Ćwiczenie

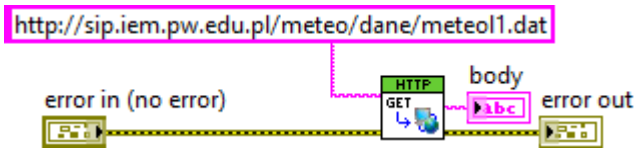
Programowanie w środowisku LabVIEW - Odczyt danych ze stacji meteorologicznej Davis Vantage Pro 2.

Celem ćwiczenia jest zapoznanie studenta z metodyką programowania, projektowania i tworzenia aplikacji, których zadaniem jest przybliżenie zagadnień i problemów występujących w systemach pomiarowych. W trakcie ćwiczenia student nabędzie podstawowe informacje dotyczące środowiska i umiejętności posługiwania się nim w procesie tworzenia aplikacji pomiarowych.

Zakład Systemów Informacyjno-Pomiarowych

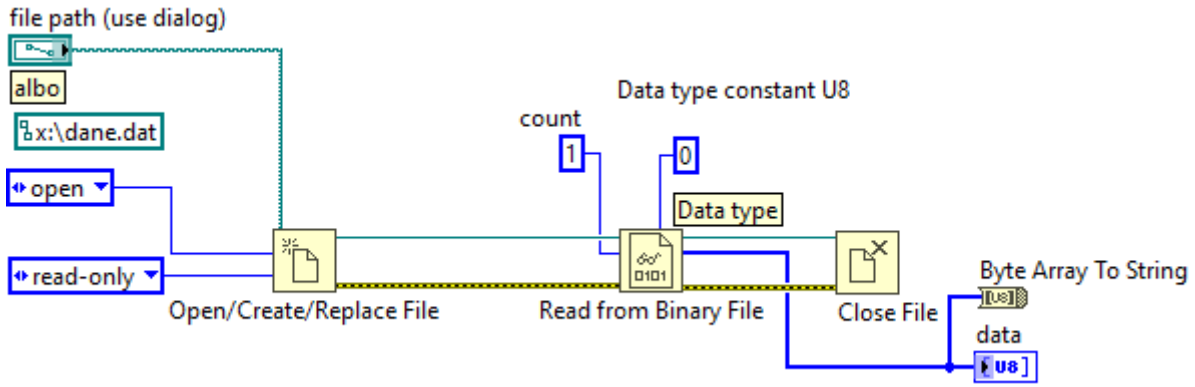
IETiSIP, Wydział Elektryczny, PW

Zadaniem aplikacji jest nawiązanie połączenia z komputerem "kontrolerem" stacji meteorologicznej i odbieranie pakietów z danymi. Połączenie należy zrealizować za pomocą funkcji **GET.vi** pozwalającą za pomocą protokołu HTTP pobrać zawartość danych plików (rysunek 1a). Adres serwisu obsługującego dane pogodowe jest następujący: **http://sip.iem.pw.edu.pl/meteo/dane/meteol?.dat**. W adresie znak zapytania oznacza cyfrę 1 lub 2.



Rysunek 1a. Przykład użycia funkcji GET.vi

Alternatywnie pliki należy odczytywać z dysku lokalnego komputera za pomocą standardowej (nie tylko w LabVIEW) sekwencji funkcji obsługujących pliki:

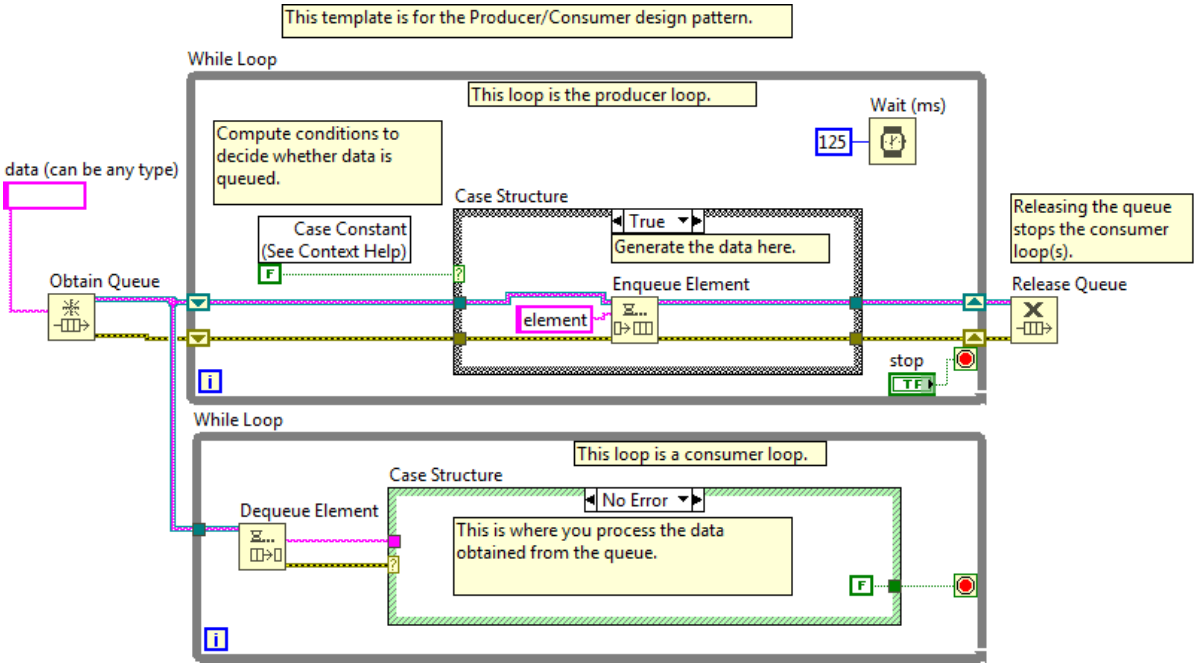


Rysunek 1b. Przykład użycia funkcji obsługujących pliki.

W ćwiczeniu pliki te (meteol1.dat oraz meteol2.dat) zawierają dane pogodowe zapisane w postaci binarnej. Dobrze jest również rozważyć wykonanie programu w oparciu o szablon Producent/Konsument, tak aby dane (zasoby WWW albo pliki lokalne) pobierać w wątku producenta a przetwarzać je w wątku konsumenta (rysunek 2a). Gotowy szablon można wstawić do kodu wybierając z menu File->New...->Frameworks->Design Pattern->Producer/Consumer (Data).

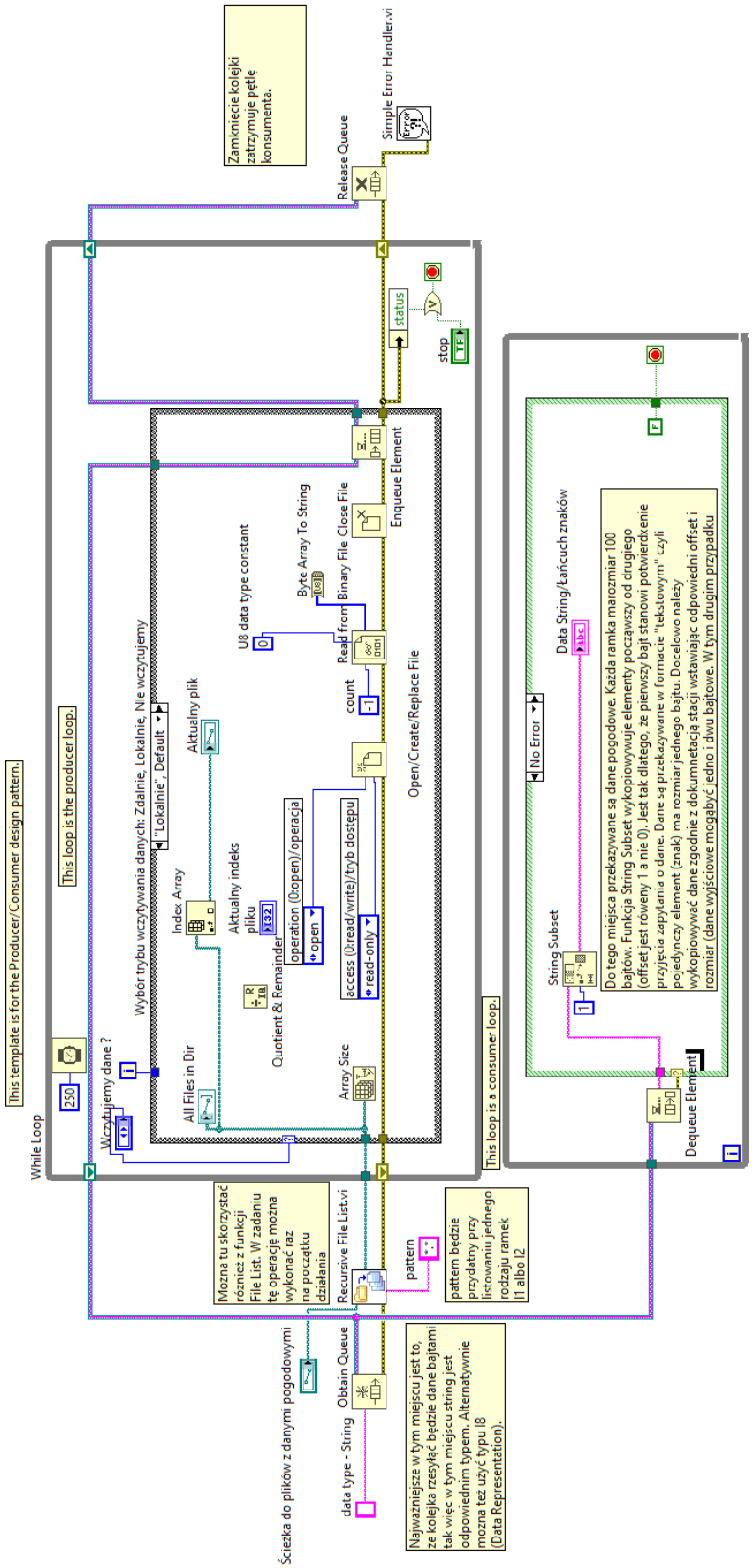
W tym celu wyjście **body** funkcji **GET.vi** lub **text** funkcji **Read from Text File** (rysunek 1a i 1b) należy podawać do funkcji **Enqueue Element** zamiast

stałej **element** na rysunku 2a. Instrukcja **Case** w pętli Producenta powinna być rozbudowana poprzez dodanie kolejnych przypadków dla trybu zdalnego, lokalnego i trybu kiedy dane nie są pobierane. Case zatem powinien być sterowany z płyty czołowej za pomocą przełącznika wielopozycyjnego, utworzonego jako enumerator (**Enum**). Opóźnienie **Wait** w pętli Producenta również można ustawiać za pomocą nastawy na panelu aplikacji lecz pomimo, iż dane odświeżane są w odstępach minutowych (tryb zdalny) to wartość ta powinna raczej pozostać na poziomie do 1s. Można również przyjąć też osobne wartości stałe dla trybu zdalnego i lokalnego. Tak więc w trybie zdalnym wynik działania funkcji **GET.vi** przekazywany będzie na wejście element funkcji **Enqueue Element** a w trybie lokalnym będzie to rezultat działania funkcji **Read from Binary File** z typem danych wejściowych ustawionym na **I8** lub **U8**. Wyjście **data** funkcji **Read from Binary File** może być odczytywanie binarnie (bezpośrednio jako tablica bajtowa) albo po wykorzystaniu funkcji **Byte Array To String** jako ciąg tekstowy – **string**. Wszystko zależy od tego jak będzie wygodniej (wykorzystywane funkcje). Najważniejsze jest to, że dane źródłowe muszą być czytane bajtami. Zarówno ciąg tekstowy (**string**) jak i tablica bajtowa (**I8** lub **U8**) mogą być tu zastosowane. Odczyt danych lokalnych może odbywać się automatycznie na podstawie zawartości katalogu z plikami (w materiałach do przedmiotu znajduje się stosowane archiwum). Wystarczy odczytać listę plików (funkcja **Recursive File List** lub prostsza i w zupełności tu wystarczająca **File List**). Obydwie zwrócą zawartość plików w podanym katalogu. W tym miejscu należy wyjaśnić iż dane meteorologiczne zawarte są w dwóch plikach meteol1.dat oraz meteol2.dat. Zawierają one różne dane pomiarowe i są inaczej zakodowane. Dlatego przydatne będzie wykorzystanie wejścia **pattern** aby wybrać tylko pliki jednego typu.



Rysunek 2a. Szablon Producer/Consumer

Na rysunku 2b przedstawiono szczegółowy schemat programu z licznymi komentarzami do kodu. Na pierwszym planie znajduje się **Case** z obsługą plików lokalnych.

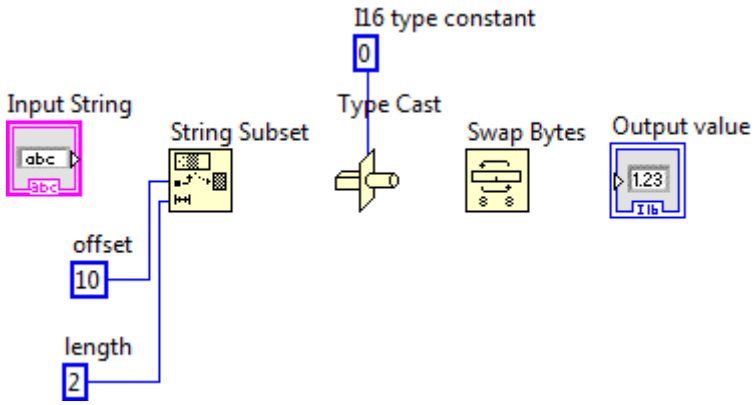


Rysunek 2b. Szczegółowy schemat programu

Po odczytaniu zawartości katalogu z plikami danych meteorologicznych (**All Files in Dir**) można odczytać liczbę tych plików (**Array Size**) i wykorzystując funkcję **Q&R** (wyjście **R**) wyznaczać resztę z dzielenia aktualnej wartości iteratora pętli **while** (**i**) przez liczbę plików w katalogu. Wynik można podać na wejście indeksu funkcji **Index Array** tak aby otrzymać nazwę konkretnego pliku. Teraz wartość tę należy podać na odpowiednie wejście funkcji **Open/Create/Replace File** uzupełniając jednocześnie w tym przypadku pozostałe parametry **operation** i **access**. W przypadku funkcji plikowych należy uzupełnić połączenia **refnum/file** oraz odpowiednio podłączyć wyjście **text** do funkcji **Enqueue Element**.

Case do obsługi danych zdalnych powinien zawierać kod jak na rysunku 1a z odpowiednio wpiętymi liniami błędu oraz stosownie podłączone wyjście **body**. Case wstrzymujący wczytywanie danych powinien zawierać jedynie przesłania linii błędu oraz identyfikatora kolejki.

Format pojedynczego pakietu danych odebranych ze stacji opisany jest na stronie 22 dokumentu VantageSerialProtocolDocs_v261.pdf. W dokumentacji odnaleźć należy położenie i format wybranych parametrów meteorologicznych (temperatury, ciśnienia, wilgotności itd.) i ich prezentację na ekranie. Pakiet danych jest w formacie **string**. Poszczególne parametry zakodowane są typowo jako wartości całkowite jedno lub dwubajtowe ze znakiem i bez, ewentualnie przeskalowane. Zadanie sprowadza się do wykorzystania zestawu funkcji z rysunku 3.



Rysunek 3. Zestaw "niezbędnik" funkcji wykorzystanych do odczytu poszczególnych parametrów pogodowych.

Funkcja **String Subset** pozwoli na wybranie określonego parametru - tu należy odpowiednio ustawić **offset** i **length** tak aby zgodnie z dokumentacją stacji zmienne te wskazywały na poszukiwany przez nas parametr pogodowy. Funkcja **Type Cast** "rzutuje" ciąg bajtowy (typ **String** jest właśnie takim

ciągiem bajtów) zawierający informację o wybranej wielkości meteorologicznej do odpowiedniego formatu. Funkcja **Swap Bytes** może okazać się przydatna przy przetwarzaniu parametrów dwubajtowych. **Output value** zawiera surową wartość wybranego parametru. Pojęcie wartość surowa oznacza tylko tyle, iż parametry pogodowe podawane są w wielkościach nie koniecznie przez Nas "preferowanych". Np. temperatura jest w stopniach Fahrenheita, opady deszczu podane są w calach na godzinę, prędkość wiatru w milach na godzinę a ciśnienie w calach słupa rtęci. Zatem może okazać się przydatna dodatkowa konwersja (konwersja skal temperatur powinna być nam już znana). W programie dane można wyświetlać w postaci "surowej" i przeskalowanej.

Należy odczytać i przedstawić jak największy zbiór wielkości !

W programie należy zatem zaprezentować:

- temperaturę chwilową
- temperaturę odczuwalną
- punkt rosy
- ciśnienie hPa oraz mmHg
- trend ciśnienia
- wilgotność
- prędkość wiatru
- kierunek wiatru
- opady dobowe/tygodniowe/miesięczne
- wschód i zachód słońca
- prognozę pogody

  Inne mile widziane (np.: wykres dobowy temperatury lub ciśnienia)

Konwersja temperatury

$t_C=(t_F-32)/1.8$

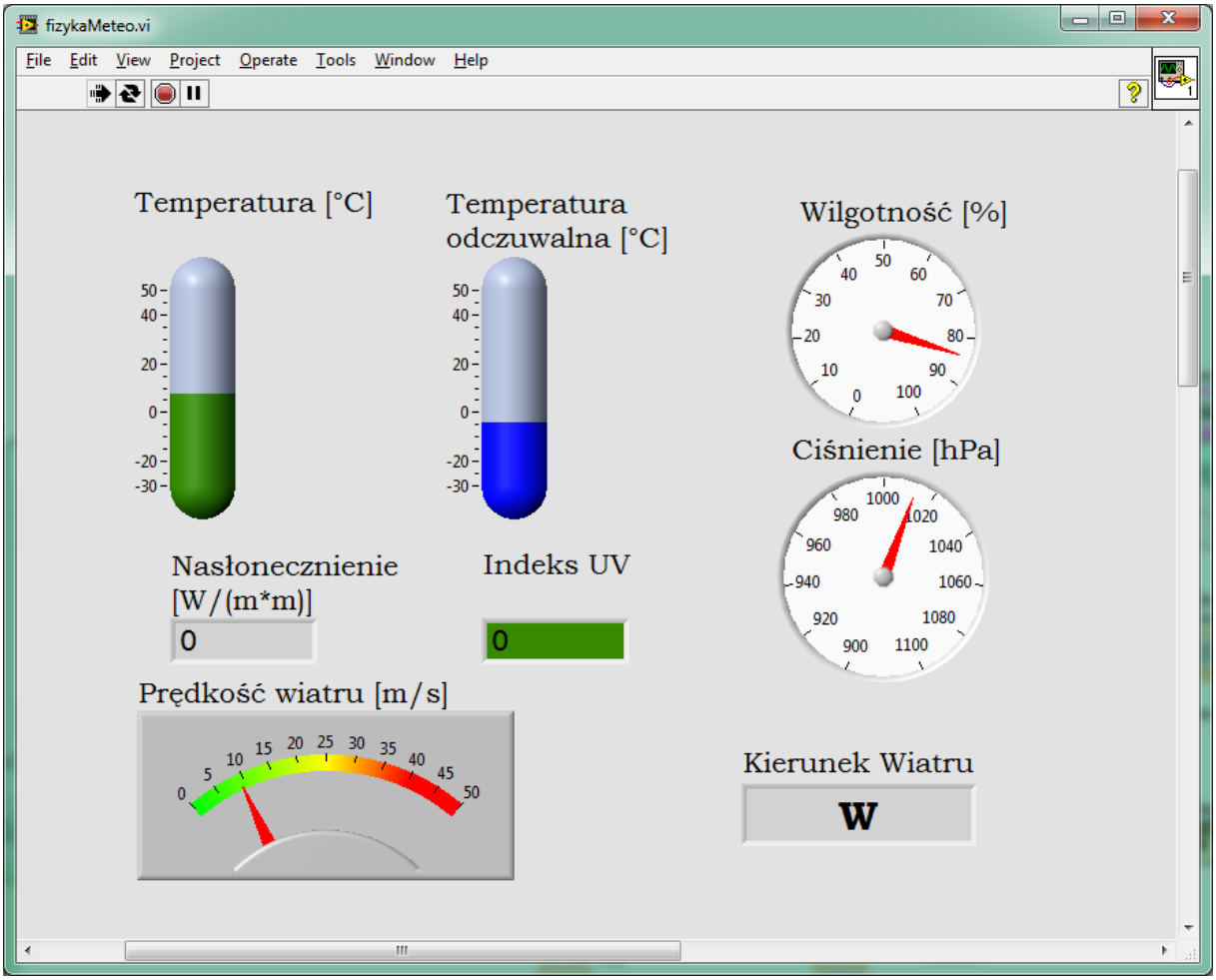
Zależności miar długości

1 cal = 25,4 mm

1 mila = 1609 m = 1,609 km

Zależności ciśnienia

1 Pa = 0,0075006167382112 mmHg



Rysunek 4. Przykładowa płyta czołowa

Zadania dodatkowe:

1. Dodać wartość maksymalną i minimalną za ostatnie N (wybrane) pomiarów lub za ostatnie N sekund (jeżeli znany będzie okres pomiarów)

2. Zmodyfikować alarmy, tak aby jednokrotne przekroczenie progów powodowało tylko ostrzeżenie a dopiero trwałe przekroczenie (dwie lub więcej kolejno po sobie następujących odczytanych wartości) powodowało właściwy alarm.

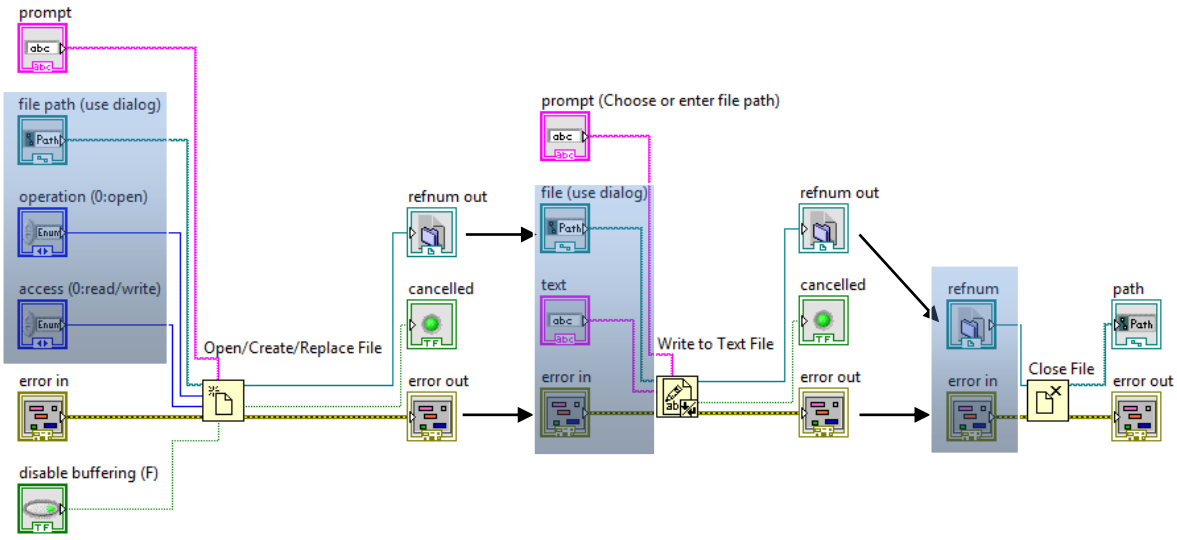
3. Zapisywać dane w postaci tekstowej do pliku.

Przykładowy pojedynczy wpis może mieć postać:

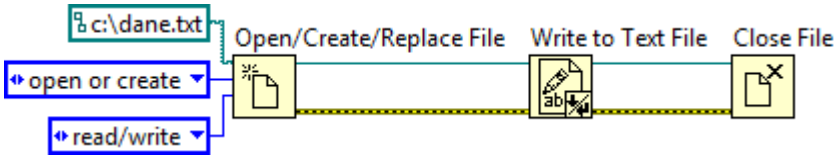
```
Data: 2007-04-10 Czas: 03:55 Temperatura: 29 stopni Celsjusza ALARM GÓRNY
```

Przydatne funkcje które można wykorzystać w rozwiązywaniu zadania:

- **Open/Create/Replace File, Write To Text File , Close File** (Functions -> File I/O)



Rysunek 5. Funkcje wykorzystywane do realizacji zapisu do pliku. Wyróżnione zostały te wejścia, które należy wykorzystać w zadaniu. Strzałki wskazują na niezbędne połączenia.

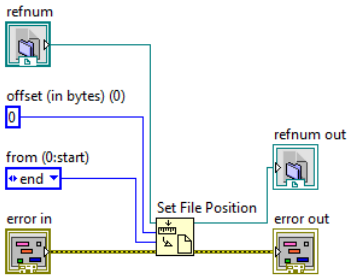


Rysunek 6. Przykładowa konfiguracja funkcji do zapisu danych w pliku.

Schemat działania funkcji operujących na pliku jest następujący:

- otwarcie pliku z odpowiednimi atrybutami (rysunek 5 i 6) **Open/Create/Replace File**;
- warunkowy ciągły zapis danych (w strukturze **Case** w głównej pętli **while** programu) realizowany za pomocą funkcji **Write to Text File**;
- po zakończeniu zapisu wykonywana jest funkcja **Close File**

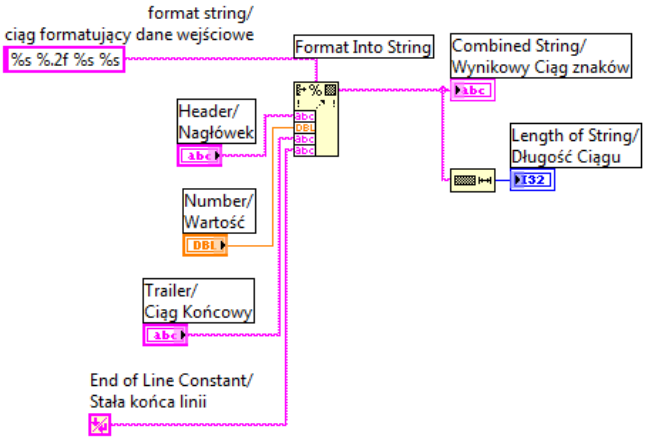
W programie może również okazać się przydatna funkcja **Set File Position** (rysunek 7). Zadaniem tej funkcji jest prawidłowe ustawienie znacznika wstawiania tekstu. Domyślnie jest on ustawiony na początek pliku i od tego miejsca wprowadzane są nowe dane. Jeżeli w programie przyjmie się zasadę każdorazowego otwierania i zamykania pliku przy pojedynczym wpisie, wtedy funkcja ta okaże się szczególnie przydatna. Jeżeli zaś plik będzie otwierany jednokrotnie na początku działania programu i pozostanie otwarty przez cały czas aktywności programu użycie funkcji **Set File Position** zależy od tego czy nowe dane będą dołączane na koniec czy też będą nadpisywały dane już istniejące.

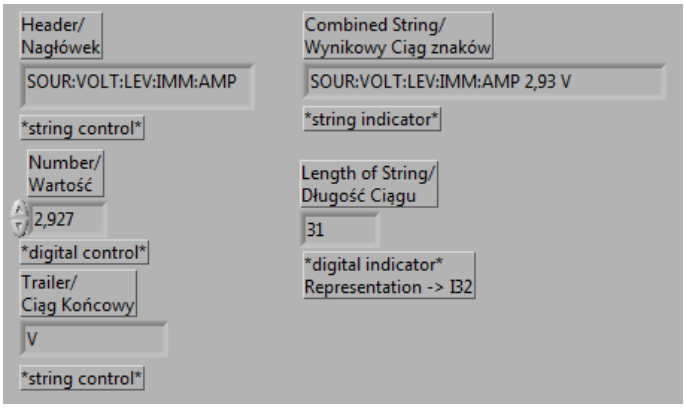


Rysunek 7. Funkcja **Set File Position** wraz ze wstępnie ustawionymi parametrami wejściowymi.

Pozostałe funkcje i elementy, które warto wykorzystać w programie:

- **Format Into String** (Functions -> String) - Rysunek 8.
- stała **End Of Line** (Functions -> String)
- stała **String Constant** (Functions -> String)

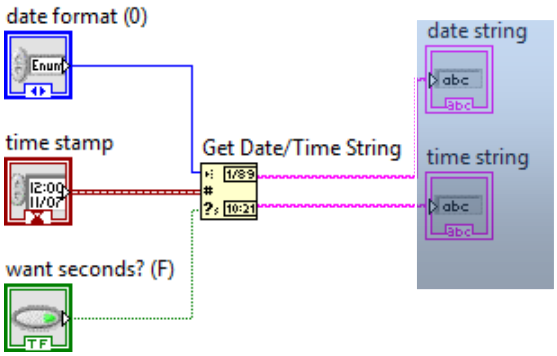




Rysunek 8. Przykładowe "wywołanie" funkcji **Format Into String** wraz z ilustracją wyniku działania. Liczba wejść może być zmieniana w zależności od potrzeb (typy danych wejściowych również). Stała **End of Line Constant** została wykorzystana, tak aby wpisy pojawiały się w kolejnych liniach.

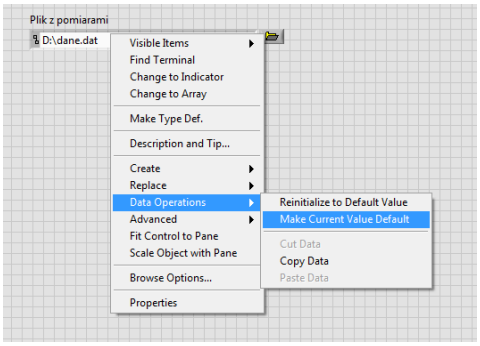
oraz

- **Get Time/Date String** (Functions -> Time & Dialog - rysunek 9) - zwraca ciągi tekstowe zawierające aktualną datę i czas. Parametr **want seconds?** warto ustawić na **True**.

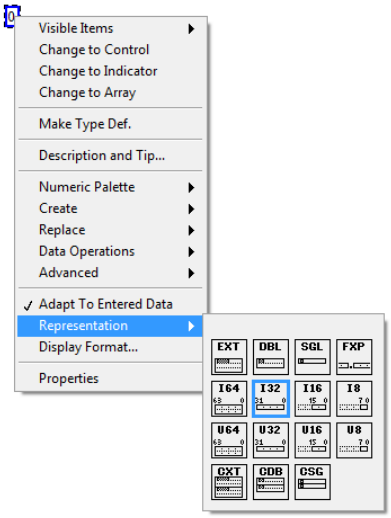


Rysunek 9. Przykład użycia funkcji **Get Date/Time String**.

Parametry pracy programu, takie jak adres komputera wykonującego pomiary, ścieżka do pliku warto zorganizować tak, aby można było je ustawiać i zmieniać na płycie czołowej. Dodatkowo też dobrze jest użyć własnych wartości domyślnych (rysunek 10).



Rysunek 10. Ustalanie własnych wartości domyślnych.



Rysunek 11. Zmiana typu danych dla stałej (również w ten sam sposób można zmienić przypisany typ dla elementów na płycie czołowej)