STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Zbigniew BIGEWSKI, Rafał CUPEK, Andrzej KWIECIEŃ
Politechnika Śląska, Instytut Informatyki

STOSOWANIE PROCEDUR JĘZYKA C W CELU ZWIĘKSZENIA CZĘSTOTLIWOŚCI DOSTĘPU DO SIECI KOMUNIKACYJNEJ ABONENTÓW SYSTEMU ROZPROSZONEGO

Streszczenie. W pracy przedstawiono problem częstości dostępu do sieci komunikacyjnej w funkcji czasu realizacji oprogramowania aplikacyjnego, rezydującego w węźle systemu rozproszonego. Przedstawiono budowę cyklu podstawowego sterownika swobodnie programowalnego, będącego węzłem systemu oraz zaprezentowano metodę fragmentaryzacji programu aplikacyjnego. Przedstawiono również propozycje zastosowania języka C do tworzenia oprogramowania aplikacyjnego w celu skrócenia czasu jego wykonywania.

Słowa kluczowe: systemy czasu rzeczywistego, determinizm czasowy, systemy rozproszone, systemy wbudowane, PLC.

THE APPLICATION OF C LANGUAGE PROCEDURES TO INCREASE ACCESS FREQUENCY TO THE COMMUNICATION NETWORK IN A DISTRIBUTED REAL-TIME SYSTEM

Summary. The paper describes the problem of access frequency to the communication network in a distributed industry system. The article shows the structure of the basic time cycle of PLC and interdependences between the access frequency and realisation time of application program. It was also proposed to use the C language to increase access frequency.

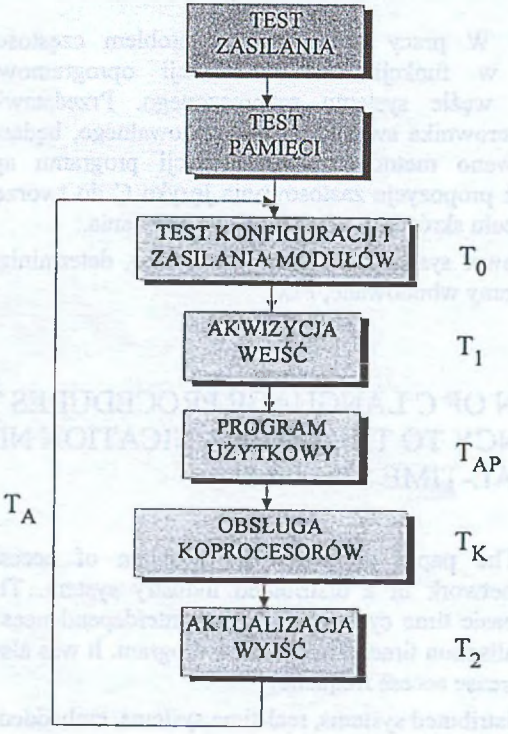
Keywords: distributed systems, real-time systems, embedded systems, PLC.

1. Wstęp

Jak dotąd najbardziej powszechnym urządzeniem programowalnym, stosowanym w procesach sterowania i regulacji fragmentów lub całych procesów przemysłowych, jest

swobodnie programowalny sterownik przemysłowy (*ang. PLC-Programable Logic Controler*). Ze względu na rodzaj oprogramowania jak i realizowane funkcje (automatu sekwencyjnego lub kombinacyjnego sterowania) jest również nazywany automatem. To określenie będzie się wielokrotnie pojawiało w niniejszej pracy zamiennie z PLC, ale zawsze dotyczyć będzie swobodnie programowalnego, logicznego sterownika przemysłowego.

Przedmiotem niniejszego rozdziału nie jest sterownik jako taki, ale jego tryb pracy. Tryb pracy bowiem ma zasadniczy wpływ na czas przetwarzania i dostępność informacji, które mają być przedmiotem transferu do systemu rozproszonego. Realizacja programu w czasie rzeczywistym wymaga, aby był on wykonywany cyklicznie. Czas trwania cyklu realizacji programu jest ważnym czynnikiem, który należy omówić. Cykliczną pracę sterownika przedstawia rys. 1.



Rys. 1. Podstawowy cykl pracy sterownika jednoprocesorowego
Fig. 1. The work cycle of single processor PLC

Na rysunku 1 przedstawiono podstawowe czynności realizowane w jednym cyklu pracy. Każda z nich wnosi swój udział czasowy w całkowity czas trwania cyklu.

Wielkości:

T0 – czas testu konfiguracji i napięć zasilających,

T1 – czas akwizycji wejść – przepisanie fizycznych stanów wejść do pamięci,

T2 – czas aktywizacji wyjść – przepisanie zawartości pamięci sterownika na fizyczne wyjścia.

są stałe dla danej konfiguracji i łatwo mierzalne. Natomiast czas T_{AP} dotyczy realizacji programu użytkownika (aplikacja) i jest zmienny, a jego precyzyjne zmierzenie nie jest łatwe. Nieco inaczej wygląda problem obsługi koprocesorów. Zależy on przede wszystkim od ich liczby i rodzaju. Inaczej na cykl automatu wpływa obsługa koprocesora sieci a inaczej na przykład obsługa koprocesora do obliczeń numerycznych, jeśli takowy został zainstalowany.

Czas pojedynczego cyklu automatu nie jest stały, a jego wartość jest sumą pięciu wielkości:

$$T_A = T_0 + T_1 + T_{AP} + T_K + T_2.$$

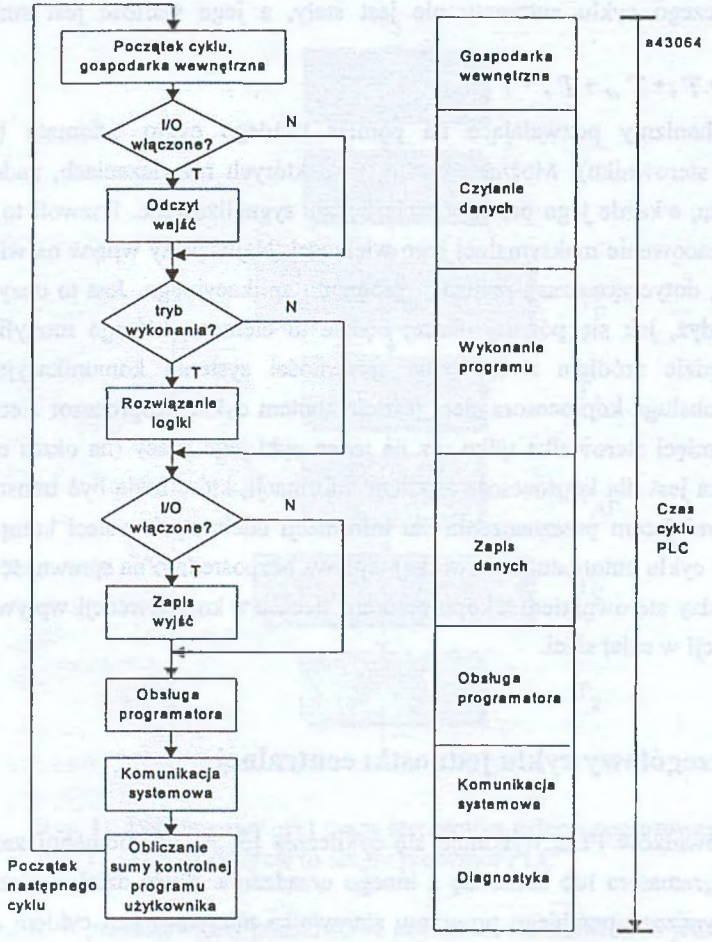
Istnieją mechanizmy pozwalające na pomiar każdego cyklu automatu (programu realizowanego w sterowniku). Można ponadto, w niektórych rozwiązaniach, zadeklarować czas trwania cyklu, a każde jego przekroczenie będzie sygnalizowane. Pozwoli to na etapie testowania na oszacowanie maksymalnej jego wielkości. Największy wpływ na wielkość T_A ma wielkość T_{AP}, dotycząca czasu realizacji programu aplikacyjnego. Jest to oczywiste, ale bardzo ważne, gdyż, jak się później okaże, będzie to element, którego modyfikacja lub optymalizacja będzie źródłem zwiększenia sprawności systemu komunikacyjnego. Jak wynika z rys. 1, obsługa koprocesora sieci jest elementem cyklu. Koprocesor sieci bowiem ma dostęp do pamięci sterownika tylko raz na jeden cykl jego pracy (na okres czasu T_K). Pamięć sterownika jest dla koprocesora źródłem informacji, które mają być transmitowane, ale jest również miejscem przeznaczenia dla informacji odebranych z sieci komputerowej. Tak więc długość cyklu automatu (sterownika) wpływa bezpośrednio na sprawność wymiany informacji pomiędzy sterownikiem a koprocesorem sieci, a w konsekwencji wpływa na czas wymiany informacji w całej sieci.

2. Opis szczegółowy cyklu jednostki centralnej

Program sterowników PLC wykonuje się cyklicznie [5] aż do momentu zatrzymania przez rozkaz programatora lub komendę z innego urządzenia. Zbiór działań niezbędny do wykonania pojedynczego przebiegu programu sterownika nazywany jest cyklem automatu. W porównaniu z wykonywaniem programu logicznego na cykl składają się: otrzymywanie danych z urządzeń wejściowych, wysyłanie danych do urządzeń wyjściowych, dokonywanie

diagnostyki wewnętrznej, obsługa programatora oraz obsługa innych połączeń. Można wyróżnić siedem (rys.1) części sekwencji wykonania standardowego cyklu programu:

- inicjalizacja cyklu,
- badanie (odczyt) wejść,
- wykonanie programu aplikacyjnego,
- aktualizacja sygnałów wyjściowych,
- transmisja danych,
- komunikacja systemowa (obsługa programatora),
- diagnostyka.



Rys. 2. Szczegółowy opis cyklu podstawowego pracy sterownika jednoprocesorowego
Fig. 2. In detail description of basic work cycle single processor PLC

Wszystkie te kroki, poza obsługą programatora, wykonywane są w każdym cyklu. Obsługa programatora następuje tylko w przypadku detekcji błędu płyty lub żądania obsługi przez urządzenie programujące. Sekwencję standardowego cyklu programu pokazuje rys.2.

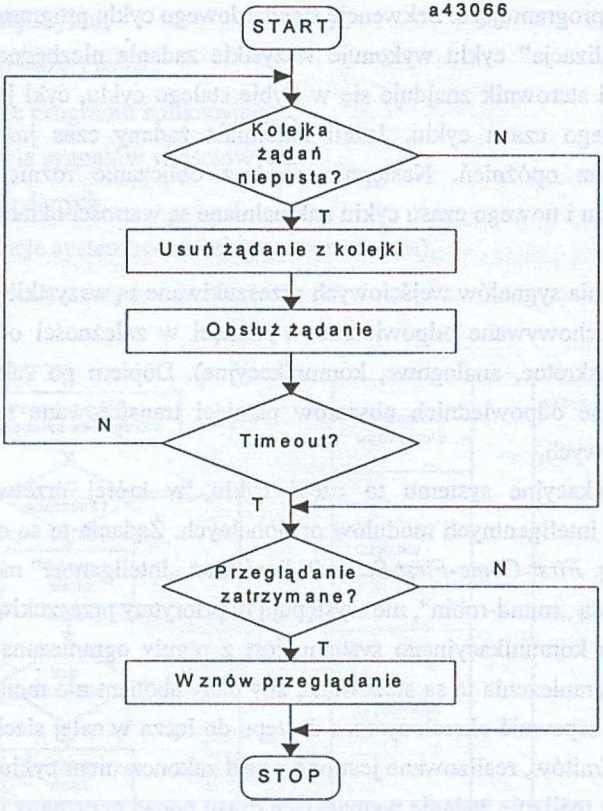
Część „Inicjalizacja" cyklu wykonuje wszystkie zadania niezbędne do przygotowania startu cyklu. Jeśli sterownik znajduje się w trybie stałego cyklu, cykl jest opóźniany aż do upłynięcia żądanego czasu cyklu. Jeżeli natomiast żądany czas już upłynął, cykl jest kontynuowany bez opóźnień. Następnie poprzez obliczanie różnic pomiędzy startem poprzedniego cyklu i nowego czasu cyklu uaktualniane są wartości układów czasowych *(ang. timers)*.

Podczas czytania sygnałów wejściowych przeszukiwane są wszystkie moduły sterownika, a ich dane są zachowywane odpowiednio w pamięci w zależności od typów zmiennych pomiarowych (dyskretne, analogowe, komunikacyjne). Dopiero po zakończeniu programu aplikacyjnego dane odpowiednich obszarów pamięci transferowane są do odpowiednich modułów wyjściowych.

Okno komunikacyjne systemu to część cyklu, w której przetwarzane są żądania komunikacyjne z inteligentnych modułów opcjonalnych. Żądania te są obsługiwane według reguły FCFS *(ang. First-Come-First-Served)*. Ponieważ „inteligentne" moduły opcjonalne są przeglądane metodą „round-robin", nie występują tu priorytety przeszukiwań.

Długość okna komunikacyjnego systemu jest z reguły ograniczana poprzez określony limit czasowy. Ograniczenia te są stosowane, aby dany abonent nie monopolizował wymian sieciowych i aby zapewnić określony czas dostępu do łącza w całej sieci. Jeżeli żądanie nie przekracza tych limitów, realizowane jest ono przed zakończeniem cyklu. Jeżeli inteligentny moduł opcjonalny realizuje żądanie wymagające czasu ponad przyznany limit, jego realizacja rozszerza się na więcej cykli tak, aby czas cyklu automatu nie został wydłużony ponad określony limit (rys.3).

Inteligentny moduł opcjonalny nie ma możliwości przerwania działania jednostki centralnej, gdy zgłasza żądanie obsługi. Jednostka centralna musi przeglądać każdy taki moduł, czekając na żądanie obsługi. Przeglądanie to zachodzi asynchronicznie w tle. Kiedy inteligentny moduł opcjonalny jest przeglądany i wysyła do jednostki centralnej żądanie obsługi, żądanie to jest kolejkowane i może być obsłużone podczas okna komunikacji systemu. Przeglądanie takich modułów jest ograniczone w ten sposób, że żaden z nich nie jest przeglądany więcej niż jednokrotnie podczas każdego cyklu. Czas cyklu składa się z odcinków stałych (inicjalizacja, diagnostyka) oraz zmiennych. Zmienne odcinki czasu zależą od konfiguracji sterownika, liczby modułów sieciowych, typu urządzenia programującego podłączonego do PLC, ale przede wszystkim od rozmiaru i jakości programu aplikacyjnego.



Rys. 3. Schemat blokowy realizacji żądania
Fig. 3. The schema of request realisation

3. Optymalizacja wymian sieciowych

Najistotniejsze, przy konfigurowaniu sieci, jest obliczanie maksymalnego czasu cyklu sieci, gdyż parametr ten będzie miał decydujące znaczenie dla określenia czasów nadawania poszczególnych abonentów. Maksymalny czas cyklu rzutuje bowiem na podjęcie decyzji o przydatności określonej sieci do realizacji zadania. Zmiany w konfiguracji sieci mogą polegać na:

- eliminacji "dziur" (dotyczy sieci z przekazywaniem żetonu do następnego abonenta),
- zminimalizowaniu liczby oraz wielkości przesyłanych informacji,
- zmniejszeniu liczby abonentów,

- zmianie priorytetów komunikatów.

Przy obliczaniu czasu cyklu sieci należy brać pod uwagę następujące czynniki:

- czas cyklu transmisji żetonu z uwzględnieniem planowanej maksymalnej liczby "dziur",
- czas transmisji komunikatu ramki o maksymalnej długości,
- czas przetwarzania ramki przy jej odbiorze i emisji,
- czas detekcji ramki,
- prędkość transmisji.

Czas potrzebny na dostarczenie informacji z potwierdzeniem z jednego sterownika do drugiego składa się z następujących elementów i jest jednocześnie minimalnym czasem odpowiedzi:

$$T_{RESP} = 2(T_{CPU\,1}) + (Tmsg_{\,2\text{-}1} + Tmsg_{\,1\text{-}2}) + 2(T_{CPU\,2}),$$

gdzie:

$T_{CPU\,1}$ i $T_{CPU\,2}$ są to czasy trwania cykli sterowników,

$Tmsg_{\,2\text{-}1}$, $Tmsg_{\,1\text{-}2}$ są to czasy transmisji, uwzględniające czasy detekcji, przygotowania i analizy obu ramek.

Czas przesyłu danych globalnych zależny jest zarówno od czasu cyklu sieci, czyli obiegu żetonu, jak również od czasu trwania cyklu jednostki centralnej danego sterownika. Ponieważ koprocesor sieci wystawia bity stanu określające, który z abonentów jest obecny, każdy cykl jednostki centralnej wydłużony jest o czas potrzebny na ustawienie tych bitów. Wartość tego czasu zwana jest czasem bazowym odbioru. Cykl jednostki centralnej sterownika wydłuża się również o czas bazowy nadawania, lecz tylko wtedy, gdy sterownik wysyła dane globalne. Jak widać, ogromny wpływ na czas dostarczenia informacji ma czas cyklu automatu. Poza podziałem wiadomości ze względu na ich ważność oraz doborem odpowiednich odstępów czasu pomiędzy kolejnymi wymianami sieciowymi najistotniejsze znaczenie dla usprawnienia komunikacji ma skrócenie czasu cyklu jednostki centralnej.

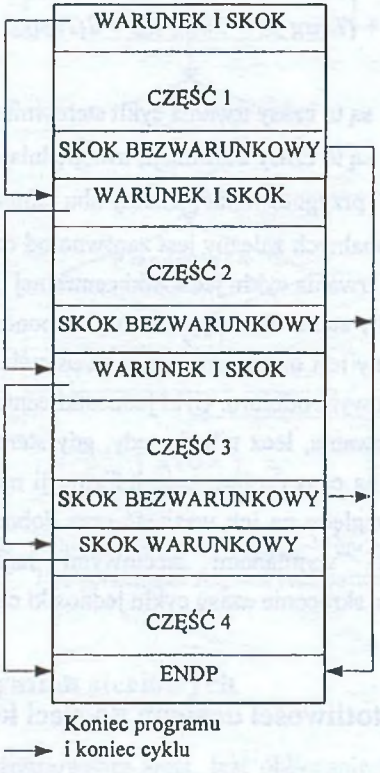
4. Zwiększenie częstotliwości dostępu do sieci komunikacyjnej

Skrócenie czasu cyklu programu sterownika jest, poza hierarchizacją komunikatów ze względu na ich ważność i doborem odpowiednich odstępów czasu pomiędzy kolejnymi wymianami sieciowymi jest zadaniem najistotniejszym [4]. Skrócenie cyklu automatu w prostej linii oznacza skrócenie czasu wykonywania programu. Jego długość, a zatem czas realizacji pętli, wpływa na częstość dostępu do pamięci jednostki centralnej przez koprocesor, a tym samym wpływa na cykl wymiany całej informacji.

Innym istotnym elementem programu mogą być występujące w nim uwarunkowania, co może powodować, że program główny musi czekać na spełnienie określonego warunku, aby

dane do transmisji były ważne. Ma to szczególne znaczenie, gdy wykorzystywane są wieloprocesorowe jednostki centralne, pracujące w trybie pracy równoległej. Dzięki odpowiedniej, właściwej synchronizacji zadań współbieżnych nie wystąpi zjawisko zbędnego oczekiwania na zakończenie pojedynczego cyklu. Zagadnienia te dotyczą problematyki równoległej pracy jednostek wieloprocesorowych, co jest problemem samym w sobie wykraczającym poza zakres niniejszej pracy.

Skrócenie cyklu automatu można również osiągnąć przez podzielenie programu na części tak, aby koprocesor co kilkanaście milisekund miał dostęp do pamięci jednostki centralnej. Sposób ten polega na realizacji całego programu w kilku cyklach (rys. 4).



Rys. 4. Podział programu sterownika na części
Fig. 4. The partition of PLC program

W każdym cyklu pracy sterownika (automatu) będzie realizowana tylko jedna część programu. Na początku każdej z nich umieszczony jest skok warunkowy decydujący o tym, czy dana część programu ma być realizowana. Jeżeli dana część była już realizowana w danym makrocyklu, to wykona się rozkaz skoku do początku następnej części programu. W przeciwnym przypadku skok się nie wykona i dana część programu będzie realizowana.

Na końcu każdej części programu umieszczony jest skok do końca programu. Na rysunku 4 widać, że w tym przykładzie koprocesor sieci w czasie realizacji całego programu w sterowniku będzie miał czterokrotny dostęp do pamięci jednostki centralnej. Rozwiązanie to ma pewne wady. Bardzo rozważnie należy bowiem podzielić program sterowania procesem na etapy. W tak podzielonym programie utrudnione jest wykrywanie szybkich zmian stanów wejściowych, a także tak zwane różniczkowanie sygnałów, polegające na wykrywaniu zboczy sygnałów wejściowych. Dodatkową wadą jest to, że realizowane układy czasowe mogą działać niedokładnie, co w wielu przypadkach jest niedopuszczalne. Trzeba również pamiętać o tym, że taki sposób realizacji programu spowoduje zwolnienie reakcji układu sterowania na zmiany stanów wejściowych, a tym samym zmniejszy dynamikę zmian stanów wyjściowych. W praktyce, szczególnie tam gdzie procesy nie są zbyt szybkie, możliwe jest realizowanie całego programu sterowania w kilku czy nawet kilkunastu cyklach sterownika. Z praktyki znane są przykłady[1], w których sterownik realizujący sterowanie kilkudziesięcioma pompami musi w każdym swym cyklu obsługiwać tylko kilka z nich. Reakcja na zmiany wejść związanych z sygnałami zabezpieczeń będzie wolniejsza o kilkadziesiąt milisekund, co nie powoduje dyskwalifikacji takiego rozwiązania. Reasumując można powiedzieć, że należy zwracać baczną uwagę na optymalne, pod względem czasu wykonania, opracowywanie algorytmów sterowania i regulacji. Trudno w tym miejscu, oprócz opisanych w tym rozdziale, dać jakieś konkretne recepty, poza ogólnikowymi wskazówkami. Dopiero doświadczenie programisty połączone z bardzo wnikliwą analizą procesu technologicznego, który ma być algorytmizowany, może dać gwarancję optymalnego wykorzystania sieci.

5. Wykorzystanie języka C do tworzenia procedur wbudowanych w część aplikacyjną oprogramowania jednostki centralnej

W zależności od standardów preferowanych przez producenta interfejs programistyczny bazować może na bezpośrednim wprowadzaniu kodów rozkazów bądź też na tworzeniu oprogramowania w sposób graficzny, w postaci bloków funkcjonalnych odpowiedzialnych za realizację poszczególnych operacji elementarnych oraz prezentowanej w sposób graficzny struktury połączeń odpowiadającej za przekazywanie sygnałów sterujących pomiędzy poszczególnymi elementami oprogramowania. Często dany typ urządzenia można programować zamiennie w jeden bądź w drugi sposób, jednak nie zawsze wybór operacji oferowanych w postaci graficznych bloków funkcjonalnych pokrywa w całości zakres rozkazów możliwych do realizacji z wykorzystaniem ciągu instrukcji. Jeszcze większą swobodę w realizacji oprogramowania aplikacyjnego jednostki centralnej daje programiście

dostępna dla części sterowników swobodnie programowalnych możliwość zastosowania procedur tworzonych bezpośrednio w języku C.

Narzędzia pozwalające na wykorzystanie procedur tworzonych w języku C umożliwiają tworzenie rozwiązań ograniczonych jedynie fizycznymi możliwościami sprzętu i zakresem posiadanych funkcji bibliotecznych. Wspomagające realizację procedur w języku C biblioteki umożliwiają dostęp do zasobów i funkcji systemowych programu monitora kontrolującego pracę jednostki centralnej. Kod języka C kompilowany jest zewnętrznie do kodu maszynowego procesora jednostki centralnej i po załadowaniu realizowany jest w sposób bezpośredni. Komunikacja z zasobami automatu dokonywana jest poprzez wspomniane procedury biblioteczne. System operacyjny jednostki centralnej odpowiada za załadowanie i uruchomienie stworzonej w języku C procedury. Dzięki takiemu rozwiązaniu sposób realizacji poszczególnych funkcji programu zależy bardzo silnie od intencji i możliwości tworzącego je programisty.

Wykorzystanie procedur tworzonych w języku C przekładanych na kod maszynowy procesora jednostki centralnej i wbudowywanych w oprogramowanie aplikacyjne jednostki centralnej stwarza z punktu widzenia czasu cyklu jednostki centralnej zarówno nowe możliwości, jak również i nowe niebezpieczeństwa. Kod tworzony w języku C umożliwia skorzystanie z szeregu konstrukcji programistycznych niedostępnych dla większości jednostek centralnych PLC. W szczególności możliwość występowania niekończących się pętli oraz możliwość znacznego wydłużenia czasu realizacji procedury w zależności od aktualnej ścieżki wewnętrznej programu sprawiają, iż wbudowane procedury tworzone w języku C wymagają szczególnie starannego przygotowania oraz dokładnych testów.

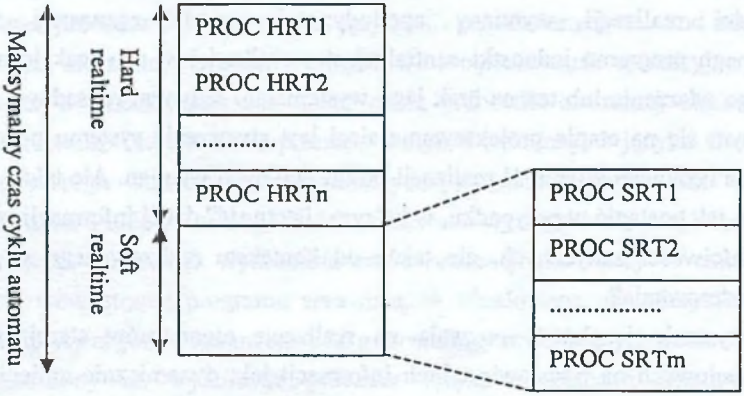
Czas przeznaczony na wykonanie procedur języka C stanowi składową części aplikacyjnej oprogramowania automatu i podlega on takim samym ograniczeniom jak pozostałe instrukcje realizowane przez jednostkę centralną. Procedury tworzone z wykorzystaniem języka C realizowane są dla sterowników z jednoprocesorową jednostką centralną w sposób szeregowy razem z pozostałymi rozkazami realizowanymi przez tę jednostkę, dając w ten sposób bezpośrednie przełożenie na długość cyklu automatu. Z drugiej strony dzięki możliwości efektywnego wykorzystania zasobów sprzętowych PLC, wykorzystaniu niedostępnych dla klasycznych metod programowania PLC konstrukcji programowych, jak również dzięki możliwości bezpośredniej optymalizacji kodu wynikowego zastosowanie procedur tworzonych w języku C może stać się bardzo ważnym narzędziem pozwalającym na skrócenie czasu cyklu jednostki centralnej, a co za tym idzie na zwiększenie częstotliwości jej dostępu do zasobów sieciowych. Zakres zastosowania procedur tworzonych w języku C może być rozpatrywany w odniesieniu do trzech płaszczyzn wpływających na długość cyklu wymiany informacji przedstawionych w rozdziałach 2, 3 i 4.

Wykorzystanie procedur tworzonych w języku C pozwala w wielu przypadkach na uzyskanie bardziej optymalnego kodu wynikowego zarówno pod względem czasu realizacji poszczególnych procedur, jak i ze względu na obszar pamięci operacyjnej wykorzystywanej do ich realizacji. Co więcej, wykorzystanie języka C stwarza możliwość realizacji własnych procedur i algorytmów nierealizowalnych lub bardzo trudno realizowalnych ze względu na złożoność czasową, zajętość pamięci danych i pamięci programu realizowanego za pomocą klasycznych metod programowania PLC. W rozdziale trzecim zaprezentowana została istotna rola, jaką odgrywa zagadnienie optymalizacji wymian sieciowych na czas obiegu informacji w rozproszonym systemie sterowania. O ile zagadnienie optymalizacji wymian cyklicznych wydaje się być zagadnieniem dosyć dobrze rozwiązanym za pomocą metod wykorzystywanych na etapie projektowania sieci przemysłowej najniższego poziomu [4], o tyle zagadnienie optymalizacji związanych z dostępem procesu technologicznego do wymian aperiodycznych nie może być prosto rozwiązane za pomocą mechanizmów predykcyjnych. Decyzja o konieczności realizacji wymiany aperiodycznej zapada zazwyczaj na poziomie realizowanego programu jednostki centralnej sterownika i jest ona reakcją na wystąpienie określonego zdarzenia lub też na brak jego wystąpienia. Jedynym rozsądnym rozwiązaniem nasuwającym się na etapie projektowania sieci jest stworzenie systemu priorytetów, który odpowiadać będzie za kolejność realizacji poszczególnych wymian. Ale także i tutaj nasuwa się pytanie, jak postąpić w przypadku, w którym „istotność" danej informacji zależy nie tylko od jej właściwości statycznych, ale także od kontekstu realizowanego w danym czasie algorytmu sterowania?

Wykorzystanie języka C pozwala na realizację algorytmów sterujących realizacją wymian sieciowych na podstawie takich informacji, jak: dynamicznie zmieniane priorytety przypisywane poszczególnym wymianom, uwzględnianie określonych stanów awaryjnych, czy też, dla algorytmów sekwencyjnych, numer kroku sekwencyjnego programu sterowania. Podobnie jak w przypadku problemów omawianych w punkcie drugim, również i zagadnienie sterowania przebiegiem wymian sieciowych może być w wielu przypadkach rozwiązywane za pomocą klasycznych mechanizmów programowania PLC. Jednakże wykorzystanie kodu tworzonego w języku C pozwala na realizację dynamicznych algorytmów sterujących pracą sieci, które w sposób optymalny lub quasi-optymalny realizują żądania komunikacyjne spływające z poszczególnych modułów oprogramowania aplikacyjnego jednostki centralnej.

Opisane w rozdziale czwartym zagadnienie segmentacji oprogramowania jednostki centralnej, pozwalające na skrócenie czasu cyklu automatu, da się rozwiązać w sposób statyczny na etapie projektowania oprogramowania. Występujące w systemach typu: *„hard real-time"* deterministyczne ograniczenie czasu trwania cyklu automatu i czasu realizacji poszczególnych procedur może zostać sprowadzone do analizy najgorszego przypadku i

przyjęcia spełniającego dane kryteria rozwiązania statycznego lub też na odrzucenie przyjętych algorytmów ze względu na brak gwarancji czasowych dla ich realizacji. W znacznej części systemów czasu rzeczywistego obok funkcji o ścisłych i nieprzekraczalnych ograniczeniach czasowych występują realizowane w tym samym systemie funkcje należące do klasy *„soft real-time"*. Funkcje *„soft real-time"* posiadają zdefiniowany czas realizacji, jednak określa się także granicę tolerancji, dla której przekroczenia czasów realizacji poszczególnych funkcji nie są traktowane jako awaria systemu. Jako przykład występowania obok siebie funkcji typu *„hard real-time"* i *„soft real-time"* można przytoczyć procedury realizowane przez jednostkę centralną PLC, związane z wypracowywaniem informacji dla systemu wizualizacji i nadzoru. Zastosowanie kryterium najgorszego przypadku do tej klasy problemów mogłoby prowadzić do odrzucania na etapie projektowym rozwiązań, które, jak pokazuje praktyka, funkcjonują poprawnie w rzeczywistych systemach.



Rys. 5. Podział okna oprogramowania aplikacyjnego
Fig. 5. The partition of application window

Rozwiązaniem pozwalającym na pogodzenie ze sobą tych dwóch odmiennych rodzajów ograniczeń może być podzielenie okna programu aplikacyjnego realizowanego poprzez jednostkę centralną na dwie części: część procedur realizowanych ze ścisłymi ograniczeniami czasowymi i część procedur typu *„soft real-time"*. Podział ten musi uwzględniać zasadę zachowania określonego maksymalnego czasu cyklu jednostki centralnej. Jak pokazano na rys.5, próba realizacji wszystkich procedur w jednym cyklu automatu prowadzić będzie do przekroczenia limitu czasu cyklu jednostki centralnej, jednak wykorzystanie czasu pozostałego po realizacji pierwszej części cyklu pozwala na realizację poszczególnych procedur z drugiej części zgodnie z określonymi wymaganiami związanymi z czasem ich realizacji. Ponieważ kryterium deterministyczne zakłada występowanie najgorszego przypadku, tak więc czas realizacji pierwszego z okien będzie zmieniał się stosownie do zdarzeń zachodzących w kontrolowanym obiekcie i wpływających na sposób realizacji

określonych fragmentów oprogramowania. Zastosowanie procedur pisanych w języku C daje możliwość tworzenia adaptacyjnych algorytmów sterujących kolejnością wykonywania zadań należących do okna *„soft real-time"* w taki sposób, aby poprzez efektywne wykorzystanie możliwości obliczeniowych jednostki centralnej stworzyć jak najlepsze warunki wykonywania procedur o słabszych ograniczeniach czasowych.

6. Podsumowanie

Zagadnienie zwiększenia częstotliwości dostępu do informacji transmitowanych poprzez sieci przemysłowe stanowi kluczowy element decydujący o rozszerzeniu spektrum zastosowań dla rozproszonych systemów czasu rzeczywistego. W pracy wykazano, iż zadanie to może zostać rozwiązane nie tylko w wyniku postępu technologicznego w zakresie tworzonej aparatury i urządzeń sieciowych, a przede wszystkim może się ono dokonywać poprzez stosowanie odpowiednich rozwiązań programowych wpływających zarówno na sposób realizacji warstwy aplikacyjnej oprogramowania sterownika, jak również na sposób obsługi poszczególnych funkcji komunikacyjnych.

W pracy wykazano, iż zasadniczy wpływ na parametry czasowe rozproszonego systemu sterowania mają dwa parametry: czas potrzebny na realizację oprogramowania aplikacyjnego i sposób realizacji funkcji komunikacyjnych. Zarówno w pierwszym, jak i w drugim przypadku zastosowanie procedur tworzonych z wykorzystaniem języka C może w znaczący sposób przyczynić się do skrócenia cyklu wymiany informacji dla określonej klasy zastosowań.

W zależności od uwarunkowań aplikacyjnych poszczególne rozwiązania mogą wykorzystywać mechanizmy języka C do realizacji procedur o znacznej złożoności czasowej i zajętości pamięci, do sterowania i optymalizacji realizowanych wymian aperiodycznych, czy też wreszcie do sterowania w sposób dynamiczny segmentacją oprogramowania aplikacyjnego. Przytoczone w pracy propozycje rozwiązań mają charakter przykładowy. W rzeczywistych systemach z pewnością zajdzie potrzeba łączenia ze sobą poszczególnych rozwiązań oraz tworzenia na ich bazie nowych algorytmów. Opisane podejście umożliwia rozwiązanie szerokiego wachlarza problemów występujących podczas tworzenia systemów czasu rzeczywistego stanowiąc pomost pomiędzy urządzeniami wykorzystującymi niskopoziomowe oprogramowanie wbudowane, a bardziej uniwersalnymi, lecz mniej wydajnymi klasycznymi mechanizmami programowymi stosowanymi dla sterowników PLC.

Oprócz opisanych powyżej zalet związanych z zastosowaniem procedur tworzonych z wykorzystaniem języka C do realizacji wymienionych funkcji, wspomnieć należy o dodatkowych korzyściach wynikających z możliwości tworzenia środowiska wspomagają-

cego tworzenie oprogramowania PLC, wykorzystującego komputer klasy PC do emulacji i testowania procedur stworzonych dla potrzeb jednostki centralnej PLC. Możliwość tworzenia narzędzi diagnostycznych i przeprowadzania testów zarówno poprzez emulację, jak i stosowanie wbudowanych procedur testujących stanowi dodatkowy atut przemawiający za kontynuacją badań rozwijających taki właśnie sposób tworzenia oprogramowania jednostki centralnej PLC.

Biorąc pod uwagę korzyści wynikające z zastosowania opisanych rozwiązań programowych wydaje się, że bliższe spojrzenie na możliwości realizacji wbudowanych w oprogramowanie aplikacyjne procedur tworzonych z wykorzystaniem języka C jest w pełni uzasadnione i może stać się ono przyczynkiem do realizacji nowej generacji systemów czasu rzeczywistego potrafiących sprostać coraz to wyższym wymaganiom niesionym przez obecnie wprowadzane technologie produkcyjne.

LITERATURA

1. Bigewski Z.: Optymalizacja pracy sieci przemysłowych. Zeszyty Naukowe Pol. Śl. s. Informatyka z. 28, Gliwice 1995.
2. Cupek R., Fojcik M.: Budowa modułów komunikacyjnych stacji nadzorczej z sieciami przemysłowymi. ZN Pol. Śl. s. Informatyka z. 32, Gliwice 1997.
3. Kwiecień A., Gaj P., Grzywak A., Mrówka Z.: Rozwiązania sprzętowe i programowe sieci przemysłowej FIP. ZN Pol. Śl. s. Informatyka z. 30, Gliwice 1996.
4. Kwiecień A., Gaj P.: Dobór protokołów w sieciach przemysłowych. ZN Pol. Śl. Studia Informatica Vol. 22, No 3(45), Gliwice 2001.
5. Kwiecień A.: Analiza przepływu informacji w komputerowych sieciach przemysłowych. ZN Pol. Śl. Studia Informatica Vol. 23, No 1(47), Gliwice 2002.
6. Werewka J.: Systemy rozproszone sterowania i akwizycji danych. Sterowniki programowalne i magistrale miejscowe. Krakowskie Centrum Informatyki Stosowanej, Kraków 1998.

Recenzent: Dr inż. Włodzimierz Boroń

Wpłynęło do Redakcji 14 kwietnia 2003 r.

Abstract

The time access for communication network for each subscriber is very important, especially in distributed real time systems. The frequency of receiving and transmission for each subscriber depends on time cycle of application program in the subscriber. The paper shows the structure of the cycle (Fig.1, Fig.2) and describes why the time access depends on program time realisation (Fig.3). Additionally a method is proposed which in many cases, can shorten the time access to network (Fig.4). The dividing of the program cycle allows to make access to the bus network more frequently. But in this case, each part of the application program will be longer. Sometimes it can be dangerous from the point of view of industry. The article also proposes the use of the C language for preparing PLC software. Thanks to this it is possible to make a new generation of function and procedure and make a time cycle shorter. In the same chapter one can find the method (Fig.5) of dynamic reorganization access to the network depending on priority exchange data.

Adresy

Zbigniew BIGEWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, zb@top.iinf.polsl.gliwice.pl .

Rafał CUPEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, rafal@top.iinf.polsl.gliwice.pl .

Andrzej KWIECIEŃ: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, andrzej@top.iinf.polsl.gliwice.pl .