ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996

Seria: INFORMATYKA z. 30 Nr kol. 1315

Andrzej GRZYWAK
Andrzej KWIECIEŃ
Zdzisław MRÓWKA
Piotr GAJ

ROZWIĄZANIA SPRZĘTOWE I PROGRAMOWE SIECI PRZEMYSŁOWEJ FIP

Streszczenie. Artykuł przedstawia podstawowe rozwiązania sprzętowe i programowe sieci FIP. W artykule przedstawiono informacje dotyczące specjalizowanej bazy elementowej potrzebnej do budowy kontrolerów dla tego typu sieci. Omówiono rozwiązanie kontrolera sieci FIP przystosowanego do współpracy z mikrokomputerem typu IBM PC. Przedstawiono problemy związane z oprogramowaniem tego typu sieci.

HARDWARE AND SOFTWARE SOLUTION OF INDUSTRIAL NETWORK FIP

Summary. This article presents the basic hardware and software solution of FIP network. It describes information about special elements base which is needed to the FIP controller construction. It discussions the FIP controller permits to realize the connection between any IBM PC computer and the FIP bus. This paper presents the functionalities of the software which supports full communication capabilities.

SOLUTIONS DU MATÉRIEL ET LOGICIEL DE RÉSEAU INDUSTRIEL FIP

Résumé. L'article présent les solutions principaux du matériel et logiciel de réseau FIP. Dans l'article sont présentés les informations concernés aux base des éléments spécialisée indispensable á la construction des contrôleurs pour ce type de réseau. On décrit le projet du contrôleur du réseau FIP adapté á la collaboration avec l'ordinateur du type IBM PC et les problemés de construction des logiciels pour ce type de réseau.

1. Wprowadzenie

W latach osiemdziesiątych zaczęto prowadzić intensywne prace nad rozwiązaniem sieci komputerowych przystosowanych do pracy w warunkach przemysłowych. Prace te rozpoczęte zostały w dwóch wielkich organizacjach przemysłowych – fabrykach koncernu General Motors i fabrykach koncernu Boeing. Opracowano wówczas dwa standardy sieci:

- sieci typu MAP (*Manufacturing Automation Protocol*),
- sieci typu TOP (*Technical and Office Protocol*).

Sieci typu MAP przystosowane były do sterowania procesami przemysłowymi w pełni zautomatyzowanymi (linie montażu samochodów), sieci TOP odpowiadały wymogom produkcji „gniazdowej" (produkcja samolotów).

W założeniach sieci te, a zwłaszcza sieci typu MAP, charakteryzowały się:

- deterministycznym algorytmem dostępu do łącza;
- „krótkimi" blokami przesyłanych danych;
- wysoką niezawodnością przy istnieniu silnych zakłóceń przemysłowych.

Sieci MAP realizowane były zgodnie z modelem warstwowym ISO/OSI, przy czym pracowały one wg standardu token-bus.

Rozpoczęte w drugiej połowie lat osiemdziesiątych prace doprowadziły do rozwoju tego typu sieci, które pracowały na zasadach sekwencyjnego przesyłu danych w rozproszonych systemach sterowania.

Instytut Informatyki Politechniki Śląskiej zajmował się tego typu sieciami opracowując zasady ich konfiguracji i oprogramowania [1], [5], [6].

W latach dziewięćdziesiątych zaczęto opracowywać sieci przemysłowe, których praca opierała się na tak zwanym modelu Producent–Dystrybutor–Konsument. Do tej kategorii sieci zalicza się sieci typu FIP (*Factory Instrumentation Protocol*). Ten typ sieci przemysłowych pozwala na „większe rozproszenie systemu sterowania", a przede wszystkim pozwala na szybką reakcję rozproszonego systemu na określone zdarzenia związane z procesem technologicznym. Sieciom typu FIP poświęcone było szereg publikacji Instytutu Informatyki [2], [3], [7], [8], [9]. Zrealizowane prace badawcze dotyczące sieci FIP pozwoliły na opracowanie założeń projektu celowego poświęconego temu zagadnieniu. W roku 1995 Komitet Badań Naukowych zatwierdził projekt celowy poświęcony sieciom FIP, który realizują Zakład Elektroniki Górniczej S.A. w Tychach i Instytut Informatyki Politechniki Śląskiej. Artykuł niniejszy bezpośrednio nawiązuje do wcześniej opublikowanych prac Instytutu Informatyki [1], [2], [3], [4], [5], [6], [7], [8], [9]. Celem artykułu jest przedstawienie istniejącej bazy elementowej koniecznej do realizacji kontrolerów sieci FIP oraz istniejących modułów oprogramowania stosowanych dla sieci tego typu. Przedstawiono również zrealizowane przez Instytut Informatyki

projekty kontrolerów sieci FIP przystosowanych do współpracy z mikrokomputerami typu IBM PC.

Artykuł nawiązuje do prowadzonego Projektu Celowego „Przemysłowe sieci komputerowe".

2. Standard FIP

We współczesnych systemach sterowania obiektami przemysłowymi występuje stały wzrost liczby sygnałów obiektowych, co pociąga za sobą konieczność instalacji wielu jednostek centralnych. Stanowi to dużą nadmiarowość w sensie mocy obliczeniowych. Fakt ten był jednym z głównych powodów rozpoczęcia prac nad sieciami, które umożliwiłyby bezpośrednie podłączenie do nich modułów wejścia/wyjścia, urządzeń pomiarowych, analogowych modułów wyjściowych itp. Sieci te nazwano sieciami terenowymi lub polowymi. Przykładem takiej sieci jest sieć FIP opracowana przez francuską firmę CEGELEC, będąca wynikiem europejskiego programu badawczego „EUREKA".

Sieć FIP jest siecią przemysłową pracującą z wykorzystaniem protokołu komunikacyjnego o tej samej nazwie. Jej głównym założeniem jest rozproszone przetwarzanie informacji znajdującej się w rozproszonej bazie danych. Informacje szczegółowe dotyczące zasady działania sieci oraz historii jej rozwoju zamieszczone są we wcześniejszych publikacjach [1], [2], [3], [4], [5], [6], [7], [8], [9].

W niniejszym opracowaniu pojawiają się nazwy firmowe dotyczące zarówno bazy elementowej, jak i oprogramowania. Nazwy te zostały zarekomendowane jako obowiązujące przez Międzynarodową Organizację **World FIP**, zrzeszającą placówki badawcze zajmujące się siecią FIP oraz jej użytkowników i zwolenników. Organizacja ta ma na celu propagowanie sieci oraz rozwiązań na niej bazujących, a także utrzymywanie kontroli nad nazewnictwem i przestrzeganiem standardu.

Obecnie sieć FIP stała się standardem ogólnoświatowym, uzyskując akceptację międzynarodowych organizacji standaryzacyjnych między innymi Francji, wraz z organizacją UTE oraz Stanów Zjednoczonych wraz z organizacją ISA.

3. Przegląd bazy elementowej związanej z siecią typu FIP

W związku z faktem, iż prace związane ze sprzętem i oprogramowaniem postanowiono prowadzić na bazie oferty firmy CEGELEC, dalsze rozważania będą w większości dotyczyć

komponentów tej właśnie firmy. Wybór taki jest podyktowany faktem, iż CEGELEC jako pomysłodawca sieci ma największe doświadczenie i najszerszą ofertę sprzętowo–programową związaną z siecią FIP. Rodzina elementów elektronicznych, które zostały specjalnie dla potrzeb sieci FIP zaprojektowane i wykonane przez firmę CEGELEC, to :

- **FULLFIP** procesor komunikacyjny pierwszej generacji,
- **FULLFIP2** procesor komunikacyjny drugiej generacji,
- **FULLFIP2 LP** procesor komunikacyjny drugiej generacji (low power) – przeznaczony do zastosowań w urządzeniach o zmniejszonym poborze mocy,
- **MikroFIP** prosty procesor komunikacyjny przeznaczony do zastosowań w urządzeniach nie wymagających obsługi dużej ilości zmiennych (max 8 zmiennych),
- **FIELDRIVE** układ zawierający nadajnik i odbiornik linii oraz logikę kontrolującą poprawną pracę magistrali sieci FIP,
- **FIELDUAL** układ kontrolujący pracę urządzenia podłączonego do dwóch magistrali sieci FIP,
- **FIELDTR** rodzina transformatorów izolujących (galwanicznie) magistralę sieci FIP od reszty elektroniki, dostosowanych do różnych prędkości transmisji oraz różnych klas bezpieczeństwa przeciwwybuchowego (Ex).

Do zbudowania karty sieciowej FIP lub interfejsu, za pomocą którego podłączamy urządzenie do sieci FIP, konieczne są:

- **FULLFIP, FULLFIP2, FULLFIP2 LP, MikroFIP** – procesor komunikacyjny,
- **FIELDRIVE** – nadajnik odbiornik linii,
- **FIELDTR** – transformator izolujący
- oraz ogólnie dostępne układy cyfrowe i elementy bierne.

W wersji karty sieciowej umożliwiającej pracę na dwóch magistralach sieci FIP stosowane są dodatkowe elementy :

- **FIELDUAL** – układ kontrolujący pracę na dwóch magistralach,
- **FIELDRIVE, FIELDTR** – po jednym dla każdego kanału.

3.1. Opis układu FULLFIP2

Układ FULLFIP2 jest procesorem komunikacyjnym, w którym zaimplementowano większość protokołów z warstwy łącza i aplikacji. Jest on następcą układu FULLFIP. Układ ten jest przeznaczony między innymi do wyposażenia sterowników, ale można go wykorzystać do budowy kart rozszerzających. Może obsługiwać do 4096 zmiennych oraz 4096 identyfikatorów z listy 65535. Pojęcie zmiennej omówione jest we wcześniejszych naszych publikacjach [7], [8].

Od strony aplikacji dostęp do zmiennej realizowany jest przy użyciu tzw. ACCESS_KEY. Klucz ten jest numerem zmiennej wybieranym w granicach 0000H–0FFFH i jest on lokalny dla danego węzła sieci. Od strony sieci FIP dostęp do zmiennych realizowany jest z użyciem numeru identyfikatora wybieranego z zakresu liczb 0000H–FFFFH i jest on numerem globalnym.

Aby umożliwić realizację powyższych funkcji układ musi być wyposażony w podręczną pamięć o pojemności od 32KB do 1MB. Obiekty, które wykorzystuje układ i które tworzą lokalną bazę danych, zapamiętywane są w tej pamięci.

Połączenie pomiędzy procesorem użytkownika a procesorem FULLFIP2 realizowane jest poprzez 8-bitową magistralę, przez którą wymieniane są dane (między aplikacją uruchomioną na PC a kontrolerem FULLFIP2). Rozkazy użytkownika oraz zdarzenia w sieci mogą występować asynchronicznie, a wszelkie konflikty dostępu do bazy danych, wynikające z tego powodu, rozstrzygane są poprzez wewnętrzny algorytm układu FULLFIP2.

Po stronie sieci układ FULLFIP2 jest podłączony do magistrali sieci FIP poprzez układ (tzw. *line transceiver*) zawierający nadajnik i odbiornik (możliwe jest zdublowanie układu nadajnika / odbiornika i zapewnienie w ten sposób pierwszego poziomu redundancji, czyli tzw. redundancji medium; zob. [7], [8]).

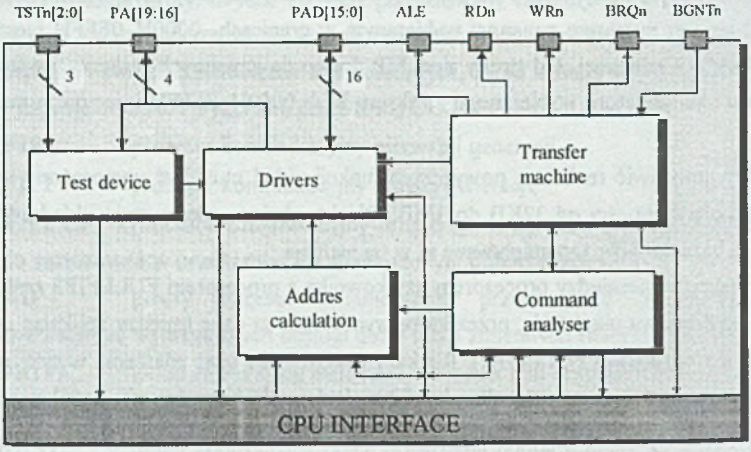
W układzie zaimplementowano standard przesyłu informacji Full-Duplex (przesył informacji w obu kierunkach w tym samym czasie - dwiema liniami), którego szybkość transmisji może być ustawiana na 31.25Kbit/s, 1Mbit/s, 2.5Mbit/s lub 5Mbit/s. Jednak w większości rozwiązań stosowana jest metoda przesyłu Half-Duplex (w obu kierunkach ale w różnym czasie - jedna linia przesyłu informacji).

3.1.1. Kontroler dostępu do pamięci

Kontroler dostępu do pamięci jest interfejsem pomiędzy jądrem układu FULLFIP2 a prywatną zewnętrzną pamięcią RAM (*EPM* - ang. *extern private memory*). Na rysunku 1 przedstawiony został schemat blokowy kontrolera.

Jak już wcześniej wspomniano, układ musi być wyposażony w pamięć podręczną. Z rysunku 1 widzimy, że układ posiada wszystkie potrzebne sygnały umożliwiające podłączenie tejże pamięci.

Wybór wielkości pamięci RAM, wynika z potrzeb użytkownika oraz z ograniczeń na szerokość magistrali adresowej (20-bitowa).



Rys. 1. Blok kontrolera dostępu do pamięci EPM
Fig. 1. EPM memory controller block

Pamięć używana jest do zapamiętywania :

- mikrokodu FULLFIP2 (FIPCODE lub FIPCODE2),
- struktury zarządzania bazą danych,
- zawartości bazy danych.

3.1.2. System interfejsu użytkownika

Transmisja danych pomiędzy układem FULLFIP2 a CPU użytkownika utrzymywana jest przy użyciu zwykłego protokołu asynchronicznego. Żądanie transmisji jest aktywowane niskim poziomem logicznym na wejściu „CSn", a potwierdzenie zakończenia transferu danych sygnalizowane jest niskim poziomem logicznym na wyjściu „DATCKn". Transmisja jest realizowana bajt po bajcie.

Układ udostępnia procesorowi użytkownika zbiór rejestrów umieszczonych w jego przestrzeni adresowej, do których mamy dostęp przez wystawienie odpowiedniego adresu na magistrali UA[2:0]. Przełączanie, na zapis lub odczyt, realizowane jest z użyciem sygnału podawanego na wejście „RWn", który to sygnał jest zależny od stanów wyjść I/OR oraz I/OW magistrali PC. Tabela 1 przedstawia szczegółowo sposób odwoływania się do poszczególnych rejestrów układu FULLFIP2.

Tabela 1

Sposób wyboru rejestrów układu FULLFIP2

UA[2:0]	RWn	Status register	RWn	Command register
000	1	USTATE	0	UCOM
001	1	UFLAG	0	USWITCH

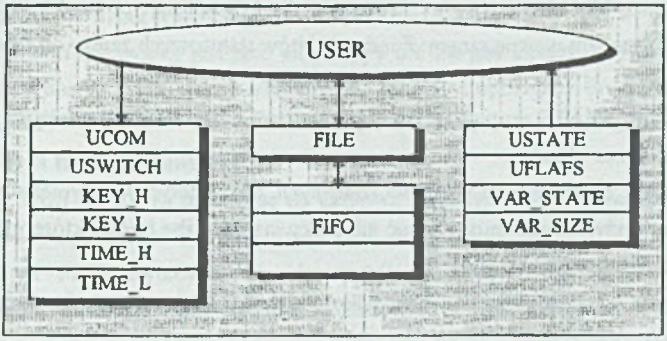
cd. tabeli 1

010	1	VAR_STATE	0	KEY_H
011	1	VAR_SIZE	0	KEY_L
100	1	unused	0	TIME_H
101	1	unused	0	TIME_L
110	1	unused	0	unused
111	1	FILE	0	FILE

Zbiór rejestrów układu FULLFIP2 dostępnych dla użytkownika możemy podzielić następująco :

- 6 rejestrów do zapisu : UCOM, USWITCH, KEY_L, KEY_H, TIME_L, TIME_H,
- 4 rejestry do odczytu : USTATE, UFLAGS, VAR_STATE, VAR_SIZE,
- FIFO - do którego mamy dostęp poprzez rejestr FILE, i który może znajdować się w trybie zapisu lub odczytu.

Na rysunku 2 przedstawiono ww. zbiór rejestrów.



Rys. 2. Model funkcjonalny systemu interfejsu użytkownika
Fig. 2. Functional model of the system interface

Część rejestrów połączona została w pary, tworząc w ten sposób 16-bitowe słowa statusowe. Taki przypadek występuje dla rejestru KEY i TIME. Rejestr TIME jest używany do modyfikacji zmiennej wewnętrznej zegara czasu rzeczywistego układu FULLFIP2. Umożliwia on synchronizację czasu zarządzanego przez układ z czasem użytkownika. Podstawowy okres zliczania ustawiany jest programowo w granicach od 0.1ms do 5ms. Rejestry raportów USTATE i UFLAGS grupują bity reprezentujące globalny status układu oraz status zmiennych. Rejestr USTATE grupuje wszystkie sygnały statusowe uaktywniane na końcówkach układu FULLFIP2 (tabela 2). Reprezentuje on globalny status interfejsu.

Tabela 2

Format rejestru USTATE

7	6	5	4	3	2	1	0
BUSY	EOC	IRQ	SV	FR	AE	FE	IC

- **BUSY** Transakcja w toku
- **EOC** Koniec cyklu
- **IRQ** Zgłoszenie przerwania
- **SV** Dane ważne
- **FR** Bufor FIFO gotowy
- **AE** Zgłoszenie błędu dostępu do procesora
- **FE** Zgłoszenie błędu formatu rozkazu
- **IC** Nielegalny rozkaz

Rejestr UFLAG jest wyłącznie uaktualniany przez wewnętrzny program FULLFIP2, grupuje on również bity statusowe zarezerwowane dla komunikacji pomiędzy aplikacją użytkownika a programem wewnętrznym. Znaczenie bitów statusowych zależy od uruchomionego oprogramowania.

Rejestr VAR_STATE grupuje razem bity statusowe właściwe dla transakcji na zmiennych rozpoznanych przez układ.

Rejestr VAR_SIZE zawiera wielkości zmiennych. Może dotyczyć również dokładnej liczby bitów, które użytkownik musi zapisać lub maksymalną liczbę bitów, które może odczytać lub zapisać, zależnie od okoliczności.

Użytkownik ma dostęp do zmiennej tylko podczas trwania transakcji, która składa się z trzech faz:

- otwarcia transakcji,
- dostępu do zmiennej,
- zamknięcia transakcji.

Podczas fazy „otwarcia" FULLFIP2 szuka w lokalnej bazie danych zmiennej i przydziela kanał DMA pomiędzy systemem interfejsu a pamięcią. Kanał ten jest używany podczas wszystkich faz dostępu do zmiennej i jest zamykany, gdy transakcja jest zamykana. Rejestry KEY_H i KEY_L pozwalają użytkownikowi na transmisję 16-bitowego parametru, który definiuje obiekt dotyczący transakcji. Ten parametr jest generalnie kluczem dostępu do tej zmiennej. Komendy identyfikowane poprzez układ dzielą się na trzy grupy, różniące się przedrostkiem zakodowanym na trzech najbardziej znaczących bitach rejestru komend. Zawartość rejestru komend UCOM jest interpretowana przez wewnętrzny „mikrokod" układu FULLFIP2. W przypadku zaistnienia sytuacji, w której komenda nie zostałaby poprawnie zinterpretowana, przesyłany jest do użytkownika kod błędu (za pośrednictwem rejestru

VAR_STATE). Interfejs systemu monitoruje kolejność komend i zapobiega jednoczesnemu otwarciu kilku transakcji. Jeśli wykryta jest komenda otwarcia podczas aktywnego sygnału BUSY, wówczas sygnalizowany jest błąd, bit IC w rejestrze USTATE ustawiany jest na 1. Układ może obsługiwać do 96 komend typu „odczyt" i do 128 typu „zapis". Kontroluje dostęp użytkownika do pewnych rejestrów w celu zabezpieczenia integralności danych (może to dotyczyć rejestrów VAR_STATE i VAR_SIZE oraz FIFO). Użytkownik może odczytywać zawartość rejestrów VAR_STATE i VAR_SIZE tylko podczas transakcji i tylko wtedy, gdy bit SV w rejestrze USTATE jest aktywny (SV=1). Zawartość odczytywanych bitów w rejestrach nie ma znaczenia (jest przypadkowa), jeśli nie dokonano odpowiedniego wyboru wcześniejszą komendą (rejestru, zmiennej, itp.).

Układ kontroluje dostęp do rejestrów w sposób podany poniżej :

- **W czasie nieaktywnej transakcji**
  Bity BUSY i SV w rejestrze USTATE są w stanie niskim (logiczne 0), a dostęp do rejestrów VAR_STATE i VAR_SIZE powoduje sygnalizację błędu wskazywaną przez bit AE w rejestrze USTATE.
- **Podczas otwarcia transakcji**
  Bit BUSY ustawiony jest jako aktywny, a SV nie jest jeszcze aktywny. W tym przypadku dostęp do VAR_STATE i VAR_SIZE jest opóźniany do czasu uaktywnienia bitu SV.
- **Podczas trwania transakcji**
  Bity BUSY i SV są ustawione na aktywne, a dostęp jest realizowany z maksymalną szybkością interfejsu.

Do bufora FIFO użytkownik ma dostęp tylko poprzez rejestr FILE i tylko podczas transakcji oraz tylko wtedy, gdy bit FR w rejestrze USTATE jest aktywny. Wartości odczytywane z rejestru nie mają znaczenia oraz zapisywane wartości są ignorowane, jeśli ten warunek nie jest spełniony.

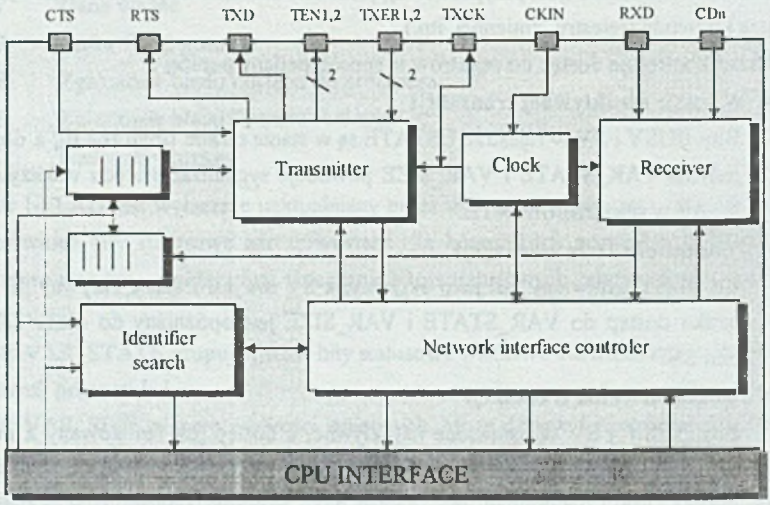
Układ kontroluje dostęp do FIFO w sposób podany poniżej :

- **W czasie nieaktywnej transakcji**
  Bity BUSY i FR w rejestrze USTATE są wyzerowane, a dostęp do FIFO powoduje sygnalizację błędu wskazywaną przez bit AE w rejestrze USTATE.
- **Podczas otwarcia transakcji do odczytu**
  Bit BUSY ustawiony jest jako aktywny, a FR jeszcze jest nieaktywny. W tym przypadku dostęp do FIFO opóźniany jest do czasu ustawienia bitu FR.
- **Podczas zapisu lub odczytu**
  Jeśli bit FR jest ustawiony na aktywny, wówczas dostęp jest realizowany z maksymalną szybkością interfejsu, jeśli nie, opóźniany jest do momentu, gdy bufor FIFO ponownie stanie się dostępny. Jeśli zostanie wykryte przekroczenie rozmiaru (zapisu lub odczy-

tu), wówczas bit FE w rejestrze USTATE jest ustawiany na aktywny i dostęp jest ignorowany. Bit FE może być kasowany przez zamknięcie transakcji (CLOSE).

3.1.3. Opis interfejsu sieci

Interfejs sieci zarządza funkcjami nadawania i odbierania ramek oraz funkcją generacji sygnału zegarowego dla układu FIELDRIVE. Może on obsługiwać dwa kanały nadawania oraz jeden kanał odbioru. Na rysunku 3 przedstawione są wszystkie sygnały układu kontrolera, za pomocą których sterowany jest układ FIELDRIVE.



Rys. 3. Blok interfejsu sieci
Fig. 3. Network interface block

3.2. Opis układu FIELDRIVE

Układ FIELDRIVE jest w pełni zintegrowanym nadajnikiem / odbiornikiem linii dedykowanym do zastosowania jako interfejs pomiędzy procesorem komunikacyjnym (FULLFIP2, FULLFIP) a transformatorem izolującym magistrale sieci FIP (*Medium Attachement Unit - MAU*).

Układ składa się z dwóch części:

- analogowej - zawierającej blok nadajnika i odbiornika,
- cyfrowej - zawierającej układy kontroli stanu sieci oraz poprawnej pracy nadajnika i odbiornika.

Sekcja nadajnika podczas nadawania kontroluje stan sieci oraz sygnalizuje odpowiednim komunikatem nieprawidłową jej pracę w następujących przypadkach :

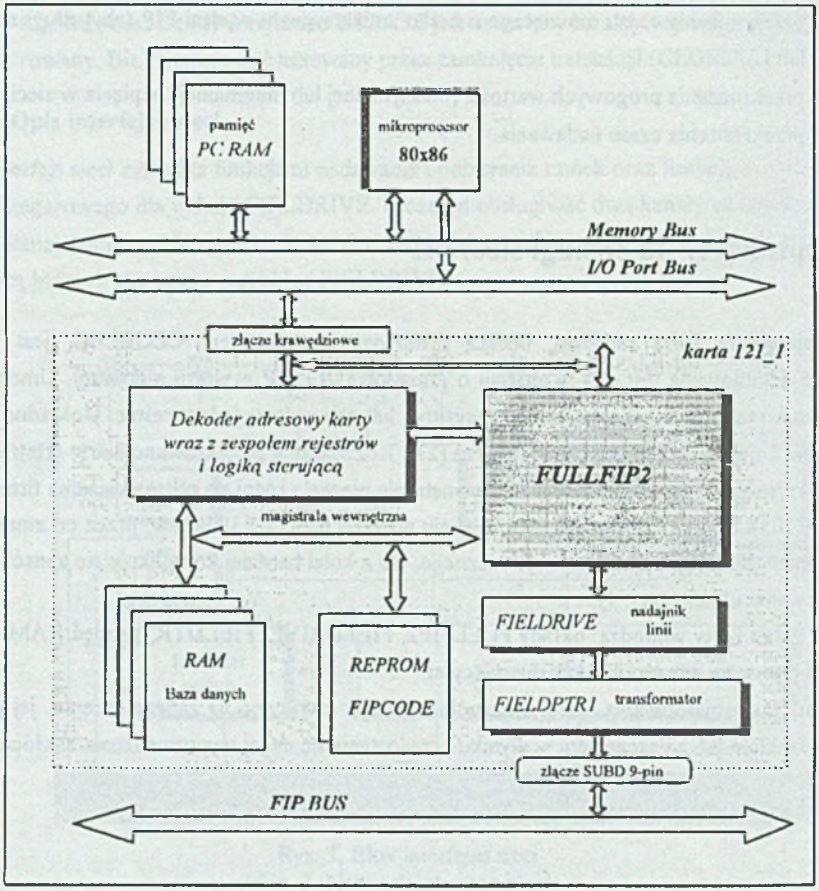
- przekroczenia czasu mówiącego o braku zmian sygnału w sieci FIP (zbyt długi stan H lub L),
- przekroczenia progowych wartości (maksymalnej lub minimalnej) napięcia w sieci,
- przekroczenia czasu nadawania.

4. Opis karty do obsługi sieci FIP

Najprostszą kartą sieciową, będącą firmowym rozwiązaniem CEGELEC, jest karta CC105. Zbudowana jest ona w oparciu o procesor FULLFIP w wersji pierwszej. Umożliwia pracę stacji jako abonent sieci, arbiter magistrali lub oba rodzaje jednocześnie. Dokładne dane na temat karty można znaleźć w literaturze [21]. Rozwiązanie projektowanej karty arbitra sieci FIP do współpracy z komputerem PC zewnętrznie niewiele różni się od rozwiązania firmowego (CC105). Różnica polega na zastosowaniu nowszej generacji układów, przez co zmniejsza się stopień złożoności układu elektronicznego, ale z kolei bardziej komplikuje się sposób programowania układu (rys. 4).

W skład karty wchodzą: układy FULLFIP2, FIELDRIVE, FIELDTR, pamięć RAM oraz zespół rejestrów wraz z układami sterującymi.

Podczas projektowania karty szczególną uwagę zwrócono na zabezpieczenie jej przed uszkodzeniem lub zniszczeniem w wyniku przedostania się na jej wyprowadzenia zakłóceń.



Rys. 4. Schemat blokowy projektowanego rozwiązania karty rozszerzającej FIP
Fig. 4. Block schematic diagram of FIP card solution

5. Oprogramowanie sieci FIP

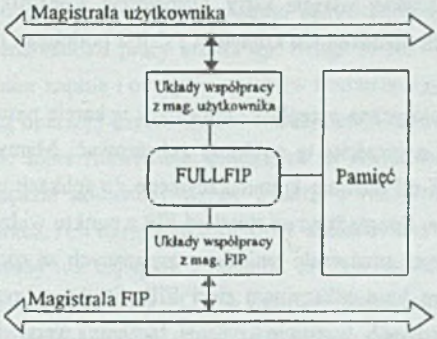
Ponieważ sieć FIP jest siecią terenową, cała baza danych oraz przetwarzanie tej bazy jest rozproszone. Każdy z abonentów sieci musi posiadać podwarstwę aplikacji zajmującą się konfiguracją i przetwarzaniem danych związanych z daną stacją. Jest to tzw. warstwa aplikacji użytkownika. Niniejszy rozdział zawiera informacje, jakie narzędzia i jakie pakiety oprogramowania mogą posłużyć do tworzenia programów użytkownika abonantów sieci FIP.

Aby rozpocząć omawianie sposobów konfiguracji oraz programowania interfejsów sieciowych sieci FIP opartych na procesorze komunikacyjnym FULLFIP/FULLFIP2, należy

przeanalizować struktury samego interfejsu pod względem przepływu informacji i rodzaju tej informacji.

Niezbędne jest również przedstawienie struktury sieci FIP w odniesieniu do siedmiowarstwowego modelu sieci otwartych ISO/OSI.

Interfejs sieciowy można ogólnie podzielić na dwie współpracujące ze sobą części realizujące dwa równoległe obiegi danych. Część pierwsza to komponenty współpracujące z magistralą sieci i umożliwiające wymianę informacji przy wykorzystaniu sieci FIP. Należą do nich transformatory separacyjne, nadajniki linii, układy kontroli magistrali, układy kontroli redundancyjności medium transmisyjnego oraz część procesora FULLFIP przeznaczona do realizacji transakcji sieciowych. Druga część karty to część procesora FULLFIP przeznaczona do komunikacji z procesem użytkownika oraz specjalizowany układ stanowiący interfejs komunikacyjny z magistralą urządzenia, dla którego zbudowany jest interfejs (rys. 5).



Rys. 5. Uproszczona struktura interfejsu komunikacyjnego
Fig. 5. Simplify communication structure

Powyższy podział, pomimo że bazuje na strukturze wewnętrznej interfejsu, dotyczy również zewnętrznego przepływu danych. Interfejs sieci FIP jest niczym innym jak inteligentnym urządzeniem komunikacyjnym, zapewniającym wymianę informacji pomiędzy siecią FIP a aplikacją użytkownika czy też samym użytkownikiem. Można zatem stwierdzić, iż procesor FULLFIP, a zatem i cały interfejs, realizuje dwa odrębne zadania. Pierwsze to zadanie sieciowe. Polega ono na zapewnieniu pracy abonenta w sieci, czyli realizacji transakcji, detekcji błędów itp. Zadanie drugie to zadanie użytkownika. Ma ono na celu zapewnić wymianę danych z aplikacją użytkownika, czyli umożliwić dostęp do zmiennych, konfiguracji, raportów itp.

Wewnętrzna architektura oprócz opisanego pionowego obiegu danych zapewnia obieg lokalny. Jest to prywatna magistrala karty, służąca do wymiany informacji pomiędzy procesorem, pamięcią oraz innymi układami współpracującymi. Z punktu widzenia konfiguracji i oprogramowania interfejsu lokalny obieg danych jest nieistotny, gdyż służy on samej karcie do realizacji współpracy międzyukładowej.

Ze względu na zagadnienia związane z konfiguracją i oprogramowaniem interfejsów komunikacyjnych sieci FIP istotna dalej będzie jedynie część służąca do wymiany informacji pomiędzy układem procesora FULLFIP oraz aplikacją użytkownika. Problem, jaki należy rozwiązać, polega na przesyłaniu informacji pomiędzy warstwą aplikacji użytkownika a warstwą komunikacyjną sieci i odwrotnie oraz na konfiguracji warstwy komunikacyjnej z poziomu warstwy aplikacji użytkownika. Konfigurując, oprogramowując czy też interaktywnie współpracując z interfejsem z poziomu komputera PC lub innego urządzenia, nieistotne jest jak interfejs realizuje lokalne przesyły danych, czy też jak część współpracująca z siecią FIP realizuje transakcje. Oczywiście, niezbędna jest znajomość zasad pracy sieci oraz całej filozofii modelu PDC, jednak przyjęte rozwiązania pozwalają na programową współpracę z interfejsem z pewnego wyższego poziomu abstrakcji. Ponieważ problem pojawił się przy okazji prób konstruowania karty sieciowej dla komputera klasy PC, w dalszych rozważaniach odnosić się będziemy do przypadku takiej właśnie karty. Oczywiście, konstrukcja interfejsów i sposób współpracy z nimi innych urządzeń niż komputer PC jest podobna i nie zmienia filozofii tutaj przedstawionej.

Analizując strukturę logiczną przepływu informacji w karcie nasuwa się model ISO/OSI, na przykładzie którego najprościej tę strukturę zobrazować. Mamy w końcu do czynienia z przepływem informacji od medium komunikacyjnego do aplikacji użytkownika lub samego użytkownika i odwrotnie. Spoglądając na standard FIP z punktu widzenia modelu sieci otwartych ISO/OSI można wyciągnąć wiele wniosków związanych ze sposobem współpracy programowej z urządzeniem komunikacyjnym sieci FIP. Struktura systemu komunikacyjnego oparta jest na trzech warstwach: warstwie fizycznej, łączenia i warstwie aplikacji. Każda z tych warstw realizuje usługi dla niej zdefiniowane. Warstwa fizyczna zajmuje się ostatecznym przygotowaniem ramek i ich fizyczną transmisją na magistralę. Warstwa łączeniowa nadzoruje przebieg poszczególnych transakcji, czyli realizuje algorytm logicznej dystrybucji ramek. Warstwa aplikacji zajmuje się zarządzaniem pracy stacji abonenckiej, czyli udostępnia usługi dla aplikacji użytkownika. Konstruując zatem urządzenie komunikacyjne, które z założenia oparte ma być na standardzie sieci FIP, powinniśmy otrzymać urządzenie oferujące usługi zdefiniowane w każdej z trzech warstw opisanych powyżej. Aplikacja użytkownika powinna mieć możliwość korzystania z tych usług oraz powinna mieć możliwość konfigurowania danych wykorzystywanych przez każdą z warstw. Z punktu widzenia użytkownika istnieją zatem dwie grupy funkcji służących do współpracy aplikacji użytkownika z systemem komunikacyjnym sieci FIP. Są to funkcje interaktywnej współpracy z siecią oraz funkcje konfiguracyjne.

5.1. FIPCODE

Układy nadajników i odbiorników linii umożliwiają fizyczną transmisję ramek, natomiast procesor FULLFIP realizuję logikę komunikacji oraz zarządzania. Aby wszystkie komponenty sprzętowe mogły poprawnie działać, potrzebny jest program, który umożliwi zarządzanie pamięcią karty oraz współpracę procesora z resztą układów. Można powiedzieć, że potrzebny jest specyficzny mikrosystem operacyjny karty, udostępniający prymitywy realizujące podstawowe polecenia dla procesora FULLFIP. Należy tutaj zaznaczyć, iż procesor posiada listę mikroinstrukcji umożliwiających sterowanie jego pracą. Zorganizowanie tych mikroinstrukcji w pewne prymitywy jest głównym zadaniem, jakie musi spełniać mikrosystem karty. Prymitywy te powinny spełniać dwie podstawowe funkcje. Pierwsza to umożliwienie użytkownikowi konfigurowanie baz danych arbitra i abonenta. Polega ona na zorganizowaniu prywatnej pamięci karty w sposób umożliwiający przechowywanie danych o zmiennych, komunikatach, listach skaningowych, scenariuszach pracy arbitra itp. Druga to sterowanie pracą procesora. Polega ona na odpowiednim zapisie i odczycie rejestrów komunikacyjnych układu FULLFIP w zależności od tego, jaką operację użytkownik chce rozpocząć, zakończyć czy też przeprowadzić. Można wyobrazić sobie rozwiązanie polegające na stworzeniu oprogramowania na komputerze PC, które będzie komunikować się z kartą, a konkretnie z rejestrami układu FULLFIP, sterując jego pracą. Jest to rozwiązanie, które właściwie nie wymagałoby prywatnej pamięci karty, jednak byłoby nie najlepsze z powodu silnego uzależnienia pracy procesora FULLFIP od pracy procesora komputera PC, i w zasadzie mocno ograniczałoby możliwości urządzenia. Rozwiązaniem znacznie bardziej optymalnym jest umieszczenie takiego programu w pamięci prywatnej karty i korzystanie z jego usług przez aplikację użytkownika. Jednokrotne skonfigurowanie, zaprogramowanie i uruchomienie karty spowodowałoby, iż karta stałaby się logicznie niezależnym urządzeniem, do którego aplikacja użytkownika pracująca na komputerze PC miałaby dostęp. Łatwo zauważyć, że tego typu mikrosystem operacyjny będzie musiał realizować usługi warstwy łączenia i aplikacji, więc jego działanie musi bazować na zdefiniowanym standardzie sieci FIP.

Pisanie takiego oprogramowania od podstaw jest niezmiernie skomplikowane i trudne. Każdy dostęp do procesora polega na wpisaniu kodu rozkazu do rejestru rozkazów UCOM, oczekiwaniu na zrealizowanie polecenia, odczycie wyniku i statusów oraz na zamknięciu polecenia. Przy założeniu, iż korzysta się z istniejącego procesora komunikacyjnego, jakim jest układ FULLFIP, rozsądniejsze wydaje się stosowanie programów już istniejących. Należy do nich program FIPCODE. Sprzedawany jest on w kilku wersjach. Najnowsza z nich, współpracująca z układami FULLFIP oraz FULLFIP2, ma numer wersji 5. 0. Stosując gotowy system można uniknąć błędów, które powstałyby przy tworzeniu własnego systemu, a co za tym idzie, uzyskujemy na czasie, który pochłonęłyby testy i korekcje błędów.

FIPCODE w wersji 5. 0 realizuje następujące usługi warstwy łączenia:

- periodyczny transfer zmiennych
- aperiodyczny transfer zmiennych
- periodyczną i aperiodyczną obsługę komunikatów bez potwierdzenia
- periodyczną i aperiodyczną obsługę komunikatów z potwierdzeniem z możliwością powtórzeń

W warstwie aplikacji udostępnia usługi:

- lokalnego zapisu i odczytu zmiennych
- kontroli statusu ważności produkcji i transmisji
- sygnalizacji
- żądania transmisji aperiodycznych zmiennych lub komunikatów

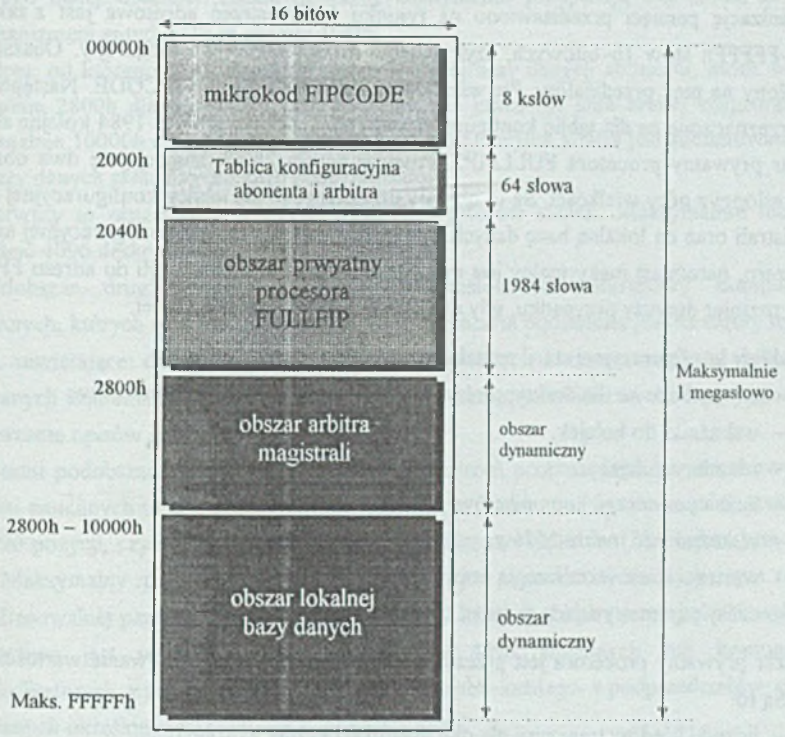
Funkcje dodatkowe, jakie spełnia FIPCODE, to:

- lokalny dostęp do pamięci
- konfiguracja lokalnej bazy danych
- inicjalizacja układów sprzętowych
- obsługa arbitra magistrali

Rozmiar programu FIPCODE wynosi osiem kilosłów szesnastobitowych. Umożliwia on obsługę do 4000 zmiennych z zakresu 0000h..FFFFh. Transmisję komunikatów realizuje w dziewięciu kanałach stanowiących osiem kolejek dla komunikatów cyklicznych i jedną dla komunikatów aperiodycznych. Każda z kolejek może pomieścić do 32 komunikatów. FIPCODE organizuje jeden bufor odbiorczy dla komunikatów o rozmiarze do 64 komunikatów oraz jeden bufor dla zdarzeń o rozmiarze do 32 zdarzeń. Umożliwia pracę w jednej z trzech standardowych prędkości transmisji.

FIPCODE wykorzystuje polecenia procesora FULLFIP zapisywane do rejestru UCOM w celu zrealizowania określonych usług, udostępniając użytkownikowi funkcje realizujące te usługi. Funkcje udostępniane użytkownikowi można pogrupować następująco:

- funkcje inicjalizacyjne,
- funkcje dostępu do pamięci,
- funkcje konfigurujące bazy danych,
- funkcje obsługi zmiennych,
- funkcje obsługi komunikatów,
- funkcje synchronizacji zdarzeniowej,
- instrukcje arbitra magistrali,
- funkcje obsługi arbitra magistrali.



Rys. 6. Przestrzeń adresowa układu FULLFIP
Fig. 6. General organization of the FULLFIP memory

Można ustalić jak FIPCODE realizuje polecenie lokalnego odczytu zmiennej. Do rejestrów KEY_H i KEY_L wpisywany jest klucz dostępu do zmiennej. Następnie do rejestru UCOM wpisywany jest rozkaz OPEN_R_VAR otwierający w procesorze sekwencję odczytu zmiennej. FIPCODE odczytuje rejestr USTATE sprawdzając kody błędów i oczekuje na ustawienie bitu SV sygnalizującego gotowość procesora do przesyłu wartości zmiennej. Równolegle po wydaniu rozkazu OPEN_R_VAR układ FULLFIP ustawia znacznik BUSY w rejestrze USTATE i przystępuje do przeszukiwania bazy danych. Po odnalezieniu szukanej zmiennej wylicza statusy ważności, uaktualnia rejestry VAR_SIZE oraz VAR_STATE i aktywuje bit SV. Następnie zapełnia wewnętrzny bufor FIFO wartością zmiennej i inicjuje pętle odczytu. Po zrealizowaniu odczytu, czyli po opróżnieniu bufora, procesor czeka na zamknięcie sekwencji. Jednocześnie FIPCODE po wykryciu ustawienia bitu SV sprawdza zawartość rejestrów VAR_SIZE oraz VAR_STATE i przystępuje do realizacji pętli odczytu z rejestru FILE. Po zakończeniu pętli, czyli po odczytaniu liczby bajtów równej liczbie odczytanej z rejestru VAR_SIZE, przechodzi do kontroli błędów na podstawie rejestru VAR_STATE i zamyka sekwencję odczytu zmiennej.

Organizację pamięci przedstawiono na rysunku 6. Przestrzeń adresowa jest z zakresu 00000h–FFFFFh słów 16–bitowych, czyli zajmuje maksymalnie dwa megabajty. Obszar ten jest dzielony na pięć przedziałów. Pierwsze osiem kilosłów zajmuje FIPCODE. Następne 64 słowa przeznaczone są dla tablic konfiguracyjnych abonenta oraz arbitra. 1984 kolejne słowa to obszar prywatny procesora FULLFIP2. Powyżej adresu 2800h znajdują się dwa obszary o nieokreślonej z góry wielkości. Są to obszary przeznaczone dla tablicy konfiguracyjnej arbitra magistrali oraz na lokalną bazę danych. Minimalny rozmiar tablicy konfiguracyjnej arbitra wynosi zero, natomiast maksymalny jest ograniczony do 54 kilosłów, czyli do adresu FFFFh. Zerowy rozmiar dotyczy przypadku, gdy dana stacja nie pracuje jako arbiter.

W tablicy konfiguracyjnej stacji przechowywane są następujące dane:

- adresy bazowe dla deskryptorów ramek oraz zmiennych i komunikatów,
- wskaźniki do kolejek,
- rozmiary kolejek,
- liczba powtórzeń komunikatów,
- wskaźniki do makrocyklów,
- wartości czasów: milczenia oraz odpowiedzi,
- czasy parametryzujące moment uruchomienia arbitra.

Obszar prywatny procesora jest przeznaczony głównie na przechowywanie wartości liczników. Są to:

- liczniki błędów transmisji dla obu magistral i każdej z osobna,
- liczniki odebranych ramek bez błędu,
- liczniki odebranych ramek z błędem,
- licznik timeout-u,
- liczniki błędów CRC, itp.

Obszar programu arbitra magistrali składa się z dwóch podprzedziałów. Pierwszy zawiera programy arbitra wraz z tablicami antypowtórzeniowymi, natomiast drugi, począwszy od adresu określonego w tablicy konfiguracji, zawiera kolejki dla aperiodycznej transmisji komunikatów oraz zmiennych. Długości obszarów przeznaczonych na kolejki określone są również w tablicy konfiguracji. Istnieje możliwość przechowywania w pamięci kilku różnych programów dla arbitra. Każdy taki makrocykl posiada swoją prywatną tablicę antypowtórzeniową, służącą do eliminacji sytuacji, w których ten sam abonent żąda aperiodycznego uaktualnienia zmiennej więcej niż jeden raz przed obsługą tego żądania. Tablica antyduplikacyjna składa się z dwusłowowych rekordów zawierających identyfikator zmiennej oraz specjalne słowo kontrolne. Rekord taki zwany jest elementem antyduplikacyjnym. Z każdym identyfikatorem zmiennej związany jest taki rekord w tablicy. W skład instrukcji makrocyklu wysłania identyfikatora wchodzi oprócz kodu instrukcji i wartości identyfikatora adres do elementu antydupli-

kacyjnego danej zmiennej. Zatem na każdy identyfikator przypadają trzy słowa związane z mechanizmem antyduplikacji obsługi żądań.

Adres, od którego rozpoczyna się obszar lokalnej bazy danych abonenta, może wynosić minimalnie 2800h dla przypadku, gdy stacja ma nie pracować jako arbiter magistrali oraz maksymalnie 10000h dla przypadku, gdy tablica konfiguracyjna arbitra jest zdefiniowana. Obszar bazy danych składa się z trzech podprzedziałów.

Pierwszy to obszar szesnastosłowowych deskryptorów ramek. Maksymalnie może on pomieścić 4096 deskryptorów.

Podobszar drugi zawiera również szesnastosłowowe deskryptory komunikatów i zmiennych, których również może być do 4096. Tablica ta podzielona jest na cztery logiczne części, zawierające: deskryptory zmiennych produkowanych i konsumowanych, deskryptory odbieranych komunikatów, deskryptory wysyłanych komunikatów oraz obszar przeznaczony na tworzenie opisów zmiennych dynamicznych.

Ostatni podobszar lokalnej bazy danych to przestrzeń przeznaczona na przechowywanie wartości zmiennych i treści komunikatów. Oczywiście również może on zawierać maksymalnie 4096 pozycji, czyli 64 słowa (128 bajtów) na zmienną i 128 słów (256 bajtów) na komunikat. Maksymalny rozmiar przestrzeni bazy danych jest ograniczony maksymalnym rozmiarem adresowalnej pamięci, czyli adresem FFFFFh (2 Mbajty). Liczba pozycji 4096 wskazuje, iż możliwe jest wykorzystywanie maksymalnie 4096 zmiennych lub komunikatów o identyfikatorach z zakresu 0000h . . FFFFh. Początek każdego z podprzedziałów obszaru bazy danych określony jest w tablicy konfiguracji stacji.

Podział opisu zmiennej na opis ramki i opis samej zmiennej wynika z rozgraniczenia zadań, jakie na danej zmiennej są wykonywane. Deskryptory ramek związane są z zadaniem, jakie na zmiennej wykonuje procesor od strony sieci FIP, natomiast deskryptor zmiennej związany jest z zadaniem wykonywanym przez użytkownika. Jest to widoczne w sposobie odwołań do zmiennej. Na sieci zmienna jest reprezentowana przez identyfikator, natomiast od strony użytkownika po inicjalizacji zmiennej identyfikator jest niedostępny. Użytkownik otrzymuje dostęp do zmiennej na podstawie tzw. ACCESS-KEY, który można traktować jako nazwę identyfikującą zmienną dla użytkownika.

Istotne dla programowania karty sieciowej są formaty deskryptorów ramek, komunikatów i zmiennych. Wszystkie deskryptory składają się z szesnastu słów. Deskryptor ramki na dwóch pierwszych słowach zawiera identyfikator zmiennej oraz jej typ. Typ zmiennej określa charakterystykę zmiennej o danym identyfikatorze.

Kolejne dwa słowa zawierają względne adresy do deskryptora zmiennej produkowanej lub konsumowanej. Słowo siódme zawiera wymagany okres produkcji dla danej zmiennej. Jest to czas określający maksymalny przedział pomiędzy produkcją zmiennej przez użytkownika a jej transmisją na sieć. Oczywiście jest on istotny tylko wtedy, gdy wykorzystywany jest status

produkcji zmiennej. Słowo dziewiąte zawiera wskaźnik dla komunikatu. Słowo to powinno być inicjowane na jedną z dziewięciu wartości adresów, oczywiście o ile zmienna jest typu producent. W sytuacji gdy stacja otrzyma ramkę ID_MSG, przed transmisją treści komunikatu stacja odczytuje tę treść z adresu podanego w słowie siódmym. W sytuacji gdy stacja nie obsługuje danego komunikatu, wartość w tym słowie powinna wynosić zero. Pozostałe słowa deskryptora ramki są zarezerwowane i niedostępne dla użytkownika. Łatwo zauważyć, iż deskryptor ramki zawiera tylko dane potrzebne sieci na przeprowadzenie transakcji z danym identyfikatorem. Informacje pozostałe, określające zmienną lub komunikat, zawarte są w deskryptorze zmiennych lub komunikatów. Deskryptory te są różne dla transmisji i odbioru, czyli produkcji i konsumpcji.

Deskryptor komunikatów otrzymywanych zawiera na najmłodszym bajcie określenie typu deskryptora, na pierwszym słowie klucz dostępu identyfikujący komunikat oraz na słowie czwartym względny wskaźnik do treści komunikatu w strefie alokacji wartości zmiennych. Słowa pozostałe są zarezerwowane. Każde otrzymanie komunikatu powoduje wygenerowanie przerwania z raportem zdarzenia 254, co użytkownik może wykorzystać do odczytu jego treści identyfikując komunikat przez klucz dostępu. Strefa deskryptorów komunikatów otrzymanych w tablicy deskryptorów jest ograniczona maksymalnie do 64 pozycji, w przeciwieństwie do strefy deskryptorów komunikatów wysyłanych, w której ilość pozycji jest ograniczona jedynie ilością pamięci.

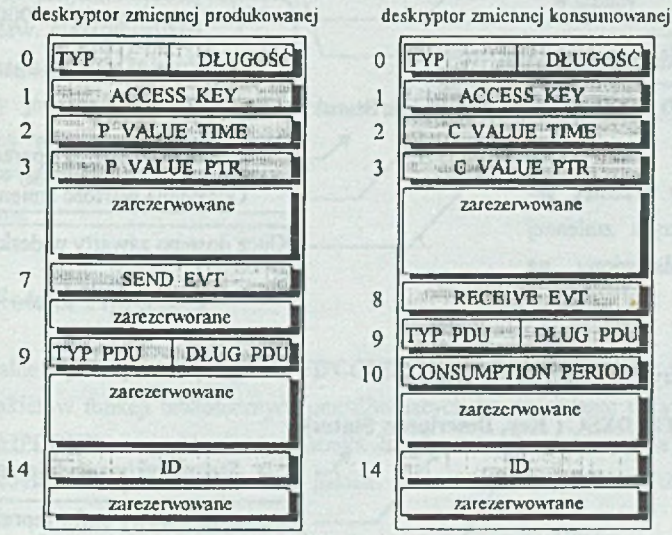
Format deskryptora komunikatów wysyłanych jest podobny do deskryptora komunikatów odbieranych. Zawiera on dodatkowe pole (słowo 8) z numerem zdarzenia identyfikującym wysłanie komunikatu. Stacja po wysłaniu komunikatu wystawia przerwanie z kodem zdarzenia równym numerowi zapisanemu w słowie ósmym deskryptora. Określenie numeru zdarzenia odpowiadającego otrzymaniu komunikatu nie jest wymagane, gdyż odebranie jakiegokolwiek komunikatu powoduje automatyczne wygenerowanie zdarzenia o numerze 254.

Adresy rekordów z wartościami zmiennych lub komunikatów zawarte w deskryptorach nie są adresami fizycznymi. Adres fizyczny otrzymuje się przez przemnożenie wskaźnika zawartego w deskryptorze przez wartość 40h, czyli maksymalny rozmiar zmiennej liczony w słowach. Jest to rozmiar tzw. strony pamięci. Tego typu wyliczenie jest możliwe dzięki faktowi, iż rozmiar alokowanej pamięci dla zmiennych jest zawsze taki sam, niezależnie od zdefiniowanej długości zmiennej i wynosi 64 słowa. W przypadku komunikatów na ich treść alokowane są niejako dwie strony pamięci po 64 słowa każda.

Podobnie jest ze wskaźnikami do deskryptorów zmiennych przechowywanych w deskryptorach ramki. Aby wyliczyć adres fizyczny, określający położenie deskryptora w pamięci zewnętrznej, należy do wskaźnika dodać wartość bazową położenia deskryptorów zmiennych pomnożoną przez stałą 400h.

W pierwszych dwóch słowach deskryptora zmiennej (rys. 7), zarówno produkowanej jak i konsumowanej, znajdują się, podobnie jak w przypadku komunikatów, typ określający rodzaj zmiennej, długość zmiennej oraz klucz dostępu. Kolejne pola w przypadku zmiennej produkowanej, to: czas wyprodukowania ostatniej wartości zmiennej przez użytkownika, adres rekordu przechowującego tę wartość, numer zdarzenia identyfikującego wysłanie zmiennej na sieć, typ PDU określający typ danych wg standardu ISO, długość danych związanych z danym typem PDU, identyfikator zmiennej.

Format deskryptora zmiennej konsumowanej jest podobny. Zawiera on: czas ostatniego odczytu wartości z sieci, wskaźnik do rekordu przechowującego wartość zmiennej, numer zdarzenia identyfikującego odczyt zmiennej z sieci, typ PDU zmiennej, długość danych związanych z danym typem PDU, period konsumpcji określający maksymalny czas pomiędzy odczytem zmiennej z sieci a odczytem zmiennej przez użytkownika, identyfikator zmiennej.



Rys. 7. Formaty deskryptorów zmiennych
Fig. 7. Descriptors of variables format

Spoglądając na całą strukturę opisu zmiennej można powiedzieć, iż jest on trójstopniowy. Na poziomie sieci zmienną opisuje deskryptor ramki zawierający wskaźnik do deskryptora zmiennej, który opisuje zmienną z poziomu użytkownika. Wartość rzeczywista zmiennej przechowywana jest w rekordzie o stałej długości 64 słów, do którego wskaźnik znajduje się w deskryptorze zmiennej. Położenie tego rekordu może ulegać zmianie w trakcie pracy sieci, gdyż jest to obszar alokowany dynamicznie.

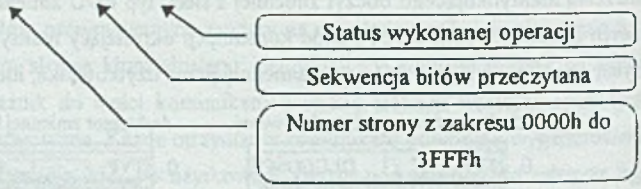
Analizując budowę deskryptorów można zauważyć, iż możliwe jest istnienie jednoczesne dwóch deskryptorów zmiennej, jednego dla produkcji, drugiego dla konsumpcji. W praktyce

FIPCODE nie dopuszcza możliwości, aby jedna stacja była producentem i konsumentem tej samej zmiennej.

FIPCODE, jak już wspomniano, udostępnia zbiór funkcji realizujących sekwencję poleceń układu FULLFIP, mających na celu zrealizowanie odczytu lub zapisu na pamięci, transmisji na sieci lub operacji na arbitrze magistrali. Poniżej przedstawiono niektóre z tych funkcji wraz z krótkim opisem.

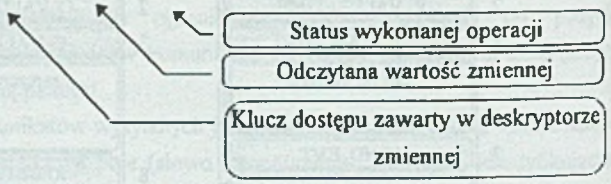
- Odczyt z pamięci zewnętrznej

READ_PHY (Page_number, Value, Status)



- Odczyt zmiennej

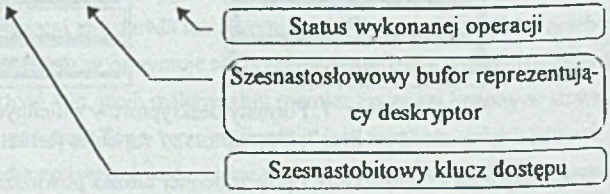
READ_VAR (Access_key, Value, Status)



Dokładny przebieg sekwencji odczytu zmiennej został przedstawiony na stronie 501.

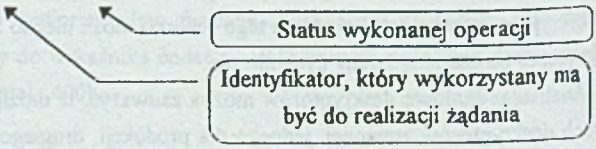
- Zapis deskryptora

WRITE_DESC (Key, Descriptor, Status)



- Żądanie aperiodycznej transmisji zmiennej

SEND_APER (Identifier, Status)



- Uruchomienie arbitra magistrali

START_BA (Idle_Fr_ID, Status)

Status wykonanej operacji

Identyfikator ramki bezczynności. Musi to być identyfikator nie obsługiwany przez sieć.

- Transmisja ramki ID_DAT

SEND_ID (Frame_Code, Identifier, Antiduplic_Element_Addr)

Adres elementu antyduplikacyjnego w tablicy

Identyfikator, który ma być rozesłany.

Kod ramki

Jest to przykład jednej z siedmiu tzw. makroinstrukcji arbitra magistrali, z których zbudowany jest opis makrocyklu pracy arbitra przechowywany w tablicy konfiguracyjnej arbitra.

5.2. Biblioteka FipAccess

Optymalne wykorzystanie programu FIPCODE z poziomu komputera PC wymusza stosowanie pakietów funkcji bibliotecznych umożliwiających współdziałanie układu FULLFIP, programu FIPCODE oraz aplikacji użytkownika. Istnieją dwa takie pakiety, oba dla języka C. Pierwszy z nich to FipAccess. W skład pakietu wchodzą biblioteki FIPHANDLER oraz FIPGEN.

Biblioteka FIPHANDLER zapewnia połączenie interfejsu użytkownika z komponentami FULLFIP za pośrednictwem programu FIPCODE, z poziomu procesora użytkownika. Funkcje FIPHANDLER są przeznaczone dla obsługi abonenta sieci lub arbitra magistrali i umożliwiają:

- dostęp fizyczny do pamięci karty,
- lokalny odczyt i zapis zmiennych,
- zapis i modyfikację deskryptorów,
- odczyt statusów,
- definiowanie i odczyt zdarzeń,
- obsługę komunikatów,

- obsługę żądań,
- obsługę arbitra magistrali.

Wszystkie możliwości funkcjonalne biblioteki są zgodne z wymaganiami standardu FIP. Biblioteka FIPHANDLER zawiera mikrokod programu FIPCODE.

Poniżej opisanych zostało kilka przykładowych funkcji udostępnianych przez bibliotekę FIPHANDLER.

```
FIP_DOWNLOAD_PRG ()
FIP_DOWNLOAD_FILE ()
```

Funkcje służą do załadowania mikrokodu FIPCODE do prywatnej zewnętrznej pamięci karty.

```
FIP_START_CIRCUIT ()
```

Funkcja uaktywniająca działanie programu FIPCODE i włączająca testy układu FULLFIP. Przed wywołaniem tej funkcji użytkownik musi załadować FIPCODE i wykonać wszystkie niezbędne konfiguracje.

```
FIP_END_DIAG ()
```

Zakończenie testowania układu FULLFIP.

```
FIP_WRITE_PHY     ( short Page_Number, unsigned char
                    *Buffer_ptr, short Size )
FIP_READ_PHY      ( short Page_Number, unsigned char
                    *Buffer_ptr, short Size )
FIP_WRITE_PHY_INTEL ( short Page_Number, unsigned char
                      *Buffer_ptr, short Size )
FIP_READ_PHY_INTEL  ( short Page_Number, unsigned char
                      *Buffer_ptr, short Size )
```

Fizyczny zapis lub odczyt pamięci karty bloku danych o rozmiarze 64 słów 16-bitowych. Realizacja polecenia dokonywana jest w formacie MOTOROLA, natomiast umieszczenie danych w buforze może być wykonane w formacie MOTOROLA lub INTEL.

```
FIP_WRITE_VAR     ( short Access_key, unsigned char
                    *Buffer_ptr)
FIP_READ_VAR      ( short Access_key, VarStruct
                    *Buffer_ptr )
```

Lokalny zapis i odczyt zmiennej. VarStruct zawiera typ PDU, długość zmiennej, dane oraz status.

```
FIP_WRITE_DESC    ( short Key, unsigned short *Buffer_ptr)
FIP_READ_DESC     ( short Key, unsigned short
                    Buffer_ptr[16] )
```

Zapis i odczyt deskryptorów. Funkcja umożliwia operowanie na wszystkich formatach deskryptorów. Fizyczny adres deskryptora wyliczany jest z zależności:

$$\text{Adres_fizyczny} = (\text{DESC_BASE} * 400\text{H}) + (\text{Key} + 10\text{H})$$

Wartość DESC_BASE jest adresem bazowym tablicy deskryptorów z tablicy konfiguracji stacji, natomiast wartość Key analogiczna do Access_Key w przypadku odwołań do zmiennych.

```
FIP_MODIF_DESC    ( short Key, short Base, short Offset,
                    unsigned short Word )
```

Funkcja modyfikacji deskryptora. Umożliwia dostęp do poszczególnych słów deskryptora.

```
FIP_DEF_EVT       ( short Access_Key, unsigned short
                    Event_Definition )
```

Funkcja umożliwia wykonanie dynamicznej definicji nowego zdarzenia związanego ze zmienną identyfikowaną przez klucz dostępu.

```
FIP_UPDATE        ( unsigned short Identifier, short
                    Priority)
```

Funkcja realizująca żądanie aperiodycznego transferu zmiennej z określonym priorytetem.

```
FIP_WRITE_MESS    ( short Access_Key, unsigned char
                    *Buffer_ptr, short Size )
```

Funkcja realizująca z poziomu użytkownika zapis komunikatu do bufora transmisyjnego wskazywanego pośrednio przez klucz dostępu.

```
FIP_SEND_MESS     ( short Access_Key, HEADER &Msg_Header )
FIP_READ_MESS     ( short Access_Key, MSG_BUFFER
                    &Msg_Header )
```

Żądanie przesyłu oraz odczyt komunikatu. Struktury HEADER oraz MSG_BUFFER zawierają dane typu: adresy, numery związanych zdarzeń, numer kanału, dane.

```
FIP_START_BA      ( unsigned short Idle_frame_ID )
```

Funkcja uruchomienia arbitra magistrali.

```
FIP_CHANGE_MC     ( unsigned short Macro_Cycl_Addr )
```

Funkcja umożliwiająca przełączenie wykonywania makrocyklu z aktualnego na inny.

```
SEND_ID ()
```

Makroinstrukcja umożliwiająca wytransmitowanie przez arbitra ramki ID_xxx.

```
SEND_APER ()
```

Makroinstrukcja umożliwiająca obsługę aperiodycznych transmisji zmiennych.

```
WAIT ()
```

Makroinstrukcja umożliwiająca synchronizację cyklu elementarnego do określonego czasu.

Druga z bibliotek pakietu FipAccess zawiera zbiór funkcji umożliwiających generowanie komunikacyjnej bazy danych dla stacji abonenckiej. Biblioteka FIPGEN zapewnia:

- konfigurację układu FULLFIP,
- generowanie bazy danych dla zmiennych,

- konfigurację obsługi komunikatów,
- dynamiczną modyfikację baz danych,
- generację programu dla arbitra magistrali,
- obsługę transmisji komunikatów.

FIPGEN zawiera kod biblioteki FIPHANDLER oraz mikrokod FIPCODE.

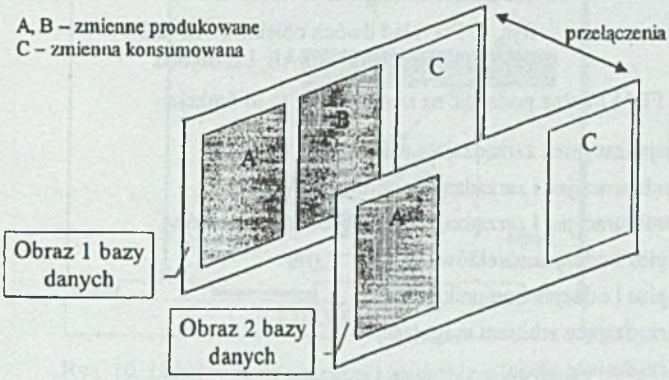
5.3. Biblioteka FDM

Kolejną generacją funkcji umożliwiających programowanie i konfigurację karty sieci FIP opartej na procesorze FULLFIP oraz programie FIPCODE jest pakiet Fip Device Menager, zwany dalej FDM. Obecna jego wersja (FDM v. 2.0) bazuje na bibliotekach FIPHANDLER oraz FIPGEN, producent oprogramowania zapowiada jednak kolejną wersję nie wykorzystującą biblioteki FipAccess. FDM jest pakietem funkcji bibliotecznych przeznaczonym do realizacji połączenia komponentów rodziny FULLFIP z aplikacją użytkownika pracującą na procesorze użytkownika urządzenia takiego, jak: sterownik programowalny, komputer klasy PC, komputerowa stacja robocza, elementy pomiarowe i wykonawcze armatury przemysłowej, itp. Dzięki realizacji takiego połączenia umożliwia przekształcenie danego urządzenia w aktywnego abonenta sieci FIP, arbitra magistrali lub w oba jednocześnie. Pakiet jest dostarczany w wersji źródłowej języka C w formacie ANSI-C. Poniżej przedstawiono opis możliwości pakietu.

- Lokalna inicjalizacja dwuobrazowej komunikacyjnej bazy danych (rys. 8)
- Konfiguracja on-line lokalnej dwuobrazowej komunikacyjnej bazy danych na podstawie obiektów AE_LE,
- Modyfikacja on-line lokalnej dwuobrazowej komunikacyjnej bazy danych opartej na obiektach AE_LE
- Konfiguracja on-line obsługi komunikatów
- Lokalny i zdalny zapis i odczyt zmiennych
- Obsługa statusów ważności zmiennych
- Obsługa zdarzeń
- Obsługa periodycznych i aperiodycznych komunikatów bez potwierdzeń oraz z potwierdzeniem, z możliwością powtórzeń
- Zarządzanie on-line lokalną komunikacyjną bazą danych
- Zarządzanie on-line wymianami periodycznymi zmiennych
- Zarządzanie siecią on-line
- Zarządzanie on-line mechanizmem redundancji medium
- Zarządzanie on-line pracą arbitra magistrali

- Obsługa on–line programów użytkownika na arbitra magistrali
- Testy komunikacji on–line lub off–line

Pakiet umożliwia obsługę abonentów ze zdublowanymi obrazami bazy danych oraz ze zdublowanymi interfejsami komunikacyjnymi. Istnieje możliwość wyboru aktywnego obrazu bazy z poziomu aplikacji użytkownika. Istotny jest fakt, iż nie mamy do czynienia ze zdublowanymi bazami danych, lecz tylko ze zdublowanymi jej obrazami. Oznacza to, że pewne zmienne należące do bazy danych stacji mogą być widoczne na jednym obrazie, natomiast nie widoczne na drugim.

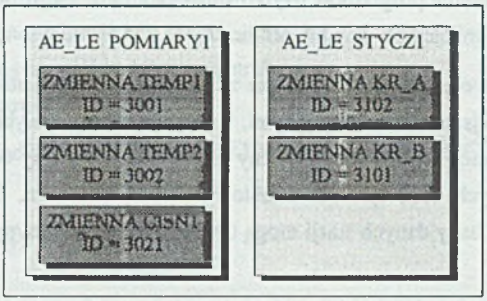


Rys. 8. Dwuobrazowa baza danych
Fig. 8. Two–image data base

Ciekawym rozwiązaniem reprezentacji danych, jakie pojawiło się w pakiecie FDM, są obiekty AE_LE (Application Entity – Link Entity). Dany obiekt AE_LE jest zbiorem zmiennych lub definicji zdarzeń synchronizacyjnych powiązanych z jednym lub dwoma obrazami bazy. Każdy obiekt AE_LE może znajdować się w jednym z następujących stanów:

1) stan konfiguracji,
2) stan zawieszenia,
3) stan aktywności,

Obiekt w stanie pierwszym znajduje się w pamięci użytkownika. Po wykonaniu funkcji AE_LE_DOWNLOAD obiekt zostaje przeniesiony do pamięci prywatnej karty i przechodzi do stanu zawieszenia. Uaktywnienie obiektu wewnątrz wybranej bazy możliwe jest dzięki funkcji AE_LE_START (rys. 9).



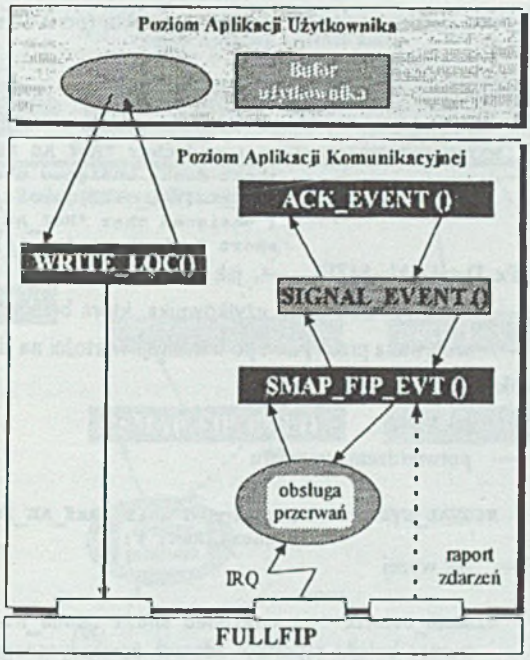
Rys. 9. Przykład dwóch obiektów AE_LE
Fig. 9. Example of two AE_LE objects

Funkcje FDM można podzielić na siedem grup, są to funkcje:

1) inicjalizacyjne i zarządzające redundancją medium,
2) konfiguracyjne i zarządzające obiektami AE_LE,
3) konfiguracyjne i zarządzające przesyłem komunikatów,
4) zapisu i odczytu obiektów AE_LE,
5) zapisu i odczytu komunikatów,
6) zarządzające arbitrem magistrali,
7) zarządzające siecią.

Na przykładzie lokalnego zapisu zmiennej prześledźmy mechanizm współpracy warstw aplikacji użytkownika i komunikacji oraz mechanizm indykacji (rys. 10). Aplikacja użytkownika wywołuje funkcję WRITE_LOC() w celu przeprowadzenia lokalnego zapisu zmiennej. FDM transferuje dane z bufora użytkownika do bufora komunikacyjnego zmiennej. Wartość z tego bufora zostanie rozesłana na sieć dopiero w momencie, gdy arbiter magistrali wyśle odpowiedni identyfikator identyfikujący daną zmienną. Funkcja WRITE_LOC() zwraca typowy raport wywołania. Do momentu wysłania wartości na sieć w temacie zapisu zmiennej stacja będzie milczała. Po wykonaniu transmisji wartości na magistralę rozpocznie się proces indykacji, polegający na uaktywnieniu sygnału przerwania IRQn przez układ FULLFIP. Kolejnym krokiem będzie uaktywnienie funkcji SMAP_FIP_EVT() przez proces obsługi przerwań stworzony przez użytkownika. FDM przeczyta raport zdarzenia z kolejki raportów zdarzeń i wywoła funkcję SIGNAL_EVENT(), będącą specyficzną funkcją użytkownika. Następnie powinno nastąpić wywołanie funkcji ACK_EVENT() w celu poinformowania systemu o potwierdzeniu przyjęcia zdarzenia.

W przypadku zdalnych zapisów zmiennych FDM udostępnia funkcję WRITE_UNIVERSAL() umożliwiającą zarówno zapis zdalny, jak i zapis lokalny. Mechanizm zapisu jest realizowany dwoma sposobami.



Rys. 10. Lokalny zapis zmiennej i mechanizm indykacji
Fig. 10. Local write of variable and transmission indication service

Sposób pierwszy to tzw. *immediate processing*, który niewiele różni się od zapisu lokalnego. Jedynie zamiast funkcji SIGNAL_EVENT() wywoływana jest funkcja użytkownika określona przy wywołaniu WRITE_UNIVERSAL().

Przypadek drugi to tzw. *postponed processing* (rys. 11), polegający na przekazywaniu informacji zwrotnej o zapisie przez wewnętrzny system buforowania. Funkcja SMAP_FIP_EVT() wywołuje funkcję SIGNAL_UPDATE(), która powiadamia warstwę aplikacji użytkownika o nadejściu zdarzenia. Informacja ta jest kolejkowana i w odpowiednim czasie obsługiwana przez funkcję PROCESS_UPDATE(). Dopiero wówczas wywoływana jest funkcja użytkownika podana jako parametr dla WRITE_UNIVERSAL().

Podobne mechanizmy są zastosowane w przypadku funkcji odczytu lokalnego i zdalnego zmiennej oraz w przypadku transmisji i odbioru komunikatów.

Poniżej zamieszczono przykłady definicji niektórych z omówionych powyżej funkcji.

```
WRITE_LOC        ( unsigned char *Ref_AE_LE, unsigned
                   short Rank, unsigned char
                   Tx_Data[VAL_SIZE] );
```

Ref_AE_LE — wskaźnik do odpowiedniego obiektu AE_LE znajdującego się w stanie aktywnym

Rank — zakres zmiennych wewnątrz obiektu (od 0 do maksymalnej liczby zmiennych w obiekcie)

Tx_Data[VAL_SIZE] — bufor zawierający dane do zapisu

```
WRITE_UNIVERSAL ( unsigned char *Ref_AE_LE, unsigned
                  short Rank, unsigned char
                  Tx_Data[VAL_SIZE], void User_Function
                  ( unsigned char *Ref_AE_LE, unsigned
                  short Rank,short Ack ) );
```

Ref_AE_LE, Rank, Tx_Data[VAL_SIZE] — jak wyżej

User_Function() — wskaźnik do funkcji użytkownika, która będzie automatycznie wywoływana przez FDM po transmisji wartości na sieć

Parametry dla funkcji użytkownika:

Ref_AE_LE, Rank — jak wyżej

Ack — potwierdzenie transferu

```
SIGNAL_EVENT    ( unsigned char *Ref_AE_LE, unsigned
                  short Rank );
```

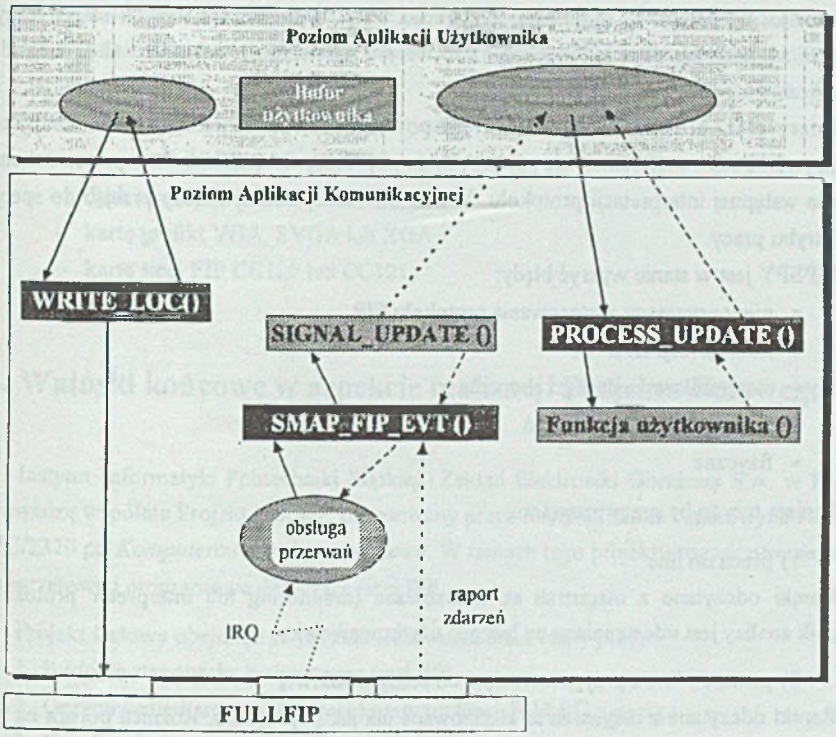
Ref_AE_LE, Rank — jak wyżej

```
SIGNAL_UPDATE   ( unsigned short Queue_Nr );
```

Queue_Nr — numer kolejki raportów zdarzeń aperiodycznego bufora transferu; istnieją trzy kolejki zdarzeń: dla przesyłów aperiodycznych pilnych, normalnych oraz aperiodycznych przesyłów zmiennych systemowych służących do obsługi sieci.

```
SMAP_FIP_EVT    ()
```

Funkcja bezparametrowa



Rys. 11. Zapis uniwersalny zmiennej z indykacją *postponed*
Fig. 11. Uniwersal write of variable with *postponed* processing

5.4. Program narzędziowy FIPSPY

Program narzędziowy FIPSPY służy do analizy pracy sieci FIP. Pakiet FIPSPY pracuje pod kontrolą systemu operacyjnego UNIX. Ułatwia on laboratoryjne tworzenie i kontrolę systemów wykorzystujących magistralę FIP oraz ich instalację i utrzymanie na rzeczywistym obiekcie przemysłowym. Działanie programu opiera się na podglądaniu ramek przesyłanych magistralą i ich analizie. FIPSPY może pracować tylko w trybie podglądu. Jest on przeznaczony do testowania wszystkich systemów podłączonych do sieci bez jakichkolwiek zmian i modyfikacji konfiguracji tej sieci. Nie jest zatem konieczne tworzenie nowego abonenta, służącego do testów. FIPSPY jest całkowicie przeźroczysty dla testowanego systemu. Obecnie najnowsza wersja tego pakietu przeznaczona jest dla śledzenia pojedynczej magistrali i testowania jedynie warstwy drugiej systemu dla prędkości transmisji 31.25 kb/s lub 1 Mb/s. Umożliwia ona analizę danych przez tzw. monitoring lub interpretację protokołu.

Monitoring polega na obserwacji aktywności sieci i wyświetlaniu danych statystycznych i historii zdarzeń. Możliwe jest uzyskanie informacji o starcie sieci, liczbie transakcji względem typu, liczbie błędów, itp.

Interpreter protokołu umożliwia grupowanie ramek w elementarne transakcje i wykrywanie sekwencji ramek nie pasujących do poprawnych transakcji. Interpreter dokonuje jedynie wstępnej interpretacji protokołu. W celu dokładnej analizy należy przejść do specjalnego trybu pracy.

FIPSPY jest w stanie wykryć błędy:

- niepoprawnego zastosowania protokołu FIP
- konfiguracji sieci
- stanu milczenia danego abonenta
- periodyczności odświeżeń
- fizyczne

Istnieją trzy tryby pracy programu:

1) praca on line

Ramki odczytane z magistrali są analizowane (monitoring lub interpreter protokołu), a wynik analizy jest udostępniany na bieżąco użytkownikowi.

2) praca on line z przygotowaniem do pracy wsadowej

Ramki odczytane z magistrali są analizowane tak jak poprzednio. Różnica polega na tym, że ramki przed analizą są składowane w specjalnym zbiorze w pamięci masowej. Przedział czasu, w którym będzie dokonywany zapis do zbioru i interpretacja, jest definiowany. Definicja może dotyczyć czasu rozpoczęcia i zakończenia obserwacji oraz ustawienia specjalnego filtru umożliwiającego rozpoczęcie obserwacji po wykryciu odpowiedniego znacznika w określonej transakcji. Istnieją trzy typy filtrów. Pierwszy dotyczy wyłapywania określonego rodzaju transakcji sieciowych, drugi możliwości rozpoczęcia lub zakończenia zapisu po wystąpieniu określonego zdarzenia, natomiast trzeci umożliwia wydobycie jednej określonej części zapisu komunikacji.

3) analiza wsadowa

Realizowana jest tak samo jak analiza on line, lecz źródłem danych nie jest magistrala, tylko wcześniej stworzony zbiór danych przy użyciu pracy w trybie drugim. Ten rodzaj analizy ma przewagę na analizą on line, gdyż pozwala na analizę danych zapamiętanych w pewnym przedziale czasu. Analiza on line umożliwia jedynie dostęp do informacji chwilowej, dostępnej aktualnie na sieci.

Aby możliwa była instalacja i wykorzystanie pakietu, należy spełnić następujące wymagania sprzętowe i programowe:

- konieczne jest posiadanie systemu UNIX SCO ODT 3.0 oraz serwera X11,
- konieczne jest posiadanie pakietu FIPSPY wraz z zabezpieczającym kluczem sprzętowym,
- dowolny komputer 486 PC z magistralą EISA,
- minimum 12 Mb pamięci RAM,
- dysk twardy (minimum 210 Mb),
- kartę grafiki VGA, SVGA lub XGA,
- kartę sieci FIP CC120 lub CC121.

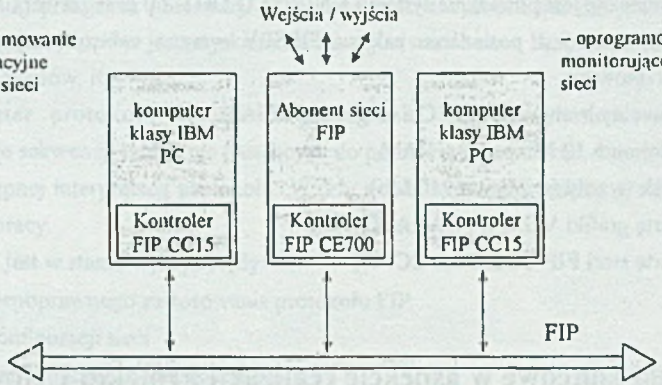
6. Wnioski końcowe w aspekcie realizacji Projektu Celowego

Instytut Informatyki Politechniki Śląskiej i Zakład Elektroniki Górniczej S.A. w Tychach prowadzą wspólnie Projekt Celowy, finansowany przez Komitet Badań Naukowych Nr 88535 95C/2370 pt. *Komputerowa sieć przemysłowa*. W ramach tego projektu rozwiązywane są prace sprzętowe i programowe dotyczące sieci FIP.

Projekt Celowy obejmuje swym zakresem następujące etapy pracy:

1. Budowę stanowiska badawczego sieci FIP
2. Opracowanie karty sprzęgającej komputer klasy IBM PC z siecią
3. Uruchomienie oprogramowania sieci FIP w oparciu o istniejące moduły programowe i opracowane we własnym zakresie programy
4. Ustalenie zasad realizacji pierwszej w Polsce pilotowej instalacji przemysłowej opartej o sieć FIP.

Na rysunku 12 przedstawiono zrealizowane stanowisko badawcze sieci FIP. W stanowisku tym zastosowano oryginalne kontrolery sieci FIP produkcji firmy CEGELEC – Francja, jak również oprogramowanie sieci bazujące również na rozwiązaniach tej samej firmy. Program monitorujący pracę sieci pozwala na badanie większości zjawisk, które zachodzą w sieci FIP. Na tym stanowisku można uruchamiać własne rozwiązania sprzętowe i programowe.

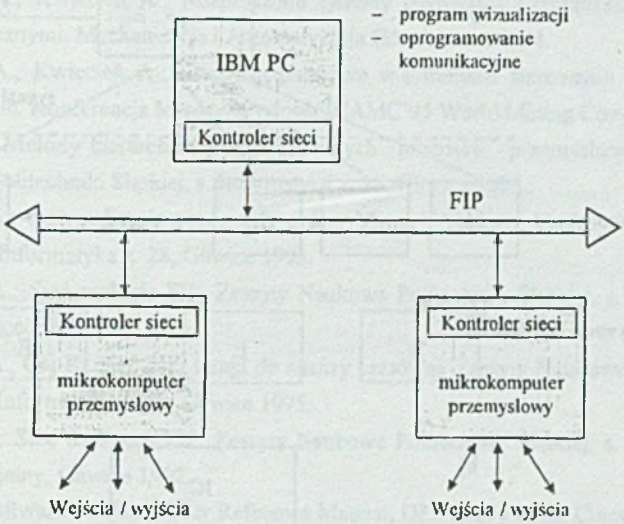


Rys. 12. Stanowisko badawcze sieci FIP
Fig. 12. Research stand of FIP network

W zakresie prac sprzętowych zaprojektowano kartę kontrolera sieci, której koncepcja budowy przedstawiona jest w niniejszym artykule.

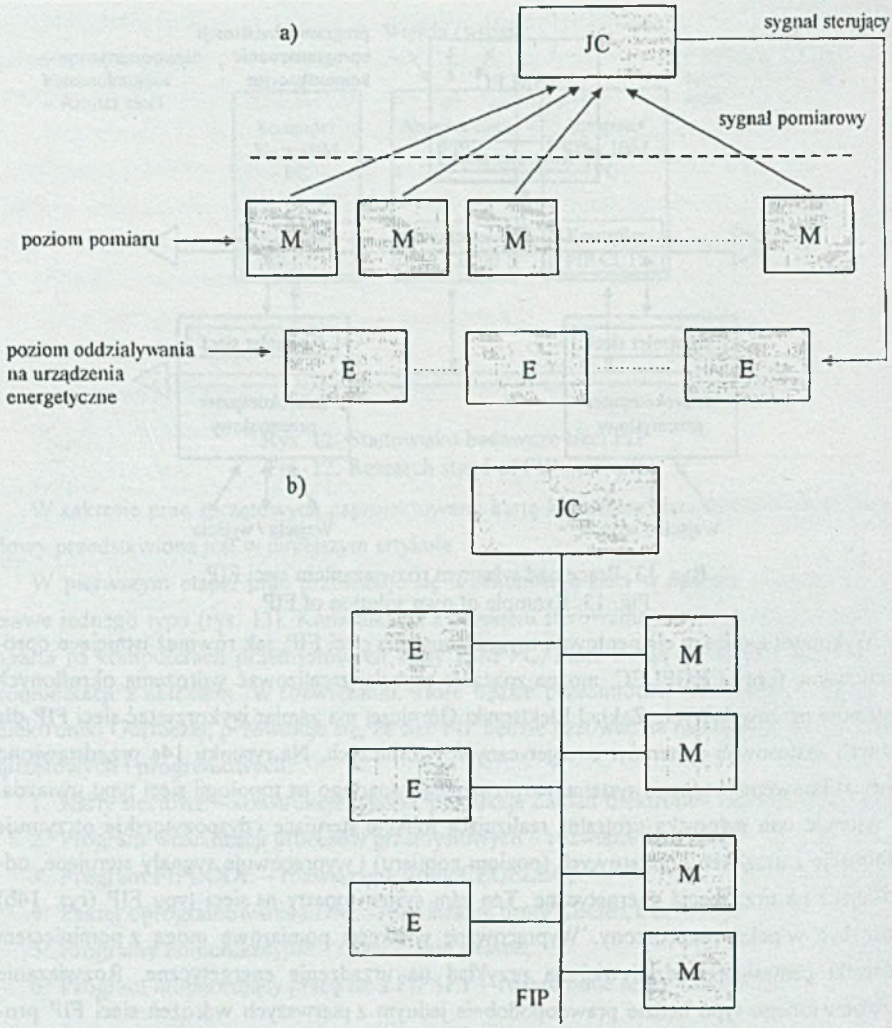
W pierwszym etapie prac przewiduje się uruchomienie sieci w oparciu o kontrolery sieciowe jednego typu (rys. 13). Komunikacja z obiektem sterowania w tym rozwiązaniu będzie oparta na komputerach przemysłowych klasy IBM PC, które mogą zawierać różne moduły komunikacji z obiektem. W rozwiązaniu, które będzie przedmiotem wdrożenia w Zakładzie Elektroniki Górniczej, przewiduje się, że sieć FIP będzie bazować na następujących modułach sprzętowych i programowych:

1. Karty sieciowe – konstrukcja własna, produkcja Zakład Elektroniki Górniczej S.A.;
2. Program wizualizacji procesów przemysłowych – rozwiązanie własne;
3. Program FIPCODE – rozwiązanie firmy CEGELEC;
4. Pakiet oprogramowania FDM – rozwiązanie firmy CEGELEC;
5. Programy komunikacyjne – rozwiązanie własne;
6. Program monitorujący pracę sieci FIPSPY – rozwiązanie firmy CEGELEC.



Rys. 13. Prace nad własnym rozwiązaniem sieci FIP
Fig. 13. Example of own solution of FIP

Wykorzystując bazę elementową przygotowaną dla sieci FIP, jak również istniejące oprogramowanie firmy CEGELEC, można znacznie szybciej zrealizować wdrożenia określonych systemów przemysłowych. Zakład Elektroniki Górniczej ma zamiar wykorzystać sieci FIP dla różnych zastosowań systemów energetycznych i górniczych. Na rysunku 14a przedstawiono schemat konwencjonalnego systemu rozproszonego opartego na topologii sieci typu gwiazda. W systemie tym jednostka centralna realizująca funkcje sterujące i dyspozytorskie otrzymuje informację z urządzeń pomiarowych (poziom pomiaru) i wypracowuje sygnały sterujące, oddziałujące na urządzenia energetyczne. Ten sam system oparty na sieci typu FIP (rys. 14b) może być w pełni rozproszony. Wypracowane wielkości pomiarowe mogą z pominięciem jednostki centralnej oddziaływać na przykład na urządzenie energetyczne. Rozwiązanie przedstawionego typu będzie prawdopodobnie jednym z pierwszych wdrożeń sieci FIP prowadzonych przez Zakład Elektroniki Górniczej.



Rys. 14. Systemy dyspozytorskie oparte na rozwiązaniach konwencjonalnych (a) i na sieci FIP (b)

Fig. 14. Standard solution of distributed system (a) and FIP network (b)

LITERATURA

[1] Praca zbiorowa: Rozproszone systemy komputerowe. Pronet, Gliwice 1994.

[2] Grzywak A., Kwiecień A.: Perspektywy rozwoju zastosowań sieci komputerowych w górnictwie. Mechanizacja i Automatyzacja Górnictwa nr 5, 6/94.

[3] Grzywak A., Kwiecień A.: Rozproszone systemy sterowania i zarządzania procesami technologicznymi. Mechanizacja i Automatyzacja Górnictwa nr 8/94.

[4] Grzywak A., Kwiecień A.: Sieci komputerowe w systemach sterowania i zarządzania w górnictwie. Konferencja Międzynarodowa ICAMC'95 World Mining Congress.

[5] Cupek R.: Metody hierarchizacji rozproszonych procesów przemysłowych. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 28, Gliwice 1995.

[6] Bigewski Z.: Optymalizacja pracy sieci przemysłowych. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 28, Gliwice 1995.

[7] Kwiecień A.: Sieć rozległa FIP. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 24, Gliwice 1993.

[8] Kwiecień A., Gaj P.: Sieć FIP, wstęp do analizy czasowej. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka z. 28, Gliwice 1995.

[9] Chapuis H.: Sieć terenowa FIP. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka Zeszyt specjalny, Gliwice 1992.

[10] FIPGEN Software version 4 User Reference Manual; DPS 50254 a-en, Clamart 1992.

[11] FIPHANDLER Software version 4 User Reference Manual; DPS 50253 a-en, Clamart 1992.

[12] FIPCODE Software version 5 User Reference Manual; DPS 50263 b-en, Clamart 1994.

[13] FIP NETWORK General Introduction; DPS 50249 aA, Clamart 1990.

[14] FIP DEVICE MENAGER Software Version 2 User Reference Manual; DPS 50269 a-em, Clamart 1994.

[15] FIP DEVICE MENAGER Software Version 2 User Reference Manual; DPS 50255 a-en, Clamart 1992.

[16] FIPSPY Version 2 User Reference Manual; DPS 50275 b-en, Clamart 1995.

[17] CC120/1/2 Board User Manual; DPS 50267 b-en, Clamart 1994.

[18] FIELDRIVE Component Data Sheet; DPS 50261 a-en, Clamart 1994.

[19] FULLFIP2 component User Reference Manual; DPS 50262 a-en, Clamart 1994.

[20] FULLFIP2-LP component User Reference Manual; DPS 50270 a-en, Clamart 1994.

[21] FIP STARTER KIT User Manual; DPS 50252 b-en, Clamart 1993.

[22] Klubowe materiały szkoleniowe; 2 tomy WorldFIP, Clamart 1995.

[23] FIP STARTER KIT materiały klubowe WorldFip; Clamart 1995.

Recenzent: Dr hab. inż. Adam Mrózek

Wpłynęło do Redakcji 4 stycznia 1996 r.

Abstract

The paper describe the problems connected witch software and hardware solutions of FIP networks.

The first part of article describes the chips base used for FIP controllers (network communication coprocessor, fully integrated line driver circuit, bus redundancy management device, line isolation transformer). The next part include the project of hardware FIP board and information about software solutions.

The last part of article describes the state of hardware and software job which realise the Politechnical Institute together with Zakład Elektroniki Górniczej SA factory from Tychy.