LabVIEW Execution Trace Toolkit

1. Pierwszy, prosty program.
   a) Tworzymy projekt, np.: Nazwisko_Proj_10_1, zawierający tylko jeden program (np. Nazwisko_10_target_1.vi), który ma działać na komputerze docelowym.
   b) W programie tworzymy jedną pętlę czasową o okresie 100 i nazwie P_1 podpiętą do **1MHz** źródła czasu, co oznacza, że pętla powinna się wykonywać co 100 µs. Pętla ma się wykonać czterokrotnie i zatrzymać. Nie zapomnijmy o przekazywaniu informacji o błędzie wewnątrz pętli.
   c) Za **pętlą czasową** wstawiamy **czasową strukturę sekwencyjną** i nadajemy jej nazwę S_1. Jest ona podpięta do tego samego źródła czasu, co pętla czasowa. W strukturze S_1 umieszczamy opóźnienie czasowe 3 ms. Tu również przekazujemy wewnątrz informację o błędzie. Struktura sekwencyjna ma się wykonywać po zakończeniu pętli czasowej.
   d) Podłączamy moduły śledzenia programu:
      – *Functions> Real-Time> RT Trace Tool> TraceTool Start Trace* – ma być tak podłączona, by wykonała się **po** module *Create Timing Source*, a **przed** pętlą czasową. Do wejścia *Detailed Tracing* podłączamy stałą logiczną *True*.
      – *TraceTool Stop Trace And Send* – na samym końcu. Do wejścia *Trace Host Network Address* podłączamy stałą z adresem komputera głównego: **169.254.148.100** Adres ten można odczytać w systemowym Centrum Sieci i Udostępniania w zakładce dedykowanej karcie sieciowej wykorzystywanej do połączenia z kasetą PXI (Połączenie sieciowe 7).
   e) Zapisujemy program Nazwisko_10_target_1.vi oraz uruchamiamy program pozwalający na oglądnięcie zarejestrowanych danych śledzenia. Znajduje się on zarówno w Menu okna naszego programu vi, jak i w Menu okna projektu w zakładce Tools > *Real-Time Module> Trace Viewer….* Na ewentualnie zadane pytania odpowiadamy twierdząco.
   f) Uruchamiamy program Nazwisko_10_target_1.vi. Po krótkiej chwili program się zatrzyma, a okno programu *Execution Trace Tool* wypełni się zebranymi danymi.
   g) Proszę zaobserwować tykanie zegara (OS Timer Thread), odległości pomiędzy uruchomieniami pętli P1, umiejscowienie na diagramie struktury S1, priorytety różnych procesów itd. Można zmienić takt wykonywania pętli czasowej ze 100 µs na 200, 50 lub inną wartość i powtórnie uruchomić program. Nowy diagram będzie znajdował się **pod** zakładkami sesji, znakowanymi kolejnymi czasami powstania.
   h) Zapisujemy i zamykamy niniejszy program i projekt.
2. Drugi program, który będzie zawierał podprogramy.
   a) Możemy skorzystać z pierwszego programu. Proszę wpierw na dysku przekopiować program Nazwisko_10_target_1.vi na Nazwisko_10_target_2.vi. Następnie należy stworzyć nowy projekt Nazwisko_Proj_10_2, do którego wstawiamy Nazwisko_10_target_2.vi – oczywiście, program umieszczamy w projekcie tak, by działał na kontrolerze PXI.

b) W istniejącej pętli czasowej (nazwa P_1, okres 100 µs, priorytet 50) wstawiamy generator sinusoidalny. Wyjścia sygnału nigdzie nie podpinamy – będzie generował *sobie, a muzom*. Żądamy zatrzymania pętli po upływie 1 ms po rozpoczęciu działania programu (korzystamy z bloczka *Tick Count* podobnie, jak na poprzednich zajęciach). Umieszczamy również dodatkowy znacznik (flagę) używając modułu *TraceTool Log User Event.* Numer znacznika (*Event ID)* jest dowolny z liczb między 0 i 255 (np. 1). Kabelek z informacją o błędzie przeprowadzamy przez generator oraz *TraceTool Log User Event.*

c) Z generatora sinusoidalnego, bloczka porównania (jest nam potrzebny do stworzenia sygnału stopu pętli) oraz modułu *TraceTool Log User Event* tworzymy podprogram, który nazywamy np. Nazwisko_10_sub_1.vi. Przypominam, że podprogram robi się tak, że się zaznacza elementy, które chcemy umieścić w podprogramie i następnie kilka w Menu diagramu blokowego w *Edit> Create SubVI*.

d) Dokładamy drugą pętlę czasową o nazwie P_2, z priorytetem 500, z okresem pętli dwukrotnie większym niż pętli P_1. Pętla ta również ma się zatrzymywać po 1 ms od momentu rozpoczęcia działania programu. Umieszczamy tu również *TraceTool Log User Event* – numer znacznika *Event ID* oczywiście inny (np. 2). Z bloczka porównania oraz *TraceTool Log User Event* tworzymy podprogram o nazwie np. Nazwisko_10_sub_2.vi.

e) Za pętlą P_1, tak jak poprzednio, znajduje się niezmieniona struktura S_1. Moduł *Stop Trace and Send* musi być umieszczony tak, by wykonywał się na samym końcu. Tu wygodnie jest skorzystać z modułu *Merge Errors*, który znajduje się w zakładce funkcji: *Programming >> Dialog & User Interface,* by połączyć linie błędu wychodzące z S_1 oraz P_2.

f) Obie pętle P_1 i P_2 oraz struktura S_1 powinny mieć podpięte to samo źródło czasu **1MHz**.

g) Uruchamiamy program Prac_10_target_2.vi. Na diagramie śledzenia obserwujemy zależności pomiędzy pętlami P_1, P_2 oraz S_1, kolory priorytetów im przydzielone oraz zależności czasowe. Odnajdujemy również programy Nazwisko_10_sub_1.vi i _sub_2.vi. Aby zobaczyć wstawione przez nas znaczniki (flagi użytkownika) 1 i 2, to w Menu panelu śledzenia *Edit> Configure Flags* należy własne flagi włączyć i przydzielić im różne kolory. Dla czytelności można odznaczyć wszystkie inne flagi. Proszę zmienić priorytety lub inne parametry pętli i zobaczyć efekty na diagramie.

3. <u>Inne działania</u>

– Można również otworzyć program przykładowy:

W pomocy kontekstowej najechać na moduł np. *Trace Tool Start Trace*.vi, kliknąć w *Detailed Help*, a w otwartym oknie kliknąć w *Open Example*. Wpisać do projektu poprawny adres komputera docelowego, obejrzeć diagram blokowy, uruchomić i obejrzeć wynik. Aby lepiej zobaczyć zależności pomiędzy procesami należy włączyć szczegółowe śledzenie działania programu i go uruchomić ponownie.