrednostmi.

- Enota časovne intervale definira na podlagi signala ure, tega pa izberemo kot opcijo (»Sample Clock«) v meniju ob spodnjem robu bloka 2.
- V naprej se odločimo, da naj bo časovni interval med sosednjima generiranima vrednostima 100µs kar ustreza frekvenci 10000s^{-1}. Frekvenco (10000) posredujemo bloku preko drugega priključka od zgoraj na levi strani.



Slika 5: Program za generiranje periodičnega signala s ponavljanjem

- Vmesnik naj kontinuirano generira signal, zato mora enota za definiranje časovnih intervalov delovati neprekinjeno. Tako delovanje izberemo kot opcijo (»Continuous Samples«) v meniju nad blokom.

Kode morebitne napake inicializacije posreduje blok naslednjemu bloku v verigi, prav tako tudi identifikacijsko kodo vmesnika. V naslednjem koraku pošljemo vmesniku niz števil, ki jih je treba pretvoriti v napetosti. V zgledu niz izračuna generator trikotnega signala, ki potrebuje parametre. Zdi se smiselno, da ta generator izračuna signal pri isti frekvenci vzorčenja, kot jo uporabljamo za generiranje v vmesniku. Dolžina niza naj bo taka, da ne preobremenjuje USB vodila. V zgledu sta zato v dvojnem okviru ob generatorju izbrani vrednosti 10000 za frekvenco vzorčenja in 1000 za dolžino niza. V meniju nad njim je izbrana trikotna oblika signala, na desni strani pa frekvenca 500Hz.

- Bloku definiramo format, v katerem so mu poslana števila za posredovanje vmesniku. V meniju pod blokom izberemo »Analog«, saj želimo pošiljati števila analognemu delu vmesnika. Števila so namenjena enemu samemu digitalno-analognemu pretvorniku, zato v pod-meniju izberemo »Single Channel«, kar odpre nov pod-pod-meni. Pošiljamo niz števil, zato tu izberemo »Multiple Samples« in v naslednjem, zadnjem pod-pod-pod-meniju še »Waveform«, saj z generatorja trikotnega signala dobimo števila v formatu »Waveform«, torej niz realnih števil, časovni interval med njimi ter trenutek generiranja.
- Izračunani niz iz generatorja trikotnega signala posredujemo tretjemu bloku preko vhoda na levi strani.

Niz števil je sedaj v pomnilniku vmesnika, zato lahko startamo generiranje. Temu služi četrti blok, ki starta generiranje. Vmesnik ponavlja generiranje trikotnega signala iz svojega pomnilnika in posegi iz našega programa niso več potrebni. Program naj obtiči v neskončni zanki dokler se uporabnik ne odloči končati generiranja signala. Zato četrtemu bloku sledi neskončna zanka, ki jo uporabnik lahko prekine s pritiskom na gumb »STOP«. Izvajanje te zanke se zdi smiselno prekiniti tudi takrat, ko pri vmesniku pride do napake, zato je v zanko vstavljen peti blok, ki preverja status vmesnika. Če pride do napake, se izvajanje neskončne zanke prekine. Neskončni zanki (ki zdaj ni več neskončna) sledi še blok sedem za prekinitev povezave med našim programom in gonilnikom vmesnika ter izpis morebitne napake na zaslon.

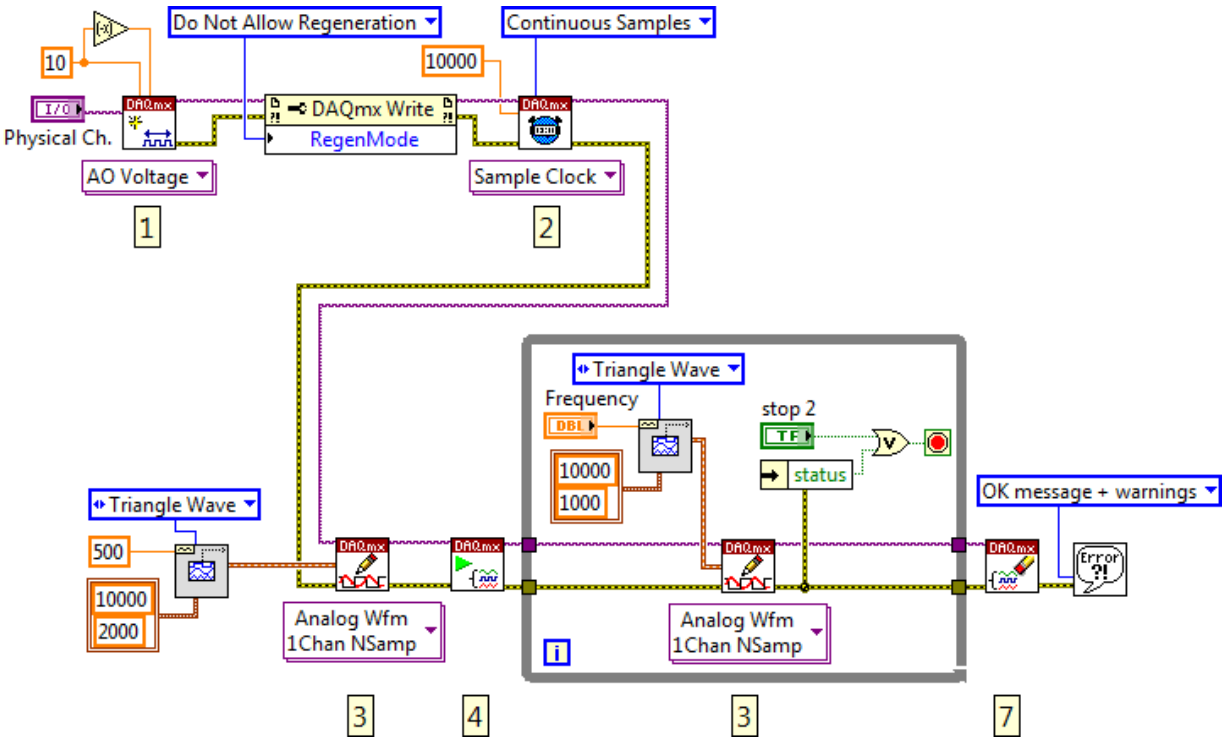
Generiranje poljubnega signala s sprotnim računanjem

Pri obdelavi podatkov večinoma ne vemo v naprej kakšen signal bo treba generirati, saj je ta posledica vhodnih signalov in akcij uporabnika. Takrat vrednosti signala računamo sproti. Ker moramo števila, ki predstavljajo signal, pošiljati vmesniku po paketih, se zdi smiselno, da tudi računanje opravljamo po enako velikih paketih. V naslednjem zgledu na sliki 6 željeni signal izračuna blok generatorja trikotne napetost, ki je vstavljen v neskončno zanko. Za demonstracijo je mogoče frekvenco generiranega signala spreminjati preko uporabniškega vmesnika (»Frequency«).

Program začne enako kot v prejšnjem primeru z vzpostavljanjem programske povezave med našim programom in gonilnikom vmesnika. Rezultat prvega bloka sta identifikacijska koda povezave in koda morebitne napake. Ob vzpostavitvi povezave gonilnik privzame, da želimo ponavljati generiranje signala na podlagi cikličnega branja vrednosti, ki so shranjene v pomnilniku vmesnika, kot je bilo to

narejeno v prejšnjem primeru. Tokrat tega ne želimo, zato moramo na roko spremeniti parametre delovanja vmesnika, kar dosežemo s klicem bloka »DAQmx Write«. Ta blok lahko spreminja parametre mnogo bolj podrobno kot to zmorejo običajni bloki iz knjižnic. Na zaslon ga dobimo tako, da pokažemo na prvi blok, pritisnemo desno tipko na miški in iz menija izberemo opcijo »DAQmx - Data Acquisition Pallete« ter nato iz odprtega pod-menija »Write Node«. Ko ta blok položimo v program, pokažemo z miško na moder napis, pritisnemo levo tipko na miški ter iz odprtega menija izberemo opcijo »RegenMode«. V nadaljevanju priključimo konstantot »Do Not Allow Regeneration« na spodnji priključek bloka. Blok potrebuje za delovanje identifikacijsko kodo vmesnika in morebitno kodo napake, kar povežemo na levem robu bloka, enakovredne kode blok posreduje naprej preko priključkov na desnem robu.

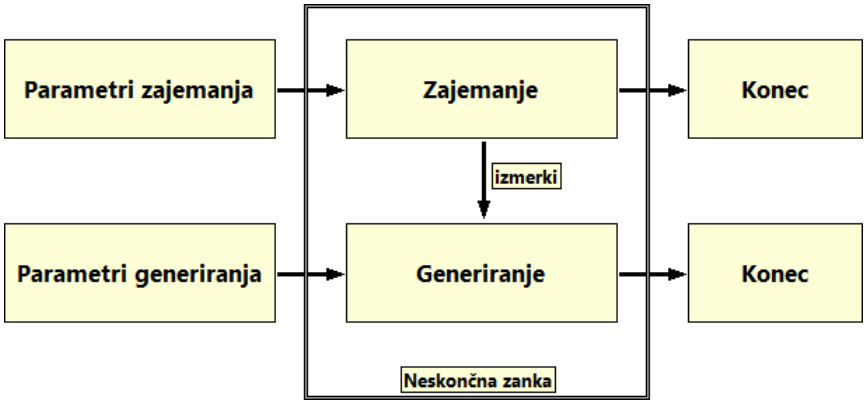
Sledi blok za definiranje časovnih intervalov med zaporednimi generiranimi vrednostmi, kot je to že bilo narejeno v prejšnjem primeru, nato v tretjem koraku pošljemo v pomnilnik vmesnika začetni niz števil. Razlika je malenkostna: tokrat v tretjem koraku pošljemo dvojno količino števil in tako na začetku dodobra napolnimo pomnilnik, ter v četrtem koraku startamo generiranje. Sledi neskončna zanka, v kateri vsakokrat sproti izračunamo normalno količino novih števi (glej spodnji parameter v dvojno obrobljeni listi parametrov za generator trikotnega signala) ter jih pošljemo v pomnilnik vmesnika s ponovljenim tretjim korakom. Izvajanje neskončne zanke lahko prekine uporabnik s pritiskom na gumb »STOP«, izvajanje se lahko prekine tudi zaradi napake pri prenašanju podatkov. Sledi še prekinitev povezave med našim programom in vmesnikovim gonilnikom ter izpis morebitnih napak na zaslon.



Slika 6: Program za generiranje poljubnega signala s sprotnim računanjem

Neprekinjeno sočasno zajemanje in generiranje signalov

Združimo zgleda za neprekinjeno zajemanje in generiranje s sprotnim računanjem v enoten program. Ta program naj signal neprekinjeno zajema na analognem vhodu vmesnika ter ga nespremenjenega posreduje na analogni izhod vmesnika, slika 7. Ko imamo na razpolago ta mehanizem, lahko dopolnimo program v neskončni zanki z ukazi, ki vhodni zajeti signal po izbranem algoritmu spremenijo in takega generirajo na izhodu.



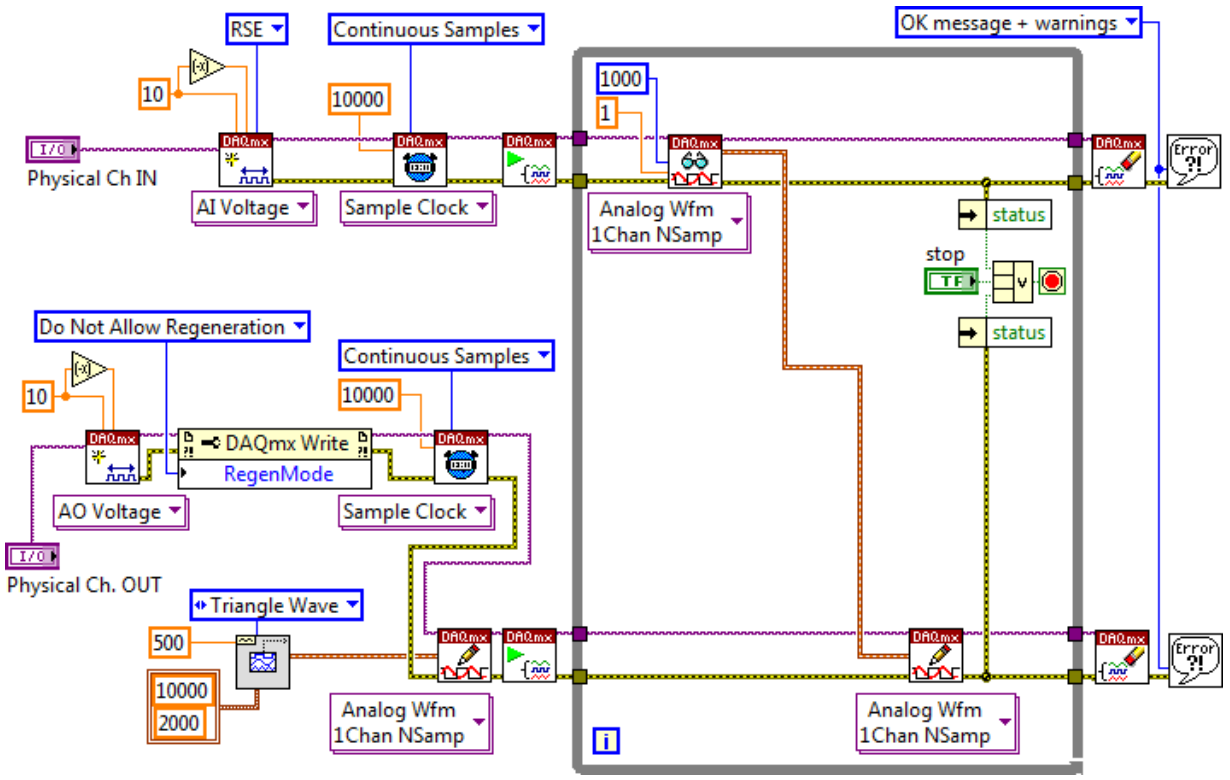
Slika 7: Posredovanje signala od vhoda v analogno-digitalni pretvornik preko našega programa na izhod digitalno-analognega pretvornika

Na sliki 8 je sestavljeni program. Zgornja veriga je namenjena zajemanju, spodnja pa generiranju. Obe verigi se stikata samo v notranjosti neskončne zanke na dveh mestih. Zajeti izmerki se iz zgornje verige posredujejo kot nove izhodne vrednosti spodnji verigi, generatorja trikotnega signala ni več. Izvajanje neskončne zanke je mogoče prekiniti na tri načine: s pritiskom na gumb »STOP«, zaradi napake v zgornji verigi ali zaradi napake v spodnji verigi.

Na zgledu s slike 8 je niz podatkov, ki jih prenašamo iz vmesnika v naš program in (predelane) nazaj v vmesnik definiran z dolžino niza za branje in znaša 1000 elementov. Pri izbrani frekvenci vzorčenja (10 kHz) torej računalnik opravi prenos podatkov iz vmesnika v računalnik in nazaj desetkrat v sekundi, vsakokrat prenese po 1000 elementov x 2 byta = 2 kB podatkov. To USB vodilo brez težav zmore. Če povečamo frekvenco vzorčenja na 200 kHz, se prenaša še vedno enaka količina podatkov v vsakem prenosu, le prenosi so bolj pogosti, na vsakih 5 ms. Zaradi tega se lahko zgodi, da preko USB vodila ni več mogoče v tako kratkih časovnih intervalih paketa podatkov poslati pravočasno in vmesniku se zatakne, zaradi tega gonilnik javi napako, izvajanje programa se prekine in na zaslonu se izpiše napaka. Problemu se ognemo, če v posameznem prenosu zahtevamo več podatkov, na primer 10000. Potem bo spet potrebnih le 20 prenosov v sekundi, vsakokrat pa se prenaša 20 kB podatkov. To USB vodilo brez težav zmore.

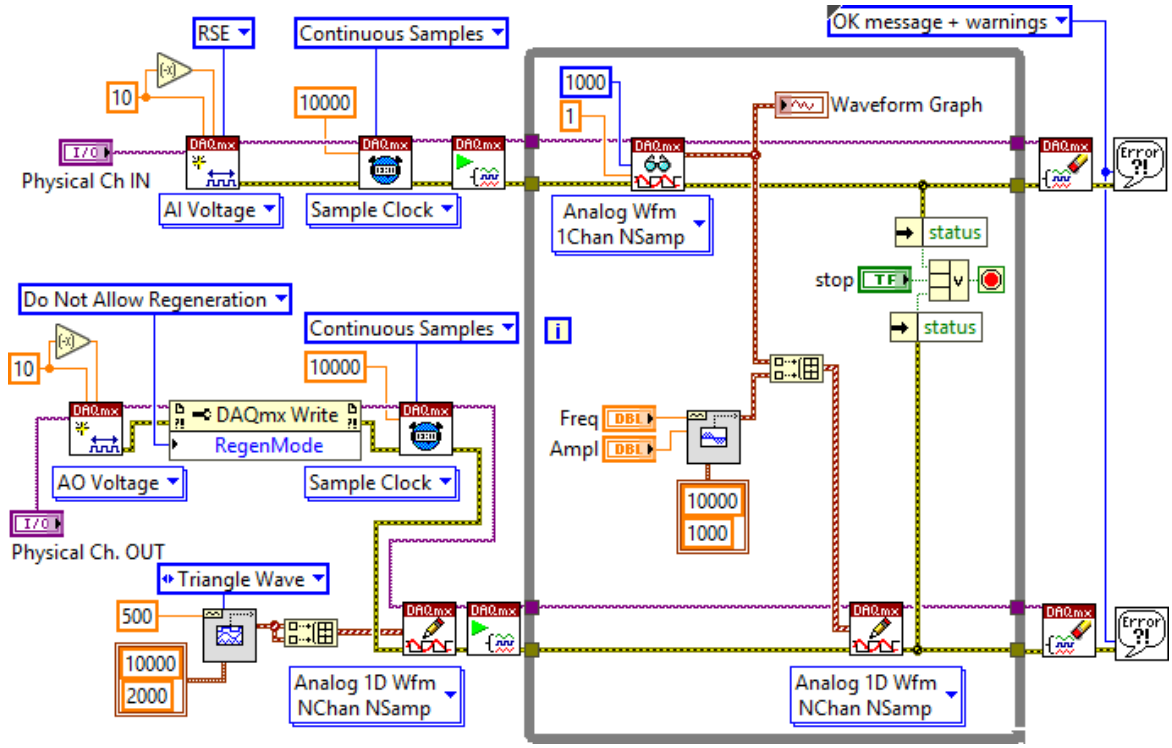
Ker morata obe verigi delovati istočasno, je nujno uskladiti število vzorcev, ki jih čakamo v zgornji verigi s številom vzorcev, ki jih pošiljamo v spodnji verigi. V zgledu, ki je na sliki 8, je to samoumevno, saj vse, kar zajamemo v zgornji verigi tudi pošljemo v spodnjo. Marsikdaj pa želimo več. Za primer naj navedemo zgled, kjer v spodnji verigi po prvem kanalu (AO0) generiramo vzbujalni signal za eksperiment, v zgornji pa zajemamo rezultate tega eksperimenta. Dodatno v spodnji verigi generiramo še signal, ki je posledica izmerkov (v zgledu naj bo to kar ponovljeni niz izmerkov). V tem primeru

potrebujemo v spodnji verigi generator, ki deluje pri isti frekvenci vzorčenja in daje enako število vzorcev, kot jo zajemamo v zgornji verigi, slika 9, le tako lahko obe delujeta sinhrono.



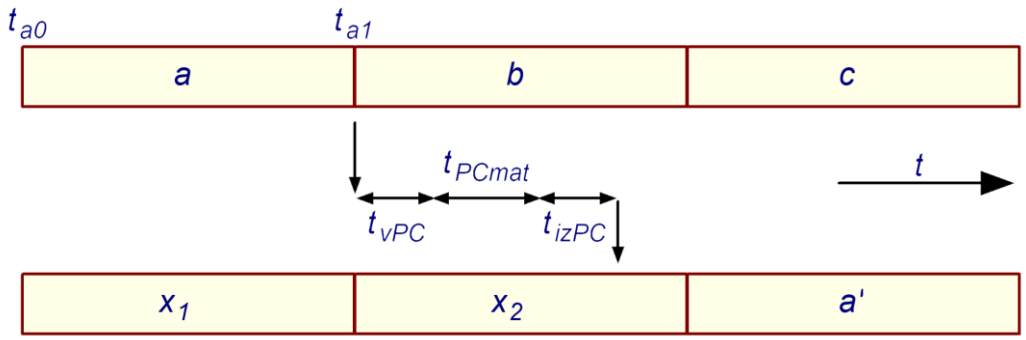
Slika 8: Zgled programa za neprekinjeno zajemanje in generiranje signalov

Poudariti je treba še začetno število vzorcev, ki jih pošljemo v spodnjo verigo pred izvajanjem »while« zanke. Iz slike 8 je razvidno, da v zanki preberemo vsakokrat 1000 izmerkov, v spodnjo verigo pa smo pred zanko poslali dvakrat toliko vzorcev. Ilustracija na sliki 10 pojasnjuje vzroke. Naj pravokotniki *a*, *b* in *c* označujejo zaporedne nize izmerkov, čas pa teče v horizontalni smeri; za program na sliki 8 bi vsak pravokotnik predstavljal niz 1000 vzorcev. Podrobneje poglejmo situacijo okoli niza *a*. Ta niz izmerkov vmesnik začne zajemati ob času t_{a0}, in po tisoč zajetih izmerkih konča ob času t_{a1}. Med zajemanjem naslednjega niza *b* sledi prenos niza izmerkov iz vmesnika preko USB vodila v računalnik, kar vzame t_{vPC} časa. Potem je na vrsti računalnik, ki v času t_{PCmat} opravi preračunavanje zajetega niza v zaporedje vzorcev za generiranje, te pa je treba v času t_{izPC} poslati iz računalnika preko USB vodila v vmesnik. Generiranje v spodnji in zajemanje zgornji verigi poteka bolj ali manj sočasno, spodnja veriga pa zahteva niz novih vzorcev še preden začne z generiranjem tega niza; če nov niz vzorcev ne prispe v vmesnik pravočasno, vmesnik javi napako in program se ustavi. Niz novih vzorcev bo torej mogoče generirati šele v paketu *a'*. Ker pa obe verigi v vmesniku začneta teči bolj ali manj sočasno, moramo verigo za generiranje zaposliti do takrat, ko do nje prispe novo poslani niz *a'*. Iz slike je razbrati, da bo začetek generiranja tega niza (*a'*) sovpadal z začetkom zajemanja niza *c*, torej generiranje kasni za zajemanjem za dva niza. Zato predhodno pred »while« zanko natlačimo v vmesnik za dva niza nepomembnih podatkov (na primer trikotnega signala, 2000 vzorcev), ki bosta služila generiranju v paketih x_1 in x_2. Generirani signal torej kasni za zajetim za dva niza, v našem primeru za 200 ms. Predhodno lahko v vmesnik natlačimo tudi več podatkov, a to običajno ni potrebno, saj podaljša kasnitev izhodnega signala za vhodnim. Le pri izjemno dolgem preračunavanju bi bilo morda smiselno na začetku natlačiti več vzorcev v verigo za generiranje.



Slika 9: Zgled programa za neprekinjeno zajemanje in generiranje s sprotnim definiranjem vzbujalnega signala (izhodna signala sta dva, vhodni je en sam !)

Ker so v prenašanje paketov preko USB vodila vpleteni še gonilniki USB s svojimi spominskimi enotami, se lahko zdi kasnitev izhodnega signala za vhodnim, ko ju opazujemo z osciloskopom, še daljša.

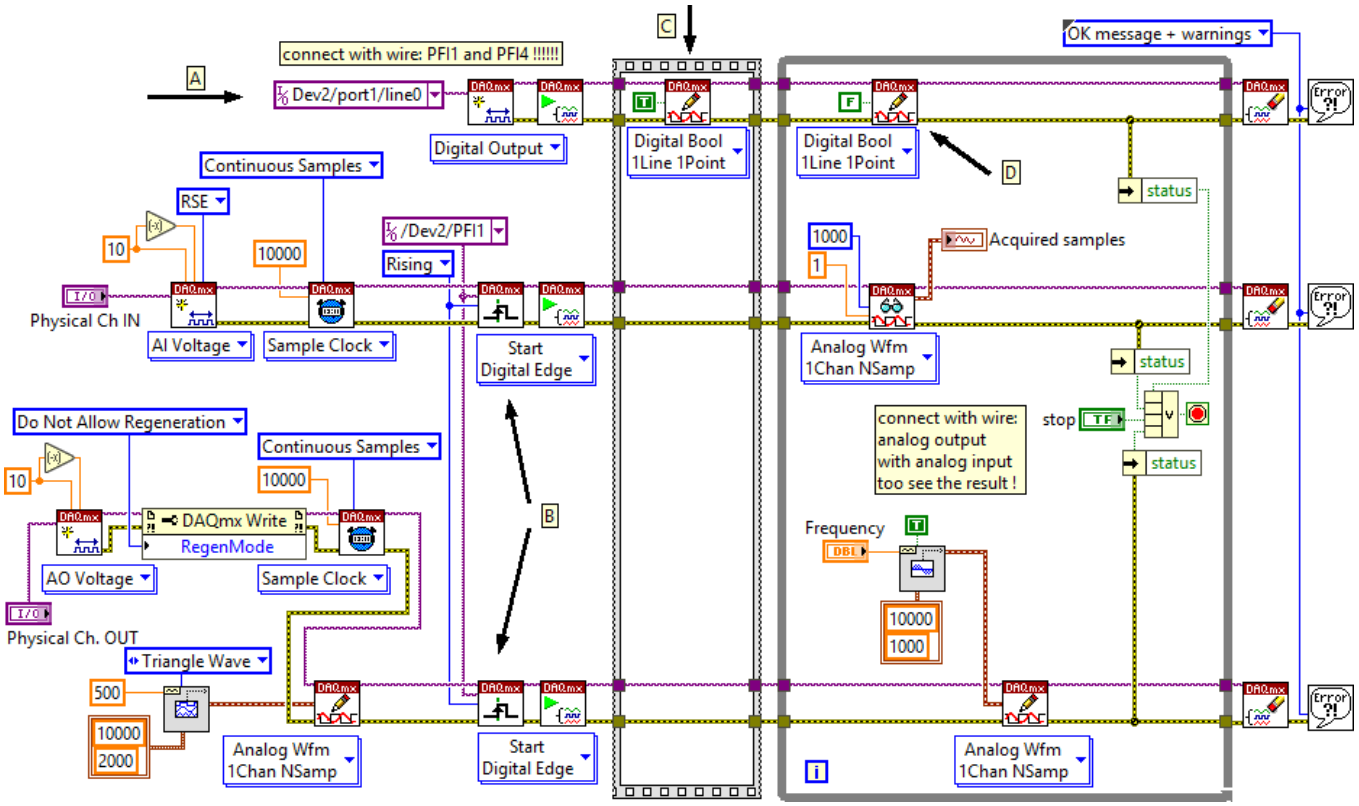


Slika 10: Prenos nizov podatkov iz in v vmesnik

Precej nedefinirane so tudi okoliščine okoli časovne usklajenosti zajemanja in generiranja. Programsko okolje Labview se samo odloča o vrstnem redu izvajanja blokov, na vrstni red (ne sočasnost!) lahko vplivamo le z medsebojnimi povezavami med bloki in »film-strip« opcijo. V programu na slikah 8 in 9 ne vemo, ali se bo prej izvedel zgornji ali spodnji blok »start« (), zato meje med bloki *a*, *b* in *c* spodnje in zgornje veje v realnem svetu niso popolnoma poravnane v času, kot so na sliki 10. Dodatno k ne-sočasnosti pripomore tudi USB vodilo. Ukaza »start« za zgornjo in spodnjo verigo sta namreč dva ločena ukaza, ki ju je treba prenesti v vmesnik in si sledita drug za drugim s kasnitvijo, ki jo vnaša vodilo USB. Zaradi vsega tega oblika signala, ki ga generira vmesnik v spodnji verigi sicer pravilno sledi vhodnemu signalu v zgornji verigi, na vedno enako kasnitev generiranega signala za merjenim pa ne

gre računati; ob vsakem zagonu programa bo kasnitev izhodnega signala za vhodnim lahko malo drugačna. To navadno ne moti. Morda je potrebno za pravilno obdelavo zajetega signala poleg tistega, ki je posledica vzbujanja eksperimenta, zajemati še vzbujalni signal, kar pa je tako ali tako koristno, saj se lahko prepričamo o velikosti, frekvenci in fazi vzbujalnega signala.

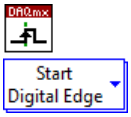
Pri nekaterih eksperimentih pa je nujno potrebna časovna usklajenost generiranja in zajemanja in takrat se ne moremo zanesti samo na programsko opremo, ampak zaupamo strojni opremi vmesnika. Obe verigi za zajemanje (ADC) in generiranje (DAC) je v vmesniku mogoče sprogramirati tako, da začneta delovati ob izbranem robu digitalnega sunka, ki ga privedemo na enega od digitalnih vhodnih priključkov vmesnika. Pravimo, da sta verigi digitalno proženi (»Start: Digital edge«). Zgled za takšen program je na sliki 11. Tudi tukaj velja, da je treba pred startom v spodnjo verigo natlačiti dovolj vzorcev; zadošča dvakratnik izmerkov v paketu zgornje verige.



Slika 11: Zgled programa za s strojno opremo povzročenim začetkom delovanja verig za zajemanje in generiranje signalov; zajemamo in generiramo po en signal

Digitalni signal, ki sproži delovanje verige za zajemanje in verige za generiranje lahko daje vmesnik preko svojih digitalnih izhodov PF0 do PF7. Štirje priključki od PF0 do PF3 so rezervirani za vhodne signale v vmesnik, preostali štirje priključki od PF4 do PF7 pa za izhodne signale. Vmesnik zato lahko generira poskok digitalnega signala (na primer) na priključku PF4 (to je »line0« v besedišču NI vmesnika), ta signal pa uporabimo na vhodnem priključku (na primer) PF1 za sproženje začetka zajemanja in generiranja. Z žico moramo torej povezati omenjena dva priključka.

Program je potrebno dopolniti:

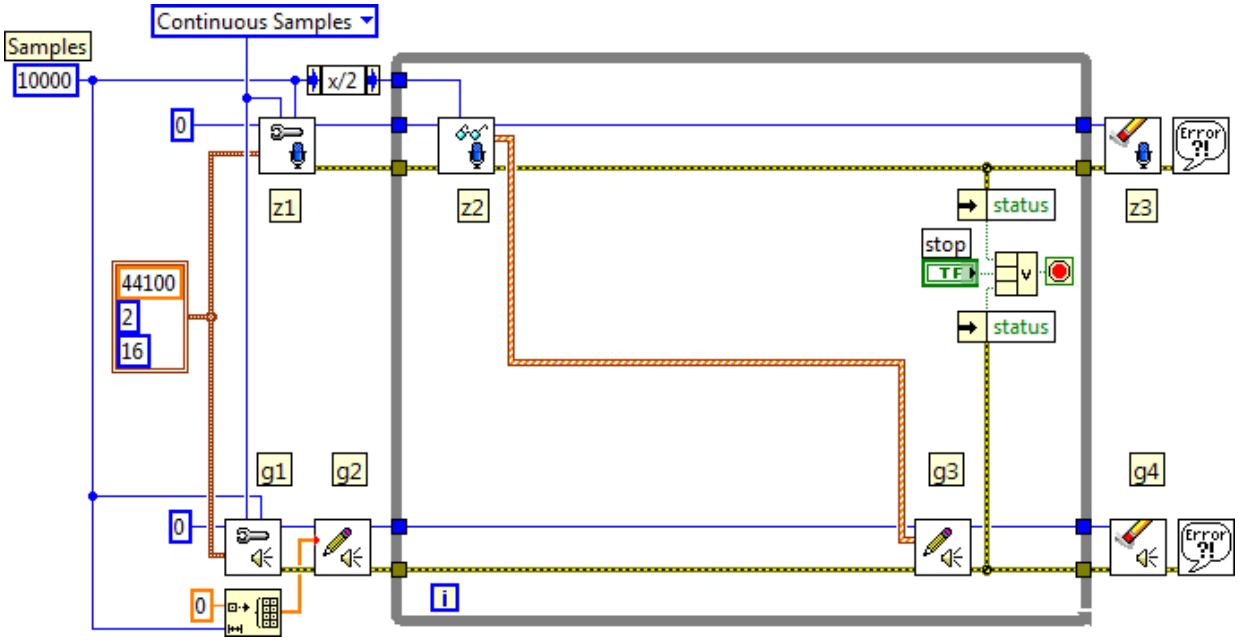
- V verigo za zajemanje in v verigo za generiranje dodamo bloka (puščica »B«), ki definirata način začetka zajemanja in generiranja (

), in tema blokoma dopovemo, da nas zanima rastoči rob digitalnega signala, ki se pojavi na vhodnem priključku PF1.
- Dodamo verigo za krmiljenje digitalnega izhodnega priključka PF4, na sliki zgoraj označena s puščico »A«. Najprej izberemo izhodni priključek (»Dev2/port1/line0«) in vzpostavimo povezavo med programom in vmesnikom, rezultata sta karakteristična številka opravila in sporočilo o morebitni napaki. Takoj zatem poženemo delovanje te verige. V tako-imenovanem »Film strip« pošljemo na ta digitalni izhod logično ena (»T«, puščica »C«), skozi isti »Film strip« vodimo tudi signale za preostali dva verigi; to storimo zato, da se ne bi preostali dve verigi sprožili predčasno. V glavni »While« zanki lahko digitalni signal vrnemo na logično nič (»F«, puščica »D«) in vmesnik s tem pripravimo na naslednji zagon programa. Zgornja veriga je tako svoje delo opravila, generirala je en digitalni sunek, katerega rastoči rob sproži delovanje preostalih dveh verig.

Neprekinjeno sočasno zajemanje in generiranje signalov z zvočno kartico v okolju Labview

Zvočno kartico sestavlja dvokanalni analogno-digitalni pretvornik, dvokanalni digitalno-analogni pretvornik ter nadzorna enota. Vse skupaj je prilagojeno zajemanju in generiranju signalov v področju slišnih frekvenc od 20 Hz do 20 kHz. Tudi tu je delovanje zvočne kartice samostojno, parametre za delovanje zvočne kartice in podatke pa kartici posreduje gonilnik, do katerega uporabnik dostopa preko knjižnjic.

Tipična frekvenca vzorčenja je 44100 Hz, kar je malo nad Nyquistovo frekvenco. Na zvočni kartici so večinoma vgrajeni analogni filtri, ki izločijo signale s frekvencami izven slišnega območja. Tipična velikost na zvočno kartico priključenih signalov znaša do nekaj 100 mV.



Slika 12: Zgled programa za neprekinjeno sočasno zajemanje in generiranje signalov z zvočno kartico

Program s slike 12 je namenjen sočasnemu zajemanju in generiranju z zvočno kartico. Tudi tu imamo opravka z nekaj zaporednimi klici v knjižnico, preko teh klicev pošiljamo zvočni kartici podatke in parametre za delovanje. Na sliki 12 je zgornja veriga spet namenjena neprekinjenemu zajemanju, spodnja pa neprekinjenemu generiranju. Obe verigi sva povezani v dveh točkah. Podatki se iz zgornje verige (zajem) prenesejo v spodnjo verigo (generiranje), zato kartica na svojem izhodu le ponovi tisto, kar je priključeno na njen vhod. Izvajanje neskončne zanke lahko prekine katerakoli od verig, če pride do napake.

Obe verigi poganjamo pri istih parametrih: zajemamo s hitrostjo 44100 s^{-1}, na 2 kanalih s po 16 biti. Ti parametri so navedeni na levi strani slike 9 v dvakrat obrobljenem pravokotniku in so v obliki grozda (»cluster«) posredovani prvima dvema klicema v knjižnico za vsako verigo posebej. Prvi korak (bloka z1 in g1) v vsaki verigi je namenjen vzpostavitvi povezave med našim programom in gonilnikom zvočne kartice. Za parameter navedemo oznako zvočne kartice, s katero želimo delati. Ker je v računalnik tokrat vstavljena le ena zvočna kartica, je njena oznaka 0, kar navedemo obema verigama na levih

robovih blokov z1 in g1. Ob času vzpostavljanja je treba definirati dolžino niza, ki ga bomo brali iz zvočne kartice ali vanjo pošiljali, to definira konstanta »Samples«. Obe verigi naj delujeta neprekinjeno brez presledkov do takrat, ko uporabnik delovanje prekine (ali do napake pri delovanju), zato v tem koraku izberemo opcijo »Continuous Samples« v meniju nad klicema blokov z1 in g1. Kot rezultate teh dveh blokov dobimo dve identifikacijski kodi na desnih zgornjih robovih blokov ter dve kodi morebitne napake na desnih spodnjih robovih blokov. Vse to posredujemo naslednjim blokom, ki se izvedejo le, če v prejšnjem bloku ni prišlo do napake.

Podobno kot pri rabi vmesnika NI6211 tudi tokrat pred neskončno zanko prvič pred zanko pošljemo podatke zvočni kartici. Temu je namenjen blok g2. Podatke tokrat določimo kot polje elementov z vrednostjo nič, dolžina polja je spet podana s konstanto »Samples«.

V neskončni zanki nato v zgornji verigi zahtevamo branje niza izmerkov iz zvočne kartice. Ko se nabere »Samples« izmerkov, jih blok z2 prenese v naš program. Format podatkov za ta blok je v naprej definiran in je polje z dvema elementoma, za vsak kanal en element. Posamezen element je grozd tipa »Waveform«, sestavljen iz polja izmerkov v dvojni natančnosti, časovnega intervala med izmerkoma ter oznake časa začetka niza. Tako formatirane podatke direktno posredujemo bloku g3 za prenos podatkov v zvočno kartico. Prenašanje zajetih podatkov iz zvočne kartice in nazaj v zvočno kartico se ponavlja dokler uporabnik ne stisne tipke »STOP«.

Takrat se izvršita še bloka z3 in g4, ki prekineta programsko povezavo med našim programom in gonilnikom zvočne kartice ter posredujeta kodo morebitne napake zadnjemu bloku za izpis na ekran.

Blok za pisanje g2/g3 je polimorfičen, kar pomeni, da je zadovoljen z različno formatiranimi podatki, ki jih skuša po svojih najboljših močeh predelati v sebi razumljive. Kadar do predelave pride, je ob priključku v blok narisana majhna rdeča pika, glej blok g2.