

Laboratorium Komputerowych Systemów Pomiarowych

ĆWICZENIE NR 2

SYSTEM POMIAROWY Z INTERFEJSEM IEC-625 / IEEE-488

(opracował Eligiusz Pawłowski)

Cel i zakres ćwiczenia

Celem ćwiczenia jest praktyczne zapoznanie się studentów z problematyką programowania w środowisku LabVIEW systemu pomiarowego złożonego ze standardowych przyrządów pomiarowych wyposażonych w interfejs IEC-625 / IEEE-488.

Program laboratorium obejmuje następujące zagadnienia:

- zapoznanie się ze środowiskiem LabVIEW i podstawami programowania,
- umieszczanie kontrolki na panelu programu i tworzenie interfejsu użytkownika,
- tworzenie diagramu sterującego przebiegiem programu,
- zapoznanie się z systemem interfejsu IEC-625 / IEEE-488,
- programowanie przyrządów pomiarowych poprzez interfejs IEC-625 / IEEE-488,
- zdalne sterowanie poprzez interfejs IEC-625 / IEEE-488: zasilaczem prądu stałego, komutatorem sygnałów analogowych i multimetrem cyfrowym,
- zaprogramowanie w środowisku LabVIEW eksperymentu pomiarowego polegającego na automatycznym wyznaczaniu charakterystyk prądowo-napięciowych nieliniowych dwójników pasywnych z wykorzystaniem standardowych przyrządów pomiarowych wyposażonych w interfejs IEC-625 / IEEE-488.

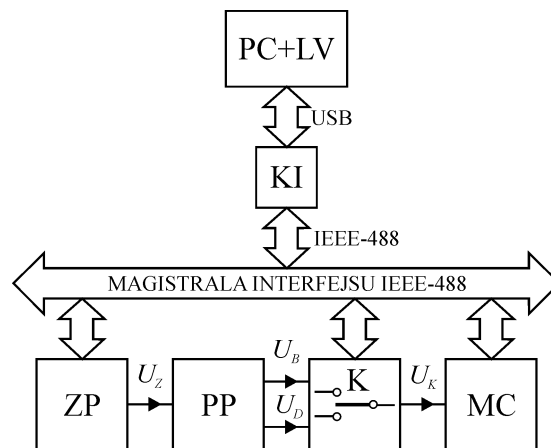
1. Opis stanowiska laboratoryjnego

Wykorzystywane w ćwiczeniu stanowisko pomiarowe umożliwia automatyczne wyznaczanie charakterystyk prądowo-napięciowych dowolnych dwójników pasywnych. Schemat blokowy stanowiska laboratoryjnego przedstawia rysunek 1. Składa się ono z następujących elementów składowych:

- programowalnego zasilacza napięcia stałego ZP (konstrukcja własna - praca dyplomowa),
- podstawki pomiarowej PP z rezystorem bocznikowym i badanym dwójnikiem,
- komutatora analogowych sygnałów pomiarowych K (MERATRONIK typ I201),
- multimetru cyfrowego MC (Hewlett Packard HP34401A),
- konwertera interfejsu KI (National Instruments GPIB-USB-HS),
- komputera PC z zainstalowanym oprogramowaniem LabVIEW.

Zasilacz programowalny ZP umożliwia zadawanie napięcia U_Z zasilającego układ pomiarowy w przedziale od -24V do +24V z wydajnością prądową do 1,5A. Na podstawie pomiarowej PP znajduje się rezystor bocznikowy do pomiaru prądu oraz badany dwójnik. Oba te elementy mogą być dowolnie wymieniane. Jako badane obiekty w ćwiczeniu będą wykorzystywane nieliniowe dwójniki pasywne takie jak: diody prostownicze, diody Zenera, diody LED, termistory, żarówki z włóknem wolframowym i inne. Komutator K umożliwia naprzemienne dołączanie napięcia U_B na rezystorze bocznikowym i napięcia U_D na badanym dwójniku do wejścia multimetru cyfrowego MC. Multimetr MC jest skonfigurowany do pomiaru napięcia stałego U_K wyjściu komutatora K. Wszystkie przyrządy pomiarowe są dołączone do magistrali interfejsu IEEE-488. Komputer PC steruje przyrządami pomiarowymi dołączonymi do magistrali IEEE-488 dzięki zastosowaniu konwertera interfejsów KI, która jest dołączony do portu USB. Oprogramowanie LabVIEW

zainstalowane na komputerze PC umożliwia przygotowanie odpowiedniego programu sterującego pracą całego stanowiska.



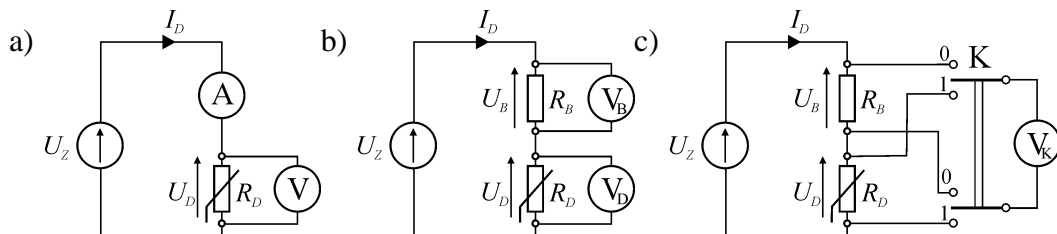
Rys. 1. Schemat blokowy stanowiska laboratoryjnego

Należy zauważyć, że możliwości pomiarowe przedstawionego stanowiska są bardzo szerokie dzięki zastosowaniu 25-kanalowego, czteroprzewodowego komutatora K oraz multimetru cyfrowego MC umożliwiające pomiar napięć i prądów stałych oraz przemiennych, rezystancji, częstotliwości i okresu. Po zastosowaniu dodatkowych źródeł zasilających możliwe jest więc również badanie innych elementów, takich jak tranzystory, tyrystory, transoptory, różnego rodzaju czujniki i przetworniki, wzmacniacze i wiele innych.

Wszystkie elementy składowe są ze sobą połączone i gotowe do realizacji ćwiczenia. W trakcie wykonywania ćwiczenia studenci modyfikują układ połączeń odpowiednio do realizowanego zadania. Zadania realizowane podczas ćwiczenia polegają na samodzielnym przygotowaniu odpowiednich programów w środowisku LabVIEW, uruchomieniu ich, przetestowaniu poprawności działania, usunięciu błędów i wykonaniu przykładowych pomiarów. W pierwszej kolejności tworzone będą programy obsługujące każdy przyrząd z osobna, które następnie łączone będą w jeden większy program obsługujący wszystkie przyrządy jednocześnie.

1.1. Układ pomiarowy do wyznaczenia charakterystyki prądowo-napięciowej dwójnika

Podstawowy układ pomiarowy do wyznaczenia charakterystyki prądowo-napięciowej dwójnika przedstawiono na rysunku 2a. Napięcie zasilające U_Z wymusza przepływ prądu I_D przez badany dwójnik, który jest reprezentowany przez rezystancję R_D . Prąd I_D jest mierzony amperomierzem A, a spadek napięcia U_D na dwójniku jest mierzony woltmierzem V. Wykonując serię pomiarów prądu I_D i napięcia U_D dla różnych wartości napięcia zasilającego U_Z można wyznaczyć charakterystykę prądowo-napięciową badanego dwójnika.



Rys. 2. Układ pomiarowy do wyznaczenia charakterystyki prądowo-napięciowej dwójnika:
a) - układ podstawowy, b) - układ używany w ćwiczeniu, c) zastosowanie komutatora

Jeśli otrzymana charakterystyka prądowo-napięciowa jest liniowa, to badany element uznajemy za dwójnik liniowy, co jest równoznaczne z wnioskiem, że jego rezystancja R_D ma stałą wartość, niezależną od przepływającego prądu I_D . W przeciwnym razie, gdy

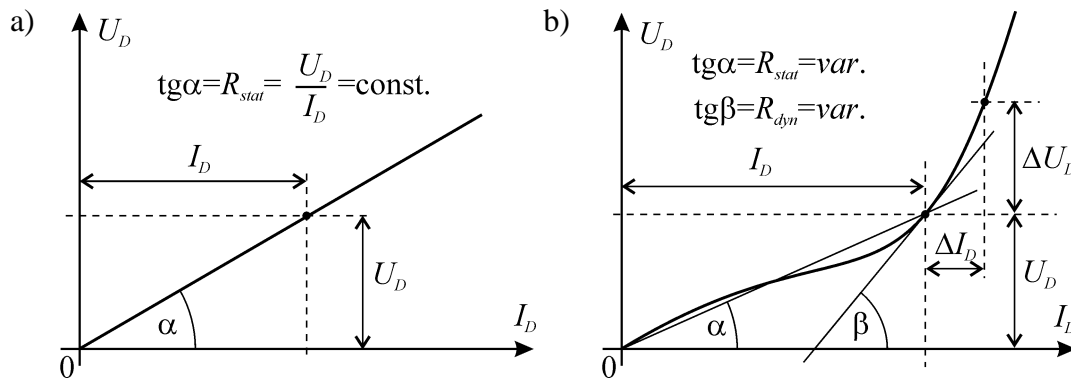
charakterystyka prądowo-napięciowa jest nieliniowa, badany element także uznajemy za dwójnik nieliniowy. Oznacza to również, że jego rezystancja R_D jest zależna od wartości przepływającego przez niego prądu I_D .

Przykładowe charakterystyki prądowo-napięciowe dla dwójnika linowego i nieliniowego przedstawiono na rysunku 3. Dla dwójnika liniowego (rys. 3a) tangens kąta α nachylenia charakterystyki wyznacza rezystancję statyczną R_{stat} dwójnika:

$$R_{stat} = \frac{U_D}{I_D} = \text{tg} \alpha \quad (1)$$

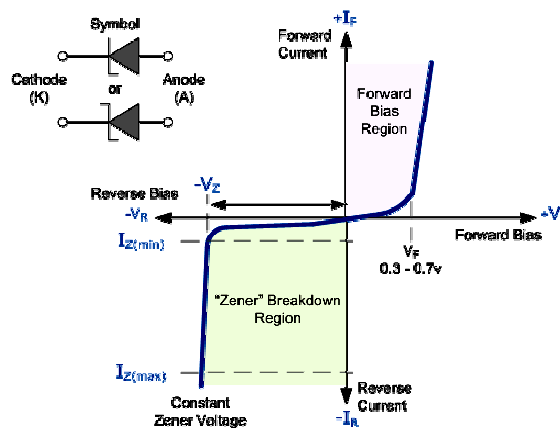
która ma wartość stałą. Dla dwójnika nieliniowego (rys. 3b) rezystancja statyczna R_{stat} zmienia się zależnie od wartości prądu I_D . Dla dwójników nieliniowych dodatkowo definiuje się również rezystancję dynamiczną R_{dyn} , jako tangens kąta β nachylenia stycznej do charakterystyki prądowo-napięciowej:

$$R_{dyn} = \frac{dU_D}{dI_D} = \text{tg} \beta \approx \frac{\Delta U_D}{\Delta I_D} \quad (2)$$



Rys. 3. Przykładowe charakterystyki prądowo-napięciowe dwójników: a) - dla dwójnika liniowego, b) - dla dwójnika nieliniowego

Charakterystyki prądowo-napięciowe przedstawione na rysunku 3 mają postać zależności funkcyjnej $U_D=f(I_D)$. Należy jednak zauważyć, że w praktyce stosuje się również przedstawianie charakterystyk prądowo-napięciowych w postaci zależności funkcyjnej $I_D=f(U_D)$, szczególnie dla elementów półprzewodnikowych. Przykładowo, na rysunku 4 przedstawiono typową charakterystykę prądowo-napięciową dla diody Zenera.

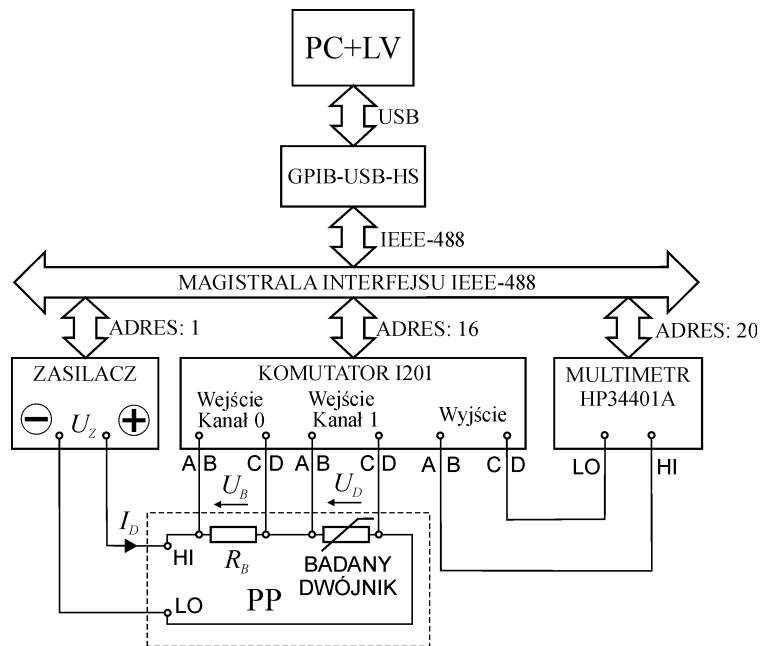


Rys. 4. Przykładowa charakterystyka prądowo-napięciowa diody Zenera

W układach praktycznych bardzo często pomiar prądu I_D amperomierzem w sposób przedstawiony na rysunku 2a zastępuje się pomiarem spadku napięcia U_B na znanej rezystancji R_B , tak jak przedstawiono to na rysunku 2b. Wartość prądu I_D wyznacza się wtedy na podstawie prawa Ohma:

$$I_D = \frac{U_B}{R_B} . \quad (3)$$

Pomiar taki wymaga zastosowania dwóch woltomierzy: woltomierza V_B do pomiaru spadku napięcia U_B na rezystorze R_B i woltomierza V_D do pomiaru napięcia U_D na badanym dwójniku. W komputerowych systemach pomiarowych bardzo często w takim przypadku stosuje się tylko jeden woltomierz V_K , w sposób przedstawiony na rysunku 2c. Za pomocą specjalnego przełącznika K (nazywanego komutatorem i sterowanego z komputera poprzez interfejs) przełącza się woltomierz V_K w pozycję 0-0 w celu pomiaru napięcia U_B na rezystorze R_B , a następnie w pozycję 1-1 w celu pomiaru napięcia U_D na badanym dwójniku. Po wykonaniu pomiaru oblicza się wartość prądu według zależności (3). Taki właśnie sposób pomiaru zrealizowano w ćwiczeniu. Schemat połączeń stanowiska pomiarowego przedstawiono na rysunku 5.



Rys. 5. Schemat połączeń stanowiska pomiarowego

Na podstawie pomiarowej PP umieszczono połączone szeregowo: rezystor bocznikowy R_B oraz badany dwójnik. Układ pomiarowy zasilany jest napięciem U_Z z zasilacza programowalnego. Napięcie U_B z rezystora bocznikowego dołączone jest do kanału 0 komutatora I201, a napięcie U_D z dwójnika dołączone jest do kanału 1. Zastosowany w układzie komutator I201 posiada w każdym kanale cztery przełączane jednocześnie linie sygnałowe oznaczone kolejno literami A, B, C i D. Ponieważ w realizowanych pomiarach wystarczające jest przełączanie tylko dwóch linii sygnałowych, to w każdym kanale linie A i B oraz C i D połączone parami ze sobą. Wyjście komutatora I201 dołączone jest do wejścia multimetru HP34401A. Wszystkie przyrządy dołączone są do magistrali IEEE-4888 i posiadają unikalne adresy: zasilacz adres 1, komutator adres 16 i multimetr adres 20. Magistrala IEEE-4888 dołączona jest poprzez konwerter interfejsów GPIB-USB-HS do portu USB komputera PC z zainstalowanym środowiskiem LabVIEW.

1.2. System interfejsu IEC-625 / IEEE-488

System interfejsu do programowalnej aparatury pomiarowej został opracowany pod koniec lat 60-tych XX wieku w przez firmę Hewlett-Packard i był jednym z pierwszych ujednoliconych standardów komunikacyjnych między urządzeniami wejścia/wyjścia. Jako standard firmy Hewlett-Packard otrzymał on nazwę HP-IB (Hewlett-Packard Interface Bus), a po uzyskaniu patentu stał się popularny na rynku amerykańskim pod nawą GPIB (General Purpose Interface Bus). W późniejszym czasie został on zatwierdzony w USA jako standard IEEE-488 (w roku 1974 z późniejszymi zmianami i rozszerzeniami w latach 1978 i 1987). W tym samym czasie w Europie przyjęto go jako standard IEC-625. Standard ten został wprowadzony do stosowania również w Polsce w postaci Polskiej Normy PN-83/T-06563 – System Interfejsu dla programowalnej aparatury pomiarowej. Obecnie wszystkie te nazwy: IEC-625, IEEE-488 oraz GPIB są stosowane na całym świecie zamiennie i oznaczają ten sam system interfejsu, za jednym tylko wyjątkiem: standard IEC-625 (i wzorowana na nim polska norma PN-83/T-06563) stosuje system złącz Canon 25-stykowych, a standard IEEE-488 (i zgodny z nim GPIB) stosuje system złącz Amphenol 24-stykowych. Jest jednak możliwe łączenie ze sobą przyrządów wykonanych w tych dwóch różnych standardach za pomocą odpowiedniego kabla, ponieważ poza różnymi rodzajami złącz, wszystkie pozostałe parametry tych standardów są identyczne.

W związku z powyższym, w niniejszej instrukcji wszystkie wymienione nazwy systemów interfejsu: IEC-625, IEEE-488 oraz GPIB będą stosowane zamiennie, podobnie jak we współcześnie spotykanej literaturze technicznej.

Początkowo standard ten opisywał jedynie elektryczne parametry systemu interfejsu (8 linii danych, 8 linii kontrolnych, złącze 24 stykowe, poziomy napięcie zgodne z logiką TTL 5V) oraz protokół transmisyjny, a w szczególności sposób przesyłu danych za pomocą 3 linii sprzętowej synchronizacji (*ang. handshake*). Nie został jednak zdefiniowany format przesyłanych danych, składnia komend sterujących przyrządami pomiarowymi, sposób zakończenia linii polecenia (np. CR i LF albo EOI). Była to duża wada, gdyż bardzo utrudniała wymianę sprzętu pomiarowego na inny tego samego rodzaju, ale pochodzący od innego producenta, co wiązało się z koniecznością poważnej modyfikacji oprogramowania.

W 1985 roku firma Tektronix zaproponowała standardowy zestaw komend dla magistrali GPIB, który został zaakceptowany w 1987 roku jako standard IEEE-488.2 a dotychczasowa norma została przemianowana jako IEEE-488.1 i określała sposób transmisji w warstwie fizycznej. Dzięki takiemu rozwiązaniu pewne komendy stały się uniwersalne dla wielu urządzeń (np. identyfikacja *IDN?, zerowania *CLS), ale wiele komend dalej było swobodnie interpretowanych przez różnych producentów, co w dalszym ciągu uniemożliwiało podmianę tego samego typu sprzętu na taki sam, ale innego producenta.

Kolejnym krokiem było zdefiniowanie standardu komend do przyrządów pomiarowych określanego jako język SCPI (Standard Commands for Programmable Instrumentation). Zdefiniowana została w nim forma wysyłania do przyrządów komend programujących oraz forma uzyskiwanych od nich odpowiedzi z wynikami pomiarów. Większość komend są to skróty słów angielskich wraz z dodatkowymi argumentami. Jednak nie rozwiązało to całkowicie problemu, ponieważ różni producenci dalej używają zróżnicowanych komend dla tych samych typów urządzeń (np. podmiana multimetru Agilent na multimetr Kethley wymaga modyfikacji komend programujących, mimo, że jest to podobny rodzaj aparatury pomiarowej).

Każde z urządzeń GPIB ma swój adres fizyczny w zakresie od 0 do 30. Adresy 0 i 21 są najczęściej używane przez kartę interfejsu PC-GPIB, adres 31 jest specjalnym adresem tzw. Unlisten/Untalk i dotyczy wszystkich urządzeń na magistrali. Ze względu na to, że wiele urządzeń używa podwójnych adresów maksymalna liczba urządzeń wynosi 14. W praktyce nie stanowi to ograniczenia, gdyż można zastosować drugi kontroler GPIB lub też użyć

ekspandera. Urządzenia mogą być połączone w gwiazdę (czyli wszystkie urządzenia łączone są od kontrolera), w kaskadę (każde następne urządzenie podłączane jest do poprzedniego) lub też można stosować połączenie mieszane. Fabryczne kable GPIB mają złącza przelotowe umożliwiające różne typy połączeń. Ograniczeniem jest maksymalna długość kabla pomiędzy dwoma urządzeniami 4 metry a między pierwszym urządzeniem a kontrolerem 2 metry. Maksymalna długość kabla nie może przekraczać 20 metrów. Ograniczenia te można ominąć stosując tzw. ekspandery. Interfejs GPIB do komputera PC może być wykonany w postaci karty rozszerzeń (PCI lub ISA), modułu USB-GPIB, modułu Serial-GPIB, modułu Paralell-GPIB, a w przypadku stosowania komputerów przenośnych także jako moduł rozszerzeń PCMCIA-GPIB lub ExpressCard-GPIB. Dostępne są także moduły LAN-GPIB umożliwiające sterowanie urządzeń poprzez sieć Internet.

Zestaw urządzeń z interfejsem GPIB jest bardzo szeroki - mogą to być multimetry, oscyloskopy cyfrowe, generatory a także multipleksery. Warto wspomnieć tu o urządzeniach, które mogą tylko przyjmować polecenia (tzw. Listener) np. multiplekser oraz przyjmować i wysyłać polecenia (Listener/Talker) np. multimetr.

W praktyce nie trzeba jednak zagłębiać się w fizyczny sposób transmisji danych w interfejsie, ponieważ w specjalizowanych środowiskach programistycznych (np. LabVIEW) są gotowe procedury obsługujące interfejs GPIB. W najprostszym przypadku obsługa polega na wysłaniu pod określony adres na magistrali odpowiedniego zestawu komend programujących dane urządzenie i ewentualnie odczytanie danych wysłanych przez urządzenie z tego adresu. W przypadku starszych urządzeń trzeba jeszcze podać rozkaz wyzwalający (Trigger), aby urządzenie zastosowało się do wysłanych komend.

Istotnym ułatwieniem w posługiwaniu się komendami sterującymi w systemie IEEE488 oraz SCPI jest sposób kodowania komunikatów za pomocą kodów ASCII. Wszystkie komendy są więc przesyłane jako łańcuchy tekstowe i są od razu czytelne dla człowieka. Konsekwencją tego jest jednak konieczność konwersji zmiennych tekstowych na liczbowe i odwrotnie. W środowisku LabVIEW nie stanowi to jednak istotnego problemu.

Urządzenia używane w ćwiczeniu pracują w standardzie IEEE 488.1. Niektóre z nich są również zgodne ze specyfikacją standardu SCPI. Wszystkie komendy wysyłane są jako łańcuchy tekstowe w kodzie ASCII. Liczby przesyłane są więc poprzez magistralę systemu interfejsu IEEE-488 jako kody ASCII kolejnych cyfr składających się na daną liczbę. Program sterujący musi więc konwertować zmienne liczbowe na zmienne tekstowe podczas programowania np. numeru kanału komutatora czy też programowania napięcia wyjściowego zasilacza. Dla przykładu, chcąc załączyć kanał komutatora o numerze określonym przez liczbę 24, należy wysłać na magistralę interfejsu tekst zawierający cyfry 2 i 4, czyli kolejno kody ASCII cyfr 2 i 4, czyli liczby 32 i 34 (dziesiętnie). Podczas odczytu wyników pomiarów, np. napięcia z multimetru, program musi konwertować tekst zawierający znaki kolejnych cyfr na zmienną liczbową. Szczególną uwagę należy zwrócić przy tym na problem odpowiedniego znaku oddzielającego część ułamkową w ułamku dziesiętnym. W tym celu stosowany jest zarówno przecinek (polska wersja systemu Windows) jak i kropka (angielska wersja systemu Windows i większość przyrządów pomiarowych). Analogiczny problem zachodzi przy przenoszeniu wyników pomiarów do arkusza kalkulacyjnego (np. Excel) w celu wykonania obliczeń. Przy niewłaściwej konwersji danych liczbowych na zmienne tekstowe i odwrotnie) może więc zostać pominięta część ułamkowa. W środowisku LabVIEW problem ten można łatwo rozwiązać odpowiednio konfigurując funkcje konwertujące łańcuchy tekstowe na zmienne liczbowe i odwrotnie.

W dalszej części instrukcji zestawiono formaty komend sterujących przyrządy pomiarowe zastosowane w ćwiczeniu oraz formaty wyników pomiarów odczytywanych z tych przyrządów. Zamieszczone w instrukcji informacje są tylko częścią dokumentacji tych przyrządów, niezbędną do realizacji ćwiczenia. Pełna dokumentacja każdego z przyrządów

jest bardzo obszerna, często obejmuje nawet kilkaset stron tekstu. W laboratorium będą dostępne pełniejsze informacje a tym zakresie.

1.3. Komutator MERATRONIK typ I201

Komutator (inne spotykane nazwy to multiplekser i skaner) firmy MERATONIK typ I201 jest programowanym zdalnie przełącznikiem realizującym połączenie pomiędzy jednym z 25 wejść, a wspólnym wyjściem. Pozwala on w dowolnej kolejności dołączać sygnały z jego wejść do przyrządu pomiarowego dołączonego do wyjścia komutatora. W ten sposób jeden przyrząd, np. multimetr cyfrowy, można wykorzystać w automatycznym systemie pomiarowym do mierzenia sygnałów kolejno z 25 różnych źródeł.

Każdy z 25 kanałów komutatora składa się z czterech komutowanych linii połączeniowych, oznaczonych literami A, B, C i D. Daje to możliwość realizacji pomiarów czteropunktowych (ważne w pomiarach rezystancji). Przełączenie zespołu czterech linii stanowiących dany kanał realizuje polecenie włączenia danego kanału, które steruje odpowiednim przekaźnikiem kontaktronowym z czterema stykami. Wyjście komutatora posiada dodatkowy przekaźnik, również z czterema stykami, pozwalający odłączyć cały zespół przełączników wejściowych od przyrządu pomiarowego dołączonego do wyjścia komutatora. Stan przekaźnika wyjściowego jest ustawiany oddzielnym poleceniem programującym komutator.

Sygnały wejściowe doprowadza się do komutatora za pomocą specjalnych, ekranowanych, czteroprzewodowych kabli dołączanych do gniazd wielostykowych, umieszczonych na tylnej ścianie urządzenia. Wyjście komutatora umieszczone jest na przedniej ścianie obudowy w postaci czterech zacisków laboratoryjnych opisanych literami A, B, C, i D. W poszczególnych liniach komutatora zastosowano następujące kolory przewodów:

Linia A : kolor Biały	Linia C : kolor Zielony
Linia B : kolor Niebieski	Linia D : kolor Czerwony
Ekran : kolor Czarny	

Wszystkie kanały komutatora pogrupowane są w 5 grup po 5 kanałów (razem 25 kanałów). Każde 5 kanałów jednej grupy dołączone są do jednego z 5 modułów przełączających, których gniazda znajdują się na tylnej ścianie komutatora. Kanały każdej grupy ponumerowane są liczbami rzymskimi: I, II, III, IV, V. Tak więc np.: kanał I modułu pierwszego jest kanałem o numerze 0, kanał I modułu drugiego jest kanałem o numerze 5.

Programowanie komutatora realizuje się za pomocą dwóch prostych poleceń tekstowych: ustawienia aktywnego kanału (polecenia z nagłówkami X i Y) oraz ustawienia przełącznika wyjściowego (polecenie z nagłówkiem H).

Programowanie włączenia wybranego kanału realizuje się przez podanie jego numeru z zakresu od 0 do 24. Służy do tego polecenie o następującym formacie :

X n Y m H w

gdzie poszczególne znaki oznaczają kolejno:

- X** - nagłówek oznaczający wagę dziesięć zapisu numeru kanału,
- n** - cyfra wagi dziesięć zapisu numeru kanału (możliwe znaki 0, 1 lub 2),
- Y** - nagłówek oznaczający wagę jeden zapisu numeru kanału,
- m** - cyfra wagi jeden zapisu numeru kanału (możliwe znaki 0, 1, 2 ... 9),
- H** - nagłówek oznaczający ustawianie przełącznika wyjściowego,
- w** - dołączenie wyjścia (jeżeli wpisujemy 0) lub odłączenie wyjścia (jeżeli wpisujemy 8).

Znaki **X**, **Y**, **H** muszą być zapisane wielkimi literami, znaki **n**, **m**, **w** są kodami ASCII odpowiednich cyfr. Przykładowo wysłanie komunikatu **X2Y4H0** dołączy sygnał z wejścia 24 kanału do wyjścia komutatora, a komunikat **X2Y4H8** odłączy go otwierając styki

przełącznika wyjściowego. Komutator I201 pracuje w standardzie IEEE 488.1, dlatego aby wykonał on polecenie musi zostać dodatkowo wyzwolony komendą TRIGGER.

Numer aktualnie załączonego kanału pokazuje wyświetlacz na płycie czołowej komutatora. Dodatkowo, kropka dziesiąta na tym wyświetlaczu sygnalizuje załączenie i odłączenia przełącznika wyjściowego.

1.4. Zasilacz programowalny

Wykorzystywany w ćwiczeniu zasilacz programowalny jest stosunkowo prostą konstrukcją powstałą w wyniku realizacji pracy dyplomowej. Spełnia on wymagania standardu IEEE-488 oraz rozpoznaje komendy zgodne z językiem SCPI. Umożliwia on zadawanie napięcia stałego z przedziału od -24V do +24V z krokiem 0,5V. Posiada on również układ ograniczenia prądowego ustawianego na wartości: 0,1A ; 0,5A ; 1A oraz 1,5A.

Zasilacz może pracować w trzech trybach programowania:

- ręczne wybieranie wartości napięcia i wartości ograniczenia prądowego;
- zdalne przesyłanie rozkazów programujących wartość napięcia i ograniczenia prądowego poprzez łącze szeregowe w standardzie RS232;
- zdalne przesyłanie rozkazów programujących wartość napięcia i ograniczenia prądowego przez interfejs pracujący w standardzie IEEE488.

Po włączeniu zasilacza zgłasza się on komunikatem **STER. RECZNE** w dolnej linii wyświetlacza z pulsującą literą **R** sygnalizującą możliwość zmiany typu sterowania.

Jeśli zasilacz ma być sterowany ręcznie należy zatwierdzić ustawiony typ sterowania naciskając klawisz **ENTER ↵**, nastąpi wówczas przejście do wybierania nastawy programującej wartość napięcia. W górnym wierszu po napisie **U=** pojawia się pierwsza dostępna nastawa napięcia 0.5 z pulsującą cyfrą 0. Wyboru wartości napięcia dokonujemy za pomocą klawisza strzałek **↑** i **↓**, do momentu pojawienia się żądanej wartości, którą zatwierdza się naciskając klawisz **ENTER ↵**. Na drugim polu wyboru ograniczenia prądowego pojawi się migająca pierwsza cyfra nastawy prądu, a wyboru dokonujemy przewijając listę dostępnych nastaw używając strzałek **↑** i **↓**. Zatwierdzenie nastawy napięcia, a potem prądu, za pomocą klawisza **ENTER ↵**, powoduje ręczne zaprogramowanie zasilacza i ustawienie na jego wyjściu wartości napięcia i ograniczenie prądowego zgodnego z nastawami. Wyświetlacz powraca do stanu pozwalającego ponownie zmienić nastawy napięcia i prądu (pulsuje pierwsza cyfra w polu napięcia i dostępne są klawisze strzałek do wyboru nowych nastaw).

Wartości nastaw napięcia wybierane są z listy zdefiniowanej w oprogramowaniu i pojawiają się w sekwencji: 0.5; -0,5; 1.0; -1.0; ... ;23.5; -23,5; 24.0; -24.0 sterowanej klawiszami strzałek **↑** i **↓**. Wartości nastaw prądu wybierane są z listy zdefiniowanej w oprogramowaniu i pojawiają się w sekwencji: 0.1 ; 0.5 ; 1.0 ; 1.5. sterowanej klawiszami strzałek **↑** i **↓**. Zatwierdzenie wybranej nastawy napięcia i prądu następuje po naciśnięciu klawisza **ENTER ↵** po każdej z nastaw.

Wybór trybu programowania zasilacza ręcznego lub zdalnego może być wykonany w dowolnym momencie stanu pracy zasilacza po naciśnięciu klawisza menu **M**. Wyświetlacz powraca wtedy do stanu jak po włączeniu z dostępnym ręcznym trybem programowania. Zmiana trybu na programowanie zdalne następuje po wybraniu za pomocą klawiszy strzałek **↑** i **↓** odpowiedniego rodzaju interfejsu zdalnego programowania: **RS232** lub **IEEE**.

Wybór interfejsu **IEEE** wiąże się z przypisaniem zasilaczowi indywidualnego adresu w systemie, stąd też mamy dostępne nastawy **IEEE 1; IEEE2 ; ... ; IEEE9**. Zatwierdzenie wybranego typu interfejsu następuje po naciśnięciu klawisza **ENTER ↵**. Gdy wybrano tryb **RS232** to w dolnym wierszu wyświetlacza pojawia się napis **STER. RS232 R**. Zasilacz oczekuje na wysłany z terminala rozkaz programujący obie nastawy, zakończony naciśnięciem klawisza **ENTER** na klawiaturze terminala. Odebrany rozkaz jest wyświetlany

w dolnym wierszu w postaci pola rozkazu, które definiuje bezpośrednio wartości nastaw napięcia i prądu (pomijany jest nagłówek rozkazu w postaci SOUR:). Dolny wiersz wyświetlacza może wyglądać tak:

VOLT 24.0, 1.0 R

Litera R występuje zawsze na końcu dolnego wiersza wskazując, że rozkaz odebrany jest przez interfejs szeregowy. Po odebraniu komendy zgodnie z prawidłowymi nastawami napięcia i prądu uzupełniane są górne pola wyświetlacza, a na jego wyjściu ustawiana jest zaprogramowana wartość napięcia.

Wybór interfejsu IEEE do programowania zasilacza różni się od trybu RS232 tym, że po zaakceptowaniu wyboru klawiszem ENTER ↵ w dolnym wierszu pojawia się odpowiedni tekst, na przykład: STER. IEEE 1 I

Cyfra "1" po napisie IEEE oznacza adres zasilacza w systemie, zaś ostatnia litera "I" oznacza typ wybranego interfejsu. Dolna linia po odebraniu rozkazu programującego jest taka sama jak w opisanym sposobie odbioru rozkazu przez interfejs RS232 i może wyglądać tak:

VOLT 24.0, 1.0 I

Pozostałe reakcje zasilacza na rozkaz programujący są takie same. W obu trybach zdalnego programowania t.j. RS232 i IEEE należy wysłać standardowy rozkaz w języku SCPI. Przykładowa składnia komendy w języku SCPI programującej zasilacz do napięcia 24V i ograniczenia prądowego 0,5A, jest następująca:

SOUR:VOLT 24.0, 0.5

Oba pola cyfrowe stanowią, zgodnie z dostępnymi nastawami zasilacza opisanymi w trybie programowania ręcznego, odpowiednio wartość ustawianego napięcia na wyjściu zasilacza oraz wartość prądu ograniczenia.

Wysłanie tego rozkazu dla trybu programowania przez łącze RS232 wymaga umieszczenia na końcu rozkazu znaku CR (powrót karetki) o kodzie OD-hex (w LabVIEW należy wybrać z biblioteki PROGRAMING>STRING stałą o nazwie Carriage Return Constant i umieścić ją za pomocą funkcji konkatenacji na końcu rozkazu, zaś w przypadku wysyłania rozkazu z terminala znak ten zostanie automatycznie dostawiony po naciśnięciu ENTER na jego klawiaturze).

Tryb programowania przez interfejs IEEE nie wymaga dodawania znaku CR na końcu rozkazu. Należy zwrócić szczególną uwagę na to aby nastawy napięcia i prądu wysyłane za pomocą rozkazu programującego pochodziły ze zbioru dostępnych wartości zdefiniowanych przez oprogramowanie zasilacza (opisane dla trybu ręcznego), a także aby użyć właściwej składni rozkazu (dwukropek, spacja, przecinek) i oddzielać część całkowitą od części dziesiętnej nastawy za pomocą kropki tak jak przyjęto to w standardzie języka SCPI.

Wysłanie do zasilacza rozkazu o błędnej składni w polu SOUR : VOLT skutkuje pojawieniem się w górnym wierszu komunikatu

U = ER_RZK I = ER_RZK

Wysłanie do zasilacza rozkazu, w którym umieszczono wartość jednej z nastaw niezgodną ze zbiorem dostępnych, predefiniowanych nastaw, spowoduje umieszczenie w górnym wierszu komunikatu:

U = ERRWRT I = ERRWRT

Komunikat postaci:

U = ER_NST I = ER_NST

pojawi się, gdy w odebranych rozkazach brak jest przecinka w części pola nastaw lub brakuje znaku CR na końcu rozkazu przesyłanego łączem szeregowym RS232.

W wymienionych przypadkach błędów na wyjściu zasilacza ustawiona jest wartość napięcia i prądu równa 0, zaś kolejny poprawny rozkaz spowoduje ustawienie nowej przesłanej wartości napięcia i prądu.

Błędy są sygnalizowane wyłącznie na wyświetlaczu zasilacza i nie jest możliwe odebranie potwierdzenia o prawidłowym lub błędnym zaprogramowaniu zasilacza.

1.5. Multimetr HP34401A

Multimetr cyfrowy Hewlett Packard (obecnie Agilent) HP34401A jest nowoczesnym przyrządem pomiarowym spełniającym wymagania standardu interfejsu IEEE-488 oraz języka SCPI. Umożliwia on pomiar napięć i prądów stałych oraz przemiennych, rezystancji, częstotliwości i okresu. Pełna dokumentacja miernika obejmuje kilkaset stron tekstu i nie jest możliwe omówienie jej w instrukcji do ćwiczenia nawet w ograniczonej formie. Dlatego zostaną przedstawione tylko najważniejsze informacje, umożliwiające najprostszą programową obsługę multimetru do pomiaru napięcia stałego.

Jedną z najprostszych i często wykorzystywaną komendą języka SCPI jest komenda identyfikacji przyrządu pomiarowego dołączonego do interfejsu. Jej składnia jest następująca:

***IDN?**

Jeśli powyższa komenda zostanie wysłana pod adres, który posiada przyrząd pomiarowy zgodny ze standardem IEEE-488.2, to odeśle on swoje podstawowe informacje identyfikacyjne. Umożliwia to sprawdzenie, czy przyrząd jest prawidłowo dołączony do interfejsu i czy w ogóle ma włączone zasilanie. W przypadku opisywanego multimetru odpowiedź będzie wyglądała następująco:

HEWLETT-PACKARD,34401A,0,XX-XX-XX

gdzie pole XX-XX-XX będzie zawierało numer wersji oprogramowania.

Przykładowa składnia komendy w języku SCPI programującej multimetr do pomiaru napięcia stałego na zakresie 100V z domyślną rozdzielczością jest następująca:

MEAS:VOLT:DC? 100,DEF

gdzie:

MEAS - oznacza komendę wykonania pomiaru MEASure,

VOLT - oznacza konfigurację do pomiaru napięcia VOLTage,

DC - oznacza pomiar przy prądzie stałym Direct Current,

? - znak zapytania oznacza wykonanie pomiaru,

100 - oznacza wybranie zakresu 100V,

DEF - oznacza wybranie domyślnej rozdzielczości DEFault.

: - dwukropek oddziela kolejne komendy,

, - przecinek oddziela kolejne parametry komendy.

Wysłanie powyższej komendy do multimetru spowoduje zaprogramowanie go do pomiaru zgodnie z podanymi parametrami, wykonanie pomiaru i umieszczenie wyniku w buforze wyjściowym, skąd może zostać odczytany.

Format danych wydawanych na magistralę IEEE-488 z multimetru HP34401A przy odczycie pojedynczego wyniku jest następujący:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
S	D	.	D	D	D	D	D	D	D	D	D	E	D	D	D	NL

gdzie:

S - oznacza znak plus lub minus,

D - oznacza cyfrę 0-9,

E - oznacza znak Exponenty

NL - oznacza znak kodu ASCII (New Line) kończący transmisję.

Jak widać na powyższym przykładzie, wynik pomiaru z multimetru wysyłany jest w formacie zmiennoprzecinkowym (floating point) i zawiera zawsze 16 znaków ASCII. Jest to bardzo istotna informacja, gdyż umożliwia poprawną konwersję liczby zapisanej tekstowo (string) na zmienną liczbową. Należy również zwrócić uwagę, że znakiem rozdzielającym część ułamkową jest kropka, co jest istotne podczas pracy z polską wersją językową systemu Windows, w którym znakiem rozdzielającym jest przecinek. Wymaga to odpowiedniego skonfigurowania funkcji *Fract_Exp String To Number* dokonującej konwersji tekstu na liczbę: konieczne jest podanie wartości FALSE na wejście *use system decimal point*.

1.6. Konwerter interfejsu GPIB-USB-HS

Komputer sterujący przyrządami pomiarowymi z interfejsem w standardzie IEEE-488 również musi być wyposażony odpowiedni układ interfejsu w tym standardzie. Zazwyczaj nie stanowi on typowego wyposażenia komputera i jest konieczne jego dołączenie. W praktyce można spotkać się z układami interfejsu IEEE-488 współpracującymi z komputerem PC poprzez magistralę ISA, PCI, PCIe, złącze USB, RS232 oraz sieć Ethernet. W ćwiczeniu zastosowano konwerter interfejsów GPIB-USB-HS firmy National Instruments. Po jednorazowym zainstalowaniu odpowiedniego oprogramowania i dołączeniu konwertera do portu USB, jest on gotowy do pracy i nie wymaga żadnej dodatkowej obsługi. Wygląd konwertera GPIB-USB-HS przedstawiono na rysunku 6.



Rys. 6. Konwerter interfejsów GPIB-USB-HS National Instruments


2. Programowanie przyrządów pomiarowych w środowisku LabVIEW








2.1. Podstawy programowania w środowisku LabVIEW

Studenci samodzielnie powinni zaopatrzyć się w literaturę wspomagającą naukę programowania w LabView. Przykładowe pozycje, w tym dostępne w Internecie darmowe podręczniki, zamieszczono w wykazie literatury.

LabView jest graficznym środowiskiem programistycznym przeznaczonym do tworzenia programów zorientowanych na obsługę systemów pomiarowych. Aby napisać program w środowisku LabView należy:

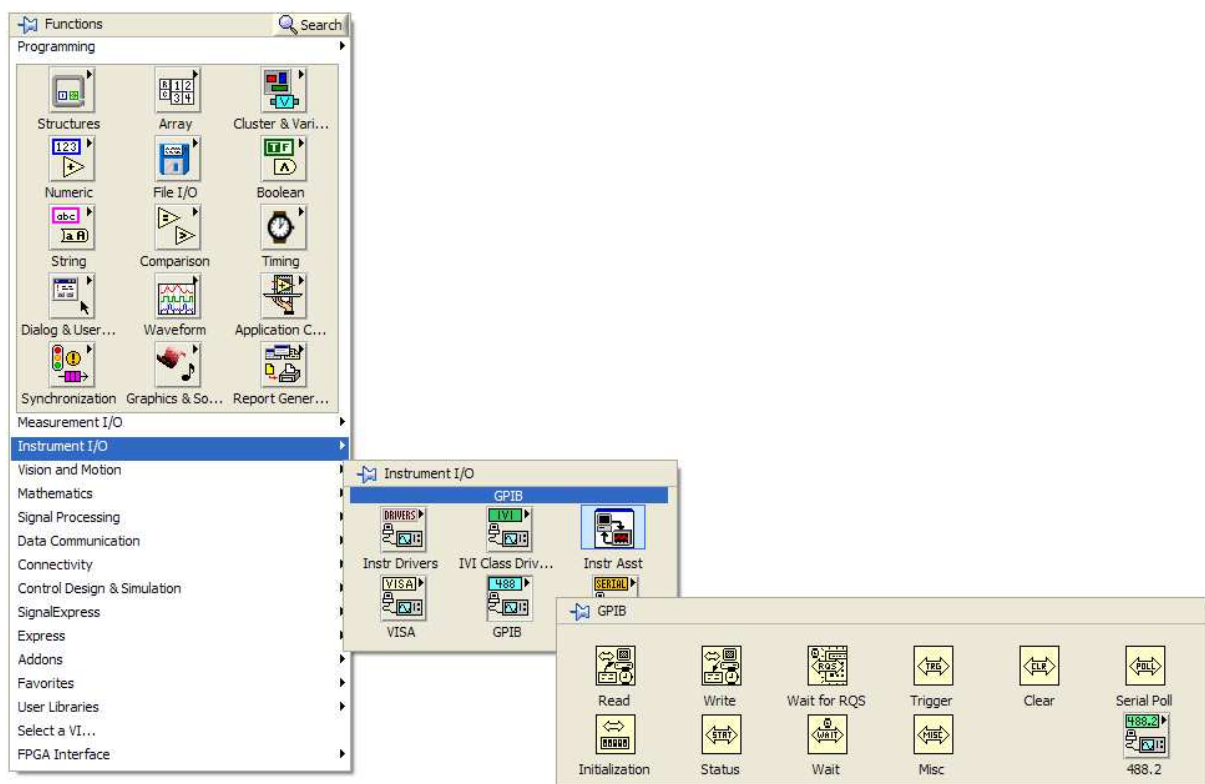
- a- uruchomić środowisko LabView,
- b- utworzyć nowy plik programu: New\Blank_VI,
- c- rozmieścić potrzebne kontrolki (*controls*) w oknie panelu programu,
- d- rozmieścić potrzebne elementy funkcyjne (*functions*) w oknie diagramu programu,
- e- wykonać odpowiednie połączenia na diagramie realizujące algorytm programu,
- f- uruchomić program, ocenić poprawność działania i wyszukać błędy,
- g- usunąć błędy w programie modyfikując zawartość panelu i diagramu programu,
- h- powtarzać punkty f oraz g aż do osiągnięcia oczekiwanego rezultatu.

Każda aplikacja przygotowana w środowisku LabView składa się z dwóch części: Panelu i Diagramu. Panel stanowi graficzny interfejs użytkownika aplikacji, natomiast Diagram jest graficznym zapisem algorytmu realizowanego przez tę aplikację. Po otwarciu aplikacji w środowisku LabView widoczny jest jej Panel sterujący. Przełączanie pomiędzy widokiem Panelu i Diagramu jest możliwe za pomocą kombinacji klawiszy **CTRL+E**. Kombinacją klawiszy **CTRL+T** możemy wybrać jednoczesny widok okna Panelu i Diagramu. Analizę Diagramu programu można sobie znacznie ułatwić włączając przyciskiem  okno pomocy kontekstowej **Context Help**.

Do uruchomienia programu służą przyciski  i . Proste programy nie posiadające w swej strukturze pętli programowych należy zazwyczaj uruchamiać przyciskiem , dzięki czemu pracują one w sposób ciągły i można je zatrzymać przyciskiem . Programy bardziej złożone posiadające w swej strukturze pętle programowe należy uruchamiać zazwyczaj przyciskiem , a do ich zatrzymywania służy odpowiedni przycisk sterujący w tej aplikacji. Uruchomienie programu sygnalizowane jest zmianą postaci przycisków na  i .

2.2. Obsługa interfejsu IEEE-488 w środowisku LabVIEW

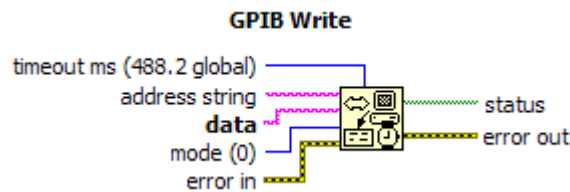
Do obsługi interfejsu IEEE-488 w środowisku LabVIEW można wykorzystać funkcje z biblioteki GPIB. Funkcje te są dostępne w widoku Diagramu na palecie *Function* w opcji \rightarrow Instrument I/O \rightarrow GPIB, tak jak pokazano to na rysunku 7. W ćwiczeniu wykorzystywane będą następujące funkcje: *GPIB Write*, *GPIB Read* oraz *GPIB Trigger*.



Rys. 7. Funkcje *Instrument I/O* do obsługi interfejsu GPIB

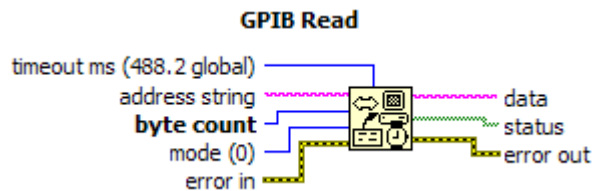
Funkcję *GPIB Write* przedstawiono na rysunku 8. Służy ona do wysyłania danych przez interfejs GPIB. Wymaga ona podania adresu przyrządu dołączonego do magistrali GPIB (*address string*) i danych do wysłania (*data*). Adres i dane do wysłania są zmiennymi typu tekstowego (różowy kolor linii). Wejścia i wyjścia: *error in*, *error out* mogą być

wykorzystane do kaskadowego łączenia funkcji. Należy zauważyć, że dane wysyłane za pomocą funkcji *GPIB Write* są typu tekstowego, a więc podczas wysyłania danych liczbowych należy je uprzednio przekonwertować na zmienną tekstową.



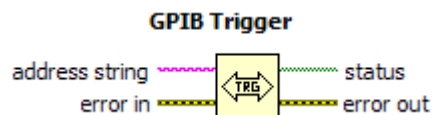
Rys. 8. Funkcja *GPIB Write*

Funkcję *GPIB Read* przedstawiono na rysunku 9. Służy ona do odbierania danych z interfejsu GPIB. Wymaga ona podania adresu przyrządu dołączonego do magistrali GPIB (*address string*) i liczby bajtów danych do odczytania (*byte count*). Przy błędnym podaniu liczby bajtów wystąpią błędy: albo nie zostaną odczytane wszystkie dane (gdy podano za małą wartość *byte count*) lub funkcja będzie bezskutecznie oczekiwała na kolejne dane (gdy podano za dużą wartość *byte count*), aż do przekroczenia czasu *timeout ms*. Podobnie jak dla poprzedniej funkcji, wejścia i wyjścia: *error in*, *error out* mogą być wykorzystane do kaskadowego łączenia funkcji. Odebrane znaki dostępne są na wyjściu *data*. Należy zauważyć, że dane odebrane za pomocą funkcji *GPIB Read* są typu tekstowego, a więc podczas transmisji przez interfejs GPIB danych liczbowych należy odebrane znaki przekonwertować ze zmiennej tekstowej na zmienną numeryczną odpowiedniego typu. Szczególną uwagę należy zwrócić na problem stosowania różnych znaków oddzielających część ułamkową w zapisie dziesiętnym (stosowana jest kropka lub przecinek).



Rys. 9. Funkcja *GPIB Read*

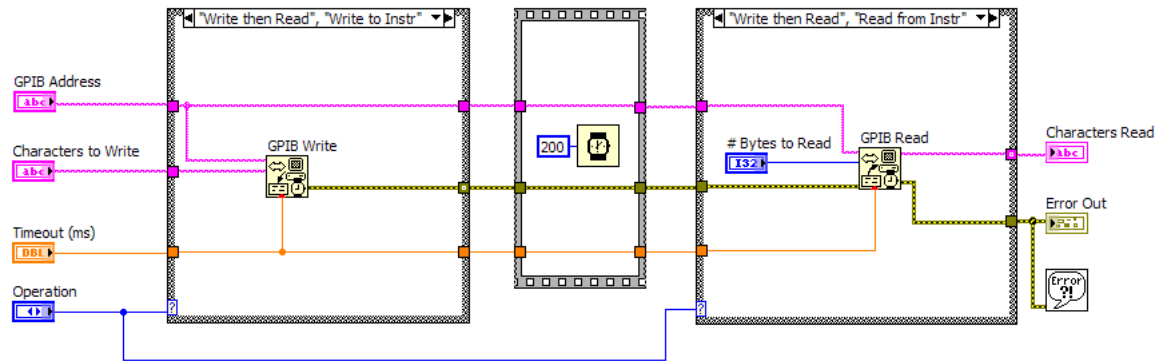
Funkcję *GPIB Trigger* przedstawiono na rysunku 10. Służy ona do wysłania rozkazu *Trigger* do urządzenia o wskazanym adresie. Wymaga ona podania tylko adresu przyrządu dołączonego do magistrali GPIB (*address string*). Podobnie jak dla poprzedniej funkcji, wejścia i wyjścia: *VISA resource name*, *error in*, *error out* mogą być wykorzystane do kaskadowego łączenia funkcji.



Rys. 10. Funkcja *GPIB Trigger*

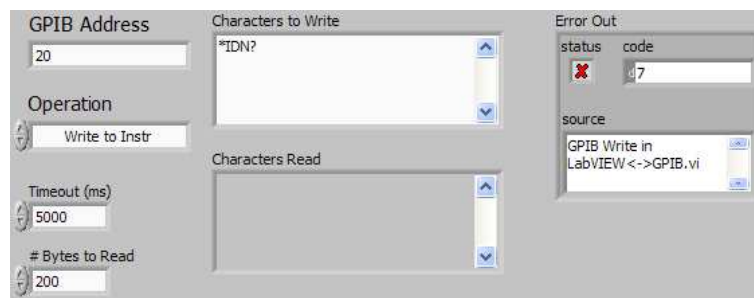
Kompletna obsługa przyrządu pomiarowego dołączonego do magistrali interfejsu GPIB jest możliwa dzięki zastosowaniu jednej lub kilku spośród funkcji: *GPIB Write*, *GPIB Read* oraz *GPIB Trigger*. Firma National Instruments dołączyła do LabVIEW prosty przykład (*NI Example Finder/Hardware Input and Output/GPIB*) w postaci programu *GPIB.VI*,

umożliwiającego podstawową obsługę przyrządów dołączonych do magistrali interfejsu GPIB. Na rysunku 11 przedstawiono Diagram programu GPIB.VI, a na rysunku 12 jego Panel. Program wykorzystuje funkcje: *GPIB Write* oraz *GPIB Read*. Zastosowano również strukturę Case do sterowania przebiegiem programu oraz połączono ze sobą odpowiednio wszystkie wejścia *error in* i wyjścia *error out* kolejnych funkcji.



Rys. 11. Diagram przykładowego programu GPIB.VI

Po wpisaniu w polu GPIB Address adresu urządzenia (np. w ćwiczeniu adres 20 posiada multimetr HP34401A) i w polu Characters to Write odpowiedniej komendy (np.: *IDN?) można uruchomić program i obserwować reakcję przyrządu.



Rys. 12. Panel przykładowego programu GPIB.VI

Przedstawiony program GPIB.VI jest bardzo uproszczony i jego możliwości są niewielkie, można jednak z powodzeniem wykorzystać go do testowania poprawności transmisji, sprawdzenia działania pojedynczych przyrządów pomiarowych, konfigurowania ich, wykonywania pojedynczych pomiarów itp. Należy w tym celu ustalić adres urządzenia, postać komendy do wysłania oraz liczbę znaków do odebrania i wpisać je w odpowiednie miejsca na Panelu.

2.3. Sterowanie komutatorem MERATRONIK typ I201 w środowisku LabVIEW

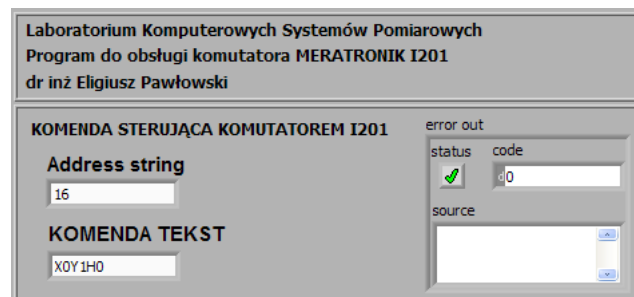
Sterownie komutatorem I201 wymaga wysłania do niego komendy sterującej, która spowoduje załączenie wybranego kanału wejściowego oraz załączenie przekaźnika wyjściowego. Strukturę odpowiedniej do tego celu komendy sterującej **X n Y m H w** szczegółowo omówiono w p. 1.3. Na rysunku 13 przedstawiono przykładowy Diagram programu sterującego komutatorem I201. Do obsługi interfejsu GPIB zastosowano funkcje: *GPIB Write* oraz *GPIB Trigger*. Funkcja *GPIB Read* do tego celu nie jest przydatna, gdyż komutator I201 nie wysyła żadnych danych na magistralę interfejsu. Do wysłania komendy programującej komutator wykorzystano funkcję *GPIB Write*. Na jej wejściach *address string*

oraz *data* utworzono prawym kliknięciem zadajniki (*Create/Control*), dzięki którym na Panelu programu można wprowadzić adres skanera (skaner posiada adres 16) i komendę programującą (np.: X0Y1H) - załączenie kanału nr 1). Aby komutator wykonał wysłaną do niego komendę, należy wykorzystać funkcję *Trigger*. Na jej wyjściu *error out* utworzono wskaźnik (*Create* → *Indicator*) pokazujący na Panelu ewentualne błędy w transmisji.

Aby użyte funkcje były wykonane w odpowiedniej kolejności (wysłanie komendy *Write* → wysłanie rozkazu *Trigger*) połączono ze sobą odpowiednio wejście *error in* i wyjście *error out* tych funkcji. Przykładowy Panel programu sterującego komutatorem I201 przedstawiono na rysunku 14.



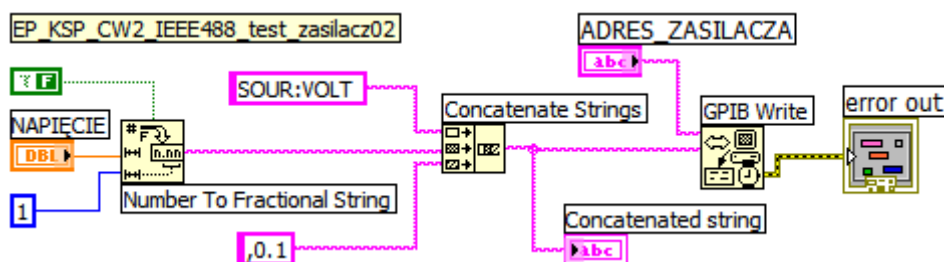
Rys. 13. Diagram programu sterującego komutatorem I201



Rys. 14. Panel programu sterującego komutatorem I201

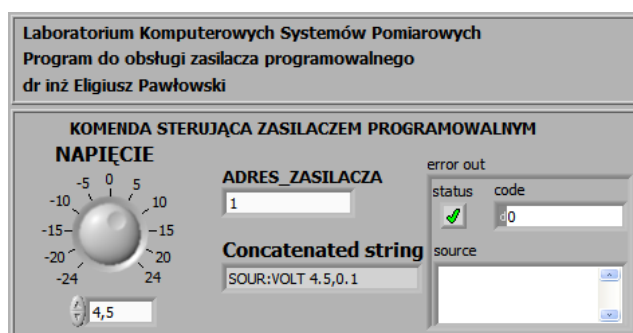
2.4. Sterowanie zasilaczem programowalnym w środowisku LabVIEW

Sterownie zasilaczem programowalnym wymaga wysłania do niego komendy sterującej, która spowoduje zaprogramowanie wybranych wartości napięcia wyjściowego oraz wartości ograniczenia prądowego. Strukturę odpowiedniej do tego celu komendy sterującej szczegółowo omówiono w p. 1.4. Przykładowo komenda **SOUR:VOLT 24.0,0.5** zaprogramuje na wyjściu zasilacza napięcie +24V i ograniczenia prądowe 0,5A. Inne wartości napięcia i prądu wymagają zastosowania w odpowiednich polach komendy odpowiednich liczb. Komenda wysyłana jest w postaci tekstowej, co wymaga odpowiedniego przekonwertowania zmiennych liczbowych. Na rysunku 15 przedstawiono przykładowy Diagram programu sterującego zasilaczem programowalnym.



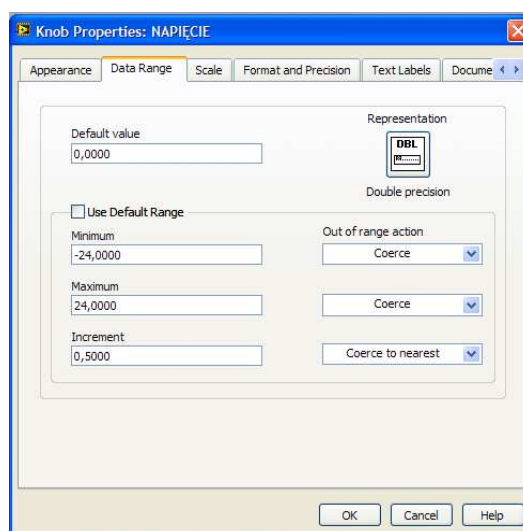
Rys. 15. Diagram programu sterującego zasilaczem programowalnym

Do obsługi interfejsu GPIB zastosowano funkcję *GPIB Write*. Funkcja *GPIB Trigger* nie jest wymagana, ponieważ zasilacz jest zgodny z językiem SCPI. Funkcja *GPIB Read* również nie jest przydatna, gdyż zasilacz nie wysyła żadnych danych na magistralę interfejsu. Do wysłania komendy programującej zasilacz wykorzystano więc tylko funkcję *GPIB Write*. Na jej wejściu *address string* utworzono prawym kliknięciem zadajnik (*Create/Control*), dzięki którym na Panelu programu można wprowadzić adres zasilacza (zasilacz posiada adres 1). Komendę programującą napięcie wyjściowe i ograniczenia prądowe utworzono funkcją *Concatenate String* i dołączyło do wejścia *data* funkcji *GPIB Write*. Na jej wyjściu *error out* utworzono wskaźnik (*Create →Indicator*) pokazujący na Panelu ewentualne błędy w transmisji. Przykładowy Panel programu sterującego zasilaczem programowanym przedstawiono na rysunku 16.



Rys. 16. Panel programu sterującego zasilaczem programowanym

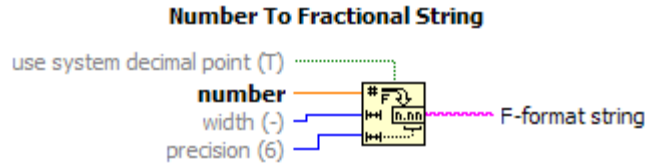
Do zadawania napięcia programującego zasilacz zastosowano zadajnik numeryczny w postaci pokrętki (*Controls/Numeric/Knob*) z aktywnym dodatkowo wskaźnikiem cyfrowym (*Visible Items/Digital Display*). Ponieważ zasilacz można zaprogramować napięciami tylko z zakresu $-24\text{V} \dots +24\text{V}$ z krokiem $0,5\text{V}$, konieczne jest odpowiednie skonfigurowanie tego zadajnika (*Knob Properties/Data Range*), co przedstawiono na rysunku 17.



Rys. 17. Konfiguracja zadajnika wartości napięcia do programowania zasilacza

Wartość liczbową uzyskaną z zadajnika jest następnie konwertowana do zmiennej tekstowej za pomocą funkcji *Number To fractional String*, przedstawionej na rysunku 18. Należy podać precyzję jednego miejsca dziesiętnego oraz zdefiniować znak oddzielający

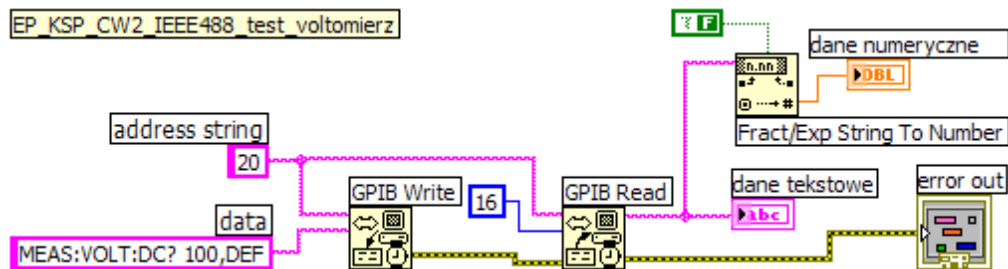
część ułamkową w postać kropki dziesiętnej, zgodnie z wymaganiami komendy sterującej zasilaczem. Stała część komendy **SOUR:VOLT** oraz ograniczenie prądowe, **0.5** nie ulegają zmianie i są zdefiniowane jako stałe tekstowe. Na wyjściu funkcji *Concatenate String* utworzono dodatkowo wskaźnik tekstowy ((*Create* → *Indicator*) umożliwiający na Panelu sprawdzenie postaci utworzonej komendy programującej zasilacz.



Rys. 18. Funkcja *Number To Fractional String* konwertująca liczbę na tekst

2.5. Pomiary multimetrem HP34401A w środowisku LabVIEW

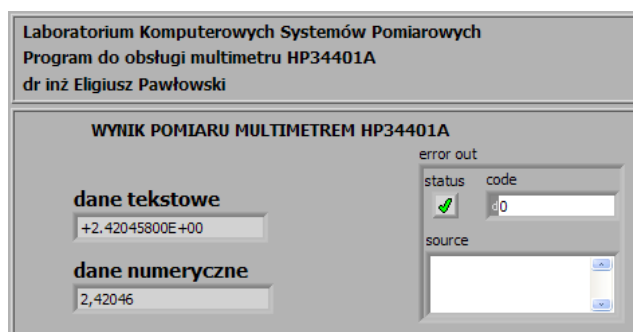
Realizacja pomiarów za pomocą multimetru HP34401A wymaga wysłania do niego komendy programującej funkcję pomiarową, zakres pomiarowy i rozdzielczość oraz wymuszającej wykonanie pomiaru i następnie odczytanie wartości zmierzonej wielkości. Strukturę odpowiedniej do tego celu komendy **MEAS:VOLT:DC? 100,DEF** szczegółowo omówiono w p. 1.5. Na rysunku 19 przedstawiono przykładowy Diagram programu realizującego pomiar multimetrem HP34401A napięcia stałego na zakresie 100V z domyślną rozdzielczością.



Rys. 19. Diagram programu do pomiarów napięcia multimetrem HP34401A

Program wykorzystuje funkcje: *GPIB Write* oraz *GPIB Read*. Adres multimetru (*adres string*) oraz komendę programującą (*data*) zdefiniowano jako stałe tekstowe. Zastosowano funkcję *Fract/EXP String To Number* w celu przekonwertowania danych tekstowych odczytanych z multimetru na dane liczbowe. Połączono również ze sobą odpowiednio wszystkie wejścia *error in* i wyjścia *error out* kolejnych funkcji dla zapewnienia odpowiedniej kolejności realizacji programu. Na wyjściu funkcji *GPIB Read* utworzono wskaźnik tekstowy (prawe kliknięcie >Create>Indicator), a na wyjściu funkcji *Fract/EXP String To Number* analogicznie utworzono wskaźnik numeryczny. Stała numeryczna całkowita (kolor niebieski) o wartości 16 ustala liczbę znaków wyniku pomiaru odczytywanego z multimetru. Należy przy tym pamiętać, że wynik pomiaru otrzymany z multimetru zawiera kropkę rozdzielającą część całkowitą od ułamkowej w zapisie dziesiętnym, podczas gdy w komputerze z zainstalowaną polską wersją systemu Windows oczekiwany jest znak rozdzielający w postaci przecinka. Wymaga to szczególnej uwagi podczas konwersji wyniku pomiaru z postaci tekstowej na liczbową. Przykładowy Panel programu realizującego pomiary multimetrem przedstawiono na rysunku 20. Można na nim

zauważyć różnicę w stosowanych znakach oddzielających część ułamkową w wyniku zapisywanym jako tekst i jako zmienna numeryczna.



Rys. 20. Panel programu do pomiarów napięcia multimetrem HP34401A

3. Ogólne zasady realizacji ćwiczenia

Ćwiczenie opiera się na wykorzystaniu środowiska LabVIEW i opisanego w rozdziale 1 stanowiska laboratoryjnego do oprogramowania automatycznego wyznaczania charakterystyk prądowo-napięciowych wybranych dwójników pasywnych. Szczegółowa dokumentacja wykorzystywanej aparatury będzie dostępna w laboratorium. Studenci samodzielnie powinni zaopatrzyć się w literaturę wspomagającą naukę programowania w LabVIEW. Przykładowe pozycje, w tym dostępne w Internecie darmowe podręczniki, zamieszczono w wykazie literatury.

3.1. Organizacja systemu plików na komputerze

Przystępując do realizacji ćwiczenia należy najpierw uruchomić komputer, a dopiero w drugiej kolejności włączyć zasilanie pozostałej aparatury. Wyłączamy aparaturę w kolejności odwrotnej.

Wszystkie pliki z programami napisanymi podczas realizacji ćwiczenia należy zapisywać w katalogu C:\LABORKA_KSP\STUDENT\RRRRMMDD_G_Z, gdzie RRRR, MM, DD oznaczają kolejno: rok, miesiąc i dzień rozpoczęcia wykonywania ćwiczenia, a G i Z oznaczają grupę i zespół laboratoryjny. Zapisywane pliki powinny posiadać nazwę utworzoną z początkowych liter nazwiska jednego z członków zespołu oraz kolejnych cyfr 1, 2, 3 itd., np.: **KOWALS_1.Vi**, **KOWALS_2.Vi** itd. Pliki utworzonych w czasie zajęć przyrządów wirtualnych do obsługi pojedynczych przyrządów powinny również zawierać w nazwie pliku programu symbol obsługiwanego przyrządu (np. zasilacz, komutator, woltomierz).

Podczas realizacji kolejnego zadania często należy wykorzystać program opracowany w ramach wcześniejszego punktu. Nie wolno jednak w tym celu modyfikować programu będącego finalną wersją wcześniej zrealizowanego zadania. Po zapisaniu finalnej wersji programu realizującego w pełni określony punkt ćwiczenia, należy wykonać jego kopię nadając mu nową nazwę, zmieniając końcową cyfrę na kolejną wartość. Kopię programu należy wykonywać opcją *File\Save As*, wybierając następnie opcję *Copy - create copy on disk\Substitute copy for original (Copy will be in memory. Original will be closed)*. Pracując na tak utworzonej kopii programu należy ją uzupełniać o nowe elementy, aż do zrealizowania kolejnego punktu ćwiczenia.

Wszystkie realizowane podczas zajęć programy powinny zawierać dane osobowe zespołu laboratoryjnego, numer grupy dziekańskiej oraz datę wykonywania ćwiczenia. Informacje te należy podać w postaci **komentarza umieszczonego na panelu sterującym i na diagramie**.

W każdym realizowanym punkcie oczekiwana jest inwencja własna studentów. Wydruki programów oraz przykładowe pliki udostępniane ćwiczącym należy traktować wyłącznie jako

przykłady, a nie jako obowiązujące wzorce. Przykłady te zazwyczaj realizują zadania stawiane w ćwiczeniach tylko w ograniczonym zakresie, **mniej niż wymagane**.

3.3. Wydruk dokumentacji programu

Środowisko LabVIEW pozwala w prosty sposób utworzyć na dysku twardym komputera pliki z dokumentacją zrealizowanego przyrządu wirtualnego. Pliki będą zawierać obraz Panelu oraz Diagramu. Kolejność postępowania jest następująca:

- wybrać opcję **File/Print..** i w oknie **Select VI(s)** zaznaczyć nazwę aplikacji do wydruku, wcisnąć **NEXT**,
- w oknie **Print Contents** zaznaczyć opcję **VI documentation**, wcisnąć **NEXT**,
- w oknie **VI Documentation** zaznaczyć opcje: **Front Panel, Controls (connected Controls), Descriptions, Data type information, Label, Block Diagram**, wcisnąć **NEXT**,
- w oknie **Destination** wybrać opcję **HTML File**, wcisnąć **NEXT**,
- w oknie **HTML** wybieramy **Image format: GIF (uncompressed), color depth: 256 colors**, wcisnąć **SAVE**,
- w oknie **SAVE** wybrać katalog (jeśli go jeszcze nie ma, to należy go utworzyć zgodnie z podaną wcześniej regułą). Zapisać plik.
- Odszukać zapisane pliki na dysku i sprawdzić ich zawartość.

Zanotować w protokole nazwę utworzonego katalogu i nazwy zapisanych w nim plików z opisem zawartości.

3.4. Zapisywanie wykresów do plików graficznych

Aby zapisać do pliku dyskowego uzyskany w programie na wykresie przebieg sygnału, należy wykonać kolejno następujące czynności:

- ustawić kursor myszki na oknie przebiegu sygnału i kliknąć prawym przyciskiem myszki,
- z otworzonego menu wybrać opcję **Export Simplified Image**, zaznaczyć opcję **Bitmap (BMP) i Save to file**,
- wybrać katalog utworzony przez grupę laboratoryjną na początku zajęć i wpisać nazwę pliku odpowiednio do zawartości, zatwierdzić **OK** i zapisać **Save**,
- sprawdzić zawartość pliku i zanotować w protokole nazwę pliku z zapisanym przebiegiem i opisać jego zawartość.

3.5. Przekształcanie przyrządu wirtualnego w podprogram

Środowisko LabVIEW umożliwia przekształcenie napisanego samodzielnie programu w podprogram, który może być potem wykorzystywany w innych aplikacjach w ten sam sposób, jak wszystkie inne elementy dostępne w palecie *Functions*. W tym celu należy utworzyć ikonę, której obraz będzie identyfikował na diagramie dany podprogram, oraz zdefiniować konektory, które będą służyć jako wejścia i wyjścia danych w diagramie programu. Kolejność postępowania jest następująca:

- utworzyć w LabVIEW program, który chcemy przekształcić w podprogram,
- przełączyć się na widok Panelu,
- w prawym górnym rogu okna Panel znajduje się ikona utworzonego programu, należy umieścić kursor na tej ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać „*Edit Icon*”,
- dokonać edycji ikony w sposób umożliwiający łatwą identyfikację podprogramu po umieszczeniu go na diagramie,
- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*Show Connector*”. W prawym górnym rogu zostanie wyświetlona siatka wejść i wyjść programu,

- ponownie umieścić kursor na ikonie i prawym kliknięciem rozwinąć listę opcji, wybrać opcję „*Patterns*” i z dostępnych wzorów zacisków wybrać konfigurację odpowiadającą potrzebom tworzonego podprogramu. W miarę potrzeby dla zwiększenia czytelności połączeń można siatkę konektorów obrócić o 90 stopni opcją *Rotate 90 Degrees*,

- w celu przypisania wejść i wyjść danych do konektorów należy kliknąć na wybrany konektor (zmieni on kolor) i następnie kliknąć w oknie Panelu na kontrolkę reprezentującą wybrane wejście lub wyjście danych w naszym podprogramie. Konektor zmieni kolor odpowiednio do typu danych z nim powiązanych,

- przypisać kolejno wszystkie konektory do wejść i wyjść podprogramu,

- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*Show Icon*”.

- ponownie umieścić kursor na ikonie, prawym kliknięciem rozwinąć listę opcji i wybrać opcję „*VI properties*”. W celu utworzenia opisu podprogramu w widoku Panelu wybrać opcję *Documentation* i wpisać w polu "*VI description*" tekst opisujący przeznaczenie programu, który będzie pojawiał się w okienku helpu kontekstowego,

- zapisać program do pliku dyskowego i zamknąć,

W celu wywołania utworzonego podprogramu należy w kolejnym tworzonej nowym programie przejść do okna Diagramu i z palety „*Functions*” wybrać „*Select a VI ..*”, odnaleźć na dysku komputera wcześniej utworzony podprogram i wstawić do diagramu tworzonego nowego programu.

4. Wykonanie ćwiczenia

4.1. Uruchomienie stanowiska ćwiczeniowego

- Zapoznać się z konstrukcją stanowiska i porównać ze schematami przedstawionymi w rozdziale 1 instrukcji. W razie zauważenia niezgodności połączeń ze schematem, po uzgodnieniu z prowadzącym zajęcia, wprowadzić niezbędne poprawki w układzie połączeń.

- Zgodnie z zaleceniami podanymi w rozdziale 3 uruchomić stanowisko: włączyć komputer i poczekać na uruchomienie systemu operacyjnego. Uruchomić środowisko LabVIEW. Zanotować do protokołu konfigurację komputera: wersję systemu operacyjnego, typ procesora, częstotliwość taktowania, ilość pamięci RAM, wielkość dysku twardego, wersję LabVIEW. Odczytać rodzaje i typy przyrządów pomiarowych zastosowanych na stanowisku i zapisać do protokołu.



- Włączyć zasilanie wszystkich przyrządów pomiarowych.

- Po włączeniu zasilacza należy zaprogramować go do pracy z interfejsem IEEE-488 z adresem 1. Zanotować do protokołu adresy wszystkich przyrządów pomiarowych.



- Utworzyć na dysku komputera katalog do zapisywania programów i pozostałych danych zgodnie z zasadą przedstawioną w punkcie 3.1. Zanotować nazwę utworzonego katalogu do protokołu.



4.2. Obsługa interfejsu IEEE-488 przykładowym programem

- Otworzyć w środowisku LabVIEW przykładowy program do obsługi interfejsu GPIB opisany w punkcie 2.2. i zapisać go pod własną nazwą zgodnie z regułami przedstawionymi w punkcie 3.1. Zapoznać się ze strukturą diagramu i organizacją panelu. **W dalszej części ćwiczenia należy wykorzystywać wyłącznie własną kopię zapisaną programu testowego.**

- Na podstawie punktu 1.4 instrukcji wpisać w odpowiednie okienka Panelu adres i dane programujące zasilacz. Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu



program samoczynnie powinien się zatrzymać. Sprawdzić reakcję zasilacza i programu na wysłaną komendę. Zanotować wnioski do protokołu.

- Na podstawie punktu 1.5 instrukcji wpisać w odpowiednie okienka Panelu adres i dane programujące multimetr HP34401A. Dołączyć wyjście zasilacza bezpośrednio do wejścia woltomierza. Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać. Sprawdzić reakcję woltomierza i programu na wysłaną komendę. Zanotować wnioski do protokołu.

- Na podstawie punktu 1.3 instrukcji wpisać w odpowiednie okienka Panelu adres i dane programujące komutator I201. Dołączyć wyjście zasilacza i woltomierz zgodnie ze schematem stanowiska przedstawionym na rysunku 5. Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać. Sprawdzić reakcję komutatora i programu na wysłaną komendę. Zanotować wnioski do protokołu.

4.3. Programowanie obsługi komutatora I201

- Przygotować program sterujący komutatorem I201 według punktu 2.3. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.

- Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać. Sprawdzić, czy zadziałał odpowiedni kanał komutatora i zapaliła się odpowiednia dioda LED (kropka dziesiętna). Ocenić poprawność działania programu i usunąć ewentualne błędy. Zmienić wartość danych programujących komutator i sprawdzić działanie innych kanałów. Zanotować do protokołu postać wysyłanych komend i odpowiadające im zachowanie się przekaźników oraz diody LED.

- Zwrócić uwagę na zachowanie się diody LISTEN na obudowie komutatora, zanotować wynik obserwacji do protokołu.



- Jeśli program działa poprawnie, uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.4. Programowanie obsługi zasilacza programowalnego

- Przygotować program sterujący zasilaczem programowalnym według punktu 2.4. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.

- Dołączyć multimetr bezpośrednio do wyjścia zasilacza programowalnego. Jeśli multimetr znajduje się w trybie pracy zdalnej (REMOTE - na wyświetlaczy widoczny napis Rmt), należy przełączyć go tryb LOCAL.



- Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać. Sprawdzić, czy zasilacz poprawnie zaprogramował się do zaplanowanej wartości napięcia, odczytać wartość napięcia zmierzoną przez multimetr. Ocenić poprawność działania programu i usunąć ewentualne błędy. Zmienić wartość danych programujących zasilacz i sprawdzić działanie programu. Zwrócić uwagę na postać znaku dziesiętnego w komendzie tekstowej programującej zasilacz i w zapisie liczbowym. Zanotować do protokołu postać wysyłanych komend i odpowiadające im zachowanie się zasilacza oraz wskazania multimetru.

- Jeśli program działa poprawnie, uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.

- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.5. Programowanie obsługi multimetru HP34401A

- Przygotować program sterujący multimetrem HP34401A według punktu 2.5. Zapisać program we wskazanym katalogu pod nazwą utworzoną zgodnie z zaleceniami podanymi w punkcie 3.1.

- Uruchomić program przyciskiem . Ponieważ **program nie zawiera żadnej pętli**, przycisk  pojawi się tylko na chwilę i po jednorazowym wykonaniu program samoczynnie powinien się zatrzymać. Sprawdzić, czy multimetr przełączył się w tryb REMOTE oraz czy wykonał poprawny pomiar napięcia. Ocenić poprawność działania programu i usunąć ewentualne błędy. Zmienić wartość danych programujących woltomierz i sprawdzić jego działanie na innych zakresach pomiarowych. Zwrócić uwagę na postać znaku dziesiętnego w danych tekstowych odczytanych z multimetru i po przekonwertowaniu wyniku pomiaru na zapis liczbowy. Zanotować do protokołu postać wysyłanych komend i odpowiadające im zachowanie się multimetru.

- Jeśli program działa poprawnie, uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać finalną wersję programu na dysku, zanotować jego nazwę do protokołu.



- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

4.6. Programowanie automatycznego badania zasilacza

- Przygotować program sterujący zasilaczem programowalnym i multimetrem cyfrowym, który będzie automatycznie programował zasilacz do kolejnych wartości napięć, mierzył napięcie na wyjściu zasilacza woltomierzem, obliczał błąd tego napięcia względem wartości zaprogramowanej i wykresował wykres. Docelowy program będzie tworzony w kilku krokach.

- Punktem wyjściowym będzie program sterujący zasilaczem przygotowany w punkcie 4.4. W pierwszej kolejności należy program z punktu 4.4 **zapisać pod nową nazwą**. Następnie cały program należy objąć pętlą *For Loop*. Stałe tekstowe tworzące rozkaz sterujący zasilaczem przenieść na zewnątrz pętli. Adres zasilacza utworzyć jako stałą tekstową. Usunąć zadajnik numeryczny zadający wartość napięcia i w jego miejsce umieścić algorytm obliczający wartość napięcia na podstawie zmiennej *i* indeksującej kolejne obiegi pętli *For Loop*. Przykładowy algorytm przedstawiono na rysunku 21. Na panelu należy umieścić zadajniki numeryczne dla wartości początkowej napięcia (Start wolty), kroku zmiany napięcia (Krok wolty) oraz liczby iteracji dla pętli *For Loop*. Następnie umieścić na Panelu wskaźniki do kontroli poprawności działania programu: Numer kroku oraz Napięcie zaprogramowane.

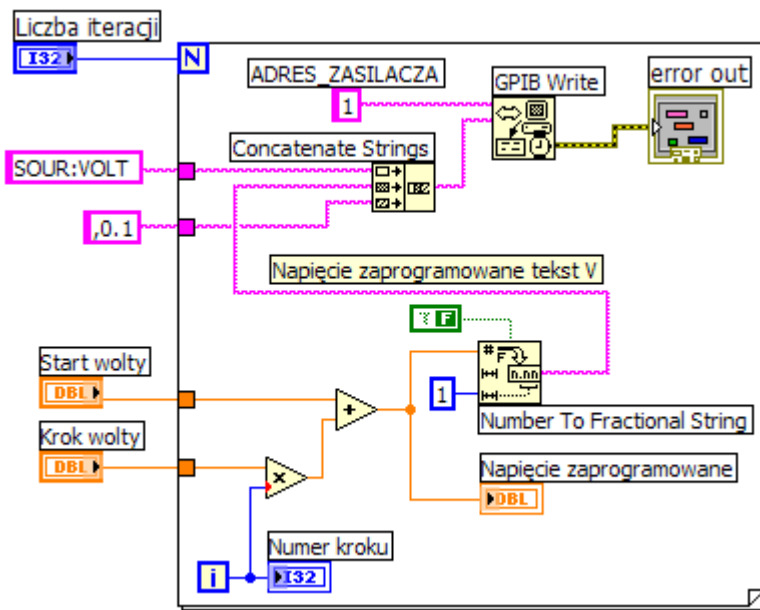
- Dołączyć multimetr bezpośrednio do wyjścia zasilacza programowalnego. Jeśli multimetr znajduje się w trybie pracy zdalnej (REMOTE - na wyświetlaczu widoczny jest napis Rmt), należy przełączyć go tryb LOCAL.

- Na Panelu programu wprowadzić wartości początkowe: Start wolty -24, Krok wolty 0,5, liczba iteracji 97. Uruchomić program przyciskiem . Sprawdzić, czy po pojawieniu się ikony  program zaprogramuje wszystkie wartości napięcia od -24V do +24V z krokiem 0,5V. Obserwować wyświetlacz zasilacza oraz wskazania multimetru.

- Jeśli program działa poprawnie uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać program na dysku jako wersję przejściową, zanotować jego nazwę do protokołu.

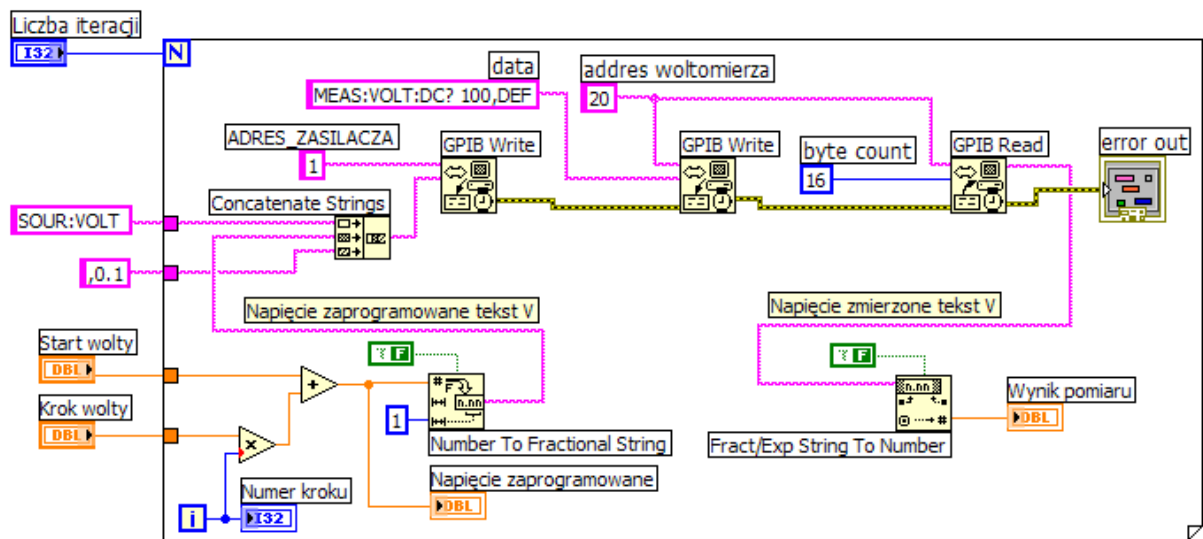
- **Skopiować utworzony program pod nową nazwą** w celu jego dalszej modyfikacji. Kolejnym etapem będzie uzupełnienie programu o pomiar napięcia wyjściowego zasilacza za

pomocą multimetru. W tym celu należy powiększyć obszar objęty pętlą *For Loop* i skopiować do jej wnętrza algorytm obsługi multimetru przygotowany w punkcie 4.5.




Rys. 21. Algorytmu programujący zasilacz kolejnymi wartościami napięcia w pętli *For Loop* od wartości Start wolty z krokiem Krok wolty

- Połączyć wewnątrz pętli *For Loop* algorytm programujący zasilacz z algorytmem pomiaru napięcia multimetrem. Przykładowy Diagram uzyskanego programu przedstawiono na rysunku 22. Uporządkować Panel programu.



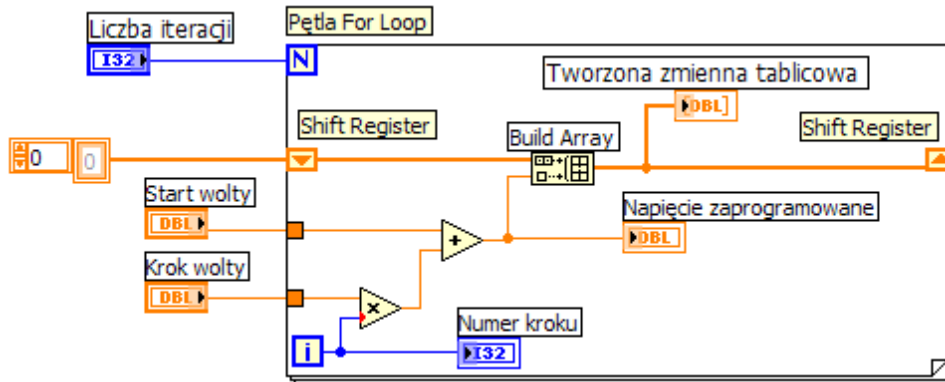
Rys. 22. Algorytmu programujący zasilacz kolejnymi wartościami napięcia w pętli *For Loop* oraz wykonujący pomiar napięcia multimetrem

- Uruchomić program przyciskiem . Sprawdzić poprawność programowania zasilacza oraz wyniki pomiarów napięcia odczytywane przez program z multimetru. Upewnić się, czy program zaprogramuje wszystkie wartości napięcia od -24V do +24V z krokiem 0,5V. Obserwować wyświetlacz zasilacza, wskazania multimetru oraz wartości pokazywane na

Panelu programu. Sprawdzić czy multimetr przełączył się z trybu LOCAL w tryb pracy zdalnej REMOTE - na wyświetlaczu powinien być widoczny napis Rmt.

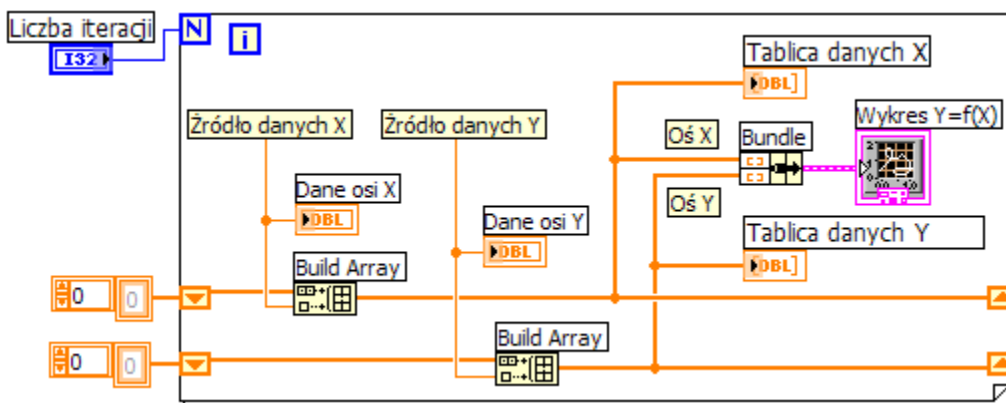
- Jeśli program działa poprawnie, uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać program na dysku jako wersję przejściową, zanotować jego nazwę do protokołu.

- **Skopiować utworzony program pod nową nazwą** w celu jego dalszej modyfikacji. Kolejnym etapem będzie uzupełnienie programu o gromadzenie kolejnych wyników w zmiennej tablicowej, obliczenie błędów napięcia wyjściowego zasilacza i wykreślenie wykresu tych błędów w funkcji napięcia programowanego. Zasadę tworzenia zmiennej tablicowej przedstawiono na rysunku 23.



Rys. 23. Algorytmu tworzący zmienną tablicową *Array* w celu gromadzenia w niej kolejnych wartości napięcia programującego zasilacz

Na krawędzi pętli *For Loop* należy utworzyć rejestr przesuwany (prawe kliknięcie>Add Shift Register). Do tworzenia tablicy służy funkcja *Build Array*. Do zacisku wejściowego rejestru przesuwanego (na lewej krawędzi pętli) należy dołączyć stałą tablicową o wartości zerowej (*Create>Constant*). Przedstawiony algorytm zapisze w tablicy *Array* wszystkie wartości napięcia programującego zasilacz. Aby wykreślić wykres konieczne jest utworzenia dwóch tablic, gromadzących dane dla osi X i dla osi Y. Odpowiedni fragment algorytmu przedstawiono na rysunku 24.

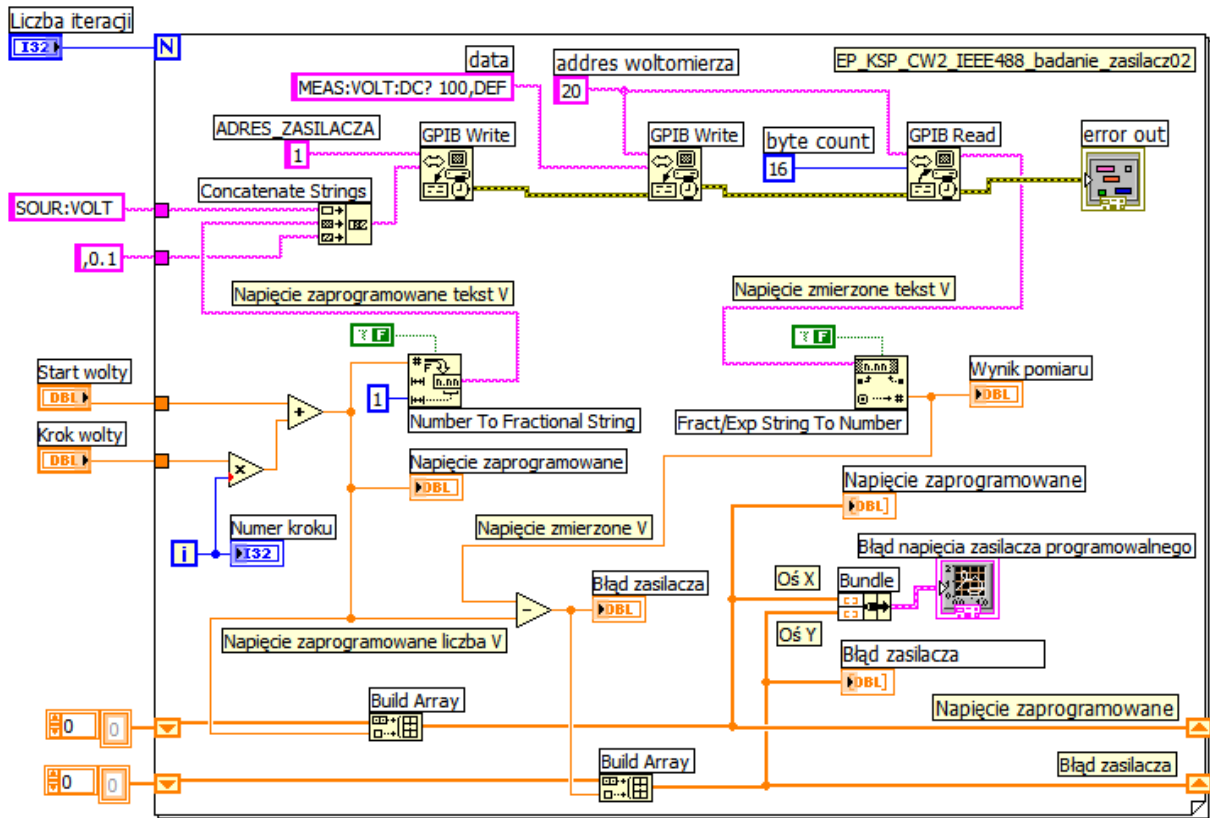


Rys. 24. Fragment algorytmu tworzący dwie zmienne tablicowe *Array* w celu gromadzenia danych dla osi X, Y i wykreślenia wykresu XY Graph

Do gromadzenia danych dla osi X i Y należy wykorzystać dwie funkcje *Build Array* analogicznie jak to przedstawiono wcześniej na rysunku 23. Następnie obie tablice należy


połączyć ze sobą w jeden klaster za pomocą funkcji Bundle. Uzyskany klaster łączący dwie tablice zawierające dane dla osi X i Y należy dołączyć do wejścia funkcji XY Graph, która wykreśla na Panelu wykres typu $Y=f(X)$.


- Korzystając ze wskazówek zawartych w opisach do rysunków 23 i 24 należy uzupełnić program o wykreślanie wykresu błędów napięcia wyjściowego zasilacza w funkcji napięcia programującego. Odpowiedni Diagram oraz Panel przedstawiono na rysunkach 25 i 26. Błąd napięcia wyjściowego należy obliczyć jako różnicę wyniku pomiaru napięcia multimetrem i wartości napięcia programującego zasilacza. Do tablicy osi X należy zapisywać wartości napięcia programującego zasilacza, a do tablicy osi Y błędy zasilacza.



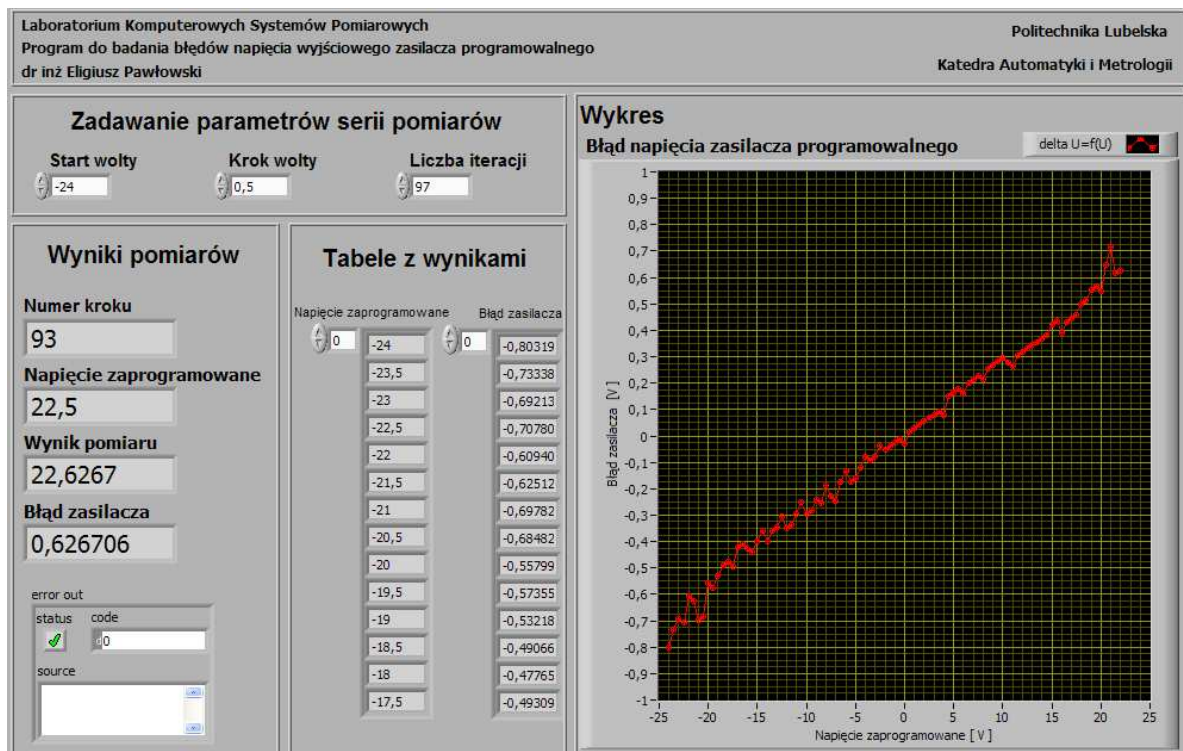
Rys. 25. Diagram programu do automatycznego badania zasilacza programowalnego

Na Panelu programu powinny znajdować się zadajniki numeryczne do ustawiania napięcia początkowego, kroku napięcia i liczby kroków. Jako dane wyjściowe należy na Panelu umieścić przede wszystkim wykres błędów w funkcji napięcia programującego oraz: numer kroku iteracji pętli For Loop, wartość napięcia zaprogramowanego, napięcia zmierzonego oraz obliczony błąd zasilacza. Dodatkowo na Panelu należy umieścić wskaźniki numeryczne umożliwiające przeglądanie tablic z wynikami przedstawionymi na wykresie.

- Dołączyć multimetr bezpośrednio do wyjścia zasilacza programowalnego.
- Na Panelu programu wprowadzić wartości początkowe: Start wolty -24, Krok wolty 2, liczba iteracji 25. Uruchomić program przyciskiem . Sprawdzić poprawność działania programu i ocenić zgodność otrzymanego wykresu z oczekiwaniami.
- Jeśli program działa poprawnie uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać program na dysku jako wersję finalną dla punktu 4.6, zanotować jego nazwę do protokołu.
- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.

- Na Panelu programu wprowadzić wartości początkowe: Start wolty -24, Krok wolty 0,5, liczba iteracji 97. Uruchomić program przyciskiem . Podczas działania programu wykonać zrzut ekranu (klawisz PrtScr na klawiaturze) i korzystając z programu graficznego Paint zapisać go na dysku w celu wykorzystania do sprawozdania. Uzyskany wykres zapisać do pliku graficznego wykorzystując opcję **Export Simplified Image** (według punktu 3.4) i umieścić w sprawozdaniu.

- **Skopiować utworzony program pod nową nazwą** w celu wykorzystania go do realizacji kolejnego punktu.



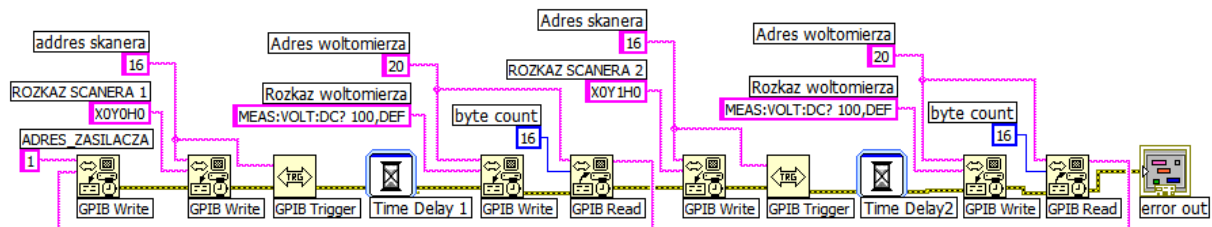
Rys. 26. Panel programu do automatycznego badania zasilacza programowalnego

4.7. Programowanie automatycznego wyznaczenia charakterystyki dwójnika

- Przygotować program sterujący zasilaczem programowalnym, multimetrem cyfrowym i komutatorem w celu automatycznego wyznaczania charakterystyki prądowo-napięciowej dwójnika pasywnego. Program będzie automatycznie programował zasilacz do kolejnych wartości napięć, przełączał multimetr za pomocą komutatora, mierzył napięcia na badanym dwójniku i na rezystorze bocznikowym o znanej rezystancji, obliczał prąd płynący przez badanych dwójnik oraz moce tracone w rezystorze i dwójniku. Docelowy program będzie tworzony w kilku krokach.

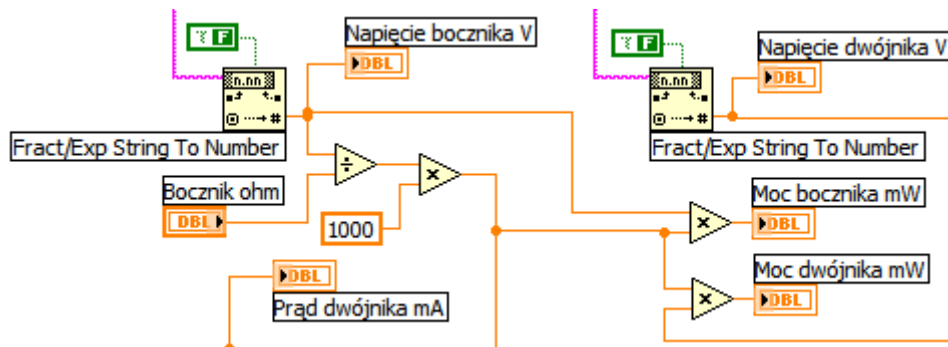
- Punktem wyjściowym będzie program sterujący zasilaczem i multimetrem przygotowany w punkcie 4.6. W pierwszej kolejności należy program z punktu 4.6 **zapisać pod nową nazwą**. Następnie program ten należy uzupełnić o sterowanie komutatorem. W tym celu należy powiększyć obszar objęty pętlą *For Loop* i skopiować do jej wnętrza algorytm sterowania komutatorem przygotowany w punkcie 4.3. Wszystkie teksty programujące adresy i komendy należy utworzyć jako stałe tekstowe, za wyjątkiem programowania napięcia zasilacza, gdyż ten fragment algorytmu będzie taki sam jak w p. 4.6. Programowanie komutatora należy umieścić po programowaniu zasilacza, ale przed programowaniem multimetru. Dodatkowo pomiędzy programowaniem komutatora i multimetru należy

umieścić opóźnienie czasowe (0,2s) za pomocą funkcji Time Delay. Jest to konieczne ze względu na wolne działanie przekaźników w komutatorze. Fragment obsługi komutatora i multimetru należy zastosować dwukrotnie, mierząc napięcie kolejno w kanale 0 i w kanale 1. Przykładowy fragment tej części algorytmu przedstawiono na rysunku 27. Na panelu powinny znajdować się zadajniki numeryczne dla wartości początkowej napięcia (Start wolty), kroku zmiany napięcia (Krok wolty), liczby iteracji dla pętli *For Loop*. oraz dodatkowy zadajnik do wprowadzenia wartości rezystora bocznikującego R_B (Bocznik ohm).



Rys. 27. Fragment diagramu realizującego pomiar charakterystyki prądowo-napięciowej


- Kolejnym etapem będzie przeliczenie wartości napięcia na boczniku U_B (zmierzonego w kanale 0) na wartość prądu I_D płynącego przez badany dwójnik, według wzoru (3). Ze względu na niewielkie wartości prądu należy obliczyć go w miliamperach (mA). Następnie obliczamy moce tracone w boczniku i dwójniku. Moce obliczamy w miliwatach (mW). Przykładowy fragment tej części algorytmu przedstawiono na rysunku 28.



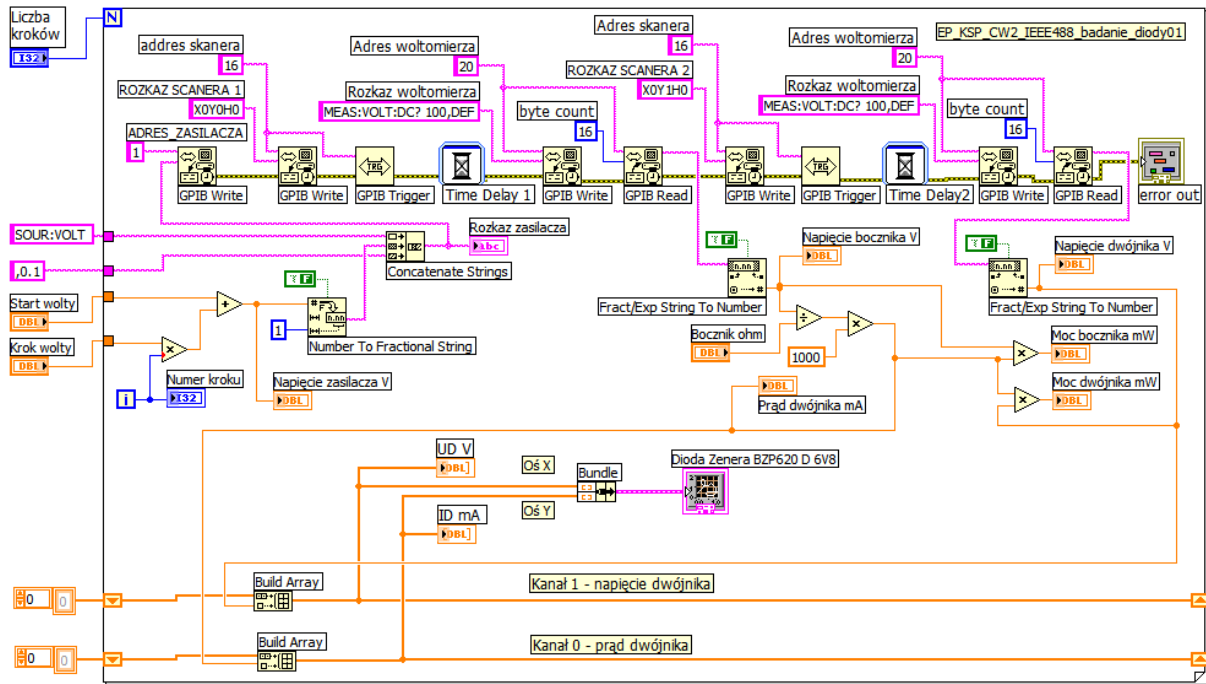
Rys. 28. Fragment diagramu realizującego obliczanie prądu płynącego przez dwójnik oraz mocy traconej w boczniku i dwójniku

- kolejnym etapem jest utworzenia tablic do przechowywania wartości napięcia U_D na dwójniku (oś X, pomiary z kanału 1) oraz prądu I_D płynącego przez dwójnik (oś Y, przeliczone pomiaru z kanału 0) i wykreślenie wykresu $I_D = f(U_D)$. Ten fragment programu jest praktycznie taki sam, jak w p. 4.6, należy tylko do wejść funkcji *Build Array* tworzących macierze dołączyć inne zmienne oraz zmienić opisy umieszczone na wykresie i skalowanie jego osi. Po wprowadzeniu wszystkich zmian Diagram programu powinien wyglądać jak na rysunku 29, a Panel jak na rysunku 30.

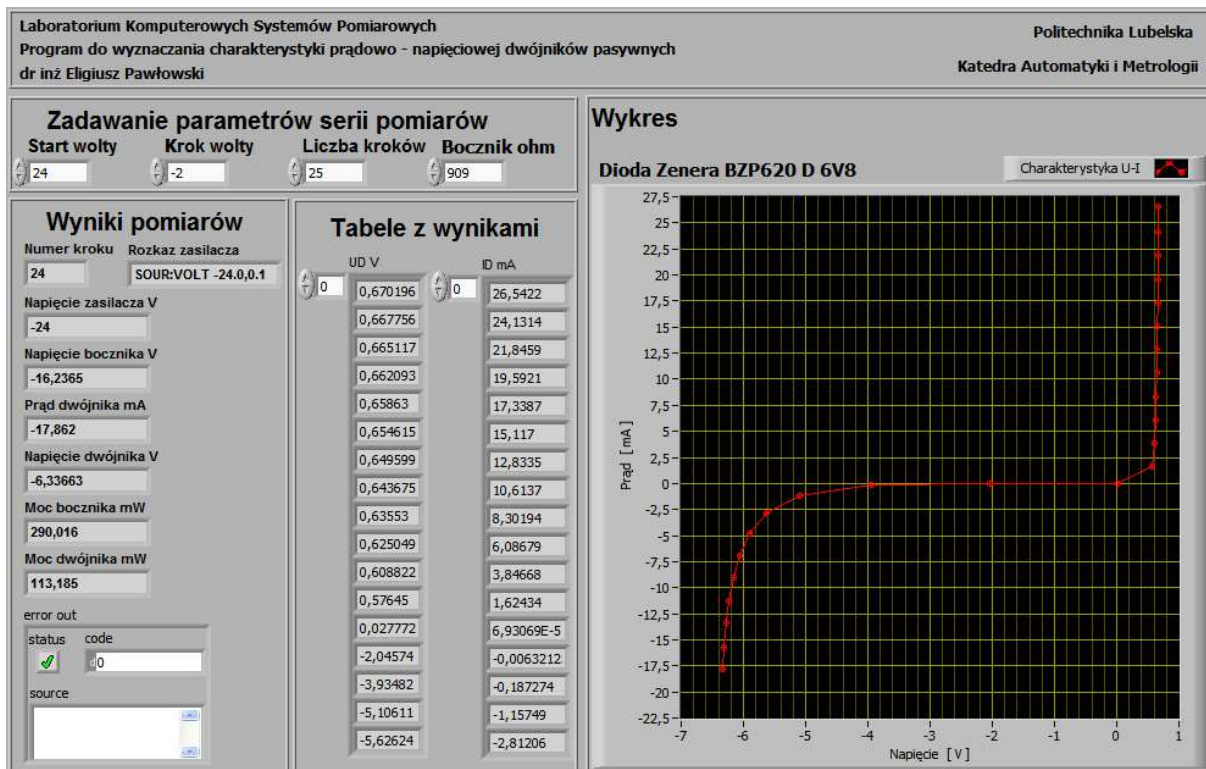
- Połączyć układ zgodnie ze schematem stanowiska (rys. 5).

- Na Panelu programu wprowadzić wartości początkowe: Start wolty -24, Krok wolty 2, liczba iteracji 25. Uruchomić program przyciskiem . Sprawdzić poprawność działania programu i ocenić zgodność otrzymanego wykresu z oczekiwaniami.

- Jeśli program działa poprawnie uzupełnić Panel oraz Diagram w dane osobowe, numer grupy, datę, temat ćwiczenia. Zapisać program na dysku jako wersję finalną dla punktu 4.7, zanotować jego nazwę do protokołu.



Rys. 29. Diagram programu do automatycznego badania diody pasywnej

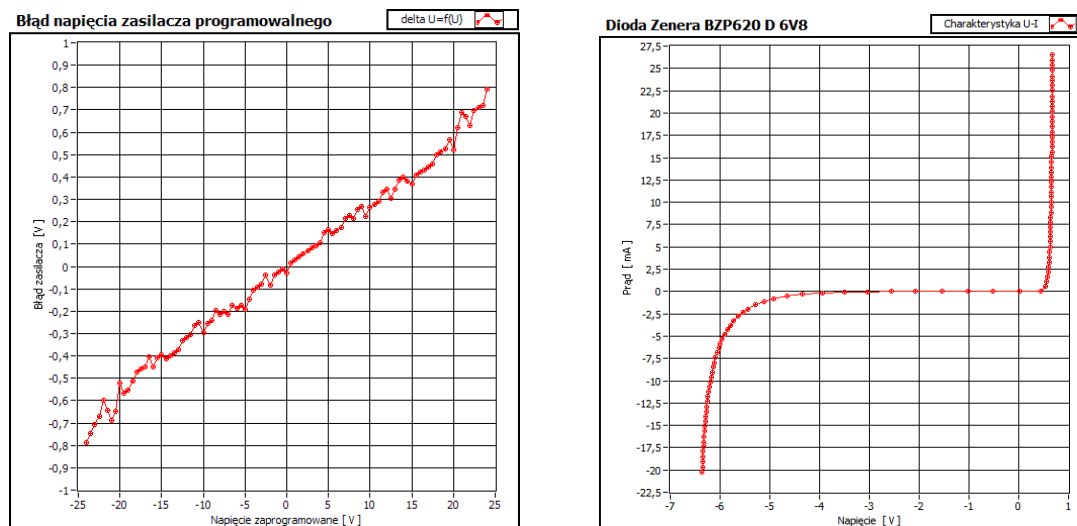


Rys. 30. Panel programu do automatycznego badania diody pasywnej

- Wydrukować (zapisać do pliku) dokumentację programu według punktu 3.3.
- Na Panelu programu wprowadzić wartości początkowe: Start wolty -24, Krok wolty 0,5, liczba iteracji 97. Uruchomić program przyciskiem . Podczas działania programu wykonać zrzut ekranu (klawisz PrtScr na klawiaturze) i korzystając z programu graficznego Paint

zapisać go na dysku w celu wykorzystania do sprawozdania. Uzyskany wykres zapisać do pliku graficznego wykorzystując opcję **Export Simplified Image** (według punktu 3.4) i umieścić w sprawozdaniu. Przykładowe wykresy przedstawiono na rysunku 31.

- Zapoznać się z kartą katalogową badanego dwójnika i ocenić zgodność uzyskanych parametrów i charakterystyk z danymi podanymi przez producenta. Wnioski zamieścić w sprawozdaniu.



Rys. 31. Przykładowe wykresy zapisane opcją **Export Simplified Image**

5. Sprawozdanie ze zrealizowanego ćwiczenia

Każda grupa przygotowuje z ćwiczenia jedno całościowe sprawozdanie zawierające:

- stronę tytułową z danymi osobowymi zespołu, datą wykonania ćwiczenia i jego tytułem,
- opis wykorzystywanego środowiska ćwiczeniowego z podaniem wykazu wykorzystywanej aparatury oraz zestawieniem jej parametrów,
- schemat wykorzystywanego układu pomiarowego,
- tematy kolejnych zadań przewidzianych programem ćwiczenia,
- opis słowny każdego ze zrealizowanych zadań lub podanie przyczyny niezrealizowania,
- dokumentację samodzielnie opracowanych i skutecznie uruchomionych programów (niezbędne są **opisy na panelu** oraz **komentarze na diagramie** programu !!!),
- schematy blokowe algorytmów dla zrealizowanych **samodzielnie programów**,
- słowny opis zrealizowanych **samodzielnie** algorytmów,
- wyniki pomiarów uzyskane za pomocą przygotowanych programów,
- wnioski ze zrealizowanych ćwiczeń, a w szczególności: napotkane trudności i przyjęte sposoby ich rozwiązania, propozycje zmian i rozszerzeń programu realizowanych zadań, propozycje dalszych modyfikacji i ulepszeń realizowanego zadania, propozycje nowych tematów zadań, propozycje modyfikacji i rozszerzeń układu pomiarowego.

W sprawozdaniu ocenie podlegają:

- Poprawność działania programu oraz jego odporność na nietypowe sygnały i złą obsługę.
- Czytelność i przejrzystość diagramu,
- Estetyka utworzonego Panelu, dobór kolorystyki, elementów graficznych, wyrównanie położenia obiektów, funkcjonalność panelu pod kątem obsługi,
- Zakres zrealizowanych zadań,
- Trafność wniosków końcowych.

Uwaga! Nie należy stosować systemowej funkcji Print Screen (przycisk PrtScr na klawiaturze) do zapisywania Panelu i Diagramu na dysku jako dokumentacji programu. Sprawozdania zawierające zamiast dokumentacji programu wydruk okna Windows z Panelem i Diagramem uzyskane klawiszem PrtScr będą oceniane słabiej. Można i należy natomiast wykonywać za pomocą funkcji PrtScr kopie Panelu w celu udokumentowania sposobu działania zrealizowanego programu. Do zapisywania wykresów uzyskanych podczas działania programów należy wykorzystywać opcję **Export Simplified Image**. Przykłady poprawnie zapisanych wykresów przedstawiono na rysunku 31.

6. Literatura

1. Nawrocki W.: Komputerowe systemy pomiarowe, WKiŁ, Warszawa 2002.
2. Winiecki W.: Organizacja komputerowych systemów pomiarowych, Oficyna Wydawnicza PW, Warszawa 1997.
3. Świsulski D.: Komputerowa technika pomiarowa Oprogramowanie wirtualnych przyrządów pomiarowych w LabView, Wyd. PAK, Warszawa 2005.
4. Świsulski D.: Laboratorium z systemów pomiarowych ; Wydawnictwo Politechniki Gdańskiej, 1998
5. Tłaczała W.: Środowisko LabVIEW w eksperymencie wspomaganym komputerowo, WNT Warszawa 2002.
6. Chruściel M.: LabVIEW w praktyce, Wyd. BTC, Warszawa 2008.
7. Mielczarek W.: Szeregowe interfejsy cyfrowe ; Helion, 1993
8. Mielczarek W.: Urządzenia pomiarowe i systemy kompatybilne ze standardem SCPI ; Helion, 1999
9. Mielczarek W.: Komputerowe systemy pomiarowe; Standardy IEEE488.2 i SCPI ; Wydawnictwo Politechniki Śląskiej, Gliwice 2002
10. Nowakowski W.: Systemy interfejsu w miernictwie ; WKiŁ, Warszawa 1987
11. Witold Kaczurba: [Kurs dla początkujących w Labview](http://www.kaczurba.pl), <http://www.kaczurba.pl>.
12. National Instruments: LabVIEW Tutorial Manual.