|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP13 | |  |
| Tytuł ćwiczenia: | | **Programowe sterowanie interfejsem użytkownika** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Klaster błędu** | 3 |
| 2.2. | **Zastosowanie klastra błędu w „dataflow”** | 4 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 6 |
| 3.1. | **Obiekt badany** | 6 |
| 3.2. | **Urządzenia dodatkowe** | 6 |
| 3.3. | **Oprogramowanie** | 6 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 6 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – działanie węzła Merge Errors** | 7 |
| 6. | **RAPORT** | 8 |
| 7. | **PYTANIA** | 8 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. |  | 3 |
| 2. | Widok okna Explain Error w przypadku sprawdzenia kodu błędu i ostrzeżenia. | 4 |
| 3. | Widok węzła Simple Error Handler z dostępnymi opcjami informowania o błędach. | 4 |
| 4. | Zamknięcie węzła nie posiadającego terminali błędu w strukturze Case (a), w subVI (b). | 5 |
| 5. |  | 5 |
| 6. | Pomoc kontekstowa z opisem węzła Merge Errors. | 5 |
| 7. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

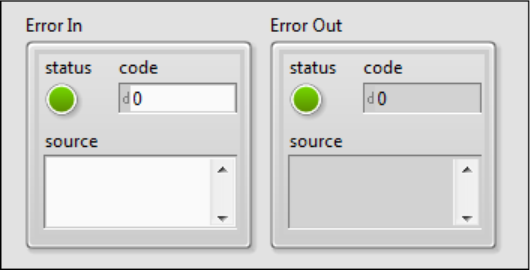
1. **CELE ĆWICZENIA**

Celem ćwiczenia jest poznanie się z:

* Strukturą klastra błędu,
* Podstawowymi metodami obsługi błędów,
* Zastosowaniu klastra błędów w „dataflow”.

1. **WPROWADZENIE**
   1. **Klaster błędu**

Klaster błędu jest kontrolką typu Cluster zawierająca następujące elementy: *status* (**Boolean**), *code* (**I32**), *source* (**String**). Widok kontrolki i wskaźnika w strukturze klastra błędu przedstawia Rys. 1.



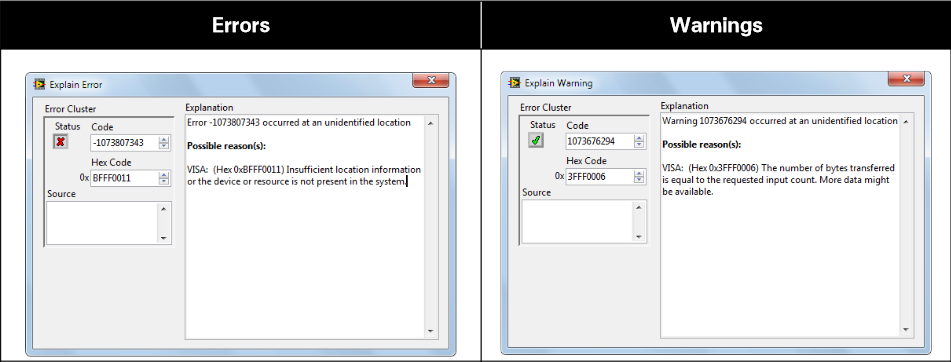
1. Kontrolka i wskaźnik błędu.

Obiekty klastra zawierają następujące informacje:

* *status* – wskazuje wystąpienie błędu, wartość **FALSE** oznacza brak błędu, wartość **TRUE** oznacza wystąpienie błędu,
* kod (*code*) – wskazuje wartość błędu, wartość 0 oznacza brak błędu, wartość inna niż 0 oznacza wystąpienie błędu lub ostrzeżenie,
* źródło (*source*) wskazuje miejsce wystąpienia błędu (plik vi).

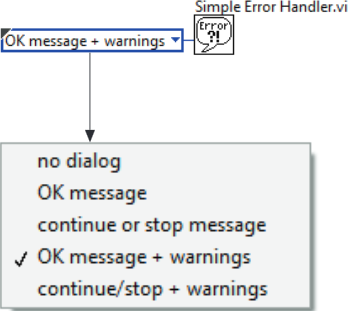
Programy napisane w LabVIEW domyślnie mają włączoną automatyczną obsługę błędów polegającą na natychmiastowym przerwaniu działania aplikacji i wyświetleniu odpowiedniego komunikatu w przypadku wystąpienia błędu. Nie jest to rozwiązanie zalecane zwłaszcza w aplikacjach obsługujących dostęp do plików lub wymieniających informacje z urządzeniami zewnętrznymi (np. karty DAQ). Wyłączenie domyślnej obsługi błędów możliwe jest na dwa sposoby: wyłączenie obsługi błędów w opcjach środowiska LabVIEW, połączenie przewodami błędów wejść i wyjść błędu zachowując odpowiednią kolejność wykonywania subVI zgodnie z oczekiwanym „dataflow”. W drugim przypadku przerwanie aplikacji wynikające z pojawienia się błędu odbywa się w miejscu przerwania przewodu błędu.

Klaster błędu informuje o wystąpieniu błędu lub ostrzeżenia. W przypadku błędu status ma wartość TRUE. Dokładniejsze informacje nt. błędów i ostrzeżeń można odczytać w **Help** 🡪 **Explain Error** (Rys. 2).



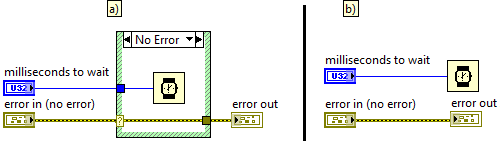
1. Widok okna Explain Error w przypadku sprawdzenia kodu błędu i ostrzeżenia.
   1. **Zastosowanie klastra błędu w „dataflow”**

Klaster błędu jest najpopularniejszą metodą zapewnienia właściwej kolejności wykonywania działań w LabVIEW, szczególnie w przypadku, jeżeli kolejne subVI nie zawsze przesyłają inne sygnały między sobą. Najprostszym zakończeniem takiej sekwencji jest zastosowanie węzła **Simple Error Handler** znajdującego się w **PPM** 🡪 **Programming** 🡪 **dialog & User Interface** (Rys. 3). Węzeł ten umożliwia wygenerowanie komunikatu w zależności od konfiguracji, najpopularniejszym i domyślnym jest *OK Message + warnings*.



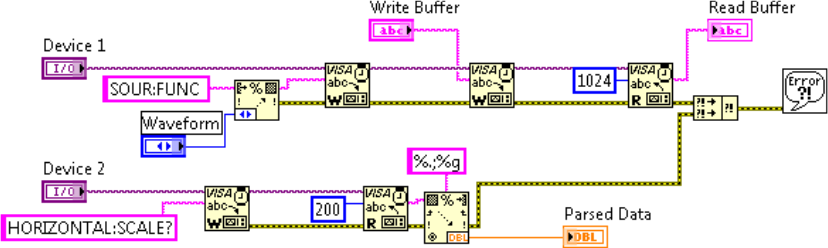
1. Widok węzła Simple Error Handler z dostępnymi opcjami informowania o błędach.

Jeżeli wybrany węzeł nie posiada wejść i wyjść błędu można to uzupełnić zamykając go w strukturze **Case**, i/lub w subVI (strukturę Case można oczywiście także zamknąć w subVI), takie rozwiązania przedstawia Rys. 4.



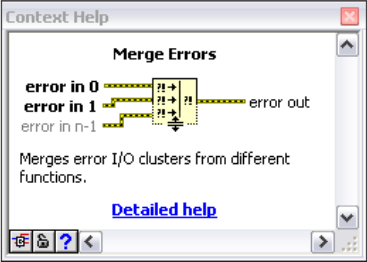
1. Zamknięcie węzła nie posiadającego terminali błędu w strukturze Case (a), w subVI (b).

Zdarzają się także przypadki, że trzeba scalić kilka przewodów błędu w jeden. Przykład został przedstawiony na Rys. 5. W tym celu należy skorzystać z węzła **Merge Errors**. Węzeł ten sprawdza wszystkie wejścia, czy nie pojawiła się na którymś błąd lub ostrzeżenie.



1. Komunikacja z dwoma urządzeniami – połączenie przewodów błędu.

Węzeł **Merge Errors** można rozszerzyć na dowolną liczbę wejść. Jeżeli na co najmniej jednym wejściu pojawi się błąd to pierwszy błąd zostanie przesłany dalej, informacja o pozostałych błędach jest tracona. W przypadku wystąpienia ostrzeżeń sytuacja jest podobna. Jeżeli wystąpią zarówno błędy jak i ostrzeżenia to priorytet mają błędy więc przesłany zostanie pierwszy błąd niezależnie, czy wyżej pojawiło się ostrzeżenie.



1. Pomoc kontekstowa z opisem węzła Merge Errors.
2. **LABORATORYJNE STANOWISKO BADAWCZE**
   1. **Obiekt badany**

- Środowisko programistyczne LabVIEW,

* 1. **Urządzenia dodatkowe**

- brak,

* 1. **Oprogramowanie**
* LabVIEW 2013 lub nowszy

1. **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI**

Kolejne kroki do wykonania podczas zajęć:

* Obsługa błędów w programach LabVIEW

- otworzyć projekt „*System pomiarowy DAQ*”,

- otworzyć plik „*Main\_maszyna.vi*”,

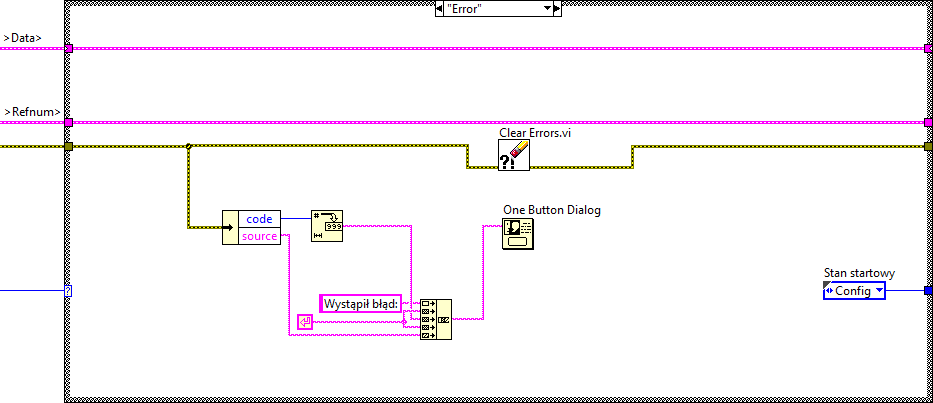
- uzupełnić maszynę stanów o stany **Error** i **Error end**:

* dodać odpowiednie stany w kontrolce „*Stany.ctl*”,
* dodać stany w strukturze Case,

- uzupełnić stany: **Config**, **Measure**, **Analyze**, **Save**, **STOP** tak, aby w przypadku wystąpienia błędu nastąpiło przejście do stanu **Error** (zalecam zrobić jeden subVI),

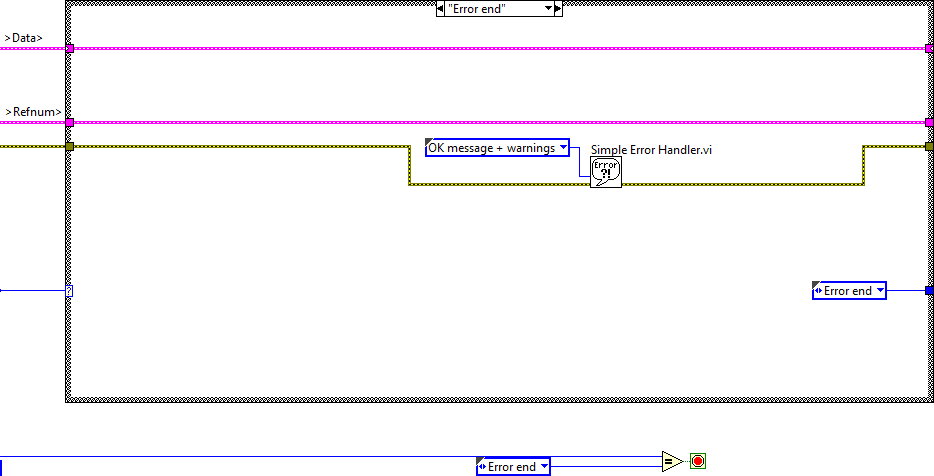
- uzupełnić stan **STOP**, aby domyślnie przechodził do stanu **Error out**,

- uzupełnić stan **Error** zgodnie z rysunkiem:



- zmienić zatrzymanie aplikacji na stan **Error end** (rysunek poniżej),

- uzupełnić stan **Error end** zgodnie z rysunkiem:



- korzystając z dodatkowego VI „Generuj\_Error” wygenerować błąd w wybranym stanie,

- sprawdzić działanie aplikacji.

1. **PRZYKŁAD REALIZACJI ZADANIA – działanie węzła Merge Errors**

Przykład działania węzła Merge Errors. Podłączono cztery kontrolki **error in** do wejść węzła **Merge Errors**, wynik wyświetlany jest na wskaźniku **error out**. Na Rys. 7 przedstawiono aplikację.

|  |  |
| --- | --- |
| a) | b) |
| 1. Program do sprawdzania działania węzła Merge Errors. | |

Sprawdzono wybrane możliwości wystąpienia błędów i ostrzeżeń, wynik przedstawiono w tabeli:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| error in 1 | error in 1 | error in 1 | error in 1 | error out |
| No error, 0 | No error, 0 | No error, 0 | No error, 0 | No error, 0 |
| No error, 0 | **Error, 100** | No error, 0 | No error, 0 | Error, 100 |
| No error, 0 | **Error, 100** | **Error, -100** | No error, 0 | Error, 100 |
| **Warning, 504** | No error, 0 | No error, 0 | No error, 0 | Warning, 504 |
| No error, 0 | **Warning, 504** | No error, 0 | **Warning, -5504** | Warning, 504 |
| No error, 0 | **Error, -5504** | **Warning, 504** | No error, 0 | Error, -5504 |
| No error, 0 | **Warning, 504** | **Error, 5504** | No error, 0 | Error, 5504 |

1. **RAPORT**

Raport z przeprowadzonego ćwiczenia laboratoryjnego powinien zawierać opis kolejnych czynności wykonywanych w trakcie realizacji ćwiczenia, zrzuty ekranu dokumentujące wykonane kroki oraz zanotowane parametry konfiguracyjne kart DAQ (mogą być zawarte  
w tabeli).

1. **PYTANIA**
2. Które z następujących są komponentami struktury klastra błędu
3. Status: Boolean,
4. Error: String,
5. Code: 32-bit integer,
6. Source: String.
7. Wszystkie błędy mają ujemny kod błędu a ostrzeżenia dodatni?
8. Prawda,
9. Fałsz.
10. Węzeł **Merge Errors** łączy informacje o błędach z wielu źródeł?
11. Prawda,
12. Fałsz.

# LITERATURA

1. LabVIEW Core 1 Course manual.
2. LabVIEW Core 1 Exercise book.
3. LabVIEW Core 2 Course manual.
4. LabVIEW Core 2 Exercise book.
5. https://www.youtube.com/watch?v=iNm0zWY7o8g&list=PLUnVykytJXxPxm5u0vRKpPRVVg2u\_WgFg&index=3
6. https://www.youtube.com/watch?v=RuIN31rSO2k

Opracowanie: Marek Kciuk

# ZADANIA DO REALIZACJI PRZED PRZYSTĄPIENIEM DO ĆWICZENIA

- ukończyć zadania z instrukcji nr 12.

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

-

# ZAŁĄCZNIKI

## Opóźnienia i zarządzanie czasem działania aplikacji