442 BE10

PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Ośrodek Automatyzacji Procesów Produkcji

Pracownia Oprogramowania Cyfrowych Systemów Sterowania

Główny wykonawca mgr inż. Bożena Dąbrowska

Wykonawcy mgr inż. Wiktor Culicki, mgr inż. Sławomir Grzechnik, mgr inż. Jacek Jurkowski, mgr inż.Wojciech Nikiel, mgr inż. Stanisław Wółtański

Konsultant mgr inż. Krzysztof Celiński (CNTK w Warszawie)

Nr zlecenia 1119

AUTOMATYCZNY SYSTEM ROZRZĄDZANIA NA STACJI LUBLIN - TATARY
Etap 6.

Oprogramowanie testujące i wspomagające uruchamianie systemów ZWH i HAD.

Zleceniodawca Wschodnia Dyrekcja Okręgowa Kolei Państwowych w Lublinie

Pracę rozpoczęto dnia 22.12.1987 **zakończono dnia** 30.09.1989

Kierownik Pracowni
Dąbrowska
mgr inż.B.Dąbrowska

Z-ca Dyrektora d/s Automatyki
dr inż.T.Gałązka

Kierownik Ośrodka
dr inż.M.Wrzesień

Praca zawiera:

stron	25
rysunków	—
fotografii	—
tabel	—
tablic	—
załączników	3

Nr rejestr. 6334

Rozdzielnik - ilość egz:

Egz. 1	BOINTE
Egz. 2	WDOKP
Egz. 3	Stacja Lublin - Tatary
Egz. 4	OAP-5
Egz. 5	--
Egz. 6	--

Analiza deskryptorowa

System sterowania, stacja rozrządowa, urządzenia mikroprocesorowe

TRANSPORT SZYNOWY, AUTOMATYZACJA, SYSTEM KOMPUTEROWY

Analiza dokumentacyjna

Oprogramowanie testujące moduły zestawu INTELDIGIT-PROWAY oraz programy wspomagającę uruchamianie oprogramowania użytkowego systemów sterowania ZWH i HAD.

Tytuły poprzednich sprawozdań

AUTOMATYCZNY SYSTEM ROZRZĄDZANIA NA STACJI LUBLIN - TATARY

Etap 3. Mały System Operacyjny Czasu Rzeczywistego. Projekt i instrukcja obsługi.
Projekt oprogramowania systemu ZWH.
Projekt oprogramowania systemu HAD.
Projekt oprogramowania systemu LTB.

Etap 1. Specyfkacja wymagań dla systemu ASR.

681.518.5 Systemy nadzoru. automatyczne

UKD 625.1 Kolejnictwo

PIAP 41/88 10000

2

Spis treści

1. Testy modułów systemu INTELDIGIT - PROWAY.

Poniższy opis dotyczy testu następujących modułów systemu INTELDIGIT - PROWAY:

MC-42 moduł we/wy dwustanowych
MC-Ø2 moduł wejść dwustanowych
MA-Ø1 moduł komutatora wejść analogowych
MA-11 moduł przetwornika a/c

Testy są zestawem funkcji napisanych w języku C. Źródła tych funkcji zamieszczono w Załączniku 1.

Ze względu na konstrukcję testów oraz konfigurację zestawów HAD i ZWH, testy zostały podzielone na dwie części uruchamiane różnymi komendami monitora SRTS. Komendy te są następujące:

E - uruchomienie testu modułów MA-Ø1 i MA-11
X - uruchomienie testów modułów MC-Ø2 i MC-42

Komenda E od razu wprowadza użytkownika do testu komutatora i przetwornika a/c (w każdym z zestawów jest tylko jedna para takich modułów). Natomiast w przypadku testowania modułów dwustanowych po wydaniu komendy X, uruchamiany jest system testów dwustanowych. Po jego uruchomieniu użytkownik podaje tylko adres przeznaczonego do badania modułu, na co system odpowiada zgłoszeniem gotowości odpowiedniego (ze względu na typ modułu) testu dwustanowego. Po zakończeniu testowania można wybrać do sprawdzenia inny moduł lub wogóle wyjść z systemu testów.

Konfigurację sprzętu (adresy modułów) należy wprowadzić na etapie kompilacji systemu testów dwustanowych jak i dla testu pary przetwornik - komutator.

Po uruchomieniu system testów dwustanowych zgłasza się na ekranie monitora następującym obrazem:

```
**************************************
H  Testy uruchomieniowe modułow    H
H            PROWAY                H
H                                  H
H            v. 1.Ø.               H
**************************************

       Podaj adres (hex; Ø = monitor):...
```

Rys. 1. Zgłoszenie gotowości testów

Po podaniu adresu modułu, system przechodzi do odpowiedniego testu, w zależności od typu modułu. Można również podać adres Ø co spowoduje wyjście z systemu testów.

1.1 Test modułu MC-Ø2.

Po podaniu adresu odpowiadającego modułowi MC-Ø2 zgłasza się test tego modułu:

```
**********************************************
H  Test uruchomieniowy modułu                 H
H           MCØ2                              H
H                                             H
H          v. 1.Ø.                            H
H---------------------------------------------H
H  Zlecenia testu:                            H
H     R <parametr>  = czytaj rejestr          H
H           Ø  - rej. danych                  H
H           1  - rej. przerwan                H
H           2  - oba rejestry                 H
H     H = ratunku!                            H
H     E = wyjscie z testu                     H
**********************************************

 ....Podaj zlecenie (R/H/E):
```

Rys. 2. Zgłoszenie gotowości testu modułu MCØ2

Test ten wykonuje następujące zlecenia:

E – wyjście z testu, powrót do poziomu na którym można wybrać adres następnego modułu do sprawdzenia (rys.3.)

```
Podaj adres (hex; Ø = monitor):...
```

Rys. 3. Rezultat wykonania zlecenia E

H – opis testu (rys.4.)

```
*****************************************
H  Test uruchomieniowy modułu          H
H          MCØ2                        H
H                                      H
H         v. 1.Ø.                      H
H--------------------------------------H
H  Zlecenia testu:                     H
H     R <parametr>  = czytaj rejestr   H
H           Ø  - rej. danych           H
H           1  - rej. przerwan         H
H           2  - oba rejestry          H
H     H = ratunku!                     H
H     E = wyjscie z testu              H
*****************************************

 ....Podaj zlecenie (R/H/E):
```

Rys. 4. Rezultat wykonania zlecenia H

R - odczyt rejestru danych i/lub przerwań w zależności od parametru:

RØ - odczyt rejestru danych (rys.5.)

R1 - odczyt rejestru przerwań (rys.6.)

R2 - odczyt rejestru danych i rejestru przerwań (rys.7.)

```
>> Zawartosc rejestru danych modułu eaØØH :

RD: IHD =   bb;  IBD = 11Ø111Ø1;  IDD = 187

    IHE =   aa;  IBE = Ø1Ø1Ø1Ø1;  IDE = 17Ø


 ....Podaj zlecenie (R/H/E):
```

Rys. 5. Przykładowy rezultat wykonania zlecenia RØ

```
>> Zawartosc rejestru przerwan modułu ea02H :
 RP: IHD =   BB;  IBD = 10111101;  IDD = 187
     IHE =   AA;  IBE = 01010101;  IDE = 170

        ....Podaj zlecenie (R/H/E):
```

Rys. 6. Przykładowy rezultat wykonania zlecenia R1

```
>> Zawartosc rejestru danych modułu ea00H :
 RD: IHD =   bb;  IBD = 10111101;  IDD = 187
     IHE =   aa;  IBE = 01010101;  IDE = 170

>> Zawartosc rejestru przerwan modułu ea02H :
 RP: IHD =   BB;  IBD = 10111101;  IDD = 187
     IHE =   AA;  IBE = 01010101;  IDE = 170

        ....Podaj zlecenie (R/H/E):
```

Rys. 7. Przykładowy rezultat wykonania zlecenia R2

Oznaczenia użyte przy wyprowadzaniu informacji są następujące:

pierwsza litera: I

druga litera: postać wyprowadzanych informacji
- H - heksadecymalnie
- B - binarnie
- D - dziesiętnie

trzecia litera: oznaczenie złącza
- D - złącze D (we0 - we7)

E - złącze E (we8 - we15)

RD: rejestr danych

RP: rejestr przerwań

Informacje w postaci heksadecymalnej i dziesiętnej są przekazywane na ekran, tak jak zostały odczytane z modułu (wejście o wyższym numerze ma większą wagę). Stan wejść w porządku "naturalnym" można odczytać w postaci binarnej, gdzie MSB odpowiada weØ lub we8, zależnie od złącza.

1.2. Test modułów MC-42.

Po wybraniu adresu modułu MC-42, test tego modułu zgłasza się w sposób następujący:

```
*****************************************
H  Test uruchomieniowy modułu           H
H           MC42                        H
H                                       H
H          v. 1.Ø.                      H
H---------------------------------------H
H  Zlecenia testu:                      H
H     R         = czytaj rejestr d.     H
H     W <info> = pisz do rejestru       H
H          info - l.hex                 H
H     H = ratunku!                      H
H     E = wyjscie z testu               H
*****************************************

      ....Podaj zlecenie (R/W/H/E):
```

Rys.8. Zgłoszenie gotowości testu dla modułu MC-42

Test ten wykonuje następujące zlecenia:

R - czytanie rejestru danych modułu (rys.9)

```
>> Zawartosc rejestru danych modułu ea56H :
 RD: IHD =   bb;  IBD = 11Ø111Ø1;  IDD = 187
     IHE =   aa;  IBE = Ø1Ø1Ø1Ø1;  IDE = 17Ø

      ....Podaj zlecenie (R/W/H/E):
```

Rys.9. Przyklad wykonania zlecenia R

W - pisanie do modułu (rys.1Ø.); argumentem zlecenia musi być czterocyfrowa liczba heksadecymalna, np.:

W ffff

```
    << Dane do modułu ea56 :
        IH = ffff; ID =      -1;

....Podaj zlecenie (R/W/H/E):
```

Rys.1Ø. Przykład wykonania zlecenia W

H - przywołanie skróconego opisu testu (rys.11.)

```
**********************************************
H  Test uruchomieniowy modułu              H
H           MC42                            H
H                                           H
H          v. 1.Ø.                          H
H-------------------------------------------H
H  Zlecenia testu:                          H
H     R          = czytaj rejestr d.        H
H     W <info> = pisz do rejestru           H
H           info - 1.hex                    H
H     H = ratunku!                          H
H     E = wyjscie z testu                   H
**********************************************

  ....Podaj zlecenie (R/W/H/E):
```

Rys.11. Wykonanie zlecenia H

E - wyjście z testu modułu MC-42 (rys.12.), powrót do poziomu na którym można wywołać test innego modułu

```
Podaj adres (hex; Ø = monitor):...
```

Rys.12. Wykonanie zlecenia E

Wpisując liczbę heksadecymalną do modułu MC-42, należy pamiętać, że wyjścia o najniższych numerach mają największe wagi.

1.3. Test modułów MA-11 i MA-Ø1.

Test ten został napisany w języku AZTEC C i uruchomiony pod kontrolą systemu czasu rzeczywistego SIRTOS. Test działa na sterowniku INTELDIGIT - PROWAY jako cyklicznie uruchamiane zadanie ZVOBS. Test jest uruchamiany zleceniem T monitora operatorskiego SRTS.

Test ma za zadanie wykryć wszelkie możliwe stany błędnej pracy pakietów komutatora i przetwornika a/c.

W wyniku pracy testu, na ekranie monitora, zostaje wypisany tekst odpowiedniego komunikatu z następującej listy:

- odczyt prawidłowy
- brak komutatora
- komutator zajęty
- błąd przełączenia komutatora
- brak przetwornika a/c
- błędne przerwanie przetwornika a/c
- zajętość przetwornika a/c
- nadmiar przetwornika
- timeout przetwornika
- pomiar OK, błąd rozłączenia komutatora
- błąd pomiaru i rozłączenia komutatora
- awaria kanału przy włączaniu

Testowanie polega na krokowym wywoływaniu funkcji obsługi komutatora i przetwornika a/c - funkcja ipac(prn,&pom,&pend,zvobs) - (przełączenie kanału komutatora, inicjacja pomiaru, obsługa przerwania od przetwornika, odczyt). Funkcja zwraca wartość będącą numerem kolejnego komunikatu z listy i wartość zmierzoną.

W programie testu użyte są ekstrakody systemu SIRTOS. Mechanizm semaforów systemu SIRTOS, w który jest wyposażona funkcja obsługi komutatora i przetwornika gwarantuje ochronę obszaru krytycznego - dostępu do w/w modułów - co umożliwia testowanie pakietów nawet podczas wykonywanie przez nie pomiarów przewidzianych w programie.

Test jest uruchamiany cyklicznie co 13 sekund.

2. Monitor operatorski SRTS.

Program SRTS służy do uruchamiania zadań użytkowych pod wielozadaniowym systemem operacyjnym R-T SIRTOS. Jest on konsolidowany, podobnie jak jądro systemu, z zadaniami użytkowymi. Dołączenie monitora jest uwarunkowane nadaniem wartości i symbolowi SRTS znajdującemu się w zbiorze exm.h i dołączeniem do konsolidacji zbiorów msrts.o, mints.o i mon.o. Z punktu widzenia systemu operacyjnego monitor jest traktowany jako obsługa przerwania (w zależności od sytuacji programowego albo sprzętowego CTRL-C). Jeśeli monitor nie jest aktywny to nie ingeruje w najmniejszym stopniu w pracę systemu (w szczególności nie powoduje żadnych zmian w istniejących zależnościach czasowych). Wywołanie monitora powoduje zablokowanie układu przerwań i wstrzymanie pracy wszystkich zadań. Nie można oczywiście wstrzymać procesów poza komputerem (np. przychodzących przerwań zewnętrznych), powoduje to nieuniknione zmiany w istniejących zależnościach czasowych. Wady tej nie można niestety w żaden sposób (i w żadnym z podobnych programów) uniknąć. Dzięki wykorzystywaniu przez program monitora mechanizmów systemu operacyjnego istnieje łatwa możliwość śledzenia pracy systemu, której nie mogą zapewnić monitory uniwersalne np. śledzenie bloku kontrolnego zadań, stosów zadań, startowanie zadań dyrektywą monitora itp. Monitor SRTS zgłasza się po inicjacji zasobów komputera i systemu operacyjnego, przed uruchomieniem zadań użytkowych oraz na żądanie operatora po wciśnięciu klawiszy CTRL-C. W opisie komend przyjęto następujące oznaczenia:

segm:offs	- adres heksadecymalny w postaci segment:offset,
count	- liczba heksadecymalna z przedziału 0000-FFFF,
addr	- liczba heksadecymalna z przedziału 0000-FFFF traktowana jako adres bezwzględny,
byte	- liczba heksadecymalna z przedziału 00-FF,
arg	- liczba heksadecymalna jedno lub dwuznakowa traktowana jako bajt albo trzy lub czteroznakowa traktowana jako słowo,

task – nazwa zadania (identyfikator),

[...] – element opcjonalny – pominięty, przyjmuje wartość domyślną (różną w zależności od dyrektywy).

Wszystkie komendy monitora są jednoliterowe. Po komendzie może wystąpić ciąg parametrów oddzielony przecinkami. Akceptowane są zarówno duże jak i małe litery. Monitor nie wymaga wprowadzania nie znaczących zer, nie akceptuje jednak spacji występujących w parametrach.

2.1. Komendy monitora operatorskiego SRTS.

B

Ustawienie pułapki programowej (ang. breakpoint). Komenda jest bezparametrowa. Powoduje wyświetlenie trzech wierszy w postaci:

```
    B0          B1          B2          B3          B4          B5  ..
0000:0000 0000:0000 0000:0000 0000:0000 0000:0000 0000:0000..
_
```

Kursor zostaje ustawiony w pierwszej kolumnie trzeciego wiersza w oczekiwaniu na wprowadzenie adresu pułapki w postaci [segm:]offs. Jeśli segm: jawnie nie wystąpi zostanie przyjęta domyślnie wartość bieżąca rejestru segmentowego CS. Wprowadzenie adresu pułapki powoduje zmianę jej dotychczasowej wartości i przejście kursora na następną pozycję. Możliwe jest jednoczesne ustawienie do ośmiu niezależnych pułapek. Przesuwanie kursora możliwe jest również przez wciśnięcie "klawisza strzałki w prawo" → lub w lewo ←. Powoduje to przeskoczenie przez kursor jednego adresu pułapki zgodnie z kierunkiem strzałki. Po dojściu do skrajnych pozycji kursor przeskakuje na przeciwny koniec wiersza. Wpisanie adresu 0:0 powoduje skasowanie ustawionej pułapki. Wpisanie litery Z powoduje skasowanie wszystkich ustawionych pułapek od pozycji kursora do końca wiersza. Wciśnięcie klawicza CR bez wpisania żadnego znaku powoduje wyjście z trybu ustawiania pułapek. Po dojściu programu do adresu z ustawioną pułapkąnastępuje automatyczne wejście do monitora z podaniem numeru pułapki i adresu.

C[seg1:]off1,[seg2]:off2[,count]

Kopiowanie obszaru pamięci. Adres seg1:offs1 podaje od jakiego adresu kopiujemy, adres seg2:offs2 na jaki kopiujemy. Parametr "count" podaje liczbę kopiowanych bajtów. Jeżeli seg1: nie zostanie podany jawnie, zastaje nadana mu aktualna wartość rejestru DS. Nienadanie jawnie wartości seg2: powoduje przyjęcie wartości seg1:. Jeżeli parametrowi "count"

nie zostanie jawnie nadana wartość to kopiowany jest jeden bajt. Nie sprawdzane jest pokrywanie się kopiowanych obszarów. Jeżeli adres fizyczny seg1:off1 + count > seg2:off2 wystapi błąd.

D[segm:]offs[,count]

Wyświetlanie obszaru pamięci w kodzie HEX i kodzie ASCII. Brak jawnie nadanej wartości segm: powoduje przyjęcie aktualnej wartości rejestru DS. Parametr "count" określa liczbę wyświetlanych bajtów. Pominięcie parametru powoduje nadanie mu wartości 10 (heksadecymalnie).

E

Wyjście z monitora operatorskiego.

F[segm:]offs,arg[,count]

Wypełnianie obszaru pamięci stałą wartością. "Segm" przyjmuje domyślnie wartość rejestru DS. Stała "arg" w zależności od liczby znaków traktowana jest jako bajt lub słowo. Licznik określa liczbę powtórzeń argumentu. Domyślnie licznik "count" przyjmuje wartość 1.

Gtask

Uruchomienie zadania o identyfikatorze "task" (START).

Ktask

Przerwanie pracy zadania o identyfikatorze "task" (KILL).

Iaddr

Odczyt bajtu z portu o podanym adresie. Odczytana wartość zostaje wypisana na ekranie w interpretacji "unsigned dec", "hex" i "ascii".

Jaddr

Odczyt słowa z portu o podanym adresie. Odczytana wartość zostaje wypisana na ekranie w interpretacji "unsigned dec",

"hex" i "ascii".

N

Praca krokowa. Po naciśnięciu klawisza N monitor przechodzi w tryb pracy krokowej. Wypisuje zawartość rejestrów oraz kod instrukcji do wykonania. Naciśnięcie klawisza CR powoduje wykonanie jednej instrukcji śledzonego programu. Naciśnięcie dowolnego innego znaku kończy pracę krokową i powoduje przejście do normalnego trybu pracy. W czasie pracy krokowej ustawione pułapki nie są aktywne.

Oaddr,arg

Wysłanie bajtu albo słowa na podany adres portu.

Ptask

Wyświetlenie obszaru stosu zadania użytkowego "task".

R

Wyświetlenie i zmiana rejestrów. Na ekranie pojawia się wiersz z wartościami rejestrów procesora. Kursor ustawiany jest pod rejestrem którego wartość ma zostaćzmieniona. Możliwe jest cykliczne przesuwanie kursora w wierszu za pomocą strzałek w prawo → lub w lewo ← .Wciśnięcie "pustego" znaku karetki powoduje zakończenie modyfikacji rejestrów i zgłoszenie się monitora.

S[segm:]offs

Modyfikacja zawartości pamięci. W jednym wierszu zostaje wyświetlona zawartość 16 bajtów pamięci (w postaci identycznej jak przy dyrektywie D). Kursor zostaje ustawiony w następnym wierszu pod podanym adresem w oczekiwaniu na wprowadzenie nowej zawartości pamięci, podawanej za pomocą kodu "hex" bajtami albo słowami. Wprowadzenie nowej wartości powoduje automatycznie przejście kursora o bajt lub słowo w prawo (rozróżnienie dokonywane jest na podstawie liczby wprowadzonych znaków). W ramach wiersza można zmieniać

położenie kursora za pomocą strzałek - w prawo → lub w lewo ←. Próba przejścia strzałki w prawo poza wiersz powoduje wyświetlenie nowego wiersza. Wciśnięcie "pustego" znaku karetki powoduje zakończenie modyfikacji rejestrów i zgłoszenie się monitora. Jeśli "segm:" jawnie nie wystąpi to domyślnie zostanie przyjęta bieżąca wartość rejestru segmentowego DS.

T

Wyświetlenie tablicy kontrolnej zadań (TCB). Jej zawartością są: stan zadania, adres wejścia do zadania, początek stosu, bieżący wskaźnik stosu ,adres semafora albo bufora cyklicznego, licznik taktów, cykl restartu zadania, licznik czasu w cyklu i wskaźnik wejścia do koordynatora (dokładne znaczenie wszystkich tych parametrów zostało podane w opisie systemu operacyjnego SIRTOS). Dodatkowo dla zadań oczekujących na semaforze podawana jest aktualna i oczekiwana wartość semafora.

U[segm:]offs[,count]

Dezasemblacja programu od podanego adresu. Jeżeli parametr segm: nie wystąpi jawnie to przyjmuje domyślnie wartość bieżącą rejestru segmentowego CS. Licznik count ogranicza długość analizowanego pola pamięci (w bajtach). Jeżeli nie jest on podany dekodowana jest tylko jedna instrukcja.

+,-,*,/

Arytmetyka na liczbach heksadecymalnych (dodawanie odejmowanie, mnożenie i dzielenie z resztą). Wynik obliczeń obcinany jest do jednego słowa.

CR (karetka)

Przewinięcie ekranu o dwa wiersze w górę i zgłoszenie monitora symbolem "SRTS>" w oczekiwaniu na komendę operatora.

3. Zadania użytkowe systemu operacyjnego SIRTOS wspomagające uruchamianie oprogramowania.

Do systemu operacyjnego SIRTOS dołączono uniwersalne zadania użytkowe KBD, MON i PRT, obsługujące odpowiednio klawiaturę, ekran monitora i drukarkę. Zadania te współpracują z przerwaniami RxRDY, TxRDY i INTB, obsługiwanymi przez funkcje "txrdy", "rxrdy" i "intb". Funkcje te znajdują się w zbiorze "uints.c". Do komunikacji pomiędzy zadaniami wykorzystano mechanizm buforów cyklicznych, obsługiwanych przez funkcje systemowe SENDM i WAITM oraz PUTBUF i GETBUF. System uzupełniono dodatkowo o obsługę przerwania zegara czasu rzeczywistego - funkcja "timer".
Od strony sprzętowej klawiatura i ekran monitora współpracują z mikrokomputerem przez łącze szeregowe V24. Zarówno ze strony mikrokomputera jak i monitora łącze to jest obsługiwane przez programowalny układ USART-a INTEL 8251A. Przyjęta szybkość transmisji wynosi 4800 bitów/s. Ze strony monitora jest ona ustawiana za pomocą połaczeń krosowych, natomiast ze strony procesora jest zadawana przez zaprogramowanie układu timera INTEL 8253 (realizuje to funkcja "timinit") i USART-a (funkcja "initV24"). Układ USART-a jest zaprogramowany jednocześnie na odbiór i nadawanie. W czasie pracy generuje on dwa przerwania RxRDY - gotowość do odczytu znaku, oraz TxRDY - gotowość do transmisji. Łącze równoległe, zarówno od strony mikroprocesora jak i drukarki, obsługuje programowalny układ typu INTEL 8255. Brama B tego układu została zainicjowana do pracy w trybie 1 jako wyjście (funkcja "piainit"). Układ ten generuje przerwanie INTB, zgłaszające swoją gotowość do transmisji.

3.1. Obsługa ekranu monitora.

Do obsługi ekranu monitora służy zadanie MON. Steruje ono wysyłaniem komunikatów na ekran. Komunikatem jest ciąg bajtów, z których pierwszy stanowi semafor, a ostatni - kończący komunikat - jest zerowy (ang. null byte). Pierwszy i ostatni bajt komunikatu nie są wysyłane na ekran. Komunikat musi zawierać sekwencje sterujące pracą monitora. Zadania użytkowe albo procedury obsługi przerwań wysyłają adresy komunikatów do bufora ("skrzynki pocztowej") "buc_1" za pomocą funkcji systemowych SENDM albo PUTBUF. Zadanie MON czeka zawieszone na pojawienie się adresu komunikatu w buforze. Po pojawieniu się adresu komunikatu zadanie zostaje uruchomione przez system operacyjny. Odmaskowuje ono przerwanie TxRDY i sprawdza gotowość USART-a do wysłania znaku (zakłada się, że ten port szeregowy nie jest wykorzystywany przez inne zadania); następnie wysyła drugi znak komunikatu na ekran monitora, po czym zawiesza się w oczekiwaniu na zakończenie komunikatu. Wysłanie komunikatu kolejno znak po znaku, aż do napotkania znacznika końca informacji (ang. null byte), przejmuje obsługa przerwania TxRDY (funkcja "txrdy"). Po wysłaniu komunikat zostaje oznaczony przez podniesienie semafora, który jest jego pierwszym bajtem. W razie potrzeby umożliwia to synchronizację zadań z wypisywaniem komunikatów na ekran. Jeżeli wypisywanie komunikatu nie zakończy się w określonym czasie (jest on zdefiniowany przez parametr MAXKOM), to system operacyjny uruchamia zadanie MON, które rozpoczyna obsługę stanu awaryjnego. Polega ona na wysłaniu odpowiedniego komunikatu o awarii na drukarkę, ponowieniu wysłania komunikatu podczas transmisji którego wystąpił błąd i podniesieniu jego semafora (w przypadku niepowodzenia ponownej transmisji). Analogicznie postępuje się w przypadku braku gotowości USART-a do transmisji. Po wysłaniu wszystkich znaków komunikatu przez obsługę przerwania TxRDY, zadanie MON zostaje uruchomione ponownie przez funkcję "txrdy". Zadanie sprawdza zawartość bufora "buc_1". Jeżeli znajdują się tam adresy następnych komunikatów, zostają one wysłane na ekran w ten

sam sposób. Adresy komunikatówsą pobierane według kolejności przyjścia (kolejka typu FIFO). Jeżeli bufor "buc_1" jest pusty, to zostaje wysłany komunikat sterujący "akur", który powoduje powrót kursora na pozycję, na jakiej znajdował się przed rozpoczęciem wysyłania komunikatów. Następnie zadanie MON maskuje przerwanie TxRDY na poziomie układu przerwań procesora (USART nie jest przeprogramowywany) i zawiesza się w oczekiwaniu na nowy komunikat. Zmianę położenia kursora na ekranie realizuje się przez zmianę jego adresu w komunikacie "akur". Musi ona być dokonywana przy zamkniętym semaforze. Następnie należy wysłać na monitor jakikolwiek niepusty komunikat (w szczególności może to być komunikat "akur"), np.:

```
            .
            .
            .
WAIT ( akur, 1 ) ;                  /* zamknięcie semafora      */
*( akur + 3 ) = wiersz ;            /* ustawienie numeru wiersza */
*( akur + 4 ) = kolumna ;           /* ustawienie numeru kolumny */
SIGNAL ( akur, 1 ) ;                /* otwarcie semafora        */
SENDM ( &buc_1, kom1 ) ;            /* wysłanie komunikatu kom1  */
            .
    /* kursor już na nowej pozycji */
            .
```

3.2. Obsługa klawiatury.

Obsługę klawiatury realizuje zadanie KBD i funkcja obsługi przerwania RxRDY. Po naciśnięciu klawisza jest generowane przerwanie RxRDY. Obsługa tego przerwania przez funkcję "rxrdy", polega na odczytaniu znaku z klawiatury, umieszczeniu go w buforze i uruchomieniu zadania KBD przez podniesienie semafora. Zadanie odczytuje przesłany znak i przeprowadza jego interpretację. "Zwykłe" znaki tworzą komunikat w buforze "buc_2" z jednoczesnym wysłaniem echa znaku na ekran monitora. Wykonuje się to poprzez wysłanie odpowiedniego komunikatu do zadania MON. Jednocześnie zadanie KBD aktualizuje adres kursora tak, aby wskazywał na następną pozycję za wysłanym echem. Następnie zadanie wykonuje funkcję systemową WAIT. Semafor jest jednocześnie licznikiem znaków w buforze i jeżeli jest on pusty (semafor przyjmuje wartość Ø), zadanie zawiesza się w oczekiwaniu na kolejny znak. W przeciwnym przypadku cykl pobierania znaku z bufora powtarza się. Znaki rozpoznane jako "specjalne" są traktowane indywidualnie, np.: BS czyści bufor w którym gromadzone są znaki, ESC zostaje umieszczony zawsze na początku bufora i natychmiast wysłany, CR kończy kompletowanie komunikatu i powoduje przesłanie go do zadania użytkowego.

Pracą zadania KBD sterują trzy zmienne - "dkom", "secret" i "single". Zmienna "dkom" ogranicza długość przyjmowanych komunikatów. Zmienna "secret" steruje wysyłaniem echa. Z definicji przyjmuje ona wartość Ø (OFF), co oznacza wysłanie echa. Zmiana tej wartości na 1 (ON) wstrzymuje wysyłanie echa na ekran monitora. Kursor przesuwa się jedynie po ekranie. Ustawienie zmiennej "single" na 1 (z definicji przyjmuje ona wartość OFF), powoduje, że znaki są wysyłane do zadania pojedynczo, natychmiast po odczytaniu (bez czekania na znak powrotu karetki - CR).

3.3. Obsługa drukarki.

Drukarkę obsługuje zadanie PRT. Działa ono podobnie jak zadanie MON. Komunikat wysyłany na drukarkę jest zbudowany analogicznie jak wysyłany na ekran monitora. Komunikat musi zawierać sekwencje sterujące drukarką (inne niż dla monitora). Zadania użytkowe albo procedury obsługi przerwań wysyłają adresy komunikatów do bufora "buc_3" za pomocą funkcji systemowych SENDM albo PUTBUF. Zadanie PRT czeka zawieszone na pojawienie się adresu komunikatu w buforze. Po pojawieniu się adresu komunikatu zadanie zostaje uruchomione przez system operacyjny. Odmaskowuje ono przerwanie INTB i sprawdza gotowość interfejsu równoległego do wysłania znaku (zakłada się przy tym, że brama B tego portu, pracująca w trybie 1 jako wyjście, nie jest wykorzystywana przez inne zadania). Dalsze czynności i współpraca z procedurą obsługi przerwania INTB są analogiczne jak w zadaniu MON (wyłączając powrót kursora). Komunikat o wykryciu awarii drukarki wysyłany jest na ekran monitora. Po wysłaniu wszystkich komunikatów na drukarkę zadanie PRT zawiesza się w oczekiwaniu na następne.

3.4. Przerwanie zegarowe czasu rzeczywistego

Przerwanie zegarowe, o okresie 100 ms, jest otrzymywane w wyniku odpowiedniego zaprogramowania układu licznika typu INTEL 8253 (za pomocą funkcji "timinit"). Przerwanie to obsługuje funkcja "timer". Funkcjonalnie można wyodrębnić w niej część obsługującą system operacyjny oraz część użytkową, obejmującą zegar czasu astronomicznego i datownik. Część systemowa aktualizuje tablicę bloków kontrolnych zadań (TCB). Co 100 ms jest aktualizowany w niej licznik taktów - "counter", a co sekundę licznik sekund -"counts". Liczniki te umożliwiają pracę systemu operacyjnego w czasie rzeczywistym.

Część użytkowa funkcji "timer" obsługuje zegar i datownik systemowy. Zegar obsługują liczniki: cycle, cls, clm i clh - zliczające odpowiednio: cykle zegarowe 100 ms, sekundy, minuty i godziny. Aktualny czas jest umieszczany w tablicy "setcl" i co sekundę wysyłany na ekran monitora. W momencie startu systemu wszystkie liczniki są zerowe. W celu ustawienia czasu należy je uaktualnić jednocześnie z tablicą "setcl". Wszystkie te czynności należy wykonywać przy zablokowanym układzie przerwań.

Datownik systemowy obsługują liczniki: "day_number", "clday", "clmonth" i "clyear", zliczające odpowiednio: dzień tygodnia, dzień miesiąca, miesiąc i rok. Komunikaty o dniu tygodnia zawiera tablica "day_name", natomiast datę - tablica "setdata". Komunikaty te są wysyłane na ekran jedynie po ich zmianie. Aktualizację datownika należy przeprowadzać analogicznie jak zegara. Datownik działa poprawnie do roku 2100.

4. Program NEWMM.EXE transmisji kodu wynikowego w standarcie INTEL-HEX.

Program NEWMM służy do ładowania kodów wynikowych programów do sterownika przemysłowego INTELDIGIT PROWAY. Do transmisji jest wykorzystywane łącze szeregowe pomiędzy komputerm IBM AT i PROWAYem.

Kody wynikowe przeznaczone do sterownika powinny być zbudowane według standardu INTEL HEX.

Poza załadowaniem programu do sterownika NEWMM ustawia również rejestr CS i opcjonalnie może ustawić dodatkowo rejestry IP i DS.

Użycie programu:

Należy ustawić komendą DOS parametry transmisji dla portu szeregowego używanego do komunikacji z PROWAYem i następnie wywołać program jak niżej:

```
NEWMM  file  CS  [ IP  [ DS ] ]
```

gdzie

- file - nazwa zbioru INTELowskiego
- CS - adres segmentu ładowania, na tę wartość będzie jednocześnie ustawiony rejestr CS
- IP - jeżeli występuje, to na tę wartość będzie ustawiony rejestr IP.
- DS - jeżeli występuje to na tę wartość będzie ustawiony rejestr DS

Po wywołaniu na ekranie pojawią się wprowadzone parametry z prośbą o akceptację. Po jej uzyskaniu rozpoczyna się transmisja poprzedzona napisem

```
LOADING...
```

Bezpośrednio po transmisji ustawiane są rejestry według wartości podanych w czasie wywołania.

W czasie transmisji program NEWMM współpracuje z dyrektywą L

(load) Monitora PROWAYa. Ta dyrektywa nie zachowuje w pełni standardu INTELowskiego (przyjmuje inną postać rekordu z adresem ładowania i nie przyjmuje rekordów do ustawiania rejestrów). Program NEWMM buduje rekord ładowania według wymagań tej dyrektywy. Rejestry są ustawiane przez wykorzystanie dyrktywy Monitora R.

Załącznik 1. Tabulogramy testów

Testy modułów dwustanowych

Testy - funkcja glowna
TESTY.C

Line# Source Line Microsoft C Compiler Version 5.00

```
 1  #include "jacek.h"
 2  #include <stdio.h>
 3
 4  #define ILPAK 9 /* liczba modułow w zestawie */
 5
 6  testy()
 7
 8  begin
 9
10   int i;
11   int rtnval;        /* wartosc zwracana przez funkcje testujaca */
12   int jaktam;        /* sygnalizuje czy podano prawidlowy adres */
13   unsigned int adres;    /* adres modułu */
14
15   extern int mc02();       /* test modułu MC02 */
16   extern int mc42();       /* test modułu MC42 */
17
18   struct proent_t
19     begin
20      unsigned int adres;         /* adres modułu */
21      int (*funct)();       /* funkcja testujaca moduł */
22     end;
23
24   static struct proent_t proway[ILPAK] =
25     begin
26      {0XEA00,mc02},
27      {0XEA04,mc02},
28      {0XEA08,mc02},
29      {0XEA0C,mc02},
30      {0XEA10,mc02},
31      {0XEA14,mc02},
32      {0XEA36,mc42},
33      {0XEA56,mc42},
34      {0XEA66,mc42},
35     end;
36
37   static char witaj[6][50] =
38     begin
39      {"  ************************************ "},
40      {"  H  Testy uruchomieniowe modułow  H "},
41      {"  H            PROWAY                H "},
42      {"  H                                  H "},
43      {"  H            v. 1.0.               H "},
44      {"  ************************************ "},
45     end;
46
47   static char pytanie[] = "Podaj adres (hex; 0 = monitor):...";
48
49   extern void wmark();       /* pisanie znaku na monitor */
```

```
50   extern void wstring();        /* pisanie stringu na monitor */
51   extern void rmes();       /* czytanie wiersza z klawiatury */
52   extern void dis();       /* di */
53   extern void eis();       /* ei */
54
```

Testy - funkcja główna
TESTY.C

Line# Source Line — Microsoft C Compiler Version 5.00

```
 55
 56   extern char *workarea;
 57
 58   /* ============================================================
 59
 60  Program wlasciwy
 61
 62   ============================================================ */
 63
 64   dis();
 65
 66   /* ..........................................................
 67    Powitanie
 68   ..........................................................*/
 69   wstring("");
 70   wstring("");
 71   for (i = 0; i <= 5; i++)
 72     wstring(&(witaj[i][0]));
 73   wstring("");
 74
 75   /*..........................................................
 76     Pytanie o adres i testowanie
 77   ..........................................................*/
 78   rtnval = 0;
 79   while (rtnval == 0)
 80     begin
 81      jaktam = 1;
 82      while (jaktam)
 83        begin
 84         wstring(pytanie);
 85         rmes(4);
 86         sscanf(workarea,"%4x",&adres);
 87         if (adres == 0)
 88   begin
 89    eis();
 90    return(0);
 91   end
 92         for (i = 0; i <= ILPAK; i++)
 93    begin
 94     if (adres == proway[i].adres)
 95       begin
 96        jaktam = 0;
 97        break;
 98       end
 99    end
100        end
101      rtnval = (*proway[i].funct)(adres);
102     end
103  end
```

Testy - funkcja glowna
TESTY.C

Microsoft C Compiler Version 5.00

testy Local Symbols

Name	Class	Type	Size	Offset	Register
i	auto			-0008	
jaktam.	auto			-0006	
rtnval.	auto			-0004	
adres	auto			-0002	
proway.	static	struct/array	36	0000	
workarea.	extern	near pointer	2	***	
witaj	static	struct/array	300	0024	
pytanie	static	struct/array	38	0150	

Global Symbols

Name	Class	Type	Size	Offset
dis	extern	near function	***	***
eis	extern	near function	***	***
mc02.	extern	near function	***	***
mc42.	extern	near function	***	***
rmes.	extern	near function	***	***
sscanf.	extern	near function	***	***
testy	global	near function	***	0000
wstring	extern	near function	***	***

Code size = 00d0 (208)
Data size = 0186 (390)
Bss size = 0000 (0)

No errors detected

Test modułu MC-Ø2
MCØ2.C

PAGE 1
Ø7-12-89
16:14:Ø9

Line# Source Line Microsoft C Compiler Version 5.ØØ

```
 1  #include "jacek.h"
 2  #include <stdio.h>
 3
 4  int mcØ2(adres)
 5     unsigned int adres;       /* adres testowanego modułu */
 6
 7  begin
 8
 9   int loop1;       /* warunek petli 1 */
10   int loop2 ;       /* warunek petli 2 */
11
12   extern void wstring();
13   extern void rmes();
14
15   extern char *workarea;
16
17   loop1 = 1;
18   while (loop1)
19     begin
20      wstring("");
21      wstring("       ****************************************");
22      wstring("       H  Test uruchomieniowy modułu         H");
23      wstring("       H           MCØ2                      H");
24      wstring("       H                                     H");
25      wstring("       H          v. 1.Ø.                    H");
26      wstring("       H-------------------------------------H");
27      wstring("       H  Zlecenia testu:                    H");
28      wstring("       H     R <parametr>  = czytaj rejestr  H");
29      wstring("       H           Ø  - rej. danych          H");
30      wstring("       H           1  - rej. przerwan        H");
31      wstring("       H           2  - oba rejestry         H");
32      wstring("       H     H = ratunku!                    H");
33      wstring("       H     E = wyjscie z testu             H");
34      wstring("       ****************************************");
35
36      loop2 = 1;
37      while (loop2)
38        begin
39         wstring("....Podaj zlecenie (R/H/E):");
40         rmes(4);
41         if ((*workarea == 'E') || (*workarea == 'e'))
42  return(Ø);
43         if ((*workarea == 'H') || (*workarea == 'h'))
44  break;
45         if ((*workarea == 'R') || (*workarea == 'r'))
46  mcØ2p(adres);
47        end
48     end
49  return(Ø);
```

```
50  end
```

Test modułu MC-Ø2 PAGE 2
MCØ2.C Ø7-12-89
16:14:Ø9

Microsoft C Compiler Version 5.ØØ

mcØ2 Local Symbols

Name	Class	Type	Size	Offset	Register
loop2	auto			-ØØØ4	
loop1	auto			-ØØØ2	
adres	param			ØØØ4	
workarea	extern	near pointer	2	***	

```
51
52  /* ..............................................................
53  Test wlasciwy
54  ..............................................................*/
55  int mcØ2p(adres)
56     unsigned int adres;
57
58  begin
59
6Ø   int parametr;    /* parametr do R */
61
62   extern char *workarea;
63
64   sscanf(workarea+1,"%1d",&parametr);
65   switch (parametr)
66     begin
67      case Ø :
68      dane(adres);
69      break;
7Ø      case 1 :
71      przerw(adres);
72      break;
73      case 2 :
74      dane(adres);
75      przerw(adres);
76      default:
77      return(Ø);
78     end
79   return(Ø);
8Ø  end
```

mcØ2p Local Symbols

Name	Class	Type	Size	Offset	Register
parametr	auto			-ØØØ2	
adres	param			ØØØ4	
workarea	extern	near pointer	2	***	

```
81
82  /*.....................................................
83          Odczyt rejestru danych, z wydrukiem
84  .....................................................*/
85  int dane(adres)
```

Line# Source Line — Microsoft C Compiler Version 5.00

```
     86     unsigned int adres;
     87
     88  begin
     89
     90   struct ent_t
     91     begin
     92      unsigned int we1 : 1;
     93      unsigned int we2 : 1;
     94      unsigned int we3 : 1;
     95      unsigned int we4 : 1;
     96      unsigned int we5 : 1;
     97      unsigned int we6 : 1;
     98      unsigned int we7 : 1;
     99      unsigned int we8 : 1;
    100     end;
    101
    102   union rej_t
    103     begin
    104      unsigned int rej;
    105      struct ent_t wej;
    106     end;
    107
    108   union rej_t rejestr_d;        /* zawartosc rejestrow */
    109   union rej_t rejestr_e;
    110
    111   char infoØ[8Ø];
    112   char info1[8Ø];
    113   char info2[8Ø];
    114
    115   extern unsigned int readw();
    116   extern void wstring();
    117
    118   rejestr_d.rej = (readw(adres) << 8) >> 8;
    119   rejestr_e.rej = readw(adres) >> 8;
    120   sprintf(infoØ,">> Zawartosc rejestru danych modułu %4xH :",a
dres);
    121   sprintf(info1,
    122   " RD: IHD = %4x;  IBD = %1d%1d%1d%1d%1d%1d%1d%1d;  IDD = %3d",
    123   rejestr_d.rej,
    124   rejestr_d.wej.we1,
    125   rejestr_d.wej.we2,
    126   rejestr_d.wej.we3,
    127   rejestr_d.wej.we4,
    128   rejestr_d.wej.we5,
    129   rejestr_d.wej.we6,
    130   rejestr_d.wej.we7,
    131   rejestr_d.wej.we8,
    132   rejestr_d.rej);
    133   sprintf(info2,
```

```
134     "       IHE = %4x;   IBE = %1d%1d%1d%1d%1d%1d%1d%1d;   IDE = %3d",
135     rejestr_e.rej,
136     rejestr_e.wej.we1,
137     rejestr_e.wej.we2,
138     rejestr_e.wej.we3,
```

Test modułu MC-Ø2 PAGE 4
MCØ2.C Ø7-12-89
16:14:Ø9

Line# Source Line Microsoft C Compiler Version 5.ØØ

```
   139    rejestr_e.wej.we4,
   14Ø    rejestr_e.wej.we5,
   141    rejestr_e.wej.we6,
   142    rejestr_e.wej.we7,
   143    rejestr_e.wej.we8,
   144    rejestr_e.rej);
   145    wstring("");
   146    wstring(infoØ);
   147    wstring(info1);
   148    wstring(info2);
   149   end
```

dane Local Symbols

Name	Class	Type	Size	Offset	Register
info2	auto			-ØØf4	
info1	auto			-ØØa4	
infoØ	auto			-ØØ54	
rejestr_e	auto			-ØØØ4	
rejestr_d	auto			-ØØØ2	
adres	param			ØØØ4	

```
   15Ø
   151  /*.....................................................
   152          Odczyt rejestru przerwan, z wydrukiem
   153  .....................................................*/
   154  int przerw(adres)
   155     unsigned int adres;
   156
   157  begin
   158
   159   unsigned int rejestrp;
   16Ø
   161   struct ent_t
   162     begin
   163      unsigned int we1 : 1;
   164      unsigned int we2 : 1;
   165      unsigned int we3 : 1;
   166      unsigned int we4 : 1;
   167      unsigned int we5 : 1;
   168      unsigned int we6 : 1;
   169      unsigned int we7 : 1;
   17Ø      unsigned int we8 : 1;
   171     end;
   172
   173   union rej_t
   174     begin
```

```
175       int rej;
176       struct ent_t wej;
177      end;
178
179   union rej_t rejestr_d;        /* zawartosc rejestrow */
```

Test modułu MC-02 PAGE 5
MC02.C 07-12-89
16:14:09

Line# Source Line Microsoft C Compiler Version 5.00

```
180   union rej_t rejestr_e;
181
182   char info0[80];
183   char info1[80];
184   char info2[80];
185
186   extern unsigned int readw();
187   extern void wstring();
188
189   adres = adres + 2;
190   rejestrp = (unsigned)readw(adres);
191   rejestr_d.rej = (rejestrp << 8) >> 8;
192   rejestr_e.rej = rejestrp >> 8;
193   sprintf(info0,">> Zawartosc rejestru przerwan modułu %4xH :"
,adres);
194   sprintf(info1,
195   " RP: IHD = %4x;  IBD = %1d%1d%1d%1d%1d%1d%1d%1d;  IDD = %3d",
196   rejestr_d.rej,
197   rejestr_d.wej.we1,
198   rejestr_d.wej.we2,
199   rejestr_d.wej.we3,
200   rejestr_d.wej.we4,
201   rejestr_d.wej.we5,
202   rejestr_d.wej.we6,
203   rejestr_d.wej.we7,
204   rejestr_d.wej.we8,
205   rejestr_d.rej);
206   sprintf(info2,
207   "     IHE = %4x;  IBE = %1d%1d%1d%1d%1d%1d%1d%1d;  IDE = %3d",
208   rejestr_e.rej,
209   rejestr_e.wej.we1,
210   rejestr_e.wej.we2,
211   rejestr_e.wej.we3,
212   rejestr_e.wej.we4,
213   rejestr_e.wej.we5,
214   rejestr_e.wej.we6,
215   rejestr_e.wej.we7,
216   rejestr_e.wej.we8,
217   rejestr_e.rej);
218   wstring("");
219   wstring(info0);
220   wstring(info1);
221   wstring(info2);
222  end
```

przerw Local Symbols

Name	Class	Type	Size	Offset	Register

```
info2 . . . . . auto                    -00f6
info1 . . . . . auto                    -00a6
info0 . . . . . auto                    -0056
rejestr_e . . . . . auto                -0006
```

Test modułu MC-Ø2 PAGE 6
MCØ2.C Ø7-12-89
16:14:Ø9

Microsoft C Compiler Version 5.ØØ

przerw Local Symbols

Name	Class	Type	Size	Offset	Register
rejestr_d	auto			-ØØØ4	
rejestrp.	auto			-ØØØ2	
adres	param			ØØØ4	

Global Symbols

Name	Class	Type	Size	Offset
dane.	global	near function	***	Ø16a
mcØ2.	global	near function	***	ØØØØ
mcØ2p	global	near function	***	Ø116
przerw.	global	near function	***	Ø2a2
readw	extern	near function	***	***
rmes.	extern	near function	***	***
sprintf	extern	near function	***	***
sscanf.	extern	near function	***	***
wstring	extern	near function	***	***

Code size = Ø3da (986)
Data size = Ø44b (1Ø99)
Bss size = ØØØØ (Ø)

No errors detected

Test modułu MC-42 PAGE 1
MC42.C 07-12-89
16:14:59

Line# Source Line Microsoft C Compiler Version 5.00

```
 1  #include "jacek.h"
 2  #include <stdio.h>
 3
 4  int mc42(adres)
 5    unsigned int adres;
 6
 7  begin
 8
 9   int loop1;      /* warunek petli 1 */
10   int loop2 ;      /* warunek petli 2 */
11
12   extern void wstring();
13   extern void rmes();
14
15   extern char *workarea;
16
17   loop1 = 1;
18   while (loop1)
19     begin
20      wstring("");
21      wstring("       ****************************************");
22      wstring("       H  Test uruchomieniowy modułu         H");
23      wstring("       H          MC42                        H");
24      wstring("       H                                      H");
25      wstring("       H         v. 1.0.                      H");
26      wstring("       H--------------------------------------H");
27      wstring("       H  Zlecenia testu:                     H");
28      wstring("       H     R        = czytaj rejestr d.     H");
29      wstring("       H     W <info> = pisz do rejestru      H");
30      wstring("       H           info - 1.hex               H");
31      wstring("       H     H = ratunku!                     H");
32      wstring("       H     E = wyjscie z testu              H");
33      wstring("       ****************************************");
34
35      loop2 = 1;
36      while (loop2)
37        begin
38         wstring("....Podaj zlecenie (R/W/H/E):");
39         rmes(5);
40         if ((*workarea == 'E') || (*workarea == 'e'))
41  return(0);
42         if ((*workarea == 'H') || (*workarea == 'h'))
43  break;
44         if ((*workarea == 'R') || (*workarea == 'r'))
45  mc42r(adres);
46         if ((*workarea == 'W') || (*workarea == 'w'))
47  mc42w(adres);
48        end
49     end
```

```
50   return(0);
51  end
```

mc42 Local Symbols

Name	Class	Type	Size	Offset	Register
loop2	auto			-0004	
loop1	auto			-0002	
adres	param			0004	
workarea	extern	near pointer	2	***	

```
52
53  /* ..............................................................
54  Test wlasciwy - czytanie
55  ..............................................................*/
56  int mc42r(adres)
57     unsigned int adres;
58
59  begin
60   dane(adres);
61   return(0);
62  end
```

mc42r Local Symbols

Name	Class	Type	Size	Offset	Register
adres	param			0004	

```
63
64  /*..............................................................
65   Test wlasciwy - pisanie
66  ..............................................................*/
67  int mc42w(adres)
68    unsigned int adres;
69
70  begin
71
72   unsigned int info; /* dane do zapisania */
73   char buf0[80];
74   char buf1[80];
75
76   extern char *workarea;
77
78   extern void writew();
79   extern void wstring();
80
81   sscanf(workarea+1,"%4x",&info);
82   writew(adres,info);
83   sprintf(buf0,"<< Dane do modułu %4x :",adres);
```

```
84    sprintf(buf1,"     IH = %4x; ID = %7d;    ",info,info);
85    wstring("");
86    wstring(buf0);
87    wstring(buf1);
88    return(0);
```

```
     89  end

mc42w  Local Symbols

Name                          Class   Type              Size    Offset  Register

buf1. . . . . . . . . . . .   auto                              -00a2
buf0. . . . . . . . . . . .   auto                              -0052
info. . . . . . . . . . . .   auto                              -0002
adres . . . . . . . . . . .   param                              0004
workarea. . . . . . . . . .   extern  near pointer         2     ***

Global Symbols

Name                          Class   Type              Size    Offset

dane. . . . . . . . . . . .   extern  near function       ***     ***
mc42. . . . . . . . . . . .   global  near function       ***    0000
mc42r . . . . . . . . . . .   global  near function       ***    0126
mc42w . . . . . . . . . . .   global  near function       ***    013c
rmes. . . . . . . . . . . .   extern  near function       ***     ***
sprintf . . . . . . . . . .   extern  near function       ***     ***
sscanf. . . . . . . . . . .   extern  near function       ***     ***
writew. . . . . . . . . . .   extern  near function       ***     ***
wstring . . . . . . . . . .   extern  near function       ***     ***

Code size = 01ac (428)
Data size = 02f7 (759)
Bss size  = 0000 (0)

No errors detected
```

Test modułów komutatora i przetwornika a/c

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

                The PROWAY addresses : "proway.h"  file

                                Adresy PROWAY-a
*/

#define MWA     0xEAE0
#define MWB     0xEBE0

#define ACA     0xEA40
#define ACB     0xEB80

#define AS1     0xEA20
#define AS2     0xEA22
#define AS3     0xEA24
#define AS4     0xEA26
#define AS5     0xEA28
#define AS6     0xEA2A

#define CO1     0xEA60
#define CO2     0xEA64
#define CO3     0xEA68
#define CO4     0xEA6C

#define CI1     0xEA70
#define CI2     0xEA74
#define CI3     0xEA78
#define CI4     0xEA7C

#define BS1     0xEB88
#define BS2     0xEB8A
#define BS3     0xEB8C
#define BS4     0xEB8E
#define BS5     0xEB90

#define BS6     0xEB92
#define BS7     0xEB94
#define BS8     0xEB96
#define BS9     0xEB98

#define MX1     0xEBA0
#define MY1     0xEBA2


#define V10        0
#define V01        1
#define V1         2

#define BK       1  /* brak komutatora */
#define KZ       2  /* komutator zajety */
#define BPK      3  /* blad przelaczenia komutatora */
#define BPAC     4  /* brak przetwornika a/c */
```

```
#define PACZ     5  /* pakiet przetwornika a/c zajety */
#define PINT     6  /* bledne przerwanie a/c */
#define NPAC     7  /* nadmiar przetwornika a/c */
#define TPAC     8  /* timeout przetwornika a/c */
#define PPBK     9  /* (1) pomiar OK, blad rozl. komutatora */
#define BPBK    10  /* (2) blad pomiaru i blad rozl. komutatora */
#define AWK     11  /* awaria kanalu przy wlaczaniu */
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              Global functions: ipok, ipac, ipco, ipci, ipca
                 Local functions: ipk
*/
#include "hadh.h"
#include "gcc.h"
#include "guc.h"
#include "gue.h"
#include "mmc.h"
#include "gfe.h"
#include "proway.h"
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/

#define TAC      2      /* czas oczekiwania na WAITOUT-cie na przetw. A/C MA11
*/
#define LPBK     1      /* liczba powtorzen przy bledzie komutatora */
#define LPBP     1      /* liczba powtorzen przy bledzie przetwornika */
#define CPS      5      /* oczekiwanie na przelaczenie stykow dla MA01 > 3.6 ms
*/
#define DSTAC    7      /* oczekiwanie na osiagniecie wartosci zadanej > 7 ms
*/
#define DELTA    2

#define MASK03 0x0008
#define MASK06 0x0040
#define MASK07 0x0080
#define MASK15 0x8000

#define XF 0x000F
#define X7 0x0007

#define JUST     3      /* wlasnie uplynal czas pracy radaru dla danego czujn.
*/


#define TAL      5      /* ilosc zapamietywanych predkosci aktualnych */
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/

int *ac ;
unsigned int swac ;
char semzegar ;
int c_sem ;          /* kwant czasu */
int wsk4 ;           /* wskaznik dla A/C gdy tylko 1 tor zajety */


struct tacr acr [] = {
                        ACA, V10,  ACA, V1,  ACA, V01,
                        ACB, V10,  ACB, V1,  ACB, V01
} ;

struct prom rom [] = {
                    /* adres pakietu komutatora, nr we MA01,
```

```
                                identyfikator przetwornika i jego zakresu */
                        AS1, Ø, Ø,  AS1, 1, Ø,  AS1, 2, Ø,  AS1, 3, Ø,
                        AS1, 4, Ø,  AS1, 5, Ø,  AS1, 6, Ø,  AS1, 7, Ø,
                        AS2, Ø, Ø,  AS2, 1, Ø,  AS2, 2, Ø,  AS2, 3, Ø,
                        AS2, 4, Ø,  AS2, 5, Ø,  AS2, 6, Ø,  AS2, 7, Ø,
                        AS3, Ø, Ø,  AS3, 1, Ø,  AS3, 2, Ø,  AS3, 3, Ø,
                        AS3, 4, Ø,  AS3, 5, Ø,  AS3, 6, Ø,  AS3, 7, Ø,
                        AS4, Ø, Ø,  AS4, 1, Ø,  AS4, 2, Ø,  AS4, 3, Ø,
                        AS4, 4, Ø,  AS4, 5, Ø,  AS4, 6, Ø,  AS4, 7, Ø,
                        AS5, Ø, Ø,  AS5, 1, Ø,  AS5, 2, Ø,  AS5, 3, Ø,
                        AS5, 4, Ø,  AS5, 5, Ø,  AS5, 6, Ø,  AS5, 7, Ø,
                        AS6, Ø, Ø,  AS6, 1, Ø,  AS6, 2, Ø,  AS6, 3, Ø,
                        AS6, 4, Ø,  AS6, 5, Ø,  AS6, 6, Ø,  AS6, 7, Ø,
} ;

char ramci [ NUVI ] ;                                   /* wartosc odczytana
*/

struct rwm ram [ NUV ] ;

struct rwm reg [ NREG ] ;

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
void cdot ( liczba, str, mode, skala )
                                /* dekodowanie dodatniej liczby na kod ASCII
*/
    int liczba,                 /* dekodowana liczba */
        mode,                   /* typ liczby INT = Ø albo FLOAT = 1, 2, 3 */
        skala ;                 /* wskaznik skalowania ( = 5, gdy w [ mV ] )
*/
    char *str ;                 /* wskaznik na pole jednosci */
    {
    char *st ;

    if ( liczba < Ø ) {
        *( str - 1 ) = '-' ;    /* wpisanie "-" przed polem jednosci */
        liczba = - liczba ;
    }

    liczba *= skala ;
    *str = 'Ø' ;                /* wpisanie zera przed kropka dziesietna */
    st = str + ONE ;            /* wskaznik na kropke */
    str += mode ;
    do {
        *str-- = liczba % TEN + 'Ø' ;
                                /* wpisywanie kolejnych cyfr dziesietnych */
    } while ( liczba /= TEN ) ;
    if ( mode ) {
        memcpy( st, st + ONE, mode ) ;
        *st = DOT ;
    }
    return ;
}                                                       /* end cdot
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
*/
int ipok ( pvn, pom, mode )        /* pvn - numer obslugiwanej zmiennej   */
    char *pom ;                    /* pom - semafor                       */
    int pvn, mode ;                /* mode = 1  wlaczenie kanalu  */
    {                              /* mode = Ø  odlaczenie kanalu */
    int  wsk, i ;
    int *ap, nw ;

    nw = rom[ pvn ].nw ;
    ap = rom[ pvn ].ap ;

/* OBSLUGA KOMUTATORA MAØ1 ------------------------------------------------
*/

    WAIT ( pom, 1 ) ;
    for ( i = Ø ; i < LPBK ; ++i )
        if ( ! ( wsk = hipok ( ap, nw, mode ) ) )
            break ;
    SIGNAL ( pom, 1 ) ;
    return ( wsk ) ;
}                                                          /* end ipok
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
static int hipok ( ap, nw, mode )
    int *ap,                           /* ap - adres pakietu komutatora */
        nw,                            /* nw - numer styku komutatora   */
        mode ;                         /* mode = 1    wlaczenie kanalu  */
                                       /* mode = Ø    wylaczenie kanalu */
    {
    int sk, stp, j ;

   /* ap -= halfds ; */

 /*   dis () ; */
    sk = readb1( ap ) ;      /*  *ap */  /* czytaj slowo stanu komutatora
MAØ1*/

  /* if ( btei () )
        return ( BK ) ;  */                    /* brak komutatora */
    if ( sk & MASKØ7 )
        return ( KZ ) ;                       /* komutator zajety */
    stp = ( mode ? MASKØ3 : X7 ) | nw ;       /* ustaw kanal */
   /* dis () ; */
    writeb1( ap , stp );         /*  *ap = stp ; */
   /* if ( btei () )
        return ( BK ) ; */                    /* brak komutatora */
    for ( j = Ø ; j < CPS ; ++j )
        delay () ;                  /* czekaj na przelaczenie stykow > 3.6 ms
*/
    /* dis () ; */
    sk = readb1(ap) ;                /*    *ap & XF ; */    /* czytaj slowo
stanu
komutatora MAØ1 */
 /*   if ( btei () )
```

```
        return ( BK ) ; */                          /* brak komutatora */
    if ( mode ) {
        if ( sk == MASKØ3 ! XF & nw )
            return ( OFF ) ;                      /* przelaczenie prawidlowe */
    }
    else
        if ( ! ( MASKØ3 & sk ) )
            return ( OFF ) ;                      /* odlaczenie prawidlowe */
    return ( BPK ) ;                             /* blad przelaczenia komutatora
*/
}                                                                  /* end hipok
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/

int ipac ( pvn, pom, pend , nazwa )
                                        /* pvn - numer obslugiwanej zmiennej */
    char *pom, *pend ;                  /* pom, pend - semafory
*/
    int pvn , nazwa  ;                  /* wlaczenie kanalu + pomiar a/c
*/

    {
    int i, wsk1, wsk2, wsk3 ;
    int *apk, nw ;
    int *acp, range, pomiar ;
    char mod ;

    nw = rom[ pvn ].nw ;
    apk = rom[ pvn ].ap ;
    mod = rom[ pvn ].mod ;
    acp = acr[ mod ].ac ;
    ac  = acp;                       /*- halfds ;*/ /* extern z obsluga przerwania
*/
    range = acr[ mod ].r ;

/* OBSLUGA PRZETWORNIKA AC MA11 ----------------------------------------------
*/

    WAIT ( pom, i ) ;              /* ochrona obszaru krytycznego */

/* zalaczenie komutatora
------------------------------------------------------*/

        for ( i = Ø ; i < LPBK ; ++i )
            if ( ! ( wsk1 = hipok ( apk, nw, ON ) ) )
                break ;

/* zalaczenie przetwornika A/C
----------------------------------------------*/

        for ( i = Ø ; i < LPBP ; ++i )
            if ( ! ( wsk2 = hipac ( acp, range, pend, &pomiar ) ) ) {
                ram[ pvn ].ws = ram[ pvn ].w ;
                ram[ pvn ].w = pomiar ;
```

```
                  break ;
             }

/* wylaczenie komutatora
-----------------------------------------------------------*/

         for ( i = Ø ; i < LPBK ; ++i )
         if ( ! ( wsk3 = hipok ( apk, nw, OFF ) ) )
            break ;

    SIGNAL ( pom, 1 ) ;         /* ochrona obszaru krytycznego */
    if ( wsk1 )
        if ( wsk3 )
            return ( wsk3 ) ;           /* blad komutatora */
        else
            return ( AWK ) ;            /* awaria kanalu przy wlaczaniu */
    else
        if ( wsk3 )
            if ( wsk2 )
                return ( BPBK ) ;   /* blad pomiaru i blad rozl. komutatora */
            else
                return ( PPBK ) ;   /* pomiar OK, blad rozlaczenia komutatora
*/
        else
            return ( wsk2 ) ;
}                                                                /* end ipac
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
static int hipac ( acp, range, pend, pomiar )   /* obsluga przetwornika AC
*/
    int *acp,                           /* acp - adres pakietu przetwornika
*/
        range,                          /* zakres Ø - 1ØV, 2 - 1V, 1 - Ø.1V
*/
        *pomiar ;                       /* wynik pomiaru - "int" ze znakiem
*/
    char *pend ;                        /* pend - semafor do WAITOUT-u
*/
    {
    char mod ;
    int sp, i ;

  /*  dis () ; */
    sp = readbl(acp + 3 );                          /*    *( acp + 3 ) ; */
/* czytaj slowo stanu przetwornika a/c MA11 */
  /*  if ( btei () )
        return ( BPAĆ ) ;  */          /* brak przetwornika a/c */
    if ( sp & MASKØ7 )
        return ( PACZ ) ;          /* przetwornik a/c zajety */
    *pend = OFF ;                  /* opuszczenie semafora dla WAITOUT-u */
   /*  dis () ; */
    writebl( acp + 2 , range );        /* *( acp + 2 ) = range ; */
                                   /* wpis zakresu z inicjacja pomiaru */
  /*  if ( btei () )
```

```
        return ( BPAC ) ; */             /* brak przetwornika a/c */
    if ( WAITOUT ( pend, 1, TAC ) ) /* czekaj na przerwanie <= timeout */
        return ( TPAC ) ;            /* timeout przetwornika a/c */
    if ( swac & MASK07 )
        return ( PACZ ) ;            /* przetwornik a/c zajety */
    if ( ! swac & MASK15 )
        return ( PINT ) ;            /* bledne przerwanie a/c */
   if ( swac & MASK06 )
        return ( NPAC ) ;            /* nadmiar przetwornika a/c */
    /* dis () ; */
    *pomiar = readbl(acp + 1);        /*   *( acp + 1 ) */
                                      /* odczyt przetwornika a/c
                                         bez inicjacji nastepnego pomiaru */
  /*  if ( btei () )
        return ( BPAC ) ; */           /* brak przetwornika a/c */
    if ( *pomiar < 0 )
        *pomiar =  - ( *pomiar & 0x7FFF ) ;
    return ( OFF ) ;                 /* pomiar poprawny */
}                                                          /* end of hipac
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

                TEST PAKIETOW KOMUTATORA I PRZETWORNIKA A/C

                    The user's file "mzvobs"          Task: ZVOBS
Global functions: mmc
Local  functions: ident, cident, data, zegar, number_of_day,
                   ekran1, ekran2,
                  inpserv, cdekod, dekods
*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
#include "gcc.h"
#include "ees.h"
#include "guc.h"
#include "gue.h"
#include "gic.h"
#include "gfe.h"
#include "mmc.h"
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
static char clearsc [] = { ON, "\033H\033J" } ; /* clear screen */


static char *buf5[LBUC1];
struct bch buc_5 = { EMPTY , LBUC1 ,0 , 0, buf5 };

static char  *kom[] = {                    /*KOMENTARZE DO TESTU */
    "\001\033Y NPOMIAR PRAWIDLOWY",
    "\001\033Y NBrak komutatora"  ,
    "\001\033Y NKomutator zajety",
    "\001\033Y NBlad przelaczenia komutatora",
    "\001\033Y NBrak przetwornika A/C",
    "\001\033Y NBledne przerwanie przetw. A/C",
    "\001\033Y NPakiet przetw. A/C zajety",
    "\001\033Y NNadmiar przetw.A/C",
    "\001\033Y NTimeout przetw. A/C",
    "\001\033Y NPomiar OK, blad rozl. komutatora",
    "\001\033Y NBlad pomiaru i rozl. komutatora",
    "\001\033Y NAwaria kanalu przy wlaczaniu "
};

extern struct rwm ram [ ] ;
char  pend ;
char  pom = ON ;
int  wsk ,pvn ;

static char message [] =  "\000\033Y 'Num. pom.=    ," ;
static char clss [] = "\000\033Y      \033Y  ==>\033K";
static char outpom [] = "\000\033Y 9Pomiar=       mV," ;
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
void  mzvobs()
{
static  char per  ;
char *adr, k ;
SENDM ( &buc_1 ,clearsc );
PAUSE(10);
```

```
    *(outpom + 3 ) = (*(message + 3 ) ) = ' ' ;
    *( clss + 11) = ' ' ;
while(){
for( pvn = 0 ; pvn < 48 ; ++pvn ) {          /*testowanie na 48 we.
                                               komutatora*/
    dec2(pvn , message + 17 );          *dekod. liczby na kod ASCII*/
    SENDM( &buc_1 ,  clss    ) ;
    SENDM( &buc_1 , message ) ;
    WAIT(message, 1) ;                          /*semafor systemowy*/

    wsk = ipac( pvn , &pom, &pend , ZVOBS );  /*wywolanie f.cji
                                                 obslugi komu -
                                       tatora i przetw. A/C z pliku
                                       handlers.c */
    for( k = 13; k < 19 ; ++k )
        *( outpom + k ) = ' ' ;
    for(k=12; k<19; k++)
    *(outpom + k) = ' ';
    cdot( ram[pvn].w, outpom + 16, 0, 5 );
    SENDM(&buc_1 , outpom );
    WAIT ( outpom, 1 ) ;
    *( *( kom + wsk ) + 3 ) = *(message + 3 ) ;
    *( clss + 11) = *(message + 3 ) + 1 ;
    *( clss + 3 ) = *(message + 3 ) ;
    SENDM( &buc_1 , *( kom + wsk ) );  /*wysylanie komunikatu na
                                         ekran*/
    if ( ( *(outpom + 3 ) = ++(*(message + 3 ) ) ) > '7' )
        *(message + 3 ) =  *(outpom + 3 ) = *( clss + 11 ) = ' ' ;
    if(wsk)
        PAUSE( 50 ) ;                 /* zatrzymanie testu na 5sek
                                    gdy niprawidlowe dzial.pakietu*/

   }
    }
    if( !per++)                      /* periodyczne wywolanie testu
                                       co 15sek. */
        PERIOD(ZVOBS , 15);
    STOP ( ) ;
}
/*~~~~~~~~~~~~~koniec zadania testujacego~~~~~~~~~~~~~~~~~~*/
```

Załącznik 2. Tabulogram programu NEWMM.EXE

```
/*
 *csumSKG  22-Feb-1989
 *
 *oblicz dla rekordu INTELowskiego sume kontrolna i umiesc pod
 *wskazanym offsetem.
 */

#include <stdio.h>
#include <string.h>

static unsigned char aux[120];

void control ( buf, s1, s2, s3 )
unsigned char *buf;/* bufor z rekordem*/
int s1; /* od buf[s1] do*/
int s2; /* buf[s2] oblicz sume kontrolna*/
int s3; /* i umiesc w buf[s3]*/
{
    unsigned char sum, *pc;
    unsigned int 1;

    sum = (unsigned char)0;
    for ( pc = buf + s1; pc <= buf + s2; pc += 2 ) {
aux[0] = pc[0];
aux[1] = pc[1];
sscanf ( aux, "%x", &1 );
sum += (unsigned char)1;
    }
    sum = ~sum;
    sum += (unsigned char)1;
    sprintf ( aux, "%02x", (unsigned int)sum );
    strupr ( aux );
    buf[s3] = aux[0]; buf[s3+1] = aux[1];
}
```

```
/*
 *      dirtran                     SKG  26-Feb-1989
 *
 *      wyslij komende do PROWAYa z odbieraniem echa
 *      i odzbierz odpowiedz na komende
 */

#include <stdio.h>
#include <memory.h>
#include <string.h>
#include <dos.h>

extern m86_rec ( char *, int );

static union REGS reg1, reg2;
static char buf[100];

void dirtran ( bbb )
unsigned char *bbb;
{
    unsigned char *pc1;
    int i, k;

    memset ( buf, '\0', 90 );
    reg1.x.dx = 1;
    k = strlen ( bbb );
    i = 0;
    for ( pc1 = bbb, i = 0; i < k; i++, pc1++ ) {
        reg1.h.al = *pc1;
        reg1.h.ah = 1;
        int86 ( 0x14, &reg1, &reg2 );
        reg1.h.ah = 2;
        int86 ( 0x14, &reg1, &reg2 );
        buf[i] = reg2.h.al & 0x7F;
    }
    m86_rec ( buf + i, 256 );
    printf ( "%s", buf );
}
```

```
/*
 *      m86_rec                                         SKG  13-Mar-1989
 *
 *  odbierz zadana liczbe znakow z COM1: lub przestan odbierac po
 *  dwoch timeoutach
 */

#include <stdio.h>
#include <dos.h>

static union REGS reg1, reg2;

int m86_rec ( buf, n )
char *buf;                          /* buffer to store bytes        */
int n;                              /* no of bytes to receive       */
{
    int i, count = 0;

    reg1.x.dx = 1;                          /* port 0               */
    buf[0] = '\0';
    for ( i = 0; i < n; i++ ) {
        reg1.h.ah = 2;                     /* receive code          */
        int86 ( 0x14, &reg1, &reg2 );      /* receive byte          */
        if ( reg2.h.ah & 0x80 ) {
            if ( count == 1 ) {
                count = 0;
                break;
            }
            count++;
        }
        buf[i] = reg2.h.al & 0x7F;
    }
    buf[n] = '\0';
    return ( 0 );
}
```

```
/*
 *  make1                SKG  22-Feb-1989
 *
 *  dla podanego adresu zrob NIESTANDARDOWY rekord rozszerzonego
 *  adresu dla PROWAY'a ZAP
 */

#include <stdio.h>
#include <string.h>

extern void control ( char *, int, int, int );

static char buf[20];
/*       :00xxxx02xx          */
/*       01234567890          */
static char *pattern = ":00000000200";

char *make1 ( pc )
char *pc;
{
    int i, j, k;

    k = strlen ( pc );
    strcpy ( buf, pattern );
    for ( i = k - 1, j = 6;   i >= 0;   i--, j-- )
        buf[j] = pc[i];
    control ( buf, 1, 7, 9 );

    return ( buf );
}
```

```
/*
 *      newmm.c                                  SKG 13-Mar-1989
 *                  dopasowany do PROWAY'a ZAP
 *
 * Program czyta zbior INTELowski i wysyla do PROWAY'a przez COM1:
 */

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>

#define BUF_L 120

extern int errno;
extern int change ( unsigned char * );
extern int newtran ( char * );
extern char *make1 ( char * );
extern char *make2 ( char *, char * );
extern control ( char *, int, int, int );
extern change1 ( unsigned char *, char * );
extern int m86_rec ( char *, int );
extern void setreg ( char *, char * );

int ster = 1;
int first = 1;

static unsigned char buf[BUFSIZ], buf1[BUFSIZ];
static char *not = "\007\n\tTRANSMISSION  NOT  OK\n";
static unsigned char ds[3] = "00";
char comm[120] = "copy ";

main ( ac, av )
int ac;
char *av[];
{
    int i, count, kilo, k;
    char *pc;
    FILE *ptri;

    if ( ac < 3 ) {
        printf ( "Uzycie: mm file cs ip ds\n" );
        return ( 1 );
    }
    if (( ptri = fopen ( av[1], "r+" )) == NULL ) {
       printf ( "File open errno %d", errno );
        return ( 2 );
    }
    for ( i = 1; i < ac; i++ )
        strupr ( av[i] );
    printf ( "\nMM ver 3.0              1987, 1988, 1989,  SKG\n\n" );
    printf ( "\nUwaga: port powinien byc zainicjalizowany 4800,e,7,2\n\n" );
    printf ( "Zbior : %s\n", av[1] );
    printf ( "CS     : %s    ", av[2] );
    if ( ac >= 4 )
```

```
        printf ( "IP      : %s   ", av[3] );
    if ( ac >= 5 )
        printf ( "DS      : %s", av[4] );
    printf ( "\n\nOK ? [Y/N]:" );
    i = getch();
    if ( (char)i == 'Y' || (char)i == 'y' )
        ;
    else
        return ( 2 );
    newtran ( "L" );
    m86_rec ( buf, 12 );
    printf ( "%s\n", buf );
    pc = make1 ( av[2] );                /* segment addr equal to CS       */
    if ( newtran ( pc ) != 0 ) {
        printf ( "After segment addr record\n" );
        printf ( "%s", not );
        return ( 5 );
    }
    printf ( "START\n" );

    count = kilo = 0;
    k = 1;
    errno = 0;
    for ( ;; ) {
        fgets ( buf, BUF_L, ptri );
        if ( feof ( ptri ))
            break;
        if ( ferror ( ptri )) {
            perror ( "fread " );
            return ( 1 );
        }
        if ( first ) {
            first = 0;
            if ( ac == 5 ) {
                strupr ( av[4] );
                change1 ( buf, av[4] );
            }
        }
        if ( ster ) {
            change ( buf );
        }
        if ( newtran ( buf ) != 0 ) {
            k = 0;
            break;
        }
        count++;
        if ( count == 64 ) {
            kilo++;
            count = 0;
            printf ( "%3d\r", kilo );
        }
    }
    printf ( "koniec ladowania\n" );
    m86_rec ( buf, 20 );
    for ( i = 0; i <10000; i++ )
```

```
        ;

/*
    setreg ( "CS", av[2] );
    if ( ac >= 4 )
        setreg ( "IP", av[3] );
    if ( ac >= 5 )
        setreg ( "DS", av[4] );
*/

    return ( 0 );
}

static char aaa[6];

int change1 ( pc, ds )
unsigned char *pc;
char *ds;
{
    int k, i;

    if ( ds != NULL ) {
        strupr ( ds );
        strcpy ( aaa, "0000" );
        k = strlen ( ds );
        for ( i = 0; i < k; i++ )
            aaa[3-i] = ds[k-i-1];
        pc[23] = aaa[2];
        pc[24] = aaa[3];
        pc[25] = aaa[0];
        pc[26] = aaa[1];

        control ( pc, 1, 39, 41);
    }
    return ( 0 );
}


int change ( pc )
unsigned char *pc;
{
    char *bc;

    for ( bc = pc + 9; *bc != '\0'; bc += 2 ) {
        if ( *bc == 'F' && *( bc + 1 ) == 'B' ) {
            *bc = '9';
            *( bc + 1 ) = '0';
            control ( pc, 1, 39, 41 );
            ster = 0;
            break;
        }
    }
    return ( 0 );
}
```

```
/*
 *      newtran                                 SKG  3-Sep-1989
 *
 *      wyslij INTELowski record HEX do PROWAYa
 */

#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <process.h>

extern int m86_rec ( char *, int );

static unsigned char rec[4];
static union REGS reg1, reg2;
static int first = 1;

int newtran ( bbb )
unsigned char *bbb;
{
    unsigned char *pc1;
    int i, k;

    reg1.x.dx = 1;
    k = strlen ( bbb );
    for ( pc1 = bbb, i = 0; i < k; i++, pc1++ ) {
        reg1.h.al = *pc1;
        reg1.h.ah = 1;
        int86 ( 0x14, &reg1, &reg2 );
    }
    return ( 0 );
}
```

```
/*
 *   setreg.c                  SKG  23-Feb-1989
 *
 *  ustaw rejestr ZAPowskiego PROWAYa
 */

#include <stdio.h>
#include <string.h>

extern int dirtran ( char * );

char buf[BUFSIZ];
char aux[20];

void setreg ( reg, val )
char *reg, *val;
{
    aux[0] = 'R';
    aux[1] = '\0';
    strcat ( aux, reg );
    dirtran ( aux );
    strcpy ( aux, val );
    strcat ( aux, "\r" );
    dirtran ( aux );
}
```

Załącznik 3. Tabulogram Instrukcji batch'owej SI.BAT

Instrukcja ta służy do kompilacji systemu SIRTOS i transmisji jego kodu wynikowego do zestawu INTELDIGIT-PRÓWAY.

```
set PATH=c:\;c:\aztec;d:\util;c:\dos;c:\msc5\bin;c:\tc5;
c:\util\pctools;
set CLIB=\aztec\
set INCLUDE=c:\aztec\
make -f nu
\trans\newmm sirtos.hex 800 0 1f
\kermit\mskermit
```