|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP07 | |  |
| Tytuł ćwiczenia: | | **Struktura i obsługa klastra błędu** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Wektory** | 3 |
| 2.2. | **Klastry** | 5 |
| 2.3. | **Klaster błędu** | 6 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 7 |
| 3.1. | **Obiekt badany** | 7 |
| 3.2. | **Urządzenia dodatkowe** | 7 |
| 3.3. | **Oprogramowanie** | 7 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 7 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – obliczanie histogramu danych** | 8 |
| 6. | **RAPORT** | 9 |
| 7. | **PYTANIA** | 9 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
| 8. |  |  |
| 9. |  |  |
|  |  |  |
|  |  |  |

1. **CELE ĆWICZENIA**

Celem ćwiczenia jest zapoznanie się z budowa i obsługą klastra błędu:

* Struktura klastra błędu,
* Zastosowanie klastra błędu do sterowania wykonywaniem programu,
* Łączenie klastrów błędu.
* Obsługa błedów.

1. **WPROWADZENIE**
   1. **Struktura klastra błędu**

Klaster błędu to struktura typu Klaster zawierająca składowe:

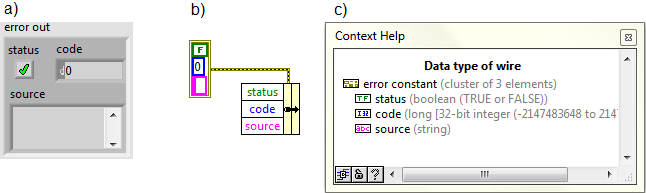
- **status** (typu boolean): wartość **False** oznacza brak błędu,

- **kod** (typu 32-integer): zawiera numer błędu, w przypadku gdy kod ma wartość niezerową a status ma wartość **True** wystąpił błąd (**error**), w przypadku gdy kod ma wartość niezerową a status ma wartość **False** wystąpiło ostrzeżenie (**warning**),

**- źródło** informuje w którym subVI nastąpił błąd.

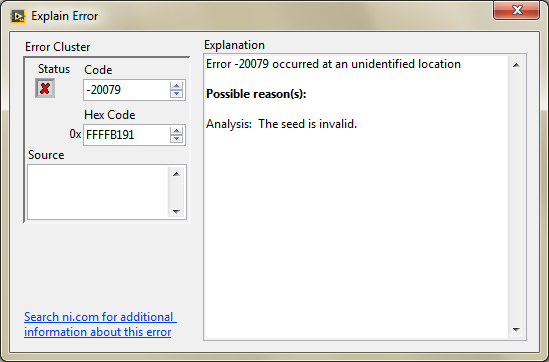
W tabeli

|  |  |  |  |
| --- | --- | --- | --- |
| Element | Brak błędu  No Error | Ostrzeżenie  Warning | Błąd  Error |
| Status | False | False | True |
| Kod | 0 | ≠0 | ≠0 |



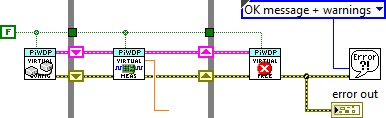
1. Wskaźnik błędu (a), stała błędu rozłożona na składowe (b), informacja o typie danych klastra błędu (c).

Kod błędu może mieć wartość dodatnią lub ujemną, każda wartość różna od zera informuje, że program nie do końca wykonał się w sposób prawidłowy. Informację o błędzie wyświetla komunikat, dokładniejsze informacje można uzyskać wpisując kod błędu w oknie **Explain Error** znajdującym się w **Help** 🡪 **Explain Error…** .



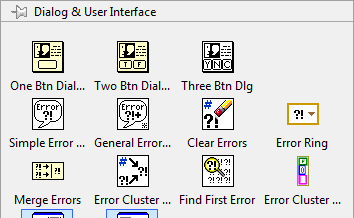
1. Widok okna Explain Error.
   1. **Sterowanie wykonywaniem programu**

Klaster błędu jest preferowanym mechanizmem kontroli wykonywania programu. Stosowanie klastra błędu łączącego kolejne subVI realizuje dwie funkcjonalności. (1) określa kolejność wykonywania programu jednoznacznie określając strumień Dataflow zapewniając wykonywanie węzłów w programie zgodnie z kolejnością podpięcia do przewodu błędu. (2) wyłącza domyślną obsługę błędów polegającą na natychmiastowym przerwaniu działania aplikacji. W przypadku klastra błędu błąd jest propagowany przez wszystkie subVI aż do punktu, w którym program obsługuje zdarzenie. Przykład prostego programu akwizycji pracującego z prawidłowym zastosowaniem i obsługą klastra błędu przedstawiono na Rys. 3. Podczas zakończenia programu następuje obsługa błędu za pomocą subVI **Simple Error Handler.vi**.



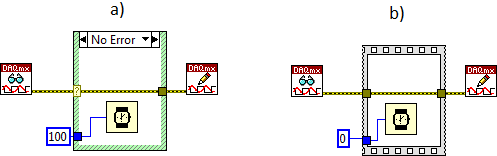
1. Aplikacja pomiarowa, której wykonanie jest sterowane za pomocą klastra błędu.

Domyślną strukturą podprogramów obsługującą mechanizm kontroli błędu jest **SubVI with Error Handling** dostępna w szablonach **File** 🡪 **New…** . Najczęściej w strukturze tej właściwy kod funkcji pisany jest w stanie **No Error**, obsługę błędów realizuje się w stanie **Error**. Najczęściej jednak ten stan pozostawia się pusty, aby błąd był propagowany dalej nie realizując funkcjonalności podprogramu. Podstawową metodą obsługi i informowania użytkownika o występujących błędach na poziomie CLAD jest węzeł **Simple Error Handler** znajdujący się w **Functions Palette** 🡪 **Programming** 🡪 **Dialog & User Interface**. Zastosowanie węzła przedstawiono na Rys. 3. Inne funkcje obsługi błędów znajdują się w tej samej palecie (Rys. 4).



1. Część palety Dialog & User Interface dotycząca obsługi błędów.

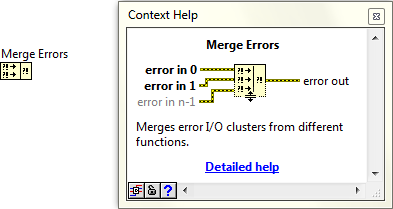
Niektóre węzły nie posiadają tuneli obsługi klastra błędu (np. Wait (ms)), w celu wprowadzenia ich do struktury dataflow stosuje się rozwiązanie z zastosowaniem struktur: wyboru **Case Structure** lub sekwencyjnej **Flat Sequence Structure** przedstawione na Rys. 5.



1. Zastosowanie struktur do obsługi klastra błędu w przypadku węzła Wait (ms).

Przy czym rozwiązanie Rys. 5a jest preferowane, można zamknąć w subVI i stosować wielokrotnie.

Często w programie prowadzonych jest kilka linii błędu, które w pewnym momencie trzeba połączyć. Do tego celu służy węzeł **Merge Errors** przedstawiony na Rys. 6 znajdujący się w tej samej palecie (Rys. 4). Działanie węzła Merge Errors polega na wyszukaniu pierwszego błędu lub (w przypadku braku błędów) ostrzeżenia i przesłania na wyjście. W tym rozwiązaniu tylko jeden (pierwszy) błąd jest przenoszony, informacja o pozostałych jest tracona.



1. Widok węzła Merge Errors oraz jego opis w pomocy kontekstowej.

Pełniejszą obsługę i łączenie błędów należy napisać samodzielnie, jeśli zachodzi taka potrzeba.

* 1. **Dodawanie własnych kodów błędów do programu**

System obsługi błędów w LabVIEW jest złożony i uwzględnia wiele możliwych błędów przypisując im różne kody. Jednak możliwe jest również zdefiniowanie własnego błędu wynikającego z potrzeby aplikacji. W tym celu zostały przewidziane trzy zakresy kodów dostępne dla użytkownika tzw. **Custom Error Codes**:

|  |  |  |
| --- | --- | --- |
| -8999 – -8000, | 5000 – 9999, | 500 000 – 599 999. |

1. **LABORATORYJNE STANOWISKO BADAWCZE**
   1. **Obiekt badany**

- Środowisko programistyczne LabVIEW,

* 1. **Urządzenia dodatkowe**

- brak,

* 1. **Oprogramowanie**
* LabVIEW 2014 lub nowszy

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

Kolejne kroki do wykonania podczas zajęć:

* Zastosowanie klastrów:

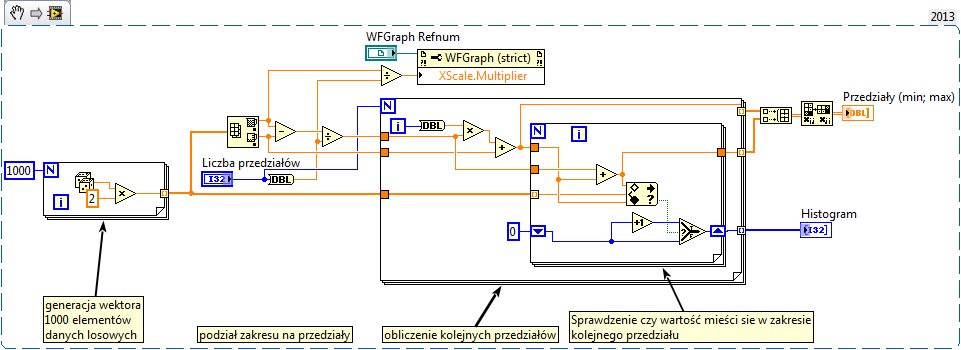
- zapisać plik,

* Zastosowanie wektorów:

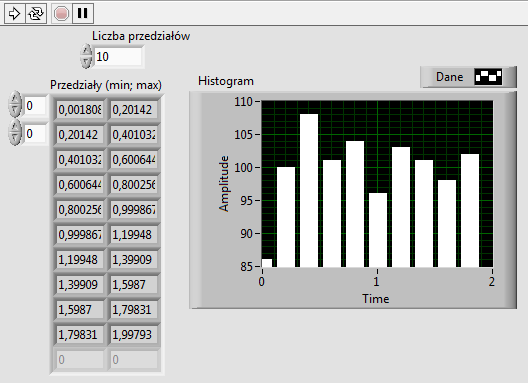
- otworzyć plik „*main.vi*”,

1. **PRZYKŁAD REALIZACJI ZADANIA – generacja błędu użytkownika**

Przykładowa realizacja obliczania histogramu danych została przedstawiona na Rys. 8. Algorytm został podzielony na trzy części: obliczenie zakresów, przejście przez kolejne przedziały (zewnętrzna pętla **For**), sprawdzenie czy wartość mieści się w aktualnym zakresie (wewnętrzna pętla **For**). Wadą tego rozwiązania jest konieczność przeszukiwania całego zbioru danych tyle razy, ile wynosi liczba przedziałów.



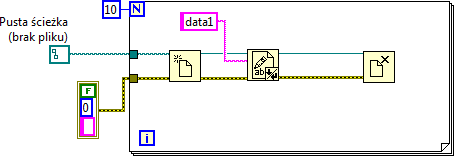
1. Przykładowa realizacja zadania obliczania histogramu – kod programu.



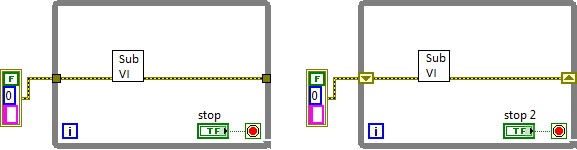
1. Przykładowa realizacja zadania obliczania histogramu – panel programu.
2. **RAPORT**

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. Przedstawić budowę klastra błedu.
3. Omówić zastosowanie przewodu błędu w ustalaniu kolejności wykonania programu.
4. Do czego służy węzeł Merge Errors i jak działa?
5. Ile razy zostanie wywołany błąd?



1. Wyjaśnić różnicę w działaniu linii błędu dla przedstawionych kodów:



# 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. Nota katalogowa LM35

Opracowanie: Marek Kciuk

# ZADANIA DO REALIZACJI PRZED PRZYSTĄPIENIEM DO ĆWICZENIA

- ukończyć zadania z instrukcji nr 6.

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

- wyprowadzić wspólnie dane zmierzone i uśrednione z pętli akwizycji za pomocą klastra,

- zmienić funkcję obliczania histogramu z zastosowaniem węzła sortowania Sort 1D Array,

# ZAŁĄCZNIKI

## Waveform Graph vs Waveform Chart

LabVIEW