Model czasu rzeczywistego systemu zbiorników

Na Rys. 1 pokazany jest model *tanki_rtwt.mdl*, który umożliwia pomiary i sterowanie w czasie rzeczywistym systemem zbiorników. Model jest skonfigurowany do generacji kodu wykonywalnego czasu rzeczywistego z wykorzystaniem przyborników RTW i RTWT. Jest on bazą, na której można skonstruować dowolny układ regulacji dla układu zbiorników podczas zajęć laboratoryjnych.

W niniejszej instrukcji opisano szczegóły budowy i obsługi tego modelu czasu rzeczywistego.



Rys. 1. Schemat modelu czasu rzeczywistego zrealizowany w Simulinku

Przedstawiony model posiada bloki wejść i wyjść komunikujące model Simulinka z obiektem rzeczywistym poprzez kartę I/O RTDAC/PCI. Są dwa podstawowe bloki:

- blok pomiarów *Levels of Tanks* zawierający device driver obsługi kanałów A/C karty I/O,
- blok sygnałów sterujących *Power Output* zawierający device drivery kanałów PWM karty I/O.

Device drivery, czyli moduły obsługi funkcji karty I/O są S-funkcjami napisanymi w języku C umożliwiającymi czytanie rejestrów karty (czyli odczyt pomiarów) oraz wpisywanie do karty sygnału sterującego (w naszym przypadku sygnały sterujące to sygnały PWM).

Opisywane bloki używa się w ten sam sposób jak inne bloki biblioteczne Simulinka. Trzeba jeszcze zaznaczyć pewne różnice w stosunku do znanych nam modeli Simulinka. A mianowicie:

• przedstawiony model nie jest wykonywany w trybie symulacji *Normal,* jak zwykły model symulacyjny, ale musi być przed użyciem skompilowany do kodu czasu rzeczywistego za pomocą RTW i RTWT, i dopiero potem wykonany w trybie *External.*

• Model jest specjalnie skonfigurowany. Konfiguracja ta jest ukryta w opcjach okna modelu i będzie wyjaśniona w następnym punkcie.

Na Rys. 2 pokazano maskę device drivera obsługującego kanały A/C karty I/O. W polu *Channel* wybrane są trzy kanały 0,1 i 2 (ogółem karta posiada 16 wejść A/C). Każdy z wybranych kanałów ma ustawione wzmocnienie równe 1. Okres próbkowania jest równy *T0*. Czyli w przestrzeni roboczej Matlaba trzeba przypisać wartość zmiennej *T0*. Blok ten mierzy sygnały napięciowe z czujników ciśnienia umieszczonych w każdym z trzech zbiorników. Sygnały odczytywane z kanałów A/C są zawarte w zakresie [0÷10 V].

Block Parameters: RT-DAC Analog Inputs	×
Analog Input Module (mask)	
RT-DAC PCI Device Driver	
Parameters	
Base Address:	
mex_baseaddress(-1,-1)	
Channel:	
[012]	
Gain:	
[1 1 1]	
Sample Time (sec):	
ТО	
OK Cancel <u>H</u> elp <u>A</u> pply	

Rys. 2 Maska modułu pomiarów A/C

Ponieważ naszym celem jest pomiar poziomów wody w zbiornikach, mierzone sygnały muszą być przeliczone z [V] na [cm]. Zadanie to nosi nazwę "kondycjonowanie sygnałów". Blok *Subsystem1* realizujący przeliczanie jest pusty i musi być zaprojektowany przez użytkownika. Funkcję jego można opisać zależnością: h[cm] = ax[V] + b. Na Rys. 3 przedstawiono realizacje tego bloku. Należy jedynie dobrać parametry *a* i *b*.



Rys. 3 Kondycjonowanie pomiarów

Blok *Power Output* przekazuje sterowanie z modelu Simulinka do silnika pompy. Sygnał *PUMP PWM* wychodzący z suwaka *Pompa* jest bezwymiarowy i ograniczony do przedziału [0÷1]. Wartość tego sygnału odpowiada stopniowi wypełnienia (duty cycle) sygnału PWM, który fizycznie jest generowany przez logikę karty. Z kolei ten sygnał jest podawany na zewnętrzny moduł mocy i stąd przekazywany na silnik pompy.

Na Rys. 4 pokazana jest maska modułu PWM (jednego z czterech dostępnych na karcie RTDAC/PCI). Zauważmy, że pompa sterowana jest wyjściem z kanału trzeciego. Ponieważ zaznaczona jest opcja *Termination* i *Termination output* jest równe zeru oznacza to, że w momencie zakończenia działania modelu wyjściowa wartość kanału PWM ustawiana jest na zero. Czyli pompa zatrzymuje się.

Block Parameters: POMP PWM 3
PWM Module (mask)
RT-DAC PCI Device Driver
Parameters
IO Base Address:
[mex_baseaddress(-1,-1)]
Channel: 3
Channel Mode: 12 Bit
Channel Prescaler:
30
Set Outputs At Termination
Termination Outputs:
0
Sample Time (sec):
ТО
OK Cancel <u>H</u> elp <u>Apply</u>

Rys. 4 Maska modułu PWM sterującego pompą

Podsumowując, device drivery modelu zbiorników realizują dwa główne zadania:

- Udostępniają pomiary z czujników ciśnienia, które to sygnały są proporcjonalne do wysokości cieczy w zbiornikach, i przekazują je do komputera.
- Generują sygnał sterujący silnikiem pompy. Sygnał sterujący są bezwymiarowy i należy do przedziału [0÷1].

Projektowanie własnego modelu czasu rzeczywistego w środowisku Matlaba/Simulinka

Żeby zbudować system, który będzie działał w czasie rzeczywistym należy kolejno:

- zbudować w Simulinku model systemu sterowania używając dedykowanego dla zbiorników device driverów (jak to jest po polsku ?) oraz potrzebnych bloków bibliotecznych,
- wygenerować kod czasu rzeczywistego wybierając odpowiednią opcję w menu modelu (patrz Rys. 5) lub stosując skrót klawiaturowy *Ctrl+B*,
- uruchomić kod poprzez kliknięcie kolejno opcji w menu modelu: *Simulation/Connect to target* i *Simulation/Start real-time code*.



Rys. 5. Generacja kodu czasu rzeczywistego

Budowa modelu

Najprostszym sposobem zbudowania własnego modelu czasu rzeczywistego jest wykorzystanie jako wzorca dostępnego modelu *Tanki_rtwt.mdl*. Należy otworzyć model, zapisać go pod inną nazwą (np. My_System) i zmodyfikować. W prezentowanym na Rys. 6 modelu zaimplementowano regulator PID śledzący zadany (*Ref*) poziom w zbiorniku dolnym h_3 .

Budowa własnego modelu na bazie już istniejącego zapewnia, że wszystkie wewnętrzne opcje modelu zostaną właściwie ustawione. Te opcje są konieczne do poprawnego przeprowadzenia procesów generacji, kompilacji i linkowania kodu czasu rzeczywistego.

Użytkownik ma całkowitą dowolność przy projektowaniu własnego układu regulacji. Może użyć większości bibliotecznych bloków dostępnych w bibliotece Simulinka. Nie wolno mu jedynie usunąć device drivera, ponieważ model utraci połączenie z obiektem rzeczywistym



Rys. 6. Model układu regulacji w Simulinku

Chociaż nie jest to konieczne zalecane jest również pozostawienie oscyloskopu. Będzie on potrzebny do obserwacji zachowania się modelu. Właściwości oscyloskopu są dostępne w zakładce *Scope/Properties* (patrz Rys. 7). Zapamiętanie danych pomiarowych, w celu późniejszej obróbki off line, można wymusić zaznaczając opcję *save data to workspace*. Dane zostaną zapisane w zmiennej o nazwie wpisanej do okienka edycyjnego *Variable name*. Ten sposób jest jedyny dla pomiarów w czasie rzeczywistym, ponieważ normalnie używany blok *To Workspace* nie działa w RTWT.

Należy jeszcze zwrócić uwagę na ustawienie próbkowania. Jeżeli *Decimation* jest równe 1 to znaczy, że każda próbka jest rysowana na wykresie i równocześnie jest zapamiętywana w zmiennej. Ustawienie *Decimation* równe 10 oznacza, że jedynie co dziesiąta próbka jest zapamiętywana i wyświetlana.

🛃 'Poziomy' parameters	🛃 'Poziomy' parameters
General Data history Tip: try right clicking on axes	General Data history Tip: try right clicking on axes
Axes Number of axes: 3 floating scope	☑ Limit data points to last: 20000
Time range: 120	Save data to workspace
Tick labels: bottom axis only 💌	Variable name: Poziomy
Sampling Decimation 💌 1	Format: Structure with time
OK Cancel Help Apply	OK Cancel Help Apply

Rys. 7. Parametry bloku oscyloskopu

Sposób zbierania danych pomiarowych w czasie rzeczywistym określa się również w opcji *Tools/External Mode Control Panel*. Po kliknięciu klawisza *Signal Triggering* otworzy nam się okno pokazane na Rys. 8. Należy zaznaczyć blok oscyloskopu czyli *Poziomy* (tzn., że tam będą zbierane dane pomiarowe), ustawić *Source* jako manual, a *Duration* równe liczbie próbek, którą będziemy chcieli zapamiętać na wykresach w oscyloskopie. Wielkość ta nie powinna być mniejsza niż zadeklarowana długość bufora w bloku *Scope*. Należy także zaznaczyć opcję *Arm when connect to target*.

🜗 My_System: External Signal & Triggering		
Signal selection Block	Path	
X Poziomy	My System/Poziomy	I Select all Clear all € orn € off
		Trigger signal Go to block
Trigger Source: manual ✓ Mode: normal ▼ Duration: 10000 Delay: 0 ✓ Arm when connect to target	Trigger signal:	Port: 1 Element any
		Revert Help Apply Close

Rys. 8. Okno External Signal & Triggering

Proces generacji kodu czasu rzeczywistego

Kod jest generowany z użyciem Target Language Compiler (TLC) (patrz opis *Simulink Target Language*). Plik makefile jest używany do automatycznego budowania i załadowania zbiorów *.*obj* związanych z zastosowanymi driverami systemu rzeczywistego Rys. 9 przedstawia jak muszą być ustawione opcje dotyczące procesu tworzenia kodu czasu rzeczywistego, żeby proces ten nie generował błędów.

Configuration Parameters:	MySystem/Configuration (Active)					x
Select:	Simulation time					
Solver Data Import/Export	Start time: 0.0		Stop time: 9	39999		
Optimization	Solver options					
Sample Time	Type: Fixed-step	•	Solver: ode5	(Dormand-Prince)		-
Data Validity Type Conversion Connectivity	Periodic sample time constraint: Fixed-step size (fundamental sample time):	Unconstraine me): <mark>0.02</mark> s: SingleTaskin	ed			
Model Referencing	Higher priority value indicates high	er task prioritu	9			-
Hardware Implementation	Automatically bandle data transfer	s bobucon tooko				
Model Referencing Gradiene Workshop Comments Symbols Custom Code Debug Meal-Time Windows	Automatically handle data transfer	s between tasks				
Configuration Parameters	s: MySystem/Configuration (Acti	re)	<u>o</u> k	<u>C</u> ancel <u>I</u>	Help Apply	×
Select:	Target selection					
Solver	System target file: rtwin.tlc				Browse	
Optimization	Language: C				•	
⊡- Diagnostics	Description: Real-Time Wind	lows Target				
Sample Time Data Validity	Documentation					
Type Conversion	🔲 Generate HTML report					
···· Connectivity ···· Compatibility	Launch report automatically					
Model Referencing	Build process					
Model Referencing	TLC options:					
E-Real-Time Workshop	Makefile configuration					
- Symbols	🔽 Generate makefile					
- Custom Code	Make command: make_r	tw				
Debug Beal Time Windows	Template makefile: rtwin.tm	f				
	Generate code only				Build	
			к <u>(</u>	Cancel <u>H</u> e	elp Apply	

Rys. 9 Poprawnie ustawione parametry w opcji Simulation parameters

Plik o nazwie *make_rtw* zarządza procesem generacji. Plik *rtwintmf* to tzw. *template makefile*. Plik ten jest odpowiedzialny za generację kodu w C z użyciem zainstalowanego w systemie kompilatora Open Watcom.

Zakładka *Solver* pozwala ustawić parametry symulacji. Konieczne jest ustawienie opcji *Fixed Step* czyli stałego kroku próbkowania. Wartość kroku próbkowania należy ustawić równą 0.01 [s].

Uwaga – jeżeli w modelu używane są bloki dyskretne należy pamiętać, że bloki te i ustawiony krok próbkowania muszą mieć wspólny podzielnik.

Po ustawieniu wszystkich parametrów możemy wygenerować kod czasu rzeczywistego. W tym celu naciskamy klawisz *Build* w zakładce *Real Time Workshop* opcji *Simualtion/Simulation Parameters* lub przy aktywnym oknie modelu naciśniemy kombinację klawiszy Ctrl+B. Pomyślna generacja kodu kończy się informacją w oknie Matlaba:

Model My_System.rtd successfully created

Successful completion of Real-Time Workshop build procedure for model: My_System

Uruchomienie modelu czasu rzeczywistego

W celu uruchomienia kodu czasu rzeczywistego musimy kolejno kliknąć klawisz *Simulation/Connect to target* – kod zostanie załadowany do pamięci. Następnie należy kliknąć opcję *Run real-time code* co uruchamia działanie modelu w czasie rzeczywistym.

Klikając opcję *Stop* w menu okna modelu zatrzymujemy uruchomiony model w dowolnej chwili.