Dodatek

INTERAKCJA CZŁOWIEK MASZYNA

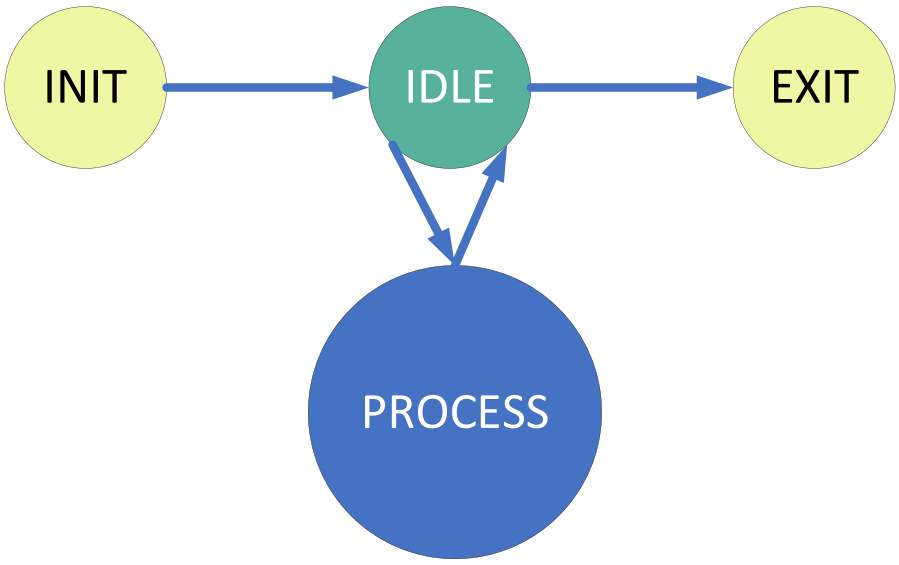
Programowanie w środowisku LabVIEW – zmienne w aplikacji.

Celem niniejszego materiału jest zapoznanie studenta z metodyką programowania, projektowania i tworzenia aplikacji. Podstawowym zadaniem jest przybliżenie zagadnień i problemów występujących w systemach SCADA. W trakcie ćwiczenia student nabędzie podstawowe informacje dotyczące środowiska i umiejętności posługiwania się nim w procesie tworzenia aplikacji pomiarowych. Niniejsze zagadnienie dotyczy przestawienia podstawowej struktury programu jaką może być maszyna stanu.

Zakład Systemów Informacyjno-Pomiarowych

IETiSIP, Wydział Elektryczny, PW

Program symulatora generatora i analizatora przebiegów okresowych w podstawowej wersji oparty jest na pętli **while** w wariancie prostym i pętli **while** z konstrukcją **Event Structure** w wariancie rozbudowanym. Program ten warto zmodyfikować, rozdzielając poszczególne fazy pracy aplikacji i wykorzystać maszynę stanu.



Rysunek 1. Podstawowa struktura Maszyny Stanu

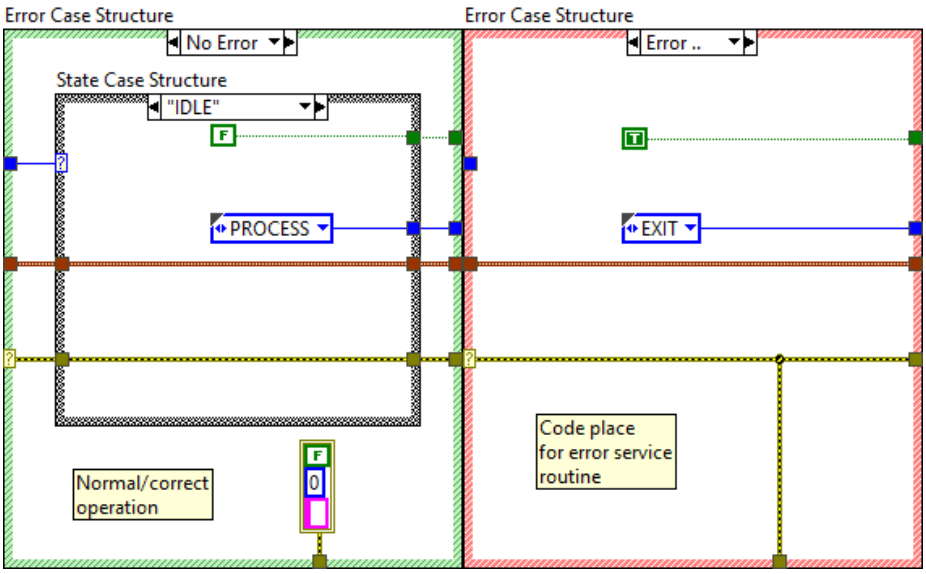
Przestawiona na rysunku 1 struktura Maszyny Stanu zawiera 4 elementy/stany:

- **INIT** – stan początkowy pracy aplikacji. Może być wykorzystany np.: na ustawienie nastaw płyty czołowej do wartości domyślnych albo wczytanie konfiguracji z pliku i aktualizację nastaw płyty czołowej na odpowiadające wartości.
- **IDLE** – „nasłuchiwanie" na przychodzące zdarzenia (od operatora płyty czołowej albo w wyniku zmiany wartości wybranego parametru) i pobieranie danych
- **PROCESS** – przetwarzanie i zasadnicza obsługa przychodzących zdarzeń
- **EXIT** – obsługa fazy kończenia działania programu polegająca np.: na zapisaniu ostatniej konfiguracji urządzenia.

Tak więc działanie programu wykonującego działanie zgodnie z rysunkiem 1 rozpoczyna się od stanu **INIT**, po którym następuje przejście do stanu **IDLE**. Ten kolei, jeżeli nie pojawią się żadne zdarzenia ma możliwość wywołania samego siebie. Możliwe jest zatem w takim razie np.: choćby odświeżenie ekranu. Jeżeli pojawia się zdarzenie (np.: z płyty czołowej), jest ono odbierane wraz z danymi. W proponowanym rozwiązaniu odczyt danych

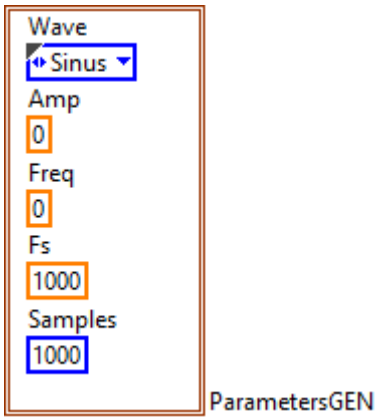
w następuje stanie **IDLE** ale nie jest to konieczne i można to zrobić w stanie **PROCESS**. Jeśli natomiast pojawi się zdarzenie naciśnięcia przycisku **Stop**, maszyna przechodzi do stanu EXIT gdzie kończy działanie. Początkowo stan ten jest pusty, ale można go wypełnić np.: kodem do zapamiętania ostatniej konfiguracji.

W projektowanym schemacie/szablonie maszyny stanu warto uwzględnić następującą regułę. Maszyna stanu działa wg. opisanego porządku tylko wtedy jeśli nie pojawiają się błędy. Jeśli natomiast one wystąpią, Maszyna Stanu przekazuje sterowanie do kodu obsługującego błędy i jeśli nie uda się ich usunąć program jest zamykany – rysunek 2.



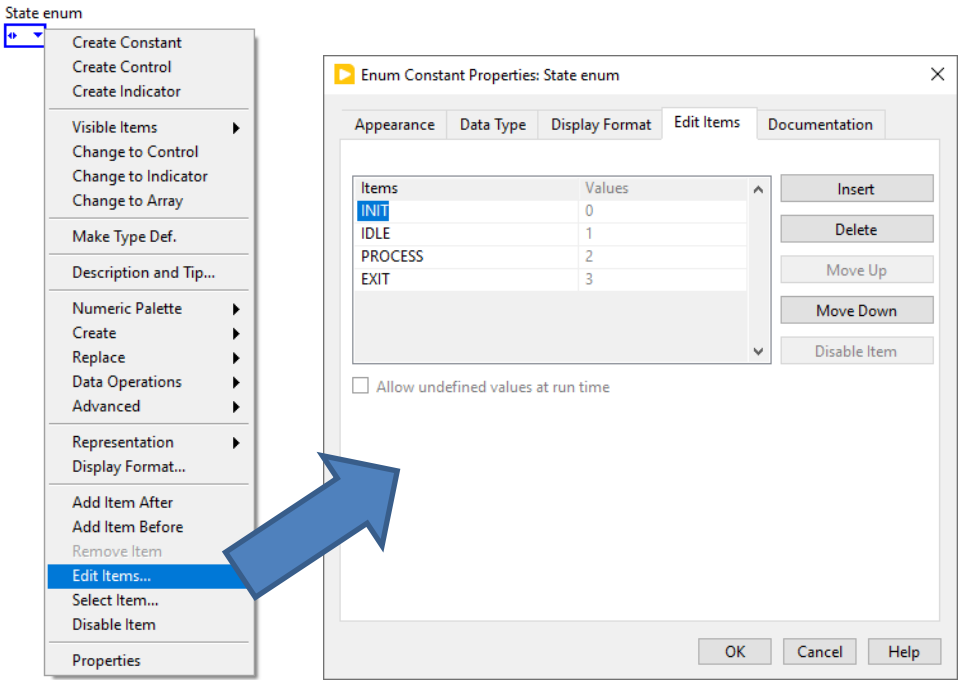
Rysunek 2. Postać instrukcji warunkowej (Case Structure) dla przypadku prawidłowego No Error – zielony i dla obsługi błędów Error.. – czerwony.

Kolejnym założeniem jest możliwość przechowywania aktualnych parametrów obiektu/urządzenia sterowanego maszyną stanu. W tym celu należy przygotować strukturę (Cluster w LabVEW), którego zadaniem będzie gromadzenie niezbędnych danych – rysunek 3.

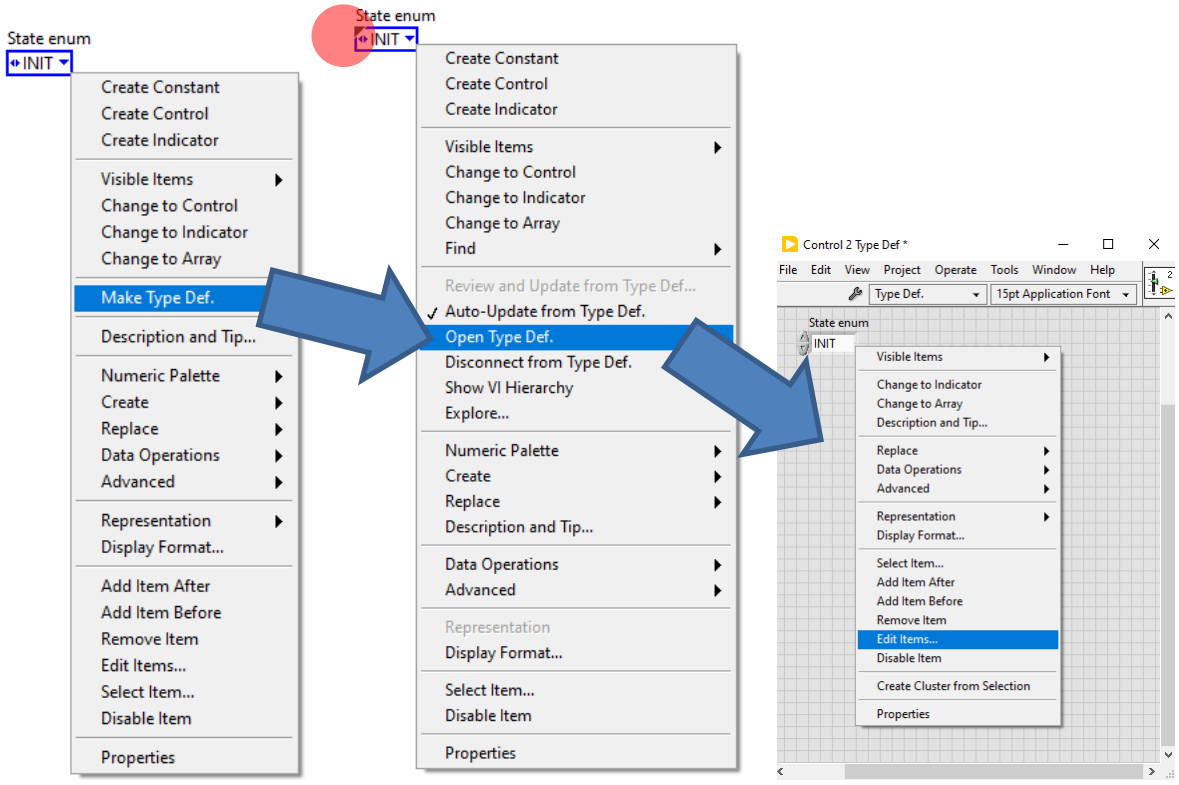


Rysunek 3. Przykład struktury/klastra przechowującego aktualne parametry urządzenia.

Definicja/zbiór stanów przechowywany będzie w zmiennej wyliczeniowej (enumerator), na podstawie należy utworzyć „własny” Typedef, tak aby móc łatwo i jednocześnie modyfikować wszystkie znajdujące się w kodzie jego instancje (użycia, wystąpienia) – rysunek 4.

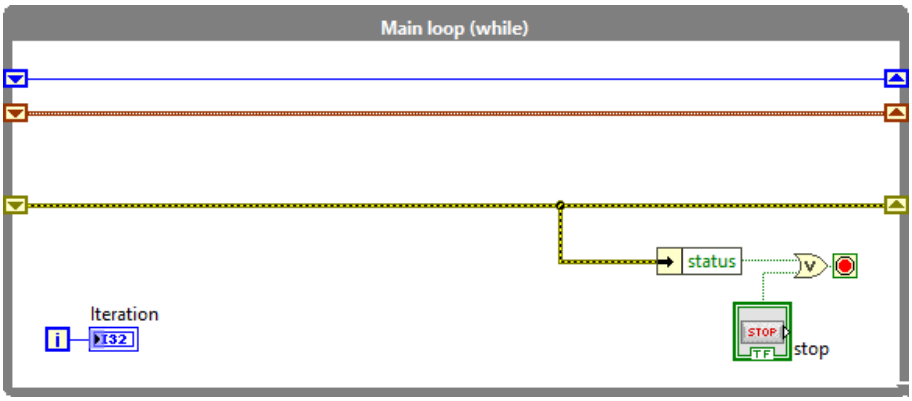


Rysunek 4. Konfiguracja enumeratora.



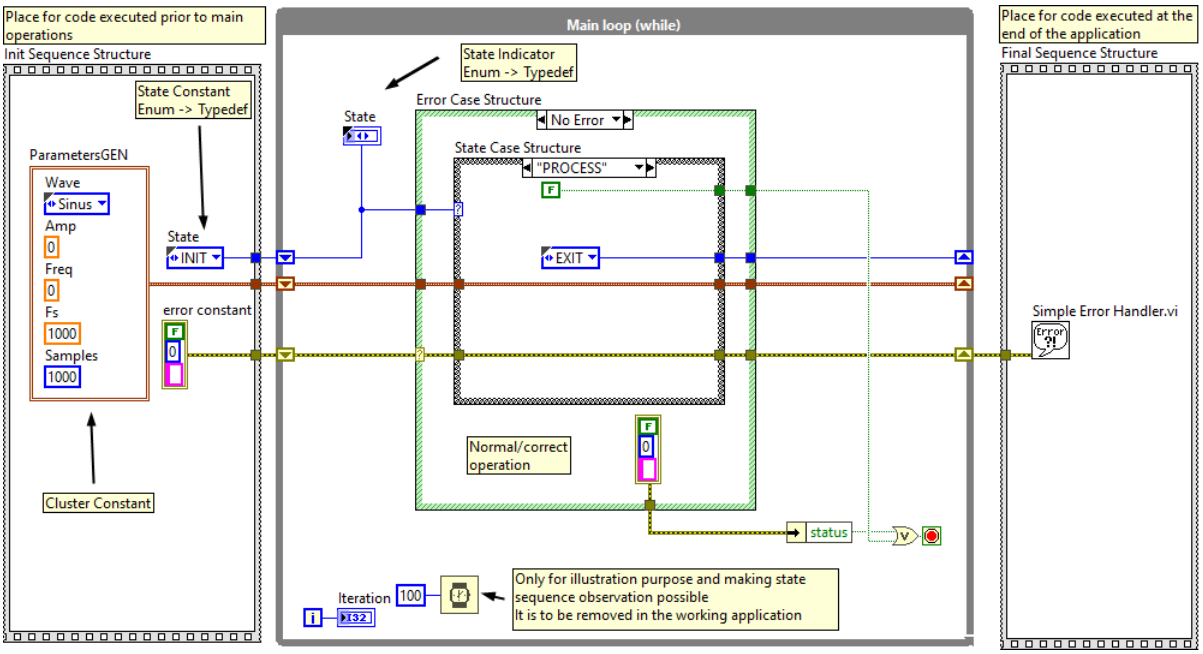
Rysunek 5. Tworzenie i edycja **typedef**.

Główna pętla programu, pętla **while** wykorzystywać będzie rejestry przesuwne dla linii: błędów, stanów i struktury/klastra parametrów aktualnych – rysunek 6.



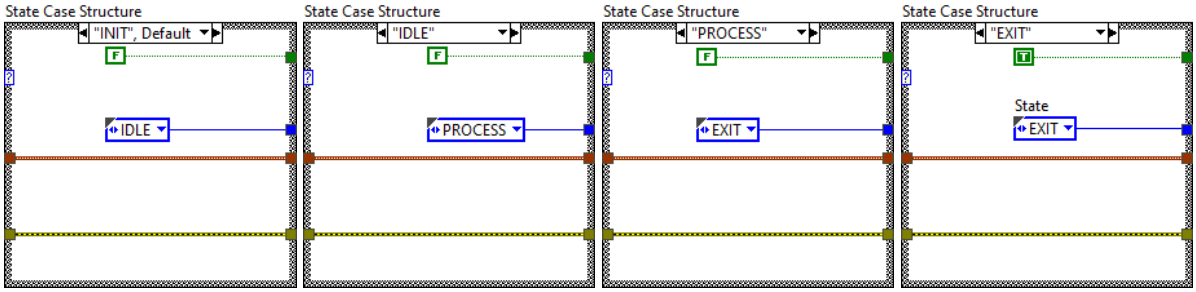
Rysunek 6. Schemat rejestrów przesuwne w pętli głównej **while**.

Zgodnie z powyższymi założeniami stworzony szkielet maszyny stanu może wyglądać tak jak na rysunku 7.



Rysunek 7. Schemat ogólny kodu maszyny stanu w LabVIEW.

W tym momencie poszczególne stany nie zawierają kodu programu urządzenia a jedynie kod sterujący wykonaniem programu, który na tym etapie przechodzi po prostu przez wszystkie stany począwszy od stanu INIT na stanie EXIT zakończywszy. W tym momencie w pętli głównej programu umieszczone jest dodatkowe opóźnienie **Wait,** którego zadaniem jest „przytrzymywanie" poszczególnych stanów tak aby można było je zaobserwować w okienku wyświetlacza **State**. Zawartość poszczególnych stanów (**State Case Structure**) zilustrowana została na rysunku 8.



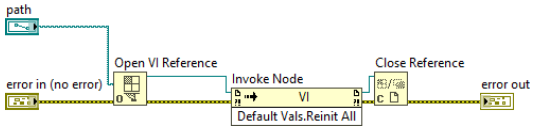
Rysunek 8. Zawartość poszczególnych stanów szkieletu programu Maszyny Stanu.

Stany INIT, IDLE oraz PROCESS są na chwilę obecną w zasadzie puste. Założenie dla wersji startowej programu jest następujące:

- stan INIT ustawia elementy płyty czołowej na wartości domyślne oraz synchronizuje je z danymi w kodzie.
- stan IDLE zawiera kod do przechwytywania zdarzeń z płyty czołowej, odczytu nowych nastaw parametrów i ich aktualizacji w strukturze

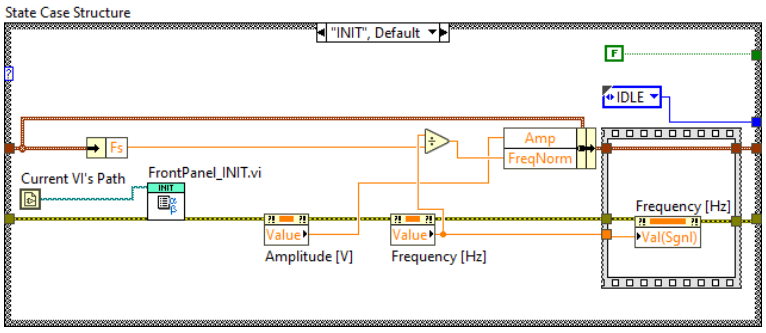
- stan PROCESS zawiera zasadniczy kod reakcji na przychodzące zdarzenia i aktualizacji zawartości płyty czołowej programu.

Ustawienie (resetowanie do) wartości domyślnych można przeprowadzić „standardową” procedurą przedstawioną na rysunku 9. Zasadniczą funkcją jest tu **Invoke Node**, która pozwala „zarządzać” aplikacją. Jedną z możliwości jest nastawienie elementów płyty czołowej na wartości domyślne. W tym celu po umieszczeniu **Invoke Node** na diagramie należy kliknąć prawym klawiszem na tę funkcję i wybrać **Select Method** -> **Default Values** -> **Reinitialize All to Default** i gotowe.



Rysunek 9. Ustawienie wartości domyślnych na płycie czołowej.

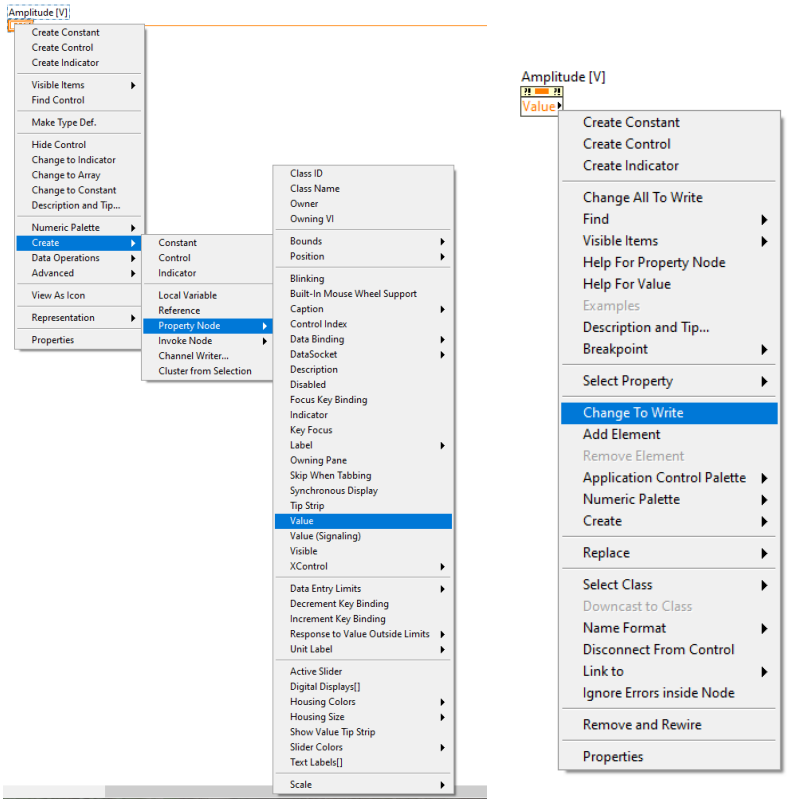
Ze względu na fakt, że czynność ta może być wykonywana wielokrotnie w różnych miejscach kodu ale też i w efekcie końcowym zajmować mniejszą część ekranu, warto rozważyć utworzenie subVI (funkcji). Wtedy stan IDLE może mieć postać jak na rysunku 10. Ustawienie wartości domyślnych zostało przeniesione do subVI FrontPanel_INIT.vi. Widoczne w dalszej części diagramu kodu terminale to „węzły” **PropertyNode**. Węzły te pozwalają odczytać i ustawić nowe własności elementów znajdujących się na płycie czołowej. W tym przykładzie służą one do odczytu aktualnej wartości nastawy z płyty czołowej i zsynchronizowaniu jej z danymi w kodzie programu. Pojedynczy węzeł własności (**Property Node**) przypisany jest do konkretnego elementu na płycie czołowej. Tak więc, aby za ich pomocą zsynchronizować wszystkie parametry/nastawy na płycie czołowej, należy wywołać je każdorazowo dla wszystkich elementów.



Rysunek 10. Przykładowy kod stanu INIT.

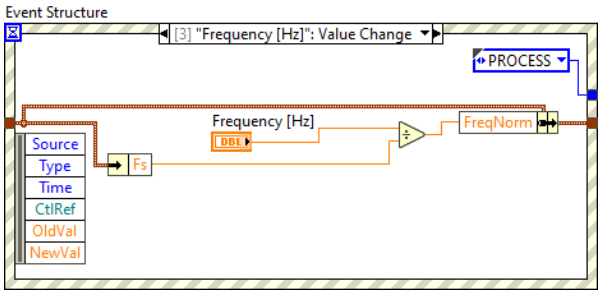
Węzeł własności zwykle zwiera pokaźny zbiór parametrów. W przykładzie wykorzystano Parametr **Value** do odczytu aktualnych wartości

poszczególnych nastaw. Rysunek 11 przedstawia kolejne kroki tworzenia węzła własności (na przykładzie potencjometru Amplitude [V]).



Rysunek 11. Tworzenie węzła własności (**Property Node**) dla potencjometru Amplitude [V] – wybierany parametr to w tym przypadku **Value**.

Na rysunku 11 po prawej stronie widoczny jest utworzony właśnie węzeł – **Property Node**. Często zachodzi potrzeba zmiany trybu dostępu do parametru tj. ustawienia zapisu czy odczytu. Służy do tego wyróżniona pozycja menu kontekstowego. Pojedynczy **Property Node** może zawierać więcej parametrów. Wtedy Należy wtedy być ostrożnym z funkcją **Change All to Write/Read** – działa zgodnie z nazwą a węzeł może jednocześnie zawierać parametry do odczytu albo zapisu. Na rysunku 10 zamieszczono dodatkowo węzeł własności dla nastawy Frequency [V] z parametrem **Val(Sgnl)**. Jest to ten sam parametr **Value** z tą różnicą, że dodatkowo generuje zdarzenie związane ze zmianą wartości. Po przejściu zatem do stanu IDLE wykonywany jest odpowiedni kod w **Event Structure** odpowiadający za obsługę tego zdarzenia. W przykładzie jest to zdarzenie **Frequency [Hz] Value changed** i kod mu kod, przedstawiony na rysunku 12.



Rysunek 12. Kod obsługi zdarzenia **Frequency [Hz] Value changed**.

Po wykonaniu kodu obsługi zdarzenia (rysunek 12) sterowanie programu przechodzi do stanu PROCESS, w którym następuje generacja nowych danych (przebiegów) i ich wyświetlenie na ekranie. Rozwiązaniem alternatywnym dla przedstawionego powyżej może być może być warunkowe przejście do stanu PROCESS bezpośrednio z INIT.