inż. GRAŻYNA MIROSZ

Przemysłowy Instytut
Automatyki i Pomiarów MERA–PIAP

Warszawa

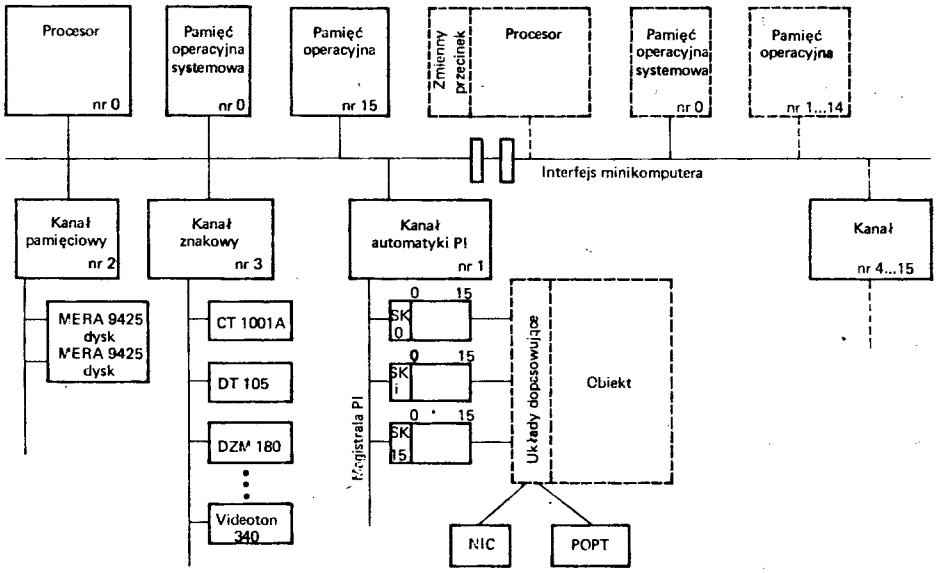
WYKORZYSTANIE SYSTEMU OPERACYJNEGO SOM 3 DO OPROGRAMOWANIA ZESTAWÓW MERA 400-PI

Przedstawiono cechy charakteryzujące system operacyjny minikomputera MERA 400, i SOM 3 w wersji CORE. Opisano przykładowe wykorzystanie tego systemu do oprogramowania PI oraz sposób użycia ekstrakodów INPI i IOPI a także realizowane przez funkcje. System SOM3 do PI jest wykorzystywany w bieżących pracach badawczych MERA–PIAP.

1. Wstęp

Zestawy INTELDIGIT PI są urządzeniami sprzęgającymi komputery cyfrowe różnych typów ze sterowanymi lub kontrolowanymi obiektami. Warunkiem wykorzystania sprzętu PI jest wyspecjalizowane oprogramowanie systemowe i użytkowe.

Do współpracy z PI może być wykorzystany między innymi minikomputer MERA 400. Przedstawione niżej opracowanie dotyczy oprogramowania użytkowego, pracującego na tym minikomputerze połączonym z zestawem PI, pod kontrolą systemu operacyjnego SOM 3.



Rys. 1. Schemat blokowy minikomputera MERA 400 i zestawu PI

2. Ogólne informacje o systemie operacyjnym SOM 3 wersji CORE*)

Istnieją 3 wersje systemu operacyjnego do MERA 400: CORE, BASIC, EXTENDED. Instytut MERA–PIAP badał system operacyjny w wersji CORE. Wersja ta jest przeznaczona głównie dla zamkniętej klasy systemów czasu rzeczywistego, która nie wymaga zewnętrznych pamięci magnetycznych (dla minimalnej konfiguracji pamięci jednostki centralnej). Zakres wprowadzania zmian w programach po dokonaniu instalacji jest ograniczony. Dopuszcza się opcjonalne rozszerzenia systemu o przetwarzanie wsadowe i overlayowanie zadań, jeżeli przewidziano odpowiednie urządzenia wejście/wyjście i pojemność pamięci operacyjnej.

Wersja CORE pracuje w następującej konfiguracji:

jednostka centralna (32K słów szesnastobitowych)
czytnik taśm papierowych CT–1001A
dziurkarka taśm papierowych DT–105,
drukarka mozaikowa DZM–180,
monitor ekranowy Videoton 340.

System operacyjny SOM 3, przewidziany dla takiej konfiguracji, charakteryzuje się następującymi możliwościami:

- Operowanie tylko na zadaniach rezydujących. Liczba zadań jest ustalona w czasie generacji systemu (praktycznie – wersja handlowa jest jednozadaniowa).
- Procedura planująca systemu dzieli czas maszyny, niewykorzystany na obsługę przerwań, pomiędzy poszczególne zadania, zgodnie z priorytetową kolejką prac systemu.
- Każde zdarzenie systemowe powoduje przekazanie sterowania najstarszemu zadaniu. Zadania mogą powodować przekazanie sterowania najstarszemu lub następnemu zadaniu.
- Zadania mogą być wykonywane na jednym z 256 poziomów priorytetu. Każdy poziom dopuszcza istnienie własnej kolejki zadań.
- W zależności od priorytetu dokonuje się podziału zadań na systemowe (pełny dostęp do sprzętu i wewnętrznej struktury systemu), uprzywilejowane (prawo korzystania ze wszystkich ekstrakodów) i zwykłe.
- Przydzielanie zadaniom pamięci operacyjnej w czasie generacji systemu i tworzenie dowolnej konfiguracji bloków pamięci (tzw.podział programowy). Wstępnie ustalone przedziały nie podlegają żadnym zmianom.
- Ochrona pamięci w przypadku lokalizacji jednego zadania w bloku. Jest dopuszczalne istnienie wielu zadań w bloku bez ochrony. Możliwość utworzenia wspólnych dla zadań obszarów komunikacji (COMMON) w bloku pamięci.
- System zapewnia standardową osbługę przerwań. Użytkownik ma możliwość dołączenia własnej obsługi przerwań specjalnych (w przypadku występowania krytycznych czasów obsługi, np.przerwań z PI).
- Ekstrakody wielodostępne mogą być używane we wszystkich zadaniach. Użytkownik ma możliwość dołączenia własnych ekstrakodów. Podstawowa lista ekstrakodów zapewnia: wykonywanie operacji we/wy, sterowanie przebiegiem zadań, obsługę sygnałów zewnętrznych, kontrolę zasobów zadania, analizę ciągów znakowych, konwersję postaci informacji i inne funkcje użytkowe.
- Specjalny podsystem we/wy zapewnia obsługę wszystkich rzeczywistych urządzeń peryferyjnych i pseudourządzeń (symulatorów urządzeń rzeczywistych). Każdy typ urządzenia posiada indywidualną

*) *opracowanym przez OBR TKiP*

procedurę obsługi, zwaną handlerem. Pojedynczy handler może obsługiwać dowolną liczbę urządzeń tego samego typu. Użytkownik ma możliwość dołączenia własnych handlerów. Wszystkie handlery działają pod kontrolą podsystemu we/wy.

- Operacja we/wy może wstrzymać bieg zadania lub odbywać się współbieżnie z nim (time — sharing). Dopuszcza się równoległe prowadzenie wielu operacji we/wy (wielodostępność) w jednym zadaniu.

- Wszystkie operacje we/wy są buforowane. Podsystem we/wy gwarantuje konwersję kodów do jednolitej postaci wewnętrznej (ISO–7). Obsługa błędów operacji we/wy może mieć miejsce w zadaniu lub w systemie.

- Wszystkie operacje we/wy odbywają się za pośrednictwem pseudokanałów przesyłania (tzw.strumieni we/wy), które są niezależne od typu urządzenia. Interpretacja poszczególnych operacji jest dokonywana w handlerach. Dopuszcza się istnienie strumieni lokalnych (tylko dla danego zadania) i globalnych (wspólnych dla wszystkich zadań, przy narzuceniu ograniczeń dostępu).

- Handlery zapewniają jednolity sposób obsługi urządzeń zewnętrznych. Strumienie zadania są przydzielone do urządzeń w czasie generacji systemu, a przydział ten może być zmieniony przez operatora bądź przez program użytkowy.

- System dopuszcza standardowe i niestandardowe, binarne i znakowe typy organizacji danych. Organizacja standardowa umożliwia jednolitą współpracę z każdym urządzeniem, niezależnie od jego typu. Wraz z koncepcją strumieni we/wy stanowi to całkowite uniezależnienie pisanych programów od zestawu urządzeń we/wy.

- Program użytkowy nie jest zaangażowany w obsługę operacji we/wy. Są one kolejkowane do autonomicznego podsystemu we/wy i obsługiwane zgodnie z priorytetem zadań.

- Specjalna klasa pseudourządzeń jest obsługiwana przez systemowe zadania — symbionty. Symulują one urządzenia i wykonują: konwersję kodów, formatowanie i redagowanie, buforują informacje itp. Podsystem we/wy nie odróżnia pseudourządzeń od urządzeń rzeczywistych gwarantując jednolitość aparatu we/wy.

- Standardowa obsługa przerwań sprzętowego zegara pozwala w dowolny sposób zdefiniować programowy dzielnik częstotliwościowy. Jest możliwe dołączanie MONITORA CZASU RZECZYWISTEGO, kontrolującego systemowy zegar i datownik dostępny zadaniom.

- Dopuszcza się rozszerzenie zakresu możliwości monitora czasu rzeczywistego o obsługę dowolnej liczby lokalnych i globalnych zegarów programowych. Zegary mogą działać jako timery (sygnały wielokrotne) lub budziki (sygnał pojedynczy). Pełną kontrolę programową zapewniają odpowiednie ekstrakody.

- Dołączenie MONITORA PRZERWAŃ ZEWNĘTRZNYCH, obsługującego zewnętrzne sygnały dwustanowe. Pełną kontrolę programową zapewniają odpowiednie ekstrakody.

- Zadania mogą być sterowane (startowane, zawieszane, wznawiane, odrzucane, zmiana priorytetu) przez: operatora, podsystem we/wy, zegar programowy, zewnętrzne przerwania lub inne zadania.

- Każdemu zadaniu system przydziela zasoby z globalnej puli systemowej (takie jak: pamięć operacyjna, urządzenia, strumienie we/wy, zegary programowe, przerwania zewnętrzne). Przydziału można dokonać w trakcie generacji systemu, przez operatora lub dynamicznie (samo zadanie w trakcie działania).

- Zasoby dzielą się na permanentne (przydzielone na stałe) i tymczasowe (przydzielone na czas działania). Wszystkie zasoby są rejestrowane na bieżąco i podlegają zwrotowi do puli systemu, w momencie zakończenia zadania (tymczasowe) lub też są zatrzymane do dalszego wykorzystania (permanentne).

- Opcjonalne wyposażenie systemu mogą stanowić: procedura SYSLOADER (ładująca overlay'e

zadań z perferyjnych urządzeń sekwencyjnych), symulator arytmometru wielokrotnej precyzji (uruchamiany standardowymi rozkazami).

- System obejmuje rezydujące zadania komunikacji (interpretuje język operacyjny maszyny), standardowy pakiet dyrektyw zapewniających podstawowe funkcje z zakresu kontroli zadań, urządzeń, zasobów i inne funkcje użytkowe. Jest możliwe uzupełnienie zadania komunikacji własnymi dyrektywami.
- Dołączanie zadania BACKGROUND wraz z programami zwanymi procesorami systemowymi przeznaczonymi do przetwarzania wsadowego w zakresie: kompilacji, translacji, konsolidacji programów, organizacji źródłowych i binarnych bibliotek systemu, oraz uruchamiania programów.

3. Możliwości wykorzystania systemu do oprogramowania zestawu PI*)

Do eksploatacji zestawu MERA 400—PI niezbędne są dwa systemy operacyjne. Jeden system będzie służył do przetwarzania wsadowego w zakresie omówionym w poprzednim rozdziale. Natomiast do współpracy z PI trzeba mieć specjalnie wygenerowany system różniący się nieco od opisanego wyżej. SOM 3 do PI zajmuje pamięć 12K słów 16 bitowych. Pozostałe 20 K jest podzielone na 4 zadania w następujący sposób: jedno zadanie w sysground'zie (uprzywilejowane zadanie systemowe — w systemowym bloku pamięci) nazwane PIØ, posiadające najwyższy priorytet oraz trzy zadania w foreground'zie (zadania uprzywilejowane — we wspólnym wydzielonym bloku pamięci) zwane PI1, PI2, PI3. Pula pamięci odpowiednio 4K, 6K, 5K, 5K.

Do komunikacji z PI służą dwa dodatkowe ekstrakody umieszczone w tablicy ekstrakodów. Są to ekstrakody IOPI i INPI. Ekstrakod IOPI umożliwia realizację funkcji sprzężenia pakietu z komputerem oraz rozkazów do kanału PI. W systemie SOM—3, operacja ekstrakodu IOPI jest wykonywana przez rozkaz IN lub OU. Ekstrakod IOPI umożliwia również deklarację lub odwołanie deklaracji zadania głównego. Ekstrakod INPI umożliwia deklarację obsługi dla pakietu zgłaszającego przerwanie. System ten zawiera również aparat obsługi przerwań z pakietów. Obsługa przerwania jest dokonywana według wcześniejszej deklaracji ekstrakodem INPI.

Realizowane przez ekstrakod IOPI funkcje sprzężenia pakietu z komputerem są następujące:

B1	— funkcja dodatkowa	FETCH
B2	— pisz do pakietu pierwsze słowo	FETCH
B3	— funkcja dodatkowa	FETCH
B4	— pisz do pakietu drugie słowo	FETCH
B6	— czytaj z pakietu pierwsze słowo	SEND
B7	— czytaj zgłoszenia przerwań z pakietów	SEND
B8	— czytaj z pakietu drugie słowo	SEND

Są realizowane następujące rozkazy do kanału PI:

A1	— zeruj kanał	SEND
A2	— blokuj przerwanie wg PN	SEND
A3	— blokuj przerwanie wg $\overline{PN}$	SEND
A4	— ustaw rejestr masek	SEND

*) *W OBR TKiP opracowywany jest translator FORTRANU oraz nowy SOM 3 do współpracy z PI.*

A6 — sprawdź istnienie PI
A7 — podaj specyfikację przerwań kaset
A8 — podaj specyfikację przerwania o najwyższym priorytecie
A9 — ustaw numer procesora

} FETCH

W systemie operacyjnym do PI są stosowane takie pojęcia jak: *zadanie główne* i *zadanie wykonawcze*. Zadanie główne może realizować wszystkie funkcje sprzężenia oraz rozkazy do kanału. Jest ono deklarowane ekstrakodem IOPI. W zadaniu wykonawczym istnieje możliwość realizacji wszystkich funkcji sprzężenia oraz jednego rozkazu do kanału (A7 — podaj specyfikację przerwań kaset).
Przykład deklaracji zadania głównego:

```
LWD,2, # 8000
LWD,4,AB
EXL,IOPI
```

Rejestr 4 — adres pod który będzie zapisana informacja o ewentualnym błędzie.

Wyżej wymienioną sekwencję rozkazów umieszcza się na ogół na początku programu użytkowego, który chcemy uznać za zadanie główne (najczęściej program obsługi przerwań z PI).
Istnieje również możliwość odwołania wcześniejszej deklaracji zadania głównego. Umożliwia to następująca sekwencja rozkazów:

```
LWD,2, # 8001
LWD,4,AB
EXL,IOPI
```

Użytkownicy piszący oprogramowanie MERA 400 z PI powinni pamiętać o tym, że cykl programu powinien zachować następującą kolejność:

I rozkaz „zeruj kanał" (A1),
II ewentualną deklarację zadania głównego,
III wstępną obsługę przypadkowych przerwań,
IV właściwy program realizujący żądane funkcje.

Przykładowo, zerowanie kanału (A1) w zadaniu głównym może mieć następującą postać:

```
RZD,AC
LWD,2,AC
LWT,1,2
LWD,4,AB
EXL,IOPI
```

Rejestr 1 — kod rozkazu A1 i adres urządzenia (kanał automatyki nr 1).
Rejestr 2 — zawiera adres spod którego należy wysłać informację.

Stan pakietów przerywających odczytujemy rozkazem „podaj specyfikację przerwania o najwyższym priorytecie" (A8), stosując ekstrakod IOPI. Ponieważ rozkaz ten może być wysłany tylko przez zadanie główne, należy pamiętać o konieczności wcześniejszego zadeklarowania tego zadania. Wyjście pozytywne z ekstrakodu po wysłaniu rozkazu A8 (S(R4) = 0; R4 — adres błędu) oznacza, że istnieją zgłaszające się przerwania. Specyfikację przerwania umieszczoną w R2 należy załadować do R1 i ponownie użyć ekstrakodu IOPI. Cykl dwukrotnego użycia IOPI należy powtarzać tak długo, aż na wysłany rozkaz A8 nie otrzyma się odpowiedzi PE (S(R4) = 14), która oznacza, że wszystkie przerwania zostały obsłużone. Numery błędów można znaleźć w opisie ekstrakodów IOPI i INPI, [2].

Przykład wstępnej obsługi przerwań:

```
*WYSLANIE A8
XX    LWD,1,#1402     KOD A8 ORAZ ADRES URZADZENIA (NR. KAN.=1)
      LWD,2,AC        ADRES POD KTORY NALEZY WYSLAC SPECYFIKACJE
*                     PRZERWANIA
      LWD,4,AB        ADRES BLEDU
      EXL,IOPI        WYSLANIE A8
*SPRAWDZENIE NUMERU BLEDU
      LWI,1,AB
      CWT,1,14
      JES,XX2-$
      CWT,1,0
      JES,XX3-$
      EXL,BINDEC+1
      RFD,BUF1+4      ADRES TEKSTU PRZY WYDRUKU NR. BLEDU
      LWD,4,BUF1
      EXL,MESSAGE+1   WYDRUK (DLA OPERATORA)NR. BLEDU ORAZ TEKSTU
*OBSLUGA ZGLOSZONEGO PRZERWANIA
XX3   LWI,1,AC        SPECYFIKACJA PRZERWANIA
      LWD,2,AC+1      ADRES ODCZYTANEJ INFORMACJI(PRZYCZYNY)
      LWD,4,AB+2
      EXL,IOPI
*WLASCIWY PROGRAM
XX2   ..............................
```

Rys.2

Po wyżej wymienionej sekwencji rozkazów można przystąpić do normalnego działania. Następne przerwania będą dochodziły do minikomputera MERA 400. Przerwania te będą obsługiwane przez system automatycznie, według wcześniejszej deklaracji – przy użyciu ekstrakodu INPI (maksymalnie można użyć 30 deklaracji).

Przykład deklaracji obsługi przerwań z pakietu PI–Ø1 umieszczonego w kasecie nr Ø na pozycji 1/ØØ/Ø1/ jest następujący:

```
LWD,1 # 62Ø2    kod „czytaj z pakietu 2 słowo" + adres
LWD,2,AB        adres odczytanej informacji
LWD,3,@PI1      nazwa zadania obsługującego
LWD,4,AC        adres informacji o błędzie
EXL,INPI
```

Przy użyciu ekstrakodu INPI możemy odwołać deklarację obsługi przerwania z konkretnego pakietu. Przykład na odwołanie deklaracji obsługi pakietu PI–Ø1:

```
LWD,1 # Ø2Ø2
LWD,4,AC+1
EXL,INPI
```

Przy użyciu ekstrakodów IOPI i INPI, użytkownik ma zapewnioną pełną komunikację jednostki centralnej MERA 400 z urządzeniami PI i kanałem automatyki.

Ze względu na ograniczoną pamięć (32K) i zastosowany w systemie przeznaczonym dla MERA–PIAP podział pamięci na zadania, system ten nie może pracować z programami usługowymi (według nomenklatury OBR TKiP zwanymi procesorami systemowymi), ponieważ pamięć zajmowana przez niektóre

procesory jest większa niż pojemność zadania. Dlatego chcąc na przykład przygotować moduł ładowania, trzeba korzystać z jednozadaniowego systemu SOM 3, wersji CORE.
Programy użytkowe są pisane obecnie w MACROASSEMBLERZE – jednym z języków przewidzianych dla MERA 400.

4. Podsumowanie

Minikomputer MERA 400 wykazuje szereg zalet sprzętu i pod wieloma względami nie ustępuje innym rozwiązaniom minikomputerów wykorzystywanych w automatyce. Jednak przy wyżej przedstawionym stanie oprogramowania podstawowego MERA 400 (procesory systemowe), realizacja oprogramowania konkretnej realizacji układu automatyki jest pracochłonna.
Obecna wersja SOM 3 do PI nie jest wersją docelową i może być rozszerzona. Na ukończeniu jest opracowanie translatora FORTRAN oraz systemu operacyjnego do współpracy z PI.

Literatura

[1] Informacje o systemie MERA 400. 1977, ZSM.

[2] Domachowski M.: Oprogramowanie kanału automatyki sprzęgające minikomputer MERA 400 z systemem PI. OBR–TKiP.