dr inż. Piotr Szymczyk, dr inż. Magdalena Szymczyk
Akademia Górniczo-Hutnicza, Katedra Automatyki

SYMULATOR SYSTEMU OPERACYJNEGO CZASU RZECZYWISTEGO

Tematem artykułu jest przedstawienie przykładów istniejących symulatorów systemów operacyjnych czasu rzeczywistego ze szczególnym uwzględnieniem symulatora VxSim firmy Wind River i symulatora zawartego w AVR Studio firmy Atmel oraz opis tworzonego w ramach prowadzonych badań symulatora systemu czasu rzeczywistego uCRTOS.

REAL TIME OPERATING SYSTEM SIMULATOR

The main aim of this article is presentation of existing real-time operating system simulators with special consideration of VxWorks simulator from Wind River corporation, AVR Simulator from Atmel corporation and simulator which is created during our research uCRTOS.

1. WPROWADZENIE

Pisząc aplikacje komputerowe, w tym aplikacje czasu rzeczywistego zawsze nadchodzi taki moment, że należy sprawdzić poprawność ich działania i dokonać niezbędne poprawki zanim oprogramowanie trafi na sprzęt docelowy. Etap testowania i poprawiania kodu jest etapem długotrwałym i żmudnym. Wymaga dużego doświadczenia od osób prowadzących te prace i jest dość kosztowny. Oczywiście skala tego etapu zależy od wielkości systemu oraz od roli, jaką będzie odgrywał.

W systemach czasu rzeczywistego, które w większości pełnią istotną i odpowiedzialną funkcję, szczególnie ważne jest zatem sprawdzenie ich poprawnego działania. Skutki błędów w takich systemach mogą mieć poważne konsekwencje. Źle funkcjonujący system w urządzeniach stosowanych w medycynie do naświetlań pacjentów promieniami X może spowodować ich ciężkie obrażenia. Innym znanym przykładem jest błąd w oprogramowaniu lądownika marsjańskiej misji Pathfinder, gdzie nie wykryto na etapie symulacji i testowania błędu w implementacji semafora wzajemnego wykluczania [2]. Spowodowało to awarię systemu i dopiero po analizie sytuacji wykryto problem, poprawiono oprogramowanie, przetransmitowano je z Ziemi na Marsa, uruchomiono i przywrócono system do prawidłowego działania. Spowodowało to opóźnienie realizacji misji, a gdyby cała operacja nie powiodła się, NASA poniosłaby znaczne straty finansowe i konieczne byłoby powtórzenie tej misji.

Zdarza się, że podczas tworzenia aplikacji, sprzęt docelowy powstaje równolegle. Nie istnieje więc możliwość uruchamiania i testowania oprogramowania na urządzeniach. Proces tworzenia oprogramowania wymaga czasu i im wcześniej rozpocznie się jego sprawdzanie tym szybciej i łatwiej można poprawić ewentualne błędy [9].

W celu ułatwienia tej pracy, przyspieszenia jej, a także w celu umożliwienia dokładniejszej weryfikacji czy zaimplementowany system działa właściwie, powstają narzędzia wspomagające projektantów i wykonawców systemów. Narzędziami takimi są między innymi symulatory systemów operacyjnych czasu rzeczywistego (SOCR).

Są one również często stosowane do celów dydaktycznych, gdyż uruchamianie aplikacji nie wymaga posiadania kosztownego sprzętu docelowego.

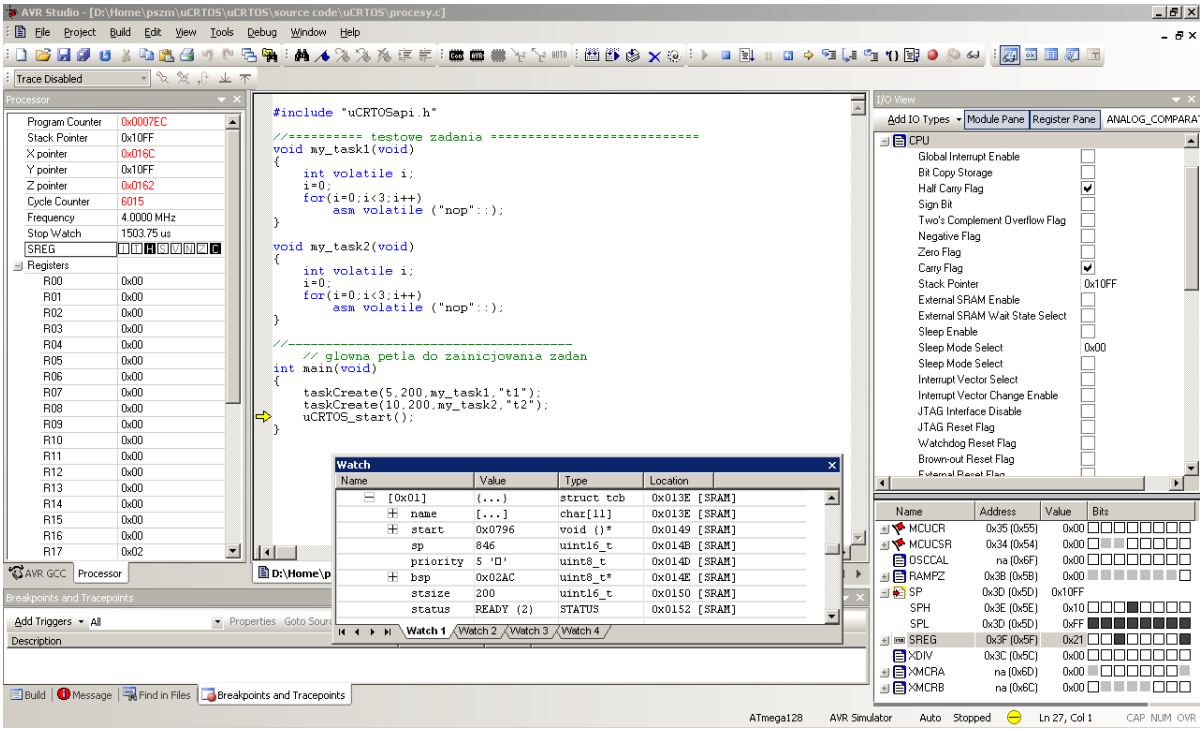
2. SYMULATORY

Istnieje szereg profesjonalnych [10] jak i darmowych symulatorów [3, 4, 5, 7, 8]. Niektóre z nich pozwalają symulować jedynie część funkcjonalności systemu operacyjnego, takie jak na przykład harmonogramowanie procesów, czy komunikację między nimi. Stosowanie symulatora nigdy nie może w pełni zastąpić sprzętu docelowego, ze względu na to, że symulator często nie potrafi idealnie odwzorować otoczenia systemu i środowiska docelowego. Niemniej jednak możliwość użycia symulatora jest cenna i często wykorzystywana w praktyce.

W dalszej części rozdziału przyjrzymy się dokładniej dwom wybranym symulatorom, AVR Studio firmy Atmel [1, 6] oraz VxSim firmy Wind River [10].

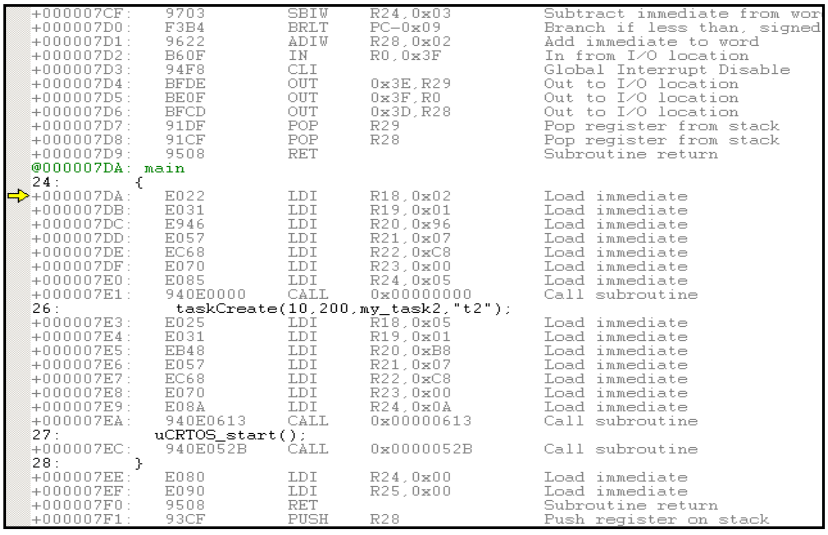
2.1. AVR Studio

Oprogramowanie AVR Studio firmy Atmel jest kompletnym środowiskiem do tworzenia i rozwijania aplikacji uruchamianych na mikrokontrolerach tego producenta. Zawiera wiele różnych dodatkowych narzędzi, między innymi symulator pracy mikrokontrolera. Na rys. 1 pokazane jest całe środowisko z uruchomionym symulatorem. W trakcie symulacji można obserwować postęp wykonania linii kodu, stany rejestrów procesora, podglądać zawartość poszczególnych interesujących nas obszarów pamięci oraz stany portów wyjściowych, a nawet zmieniać stany portów wyjściowych, wartości poszczególnych liczników i rejestrów, generować przerwania tak, aby maksymalnie odwzorować zachowanie zewnętrznego środowisko docelowego.



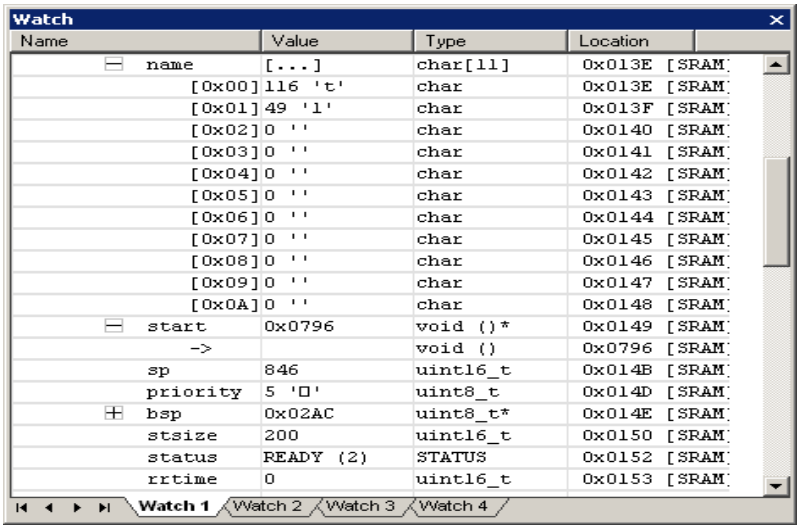
Rys. 1. Symulator środowiska AVR Studio

Ciekawą opcją jest również możliwość równoczesnej obserwacji kodu źródłowego napisanego w języku C oraz kodu wykonywalnego powstałego w wyniku kompilacji, co pokazano na rys. 2.



Rys. 2. Obserwacja kodu źródłowego w wynikowego w trakcie symulacji

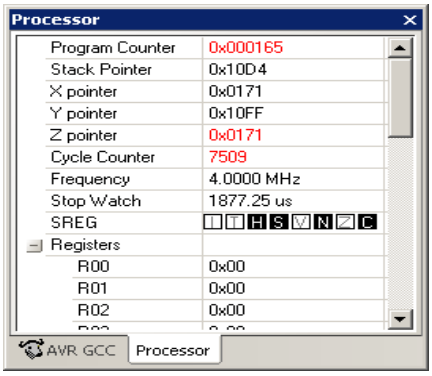
Szczególnie przydatna jest m ożliwość obserwacji zawarto ści pamięci, w której umiejscowione są dane na tem at procesów system u operacyjnego, co pokazano z kolei na rys. 3.



Rys. 3. Stan procesu o nazwie t1 obserwowany w trakcie symulacji

Na rysunku tym można zauważyć, że proces o nazwie t1, który m a aktualny priorytet o wartości 5 jest w stanie wykonywania (READY). Bardzo wa żnym parametrem systemów czasu rzeczywistego jest oczywiście czas. Jak wiadom o nie tylko poprawne wyniki oblicze ń, ale uzyskiwane ich w oczekiwanym momencie czasowym świadczą o poprawnym działaniu całego systemu. Dlatego te ż znajomość czasu wykonania poszczególnych elem entów oprogramowania jak i sam ych operacji systemu operacyjnego (takich jak prze łączanie kontekstu, czy propagacja kom unikatu) jest konieczna do weryfikacji poprawno ści działania takiego systemu.

Symulator AVR Studio um ożliwia pomiary czasu poprzez odczyt licznika cykli (ang. *cycle counter*), co jest możliwe dzięki obserwacji danych znajduj ących się podczas sym ulacji w oknie stanu procesora, które pokazano na rys. 4.



Rys. 4. Stan procesora podczas symulacji z widocznym licznikiem cykli

Do obliczenia czasu można użyć wzoru (1):

$$T_{ij} = \frac{CC_i - CC_j}{F_{proc}} \tag{1}$$

gdzie i oznacza punkt czasowy początkowy, j oznacza punkt czasowy końcowy, T_{ij} oznacza czas między punktem czasowym i, a punktem czasowym j, CC_i – wartość licznika cykli dla punktu czasowego i, CC_j – wartość licznika cykli dla punktu czasowego j, a F_{proc} to częstotliwość procesora.

Użycie środowiska AVR Studio jest proste i intuicyjne. Daje pełny pogląd na wykonanie aplikacji i pracę systemu operacyjnego, umożliwia pomiary czasów oraz emulację środowiska zewnętrznego. Wadą tego rozwiązania jest brak dodatkowych narzędzi umożliwiających wizualizację oraz automatyzację analizy pracy systemu.

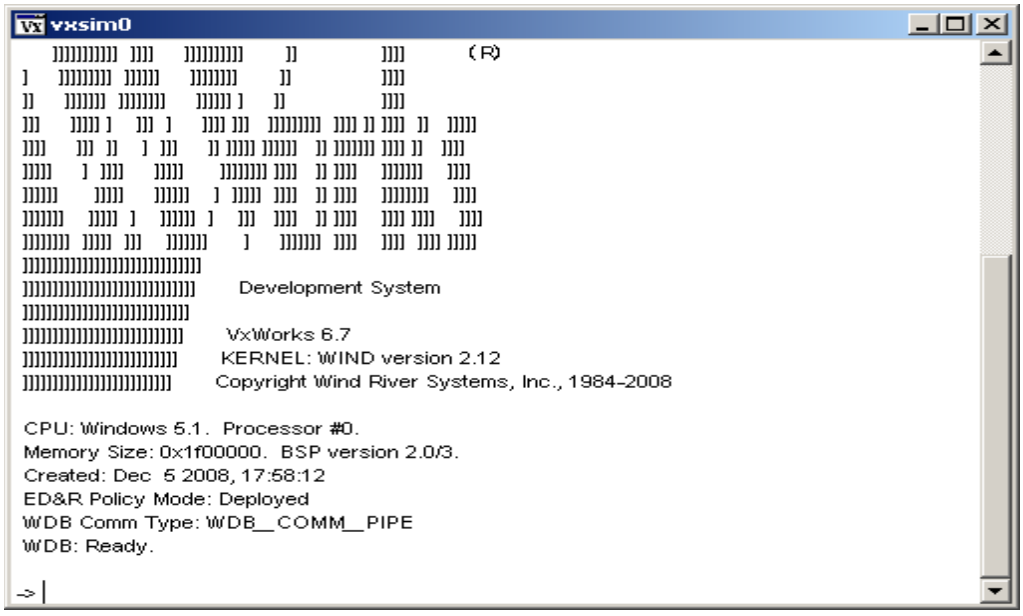
2.2. VxSim

Symulator VxSim [10] jest symulatorem systemu operacyjnego czasu rzeczywistego VxWorks firmy Wind River. Pozwala on na symulowanie pracy zarówno sprzętu docelowego oraz sieci komputerowej i aplikacji w niej pracujących. Ma zaimplementowaną również symulację wieloprocesorowości, NVRAM (ang. non-volatile RAM), MMU (ang. memory management unit), RTC (ang. real-time clock), docelowy system plików i wiele innych funkcjonalności sprzętu docelowego.

Symulator jest programem uruchamianym na komputerze pracującym pod konsolą innego systemu operacyjnego (Microsoft Windows, Linux, Solaris) na platformie innej niż sprzęt docelowy (na przykład na komputerze PC). Powoduje to pewne ograniczenia wynikające z innych niż na sprzęcie docelowym instrukcji maszynowych. Nie ma on zaimplementowanych sprzętowych funkcji zmiennoprzecinkowych, ale posiada zaimplementowaną emulację matematycznych funkcji zmiennoprzecinkowych. Są dostępne nastepujące funkcje: acos(), asin(), atan(), atan2(), cos(), cosh(), exp(), fabs(), floor(), fmod(), log(), log10(), pow(), sin(), sinh() i sqrt(). Nie można natomiast używać następujących funkcji zmiennoprzecinkowych: cbrt(), ciel(), infinity(), irint(), iround(), log2(), round(), sincos(), trunc(), cbrtf(), infinityf(), irintf(), iroundf(), log2f(), roundf(), sincosf(), truncf(), tan( ) tanh( ) oraz funkcji pojedynczej precyzji: acosf(), asinf(), atanf(), atan2f(), cielf(), cosf(), expf(), fabsf(), floorf(), fmodf(), logf(), log10f(), powf(), sinf(), sinhf(), sqrtf(), tanf(), tanhf().

Z kolei symulacja wieloprocesorowości jest bliższa rzeczywistości jeśli platforma na której jest uruchamiany sam symulator też jest wieloprocesorowa lub hiperwątkowa (ang. hyperthreading).

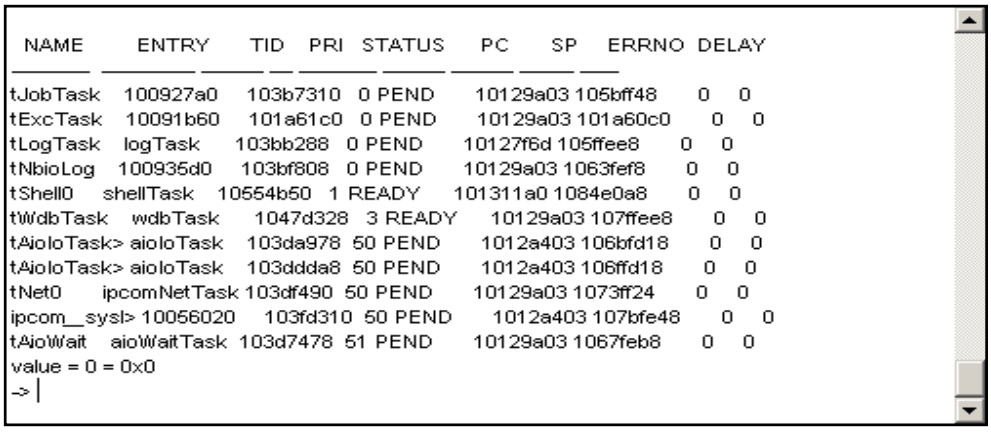
Po skonfigurowaniu i uruchom ieniu symulator zgłasza się otwierając okno konsoli, jak na rys. 5, umożliwiając wykonywanie poleceń tej konsoli i urucham ianie testowanej aplikacji. Możliwe jest, je śli zajdzie taka potrzeba, prze ładowanie (ang. *reboot*) symulatora dający identyczny efekt jak na sprzęcie docelowym.
Możliwe jest zalogowanie się do symulatora z systemu zewnętrznego poprzez otwarcie sesji terminalowej i pracę w linii komend.



Rys. 5. Konsola terminalowa symulatora VxSim

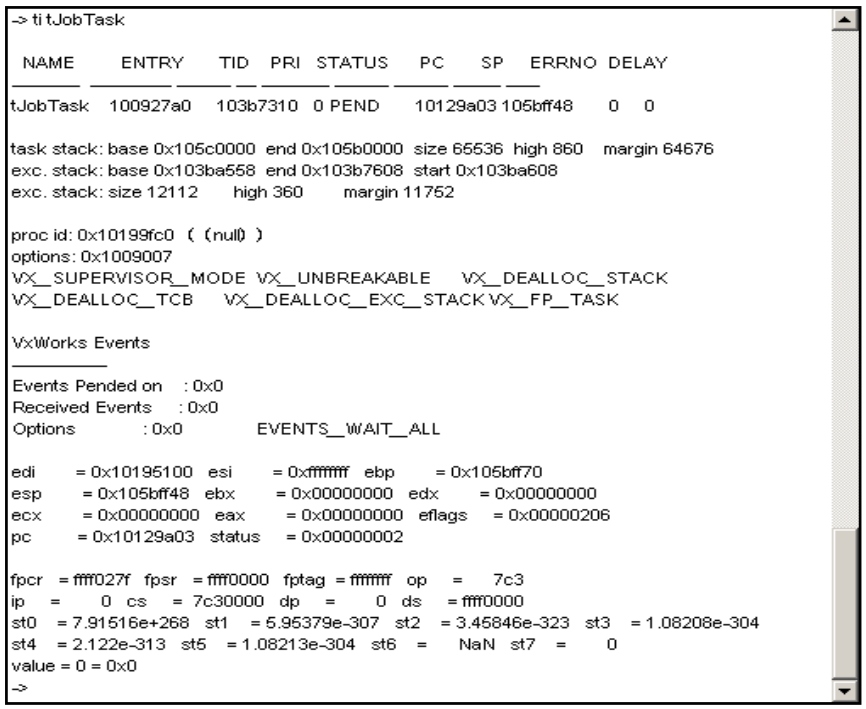
Mając uruchomiony symulator i dostęp do konsoli można wydawać komendy z linii komend. W pewnym stopniu ten tryb pracy przypom ina pracę z terminalem Uniksowych, podobne są nawet niektóre komendy takie jak na przykład pwd, ls, cp, mv.
Ważniejsze komendy dostępne w powłoce:

- help wypisuje listę komend z krótkim ich opisem,
- dbgHelp wypisuje informację na temat dostępnych opcji debugowania,
- edrHelp wypisuje informację na temat dostępnych opcji ED&R (ang. Error Detection and Reporting)
- ioHelp wypisuje informację na temat komend I/O,
- nfsHelp wypisuje informację na temat komend nfs,
- netHelp wypisuje informację na temat komend związanych z siecią,
- rtpHelp wypisuje informację na temat komend związanych z RTP
- spyHelp wypisuje informację na temat komend związanych histogramami zadań,
- timexHelp wypisuje informację na temat komend związanych z czasem wykonania procesów,
- h [n] pokazuje lub ustawia mechanizm historii powłoki,
- i [task] pokazuje zestawienie informacji o uruchomionych zadaniach (patrz rys. 6),
- ti task pokazuje informacje z TCB wybranego zadania (rys. 7),
- sp adr,args... uruchamia zadanie,
- taskSpawn name,pri,opt,stk,adr,args... uruchamia zadanie,
- td task kasuje zadanie,



Rys. 6. Wynik wykonania komendy i w powłoce symulatora

- ts task zadanie zostaje przesunięte w stan suspend,
- tr task zadanie przechodzi ze stanu suspend w stan ready,
- tw task pokazuje szczegółowe informacje na temat zadania znajdującego się w stanie pended,
- w [task] pokazuje zbiorczo informacje na temat wszystkich zadań znajdujących się w stanie pended,

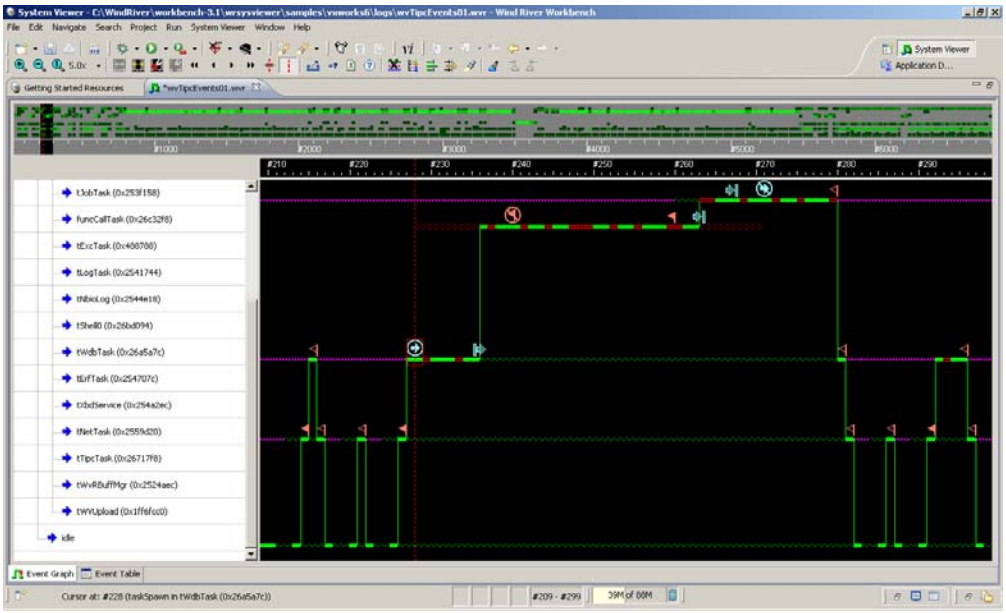


Rys. 7. Wynik wykonania komendy ti dla zadania o nazwie tJobTask w powłoce symulatora

- d [adr[,nunits[,width]]] pokazuje zawartośc pamięci,
- m adr[,width] modyfikuje pamięć o podanym adresie,
- pc [task] zwraca PC (ang. program counter) zadania,
- devs pokazuje listę dostępnych urządzeń,
- ld [syms[,noAbort][,"name"]] ładuje zadanie do pamięci,
- checkStack [task] pokazuje rozmiar i użycie stosu zadania,

- period secs,adr,args... uruchamia zadanie periodycznie,
- repeat n,adr,args... uruchamia zadanie n razy,
- version pokazuje wersję symulatora i systemu VxWorks,
- shConfig ["config"] pokazuje lub ustawia zmienne powłoki.

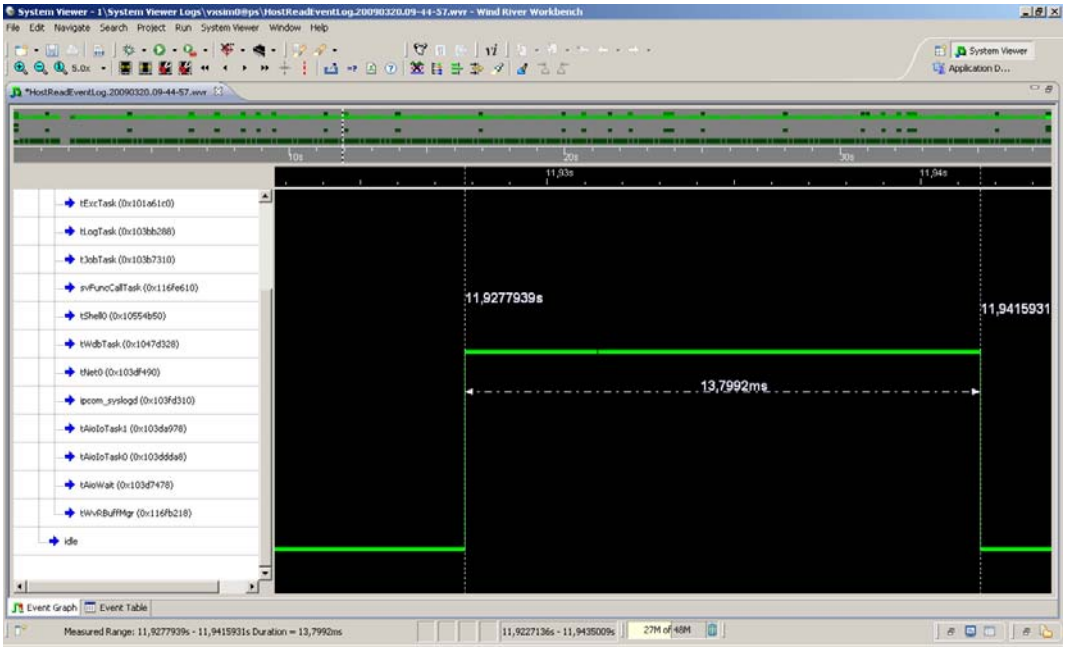
Argument task w powy ższych komendach może być określony zarówno poprzez nazw ę zadania jak i przez jego identyfikator.
Można również uruchamiać aplikacje bezpo średnio z zintegrowanego środowiska rozwoju aplikacji o nazwie Workbench. Takie rozwiązanie jest o wiele wygodniejsze i daje dodatkowe możliwości. W skład tego środowiska wchodzi równie ż oprogramowanie VxView, które z kolei umożliwia pomiary czasów oraz obserwacje wszystkich zdarze ń w system ie, stanów procesów, przełączanie kontekstu, obsługi przerwań i pętli jałowej.
Na rys. 8 pokazano wykres wizualizuj ący pracę aplikacji uruchom ionej na sym ulatorze VxSim.



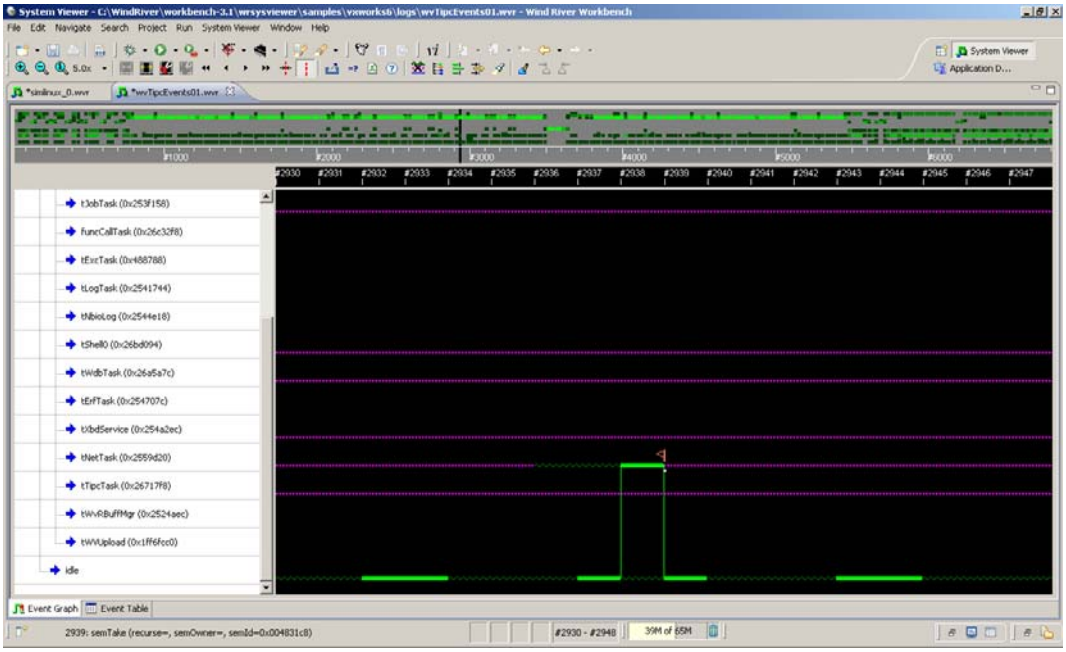
Rys. 8. Wizualizacja symulacji za pomocą modułu Data Monitor (VxView)

Symulator ten potrafi tak że zbierać informację na temat historii zdarzeń, stanu systemu oraz zadań w kolejnych krokach sym ulacji. Tworzy plik z histori ą symulacji do pó źniejszej jej analizy. Analiza taka m oże być przeprowadzona za pom ocą VxView lub innych narz ędzi dostępnych w środowisku Workbench takich jak Perf ormance Profiler, Code Coverage Analizer, Memory Analizer lub narzędzi innych firm takich jak CodeTest, StethoScope.
Rys. 8 w sposób poglądowy pokazuje zmiany stanów poszczególnych procesów, przełączanie kontekstu oraz zdarzenia, jakie m ają miejsce w obserwowanym czasie. Rys. 9 prezentuje możliwość wyznaczenia czasu wyst ąpienia danego zdarzenia oraz pom iaru przedziału czasu, z kolei na rys. 10 pokazano inspekcję interesującego nas zdarzenia.

Symulator VxSim wraz z dodatkowymi narzędziami stanowi bogate narzędzie umożliwiające przeprowadzenie kompleksowych testów i analiz w sytuacji gdy sprz ęt docelowy nie jest dostępny. Środowisko to jest również ciekawe pod względem dydaktycznym i badawczym.



Rys. 9. Pomiar czasu wykonany przy pomocy VxView

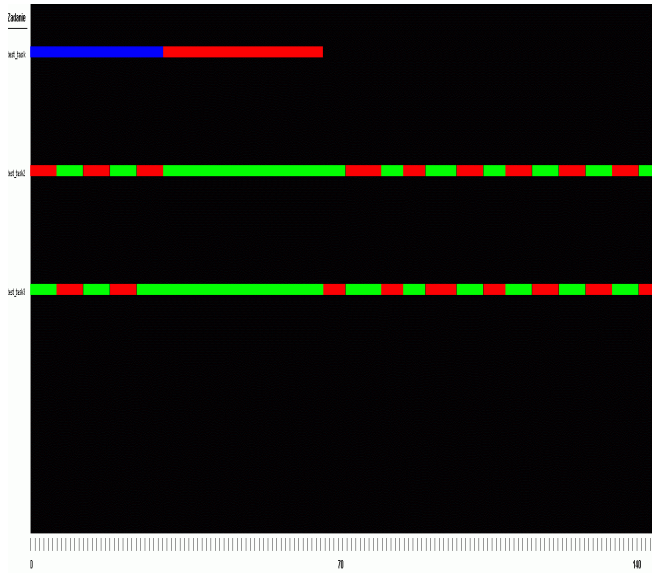


Rys. 10. Inspekcja zdarzenia wykonana za pomocą VxView

2.3. Symulator uCRTOS

Symulator systemu operacyjnego uCRTOS dzia ła na nieco odm iennej zasadzie ni ż opisane powyżej środowiska. Scenariusz sym ulacji zawierający parametry procesów oraz param etry symulacji są przygotowywane i zapisywan e w zbiorze dyskowym . Następnie symulator wczytuje dane, przeprowadza symulację i zapisuje jej wynik na dysku w postaci zbioru.

Symulowane są zmiany stanów procesów, ich okresowe uruchamianie oraz różne algorytmy szeregowania procesów w systemie. Nie są jeszcze w chwili obecnej w pełni zaimplementowane mechanizmy komunikacji między procesami.
Symulator ten jest intensywnie rozwijany i planowane jest wyposażenie go również w dodatkowe narzędzia umożliwiające zarówno analizę jak i wizualizację wyników symulacji.



Rys. 11. Przykładowy ekran z rozwijanego symulatora systemu uCRTOS

3. PODSUMOWANIE

Symulacja systemu operacyjnego czasu rzeczywistego i aplikacji na nim uruchamianych jest ważnym etapem tworzenia aplikacji. Oczywiście kolejny krok to uruchomienie aplikacji na sprzęcie docelowym. Zdarza się jednak czasem, że sprzęt docelowy nie jest dostępny w chwili, gdy należy rozpocząć testy, wtedy użycie symulatora pozwala na przezwyciężenie tego problemu. Symulatory mają również duże znaczenie w badaniach oraz dydaktyce. Jednym z najlepszych symulatorów systemu czasu rzeczywistego jest zaprezentowany powyżej VxSim symulujący pracę systemu VxWorks.

4. LITERATURA

[1] Baranowski R.: Mikrokontrolery AVR ATmega w praktyce BTC, Warszawa, 2005
[2] Cottet F., Delacroix J., Kaiser C., Mammeri Z.: Scheduling in Real-Time Systems, Wiley, NY, 2002
[3] http://carbonkernel.org
[4] http://cpuss.codeplex.com
[5] http://wsim.gforge.inria.fr/
[6] http://www.atmel.com/default.asp
[7] http://www.cise.ufl.edu/~fishwick/CPUDisk
[8] http://www.ontko.com/moss
[9] Szymczyk P.: Systemy operacyjne czasu rzeczywistego. AGH, Kraków,2003
[10] Wind River VxWorks Simulator USER'S GUIDE 6.7, Wind River Systems, Alameda, 2008