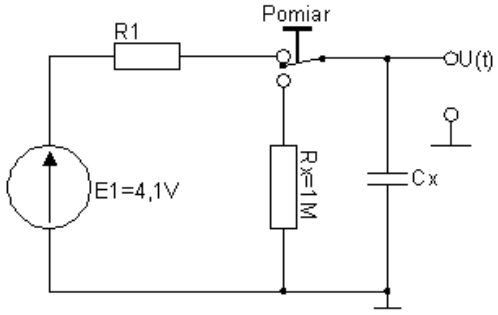
Ćwiczenie 6

WIZUALIZACJA PROCESU WYZNACZANIA SKŁADOWYCH IMPEDANCJI *LC* METODĄ DYNAMICZNĄ Z WYKORZYSTANIEM ŚRODOWISKA *LabVIEW*

6.1. Wprowadzenie

Istotę metody dynamicznej wyznaczania składowych impedancji, na przykładzie pomiaru pojemności C_x, objaśnia rys. 6.1 [3]. W pierwszej fazie badany kondensator C_x jest ładowany do napięcia $U_m=E1$ a następnie, po załączeniu przycisku „POMIAR", rozładowywany w obwodzie R_xC_x. Kondensator rozładowuje się przez rezystancję wzorcową $R_x=1\ M\Omega$. System pomiarowy na bazie karty *NI USB 6008* mierzy czas rozładowania kondensatora C_x do momentu, w którym napięcie na nim osiągnie wartość $u_{Cx}(t=T)=U_m/e$. W tym momencie zmienia swój stan komparator, na którym ustawiono taką wartość napięcia odniesienia oraz zatrzymuje się pomiar czasu. Stała czasowa obwodu wynosi $T=R_x\cdot C_x$.



Rys. 6.1. Objaśnienie dynamicznej metody wyznaczania składowych impedancji na przykładzie pomiaru pojemności C_x [3]

Przyjęto następujące oznaczenia:
t – czas,
T – stała czasowa obwodu $T=R_x\cdot C_x$,
e – podstawa logarytmu naturalnego.

Uwzględniając, że kondensator C_x rozładowuje się według funkcji:

$$u_{Cx}(t) = U_m \exp\left(-\frac{t}{R_x C_x}\right), \tag{1}$$

po logarytmowaniu można napisać:

$$\ln[u_{Cx}(t)] = -\frac{1}{R_x C_x} t + \ln U_m . \tag{2}$$

Równanie (2) odpowiada równaniu prostej:

$$y = ax + b , \tag{3}$$

gdzie:

$$y = \ln[u_{Cx}(t)], \qquad a = -\frac{1}{R_x C_x}, \quad x=t, \qquad b = \ln U_m . \tag{4}$$

System pomiarowy pracuje dyskretnie, dlatego napięcie na kondensatorze $u_{Cx}(t_i)$ jest mierzone w dokładnie określonym czasie t_i. Na podstawie pomiarów otrzymuje się tablicę danych [t_i,

$u_{Cx}(t_i)$]. Dla przypadku ujemnej wartości „współczynnika kierunkowego” *a* prostej, z równań (3, 4) można wyprowadzić następującą zależność na mierzoną pojemność C_x [3, 4]:

$$C_x = \frac{\left(\sum_{i=1}^{n} t_i\right)^2 - n\sum_{i=1}^{n} t_i^2}{R_x\left\{n\sum_{i=1}^{n} t_i \ln[u_{Cx}(t_i)] - \left(\sum_{i=1}^{n} t_i\right)\left(\sum_{i=1}^{n} \ln[u_{Cx}(t_i)]\right)\right\}}. \tag{5}$$

Wzór (5) wynika z liniowej regresji danych dla minimum błędu średniokwadratowego. Analogicznie można wyprowadzić współczynnik korelacji wzajemnej r^2 [3]:

$$r^2 = \frac{\left\{\sum_{i=1}^{n} t_i \ln[u_{Cx}(t_i)] - \frac{\sum_{i=1}^{n} t_i}{n}\sum_{i=1}^{n} \ln[u_{Cx}(t_i)]\right\}^2}{\left[\sum_{i=1}^{n} t_i^2 - \frac{\left(\sum_{i=1}^{n} t_i\right)^2}{n}\right]\cdot\left\{\sum_{i=1}^{n} \ln[u_{Cx}(t_i)]^2 - \frac{\left(\sum_{i=1}^{n} \ln[u_{Cx}(t_i)]\right)^2}{n}\right\}}. \tag{6}$$

Współczynnik ten wskazuje, czy istnieje zależność między elementami tablicy danych [t_i, $u_{Cx}(t_i)$]. Oczywiście powinien przyjmować wartości bliskie jedności.

Dla zarejestrowanych dwóch próbek (n=2), wzór (5) przyjmuje postać:

$$C_x = \frac{t_2 - t_1}{R_x\{\ln[u_{Cx}(t_1)] - \ln[u_{Cx}(t_2)]\}}. \tag{7}$$

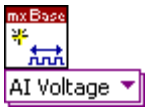
6.2. Opis oprogramowania stworzonego w środowisku *LabVIEW*

6.2.1. Uwagi wstępne

Środowisko *LabVIEW* jest graficznym językiem programowania zaprojektowanym po to by ułatwić tworzenie wirtualnych przyrządów pomiarowych (*VI’s - Virtual Instruments*). W przeciwieństwie do tradycyjnych języków programowania w *LabVIEW* używa się ikon zamiast tekstu. Biegiem wykonywania programu kieruje nie kolejność występowania instrukcji, ale przepływ strumienia danych. Oznacza to, że dany bloczek (instrukcja w formie graficznej) zostanie wykonany dopiero wówczas, gdy wszystkie dane na wejściu będą kompletne. Aplikacja zaprojektowana za pomocą *LabVIEW* składa się z dwóch części. Są to: okno edycyjne *Panelu* (Front Panel), gdzie buduje się interfejs użytkownika oraz okno edycyjne *Diagramu* (Block Diagram) w polu którego buduje się sieć działań. Ważną cechą *LabVIEW* jest to, iż firma „National Instruments” dostarcza wraz ze swoimi urządzeniami do akwizycji danych (*DAQ’s - Data AcQuisition devices*) wsparcie projektowe dla środowiska *LabVIEW*, co czyni prace nad aplikacją znacznie bardziej przyjazną dla użytkownika [1, 5, 6, 7].

6.2.2. Opis instrukcji *DAQmxBASE*

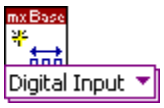
- *DAQmxBase Create Task*. Jest to instrukcja tworząca nowe zadanie. Dostarcza ona „*Task ID*” dla pozostałych bloczków w programie. Większość parametrów wejściowych tej funkcji jest ignorowana. W aplikacji użyta jest jako instrukcja bez parametrów.



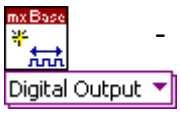
- *DAQmxBase Create Virtual Channel*. Tworzy kanał wirtualny i dodaje go do zadania. Aby użyć tego *VI* należy najpierw wykonać *DAQmxBase Create Task*. W powyższej wersji utworzony zostanie kanał wejścia analogowego. Do wejścia *VI* należy doprowadzić następujące dane:
  - *Physical channels*: nazwa kanału fizycznego w celu skojarzenia go z utworzonym kanałem wirtualnym. W zaprezentowanym rozwiązaniu skorzystano z pierwszego kanału pomiarowego Dev1/ai0 .
  - *Maximum value*: maksymalna wartość (domyślnie napięcia), jaką spodziewamy się mierzyć.
  - *Minimum value*: minimalna wartość (domyślnie napięcia), jaką spodziewamy się mierzyć.
  - *Input terminal configuration*: w tym miejscu należy wybrać tryb, w jakim ma pracować urządzenie *DAQ*. Są dwa tryby pracy: RSE (Reference Single-Ended - sygnał doprowadzamy między masę a pin „+AI0") - wówczas mamy do dyspozycji osiem analogowych kanałów pomiarowych oraz tryb DIFFERENTIAL (sygnał pomiarowy doprowadzamy między piny „+AI0" a „-AI0") - liczba kanałów pomiarowych spada do czterech, natomiast rozdzielczość przetwornika A/C wzrasta do 12 bitów.
  - *Task In*: do tego wejścia doprowadzamy „*Task ID*" uzyskany np. za pomocą *VI DAQmxBase Create Task.*

Możliwe są także inne opcje pracy tego VI:

- Wyjście analogowe.

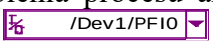


- Wejście cyfrowe.



- Wyjście cyfrowe. Jako parametry wejściowe dla tego *VI* należy doprowadzić:
  - *Lines* - nazwa linii cyfrowej lub portu w celu skojarzenia go z utworzonym kanałem wirtualnym Dev1/port0.
  - *Line grouping* - parametr określa czy pogrupować linie portu w jeden wirtualny kanał czy dla każdej linii portu stworzyć oddzielny. W aplikacji zastosowano pierwszy sposób - jeden kanał dla wszystkich linii one channel for all lines

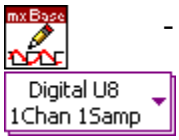
- *DAQmxBase Timing*. Za pomocą tego *VI* ustawia się częstotliwość próbkowania, ilość zebranych próbek a także dokonuje wyboru czy nabywanie danych ma się odbywać ciągle, czy zakończyć po jednym cyklu. Jako dane wejściowe należy doprowadzić:
  - *Rate* - częstotliwość próbkowania ustawiana w próbkach na kanał na sekundę.
  - *Source* - źródło sygnału zegarowego (niepołączone oznacza, że wykorzystywany jest wewnętrzny zegar karty).
  - *Active edge* - pole wyboru: czy próbki mają być nabywane przy rosnącym (domyślnie) czy przy opadającym zboczu sygnału zegarowego.
  - *Sample mode* - określa, w jaki sposób urządzenie nabywa lub generuje próbki. W opisywanej aplikacji użyto trybu Finite Samples, co oznacza, że nabyta zostanie określona ilość próbek.
  - *Samples per channel* - określa ilość próbek do nabycia lub wygenerowania (pod warunkiem, że pole *Sample mode* zostanie ustawie w tryb *Finite Samples*).

- *DAQmxBase Trigger*. Ten *VI* konfiguruje urządzenie *DAQ* tak, aby zacząć nabywać próbki wówczas, gdy na odpowiedni pin karty przyjdzie narastające lub opadające (niepopierane dla *NI USB 6008*) zbocze sygnału cyfrowego. Jako parametry należy podać:
  - *Source* - określa gdzie znajduje się źródło sygnału cyfrowego, które zostanie użyte do wyzwolenia procesu akwizycji danych. W opisywanej aplikacji podłączono tam 

.

- *DAQmxBase Start Task*. Przejście zadania w tryb wykonania. Ten *VI* jest wymagany dla wszystkich aplikacji *DAQmxBASE*. Umieszcza się go po dokonaniu pełnej konfiguracji kanałów.

- *DAQmxBase Read*. Ten *VI* czyta próbki z wcześniej utworzonego zadania. W zaprezentowanej wersji, na wyjściu otrzymuje się tablicę danych, np. z wejść analogowych. Jako dane wejściowe należy podać:
  - *Number of samples per channel* - określa ile próbek należy odczytać. Jeśli zostawia się niepołączone, a przy pomocy *VI DAQmxBase Timing* zadanie zostało ustawione na nabywanie skończonej liczby próbek, wówczas *VI* czeka aż zadanie zgromadzi wszystkie żądane próbki i dopiero je przeczyta.
  - *Timeout* - określa czas, jaki *VI* ma czekać aż zażądane próbki staną się dostępne. Jeśli czas mija, *VI* zwraca błąd wraz z próbkami, które zdołał odczytać.



- *DAQmxBase Write*. Ten *VI* wysyła do zadania, które jest skojarzone z wyjściem cyfrowym, pojedynczą 8 bitową zmienną typu całkowitego. Inaczej mówiąc tego *VI* używa się, aby wysłać dane od razu na wszystkie linie portu. Przykładowo, aby uaktywnić pierwszy pin portu, należy przypisać polu data wartość 1; aby uaktywnić wszystkie linie portu, pole data powinno przyjąć wartość 255 (255 Dec =11111111 Bin).

- *DAQmxBase Stop Task*. Ten *VI* zatrzymuje zadanie i zwraca je do stanu, w jakim było przed wykonaniem *DAQmxBase Start Task*. Użycie tego *VI* jest wymagane dla każdej aplikacji *DAQmxBase*. Jeśli w pętli zostaje użyty *DAQmxBase Start Task* w tej samej pętli musi zostać użyty *DAQmxBase Stop Task*, choć ciągłe używanie tych *VI's* w aplikacji zmniejsza efektywność programu.

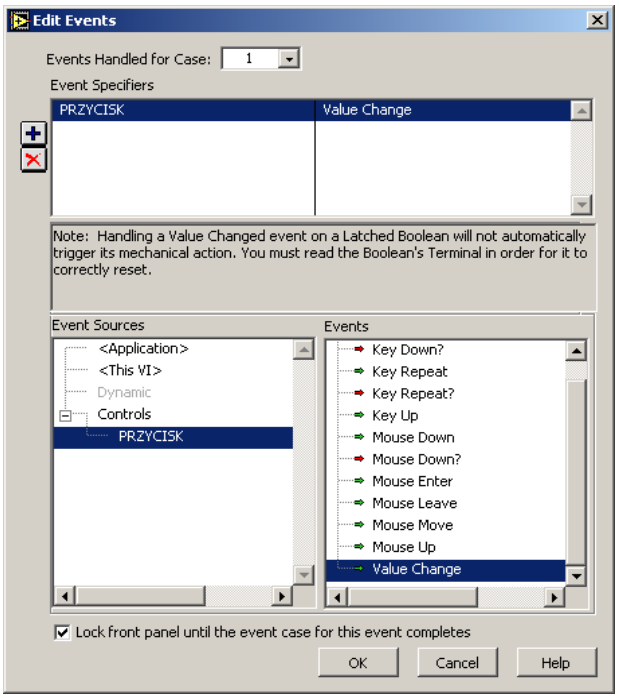
- *DAQmxBase Clear Task*. Ten *VI* czyści zadanie, zwalnia, zatrzymuje je (jeśli to jest konieczne) oraz zwalnia zasoby pamięci zarezerwowane przez zadanie. Po wykonaniu instrukcji nie można ponownie użyć zadania. Jeśli w pętli występuje *DAQmxBase Create Task* lub *DAQmxBase Create Virtual Channel* w tej samej pętli musi wystąpić *DAQmxBase Clear Task.*

6.2.3. Event Structure

Event Structure zawiera jeden lub więcej diagramów, z których jeśli wykonuje się struktura, wykonuje się dokładnie jeden diagram - rys. 6.2. *Event Structure* czeka, aż na głównym pulpicie wystąpi zaprogramowane zdarzenie, aby następnie wykonując odpowiednią ramkę obsłużyć dane zdarzenie. Jeśli w ciągu czasu, który jest podłączony do klepsydry w lewym górnym rogu struktury, żadne z zaprogramowanych zdarzeń nie wystąpi, wówczas wykonuje się ramka *Timeout.* Aby wyłączyć wykonywanie się tej ramki i po to by wydłużyć czas oczekiwania struktury na zdarzenie w nieskończoność, należy podłączyć do terminalu w lewym górnym rogu struktury wartość (-1). Aby stworzyć procedurę, która wykona się po wciśnięciu określonego przycisku, należy [5]:

- umieścić przycisk na panelu użytkownika,

- w obrębie struktury *Event* kliknąć prawym przyciskiem myszy i wybrać „*Add Event Case…*” oraz postępować zgodne z rys. 6.2, a utworzona zostanie ramka wewnątrz której będzie można oprogramować wybrane zdarzenie.

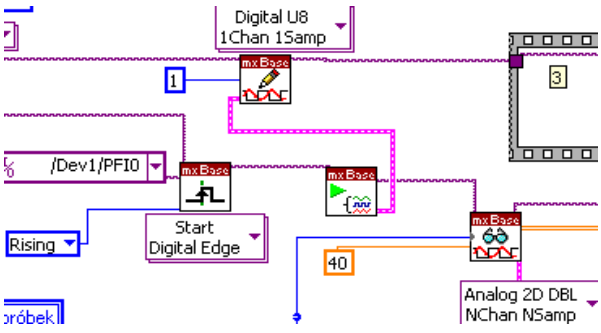


Rys. 6.2. Sposób tworzenia nowej ramki w *Event Structure*

6.2.4. Kontrolowanie biegu programu w *LabVIEW*

Jak wspomniano w p. 6.2.1, w *LabVIEW* kolejność wykonywania poszczególnych instrukcji, zależy od przepływu danych pomiędzy kolejnymi *VI*. W momencie, gdy do określonej instrukcji dostarczone zostaną wszystkie dane wejściowe będzie ona wykonana. Problem powstaje wówczas, gdy nie można przewidzieć przepływu danych, a należy utrzymać kolejność wykonywania się poszczególnych operacji. Można wtedy wykorzystać „*Flat Sequence Structure*”. Kolejne ramki tej struktury będą wykonywane po sobie.

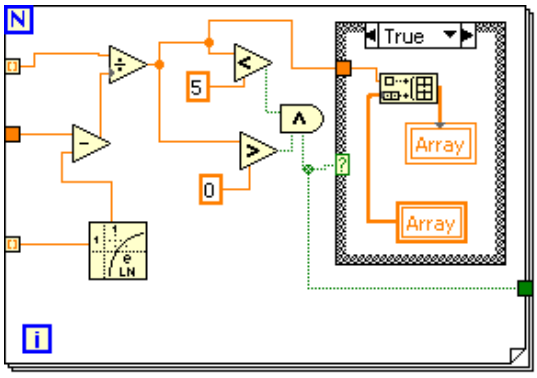
Jednak czasami można uniknąć konieczności uciekania się do stosowania wyżej wymienionej struktury. Mamy wtedy do czynienia z dwoma niezależnymi zadaniami, które należy zsynchronizować. Ważne jest, aby najpierw uruchomić zadanie akwizycji danych, a następnie wystawić sygnał logicznej jedynki na Pin 1 Port 0 karty. Synchronizację osiąga się za pomocą linii błędu, czyli za pomocą przepływu danych - rys. 6.3. Jest to sposób najbardziej zalecany podczas programowania w *LabVIEW*.



Rys. 6.3. Kierowanie biegiem programu z wykorzystaniem linii błędu

6.2.5. Zmienne lokalne

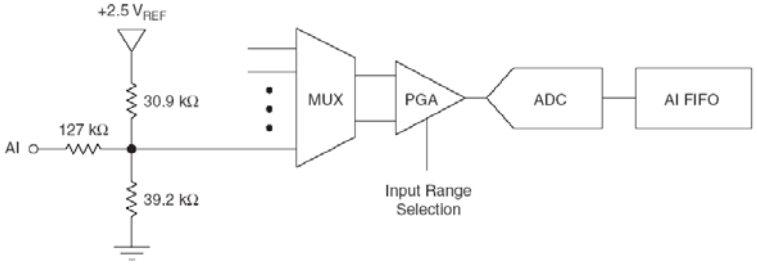
Zmienne lokalne są wygodnym sposobem na wymianę danych pomiędzy poszczególnymi pętlami w programie, jak i poszczególnymi ramkami *Event Structure*. Zmienna lokalna może reprezentować dowolną zmienną w programie i ma zasięg zarówno na program, jak i na podprogramy. W przykładzie pokazanym na rys. 6.4 przedstawiono, jak za pomocą zmiennej lokalnej można rozszerzyć tablicę o kolejny element. Rozszerzenie tablicy następuje dopiero wówczas, gdy element spełnia określone kryteria. Zrealizowanie takiego pomysłu bez użycia zmiennych lokalnych byłoby bardzo kłopotliwe.



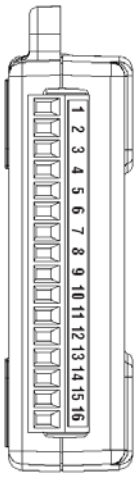
Rys. 6.4. Dodawanie nowego elementu do tablicy z przykładu
z wykorzystaniem zmiennych lokalnych

6.3. Charakterystyka karty pomiarowej *NI USB 6008*

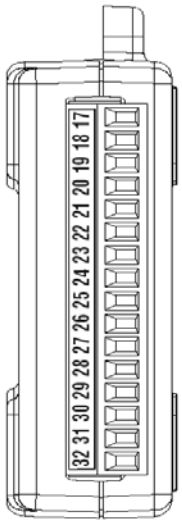
Zastosowana karta pomiarowa posiada cztery wejścia analogowe typu różnicowego (*Differential*), dwa wyjścia analogowe, jeden pełny port cyfrowy P0.<0..7> oraz drugi port okrojony P1.<0..3>. Karta posiada 12 bitowy przetwornik A/C oraz możliwość wyzwalania pomiaru zewnętrznym sygnałem cyfrowym. Maksymalna częstotliwość próbkowania wynosi 10 [kS/s] (kS = kiloSample, ang. sample – próbka). O wyborze karty zadecydowała możliwość zewnętrznego wywołania pomiaru. Cecha ta zyskuje na znaczeniu wówczas, gdy mamy do czynienia z szybkimi pomiarami i programowe wyzwolenie karty może okazać się nieskuteczne. Wadą karty jest to, iż wszystkie piny AI mają podniesiony potencjał do około 1,4 [V] - rys. 6.5 [1]. Rozwiązanie takie uniemożliwia bezpośredni pomiar napięcia na rozładowującym się kondensatorze. W celu odizolowania układu pomiarowego - rys. 6.8 od obwodów wejściowych karty, konieczne stało się zastosowanie wzmacniacza operacyjnego o wzmocnieniu równym 1. Na rys. 6.6 przedstawiono opis wyprowadzeń terminala analogowego karty, a na rys. 6.7 - cyfrowego [1].



Rys. 6.5. Schemat analogowego obwodu wejściowego karty pomiarowej (MUX - multiplekser, PGA - programowalny wzmacniacz, ADC - przetwornik analogowo-cyfrowy)

Module	Terminal	Signal, Single-Ended Mode	Signal, Differential Mode
	1	GND	GND
	2	AI 0	AI 0+
	3	AI 4	AI 0–
	4	GND	GND
	5	AI 1	AI 1+
	6	AI 5	AI 1–
	7	GND	GND
	8	AI 2	AI 2+
	9	AI 6	AI 2–
	10	GND	GND
	11	AI 3	AI 3+
	12	AI 7	AI 3–
	13	GND	GND
	14	AO 0	AO 0
	15	AO 1	AO 1
	16	GND	GND

Rys. 6.6. Opis wyprowadzeń terminala analogowego karty [1]

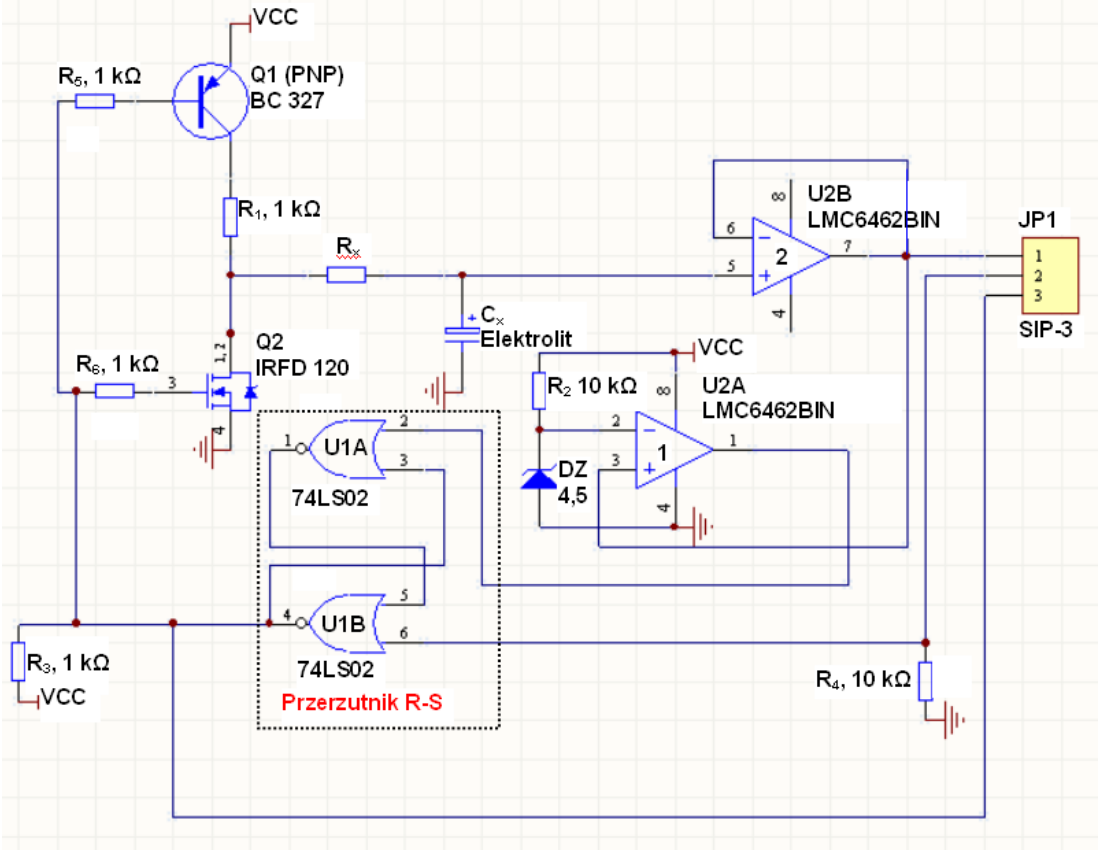
Module	Terminal	Signal
	17	P0.0
	18	P0.1
	19	P0.2
	20	P0.3
	21	P0.4
	22	P0.5
	23	P0 6
	24	P0.7
	25	P1.0
	26	P1.1
	27	P1.2
	28	P1.3
	29	PFI 0
	30	+2.5 V
	31	+5 V
	32	GND

Rys. 6.7. Opis wyprowadzeń terminala cyfrowego karty [1]

6.4. Opis układu pomiarowego

Pomiar rozpoczyna się w momencie naciśnięcia przycisku „Pomiar” - rys. 6.1. Wówczas na jeden z pinów karty zostanie wystawiony sygnał wysoki i rozpocznie się proces ładowania kondensatora C_x. Gdy napięcie na kondensatorze C_x przekroczy napięcie na diodzie Zenera *DZ* (około 4 [V]), wówczas komparator uruchomi proces jego rozładowywania oraz jednocześnie poda sygnał do karty w celu rozpoczęcia rejestracji charakterystyki rozładowania się kondensatora. W tym czasie, w programie napisanym w *LabVIEW*, zapisywana jest

tablica z danymi - kolejnymi próbkami napięcia: $[t_i, u_{Cx}(t_i)]$, które wykorzystane zostaną do wyznaczenia stałej czasowej obwodu rozładowania się kondensatora - p. 6.1. Schemat układu pomiarowego zaprojektowanego do współpracy z kartą pomiarową przedstawiono na rys. 6.8. Opis wyprowadzeń jest następujący [2]:



Rys. 6.8. Schemat układu pomiarowego [2]

- Pin1 - sygnał pomiarowy, napięcie na rozładowującym się kondensatorze (Wyj).
- Pin2 - wyzwalanie procedury pomiarowej (Wej).
- Pin3 - sygnalizacja rozpoczęcia rozładowywania kondensatora (Wyj).

W układzie z rys. 6.8 zastosowano następujące elementy:

U1A, U1B - dwie bramki NOR połączone w układ przerzutnika R-S. Zadaniem przerzutnika jest sterowanie tranzystorami *Q1*, *Q2* oraz wygenerowanie dla karty (Pin3) sygnału rozpoczęcia rejestracji próbek $[t_i, u_{Cx}(t_i)]$ rozładowania kondensatora.

Q1 - tranzystor ładujący kondensator C_x. Zastosowano tranzystor BC327.

Q2 - tranzystor rozładowujący kondensator C_x. Zastosowano tranzystor *IRFD120* typu CMOS o małej rezystancji kanału, wynoszącej zgodnie z danymi katalogowymi 0,25 [Ω], co umożliwia prawie całkowite rozładowanie kondensatora.

U2A - wzmacniacz operacyjny pracujący w układzie komparatora. Wykrywa moment naładowania się kondensatora i daje sygnał do przerzutnika RS w celu rozpoczęcia procedury pomiarowej, tj. rejestracji próbek $[t_i, u_{Cx}(t_i)]$ rozładowania kondensatora. Zastosowano wzmacniacz *LMC6462BIN* firmy National Semiconductor.

U2B - wzmacniacz operacyjny o wzmocnieniu równym 1. Pełni rolę bufora pomiędzy układem pomiarowym, a kartą. Konieczność jego zastosowania wynika ze wspomnianej wcześniej (p. 6.3) budowy układu wejściowego karty *NI USB 6008*. Zastosowano wzmacniacz *LMC6462BIN* firmy National Semiconductor o bardzo dobrych parametrach. Charakteryzuje się on dużą rezystancją wejściową, bardzo niskim napięciem

niezrównoważenia oraz dużym wzmocnieniem, dlatego nie obciąża obwodu pomiarowego oraz wiernie odtwarza mierzone napięcie.

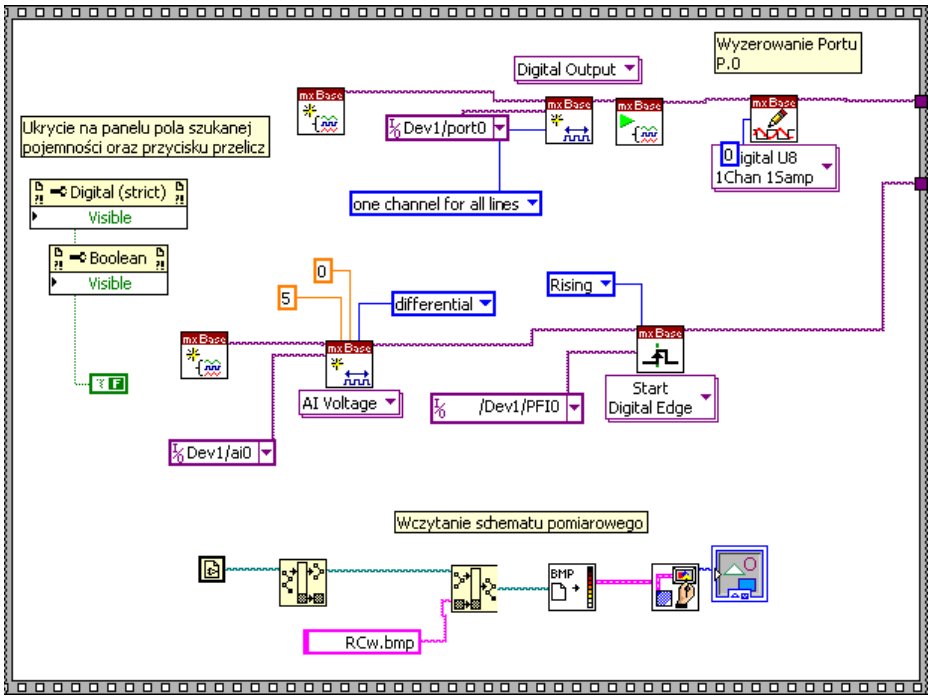
R_x, C_x - obwód pomiarowy.

W celu zminimalizowania zakłóceń, układ elektroniczny z kartą pomiarową połączono przewodem ekranowanym [8].

6.5. Opis oprogramowania do obsługi karty pomiarowej.

Do obsługi wyżej zaproponowanego układu wykorzystano program napisany w graficznym środowisku programowania *LabVIEW*. W związku z faktem, że pakiet *LabVIEW* jest tego samego producenta, co karta pomiarowa, firma oferuje wraz z kartą wsparcie projektowe dla tego środowiska programistycznego. Zaproponowany program swoje działanie opiera na *Event Structure*, co znacznie poprawia jego funkcjonalność. Dzięki temu możliwe stało się programowanie poszczególnych zdarzeń takich jak wciśnięcie przycisku oraz wielokrotne ich wywoływanie np. ze zmienionymi parametrami.

W pierwszej części programu następuje inicjacja karty, budowa panelu użytkownika oraz wczytanie schematu pomiarowego zapisanego w formacie *.jpg. Inicjacja karty - rys. 6.9 polega między innymi na wyzerowaniu portu P0 karty pomiarowej. Następna logiczna 1 wystawiona na port P0.1 (Port 0 Pin1) powoduje rozpoczęcie procedury pomiarowej.

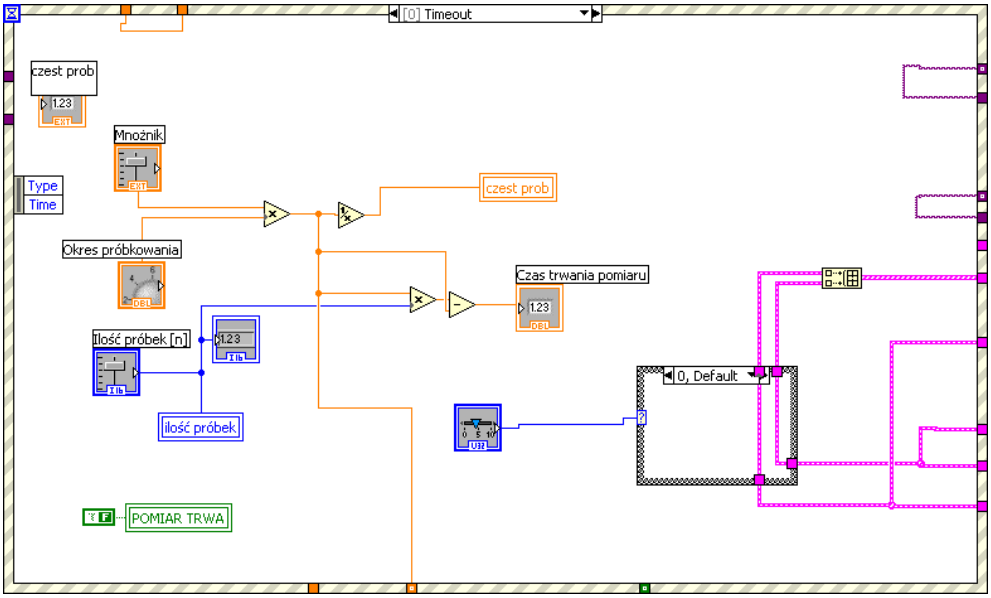


Rys. 6.9. Inicjacja karty pomiarowej

Kolejnym etapem programu jest *Event Structure* - rys. 6.10÷6.14, zawarta w pętli „*while loop*”, w której zaprogramowano następujące zdarzenia:

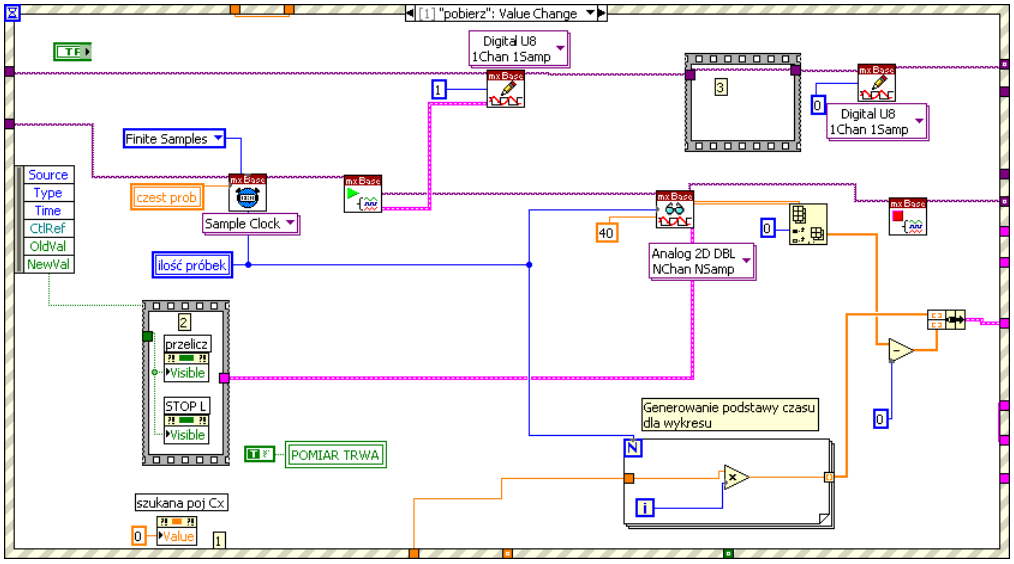
- *Timeout* - wykonywane w niekończącej się pętli - rys. 6.10
- „POMIAR”: *value change* - wykonane po wciśnięciu przycisku „POMIAR” - rys. 6.11
- „PRZELICZ”: *value change* - wykonane po wciśnięciu przycisku „PRZELICZ” - rys. 6.12
- „TEST”: *value change* - wykonane po wciśnięciu przycisku „TEST” - rys. 6.13
- „STOP”: *value change* - wykonane po wciśnięciu przycisku „STOP” - rys. 6.14.

Timeout - wykonuje się wówczas, gdy żadne z zaprogramowanych zdarzeń nie jest wywołane. W tym miejscu następuje ciągły odczyt parametrów próbkowania oraz wybór wykresu, który chcemy obserwować - rys. 6.10.



Rys. 6.10. Diagram części *Timeout Event Structure*

Najważniejszą częścią aplikacji jest sekcja wykonywana po naciśnięciu przycisku „POMIAR” - rys. 6.11. W tej części obsługiwana jest cała procedura akwizycji danych. Algorytm programu jest następujący:



Rys. 6.11. Diagram części „POMIAR”: *value change Event Structure*

- ustawienie częstotliwości oraz ilości zebranych próbek,
- ustawienie pracy karty w tryb umożliwiający wyzwolenie akwizycji danych zboczem narastającym sygnału cyfrowego - zbocze opadające jest niepopierane dla *NI USB 6008*,
- ustawienie portu P0.1 (Port 0 Pin1) karty pomiarowej w stan wysoki - rozpocznie się procedura pomiarowa,

- wykonanie zadania z zadanymi parametrami,
- odczyt danych z bufora w celu ich dalszej obróbki.

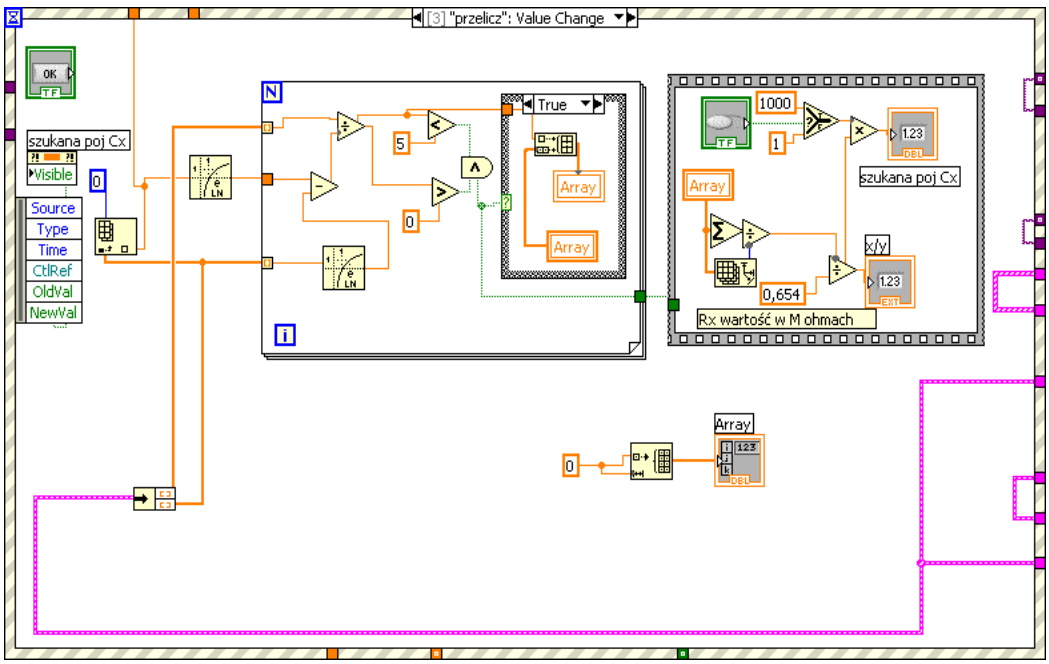
Z danych zebranych podczas procesu pomiaru należy wyznaczyć stałą czasową $T=R_xC_x$. Odbywa się to poprzez przekształcenie wzoru (7) do postaci (7'):

$$T = \frac{t}{\ln U_m - \ln[u_{Cx}(t)]}. \qquad (7')$$

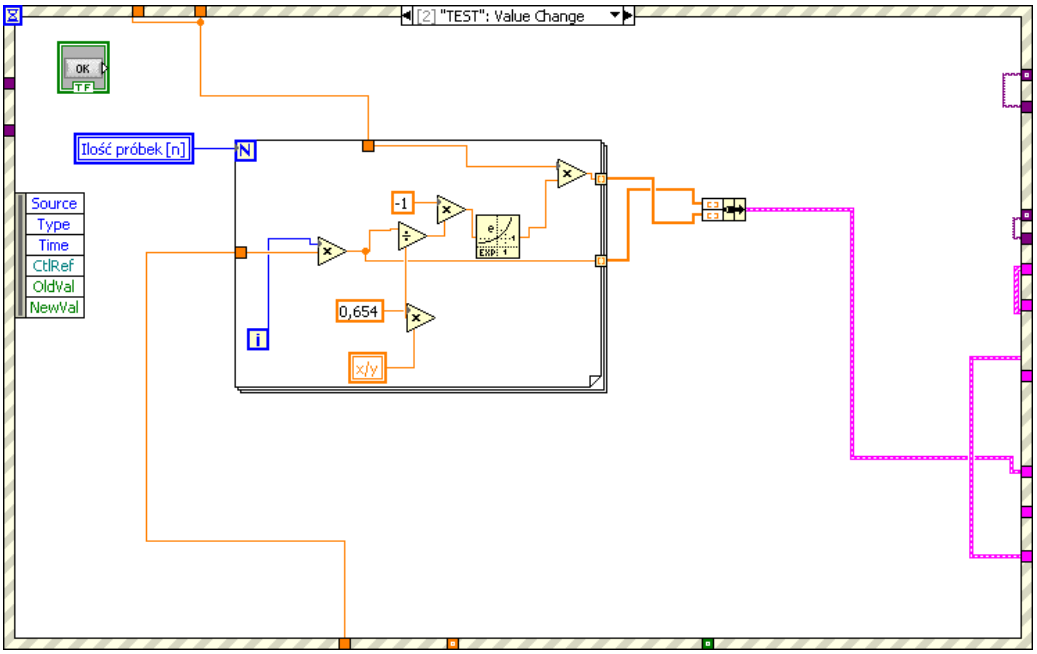
Wzór (7') przedstawia sposób wyznaczenia stałej czasowej obwodu R_xC_x za pomocą pojedynczej próbki.

W diagramie „PRZELICZ" - rys. 6.12 zaimplementowano również pomijanie błędów grubych. Jeśli wyliczona stała czasowa obwodu R_xC_x nie mieści się w zakresie od zera do pięciu sekund, wynik jest pomijany przy budowie tablicy do uśredniania.

Kolejną częścią programu jest sekcja „TEST" - rys. 6.13. Na podstawie wyliczonej pojemności generuje się charakterystykę rozładowania kondensatora, którą można oglądać na jednym wykresie. Wyboru, którą z charakterystyk przedstawić na wykresie, dokonuje się suwakiem na panelu użytkownika.

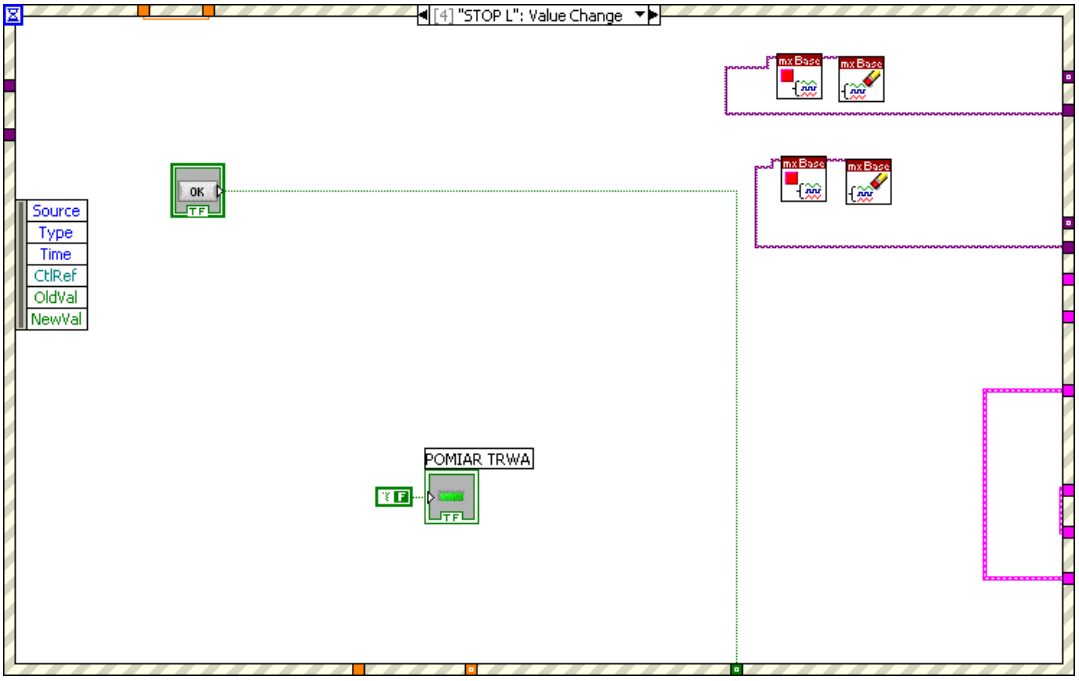


Rys. 6.12 Diagram części „PRZELICZ"*: value change Event Structure*

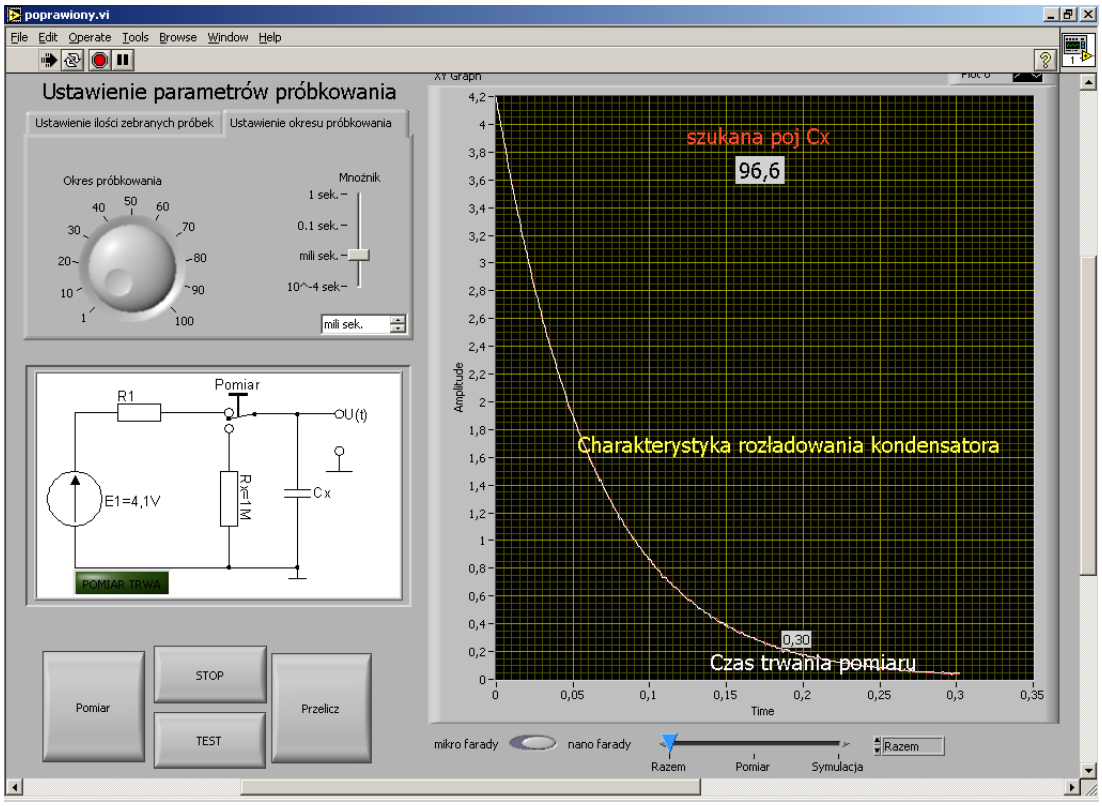


Rys. 6.13. Diagram części „TEST”*: value change Event Structure*

W sekcji „STOP”*: value change* - rys. 6.14, wykonywana jest procedura zwalniania zasobów karty pomiarowej oraz wystawiony jest sygnał zakończenia do pętli głównej, kończący wykonywanie programu.



Rys. 6.14. Diagram części „STOP”*: value change Event Structure*



Rys. 6.15. Wygląd panelu użytkownika programu napisanego w *LabVIEW*

Panel użytkownika – rys. 6.15 zawiera:

- Pole wykresu, na którym jest wyświetlony przebieg czasowy napięcia na rozładowywanym kondensatorze.
- Suwaki umożliwiające zadanie parametrów próbkowania (okres próbkowania, ilość zebranych próbek).
- Diodę, która sygnalizuje czas trwania pomiaru - pomiar widoczny na rys. 6.15 trwał 0.3 sek.
- Przyciski, z których: „POMIAR” - inicjuje procedurę pomiarową, „STOP” - kończy wykonywanie programu, „PRZELICZ” - wylicza szukaną pojemność C_x na podstawie charakterystyki rozładowania, „TEST” - na podstawie wyliczonej pojemności symuluje charakterystykę rozładowania kondensatora (charakterystyka ta na rys. 6.15 widoczna jest w kolorze czerwonym).
- Suwak wyboru (u dołu) określa, która z charakterystyk ma zostać przedstawiona na wykresie. Program wyposażono w opcję, która na podstawie wyznaczonej pojemności C_x generuje charakterystykę rozładowywania kondensatora, a następnie wyświetla ją na tym samym wykresie, co charakterystyka uzyskana z pomiaru. Daje to możliwość porównania przebiegu obu charakterystyk i wizualnej oceny jakości aproksymacji charakterystyk uzyskanych z pomiaru i symulacji..

W części wizualizacyjnej ćwiczenia, charakterystyka rozładowania kondensatora jest generowana na dwa sposoby: na podstawie algebraicznego rozwiązania równania różniczkowego obwodu oraz poprzez dyskretyzację równania różniczkowego obwodu. Sposób pierwszy jest bardziej dokładny i pozwala wyciągnąć wniosek, że dla zastosowanej karty pomiarowej *NI USB 6008* zarówno ilość próbek, jak i okres próbkowania nie mają znaczącego wpływu (innego niż statystyczny) na dokładność pomiaru pojemności C_x. Sprawdzenie tego stwierdzenia będzie tematem jednego z punktów programu ćwiczenia - p. 6.6.

Dodatkowo, w analogiczny sposób, w ćwiczeniu można badać charakterystykę rozładowania cewki i mierzyć jej wartość L_x.

6.6. Wykonanie ćwiczenia

1. Wykonać dwie serie pomiarów dla kondensatorów elektrolitycznych o pojemności znamionowej C_x= 2,2 µF oraz C_x= 100 nF (tj. podanej przez producenta na obudowie), dostarczonych przez Prowadzącego, dla różnych czasów t pomiaru oraz różnej ilości n próbek. Zapewnić, aby czas pomiaru oraz ilość zebranych próbek zmieniały się w szerokich granicach. Pojemności te, zmierzone miernikiem cyfrowym ALDA M89OG klasy 0,5, wynosiły odpowiednio: 1,89 µF oraz 94,1 nF.
2. Dla dwóch badanych kondensatorów, wykonać odnośne wykresy błędów $\delta_{Cx}=f(t)$ dla różnych czasów t pomiaru oraz $\delta_{Cx}=f(n)$ dla różnych ilości n zebranych próbek.
3. Punkty 2, 3 powtórzyć dla wybranych dwóch wartości indukcyjności L_x cewki.
4. Dokonać analizy metrologicznej dokładności pomiaru składowych L_x, C_x impedancji z uwzględnieniem dokładności wyznaczania stałej czasowej obwodu R_xC_x oraz wartości rezystancji R_x.
5. Wyciągnąć odnośne wnioski z przeprowadzonych pomiarów, określić przyczyny występowania poszczególnych rodzajów błędów. Określić czy metoda lepiej nadaje się do pomiaru dużych, np. kilkadziesiąt µF, czy małych wartości C_x. To samo sprawdzić dla podanej przez Prowadzącego indukcyjności L_x.

Literatura

[1] Dokumentacja karty pomiarowej *NI USB 6008* firmy National Instruments.

[2] Kozera M.: „*Wyznaczanie składowych impedancji, pojemności oraz indukcyjności, metodą dynamiczną. Wizualizacja z wykorzystaniem środowiska LabVIEW*” praca dyplomowa magisterska, Politechnika Częstochowska, Częstochowa 2005 (promotor: Prof. dr hab. inż. Waldemar Minkina).

[3] Minkina W., Sołtysiak W.: „*Methode der dynamischen Zustände zur Kapazitätsmessung unter Verwendung eines Mikrorechners*” Messen, Prüfen, Automatisieren (mpa), 25 (1989) Nr 1/2, S. 48-53.

[4] Strzałkowski A., Śliżyński A.: „*Matematyczne metody opracowywania wyników pomiarów*” PWN, Warszawa 1978.

[5] Tłaczała W.: „*Środowisko LabVIEW w eksperymencie wspomaganym komputerowo*” WNT, Warszawa 2002.

[6] WWW.ni.com/LabVIEW

[7] WWW.LabVIEW.pl

[8] WWW.elektroda.pl

Instrukcję opracował:

KIEROWNIK
Zakładu Systemów Pomiarowych

prof. dr hab. inż. Waldemar MINKINA

Częstochowa, luty 2018

Dodatek

Wyprowadzenie wzoru (7) na podstawie wzoru (5):

$$C_x = \frac{(t_1+t_2)^2 - 2(t_1^2+t_2^2)}{R_x\{2[t_1 \ln u_{Cx}(t_1) + t_2 \ln u_{Cx}(t_2)] - (t_1+t_2)[\ln u_{Cx}(t_1) + \ln u_{Cx}(t_2)]\}} =$$

$$\frac{t_1^2 + 2t_1t_2 + t_2^2 - 2t_1^2 - 2t_2^2}{R_x[2t_1 \ln u_{Cx}(t_1) + 2t_2 \ln u_{Cx}(t_2) - t_1 \ln u_{Cx}(t_1) - t_1 \ln u_{Cx}(t_2) - t_2 \ln u_{Cx}(t_1) - t_2 \ln u_{Cx}(t_2)]} =$$

$$\frac{2t_1t_2 - t_1^2 - t_2^2}{R_x[t_1 \ln u_{Cx}(t_1) + t_2 \ln u_{Cx}(t_2) - t_1 \ln u_{Cx}(t_2) - t_2 \ln u_{Cx}(t_1)]} =$$

$$\frac{-(t_1-t_2)(t_1-t_2)}{R_x[(t_1-t_2)\ln u_{Cx}(t_1) - (t_1-t_2)\ln u_{Cx}(t_2)]} = \frac{t_2-t_1}{R_x[\ln u_{Cx}(t_1) - \ln u_{Cx}(t_2)]}.$$