EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vývoj aplikácií v LabVIEW pre meracie úlohy s CompactRIO systémami

Učební texty k semináři

Autoři:

Ing. Mgr. Márk Jónás (ANV s.r.o., Bratislava)

Ing. Zuzana Petráková (ANV s.r.o., Bratislava)

Mgr. Silvia Mókosová (ANV s.r.o., Bratislava)

Ing. Gregor Izrael, PhD. (ANV s.r.o., Bratislava)

Datum:

13.-15. 8. 2012 a 20.-24. 8. 2012

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

OBSAH

1. ORINETÁCIA V LABVIEW

Táto lekcia uvádza navigáciu v prostredí LabVIEW. Zahŕňa používanie menu, nástrojovej lišty, palety, nástroje, pomocníka a všeobecné dialógov okná v LabVIEW. Zároveň sa naučíte spustiť VI a získate všeobecný prehľad o čelnom panely a blokovom diagrame.

Programy v LabVIEW sa nazývajú virtuálne inštrumenty alebo VI, pretože ich vzhľad a chod imituje fyzické inštrumenty, ako napríklad osciloskopy a multimetre. LabVIEW obsahuje obsiahlu skupinu nástrojov pre akvizíciu, analýzu, zobrazenie a uloženie dát ako aj nástroje ktoré vám pomôžu riešiť problémy vo vyvíjanom kóde.

LabVIEW VI obsahuje tri hlavné súčasti - okno čelného panelu, blokový diagram, a ikonku/konektorový panel.

1.1. Okno čelného panelu

Okno čelného panelu je používateľským rozhraním pre VI. Okno čelného panelu si vytvoríte pomocou kontroliek a indikátorov, ktoré sú interaktívnymi vstupnými a výstupnými terminálmi VI.

1.2. Okno blokového diagramu

Po vytvorení okna čelného panelu pridáte kód použitím grafickej reprezentácii funkcií na prácu s objektmi čelného panelu. Okno blokového diagramu obsahuje zdrojový grafický kód. Objekty čelného panela sú zobrazené ako terminály na blokovom diagrame.

1.3. Ikona a konektorový blok

Ikona a konektorový blok umožnia použiť a zobraziť VI v rámci iného VI. VI ktoré je použité v rámci iného VI sa nazýva subVI, čo je podobné ako funkcia v textovo orientovanom programovacom jazyku. K tomu aby ste mohli použiť VI ako subVI, VI musí disponovať s ikonou a konektorovým blokom.

ikona Každé VI zobrazuje ikonu v pravom hornom rohu okna čelného panelu a blokového diagramu. Príklad predvolenej ikony je zobrazený naľavo. Ikona je grafická reprezentácia VI. Ikona môže obsahovať text aj obrázky. Ak používate VI ako subVI, ikona identifikuje subVI na blokovom diagrame VI. Predvolená ikona obsahuje číslo ktoré zobrazuje koľko nových VI ste otvorili po spustení LabVIEW.

ikona K tomu, aby ste mohli použiť VI ako subVI musíte vytvoriť konektorový panel, zobrazený naľavo. Konektorový panel je súbor terminálov na ikone ktoré zodpovedajú kontrolkám a indikátorom pre dané VI, podobne ako je to u parametrov volania funkcie v textovo orientovaných programovacích jazykoch. Prístup ku konektorovému panelu je pomocou pravého kliku na ikonu v pravom hornom rohu čelného panelu. Prístup ku konektorovému panelu z blokového diagramu nie je možný.

1.4. Spustenie VI

Pri spustení LabVIEW sa vám zobrazí okno Getting Started. Použite toto okno k vytváraniu nových VI, projektov, k výberu naposledy otvorených LabVIEW súborov, k vyhľadávaniu príkladov a k vyhľadávaniu v LabVIEW Help. Taktiež máte prístup k informáciám a k iným zdrojom ktoré vám môžu byť nápomocné pri práci s LabVIEW, ako napríklad špecifické manuály, témy pomocníka, a zdroje na stránke ni.com/manuals.

Okno Getting Started sa zatvorí ako náhle otvoríte existujúci súbor alebo vytvoríte nový súbor. Toto okno môžete zobraziť pomocou voľby View»Getting Started Window.

Okno LabVIEW Getting Started

LabVIEW si môžete nakonfigurovať tak aby otvoril nové, prázdne VI namiesto zobrazenia Getting Started okna. Vyberte **Tools»Options**, následne zvoľte Environment zo zoznamu **Category**, a zaškrtnite voľbu **Skip Getting Started window on launch**.

1.5. Tvorba a otvorenie VI alebo projektu

Začať prácu v LabVIEW môžete pomocou prázdneho VI alebo projektu, otvorením a modifikáciou existujúceho VI alebo projektu, alebo otvorením šablóny pomocou ktorej môžete začať vyvíjať nové VI alebo projekt.

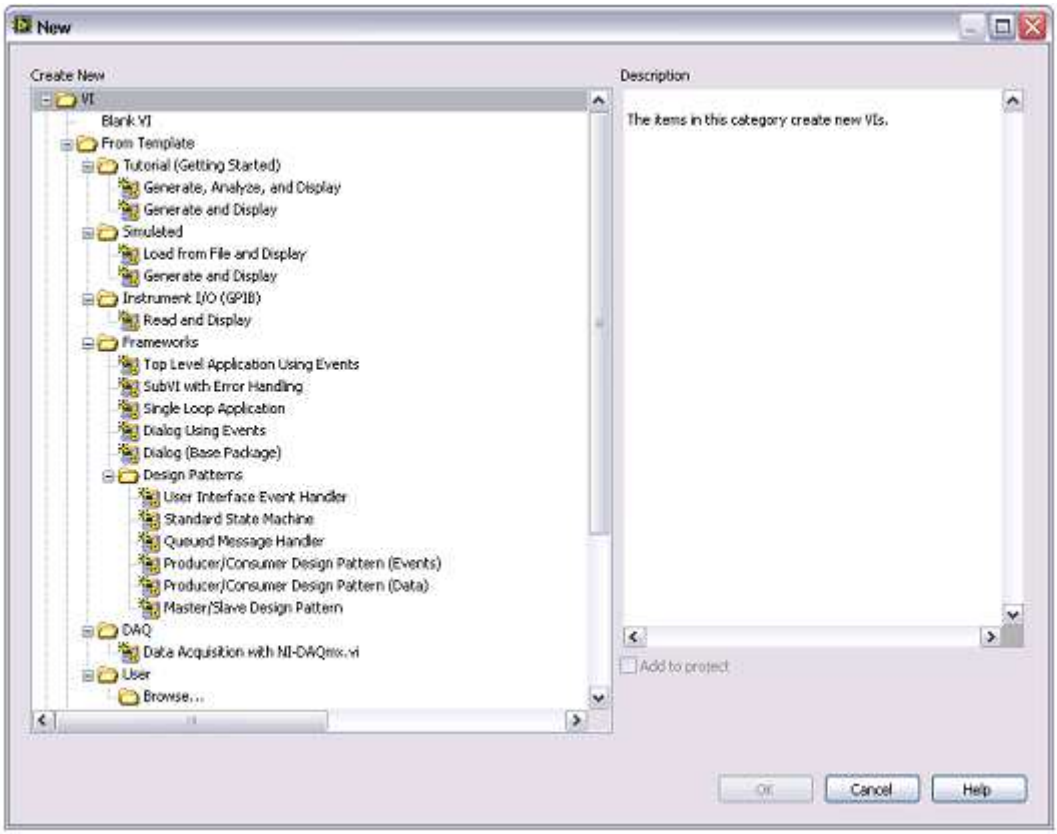
1.6. Tvorba nových projektov a VI

K otvoreniu nového projektu z okna **Getting Started**, vyberte **Empty Project** zo zoznamu **New**. Otvorí sa nový, nepomenovaný projekt do ktorého môžete pridávať súbory a uložiť projekt.

K otvoreniu nového, prázdneho VI, ktoré nie je pričlenené k projektu zvoľte v okne **Getting Started** voľbu **Blank VI** v zozname **New**.

1.7. Tvorba VI zo šablóny

Zvoľte **File»New** k zobrazeniu dialógového okna **New**, ktoré obsahuje vstavané šablóny. Dialógové okno **New** môžete taktiež zobraziť pomocou odkazu New v okne **Getting Started**.



1.8. Otvorenie existujúceho VI

V okne **Getting Started** zvoľte **Browse** v zozname **Open** k prístupu a otvoreniu existujúcich VI

Ak LabVIEW nedokáže okamžite lokalizovať subVI, potom začne hľadať adresároch špecifikovaných pomocou VI search path. VI search path môžete meniť v časti **Tools»Options** zvolením **Paths** zo zoznamu Category.

1.9. Uloženie VI

K uloženiu nového VI zvoľte **File»Save**. V prípade, že ste si už uložili VI a chcete zobraziť dialógové okno **Save As** zvoľte **File»Save As**. Z dialógového okna **Save As** môžete vytvoriť kópiu VI, alebo vymazať pôvodné VI a nahradiť ho s novým.

1.10. Project Explorer

Použite projekt na tvorbu zoskupenia LabVIEW súborov a nie LabVIEW súborov, na tvorbu build špecifikácií, a k prenosu súborov na cieľové zariadenia. Keď uložíte projekt, LabVIEW vytvorí projektový súbor (.lvproj), ktorý obsahuje referencie na súbory v projekte, informácie o konfigurácii, build informáciu, informáciu rozmiestnení a pod.

K tvorbe aplikácií a zdielaných knižníc musíte použiť projekt. Projekt musíte taktiež použiť pri práci s cieľovými zariadeniami ako real-time (RT), programovateľné hradlové pole (FPGA), alebo osobný digitálny asistent (PDA). Pre viac informácií ohľadne používania projektu s modulmi LabVIEW Real-Time, FPGA, a PDA si pozrite dokumentáciu k špecifickým modulom.

1.11. Okno Project Explorer

Na tvorbu a zmenu LabVIEW projektov použite okno **Project Explorer**. Na zobrazenie okna **Project Explorer** zvoľte **File»New Project**. K zobrazeniu okna **Project Explorer** môžete použiť aj voľbu **Project»New Project** alebo zvoliť **Empty Project** v dialógovom okne **New**.

Okno **Project Explorer** sa skladá z nasledovných štandardných častí:

- **Project root**—obsahuje všetky ostatné položky okna **Project Explorer**. Návestie tejto súčasti obsahuje meno projektového súboru.
- **My Computer**—reprezentuje lokálny počítač ako cieľové zariadenie
- **Dependencies**—obsahuje komponenty ktoré sú požadované VI pre dané cieľové zariadenie
- **Build Specifications**—obsahuje build konfigurácie pre distribúciu kódu a iné typy buildov ktoré sú dostupné v LabVIEW toolkitoch a moduloch. Ak máte nainštalovaný LabVIEW Professional Development System alebo Application Builder, tak môžete použiť **Build Specifications** na konfiguráciu samostatných aplikácií, zdielaných knižníc, inštalátorov, a zip súborov.

Keď pridáte ďalšie cieľové zariadenie do projektu, LabVIEW vytvorí ďalšiu položku v okne **Project Explorer** na reprezentáciu cieľového zariadenia. Každé cieľové zariadenie obsahuje sekcie **Dependencies** a **Build Specifications**. Súbory môžete pridať pod každé cieľové zariadenie.

1.12. Nástrojové lišty týkajúce sa projektu

Používajte tlačidlá **Standard, Project**, **Build**, a **Source Control** na nástrojovej lište k vykonaniu úkonov v LabVIEW projekte. K zobrazeniu všetkých nástrojových líšt často potrebujete roztiahnuť okno **Project Explorer**

Nástrojové lišty môžete zobraziť alebo ukryť pomocou voľby **View»Toolbars** a následným zvolením nástrojovej lišty ktorú chcete zobraziť alebo ukryť. Taktiež môžete pravým tlačidlom myši kliknúť na voľný priestor na nástrojovej lište a zvoliť si nástrojové lišty k zobrazeniu alebo k ukrytiu.

1.13. Tvorba LabVIEW projektu

K vytvoreniu projektu vykonajte nasledovné kroky.

1. Zobrazte okno **Project Explorer** pomocou voľby **File»New Project**. K zobrazeniu okna **Project Explorer** môžete použiť aj voľbu **Project»Empty Project** v dialógovom okne **New**.

2. Položky ktoré chcete zahrnúť do projektu pridajte pod cieľové zariadenia.

3. Projekt uložíte pomocou voľby **File»Save Project**.

1.14. Pridanie existujúcich súborov do projektu

Do projektu môžete vložiť existujúce súbory. Na pridávanie súborov ako napr. VI alebo textové súbory do LabVIEW projektu použite položku **My Computer** (alebo iné cieľové zariadenie) v okne **Project Explorer**.

Položky do projektu môžete pridávať nasledovne:

- Pravý klik na My Computer a zvoľte **Add»File** z kontextovej ponuky. Pridať súbor môžete aj pomocou voľby **Project»Add To Project»File** z nástrojovej lišty Project Exploreru.
- Adresár ktorý automaticky zobrazuje verný stav súborov na disku (auto-populating) pridáte pravým klikom na cieľové zariadenie a voľbou **Add»Folder** (Auto-populating). Ďalšou možnosťou pridania auto-populating adresára je pomocou nástrojovej lišty **Project»Add To Project»Add Folder** (Auto-populating). LabVIEW neustále monitoruje a aktualizuje adresár podľa zmien ktoré sa vykonali v projekte a na samotnom disku.
- Virtuálny adresár pridáte pravým klikom na cieľové zariadenie a voľbou **Add»Folder (Snapshot)**. Ďalšou možnosťou pridania virtuálneho adresára je pomocou nástrojovej lišty **Project»Add To Project»Add Folder (Snapshot)**. Keď si zvolíte adresár na disku, LabVIEW vytvorí nový virtuálny adresár v projekte s totožným názvom ako adresár na disku. LabVIEW taktiež vytvorí položky v projekte ktoré reprezentujú obsah celého adresára, zahrňujúc súbory a obsah podadresárov. Zvolením adresára z disku sa pridá obsah celého adresára, vrátane súborov a obsahu podadresárov.

- Nové, prázdne VI pridáte pomocou pravého kliku na cieľové zariadenie a voľbou **New»VI**. Pridať nové, prázdne VI môžete aj pomocou **File»New VI** alebo **Project»Add To Project»New VI**.
- Vyberte ikonku VI v pravom hornom rohu v okne čelného panelu alebo blokového diagramu a pretiahnite ikonku do cieľového zariadenia.
- Vyberte položku alebo adresár zo súborového systému na počítači a pretiahnite ho do cieľového zariadenia.

1.15. Odstránenie položiek z projektu

Súbory z okna **Project Explorer** môžete odstrániť nasledujúcimi spôsobmi:

- Pravým klikom vyberte položku ktorú chcete odstrániť a zvoľte **Remove from Project** z kontextovej ponuky.
- Vyberte položku ktorú chcete odstrániť a stlačte <Delete>.
- Vyberte položku ktorú chcete odstrániť a kliknite na tlačidlo **Remove From Project** na Standard nástrojovej lište.

1.16. Organizácie položiek v projekte

Okno **Project Explorer** obsahuje dve strany, stranu **Items** a stranu **Files**. Strana **Items** zobrazuje položky projektu tak ako existujú v projektovom strome. Strana **Files** zobrazuje položky projektu ktoré majú prislúchajúci súbor na disku. Na tejto strane môžete usporiadať názvy súborov a adresárov. Úkony v rámci projektu robené na strane **Files** zmenia obsah aj na disku. Prepnúť sa medzi stránkami môžete pomocou pravého kliku na adresár alebo položku pod cieľovým zariadením a voľbou **Show in Items View** alebo **Show in Files View** z kontextovej ponuky.

Použite adresáre na organizáciu položiek v okne **Project Explorer**. Do LabVIEW projektu môžete pridať dva typy adresárov, virtuálne adresáre a auto-populating adresáre. Virtuálne adresáre slúžia na organizáciu položiek projektu. Nový virtuálny adresár vytvoríte pravým klikom na cieľové zariadenie v Project Explorer a voľbou **New»Virtual Folder** z kontextovej ponuky. Auto-populating adresáre sú zaktualizované v reálnom čase aby odzrkadľovali obsah adresára na disku. Auto-populating adresáre pridávate do projektu ak potrebujete mať prehľad o aktuálnych položkách na disku.

Auto-populating adresáre sú viditeľné iba na stránke **Items** v okne **Project Explorer**. V prípade auto-populating adresárov si obsah disku môžete prezerať, ale vykonávanie operácií na disku, ako napr. premenovanie, odstránenie, nie je možné. Na vykonanie diskových operácii nad položkami v auto-populating adresári použite stranu **Files** v okne **Project Explorer**. Strana **Files** zobrazuje umiestnenie projektových adresárov na disku. Úkony v rámci projektu robené na strane **Files** zmenia obsah aj na disku. Obdobne, LabVIEW automaticky zaktualizuje auto-populating adresár v projekte ak spravíte zmeny v adresári mimo LabVIEW.

Položky v adresári môžete usporiadať. Usporiadanie položiek podľa alfabetického poradia vykonáte pravým klikom na adresár s voľbou **Arrange By»Name** z kontextového menu. Usporiadanie položiek podľa typu položky vykonáte pravým klikom na adresár s voľbou **Arrange By»Type** z kontextového menu.

1.17. Prezeranie súborov v Projekte

Keď pridáte súbor do LabVIEW projektu, LabVIEW si pridá odkaz na súbor na disku. Súbor otvoríte v jeho prednastavenom editore pravým klikom na súbor v okne **Project Explorer** a voľbou **Open** z kontextovej ponuky.

K nahliadnutiu kde sú uložené súbory na disku pravým tlačidlom myši kliknite na projekt a zvoľte **View»Full Paths** z kontextovej ponuky.

K nahliadnutiu kde sa súbory na ktoré sa projekt odkazuje nachádzajú na disku a v okne **Project Explorer** môžete použiť dialógové okno **Project File Information**. Dialógové okno **Project File Information** zobrazíte voľbou **Project»File Information**. Pravým klikom na projekt a voľbou **View»File Information** z kontextovej ponuky taktiež viete zobraziť dialógové okno **Project File Information**.

1.18. Uloženie projektu

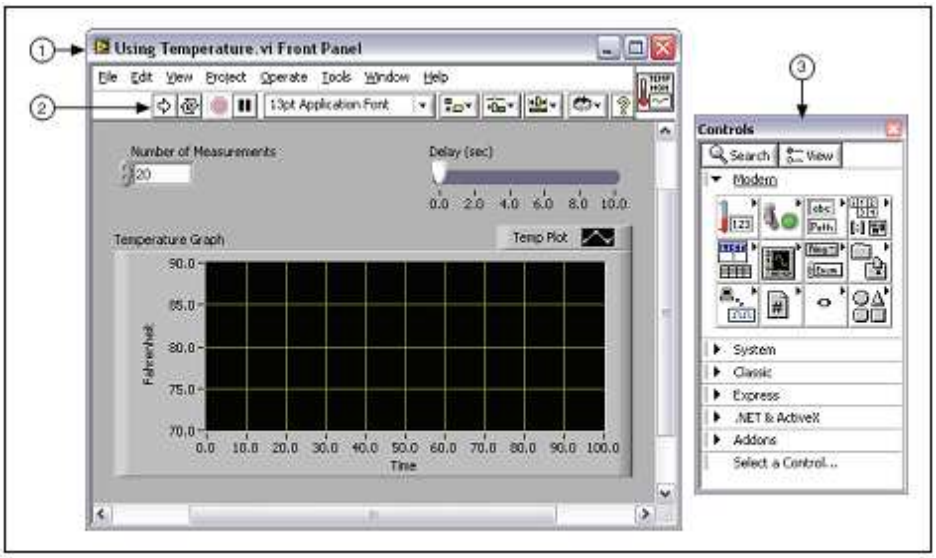
Uložiť projekt môžete nasledovnými spôsobmi:

- Voľbou **File»Save Project**.
- Voľbou **Project»Save Project**.
- Pravým klikom na projekt a voľbou **Save** z kontextového menu.
- Kliknutím na tlačidlo **Save Project** na **Project** nástrojovej lište.

Pred uložením samotného projektu musíte uložiť nové, neuložené súbory v projekte. Pri uložení projektu, LabVIEW neuloží závislé položky (dependencies) ako časť projektového súboru.

1.19. Čelný panel

Pri otvorení nového alebo existujúceho VI sa vám zobrazí čelný panel VI. Okno čelného panelu je používateľským rozhraním pre VI.



1 Okno čelného panelu 2 Nástrojové lišty 3 Paleta kontroliek

Príklad čelného panelu

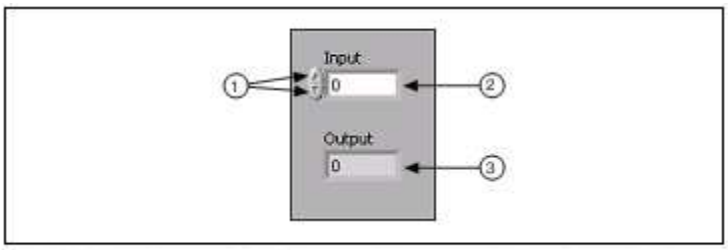
1.20. Kontrolky a indikátory

Čelný panel si vytvoríte pomocou kontroliek a indikátorov, ktoré sú interaktívnymi vstupnými a výstupnými terminálmi VI. Kontrolky sú otočné gombíky, tlačidlá, číselníky, a iné vstupné objekty. Indikátory sú grafy, LEDky, a iné objekty na zobrazenie hodnôt. Kontrolky simulujú vstupné zariadenia prístrojov a poskytujú dáta do blokového diagramu VI. Indikátory simulujú výstupy prístrojov a zobrazujú dáta ktoré blokový diagram zhromaždí alebo generuje.

Každá kontrolka a indikátor má dátový typ ktorý je jej priradený. Napríklad, vertikálny posúvač Delay(sec) má numerický dátový typ. Najbežnejšie používané dátové typy sú: numeric, boolean a string.

1.21. Kontrolky a indikátory

Numerický dátový typ môže reprezentovať čísla rôznych typov, ako napr. celé čísla alebo reálne čísla. Zaužívané numerické objekty sú numerická kontrolka a numerický indikátor Objekty ako merací panel a číselník taktiež reprezentujú numerické dáta.



1 Tlačidlá na inkrementáciu/dekrementáciu 3 Numerický indikátor
2 Numerická kontrolka

Numerické kontrolky a indikátory

Na zápis alebo zmenu hodnôt v numerickej kontrolke kliknite na inkrement a dekrement tlačidlá pomocou nástroja Operating tool, alebo dvojklikom na samotné číslo pomocou nástroja Labeling tool alebo Operating tool, zapíšte nové číslo a stlačte <Enter>.

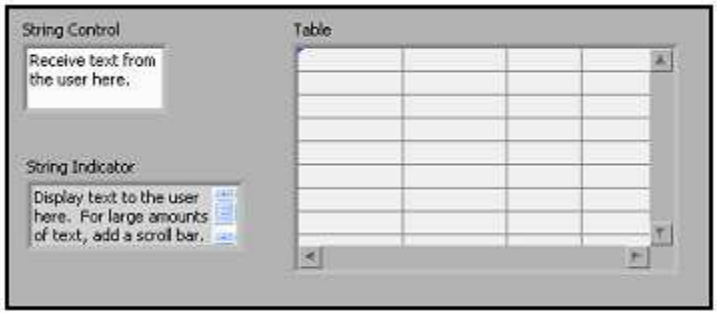
1.22. Boolean kontrolky a indikátory

Dátový typ Boolean reprezentuje dáta ktoré majú iba dva možné stavy, ako napríklad PRAVDA a NEPRAVDA alebo ON a OFF. Na zápis a zobrazenie boolean hodnôt použite boolean kontrolky a indikátory. Boolean objekty simulujú prepínače, tlačidlá a LEDky.

Boolean kontrolky a indikátory

1.23. String kontrolky a indikátory

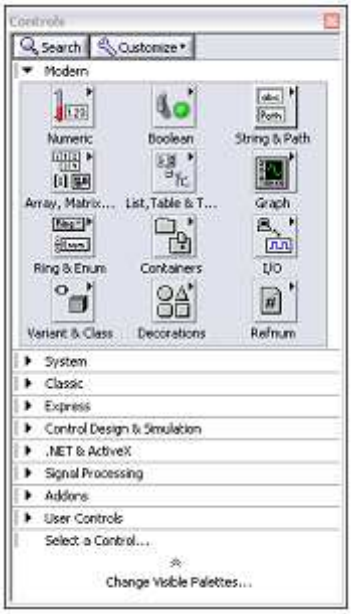
Dátový typ string reprezentuje sekvenciu ASCII znakov. String kontrolky použite na príjem textu od používateľa, napr. na zadanie hesla alebo používateľského mena. String indikátory použite na zobrazenie textu používateľovi. Najbežnejšie string objekty sú tabuľky a objekty na vstup textu.



String kontrolky a indikátory

1.24. Paleta kontroliek

Paleta Controls obsahuje kontrolky a indikátory ktoré použijete pri tvorbe čelného panelu. Paletu **Controls** si zobrazíte z okna čelného panelu pomocou **View»Controls Palette**. Paleta **Controls** je rozdelená do rôznych kategórií, podľa vašich potrieb si dajte zobraziť všetky alebo iba niektoré kategórie.



Paleta kontroliek

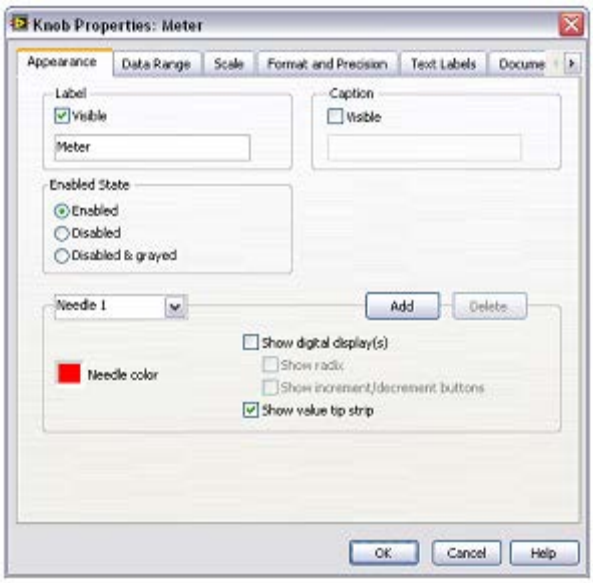
K zobrazeniu alebo k ukrytiu kategórií (subpaliet) použite tlačidlo **Customize** na palete a následne použite možnosť **Change Visible Categories**.

1.25. Kontextové ponuky

Všetky LabVIEW objekty majú priradené kontextové ponuky, nazývané aj ako pop-up ponuky. Počas tvorby VI používate kontextové ponuky jednotlivých položiek na zmenu vizáže alebo vlastností objektov na čelnom panely alebo blokovom diagrame. Kontextovú ponuku si zobrazíte pomocou pravého kliku myšou.

1.26. Dialógové okno vlastností

Objekty na čelnom panely disponujú dialógovými oknami na zobrazenie ich vlastností. Pomocou tejto možnosti môžete meniť vizáž a vlastnosti objektu. Dialógové okno vlastností objektu zobrazíte pomocou pravého kliku na objekt a voľbou **Properties** z kontextovej ponuky.



Dialógové okno vlastností pre objekt merací panel

Súčasne si môžete zvoliť viacero objektov na čelnom panely alebo blokovom diagrame a meniť vlastnosti ktoré tieto objekty zdielajú. K výberu viacerých objektov použite nástroj Positioning tool a nakreslite obdĺžnik okolo všetkých objektov ktoré chcete pridať do výberu alebo podržte stlačenú klávesu <Shift> a kliknite na každý objekt ktorý chcete pridať do výberu. K zobrazeniu dialógového okna Properties pre všetky spoločné vlastnosti objektov vo výbere kliknite pravým tlačidlom myši na objekt vo výbere a zvoľte **Properties** z kontextovej ponuky. Dialógové okno **Properties** zobrazuje iba vlastnosti ktoré sú spoločné pre všetky objekty vo výbere. Na zobrazenie väčšieho počtu možností vyberte podobné objekty. Ak vyberiete objekty ktoré nemajú žiadne spoločné vlastnosti, dialógové okno **Properties** nezobrazí žiadnu stránku alebo vlastnosť.

1.27. Nástrojová lišta okna čelného panelu

Každé okno disponuje s priradenou nástrojovou lištou. Na spustenie a zmenu VI použite tlačidlá na nástrojovej lište čelného panelu.

Nasledovná nástrojová lišta sa zobrazuje na okne čelného panela.

K spusteniu VI kliknite na tlačidlo **Run**. Ak je to potrebné, LabVIEW skompiluje VI. VI je spustiteľné ak tlačidlo **Run** sa zobrazí ako jednoliata biela šípka, zobrazená naľavo. Jednoliata biela šípka taktiež vyjadruje, že toto VI môžete použiť ako subVI ak pre toto VI vytvoríte konektorový panel.

Počas behu VI, tlačidlo **Run** nadobudne tvar ako je to zobrazené na obrázku naľavo ak VI ktoré beží je na najvyššej úrovni volaní, tj. nejedná

sa o subVI.

V prípade, že VI ktoré beží je volané subVI, tlačidlo **Run** sa zobrazí tak, ako je to znázornené naľavo.

Tlačidlo **Run** za zobrazí ako zlomená šípka keď VI ktoré vytvárate alebo meníte obsahuje chyby. Ak je tlačidlo **Run** po ukončení editovania blokového diagramu stále v tvare zlomenej šípky, znamená to, že VI nie je spustiteľné. Kliknite na toto tlačidlo, zobrazí sa vám okno **Error list** ktoré obsahuje zoznam všetkých chýb a upozornení.

Kliknite na tlačidlo **Run Continuously** ak chcete VI spúšťať cyklicky až pokým nezrušíte alebo nepauzujete vykonávanie. Na zrušenie nepretržitého behu znovu kliknite na tlačidlo.

Počas behu VI sa zobrazí tlačidlo **Abort Execution**. Kliknite na toto tlačidlo ak chcete okamžite ukončiť vykonávané VI, a ak neexistuje iný spôsob na ukončenie behu VI. Ak viacero súčasne bežiacich VI volá dané VI, toto tlačidlo je neaktívne.

K pauzovaniu VI kliknite na tlačidlo **Pause**. Keď kliknete na tlačidlo **Pause**, LabVIEW zvýrazní na blokovom diagrame tú časť pri ktorej došlo k pauze pri behu, a tlačidlo **Pause** sa zobrazí v červenej farbe. K opätovnému behu VI znovu kliknite na tlačidlo **Pause**.

Použite výsuvnú ponuku **Text Settings** na zmenu typu písma pre zvolenú časť VI, vrátane veľkosti, štýlu a farby.

Použite výsuvnú ponuku **Align Objects** k usporiadaniu objektov pozdĺž ich osi, horných, dolných hrán a pod.

Zvoľte výsuvnú ponuku **Distribute Objects** na rovnaké rozmiestnenie objektov.

Zvoľte výsuvnú ponuku **Resize Objects** na zmenu rozmerov viacerých objektov čelného panelu na totožnú veľkosť.

Zvoľte výsuvnú ponuku **Reorder** na určenie poradia prekrývajúcich sa objektov. Pomocou nástroja Positioning tool vyberte jeden z objektov a následne zvoľte jednu z možností **Move Forward**, **Move Backward**, **Move To Front**, a **Move To Back**.

Použite tlačidlo **Show Context Help Window** za zobrazenie okna **Context Help**.

Enter Text tlačidlo sa zobrazí aby vám pripomenulo, že nová hodnota je dostupná namiesto starej hodnoty. Tlačidlo **Enter Text** zmizne keď na

neho kliknete, keď stlačíte kláves <Enter> alebo keď kliknete na pracovnú plochu čelného panelu alebo blokového diagramu.

1.28. Blokový diagram

Objekty na blokovom diagrame zahrňujú terminály, subVI, funkcie, konštanty, štruktúry, a spojenia ktoré prenášajú dáta medzi jednotlivými objektmi blokového diagramu.

1.29. Terminály

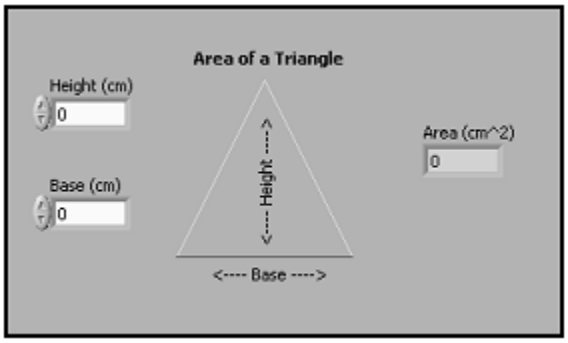
Objekty čelného panela sú zobrazené ako terminály na blokovom diagrame. Terminály sú vstupné a výstupné porty ktoré prenášajú informácie medzi čelným panelom a blokovým diagramom. Terminály sú obdobou parametrov a konštánt v textových programovacích jazykoch. Typy terminálov zahrňujú terminály kontroliek, indikátorov a uzlov. Controla indicator terminály patria ku kontrolkám a indikátorom na čelnom panely.

1.30. Kontrolky, indikátory a konštanty

Kontrolky, indikátory a konštanty reprezentujú vstupy a výstupy algoritmov na blokovom diagrame. Implementácia algoritmu pre výpočet obsahu trojuholníka:

Obsah = .5 * základňa * výška

V tomto algoritme sú vstupy základňa avýška, a výstupom je obsah



Čelné okno - Obsah trojuholníka

Používateľ nebude meniť konštantu .5, preto sa nezobrazuje na čelnom panely (pokiaľ nie je zobrazená ako dokumentácia algoritmu).

1.31. Okno blokového diagramu

Uzly sú objekty na blokovom diagrame ktoré majú vstupy a/alebo výstupy a vykonávajú operácie keď VI beží. Sú analogické k výrazom, operátorom, funkciám a podprogramom v textovo orientovaných programovacích jazykoch. Uzly môžu byť funkcie, subVI, alebo štruktúry. Štruktúry sú prvky riadenia toku programu, ako napríklad Case štruktúry, For slučky, alebo While slučky.

1.32. Funkcie

Funkcie sú základnými ovládacími časťami LabVIEW. Funkcie nemajú čelný panel a blokový diagram, ale disponujú s konektorovým panelom. Dvojklik na funkciu iba vyberie funkciu. Funkcia má ikonu so žltým pozadím.

1.33. SubVI

SubVI je VI ktoré používate v rámci iného VI alebo VI ktoré sa nachádza na Functions palete.

Ľubovoľné VI sa môže stať subVI. Dvojklikom na subVI na blokovom diagrame otvoríte jeho čelný panel. Čelný panel obsahuje kontrolky a indikátory. Blokový diagram obsahuje spojenia, ikony, funkcie, prípadne subVI a iné LabVIEW objekty. Pravý horný roh na okne čelného panela a blokového diagramu zobrazuje ikonu pre VI. Táto ikona bude reprezentovať VI na blokovom diagrame, ak ho použijete ako subVI.

SubVI môže byť aj tak zvaný Express VI. Express VI je uzol ktorý požaduje minimálny počet spojov nakoľko je konfigurovateľný pomocou dialógových okien. Používajte Express VI pre bežné meracie úlohy. Konfiguráciu Express VI si môžete uložiť ako subVI. Pre viac informácií ohľadne tvorby subVI z Express VI si pozrite tému Express VIs v LabVIEW Help.

LabVIEW používa farebné ikony na rozlíšenie medzi Express VI a inými VI na blokovom diagrame. Ikony pre Express VI disponujú s modrým podkladom, pričom subVI ikony majú žltý podklad.

1.34. Rozšíriteľné uzly verzus ikony

VI a Express VI môžete zobraziť ako ikony alebo ako rozšíriteľné uzly. Rozšíriteľné uzly sa zobrazia ako ikony obklopené vyfarbeným pozadím. SubVI sa zobrazia so žltým pozadím, Express VI s modrým pozadím. Použite ikony ak chcete šetriť priestorom na blokovom diagrame. Použite rozšíriteľné uzly ak chcete uľahčiť spájanie uzlov a pri podpore dokumentácii blokového diagramu. Podľa predvoleného nastavenia SubVI sa zobrazí na blokovom diagrame ako ikona, Express VI ako rozšíriteľný uzol. K zobrazeniu subVI alebo Express VI ako rozšíriteľný uzol klikniteuzol. K zobrazeniu subVI alebo Express VI ako rozšíriteľný uzol kliknite pravým tlačidlom myši na subVI alebo Express VI a odstráňte zaškrtnutie políčka vedľa položky View As Icon v kontextovej ponuke.

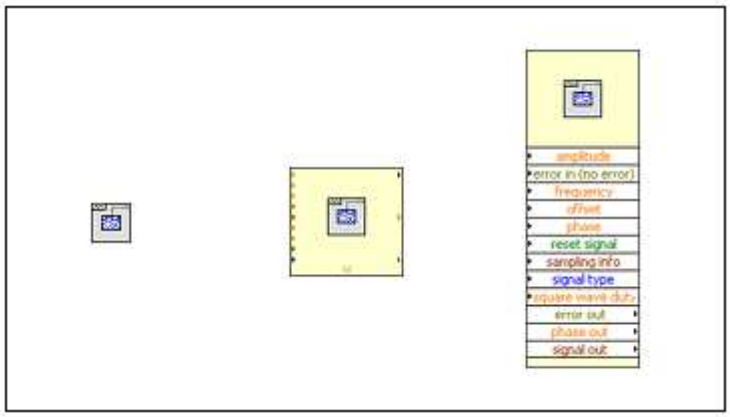
Môžete zmeniť veľkosť rozšíriteľného uzla tak aby spájanie uzlov bolo ešte jednoduchšie, tým ale zaberiete veľa miesta na blokovom diagrame. K zmene veľkosti uzlu na blokovom diagrame vykonajte nasledovné kroky:

1. Umiestnite nástroj Positioning tool nad uzol. Na vrchu a spodku uzla sa zobrazia úchytky na rozšírenie.

2. Umiestnite kurzor na úchytku tak aby sa kurzor zmenil na kurzor pre zmenu rozšírenia.

3. Použite kurzor pre rozšírenie, potiahnite hranu uzla smerom dole tak aby sa zobrazili ďalšie terminály.

4. Uvoľnite tlačidlo myšiNa zrušenie tohto úkonu rozšírenia pretiahnite kurzor mimo blokového diagramu pred uvoľnením myši.



Basic Function Generator VI v rôznych módoch zobrazenia

1.35. Spojenia

Cez spojenia sa prenášajú údaje medzi jednotlivými uzlami blokového diagramu. Každé spojenie má jediný zdroj dát, ale môže ústiť do viacerých VI a funkcií ktoré tieto dáta čítajú. Spojenia majú rôzne farby, štýly, hrúbku, ktoré závisia od dátového typu.

Porušené spojenie je zobrazené pomocou čiarkovanej čiary s X v strede, ako je to zobrazené naľavo. Porucha spojenia môže nastať z viacerých dôvodov, ako napríklad pri snahe spojiť dva objekty s nekompatibilnými dátovými typmi.

Bežné typy spojení

Typ spojenia	Škalár	1D Pole	2D PoleD	Farba
Numeric				oranžová (floating-point), modrá (integer)
Boolean				zelená
String/Text				ružová

V LabVIEW používate spojenia na prepojenie viacerých terminálov pri prenose dát vo VI. Spojenia musíte pripojiť k vstupom a výstupom ktoré sú kompatibilné s prenášanými dátami. Napríklad, nemôžete prepojiť výstup typu pole k vstupu typu numeric. Zároveň, smer spojenia musí byť správny. Spojenie musíte použiť iba s jedným vstupom ale aspoň s jedným výstupom. Napríklad, nemôžete prepojiť dva indikátory navzájom. Prvky ktoré určujú kompatibilitu spojení zahrňujú dátový typ kontrolky a/alebo indikátora a dátový typ terminálu.

1.36. Dátové typy

Dátové typy vyjadrujú aké objekty, vstupy, a výstupy môžete spolu prepojiť. Napríklad, ak prepínač má zelenú hranu, môžete ho prepojiť so ľubovoľným zeleným návestím na Express

VI. Ak otočné tlačidlo má oranžovú hranu, môžete ho prepojiť k ľubovoľnému vstupu s oranžovým návestím. Avšak nemôžete prepojiť oranžové otočné tlačidlo so vstupom so zeleným návestím. Všimnite si, že spojenia sú totožnej farby ako terminály.

1.37. Automatické spájanie objektov

Ak premiestnite vybraný objekt blízko k iným objektom na blokovom diagrame, LabVIEW nakreslí dočasné spojenia na ukážku možných spojení. Keď uvoľníte myš aby ste vložili objekt na blokový diagram LabVIEW automaticky vytvorí spojenia. Objekty, ktoré už sú na blokovom diagrame môžete taktiež automaticky prepojiť. LabVIEW prepojí terminály ktoré sa najlepšie párujú a neprepojí terminály ktoré sa nedajú spárovať.

Automatické spájanie docielite pomocou stlačeného medzerníka počas presunu objektu pomocou nástroja Positioning tool.

Podľa predvoleného nastavenia, automatické spájanie je aktívne ak vyberiete objekt z palety Functions alebo keď skopírujete objekt ktorý je už prítomný na blokovom diagrame s pomocou stlačenej klávesy <Ctrl> počas presunu objektu. Automatické spájanie je neaktívne pri použití nástroja Positioning tool na presun existujúcich objektov na blokovom diagrame.

Možnosti automatického spájania môžete meniť v časti **Tools»Options** zvolením **Block Diagram** zo zoznamu **Category**.

1.38. Manuálne spájanie objektov

Keď na terminál presuniete nástroj Wiring tool, zobrazí sa náveska s menom terminálu. Navyše terminál začne blikať v okne **Context Help** a n ikone tak, aby vám pomohol overiť si, že pripájate ten správny terminál. Kvytvoreniu spojenia medzi objektmi, použite nástroj Wiring tool, presuňte sja prvý terminál, kliknite myšou, presuňte sa na druhý terminál a zasa kliknite myšou. Po prepojení môžete spojenie vyhladiť pomocou voľby **Clean Up Wire** z kontextovej ponuky. LabVIEW automaticky vyberie vhodnú trasu pre spojenie. Ak chcete zrušiť nefunkčné spojenia stlačte <Ctrl-B> na výmaz všetkých nefunkčných spojení na blokovom diagrame

1.39. Paleta funkcií

Paleta **Functions** obsahuje VI, funkcie a konštanty ktoré používate pri tvorbe blokového diagramu. Paletu **Functions** si zobrazíte z okna blokového diagramu pomocou **View»Functions Palette**. **Paleta Functions** je rozdelená do rôznych kategórií, podľa vašich potrieb si môžete dať zobraziť alebo ukryť niektoré kategórie.

Paleta funkcií

K zobrazeniu alebo k ukrytiu kategórií (subpaliet) použite tlačidlo **Customize** na palete a následne použite možnosť **Change Visible Categories**.

1.40. Nástrojová lišta okna blokového diagramu

Po spustení VI sa na nástrojovej lište objavia tlačidlá ktoré môžete použiť na ladenie VI. Zobrazí sa nasledovná nástrojová lišta:

Použite tlačidlo **Highlight Execution** na zobrazenie animácie vykonávania kódu na blokovom diagrame keď beží VI. Všímajte si tok dát cez blokový diagram. Kliknite znovu na to isté tlačidlo ak chcete ukončiť animáciu vykonávania kódu.

Použite tlačidlo **Retain Wire Values** na nastavenie uchovania poslednej hodnoty ktorá prešla spojením počas vykonávania kódu. Túto hodnotu viete zobraziť následným použitím sondy (probe). VI musíte spustiť aspoň raz k tomu aby ste mohli uchovávať hodnoty spojení

Kliknite na tlačidlo **Step Into** k otvoreniu a k pauze vykonávania VI. Keď znovu kliknete na tlačidlo Step Into, tak sa vykoná prvá akcia nasledovaná pauzou. Takisto môžete stlačiť klávesy <Ctrl> a šípku smerom dole. Jednoduché krokovanie prechádza cez VI uzol po uzle. Každý uzol blikne keď je pripravený k vykonaniu.

Kliknite na tlačidlo **Step Over** k vykonaniu uzlu a k pauze pred nasledovným uzlom. Takisto môžete stlačiť klávesy <Ctrl> a šípku smerom doprava. Použitím možnosti step over sa uzol vykoná bez jednoduchého krokovania v rámci uzlu.

Kliknite na tlačidlo **Step Out** k ukončeniu vykonávania aktuálneho uzlu a k následnej pauze. Keď VI ukončí beh, tlačidlo **Step Out** je deaktivované. Takisto môžete stlačiť klávesy <Ctrl> a šípku smerom hore. Pomocou Step Out ukončíme jednoduché krokovanie uzlu a prejdeme k nasledujúcemu uzlu.

Použite tlačidlo **Clean Up Diagramna** automatický rozvrh spojení a objektov na blokovom diagrame tak aby konečný výsledok bol usporiadanejší. K nastaveniu možností rozvrhu zobrazte Options dialógové okno pomocou **Tools»Options** a následne zvoľte **Block Diagram** zo zoznamu **Category**. Konfiguráciu môžete nastaviť v časti Block Diagram Cleanup.

Tlačidlo **Warning** sa zobrazí ak VI obsahuje výstrahu (warning) a ste zaškrtli políčko **Show Warnings** v okne **Error List**. Výstraha vyjadruje, že existuje potenciálny problém s blokovým diagramom, ktorý ale nebráni behu VI.

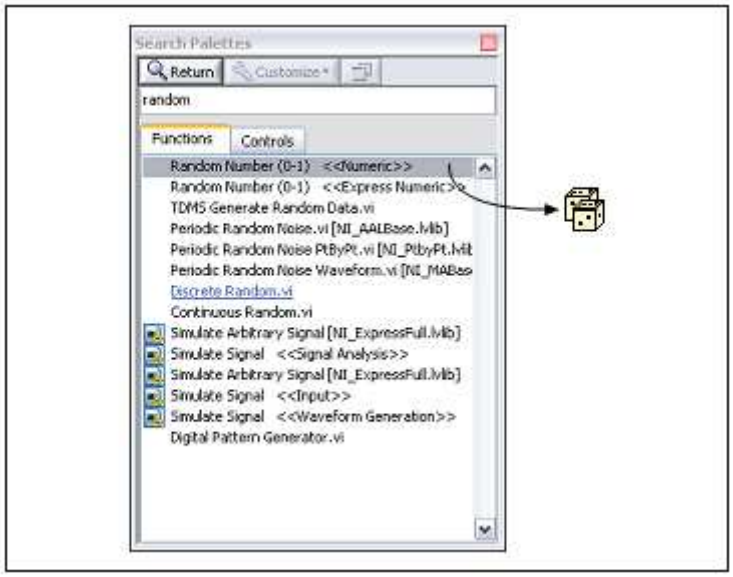
1.41. Vyhľadávanie kontroliek, VI a funkcií

Keď si zvolíte View»Controls alebo View»Functions k zobrazeniu palety Controls a Functions, zobrazia sa dve tlačidlá v hornej časti palety

Search—nastaví paletu do vyhľadávacieho módu, tak, aby ste mohli použiť vyhľadávanie pomocou textu na lokalizáciu kontroliek, VI a funkcií. Kliknite na tlačidlo Return ak chcete ukončiť vyhľadávanie a vrátiť na pôvodnú paletu z vyhľadávacieho módu.

Customize—poskytuje možnosti pre výber formátu aktuálnej palety, na zobrazenie a skrytie kategórií pre všetky palety, a usporiadanie položiek v Textovom a Stromovom formáte, alfabeticky. Zvoľte položku Options z menu na zobrazenie strany Controls/Functions Palettes dialógového oknaOptions . Tu si môžete zvoliť formát pre všetky palety. Toto tlačidlo je viditeľné iba ak kliknete na ikonu špendlíka na ľavej hornej strane palety.

Pokým nie ste oboznámený s umiestnením VI a funkcií, použite vyhľadávanie pomocou tlačidla Search. Napríklad, ak chcete nájsť funkciu Náhodného čísla, kliknite na tlačidlo Search na palete Functions a zadajte Random Number do vyhľadávacieho pola v hornej časti palety. LabVIEW zobrazí všetky zhodujúce sa položky ktoré začínajú daným textom alebo obsahujú daný text. Po vyhľadaní kliknite na výsledok vyhľadávania a premiestnite objekt na blokový diagram.



Vyhľadávanie objektu na palete funkcií

Dvojklikom na výsledok vyhľadávanie zobrazíte umiestnenie objektu na palete. Ak daný objekt potrebujete často používať, môžete si ho pridať do kategórie Favorites (Obľúbené). Kliknite pravým tlačidlom myši na objekt v palete a vyberte.

1.42. Výber nástroja

Vytvárať, modifikovať a ladiť VI môžete pomocou nástrojov v LabVIEW. Nástroj je špeciálna funkcionalita kurzora myši. Táto funkcia korešponduje ikone pomocou ktorej je daný nástroj zvolený. LabVIEW určí ktorý nástroj zvoliť podľa aktuálneho umiestnenia myši.

Paleta nástrojov

1.43. Nástroj Operating tool

Keď sa kurzor myši zmení na ikonu ako je zobrazené naľavo, nástroj Operating tool je je aktívny. Nástroj Operating tool používajte na zmenu hodnôt kontrolky. Nástroj Operating tool sa automaticky aktivuje keď sa myšou zastavíte nad ukazovateľom.Nástroj Operating tool sa väčšinou používa na čelnom panely, ale taktiež ho môžete použiť na blokovom diagrame pri zmene hodnôt boolean konštánt.

1.44. Nástroj Positioning Tool

Keď sa kurzor myši zmení na ikonu ako je zobrazené naľavo, nástroj Positioning tool je je aktívny. Tento nástroj používate na výber, označenie a zmenu veľkosti objektov. Po výbere objektu ho môžete posúvať, kopírovať a vymazať. Nástroj Positioning tool sa automaticky aktivuje keď sa myšou prenesiete nad hranu objektu. Nástroj Positioning tool môžete použiť na čelnom panely ako aj na blokovom diagrame.

1.45. Nástroj Labeling tool

Keď sa kurzor myši zmení na ikonu ako je zobrazené naľavo, nástroj Labeling tool je je aktívny. Nástroj Labeling tool použite na vloženie textu do kontrolky, na zmenu textu, a na tvorbu voľných návestí. Nástroj Labeling tool sa automaticky aktivuje keď sa myšou prenesiete do vnútorného priestoru kontrolky. Jedným klikom vložíte kurzor do vnútra kontrolky. Následne dvojklikom označíte aktuálny text. Keď kurzor nie je nad špecifickým objektom čelného panelu alebo blokového diagramu ktorý by mohol aktivovať niektorý z módov/nástrojov, tak má tvar kríža. V prípade, ak automatický výber nástroja je zapnutý, dvojklikom v prázdnom priestore zaktivujete nástroj Labeling tool na tvorbu voľných návestí.

1.46. Nástroj Wiring Tool

Keď sa kurzor myši zmení na ikonu ako je zobrazené naľavo, nástroj Wiring tool je je aktívny. Nástroj Wiring tool použite na spájanie objektov v blokovom diagrame Nástroj Wiring tool sa stane automaticky aktívnym ak kurzor posuniete nad vstupný alebo výstupný bod terminálu alebo nad spojenie. Nástroj Wiring tool používate v blokovom diagrame na prepojenie objektov a pri tvorbe konektorového panelu v okne čelného panelu.

1.47. Ostatné nástroje z palety nástrojov

Nástroje Operating, Positioning, Labeling, a Wiring tool môžete zaktivovať aj priamo z palety Tools, namiesto automatického výberu nástroja. Paletu Tools zobrazíte pomocou voľby **View»Tools Palette**

Paleta nástrojov

Prvou položkou v palete Tools je tlačidlo automatického výberu nástrojov. V prípade zapnutého stavu LabVIEW automaticky vyberie nástroj na základe pozície kurzora. Voľbu automatického výberu nástrojov deaktivujete ak vypnutým tlačidlom alebo pomocou výberu inej položky z palety. V palete sa nachádzajú aj ďalšie nástroje:

Použite nástroj Object Shortcut Menu tool k prístupu kontextovej ponuke objektu pomocou ľavého tlačidla myši

Použite nástroj Scrolling tool na posun okna bez použite rolovacieho pruhu

Použite nástroj Breakpoint tool na nastavenie prerušení u VI, funkciách, uzloch, spojení a štruktúr

Použite nástroj Probe tool na tvorbu sond na spojeniach v blokovom diagrame Nástroj Probe tool použijete na kontrolu dočasných hodnôt vo VIktoré vykazuje otázne alebo neočakávané výsledky

Použite nástroj Color Copy tool na kopírovanie farieb pre účely nástroja Coloring tool.

Použite nástroj Coloring tool na zafarbenie objektu. Nástroj Coloring tool taktiež zobrazí aktuálne farebné nastavenia pozadia a popredia.

1.48. Tok dát

LabVIEW vykonáva VI na základe modelu toku dát. Uzol na blokovom diagrame sa vykoná až keď má prístupné všetko požadované vstupy. Po ukončení vykonania, uzol vytvorí výstupné dáta ktoré sa prenesú do ďalšieho uzla v toku dát. Pohyb dát cez uzly určí poradie vykonávania VI a funkcií v blokovom diagrame.

Visual Basic, C++, JAVA a väčšina iných textovo orientovaných programovacích jazykov sa vykonáva na základe toku kontroly. V takejto schéme sa poradie vykonávania určí podľa poradia príkazov v sekvencii.

1.49. Vývoj jednoduchého VI

Väčšina LabVIEW VI má tri hlavné úlohy - zber dát, analýza a následné zobrazenie výsledkov. V prípade ak všetky tieto úlohy sú jednoduché, celé VI môžete poskladať pomocou malého počtu objektov na blokovom diagrame. Express VI sú navrhnuté špecificky k bežným a často používaným činnostiam. Táto sekcia pojednáva o niektorých Express VI na zber, analýzu a zobrazovanie dát.

Na palete **Functions** sú Express VIs zhromaždené v kategórii **Expressn.** Express VI používajú dynamický dátový typ na prenos údajov medzi Express VI.

1.50. Zber dát

Express VI používané pre zber dát nahrňujú nasledovné VI: DAQ Assistant, Instrument I/O Assistant, Simulate Signal, a Read from Measurement File.

DAQ Assistant

DAQ Assistant nadobudne dáta cez zariadenie na zber dát. Počas kurzu budete toto Express VI používať často. Pokým sa nenaučíte viac ohľadne akvizície dát, budete používať iba jeden kanál,CH0, zariadenia na zber dát. Tento kanál je pripojený na teplotný senzor v DAQ Signal Accessory. Pomocou dotyku teplotného senzora môžete meniť snímané hodnoty.

Instrument I/O Assistant

Instrument I/O Assistant vykoná zber dát z prístrojov, obyčajne pripojených cez rozhranie GPIB alebo serial.

Simulate Signal

Simulate Signal Express VI vytvorí simulované dáta, ako napríklad sínusový signál.

Read From Measurement File

Read From Measurement File Express VI načíta súbor ktorý bol vytvorený pomocou Write To Measurement File Express VI. Dokáže načítať LVM a TDMS formáty súborov. Toto Express VI nenačíta ASCII súbory.

1.51. AnalýzaExpress

VI používané pre analýzu zahrňujú nasledovné - Amplitude and Level Measurements, Statistics, Tone Measurements a pod.

Amplitude and Level Measurements

Amplitude and Level Measurements Express VI uskutoční merania napätí na signály. Tieto merania zahrňujú jednosmernú zložku, efektívnu hodnotu, maximálny vrchol, minimálny vrchol, priemer cyklu, efektívnu hodnotu cyklu.

Statistics

Statistics Express VI vypočíta štatistické dáta priebehu. Tieto zahrňujú strednú hodnotu, súčet, štandardnú odchýlku, a extrémne hodnoty.

Spectral Measurements

Spectral Measurements Express VI urči parametre spektra priebehu, ako napríklad magnitúda, spektrálna hustota.

Filter

Filter Express VI aplikuje na signál filtre a okná. Použité filtre zahrňujú: horno priepustný, dolno priepustný, pásmový priepustný,a pásmovo zádržný. Typy okien zahrňujú Butterworth, Chebyshev, inverzný Chebyshev, Elliptical, a Bessel.

1.52. Zobrazenie

Výsledky môžete zobraziť pomocou Express VI ktoré vykonajú nejakú funkciu ako napríklad Write to Measurement File Express VI, alebo pomocou indikátorov na zobrazenie dát v čelnom panely. Najbežnejšie používané indikátory pre tento účel sú Waveform Chart, Waveform Graph, a XY Graph. Častou používané Express VI zahŕňajú Write to Measurement File Express VI, Build Text Express VI, DAQ Assistant,a Instrument I/O Assistant. V tomto prípade DAQ Assistant a Instrument I/O Assistant poskytujú výstupné dáta z počítača na DAQ zariadenie alebo do externého pristroja.

Write to Measurement File

Write to Measurement File Express VI zapíše dáta do LVM or TDMS súborového formátu.

Build Text

Build Text Express VI vytvorí text, obyčajne pre účely zobrazenia na čelnom panely alebo exportovania do súboru či prístroja.

1.53. Spustenie VI

Po konfigurácii Express VI a vytvorení prepojení môžete spustiť VI. Po ukončení vytvárania, VI spustíte klikom na tlačidlo Run na nástrojovej lište .

Počas behu VI ikona Run sa zmení na tvar ktorí je zobrazený na obrázku vľavo . Po ukončení behu tlačidlo Run sa zmení späť na pôvodný stav a indikátory čelného panelu budú obsahovať dáta.

1.54. Chyby VI

Ak sa VI nedá spustiť tak sa jedná o nefunkčné, nespustiteľné VI. Tlačidlo Run sa zobrazí ako zlomená šípka keď VI ktoré vytvárate alebo meníte obsahuje chyby.

Ak je tlačidlo Run po ukončení editovania blokového diagramu stále v tvare zlomenej šípky, znamená to, že VI nie je spustiteľné.

Vo všeobecnosti to znamená, že požadovaný vstup nie je zapojený, alebo spojenie je porušené. Zobrazte okno Error list kliknutím na tlačidlo Run (v tvare zlomenej šípky). Okno Error list obsahuje zoznam chýb a opisuje problém. Dvojklikom na chybu sa nastavíte priamo na problematický bod.

2. RIEŠENIE PROBLÉMOV A LADENIE APLIKÁCIÍ

K tomu, aby VI bolo spustiteľné musíte prepojiť všetky subVI, funkcie, štruktúry so správnymi dátovými typmi. Niekedy sa môže stať, že VI produkuje dáta alebo beží spôsobom akým by ste to neočakávali. LabVIEW môžete použiť na konfiguráciu behu VI ako aj na identifikáciu problémov s blokovým diagramom alebo dát ktoré blokovým diagramom prechádzajú.

2.1. Pomocné nástroje LabVIEW Help

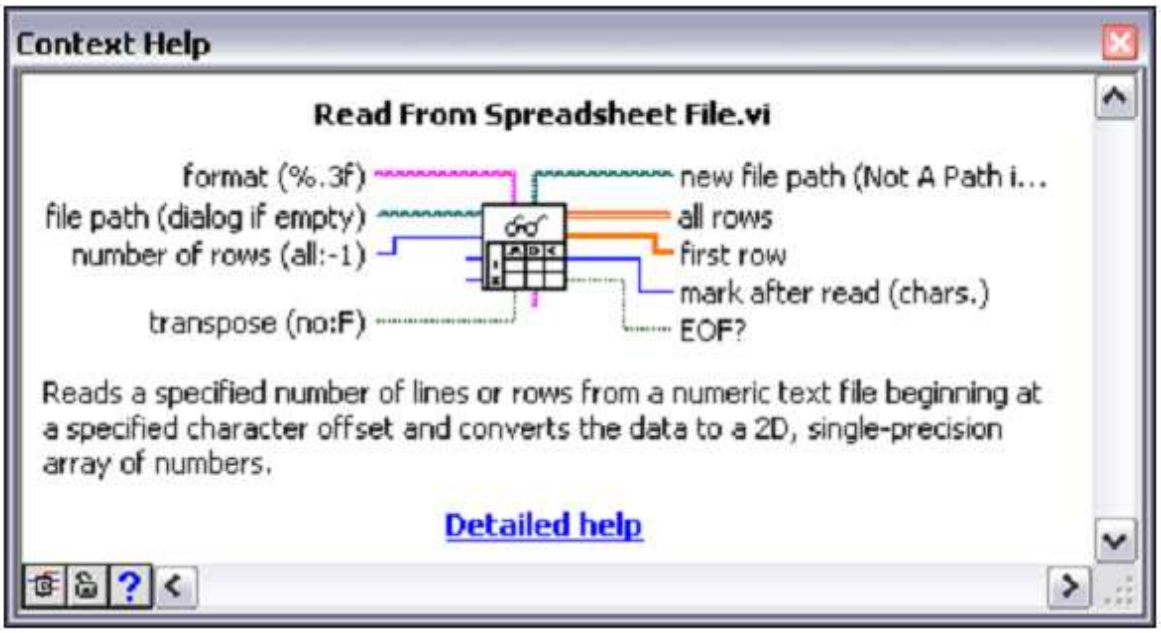
Použite **Context Help**, LabVIEW Help, a NI Example Finder na pomoc pri tvorbe a zmene VI. Pre viac informácií o samotnom LabVIEW si pozrite LabVIEW Help a dostupné manuály.

2.2. Context Help

Okno **Context Help** zobrazuje základné informácie ohľadne LabVIEW objektov pri posune kurzora myši nad objekt. Na zobrazenie okna **Context Help** použite voľbu **Help»Show Context Help**, klávesovú skratku <Ctrl-H>, alebo kliknite na tlačidlo **Show Context Help Window** na nástrojovej lište.

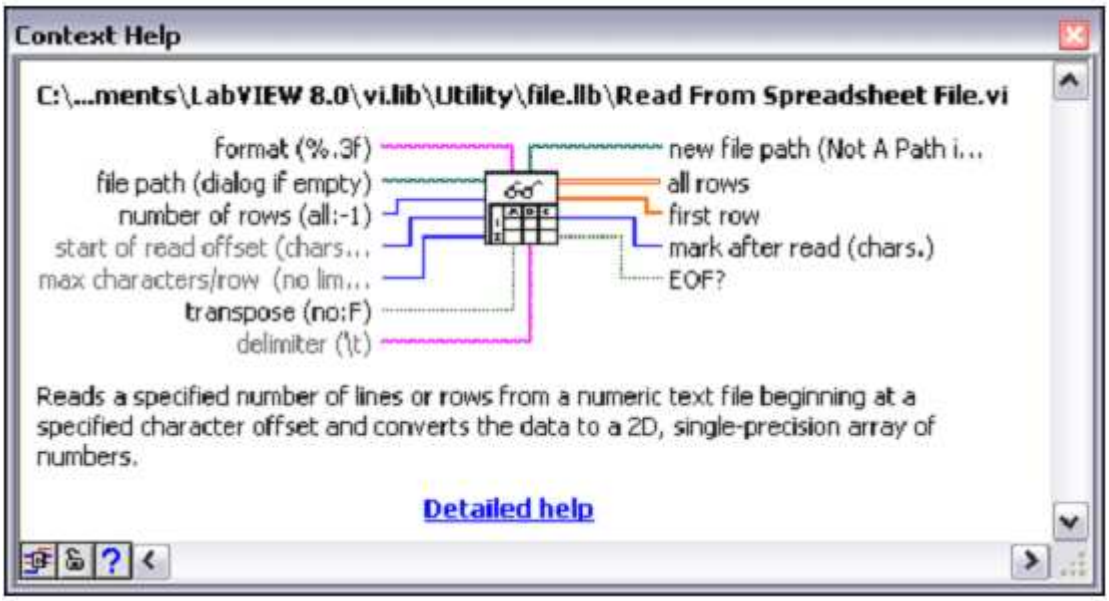
Pri presune kurzora myši na objekty na čelnom panely alebo blokovom diagrame okno **Context Help** zobrazí ikony pre subVI, funkcie, konštanty, kontrolky a indikátory spolu so spojeniami pre každý terminál. Pri presune kurzora myši nad jednotlivé možností nastavenia v rámci dialógových okien, okno **Context Help** zobrazí opis týchto možností.

V okne **Context Help** sú návestia požadovaných terminálov zobrazené zvýrazneným, doporučené terminály obyčajným, a opcionálne terminály nezvýrazneným typom písma. Návestia opcionálnych terminálov sa nezobrazia ak kliknete na tlačidlo **Hide Optional Terminals and Full Path** v okne **Context Help**.



Context Help

Na zobrazenie opcionálnych terminálov konektorového panelu a na zobrazenie celej cesty k VI kliknite na tlačidlo Show Optional Terminals and Full Path v ľavom dolnom rohu okna Context Help. Opcionálne terminály sú zobrazené pomocou otvorených spojení.



Detailný Context Help

Použite tlačidlo Lock Context Help Window na uzamknutie aktuálneho obsahu okna Context Help. Pri uzamknutom obsahu, presun kurzora nad iný objekt nespôsobí zmenu obsahu okna. Na odomknutie okna stlačte

znovu tlačidlo. Túto možnosť máte taktiež prístupnú z ponuky Help.

Ak existuje korešpondujúca téma v LabVIEW Help pre objekt opisovaný v okne Context Help, tak sa vám zobrazí modrý odkaz Detailed helpv okne Context Help. Zároveň je tlačidlo More Help je aktívne. Kliknite na odkaz alebo na tlačidlo k zobrazeniu LabVIEW Help pre viac informácií ohľadne daného objektu.

2.3. LabVIEW Help

LabVIEW Help si môžete zobraziť pomocou tlačidla More Help v okne Context Help, voľbou Help»LabVIEW Help, alebo kliknutím na modrý odkaz Detailed Help v okne Context Help. Ďalšou možnosťou je pravým tlačidlom myši kliknúť na objekt a zvoliť Help z kontextovej ponuky.

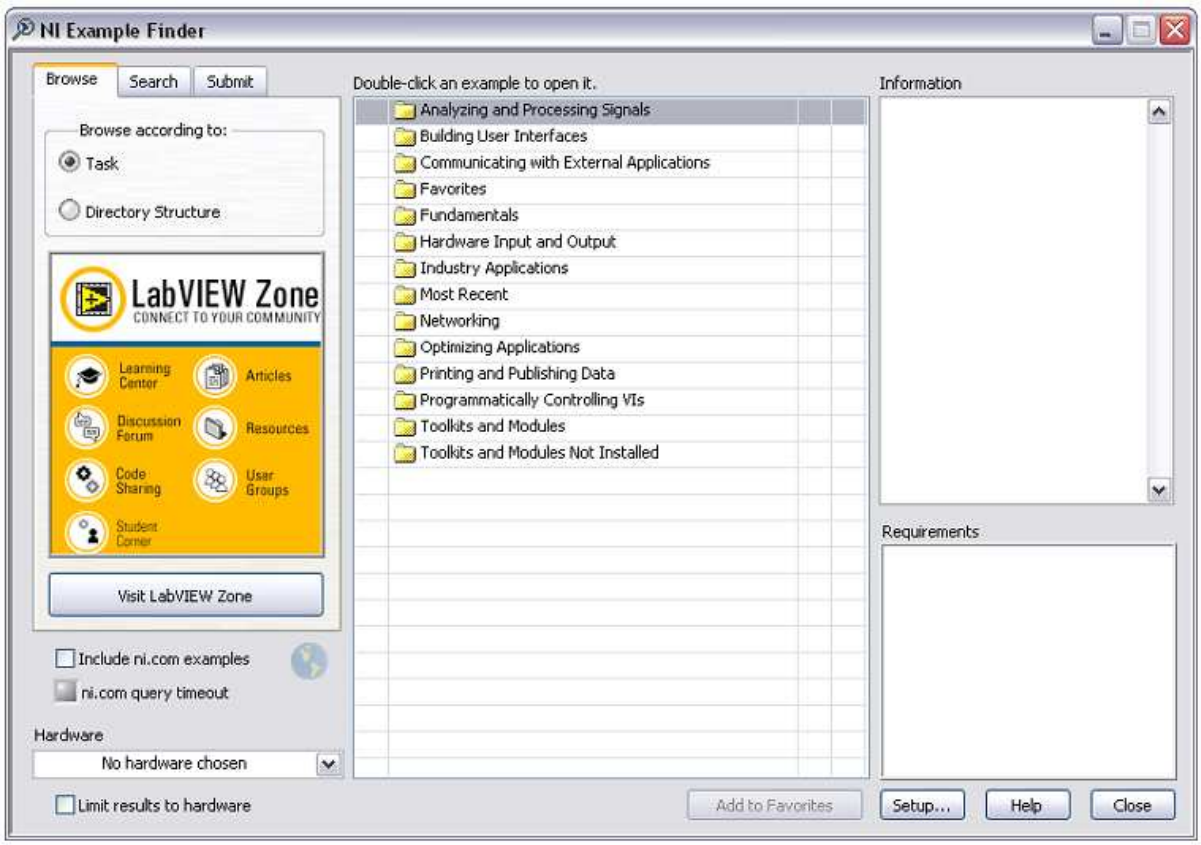
LabVIEW Help obsahuje detailný opis väčšiny paliet, položiek, nástrojov, VI, a funkcií. LabVIEW Help takisto obsahuje detailné inštrukcie na používanie možností LabVIEW. LabVIEW Help obsahuje odkazy na nasledovné zdroje:

- LabVIEW Documentation Resources, ktoré opisujú elektronické a tlačené verzie dokumentov na pomoc novým aj skúseným používateľom a obsahujú PDF verzie všetkých LabVIEW manuálov.
- Zdroje technickej podpory na stránkach National Instruments, ako napríklad NI Developer Zone, KnowledgeBase, a Product Manuals Library.

2.4. NI Example Finder

Použite NI Example Finder na vyhľadávanie príkladov ktoré sú nainštalované na vašom počítači alebo na NI Developer Zone, ni.com/zone. Tieto príklady znázorňujú ako používať LabVIEW na vykonanie širokého okruhu testovacích, meracích, riadiacich a návrhových úloh. NI Example Finder spustíte pomocou voľby Help»Find Examples alebo kliknite na odkaz Find Examples v sekcii Examples v okne Getting Started.

Príklady poukážu na použite špecifických VI a funkcií. Pravým klikom myši na funkciu na blokovom diagrame a následnou voľbou Examples z kontextovej ponuky vyvoláte pomocníka s odkazmi na príklady pre danú funkciu. Môžete modifikovať už existujúci príklad tak aby vám vyhovoval, alebo môžete skopírovať jeden alebo viacero príkladov do VI ktoré vytvárate.



2.5. Oprava nefunkčného VI

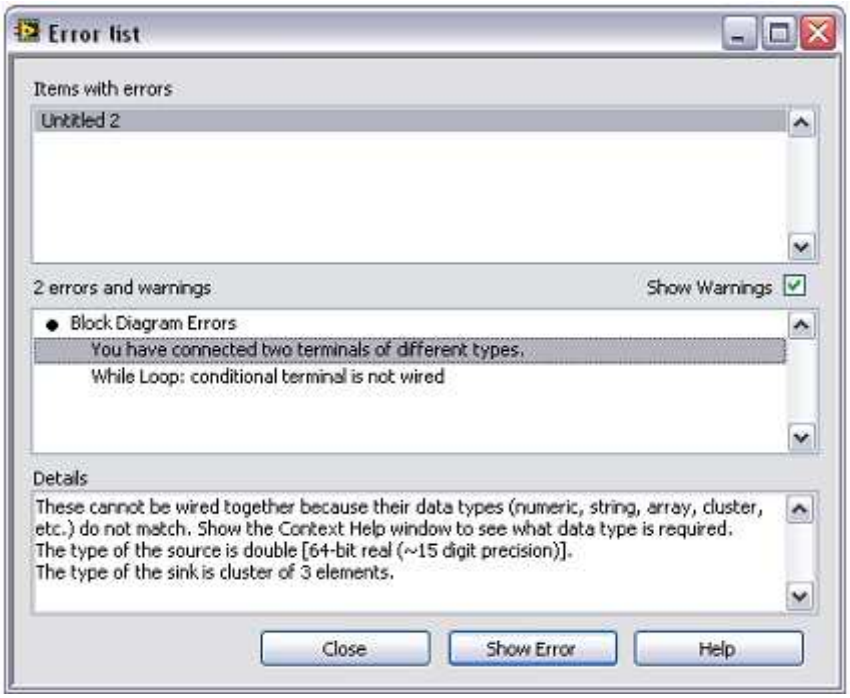
Ak sa VI nedá spustiť tak sa jedná o nefunkčné, nevykonávateľné VI. Tlačidlo Run sa zobrazí ako zlomená šípka keď VI ktoré vytvárate alebo meníte obsahuje chyby.
Ak je tlačidlo Run po ukončení editovania blokového diagramu stále v tvare zlomenej šípky, znamená to, že VI nie je spustiteľné.

2.6. Hľadanie zdrojov chýb

Upozornenia (Warnings) nebránia spusteniu VI. Sú navrhnuté pre notifikáciu na možné problémy. Na rozdiel od upozornení, chyby zabraňujú spusteniu VI. K spusteniu VI musíte mať všetky chyby odstránené.

Kliknite na zlomenú šípku Run alebo zvoľte View»Error List na zistenie dôvodov chyby. Okno Error list obsahuje zoznam všetkých chýb. Sekcia Items with errors obsahuje zoznam všetkých položiek načítaných v pamäti, ako napr. VI a projektové knižnice, ktoré obsahujú chyby. Ak dva alebo viac položiek má totožný názov, táto sekcia zobrazuje špecifickú inštanciu aplikácie pre každú položku. Sekcia errors and warnings obsahuje zoznam všetkých chýb a upozornení pre VI ktoré vyberiete v sekcii Items with errors. Sekcia Details opisuje chyby a v niektorých prípadoch aj odporučenia k oprave chýb. Kliknite na tlačidlo Help k zobrazeniu témy v LabVIEW Help ktorá detailne opisuje chybu a obsahuje inštrukcie na opravu chyby.

Kliknite na tlačidlo Show Error alebo dvojklikom na opis chyby k zvýrazneniu sekcie na blokovom diagrame alebo čelnom panely ktorá obsahuje chybu.



Príklad na dialógové okno Error List

2.7. Bežné zdroje chýb

Nasledovný zoznam obsahuje časté dôvody nefunkčných VI počas ich editácie:

- Blokový diagram obsahuje nefunkčné spojenia kvôli nekompatibilným dátovým typom alebo neukončeným spojeniam. Viac informácií ohľadne oprave nefunkčných spojení nájdete v téme Correcting Broken Wires v pomocníkovi LabVIEW Help.
- Požadovaný terminál blokového diagramu nie je pripojený Viac informácií ohľadne nastavenia požadovaných vstupov a výstupov nájdete v téme Using Wires to Link Block Diagram Objects v pomocníkovi LabVIEW Help.
- SubVI je nefunkčné alebo došlo k zmene konektorového panelu po jeho použití v blokovom diagrame volajúceho VI. Viac informácií ohľadne subVI nájdete v téme Creating SubVIs v pomocníkovi LabVIEW Help.

2.8. Ladiace techniky

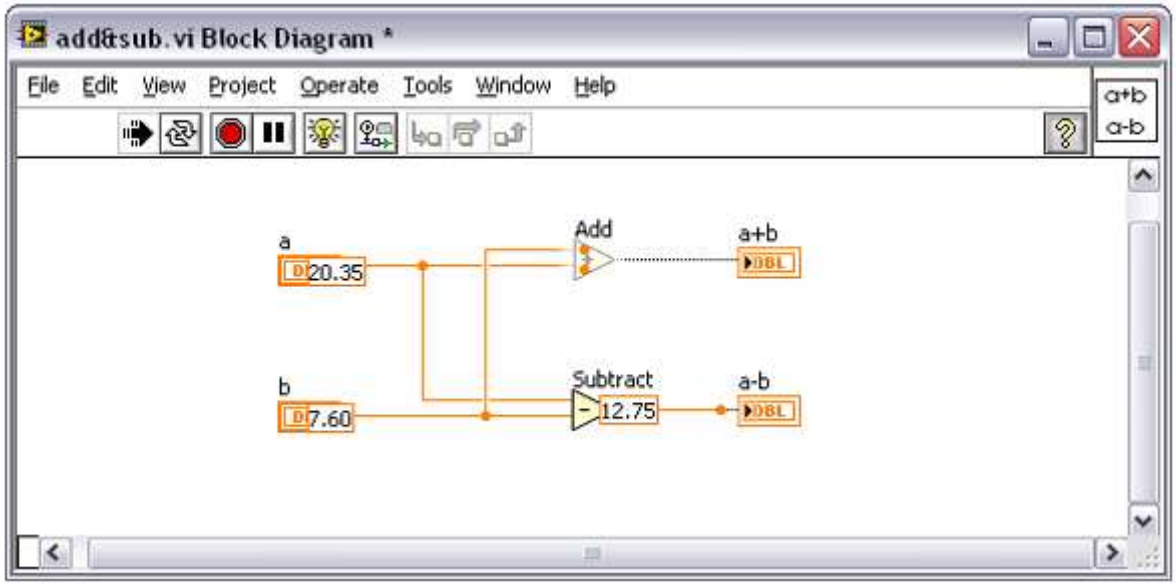
- V prípade, že VI je spustiteľné, ale výsledkom sú neočakávané dáta, môžete použiť nasledujúce techniky na identifikáciu a opravu problémov v rámci VI:
- Prepojte error in a error out parametre VI a funkcií. Tieto parametre detekujú chyby ku ktorým mohlo dôjsť pre každý uzol v blokovom diagrame a indikujú či a kde chyba nastala. Tieto parametre môžete taktiež použiť vo vami vyvinutými VI.
- Na elimináciu upozornení a zobrazenie upozornení zvoľte View»Error List a zaškrtnite políčko Show Warnings. Zistite dôvody chýb a upozornení a opravte ich vo VI.

- Na zvýraznenie celého prepojenia použite nástroj Positioning tool s trojklikom na prepojenie.
- Použite Context Help na kontrolu predvolených hodnôt pre každú funkciu a subVI na blokovom diagrame. VI a funkcie prenášajú prednastavené hodnoty ak doporučené alebo požadované vstupy sú nezapojené. Napríklad, boolean vstup môže byť nastavený na TRUE ak je nezapojený.
- Použite Find dialógové okno na vyhľadávanie subVI, textu a iných objektov v rámci VI.
- Zvoľte View»VI Hierarchy k vyhľadaniu neprepojených subVI. Na rozdiel od nezapojených funkcií, nezapojené VI generujú chybu len za prípadu, ak nie sú zapojené ich požadované vstupy. Ak omylom vložíte nezapojené subVI do blokového diagramu, subVI sa vykoná spolu s blokovým diagramom. Toto VI môže vykonať vedľajšie akcie.
- Použite voľbu animácie vykonávania kódu na získanie prehľadu o pohybe dát.
- Jednoduchým krokovaním môžete preskúmať každý uzol na blokovom diagrame.
- Použite sondu na pozorovanie aktuálnych dát a na kontrolu chybových výstupov VI a funkcií, obzvlášť pre V/V.
- Na uchovanie prenesenej hodnoty pozdĺž prepojenia kliknite na Retain Wire Values. Táto funkcionalita umožní jednoduchú kontrolu naposledy prenesených hodnôt.
- Použite breakpoint na pauzu vo vykonávaní kódu a následne využite možnosť krokovania alebo použitia sond.Prerušte vykonávanie subVI ak chcete zmeniť hodnoty kontroliek a indikátorov, pri zmene počtu cyklov, alebo ak sa chcete vrátiť za začiatok vykonávania kódu v subVI.
- Určte, či dáta ktoré subVI alebo funkcia používa sú definované. Nedefinované dáta sú časté pri práci s číslami. Napríklad, v nejakom bode vykonávania, VI mohlo vykonať operáciu delenia s hodnotou 0, a vrátiť návratovú hodnotu Inf (infinity), zatiaľ čo nasledovné funkcie očakávali číslo.
- Ak VI beží omnoho pomalšie ako je to očakávané, potvrďte, že animácia vykonávania kódu je vypnutá. Zároveň, zavrite čelné panely a blokové diagramy subVI keď ich nepoužívate, nakoľko otvorené okná môžu mať vplyv na rýchlosť vykonávania kódu.
- Skontrolujte reprezentáciu kontroliek a indikátorov kvôli možnému pretečeniu ktoré by mohlo nastať ak ste skonvertovali číslo s pohyblivou rádovou čiarkou na integer alebo integer na ešte menší integer. Napríklad, keby ste prepojili výstup 16-bit integer do funkcie so vstupom pre 8-bit integer. Takéto prepojenie by malo za dôsledok konverziu z 16-bit na 8-bit reprezentáciu, čo by mohlo viesť k potencionálnej strate dát.
- Zistite, či sa nachádzajú v kóde For cykly ktoré sa nevykonajú ani raz a tvoria prázdne polia.
- Uistite sa, že ste správne inicializovali shift registre - pokiaľ nemáte v úmysle prenášať dáta medzi jednotlivými volaniami VI.
- Skontrolujte poradie prvkov clustra pri zdrojových aj cieľových bodoch. LabVIEW zistí nezhody medzi dátovými typmi a rozmerom clustra pri editovaní kódu, ale nezistí nezhody medzi prvkami toho istého typu.
- Skontrolujte poradie vykonávania uzlov.

- Skontrolujte, že VI neobsahuje skryté subVI. Z nedbanlivosti ste mohli ukryť subVI tak, že ste ho prekryli s iným subVI alebo zmenou veľkosti štruktúry bez ponechania subVI v dohľade.
- Skontrolujte inventár subVI ktoré sú použité vo VI a porovnajte to s výsledkami voľby View»Browse Relationships»This VI's SubVIs a View»Browse Relationships»Unopened SubVIs. Zároveň si otvorte okno VI Hierarchy na zobrazenie subVI pre dané VI. Navrhnite vstupy VI ako požadované vstupy aby ste sa vyhli nesprávnym výsledkom spôsobených ukrytými VI.

2.9. Animácia vykonávania kódu

Pre zobrazenie animácie vykonávania kódu v blokovom diagrame použite tlačidlo Highlight Execution.
Animácie vykonávania znázorňuje tok dát na blokovom diagrame medzi uzlami pomocou návestí ktoré sa pohybujú po spojeniach. Použite animáciu vykonávania v spolupráci s krokovaním na pozorovanie toku dát medzi uzlami.
V uzloch MathScript Nodes, (MathScript RT Module), animácia vykonávania zobrazuje progres behu kódu pomocou modrej šípky ukazujúcej na aktuálne vykonávaný riadok kódu.



Príklad na použitie animácie vykonávania kódu

2.10. Krokovanie

Jednoduchým krokovaním môžete preskúmať každý uzol na blokovom diagrame. Tlačidlá pre krokovanie ovplyvnia vykonávanie iba v rámci VI alebo subVI v móde krokovania.

Krokovanie spustíte pomocou tlačidiel Step Into alebo Step Over na blokovom diagrame. Presuňte kurzor myši nad tlačidlá Step Into, Step Over, alebo Step Out na zobrazenie návestia ktoré zobrazuje ďalší krok ktorý sa vykoná ak kliknete na to tlačidlo. Môžete krokovať cez VI alebo ho spustiť obvyklým spôsobom.

Pri krokovaní uzly bliknú keď sú pripravené k vykonaniu. Pri krokovaní so zapnutou animáciou vykonávania sa zobrazí glyf na ikone aktuálne vykonávaného subVI.

2.11. Sonda

Použite sondu ak máte komplikovaný blokový diagram so viacerými operáciami ktoré môžu vracať nesprávne hodnoty. Použite sondu spolu s animáciu vykonávania kódu, krokovaním a s použitím prerušení (breakpoints) na zistenie či a prípadne kde sú dáta nesprávne. Ak sú dáta
dostupné, sonda okamžite zaktualizuje a zobrazí dáta v okne Probe WatchWindow počas animácie vykonávania kódu, krokovania, alebo pri pauze pri prerušení (breakpoint). Keď vykonávanie kódu je pozastavené kvôli krokovaniu alebo prerušeniu (breakpoint), môžete použiť sondu na zistenie hodnoty dát na spojení ktoré sa vykonalo ako posledné.

2.12. Typy sond

Na kontrolu aktuálnych hodnôt na spojení počas behu VI môžete použiť generickú sondu, indikátor z Controls palety, predom dodanú sondu alebo vytvoriť novú sondu.

2.12.1. *Generické*

Použite generickú sondu na zobrazenie dát prenášaných spojením. Pravým klikom myši na spojenie zobrazte kontextovú ponuku a zvoľte Custom Probe»Generic Probe.

Generická sonda zobrazí dáta. Generická sonda sa nedá nakonfigurovať tak, aby reagovala na dáta.

LabVIEW zobrazí generickú sondu keď pravým klikom na spojenie zvolíte Probe, okrem prípadu, keď už máte definovanú na mieru šitú alebo predom dodanú sondu pre dátový typ.

Na mieru šitú sondu môžete ladiť podobne ako VI. Avšak, sonda sa nemôže použiť na skúmanie vlastného blokového diagramu, ani blokového diagramu použitej subVI. Pri ladení sond použite generickú sondu.

2.13. Použitie indikátorov na zobrazenie dát

Na zobrazenie prenášaných dát cez vodič môžete použiť indikátor. Napríklad, na získanie prehľadu o numerických dátach môžete použiť graf. Pravým klikom myši na spojenie zobrazte kontextovú ponuku a zvoľte Custom Probe»Controls. Môžete zvoliť aj ikonu Select a Control na palete Controls a následne zvoliť ľubovoľnú kontrolku alebo typovú definíciu uloženú na počítači alebo v zdielanom adresári. LabVIEW pristupuje k typovým definíciám podobne ako ku vlastným kontrolkám keď ich používate na zobrazenie sondovaných dát.

Ak sa dátový typ vybraného indikátora nezhoduje s dátovým typom vybraného spojenia tak LabVIEW neumiestni indikátor na spojenie.

2.13.1. *Dodané*

Dodané sondy sú VI ktoré zobrazia súhrnné informácie o dátach ktoré sú prenášané cez spojenie. Napríklad, sonda VI Refnum vráti informácie o názve VI, ceste k VI a hexadecimálnej hodnote referencie. Dodaná sonda sa dá použiť tak aby reagovala na prenášané dáta cez spojenie. Napríklad, použite sondu Error pri error clusteri na zobrazenie statusu, kódu, zdroja, a opisu chyby a špecifikujte, či chcete nastaviť podmienený breakpoint ak dôjde k chybe alebo k výstrahe (warning).

Dodané sondy sú k dispozícii na vrchnej časti kontextovej ponuky Custom Probe. Pravým klikom myši na spojenie zobrazte kontextovú ponuku a zvoľte Custom Probe. Iba tie sondy, ktoré sa zhodujú s dátovým typom spojenia sa objavia v kontextovej ponuke.

2.13.2. *Vlastné*

Použite dialógové okno Create New Probe na vytvorenie sondy na základe existujúcej sondy alebo úplne novej sondy. Pravým klikom myši na spojenie zobrazte dialógové okno Create New Probe pomocou kontextovej ponuky a voľby Custom Probe»New. Vytvorte novú sondu ak chcete mať viac kontroly nad tým ako LabVIEW sonduje dáta ktoré tečú cez spojenia. Keď vytvoríte novú sondu, dátový typ sondy sa bude zhodovať s dátovým typom linky na ktorú ste klikli pravým. Ak chcete editovať existujúcu sondu, musíte ju otvoriť z adresára kde ste si ju uložili.

Po výbere sondy z kontextovej ponuky Custom Probe, vyberte ju pomocou Select a Control opcii na palete, alebo vytvorte novú sondu pomocou dialógového okna Create New Probe. Táto sonda sa stane predvolenou sondou pre daný dátový typ, a LabVIEW načíta túto sondu pri pravom kliku na prepojenie a voľby Probe z kontextovej ponuky. LabVIEW načíta iba tie sondy ktorých dátový typ sa presne zhoduje s dátovým typom spojenia pri pravom kliku. Tj., sonda ktorá má dátový typ double sa nedá použiť na zobrazenie hodnôt typu 32-bit unsigned integer na spojení, ak keď tieto dátové typy LabVIEW dokáže skonvertovať.

2.14. Prerušenia

Použite nástroj Breakpoint na vloženie prerušenia do VI, uzla, na spojenie a vykonanie pauzy na danom mieste.
Keď vložíte prerušenie (breakpoint) na spojenie, vykonávanie kódu sa preruší a tlačidlo Pause sa označí červeným po tom, čo dáta pretiekli spojením. Vložte prerušenie na blokový diagram k nastaveniu prerušenia vykonávania po tom, čo sa všetky uzly vykonali na blokovom diagrame. Hrany blokového diagramu sa zafarbia na červeno a budú blikať - takto sa
zvýrazní umiestnenie prerušenia.
Keď VI sa zastaví pri prerušení, LabVIEW prenesie fókus na blokový diagram a zvýrazní uzol, spojenie alebo riadok skriptu ktorý obsahuje breakpoint. Keď presuniete kurzor nad existujúce prerušenie, tak čierna oblasť nástroja Breakpoint tool sa zafarbí na bielo.

Keď vykonávanie kódu dosiahne bod prerušenia, VI sa preruší a tlačidlo Pause za zobrazí v červenej farbe. Môžete vykonať nasledovné akcie:

- Krokovanie pomocou tlačidiel na krokovanie.
- Použiť sondu na kontrolu aktuálnych hodnôt.
- Zmeniť hodnoty kontroliek na front panely.
- Kliknite na tlačidlo Pause na pokračovanie vo vykonávaní až do ďalšieho prerušenia alebo až pokým VI neukončí beh.

2.15. Zastavenie vykonávania

Prerušte vykonávanie subVI ak chcete zmeniť hodnoty kontroliek a indikátorov, počet cyklov vykonávania subVI, alebo ak sa chcete vrátiť za začiatok vykonávania kódu v subVI. Môžete nastaviť všetky volania subVI na začatie vykonania subVI so zastavením, alebo môžete pozastaviť vykonávanie subVI pri špecifickom volaní.

Na začatie vykonania subVI so zastavením otvorte subVI a zvoľte Operate»Suspend when Called. SubVI automaticky zastaví vykonávanie pri volaní z iného VI. Pri voľbe tejto možnosti počas krokovania subVI nezastaví vykonávanie okamžite. SubVI zastaví vykonávanie keď je volané.

K zastaveniu vykonávania špecifického subVI, kliknite pravým tlačidlom myši na blokový diagram a z kontextovej ponuky zvoľte SubVI Node Setup. Zaškrtnite políčko Suspend when called k zastaveniu vykonávania iba pre danú inštanciu subVI.

Okno VI Hierarchy ktoré zobrazíte voľbou View»VI Hierarchy zobrazí, či VI je pauzované alebo zastavené. Šípka ukazuje, že VI beží regulárne alebo je krokované.

Znak pauzy zobrazuje, že je pauzované alebo zastavené.

Zelený znak pauzy, alebo dutý čierno-biely znak zobrazpozastavené keď bude volané. Červený znak pauzy, alebo jednoliaty čierno-biely znak zobrazuje, že VI je aktuálne pauzované zobrazuje, že VI je pozastavené.

VI môže byť suspendované a pauzované súčasne.

2.16. Zistenie aktuálnej inštancie subVI

Pri pauze subVI, výsuvná ponuka Call list na nástrojovej lište zobrazuje zoznam volajúcich VI. Tento zoznam je rozdielny od zoznamu View»Browse Relationships»This VI's Callers, ktorý zobrazuje všetky volajúce VI nezávisle od toho, či sú aktuálne spustené. Použite ponuku Call list za zistenie aktuálnej inštancie subVI ak blokový diagram obsahuje viac než jednu inštanciu. Pri výbere VI z ponuky Call list sa otvorí prislúchajúci blokový diagram a LabVIEW zvýrazní aktuálnu inštanciu subVI.

Takisto môžete použiť funkciu Call Chain na zobrazenie zoznamu volajúcich VI.

2.17. Nedefinované alebo neočakávané dáta

Nedefinované dáta, ako sú NaN (not a number) alebo Inf (infinity), zrušia platnosť všetkých nasledujúcich operácií. Operácie s pohyblivou rádovou čiarkou vracajú nasledovné symbolické hodnoty na vyjadrenie chybných výpočtov alebo nezmyselných výsledkov.

- NaN (not a number) reprezentuje dáta s pohyblivou rádovou čiarkou ktoré sú výsledkom neplatných operácií, ako napr. odmocnina zo záporného čísla.
- Inf (infinity) reprezentuje dáta s pohyblivou rádovou čiarkou ktoré sú tvorené platnými operáciami, ako napríklad delenie čísla nulou.

LabVIEW nerobí kontrolu pretečenia alebo podtečenia pri typu integer. Pretečenie a podtečenie pre čísla s pohyblivou rádovou čiarkou je v súlade s IEEE 754, Standard for Binary Floating-Point Arithmetic.

Operácie s pohyblivou rádovou čiarkou spoľahlivo prenášajú NaN a Inf. Pri explicitnej alebo implicitnej konverzii NaN alebo Inf na integer alebo boolean dátové typy budú hodnoty bezvýznamné. Napríklad, delenie 1 s nulou končí s výsledkom Inf. Výsledkom konverzie Inf na 16-bit integer je hodnota 32,767, ktorá sa javí byť v poriadku.

Pred konverziou dát na dátový typ integer, použite sondu na kontrolu platnosti aktuálnych dát s pohyblivou rádovou čiarkou. Vykonajte kontrolu na NaN pomocou porovnávacej funkcie Not A Number/Path/Refnum?.

Nespoliehajte sa na špeciálne hodnoty ako napríklad NaN, Inf, alebo prázdne pole ak VI produkuje nedefinované dáta. Namiesto toho potvrďte, že VI produkuje definované dáta pomocou výpisu hlásenia o chybe ak nastane situácia v ktorej by sa pravdepodobne vytvorili nedefinované dáta.

Napríklad, ak vytvoríte VI ktoré používa vstupné pole na -auto indexáciu For slučky, určte, čo má VI vykonať ak je vstupné pole prázdne. Môžete vytvoriť výstupnú chybovú hlásku, nahradiť definované dáta s hodnotami vytvorenými slučkou, alebo použiť Case štruktúru ktorá nevykoná For slučku ak pole je prázdne.

2.18. Kontrola chýb a obsluha chýb

Nezávisle na tom, nakoľko ste si istý vo vyvíjanom VI, nemôžete predpovedať každý problém s ktorým sa môže používateľ stretnúť. Bez kontrolného mechanizmu zistíte iba, že VI nefunguje správne. Pomocou kontroly chýb určíte prečo a kde nastala chyba.

2.18.1. *Automatická obsluha chýb*

Každá chyba má numerický kód a prislúchajúce chybové hlásenie.

V súlade s predvoleným nastavením LabVIEW automaticky obslúži ľubovoľnú chybu počas behu VI pomocou prerušenia vykonávania, zobrazenia subVI alebo funkcie kde nastala chyba, a zobrazením dialógového okna s výpisom chyby.

Na zrušenie automatickej obsluhy chýb pre aktuálne VI zvoľte File»VI Properties a následne vyberte Execution zo ponuky Category. Na zrušenie automatickej obsluhy chýb pre nové,

prázdne VI zvoľte Tools»Options ak následne vyberte Block Diagram z ponuky Category. Na zrušenie automatickej obsluhy chýb pre subVI alebo funkciu vo VI prepojte parametre error out s parametrami error in jednotlivých subVI alebo funkcií, alebo s indikátorom error out.

2.18.2. *Manuálna obsluha chýb*

Môžete si zvoliť iné metódy obsluhy chýb. Napríklad, ak dôjde k timeout V/V na blokovom diagrame, nie nutne chcete celú aplikáciu zastaviť a zobraziť error dialog box. Zároveň môžete chcieť zaistiť aby VI sa pokúšalo operáciu vykonať počas určitej doby. V LabVIEW môžete tieto rozhodnutia o obsluhe chyby implementovať v blokovom diagrame.

Na riadenie chýb použite LabVIEW VI určené na obsluhu chýb, nachádzajúcich sa na palete Dialog & User Interface a parametre error in a error out vo väčšine VI. Napríklad, ak dôjde k chybe, môžete zobraziť chybové hlásenie v rôznych typoch dialógového okna. Použite obsluhu chýb v spojení s ladiacimi nástrojmi na odhalenie a riadenie chýb.

VI a funkcie vrátia chybu iba prostredníctvom numerických chybových kódov alebo chybového clustra. Typicky, funkcie používajú numerické chybové hlásenia, VI používajú error clustre, obyčajne s chybovými vstupmi a výstupmi. Error clustre typicky poskytujú totožnú vstupnú a výstupnú funkcionalitu.

Pri hociakých V/V operáciách berte ohľad na možnosť výskytu chyby. Takmer všetky V/V operácie vracajú informáciu o chybe. Do VI zahrňte kontrolu chýb, obzvlášť pre V/V operácie (súbor, sériový port, inštrumentácia, zber dát, komunikácia) a zabezpečte mechanizmus na primeranú obsluhu chýb.

Na riadenie chýb použite obslužné LabVIEW VI na obsluhu chýb, funkcie a parametre. Napríklad, ak dôjde k chybe, môžete zobraziť chybové hlásenie v dialógovom okne. Ďalšou možnosťou je vykonať opravu chyby a následne vymazať chybu pomocou prepojenia error out výstupu subVI alebo funkcie so vstupom error in Clear Errors VI. Použite obsluhu chýb v spojení s ladiacimi nástrojmi na odhalenie a riadenie chýb. National Instruments odporúča používať obsluhu chýb.

2.18.3. *Error clustre*

Použite error cluster kontrolky a indikátory na vytvorenie vstupov a výstupov pre chyby v subVI.

Clustre error in a error out obsahujú nasledovné prvky:

- status je boolean hodnota ktorá má hodnotu TRUE ak nastala chyba
- code je 32-bit signed integer ktorý numericky identifikuje chybu. Nenulový kód chyby so statusom nastaveným na FALSE označuje upozornenie (warning).
- source je string ktorý identifikuje kde nastala chyba.

Obsluha chýb v LabVIEW sa vykoná v súlade s modelom toku dát. Obdobne ako dáta, aj chyby sa prenášajú cez spojenia. Informáciu o chybe prepojte zo začiatku po koniec VI. Pripojte VI na obsluhu chyby na koniec VI, tak aby ste zistili či VI zbehlo bez chýb. Použite error in a error out clustre na prenos informácií o chybe cez VI.

Počas behu VI LabVIEW zistí výskyt chyby pri každom uzle. Ak LabVIEW nezistí žiadnu chybu, tak sa uzol vykoná bežným spôsobom. Ak LabVIEW zistí chybu, uzol pošle chybu do ďalšieho uzla bez vykonania danej časti kódu. Ďalší uzol sa správa obdobne. LabVIEW hlási chybu na konci toku vykonávania.

2.18.4. *Vysvetlenie chyby*

Keď nastane chyba, pravým kliknite na okraj error clustra a zvoľte Explain Error z kontextovej ponuky na zobrazenie dialógového okna Explain Error. Dialógové okno Explain Error obsahuje informácie o chybe. Kontextové menu zahrňuje možnosť Explain Warning ak VI neobsahuje chyby ale obsahuje upozornenie (warning).

Dialógové okno Explain Error môžete zobraziť aj z ponuky Help»Explain Error.

3. IMPLEMENTÁCIA ALGORITMOV

Táto lekcia sa zaoberá s implementáciou kódu v LabVIEW. Zahrňuje návrh používateľského rozhrania, výber dátového typu, dokumentáciu kódu, používanie slučiek ako napr. While slučka, For slučka, pridanie softvérového časovania do kódu, zobrazenie dát v grafe, a implementáciu rozhodovania v kóde pomocou Case štruktúry.

3.1. Čelný panel

Vo fáze návrhu vývoja softvéru sa identifikujú vstupy a výstupy riešeného problému. Identifikácia vstupov a výstupov priamo vedie k návrhu okna čelného panelu.

Vstupy môžu pochádzať z viacerých zdrojov:

- zberom dát zo zariadenia ako je napr. DAQ zariadenie alebo multimeter
- načítaním priamo zo súboru
- zmenou kontroliek

Výstupy sa môžu zobraziť pomocou indikátorov, ako napríklad grafy, charty, LED indikátory, alebo sa môžu zapísať do súboru. Ďalšou možnosťou výstupu dát je generovať výstupný signál na zodpovedajúcom zariadení. S akvizíciou dát, tvorbou signálov, zapisovaním do súboru sa zaoberáme v ďalších častiach kurzu.

3.2. Návrh kontroliek a indikátorov

Pri voľbe kontroliek a indikátorov sa uistite, že sú vhodné resp. primerané k úlohe na ktorú ich používate. Napríklad, na nastavenie frekvencie sínusového priebehu, zvoľte kontrolku dial, na zobrazenie teploty, zvoľte indikátor thermometer.

3.3. Návestia (label) a popisy (caption)

Uistite sa, že kontrolky a indikátory sú jasne označené pomocou návestí. Správne označenie pomáha používateľom identifikovať účel jednotlivých kontroliek a indikátorov. Zároveň je to pomôcka pri dokumentácii kódu na blokovom diagrame. Návestia kontroliek a indikátorov korešpondujú s názvami terminálov na blokovom diagrame.

Popisy (caption) slúžia na opis kontrolky a indikátora na čelnom panelu. Popisy nie sú zobrazené na blokovom diagrame. Pomocou popisov môžete dokumentovať používateľské rozhranie bez toho aby ste pridali zbytočný text do blokového diagramu.

3.4. Nastavenia kontroliek a indikátorov

Ku kontrolkám môžete nastaviť predvolené hodnoty. Pomocou predvolenej hodnoty zaistíte zmysluplnú vstupnú hodnotu v prípade, ak používateľ nezadal inú počas behu VI. Jednotlivé položky vzťahujúce sa ku kontrolkám a indikátorom sa dajú osobite zobraziť resp. skryť.

3.5. Používanie farby

Správe používanie farby môže zlepšiť vzhľad a funkcionalitu vášho používateľského rozhrania. Avšak použitie nadmerného počtu rôznych farieb môže spôsobiť neharmonický, rušivý efekt používateľského rozhrania.LabVIEW poskytuje nástroj color picker, ktorý môže pomôcť pri výbere vhodných farieb. Vyberte nástroj Coloring tool a následne pravým klikom na objekt alebo pracovnú plochu zobrazíte color picker. Horná časť nástroja color picker obsahuje spektrum šedej farby a možnosť nastavenia priehľadnosti objektov. Druhé spektrum obsahuje tmavšie farby ktoré sú vhodné na pozadie a objekty čelného panela. Tretie spektrum obsahuje farby ktoré sú vhodné na zvýraznenie. Pohybom myši po spektre sa dynamicky mení farba objektov, čo pomôže pri výbere vhodnej farebnej kombinácie.

Užitočné tipy pri voľbe farieb:

- Použite predvolené LabVIEW farby. LabVIEW nahradí farby obdobne ako nahradí typy písma. Ak niektorá z farieb VI nie je dostupná, LabVIEW ju nahradí s najviac podobajúcou sa.Môžete použiť aj systémové farby na prispôsobenie okna čelného panelu k systémovým farbám počítača na ktorom je spustené VI.
- Začnite so sivou schémou. Voľte jednu alebo dve odtiene sivej, a farby na zvýraznenie ktoré majú dobrý kontrast k pozadiu.
- Farby na zvýraznenie pridávajte opatrne - ku grafom, na ukončovanie tlačidlo a prípadne na posúvatko (objekt slider) - pre dôležité nastavenia. Malé objekty môžu potrebovať svetlejšie farby a viac kontrastu ako väčšie objekty.
- Používajte rozdiely v kontraste častejšie ako farebné rozdiely. Farboslepý používatelia ťažšie rozoznajú rozdiely medzi farbami ako medzi kontrastom.
- Na zoskupenie objektov ich správne rozmiestnite, namiesto rozlíšenia pomocou farieb.
- Používanie farieb si môžete naštudovať aj z panelov prístrojov, máp a z magazínov.
- Pre použitie systémových farieb pri kontrolkách čelného panela zvoľte objekty z kategórie System z palety Controls.

3.6. Rozmiestnenie a zarovnanie

Rozmiestnenie a zarovnanie sú najdôležitejšie techniky pre zoskupenie a rozdeľovanie objektov. Organizácia objektov sa javí čistejšou a kohéznejšou ak sú objekty zoradené podľa vodiacej čiary. Keď sú objekty zoradené na čiare, oko používateľa sa pohybuje po čiare zľava doprava resp. zhora dole. Súvisí to s čítaním textu. Hoci v niektorých kultúrach je zaužívaný smer sprava doľava, smer zhora dole je zaužívaný takmer všade.

Použite nasledovné pokyny na návrh čelného panela ktorý má slúžiť ako používateľské rozhranie.

- Neumiestnite objekty príliš blízko k sebe. Kvôli lepšej čitateľnosti čelného panela nechajte prázdne miesto medzi objektmi. Zároveň predídete tomu aby používateľ náhodou klikol na nesprávnu kontrolku alebo tlačidlo.

- Neumiestnite objekty na seba. Už aj čiastočné prekrytie kontroliek alebo indikátorov s inými objektmi alebo návestiami spomalí obnovovanie obrazovky a môže spôsobiť blikanie indikátora.

3.7. Text a fonty

Te x t j e ľahšie čitateľný a informácia je zrozumiteľnejšia keď je zobrazenie systematické. Použite predvolené LabVIEW fonty. Napríklad, LabVIEW definuje vstavané fonty ako predvolené systémové fonty. Ak prenesiete VI medzi rôznymi platformami, tak LabVIEW automaticky zaktualizuje vstavané fonty tak, aby sa zhodovali s predvoleným systémovým fontom na aktuálnej platforme. Pri otvorení VI ktoré používa nedostupný font na danej platforme, LabVIEW automaticky nahradí font s najviac podobajúcim sa fontom ktorý je dostupný. Napríklad, keď otvoríte VI ktoré používa typ písma Arial na počítači ktorý nemá typ Arial, LabVIEW nahradí písmo s podobným typom ktorý je nainštalovaný.

Používanie väčšieho počtu štýlov písma na čelnom panely môže pôsobiť rušivo a neorganizovane. Namiesto viacerých typov písma, použite dva alebo tri rôzne veľkosti toho istého typu písma. Typ Serif pomôže používateľom rozoznať celé slová z diaľky. Ak používate viac než jednuveľkosť písma, uistite sa, že tieto veľkosti sú značne rozdielne. V opačnom prípade môžu tvoriť dojem, že sa jedná o chybu. Podobne, ak používate dva rôzne typy písma, uistite sa, že zreteľne rozdielne.

Pri tvorbe priemyselných staníc pre operátorov navrhnite používateľské rozhranie s väčšími typmi písma a s vyšším kontrastom. Odrazy svetla a potreba čítať informácie z diaľky robí normálne typy písma nevhodnými. Dotykové obrazovky vo všeobecnosti požadujú väčšie typy písma a väčšie odstupy medzi objektmi.

3.8. Tipy a nástroje používateľského rozhrania

Vstavané LabVIEW nástroje na tvorbu používateľsky prívetivého používateľského rozhrania zahŕňajú systémové kontrolky, tab kontrolky, dekorácie, menu a automatickú zmenu objektov front panelu.

3.9. Systémové kontrolky

Bežnou technikou používateľského rozhrania je zobrazenie dialógového okna vo vhodnom čase pre interakciu s používateľom. Zobrazenie vo forme dialógového okna zvolíte pomocou File»VI Properties, následne voľbou kategórie Window Appearance a výberom možnosti Dialog.

V dialógových oknách použite systémové kontrolky a indikátory nachádzajúce sa na System palete. Systémové kontrolky menia svoj vzhľad podľa toho na akej platforme beží VI. Keď spustíte VI na inej platforme, systémové kontrolky prispôsobia svoju farbu a vzhľad tak aby sa zhodovali s kontrolkami zo štandardného dialógového okna pre danú platformu.

Systémové kontrolky typicky ignorujú všetky farby okrem priehľadnej. Pri integrácii grafu alebo nesystémovej kontrolky do čelného panelu upravte objekty pomocou ukrytia niektorých okrajov alebo voľbou farieb podobných k systémovým farbám.

3.10. Tab kontrolky

Fyzické prístroje obyčajne disponujú s dobrým používateľským rozhraním. Inšpirujte sa s ich princípmi návrhu, ale v prípade vhodnosti použite menšie alebo spôsobilejšie kontrolky, ako napríklad ring kontrolky, tab kontrolky. Použite tab kontrolky na prekrytie kontroliek a indikátorov čelného panela.

Nové strany do tab kontrolky pridáte pomocou pravého kliku a voľbou Add Page Before alebo Add Page After z kontextovej ponuky. Zmeňte návestia strán pomocou nástroja Labeling tool, a umiestnite objekty čelného panela na adekvátne strany. Ako pre všetky ostatné objekty (okrem dekorácií) čelného panela, tak aj pre objekty na tab kontrolke sú terminály dostupné na blokovom diagrame.

Môžete prepojiť výstup z terminálu tab kontrolky do selektora Case štruktúry a tým upraviť blokový diagram. Pri tejto metóde pridružíte ku každej strane tab kontrolky subdiagram v Case štruktúre. Umiestnite terminály kontroliek a indikátorov z každej strany tab kontrolky - spolu s priradenými uzlami a spojeniami na blokovom diagrame - do subdiagramu v Case štruktúre.

3.11. Dekorácie

Použite dekorácie z palety Decorations na zoskupenie alebo oddelenie objektov pomocou priehradok, čiar a šípok.

3.12. Ponuky (Menus)

Použite na mieru šité ponuky na zobrazenie funkcionality čelného panelu systematickým spôsobom a na relatívne malom priestore. Šetrenie miestom zabezpečí viac priestoru na čelnom panely pre kritické kontrolky, indikátory, položky pre začiatočníkov, položky potrebné pre produktivitu, a položky ktoré sa nehodia do ponúk. Môžete použiť kombináciu klávesov pre voľbu z ponuky.

Run-time ponuky pre objekty čelného panela vytvoríte pomocou pravého kliku na objekt a voľbou Advanced»Run-Time Shortcut Menu»Edit. Na tvorbu na mieru šitej run-time ponuky pre VI zvoľte Edit»Run-Time Menu.

3.13. Automatická zmena veľkosti objektov čelného panela

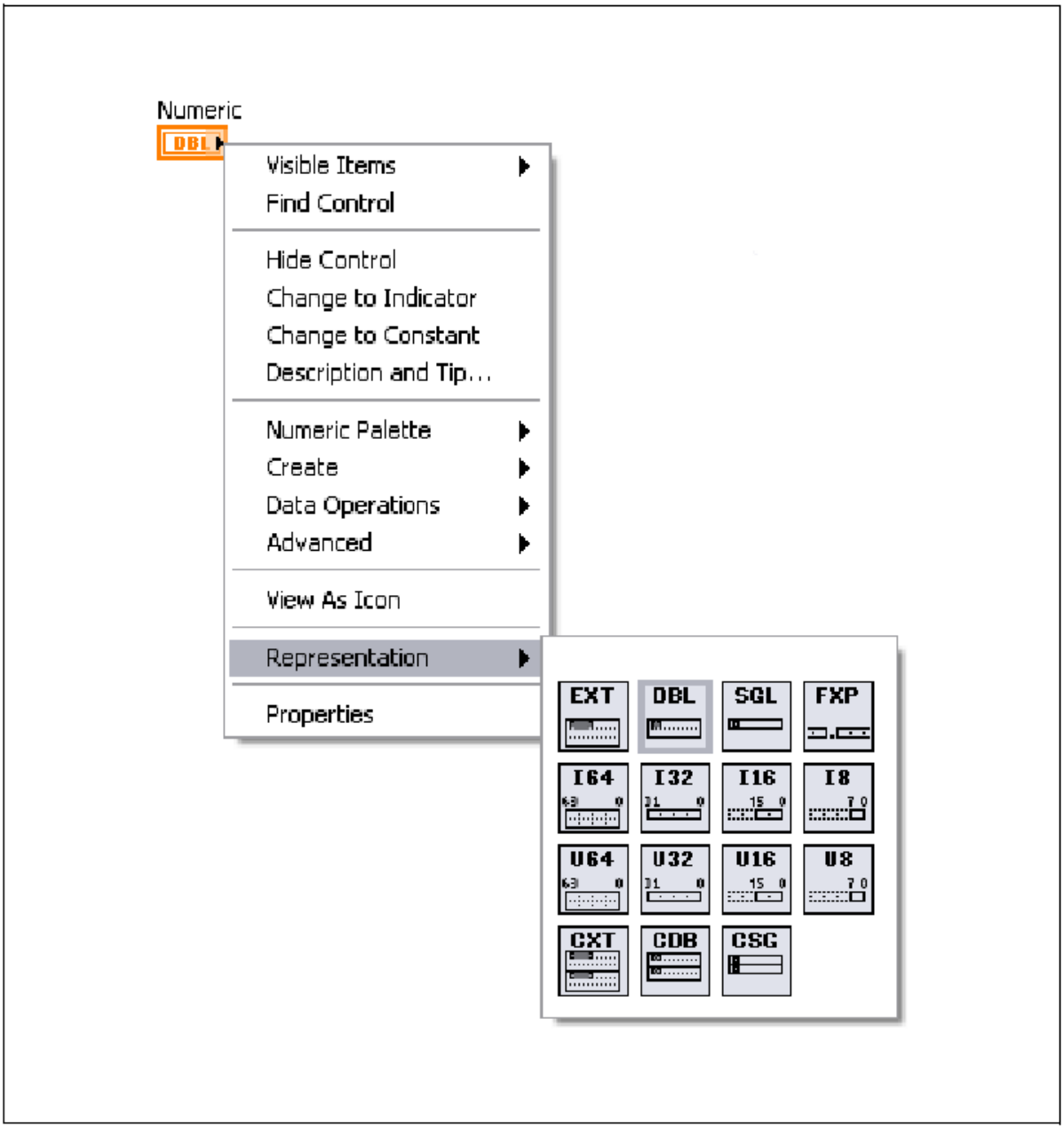
Na nastavenie minimálnej veľkosti okna, na uchovanie pomerov strán počas zmeny obrazovky a nastavenia zmeny veľkosti objektov čelného panela v dvoch rozdielnych módoch použite voľbu File»VI Properties»Window Size. Pri návrhu VI berte ohľad na zobrazenie čelného panela na obrazovkách s rozdielnymi rozlíšeniami. Na uchovanie pomerov strán okna čelného panela pri rôznych rozlíšeniach obrazovky zvoľte File»VI Properties, následne

vyberte Window Size z výsuvnej ponuky Category a zaškrtnite políčko Maintain proportions of window for different monitor resolutions.

3.14. Dátové typy LabVIEW

3.14.1. *Dátové typy numeric*

Dátový typ numeric reprezentuje rôzne typy čísel. Zmenu reprezentácie typu čísla vykonáte pravým klikom na kontrolku, indikátor alebo konštantu a voľbou Representation



Numerická reprezentácia

Keď pripojíte dva alebo viac číselných vstupov s rôznou reprezentáciu k funkcii, funkcia obyčajne vráti dáta vo väčšej, alebo širšej reprezentácii. Funkcie pretypujú menšiu reprezentáciu na väčšiu reprezentáciu čísla pred vykonaním funkcie a LabVIEW graficky zobrazí bodku na terminály kde nastala konverzia.

Numerický dátový typ zahŕňa nasledovné podkategórie - čísla s pohyblivou rádovou čiarkou (floating-point), celé čísla so znamienkom (signed integers), celé čísla bez znamienka (unsigned integers), a komplexné čísla.

3.14.2. *Čísla s pohyblivou rádovou čiarkou*

Čísla s pohyblivou rádovou čiarkou reprezentujú desatinné čísla. V LabVIEW, čísla s pohyblivou rádovou čiarkou sú reprezentované oranžovou farbou.

Single-precision (SGL) - jednoduchá presnosť, 32-bit IEEE čísla s jednoduchou presnosťou s pohyblivou rádovou čiarkou Single-precision čísla používajte na šetrenie s pamäťou a dbajte na pretečenie rozsahu.

Double-precision (DBL) - dvojitá presnosť, 64-bit IEEE čísla s dvojitou presnosťou s pohyblivou rádovou čiarkou Dvojitá presnosť je predvolená reprezentácia numerických objektov. Pre väčšinu situácií používajte čísla s dvojitou presnosťou s pohyblivou rádovou čiarkou.

Extended-precision (EXT)—Veľkosť a presnosť čísiel s rozšírenou presnosťou sa mení v závislosti od platformy. V systéme Windows majú tieto čísla formát 80-bit IEEE číslo s rozšírenou presnosťou.

3.14.3. *Čísla s pevnou rádovou čiarkou (Fixed-point)*

Čísla s pevnou rádovou čiarkou reprezentujú racionálne čísla pomocou binárnych číslic, alebo bitov. Na rozdiel od čísel s pohyblivou rádovou čiarkou, ktoré umožnia aby LabVIEW použil meniaci sa počet bitov na reprezentáciu čísla, čísla s pevnou rádovou čiarkou používajú špecifický počet bitov. Hardvér a cieľové zariadenia, ktoré dokážu uložiť a spracovať dáta s limitovaným alebo pevným počtom bitov, môžu uložiť a spracovať tieto čísla. Môžete špecifikovať rozsah a presnosť čísel s pevnou rádovou čiarkou.

Dátový typ s pevnou rádovou čiarkou použite ak nepotrebujete dynamickú funkcionalitu pohyblivej rádovej čiarky alebo v prípade ak chcete pracovať s cieľovým zariadením ktoré nepodporuje aritmetiku s pohyblivou rádovou čiarkou, ako napr. FPGA cieľové zariadenia.

Pri prispôsobovaní čísla určte znamienko, šírku slova, a šírku celej časti čísla s pevnou rádovou čiarkou.

Znamienko (encoding)— Binárne zakódovanie čísla s pevnou rádovou čiarkou. Môžete vybrať so alebo bez znamienka. V prípade výberu so znamienkom, bit určujúci znamienko je prvý bit v reprezentácii dát.

Dĺžka slova (Word length) - Celkový počet bitov ktoré LabVIEW používa na reprezentáciu všetkých kladných hodnôt čísla s pevnou rádovou čiarkou. LabVIEW akceptuje maximálnu dĺžku slova 64 bitov. Určité cieľové zariadenia môžu obmedziť dáta na kratšiu dĺžku slova. V

prípade ak otvoríte VI na cieľovom zariadení a VI obsahuje dáta s pevnou rádovou čiarkou s väčšou dĺžkou slova ako dokáže cieľové zariadenie akceptovať, VI bude obsahovať nefunkčné spojenia. Informácie o maximálnej dĺžke slova nájdete v dokumentácii daného cieľového zariadenia.

Šírka celej časti čísla (Integer word length) - Počet bitov na reprezentáciu všetkých možných celých čísel pri pevnej rádovej čiarke,alebo bitová vzdialenosť rádovej čiarky od najvýznamnejšieho bitu v slove. Táto šírka môže byť väčšia ako dĺžka slova, a môže byť záporná aj kladná.

3.14.4. *Celé čísla (Integer)*

Integer reprezentuje celé čísla. Celé čísla so znamienkom môžu byť záporné alebo kladné. Použite celé čísla bez znamienka keď viete, že čísla budú iba kladné. LabVIEW zobrazuje celé čísla s modrou farbou.

Pri konverzii čísel s pohyblivou rádovou čiarkou na celé čísla LabVIEW zaokrúhli číslo na najbližšie párne celé číslo. Napríklad, LabVIEW zaokrúhli 2,5 na 2 a 3,5 na 4.

Byte (I8)—Byte integer čísla disponujú s 8 bitovou dĺžkou a rozsahom od –128 do 127.

Word (I16))—Word integer čísla disponujú s 16 bitovou dĺžkou a rozsahom od -32,768 do 32,767.

Long (I32)—Long integer čísla disponujú s 32 bitovou dĺžkou a rozsahom od –2,147,483,648 do 2,147,483,647. Vo väčšine prípadov je najlepšie použiť 32-bit integer čísla.

Quad (I64))—Quad integer čísla disponujú s 64 bitovou dĺžkou a rozsahom od –1e19 do 1e19.

Byte (U8)—Byte unsigned integer čísla disponujú s 8 bitovou dĺžkou a rozsahom od 0 do 255.

Word (U16))—Word integer čísla disponujú s 16 bitovou dĺžkou a rozsahom od 0 do 65,535.

Long (U32)—Long unsigned integer čísla disponujú s 32 bitovou dĺžkou a rozsahom od 0 do 4,294,967,295.

Quad (U64)—Quad unsigned integer čísla disponujú s 64 bitovou dĺžkou a rozsahom od 0 do 2e19.

3.14.5. *Komplexné čísla*

Komplexné čísla sú reprezentované pomocou dvoch v pamäti spojených hodnôt - jedna hodnota na reprezentáciu reálnej časti a jedna na reprezentáciu imaginárnej časti čísla. Nakoľko komplexné čísla sú typom čísiel s pohyblivou rádovou čiarkou, sú reprezentované oranžovou farbou v LabVIEW.

Complex Single (CSG)—Komplexné číslo s jednoduchou presnosťou, reálna a imaginárna časť sú 32-bit IEEE čísla s jednoduchou presnosťou s pohyblivou rádovou čiarkou.

Complex Double (CDB)—Komplexné číslo s dvojitou presnosťou, reálna a imaginárna časť sú 64-bit IEEE čísla s dvojitou presnosťou s pohyblivou rádovou čiarkou.

Complex Extended (CXT)—Komplexné číslo s rozšírenou presnosťou, reálna a imaginárna časť sú IEEE čísla s rozšírenou presnosťou. Veľkosť a presnosť čísiel s rozšírenou presnosťou sa mení v závislosti od platformy. V systéme Windows majú tieto čísla formát 80-bit IEEE číslo s rozšírenou presnosťou.

3.14.6. *Boolean hodnoty*

LabVIEW reprezentuje boolean dáta ako 8 bitové hodnoty. Boolean hodnota je FALSE ak 8 bitová hodnota je nula. Ľubovoľná nenulová hodnota reprezentuje TRUE. Boolean dáta sú v LabVIEW reprezentované zelenou farbou.

Boolean hodnoty majú priradenú mechanickú akciu. Dve hlavné skupiny akcií sú latch a switch. Vybrať si môžete z nasledovných módov:

• Switch when pressed—zmení hodnotu kontrolky vždy keď na kontrolku kliknete pomocou nástroja Operating tool. Frekvencia vyčítania hodnoty z kontrolky nemá vplyv na jej fungovanie.

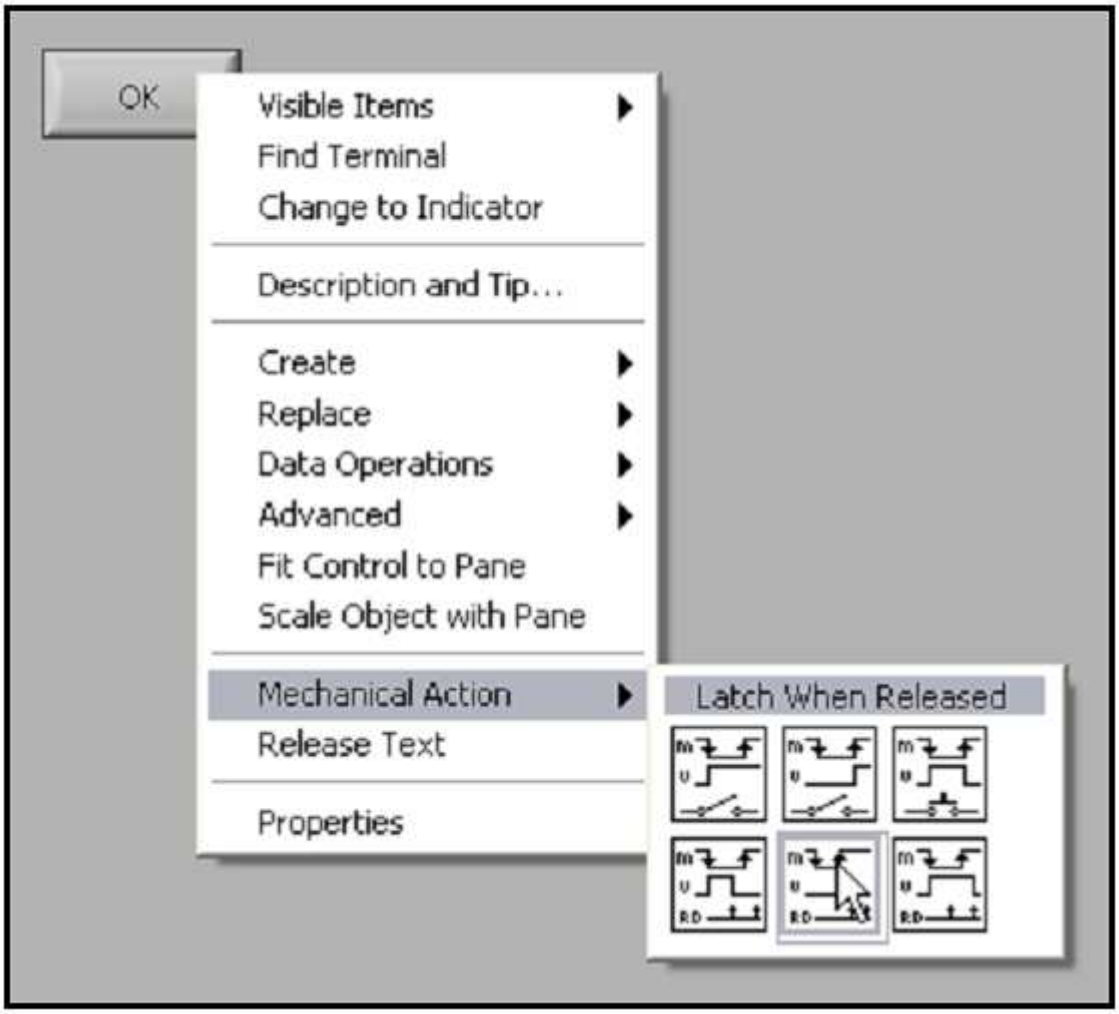
• Switch when released—zmení hodnotu kontrolky iba po tom čo ste uvoľnili tlačidlo myši počas kliknutia v na kontrolke. Frekvencia vyčítania hodnoty z kontrolky nemá vplyv na jej fungovanie.

• Switch until released—zmení hodnotu kontrolky pri kliknutí a drží túto novú hodnotu pokým neuvoľníte tlačidlo myši. Po uvoľnení sa kontrolka prepne na predvolenú (default) hodnotu, fungovanie je podobné zvončeku na dverách. Frekvencia vyčítania hodnoty z kontrolky nemá vplyv na jej fungovanie. Tento mód sa nedá zvoliť pre radio button kontrolku.

• Latch when pressed—zmení hodnotu kontrolky keď na kontrolku kliknete a drží túto novú hodnotu pokým vyčítanie hodnoty neprebehne. Kontrolka sa prepne na predvolenú (default) hodnotu aj keď držíte tlačidlo myši zatlačené. Toto správanie je podobné prerušovaču a je užitočné pri zastavení While slučky, alebo riadení vykonania kódu iba raz pri každom stlačení tlačidla. Tento mód sa nedá zvoliť pre radio button kontrolku.

• Latch when released—zmení hodnotu kontrolky iba po tom čo ste uvoľnili tlačidlo myši počas kliknutia v na kontrolke. Po vyčítaní hodnoty sa kontrolka prepne na predvolenú (default) hodnotu. Toto správanie je podobné tlačidlám na dialógovom okne alebo pri systémových tlačidlách. Tento mód sa nedá zvoliť pre radio button kontrolku.

• Latch until released—zmení hodnotu kontrolky keď na ňu kliknete a drží túto hodnotu pokým je hodnota raz vyčítaná alebo uvoľnite tlačidlo myši, v závislosti od toho čo nastane neskôr. Tento mód sa nedá zvoliť pre radio button kontrolku.



Mechanické akcie booleanov

3.14.7. *String*

String je sekvencia zobraziteľných a nezobraziteľných ASCII znakov. String poskytuje platformovo nezávislý formát pre informácie a dáta. Najbežnejšie využitie dátového typu string v aplikáciách zahrnuje:

• Tvorba jednoduchých textových správ

• Riadenie prístrojov pomocou posielania textových príkazov a prijímanie dát v tvare ASCII alebo binárny string, ktorý následne môžete prekonvertovať na numerické hodnoty.

• Uloženie numerických dát na disk. K tomu aby ste mohli uložiť numerické dáta do ASCII súboru musíte pred zápisom súboru najskôr skonvertovať numerické dáta na string.

• Komunikácia s používateľom pomocou dialógových okien.Dátový typ string je zobrazený pomocou tabuliek, textových rámčekov (text entry boxes), a návestí na čelnom panely. LabVIEW obsahuje vstavané VI a funkcie ktoré môžete použiť na manipuláciu so stringom, zahrňujúc formátovanie, analýzu a pod.

Viac informácií ohľadne ASCII kódov a konverzných funkcií nájdete v téme ASCII Codes.

LabVIEW zobrazuje typ string ružovou farbou.

Pomocou pravého kliknutia na string kontrolku alebo indikátor na čelnom panely môžete zvoliť typ zobrazenia podľa nasledovnej tabuľky: Tabuľka obsahuje aj príklad správy pre každý typ zobrazenia.

Typ zobrazenia	Popis	Správa
Normal Display	Zobrazí tlačiteľné znaky s použitím typu písma kontrolky. Nezobraziteľné znaky sa vo všeobecnosti zobrazia ako malé štvorce.	`Existujú štyri typy zobrazenia. \ je spätné lomítko.`
'\' Codes Display	Zobrazí kódy so spätným lomítkom pre všetky nezobraziteľné znaky.	`Existujú\šťyri\stypy\szobrazenia.\n\\\sje\sspätné\slomítko.`
Password Display	Zobrazí hviezdičky (*) pre každý znak, vrátane medzier.	`**********`
Hex Display	Namiesto samotného znaku zobrazí ASCII hodnotu každého znaku v hexadecimálnom tvare.	`5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074 7970 6573 2E0A 5C20 6973 2061 2062 6163 6B73 6C61 7368 2E`

LabVIEW interne reprezentuje string pomocou smerníka (pointer) na štruktúru ktorá obsahuje 4-bytovú hodnotu dĺžky a 1D pole byte integerov (8 bit znaky).

3.14.8. *Vymenovaný typ (Enum)*

Enum (vymenovaný typ) je kombinácia dátových typov. Enum reprezentuje dvojicu hodnôt, string a numeric, pričom enum môže nadobudnúť hodnotu zo zoznamu definovaných kombinácií.

3.14.9. *Dynamický dátový typ (Dynamic)*

Dynamický dátový typ slúži na prácu s dátami generovanými alebo získanými pomocou Express VI. Terminál dynamického dátového typu má tmavo-modrú farbu , ako je to zobrazené naľavo. Väčšina Express VI akceptuje a/alebo vracia dynamický dátový typ. Dynamický dátový typ môžete pripojiť k ľubovoľnému indikátoru alebo k vstupu ktorý akceptuje dátové typy numeric, priebeh (waveform) a boolean. Pripojte dynamický dátový typ k indikátoru ktorý najlepšie reprezentuje dáta. Indikátory zahrňujú grafy, charty, alebo numerické indikátory.

Väčšina zvyšných VI a funkcií neakceptuje dynamický dátový typ. K analýze, alebo k spracovaniu dynamického dátového typu vstavanými VI alebo funkciami musíte spraviť konverziu dynamického dátového typu.

Na konverziu dynamického dátového typu na numeric, boolean, priebeh (waveform) a pole použite Convert from Dynamic Data Express VI. Pri vložení Convert from Dynamic Data Express VI na blokový diagram sa zobrazí dialógové okno Configure Convert from Dynamic Data. Dialógové okno Configure Convert from Dynamic Data obsahuje nastavenia pomocou ktorých

špecifikujete aký výstupný formát dát vráti Convert from Dynamic Data Express VI.
Keď pripojíte dynamický dátový typ indikátoru pola, LabVIEW automaticky pridá Convert from Dynamic Data Express VI do blokového diagramu. Dvojklikom na Convert from Dynamic Data Express VI zobrazíte dialógové okno Configure Convert from Dynamic Data kde si môžete skontrolovať výstup dát do poľa.

3.15. Dokumentácia kódu

Profesionálny vývojári ktorí udržujú a modifikujú kód vedia oceniť hodnotu dobrej dokumentácie. Dobrá dokumentácia blokového diagramu vám uľahčí implementáciu budúcich zmien v kóde. Zároveň použite dokumentáciu na vysvetlenie účelu VI a objektov čelného panela.

Použite tip strip návestia, opisy (descriptions), vlastnosti VI (VI Properties) a princípy dobrého návrhu na dokumentáciu čelného panelu.

3.16. Tip strip návestia a opisy (description)

Tip strip návestia sú krátke opisy ktoré sa zobrazia keď sa počas behu VI kurzorom myši nastavíte na kontrolku alebo indikátor. Napríklad, návestie tip strip môže zobrazovať v akých jednotkách sa zadáva teplota, alebo opisovať význam vstupov do algoritmu. Opisy (descriptions) poskytujú dodatočné informácie o danej kontrolke alebo indikátore. Opisy sa zobrazia v okne Context Help keď sa kurzor myši nachádza nad objektom. Návestia tip strip a opisy ku kontrolkám a indikátorom pridáte pomocou pravého kliknutia na objekt a voľbou Description and Tip z kontextovej ponuky.

3.17. Vlastnosti VI (VI Properties)

Použite zložku Documentation dialógového okna VI Properties na opis VI a vytvorenie väzieb na HTML súbory alebo súbory pomocníka. VI Properties zobrazíte pomocou pravého kliknutia na ikonu VI na čelnom panely alebo blokovom diagrame a voľbou VI Properties z kontextovej ponuky, alebo pomocou voľby File»VI Properties. Následne zvoľte Documentation z ponuky Categories. Toto dialógové okno je neprístupné počas behu VI.

Strana dokumentácie obsahuje nasledovné komponenty:

- VI description—obsahuje text ktorý sa zobrazí v okne Context Help ak kurzor myši je nad ikonou VI. Použite <B> a </B> párové značky pred a za textom ktorý chcete zobraziť zvýraznene. Na programatickú zmenu opisu VI môžete použiť VI Description property uzol.
- Help tag—obsahuje názov HTML súboru alebo index kľúčového slova témy ku ktorej chcete spraviť spojenie v súbore pomocníka. Programatické nastavenie tejto vlastnosti je možné pomocou Help:Document Tag property.

- Help path—obsahuje cestu k HTML súboru alebo k súboru pomocníka na ktoré má existovať prepojenie z okna Context Help. Ak je táto položka prázdna, linka Detailed help sa neobjaví v okne Context Help, a tlačidlo Detailed help je neaktívne.
- Browse—zobrazí dialógové okno na výber HTML súboru alebo súboru pomocníka ktorý sa má použiť v časti Help path.

3.18. Grafické programovanie

Aj keď samotné grafické programovanie v LabVIEW je seba dokumentujúce, dodatočné poznámky sú nápomocné pri budúcich modifikáciách VI. Existujú dva typy poznámok na blokovom diagrame - poznámky ktoré opisujú funkciu alebo funkčnosť algoritmov a poznámky ktoré opisujú účel prenášaných dát cez spojenia. Oba typy poznámok sú zobrazené na nasledujúcom blokovom diagrame. Štandardné návestia môžete vložiť pomocou nástroja Labeling tool, alebo pomocou voľných návestí (free labels) z palety Functions»Programming»Structures»Decorations. Podľa predvoleného nastavenia majú voľné návestia žlté pozadie.

Použite nasledovnú smernicu na písanie komentárov do VI:

- Použite poznámky na blokovom diagrame k vysvetleniu funkcionality kódu.
- Použite voľné návestia na opis funkcií blokového diagramu aj keď kód v LabVIEW je seba dokumentujúci.
- Nezobrazujte návestia volania funkcií a subVI nakoľko sú obvykle obsiahle a nepraktické. Vývojári si názvy funkcií a subVI môžu zobraziť pomocou okna Context Help.
- Na opis dlhých spojení použite malé voľné návestia s bielym pozadím. Používanie popisu spojení je užitočné pre spojenia vedúcich z posuvných registrov (shift register) a pre dlhé spojenia ktoré vedú cez celý blokový diagram.Viac informácií o posuvných registroch nájdete v sekcii Štruktúra Case.
- Označte štruktúry kvôli špecifikácii ich hlavnej funkcionality.
- Označte konštanty kvôli ich charakteristike.
- Použite voľné návestia na dokumentáciu algoritmov blokového diagramu. Ak používate algoritmus z knihy alebo z iného zdroja, poskytnite referenčnú informáciu.

3.19. While slučky

Podobne ako Do slučky alebo Repeat-Until slučky v textovo orientovaných programovacích jazykoch, While slučka vykoná subdiagram pokým nenastane definovaná podmienka.

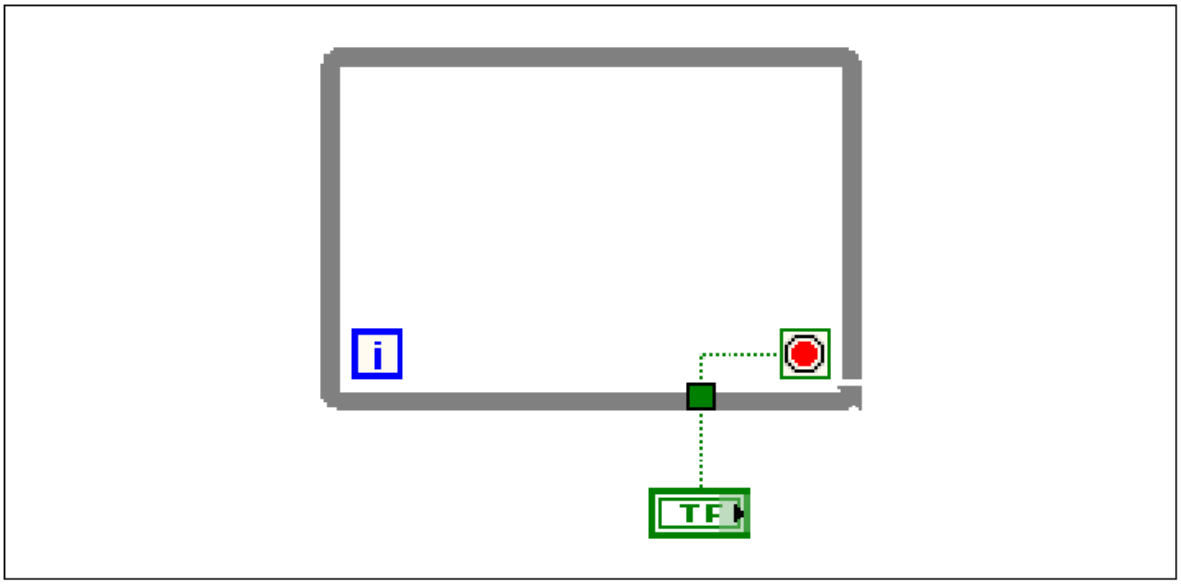
While slučka sa nachádza na palete Structures. Vyberte While slučku z palety, následne pomocou kurzora vykreslite obdĺžnik okolo časti blokového diagramu ktorú chcete vykonávať opakovane. Pri uvoľnení tlačidla myši sa vytvorí While slučka okolo vybranej časti kódu.

Objekty do While slučky pridáte pomocou drag and drop funkcionality, jednoduchým pretiahnutím objektu z palety do slučky.

Terminál iterácií (iteration terminal) je výstupný terminál ktorý poskytuje počet ukončených iterácií.
Terminál iterácií stále začína počítať ukončené iterácie od nuly.
V nasledujúcom blokovom diagrame sa While slučka vykonáva až pokým výstup funkcie Random Number je väčší alebo rovný hodnote 0.9 a kontrolka Enable má hodnotu True. Funkcia And vráti True iba ak oba vstupy majú hodnotu True. V opačnom prípade vráti False.
V nasledovnom príklade je zvýšená pravdepodobnosť nekonečnej slučky. Vo všeobecnosti je preferované mať jednu podmienku splnenú na ukončenie slučky, namiesto nutnosti súčasného splnenia viacerých podmienok.

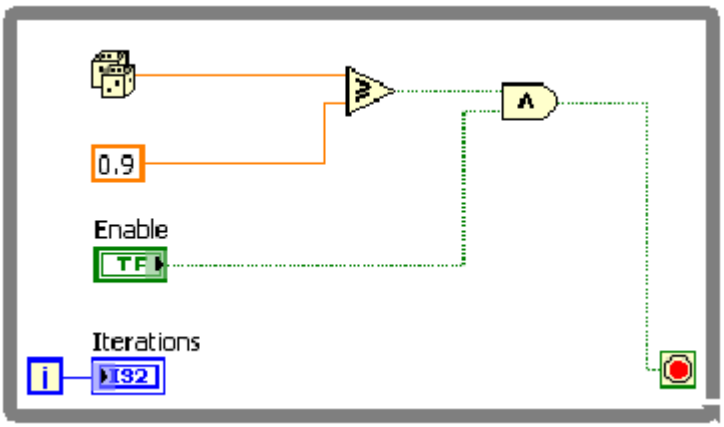
Slučka While opakovane vykonáva kód v subdiagrame až pokým špecifická boolean hodnota sa nevstúpi do podmieňovacieho terminálu (conditional terminal). Podmieňovací terminál vo While slučke sa správa totožne ako v prípade For slučky s podmieňovacím terminálom. Avšak, nakoľko For slučka obsahuje aj terminál počtu iterácií (iteration count), nebude sa vykonávať donekonečna ak podmienka nikdy nenastane. While slučka neobsahuje terminál na nastavenie počtu iterácií a beží donekonečna ak podmienka nikdy nenastane.

Ak podmieňovací terminál je nastavený na Stop if True, a pripojenú boolean kontrolku umiestnite mimo While slučky, pričom pri spustení slučky má hodnotu FALSE, tak vytvoríte nekonečnú slučku, ako je to zobrazené na nasledovnom príklade. Nekonečnú slučku vytvoríte aj v prípade ak podmienený terminál je nastavený na Continue if True a kontrolka mimo slučky je nastavená na TRUE.



Zmena hodnoty kontrolky neukončí nekonečnú slučku, nakoľko hodnota je prečítaná iba raz, pred spustením slučky. Na ukončenie nekonečnej slučky kliknite na tlačidlo Abort Execution na nástrojovej lište.

Môžete vykonať základnú obsluhu chýb pomocou podmieneného terminálu v slučke While. Keď pripojíte error cluster na podmieňovací terminál, iba True alebo False hodnota parametra status sa prenesie do terminálu. Zároveň sa Stop if True a Continue if True položky v kontextovej ponuke zmenia na Stop if Error a Continue while Error.

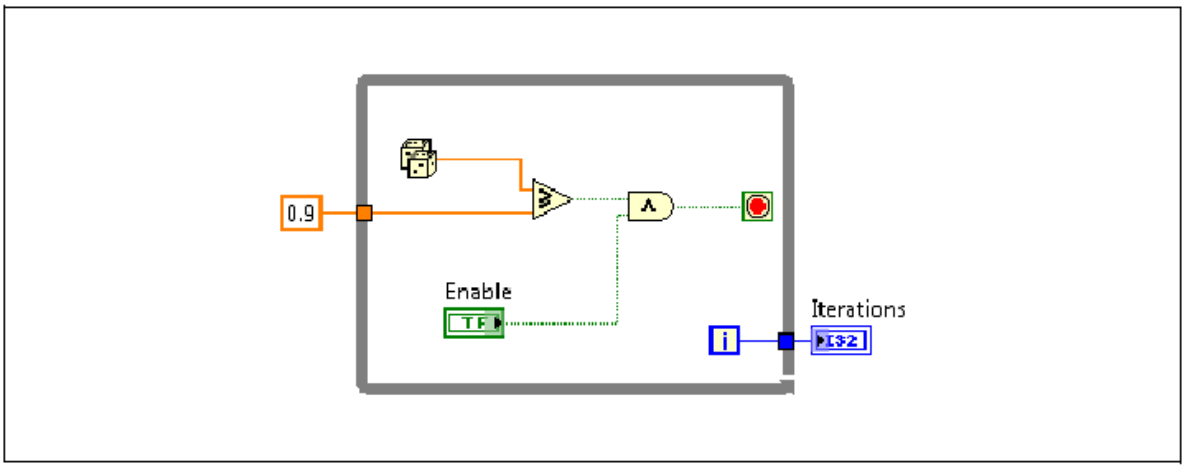


Možná nekonečná slučka

Tunely

Tunely prenášajú dáta do slučky a zo slučky. Tunel za zobrazí v tvare jednoliateho štvorca na hrane While slučky. Nadobudne farbu podľa pripojeného dátového typu. Dáta sa prenesú mimo slučku po ukončení slučky. Slučka sa začne vykonávať iba po tom, čo všetky vstupné dáta sú dostupné v tuneloch.

V nasledovnom blokovom diagrame je terminál iterácií pripojený k tunelu. Hodnota v tunely sa neprenesie do indikátora Iterations až pokým While slučka nedokončí svoj beh.



3.20. Tunel while slučky

Iba posledná hodnota terminálu iterácií je zobrazená v indikátore Iterations. Používanie While slučiek pre obsluhu chýb K ukončeniu While a For slučky môžete pripojiť error cluster s podmieňovacím terminálom. Keď pripojíte error cluster na podmieňovací terminál, iba True alebo False hodnota parametra status sa prenesie do terminálu. Slučka sa ukončí ak dôjde k chybe. V prípade For slučky s podmieňovacím terminálom musíte nastaviť maximálny počet iterácií - pripojiť hodnotu na terminál počtu iterácií (count terminal), alebo pripojiť na vstup auto-indexované pole. For slučka sa bude vykonávať až pokým nedôjde k chybe alebo slučka vykoná nastavený počet iterácií.

Položky Stop if True a Continue if True z kontextovej ponuky sa po pripojení error clusteru ku podmieňovaciemu terminálu zmenia na Stop on Error a Continue while Error.

3.21. For slučka

For slučka vykoná subdiagram nastavený počet krát. Nasledovná ilustrácia zobrazuje For slučku v LabVIEW, flowchart ekvivalentný funkcionalite For slučky, a príklad pseudo-kódu.

For slučka sa nachádza na palete Structures. Slučku While môžete prekonvertovať na slučku For pomocou pravého kliknutia na okraj While slučky a voľbou Replace with For Loop z kontextovej ponuky. Terminál počtu iterácií (count terminal) je vstupný terminál ktorého hodnota určuje koľkokrát sa subdiagram vykoná.

Terminál iterácií (iteration terminal) je výstupný terminál ktorý poskytuje počet ukončených iterácií.Terminál iterácií stále začína počítať ukončené iterácie od nuly.

Rozdiel medzi For slučkou a While slučkou je ten, že For slučka sa vykoná nastavený počet krát. While slučka ukončí vykonávanie subdiagramu iba ak je podmienka ukončenia splnená.

3.22. Pridanie podmieňovacieho terminálu do For slučky

Ak je to potrebné, môžete pridať podmieňovací terminál a tak ukončiť For slučku keď dôjde k naplneniu ukončovacej podmienky alebo k chybe. For slučka s podmieňovacím terminálom beží pokým sa nenaplní podmienka ukončenia, alebo pokým sa nevykoná nastavený počet iterácií, čo nastane skôr. For slučky s podmienkou ukončenia obsahujú červenú bodku pri terminály počtu iterácií a zároveň podmieňovací terminál v pravom dolnom rohu.

Podmieňovací terminál pridáte do For slučky pomocou pravého kliknutia na hranu slučky a voľbou Conditional Terminal z kontextovej ponuky.

Následne spravte spojenia k podmieňovaciemu terminálu a k terminálu počtu iterácií.

3.23. Numerická konverzia

LabVIEW môže reprezentovať numerické dátové typy ako celé čísla so znamienkom alebo bez znamienka, dáta s pohyblivou rádovou čiarkou, alebo komplexné dátové typy, tak ako o tom pojednáva Dátové typy LabVIEW sekcia tejto lekcii. Vo všeobecnosti platí, že ak pripojíte rôzne dátové typy na vstupy funkcie, funkcie vráti výstup vo väčšom alebo v širšom formáte. Ak používate signed integer s unsigned integer, výstup bude v tvare unsigned integer. Ak používate unsigned integer s floating point, výstup bude v tvare floating point. Ak použijete floating point číslo s komplexným číslom, výstup bude v tvare komplexného čísla. Ak použijete dva čísla zhodného dátového typu s rôznou bitovou šírkou, LabVIEW vykoná korekciu na väčšiu bitovú šírku. Ak je počet bitov zhodný, LabVIEW preferuje unsigned integer pred signed integer.

Avšak terminál počtu iterácií v slučke For funguje opačne. Ak pripojíte číslo s pohyblivou rádovou čiarkou s dvojitou presnosťou (DBL) k 32-bit terminálu počtu iterácií, LabVIEW vykoná konverziu väčšieho čísla na 32-bit signed integer. Hoci táto konverzia je v protiklade s

normálnymi konverznými štandardmi, je potrebná, nakoľko počet iterácií slučky For musí byť celé číslo.

Pre lepší výkon sa vyhnite konverzii pomocou použitia zhodných dátových typov, alebo pomocou programatickej konverzii na zhodné dátové typy

3.24. Časovanie VI

Ak sa nenaplní ukončovacia podmienka, slučka po ukončení vykonávania jednej iterácie okamžite začne vykonávaním ďalšej iterácie. Vo väčšine prípadov je potrebné riadenie frekvencie iterácií. Napríklad, ak chcete vykonať akvizíciu pri zberu dát raz za 10 sekúnd, potrebujete metódu na časovanie iterácií tak, aby nastali v 10 sekundových intervaloch.

Aj keď nepotrebujete riadiť frekvenciu vykonávania kódu, potrebujete poskytnúť procesoru čas na vykonanie iných úloh, ako napríklad reagovanie na používateľské rozhranie. Táto sekcia uvádza niektoré metódy časovania slučiek.

3.25. Funkcie Wait

K uspatiu VI na určený čas vložte funkciu wait do slučky. To umožní vášmu procesoru adresovať iné úlohy počas doby spania. Wait funkcie používajú milisekundové hodiny operačného systému.

Wait Until Next ms Multiple funkcia monitoruje milisekundový čítač a čaká pokým milisekundový čítač dosiahne násobok špecifikovanej hodnoty. Túto funkciu použite na synchronizáciu aktivít. Použite túto funkciu v slučke na riadenie frekvencie vykonávania. K tomu aby táto funkcia bola účinná, čas behu kódu musí byť kratší ako čas špecifikovaný pre túto funkciu. Frekvencia prvej iterácie slučky je neurčitá.

Wait (ms) funkcia čaká pokým milisekundový čítač sa inkrementuje špecifikovaný počet krát. Táto funkcia garantuje, že čas vykonania iterácie slučky je aspoň v trvaní špecifikovanej hodnoty.

3.26. Uplynutý čas

V niektorých prípadoch je užitočné zistiť koľko času ubehlo od nejakého bodu vo VI. Elapsed Time Express VI vyjadrí koľko času ubehlo od špecifikovaného začiatočného času. Toto Express VI sleduje plynutie času počas behu VI. Toto Express VI neposkytne procesoru čas na vykonávanie iných úloh. Elapsed Time Express VI budete používať v projekte Weather Station (Meteo stanica).

3.27. Iteratívny prenos dát

Pri programovaní pomocou slučiek, často musíte mať prístup k dátam z predošlých iterácií. Napríklad, ak nasnímate jeden údaj pri každej iterácii slučky a potrebujete vypočítať priemer z každej pätici, potrebujete uchovať dáta z predošlých iterácií slučky.

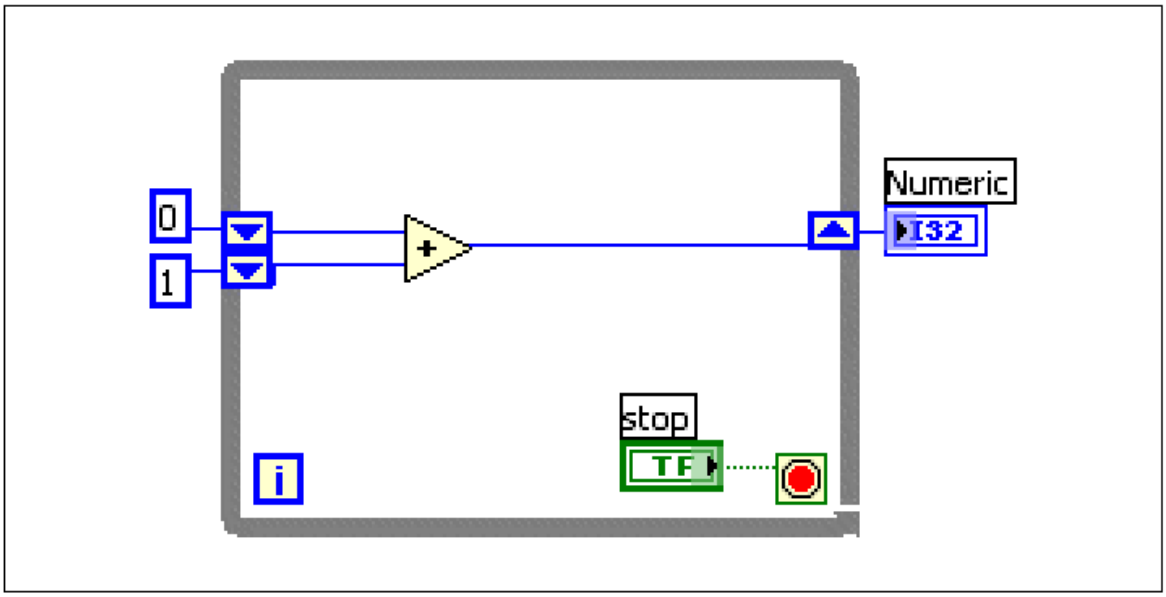
Posuvné registre sú podobné statickým premenným v textovo orientovaných programovacích jazykoch.Použite posuvné registre keď potrebujete preniesť

dáta z predošlých iterácií do nasledujúcej iterácie. Shift register sa zobrazí ako dvojica terminálov umiestnených oproti sebe na vertikálnych okrajoch slučky. Terminál na pravej strane slučky obsahuje šípku smerom hore a uloží dáta na konci iterácie. LabVIEW prenesie dáta pripojené na pravý terminál registra do ďalšej iterácii. Po ukončení behu slučky, terminál na pravej strane vracia poslednú zapísanú hodnotu do posuvného registra.
Posuvný register vytvoríte pomocou pravého kliknutia na ľavý alebo pravý okraj slučky a voľbou Add Shift Register z kontextovej ponuky.
Posuvný register prenáša ľubovoľný dátový typ a automaticky sa zmení na dátový typ prvého pripojeného objektu. Dáta ktoré pripojíte na terminály posuvného registra musia byť totožného typu.
Do slučky môžete pridať viacero posuvných registrov. Ak máte viacero výpočtov ktoré používajú hodnotu z predošlej iterácie použite viaceré posuvné registre na uloženie dát, tak ako je to zobrazené na nasledujúcom obrázku.

3.28. Hromadné posuvné registre

Hromadné posuvné registre vám umožnia prístup k dátam z predošlých iterácií slučky. Hromadné posuvné registre si uchovávajú hodnoty z predošlých iterácií a prenášajú tieto hodnoty do ďalších iterácií. K tvorbe hromadného posuvného registra, kliknite pravým na ľavý terminál registra a zvoľte Add Element z kontextovej ponuky.

Hromadné posuvné registre sú umiestnené iba na ľavej strane slučky, nakoľko pravý terminál prenáša dáta iba z aktuálne vykonanej iterácii do ďalšej iterácii



Používanie hromadných posuvných registrov

Ak pridáte ďalší prvok do ľavého terminálu v predošlom blokovom diagrame, hodnoty z posledných dvoch iterácií sa prenesú do ďalšej iterácii, pričom najaktuálnejšia hodnota je uložená v hornom terminály shift registra. Nižšie položený terminál obsahuje dáta z predošlej iterácie.

3.29. Zobrazovanie dát

Grafy a charty ste už použili na zobrazenie dát. V tejto sekcii sa dozviete viac o používaní a nastavení grafov a chartov.

3.30. Waveform Chart

Waveform chart je špeciálny numerický indikátor ktorý zobrazuje jeden alebo viacero kriviek dát, ktoré sú typicky nasnímane s konštantnou frekvenciou. Waveform chart môže zobraziť jednu alebo viacero kriviek.

Nastavte spôsob zobrazovania a zaktualizovania nových dát Pravým tlačidlom myši kliknite na chart a z kontextovej ponuky zvoľte Advanced»Update Mode na zmenu spôsobu zobrazovania dát. Chart používa nasledovné módy na zobrazovanie dát:

- Strip Chart—priebežne zobrazuje dáta, posúvaním zľava doprava, pričom staršie dáta sú naľavo a novšie napravo. Strip chart sa podobá na papierový páskový záznamník. Strip Chart je predvolený spôsob zobrazovania.
- Scope Chart—zobrazuje skupinu dát, ako napr. priebeh alebo pulz, s čiastočným posunom zľava doprava. Každá nová hodnota je vykreslená napravo od predchádzajúcej. Keď krivka dosiahne pravú hranu zobrazovacej plochy, LabVIEW zmaže krivku a začína znova vykresľovať od ľavej hrany zobrazovacej plochy. Tento spôsob zobrazovanie je podobný k osciloskopu.
- Sweep Chart—Podobá sa na scope chart s tou výnimkou, že staré dáta napravo a nové dáta naľavo sú oddelene s vertikálnou čiarou. LabVIEW nezmaže krivku keď dosiahla pravú hranu zobrazovacej plochy. Sweep chart je podobný k EKG displeju.

3.31. Waveform Graf

Aplikácie ktoré používajú grafy obyčajne zozbierajú dáta do pola a následne zobrazia dáta na grafe.

Grafy umiestnené na palete Graph Indicators zahŕňajú waveform graf a XY graf. Waveform graf zobrazí iba body opísateľné funkciou y = f(x) pomocou rovnomerne rozdelených bodov na ose x, napr. nasnímané priebehy v časovej doméne. XY grafy ľubovoľnú skupinu dát, nezávisle na tom, či boli nasnímané v rovnomerných intervaloch.

Rozšírte popis kriviek tak aby sa zobrazilo viacero kriviek. Použite viaceré krivky na jednom grafe k úspore miesta na čelnom panely a k porovnaniu jednotlivých kriviek. XY a waveform grafy sa automaticky adaptujú na zobrazenie viacerých kriviek.

3.32. Waveform graf s jednou krivkou

Waveform graf akceptuje viaceré dátové type pre zobrazenie jednej krivky. Graf akceptuje pole hodnot, interpretuje dáta ako body na grafe, a inkrementuje x index vždy o jedna, so začiatkom x = 0. Graf akceptuje aj cluster s inicializačnou hodnotu x, inkrementom x, a

polom y hodnôt. Zároveň, graf akceptuje aj dátový typ waveform (priebeh), ktorý nesie dáta o začiatočnom čase t0, delta t (časový inkrement), a pole y hodnôt.

Viac príkladov o dátových typoch nájdete vo Waveform Graph VI v knižnici labview\examples\general\graphs\gengraph.llb .

3.33. Waveform graf s viacerými krivkami

Waveform graf akceptuje viaceré dátové type pre zobrazenie viacerých kriviek. Waveform graf akceptuje 2D dáta hodnôt, kde každý riadok hodnôtreprezentuje jednu krivku. Graf interpretuje dáta ako body na krivke a inkrementuje index x o jedna, začínajúc s x = 0. Ak chcete interpretovať krivky podľa stĺpcov, pripojte 2D pole ku grafu, následne pravým tlačidlom myši kliknite na graf a zvoľte Transpose Array z kontextovej ponuky. Táto možnosť je vyslovene užitočná keď zbierate údaje z viacerých kanálov súčasne z DAQ zariadenia, nakoľko zariadenie môže vracať dáta ako 2D pole, kde každý kanál je zapísaný v osobitnom stĺpci.

Príklad na graf ktorý akceptuje zmienený dátový typ nájdete vo Waveform Graph VI v časti (Y) Multi Plot 1 graph v knižnici labview\examples\general\graphs\gengraph.llb.

Waveform graf akceptuje aj cluster so začiatočnou hodnotou x, s hodnotou delta x, a 2D pole y hodnôt y. Graf interpretuje hodnoty y ako body na krivke, a inkrementuje index x s hodnotou delta x, začínajúc od začiatočnej hodnoty x. Tento dátový typ je užitočný pre zobrazovanie viacerých signálov ktoré sú nasnímané s totožnou, nemennou frekvenciou. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete vo Waveform Graph VI v časti (Xo = 10, dX = 2, Y) Multi Plot 2 graph v knižnici labview\examples\general\graphs\gengraph.llb.

Použite pole clustrov namiesto 2D pola ak je počet prvkov pre každú krivku rôzny. Napríklad, ak snímate dáta z viacerých kanálov s rozdielnymi dobami snímania, použite túto štruktúru namiesto 2D pola, nakoľko každý riadok 2D pola musí mať rovnaký počet prvkov. Počet prvkov vo vnútornom poli tejto štruktúry sa môže meniť. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete vo Waveform Graph VI v časti (Y) Multi Plot 2 graph v knižnici labview\examples\general\graphs\gengraph.llb.

Waveform graf akceptuje aj cluster so začiatočnou hodnotou x, s hodnotou delta x, a s polom ktoré obsahuje clustre. Každý cluster obsahuje 1D pole ktoré obsahuje hodnoty y. Použite funkciu Bundle na zviazanie polí do clustrov a použite funkciu Build Arra vytvorenie pola z clustrov. Taktiež môžete použiť funkciu Build Cluster Array, ktorá vytvorí polia clustrov ktoré obsahujú špecifikované vstupy. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete vo Waveform Graph VI v časti (Xo = 10, dX = 2, Y) Multi Plot 3 graph v knižnici labview\examples\general\graphs\gengraph.llb.

Waveform graf akceptuje aj pole clustrov so začiatočnou hodnotou x, s hodnotou delta x, a s polom y hodnôt y. Tato štruktúra je najbežnejšie používaným dátovým typom pre waveform graf s viacerými krivkami, nakoľko môžete špecifikovať osobitný začiatočný bod a inkrement pre každú krivku osobite. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete vo

Waveform Graph VI v časti (Xo = 10, dX = 2, Y) Multi Plot 1 graph v knižnici labview\examples\general\graphs\gengraph.llb.

Waveform graf akceptuje aj dynamický dátový typ, ktorý sa používa s Express VI. Dynamický dátový typ okrem dátovej reprezentácii signálov obsahuje aj atribúty ktoré poskytnú informácie o signáloch, ako napríklad názov signálu, dátum a čas zberu. Atribúty určia ako sa signál zobrazí na waveform grafe. Keď dynamický dátový typ obsahuje viacero kanálov, graf pre každý kanál automaticky zobrazí krivku a naformátuje popis kriviek a časovú pečiatku osi x.

3.34. XY graf s jednou krivkou

XY graf akceptuje tri dátové type pre zobrazenie XY kriviek. XY graf akceptuje cluster ktorý obsahuje pole x a pole y. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete v XY Graph VI v časti (X and Y arrays) Single Plot graph v knižnici labview\examples\general\graphs\gengraph.llb.

XY graf akceptuje aj pole bodov, kde každý bod je cluster ktorý obsahuje hodnotu x a hodnotu y. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete v XY Graph VI v časti (Array of Pts) Single Plot graph v knižnici labview\examples\general\graphs\gengraph.llb. XY graf taktiež akceptuje aj pole komplexných dát, pričom reálna časť sa zobrazí na osi x a imaginárna na osi y.

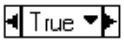
3.35. XY graf s viacerými krivkami

XY graf akceptuje viaceré tri dátové typy pre zobrazenie viacerých kriviek. XY graf akceptuje aj pole kriviek, kde každá krivka je cluster ktorý obsahuje hodnotu pole x a pole y. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete v XY Graph VI v časti (X and Y arrays) Multi Plot graph v knižnici labview\examples\general\graphs\gengraph.llb.

XY graf taktiež akceptuje pole clustrov kriviek, kde krivka je reprezentovaná polom bodov. Bod je cluster ktorý sa skladá z hodnoty x a z hodnoty y. Príklad na graf ktorý akceptuje zmienený dátový typ nájdete v XY Graph VI v časti (Array of Pts) Multi Plot graph v knižnici labview\examples\general\graphs\gengraph.llb. XY graf akceptuje aj pole clustrov kriviek, kde krivka je pole komplexných dát, pričom reálna časť sa zobrazí na osi x a imaginárna na osi y.

3.36. Štruktúra Case

Case štruktúra má 2 alebo viacero subdiagramov, resp. prípadov (case). Iba jeden subdiagram je viditeľný naraz, a štruktúra vykoná iba jeden prípad. Vstupná hodnota určí ktorý subdiagram sa vykoná. Case štruktúra je podobná k príkazom switch alebo if... then...else v textovo orientovaných jazykoch.



Návestie prípadu na hornej časti Case štruktúry obsahuje hodnotu podmienky ktorej prípad je zobrazený a šípky na inkrementáciu a dekrementáciu.K posunu na ďalšie prípady kliknite na šípky inkrementácie a dekrementácie. Môžete kliknúť na šípku dole vedľa názvu prípadu a zvoliť prípad z výsuvnej ponuky.

Na určenie ktorý prípad sa má vykonať pripojte vstupnú hodnotu, resp. selektor na terminál podmienky.
K terminálu podmienky musíte pripojiť integer, boolean, string, alebo enum. Terminál podmienky môžete umiestniť na ľubovoľnú pozíciu v rámci ľavého okraja Case štruktúry. Ak je dátový typ terminálu podmienky boolean, štruktúra bude mať prípady True a False. Ak je terminál podmienky integer, string, alebo enum, štruktúra môže mať ľubovoľný počet prípadov.
Ak nezadáte predvolený (default) prípad v Case štruktúre na obsluhu hodnôt mimo rozsahu, musíte explicitne definovať každú možnú vstupnú hodnotu. Napríklad, ak je selektor integer a špecifikujete prípady pre 1, 2, a 3, následne musíte špecifikovať predvolený prípad ktorý sa vykoná ak vstupná hodnota je 4 alebo ľubovoľná iná ktorá nie je explicitne špecifikovaná.
Ak chcete previesť Case štruktúru na Stacked Sequence štruktúru, kliknite pravým tlačidlom myši na Case štruktúru a zvoľte Replace with Stacked Sequence z kontextovej ponuky.
K pridaniu, kopírovaniu, odstráneniu, zmeny poradia jednotlivých prípadov, a výberu predvoleného prípadu, kliknite pravým tlačidlom myši na okraj case štruktúry.
Ak do návestia prípadu zapíšete selektor ktorý nie je totožného typu ako objekt pripojený na terminál podmienky, zapísaná hodnota sa vyfarbí na červeno. Značí to, že VI nie je spustiteľné pokým neodstránite alebo nezmeníte túto hodnotu. Kvôli možným chybám zaokrúhľovania case štruktúra nepodporuje návestia podmienok s číslami s pohyblivou rádovou čiarkou. Ak pripojíte číslo s pohyblivou rádovou čiarkou na terminál podmienky, LabVIEW zaokrúhli túto hodnotu na najbližšie celé číslo. Ak zapíšete číslo s pohyblivou rádovou čiarkou do návestia prípadu, zapísaná hodnota sa vyfarbí na červeno, čo znamená, že VI nie je spustiteľné pokým neodstránite alebo nezmeníte túto hodnotu.

3.37. Vstupné a výstupné tunely

Môžete vytvoriť viacero vstupných a výstupných tunelov pre Case štruktúry. Vstupy sú prístupné pre všetky prípady, ale jednotlivé prípady nemusia použiť každý vstup. Avšak, výstupný tunel musíte definovať pre každý prípad.

Napríklad, Case štruktúra na blokovom diagrame obsahuje výstupný tunel, ale aspoň v jednom zo všetkých prípadov nie je žiadna hodnota pripojená na tento tunel. Ak by sa tento prípad vykonal, LabVIEW by nevedel určiťakú hodnotu má obsahovať výstupný tunel. LabVIEW zobrazuje túto chybu pomocou nevyplneného štvorca. Prípad ktorý obsahuje nepripojený výstupný tunel nemusí byť ten ktorý je aktuálne zobrazený na blokovom diagrame.

Chybu opravíte ak pripojíte hodnotu na všetky výstupné tunely. Ďalšou možnosťou je kliknúť pravým tlačidlom myši na výstupný tunel a zvoliť Use Default If Unwired z kontextovej ponuky. Týmto spôsobom tunel použije predvolenú hodnotu pre daný dátový typ. Keď výstup je pripojený vo všetkých prípadoch, výstupný tunel má tvar vyplneného štvorca.

Vyhýbajte sa použitiu voľby Use Default If Unwired. Používanie tejto možnosti nie je ideálne z dokumentačného hľadiska a môže spliesť iných vývojárov. Zároveň, voľba Use Default If Unwired robí hľadanie chýb ťažším. Ak používate túto možnosť, nezabudnite, že predvolená

hodnota ktorá sa použije pre tá ktorá je predvolená pre dátový typ ktorý je pripojený k tunelu. Napríklad, ak tunel je boolean dátového typu, predvolená hodnota je FALSE.

4. VZŤAH MEDZI DÁTAMI

V niektorých situáciách je užitočné spolu súvisiace dáta zoskupiť. Použite polia a clustre na zoskupenie súvisiacich dát v LabVIEW. Polia zhromaždia dáta rovnakého typu do jednej štruktúry, clustre zhromaždia dáta rôznych dátových typov do jednej štruktúry. Použite typové definície na vytvorenie opisu na mieru šitých polí a clustrov. Táto lekcie pojednáva o poliach, clustroch, typových definíciách, a aplikáciách kde ich použitie môže byť výhodné.

4.1. Polia

Pole sa skladá z prvkov pola a z dimenzií. Prvky sú dáta ktoré tvoria pole. Dimenzia je dĺžka, výška, alebo hĺbka pola. Polia môžu mať jednu alebo viacero dimenzií a maximálne (231) – 1 prvkov pre každú dimenziu, ohraničené pamäťou.

Môžete vytvoriť polia numeric, boolean, path, string, waveform a cluster dátových typov. Uvážte použitie polí, ak pracujete s množinou podobných dát a keď vykonávate opakujúce sa výpočty. Polia sú ideálne na uloženie dát ktoré pochádzajú z priebehov alebo sú generované v slučkách, kde každá iterácie slučky vytvorí nový prvok pola.

4.2. Obmedzenia

Nemôžete vytvoriť pole polí. Avšak, môžete použiť viac dimenzionálne pole alebo vytvoriť pole z clustrov kde každý cluster bude obsahovať jedno alebo viac polí. Nemôžete vytvoriť pole subpanel kontroliek, tab kontroliek, .NET kontroliek, ActiveX kontroliek, chartov, alebo XY grafov s viacerými krivkami. Viac informácií ohľadne clustrov nájdete v cluster sekcii tejto lekcie.

Príkladom na jednoduché pole je textové pole ktoré ktoré obsahuje dvanásť mesiacov roka. Takúto štruktúru LabVIEW reprezentuje ako 1D pole stringov s dvanástimi prvkami.

Prvky pola sú zoradené. Pole používa index na prístup k danému prvku. Index sa začína od nuly, čo znamená, že jeho rozsah je od 0 do n – 1, kde n je počet prvkov v poli. Napríklad, pre dvanásť mesiacov roka, n = 12, tým pádom rozsah indexu je od 0 do 11. Marec je tretí mesiac, teda má index 2.

4.3. Tvorba pola kontroliek a indikátorov

Pole kontroliek alebo indikátorov na čelnom okne vytvoríte pomocou vloženia prázdneho prvku typu pole, ako je to zobrazené v nasledujúcom príklade, a vložením indikátora alebo kontrolky do poľa. Indikátory alebo kontrolky môžu byť typu numeric, boolean, string, path, refnum, alebo cluster.

LabVIEW vám neumožní vložiť nevhodný indikátor alebo kontrolku do prázdneho pola.

Pred použitím pola v blokovom diagrame musíte do pola vložiť vhodný objekt. V opačnom prípade terminál pola bude čierny s prázdnymi zátvorkami a bez asociácie na dátový typ.

4.4. Dvojdimenzionálne polia

Predošlé príklady pracovali s 1D polom. 2D pole usporiada prvky v tvare mriežky. K lokalizácii prvku je potrebný index riadka a stĺpca, pričom oba indexy sa začínajú od nuly. Obrázok 5-3 zobrazuje 2D pole s ôsmimi riadkami a s ôsmimi stĺpcami, ktoré obsahuje 8 × 8 = 64 prvkov.

Viacdimenzionálne pole vytvoríte na čelnom panely s pravým kliknutím na návestie pre index a voľbou Add Dimension z kontextovej ponuky. Môžete aj meniť veľkosť návestia s tým pridávať nové dimenzie.

4.5. Inicializácia polí

Polia môžete inicializovať alebo nechať neinicializované. Pole inicializujete pomocou určenia počtu prvkov pre každú dimenziu a určením hodnoty pre každý prvok. Neinicializované pole obsahuje nastavený počet dimenzií, ale žiadne prvky.

4.6. Tvorba konštanty typu pole

Ak chcete vytvoriť konštantu pola na blokovom diagrame, zvoľte prvok array constant z palety Functions , vložte prázdny prvok pola do blokového diagramu a následne vložte string konštantu, numeric konštantu, boolean konštantu, alebo konštantu clustra do pola. Konštantu pola môžete použiť ako úložisko nemenných dát alebo ako základ pre porovnanie s iným polom.

Auto-indexácia vstup typu pole

Ak pripojíte pole k slučke For alebo While, môžete zosynchronizovať iterácie slučky s postupným spracovaním prvkov pola pomocou auto-indexácie. Ikona tunela sa zmení z vyplneného štvorca na ikonu ktorá označuje auto-indexáciu. Pravým tlačidlom myši kliknite na tunel a zvoľte Enable Indexing alebo Disable Indexing z kontextovej ponuky pre zmenu indexácie.

4.7. Vstupy typu pole

Keď povolíte auto-indexáciu pola pripojeného na vstupný terminál For slučky, LabVIEW nastaví počet iterácii slučky podľa počtu prvkom v poli, tým pádom nemusíte túto hodnotu pripojiť na terminál počtu iterácií. Nakoľko slučky For môžete použiť na spracovanie polí po prvkoch, LabVIEW ako predvolenú možnosť, nastaví auto-indexáciu vstupného pola. Auto-indexáciu môžete zrušiť, ak nepotrebujete spracovávať pole po prvkoch.

V prípade ak povolíte auto-indexáciu u viacerých vstupných polí, alebo ak zapojíte terminál iterácií, počet skutočných iterácií bude najmenšia hodnota zo všetkých možností. Napríklad, dve auto-indexované polia s počtom prvkov 10 a 20 sú pripojené na vstup a zároveň hodnota 15 je pripojená k terminálu iterácií, slučka sa vykoná iba 10-krát, spracuje všetky prvky prvého pola a prvých 10 prvkov druhého pola.

4.8. Tvorba dvojdimenzionálnych polí

Môžete použiť dve vnorené For slučky na tvorbu 2D pola. Vonkajšia For slučka vytvorí riadky, vnútorná For slučka zasa stĺpce

4.9. Clustre

Clustre zoskupujú prvky rôznych dátových typov. Príkladom môže byť error cluster, ktorý obsahuje boolean hodnotu,numerickú hodnotu a string. Cluster je podobný štruktúram record alebo struct v textovo orientovaných programovacích jazykoch.

Zoskupenie viacerých prvkov do clustra redukuje počet potrebných spojení na blokovom diagrame a zároveň aj počet terminálov na konektorovom panely ktoré sú potrebné pre subVI. Konektorový panel môže mať maximálne 28 terminálov. Ak váš čelný panel obsahuje viac ako 28 kontroliek a indikátorov ktoré chcete priložiť na vstup iného VI, zoskupte ich do clustrov a ku clustrom priraďte terminál na konektorovom panely.

Väčšina clustrov na blokovom diagrame má ružovú farbu spojenia a terminálu. Error clustre majú tmavo žltú farbu spojení a terminálov. Clustre numerických hodnôt, označované aj ako body, majú hnedú farbu spojení a terminálov. Hnedé numerické clustre môžete pripojiť k numerickým funkciám, ako napr. Add, alebo Square Root. Vykonajú sa zvolené operácie na všetkých prvkoch clustra súčasne.

4.10. Poradie prvkov v clustri

Poradie prvkov v clustri ako aj v poli je usporiadané. V prípade clustra musíte prvky vybaliť pomocou funkcie Unbundle. Môžete použiť funkciu Unbundle By Name na rozbalenie prvkov podľa mena. Ak použijete funkciu Unbundle by Name, každý prvok v clustri musí mať návestie (label). Clustre sa odlišujú od polí aj v tom, že majú nemennú veľkosť. Podobne ako pole, cluster je alebo kontrolka alebo indikátor. Cluster nemôže obsahovať zmes kontroliek a indikátorov.

4.11. Tvorba cluster kontroliek a indikátorov

Cluster kontroliek alebo indikátorov na čelnom okne vytvoríte pomocou vloženia prázdneho prvku typu cluster, ako je to zobrazené v nasledujúcom príklade, a vložením indikátora alebo kontrolky do clustra. Indikátory alebo kontrolky môžu byť typu numeric, boolean, string, path, refnum, alebo iný cluster.

Zmeňte veľkosť prázdneho prvku typu cluster pri jeho vložení do okna čelného panelu pomocou ťahu myšou v procese vkladania (pred uvoľnením tlačidla myši).

4.12. Tvorba konštanty typu cluster

Ak chcete vytvoriť konštantu clustra na blokovom diagrame, zvoľte prvok cluster constant z palety Functions , vložte prázdny prvok clustra do blokového diagramu a následne vložte string konštantu, numeric konštantu, boolean konštantu, alebo konštantu clustra do clustra.

Konštantu clustra môžete použiť ako úložisko nemenných dát alebo ako základ pre porovnanie s iným clustrom.

Ak už máte cluster kontrolku alebo indikátor na okne čelného panelu a chcete vytvoriť v blokovom diagrame cluster konštantu s totožnými prvkami, môžete pretiahnuť daný cluster z čelného panelu do blokového diagramu, alebo pravým tlačidlom myši kliknúť na cluster v blokovom diagrame a zvoliť Create»Constant z kontextovej ponuky.

4.13. Poradie v clustri

Prvky clustra majú logické poradie ktoré sa nevzťahuje na ich umiestnenie. Prvý objekt ktorý vložíte do clustra má poradové číslo 0, druhý má1 atď. Ak vymažete prvok, poradie sa upraví automaticky. Poradie prvkov v clustri určí poradie v ktorom sa tieto prvky zobrazia ako terminály pri použití Bundle a Unbundle funkcií na blokovom diagrame. Môžete zobraziť a meniť poradie prvkov pomocou pravého kliknutia na okraj clustra a voľbou Reorder Controls In Cluster z kontextovej ponuky.

4.14. Používanie funkcií pre prácu s clustrom

Použite funkcie pre prácu s clustrom k tvorbe a modifikácii clustrov.

- Napríklad, môžete vykonať nasledovné úlohy:
- Extrahovať individuálne prvky z clustra
- Pridať individuálne prvky do clustra
- Rozložiť cluster na jednotlivé prvky

Použite funkciu Bundle na zostavenie clustra, použite funkciu Bunde a Bundle by Name na zmenu clustra, a použite funkciu Unbundle resp. Unbundle by Name na rozklad clustra. Funkcie Bundle, Bundle by Name, Unbundle, a Unbundle by Name môžete vložiť na blokový diagram aj pomocou pravého kliku na terminál clustra a voľbou Cluster, Class & Variant Palette z kontextovej ponuky. Funkcie Bundle a Unbundle automaticky obsahujú správny počet terminálov. Funkcie Bundle by Name a Unbundle by Name sa zobrazia s prvým prvkom v clustri. Použite nástroj Positioning tool na rozšírenie funkcií Bundle by Name a Unbundle by Name tak, aby zobrazili aj ostatné prvky clustra.

4.15. Zostavenie clustrov

Použite funkciu Bundle na zostavenie clustra z jednotlivých prvkov alebo na zmenu hodnôt jednotlivých prvkov v existujúcom clustri bez nutnosti špecifikácie nových hodnôt pre všetky prvky. Použite nástroj Positioning tool na rozšírenie funkcie alebo kliknite pravým tlačidlom myši na vstup funkcie a zvoľte Add Input z kontextovej ponuky.

4.16. Zmena clustra

Ak zapojíte vstup cluster do funkcie, potom môžete zapojiť iba prvky ktoré chcete zmeniť.

Môžete použiť aj funkciu Bundle by Name na nahradenie alebo prístup k pomenovaným prvkom v existujúcom clusteri. Funkcia Bundle by Name pracuje podobne ako funkcia

Bundle, ale namiesto odkazovania sa na prvky clustra podľa ich poradia, používa návestia prvkov. Pristúpiť viete iba k prvkom s návestím. Počet vstupov sa nemusí zhodovať s počtom prvkov v output cluster.

Použite nástroj Operating tool, kliknite na vstupný terminál z vyberte prvok z lišty. Môžete aj pravým kliknúť na vstup a zvoliť prvok z Select Item kontextovej ponuky.

Použite funkciu Bundle by Name pri štruktúrach ktoré sa môžu zmeniť počas vývoja. Ak pridáte nový prvok do clustra, alebo modifikujete poradie prvkov, nemusíte zmeniť spojenia funkcie Bundle by Name, nakoľko názvy sú stále platné.

4.17. Rozklad clustrov

Použite funkciu Unbundle na rozklad clustra do jednotlivých prvkov.

Použite funkciu Unbundle by Name na extrakciu prvkov ktoré určíte podľa názvu. Počet výstupných terminálov nezávisí na počtu prvkov vstupného clustra.

Použite nástroj Operating tool, kliknite na výstupný terminál z vyberte prvok z lišty. Môžete aj pravým kliknúť na výstup a zvoliť prvok z Select Item kontextovej ponuky.

4.18. Error clustre

LabVIEW obsahuje cluster ktorý sa nazýva error cluster. LabVIEW používa error cluster na prenos informácií o chybe. Error cluster obsahuje nasledovné prvky:

- status—boolean hodnota ktorá má hodnotu TRUE ak nastala chyba
- code— 32-bit signed integer ktorý numericky identifikuje chybu.
- source—string ktorý identifikuje kde nastala chyba.

4.19. Typové definície

Typové definície môžete použiť na vytvorenie na mieru šitých polí a clustrov.

4.20. Upravené kontrolky (Custom controls)

Upravené kontrolky a indikátory použite na rozšírenie existujúcej množiny objektov čelného panelu. Môžete vytvoriť upravené komponenty používateľského rozhrania, ktoré sa grafiky líšia od vstavaných LabVIEW kontroliek a indikátorov. Upravenú kontrolku alebo indikátor môžete uložiť do súboru alebo do LLB, a následne túto kontrolku alebo indikátor použiť aj na iných čelných paneloch. Môžete vytvoriť aj ikonu pre upravenú kontrolku a indikátor a pridať ju do palety Controls .

Viac informácií ohľadne tvorbe a používania upravených kontroliek a indikátorov nájdete v téme Creating Custom Controls, Indicators, and Type Definitions pomocníka LabVIEW Help.Použite okno Control Editor na úpravu kontroliek a indikátorov. Napríklad, môžete zmeniť veľkosť, farbu, relatívnu pozíciu prvkov kontrolky alebo indikátora a importovať obrázky do kontrolky alebo indikátora.

Okno Control Editor môžete zobraziť nasledovnými spôsobmi:

- Pravým tlačidlom myši kliknite na kontrolku alebo indikátor na čelnom panely a zvoľte Advanced»Customize z kontextovej ponuky.
- Použite nástroj Positioning tool na výber kontrolky alebo indikátora na čelnom panely a zvoľte Edit»Customize Control.
- Použite dialógové okno New.

Okno Control Editor sa zobrazí spolu s vybratými objektmi čelného panela. Control Editor má dva režimy, režim editácie (edit mode) a režim úprav (customize mode).

Nástrojová lišta okna Control Editor udáva v ktorom režime sa nachádzate. Okno Control Editor sa otvorí v režime editácie. Do režimu úprav sa prepnete pomocou tlačidla Change to Customize Mode . Spätne, do režimu editácie sa prepnete tlačidlom Change to Edit Mode . Prepnúť medzi režim môžete aj voľbou Operate»Change to Customize Mode alebo Operate»Change to Edit Mode.

Použite režim editácie na zmenu veľkosti alebo farby kontrolky resp. indikátora a k voľbe možností z kontextovej ponuky, ako v prípade editácie na čelnom panely.

Použite režim úprav na realizáciu rozsiahlych zmien kontroliek a indikátorov pomocou zmeny jednotlivých súčastí kontrolky a indikátora.

4.21. Režim editácie (Edit mode)

V režime editácie môžete kliknúť pravým tlačidlom myši na kontrolku a meniť nastavenia tak, ako by ste to robili v LabVIEW programovacom prostredí.

4.22. Režim úprav (Customize mode)

V režime úprav môžete navzájom premiestňovať jednotlivé súčasti kontrolky. Zoznam manipulovateľných súčastí si zobrazíte pomocou Window»Show Parts Window.

Jednou z možností úprav kontrolky je zmena statusu typovej definície. Kontrolku môžete uložiť ako kontrolka, typová definícia, striktná typová definícia, v závislosti od výberu vo voľbe Type Def. Status (Status typovej definície). Voľba control je totožná ako kontrolka ktorú by ste si vybrali z palety Controls. Môžete kontrolku modifikovať podľa vašich predstáv, pričom každá kópia kontrolky si udrží svoje individuálne vlastnosti.

4.23. Uloženie upravených kontroliek

Po vytvorení upravenej kontrolky si ju môžete uložiť pre použitie v budúcnosti. Podľa predvoleného nastavenia, kontrolky sú uložené na disk s príponou .Cl.

Pomocou Control Editoru môžete uložiť kontrolky aj s vašimi predvolenými nastaveniami. Napríklad, pomocou Control Editoru môžete modifikovať predvolené nastavanie waveform grafu, uložiť kontrolku a následne ju použiť aj v iných VI.

4.24. Typové definície

Použite typové definície a striktné typové definície na prepojenie všetkých inštancií upravenej kontrolky alebo indikátora so súborom upravenej kontrolky alebo indikátora. Môžete zrealizovať zmeny vo všetkých inštanciách upravenej kontrolky alebo indikátora pomocou úprav v uloženom súbore upravenej kontrolky alebo indikátora. Táto možnosť je užitočná keď používate tú istú upravenú kontrolku alebo indikátor vo viacerých VI.

Keď vložíte upravenú kontrolku alebo indikátor do VI, neexistuje spojenie medzi uloženou kontrolkou alebo indikátorom a inštanciou tejto upravenej kontrolky alebo indikátora vo VI. Každá inštancie upravenej kontrolky alebo indikátora je osobitná, nezávislá kópia. Tým pádom, zmeny ktoré spravíte v súbore upravenej kontrolky alebo indikátora, nemajú vplyv na VI ktoré tieto upravené kontrolky alebo indikátory už používajú. Ak chcete prepojiť inštancie upravenej kontrolky alebo indikátora so súborom upravenej kontrolky alebo indikátora, uložte upravenú kontrolku ako typová definícia alebo striktná typová definícia. Všetky inštancie typovej definície alebo striktnej typovej definície sú napojené na pôvodný súbor odkiaľ boli použité.

Keď uložíte upravenú kontrolku alebo indikátor ako typová definícia, alebo striktná typová definícia, všetky zmeny dátového typu ktoré vykonáte v typovej definícii alebo v striktnej typovej definícii ovplyvnia všetky inštancie (striktnej) typovej definície vo všetkých VI ktoré kontrolku alebo indikátor používajú. Zároveň, vizuálne zmeny ktoré zrealizujete na striktnej typovej definícii ovplyvnia všetky inštancie striktnej typovej definície na čelnom panely.

Typové definície identifikujú správny dátový typ pre každú inštanciu upravenej kontrolky alebo indikátora. Keď sa dátový typ typovej definície zmení, všetky inštancie typovej definície sa automaticky zaktualizujú. Inými slovami, dátový typ inštancií typovej definície sa zmení v každom VI kde je typová definícia použitá. Avšak, nakoľko typové definície identifikujú iba dátové typy, iba tie hodnoty, ktorý sú súčasťou dátového typu sú zaktualizované. Napríklad, pri numerických kontrolkách, rozsah dát nie je súčasťou dátového typu. Z toho dôvodu, typové definície pre numerické kontrolky nedefinujú rozsah dát pre inštancie typovej definície. Zároveň, nakoľko názvy položiek v ring kontrolke nedefinujú dátový typ, zmeny názvov položiek ring kontrolky v typovej definície nemení názvy položiek v inštanciách typovej definície. Avšak, ak zmeníte názvy položiek v typovejdefinícii enum kontrolky, inštancie sa zaktualizujú, nakoľko názvy položiek sú súčasťou enum dátového typu. Inštancia typovej definície môže mať vlastný popis, návestie, opis, tip strip, predvolenú hodnotu, veľkosť, farbu, alebo štýl kontrolky alebo indikátora, ako napríklad otočné tlačidlo namiesto posúvača.

Ak zmeníte dátový typ v typovej definícii, LabVIEW, pokiaľ je to možné, skonvertuje starú predvolenú hodnotu vo všetkých inštanciách typovej definícii na nový dátový typ. LabVIEW nedokáže zachovať predvolenú hodnotu inštancie ak dátový typ sa zmení na nekompatibilný typ, ako napríklad zmena numerickej kontrolky alebo indikátora na string kontrolku alebo indikátor. Keď dátový typ typovej definície sa zmení na nekompatibilný typ s predošlou typovou definíciou, LabVIEW nastaví predvolené hodnoty inštancií podľa predvolených hodnôt špecifikovaných v .ctl súbore. Ak neudáte predvolenú hodnotu, LabVIEW použije

predvolenú hodnotu pre daný dátový typ. Napríklad, ak zmeníte typovú definíciu z numeric na string, LabVIEW nahradí predvolené hodnoty asociované s dátovým typom numeric na prázdne stringy.

4.25. Striktné typové definície

Striktná typová definícia zabezpečí aby všetko, okrem popisu, návestia, opisu, tip stripu a predvolenej hodnoty inštancií bolo identické k striktnej typovej definícii. Podobne ako pri typových definíciách, dátový typ striktnej typovej definície vo všetkých inštanciách totožný. Striktné typové definície definujú aj iné hodnoty, ako napr. kontrola rozsahu pri numerických kontrolkách a názvy položiek v ring kontrolkách. Jediné dostupné vlastnosti VI Servera pre striktnú typovú definíciu sú tie, ktoré majú vplyv na podobu kontrolky alebo indikátora, ako napríklad Visible (viditeľné), Disabled (neaktívne), Key Focus (Fókus klávesy), Blinking (Blikanie), Position (Pozícia), and Bounds (hranice).

Automatickej aktualizácii inštancií striktnej typovej definície neviete zabrániť, iba v prípade, ak odstránite spojenie medzi inštanciou a striktnou typovou definíciou.

Typové definície a striktné typové definície môžu vytvoriť upravenú kontrolku pomocou clustra z viacerých kontroliek. Ak potrebujete pridať novú kontrolku a podať túto novú hodnotu všetkým subVI, môžete pridať novú kontrolku do clustra upravenej kontrolky. Táto možnosť nahrádza nutnosť pridania novej kontrolky na čelný panel všetkých subVI a nutnosť vytvorenia nových spojení a terminálov.

5. SPRÁVA ZDROJOV

Už ste sa naučili ako spraviť zber dát a ako ich zobraziť. V nadväznosti, aj uloženie dát je obyčajne veľmi dôležitá súčasť projektu. Naučili ste sa aj ako nastaviť hardvér a nakonfigurovať ho v Measurement & Automation Explorer. V tejto lekcii sa naučíte ako uložiť dáta, naprogramovať základnú DAQ aplikáciu pomocou DAQmx API, a ako riadiť osobitné prístroje pomocou VISA API a ovládačov prístrojov v LabVIEW.

5.1. Porozumenie súborovým V/V

Súborové V/V zapisujú dáta a čítajú dáta zo súboru Priebeh typických súborových V/V operácií

1. Vytvoríte alebo otvoríte súbor. Po otvorení súboru, jedinečný identifikátor, tak zvaný refnum reprezentuje súbor.

2. Súborové V/V VI alebo funkcie čítajú alebo zapisujú do súboru.

3. Zatvoríte súbor.

5.2. Súborové formáty

LabVIEW dokáže používať a vytvárať nasledovné súborové formáty: Binary, ASCII, LVM, a TDMS.

- Binary—binárne súbory sú základným súborovým formátom pre všetky ostatné súborové formáty.
- ASCII—ASCII súbor je špecifický typ binárneho súboru ktorý je používaným štandardom pre väčšinu programov. Obsahuje sériu ASCII kódov. ASCII súbory sa nazývajú aj ako textové súbory.
- LVM—LabVIEW measurement dátový súbor (.lvm) je tabulátorom oddelený textový súbor ktorý môžete otvoriť pomocou tabuľkového editora alebo textového editora. Súbor .lvm obsahuje informácie ohľadne dát, ako napríklad dátum a čas a generované dáta. Tento súborový formát je špecifický typ ASCII súboru vytvorený pre LabVIEW.
- TDMS—Tento súborový formát je špecifický typ binárneho súboru vytvoreného pre produkty National Instruments. Skladá sa z dvoch oddelených súborov - z binárneho súboru ktorý obsahuje dáta a atribúty dát, a z binárneho index súboru ktorý poskytuje konsolidované informácie o všetkých atribútoch a smerníkoch v binárnom súbore.

V tomto kurze sa zaoberáme s tvorbou textových (ASCII) súborov. Použite textové súbory keď chcete mať prístup k súboru z inej aplikácie, ak obsadený priestor na disku a rýchlosť súborových V/V nie je kľúčová, ak nepotrebujete prístup k ľubovoľnej pozícii pri operáciách čítania a písania, a ak numerická presnosť nie je dôležitá.

Adresár LabVIEW Data

Predvolený adresár LabVIEW Data môžete použiť na uloženie dát ktoré LabVIEW generuje, ako napríklad .lvm alebo .txt súbory. LabVIEW vytvorí adresár LabVIEW Data v predvolenom súborovom adresári pre vás operačný systém,

uľahčí tým organizáciu a zisťovanie umiestnenia dátových súborov ktoré sú vytvorené v LabVIEW. Štandardne, Write LabVIEW Measurement File Express VI uloží do tohto adresára .lvm súbory ktoré vytvorí a Read LabVIEW Measurement File Express VI číta z tohto adresára. Konštanta Default Data Directory a property uzol Default Data Directory štandardne vracajú adresár LabVIEW Data.

K zmene tohto predvoleného adresára zvoľte Tools»Options a následne Paths zo zoznamu Category. Predvolený adresár pre dáta je rozdielny od predvoleného adresára, ktorý špecifikuje umiestnenie pre nové VI, upravené kontrolky, VI šablóny, a iné LabVIEW dokumenty ktoré vytvoríte.

5.3. Porozumenie vysokoúrovňovým súborovým V/V

Niektoré VI pre súborové operácie vykonajú všetky tri kroky V/V procesu - otvorenie, načítanie/zápis a zatvorenie. Ak VI vykoná všetky tri kroky, považuje sa za vysokoúrovňové VI. Avšak, tieto VI nemusia byť také účinné ako nízkoúrovňové VI a ako funkcie ktoré sú navrhnuté pre jednotlivé kroky procesu. Ak zapisujete do súboru v slučke, použite nízkoúrovňové V/V VI. Ak zapisujete do súboru v jednom kroku a to neopakovane, môžete použiť vysokoúrovňové V/V VI.

LabVIEW obsahuje nasledovné vysokoúrovňové V/V VI:

- Write to Spreadsheet File—skonvertuje 2D a1D pole čísiel s dvojnásobnou presnosťou na text string a zapíše string do nového ASCII súboru, alebo doplní string to existujúceho súboru. Dáta môžete aj transponovať. VI otvorí alebo vytvorí súbor pred zápisom a po zápise ho zatvorí. Toto VI môžete použiť na vytváranie textových súborov čitateľných pre väčšinu tabuľkových editorov.
- Read From Spreadsheet File—načíta špecifikovaný počet riadkov alebo stĺpcov z numerického textového súboru, začínajúc od špecifikovaného ofsetu a skonvertuje dáta do 2D pola čísiel s dvojnásobnou presnosťou. VI otvorí súbor pred načítaním a po načítaní ho zatvorí. Toto VI môžete použiť na načítanie tabuľky uloženej v textovom formáte.
- Write to Measurement File—expresné VI ktoré zapíše dáta do textovo orientovaného (.lvm) súboru alebo do binárneho (.tdms) formátu. Môžete určiť spôsob uloženia, súborový formát (.lvm alebo .tdms), typ hlavičky a oddeľovači znak.
- Read from Measurement File—expresné VI ktoré načíta dáta z textovo orientovaného (.lvm) súboru alebo z binárneho (.tdms) formátu. Môžete špecifikovať názov súboru, formát súboru a veľkosť segmentu.

Porozumenie nízkoúrovňovým súborovým V/V

Nízkoúrovňové V/V VI a funkcie vykonajú iba jeden krok z procesu práce so súborom. Napríklad, osobitné funkcie existujú na otvorenie ASCII súboru, na načítanie ASCII súboru, a na uzavretie ASCII súboru. Použite nízkoúrovňové funkcie keď zapisujete alebo čítate zo súboru v slučke.

5.4. Streaming na disk pomocou nízkoúrovňových funkcií

Tieto funkcie použite na vytvorenie streamingu na disk, nakoľko dochádza k úspore pamäte pomocou zníženia počtu interakcií aplikácie s operačným systémom, ktorá je nutná kvôli otvoreniu a uzavretiu súboru. Streaming a disk je technika pri ktorej nechávate súbor otvorený počas vykonania viacerých zápisov, napríklad v slučke.

Pripojením path kontrolky alebo konštanty na vstup funkcií Write to Text File, Write to Binary File, alebo Write To Spreadsheet File VI zvýšite réžiu týchto operácií o otvorenie a uzavretie súboru pri každom volaní danej funkcie. VI sú účinnejšie, ak sa vyhnete častému otváraniu a uzatváraniu tých istých súborov.

Otváraniu a uzatváraniu toho istého súboru sa vyhnete pomocou prepojenia refnum identifikátora do slučky. Keď otvoríte súbor, zariadenie, alebo sieťové spojenie, LabVIEW vytvorí refnum ktorý je asociovaný s daným súborom, zariadením alebo sieťovým pripojením. Všetky operácie ktoré vykonáte na súboroch, zariadeniach, alebo sieťových spojeniach použijú refnum na identifikáciu objektu.

5.5. Programovanie DAQ

Základy o snímačoch, signáloch, konfigurácii DAQ zariadení a MAX už poznáte. Teraz sa naučíte používať LabVIEW na vývoj aplikácií za zber dát. Viac informácií ohľadne základov merania, ako napríklad spojenia a šum, nájdete v Prílohe A, Measurement Fundamentals. Tento kurz opisuje vývoj DAQ aplikácií v LabVIEW, hoci NI-DAQmx podporuje aj iné vývojárske prostredie ako LabVIEW.

5.6. Kontrolky DAQmx názvov

Paleta DAQmx Name Controls obsahuje kontrolky pre meno úlohy, kanálu, fyzického kanálu, terminálu, škály, zariadenia a prepínača. Tieto kontrolky môžete vytvoriť aj pomocou pravého kliknutia na príslušný vstupný terminál DAQmx VI a voľbou Create»Control. Viac informácií o týchto kontrolkách nájdete v NI-DAQmx Help.

5.7. DAQmx - VI pre zber dát

NI-DAQmx VI použite s NI-DAQ hardvérom k vývoju aplikácií pre inštrumentáciu, zber a riadenie. Zoznam podporovaných zariadení ovládačom NI-DAQmx nájdete v DAQ Getting Started Guide alebo v NI-DAQ Readme.

Paleta DAQmx - Data Acquisition obsahuje nasledovné konštanty a VI.

5.8. Konštanty

- DAQmx Task Name Constant—obsahuje všetky názvy úloh ktorý ste vytvorili a uložili pomocou DAQ Assistant K obmedzeniu úloh ktoré konštanta zobrazí a k obmedzeniu vstupu do konštanty kliknite pravým na konštantu a zvoľte I/O Name Filtering z kontextovej ponuky.
- DAQmx Global Channel Constant—obsahuje všetky globálne kanály ktoré ste vytvorili a uložili pomocou DAQ Assistant. K výberu viacerých kanálov kliknite na konštantu a vyberte Browse. K obmedzeniu kanálov ktoré konštanta zobrazí a k obmedzeniu vstupu do konštanty kliknite pravým na konštantu a zvoľte I/O Name Filtering z kontextovej ponuky.

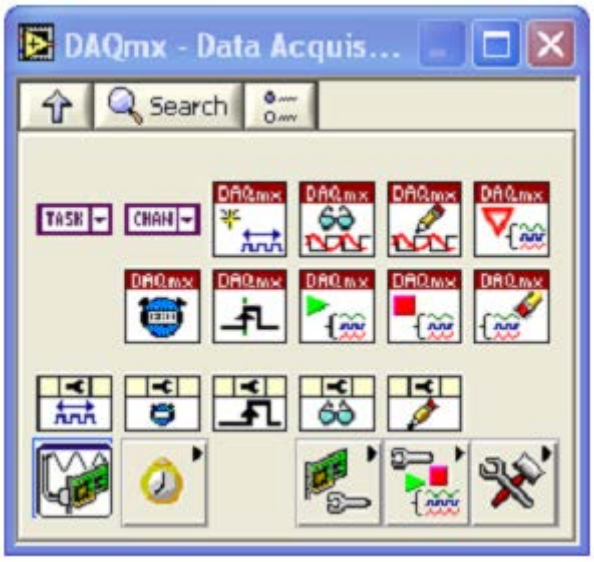
5.9. VI

• DAQmx Create Virtual Channel VI—vytvorí virtuálny kanál, alebo skupinu virtuálnych kanálov a pridá ich do úlohy. Inštancie tejto polymorfickej VI korešpondujú s typom kanála (napr. analógový vstup, digitálny výstup, výstup z čítala), meraním alebo generovaním ktoré sa má vykonať (napr. meranie teploty, generovanie výstupného napätie, počítanie udalostí), a v niektorých prípadoch aj s použitým senzorom (napr. termočlánok alebo RTD pre meranie teploty).

Táto funkcia nastaví tie isté parametre ktoré by ste nakonfigurovali v MAX pri tvorbe virtuálneho kanálu. Použite túto funkciu ak používateľ aplikácie často mení fyzické zapojenia kanálov, pričom ostatné dôležité nastavenia (ako napr. konfigurácia terminálov, aplikovaná škála) zostávajú nemenné. Použite výsuvnú ponuku physical channel na špecifikáciu čísla DAQ zariadenia a reálnych kanálov na ktoré je signál pripojený.

- DAQmx Read VI—načíta vzorky zo špecifikovanej úlohy alebo kanálov Inštancie tejto polymorfickej VI špecifikujú v akom formáte sa majúvzorky vrátiť, či sa má načítať iba jedna vzorka alebo viacero vzoriek naraz, a či načítať vzorky z jedného alebo z viacerých kanálov. Inštanciu vyberiete pomocou pravého kliknutia na DAQmx Read VI a voľbou Select Type.
- DAQmx Write VI—zapíše vzorky do špecifikovanej úlohy alebo na kanály. Inštancie tejto polymorfickej VI špecifikujú formát vzoriek na zápis, či sa zapíše jedna alebo viacero vzoriek, a či sa použije jeden alebo viacero kanálov. Inštanciu vyberiete pomocou pravého kliknutia na DAQmx Write VI a voľbou Select Type.
- DAQmx Wait Until Done VI—Čaká na ukončenie merania alebo generácie dát. Použite toto VI k tomu, aby ste sa uistili, že daná operácia sa ukončila predtým ako ukončíte úlohu.

- DAQmx Timing VI—nakonfiguruje počet vzoriek na akvizíciu alebo generovanie dát a vytvorí zásobník ak je to potrebné. Inštancie tejto polymorfickej VI korešpondujú typu časovania úlohy.
- DAQmx Trigger VI—nakonfiguruje spúšťanie úlohy. Inštancie tejto polymorfickej funkcie korešpondujú k samotnému signálu použitého na spúšťanie.
- DAQmx Start Task VI—zrealizuje prechod úlohy do spusteného stavu (running state) na začatie merania alebo generácie. Použitie tohto VI je povinné pre niektoré aplikácie, pre iné to môže byť iba voliteľné.
- DAQmx Stop Task VI—ukončí úlohu a vráti ju do stavu v ktorom bola pred volaním DAQmx Start Task VI alebo DAQmx Write VI s nastaveným vstupom autostart na TRUE.
- DAQmx Clear Task VI—zmaže úlohu. Pred zmazaním, toto VI ukončí úlohu, a ak je to potrebné, uvoľní všetky zdroje rezervované úlohou. Po zmazaní úlohy ju už nedokážete použiť, len ak ju znova vytvoríte.



5.10. Programovanie riadenia prístrojov

VISA je vysokoúrovňové API ktoré volá nízkoúrovňové ovládače. VISA môže ovládať VXI, GPIB, sériové, alebo na computer-based prístroje a uskutoční volanie priradeného ovládača na základe typu použitého prístroja. Pri ladení VISA problémov pamätajte na to, že problém s VISA môže tkvieť v inštalácii niektorého ovládača ktorý VISA volá.

V LabVIEW, VISA je knižnica funkcií ktoré používate na komunikáciu s GPIB, sériovými, alebo na computer-based prístrojmi. Nepotrebujete používať osobitné V/V palety na programovanie prístrojov. Napríklad, niektoré prístroje disponujú s viacerými typmi rozhraní. Ak by LabVIEW ovládač prístroja bol napísaný pomocou funkcií z palety Instrument I/O»GPIB, ovládač by nefungoval v prípade pripojenia prístroja cez sériové rozhranie. VISA rieši tento problém pomocou skupiny funkcií ktorý nie sú závislé na type rozhrania. Tým pádom väčšina LabVIEW ovládačov prístrojov používa VISA ako V/V jazyk.

5.11. Programovacie terminológia VISA

Nasledovná terminológia je podobná k terminológii použitej pri ovládačoch prístrojov:

- Zdroj (Resource)—ľubovoľný prístroj v systéme, zahrňujúc sériové a paralelné porty.
- Relácia (Session)—ku komunikácii so zdrojom musíte otvoriť reláciu ku zdroju, je to podobné komunikačnému kanálu. Keď otvoríte reláciu na zdroj, LabVIEW vráti VISA číslo relácie, čo je jedinečný refnum na daný prístroj. Toto číslo relácie musíte použiť pri všetkých následných VISA funkcií.
- Deskriptor prístroja (Instrument Descriptor)—presný názov zdroja. Deskriptor špecifikuje typ rozhrania (GPIB, VXI, ASRL), adresu zariadenia (logickú alebo primárnu) a typ VISA relácie (INSTR alebo Event).

Deskriptor pristroja je podobný telefónnemu číslu, zdroj je podobný osobe s ktorou chcete hovoriť, a relácia je podobná telefónnej linke. Každé volanie používa vlastnú linku, skríženie týchto liniek má za následok chybu.

Syntax pre rôzne rozhrania prístrojov

Rozhranie	Syntax
Asynchrónne sériové	`ASRL[`prístroj`][::INSTR]`
GPIB	`GPIB[`prístroj`]::`primárna adresa`[::`sekundárna adresa`][::INSTR]`
VXI pristroj cez vnorený alebo MXIbus kontrolér	`VXI[`prístroj`]::`VXI logická adresa`[::INSTR]`
GPIB-VXI kontrolér	`GPIB-VXI[`prístroj`][::`GPIB-VXI primárna adresa`]::`VXI logická adresa`[::INSTR]`

Namiesto deskriptora prístroja môžete použiť jeho alternatívne meno (alias), ktoré si nastavíte v MAX.

Najviac používané VISA komunikačné funkcie sú VISA Write a VISA Read.

Väčšina prístrojov požaduje zaslanie informácie vo forme príkazu alebodopytu predtým ako môžete prečítať informáciu späť od prístroja. Kvôli tomu, za funkciou VISA Write zvyčajne nasleduje funkcia VISA Read. VISA Write a VISA Read funkcie spolupracujú s ľubovoľným typom komunikácie s prístrojom, a sú rovnaké nezávisle od toho, či sa jedná napr. o GPIB alebo sériovú komunikáciu. Avšak, nakoľko sériová komunikácia požaduje konfiguráciu dodatočných parametrov, komunikáciu po sériovej linke musíte začať volaním VISA Configure Serial Port VI.

5.12. VISA a sériová komunikácia

VISA Configure Serial Port VI vykoná inicializáciu portu identifikovaného pomocou VISA resource name. Timeout nastaví časový limit pre sériovú komunikáciu. Baud rate, data bits,

parity, a flow control určia špecifické parametre sériového portu. Clustre error in and error out slúžia na obsluhu a prenos chýb.

5.13. Používanie ovládačov prístrojov

Predstavte si nasledovný scenár. Napísali ste LabVIEW VI ktoré komunikuje so špecifickým osciloskopom vo vašom laboratóriu. Žiaľ, osciloskop sa pokazil a musíte ho nahradiť. Avšak tento typ osciloskopu už nie je dostupný. Našli ste iný osciloskop, ktorý by ste radi použili, ale vaše VI nespolupracuje s novým osciloskopom. Musíte tým pádom prepísať VI.

Keď používate ovládač prístroja, ovládač obsahuje špecifický kód pre prístroj. Tým pádom, ak zmeníte prístroj, musíte nahradiť iba VI ovládačaprístroja s VI ovládačom nového prístroja. To značne ušetrí čas potrebný na zmenu kódu. Ovládače prístrojov pomáhajú k lepšej a ľahšej udržiavateľnosti testovacích aplikácií, nakoľko ovládače obsahujú všetky V/V pre prístroj v jednej knižnici, osobite od iného kódu. Keď zmeníte hardvér, následná zmena aplikácie je ľahšia, nakoľko ovládač obsahuje kód ktorý je špecifický pre daný prístroj.

5.14. Úvod do ovládačov prístrojov

LabVIEW Plug and Play ovládač prístroja je skupina VI ktoré riadia programovateľný prístroj. Každé VI korešponduje s niektorou funkciou prístroja, ako napríklad konfigurácia, spúšťanie, čítanie hodnôt meraní z prístroja. Ovládače prístrojov uľahčia rozbehnúť používanie prístroja a ušetria čas a náklady na vývoj, nakoľko používateľ nemusí naštudovať programovací protokol pre každý prístroj. Pri dobre zdokumentovaných ovládačoch s otvoreným zdrojovým kódom môžete upraviť ich funkcionalitu pre lepší výkon. Modulárny návrh uľahčí úpravu ovládačov.

5.15. Lokalizácia ovládačov prístrojov

Väčšinu LabVIEW Plug and Play ovládačov nájdete v Instrument Driver Finder. Instrument Driver Finder spustíte pomocou voľby Tools»Instrumentation»Find Instrument Drivers alebo Help»Find Instrument Drivers. Instrument Driver Finder vás spojí s ni.com na vyhľadávanie ovládačov. Pri inštalácii ovládača prístroja sa pridá príklad ako ho použiť do NI Example Finder.

Viacero programovateľných prístrojov má väčší počet funkcií a režimov práce. Pri tejto zložitosti je potrebné ustanoviť konzistentný model návrhu ktorý je nápomocný vývojárom ovládačov prístrojov ako aj koncovým používateľom riadiacich aplikácií. LabVIEW Plug and Play model ovládačov prístrojov obsahuje smernice pre externú aj internú štruktúru. Externá štruktúra sa vzťahuje na rozhranie ovládača prístroja s používateľom a s inými softvérovými komponentmi v systéme. Interná štruktúra sa vzťahuje na internú organizáciu softvérových modulov ovládača.

6. VÝVOJ MODULÁRNYCH APLIKÁCIÍ

Táto lekcia opisuje vývoj modulárnych aplikácií. Sila LabVIEW tkvie v hierarchickej náture VI. Po vytvorení VI, toto VI môžete použiť na blokovom diagrame iného VI. Neexistuje obmedzenie na počet takýchto hierarchických vrstiev. Používanie modulárneho programovania pomôže pri organizácii zmien a rýchlejšom ladení blokového diagramu.

6.1. Základy modularity

Modularita definuje mieru do ktorej je program zostavený z diskrétnych modulov tak, že zmena v jednom moduly má minimálny dopad na ostatné moduly. Moduly v LabVIEW sa nazývajú subVI.

VI v rámci iného VI sa nazýva subVI. SubVI zodpovedá rutine v textovo orientovaných programovacích jazykoch. Pomocou dvojkliku na subVI otvoríte jeho čelný panel a blokový diagram. Čelný panel obsahuje kontrolky a indikátory. Blokový diagram obsahuje spojenia, ikony, funkcie, prípadne subVI a iné LabVIEW objekty ktoré vám už môžu byť známe.

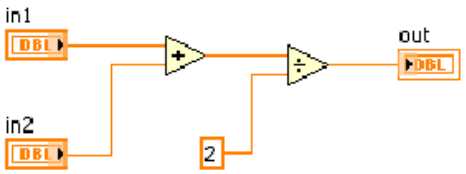
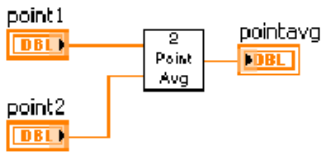
V pravom hornom rohu čelného panela a blokového diagramu je zobrazená ikonu pre VI. Táto ikona sa zobrazí na blokovom diagrame keď vložíte toto VI na blokový diagram iného VI.

Počas vývoja VI môžete zistiť, že určité operácie vykonávate často. Zvážte použitie subVI alebo slučiek na vykonávanie častých operácií. Napríklad, nasledujúci blokový diagram obsahuje dve identické operácie.

Nasledovný pseudo-kód a blokový diagram zobrazujú analógiu medzi subVI s rutinou.

Kód funkcie	**Kód volajúceho programu**

`function average (in1, in2, out)`
`{`
`out = (in1 + in2)/2.0;`
`}`
	`main`
	`{`
	`average (point1, point2, pointavg)`
	`}`

Blokový diagram subVI	**Blokový diagram volajúceho VI**
	

6.2. Tvorba ikony konektorového panelu

Po vývoji čelného panelu a blokového diagramu vytvorte ikonu a navrhnite konektorový panel, tým pádom môžete použiť toto VI ako subVI. Ikona a konektorový panel korešpondujú s prototypom funkcie v textovo-orientovaných programovacích jazykoch. Každé VI zobrazuje ikonu v pravom hornom rohu okna čelného panelu a blokového diagramu.

Ikona je grafická reprezentácia VI. Môže obsahovať text, obrázky a ich kombináciu. Ak používate VI ako subVI, ikona identifikuje subVI na blokovom diagrame VI. Ak pridáte VI do palety, ikona VI sa objaví aj na palete Functions. Dvojklikom na ikonu v okne čelného panelu alebo v okne blokového diagramu ju viete na mieru prispôsobiť alebo editovať.

Ak chcete použiť VI ak subVI, musíte navrhnúť konektorový panel VI.

Konektorový panel je súbor terminálov ktoré zodpovedajú kontrolkám a indikátorom pre dané VI, podobne ako je to u parametrov volania funkcie v textovo orientovaných programovacích jazykoch. Konektorový panel definuje vstupy a výstupy ktoré môžete pripojiť k VI a tým používať VI ako subVI. Konektorový panel príjme dáta na vstupných termináloch, prenesie tieto dáta do blokového diagramu cez kontrolky čelného panela a z indikátorov čelného panela naplní výstupné terminály.

6.3. Tvorba ikony

Ikona je grafická reprezentácia VI.

Každé VI zobrazuje ikonu v pravom hornom rohu okna čelného panelu a blokového diagramu.

Štandardná ikona VI obsahuje číslo ktoré udáva koľko nových VI, maximum 9, ste otvorili od spustenia LabVIEW. Na zrušenie číslovania zvoľte Tools»Options»Front Panel a zrušte voľbu Use numbers in icons of new VIs (1 through 9).

Ikona obsahuje text alebo obrázky. Ak používate VI ako subVI, ikona identifikuje subVI na blokovom diagrame VI. Ak pridáte VI do palety, ikona VI sa objaví aj na palete Functions.

Použite dialógové okno Icon Editor na zmenu ikony VI. Dvojklikom na ikonu v pravom hornom rohu čelného panelu alebo blokového diagramu zobrazíte dialógové okno Icon Editor.

Vytvorte ikonu na grafickú reprezentáciu VI alebo upravenej kontrolky. Použite dialógové okno Icon Editor na tvorbu alebo zmenu ikony VI.

Môžete použiť banner na identifikáciu súvisiacich VI. National Instruments odporúča tvorbu a uloženie bannera ako šablóna. Následne môžete použiť túto šablónu pre tvorbu ikon súvisiacich VI. Modifikujte základnú časť ikony a poskytnite informácie špecifické pre dané VI.

6.4. Uloženie bannera ako šablóna

Vykonajte nasledovné kroky na uloženie bannera ako šablóna.

1. Zobrazte okno Icon Editor pomocou dvojkliku na ikonu v pravom hornom rohu čelného panelu alebo blokového diagramu, alebo pravým kliknutím na ikonu a voľbou Edit Icon.

2. Stlačte klávesy <Ctrl-A> k výberu všetkých používateľských vrstiev ikony a následne kláves <Delete> na vymazanie výberu. Štandardná ikona je jediná vrstva nazvaná VI Icon.

3. Na strane Templates, zvoľte šablónu _blank.png z kategórie VI»Frameworks. Šablóny môžete prehľadávať podľa kategórií alebo podľa kľúčového slova.

4. Použite nástroj Fill tool na pravej strane okna Icon Editor na vyplnenie bannera farbou.

5. Použite nástroj Text tool na zápis textu do bannera. Pokým je text aktívny, môžete posúvať text pomocou šípok.

6. Zvoľte File»Save As»Template na zobrazenie okna Save Icon As a uložte ikonu ako šablóna pre použitie v budúcnosti. LabVIEW uloží šablóny ikoniek ako .png súbory s 256 farbami.

6.5. Tvorba ikony VI zo šablóny

Vykonajte nasledovné kroky na tvorbu ikony VI ktorá používa šablónu ktorá ste vytvorili.

1. Stlačte klávesy <Ctrl-A> k výberu všetkých používateľských vrstiev ikony a následne kláves <Delete> na vymazanie výberu.

2. Na strane Templates vyberte šablónu ktorú ste vytvorili. Šablóny môžete prehľadávať podľa kategórií alebo podľa kľúčového slova.

3. Na strane Icon Text zadajte maximum štyri riadky textu do tela ikony. Môžete zvoliť štýl, zarovnanie, veľkosť a farbu textu. Ak zaškrtnete políčko Center text vertically, okno Icon Editor vycentruje text ikony vertikálne v rámci tela ikony.

4. Na strane Glyphs, vložte glyfy do oblasti Preview.Pomocou kláves <F> alebo <R> môžete počas vkladania horizontálne preklopiť, alebo otočiť glyf. Dvojklikom na glyf vložíte glyf do ľavého horného rohu ikony. Glyfy môžete prehľadávať podľa kategórií alebo podľa kľúčového slova.

5. Na presun glyfu použite nástroj Move tool. Každý glyf je umiestnený na osobitnej vrstve a tým pádom sa glyfy dajú presúvať osobite. Všimnite si, že zvyšok ikony sa zatemní keď zvolíte glyf, tým pádom viete identifikovať výber ktorý premiestňujete.

6. Na ďalšiu zmenu ikony použite nástroje na editáciu ktoré sú na pravej strane okna Icon Editor.Icon Editor vytvorí novú vrstvu pre každé nesúvislé použitie nástrojov pre editáciu. K vytvoreniu novej vrstvy počas súvislého používania editačných nástrojov zvoľte Layers»Create New Layer.

7. (Opcionálne) Zvoľte Layers»Show Layers Page k zobrazeniu strany Layers. Túto stranu použite na konfiguráciu názvu, priehľadnosti, viditeľnosti a poradia vrstiev ikony.

8. Ikonu uložíte do VI pomocou tlačidla OK. Následne môžete okno Icon Editor zatvoriť.

Môžete presunúť grafický súbor zo súborového systému priamo do pravého horného rohu okna čelného panelu, a tým nastaviť túto grafiku ako ikonu VI. Presunúť môžete súbory formátu .png, .bmp, alebo .jpg.

6.6. Nastavenie konektorového panelu

Spojenia definujete pomocou priradenia kontroliek alebo indikátorov z čelného panelu k jednotlivým terminálom konektorového panelu.

Konektorový panel zobrazíte pomocou pravého kliknutia na ikonu v pravom hornom rohu čelného panela a voľbou Show Connector z kontextovej ponuky. Konektorový panel sa zobratí namiesto ikony. Keď si zobrazíte konektorový panel prvý krát, zobrazí sa vzor usporiadania terminálov. Vzor usporiadania terminálov zmeníte pomocou pravého kliknutia a voľbou Patterns z kontextovej ponuky.

Každý obdĺžnik na konektorovom panely reprezentuje terminál. Obdĺžniky použijete na priradenie vstupov a výstupov. Štandardný konektorový panel má usporiadanie 4 × 2 × 2 × 4. Ak predpokladáte v budúcnosti zmeny VI s nutnosťou pridania nových, dodatočných vstup alebo výstupov, ponechajte štandardný konektorový panel a nechajte niektoré terminály nepriradené.

6.7. Výber a modifikácia vzoru usporiadania terminálv

Vzor usporiadania terminálov zmeníte pomocou pravého kliknutia a voľbou Patterns z kontextovej ponuky. Napríklad, môžete zvoliť konektorový panel s dodatočnými terminálmi. Nepoužité terminály môžete nechať nezapojenépokým ich nepotrebujete. Táto flexibilita vám umožní vykonanie zmien s minimálnym vplyvom na hierarchiu aplikácie.

Môžete mať viac kontroliek a indikátorov na čelnom panely ako dostupných terminálov. Použiť môžete maximálne 28 terminálov konektorového panela.

Najčastejšie používaný vzor je zobrazený naľavo. Tento vzor je používaný ako štandard kvôli zjednodušeniu spojení.

Vyhnite sa používaniu konektorového panelu s viac než 16 terminálmi. Aj keď vzory konektorových panelov s viacerými terminálmi sa môžu zdať užitočné, ich napojenie je ťažšie. Ak potrebujete preniesť viac dát, použite clustre.

6.8. Priradenie kontroliek a indikátorov k terminálom

Potom, čo ste si zvolili vzor pre konektorový panel, môžete priradiť kontrolky a indikátory z čelného panelu k terminálom konektorového panelu. Pri priraďovaní kontroliek a indikátorov ku terminálom konektorového panelu, umiestnite vstupy naľavo a výstupy napravo. Zabránite tým komplikovaným resp. mätúcim spojeniam.

K priradeniu terminálu ku kontrolke alebo k indikátoru čelného panela kliknite na terminál na konektorovom panely, následne kliknite na kontrolku alebo indikátor na čelnom panely ku ktorému chcete terminál priradiť. Kliknite na prázdne miesto na čelnom panely. Farba terminálu sa zmení podľa dátového typu priradenej kontrolky alebo indikátora.

Priradenie môžete vykonať aj v opačnom poradí, najskôr vyberte kontrolku alebo indikátor a potom terminál.

Hoci používate nástroj Wiring tool na priradenie terminálov konektorového panelu k kontrolkám a indikátorom čelného panela, žiadne grafické spojenia sa nezobrazia medzi konektorovým panelom a priradenými kontrolkami a indikátormi.

6.9. Používanie subVI

SubVI vložíte na blokový diagram ak kliknete na tlačidlo Select a VI na palete Functions. Dvojklikom vyberiete a vložíte VI do blokového diagramu.

Môžete vložiť aj otvorené VI do blokového diagramu na iného otvoreného VI. Použite nástroj Positioning tool, kliknite na ikonu na čelnom panely alebo na blokovom diagrame VI ktoré chcete použiť ako subVI a pretiahnite ikonu do blokového diagramu iného VI.

6.10. Otvorenie a úprava subVI

Použitím nástroja Operating alebo Positioning tool a dvojklikom na subVI z volajúceho VI otvoríte čelný panel subVI. K zobrazeniu blokového diagramu subVI z volajúceho VI stlačte kláves <Ctrl> a použite nástroj Operating alebo Positioning tool s dvojklikom na subVI v blokovom diagrame volajúceho VI.

Upravovať subVI môžete pomocou nástroja Operating alebo Positioning tool s dvojklikom na subVI v blokovom diagrame. Keď uložíte subVI, zmeny ovplyvnia všetky volania tohto subVI, nie len aktuálnu inštanciu.

6.11. Nastavenie požadovaných, odporučených a opcionálnych vstupov a výstupov

V okne Context Help sú návestia požadovaných terminálov zobrazené zvýrazneným, doporučené terminály obyčajným, a opcionálne terminály nezvýrazneným typom písma. Návestia opcionálnych terminálov sa nezobrazia ak kliknete na tlačidlo Hide Optional Terminals and Full Path v okne Context Help.

Pomocou určenia ktoré vstupy a výstupy sú požadované, doporučené a opcionálne môžete predísť tomu, aby používateľ zapomenul pripojiť dáta k terminálom.

Kliknite pravým tlačidlom myši na terminál na konektorovom panely a zvoľte This Connection Is z kontextovej ponuky. Zaškrtnutie značí aktuálne nastavenie. Zvoľte Required, Recommended, alebo Optional. Na strane Tools»Options»Front Panel môžete zaškrtnúť políčko Connector pane terminals default to required.

Táto voľba nastaví terminály štandardne na požadované namiesto doporučených. Táto voľba sa aplikuje na spojenia vytvorené pomocou nástroja Wiring tool a na subVI ktoré boli vytvorené pomocou Create SubVI.

Pre terminály vstupu, atribút požadovaný znamená, že blokový diagram nie je vykonávateľný pokiaľ požadovaný vstup do subVI nie je zapojený. Atribút požadovaný nie je k dispozícii pre terminály výstupu. V prípadeodporučených a opcionálnych terminálov pre vstupy a výstupy, blokový diagram v ktorom sa volá dané subVI je spustiteľný aj keď tieto terminály nie sú zapojené. Ak nezapojíte tieto terminály, VI nevygeneruje žiadne upozornenie.

Vstupy a výstupy VI z vi.lib sú už označené ako Required, Recommended, alebo Optional. LabVIEW nastaví vstupy a výstupy vyvíjaného VI štandardne na Recommended. Nastavte terminál na požadovaný iba ak VI musí mať k dispozícii daný vstup k správnemu vykonaniu kódu.

6.12. Obsluha chýb v subVI

Informácie o chybách prenášame do a zo subVI pomocou error clustrov. Použitím štruktúry Case, s prípadmi No Error a Error, obslúžime prijaté chyby.

6.13. Tvorba subVI z existujúceho VI

Blokový diagram VI môžete zjednodušiť pomocou skonvertovania sekcie kódu do subVI. Konverziu časti blokového diagramu na subVI vykonáte výberom sekcie kódu, ktorú chcete skonvertovať, pomocou nástroja Positioning tool a následnou voľbou Edit»Create SubVI. Ikona nového subVI nahradí vybranú sekciu blokového diagramu. LabVIEW vytvorí kontrolky a indikátory pre nové subVI, automaticky nakonfiguruje konektorový panel podľa počtu kontroliek a indikátorov a napojí subVI na existujúce spojenia.

Nové subVI používa štandardný vzor pre rozvrh terminálov na konektorovom panely a štandardnú ikonu. Dvojklikom na subVI môžete editovať subVI, konektorový panel, ikonu a uložiť subVI.

Nevyberte viac ako 28 objektov pri tvorbe subVI, nakoľko 28 je maximálny počet možných pripojení na konektorový panel. Ak váš čelný panel obsahuje viac ako 28 kontroliek a indikátorov, zoskupte niektoré z nich do clustrov a ku clustrom priraďte terminál na konektorovom panely.

7. COMPACT RIO ARCHITEKTÚRA

7.1. OVERVIEW AND BACKGROUND

This guide provides best practices for designing NI CompactRIO control and monitoring applications using NI LabVIEW system design software along with the LabVIEW Real-Time and LabVIEW FPGA modules. Controllers based on these modules are used in applications ranging from the control of particle accelerators like the CERN Large Hadron Collider, to hardware-in-the-loop testing for engine electronic control units (ECUs), to adaptive control for oil well drilling, to high-speed vibration monitoring for predictive maintenance. This guide provides tips for developing CompactRIO applications that are readable, maintainable, scalable, and optimized for performance and reliability.

7.2. CompactRIO Architecture

CompactRIO is a rugged, reconfigurable embedded system containing three components: a processor running a real-time operating system (RTOS), a reconfigurable field-programmable gate array (FPGA), and interchangeable industrial I/O modules. The real-time processor offers reliable, predictable behavior and excels at floating-point math and analysis, while the FPGA excels at smaller tasks that require high-speed logic and precise timing. Often CompactRIO applications incorporate a human machine interface (HMI), which provides the operator with a graphical user interface (GUI) for monitoring the system's state and setting operating parameters.

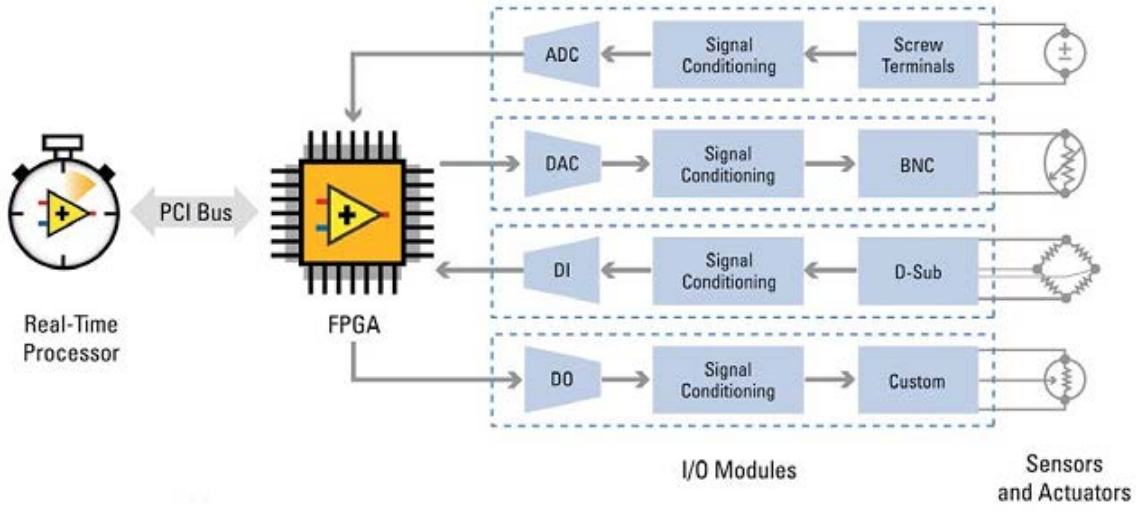


Figure 1. Reconfigurable Embedded System Architecture

7.3. LabVIEW

LabVIEW is a graphical programming environment used by millions of engineers and scientists to develop sophisticated control systems using graphical icons and wires that resemble a flowchart. It offers integration with thousands of hardware devices and provides

hundreds of built-in libraries for advanced control, analysis, and data visualization—all for creating user-defined systems more quickly. The LabVIEW platform is scalable across multiple targets and OSs, and, in the case of CompactRIO, LabVIEW can be used to access and integrate all of the components of the reconfigurable I/O (RIO) architecture.

7.4. Real-Time Controller

The real-time controller contains a processor that reliably and deterministically executes LabVIEW Real-Time applications and offers multirate control, execution tracing, onboard data logging, and communication with peripherals. Additional options include redundant 9 to 30 VDC supply inputs, a real-time clock, hardware watchdog timers, dual Ethernet ports, up to 2 GB of data storage, and built-in USB and RS232 support.

7.5. Real-Time Operating System (RTOS)

A RTOS is able to reliably execute programs with specific timing requirements, which is important for many science and engineering projects. The key component needed to build a real-time system is a special RTOS; other hardware and software pieces that make up an entire real-time system are discussed in the next section.

7.6. Precise timing

For many engineers and scientists, running a measurement or control program on a standard PC with a general-purpose OS installed (such as Windows) is unacceptable. At any time, the OS might delay execution of a user program for many reasons: running a virus scan, updating graphics, performing system background tasks, and so on. For programs that need to run at a certain rate without interruption (for example, a cruise control system), this delay can cause system failure.

Note that this behavior is by design—general-purpose OSs are optimized to run many processes and applications at the same time and provide other features like rich user interface graphics. In contrast, real-time OSs are designed to run a single program with precise timing. Specifically, real-time OSs help you implement the following:

- Perform tasks within a guaranteed worst-case timeframe
- Carefully prioritize different sections of your program
- Run loops with nearly the same timing on each iteration (typically within microseconds)
- Detect if a loop missed its timing goal

7.7. Reliability

In addition to offering precise timing, RTOSs can be set up to run reliably for days, months, or years without stopping. This is important not only for engineers building systems that need 24/7 operation but also for any application where downtime is costly. A “watchdog” feature is also typically included in real-time systems to automatically restart an entire computer if the user program stops running. Furthermore, hardware used in a real-time system is often designed to be rugged to sustain harsh conditions for long periods.

7.8. Reconfigurable FPGA

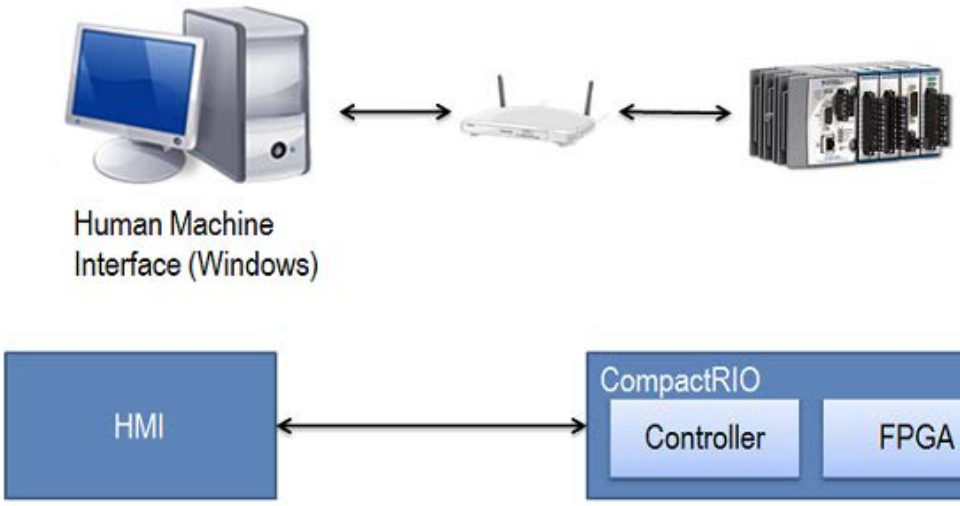
The reconfigurable FPGA chassis is the center of the embedded system architecture. The RIO FPGA is directly connected to the I/O for high-performance access to the I/O circuitry of each module and timing, triggering, and synchronization. Because each module is connected directly to the FPGA rather than through a bus, you experience almost no control latency for system response compared to other controller architectures. By default, this FPGA automatically communicates with I/O modules and provides deterministic I/O to the real-time processor. Out of the box, the FPGA enables programs on the real-time controller to access I/O with less than 500 ns of jitter between loops. You can also directly program this FPGA to further customize the system. Because of the FPGA speed, this chassis is frequently used to create controller systems that incorporate high-speed buffered I/O, fast control loops, or custom signal filtering. For instance, using the FPGA, a single chassis can execute more than 20 analog proportional integral derivative (PID) control loops simultaneously at a rate of 100 kHz. Additionally, because the FPGA runs all code in hardware, it provides the highest reliability and determinism, which is ideal for hardware-based interlocks, custom timing and triggering, or eliminating the custom circuitry normally required with nonstandard sensors and buses. For an overview of FPGA technology

7.9. I/O Modules

I/O modules contain isolation, conversion circuitry, signal conditioning, and built-in connectivity for direct connection to industrial sensors/actuators. By offering a variety of wiring options and integrating the connector junction box into the modules, the CompactRIO system significantly reduces space requirements and field-wiring costs. You can choose from more than 50 NI C Series I/O modules for CompactRIO to connect to almost any sensor or actuator. Module types include thermocouple inputs; ±10 V simultaneous sampling, 24-bit analog I/O; 24 V industrial digital I/O with up to 1 A current drive; differential/TTL digital inputs; 24-bit IEPE accelerometer inputs; strain measurements; RTD measurements; analog outputs; power measurements; controller area network (CAN) connectivity; and secure digital (SD) cards for logging. Additionally, you can build your own modules or purchase modules from third-party vendors. With the NI cRIO-9951 CompactRIO Module Development Kit, you can develop custom modules to meet application-specific needs. The kit provides access to the low-level electrical CompactRIO embedded system architecture for designing specialized I/O, communication, and control modules. It includes LabVIEW FPGA libraries to interface with your custom module circuitry.

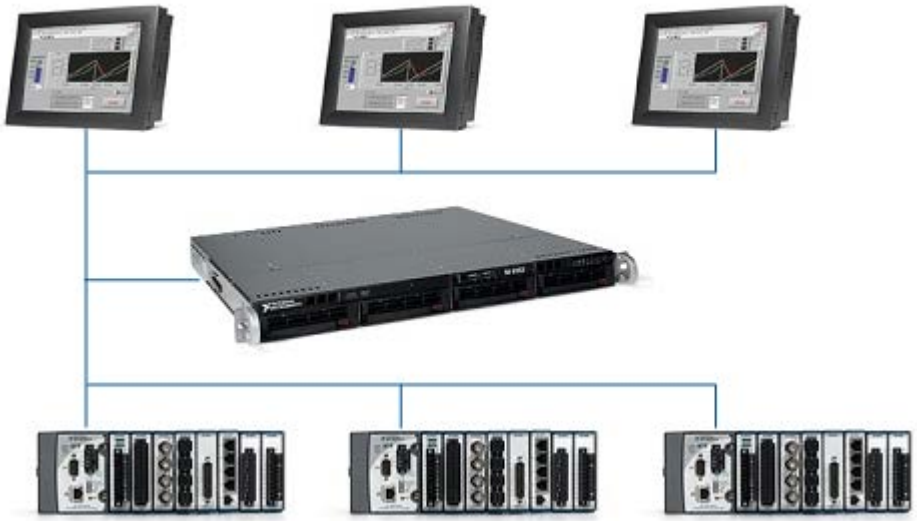
7.10. System Configurations

The simplest embedded system consists of a single controller running in a “headless” configuration. This configuration is used in applications that do not need an HMI except for maintenance or diagnostic purposes. However, a majority of control and monitoring applications requires an HMI to display data to the operator or to allow the operator to send commands to the embedded system. A common configuration is 1:1, or 1 host to 1 target, as shown in Figure 5. The HMI communicates to the CompactRIO hardware over Ethernet through either a direct connection, hub, or wireless router.



The next level of system capability and complexity is either a 1:N (1 host to N targets) or N:1 (N hosts to 1 target configuration). The host is typically either a desktop PC or an industrial touch panel device. The 1:N configuration, shown in Figure 6, is typical for systems controlled by a local operator. The N:1 configuration is common when multiple operators use an HMI to check on the system state from different locations.

Complex machine control applications may have an N:N configuration with many controllers and HMIs (Figure 7). They often involve a high-end server that acts as a data-logging and forwarding engine. This system configuration works for physically large or complex machines. Using it, you can interact with the machine from various locations or distribute specific monitoring and control responsibilities among a group of operators.



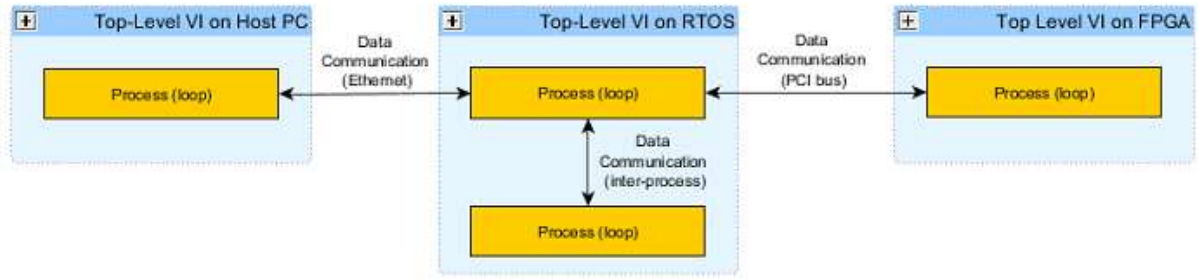
This guide walks through recommended implementations for a variety of FPGA, controller, and HMI architecture components. It also offers example code and, in some cases, alternative implementations and the trade-offs between implementations.

8. NÁVRH SOFTVÉROVEJ ARCHITEKTÚRY PRE COMPACTRIO

Almost all CompactRIO systems have a minimum of three top-level VIs executing asynchronously across three different targets: the FPGA, the real-time operating system (RTOS), and a host PC. If you begin your software development without having some sort of architecture or flowchart to refer to, you may find keeping track of all of the software components and communication paths to be challenging. Having a diagram that describes all of the system pieces at a high level helps guide development and communication about the design with stakeholders. This section describes several common CompactRIO architectures that you can use as a starting point for many applications. You also can use them as a guide for creating your own architecture. The example architectures in the following sections were created using a vector-based drawing program called the yEd Graph Editor, which is a free design tool downloadable from yworks.com.

8.1. CompactRIO Reference Architectures

A CompactRIO system architecture or diagram should provide a basic overview of the system. A basic diagram includes processes and data communication paths. A process, as discussed in this guide, is an independent, asynchronously executing segment of code—basically a loop. The architectures referenced in this guide include processes represented by yellow boxes, the hardware targets that the processes execute on represented by blue boxes, and data communication paths represented by black arrows.



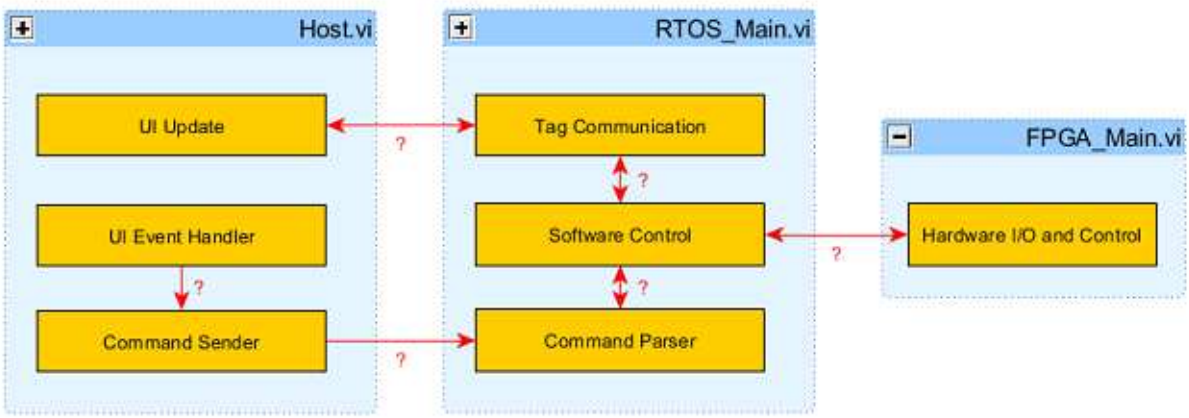
8.2. Processes

The first step in identifying the processes that your application contains is to create a list of tasks that your application needs to accomplish. Common tasks include PID control, data logging, communication to an HMI, communication to I/O, and safety logic or fault handling. The next step is deciding how the tasks are divided into processes. An application with a large number of processes requires more time to be spent on interprocess data communication. At the same time, breaking your tasks into individual processes makes your program more scalable. For example, you might have a real-time application during the first phase of a project that is communicating across the network to a user interface (UI) built with LabVIEW, but later on the application needs to communicate to a web browser. If you separate the network communication task from other tasks within the application, you can redesignyour network communication process without impacting your control process. More information

on designing processes in LabVIEW Real-Time can be found in Chapter 3: Designing a LabVIEW Real-Time Application.

8.3. Data Communication

Once you've diagrammed your processes, the next step is to identify data communication paths. Data communication is one of the most important factors to consider when designing an embedded system. LabVIEW, LabVIEW Real-Time, and LabVIEW FPGA include many different mechanisms for transferring data between processes on a single target as well as processes that communicate across targets.



Choosing a data communication mechanism depends on several factors, with the primary factor being the data communication model. Most CompactRIO applications can contain at least two or more of the data communication models listed below:

- Command-/message-based communication
- Current values (tags)
- Streaming/buffered communication
- Updates

Each model has different data transfer requirements. These models apply to both interprocess communication and intertarget communication. Later chapters refer to these data communication types when recommending LabVIEW mechanisms for data transfer between processes on a Windows target, real-time target, or FPGA target in addition to communication between the CompactRIO device and a host PC.

8.4. Command-/Message-Based Communication

Command- or message-based communication is something that happens relatively infrequently and is triggered by some specific event. An example is a user pushing a button on an HMI to stop a conveyor belt. In message-based communication, it is important to guarantee delivery of the message in a timely manner or with low latency. In the previous example, when the operator pushes the stop button, the operator expects an immediate response (a human's perception of "immediate" is on the order of tenths of a second). It is

often desirable to synchronize a process to the incoming message or command to conserve processor resources. This involves blocking until a message is received or a timeout occurs.

8.5. Current Values (Tags)

Tags are used to communicate current values periodically between processes, or from a controller to an HMI. Tags are commonly used to represent process variables, such as a setpoint for a control system or current temperature data. Often tags are bound to I/O channels within a high-channel-count system such as a machine controller. With current value data or tags, each data transmission does not need to be guaranteed because the controller or HMI is always interested in the latest value, not in historical values. It is often desirable to have features that you can use to look up tags dynamically and manage different groups since tags are usually associated with high-channel counts.

8.6. Streaming/Buffered Communication

Streaming or buffered communication involves the high-throughput transfer of every data point, where throughput is more important than latency. Typically, one process writes data to another process that reads, displays, or processes that data. An example is an in-vehicle data logger that sends data from the FPGA to the RTOS for data logging. Streaming often requires multilayer buffering, during which each end (the sender and the receiver) has its own uniquely sized buffer. This is ideal when you have a processor loop that wants to read large chunks of data at a time and a data acquisition loop that writes smaller chunks of data at shorter intervals.

8.7. Updates

Updates are like a combination of tags and messages. Instead of periodically polling the current value like you would with a tag, you want to wait until the value changes. For example, if you are monitoring the heartbeat of another device on the network, you only are interested in knowing when the value of the heartbeat (dead or alive) changes. It is often desirable to have broadcasting capabilities when dealing with updates.

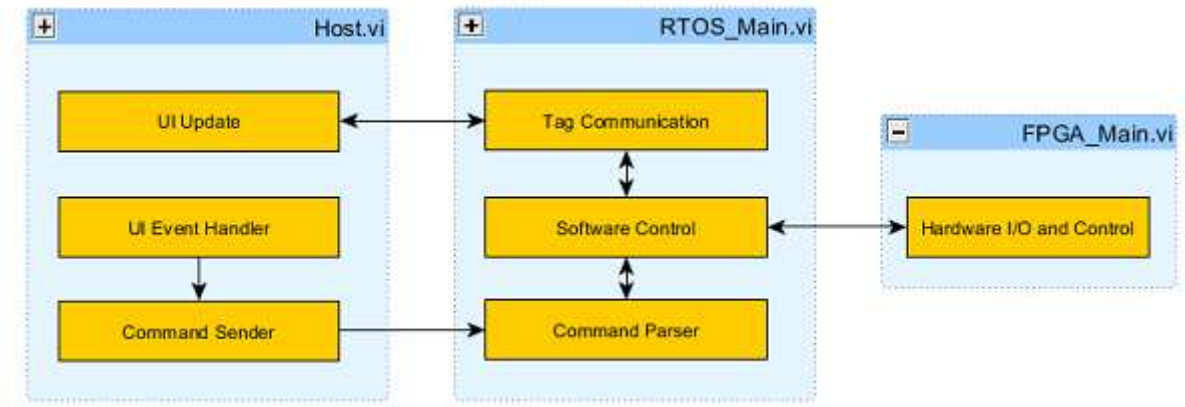
A summary of the different types of data communication models and their features is shown in Table 1.1. Keep these features in mind when selecting a data communication model.

Communication Type	Fundamental Features	Optional Features	Performance
Message-Based	Buffering, blocking (timeout)	Acknowledgment of data transfer	Low latency
Current Value Data (tags)	Nonhistorical	Dynamic tag lookup, group management	Low latency, high-channel count
Updates	Nonhistorical, blocking (timeout)	Broadcasting	Low latency
Streaming/Buffered	Buffering, blocking (timeout)	Multilayer buffering	High throughput

8.8. Typical CompactRIO Architecture

Figure 1.3 shows a common architecture that you can use as a starting point for most control and monitoring applications. The host VI provides an event-based user interface so an operator can interact with the embedded system. The RTOS executes high-level control and the FPGA executes low-level control.

There are two network communication paths: one path for sending commands from the user interface to the CompactRIO hardware and a second path for sending tags (current value data) from the CompactRIO hardware to the user interface for display. To build a scalable application, you should design your system so that it has a single command parser task that you can use to interpret and redistribute the command as needed. This ensures that ongoing critical tasks are not disturbed by the arrival of the command and makes it easy to modify the code to handle additional commands.



On the RTOS_Main.vi, the communication task and control task are designed as two separate loops. This is generally a good practice since the network communication could impact the determinism of the control algorithm if the tasks are executed within a single loop. Also, the modularity makes it easier to simulate parts of the system. For example, you could easily swap out the actual control loop with a simulated control loop to test the network communication.

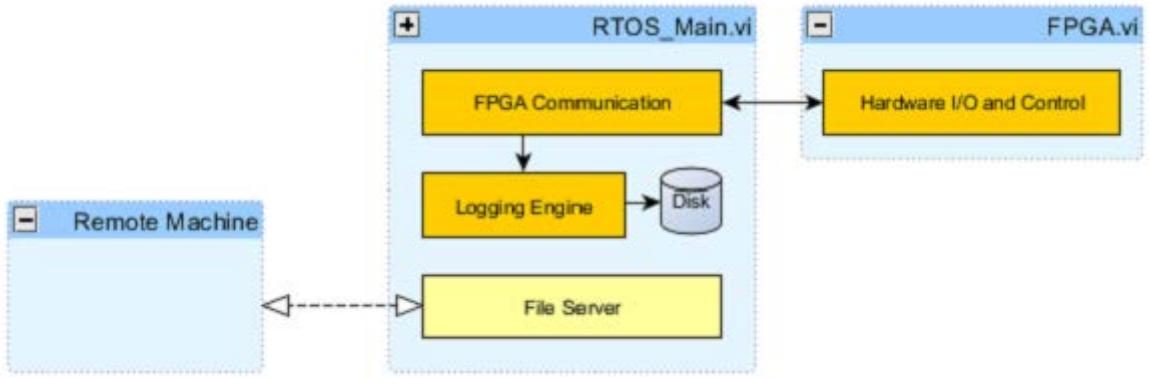
8.9. Common Variant Architectures

The control and monitoring architecture is a great starting point for most applications, but if you want to design an embedded data logger, an embedded monitoring system, or a supervisory control and data acquisition (SCADA) system, you can leverage the more application-specific architectures in the following sections.

Headless Data Logger

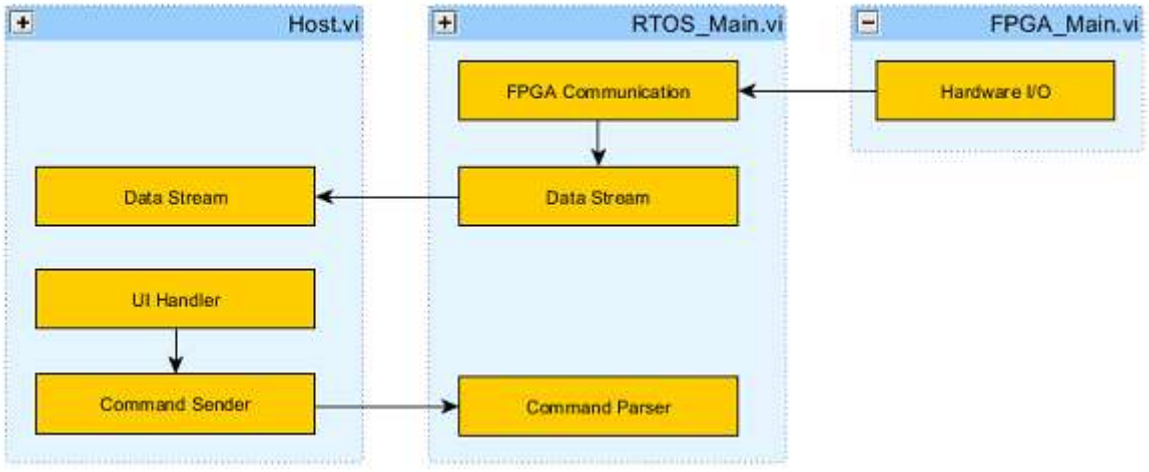
A headless CompactRIO system does not require a user interface running on a remote machine. In the case of a headless data logger, the remote machine might be used for retrieving files that contain logged data from the FTP server. In this architecture, the FPGA is performing low-level control while acquiring data from NI C Series I/O modules at a high

speed. This data is streamed from the FPGA to the RTOS, where it can be logged to the onboard memory. The FPGA communication process is separate from the logging engine processes to maximize the rate at which data is read.



8.10. Embedded Monitoring

Embedded monitoring applications typically require data to be streamed at high rates from the C Series I/O modules up to a host user interface where the data is displayed to an operator. The user interface also allows the operator to configure the hardware, requiring a mechanism for sending commands from the host down to the RTOS.

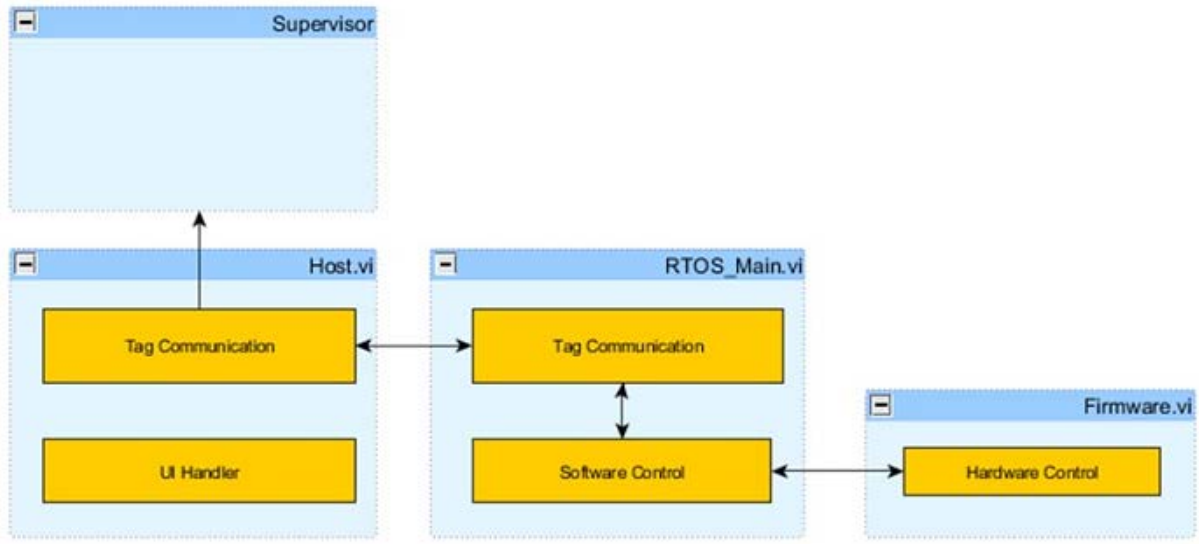


8.11. Supervisory Control and Data Acquisition (SCADA) Architecture

In a typical SCADA application, the CompactRIO device communicates I/O channels as tags to a supervisor. Some control might be implemented on either the FPGA or RTOS. Often you can use the RIO Scan Interface to handle I/O for these types of applications since they do not require high-speed data acquisition rates or customized hardware. The NI Scan Engine is discussed in Chapter 2: Choosing a CompactRIO Programming Mode. With the LabVIEW

Datalogging and Supervisory Control (DSC) Module, you can extend the software capabilities of a SCADA system to include the ability to log data to a historical database, set alarms, and manage events. LabVIEW DSC also features support for commonly used industrial protocols

including OLE for process control (OPC), so LabVIEW can communicate to third-party programmable logic controllers (PLCs).



8.12. Example—Bioreactor

LabVIEW example code is provided for this section.

An example of a bioreactor control system demonstrates how the control and monitoring architecture in Figure 1.6 could apply to an actual application. A bioreactor is a mechanical vessel in which organisms are cultivated and materials are transformed through chemical reactions. The vessel's environmental conditions like gas, flow rates, temperature, pH levels, and agitation speed need to be closely monitored and controlled. A diagram of a basic bioreactor is shown in Figure 1.7.

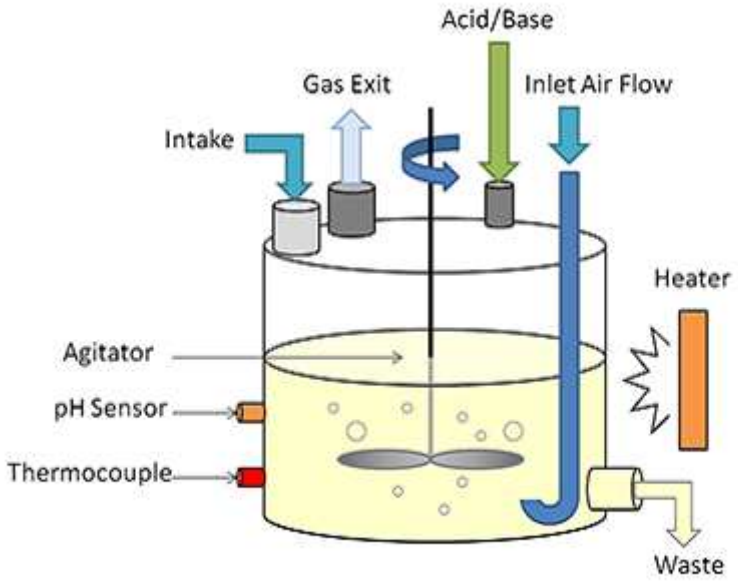


Figure 1.7 Example of a Simplified Bioreactor System

The bioreactor control system has the following software requirements:

- Control of the pH level, temperature, and agitator speed
- UI that allows the operator to do the following:
- Define the setpoint of the temperature, pH level, and agitator speed
- Define and run recipes
- Abort a running recipe and override recipe setpoints
- Monitor current pH level and temperature

Table 1.2 is an example of a recipe:

Phase 1	Phase 2	Phase 3
Temp 35 °C	Temp 40 °C	Temp 38 °C
pH 1.5	pH 1.2	pH 1.2
Agitator ON	Agitator OFF	Agitator ON
Intake: ON	Intake: OFF	Intake: OFF
Waste: OFF	Waste: ON	Waste: OFF
Time: 5 minutes	Time: 2 minutes	Time: 5 minutes

Table 1.2. Example Recipe for Bioreactor Application

To create an architecture for the bioreactor system, start with the control and monitoring architecture shown in Figure 1.8.

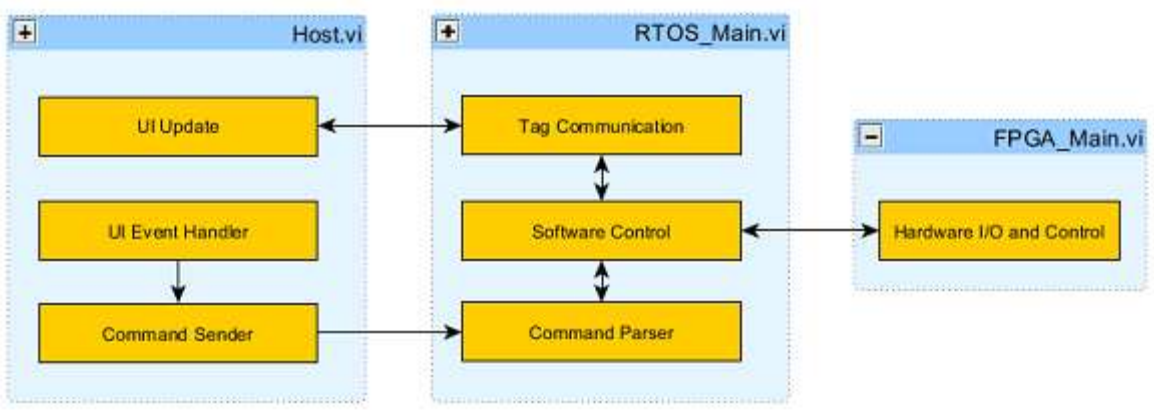


Figure 1.8. Common Software Architecture Control and Monitoring Applications

Since this application interfaces to single-point I/O and does not require update rates faster than 500 Hz, the I/O was implemented using the RIO Scan Interface. The RIO Scan Interface, included in the LabVIEW Real-Time Engine, helps you access your I/O channels as variables within the LabVIEW Real-Time development environment. Find more information on using the RIO Scan Interface versus LabVIEW FPGA in Chapter 2: Choosing a CompactRIO Programming Mode.

Figure 1.9 shows an example of a design diagram for the bioreactor application. Data communication mechanisms are labeled on each arrow. The following process components were reused from the example control and monitoring architecture in Figure 1.8:

- UI Update—Receives the latest pH and temperature values as network-published I/O variables (IOVs) from the RIO Scan Interface and displays them to a user interface.
- UI Event Handler—Uses an event structure to handle user events, such as Run Recipe or Temperature Override.
- Command Sender—Sends any commands received from the UI Event Handler to the CompactRIO controller using network streams.
- Command Parser—Receives commands from the Command Sender process and sends them to the Recipe Engine process using queues (Q). It sends override commands to the control task using single-process shared variables (SP-SV). Queues are not used because they impact control-loop synchronization.
- Control Task—Controls the pH and temperature using a PID algorithm; is executed at a higher priority than other processes on the real-time system.

The following processes were added:

- Recipe Engine—Manages and executes incoming recipes. It shares recipe data (temperature setpoint = 25, pump ON, and so on) with the control loops using single-process shared variables.
- Pump ON/OFF—Controls the state of the pumps and agitation.

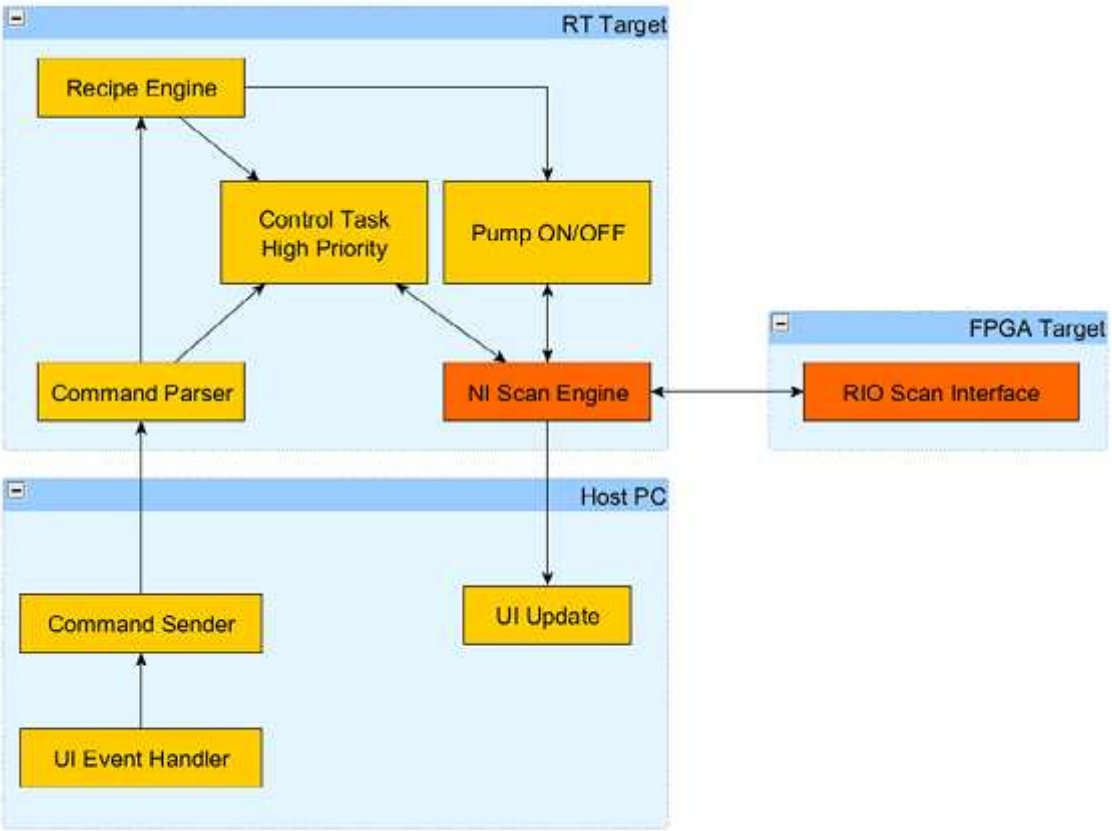


Figure 1.9. CompactRIO Architecture of a Bioreactor Application

To view and run the actual bioreactor application, download the Bioreactor Example from Section 1 Downloads.

9. PRIDANIE V/V BODOV DO COMPACTRIO SYSTÉMU PRE ROZŠÍRENIE MERANIA A RIADENIA

NI RIO products are increasingly adopted for system-level applications that require high-channel counts, intensive processing, and distributed I/O. Adding RIO expansion I/O to the NI RIO product offering enables a 1:N system topology with one controller and many FPGA I/O nodes for flexible, high-channel-count systems that can perform both distributed control and centralized processing.

NI offers three types of expansion chassis systems for expanding your I/O. Each expansion chassis features an FPGA and is based on the same C Series module architecture. The main differences between expansion I/O options stem from the choice of bus, with each bus best suited for a subset of expansion I/O applications. The NI I/O expansion chassis systems include the following:

- MXI-Express RIO
- Ethernet RIO
- EtherCAT RIO

Figure 71. Expand your CompactRIO I/O with NI expansion chassis.

Each expansion chassis has unique capabilities. Choose an expansion chassis depending on your application requirements. Table 7.1 provides a quick comparison of the three different types of expansion chassis.

	MXI-Express RIO	Ethernet RIO	EtherCAT RIO
	NI 9157, 9159	NI 9148	NI 9144
Slots	14	8	8
FPGA	Virtex-5 (LX85 or LX110)	Spartan 2M	Spartan 2M
Network Topology	Daisy chain	Same as Ethernet	Daisy chain
Distance	7 m between nodes	100 m before repeater	100 m before repeater
Multichassis Synchronization	FPGA-based DIO	FPGA-based DIO	Implicit in bus operation
Communication Jitter	*<10 µs*	No Spec	*<1 µs*
Bus Throughput	250 MB/s	100 Mbit/s	100 Mbit/s
API Support	FPGA Host Interface	FPGA Host Interface/Scan Engine	FPGA Host Interface/Scan Engine
Host	Windows/Real-Time	Windows/Real-Time	Real-Time Only

Table 7.1. Comparison of RIO Expansion Chassis

9.1. MXI-Express RIO

Featuring a high-performance Xilinx Virtex-5 FPGA and a PCI Express x1 cabled interface, an NI 9157/9159 MXI-Express RIO 14-slot chassis for C Series I/O expands the RIO platform for large applications requiring high-channel counts and a variety of signal conditioning and custom processing and control algorithms. You can use these MXI-Express RIO expansion chassis with high-performance NI cRIO-9081/9082 CompactRIO systems in addition to industrial controller and PXI systems. When using a MXI-Express RIO expansion chassis, you interface to the expansion chassis I/O within your main controller VI using LabVIEW FPGA Host Interface functions.

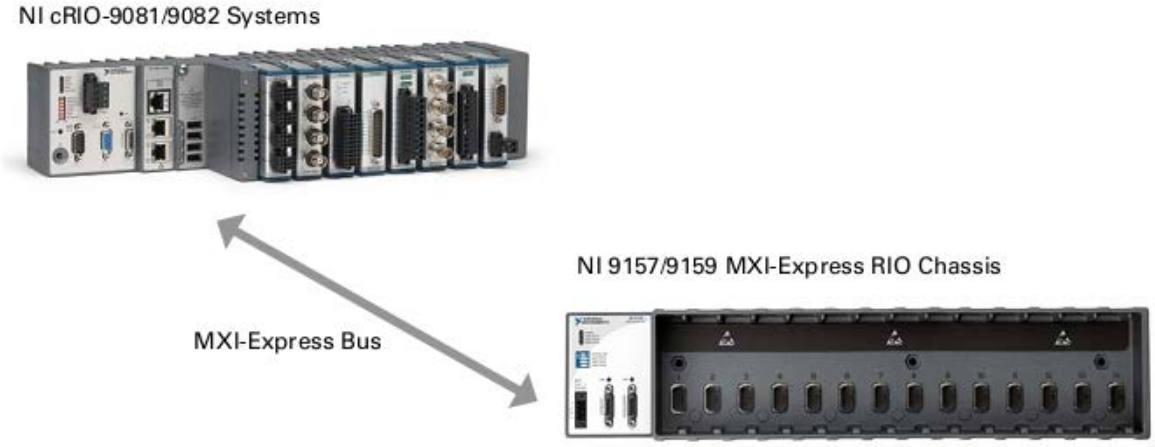


Figure 7.2. A MXI-Express RIO Chassis Paired With an NI cRIO-9081/9082 High-Performance CompactRIO System

9.2. Ethernet RIO

When expanding your I/O using the standard Ethernet protocol, you can use the NI 9148 Ethernet RIO expansion chassis. Programmers can take advantage of the existing network infrastructure such as switches and routers. Although full duplex switch networks eliminate packet collisions, switches introduce jitter, and you should use general Ethernet only in applications that do not require deterministic communication. If you need synchronization between the local I/O and expansion I/O, see the EtherCAT RIO section for more information.

Figure 7.3. The NI 9148 Ethernet RIO Expansion Chassis

When using the Ethernet RIO expansion chassis, the main controller is responsible for running the real-time control loop using I/O from its own chassis in addition to the I/O from one or more Ethernet RIO chassis. The expansion chassis provides expansion or distributed I/O for the main controller.

The Ethernet RIO expansion chassis works with both LabVIEW FPGA and the Scan Engine. If you use LabVIEW FPGA with the expansion chassis, you can embed decision-making capabilities to quickly react to the environment without host interaction. The FPGA can also offload processing from the main controller by conducting inline analysis, custom triggering, and signal manipulation.

When using LabVIEW FPGA, you should create a regular While Loop or a Timed Loop with a lower priority to handle the communication since Ethernet is nondeterministic (see Figure 7.4). This allows your control task to run deterministically and reliably because it is not affected by the possibly high-jitter I/O device. When using LabVIEW FPGA, you interface to the I/O within your real-time VI using the FPGA Host Interface functions.

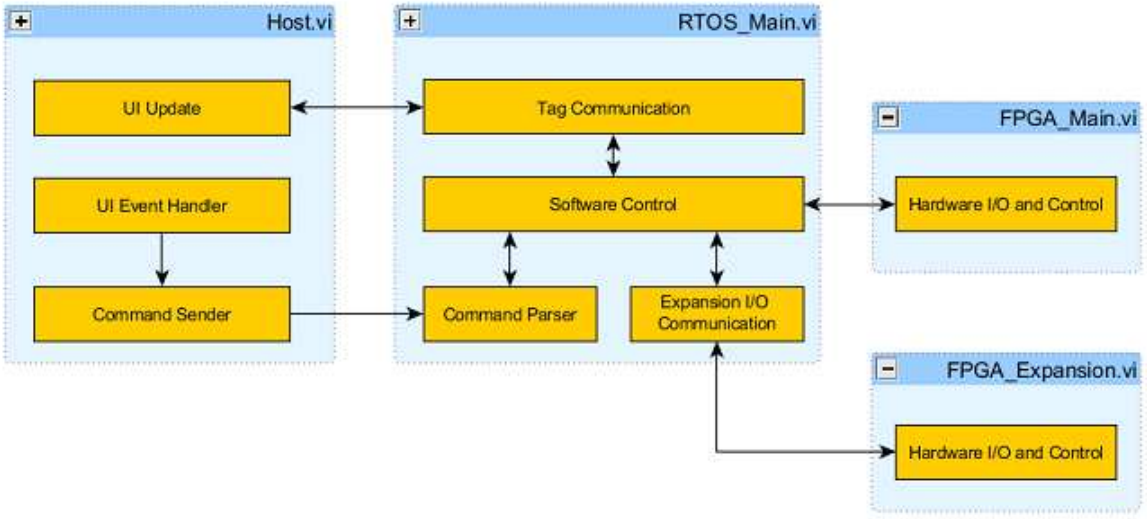


Figure 7.4. Add a new process to handle the I/O expansion task when using LabVIEW FPGA Interface Mode.

The Ethernet RIO expansion chassis also works with the Scan Engine. You have the option to choose Scan Mode or FPGA Interface Mode when adding the Ethernet RIO chassis to your LabVIEW project. When using Scan Mode, your design diagram might look like Figure 7.5, where you have all of your system I/O accessible from the Scan Engine.

When using Scan Mode, you interface to the I/O within your real-time VI using the Scan Engine I/O variables.

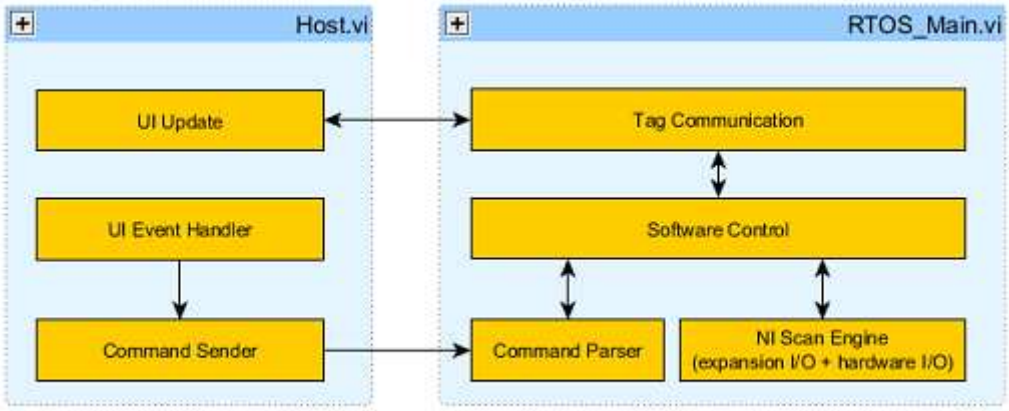


Figure 7.5. You can use the NI Scan Engine to handle I/O with the NI 9148 Ethernet RIO chassis.

To get started with the NI 9148 Ethernet RIO expansion chassis, see the NI Developer Zone tutorial Getting Started With the NI 9148 Ethernet RIO Expansion Chassis.

9.3. EtherCAT RIO

In certain applications, the main I/O and expansion I/O systems require tight synchronization—all of the inputs and outputs must be updated at the same time. Using a deterministic bus allows the main controller to know not only when the expansion I/O is updated but also exactly how long the data takes to arrive. You can easily distribute

CompactRIO systems with deterministic Ethernet technology using the NI 9144 expansion chassis.

The NI 9144 is a rugged expansion chassis for expanding CompactRIO systems using a deterministic Ethernet protocol called EtherCAT. With a master-slave architecture, you can use any CompactRIO controller with two Ethernet ports as the master device and the NI 9144 as the slave device. The NI 9144 also has two ports that permit daisy chaining from the controller to expand time-critical applications.

Figure 7.6. The CompactRIO hardware architecture expands time-critical systems using the NI 9144 deterministic Ethernet chassis.

9.4. Tutorial: Accessing I/O Using the EtherCAT RIO Chassis

The following step-by-step tutorial provides instructions for configuring your EtherCAT system and communicating I/O with both the Scan Engine and LabVIEW FPGA. This example uses an NI cRIO-9074 integrated system, but any CompactRIO system with two Ethernet ports can work with EtherCAT.

Step 1: Configure the Deterministic Expansion Chassis

1. To enable EtherCAT support on the CompactRIO controller, you must install the NI-Industrial Communications for EtherCAT driver on the host computer. This driver is on the CD included with the NI 9144 chassis and on ni.com for free download.

2. To configure the deterministic Ethernet system, connect Port 1 of the cRIO-9074 to the host computer and connect Port 2 to the NI 9144 using standard Ethernet cables. To add more NI 9144 chassis, use the Ethernet cable to connect the OUT port of the previous NI 9144 to the IN port of the next NI 9144.

Figure 7.7. Hardware Setup for Ethernet RIO Tutorial

Configure the CompactRIO Real-Time Controller to Be the Deterministic Bus Master

3. Launch MAX and connect to the CompactRIO controller.

4. In MAX, expand the controller under Remote Systems in the Configuration pane. Right-click Software and select Add/Remove Software.

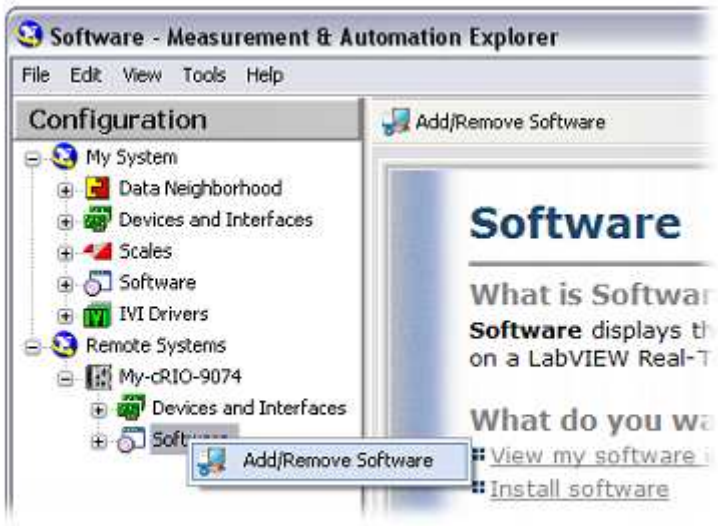


Figure 7.8. Install the appropriate software in MAX.

5. If the controller already has LabVIEW Real-Time and NI-RIO installed on it, select Custom software installation and click Next. Click Ye s if a warning dialog box appears. Click the box next to NI-Industrial Communications for EtherCAT. The required dependences are automatically checked. Click Next to continue installing software on the controller.

6. Once the software installation is finished, select the controller under Remote Systems in the Configuration pane. Navigate to the Network Settings tab in the bottom right portion of the window. In the Network Adapters window, select the secondary Ethernet Adapter (the one that does not say primary). Next to Adapter Mode, select EtherCAT in the drop-down box and hit the Save button at the top of the window.

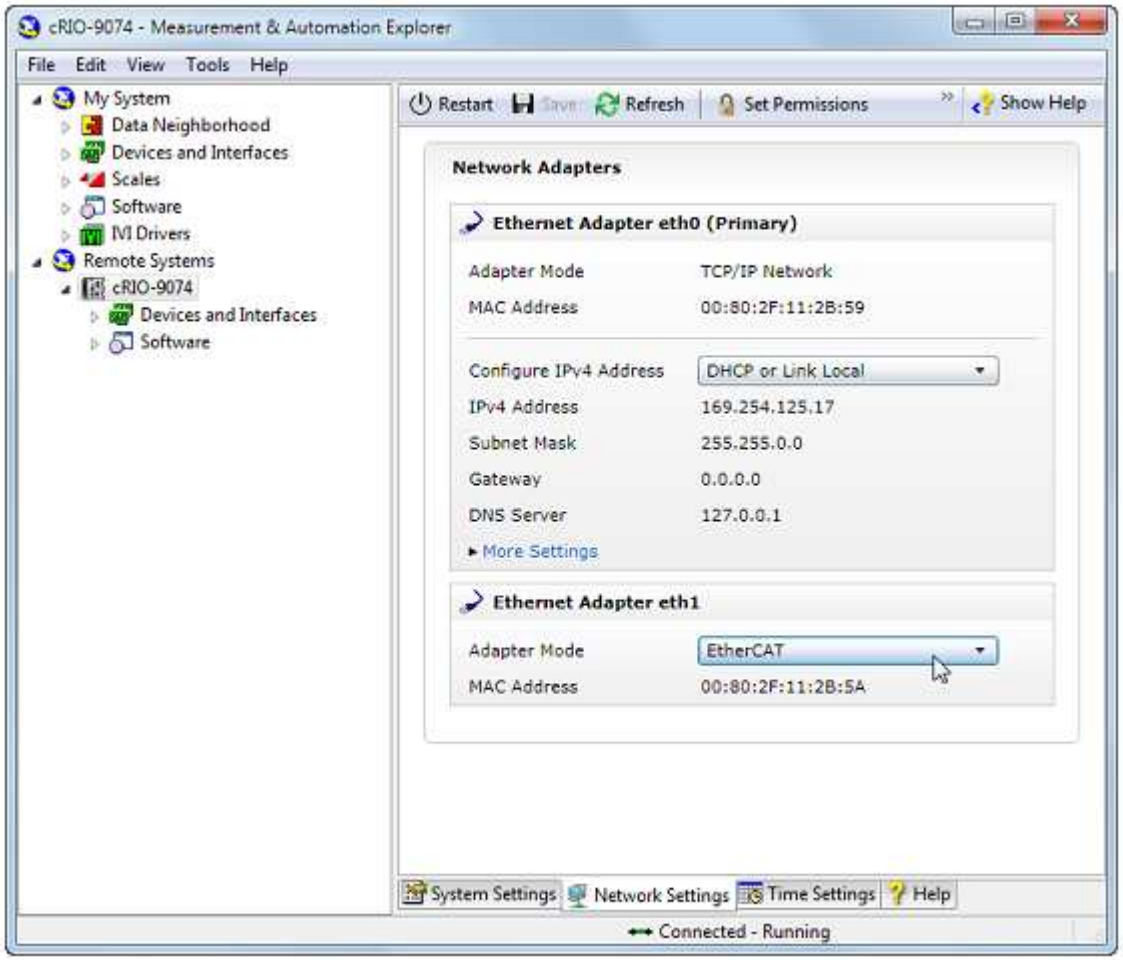


Figure 79. Select EtherCAT as the mode for the second Ethernet port of the CompactRIO controller.

Step 2: Add Deterministic I/O (Option 1—Scan Engine)

1. In the LabVIEW Project Explorer window, right-click on the CompactRIO controller and select New»Targets and Devices.

2. In the Add Targets and Devices dialog window, expand the category EtherCAT Master Device to autodiscover the EtherCAT port on the master controller. Select your EtherCAT device and click OK. The LabVIEW project now lists the master controller, the NI 9144 chassis, their I/O modules, and the physical I/O on each module.

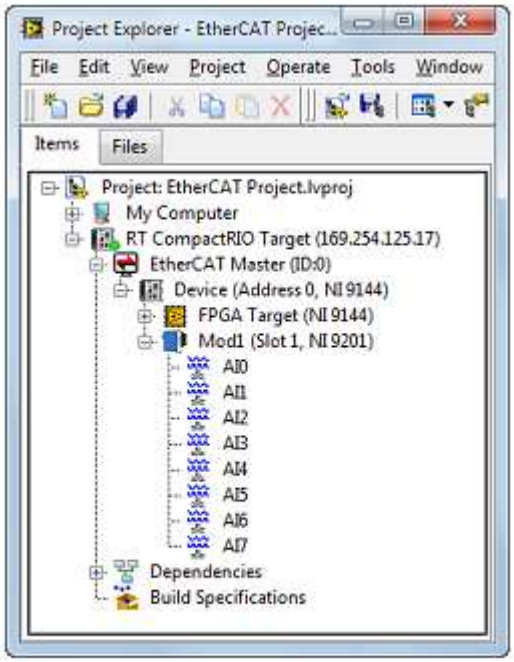


Figure 7.10. The LabVIEW project lists the master controller, NI 9144 chassis, and I/O modules.

3. By default, the I/O channels show up in the project as Scan Engine I/O variables. The Scan Engine automatically manages I/O synchronization so that all modules are read and updated at the same time once each scan cycle.

Note: For high-channel-count systems, you can streamline the creation of multiple I/O aliases by exporting and importing .csv spreadsheet files using the Multiple Variable Editor.

Step 2: Add Deterministic I/O (Option 2—LabVIEW FPGA)

1. In the LabVIEW Project Explorer window, right-click on the CompactRIO controller and select New»Targets and Devices.

2. In the Add Targets and Devices dialog window, expand the category EtherCAT Master Device to autodiscover the EtherCAT port on the master controller. Select your EtherCAT device and click OK. The LabVIEW project now lists the master controller, the NI 9144 chassis, their I/O modules, and the physical I/O on each module. By default the I/O channels show up in the project as Scan Engine I/O variables.

3. Right-click the EtherCAT device in the LabVIEW project and select New»FPGA Target. You can now create a LabVIEW FPGA VI to run on the EtherCAT target. By default the chassis I/O is added to the FPGA target but not the I/O modules. To program the C Series module I/O using LabVIEW FPGA, drag the C Series modules from the EtherCAT device target onto the FPGA target in the LabVIEW Project Explorer window.

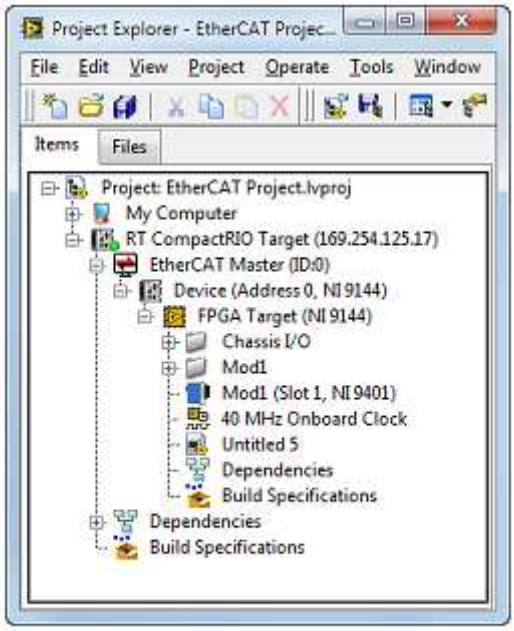


Figure 7.11. Drag C Series I/O modules under the FPGA target to access them with LabVIEW FPGA.

4. Create an FPGA VI under the EtherCAT device target and program using the EtherCAT I/O. The Figure 7.12 example uses the FPGA on an EtherCAT chassis to output a PWM signal.

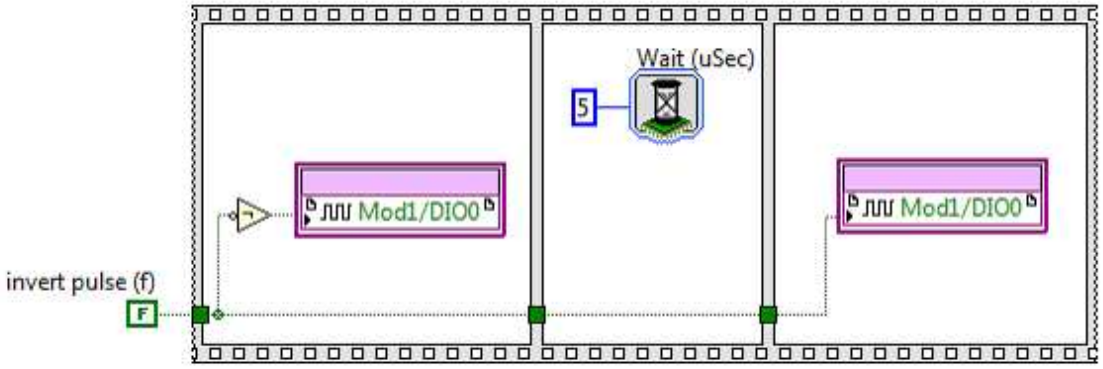


Figure 7.12. Program the EtherCAT FPGA using LabVIEW FPGA.

Note: There is a limit to the number of user-defined I/O variables that you can create in FPGA Mode. The NI 9144 can hold a total of 512 bytes of input data and 512 bytes of output data for both I/O variables in Scan Mode and user-defined I/O variables in FPGA Mode. For example, if you are using four 32-channel modules in Scan Mode and each channel takes up 32 bits of data, then Scan Mode I/O variables are using 256 bytes of input data. With the remaining 256 bytes of input data, you can create up to 64 input user-defined I/O variables (also of 32-bit length) in FPGA Mode.

5. The next step is creating an interface that allows the Real-Time VI to communicate to the FPGA VI on the EtherCAT expansion chassis. For example, you need the user to be able to control the pulse width of the PWM signal, along with the invert pulse input. To transfer data to or from an FPGA on an EtherCAT chassis to a Real-Time VI, you use a new mechanism

called user-defined I/O variables. To create a user-defined I/O variable, right-click the EtherCAT device target and select New»User-Defined Variable.

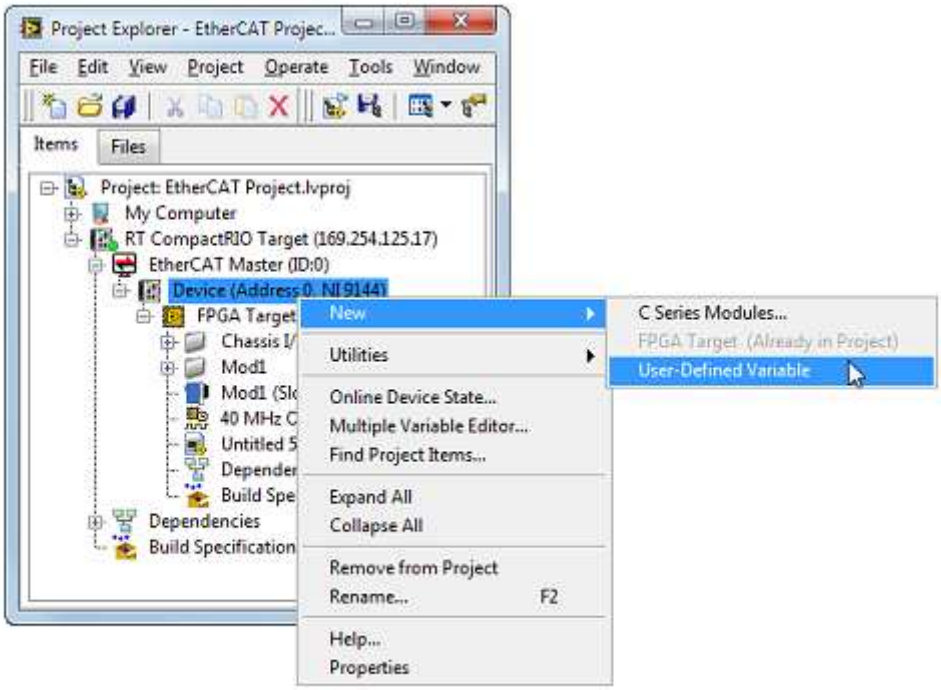


Figure 7.13. Create user-defined I/O variables to communicate between a Real-Time VI and the EtherCAT FPGA VI.

6. In the Properties dialog box, specify the variable name, data type, and direction (FPGA to host or host to FPGA). Drag and drop the user-defined I/O variables onto your FPGA VI.

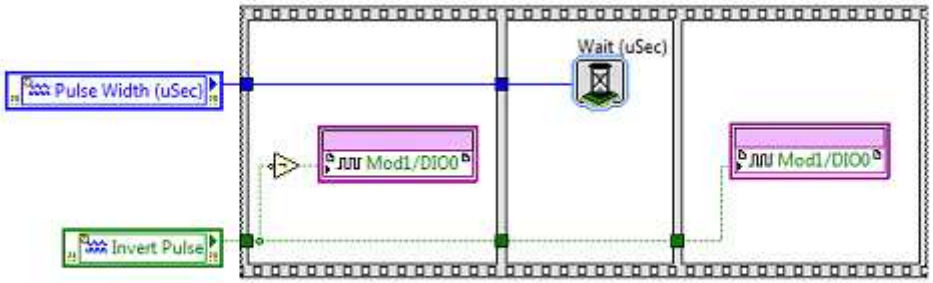


Figure 7.14. Drag and drop user-defined I/O variables onto the FPGA VI.

Note: These user-defined I/O variables transfer single-point data to and from the master controller at each scan cycle, so they are best used for passing processed data from the NI 9144 FPGA.

7. Drag and drop the same variables onto your Real-Time VI to communicate between the two targets.

10. KOMUNIKÁCIA S EXTERNÝMI ZARIADENIAMI TRETÍCH STRÁN

10.1. Overview

This section examines different methods for communicating with third-party devices from a CompactRIO controller. Because CompactRIO has built-in serial and Ethernet ports, it can easily work Modbus TCP, Modbus Serial, and EtherNet/IP protocols. You can find information on using the Ethernet port to communicate to an HMI in Chapter 4: Best Practices for Network Communication. When interfacing to a third-party device in LabVIEW Real-Time using RS232, USB, or Ethernet ports, you should consider adding a separate process for handling the communication since these protocols are nondeterministic.

This allows the control task to run deterministically and reliably because it is not affected by the possibly high-jitter I/O device.

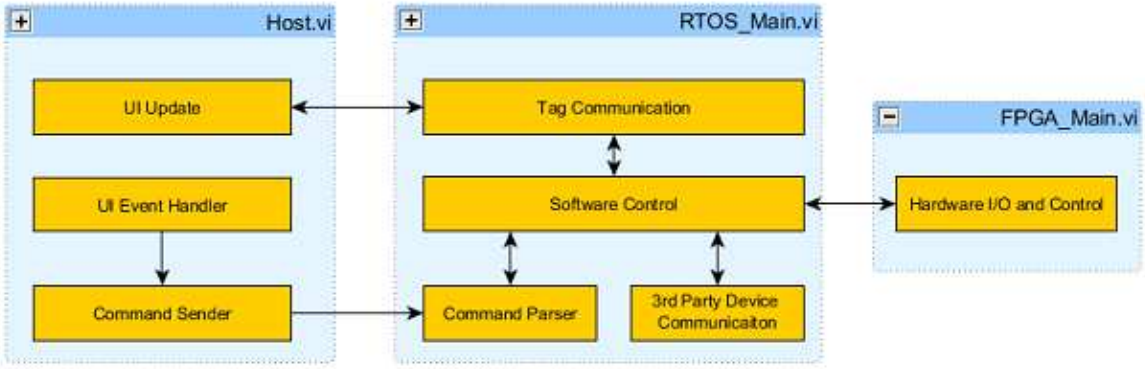


Figure 8.1. Add a separate process for handling any nondeterministic communication to a third-party device over serial or Ethernet.

Serial Communication From CompactRIOA standard for more than 40 years, RS232 (also known as EIA232 or TIA232) ports can be found on all CompactRIO systems and a wide variety of industrial controllers, PLCs, and devices. RS232 remains a de facto standard for low-cost communications because it requires little complexity, low processing and overhead, and modest bandwidth. While it is fading from the PC industry in favor of newer buses such as USB, IEEE 1394, and PCI Express, RS232 continues to be popular in the industrial sector because of its low complexity, low cost to implement, and wide install base of existing RS232 ports.

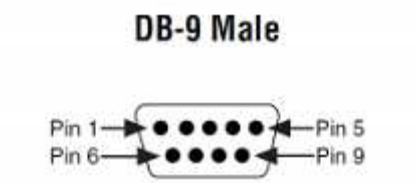
The RS232 specification covers the implementation of hardware, bit-level timing, and byte delivery, but it does not define fixed messaging specifications. While this allows for easy implementation of basic byte reading and writing, more complex functionality varies between devices. Several protocols use RS232-based byte messaging such as Modbus RTU/ASCII, DNP3, and IEC-60870-5.

This section covers how to program byte-level communications for the serial port built into CompactRIO real-time controllers using the NI-VISA API.

10.2. RS232 Technical Introduction

RS232 is a single-drop communications standard, meaning only two devices can be connected to each other. The standard defines many signal lines used in varying degrees by different applications. At a minimum, all serial devices use the Transmit (TXD), Receive (RXD), and ground lines. The other lines include flow control lines, Request to Send, and Clear to Send, which are sometimes used to

improve synchronization between devices and prevent data loss when a receiver cannot keep up with its sender. The remaining lines are typically used with modem-to-host applications and are not commonly implemented in industrial applications with the exception of radio modems.



Pin	232 DTE	232 DCE	422/485
1	DCD[2]	DCD	GND
2	RXD	TXD	CTS+ (HSI+)
3	TXD	RXD	RTS+ (HSO+)
4	DTR[2]	DSR	RXD+
5	GND	GND	RXD–
6	DSR[2]	DTR	CTS– (HSI–)
7	RTS	CTS	RTS– (HSO–)
8	CTS	RTS	TXD+
9	RI[2]	RI	TXD–

Figure 8.2. D-SUB 9-Pin Connector Pinout

As a single-ended bus, RS232 is ideal for shorter-distance communications (under 10 m). Shielded cables reduce noise over longer runs. For longer distances and higher speeds, RS485 is preferred because it uses differential signals to reduce noise. Unlike buses such as USB and IEEE 1394, RS232 is not designed to power end devices.

CompactRIO controllers have an RS232 port that features the standard D-SUB 9-pin male connector (also known as a DE-9 or DB-9). You can use this port to monitor diagnostic messages from a CompactRIO controller when the console out switch is engaged or to communicate with low-cost RS232 devices in your application.

10.3. RS232 Wiring and Cabling

RS232 ports feature two standard pinouts. You can classify RS232 devices as data terminal equipment (DTE), which is like a master or host, and data communications equipment (DCE), which is similar to slaves. DCE ports have inverted wiring so that DCE input pins connect to DTE output pins and vice versa. The serial port on CompactRIO controllers is a DTE port, and the serial port on a device such as a GPS, bar code scanner, modem, or printer is a DCE port.

When connecting a CompactRIO DTE RS232 port to a typical end device with a DCE RS232 port, you use a regular straight-through cable. When connecting two DTE or DCE devices together, such as a CompactRIO controller to a PC, you need to use a null modem cable, which swaps the transmit and receive signals in the cabling.

Device 1	Device 2	Cable Type
DTE	DTE	Null modem cable
DTE	DCE	Straight-through cable
DCE	DTE	Straight-through cable
DCE	DCE	Null modem cable

Table 8.1. Selecting a Cable to Run Between Different RS232 Ports

Loopback Testing

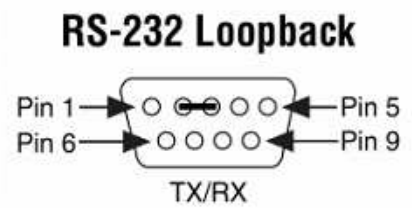


Figure 8.3. RS232 Loopback Wiring

A common technique for troubleshooting and verifying serial port functionality is loopback testing. At a basic level, all you need to do is connect the TXD to the RXD pin. Use a standard RS232 straight-through or loopback cable and a small bent paper clip to short pins 2 and 3. With the loopback cable installed, bytes sent from the read function can be read by the read function. You can verify that software, serial port settings, and drivers are working correctly with this technique.

10.4. Serial Communications From LabVIEW

The native CompactRIO RS232 serial port is attached directly to the CompactRIO real-time processor, so you should place code that accesses the serial port in the LabVIEW Real-Time VI. To send and receive bytes to the serial port, use NI-VISA functions. NI-VISA is a driver that provides a single interface for communicating with byte-level interfaces such as RS232, RS485, GPIB, and so on. Code written with the NI-VISA functions is usable on any machine with a serial port and NI-VISA installed. This means you can write and test a Serial VI on a Windows machine with LabVIEW and then reuse the same code in LabVIEW Real-Time on CompactRIO. You can then directly use thousands of prewritten and verified instrument drivers located at ni.com/idnet.

To get started with the serial port, locate the Virtual Instrument Software Architecture (VISA) functions in the palettes, under Data Communication»Protocols»Serial.

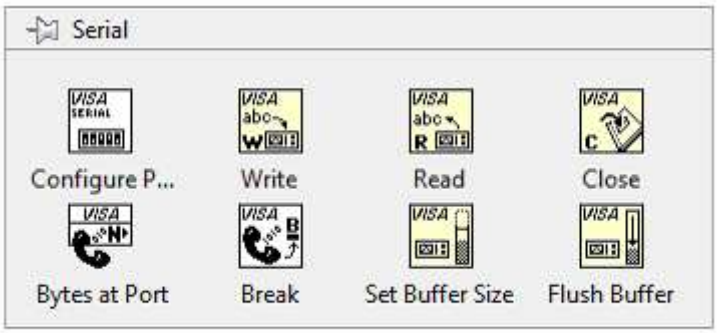


Figure 8.4. VISA Functions

For most simple instrument applications, you need only two VISA functions, VISA Write and VISA Read. Refer to the Basic Serial Write and Read VI in the labview\examples\instr\smplserl.llb for an example of how to use VISA functions. Most devices require you to send information in the form of a command or query before you can read information back from the device. Therefore, the VISA Write function is usually followed by a VISA Read function. Because serial communication requires you to configure extra parameters such as baud rate, you must start the serial port communication with the VISA Configure Serial Port VI. The VISA Configure Serial Port VI initializes the port identified by the VISA resource name.

It is important to note that the VISA resource name refers to resources on the machine for which the VI is targeted.

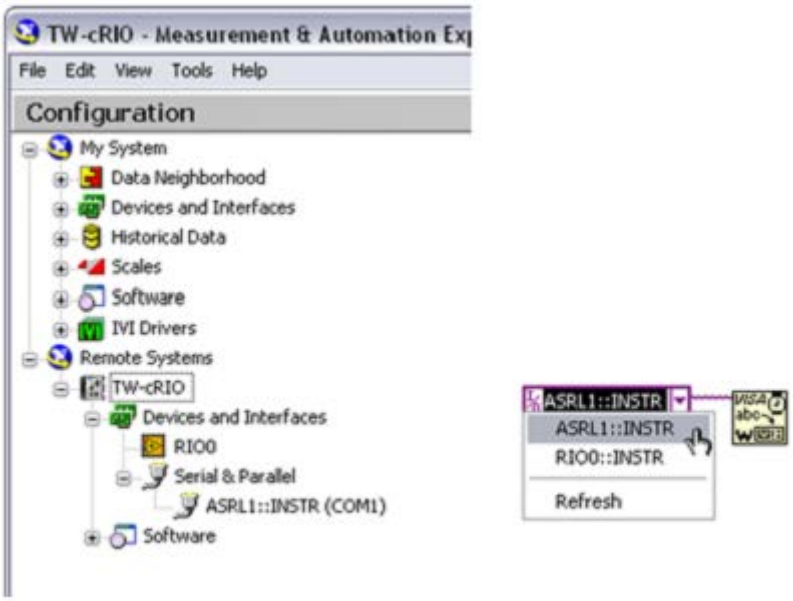


Figure 8.5. You can directly browse to the VISA resource name from the VISA resource control in LabVIEW or from MAX.

Check the bottom left corner of the VI to determine which target the VI is linked to. For CompactRIO, use COM1 to use the built-in port. COM 1 on a CompactRIO controller is ASRL1::INSTR. If you are connected to the CompactRIO controller, you can directly browse to the VISA resource name from the VISA resource control in LabVIEW or from MAX.

Timeout sets the timeout value for the serial communication. Baud rate, data bits, parity, and flow control specify those particular serial port parameters. The error in and error out clusters maintain the error conditions for this VI.

While the ports work at the byte level, the interfaces to the read and write functions are strings. In memory, a string is simply a collection of bytes with a length attached to it, which makes working with many bytes easier to program. The string functions in LabVIEW provide tools to break up, manipulate, and combine data received from a serial port. You can also convert string data to an array of bytes for using LabVIEW array functions to work with low-level serial data.

Because serial operations deal with varying-length strings, these tasks are nondeterministic. Additionally, serial interfaces by nature are asynchronous and do not have any mechanisms to guarantee timely delivery of data. To maintain the determinism of your control loops, when programming serial applications keep any communications code in a separate loop from the control code.

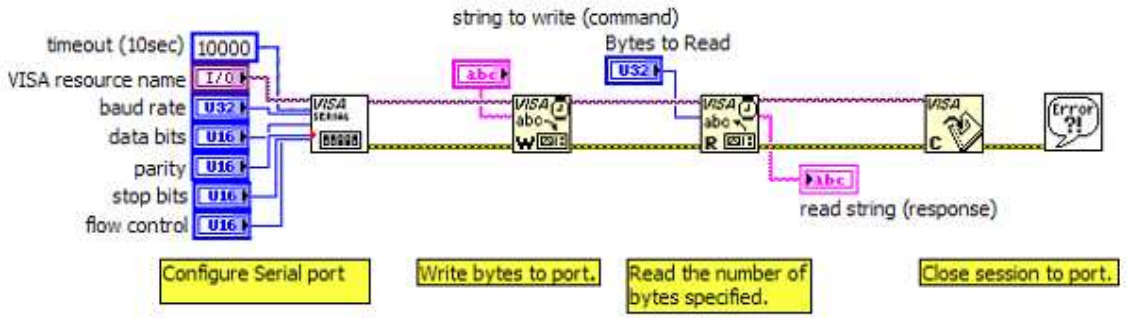


Figure 8.6. Simple Serial Communication Example

The VISA Read function waits until the number of bytes requested arrives at the serial port or until it times out. For applications where this is known, this behavior is acceptable, but some applications have different lengths of data arriving. In this case, you should read the number of bytes available to be read, and read only those bytes. You can accomplish this with the Bytes at Serial port property accessible from the Serial palette.

You can access serial port parameters, variables, and states using property nodes. If you open the VISA Configure Serial Port VI and examine the code, you see that this VI simply makes a call to many property nodes to configure the port. You can use these nodes to access advanced serial port features, including the auxiliary data lines, bytes waiting to be read/written, and memory buffer sizes. Figure 8.7 shows an example of using a property node to check the number of bytes available at the port before reading.

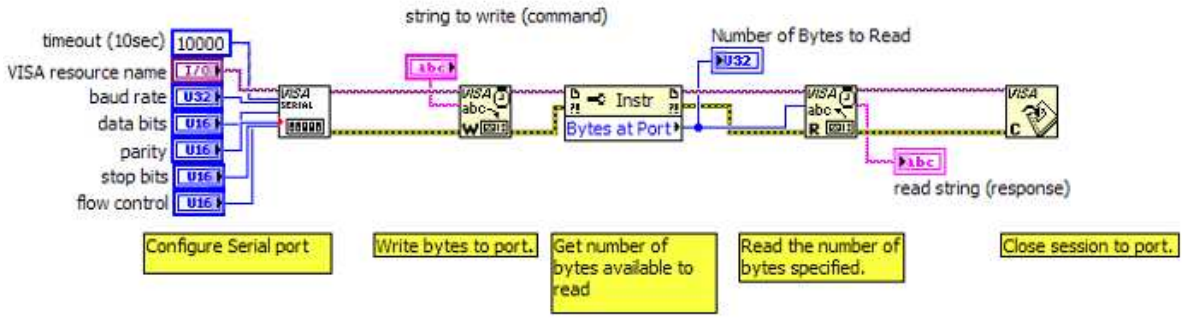


Figure 8.7 Using a VISA Property Node to Read the Number of Bytes

In general, working with devices can range from simply reading periodically broadcasted bytes such as GPS data to working with complex command structures. For some devices and instruments, you can find preprogrammed instrument drivers on ni.com/idnet, which can simplify development. Because these drivers use NI-VISA functions, they work with the onboard CompactRIO serial port.

10.5. Instrument Driver Network

To help speed development, NI has worked with major measurement and control vendors to provide a library of ready-to-run instrument drivers for more than 9,000 devices. An instrument driver is a set of software routines that control a programmable instrument. Each routine corresponds to a programmatic operation such as configuring, reading from, writing to, and triggering the instrument. Instrument drivers simplify instrument control and reduce program development time by eliminating the need to learn the programming protocol for each device.

Where to Find Instrument Drivers and How to Download ThemYou can find and download instrument drivers using two different ways. If you are using LabVIEW 8.0 or later, the easiest way is to use the NI Instrument Driver Finder. If you have an older version of LabVIEW, then you can use the Instrument Driver Network (ni.com/idnet).

Use the NI Instrument Driver Finder to find, download, and install LabVIEW Plug and Play drivers for an instrument.

Select Tools»Instrumentation»Find Instrument Drivers to launch the Instrument Driver Finder. This tool searches IDNet to find the specified instrument driver.

You can use an instrument driver for a particular instrument as is. However, LabVIEW Plug and Play instrument drivers are distributed with their block diagram source code, so you can customize them for a specific application. You can create instrument control applications and systems by programmatically linking instrument driver VIs on the block diagram.

10.6. RS232 and RS422/RS485 Four-Port NI C Series Modules for CompactRIO

If your application requires higher performance or additional RS232 ports and/or RS485/RS422 ports, you can use the NI 9870 and NI 9871 serial interfaces for CompactRIO

to add more serial ports. These modules are accessed directly from the FPGA using LabVIEW FPGA or from the Scan Engine.

10.7. Using NI 987x Serial Modules With Scan Mode

To enable support for these modules using the Scan Engine, you need to add NI-Serial 9870 and 9871 Scan Engine Support to your CompactRIO target in MAX. Once the NI-Serial 9870 and 9871 Scan Engine Support and your CompactRIO target are correctly configured, the ports of an NI 987x appear in MAX when the Serial & Parallel branch is expanded. ASRL1 is your built-in serial port, and ASRL 2 through 5 are the ports of an NI 987x.

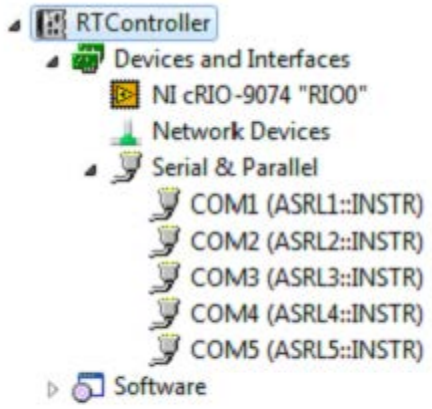


Figure 8.8. Access Your NI 987x serial ports in MAX by installing Scan Engine support.

In LabVIEW, place a VISA Configure Serial Port VI or a Visa Property Node to configure the settings of your individual ports.

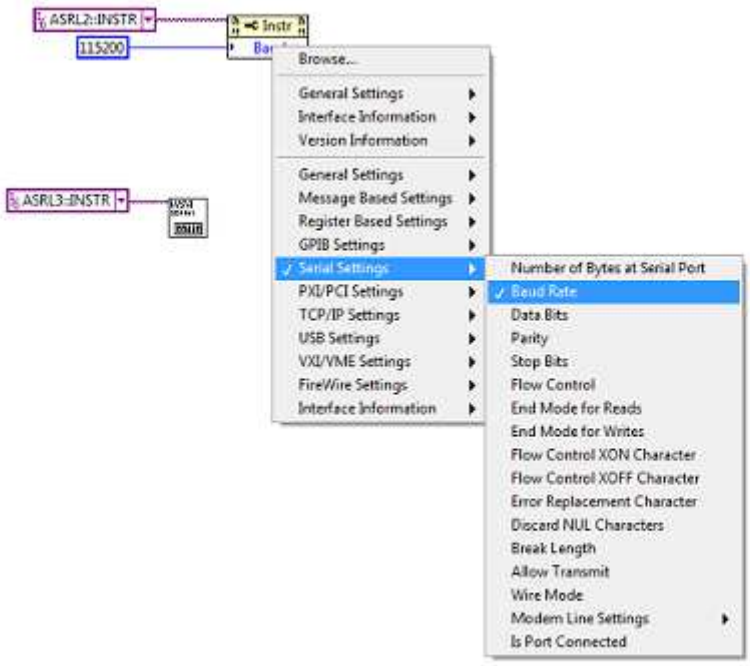


Figure 8.9. Program an NI 987x with the VISA API when using Scan Mode.

Once you have set the serial port to match that of the connected device, use the VISA API as you would when using any other VISA device.

Note: This maximum nonstandard baud rate for the NI 9871 module (3.6864 Mbit/s) can only be achieved while using the FPGA interface. The maximum baud rate while using either module in with the RIO Scan Interface is 115.2 kbit/s.

10.8. Using NI 987x Serial Modules With LabVIEW FPGA

You also can access NI 987x serial modules with LabVIEW FPGA. See the Figure 8.10 example titled NI 987x Serial Loopback.lvproj in the NI Example Finder. This example demonstrates communication to the NI 987x serial port and streaming serial data to a real-time VI.

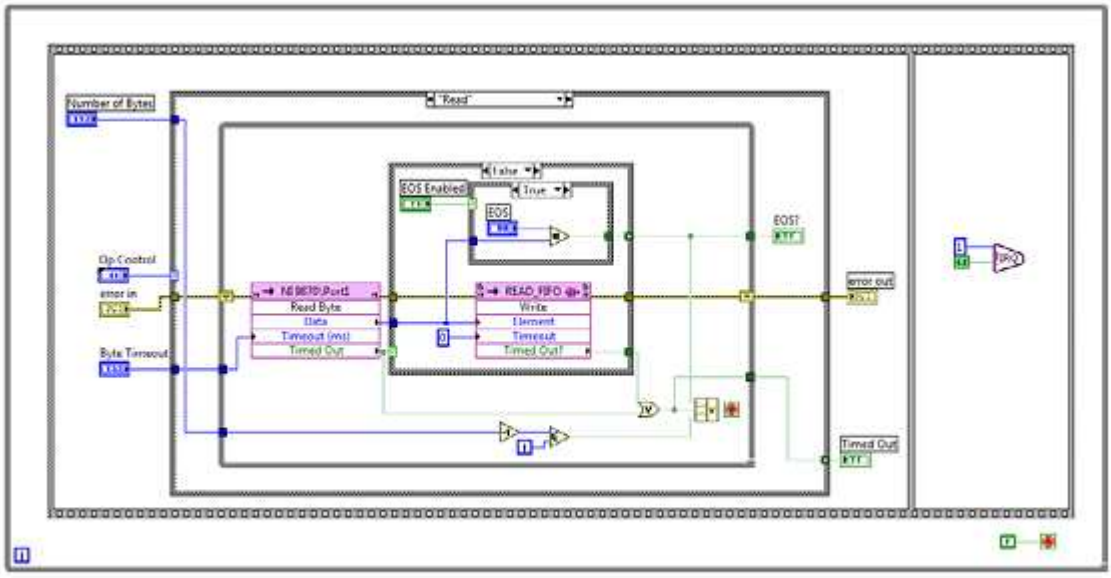


Figure 8.10. You can find the NI 987x serial loopback example for LabVIEW FPGA in the NI Example Finder.

Communicating With PLCs and Other Industrial Networked DevicesOften applications call for integrating NI programmable automation controllers (PACs) such as CompactRIO into an existing industrial system or with other industrial devices. These systems can consist of traditional PLCs, PACs, or a wide variety of specialty devices such as motor controllers, sensors, and HMIs. These devices feature a wide range of communication options from simple RS232 to specialized high-speed industrial networks.

You can choose from three common methods to connect the CompactRIO controller to an industrial system or PLC. You can directly wire the CompactRIO controller to the PLC using standard analog or digital I/O. This method is simple but scales for only small systems. For larger systems, you can use an industrial communications protocol to communicate between the CompactRIO controller and the PLC. For large SCADA applications, OPC may be a good tool. OPC scales to high channels but uses Windows technology and, therefore, requires a Windows PC to perform the communications.

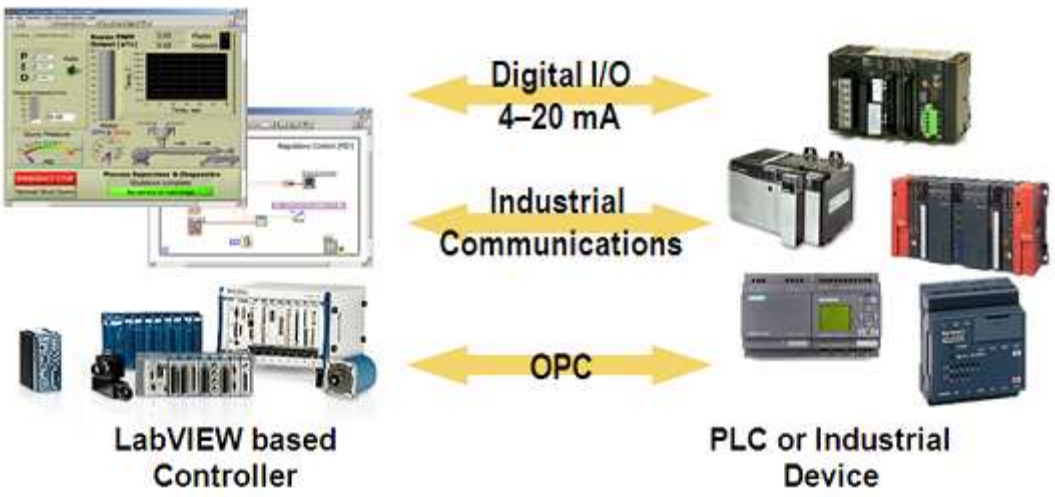


Figure 8.11. Three Methods for Connecting a CompactRIO Controller to an Industrial Device

10.9. Industrial Communications Protocols

Beyond using standard I/O, the most common way to connect a CompactRIO controller to other industrial devices is using industrial communications protocols. Most protocols use standard physical layers such as RS232, RS485, CAN, or Ethernet. As discussed earlier, buses like RS232 and Ethernet provide the physical layer for simple communications but lack any predefined methods of higher level communication, so they offer just enough for a user to send and receive individual bytes and messages. When working with large amounts of industrial data, handling those bytes and converting them into relevant control data quickly becomes tedious. Industrial communications protocols provide a framework for how data is communicated. They are fundamentally similar to the instrument drivers discussed earlier although they typically run a more sophisticated stack.

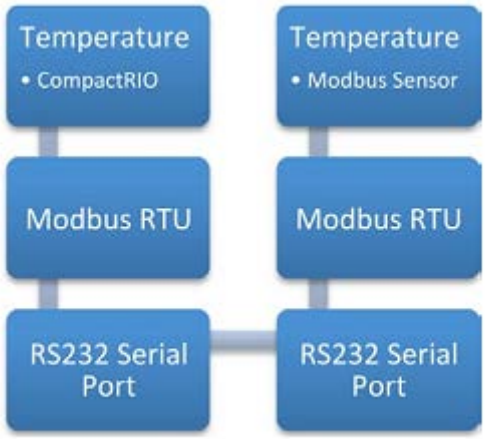


Figure 8.12. Typical Industrial Protocol Implementation

Industrial protocols take a low-level bus like RS232 or TCP/IP and add predefined techniques for communication on top. This is similar to how HTTP, the protocol for web content delivery, sits on top of TCP/IP. It is possible to write your own HTTP client like Internet Explorer or Firefox, but it takes a large effort and may not accurately connect to all servers without hours of testing and verification. Similarly, devices and controllers that support industrial networking standards are much easier to integrate at the program level and abstract the low-level communication details from the end user, so engineers can focus on the application.

Because CompactRIO has built-in serial and Ethernet ports, it can easily work with many protocols including Modbus TCP, Modbus Serial, and EtherNet/IP. You also can use plug-in modules for communication with most common industrial protocols such as PROFIBUS and CANopen.

When you cannot use native communication, another option is gateways. Gateways are protocol translators that can convert between different industrial protocols. For example, you can connect to a CC-Link network by using a gateway. The gateway can talk Modbus TCP on the PAC end and translate that to CC-Link on the other end.

10.10. Modbus Communications

Modbus is the most common industrial protocol in use today. Developed in 1979, it supports both serial and Ethernet physical layers. Modbus is an application layer messaging protocol for client/server communication between devices connected on a bus or network. You can implement it using asynchronous serial transmission to communicate with touch screens, PLCs, and gateways to support other types of industrial buses. Modbus works with a CompactRIO controller and its onboard serial or Ethernet modules. Note that the onboard serial port of CompactRIO hardware uses RS232 while some Modbus devices may use the RS485 electrical layer. In this case, you can use a common RS485-to-RS232 adapter.

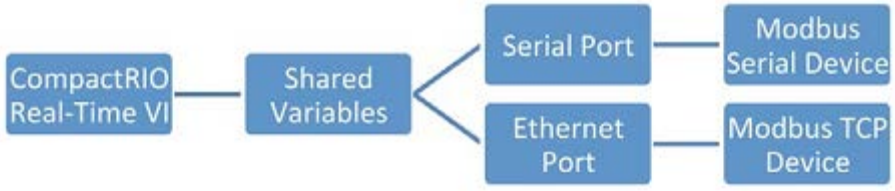


Figure 8.13. Software Architecture for Communicating With Modbus Devices

The Modbus serial protocol is based on a master/slave architecture. An address, ranging from 1 to 247, is assigned to each slave device. Only one master is connected to the bus at any given time. Slave devices do not transmit information unless a request is made by the master device, and slave devices cannot communicate to other slave devices.

Information is passed between master and slave devices by reading and writing to registers located on the slave device. The Modbus specification distinguishes the use of four register tables, each capable of 65,536 items and differentiated by register type and read-write access. Register address tables do not overlap in this implementation of LabVIEW.

Tables	Object Type	Type of Access	Comments
Discrete Inputs	Single-Bit	Read Only	Only the master device can read. Only the slave device can change its register values.
Coils	Single-Bit	Read-Write	Both master and slave devices can read and write to these registers.
Input Registers	16-Bit Word	Read Only	Only the master device can read. Only the slave device can change its register values.
Holding Registers	16-Bit Word	Read-Write	Both master and slave devices can read and write to these registers.

Table 8.2. Information is passed between master and slave devices by reading and writing to registers located on the slave device.

Before starting with Modbus programming, you must determine several important parameters of your Modbus device by consulting its documentation:

1. Master or slave? If your Modbus device is a slave, configure your CompactRIO controller as a master. This is the most common configuration. Likewise, connecting to a Modbus master (such as another PLC) requires configuring the CompactRIO controller as a slave.

2. Serial or Ethernet? Modbus Ethernet devices are sometimes called “Modbus TCP” devices.

3. For Modbus serial

- RS232 or RS485? Many Modbus devices are RS232, but some use the higher power RS485 bus for longer distances. RS232 devices can plug directly into a CompactRIO system, but an RS485 device needs an RS232-to-RS485 converter to function properly.
- RTU or ASCII mode? RTU/ASCII refers to the way data is represented on the serial bus. RTU data is in a raw binary form, whereas ASCII data is human readable on the raw bus. This parameter is required by the Modbus VIs.

4. What are the register addresses? Every Modbus device has a mapping of its I/O to registers, coils, and discrete inputs for reading and writing that is represented by a numerical address. Per Modbus convention, a register address is always one less than the register name. This is similar to the first element of an array being element 0. The Modbus LabVIEW Library requires register addresses, not register names.

10.11. Modbus Tutorial

The NI Developer Zone document How to Turn an RT Target Into a Modbus Slave Using I/O Servers walks you through the steps of turning your CompactRIO controller into a Modbus slave. It also shows you how to read from this server with LabVIEW for Windows if you do not have a third-party Modbus server. Before starting the tutorial, install the Modbus I/O Server onto your CompactRIO target through MAX.

10.12. EtherNet/IP

EtherNet/IP is a protocol built on the Common Industrial Protocol (CIP) for communications with EtherNet/IP devices. This protocol, developed and commonly found on Rockwell Automation (Allen-Bradley) PLCs, uses standard Ethernet cabling for communications with industrial I/O. EtherNet/IP is an application layer protocol that uses TCP for general messages and User Datagram Protocol (UDP) for I/O messaging and control.

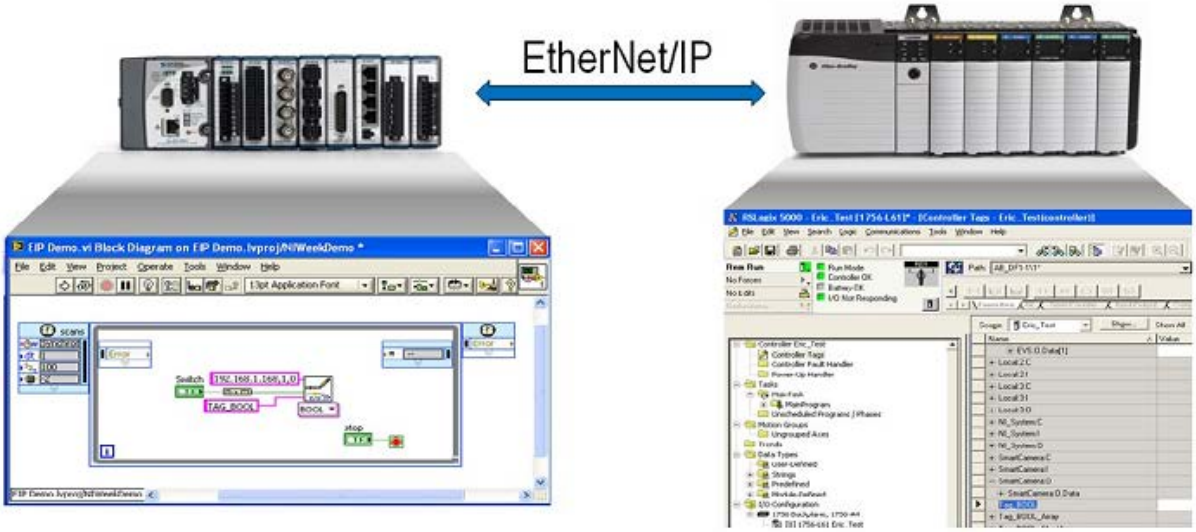


Figure 8.14. With EtherNet/IP, you can directly read and write to tags on a Rockwell PLC from CompactRIO.

The LabVIEW Driver for EtherNet/IP provides both explicit messaging API and implicit I/O data communication with a wide range of PLCs and other EtherNet/IP devices. With this driver, the CompactRIO controller supports direct communication to Rockwell Automation PLCs, allowing for easy integration into a new or existing system.

10.13. Energy Protocol: DNP3

DNP3 (Distributed Network Protocol) is a protocol specification for vendors of power grid SCADA (Supervisory Control and Data Acquisition) components. This protocol is commonly used in electrical and water utilities to communicate between a master station and other devices such as remote terminal units (RTUs) and other intelligent electronic devices (IEDs).

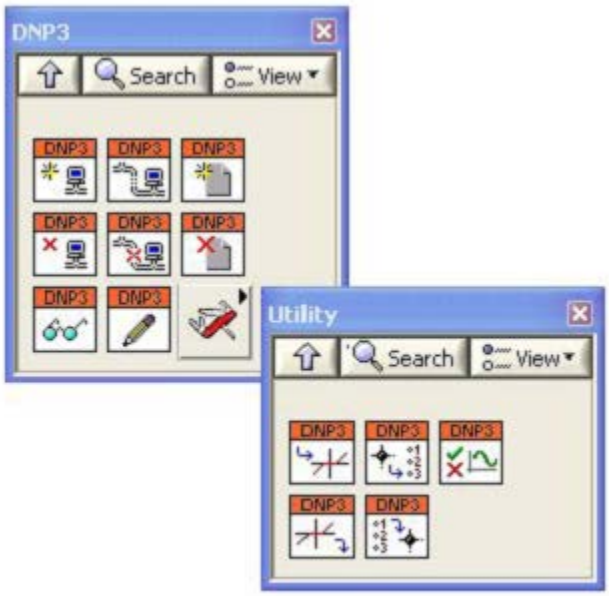


Figure 8.15. DNP3 functions provide capabilities to create outstations in LabVIEW.

You can use the LabVIEW Driver for DNP3 Outstation Support to program a real-time target, such as a CompactRIO system, as an outstation device with advanced functionality like power quality monitoring, phasor measurements, and other smart grid-related analysis. The DNP3 software driver supports Ethernet and serial communication, file transfer, and time synchronization between the master and outstation. Multiple communication channels per outstation and multiple sessions (logical devices) per channel also work with this driver.

10.14. OPC

OPC, or OLE for process control, is a common protocol used in many applications with high channel counts and low update rates, which are common in process industries such as oil and gas and pharmaceutical manufacturing. OPC is designed for connecting HMI and SCADA systems to controllers—it is not generally used for communication between controllers. A common use for OPC is to integrate a CompactRIO controller into a third-party system for HMI and SCADA applications.

The OPC specification is a large collection of standards that span many applications. This section focuses on OPC Data Access, the core specification that defines how to universally move data around using common Windows technologies.

The network-published shared variable in LabVIEW provides a gateway to OPC.

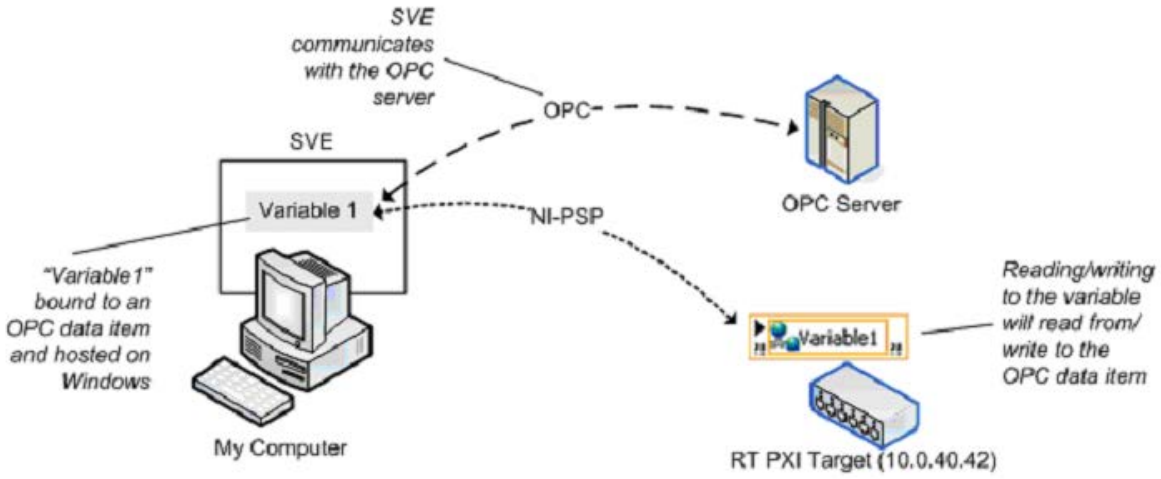


Figure 8.16. Communicate OPC data items with network-published shared variables.

OPC is a Windows-only technology and the CompactRIO controller cannot directly communicate using the OPC protocol. Instead, a Windows PC can communicate to the CompactRIO controller using the NI-PSP protocol and can translate and republish the information via OPC. When running on a Windows machine, the Shared Variable Engine functions as an OPC server and performs this translation and republishing automatically. This means other OPC clients, such as third-party SCADA packages or process control software, can access and retrieve data from the CompactRIO device.

Publishing Data From a CompactRIO System Over OPC

The following example describes how to publish network shared variables as OPC items using the Shared Variable Engine (SVE) as an OPC server. The example assumes you are already publishing data onto the network via network-published shared variables hosted by the CompactRIO system. If the network-published shared variables are hosted on a Windows PC, they are published as OPC items by default. You can verify the variable is working correctly by launching an OPC client such as the NI Distributed System Manager as described in step 6.

1. The CompactRIO controller publishes data onto the network using networked-published shared variables hosted on the CompactRIO controller. In this example, the network shared variable is named SV_PID_SetPoint.

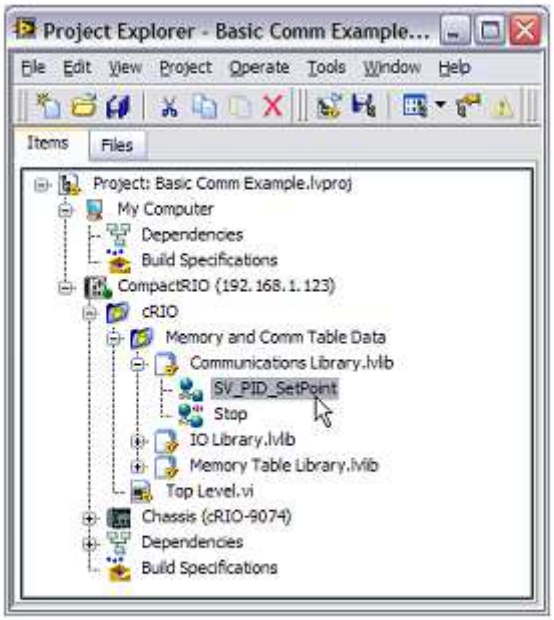


Figure 8.17. Network-Published Shared Variables Hosted on the CompactRIO System

2. To publish the variables as OPC data items, bind each variable hosted on the CompactRIO controller to a variable hosted on the Windows PC. To create variables hosted on a Windows PC, right-click My Computer in the LabVIEW Project and select New»Variable.

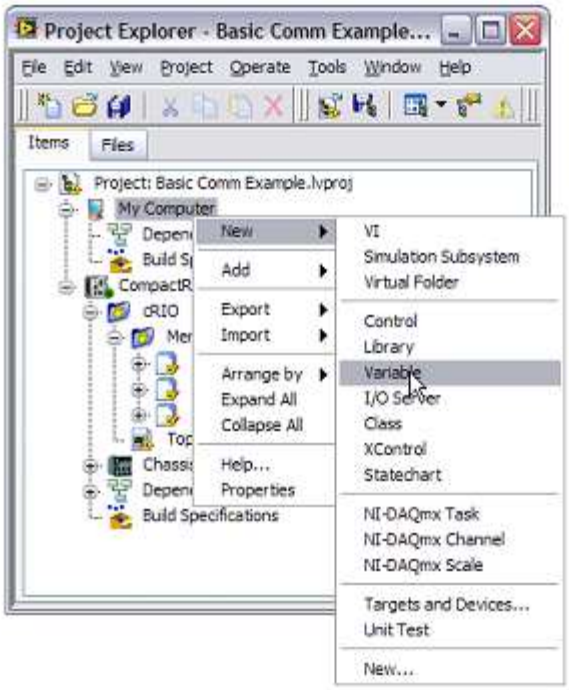


Figure 8.18. Network-published shared variables hosted on Windows are automatically published via OPC.

3. In the Shared Variable Properties window, give the variable a useful name such as SV_PID_SetPoint_OPC. Check the Enable Aliasing box and click the Browse button to browse to the SV_PID_SetPoint variable on the CompactRIO controller.

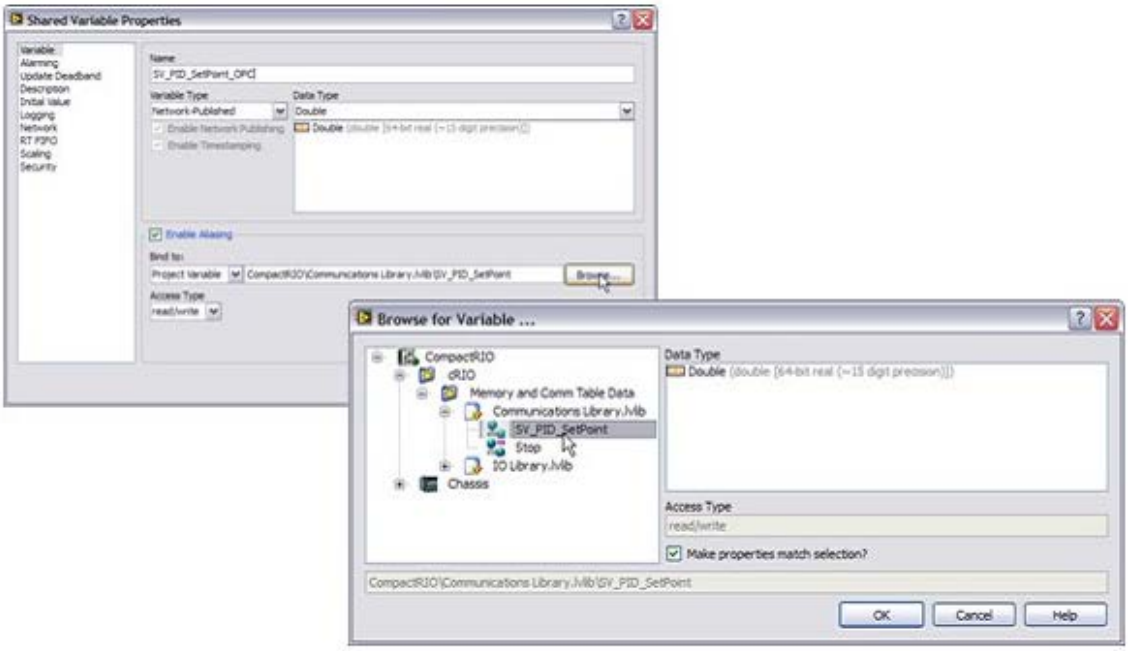


Figure 8.19. An aliased network-published shared variable provides a gateway between CompactRIO and OPC.

4. Save the new library you create with a useful name, such as OPC Library.lvlib.

5. Deploy the variables to the Shared Variable Engine by right-clicking the library and selecting Deploy.

6. The shared variable is now published as an OPC item. Other OPC clients can now connect to the Windows PC and use the data for display, HMI, and so on. You can verify the variable is working correctly by launching an OPC client such as the NI Distributed System Manager (Start»Programs»National Instruments»NI Distributed System Manager).

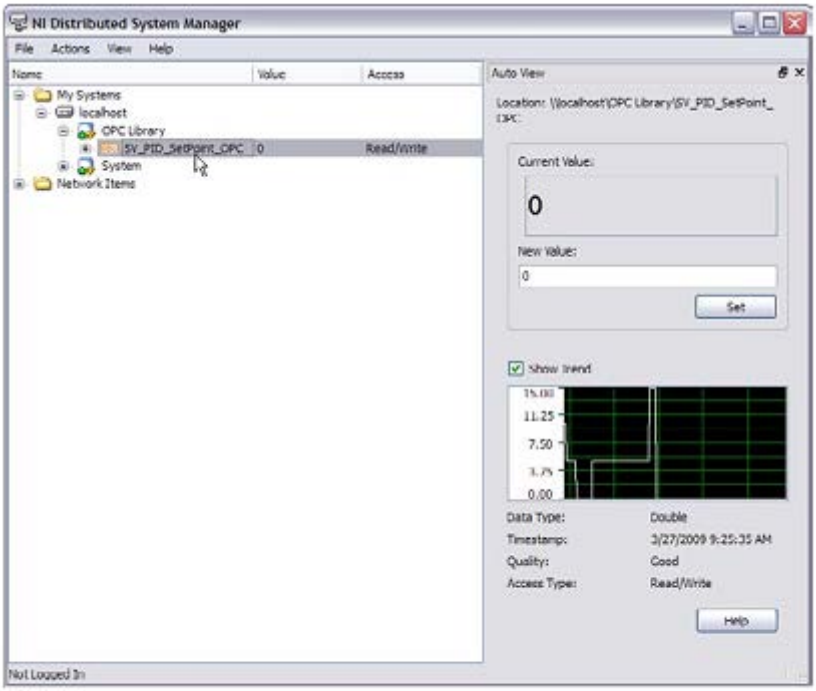


Figure 8.20. The OPC clients, such as the NI Distributed System Manager, can read and write to the OPC item.

7. If you have an OPC client, such as the LabVIEW DSC Module, you can verify the variable is working from that client as well. NI shared variable OPC items are published under the National Instruments.Variable Engine.1 server name.

8. Once variables are deployed in the Shared Variable Engine, they stay deployed with subsequent reboots of the OPC server machine.

9. Find more information on the Shared Variable Engine and OPC in the LabVIEW Help file Connecting to OPC Systems using LabVIEW (Windows Only)

Additional Resources

For information on other industrial communications protocols supported by NI, visit ni.com/comm.

PŘÍLOHY

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz

info@crr.vutbr.cz