BIULETYN TECHNICZNY

MIER BUDA

12(226)
1980

Warunki prenumeraty

Jednostki gospodarki uspołecznionej, instytucje, organizacje i wszelkiego rodzaju zakłady pracy zamawiają prenumeratę w miejscowych Oddziałach RSW "Prasa-Książka-Ruch", w miejscowościach zaś, w których nie ma Oddziałów RSW — w urzędach pocztowych. Czytelnicy indywidualni opłacają prenumeratę wyłącznie w urzędach pocztowych i u doręczycieli. Prenumeratę roczną w cenie 516 zł należy zamawiać do 25 listopada na rok następny, półroczną do 10 czerwca na II półrocze.

ZJEDNOCZENIE PRZEMYSŁU AUTOMATYKI
I APARATURY POMIAROWEJ „MERA"

„MERA"

BIULETYN PRZEMYSŁU KOMPUTEROWYCH SYSTEMÓW AUTOMATYZACJI I POMIARÓW

WARSZAWA, GRUDZIEŃ 1980

SPIS TREŚCI

Opracowanie Redakcyjne: Redakcja Biuletynu "Mera", ul. Patriotów 77, 04-950 Warszawa /tel. 12-41-71/. Wydawca: Przedsiębiorstwo Automatyki Przemysłowej "Mera-Pnefal" ul. Poezji 19. 04-994 Warszawa. Zam. 12/81. 2300 egz.

inż. BOGUSŁAW GREGORCZYK
ZUE "Unitra-Unima"

MIKROKOMPUTEROWY SYSTEM DO ZBIERANIA I REJESTRACJI DANYCH MST-1/DA

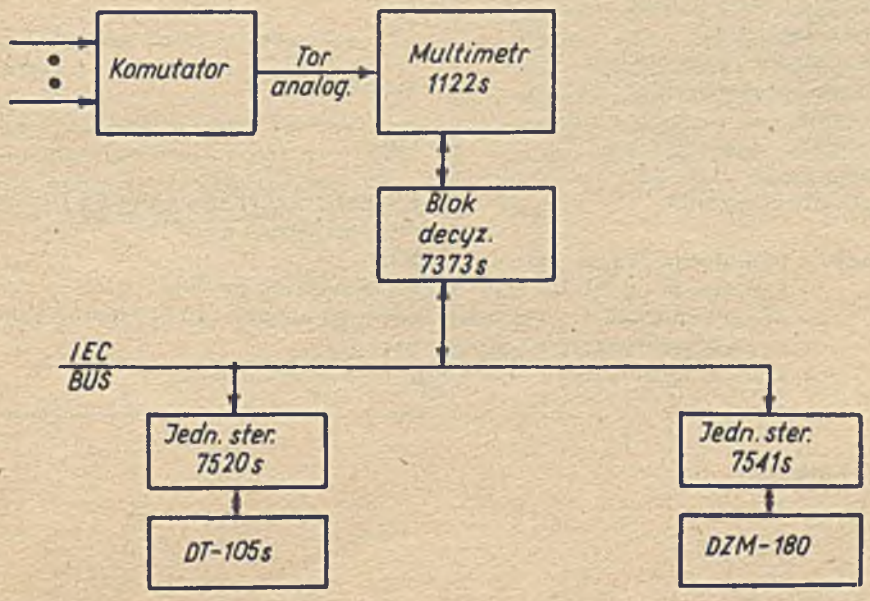
Od Redakcji

Przedstawiamy cykl artykułów poświęconych zautomatyzowanym systemom pomiarowym i badawczym z międzynarodowym interfejsem IEC-625 /wg RWPG ISP-2/.

W kraju pierwszą placówką, która opracowała i wdrożyła zestawy pomiarowe z interfejsem IEC-625 /wg USA IEEE 488/ były zakłady UNITRA-UNIMA. Z tego względu cykl ten otwieramy artykułami przygotowanymi przez inżynierów z zakładów UNITRA-UNIMA.

Bloki systemowe MST-1 wyposażone są w interface wg zaleceń IEC-625 /TSP-2/. Są one uniwersalne pod względem zastosowań. Można je wykorzystywać jako oddzielne urządzenia lub łączyć w określone zestawy przeznaczone dla określonych zadań. Jednym z takich zestawów jest znajdujący się aktualnie w opracowaniu zestaw MST-1/DA. Przeznaczony jest on do zbierania i rejestracji danych. Zestaw MST-1/DA umożliwia wykonywanie pomiaru napięć w wielu różnych punktach, a także dzięki zastosowaniu odpowiednich przetworników innych wielkości elektrycznych i nieelektrycznych oraz dokumentowanie wyników.

Zestaw MST-1/DA jest przeznaczony do następujących celów:
- kontroli procesów technologicznych;
- okresowego sprawdzenia zachowania się obiektów /np. budowlanych, drogowych, energetycznych/,

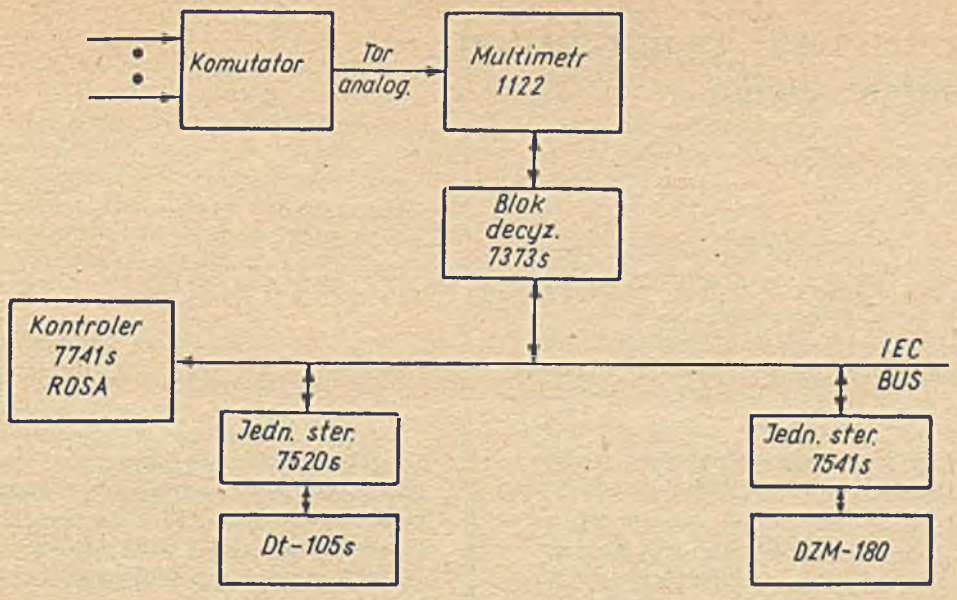


Rys.1. Zestaw MST-1/DA/wersja bez kontrolera/

Parametry podstawowych Bloków zestawu MTS-1/DA

Kontroler systemowy 7741s - ROSA	
- pojemność pamięci EPROM	- 16K x 8 bitów
- pojemność pamięci RAM	- 16K x 8 bitów
- pojemność pamięci kasetowej	- 160K x 8 bitów /na jednej stronie taśmy/
- typ mechanizmu	- PK-1
- oprogramowanie	- ASSEMBLER BASIC[x], EDYTOR, FORTRAN[x], MONITOR
- szybkość przekazywania danych	- 70 Kbajtów/s
Woltomierz cyfrowy typu 1122/V 542/	
- wskazania	- czterocyfrowe, maks. 9999
- zakresy pomiarowe	- ± 0,1; 1; 10; 100; 1000V
- dokładność	- 0,1%
- maks. czułość	- 10 V
- rezystancja wejściowa	- 1G dla 0,1 i 1V 10M dla 10, 100 i 1000V
- maks. szybkość wykonywania pomiarów	- 2 0/s
- rodzaj wejścia	- "pływające"
- wyzwalanie pomiaru	- ręczne, zdalne i automatyczne
- przełączanie zakresu	- ręczne, zdalne i automatyczne
- wskazania polaryzacji	- automatyczne
Komutator typu 1933 S	
- maks. liczba kanałów	- 200
- liczba przewodów w kanale	- 2 lub 4
- maks. napięcie sygnału przełączanego	- 100V
- maks. prąd sygnału przełączanego	- 100mA
- maks. moc sygnału przełączanego	- 3W
- rezystancja toru pomiarowego	- 1
- rezystancja izolacji toru pomiarowego w stosunku do masy	- 10^9
- pojemność toru pomiarowego w stosunku do masy	- 800pF
- pojemność między sąsiednimi torami pomiarowymi	- 12pF
- częstotliwość przełączania	- 1 ÷ 10Hz programowana co 1Hz 10÷100 Hz programowana co 10Hz
- rodzaje pracy:	- komutacja ciągła od N1 do N2. - komutacja od N1 do N2 z zatrzymaniem po N2, - programowane wybieranie żądanego kanału i przewodu w kanale, - ręczne wybieranie żądanego kanału, - ręczne wymuszanie komutacji /co jeden kanał/.

x/ w opracowaniu



Rys. 2. Zestaw MST-1/DA/wersja z kontrolerem/

- szybkiej rejestracji stanów panujących w badanym obiekcie,
- kontroli dostaw.

Konfiguracja zestawu MST-1/DA zależy od potrzeb. Najprostszy zestaw składa się /rys.1/ z:
- komutatora
- multimetru 1122 wraz z blokiem decyzyjnym 7373s; /V 542/.

Użytkownik może wykorzystać w zestawie zamiast multimetru 1122 wraz z blokiem decyzyjnym 7373s produkcji Unimy produkowany przez "Merę" multimetr V542 wraz z interfejsem I 542.
- drukarki DZM-180 wraz z jednostką sterującą 7541s,
lub
- perforatora DT-105s wraz z jednostką sterującą 7520s.

W tej konfiguracji komutator/po załączeniu kolejnego kanału wyzwala pomiar. Po jego zakończeniu wynik rejestrowany jest na drukarce lub na taśmie perforowanej. Po zakończeniu wysyłania wyniku blok decyzyjny wysyła sygnał inicjujący załączenia kolejnego kanału. Wadą tego zestawu jest wybieranie kanałów po kolei jak i rejestracja wszystkich wyników pomiarów bez wyjątku.

Znacznie większe możliwości zapewnia zestaw sterowany przez kontroler 7741s-ROSA /rys. 2/. Kontroler ten zbudowany jest na bazie mikroprocesora 8080 A. Istnieje możliwość programowania kontrolera na poziomie assemblera. Po zakończeniu prac badawczych wystąpi możliwość programowania w językach wyższego rzędu: BASIC bądź FORTRAN. Umożliwi to obróbkę, interpretację wyników, uzyskanie różnego rodzaju wydruków. Użycie kontrolera umożliwia sygnalizację przekroczenia dopuszczalnych wartości, które mogą być różne dla różnych kanałów. Kontroler pozwala też mierzyć szybkość zmian wielkości mierzonej i sygnalizację zbyt szybkich jej zmian. W tym wypadku można sygnalizować zbyt szybkie narastanie wielkości mierzonej choć aktualna jej wartość jest mniejsza od dopuszczalnej.

mgr inż. MICHAŁ CHRUDZIMSKI
ZUE "Unitra-Unima"

BLOKI STERUJĄCE ZESTAWÓW MST-1

Najprostszym blokiem sterującym szyny interfejsu IEC-625 opracowanym w ZUE UNITRA-UNIMA jest blok 7111s. Jest on przeznaczony do sterowania prostym systemem kontrolno-pomiarowym, w którym wymagany jest tylko jeden kierunek transmisji informacji. Założenie to pozwala na radykalne uproszczenie układów sterowania i sprzęgu /IEC-625/ wszystkich bloków systemu, a bloku sterującego w szczególności. Duże zapotrzebowanie przemysłu na tego typu proste systemy /POLCOLOR/ dało ekonomiczne podstawy celowości opracowania i wdrożenia ich do produkcji.

Najwcześniej opracowany blok sterujący 7341s jest przeznaczony dla systemów, w których realizuje się transmisję informacji pomiędzy blokami pomiarowymi, ale nie zachodzi potrzeba przetwarzania tej informacji /np. system MST-1/MSI-24/. Pierwsze zastosowania przemysłowe wykazały niską efektywność systemu jako całości, spowodowaną głównie małą szybkością urządzeń peryferyjnych. Radykalną poprawę tego parametru osiągnięto poprzez zastosowanie półprzewodnikowej pamięci przechowującej program pomiarowy. W eksploatowanych obecnie w zakładach TESLI systemach MST-1/MSI-24 czynnikiem ograniczającym szybkość testowania mikroukładów małej i średniej skali integracji jest wydajność podajników. Równocześnie z zatwierdzeniem planów produkcyjnych NPCP-CEMI w zakresie mikroprocesów podjęto opracowanie Kontrolera systemu w oparciu o układy INTEL 8080A.

Blok sterujący 7111s

Blok 7111s realizuje sterowanie szyną interfejsu IEC-625 zgodnie z programem zapisanym w półprzewodnikowej pamięci stałej. Jest to wymienna pamięć o maksymalnej pojemności 4k bajtów zbudowana z mikroukładów typu EPROM /INTEL 1702/.

Od strony interfejsu IEC-625 blok realizuje funkcję NADAWCA i część funkcji CONTROLER umożliwiając programowe sterowanie liniami interflejsu ATN i IEC. Niewątpliwą zaletą przyjętej organizacji bloku jest prostota programowania. Sześć wyróżnionych kodów jest interpretowanych jako rozkazy specjalne realizowane przez blok, wszystkie pozostałe są traktowane jako bajty informacji i w niezmienionej postaci transmitowane na szynę interfejsu zestawu.

Dwa spośród rozkazów specjalnych powodują zatrzymanie pracy bloku /stop/ przy czym jeden z nich powoduje automatyczne wyzerowanie licznika adresu pamięci /koniec programu/. Wznowienie pracy /start/ wymaga naciśnięcia odpowiedniego przycisku na płycie czołowej bloku lub przesłania odpowiedniego sygnału do gniazda na płycie tylnej bloku. A zatem poza sprzęgiem IEC system sterowany blokiem 7111s wymaga zastosowania prostej dwuprzewodowej szyny po której przesyłane są sygnały startu od wolno działających urządzeń wykonawczych lub operatora systemu. Płyta czołowa bloku umożliwia między innymi wymuszenie pracy krokowej, zatrzymanie programu na wskazanym adresie i skok do wskazanego adresu pamięci programu.

Blok sterujący 7341s

Blok 7341s realizuje sterowanie szyną interfejsu zgodnie z programem napisanym w języku ULAN 1/TR i programem stałym, określającym reakcję na sygnały "żądanie obsługi - SRQ" w ramach funkcji interfejsu "Szeregowa Kontrola". Blok nie posiada wewnętrznej pamięci programu. Program sterujący jest sukcesywnie pobierany z urządzenia peryferyjnego, którym może być czytnik taśmy papierowej CT-2200 lub blok: Pamięć Kasetowa 7720s. Od strony interfejsu IEC-625 blok realizuje wszystkie, poza "Równoległą Kontrolą" funkcje interfejsu. Ponieważ nie ma możliwości przetwarzania informacji funkcja interfejsu "odbiorca" jest wykorzystana do kontroli transmisji.

Sterowanie bloku zapewnia automatyczne wznowienie wstrzymanego wykonywania programu po wykryciu końca zaprogramowanej uprzednio transmisji. Układowy sposób rozwiązania obsługi przerwań systemowych zakłada ustalony porządek pobierania statusów

urządzeń systemu i identyczną ich interpretację. Brak przetwarzania informacji i wewnętrznej pamięci pozostawia tylko jedną możliwość modyfikacji programu - skok w przód. Skok następuje do miejsca określonego programem w momencie gdy wynik procesu pomiarowego jest już przesądzony. Mechanizm ten jest zrealizowany w programie stałym obsługi przerwań systemowych a jego wykorzystanie zwiększa efektywność systemu.

Bloki sterujące 7341s/7342s

Jak wspomniano we wstępie wyposażenie zestawu MST-1 w półprzewodnikową pamięć programu pozwoliło na radykalne zwiększenie szybkości jego działania. Wybrano rozwiązanie konstrukcyjne, w którym pamięć zrealizowano w postaci osobnego bloku 7342s, głównie w celu uzyskania dodatkowych funkcji zespołu sterującego:

- możliwość kontroli poprawności wprowadzania programu do pamięci,
- możliwość automatycznej eliminacji sekwencji programu nieistotnych dla procesu pomiarowego,
- możliwość poprawiania programu z płyty czołowej.

Zapewniono zarówno możliwość wymiany znaków jak i wprowadzania znaków z przesunięciem poprzedniej zawartości pamięci,

- możliwość uzyskania mapy programu, tj. listy etykiet programu uzupełnionej adresami ich lokalizacji w pamięci,
- wprowadzono dodatkowy rodzaj pracy krokowej - stop na zadanym adresie,
- uzyskano możliwość realizacji automatycznego testu autokontrolnego sprawdzającego większość układów funkcjonalnych bloku. Test może obejmować układy transmisji bloku 7341s.

W standardowym wykonaniu blok jest wyposażony w pamięć półprzewodnikową o pojemności 16k bajtów. W systemie MST-1/MSI-24 blok włączany jest pomiędzy urządzeniem peryferyjnym a sterownikiem 7341s. Program pomiarowy poprzez urządzenie peryferyjne zostaje wprowadzony do pamięci. Następnie zawartość pamięci /od adresu zerowego do komórki zawierającej ustalony znak końca/, jest cyklicznie przesyłana do bloku 7341s interpretującego program. Współpracę z urządzeniami peryferyjnymi i blokiem sterującym zapewniają trzy układy interfejsu, w które blok jest wyposażony.

- wejściowy CT 2200 do współpracy z czytnikiem taśmy papierowej,
- wejściowy IEC-625 /tylko uproszczona funkcja "odbiorca"/ do współpracy z blokiem Pamięci Kasetowej 7720s,
- wyjściowy CT 2200 do współpracy ze sterownikiem 7341s.

Mikrokomputer 7741s ROSA

Kontroler systemowy 7741s powstał w oparciu o system mikroprocesorowy 8080 mikrokomputera MKT-2 opracowanego w PIE i WAT. Z uwagi na specyfikę przewidywanych zastosowań kontrolera system ten znacznie rozbudowano. Opracowane układy są na tyle złożone, że należy je traktować jak programowane kanały a nie porty we/wy:

1. Kanał operatora zawiera 16-znakowy display alfanumeryczny oraz klawiatury uniwersalną i funkcjonalną. Display wyposażony jest w generator 128-znaków alfanumerycznych i pamięć umożliwiającą podtrzymywanie obrazu. Generator znaków wykorzystuje pamięć reprogramowalną, co zapewnia możliwość łatwej zmiany repertuaru znaków. Klawiatura uniwersalna zawiera pełny zestaw znaków kodu ISO-7. Funkcje klawiatur specjalizowanych określone są programowo. Obsługa klawiatur może być realizowana poprzez system przerwań.
2. Kanał interflejsu /IEC-625/ został opracowany pod kątem uproszczenia procedur programowania i maksymalnego zwiększenia szybkości sterowania szyną interfejsu. Między innymi zastosowano wewnętrzny 256-bajtowy bufor informacji stanowiący specyficzne przedłużenie pamięci operacyjnej mikroprocesora. Przyjęty sposób sterowania kanałem w powiązaniu z odpowiednią strukturą programu, umożliwia realizowanie złożonych sekwencji sterowania systemem pomiarowym bez angażowania mikroprocesora w te procedury. Niewątpliwą zaletą przyjętego rozwiązania jest duża podatność kanału na wewnętrzne testy kontrolne.
3. Kanał pamięci kasetowej PK-1 opracowano z uwzględnieniem identycznych wymagań funkcjonalnych jak dla kanału IEC. Dodatkowym problemem, którego rozwiązanie spowodowało znaczną rozbudowę układów była konieczność przystosowania pamięci PK-1 do roli zewnętrznej pamięci mikrokomputera, oraz realizacja procedur kontroli wierności zapisu informacji.

mgr LECH SZYNGWELSKI
ZUE "Unitra-Unima"

OPROGRAMOWANIE ZESTAWÓW MST-1

Oprogramowanie zestawów MST-1 jest determinowane przez dwa zasadnicze czynniki: możliwości układowe bloków sterujących oraz wymagania związane z konkretną aplikacją zestawu. Szczególnie istotnym czynnikiem są możliwości układowe bloków sterujących. W przypadku prostych bloków sterujących, takich jak Blok Sterujący 7111s czy Pamięć Systemowa Buforowa 7342s, możliwości programowania są niewielkie. Sytuacja zmienia się całkowicie, gdy blok sterujący umożliwia swobodne programowanie, tak jest w przypadku mikrokomputera ROSA 7741s.

Oprogramowanie zestawów MST-1 z prostym blokiem sterującym

Na sterowanie pracą zestawu MST-1 składają się następujące elementy:

- wysyłanie w linie interfejsu znaków wysterowujących /programujących/ urządzenia wchodzące w skład zestawu,
- kontrolowanie transmisji znaków na liniach DIO interfejsu,
- przetwarzanie informacji napływających do bloku sterującego z urządzeń zewnętrznych /połączonych z blokiem sterującym przez interfejs IEC bądź przez interfejs opracowany specjalnie dla danego urządzenia/.

Blok Sterujący 7111s umożliwia realizację jedynie pierwszego z podanych wyżej elementów. Program stanowi szereg bajtów zapisanych w pamięci typu PROM o maksymalnej wielkości równej 4K. Pobierane kolejno bajty z pamięci są wysyłane w linie DIO interfejsu z tym, że kilka wybranych znaków jest interpretowanych jako znaki sterujące.

Poszczególne znaki sterujące umożliwiają:
- zmianę stanu linii ATN, co zezwala na wysyłanie w linie DIO adresów urządzeń bądź rozkazów interfejsu /przy aktywnej linii ATN/, względnie bajtów danych /przy wyzerowanej linii ATN/,
- wysyłanie impulsu na linii IFC, co powoduje odadresowanie urządzeń /zarówno typu nadawca jak i odbiorca/,
- wstrzymanie /do momentu naciśnięcia odpowiedniego klucza/ bądź zakończenia wykonywania programu.

Programowe sterowanie pracą zestawu polega więc na przesłaniu do urządzeń danych ustawiających takie parametry jak zakres, częstotliwość, rodzaj pracy a następnie zainicjowanie ich pracy/ inicjacja pomiaru, transmieji itp./. Nieco rozbudowanym blokiem sterującym jest Pamięć Systemowa Buforowa 7342s. Program jest pamiętany w pamięci typu RAM o maksymalnej pojemności 16 K. Język programowania w tym przypadku został nazwany UI.ANI/TR. Program pisze się przy pomocy kodów mnemonicznych. Ponadto czytelność programu znacznie zwiększają następujące elementy:
- stosowanie etykiet do oznaczenia kolejnych wierszy programu; podczas wykonywania programu aktualna etykieta jest wyświetlana na płycie czołowej,
- dopuszczalne jest stosowanie komentarzy /dowolnych ciągów znaków ograniczonych znakami dwukropka/,
- w linie DIO interfejsu można wysyłać bez interpretacji dowolny ciąg znaków ograniczonych znakami cudzysłowu.

Z funkcjonalnego punktu widzenia, możliwości bloku sterującego 7111s zostały rozszerzone o następujące elementy:
- programowe wstrzymywanie na określony czas wykonywania programu /oczekując np. na ustawienie się w urządzeniu programowanych parametrów czy też dokonanie pomiaru/,
- układowa obsługa przerwań zgłaszanych przez urządzenia przy pomocy linii SRQ. W wyniku obsługi przerwania, w zależności od napisanego programu, Pamięć Systemowa Buforowa może wznowić interpretację wstrzymanego wcześniej programu /wznowienie interpretacji może być poprzedzone pominięciem określonego fragmentu programu/ bądź zaalarmować operatora jeżeli przerwanie jest wynikiem zdarzenia określonego jako nieprawidłowe. Ważną cechą jest możliwość oczekiwania na zakończenie transmisji w interfejsie, sygnalizowanego przy pomocy odpowiedniego przerwania. W dalszym ciągu nie jest możliwe jakiekolwiek przetwarzanie informacji.

Oprogramowanie zestawów MST-1 z mikrokomputerem ROSA 7741s.

ROSA /skrót od Rationally Organized System Administrator/ jest mikrokomputerem, zbudowanym z elementów dużej, średniej i małej skali integracji. Elementy LSI należą do rodziny Intel 8080A.

Główne składniki ROSA są następujące:

- blok sterujący wykorzystujący mikroprocesor Intel 8080A,

- pamięć operacyjna typu RAM oraz pamięć stała typu ROM /łącznie 32K bajtów/,

- 16-znakowy wyświetlacz alfanumeryczny,

- pulpit operacyjny zawierający klawiaturę uniwersalną oraz klucze sterujące pracą mikrokomputera,

- kanał współpracy z interfejsem IEC,

- integralnie związana z ROSA pamięć zewnętrzna oparta na pamięci kasetowej PK-1.

Wymienione składniki zapewniają pełne, programowe sterowanie zestawem. Ze względu na wykorzystany mikroprocesor podstawowym językiem programowania jest język symboliczny dla mikroprocesora Intel 8080. Program napisany w tym języku jest tłumaczony na program wynikowy w standardowym formacie szesnastkowym przy pomocy programu assembler, umieszczonego w pamięci stałej. Z urządzeniami wejścia/wyjścia program komunikuje się przy pomocy odpowiednich podprogramów zawartych w programie monitor, zapisanym w pamięci stałej.

Monitor pełni funkcję systemu operacyjnego. Jego podstawowe zadania to:

- inicjacja pracy mikrokomputera po włączeniu zasilania,

- obsługa przerwań,

- organizacja wyświetlania i redakcji tekstów na wskaźniku,

- umożliwienie podglądania i modyfikacji zawartości pamięci operacyjnej oraz rejestrów mikroprocesora,

- sterowanie pracą programów /wczytywanie programu w postaci wynikowej, uruchamianie z ewentualnym ustawianiem pauz czyli miejsc, w których program jest wstrzymywany w oczekiwaniu na polecenia operatora/,

- wspomniana wyżej obsługa wejść i wyjść.

W celu umożliwienia przetwarzania danych /w tym obróbki statycznej wyników pomiarów/ został opracowany pakiet programów arytmetyki zmiennoprzecinkowej. Pakiet ten, zawarty w pamięci stałej realizuje następujące funkcje: normalizację, dodawanie, odejmowanie, mnożenie,dzielenie, porównywanie oraz logarytm dziesiętny. Użytkownik pisząc program w języku symbolicznym, z wykorzystaniem podprogramów dostarczanych mu przez monitor, ma do dyspozycji program edytor. Edytor również znajduje się w pamięci stałej i ułatwia redagowanie i poprawianie programów. Łącznie programy monitor, assembler, pakiet programów arytmetycznych oraz edytor zajmują 12K bajtów pamięci. Użytkownik ma do dyspozycji pamięć operacyjną o objętości 18.5K bajtów. Pierwsze 1K pamięci jest zajęte przez bufory i komórki robocze programów zapisanych w pamięci stałej. Dwie ostatnie strony /strona - 256 bajtów/ przeznaczone są na stosy: roboczy monitora oraz użytkownika.

Biorąc pod uwagę fakt, że programowanie w języku symbolicznym, mimo że daje duże możliwości, może niewprawnemu programiście przysporzyć sporo kłopotów, przewiduje się opracowanie w najbliższej przyszłości języka programowania odpowiednio wysokiego poziomu, łatwego do nauczania się a jednocześnie wykorzystującego wszystkie układowe możliwości ROSA. Będzie to język oparty na języku BASIC, dopuszczający pisanie podprogramów w języku symbolicznym oraz wzbogacony o instrukcje umożliwiające zgrabną i efektywną obsługę interfejsu IEC.

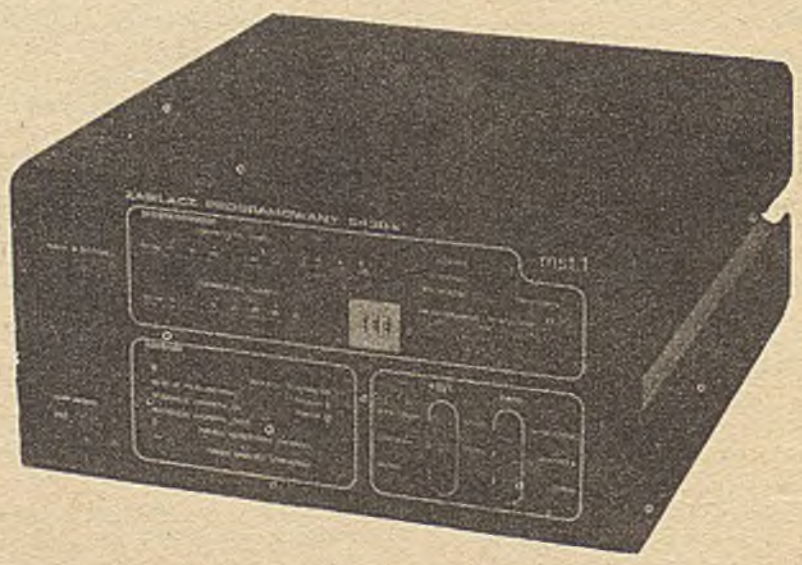
mgr inż. KRYSTYNA KIETLIŃSKA
ZUE "Unitra-Unima"

ZASILACZE PROGRAMOWANE W SYSTEMIE MST-1

Modułowy System Testujący MST-1 oparty jest na bazie interfejsu IEC-625. W oparciu o poszczególne moduły, które stanowią funkcjonalnie pełne bloki, zdolne realizować swe funkcje również poza systemem jako samodzielne przyrządy laboratoryjne, system umożliwia realizowanie praktycznie dowolnych funkcji pomiarowo-kontrolnych. Warunkiem tego, aby system mógł być wykorzystywany w szerokim zakresie zagadnień pomiarowych jest dostatecznie duży wachlarz modułów, zdolnych do współpracy w tym systemie. Jedną z podstawowych grup modułów, które potrzebne są do rozwiązania bardzo wielu zagadnień pomiarowych są źródła wymuszające prądy lub napięcia stałe.

Na terenie Zakładu Urządzeń Elektronicznych UNITRA-UNIMA opracowano rodzinę źródeł programowanych, które przeznaczone są właśnie do pracy w systemie MST-1. Źródła te mają pewne cechy wspólne, zaś różnią się między sobą parametrami elektrycznymi wyjść. Parametry źródeł przedstawiono w tabelach, zaś ich właściwości ogólne omówiono poniżej. Programowane źródła powinny mieć możliwość pracy również samodzielnie, mają więc zapewnioną możliwość programowania ręcznego za pomocą manipulatorów umieszczonych na płycie czołowej przyrządu. Ze względu na wymagania systemowe interfejs może jednak wysterować blok do pracy w systemie i wówczas programatory z płyty czołowej nie mają wpływu na stan bloku.

Ze względu na fakt, że bloki pracować mogą w systemach kontrolno-pomiarowych sterowanych kontrolerem, zaś wartości zaprogramowanych lub pomierzonych parametrów służyć mogą do dalszej obróbki w kontrolerze lub kalkulatorze, wymagane jest programowanie wartości cyfrowych w przyjętym w systemie kodzie zwanym ISO-7. Wynika stąd



Fot. 1. Zasilacz programowany w systemie MST-1.

Parametry zasilaczy

Tabela 1

Rodzaj i typ źródła	Programowane źródło napięciowe 5430s	Programowane źródło napięciowo-prądowe 5432s	Programowane źródło napięciowo-prądowe 5433s	Poczwórne programowane źródło napięcia 5434s	Programowane źródło napięciowo-prądowe 5438s	Programowane źródło prądowe 5439s
1	2	3	4	5	6	7
Zakresy wartości napięcia	0 ±1.998V 0± 19.98V	0±0.999V 0±9.99V	0±9.99V 0±99.9V	4x0±9.99V	0±9.99V 0±99.9V	———
Rozdzielczość	1000	1000	1000	1000	1000	———
Dokładność wymuszenia napięcia	± 0,1% pełnego zakresu	±0,1% pełnego zakresu	±0,1% pełnego zakresu	±0,1% pełnego zakresu	±0,1% pełnego zakresu	———
Maksymalna wydajność prądowa	300mA	300mA		10mA	1A	———
Zakresy ograniczeń prądu	±1,10,100μA ±1,10,100mA	±1,10,100μA ±1,10,100mA	±1,10,100μA ±1,10mA	———	±1,10,100μA ±1,10,100mA ±1A	———
Ograniczenie prądu w ramach zakresu	x1/2, x1/5	x1/2, x1/5	x1/2, x1/5	———	0 ±99 jednostek	———
Dokładność włączenia układu ograniczeń	± 2% zakresu	±2% zakresu	±2% zakresu	———	±1% zakresu	———
Maksymalne przekroczenie prądu granicznego	100%	100%	100%	———	10%	———
Napięcie wyjścia pomiarowego do pomiaru prądu obciążenia	0 ±1V na każdym zakresie	0 ±1V na każdym zakresie	0 ±1V na każdym zakresie	———	0 ±1V na każdym zakresie	0 ±1V na każdym zakresie

c.d. Tabeli 1

1	2	3	4	5	6	7
Zakresy wartości prądu	———	0 ±0.999mA 0 ±9.99mA 0 ±99.9mA	0 +0.999mA 0 +9.99mA	———	0 +9.99μA 0 +99.9μA 0 +999mA 0 +9.99mA 0 +99.9mA 0 +999A	0 ±9.99μA 0 ±99.9μA 0 ±999mA 0 ±9.99mA 0 ±99.9mA 0 ±999A
Rozdzielczość	———	1000	1000	———	1000	1000
Dokładność wymuszenia prądu	———	±0.2% pełnego zakresu	±0.2% pełnego zakresu	———	±0.2% pełnego zakresu	±0.2% pełnego zakresu
Maksymalna podatność napięciowa	———	±20V	+100V	———	+100V	±20V
Zakresy ograniczeń napięcia	———	±5, ±7V	+5, 7, 10, 30, 60, 80V	———	0 +10V 0 +100V	0 ±10V 0 ±20V
Ograniczenia w ramach zakresu	———	———	———	———	0 +99 programowane co 1/100 zakresu	0 +20 programowane co 1V
Dokładność zadziałania układu ograniczenia	———	±2%	±2%	———	±1%	±1%
Maksymalne przekroczenie zaprogramowanej wartości napięcia	———	25%	25%	———	25%	1V
Dokładność pomiaru napięcia na złączu pomiarowym zależy od Rwe woltomierza	$Uwy = Uk\frac{Rwe+10^5}{Rwe}$ Uk - sygnał pomiarowy	$Uwy = Uk\frac{Rwe+10^5}{Rwe}$ Uk - sygnał pomiarowy	$Uwy = Uk\frac{Rwe+10^5}{Rwe}$ Uk - sygnał pomiarowy	———	$Uwy = Uk\frac{Rwe+10^5}{Rwe}$ Uk - sygnał pomiarowy	$Uwy = Uk\frac{Rwe+10^5}{Rwe}$ Uk - sygnał pomiarowy

c. d. Tabeli 1

1	2	3	4	5	6	7
Sygnalizacja rozwarcia zacisków przy sygnale różnicowym "sense-force"	±1V	±1V	±1V	±1V	±1V	±1V
Sygnalizacja wzbudzeń przy sygnale o amplitudzie	0,5V	0.5V	0,5V	———	———	———
Czas trwania impulsu wyjściowego przy pracy impulsowej	———	———	———	———	300 µs	300 µs
Sygnał modulujący wewnętrzny	———	———	———	———	10% amplituda, 1kHz	10% amplituda, 1kHz
Praca z modulacją zewnętrzną	———	———	———	———	sygnał o amplitudzie ⩽ 10V	sygnał o amplitudzie ⩽ 10V
Rezystancja wejścia modulującego	———	———	———	———	50Ω	50Ω

konieczność użycia odpowiednich przetworników cyfrowo-analogowych. Wszystkie źródła systemowe programowane są, jeśli chodzi o wartość wymuszaną, w zakresie trzech lub czterech cyfr, przy czym jeśli występują 4 cyfry, to pierwsza, najstarsza z nich, może przybierać jedynie wartość 0 lub 1. Wszystkie źródła programuje się z rozdzielczością 1000. Ze względu na pracę w automatycznych systemach kontroli typu "go-no go" każde ze źródeł wyposażono w układy umożliwiające zaprogramowanie ograniczenia prądu przy źródle napięciowym zaś napięcia przy źródle prądowym. Układ kontroli pobieranego prądu lub występującego na wyjściu napięcia sygnalizuje stan przekroczenia zadanej wartości z dokładnością 1% zaprogramowanej wartości. Sygnał przekroczenia ograniczenia wyprowadzony jest zarówno na płytę czołową bloku jak i do jego słowa stanu, dzięki czemu można go wykorzystać do segregacji badanych elementów.

Ze względu na współpracę bloków wymuszających z układami komutacyjnymi oraz na dość duże, sięgające często 8-10 metrów połączenia między obiektem badanym a źródłem wymuszającym, wszystkie źródła wyposażone są w pary zacisków wyjściowych prądowe - "force" i kontrolne - "sense". Zaciski te powinny być zwierane bezpośrednio na obiekcie badanym. Rozwiązanie takie pozwala uniknąć błędów wymuszenia, spowodowanych spadkami napięć na przewodach doprowadzających i przekaźnikach komutacyjnych, jednakże aby zabezpieczyć się przed nieprawidłowym włączeniem bloków do układu zastosowano w źródłach układy kontroli zwarcia zacisków"sense"i"force". Sygnalizacja rozwarcia tych zacisków wyprowadzana jest zarówno na płytę czołową bloków jak i do słowa stanu. Umożliwia to również kontrolę poprawności połączeń realizowanego układu pomiarowego w systemie.

Ze względu na pracę w systemie, gdzie szereg połączeń wykonuje się szeregowo, bloki źródeł wyposażono w możliwość programowania blokady, która zapewnia, że niezależnie od stanu zaprogramowania poszczególnych rejestrów bloku wartość wymuszenia wynosi 0 ± LSB najwyższego zakresu. Umożliwia to wcześniejsze zaprogramowanie bloku i wykorzystanie go szybko we właściwym momencie przez dostęp tylko do jednego rejestru blokady. Ze względu na to, iż niektóre przełączenia w źródłach wymuszających powodują wystąpienie na wyjściu niekontrolowanych stanów przejściowych, każdy z zasilaczy programowanych wyposażono w układ blokady analogowej, która działa automatycznie, jeśli zmienia się zakres wymuszanej wartości lub rodzaj pracy źródła. Blokada ta powoduje, że niezależnie od stanu zaprogramowania źródła sygnał wysterowujący je wynosi zero ± LSB najwyższego zakresu. Urządzenie powraca do stanu zaprogramowanego po wyzwoleniu go albo w systemie, albo z płyty czołowej.

W celu umożliwienia w systemie przeprowadzenia samokontroli zestawu wszystkie źródła wyposażono w możliwość zaprogramowania zwarcia wewnętrznego zacisków kelvinowskich. Pozwala to sprawdzić, czy w razie sygnalizacji ich rozwarcia uszkodzenie występuje w komutatorze czy w źródle wymuszającym. W tym celu każde ze źródeł wyposażono w złącze służące do pomiaru wewnątrz źródła prądu jego obciążenia lub napięcia wyjściowego na jego zaciskach. Napięcie mierzone jest pomiędzy punktami zwarcia zacisków"sense"i"force". Rodzaj pomiaru, jakiego należy dokonać, programowany jest systemowo lub lokalnie. Wyposażenie źródeł w złącze pomiarowe pozwala, oprócz dokonania samokontroli bloku, na stworzenie zespołów wymuszająco-pomiarowych przy wykorzystaniu woltomierza systemowego, bez konieczności budowania układów pomiaru prądu. Pomiar prądu obciążenia dokonywany jest w zakresie 1V, a dokładność pomiaru zależy od rezystancji wejściowej woltomierza i od jego dokładności. Przy pracy w systemie z kontrolerem można wprowadzić poprawkę obliczeniową i dokładność pomiaru zwiększyć w ten sposób, eliminując zależność od rezystancji woltomierza. Wszystkie zasilacze programowane, wyposażono w ekrany aktywne, które znajdują się na potencjale wyjść. Przyspiesza to ładowanie linii pomiarowych i zapobiega upływności prądów z przewodów"sense"i"force".

Ze względu na to, że źródło wymuszające może tworzyć wraz z badanym obiektem niekorzystny układ, który będzie się wzbudzał, wyposażono niektóre źródła w układ kontroli wzbudzeń, który wykrywa wzbudzenie na wyjściu źródła, jeśli przekracza ono poziom 0,5V. Kontrola wzbudzeń może być wyprowadzona do słowa stanu i wykorzystana w systemie do podjęcia decyzji dobry-zły. Ze względu na konkretne wymagania niektórych zestawów pomiarowych niektóre z zasilaczy wyposażono w możliwość programowania pracy impulsowej. Źródło takie podaje wymuszenia na wyjście przez czas 300 µs od wyzwolenia, a następnie znajduje się w stanie zablokowanym do następnego wyzwolenia. Niektóre ze źródeł wyposażono ponadto w możliwość pracy z 10-procentowym sygnałem zmiennym /f=1kHz/ nałożonym na wartość zaprogramowaną. Umożliwia to dokonywanie pomiarów np. parametrów macierzy h tranzystorów w zestawach EDT-1. W przypadku konieczności nałożenia na napięcie lub prąd wyjściowy innego sygnału zmiennego można skorzystać ze złącza pozwalającego nałożyć na sygnał wyjściowy dowolnego przebiegu zmiennego z zewnętrznego generatora. Jednocześnie wszystkie źródła są nieuziemione, pracujące na dowolnym potencjale, co umożliwia dołączenie ich do obiektu badanego w praktycznie dowolny sposób i zapewnia możliwość dokonywania różnorodnych pomiarów.

mgr inż. MICHAŁ KLEBANOWSKI
ZUE "Unitra-Unima"

ZESTAW POMIAROWY MST-1/MSI-24 DO BADAŃ UKŁADÓW SCALONYCH TTL

Zestaw pomiarowy MST-1/MSI-24 jest automatycznym testerem przeznaczonym do badań funkcjonalnych i pomiarów stałoprądowych cyfrowych układów scalonych TTL małej i średniej skali integracji. Poszczególne bloki wymuszające i pomiarowe sterowane są programem umieszczonym w pamięci półprzewodnikowej. Program wczytuje się z czytnika taśmy papierowej lub z taśmy magnetycznej. Wczytanie programu do pamięci jest konieczne po włączeniu zasilania i przy zmianie typu badanego mikroukładu. Sterowanie pracą zestawu odbywa się za pośrednictwem interface'u IEC-625/IEEE 488/, który został przyjęty jako standard w całym systemie MST-1.

W skład zestawu pomiarowego wchodzą następujące zasadnicze zespoły konstrukcyjne:

- Stanowisko Badań Funkcjonalnych
  - część centralna typu 7210-s
  - część wykonawcza typu 7230-s
- Stanowisko Badań Stałoprądowych
  - Woltomierz typu 1122
  - Blok decyzyjny typu 7373-s
  - Zasilacz programowany typu 5432-s
  - Pamięć systemowa buforowa typu 7342-s.

Stanowisko Badań Funkcjonalnych jest urządzeniem specjalizowanym. Natomiast aparaty wchodzące w skład Stanowiska Badań Stałoprądowych są systemowymi blokami uniwersalnymi i mogą być wykorzystywane w innych zestawach pomiarowych MST-1 lub jako oddzielne przyrządy laboratoryjne. Zestaw może być dodatkowo wyposażony w drukarkę DZM-180 z interfejsem IEC-625, systemową pamięć kasetową, czytnik taśmy perforowanej i perforator taśmy. Zestaw jest w zasadzie przeznaczony do pracy w zakładach wytwarzających układy scalone. Może być również stosowany w dużych zakładach sprzętowych do kontroli dostaw układów scalonych. Obecnie pięć zestawów MST-1/MSI-24 pracuje w zakładach Tesla-Roznov w Czechosłowacji, a jeden w FP TEWA w Warszawie.

Przebieg testowania

Badany układ scalony zostaje umieszczony w gnieździe pomiarowym części wykonawczej Stanowiska Badań Funkcjonalnych, która jest głowicą pomiarową. Zespół ten zawiera płytki z układami wymuszająco-pomiarowymi, po jednej do obsługi każdego wyprowadzenia mikroukładu. W czasie testowania operator porozumiewa się z zestawem za pośrednictwem płyty czołowej Stanowiska Badań Funkcjonalnych. Test funkcjonalny rozpoczyna się od programowego dołączenia poszczególnych wyprowadzeń układu badanego do układów wymuszających stany wejściowe lub do komparatorów, zależnie od funkcji tych wyprowadzeń /wejście lub wyjście/. Ustawione zostają programowane źródła referencji, które decydują o wielkościach napięć wymuszanych na wejściach jako zera lub jedynki logiczne i o poziomach rozróżniania stanów logicznych przez komparatory wyjściowe. Zgodnie z programem do badanego układu zostaje dołączone zasilanie i ustawiona wartość napięcia zasilającego.

Po tych czynnościach wstępnych rozpoczyna się właściwe testowanie funkcjonalne. W każdym z kolejnych kroków testu zostaje z pamięci przysłany pełny wiersz "tabeli prawdy" zawierający stany logiczne podawane na wejścia i oczekiwane na wyjściach badanego układu.

Dla wszystkich wyjść komparatory podają wyniki kroku testu:

$U_{WY} < U_{KL}$, $U_{WY} > U_{KH}$ - błąd poziomu

$U_{KL} \leqslant U_{WY} \leqslant U_{KH}$ - poziom poprawny

$U_{WY} < U_{KP}$ - stan L

$U_{WY} \geqslant U_{KP}$ - stan H,

gdzie:

U_{WY} - napięcie na wyjściu układu badanego,

U_{KL} - napięcie referencji komparatora dla stanu niskiego,

U_{KH} - napięcie referencji komparatora dla stanu wysokiego,

U_{KP} - napięcie progowe /ustawiane ręcznie/.

Błędy poziomu sygnalizowane przez komparatory lub różnice między stanami na wyjściach a stanami oczekiwanymi rejestrowane są w pamięci błędów. Zawartość pamięci błędów sprawdzana jest na końcu testu i jest kryterium zakwalifikowania badanego układu do grupy dobrych lub złych funkcjonalnie.

Test stałoprądowy jest zwykle wykonywany po zakończeniu testu funkcjonalnego. Polega on przeważnie na pomiarze prądów wejściowych przy wymuszeniu napięciowym na wejściu oraz napięć wyjściowych przy wymuszeniu prądowym na wyjściu. Mierzy się także prądy zwarciowe na wyjściach i prąd zasilający. W każdym kroku testu stałoprądowego konfigurację układu pomiarowego tworzy się za pomocą sterowanej programowo matrycy komutacyjnej z przekaźników kontaktronowych, dołączającej poszczególne przyrządy pomiarowe i wymuszające do punktu pomiarowego.

Po zaprogramowaniu zasilacza /wartość i rodzaj wymuszenia/, woltomierza /zakres/ i bloku decyzyjnego /dopuszczalne granice dla wyniku/ na polecenie programu następuje pomiar. Wynik pomiaru /poprawny-niepoprawny/ przekazywany jest do Stanowiska Badań Funkcjonalnych, gdzie jest rejestrowany w pamięci błędów i wpływa na końcowy wynik testu. W teście stałoprądowym korzysta się w niektórych krokach z układów przeznaczonych zasadniczo do testowania funkcjonalnego. Za pomocą tych układów ustawia się programowo poziomy logiczne na wejściach nie mierzonych, zgodnie z wymaganiami układu pomiarowego. Wyniki testu pokazywane są na płycie czołowej oraz wyprowadzane w postaci sygnałów o standardzie TTL na gniazdo służące do podłączenia automatycznego sortownika i podajnika albo komory klimatycznej. Po podłączeniu odpowiednich urządzeń zewnętrznych możliwa jest rejestracja wyników na taśmie perforowanej, taśmie magnetycznej lub w postaci wydruków.

Dane techniczne

Liczba wyprowadzeń badanych układów:
14, 16, 24

Szybkość testowania:

- czas trwania jednego kroku testu:
  - funkcjonalnego 350 ns
  - stałoprądowego 15 ms
- czas trwania pełnego testu np. mikroukładu 74193
  - bez rejestracji wyników 1,4s
  - z rejestracją na taśmie magnetycznej 5,5s

Maksymalna liczba kroków testu funkcjonalnego - nieograniczona, zliczana modulo 1000.

Programowanie poziomów logicznych na wejściach:

- zakres	0 ÷ 5,5V
- skok	10mV
- dokładność	±20mV bez obciążenia
- szybkość zmian napięcia	40ns/V

Programowanie komparatorów wyjściowych:

- zakres	0 ÷ 5V
- skok	10mV
- dokładność	± 10mV
- napięcie progowe	ok. 1,4V /ustawiane ręcznie/

Programowanie obciążeń wyjść:

- w stanie niskim	-0,1mA ÷ -70mA co 0,1mA
- w stanie wysokim	+0,01mA ÷ +8mA co 0,01mA
- dokładność /w stanie niskim i wysokim/	± 3% ±1LSB

Przyporządkowanie wyprowadzeniom funkcji wejść i wyjść: programowane

Przyporządkowanie funkcji zasilania i masy: przez wymianę wkładek

Wymuszenia stałoprądowe: programowane

- zakres napięć -20 ÷ +20V, Imaks = 300mA
  -20 ÷ +100V, Imaks = 10mA
- zakres prądów -100 ÷ +100mA
- rozdzielczość 1000
- dokładność ±0,1% ±1LSB

Dokładność pomiarów stałoprądowych 1% ±0,1% zakresu

Napięcie zasilające układ badany - programowane

- zakres -20 ÷ +20V
- rozdzielczość 1000
- dokładność ±0,5% ±1LSB
- wydajność prądowa Imaks = 300mA

Rodzaje pracy zestawu:

- test pełny,
- test do pierwszego błędu,
- krok po kroku,
- stop na błędzie.

mgr inż. JACEK KIETLIŃSKI
ZUE "Unitra-Unima"

ZESTAW DO TESTOWANIA DIOD I TRANZYSTORÓW MST-1/ EDT-1

W zakładach UNITRA-UNIMA znajduje się w fazie opracowania zestaw do testowania diod i tranzystorów małej i średniej mocy. Zestaw ten został oznaczony symbolem MST-1/EDT-1 i stanowi kontynuację zasady organizacji stanowisk pomiarowo-kontrolnych na bazie zestawiania ich z bloków funkcjonalnie pełnych, tzn. bloków stanowiących samodzielne przyrządy, połączonych po stronie sterowania wspólną szynę interfejsu. Metoda ta podyktowana jest dążeniem do skrócenia czasu opracowywania dużych urządzeń i przedłużenia serii produkcyjnych poszczególnych bloków. Wynika to z możliwości zastosowania takich samych bloków w zestawach o różnym przeznaczeniu. Jednocześnie część bloków może być sprzedawana jako samodzielne przyrządy.

Konfiguracja zestawu

Zestaw ma umożliwić dokonywania pomiaru parametrów statycznych i niektórych parametrów m. cz. diod i tranzystorów oraz segregację ich na grupy za pomocą sortowników opracowanych w PIE-ZDUT. Opierając się na anlizie podstawowych układów pomiarowych uzyskano konfigurację toru pomiarowego zawierającą następujące bloki:

- programowane źródło napięcia i prądu o maksymalnym napięciu 500V i prądzie 10mA,
- programowane źródło napięcia i prądu o maksymalnym napięciu 100V i prądzie 1A,
- programowane źródło prądu o prądzie maksymalnym 1A i podatności napięciowej 20 V,
- woltomierz napięcia stałego,
- adapter.

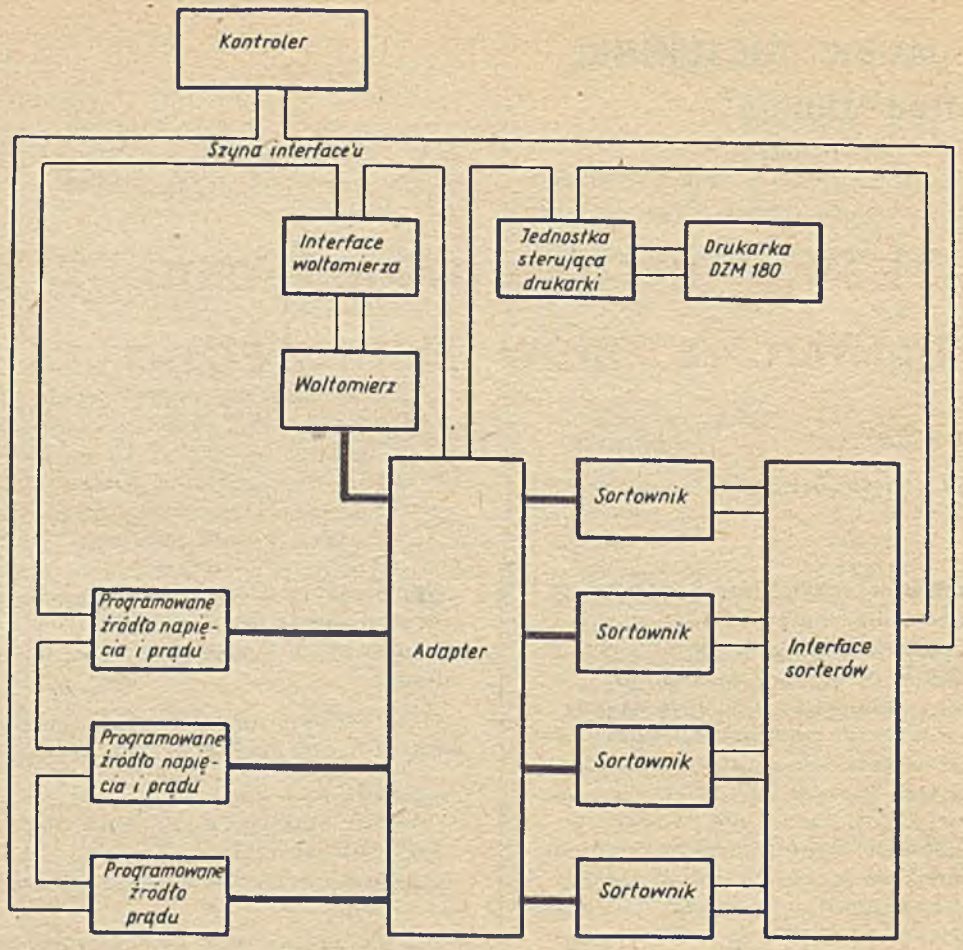
Jak widać z przedstawionego zestawienia część pomiarową stanowią cztery bloki uniwersalne i jeden blok specjalistyczny, tzn. adapter. Blok ten pozwala dostosować stanowisko do stawianych mu wymagań. Jest on wyposażony w zespoły komutujące, przetwornik prąd/napięcie, przetwornik AC/DC, układy próbkujące z pamięcią i inne układy pomocnicze.

Zastosowane w zestawie programowane zasilacze mogą pracować jako źródła:

- sygnału stałego,
- sygnału stałego ze składową zmienną o częstotliwości 1kHz,
- sygnału impulsowego o szerokości impulsu 300 us.

Dzięki takim możliwościom bloków w zestawie będzie można pomierzyć wszystkie parametry o charakterze napięciowym i prądowym diod i tranzystorów w zakresie małych i średnich mocy oraz dokonywać pomiarów typu rezystancja dynamiczna złącza, macierz /he/ itp. Wszystkie bloki niezbędne do uformowania toru pomiarowego połączone są wspólną szyną sterującą zorganizowaną w oparciu o zalecenia zawarte w IEEE-488 std. 1975/IEC/TC-66/. Jednostkę sterującą zestawu stanowić będzie kontroler ROSA opracowany w zakładzie UNITRA-UNIMA. Jednostką centralną kontrolera jest procesor zbudowany na bazie mikroprocesora Intel 8080A wyposażony w klawiaturę i pamięć kasetową. Zestaw będzie mógł współpracować z drukarką wierszową DZM 180 i czytnikiem taśmy perforowanej CT 2200. Wprowadzenie kontrolera do zestawu pozwoli na uproszczenie części pomiarowej. Wynika to z możliwości dokonywania pomiarów złożonych parametrów metodą pośrednią, czyli metodą obliczeń na bazie danych uzyskanych z pomiarów prostych. Poza tym kontroler umożliwi sowtwareowe programowanie, a więc zagwarantuje pełną dowolność formułowania sekwencji testowych. Dodatkowo umożliwi automatyczne opracowywanie raportów dziennych oraz analizę statyczną uzysku i braków dla dowolnie wybranych grup parametrów. Strukturę zestawu ilustruje rys. 1.

Konstrukcja adaptera pozwoli na dołączenie do zestawu torów pomiarowych z czte-



Rys. 1. Struktura zestawu MST-1/EDT-1

rech sortowników równocześnie. Praca sortowników sterowana będzie bezpośrednio przez kontroler poprzez blok interfejsu.

Praca zestawu

Przystępując do uruchomienia zestawu należy wprowadzić do pamięci kontrolera program. Można będzie tego dokonać przez wczytanie go z pamięci kasetowej lub z taśmy perforowanej poprzez dołączony czytnik. Teoretycznie istnieje również możliwość wprowadzania programu za pomocą klawiatury lecz metoda ta jest bardzo czasochłonna. Użycie klawiatury przewidziane jest raczej przy wprowadzaniu korekt do istniejących programów i opracowaniu nowych. Badane elementy wprowadzane będą na szczęki pomiarowe sortownika za pomocą podajnika wibracyjnego. Sortownik po zwarciu szczęk pomiarowych będzie sygnalizował poprzez słowo "stan" swoją gotowość.

W przypadku pracy z więcej niż jednym sortownikiem, kolejność ich obsługi będzie mogła być określana wg narzuconych programowo priorytetów lub w kolejności zgłaszanej gotowości. Zestaw będzie mógł pracować maksymalnie z czterema sortownikami równocześnie, przy czym każdy z sortowników będzie sortował inny typ elementów. Maksymalna ilość grup segregacyjnych będzie wynosić 49. Parametr ten jest określony konstrukcją sortowników. Czas badania jednego elementu uzależniony jest przede wszystkim od ilości kontrolowanych parametrów. Czas trwania pomiaru jednego parametru będzie się zawierał w granicach od 15 do 100ms. Przedłużanie czasu badania jest konieczne przy wymuszeniach niskoprądowych i przy pomiarach małych prądów, co wynika z konieczności ładowania pojemności toru pomiarowego. Aby zapobiec wzrostowi temperatury złącza w trakcie wykonywania pomiaru użyte w zestawie źródła umożliwiają podawanie na obiekt badania wymuszenia w postaci impulsu o szerokości nie przekraczającej 300 us. Możliwość takiej pracy jest przewidywana przy pomiarach, w których prądy wymuszeń będą przekraczały 10mA. Wszystkie źródła będą umożliwiały programowanie ograniczenia napięcia przy wymuszeniu prądowym i prądu przy wymuszeniu napięciowym; zapewni to zabezpieczenie elementu przed zniszczeniem w trakcie badania.

W bloku adaptera znajduje się układ wykrywania oscylacji, który poprzez słowo stanu

będzie informował kontroler o wystąpieniu wzbudzenia na badanym obiekcie. Jest to szczególnie ważne przy kontroli elementów półprzewodnikowych w zakresie przebicia.

Dane techniczne zestawu MST-1/EDT-1.

W zestawie zagwarantowana będzie możliwość sprawdzania następujących parametrów:

- dla diod:
  - U_F - napięcie przewodzenia
  - U_Z - napięcie stabilizacji
  - U_R - napięcie wsteczne
  - I_R - prąd wsteczny
  - r_Z - rezystancja dynamiczna w zakresie napięcia stabilizacji
  - r_{ZK} - rezystancja dynamiczna w kolanku charakterystyki napięcia stabilizacji
  - r_f - rezystancja dynamiczna charakterystyki przewodzenia - dla tranzystorów bipolarnych
  - I_{CBO}, I_{CEO} - prądy zerowe kolektora
  - I_{EBO} - prąd zerowy emitera
  - $I_{CE/R,S,X/}$ - prądy resztkowe kolektora
  - U/BR/CE/O, R, S, X/ - napięcia przebicia kolektor-emiter
  - $U_{/BR/CBO}$ - napięcia przebicia kolektor-baza
  - $U_{/BR/EBO}$ - napięcie przebicia emiter-baza
  - U_{BE} - napięcie emiter-baza
  - h_{21E} - statyczny współczynnik wzmocnienia prądowego
  - $U_{CE\ sat}$ - napięcie nasycenia kolektor-emiter
  - $U_{BE\ sat}$ - napięcie nasycenia baza-emiter
  - he - parametry macierzy h przy częstotliwości 1kHz
- dla tranzystorów unipolarnych:
  - I_{GSS} - prąd upływu bramki
  - I_{DSS} - prąd drenu
  - U_{GS} - napięcie stałe bramka-źródło
  - U_{GSoff} - napięcie odcięcia bramka-źródło
  - $U_{/BR/GSS}$ - napięcie przebicia bramka-źródło
  - y_{21S} - zwarciowa admitancji przenoszenia w przód przy częstotliwości 1kHz.

Wynik sprawdzenia - zaklasyfikowanie do jednej z grup:

Maksymalna ilość grup klasyfikacyjnych - 49

Czas trwania jednego pomiaru - 15 ÷ 100ms

Czas sprawdzania jednego elementu - uzależniony od ilości badanych parametrów

Poza tym zestaw zapewni możliwość:

- kontroli kontaktów
- wykrywania oscylacji
- wykrywania awarii zestawu.

Maksymalne parametry źródeł wymuszeń:

- napięciowe - 500V z wydajnością prądową 10mA
  - 100 V z wydajnością prądową 1A
- prądowe - 1A z podatnością napięciową 100V
  - 10mA z podatnością napięciową 500V.

mgr inż. JACEK GRONOWSKI
ZUE "Unitra-Unima"

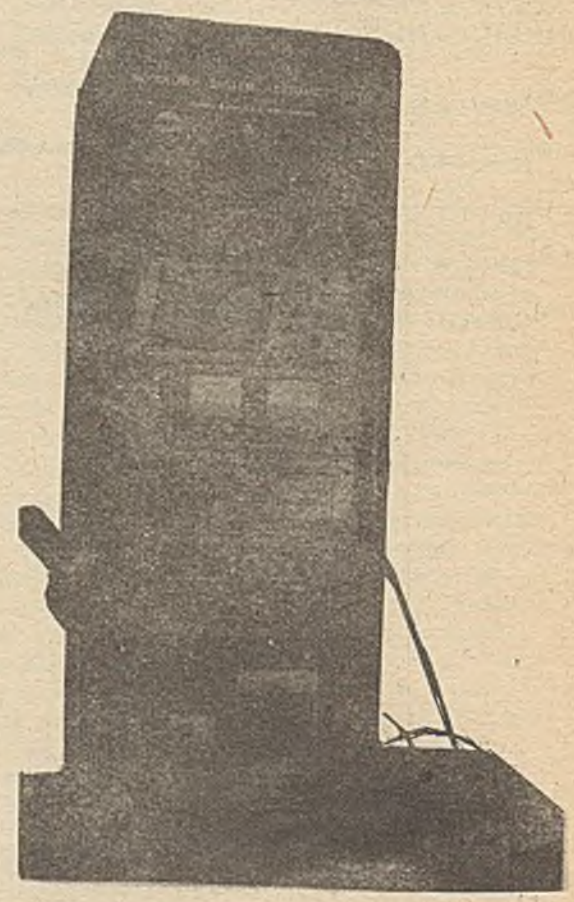
MODUŁOWY SYSTEM TESTUJĄCY OTVC

System MST OTVC jest rozwinięciem koncepcji modułowego systemu testującego MST-1 przeznaczonym do strojenia, naprawy i kontroli modułów i bloków odbiorników telewizyjnych. System ten stanowi rodzinę stanowisk technologicznych, wykorzystujących zunifikowane bloki pomiarowe zestawiane według potrzeb pomiarowych danego modułu lub bloku OTVC. Stanowiska przystosowane są do współpracy z systemem centralnej sieci sygnałowej, rozprowadzającej sygnały wobulowane i obrazy kontrolne. Ujednolicone konstrukcyjnie bloki pomiarowe zestawiane są w szafach 19", tworząc część pomiarową stanowiska. Drugą część stanowi stolik operatora zawierający blok adaptera. Blok adaptera zawiera układy szybkiego mocowania badanego wyrobu i komutacji sygnałów pomiarowych.

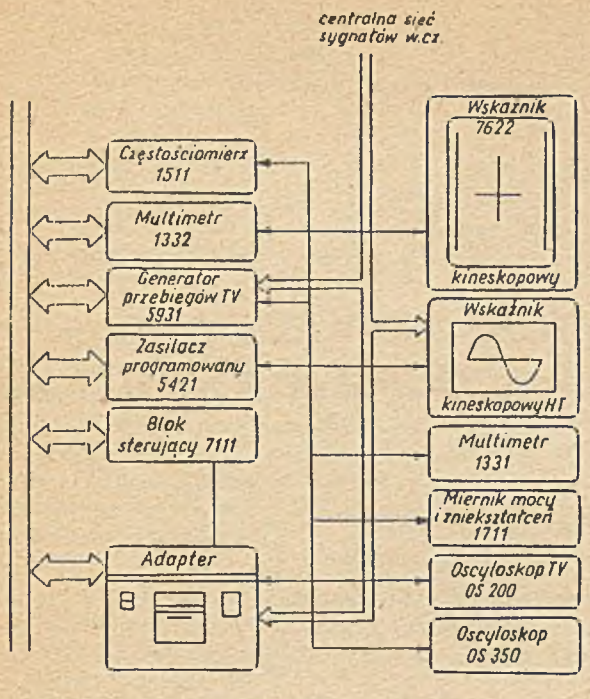
Dzięki zastosowaniu systemu interfejsu wg zaleceń IEC uzyskano możliwość sterowania nastawami przyrządów /bloków systemowych/ według ustalonego programu i w ten sposób zapewnienie dyscypliny technologicznej w zakresie kolejności postępowania. Przygotowany dla danego bloku lub modułu program pracy stanowiska zapisany jest w pamięci bloku sterującego. Program ten zapisany jest w półprzewodnikowej pamięci typu EPROM umieszczonej w wymiennej wkładce.

Na rys. 1 przedstawiona jest struktura MST OTVC. W zależności od wyposażenia w aparaturę pomiarową i informatyczną pomiary i kwalifikacja wyrobów dokonywane są przez operatora stanowiska bądź po porównaniu z zadanymi w programie tolerancjami, automatyczna decyzja DOBRY-ZŁY. W razie potrzeby wyniki pomiarów mogą być rejestrowane w celu późniejszej obróbki statycznej lub utrwalone w postaci protokółu.

Na podstawowy zestaw bloków składają się: multimetr, częstościomierz, miernik



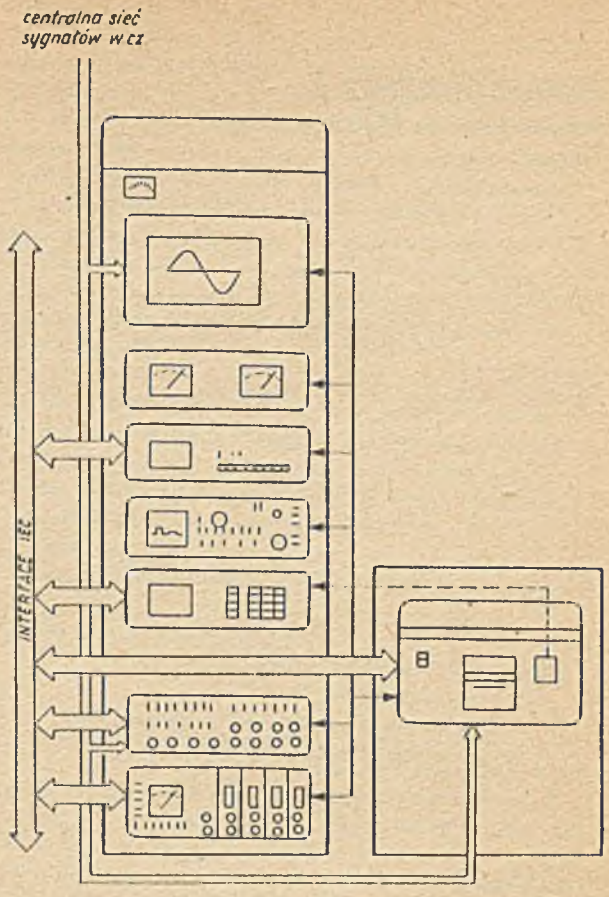
Fot. 1. Modułowy system testujący OTVC



Rys. 1. Struktura modułowego systemu testującego OTVC

mocy i zniekształceń, oscyloskop, wskaźnik kineskopowy, generator przebiegów TV, zasilacz programowany, blok sterujący oraz bloki adapterów. Schemat typowego stanowiska przedstawiony jest na rysunku 2. Do budowy bardziej zautomatyzowanych stanowisk wykorzystuje się dodatkowo bloki systemowe z zestawu MST-1 takie jak: pamięć kasetową, klawiaturę systemową, drukarkę systemową i kontroler systemowy.

Dla polskiego odbiornika telewizji kolorowej "Jowisz" opracowano 12 typów stanowisk zgodnie z jego podziałem funkcjonalnym, tj. dla modułów wzmacniacza p. cz. wizji, fonii,



Rys. 2. Schemat typowego stanowiska MST OTVC

luminancji, chrominancji i identyfikacji, matrycowania i dyskryminatorów, generatora ramki, synchronizacji i stabilizacji oraz dla bloków sygnałowego, regulacji, odchylania i zasilania ilustruje stanowisko przeznaczone do strojenia i kontroli modułu fonii/fot 1/.

TOMASZ MÜLDNER
Instytut Maszyn Matematycznych
"Mera"

PEWNE UWAGI O NOWYCH JĘZYKACH PROGRAMOWANIA WYSOKIEGO POZIOMU: LOGLAN I ADA

(Część II)

Omówienie konstrukcji językowych

TYPY, DEKLARACJE

Typ określa zbiór wartości i operacji wykonywalnych na tych wartościach.

Stałe i zmienne.

LOGLAN:

Ogólna postać: const N=W /dla stałych/, var N:T /dla zmiennych/, gdzie N jest ciągiem nazw oddzielonych przecinkami, W jest wyrażeniem typu pierwotnego zbudowanym z stałych, liczb oraz operatorów i funkcji systemowych, T jest nazwą typu. Wartość początkowa i typ stałej są wyznaczone przez wartość i typ wyrażenia W. Zmienna ma standardowo ustaloną wartość początkową, zależnie od jej typu. Wartość stałej nie może być zmieniona, wartość zmiennej może ulec zmianie wskutek wykonania np. instrukcji podstawienia.

Przykłady:

```
const a=sqrt(4),
var eps1:real, more: boolean.
```

Typ zmiennych może być typem tablicowym lub klasowym /m. in. typem współprogramu lub procesu/. Deklaracja zmiennej typu tablicowego /który to typ nie może być deklarowany explicite/ma postaci/.

```
var N : array_of T
```

gdzie N jest listą nazw, T jest nazwą typu.

Tak więc, po zadeklarowaniu, ustalony jest typ elementów tablicy oraz wymiar tablicy /typ T może być typem tablicowym postaci array_of T1/; natomiast pary graniczne są nieustalone. Ustala je instrukcja generatora tablicy, przyporządkowująca zmiennej konkretny obiekt tablicowy.

Przykłady:

```
var WEKTOR: array_of real;
var TABLICA : array_of array_of real;
```

Tablice traktuje się jako szczególny przypadek obiektów; indeksowanie jako rodzaj kropkowanego dostępu. Zmienna typu tablicowego może w "czasie swego istnienia" wskazywać na tablice z różnymi parami granicznymi.

ADA:

Ogólna postać: N:constant:=W /dla stałych liczbowych/, N:constant T:=W /dla stałych dowolnych/, N:T:=W1 /dla zmiennych/, gdzie N, T, W mają znaczenie jak wyżej, W1 jest wyrażeniem. Wartość początkowa zmiennej jest wyznaczona przez wartość wyrażenia W1.

Przykłady:

```
eps 1, eps 2 : integer := 1;
prawda : constant boolean := true.
```

Deklaracje podprogramów (funkcji i procedur).

LOGLAN:

Ogólna postać

Dla procedury:

```
type virtual P : A procedure(PF); PI fin
```

Dla funkcji:

```
type virtual F : A function (PF) : PT; PI fin
```

gdzie P, F są nazwami, A jest prefiksem, PF jest listą parametrów formalnych, PI jest ciągiem instrukcji, PT jest nazwą typu. Użycie symbolu virtual, prefiksu A i listy PF jest opcyjne. Na liście parametrów formalnych podaje się również sposób ich transmisji /por. wywołanie podprogramu/.

Parametrem może być typ lub podprogram, w ostatnim przypadku należy również podać uproszczoną listę jego parametrów formalnych /jeśli jeden z ostatnio wymienionych parametrów jest znowu podprogramem to podaje się tylko jego nazwę/. Znaczenie symbolu virtual i prefiksu będzie wyjaśnione dalej. Treścią podprogramu jest ciąg instrukcji.

Przykłady:

```
type F: function (input n : integer) : integer;
        if n > = 1 then if n = 1 then result :=1
                                 else
                                 result :=n*F(n-1)
                        fi
        fi fin;
```

```
type B : procedure (function f(x : real) : real;
         output zero :real);
   <treść procedury obliczającej miejsce ze-
   rowe funkcji f>
      fin;
type SORT : procedure (type T; A : array_of T;
            function less (x,y:T) :boolean:)
   <treść procedury sortującej wektor A,
   którego składowe są typu T>
      fin,
```

ADA:

- Deklaracja nagłówka:
  procedure P (PF)
  function P (PF) return T
  gdzie P, PF, T jak powyżej.
- Deklaracja treści /może być oddzielona od deklaracji nagłówka/.

W deklaracji treści powtarza się deklarację nagłówka

S is D; begin I; exception EH end

gdzie S jest deklaracją nagłówka, D jest ciągiem deklaracji, I jest ciągiem instrukcji, EH obsługuje wyjątki. Użycie ciągu D, oraz exception EH jest opcyjne.

Przykłady

```
procedure GET_KEY (K : out KEY ) is
      begin
        LAST_KEY := LAST_KEY +1;
        K := LAST_KEY;
      end GET_KEY;
function less (X, Y : KEY) return boolean is
      begin
        return integer (X) <integer (Y);
        - - komentarz, powyżej występuje
            zmiana kwalifikacji zmiennej z
            typu KEY do typu integer;
        end less;
```

Deklaracja typów

LOGLAN:

Klasy.

Ogólna postać

type K : KO class (PF); T; P; D; I fin

gdzie K jest nazwą klasy, KO jest prefiksem, PF - listą parametrów formalnych, T - listą atrybutów dziedziczonych z prefiksu D - listą deklaracji, tzw. wielkości lokalnych, P - listą protekcji, I - listą instrukcji. Użycie prefiksu KO, list PF, T, P, D jest opcyjne. Atrybutami treści klasy K są parametry formalne oraz wielkości lokalne tej klasy. Prefiks jest nazwą klasy.

Klasa K prefiksowana przez klasę KO jest nazywana podklasą klasy KO; jej atrybuty to zarówno atrybuty treści klasy K jak i klasy KO, a lista instrukcji powstaje z połączenia w odpowiedni sposób /instrukcja sterowania inner/ list instrukcji tych klas. Przez ciąg prefiksowy klasy K rozumiemy ciąg K1,..., Kn klas, taki, że K1 = K, klasa Ki-1 jest prefiksowana przez Ki, i=2,...,n; klasa Kn nie ma prefiksu.

Ponieważ w podklasie klasy można zdefiniować dodatkowe własności /atrybuty, operacje/ danej klasy, więc prefiksowanie umożliwia m. in. "wyciągnięcie przed nawias" wspólnych własności i uszczegółowienie na niższych poziomach, /tzn. w podklasach/.

Stosowanie wirtualnych funkcji i procedur umożliwia używanie ich w treści klasy pomimo iż nie są one w tej klasie zdefiniowane /tzn. nie są zdefiniowane ich treści/. Definicję takich podprogramów można podać na odpowiednim poziomie, tzn, w odpowiedniej podklasie danej klasy. Listy protekcji ograniczają dostęp do atrybutów klasy, /patrz widoczność nazw/.

W języku SIMULA 67, /ze względu na trudności implementacyjne/ prefiksowanie ograniczono do jednego poziomu /tekstowego/, natomiast w języku LOGLAN dopuszcza się prefiksowanie wielopoziomowe.

Przykłady.

```
type node : class (value : real), var left,
            right:node;fin
type BST  : class:
            type Member : function (x:real):
                          boolean;
            .............
            type Insert : procedure (x:real);
            .............
            fin:
```

Współprogramy i procesy.

Deklaracja jak powyżej, z zamianą słowa class na coroutine lub process.

ADA:

Typy wyliczeniowe.

Ogólna postać:

type W is (N)

gdzie W jest nazwą, N jest listą nazw, lub znaków.

Przykład:

type DAY is (MON, TUE, WED, THU);

Ograniczony typ integer

Ogólna postać:

type T is range L..R

gdzie L, R są wyrażeniami typu integer, obiekty tego typu to liczby całkowite od L do R.

Przykład:

type T is range 1..20

Typy rzeczywiste

Ogólna postać: a/ Zmienno-pozycyjne

type T is digits P

lub

type T is digits P range L..R

b/ Stało-pozycyjne.

type T is delta D

lub

type T is delta D range L..R.

gdzie D, P są wyrażeniami, których wartość jest określona w czasie kompilacji i jest dodatnia /wyrażenie P jest typu integer, wyrażenie D jest typu real/. Wyrażenia R, L są typu integer. Specyfikacja range L... R, powoduje ograniczenie zbioru wartości do przedziału /L, R/.

Wyrażenie P dostarcza liczby cyfr dziesiętnych, wyrażenie D dostarcza ograniczenie błędu dla liczb stało-pozycyjnych.

Przykłady:

```
type COEFFICIENT is digits 10
range - 1.0..1.0;
type VOLT is delta 0.125 range 0.0..255.0;
```

Typy tablicowe.
Ogólna postać
a/ Określone pary graniczne

```
type T is array (L1..U1,...,Ln:Un)of TO;
```

b/ Nieokreślone pary graniczne

```
type T is array
(T1 range<>,...., Tn range<>) of TO
```

gdzie T, TO,..., Tn są nazwami typów, L1,..., Un są wyrażeniami typu integer. Symbol < > /box/ jest użyty dla podkreślenia tego, że zbiór wartości jest nieokreślony. W czasie kompilacji nie musi być ustalona wartość wyrażeń L1,..., Un, tzn. par granicznych.
Przykłady

```
type TABLE is array(1..10) of integer;
type MATRIX is array(integer range <>,
                     integer range<>)of integer.
```

Stringi
Systemowo zdefiniowany typ wyliczeniowy elementów typu character.

```
type string is array(natural range <>) of
     character;
     stars:constant:string := "ADA"
```

Rekordy.
Ogólna postać:
a/ bez wariantów

```
type R is record D; end record;
```

b/ z wariantami

```
type R(E) is record D;
  case U of
  when A1 => D1;
  ...
  when An => Dn;
  when others  => DO
  end case;
end record;
```

gdzie R jest nazwą rekordu; E, DO, D1,..., Dn, D są ciągami deklaracji danych /stałych lub zmiennych/, U jest tzw. deskryminantem, tj. stałą z listy E, która w tym rekordzie nie ma nadanej wartości /ewentualnie opcyjną/. Użycie specyfikacji others jest opcyjne.
Dla rekordu z wariantami, jego składowymi są dane zadeklarowane w D, w Di_o jeśli wartością deskryminanta U jest Ai_o, oraz w DO w pp. Deskryminant może wystąpić w rekordzie bez wariantów.
Przykłady:

```
type DATE is record DAY : integer range
                                  1..31;
                    YEAR : integer range
                                  0...2000 :=1979
          end record;
type BUFFER (LENGHT : integer) is record
     -- LENGHT jest deskryminantem;
     ......
     end record;
type DEVICE is (PRINTER, DRUM, DISK);
type PERIPHERAL (UNIT:DEVICE) is record
     case UNIT of
        when PRINTER =>
             LINE_COUNT : integer range
                  1..PAGE_SIZE;
        when others =>
             CYLINDER : CYLINDER_INDEX;
     end case;
end record
```

Typy dostępu
Ogólna postać: type T is access T1
gdzie T1 nie jest typem dostępu.
Przykłady:

```
type T is access BUFFER;
type LIST_ITEM is access
        record VALUE : integer;
               SUCC: LIST_ITEM;
        end record;
HEAD : LIST_ITEM :=
     new LIST_ITEM (5, Y).
```

Typy wyprowadzone i podtypy
Ogólna postać:

```
type T is new TO
type T is new TO range L..U
subtype T is TO range L..U
```

gdzie TO jest nazwą typu; L, U są wyrażeniami typu integer. Typ wyprowadzony T ma dokładnie takie same własności jak typ TO. Dwa typy są zgodne tylko wtedy gdy oba są podtypami jednego typu, lub mają tą samą nazwę, lub jeden jest wyprowadzony z drugiego.
Przykłady:

```
type T is new OLD_TYPE;
type T is new OLD_TYPE range 1..20;
subtype SMALL_INT is integer range -10..
                                    ..10;
```

Powróćmy jeszcze do deklaracji zmiennych typu tablicowego i rekord.
Jeśli dana jest typu tablicowego z nieustalonymi parami granicznymi to deklaracja musi je ustalić - albo explicite albo przez podanie wartości początkowej - agregatu. Jeśli dana jest typu rekordu z wariantem - to stała, która jest deskryminantem może mieć ustaloną wartość tylko przez podstawienie danych typu rekord.
Przykłady:
Typ tablicowy:

```
type MATRIX is array (integer range <>, integer range<>)of real;
```

i zmienna tego typu.

```
SQUARE : MATRIX (1..N, 1..N);
```

Powróćmy do przykładów typów rekordu.
Stała LENGTH typu BUFFER jest deskryminantem; jej wartość nie jest ustalona. Natomiast przy deklaracji zmiennej typu BUFFER, wartość deskryminantu musi zostać ustalona:

```
CARD : BUFFER (LENGTH => 80).
```

Podobnie, nie jest ustalona wartość stałej UNIT rekordu z wariantami PERIPHERAL; zostaje ona ustalona przy deklaracji zmiennej typu PERIPHERAL:

```
WRITER : PERIPHERAL (PRINTER).
```

Wartościami zmiennej X typu access są referencje do dynamicznie tworzonych obiektów; wartością X. all jest ten obiekt. Tak więc X := Y oznacza podstawienie referencji; X. all := Y. all oznacza podstawienie wartości.
Przykłady:
Dla typu LIST_ITEM /z punktu poświęconego

typom dostępu / i zmiennej HEAD, HEAD.all jest rekordem z ze składowymi VALUE=5, SUCC=Y.

Deklaracja pakietów i jednostek generycznych.
Pakiety.
Pakiet składa się z dwu części - specyfikacji i treści. Obie części mogą być deklarowane oddzielnie. Pakiet definiuje strukturę danych, podprogramów i typów.

Nagłówek pakietu ma postać

package P is D; private D1 end

gdzie D, D1 są deklaracjami. Część private D1 jest opcyjna. Elaboracja nagłówka powoduje rezerwację pamięci dla odpowiednich nazw. Nazwy z listy D są dostępne dla otoczenia modułu, natomiast nazwy z listy D1 służą implementacji typów wymienionych na liście D, i są niewidoczne dla użytkownika.
Przykłady:

```
package WORK_DATA is
   type DAY is (MON, TUE, WED, THU, FRI)
   end WORK_DATA;
package RATIONAL_NUMBERS is
   type RATIONAL is record
                      NUMERATOR: integer;
                      DENOMINATOR: integer
                       end record;
   function plus (X, Y : RATIONAL) return
                  RATIONAL;

                       end RATIONAL
      end RATIONAL_NUMBERS;
```

Treść pakietu jest postaci

Package body P is D ; begin I end P;

gdzie D jest listą deklaracji; I listą instrukcji. Przy elaboracji treści pakietu wykonuje się listę instrukcji I inicjalizujących wartości atrybutów.
Nazwy zadeklarowane na liście D są niedostępne w otoczeniu pakietu.
Przykłady:

```
   package body RATIONAL_NUMBERS is
      function plus (X, Y : RATIONAL) return
                     RATIONAL is
                    <treść funkcji>
      end;
```

Jednostki generyczne.

Parametrami formalnymi podprogramów nie mogą być podprogramy ani typy. Istnieje natomiast możliwość deklarowania generycznych podprogramów i pakietów, przez poprzedzenie ich deklaracji klauzulą postaci

generic GP

gdzie GP jest /opcyjną/ listą generycznych parametrów formalnych, zmiennych lub nazw typów, lub podprogramów.
Przykłady:

```
    generic type ELEM
    procedure EXCHANGE (U, V : in out ELEM)
                                    is
                T : ELEM;
                begin T := U; U := V; V := T
                  end EXCHANGE;
    generic SIZE : integer ;  type ELEM
    package STACK is
        procedure PUSH (E : in ELEM);
        procedure POP  (E : out ELEM);
    end STACK;
```

Generyczne typy formalne są postaci

type T is (< >) - dowolny typ dyskretny,
type T is range < > dowolny typ integer
type T is digits < > dowolny typ zmiennopozycyjny,
type T is delta < > dowolny typ stało-pozycyjny.

bądź są typem tablicowym, prywatnym, dostępu.
Generyczne podprogramy formalne są postaci
with < specyfikacja podprogramu > is V
gdzie V jest nazwą lub symbolem <>.
Klauzula is jest opcyjna, jej wystąpienie oznacza, że odpowiedni parametr aktualny jest opcyjny, na przykład dla funkcji generycznej SQUARING postaci

```
   generic
       type item is private;
       with function "*" (U, V: item) return
       item is < >
   function SQUARING (X:item) return item
```

mamy dwa generyczne parametry formalne - typ item i funkcję "*".
W przypadku nie wyspecyfikowania parametru aktualnego - funkcji - za ten parametr będzie przyjęta systemowa funkcja mnożenie.

Jednostki generyczne nie mogą być w programie wykorzystywane, stanowią one wzorzec do dalszych deklaracji, postaci

<entity> P1 is new P (PA)

gdzie <entity> jest rodzajem jednostki /podprogram, lub pakiet/, P1 jest nazwą nowo deklarowanej jednostki, P jest nazwą jednostki generycznej, PA jest /opcyjną/ listą parametrów aktualnych. Jednostki tak zadeklarowane mogą być już wykorzystywane w programie /np. podprogramy wywoływane/. W ten sposób sprawdzenie poprawności wyrażeń typów formalnych i podprogramów formalnych jest możliwe w czasie kompilacji. Generyczne pakiety bezparametrowe pozwalają na tworzenie wielu pakietów z tą samą treścią.
Przykłady:

```
   procedure SWAP is
      new EXCHANGE (ELEM => INTEGER);
   package STACK_BOOL is
      new STACK (100, boolean);
   function SQUARING_1 is
      new SQUARING (integer);
   function SQUARING_2 is
      new SQUARING (matrix, matrix_product).
```

WYRAŻENIA

Wyrażenie jest to formuła opisująca obliczenie wartości. W obu językach, definiuje się w podobny sposób wyrażenia arytmetyczne, boolowskie i znakowe; omówimy tylko wyrażenia charakterystyczne dla danego języka.
LOGLAN:
Wyrażenia obiektowe.

Wartością wyrażenia obiektowego jest referencja do obiektu, i typ /tablicowy lub obiektowy/.

Wyrażeniem obiektowym może być:
- stała none /wartość: referencja do pustego obiektu NONE/;
- zmienna /wartość dana przez bieżące wartościowanie/;
- wywołanie funkcji /wartość obliczona wskutek wywołania funkcji/
- generator obiektu postaci new A ($PA_1, \ldots, PA_n$), gdzie A jest nazwą klasy, $PA_1, \ldots, PA_n$ są parametrami aktualnymi /wartość: referencja do nowopowstałego obiektu, utworzonego wg wzorca dostarczonego przez klasę A/,
- obiekt lokalny postaci this A /wartość : referencja do najbliższego syntaktycznie obiektu klasy A/,
- obiekt kwalifikowany, postaci X qua A gdzie X jest wyrażeniem obiektowym, A jest nazwą klasy /wartość równa wartości wyrażenia X/.

ADA:

Agregaty.

Agregat jest to wyrażenie, którego wartością jest tablica /obiekt tablicowy/ lub rekord /obiekt rekordu/. Rozróżnia się agregaty pozycyjne i niepozycyjne.

Agregat pozycyjny jest to lista wartości jakie mają być nadane kolejnym elementom tablicy lub rekordom; postaci $(a_1, \ldots, a_n)$ lub $(a_1, \ldots, a_n$, others $\Rightarrow a_o)$. W drugim przypadku wartość jest wyznaczona przez kontekst /instrukcja podstawienia/ który określa liczbę składowych danego obiektu i pierwszym "n" elementom nadaje wartości $a_1, \ldots, a_n$; a pozostałym wartość a_o.

Agregat niepozycyjny jest to lista indeksów /lub składowych rekordu/ oraz wartości jakie mają być im nadane. Postać dla tablic: $(a_{11} | \ldots | a_{1n} \Rightarrow b_1, \ldots,$ others $\Rightarrow b_o)$, /użycie others jest opcyjne/. Elementom z indeksami $a_{11}, \ldots, a_{1n}$ nadaje się wartość b_1 itp. a pozostałym /kontekst!/ wartość b_o.
Postać dla rekordów $(a_1 \Rightarrow b_1, \ldots, a_n \Rightarrow b_n$, others $\Rightarrow b_o)$.
Przekroje tablic.

Przekrój tablicy X jest to zmienna, której wartością jest tablica złożona z pewnej liczby kolejnych elementów tablicy X. Postać ogólna $A(w_1), \ldots, (w_n)$, gdzie w_i jest zmienną, bądź typem ograniczonym postaci $w_i' \ldots w_i''$.
Przykład
Jeśli A jest zmienną typu STRING (0...30) to wartością zmiennej A(11..20) jest tablica złożona z elementów A (11)... A(20).
Jeśli G jest zmienną typu H(1..5, 1..6) to wartością zmiennej:
- G(1, 2) jest (1, 2) - element tablicy G,
- G(1, .2) jest tablica złożona z elementów G(1, 2), ..., G(1, 6), G(2, 1), ..., G(2, 6).
- G(3) (4..5), jest tablica złożona z elementów G(3, 4), G(3. 5).

Wyrażenia boolowskie.

Zbiór wyrażeń boolowskich rozszerza się o tzw. "obwody logiczne" postaci
- a_1 and then a_2
- a_1 or else a_2

Znaczeniem tych wyrażeń jest koniunkcja /alternatywa/ wyrażeń a_1, a_2, liczona od lewej strony; liczenie jest przerwane przy pierwszej wartości false (true).

REGUŁY WIDOCZNOŚCI NAZW

Przez reguły widoczności nazw rozumiemy reguły ustalania odpowiedności między wystąpieniem nazwy a jej deklaracją.

W obu językach stosuje się, w zasadzie, algolowskie reguły widoczności nazw: nazwa jest widoczna, jeśli wystąpieniu nazwy odpowiada deklaracja tej nazwy /z najmniejszej /tekstowo/ jednostki syntaktycznej, zawierającej to wystąpienie/. Odstępstwa /i rozszerzenia/ od tej zasady opiszemy oddzielnie dla LOGLANu i ADY.

LOGLAN:

Jednostki prefiksowane.

Klasa, podprogram, blok i współprogram mogą być prefiksowane przez klasę. Listę atrybutów jednostki J definiuje się indukcyjnie:
- jeśli J nie ma prefiksu, to atrybutami są atrybuty jej treści /tzn. parametry formalne i wielkości zadeklarowane w J/.
- jeśli Atr/A/ jest listą atrybutów klasy A i klasa J jest postaci

type J : A taken R; class... fin

/symbol class może być zastąpiony przez function, itp/, to zakłada się, że $\emptyset \subseteq R \subseteq$ Atr/A/.

Dla przypadku R = Atr/A/, klauzula taken R może być opuszczona. Lista R jest listą wielkości dziedziczonych z prefiksu: atrybutami jednostki J są atrybuty treści tej jednostki oraz atrybuty z listy R z wyłączeniem atrybutów wirtualnych - podprogramów, które są atrybutami treści jednostki J i mają wirtualną deklarację /type virtual.../ w treści klasy B z ciągu prefiksowego klasy A. Tak więc redeklaracja wielkości wirtualnej nie zasłania poprzedniej deklaracji, ale ją usuwa.

Przez konkatenację instrukcji jednostki J z ciągiem prefiksowanym $A_1, \ldots, A_n$ rozumiemy ciąg instrukcji otrzymany w następujący sposób: Do ciągu instrukcji klasy A_n wstawia się w miejsce instrukcji inner ciąg instrukcji klasy A_{n-1}; do otrzymanego ciągu ustawia się w miejsce inner ciąg instrukcji klasy A_{n-2}, itd.

Deklaracja /a w przypadku bloku - wystąpienie/ jednostki J z ciągiem prefiksowym $A_1, \ldots, A_n$ jest równoważna zadeklarowaniu atrybutów jednostki J /w miejscu wystąpienia danej deklaracji/ i zastąpieniu instrukcji tej jednostki przez ich konkatenację; przy czym atrybuty z prefiksu są "mocniejsze" od nazw z otoczenia tekstowego: wystąpieniu O/a/ nazwy, a w treści klasy A_i odpowiada atrybut a na poziomie klasy A_j /j jest najmniejszą liczbą taką, że $i \leq j \leq n$/; lub jeśli taki atrybut nie istnieje, to deklaracja nazwy a z otoczenia tekstowego wystąpienia deklaracji jednostki J /wg. zwykłych reguł/. Widoczność atrybutów klasy w jednostkach przez tą klasę prefiksowanych może być ograniczona przez listę wielkości schowanych /hidden/.

Podane zasady w pełni określają reguły widoczności nazw w przypadku prefiksowania wielopoziomowego, przy założeniu rozłączności zbiorów atrybutów różnych jednostek /tzn. nie konfliktowości nazw/; ogólny opis p. Raport LOGLANu.

ADA:

Nazwy "przeładowane" /overloaded/.

Algolowskie reguły zasłaniania nazwy są znacznie osłabione: deklaracja podprogramu f nie zasłania deklaracji podprogramu o tej samej nazwie, jeśli jest choć jedna różnica np. w typie argumentów czy wartości /podobnie dla nazw podprogramów systemowych/:

```
function "*" (in X, Y : COMPLEX) :
                : COMPLEX ...
function "*" (in X : int, Y : COMPLEX) :
                : COMPLEX ...
```

gdzie * jest znakiem mnożenia.

Poprawne będą dla C, D : COMPLEX, I, J : int

```
C*D   /mnożenie zesp/
I*J   /zwykłe mnożenie/
C*I   /mnożenie zesp. przez int/
```

Widoczność nazw z wyższych poziomów można ograniczyć, stosując klauzulę

```
restricted (A, B, C)
```

w pakiecie np. Q. Jeśli lista nazw jest pusta, to w pakiecie Q są widoczne tylko nazwy zdefiniowane systemowo. Pierwszą nazwą na liście może być nazwa jednostki Q_1 zawierającej pakiet Q, w tym przypadku nazwy modułów lokalnych dla Q_1 są widoczne. Pozostałe nazwy na liście, muszą być nazwami modułów zewnętrznych dla Q i są widoczne w Q.

Wielkości lokalne podprogramów i modułów mogą być dostępne przy pomocy "odległego dostępu" postaci ⟨nazwa podprogramu/modułu⟩.⟨nazwa wielkości lokalnej⟩. Odległy dostęp może być uzyskany przy pomocy klauzuli use /podobnej do inspect z SIMULI/, w jednostce, w której występuje klauzula use $P_1, \ldots, P_n$; wielkości zadeklarowane w jednostkach $P_1, \ldots, P_n$ są dostępne bezpośrednio.

INSTRUKCJE

Instrukcja podstawienia

Ogólna postać /w obu językach/ N : W, gdzie N jest zmienną, W jest wyrażeniem.

Przykłady

```
LOGLAN : ORIGIN := new POINT (0, 0);
         VERTEX . LEFT. RIGHT := P;
ADA:             A := 5;
                 A(1..10) := A(11..20);
```

Instrukcja warunkowa.

LOGLAN:

Ogólna postać: if a then $I_1;\ldots;I_n$ else $J_1;\ldots;J_m$ fi,

gdzie a jest wyrażeniem boolowskim, $I_1,\ldots,J_m$ są instrukcjami.

Jeśli wartością wyrażenia a jest prawda, to wykonuje się ciąg instrukcji $I_1,\ldots,I_n$; w pp. ciąg $J_1,\ldots,J_m$.

ADA:

Ogólna postać: if a_1 then $I_1;\ldots;I_n$
elsif a_2 then $J_1;\ldots;J_m$
.....
else $K_1;\ldots;K_r$
endif

Znaczenie analogiczne do zapisanego w języku LOGLAN:

if a_1 then $I_1;\ldots;I_n$ else
if a_2 then $J_1;\ldots;J_m$ else
.....
else $K_1;\ldots;K_r$ fi

Instrukcje pętli.

LOGLAN:

Instrukcja pętli jest postaci do $I_1;\ldots;I_n$ od, gdzie $I_1,\ldots,I_m$ są instrukcjami. Zakończenie wykonywania pętli następuje w przypadku wystąpienia instrukcji sterowania exit. Wielokrotne instrukcje sterowania exit ... exit umożliwiają zakończenie zagnieżdżonych pętli. Tak więc w pętli może być wiele wyjść, z różnych poziomów zagnieżdżenia. Dodatkowo wprowadza się instrukcję pętli ze zmienną sterującą, w której jednym z warunków zakończenia pętli jest osiągnięcie przez zmienną sterującą; zwiększaną lub zmniejszaną o stałą wartość /krok/ w każdym przebiegu pętli, pewnej wartości granicznej oraz instrukcję pętli while.

Przykład

```
while r >= b do q := q+1; r := r-b od;
while i <= n do s := 1; j := m;
        while pattern (j) = text (s)
              do j := j-1;
                 if j = 0 then write (i);
                        exit exit fi ;
                 (* wyjście z obu pętli *)
                 s := s-1
              od ;
           i := i+1
        od
```

ADA:

Instrukcja pętli ma postać loop $I_1;\ldots;I_n$ endloop; i może być poprzedzona klauzulą while:

```
while a loop ... end loop
```

bądź for:

```
for I in T loop... endloop
```

Zmienna I w pętli for jest zadeklarowana implicite.

Pętla może być etykietowana. Wyjście z pętli /bez klauzuli while i for/ następuje po wykonaniu instrukcji wyjścia, jeśli jest spełniony warunek a: exit when a; lub dla pętli etykietowanych exit L when a.

Przykłady

```
loop exit when r < b; q := r := r-b end loop;
<< L >> loop  s  := 1; j := m;
   loop exit when pattern (j) /= text (s);
      j := j-1;
      exit L when j = 0;
      s := s-1
   end loop;
   ....
end loop.
```

U w a g a. Składnia pozwala na wprowadzenie do pętli wielu wyjść, ale w przeciwieństwie do LOGLANU, przy wyjściu z pętli nie można już wykonać żadnych akcji. W przypadku wielu wyjść, można wprowadzić dodatkową zmienną typu wyliczeniowego, np.:

```
var ESCAPE :(A, B, NORMAL) := NORMAL.
/wartość pocz. NORMAL/
    loop
        ...
        ESCAPE := A : exit when a1;
        ...
        ESCAPE := B ć exit when a2;
        ...
    endloop;
case ESCAPE of
        when A -- ...
        when B -- ...
    end case
```

Ogólna postać instrukcji case /z dokładnością do składni identyczna w obu językach/:

```
case D of
    when T1 -- J1
    when T2 -- J2
        ...
    when Tn -- Jn
    when others -- Jo
end case
```

Tf są postaci A|B|c .. |Z lub są typem, wyznaczają więc zbiory wartości, jeśli wartość D należy do jakiegoś z tych zbiorów, to jest wykonywany ciąg Ji /wybiera się minimalne i/.

W języku ADA dopuszcza się ograniczoną instrukcję skoku go to wewnątrz jednego podprogramu lub modułu.

Wywołanie podprogramu

LOGLAN:

Rodzaje parametrów i sposoby ich transmisji. Parametrami formalnymi mogą być zmienne, typy i podprogramy. Dla zmiennych mamy dwa rodzaje transmisji:

input: parametr formalny jest jakby zadeklarowany w treści podprogramu, z wartością początkową parametru aktualnego.

output: jak wyżej, z tym, że po zakończeniu wykonywania podprogramu na parametr aktualny podstawia się bieżącą wartość parametru formalnego.

Dla podprogramów i typów mamy jeden rodzaj transmisji: Przed rozpoczęciem wykonywania treści podprogramu oblicza się adres parametru aktualnego. Wszystkie wystąpienia formalnego podprogramu i typu w treści odnoszą się do tego adresu.

Przykłady:

```
        x := F(3)
    call B(g, result);
call SORT (vector, A, mniej);
```

ADA:

Parametrami formalnymi mogą być zmienne. Wyróżnia się trzy rodzaje transmisji:

in, out, in out

Pierwsze dwa mają znaczenie podobne jak powyżej, trzeci umożliwia zarówno odczyt jak zmianę wartości parametru aktualnego.

W deklaracji podprogramu mogą być "proponowane" wartości parametrów aktualnych i wówczas w wywołaniu, odpowiedni parametr. aktualny nie musi /ale może/ wystąpić, np.:

procedure FACT (n : in integer := 5, wynik: out integer) i wywołanie

FACT (res) dla n = 5

lub

FACT (10, res) dla n = 10.

Parametry aktualne mogą być wypisane albo w sposób pozycyjny /w kolejności odpowiadającej parametrom formalnym/, albo niepozycyjny, np:

FACT (wynik =: res, n := 10).

Wykonanie treści podprogramu.

LODLAN:

Podprogram jest zakończony albo wskutek dojścia do końca podprogramu fin, albo wykonania instrukcji sterowania return. W funkcjach może wystąpić /implicite zadeklarowana/ zmienna postaci result, jej wartość wyznacza wartość funkcji. W przypadku gdy wykonanie funkcji nie doprowadziło do nadania wartości, przyjmuje się, że tą wartością jest standardowa wartość początkowa typu /wartości funkcji/. Jeśli podprogram był prefiksowany, to jego treść wykonuje się jak treść skonkatenowanej klasy.

ADA:

Zakończenie wykonania podprogramu następuje wskutek wykonania instrukcji return dla procedury, i return W, dla funkcji /z jednoczesnym nadaniem wartości, w ostatnim przypadku/.

Generator tablicy i obiektu.

LOGLAN:

Generator tablicy umożliwia utworzenie tablicy.

Ogólna postać: array A(L1 : U1)

gdzie A jest zmienną typu array_of ... array_of T.

Generacja tablic n-wymiarowych składa się z n-kroków. Generator tablicy ustala zakresy par granicznych. Tablice traktuje się podobnie jak obiekty, w szczególności wartością zmiennej indeksowanej A(i) jest /n-1/ - wymiarowa tablica. Możliwe jest tworzenie tablic nieregularnych kształtów, np. trójkątnych. Operacją odwrotną do generacji tablicy jest operacja likwidacji tablicy.

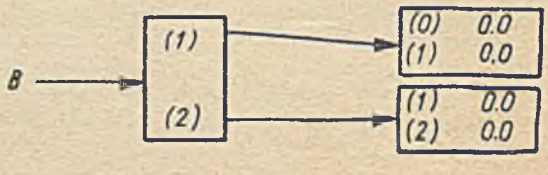
Przykłady:

Dla zmiennej var A : array_of real;

```
    array A(1:3);
    array A(-1, 17).
```

Dla zmiennej var B : array_of array_of real;

```
    array B(1 : 2);
    array B(1) (0 : 1);
    array B(2) (1 : 2).
```



Podamy teraz opis wykonania generatora obiektu klasy prefiksowanej /w podobny sposób opisuje się wywołanie podprogramu prefiksowego i instrukcje bloku prefiksowanego/.
Ogólna postać

new A(PA)

jest PA jest listą parametrów aktualnych, A jest nazwą klasy.
Wykonanie polega na:
- przekazanie parametrów, tzn. policzeniu wartości parametrów wejściowych i adresów podprogramów i typów odpowiadających odpowiednim parametrom formalnym;
- wykonanie konkatencji instrukcji klasy A;
- przekazanie parametrów, tzn. nadanie odpowiednich wartości parametrom wyjściowym.

Instrukcja kopiowania.
LOGLAN:

Instrukcja kopiowania, postać N := copy (W), umożliwia podstawienie na zmienną N referencji do kopii wartości wyrażenia W. Dzięki tej instrukcji, struktury danych /np. tablice/ mogą być przekopiowane do "podzadania" /podprogramu, procesu, itp./ i otoczenie nie może zmienić ich wartości.

Instrukcja likwidacji.
LOGLAN:
Instrukcja likwidacji, postaci kill(X) powoduje zlikwidowanie /dealokację/ obiektu, do którego referencję dostarcza wyrażenie X. Stosowanie tej instrukcji może wydatnie pomóc w gospodarce pamięcią i zmniejszyć czas pracy poświęcony na odśmiecanie.

Obsługa wyjątków /exception/.
ADA:
Wyjątek - exception - jest to rodzaj typu. Nazwy typu exception są w programie deklarowane; ogólna postać : N1,...,Nk : exception. Nazwom tym odpowiadają podprogramy obsługujące wyjątki. Obsługa wyjątków jest opisana w instrukcjach postaci

exception when M1 ⇒ I1;... when MK⇒IK;
when others ⇒ IO
end

gdzie Mi są postaci $M_i^1 | \ldots | M_i^s$; Ii są ciągami instrukcji, /wykonywanych dla wyjątków $M_i^1, \ldots, M_i^s$/.

Systemowo są zdefiniowane pewne nazwy wyjątków i ich obsługa, np. OVERFLOW. Obsługa wyjątków jest wykonywana w przypadku tzw. wywołania wyjątku - wystąpienia instrukcji raise N, gdzie N jest nazwą wyjątku /dla systemowych wyjątków instrukcja raise występuje implicite/.

Ogólne zasady łączenie nazw wyjątków z ich obsługą, są następujące:

Wywołanie wyjątku powoduje przerwanie wykonywania programu. Jeśli wywołanie wyjątku nastąpiło w bloku, lub podprogramie, w którym na zakończenie występuje obsługa wyjątków, to przechodzi się do wykonywania tej obsługi. W pp. odszukuje się najmniejszą jednostkę, w której występuje obsługa wyjątków, i która zawiera dane wywołanie, i przechodzi się do wykonywania tej obsługi /odpowiada to poszukiwaniu obsługi po łańcuchu DL - dynamic linkage/.
Przykład

```
begin
  SINGULAR : exception;
  ....
  if... then raise SINGULAR
  .....
end
exception
  when SINGULAR⇒ PUT ("MATRIX IS
  SINGULAR")
end
```

Instrukcja bloku
Instrukcja bloku jest w obu językach postaci

begin ⟨ciąg deklaracji⟩ ; ⟨ciąg instrukcji⟩
end

z tym, że w LOGLANie, blok może być prefiksowany klasą, w języku ADA, blok może być etykietowany.
Wystąpienie bloku powoduje wykonanie jego instrukcji.
Instrukcje współprogramów
LOGLAN:
Wykonywanie podprogramów i instrukcji klas, raz rozpoczęte i przerwane nie może być wznowione. W przypadku współprogramów /coroutines/, wykonywanie ich instrukcji może być wielokrotnie wstrzymywane i wznowione. W tym celu, wprowadza się instrukcje detach i attach(X). Wykonanie w współprogramie C obu instrukcji powoduje odłączenie /wstrzymanie/ współprogramu C; i w przypadku pierwszej instrukcji przyłączenie /wznowienie/ tego współprogramu, który ostatnio wznowił współprogram C, w przypadku drugiej instrukcji, wznowienie współprogramu, do którego referencję dostarcza wyrażenie X. Zakończenie wykonywania instrukcji współprogramu powoduje efekt analogiczny do wystąpienia instrukcji detach. Główny program jest traktowany jednolicie jako współprogram; zmienna systemowa MAIN umożliwia odwołanie do niego.
Instrukcje równoległe.

Jednostki programowe, które mogą być wykonywane równolegle, są nazywane w LOGLANie - procesami, w języku ADA - zadaniami /task/.
LOGLAN:

Podstawowym problemem programowania równoległego jest dostarczenie użytkownikowi odpowiednich narzędzi synchronizacyjnych, umożliwiających opóźnienie jednego procesu w stosunku do drugiego. Poniżej opiszemy operacje elementarne na procesach:
a/ proces może utworzyć nowy proces /zwany jego synem/, równolegle z nim działający przy pomocy instrukcji start (X)
b/ proces może wstrzymać swoje działanie, dopóki jego syn nie zakończy wykonywania, przy pomocy instrukcji wait (X), gdzie X wskazuje na syna danego procesu.
c/ proces może zażądać wejścia do rejonu krytycznego, identyfikowanego przez zmienną

boolowską x, wykonując instrukcję lock(x). Podstawienie x := false powoduje wyjście z rejonu krytycznego. Instrukcje rejonów krytycznych z tą samą zmienną boolowską /ograniczone wejściem i wyjściem z rejonów/ są wykonywane przez co najmniej jeden proces.es.

d/ proces może się zatrzymać, wykonując instrukcję stop.

e/ proces może się zatrzymać, zwalniając jednocześnie rejon krytyczny, identyfikowany przez zmienną boolowską x, wykonując instrukcję stop(x)

f/ proces może wznowić proces Y, wykonując instrukcję resume (Y).

Wymienione instrukcje elementarne pozwalają na bezpośrednie zarządzanie współpracą między procesami oraz na definiowanie innych narzędzi, oprócz rejonów krytycznych - np. monitorów.

ADA:

Zadanie /task/ jest, wg ostatniej wersji języka, jeszcze jednym typem. Deklaracja zadania jest podobna do deklaracji pakietu, z tym, że w nagłówku zadania mogą występować wyłącznie tzw. wejścia /entry/. Konkretne zadania są definiowane przez zmienne typu task. Przypuśćmy, że deklaracja takiej zmiennej występuje w części deklaratywnej modułu M. Elaboracja tej deklaracji powoduje utworzenie pola danych dla nowego zadania, po zakończeniu wykonywania części deklaratywnej modułu M nowe zadanie zaczyna być wykonywane równolegle z innymi zadaniami.

Przykład:

```
task type resource is
        entry seize;
        entry release;
      end resource
```

Zmienna typu task może być zadeklarowana bez uprzedniego deklarowania tego typu, np.:

```
      type producent_consumer is
       entry read (V:out elem);
       entry write (E: in elem);
   end
```

Na zmiennych typu task nie można wykonywać ani podstawień ani porównań. Dopuszcza się deklaracje typów dostępu do typu task i aktywacje takich zadań, np.:

```
   task type keyboard_driver is
              ....
        end;
   type keyboard is access keyboard_driver;
        terminal:keyboard:=
        new keyboard_ driver.
```

Podamy teraz jeszcze jeden przykład zadania, które oprócz części deklaratywnej ma treść:

```
task protected_array is
   -- komentarz: index i elem są typami glo-
                 balnymi:
entry read (N:in index; V: out elem);
entry write(N: in index; E: in elem);
end;
task body protected_array is
table:array (index) of  elem := (index ⇒0);
begin
  loop
    select
       accept read (N: in index; V: out elem)
                    do
           V:=table (N); end read;
       or
        accept write (N: in index; E: in elem)
                    do
           table (N) := E; end write;
   end select;
end loop;
end protected_array.
```

Synchronizację zadań umożliwiają tzw. wejścia oraz akceptacje.

Wystąpienie wejścia postaci

entry ⟨nazwa⟩ (⟨parametry aktualne⟩)

/gdzie nazwa identyfikuje zadanie np. A i wejście, np. E/ powoduje tzw. wywołanie tego wejścia /zadanie A nazywa się wywołane /. Wywołanie wejścia E powoduje wykonanie akceptacji tego wejścia

accept⟨nazwa E⟩ (⟨parametry formalne⟩)
do⟨ciąg instrukcji⟩ end

Jednakże wykonanie tej instrukcji może być opóźnione. Rozpatrzmy wywołanie wejścia E przez zadanie B, niech A będzie zadaniem wywołanym. Jeśli wywołanie nastąpi zanim, zadanie A dojdzie do akceptacji wejścia E, to zadanie B zostaje wstrzymane. Jeśli zadanie A dojdzie do akceptacji wejścia E zanim jakiekolwiek inne zadanie wywoła to wejście, to zadanie A zostaje wstrzymane.

Wykonanie akceptacji, nazywa się rendezvous, w tym czasie jedno z zadań jest wstrzymane. Po zakończeniu rendezvous oba zadania kontynuują równolegle swoje działanie.

Zauważmy, że zadanie wywołuje wejście w konkretnym zadaniu, podając jego nazwę. np.

X. read (v)

Natomiast akceptacja jest anonimowa, od dowolnego zadania, które wywoła to wejście, np.

```
        accept read (v : out elem) do
          v := local _ elem;
        end
```

Jeśli wiele zadań wywoła wejście w jednym zadaniu, to są one kolejkowane.

Wywołanie i akceptacja mogą być zmienione przy pomocy tzw. akceptacji selektywnej:

```
   select
     when a1 ⇒ ⟨akceptacja 1⟩
     ....
   or when an ⇒⟨akceptacja n⟩
     else ⟨ciąg instrukcji⟩
   end select
```

Wykonanie tej instrukcji polega na wykonaniu dowolnej /losowo wybranej/ akceptacji dla której warunek a_i jest spełniony, bądź ciągu instrukcji występującego po else, jeśli żaden z warunków a_i nie jest spełniony. /Opuszczenie klauzuli when rozumie się jako spełnienie warunku/.

Przyklad

```
select
    accept START;   R := R+1;
or
    when R = 0  accept WRITE (E : in elem)
                do RES := E end
    end select
```

Uwaga: Narzędzia synchronizacyjne ADY mogą być wyrażone w LOGLANie - patrz "On the properties of ADA's rendezvous and an implementation of its LOGLAN's counterpart" - T. Müldner, rękopis.

PODSUMOWANIE

W zakończeniu podamy krótkie podsumowanie ważniejszych konstrukcji językowych LOGLANu i ADY, pod względem filozofii programowania. Celowo używamy tu termin "filozofia" a nie "efektywność" czy "wyrażalność". Oba języki są jeszcze bardzo młode, brak doświadczeń zarówno co do zastosowanych implementacji jak i przydatności dla szerszego grona programistów. Nie można stwierdzić, że ".. konstrukcja a języka A jest lepsza niż konstrukcja b języka B.." gdyż może się okazać, że konstrukcji a nie można efektywnie zaimplementować.

Język ADA ma być używany do celów wojskowych i komercyjnych. Zgodnie z ideami języka PASCAL, struktury danych i typów ADY są bardzo rozbudowane. Wprowadzono dużą liczbę pojęć pierwotnych, niezbyt rozbudowano środki strukturalizacji, /co zapewne wynika z poglądu, że takie właśnie języki mogą być efektywnie implementowane/.

Język LOGLAN jest przeznaczony dla szerokiego grona odbiorców - od studentów informatyki i teoretyków oprogramowania do programistów zatrudnionych w przemyśle. Dla tych ostatnich szczególnie cenna jest możliwość łatwego budowania /i wykorzystywania/ dialektów problemowo zorientowanych. Modularność języka i oddzielna kompilacja modułów programowych umożliwiają zespołowe programowanie i uruchamianie dużych systemów.

Podstawowym funktorem programotwórczym języka LOGLAN jest prefiksowanie. Wydaje się, że właśnie prefiksowanie /w połączeniu z oddzielną kompilacją klas/ umożliwia w pełni strukturalne programowanie: budowanie złożonych programów z oddzielnych modułów, osobno sprawdzonych i skompilowanych. Trudno obecnie w pełni docenić znaczenie prefiksowania wielopoziomowego; wydaje się, że jest to poważny krok naprzód w rozwoju języków programowania.

W obu językach zdecydowano się na wprowadzenie bardziej, niż dotychczas stosowano, elastycznych deklaracji tablic, umożliwiając odłożenie ustalenia wartości par granicznych.

Podprogramy /funkcje i procedury/ są w obu językach dość podobne. Wzgląd na efektywność implementacji niewątpliwie zadecydował, że w języku ADA usunięto podprogramy i typy ze zbioru parametrów formalnych i wprowadzono "surogat" - jednostki generyczne. W LOGLANie zdecydowano się na rozwiązanie przeciwne, /skutkiem czego część kontroli musi być wykonana w czasie wykonywania programu/.

W obu językach jest dużo podobnych instrukcji. Instrukcje pętli są niemal identyczne, z tym, że w języku ADA ograniczono postać wyjścia z pętli /jeśli jest spełniony warunek wyjścia, to nie można wykonać wewnątrz pętli żadnych akcji/, zwiększając czytelność kosztem pewnego utrudnienia w programowaniu.

Na koniec omówimy instrukcje równoległe. W języku ADA, jako jedyny mechanizm synchronizacji przyjęto pojęcie /wysokiego poziomu/ - rendezvous. Natomiast w LOGLANie zdecydowano się na bardziej elastyczne rozwiązanie: w języku wprowadzono dość elementarne operacje synchronizacyjne /zaimplementowane w jądrze systemu operacyjnego minikomputera MERA 400/, przy pomocy których można budować operacje synchronizacyjne na wyższych poziomach, w szczególności w opisie języka podano definicję monitora. W ten sposób, programiści mogą wybierać różne operacje synchronizacyjne, w zależności od swych możliwości i danej klasy problemów /np. bardziej niebezpieczne i efektywne na niższym poziomie, lub bezpieczniejsze i mniej efektywne na wyższym poziomie/.

Pragnę podziękować Panu Profesorowi A. Janickiemu za inspirację w podjęciu tematu.

INFORMACJE-NOWOŚCI

NOWE AUTOMATYCZNE MIERNIKI PODZESPOŁÓW RLC

W ośrodku Badawczo-Rozwojowym Technik Komputerowych i Pomiarów "Mera-Centrum" w Warszawie opracowano automatyczny miernik danych pojemności typu E320. W wyniku współpracy ww. OBRTKiP i IKSAiP "Mera-Elwro" opracowano automatyczny miernik RLC typu F318. Oba mierniki przystosowane są do pracy w systemach pomiarowych zbudowanych zgodnie z normą IEC625.

AUTOMATYCZNY MIERNIK DUŻYCH POJEMNOŚCI TYPU E320

Automatyczny Miernik Dużych Pojemności typu F320 przeznaczony jest do pomiaru pojemności, współczynnika stratności tg δ oraz prądu upływu kondensatorów, zwłaszcza tantalowych i elektrolitycznych. Podczas pomiaru prądu upływu kondensator może być polaryzowany z zasilacza wewnętrznego napięciem stałym regulowanym w sposób ciągły w granicach 0 ÷ 100 V, bądź z zasilacza zewnętrznego napięciem do 500 V. Przyrząd wyposażony jest w układ polaryzujący stan naładowania kondensatora oraz w układ rozładowujący kondensator.

Dane techniczne:

Zakres pomiaru pojemności	0,2 µF - 0,2 F w sześciu podzakresach
Zakres pomiaru tg δ	0 - 5
Napięcie pomiarowe	ok. 2 mV
Częstotliwość napięcia pomiarowego	100 Hz
Zakres pomiaru prądu upływu	0,01 µA - 10 nA w 5 podzakresach
Napięcie polaryzacji	0-10V, 0-100V i zewnętrznie 0-500V
Podstawowa niedokładność pomiaru pojemności i prądu upływu	±2%
Wyjścia cyfrowe	Równoległe w kodzie BCD standard TTL
Sygnały programujące	Programowanie funkcji pomiarowej i podzakresu pomiarowego.
Zasilania	220 V, 50 Hz, 35 VA
Wymiary	140x297x350 mm
Masa	8 kg

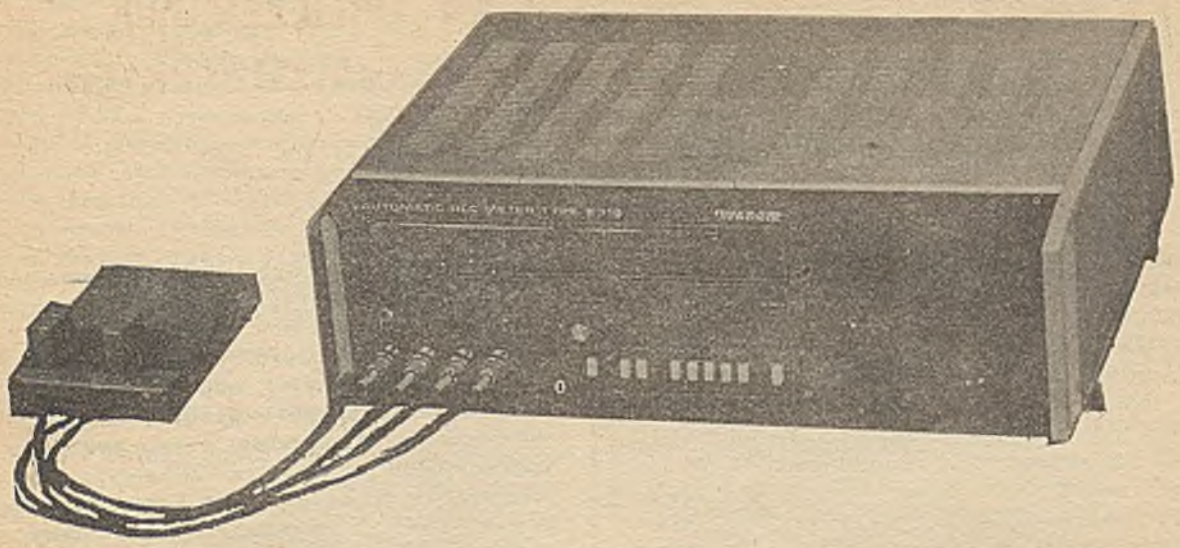
Z blokiem I 320 możliwość pracy w systemach IEC-625.

AUTOMATYCZNY MIERNIK RLC TYPU E318

Automatyczny Miernik RLC typu E318 jest w pełni automatycznym miernikiem służącym do pomiaru pojemności, przewodności, indukcyjności, rezystancji, współczynnika strat i stałej czasu. Pięciopunktowy pomiar zapewnia również dokładny pomiar małych i dużych impedancji. Wyposażenie przyrządu w płytki interfejsowe umożliwia pracę przyrządu w systemach pomiarowych wg standardu IEC-625 i współpracę z systemami kontrolowanymi przez komputer. Wyposażenie przyrządu we wkładkę komparatora cyfrowego umożliwia szybkie sortowanie podzespołów dla obu składowych - rzeczywistej i urojonej. Miernik wyposażony jest w uchwyt pomiarowy do szybkiego wkładania mierzonych elementów.

Dane techniczne:

Funkcje pomiarowe	CG, CD, LR, LD, TR
Częstotliwość pomiarowa	1 kHz
Napięcie pomiarowe	0 - 1 V
Dwa pola odczytowe	$4\frac{1}{2}$ cyfry /maks. 20000/



Fot. 1.

Wybór zakresów	automatyczny, trzymanie zakresu, zdalny
Wybór funkcji	ręczny i zdalny
Czas pomiaru	ok. 1s
Zakres pomiaru pojemności	0,01 pF - 200 µF w siedmiu podzakresach
Podstawowa niedokładność pomiaru pojemności	±0,1%
Zakres pomiaru przewodności	0,1 µS + 2 S
Podstawa niedokładności pomiaru przewodności	±0,1%
Zakres pomiaru indukcyjności	1 µH ÷ 200 H
Podstawa niedokładności pomiaru indukcyjności	±0,2%
Zakres pomiaru rezystancji	0,01 Om + 2 MOm
Podstawowa niedokładność pomiaru rezystancji	±0,1%
Zakres pomiaru współczynnika strat	D 0 ÷ 200%
Niedokładność pomiaru	D ±1%
Zasilanie	220 V -10%, 50 Hz, 60 VA
Wymiary	438x140x350 mm
Masa	ok. 10 kg.

Dodatkowe wyposażenie na zamówienia. Interfejs IEC-625, komparator cyfrowy.

mgr inż. BOGDAN WĄGROWSKI

TECHNIKA OBLICZENIOWA KRAJÓW SOCJALISTYCZNYCH

Zbiór artykułów pod redakcją M. E. Rakowskiego. Specjalistyczne wydawnictwo, wychodzące dwa razy w roku w Moskwie w języku rosyjskim. Wydawnictwo "Statystyka". Redaguje międzynarodowe kolegium w składzie: P. Popow /BRL/, B. Sowa /CSRS/, H. Czoppe /NRD/, M. Wajcen /PRL/, L. Nemet /WRL/, J. P. Seliwanow, E. N. Mielnikowa, W. W. Przałkowski, B. N. Naumow, A. E. Fatiejew, N. I. Czeszenko, A. M. Łarionow, N. W. Gorszkow /ZSRR/, I. Dmitriewa /wyd. "Statystyka"/.

Wydawnictwo przeznaczone jest dla pracowników zajmujących się problemami techniki obliczeniowej, opracowaniem i wykorzystaniem środków Jednolitego Systemu i Systemów Minikomputerowych Elektronicznych Maszyn Cyfrowych .

Do nabycia w Księgarni Wydawnictw Radzieckich, 00-042 Warszawa, ul. Nowy Swiat 47, tel. 27-48-47. Wysyłka za zaliczeniem.

TECHNIKA OBLICZENIOWA KRAJÓW SOCJALISTYCZNYCH - NUMER 7

Rozdział I. Międzynarodowa współpraca krajów socjalistycznych w dziedzinie techniki obliczeniowej.

R. I. Gogunow, M. Kuba: Kompleksowa obsługa - gwarancją efektywnego wykorzystania środków techniki obliczeniowej.

Opisano zagadnienia obsługi kompleksowej środków techniki obliczeniowej, wytwarzanych w ramach prac międzyrządowej Komisji Współpracy krajów socjalistycznych ds. Techniki Obliczeniowej, a przede wszystkim zagadnienia stworzenia jednolitego systemu kompleksowej obsługi. Omówiono nowe rodzaje usług, związane z obsługą oprogramowania. Zamieszczono krótką informację o stanie obecnym obsługi i pracy centrów nauczania i przygotowania specjalistów. Wymienia się podstawowe zadania w zakresie kompleksowej obsługi w najbliższym czasie.

L. N. Iliin, G. W. Bachmurow: Stan obecny i perspektywy rozwoju kompleksowej obsługi środków techniki obliczeniowej /JS EMC/ w ZSRR.

Zestawiono zadania organizacji kompleksowej obsługi środków JS EMC w Związku Radzieckim, a także wykaz usług, które są przez nie spełniane. Omówiono strukturę wszechzwiązkowego zjednoczenia i podstawowe pododdziały /naukowe regionalne centra obsługi, centra szkoleniowe itp/. Rozpatrzono takie rodzaje usług jak wdrożenie do eksploatacji i naprawy gwarancyjne, naprawy awaryjne, wyposażenie w części zapasowe, zaopatrzenie użytkowników w nowe wersje systemów operacyjnych i kompletów PPU, wyposażenie centrów obliczeniowych, konsultacje dla użytkowników. Wymienia się perspektywy utworzenia krajowego zbioru algorytmów i programów oraz kierunki, zgodnie z którymi rozwija się współpraca z NOTO w krajach uczestniczących w Porozumieniu.

J. Kaźmer: Opracowanie techniczne i produkcja EMC w VIDEOTON-ie.

Autor opisał etapy organizacji przemysłu produkującego środki techniki obliczeniowej w VIDEOTON-ie - centrum produkcyjnym techniki obliczeniowej na Węgrzech. Wspomniał jak zaczynała i rozwijała się produkcja emc w VIDEOTON-ie, poświęca uwagę specyfice produkcji niewielkich serii, elastyczności technologii i systemu produkcji w celu realizacji zadań - sprostania specjalnym zapotrzebowaniom użytkowników. Przedstawił środki, wykorzystywane w VIDEOTON-ie dla osiągnięcia wysokiej jakości i niezawodności produkcji. Wymienił zasady i doświadczenia z organizacji obsługi technicznej dostarczanych środków.

Rozdział II. Środki techniki obliczeniowej.

N. Botew, B. Conew, L. Jordanow: Perspektywiczne pamięci zewnętrzne na dyskach magnetycznych dla mini-emc.

Przedstawiono krótki wykaz charakterystyk produkowanych przez przemysł pamięci na dyskach magnetycznych dla mini-emc. Omówiono tendencje rozwoju pamięci na dyskach elastycznych, wymagania użytkowników i wymagania aktualnych norm. Podane zostały grupy perspektywicznych pamięci na dyskach elastycznych dla mini-emc.

Rozdział III. Oprogramowanie EMC.

W. M. Winogradow, W. A. Zołotuchin: Wybrane zagadnienia organizacji programowanych systemów baz danych zautomatyzowanych systemów zarządzania w interaktywnym reżimie.

Omówiono właściwości organizacji baz danych zautomatyzowanych systemów zarządza-

nia, funkcjonującego w interaktywnym reżimie. Uzasadnia się wymagania stawiane informacyjnemu oprogramowaniu tej klasy ZSZ i zasady jego organizacji. Jako jedno z rozwiązań omawianych zagadnień opisuje się system "SPO-baza-terminal". System ten służy do zbierania i zapamiętania danych dla korekty bazy informacyjnej ZSZ i wydania ich na żądanie, a także służy do odbierania wejściowego strumienia zgłoszeń użytkownika wpływającego z oddalonych terminali do biblioteki przetwarzających programów. Omówione zostały funkcjonalne możliwości, przeznaczenie i zasady budowy PPU "SPO-baza-terminal" wykorzystywanego dla prowadzenia bazy informacyjnej ZSZ.

I. Debreceni: Struktura podsystemów użytkowych sterowanych z wielozadaniowego monitora pracującego w czasie rzeczywistym.

Autor opisuje jeden z wielozadaniowych monitorów - RTDM, przeznaczony dla emc JS-1010, JS-1011, JS-1012 przy wykorzystaniu tych maszyn do sterowania procesami lub zdalnego przetwarzania danych. Podaje funkcjonalne możliwości dyskowego monitora, pracującego w reżimie czasu realnego, jego części składowe i ich funkcje w organizacji obliczeniowego procesu - rozdziału zasobów pomiędzy zadaniami. Autor sklasyfikował potrzebne zasoby obliczeniowe i różne programowe metody wykorzystania.

I. Kowacz, J. Madar, I. Nemet: System nauczania z wykorzystaniem monitora podziału czasu JS-1010.

Artykuł zaznajamia z właściwościami części technicznej i programowej systemu nauczania, opartego na możliwościach monitora, pracującego w reżimie podziału czasu dla maszyny JS-1010. Opisuje funkcje, pracę i obsługę części składowych systemu. Uogólnia doświadczenia praktyczne uzyskane w czasie jednego semestru pracy z systemem.

G. A. Ceniłow: Funkcjonalna struktury pakietu programów użytkowych "Kama".

Przedstawiono analizę środków pakietu programów użytkowych "Kama" dostępnych dla różnych kategorii użytkowników. Analiza przeprowadzona została z punktu widzenia możliwości wykorzystania pakietu przy projektowaniu i realizacji systemów, pracujących w reżimie czasu realnego, z pozycji ogólnej metodologii wykorzystania bazowego i ogólnego oprogramowania użytkowego. Analiza wykonana formalnie z orientacją na szerokie kręgi użytkowników JS EMC.

Rozdział IV. Zastosowanie środków techniki obliczeniowej.

I. Szyszkow, J. Bojadżijew, L. Bajadżijewa, C. Lakow: Zarządzanie magazynem przy pomocy mini-emc.

Opisany został opracowany w bułgarskim instytucie CNIKS, zautomatyzowany system sterowania w magazynie sztucznych towarów. Zarządzanie magazynem realizowane jest przez operatorów przy pomocy kompleksu obliczeniowego, utworzonego na bazie mini-emc. Omawia się techniczną strukturę typowego kompleksu obliczeniowego dla zarządzania magazynem na 20 000 miejsc składowych. System pracuje w reżimie czasu realnego. Podane jest oprogramowanie systemu. W procesie zarządzania kompleks obliczeniowy wykonuje funkcje sterowania materiałowym i informacyjnym strumieniem oraz kontroluje stan i pracę sprzętu technicznego. System informuje o stanie i pracy magazynu przy pomocy dokumentów i odpowiedzi na pytania operatora.

M. Palfi: System teleprzetwarzania danych na bazie JS-1022.

Autor opisuje opracowywany system teleprzetwarzania danych, w którym jako centralna emc wykorzystana jest JS-1022 w dużej konfiguracji urządzeń zewnętrznych, zaś jako terminal u użytkownika służy emc JS-1010. Dla synchronizacji przekazywania danych w systemie wykorzystuje się multipleksor MPMH051, synchroniczne modemy TAM-601, synchroniczne urządzenia sprzężenia emc JS-1010 z kanałami łączności. Podporządkowana asynchroniczna sieć JS-1010 zawiera cztery diplay'e JS-7168. Autor przedstawia oprogramowanie środków teleprzetwarzania danych, reżimy pracy emc JS-1010, niektóre doświadczenia eksploatacyjne, a także obecny stan systemu i perspektywy jego rozwoju.

S. Lepetow, J. Sztajer: Wybrane przykłady wykorzystania maszyny cyfrowej JS-1032.

Podano przykłady wykorzystania emc JS-1032 w przedsiębiorstwach dla technicznego przygotowania produkcji, planowania i kontroli produkcji, zarządzania gospodarką materiałową, obliczania płac, obliczania zatrudnienia itp. Wszystkie te funkcje wykonywane są w pakietowym reżimie z wykorzystaniem systemu operacyjnego DOS JS i OS JS. Krótki opis i konfiguracja podsystemu i środków programowych, opis bazy danych. Przytacza się przykłady wykorzystania emc w reżimie teleprzetwarzania: system punktów abonenckich do nauczania studentów, system planowania i kontroli produkcji w organizacji składającej się z przedsiębiorstw oddalonych od siebie oraz system zarządzania w transporcie kolejowym. Opisane zostały wykorzystywane środki programowe i metody dostępu.

Rozdział V. Zagadnienia eksploatacji i obsługi środków techniki obliczeniowej.

O. Punder, J. Zwozil, Z. Wokacz: Scentralizowana obsługa maszyn JS EMC w CSRS.

Przedstawiono etapy tworzenia i rozwoju scentralizowanej obsługi technicznej techniki obliczeniowej. Omówiono tendencje rozwoju i organizacji centralnej obsługi technicznej maszyn cyfrowych JS-1021 w CSRS. Opi-

sano usługi, świadczone w ramach scentralizowanej obsługi technicznej. Wskazane są możliwości oddziaływania na poszczególne wskaźniki niezawodności, wpływające na wydajność maszyny cyfrowej, ze strony organizacji zapewniającej obsługę techniczną. Omówiono organizację bezpośredniej obsługi technicznej i materiałowo-technicznego zaopatrzenia /magazyny, laboratoria obsługi technicznej/.

J. Trajbold: Zadanie przedsiębiorstwa "Kancelarskie stroje" i udział w organizacji służb NOTO w CSRS.

Rozpatrzono funkcje i strukturę instalowania systemów obliczeniowych. Opisany został proces formowania NOTO w CSRS, funkcje i struktura przedsiębiorstw wchodzących w skład NOTO. Podano doświadczenia przedsiębiorstw z instalowania systemów obliczeniowych u użytkowników, udział w opracowaniu konkretnych systemów /MARS/, zadania systematycznego przygotowania kadr i organizację szkolenia specjalistów w NOTO. Opisano pracę centrów obsługi technicznej i obliczeniowych centrów przyjmujących zamówienia od różnych organizacji.

H. Berner. NOTO NRD - funkcje i zadania w aspekcie współpracy międzynarodowej.

Wymienia się usługi, które świadczy NOTO NRD użytkownikom /od bilansowania potrzeb w zakresie techniki obliczeniowej do analizy danych o pracy sprzętu JS EMC i SM EMC/. Usługi te wykonują przedsiębiorstwa NOTO należące do kombinatu ROBOTRON. Opisano doświadczenia ze współpracy w tym zakresie i pierwszoplanowe zadania przy doskonaleniu obsługi technicznej, uwtorzenia jednolitych kryteriów niezawodności, a także stosunku i obowiązków NOTO krajów-dostawców i NOTO krajów-producentów.

W. S. Kuzniecow, W. P. Tichomirow: Organizacja pilotowania środków programowych zautomatyzowanych systemów zarządzania w ZSRR.

Opisano zagadnienia podniesienia efektywności wykorzystania pakietów programów użytkowych przy tworzeniu zautomatyzowanych systemów zarządzania różnego przeznaczenia. Wymieniono problemy związane z organizacją scentralizowanego pilotowania środków programowych; opisana jest klasyfikacja środków programowych ZSZ, doświadczenia pilotowania programów ZSZ w ZSRR i analiza efektywności wykorzystania pakietów programów użytkowych.

P. Jepel: Zwiększenie efektywności i sposoby łączenia ze sobą emc JS-1040.

Rozpatrzono przesłanki i przyczyny konieczności połączenia emc w konfiguracje wielomaszynowe. Opisano doświadczenia sprzężenia emc JS-1040 pomiędzy sobą i z małymi maszynami cyfrowymi, zasady sprzężenia JS EMC przez adapter "kanał-kanał", z pomocą bezpośredniego sygnałowego sterowania, wielokanałowe sprzężenie i sprzężenie przez urządzenia teletransmisji danych.

R. Pilc: Intensyfikacja pracy procesorów przetwarzania danych w centrach obliczeniowych wyposażonych w JS-1040.

Opisane zostały doświadczenia w organizacji prac w zakresie racjonalizacji procesu obliczeniowego w celu podniesienia wydajności systemu przetwarzania danych w Kombinacie ROBOTRON: zwiększenie stabilności pracy centrum obliczeniowego, stałe przygotowanie i dokształcanie personelu, doskonalenie organizacji pracy i technologii, zwiększenie intensywności strumienia zadań. Pokazano racjonalną organizację technologii obliczeniowego procesu w celu pełniejszego wykorzystania zasobów obliczeniowych. Przedstawiono doświadczenia przy racjonalizacji opracowania programów i procesu uruchomienia programów.

I. Pignicki, J. Rech, A. Zoltan: Unifikacja instalacji JS EMC w WRL.

Zaproponowano jednolite koncepcje projektowania instalacji i procesu instalowania środków technicznych JS EMC. Wskazano zadania NOTO w sterowaniu procesem instalacji środków technicznych, a także przedstawiono zawartość poszczególnych prac projektowych i informację niezbędną do ich przeprowadzenia.

B. Mroczek: Generalne dostawy i obsługa techniczna JS EMC w PRL.

Przedstawiono zagadnienia związane z dostawami PRL systemu JS-1032. Przedsiębiorstwo - generalny dostawca dostarcza techniczne i programowe środki, świadczy usługi przy przygotowaniu, wdrożeniu i eksploatacji. Omówiono organizację i zadania biblioteki programów, a także system informacyjny, zwiększający operatywność obsługi użytkowników. Opisany został system przygotowania użytkowników JS EMC w PRL.

P. Sz. Jankielewicz, L. A. Szapinskaja: Organizacja eksploatacji SM EMC /SM-3/.

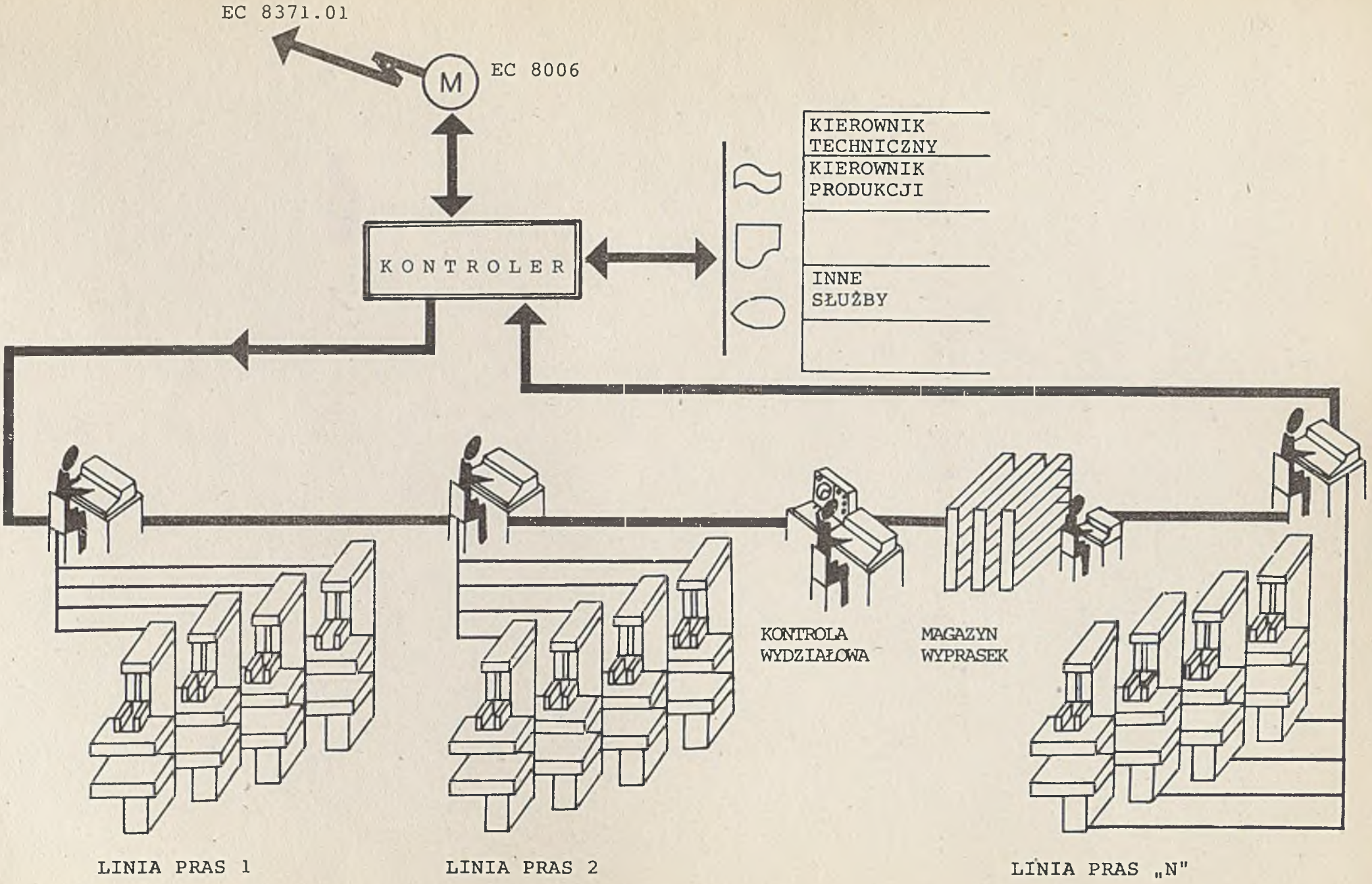
Opisana została organizacja całodobowej pracy na emc SM-3 w reżimie dostępu otwartego, a także zagadnienia związane z organizacją procesu obliczeniowego, dostawą i eksploatacją oprogramowania. Przedstawiono skład i obowiązki grupy obsługi emc, zawartość i kolejność przeprowadzania prac profilaktycznych, metody i środki kontroli maszyn, a także analizę występujących zakłóceń.

Rozdział VI. Nowe środki JS EMC i SM EMC.

J. Klouda: Elektroniczna maszyna cyfrowa JS-1025.

A. D. Michajiow, B. K. Bujuklijew: Wielostanowiskowy system przygotowania danych na taśmie magnetycznej JS-9003.

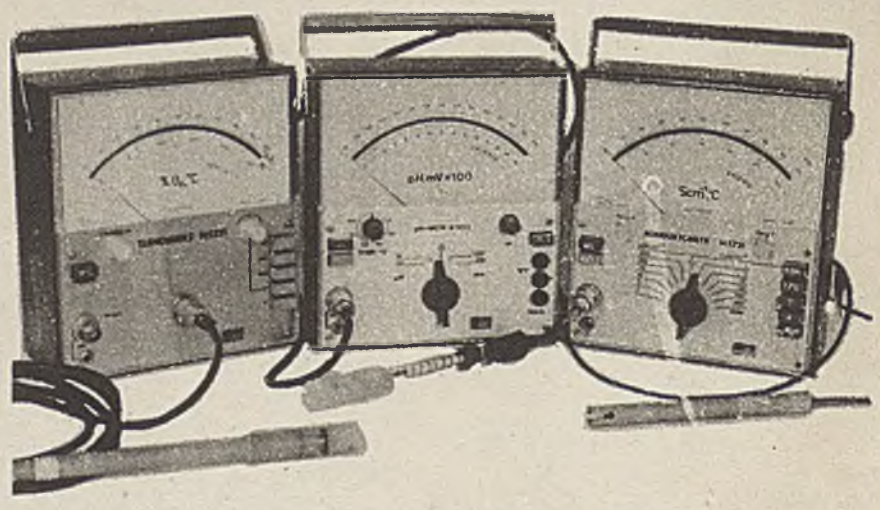
mgr RYSZARDA MALICKA-SZUMIGAJ



Podsystem terminali w zastosowaniu do zbierania danych na wydziale tłoczni

Cena zł 43

Prenumerata roczna zł 516



MERA