Mgr inż. Bartosz Trybus
Politechnika Rzeszowska

CHARAKTERYSTYKA SYSTEMU OPERACYJNEGO STEROWNIKA WIELOFUNKCYJNEGO PSW-166/CAN

W pracy opisano wielozadaniowy system operacyjny Extended RT zastosowany w prototypie sterownika wielofunkcyjnego PSW-166/CAN. Extended RT jest funkcjonalnym rozszerzeniem prostego i niedrogiego systemu operacyjnego czasu rzeczywistego RTX-Tiny firmy KEIL dla mikrokontrolera Siemens SAB 80166. RTX-Tiny został dodatkowo wyposażony w elementy wymagane w typowych zastosowaniach związanych ze sterowaniem, tj.: priorytety zadań, obszary krytyczne, komunikaty, obsługa pamięci dzielonej, sygnalizacja przekroczenia czasu zadania. Praca zawiera ponadto krótką charakterystykę sterownika PSW-166/CAN oraz opis struktury jego oprogramowania z podziałem na zadania.

CHARACTERISTIC OF PSW-166/CAN MULTIFUNCTION CONTROLLER OPERATING SYSTEM

The paper describes Extended RT multitask operating system, which has been used in PSW-166/CAN multifunction controller prototype. Extended RT is a functional extension of KEIL RTX-Tiny - a small inexpensive real-time operating system for Siemens SAB 80166 microcontroller. The system has been equipped with priority-based task switching, mailbox communications, shared memory mechanism, critical regions and task monitoring capabilities. They are necessary for typical control applications. The paper also includes brief characteristic of PSW-166/CAN controller. Its software structure, including task list, is also described.

1. WPROWADZENIE

Programowalne sterowniki logiczne (PLC) i regulatory procesowe stają się stopniowo coraz bardziej skomplikowane. Rozmiar oprogramowania nawet małych urządzeń zaczyna przekraczać 100kB. Wielozadaniowe systemy operacyjne, jak np. w modułowym regulatorze Digimatic firmy Hartmann-Braun [2], zastępują dotychczasowe jednopętlowe struktury programowe.

Dostępnych jest kilka systemów wielozadaniowych dla aplikacji wykorzystujących mikrokontrolery, jak (typ - producent): RTX - KEIL/Franklin, DCX – Intel, DCE – Iota Systems, MULTITASK! – US Software itd. [6]. W przypadku urządzeń produkowanych w małych seriach koszt zakupu komercyjnego systemu czasu rzeczywistego może stanowić duże obciążenie dla budżetu prac rozwojowych. Na przykład cena RTX-166 firmy KEIL w pełnej wersji wynosi około 10000 DM, podczas gdy pakiet kompilatora języka C kosztuje tylko około 4000 DM (a zawiera prosty system RTX-Tiny). Ponadto wiele możliwości dużych systemów pozostaje niewykorzystanych w niewielkich urządzeniach.
Warto więc rozważyć, czy dla niektórych zastosowań rozszerzenie prostego systemu, dołączonego bezpłatnie do pakietu kompilatora (jak w przypadku wspomnianego RTX-Tiny)

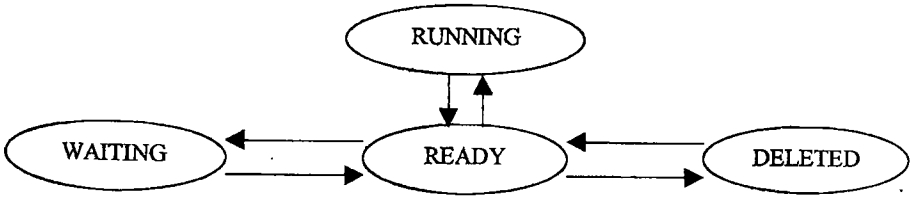
nie byłoby lepszym wyborem, niż zakup gotowego pełnowymiarowego produktu. Taki, „ekonomiczny" punkt widzenia umotywował prezentowane rozwiązanie.

Praca opisuje rozszerzenie wielozadaniowego systemu operacyjnego czasu rzeczywistego RTX-166-Tiny firmy KEIL, który jest zawarty w pakiecie kompilatora C166 [1]. Oprogramowanie KEIL jest szeroko znane i używane w całej Europie (szczególnie dla mikrokontrolerów rodziny 8051 i niektórych 16-bitowych). Opisywane rozszerzenie RTX-Tiny obejmuje mechanizmy komunikacji pomiędzy zadaniami, obsługę priorytetów w algorytmie przełączania zadań (*scheduler*) i kilka innych funkcji typowych dla systemów czasu rzeczywistego [4]. Rozszerzony system, nazywany dalej *Extended* RTX-Tiny (w skrócie Extended RT), został zaprojektowany dla sterowników wielofunkcyjnych i małych PLC. Naturalnie, może znaleźć zastosowanie w innych urządzeniach tej skali.

2. CHARAKTERYSTYKA SYSTEMU KEIL RTX-TINY

Wielozadaniowy system operacyjny czasu rzeczywistego RTX-Tiny KEIL jest częścią pakietu kompilatora języka C, który zawiera również podstawowe narzędzia programistyczne jak makroasembler, program łączący (*linker*), śledzący (*debugger*) i symulator. Istnieją wersje RTX-Tiny przeznaczone dla różnych platform sprzętowych (np. 8051 lub 80166). Praca niniejsza zajmuje się RTX166-Tiny, zaprojektowanym dla 16-bitowego mikrokontrolera 80166 [5].

RTX-Tiny udostępnia podstawowe mechanizmy wielozadaniowości jak przełączanie zadań według algorytmu karuzelowego (*round-robin*), ich synchronizacja, czy zawieszenie/wznowienie wykonywania zadania. Rozmiar jądra wynosi około 1400 bajtów. System obsługuje do 32 zadań z identyfikatorami 0-31. Zadania mogą przyjmować jeden z 4 stanów: RUNNING, WAITING, READY, DELETED, które oddziaływają jak na rys. 1.



Rys. 1 Oddziaływanie stanów zadań w RTX-Tiny

Każde zadanie może być zawieszone (w stanie WAITING) do momentu wystąpienia zdarzenia. RTX-Tiny obsługuje 3 typy zdarzeń:

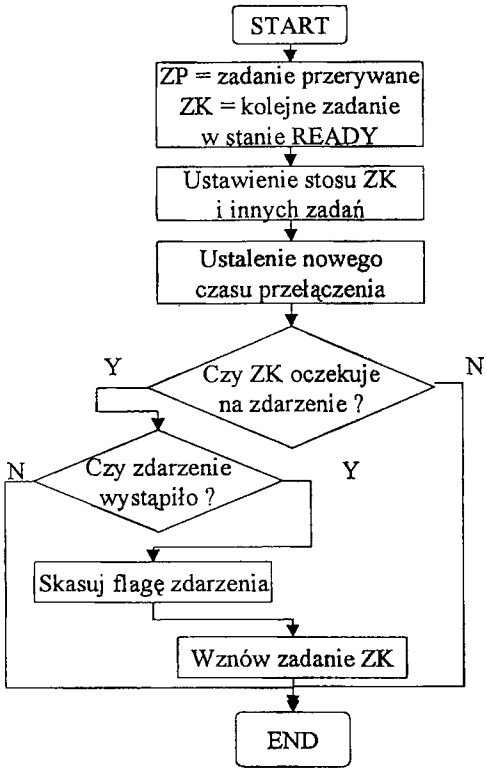
SIGNAL, TIMEOUT, INTERVAL.

Zdarzenie SIGNAL (sygnał) jest używane do synchronizacji zadań. Jedno z nich czeka, aż inne zadanie (lub przerwanie) wyśle sygnał. W systemie rozproszonym może on oznaczać, że jakieś zadanie przygotowało informację dla innego sterownika i w związku z tym należy uruchomić procedurę transmisji danych. Zdarzenie TIMEOUT jest generowane przez system operacyjny i wznawia zadanie, które zostało zawieszone na określony okres czasu. Ostatnie zdarzenie – INTERVAL – jest wykorzystywane do cyklicznego uruchamiania zadania.

Procedura przełączająca zadania (*scheduler*) przydziela zadaniom czas procesora zgodnie z algorytmem karuzelowym. Minimalny okres czasu, w którym zadanie wykonuje się bez przełączenia jest taki sam dla każdego zadania i konfigurowalny. Dla sterowników PLC może

on wynosić 2-5 ms. Podczas przełączania procedura wybiera zadanie, któremu zostanie przydzielony czas procesora. Algorytm jej działania przedstawiono na rys. 2.

Rys. 2 Uproszczony algorytm procedury przełączającej zadania w systemie RTX-Tiny



Po przerwaniu bieżącego zadania (ZP), procedura *scheduler*'a ustala identyfikator zadania, które ma być wznowione (ZK). Sprawdzane są kolejno zadania o następnych identyfikatorach, zaś wybrane zostaje pierwsze z nich, będące w stanie READY. Następnie ustawiane są wskaźniki stosów, obliczany czas kolejnego przełączenia oraz ewentualnie kasowane odpowiednie flagi zdarzeń. Procedura kończy się wznowieniem zadania ZK.

Zadanie o identyfikatorze 0 odgrywa specjalną rolę w systemie – po inicjalizacji systemu jest ono uruchamiane jako pierwsze. Musi w związku z tym utworzyć (aktywować) inne zadania związane z wykonywaną aplikacją. Procedura przełączająca traktuje je jednak jak pozostałe zadania (np. może zostać zawieszone).

Jedno z przerwań zegarowych, zwykle generowane przez *timer 0* jest zarezerwowane dla systemu. Obsługa tego przerwania odpowiada za aktualizację zmiennych systemowych związanych z zadaniami, ustawianiem flagi zdarzenia TIMEOUT i aktywowanie przełączenia zadań.

Podstawowe funkcje udostępniane przez RTX-Tiny wymieniono w tabeli 1. Parametr *task_id* oznacza identyfikator odpowiedniego zadania (0-31).

Tabela 1. Funkcje systemu operacyjnego RTX-Tiny

os_create_task(task_id)	Aktywowanie zadania – uwzględnienie go w przełączaniu karuzelowym (zadanie musi być zdefiniowane)
os_delete_task(task_id)	Wyłączenie (usunięcie) zadania z przełączania karuzelowego
os_wait(event)	Czekanie na zdarzenie: TIMEOUT, SIGNAL, INTERVAL
os_send_signal(task_id) isr_send_signal(task_id)	Wysłanie sygnału od jednego zadania (przerwania) do określonego zadania.
os_clear_signal(task_id)	Skasowanie flagi zdarzenia SIGNAL

Zadania są definiowane podobnie do standardowych funkcji języka C, lecz ze specjalnym słowem kluczowym *_task_*, np.:

```
#define PLC     1     /* Zadanie PLC ma identyfikator 1 */
plc() _task_ PLC      /* Konieczne słowo _task_ */
{
...                   /* deklaracje zmiennych lokalnych
                         oraz instrukcje zadania */
}
```

3. ZAŁOŻENIA I CECHY EXTENDED RT

Podczas tworzenia nawet skromnych aplikacji związanych z automatyką, jak oprogramowanie sterowników PLC lub regulatorów wielofunkcyjnych (spełniające jednak obecne wymagania) można szybko zdać sobie sprawę, że standardowe mechanizmy RTX-Tiny są niewystarczające. W takim przypadku istnieją trzy sposoby rozwiązania problemu: 1) zastosowanie pełnego systemu RTX-166, 2) zaprojektowanie i implementacja nowego systemu operacyjnego, 3) rozszerzenie możliwości wersji „Tiny". Jak już wspomniano, pełna wersja systemu może okazać się zbyt kosztowna dla aplikacji związanych z produkcją urządzeń w małych seriach. Druga opcja wymaga dłuższego czasu na prace rozwojowe oraz wysokiej klasy specjalistów, którzy są często niedostępni. W związku z tym rozszerzenie istniejącego, mniejszego systemu, zakładając, że jest w ogóle możliwe, wydaje się być interesującym rozwiązaniem. To rozwiązanie daje projektantowi swobodę decyzji, o które cechy i w jakim zakresie system powinien być rozszerzony oraz uwzględnić ograniczenia budżetowe, czasowe i sprzętowe [3].

Przy projektowaniu rozszerzenia należy zwrócić uwagę na ograniczenia RTX-Tiny. Prawdopodobnie najistotniejszy jest brak priorytetów zadań. Wszystkie zadania mają równe prawa i obsługiwane są przez procedurę przełączania karuzelowego w taki sam sposób. W zastosowaniach związanych z automatyczną regulacją, potrzebny wydaje się być mechanizm umożliwiający wykonywanie zadań na różnych poziomach ważności (priorytetach).
System nie pozwala na to, aby sekwencja instrukcji wykonywana była bez przełączeń. Potrzeba takiej cechy pojawia się na przykład podczas rekonfiguracji *on-line* sterowania PLC. Zablokowanie przerwań, w celu zakończenia odpowiednich instrukcji jest w wielu zastosowaniach niedozwolone.
Innym problemem jest komunikacja miedzy zadaniami. Gdy rozpatrywane są tylko dwa zadania, RTX-Tiny może zawiadomić jedno z nich o nadejściu komunikatu za pomocą zdarzenia SIGNAL. Nie jest jednak łatwo wysyłać komunikaty do jednego zadania z poziomu dwóch (lub więcej) innych procesów. Problemy pojawiają się również przy korzystaniu ze wspólnych danych o większym rozmiarze, które nie mogą być jednocześnie aktualizowane i odczytywane przez kilka zadań.
Ostre ograniczenia czasu rzeczywistego są często narzucone w zastosowaniach związanych z automatyką - na przykład niektóre zadania muszą zakończyć działanie w ciągu ustalonego okresu czasu. RTX-Tiny nie śledzi wykonywania zadań, w związku z czym błędy przekroczenia czasu nie są sygnalizowane
Należy zwrócić uwagę, że ograniczenia sprzętowe mikrokontrolerów są naturalną barierą rozpatrywanego rozszerzenia. Oprogramowanie PLC, komunikacja w systemach rozproszonych itp. wymagają krótkich czasów odpowiedzi. Kod systemu operacyjnego powinien być na tyle zwięzły, aby zminimalizować czas potrzebny na przełączanie zadań. Niektóre cechy dużych systemów operacyjnych po prostu nie mogą być zaimplementowane.

Uwzględniając powyższe rozważania, rozszerzenie Extended RT obejmuje następujące cechy:

1 *Priorytety* dla rozróżnienia zadań według ważności (pilności wykonania). Zastosowane rozwiązanie jest kombinacją algorytmów karuzelowego i wywłaszczającego (bazującego

na priorytetach). Po aktywacji i podczas wykonywania procesów o wyższym priorytecie, zadania niższe są zawieszone (wywłaszczone). Zadania o tym samym priorytecie przełączane są (jak dotychczas) według algorytmu karuzelowego.

2. *Sekcje krytyczne* dla ochrony wykonywania sekwencji instrukcji przed przerwaniem przez inne zadanie. Przełączanie zadań pozostaje zablokowane, gdy wykonywane są polecenia sekcji.
3. *Komunikaty* dla wymiany informacji pomiędzy zadaniami. Wysłany komunikat jest umieszczany w kolejce komunikatów („skrzynce pocztowej") skąd zadanie docelowe może go pobrać.
4. *Dzielony obszar pamięci* dla zapobieżenia jednoczesnemu zapisowi i odczytowi dużej ilości danych. Mechanizm ten chroni proces odczytu lub zapisu, odrzucając inne żądania dostępu do obszaru.
5. *Kolejka komunikatów o przekroczeniu czasu* wypełniana jest przez system, gdy zadanie wykonuje się dłużej niż ustalono. Użytkownik może zdefiniować procedurę ochronną, monitorującą komunikaty pojawiające się w kolejce.

Proponowane rozwiązanie dostarcza wyłącznie podstawowych mechanizmów systemu czasu rzeczywistego. W rezultacie uzyskiwany jest jedynie niewielki wzrost rozmiaru jądra systemu i małe różnice charakterystyk czasowych w stosunku do RTX-Tiny.

4. SZCZEGÓŁY ROZSZERZENIA

Procedura przełączająca zadania została nieco zmodyfikowana, w celu uwzględnienia obsługi *priorytetów*. Podczas konfiguracji, użytkownik ustala początkowy poziom priorytetów (0-4) dla każdego zadania. Jako pierwsze po przełączeniu zostaje wznowione zadanie o najwyższym priorytecie (wyższy niż dotychczas wykonywane), będące w stanie READY. Gdy nie ma takich zadań, następuje przełączenie na kolejne o takim samym priorytecie, jak dotychczas wykonywane. Oznacza to, że zadania z takim samym priorytetem przełączane są w sposób karuzelowy. Na przykład, zadanie z priorytetem 1 (regulacja PID) zostaje zawieszone, gdy aktywowane jest zadanie o priorytecie 2 (PLC). Zadanie 1 może być wznowione dopiero wtedy, gdy zadanie 2 i inne zadania o wyższym priorytecie ukończą swe działanie lub zostaną zawieszone.
Priorytety mogą być dynamicznie odczytane i zmienione za pomocą funkcji:

- `SC_GET_PRIORITY(task_id)` – odczyt priorytetu zadania
- `SC_SET_PRIORITY(task_id, new_priority)` – ustawienie nowego priorytetu dla zadania.

Gdy istnieje potrzeba ochrony sekwencji instrukcji w zadaniu przed przerwaniem przez inne zadanie, można użyć *sekcji krytycznej*. Krytyczne instrukcje muszą być zawarte pomiędzy funkcą SC_REGION_START, a SC_REGION_END, np.:

```
if (!SC_REGION_START(timeout))
{
     chronione instrukcje
     SC_REGION_END();
}
else
     brak dostępu do obszaru
```

Parametr *timeout* (występujący również w kolejnych funkcjach) określa czas oczekiwania, gdy obszar jest niedostępny. Można podać 0 (brak oczekiwania), $FFFF (czekanie

bezwarunkowe) lub inną wartość z tego zakresu. Użycie obszaru krytycznego powinno być ograniczone do sytuacji wyjątkowych, zaś sekwencja jak najkrótsza.

Komunikacja w Extended RT opiera się na *przesyłaniu wiadomości*. Odpowiednie funkcje zestawiono w tabeli 2. Zadanie (lub przerwanie) wywołuje funkcję SC_SEND_MESSAGE(), która umieszcza bajty komunikatu w określonej kolejce. Parametr *timeout* określa czas oczekiwania, gdy kolejka jest pełna. Wiadomość jest w niej przechowywana, dopóki zadanie docelowe nie wykona instrukcji SC_GET_MESSAGE(). W tym przypadku *timeout* określa maksymalny okres czekania na komunikat, gdy kolejka jest pusta.

Tabela 1. Funkcje komunikacji między zadaniami w Extended RT

`SC_SEND_MESSAGE(queue_id, message_buffer, timeout)`	Wysłanie komunikatu do kolejki
`SC_GET_MESSAGE(queue_id, message_buffer, timeout)`	Odebranie komunikatu z kolejki
`SC_QUEUE_STATE(queue_id)`	Odczyt stanu kolejki
`SC_QUEUE_EMPTY(queue_id)`	Sprawdzenie, czy kolejka jest pusta
`SC_QUEUE_FULL(queue_id)`	Sprawdzenie, czy kolejka jest pełna

Identyfikator kolejki *queue_id* występuje jako parametr każdej z funkcji. Dla prostoty założono, że tylko jedno zadanie może odczytać komunikat z danej kolejki, lecz każde z zadań może wysyłać wiadomości do wszystkich kolejek. Na przykład, zadanie wyświetlające informacje na panelu czołowym regulatora może wykorzystywać dwie kolejki, QUEUE1 i QUEUE2. Inne zadania wysyłają komunikaty do jednej z kolejek, zależnie od ich ważności. Niech QUEUE1 przechowuje pilniejsze wiadomości (np. alarmy) niż QUEUE2. W takiej konfiguracji, wiadomości z kolejki QUEUE2 zostaną wyświetlone dopiero wtedy, gdy kolejka QUEUE1 jest pusta.

Dzielony obszar pamięci jest drugim sposobem wymiany danych. Podczas konfiguracji użytkownik ustawia rozmiar wspólnej pamięci, wykorzystywany do współdzielenia zmiennych. Funkcje operujące na tym obszarze są następujące:

- `SC_GET_COMMON(data_offset, data_length, buffer, timeout)` – odczyt danych z pamięci dzielonej
- `SC_SET_COMMON(data_offset, data_length, buffer, timeout)` – zapis danych do pamięci dzielonej

Dostęp do zmiennej w obszarze dzielonym jest chroniony. Nie można zrealizować zapisu, gdy inne zadanie dokonuje odczytu zmiennej. Parametr *timeout* określa czas oczekiwania na zwolnienie obszaru.

Dodatkowo, w celu wymiany danych użytkownik może zastosować standardowe dla języka C zmienne globalne (np. do przechowywania flag i statusów). Należy jednak pamiętać, że ich rozmiar nie powinien przekraczać jednego lub dwóch bajtów. Zagwarantuje to, że dostęp d nich zostanie zrealizowany za pomocą jednej nieprzerywalnej instrukcji procesora.

Extended RT posiada również podstawowe możliwości *monitorowania zadań*. Podczas konfiguracji systemu można ustawić maksymalny czas wykonywania poszczególnych zadań (np. 25ms dla sterowania logicznego i 0.2sek dla regulacji PID). Gdy faktyczny czas wykonywania zadania przekroczy ustaloną wartość, system wysyła komunikat do specjalnej

kolejki, wskazujący na przekroczenie czasu. Jedno z zadań może monitorować stan tej kolejki i obsługiwać ewentualne błędy krytyczne.

Extended RT może być *skalowany i dopasowywany* do aktualnych potrzeb. Podczas konfiguracji, użytkownik ustawia pewną liczbę parametrów systemu (liczba kolejek komunikatów, długość komunikatu, rozmiar pamięci dzielonej itp.). Można również wybrać mechanizmy wymagane dla konkretnej aplikacji, a wyłączyć niepotrzebne. W ten sposób zmniejsza się rozmiar jądra i co za tym idzie – wzrasta prędkość operacji systemowych. Na przykład można utworzyć system z komunikacją miedzy zadaniami, lecz bez pamięci dzielonej i monitorowania zadań.

5. ZASTOSOWANIE W STEROWNIKU PSW-166/CAN

Extended RT został zaprojektowany dla sterownika PSW-166/CAN, którego prototypy są w fazie testów. PSW-166 jest wielofunkcyjnym regulatorem o wymiarze DIN, programowanym w języku bloków funkcyjnych FBD (IEC 1131). Może wykonywać zarówno sterowanie logiczne, jak i regulację ciągłą. Wersja standardowa zawiera 8 wejść i wyjść analogowych oraz 20 wejść i 16 wyjść binarnych. Istnieje też wersja binarna (40 WE i 30 WY). Sterownik ma dwa tryby działania: *off-line* (programowanie) i *on-line* (realizacja sterowania). Programowanie (inaczej *konfiguracja*) może być przeprowadzane zarówno z panelu czołowego, jak i komputera PC (za pomocą pakietu do graficznej konfiguracji). Za pośrednictwem szybkiej magistrali CAN (*peer-to-peer*), PSW może wymieniać informacje z innymi sterownikami (maks. 15). System rozproszony oparty o CAN obsługuje do 1000 sygnałów procesowych i nadaje się do automatyzacji średniej wielkości obiektów (elektrownie, miejskie urządzenia grzewcze itp.). Niezawodność takiego systemu jest większa niż jednego „dużego" sterownika, gdyż awaria jednego z urządzeń nie zatrzymuje pracy innych. Oprócz magistrali CAN, każdy PSW jest wyposażony w kanał RS-485 do komunikacji MODBUS (*master-slave*) z komputerem centralnym. Płyta główna zawiera mikrokontroler Siemens 80C166, zapewniający relatywnie dużą moc obliczeniową dla sterowania i komunikacji. Płyta analogowa jest wyposażona w procesor 80C32. Obie płyty komunikują się za pośrednictwem magistrali I^2C.

W stanie *off-line* aktywne jest jedynie zadanie konfiguracyjne CONF. W trybie *on-line* aktywowane są następujące zadania:

- PLC – sterowanie logiczne i sekwencyjne (cykl 10-20 ms, arytmetyka całkowitoliczbowa)
- REGULATOR – pętle PID, filtracja, samonastrajanie, termodynamika itp.. (0.2-0.4 sek, obliczenia zmiennoprzecinkowe)
- FACEPLATE – przyciski, wyświetlacze, linijki LED, pojedyncze LEDy (0.25 sek)
- CAN – komunikacja pozioma (*peer-to-peer*, 15 węzłów; 0.2 sek)
- MODBUS – komunikacja pionowa (*master-slave*, aktywowana na żądanie)
- PARAMETERS – współczynniki bloków funkcyjnych ustawiane *on-line* (stałe PID, na żądanie)
- TEST – pamięci NOVRAM i EPROM (zawsze aktywne).

6. PODSUMOWANIE

W pracy opisano rozszerzenie systemu operacyjnego RTX-Tiny KEIL, zawartego w pakiecie kompilatora C. System został uzupełniony między innymi o priorytetowy algorytm przełączania zadań, mechanizmy wymiany danych z wykorzystaniem kolejek komunikatów i

obszaru pamięci dzielonej. Extended RT, choć zaprojektowany początkowo dla regulatora aparatowego, wydaje się na tyle uniwersalny, by znaleźć inne zastosowania o podobnej złożoności.

Pomimo większej funkcjonalności, system pozostaje wciąż *tiny*, ponieważ kod jądra zajmuje ok. 2500 bajtów. Średni czas operacji systemowych wzrósł nieznacznie. Dostępne opcje konfiguracyjne pozwalają dopasować system do aktualnych potrzeb (skalowalność). Zastosowanie standardowych elementów spotykanych w systemach czasu rzeczywistego ułatwia specyfikację oprogramowania za pomocą narzędzi opisu wyższego poziomu, jak np. języka LACATRE [7]. Podział elementów aplikacji na zadania realizujące różne funkcje sprawia, że łatwiejsze i szybsze staje się tworzenie i testowanie oprogramowania w zespole projektowym.

Extended RT znalazł zastosowanie w regulatorze wielofunkcyjnym PSW-166/CAN. W związku z możliwością współbieżnego wykonywania zadań, PSW realizuje sterowanie logiczne i regulację PID z różnymi prędkościami, co daje dużą elastyczność w zastosowaniu (wcześniejsze 8-bitowe wersje PSW były krytykowane za jej brak).

PODZIĘKOWANIE
Autor dziękuje za wsparcie KBN Warszawa w postaci Grantu 8 T11A 023 98 C/3849.

LITERATURA

[1] *80C166 Utilities for Professional Developers Kit. User's Guide*; KEIL ELEKTRONIK / Franklin Software.

[2] *DIGIMATIC. Hardware Overview*; Hartmann & Braun, 1995.

[3] Halang A.W., Sacha K.M.: *Real-time Systems. Implementation of Industrial Computerised Process Automation*; World Scientific, 1992.

[4] Laplante P.: *Real-Time Systems Design and Analysis*; IEEE Computer Society Press, 1993.

[5] *SAB 80C166/83C166 16-Bit CMOS Single-Chip Microcontrollers for Embedded Control Applications, User's Manual*; Siemens.

[6] Schultz T.W.: *C and the 8051 Programming for Multitasking*; PTR Prentice Hall, 1993.

[7] Schwarz J.J. and Skubich J.: *Graphical Programming for Real Time Systems*; IFAC Control Engineering Practice, Vol. 1, No.1, 1993.

[8] Trybus L.: *Regulatory wielofunkcyjne*; WNT, Warszawa 1992.