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

OŚRODEK AUTOMATYKI ELEKTRYCZNEJ
ZESPÓŁ BUDOWY ROBOTÓW I SERWOMECHANIZMÓW

074 A

Główny wykonawca dr inż. Marian Wrzesień

Wykonawcy mgr inż. Zbigniew Stańczak

Konsultant

Nr zlecenia
RP-58.2.

Opracowanie i wykonanie urządzeń dla uruchamiania i testowania układów sterowania IRp-6/60
Zadanie 3.1.: Oprogramowanie nadrzędne /sterujące/ wyboru programu testującego. Oprogramowanie użytkowe dla pakietów: MZ-70, MW-31, MA-79, MC-42, P3 panelu programowania. Badania eksploatacyjne oraz modyfikacja modelu użytkowego urządzenia do uruchomienia i testowania pakietów układu sterowania robotów IRp-6/60 przy testowaniu serii próbnej układów sterowania

Zleceniodawca
CPBR-7.1.

Pracę rozpoczęto dnia 1.10.1987 **zakończono dnia** 30.11.1988

Kierownik Zespołu: dr inż. P.Jabłoński

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

Kierownik Ośrodka OAE: dr inż. R.Kontrymowicz

Praca zawiera:

stron	10
rysunków	-
fotografii	-
tabel	-
tablic	-
załączników	1

Nr rejestr. 6189

Rozdzielnik - ilość egz:

Egz. 1	BOINTE
Egz. 2	DW
Egz. 3	OAE
Egz. 4	OAE
Egz. 5	~~ZAP~~
Egz. 6	-

Analiza deskryptorowa URZĄDZENIA AUTOMATYCZNEJ REGULACJI I STEROWANIA: ROBOTY PRZEMYSŁOWE, TESTOWANIE.

Analiza dokumentacyjna Sprawozdanie zawiera opis programów testujących pakietów układu sterowania IRp-6/60

Tytuły poprzednich sprawozdań

Zlecenie UR-01.03.01K:

Etap 1: Studia wstępne oraz założenia techniczno-ekonomiczne /spr. nr rej. 5494/

Etap 2: Opracowanie i uruchomienie systemu programowania testera /spr. nr rej. 5549/

Zlecenie RP-58.2:

Zadanie 1.1: Projekt wstępny testera /spr. nr rej. 5627/

Zadanie 1.2: Wykonanie dokumentacji szkicowej testera /spr. nr rej. 5711/

Zadanie 1.3: Wykonanie, uruchomienie i przebadanie modelu użytkowego testera /spr. nr rej. 5911/

UKD 358.45:62/69].002.1/.2 Robot, przemysłowe
681.3.02 Systemy sterowania

PIAP-252/83-6000

SPIS TRESCI STRONA

1. Wstep.
   ======
   Niniejsze sprawozdanie zawiera opis wynikow pracy stanowiacej kontynu
   acje tematu pt.: "Opracowanie i wykonanie urzadzenia dla uruchamiania
   i testowania ukladow sterowania robotow przemyslowych IRp6/60", w ra-
   mach zlecenia RP-58.2, a w tym:
   - opisy dzialania programow testujacych,
   - uwagi powstale przy uruchamianiu programow testujacych oraz
   - uwagi dotyczace zmian hardware'owych w testerze.
   Komentarzem do nn. sprawozdania jest zbior programow napisanych w jez
yku
   C, ktore po zakonczeniu pracy zostana przekazane na dyskietkach do
   archiwum PIAP.
   Ponadto przedstawiono uwagi dotyczace sposobu kontynuacji zlecenia.

2. Opisy programow testujacych.
   ============================

   2.1. Pakiet MA 70.

   W zestaw programow testujacych pakiet MA-70 wchodza testy umozli-
   wiajace sprawdzanie nastepujacych ukladow pakietu:

   PRZETWORNIK CYFROWO-ANALOGOWY (DAC.C),
DEKODER ADRESOW (DEKMA70.C),
DEKODER ADRESOW WEWNETRZNYCH (DKMA70WE.C),
PAMIEC EPROM (EPROM.C),
UKLAD WYBORU PARAMETROW,
PRZELACZNIK PRZERWAN (MP1MP2.C),
UKLAD KONTROLI POLOZENIA WALU SILNIKA (PWS.C),
PAMIEC RAM (RAM.C),
REJESTR POLOZENIA RZECZYWISTEGO,
   REJESTR ZADANEGO PRZYROSTU,
UKLADY WEJSCIA,
REJESTR STANU PRZETWORNIKA POLOZENIA (WEWY.C).

Ponizej omowiono podstawy dzialania ww. testow.

OPIS DZIALANIA PROGRAMU DAC.C

I. Inicjacja.
   1. Wprowadzenie pakietu MA70 w stan HOLD.
II. Porownywanie napiec wyjsciowych pakietu testowanego i pakietu
   wzorcowego dla kazdej wartosci slowa sterujacego praca DAC.
   1. Napiecie wyjsciowe DAC pakietu testowanego jest stale ustawiane
 na poziomie przewyzszajacym napiecie wyjsciowe DAC pakietu wzor-
 cowego o wartosc odpowiadajaca dwom schodkom napiecia DAC.
      Testuje sie czy napiecie na pakiecie nie jest zbyt niskie,
      przy kazdej wartosci slowa sterujacego DAC.
   2. Napiecie Wyjsciowe DAC pakietu testowanego jest stale ustawiane
 na poziomie nizszym od napiecia wyjsciowego DAC pakietu wzorcowego
 o wartosc odpowiadajaca dwom schodkom napiecia DAC.
      Testuje sie czy napiecie na pakiecie nie jest zbyt wysokie,

przy kazdej wartosci slowa sterujacego DAC.

III. Ocena wynikow testu.

1. Ocena jest przeprowadzana na biezaco; w przypadku niedopuszczalnej wartosci odchylki jest drukowany komunikat o bledzie z zaznaczeniem wartosci slowa sterujacego dla tego stanu.

OPIS DZIALANIA PROGRAMU DEKMA70.C

I. Inicjacja.

1. wprowadzenie pakietu MA-70 w stan.

II. Testowanie sygnalow WE0C, WE1C, WY0C, WY1C.

1. Podawane sa adresy ze zbioru sterowan dla kazdej wartosci czterobitowego slowa ZA=XXXX.(hardware'owe programowanie dekodera adresow).

Wynika to z ogolnej zaleznosci funkcyjnej adresow dekodera pakietu MA70:

adr(WE0C,WY0C) = XX(4*ZA)

adr(WE1C,WY1C) = XX(4*ZA+2)

W czasie testu jest obserwowany sygnal XACK/.

III. Ocena wynikow testu.

1.W przypadku nieuzasadnionego pojawienia sie sygnalu XACK/ lub jego nieuzasadnionym braku, drukowany jest komunikat o bledzie wraz z adresem podanym do pakietu oraz wartoscia stanu ZA i nazwa sygnalu wyjsciowego dekodera.

OPIS DZIALANIA PROGRAMU DKMA70WE.C

I. Inicjacja.

1. Wprowadzenie pakietu MA-70 w stan HOLD.

II. Testowanie sygnalow WY0-WY4 oraz WE0-WE5, CSRAM/ i CSPROM/.

1. Podawane sa kolejne wartosci adresow ze zbioru sterowan oraz jest obserwowany sygnal SREADY/.

III. Ocena wyniku testu.

1. W przypadku nieuzasadnionego pojawienia sie sygnalu SREADY/ lub jego nieuzasadnionego braku drukowany jest komunikat o bledzie wraz z wartoscia wystawionego adresu podanym do pakietu oraz nazwa sygnalu wyjsciowego, przy ktorym wystapil blad.

OPIS DZIALANIA PROGRAMU EPROM.C

I. Inicjacja.

1. Wprowadzenie pakietu MA-70 w stan HOLD.

2. Podanie sygnalu PROMINH/ = 1.

II. Testowanie pamieci EPROM.

1. Przy adresach o wartosciach ze zbioru sterowan dla pamieci EPROM tego pakietu, jest odczytywana suma zawartosci pamieci.

III. Ocena wyniku testu.

1. W przypadku niezgodnosci odczytanej wartosci z wartoscia wzorcowa jest drukowany komunikat o bledzie.

OPIS DZIALANIA PROGRAMU MP1MP2.C

I. Inicjacja.

1. Wprowadzenie pakietu MA70 w stan HOLD.

II. Testowanie UKLADU WYBORU PARAMETROW oraz PRZELACZNIKA PRZERWAN.

1. Podaje sie dyrektywy o wymaganych polozeniach segmentow testowanych przelacznikow.
2. Sprawdza sie rzeczywiste stany segmentow tych przelacznikow.

III. Ocena wynikow testowania.

1. W przypadku wystapienia niezgodnosci stanu polozenia segmentow przelacznikow z oczekiwanym jest drukowany komunikat o bledzie. W tym przypadku jest podawana wartosc odczytana polozenia segmentow kazdego z przelacznikow.

OPIS DZIALANIA PROGRAMU PWS.C

I. Inicjacja.
   1. Wprowadzenie pakietu MA70 w stan HOLD.
   2. Uruchomienie generowania napiec SIN i COS zasilajacych resolwer na pakiecie MZ-70 stanowiacym wyposazenie testera.

II. Testowanie poprawnosci dzialania ukladu kontroli polozenia walu silnika.
   1. Uruchomienie silnika zgodnie z ruchem wskazowek zegara.
   2. Dynamiczne odczytywanie zawartosci rejestrow i porownywanie ich z wartosciami wpisanymi poprzednio.
   3. Sprawdzanie wystapienia sygnalu INT oraz jego zaniku w rejestrze.
   4. Powtorzenie powyzszych dzialan przy silniku obracajacym sie przeciwnie do ruchu wskazowek zegara.

III. Ocena wynikow testu.
   1. W przypadku gdy stan sygnalu INT jest niezgodny z wartoscia oczekiwana jest drukowany komunikat o wystapieniu bledu.
   2. W przypadku gdy przyrost odczytywanych wartosci z rejestru jest rozny od 1 i rozny od -1 i rozny od 1023 i rozny od -1023, jest drukowany komunikat o bledzie z podaniem wartosci przyrostu oraz wartosci, ktora powinna zostac odczytana.

OPIS DZIALANIA PROGRAMU RAM.C

I. Inicjacja.
   1. Wprowadzenie pakietu MA70 w stan HOLD.

II. Testowanie pamieci RAM.
   1. Dla kazdej wartosci adresu ze zbioru sterowan pamieci RAM pakietu sa zapisywane, a nastepnie odczytywane nastepujace wartosci danych: 0x00, 0xff, 0x55, 0xaa.

III. Ocena wynikow testu.
   1. W przypadku blednego wyniku odczytu jest drukowany komunikat o bledzie wraz z wartosciami:
   adresu pamieci RAM,
   danej wpisywanej,
   danej odczytanej.

OPIS DZIALANIA PROGRAMU WEWY.C

I. Inicjacja.
   1. Wprowadzenie pakietu MA70 w stan HOLD.

II. Testowanie REJESTRU POLOZENIA RZECZYWISTEGO.

1. Do rejestru sa wpisywane kolejno:
   pelzajaca jedynka oraz pelzajace zero.
2. Jest odczytywana zawartosc rejestru i porownywana z wartoscia oczekiwana.

III. Testowanie REJESTRU ZADANEGO PRZYROSTU.

1. Do rejestru sa wpisywane kolejno:

pelzajaca jedynka oraz pelzajace zero.
2. Jest odczytywana zawartosc rejestru i porownywana z wartoscia oczekiwana.
IV. Testowanie UKLADOW WEJSCIA.
1. Do ukladow sa podawane kombinacje sygnalow wejsciowych: ZAK.ZAPIS, REDPRED, SEARCH, ZEZWOL, SYNCHR, WRITTEN, SYNCSW, CZUJNIK, OSZSYNCHR, przy czym uklady wejscia testuje sie przy zamontowanym, a nastepnie wymontowanym obwodzie D6 (74174).
2. Jest odczytywany stan zawartosci ukladow wejscia i porownywany z wartoscia oczekiwana.
V. Testowanie REJESTRU STANU STEROWNIKA POLOZENIA.
1. Do rejestru sa wpisywane kolejno: pelzajaca jedynka i pelzajace zero.
2. Jest odczytywana zawartosc rejestru i porownywana z wartoscia oczekiwana.
3. Test przeprowadza sie przy zamontowanym, a nastepnie wymontowanym obwodzie D5(74125) pakietu.
VI. Ocena wynikow testu.
1. W kazdym z powyzszych przypadkow, w razie wystapienia niezgodnosci pomiedzy wartosciami zapisywanymi i odczytywanymi jest drukowany komunikat o bledzie ze wskazaniem wpisywanej i odczytywanej danej oraz nazwa testowanego obwodu.

2.2 Pakiet MW-31.

W zestaw programow testuja cych pakiet MW-31 wchodza testy umozliwiajace sprawdzanie nastepujacych ukladow pakietu:

BLOK KONTROLI MAGISTRALI (BKM.C),
DEKODER ADRESOW,
UKLAD GENEROWANIA SYGNALU MMAP2 (DEKMW31.C),
UKLAD KONTROLI ZASILAN (UKZ.C),
UKLAD PRZERWAN I ALARMOW,
UKLAD KONTROLI PRZERWAN SYSTEMOWYCH (UP.C).

Ponizej omowione podstawy dzialania ww. testow.

OPIS DZIALANIA PROGRAMU BKM.C

I. Testowanie bloku kontroli przesylu po magistrali systemowej na pakiecie MW31.
1. Sprawzenie prawidlowego polczenia zwory D2, umozliwiajacego jednoznaczne adresowanie pakietu.
2. Ustawienie sygnalu INIT=1.
3. Podawanie w odpowiedniej kolejnosci sygnalow sterujacych MRDC/,IORC/,MWTC/,IOWC/ na przerzutniki D9,D8 oraz C1(reset zespolu).
Dane systemowe ffff hex.
4. Odczyt z wyjscia przerzutnika D8.
II. Ocena wynikow testu.
1. Ocena jest przeprowadzana na podstawie zgodnosci slowa stanu pakietu z wartoscia oczekiwana.

OPIS DZIALANIA PROGRAMU DEKMW31.C

I. Wstepne sprawdzenie stanu sygnalu XACK/ .
II. Test pozytywny dekodera adresow pakietu
1. Podawane sa adresy, przy ktorych niezaleznie od stanu zwory D2, pakiet powinien odezwac sie sygnalem XACK/ .
III. Test dekodera adresow (przy 4 kombinacjach stanu zwory D2), w calej przestrzeni 1 Mb .[opcjonalnie]
IV. Ocena wynikow testu.
1.W przypadku pojawienia sie sygnalu XACK/ drukowany jest adres przy jakim to nastapilo .
Jesli dekoder odezwal sie niprawidlowo dodatkowo drukowany jest odpowiedni komunikat.

OPIS DZIALANIA PROGRAMU UKZ.C

I. Funkcja reseton.
1. Sprawdzenie sygnalu RESET/ przed podaniem P_RESET,
2. Sprawdzenie sygnalu RESET/ po podaniu P_RESET,
3. Sprawdzenie sygnalu OUTON po podaniu sygnalu P_RESET.
II. Funkcja detuokvb.
1. Podanie LOCK/=0,
2. Podanie P-RESET/=1,
3. Sprawdzenie Uok5VB w stanie normalnym,
4. Sprawdzenie Uok5VB przy spadku napiecia 5VB,
5. Sprawdzenie detektora Uok5VB po resecie CZYT_A.
III. Funkcja detuok.
1. Sprawdzenie detektora w stanie normalnym,
2. Sprawdzenie detektora przy kolejnych spadkach kontrolowanych napiec,
3. Sprawdzenie detektora po powrocie napiec do wartosci poczatkowych.
IV. Funkcja detcns.
1. Sprawdzenie sygnalow w warunkach normalnych (po resetach), testowane sygnaly: PFIN/, PFSN/, MPRO/,
2. Sprawdzenie sygnalow po zaniku napiecia CNS,
3. Sprawdzenie sygnalow po powrocie napiecia CNS,
4. Sprawdzenie sygnalu MPRO/ przy zaniku napiecia Uok.
V. Ocena wynikow testowania.
1. W przypadku niezgodnosci wartosci sygnalu badanego z wartoscia oczekiwana, jest drukowany komunikat.

OPIS DZIALANIA PROGRAMU UP.C

I.Inicjacja.
1. Ustawienie sygnalu INIT/ = 1,
2. Zapalenie wszystkich diod elektroluminescencyjnych,
II.Funkcja up.
1. Gaszenie kolejno kazdej z diod, (ocena wyniku badania na podstawie obserwacji),
2. Przy wskazanym ustawieniu przelacznikow C10 i E14 gaszone sa kolejne diody. Podczas tej operacji sprawdzane sa wyjscia:
a. alarm (sygnal wyjsciowy ALARM),
b. przerwanie (odczyt w slowie stanu),
c. sygnal stanu (odczyt przez bramy WY pakietu),
II.Fynkcja int.
1. podanie sygnalu INIT/ =1,
2. Sprawdzenie sygnalow przerwan systemowych INT2, INT3, INT4,
a. w stanie normalnym,

b. po podaniu odpowiednio sygnalow UST1, UST2, Przerw.1,
3. Sprawdzenie sygnalow przerwan po podaniu sygnalow ZER1 i ZER2.
III. Ocena wynikow testowania.
1. Komentarze drukowane w czasie testu, w przypadku wystapienia niezgodnosci pomiedzy wartosciami odczytywanymi i oczekiwanymi.

2.3. Pakiet MZ-70.

Program testujacy pakiet MZ-70 umozliwia sprawdzanie nastepujacych blokow pakietu:

BLOK CYFROWY,
BLOK ANALOGOWY (MZ70.C).

Ponizej omowiono podstawy dzialania ww. testu.

OPIS DZIALANIA PROGRAMU MZ-70.C

I. Blok cyfrowy.
1. Sprawdzenie dzialania przyciskow sterujacych poprzez odczyt wartosci max i min osmiobitowych slow wyjsciowych bloku.
2. Sprawdzenie dzialania bloku poprzez porownanie wszystkich mozliwych slow osmiobitowych z takimi samymi slowami generowanymi przez pakiet wzorcowy przy synchronicznym sterowaniu obu pakietow.

II. Blok analogowy.
1. Porownanie ze soba lub z napieciem wzorcowym amplitud sygnalu SIN i COS oraz skladowych stalych tych sygnalow pakietu testowanego nastepujaco:
SIN+ i SIN-,
SIN+ i COS+,
SIN+ i COS-,
COS+ i COS-,
COS+ i SIN-,
COS- i SIN-,
SIN+,
SIN-,
COS+,
COS-,
COS~,
SIN~.

III. Ocena wynikow testowania.
1.W przypadku wykrycia niezgodnosci pomiedzy sygnalami wyjsciowymi a wzorcowymi jest drukowany komunikat o bledzie z wyszczegolnieniem rodzaju bledu.

2.4. Pakiet MC-42.

W zestaw programow testujacych pakiet MC-42 wchodza testy umozliwiajace sprawdzanie nastepujacych ukladow pakietu:

UKLAD ODCZYTU PAKIETU (ODCZYT.C),
UKLAD BLOKADY SYGNALEM OUTON/ (OUTON.C),
UKLAD KONTROLI PRZECIAZENIA (PRZEC.C).
UKLAD ZAPISU PAKIETU (ZAPIS.C),

OPIS DZIALANIA PROGRAMU ODCZYT.C

I. Inicjacja.
1.Ustawienie INIT/=1.
2.Ustawienie OUTON/=1.
II. Sprawdzenie ustawienia przelacznika AP
1. Podawane sa adresy przy ktorych, w skrajnych polozeniach przelacznika AP pakiet powinien odezwac sie sygnalem XACK/ .
III: Test poprawnosci odczytu pakietem MC42.
1. Podawanie na wejscia obiektowe pakietu testowanego pelzajacej jedynki.
2. Odczyt laczem systemowym wartosci zarejestrowanej przez pakiet testowany
wartosc zanegowana (negacja na pakiecie);
3. Porownanie wartosci odczytanej z zapisana.
IV. Ocena wynikow testu.
1.W przypadku pojawienia sie niezgodnosci przedstawiana jest binarna wielkosc slowa zapisywanego i odczytywanego z pakietu.

OPIS DZIALANIA PROGRAMU OUTON.C

I. Inicjacja.
1.Ustawienie INIT/=1.
II. Sprawdzenie ustawienia przelacznika AP
1. Podawane sa adresy przy ktorych, w skrajnych polozeniach przelacznika AP pakiet powinien odezwac sie sygnalem XACK/ .
III. Test poprawnosci blokowania sygnalem OUTON/.
1. Ustawienie sygnalu OUTON/ zezwalajaco.
2. Zapis do pakietu testowanego wartosci 0000 hex, wartosc zanegowana (negacja na pakiecie);
2. Odczyt z laczy obiektowych wartosci sygnalow.
3. Porownanie wartosci odczytanej z wartoscia zerowa;
4. Ustawienie sygnalu OUTON/ zabraniajaco.
5. Zapis do pakietu testowanego wartosci 0000 hex, wartosc zanegowana (negacja na pakiecie);
6. Porownanie wartosci odczytanej z wartoscia zerowa;
IV. Ocena wynikow testu.
1.W przypadku pojawienia sie niezgodnosci w pierwszym przypadku i zgdnosci w drugim, drukowana jest binarna wielkosc slowa zapisywanego i odczytywanego z pakietu.

OPIS DZIALANIA PROGRAMU PRZEC.C

I. Inicjacja.
1.Ustawienie INIT/=1.
II. Sprawdzenie ustawienia przelacznika AP
1. Podawane sa adresy przy ktorych, w skrajnych polozeniach przelacznika AP pakiet powinien odezwac sie sygnalem XACK/ .
III. Test poprawnosci zapisu do pakieu .
IV. Test poprawnosci zapisu do pakietu przy obciazeniu wyjsc pakietu ponizej granicy przeciazenia.
V. Test poprawnosci zapisu do pakietu przy obciazeniu wyjsc pakietu

powyzej granicy przeciazenia.
VI.W przypadku bledu udostepniona jest mozliwosc regulacji detektora przeciazenia
IV. Ocena wynikow testowania.
1.W przypadku pojawienia sie niezgodnosci przedstawiana jest binarna wartosc slowa zapisywanego i odczytywanego z pakietu.

OPIS DZIALANIA PROGRAMU ZAPIS.C
I. Inicjacja INIT/=1.
II. Sprawdzenie ustawienia przelacznika AP
1. Podawane sa adresy przy ktorych, w skrajnych polozeniach przelacznika AP pakiet powinien odezwac sie sygnalem XACK/ .
III. Test poprawnosci zapisu do pakietu MC42.
1. Podawanie na wejscia systemowe pakietu testowanego pelzajacej jedynki.
wartosc zanegowana (negacja na pakiecie);
2. Odczyt laczem obiektowym stanu wyjsc pakietu testowanego.
3. Porownanie wartosci odczytanej z zapisana.
IV. Ocena wynikow testu.
1.W przypadku pojawienia sie niezgodnosci przedstawiana jest binarna wartosc slowa zapisywanego i odczytywanego z pakietu.

2.5.Plytka P3 parelu programowania.

I. Testowanie plytki.
1. Wysterowuje sie kolejne wiersze, a nastepnie kolumny wyswietlaczy i obserwuje sie swiecenie diod elektroluminescencyjnych.
2. J.w. lecz wysterowywanie jest impulsowe.
II. Ocena wynikow testowania.
1. Oceny sprawnosci diod elektroluminescencyjnych dokonuje nadzorujac y przebieg testu na podstawie obserwacji.

3. Badania testera przy uruchamianiu programow testujacych.

W czasie prac nad uruchomieniem programow testujacych zwrocono uwage m.in. na:
i) zgodnosc adresow fizycznych testera i pakietow, dla ktorych jest przeznaczony tester z adresami i sposobem adresacji odpowiednich pakietow, ktore wprowadzono do programow testujacych.
ii) sygnaly sterujace, inicjujace prace zespolu TESTER-PAKIET KONTROLOWANY,
iii)prawidlowe dolaczenie laczowek zlaczy obiektowych,
iv) koniecznosc zachowania stanu beznapieciowego testera podczas dolaczania zlaczy obiektowych testera oraz podczas umieszczania pakietu testowanego w zlaczach systemowych testera.
v) przeanalizowano wplyw uszkodzen pakietu testowanego na sprawnosc testera.
vi) zbadano skutecznosc kontrolowania niektorych sygnalow generowanych przez pakiety testowane (XACK/, SREADY/, OUTON/)
W wyniku badan testera (w czasie weryfikacji oprogramowania) stwierdzono konieczność zmodyfikowania niektorych obwodow testera. Ponizej opisano

dzialania usprawniajace dzialanie testera.

3.1. Zmiany hardware'owe przeprowadzone w wyniku badan testera.

3.1.1 Obwod dla kontroli sygnalu XACK.

Podczas prac nad uruchomieniem programow testujacych stwierdzono, ze dotychczasowa metoda badania sygnalu potwierdzenia XACK/ przy pomocy pakietu MM-86 stanowiacego wyposazenie testera nie zdaje egzaminu ze wzgledu na trudnosci w odczytaniu slowa stanu z tego pakietu. Z tego powodu zbudowano w testerze oddzielny obwod, ktory dziala jak zatrzask i po wyzerowaniu rejestruje wystapienie sygnalu XACK/. Odczyt stanu tego obwodu zapewnia interfejs testera.

3.1.2 Obwod dla kontroli sygnalu SREADY/.

Podobnie jak w przypadku sygnalu XACK, zbudowano zatrzask umozliwiajacy kontrole wystepowania sygnalu SREADY/ podczas testowania pakietu MA-70.

3.1.3 Obwod dla generowania sygnalu OUTON/.

Ze wzgledu na koniecznosc podawania sygnalu OUTON/ na pakiet MC-42 w czasie testu, zbudowano obwod sterowany programowo - umozliwiajacy generowanie tego sygnalu.

3.1.4. Okablowanie testera.

W czasie prac nad uruchomieniem programow testujacych stwierdzono, ze prowadzenie kabli laczacych magistrale wewnetrzna pakietu MA-70 z interfejsem testera nie jest wlasciwe. Miedzy innymi poprawiono prowadzenie przewodow zerowych oraz wprowadzono kondensatory filtrujace, dzieki ktorym wyeliminowano zaklocenia pojawiajace sie dotychczas w testerze.

3.1.5. Polaczenie zlacz systemowych testera z pakietem testowanym.

Badania testera podczas testowania pakietow wykazaly, ze istnieja niezwykle niebezpieczne usterki w pakietach testowanych, mogace spowodo
o
wac zniszczenia w testerze. Z tego powodu, dc obwodow testera laczacyc
h
sie bezposrednio z ukladami na pakietach kontrolowanych wprowadzono bufory, ktorych odpornosc na zwarcie chroni tester przed uszkodzeniem.

4. Uwagi dotyczace sposobu kontynuacji zlecenia RP-58.2.

Do zakonczenia prac nad testerem nalezy opracowac i wykonac:

1. Programy testujace pakiet MW-32,
2. Programy testujace pakiety P1 i P2 panelu programowania,
3. Wprowadzenie programow uzytkowych do pamieci EPROM testera,
4. Przeprowadzic konserwacje testow,
5. Dokumentacje techniczna testera,

6. Dokumentacje techniczno-ruchowa testera.
Ponadto, przy opracowywaniu oprogramowania dla testowania pakietu MW-32 oraz panelu programowania przewiduje sie dalsza modyfikacje, ktorej celem jest usprawnienie testera.

Wszelkie dotychczasowe modyfikacje sa wprowadzane na biezaco do dokumentacji technicznej, ktorej zakonczenie przypada - wraz z zakonczeniem calej pracy - na 1988-03-31.

```
/*1988-12-02*/
                        /* INT.C */
          /* TESTOWANIE OBWODU PRZERWAN SYSTEMOWYCH */
          /* opracowal dr inz. Marian Wrzesien dnia 1988.11.7 */
# include <fun.h>
# include <adr.h>
# include <adrmw31.h>
main(){intmw31();}                                        /******/
intmw31()
  {
   int od,i,k=0;
   unsigned long a,b,c,d,e,f,g,h;
   scr_clr();
   printf("ZESPOL PRZERWAN SYSTEMOWYCH INT\n");
   printf("Ustaw  I segment przelacznika C10 w pozycji gornej.\n");
   printf("Ustaw  segmenty IV,V,VI przelacznika E4 w pozycji gornej.\n");
   cr();
   scr_clr();
   printf("ZESPOL PRZERWAN SYSTEMOWYCH INT");
   outportb(IT1B1,0x9b);    /*inicjacja B1 IT1, PA,PB,PC - in */
   wyl_al;                  /* ustawienie wyl_al na 0 */
   notwyl_al;                       /* ustawienie wyl_al na 1 */
   INIT;                    /* reset kolejno trzech ukladow */
   NOTINIT;
   CZYT_S(RWADR11Q7);                       /* reset D13 */
   CZYT_S(RWADR10Q7);
   CZYT_S(RWADR01Q7);
   CZYT_S(RWADR00Q7);
   delayp(1000);
   odczyt;                  /* brama IT1B1PC */
   if(!INT2)
     k=1,printf("\nNieuzasadnione generowanie sygnalu INT2 przy sterowaniu
INIT/\n");
   if('INT3)
     k=1,printf("\nNieuzasadnione generowanie sygnalu INT3 przy sterowaniu
INIT/\n");
   if('INT4)
     k=1,printf("\nNieuzasadnione generowanie sygnalu INT4 przy sterowaniu
CZYT.S/\n");
   INIT;
   NOTINIT;
   UST1(a=RWADR11Q6);
   UST1(b=RWADR10Q6);
   UST1(c=RWADR01Q6);
   UST1(d=RWADR00Q6);
   UST2(e=RWADR11Q5);
   UST2(f=RWADR10Q5);
   UST2(g=RWADR01Q5);
   UST2(h=RWADR00Q5);
   prz_1;
   delayp(1000);
   endprz_1;
   delayp(1000);
   odczyt;
   if(INT2)
```

```
    k=1,printf("\nBrak sygnalu INT2 przy sterowaniu UST1/\n");
  if(INT3)
    k=1,printf("\nBrak sygnalu INT3 przy sterowaniu UST2/\n");
  if(INT4)
    k=1,printf("\nBrak sygnalu INT4 przy sterowaniu Przerw.1\n");
  ZER1(a);
  ZER1(b);
  ZER1(c);
  ZER1(d);
  ZER2(e);
  ZER2(f);
  ZER2(g);
  ZER2(h);
  odczyt;
  if(!INT2)
    k=1,printf("\nINT2 nie resetuje sie sygnalem ZER1/\n");
  if(!INT3)
    k=1,printf("\nINT3 nie resetuje sie sygnalem ZER2/\n");
  if(k)
    printf("Wykonana zalecona instrukcja ?\n");
  else
    printf("\t\t\t\t\t\t\t\t\tOK\n");
  getch();
  return 0;
}
                    /*koniec*/
```

```
/*1988-12-01*/
                    /* MMAP2.C */
      /* TESTOWANIE UKLADU GENERUJACEGO MMAP2 NA PAKIECIE MW31 */
   /* opracowal dr inz. Marian Wrzesien dnia 1988.09.10 */
# define ZER      0x0
# define NOTZER   0x40
# include <adr.h>
# include <fun.h>
main(){mmap2();}                                        /******/
mmap2()
  {
    unsigned int k;
    scr_clr();
    printf("\nSYGNAL MMAP2");
    outportb(IT1B3,0x8b);  /* zaprogramowanie bramy IT1B3 */
    outportb(IT1D6WY,ZER); /* reset C1 IT1 */
    outportb(IT1D6WY,NOTZER);       /* wycofanie resetu */
    if(map2)
      {
        printf("\nUsterka w zespole odczytujacym testera!");
    blad();
        return(-1);
       }
    mreadw(0x0e8ee);
    mreadw(0x0ebee);
    mreadw(0x0e9ee);
    mreadw(0x0eaee);
    if(!map2)
      {
       printf("\nPakiet MW31 nie wytwarza sygnalu MMAP2/\n");
   blad();
        return(-1);
       }
     else
      {
       printf("\t\t\t\t\t\t\t\tOK!\n");
   cr();
        return(0);
       }
  }
                    /*koniec*/
```

```
/*1988-12-01*/
                    /*      UKZ.C   */
   /*TESTOWANIE UKLADU KONTROLI ZASILAN PAKIETU MW-31 */
   /*opacowal dr inz Marian Wrzesien dnia 1988-09-21  */
# define p_reset 0xff0f                                  /* maska P_RESET/ */
# define lock 0xff1d                                     /* maska LOCK/ */
# define uol:mc42 0x100                                  /* maska na odczyt w MC42 */
# define utrprmc42 0x1                                   /*      j.w.              */
# include <fun.h>
# include <adr.h>
# include <adrmw31.h>
main(){ukz();}                                           /******/
ukz()
 {
  int k,p,m;
  k:=p=m=0;
  scr_clr();
  printf("\n\t\tUKLAD KONTROLI ZASILAN\n");
  outportb(IT1B1,0x9b);                                  /* zaprogramowanie B1 w it1 *
/
    reseton()?DR():printf("\t\t\t\tOK!\n");
    if(k)
    {
      m=1;
      goto koniec;
     }
    detuokvb()?DR():printf("\t\t\t\tOK!\n");
    if(k)
      {
       m=1;
       goto koniec;
       }
    detuok()?DR():printf("\t\tOK'\n");
    if(k)
      {
        m=1;
   goto koniec;
       }
    (detcns())?DR():printf("\t\t\t\tOK!\n");
     if(!(k!m))
       {
         delayp(30000);
    scr_clr();
    printf("UKLAD KONTROLI ZASILAN\t\t\t\t\t\t\tOK!\n");
    cr();
    return(0);
       }
  koniec:
  blad();
  return(-1);
 }

 static int v[]=
    {
/*      [0]   [1]   [2]  [3]     [4]   [5]   [6]  [7]                    */
/*     +12d, +15d, +5d,  -5d,   -15d, zero, 5vbd,cnsd                    */
```

17

```
      0x81f,0xa1f,0xe1f,0x101f,0x501f,0x1f,0xc1f,0x21f,
      0x91f,0xb1f,0xf1f,0x301f,0x701f,     0xd1f,0x31f,

/*      +12m, +15m, +5m,   -5m,    -15m,        5vbm,cnsm                    */
    };

  reseton()
   {
     int k,m,n;
     k=m=n=0;
     printf("\n\nObwod RESET/ - OUTON/");
     wy2(v[5]);
     delayp(100);
     reset?(k=1,printf("\nNieuzasadnione generowanie sygnalu RESET/\n")):(k
=0);
     wy2(v[5]&p_reset);
     delayp(100);
     reset?(m=0):(k=1,printf("\nSygnal P_RESET nie wytwarza sygnalu RESET\n
"),(m=1));
     outon?(n=1,printf("Sygnal P_RESET nie wytwarza sygnalu OUTON\n")):(n=0
);
     if(k|m|n) return(1);
     return(0);
   }

  detuok()
   {
    int i=0,k=0,l;
    printf("Detektor napiec +5V,-5V,+12V,+15V,-15V");
    wy2(v[5]),delayp(100); /* sprawdzenie det.UOK w stanie normalnym */
    if(in3)
     {
      printf("\n\nUstaw prawidlowe wartosci napiec potencjometrami P3,...,P
7\n");
      printf("Jesli wartosci ww. napiec prawidlowe - usterka w obwodach UOK
 lub UOKVB!\n");
      return(-1);
     }
    while(i<5)
     {
       wy2(v[i]);
       delayp(100);
       if(!in3)                              /* detektor nie wskazuje spadku V */
    {
      printf("\n\nDetektor nie wskazuje spadku napiecia w torze:A%d -UOK\n",i
+2);
      printf("lub nieprawidlowa wartosc napiecia potencjometru P%d\n",i+2);
      k=1;
    }
       wy2(v[5]);
       delayp(100);                    /* sprowadzenie det. UOK do stanu norm.*/
       i++;
     }
    return(k?-1:0);
```

```
    }

int  detuokvb()
    {
      unsigned long a,b,c,d;
      int k=0;
      printf("Detektor napiecia 5VB");
      wy2(0x1d);                                /* lock=0,preset=1....resp=1 */
      CZYT_A(a=RWADR11Q0);                      /* reset D17 */
      CZYT_A(b=RWADR10Q0);
      CZYT_A(c=RWADR01Q0);
      CZYT_A(d=RWADR00Q0);
      delayp(100);
      if(!in4)                          /* sprawdzenie det.UOK5vb w stanie normalnym
*/
       {
        printf("\n\nUstaw prawidlowa wartosc napiecia potencjometrem P1\n");
        printf("Jesli wartosc ww. napiecia prawidlowa - usterka w obwodach U
OK5VB:\n");
        printf("CZYT.A-L.UTR.PR.  lub  LOCK/-RESP-L.UTR.PR,\n");
        printf("lub brak zwory laczacej +5V z +5VB.\n");
        printf("(Mozliwy brak sygnalu CZYT_A)\n");
        return(-1);
       }

      wy2(v[6]&lock);                   /*spr,det, przy spadku napiecia */
      delayp(100);
      if(in4)
       {
        printf("\n\nDetektor nie wskazuje spadku napiecia\n");
        printf("Uszkodzone obwody toru: A1 - L.UTR.PR\n");
        printf("lub nieprawidlowa wartosc napiecia potencjometru P1\n");
        return(-1);
       }

      wy2(v[5]&lock),delayp(100);           /*sprawdz. det. po resecie (CZYT.A) *
/
      CZYT_A(a);
      CZYT_A(b);
      CZYT_A(c);
      CZYT_A(d);
      delayp(100);          /* reset obwodu D17 */
      if(!in4)
       {
        printf("\n\nObwod D17 nie resetuje sie\n");
        printf("Uszkodzone obwody toru: CZYT.A-L.UTR.PR  lub  LOCK/-RESP-L.U
TR.PR.\n");
        return(-1);
       }
      else
      return(0);
    }

 detcns()
  {
   int k,m,n;
   k=m=n=0;
```

```
   printf("Detektor napiecia CNS");/* na poczatku testu: warunki normalne *
/

   /*sprawdzenie sygnalow w warunkach normalnych i po resetach*/
   wy2(v[5]&p_reset);
   delayp(1000);
   pfin?(k=1,printf("\n\nNieuzasadnione generowanie sygnalu PFIN/\n")):(k=0
);
   pfsn?(wy2(v[5]&lock)):1;                          /* proba ustawienia PFSN/ */
   pfsn?(m=1,printf("\n\nSygnal PFSN/ nie da sie ustawic sygnalem LOCK/\n")
):(m=0);
   mpro?(wy2(v[5]&p_reset),delayp(1000)):1;          /* proba ustawienia MPRO/ */
   mpro?(n=1,printf("\n\nSygnal MPRO/ nie da sie ustawic sygnalem P_RESET/\
n")):(n=0);
   if(k|m|n) return(1);

            /*sprawdzenie sygnalow przy zaniku CNS*/
   wy2(v[7]&lock);
   delayp(1000);
   pfin?k=0:(k=1,printf("\n\nNie jest generowany PFIN/ przy spadku napiecia
\n"));
   pfsn?m=0:(m=1,printf("\n\nNie jest generowany PFSN/ przy spadku napiecia
\n")) ;
   mpro?n=0:(n=1,printf("\n\nNie jest generowany MPRO/ przy spadku napiecia
\n"));
   if(k|m|n) printf("Mozliwe zle ustawienie potencjometru P2\n");
   if(k|m|n) return(1);
            /*sprawdzenie sygnalow po powrocie napiecia*/
   wy2(v[5]&p_reset);                                /* lock=1 oraz p_reset=0*/
   delayp(1000);
   wy2(v[5]&lock);                          /* warunki normalne */
   pfsn?(k=1,printf("\n\nSygnal PFSN nie zanika po powrocie napiecia(przy L
OCK/=1\n")):(k=0);
   mpro?(m=1,printf("\n\nSygnal MPRO/ nie zanika po powrocie napiecia CNS \
n")):(m=0);
   if(k|m|n) return(1);

            /*sprawdzenie sygnalow przy zaniku UOK */
   delayp(1000);
   wy2(v[0]&lock);delayp(1000);                      /* spadek napiecia w det.UOK
*/
   mpro?k=0:(k=1,printf("\n\nNie jest generowany sygnal MPRO/ przy spadku n
apiecia w UOK\n"));
   wy2(v[5]&p_reset);                                /* reset B2(B) */
   delayp(1000);
   mpro?(m=1,printf("\n\nSygnal MPRO/ nie zanika po powrocie napiecia w UOK
\
                     i resecie sygnalem P_RESET\n")):(m=0);
   return(k|m);
  }
                  /*koniec*/
```

```
/*1988-12-02*/
                    /*    UP.C    */
                /* TESTOWANIE UKLADU PRZERWAN PAKIETU MW-31 */
   /* opracowal dr inz Marian Wrzesien dnia 1988-09-21 */
# define notinit 0x20
# define D0 0x0
# define D10 0x80
# define D11 0x8000
# define D12 0x40
# define D13 0x8
# define NOTD13 0x17
# define D14 0x4000
# define D15 0x2
# define WYL_AL 0x13
# include <adrmw31.h>
# include <fun.h>
# include <adr.h>
main(){up();}                                          /******/
  up()
 {
  int i,j,te,p,r,s,t,w,y;
  int a[8];
  char x;
  unsigned int dattest,b,odczyt1,odczyt2,odczyt3,maskamc42=0x200,dr,c;
  static char *kom[7]=
   {
     "przerwania 3",
     "przerwanie 2",
     "przerwania 1",
     "WYL.ZEG.",
     "ZAL.ZL",
     "GND",
     "Kout * WYL.B.",
   };
  static char *text[5]=
   {
     "przesuniecie segmentow przelacznikow C10 i E14",
     "ZESPOL WSKAZNIKOW PRZERWAN I ALARMOW",
     "Mozliwe zwarcie kilku (lub rozwarcie wszystkich) segmentow przelacznika",
   };
  r=s=t=w=y=p=0;
  scr_clr();
  outportb(IT1D6WY,notinit);                  /* sygnal INIT/=1 */
  wy1(D0);
  delay();
  wy2(NOTD13);
  delay();
  printf("\n\t\t%s\n\n",text[1]);
  printf("Ustaw segmenty przelacznika D2 w pozycji dolnej.\n");
  printf("Po wcisnieciu * CR * sprawdzaj gasniecie kolejnych diod D10-D16.\
n");
  printf("Pozytywny wynik proby jest warunkiem koniecznym dalszych prac");
  cr();
  te=10000;
```

```
wy1(D10),delayp(te);
wy1(D11),delayp(te);
wy1(D12),delayp(te);
wy1(D0),wy2(D13),delayp(te);
wy1(D14),wy2(NOTD13),delayp(te);
wy3(D15),wy1(D0),delayp(te);
wy4(D0),wy3(D0),delayp(100);
printf("Czy kontynuujesz test ? (t/n)\n");
if(getch()!='t')
  return(0);
scr_clr();
i=0;
while(i<7)                          /* 8 miejsc w liczbie datzad */
 {
  scr_clr();
  printf("\t\t%s\n\n",text[1]);
  printf("%s do dolu = rozwarcie (0)\n",text[0]);
  printf("%s do gory = zwarcie   (1)\n",text[0]);
  printf("%s dowolne =           (X)\n\n",text[0]);
  printf("\t\t\t Nr. C10 i E14:  1  2  3  4  5  6  7  8\n");
  printf("Ustaw przelaczniki C10 i E14 w pozycji:");
  dattest=pelz1(i);
  j=0;
  while(j<7)
   {
     a[j]=(dattest&pelz1(j))>>j;
     printf("  %d",a[j]);
     j++;
   }
  printf("  X\n\n\t\t\t\t\t\t\t* CR *\n");      /* !! to ma byc !! */
  getch();
  j=1;
  while(j<8)
   {
     wy2(WYL_AL);                                   /* reset D13(6) */
     wy2(NOTD13);                                   /* gasniecie d13 */
     switch(j)
      {
        case 1:                     /* przerwanie 1 !stara wersja 0x2000*/
         wy1(D10);delay();c=czyt;wy1(D0);b=0x2;dr=0x4000;
      break;
     case 2:                        /* przerwanie 2 !stara wersja 0x1000*/
      wy1(D11);delay();c=czyt;wy1(D0);b=0x4;dr=0x2000;
      break;
     case 3:                        /* przerwanie 3 !stara wersja 0x4000*/
      wy1(D12);delay();c=czyt;wy1(D0);b=0x1;dr=0x1000;
      break;
     case 4:                                    /* WYL.ZEG. */
      wy2(D13);delay();c=czyt;wy2(NOTD13);b=0x8;dr=0x2;
      break;
     case 5:                                    /* ZAL.ZL. */
      wy1(D14);delay();c=czyt;wy1(D0);b=0x10;dr=0x400;
      break;
     case 6:                                    /* GND */
      wy3(0x2);delay();c=czyt;wy3(D0);b=0x20;dr=0x200;
```

22

```
       break;
      case 7:                                   /* Kcut * WYL.B */
       wy4(D0);delay();c=czyt;delayp(20000);b=0x40;dr=0x100; /* tau przerz.*/
       break;
    }
        delayp(1000);
        odczyt1 = (inportw(MC42)&maskamc42)>>9;
        odczyt2 = c&dr;
        odczyt3 = czyt&0x8000;
        if(b==dattest)
          {
           if(odczyt1)
            p=1,printf("Brak odczytu w zespole alarmu sygnalu: %s\n",kom[j-1]
);
           if(odczyt2)
            w=1,printf("Brak odczytu w slowie stanu sygnalu: %s\n",kom[j-1]);
           if(!odczyt3)
            r=1,printf("Brak odczytu w zespole przerwan sygnalu: %s\n",kom[j-
1]);
     if(dr==0x100&&(p==1||r==1))
       printf("Mozliwa zbyt duza stala czasowa obwodu C1 (74123)\n");
         }
        else
          {
           if(!odczyt1)
            s=1,printf("Wadliwe zwarcie segmentu przelacznika E14 dla sygnalu
: %s\n",kom[j-1]);
           if(odczyt3)
            t=1,printf("Wadliwe zwarcie segmentu przelacznika C10 dla sygnalu
: %s\n",kom[j-1]);
          }
        j++;
       }
     if(p^s)
      printf("%s E14\n",text[2]);
     if(r^t)
      printf("%s C10\n",text[2]);
     if(p|r|s|t|w)              /* zatrzymanie jesli byl komentarz */
      blad();
     y+=p+r+s+t+w;              /* kontrola wyniku testu */
     p=r=s=t=w=0;
     i++;
  }
 scr_clr();
 printf("%s",text[1]);
 if('y)
   {
     printf("\t\t\t\t\tOK\n");
     cr();
     return(0);
   }
 else
  {
     blad();
     return (-1);
```

```
    }
}

/*koniec*/
```

```
/*1988-11-29*/
                        /* MZ70 */
          /* TESTOWANIE PAKIETU ZASILACZA RESOLWEROW MZ-70 */
              /* opracowal dr inz Marian Wrzesien dnia 1988.08.12 */

#define POJ_PAM 0x03ff                          /* pojemnosc pamieci SIN,COS*/
# include <fun.h>
# include <adr.h>
main(){mz70();}                                         /******/
mz70()
 {
  printf("\tTESTOWANIE PAKIETU MZ-70\n");
  if(cyf()==-1)
   {
   printf("\tTest przerwany\n");
   cr();
   return(-1);
   }
  return(anal()?-1:0);
}
cyf()
 {
   unsigned int  wyj, wyjwew, wyjzew, adr;
   int b, bi, i, j, p, s, x , t ;
   b=bi=i=j=p=s=t=0;
   scr_clr();
   printf("\tTESTOWANIE PAKIETU MZ-70\n");
   outportw(IT2B1C1,0x7c);                      /* Stan pracy    */
   printf("Wcisnij przycisk Nr1 pakietu MZ-70");
   cr();
   if((inportw(IT2A1C2)!=0xffff) || (inportw(IT2A2C3)!=0xffff))
     j++;
   printf("Wcisnij przycisk Nr2 pakietu MZ-70");
   cr();
   if( inportw(IT2A1C2)!=0x0 || inportw(IT2A2C3)!=0x0 || j!=0 )
    {
      printf("\tUsterka w zespole A5,B1,B2,B3,B4,B5,D5,przyciski sterujace\
n");
      printf("\tlub nie wykonane zalecone instrukcje\n");
      blad();
      return(-1);
     }
    else
     {
      printf("STEROWANIE PRZYCISKAMI SIN,COS min-max\t\t\t\t\tOK!\n\n");
      printf("Zwolnij przyciski Nr1,Nr2 pakietu MZ-70");
      cr();
      j=0;
      while((inportw(IT2A1C2)!=0x7f7f) || (inportw(IT2A2C3)!=0x0))
       {
         outportw(IT2B1C1,0x60);                        /*ZER + BLOK + GEN */
            outportw(IT2B1C1,0x68);                             /*ZER + BLOK*
/
     outportw(IT2B1C1,0x60);
     outportw(IT2B1C1,0x68);
     j++;
```

```
    if(j>15)
     {
       printf("\tCzesc cyfrowa pakietu MZ-70 nie zeruje sie,\n");
       printf("\tusterka w zespole (1,2,3)/F1,F2,F3,F4,E1,E4 - B2,B5,D5(wejsc
ia),\n");
       printf("\tlub nie wykonana zalecona instrukcja.");
       blad();
       return(-1);
      }
     }
    }
   j=0;
   while(j<2)
    {
      j?(adr=IT2A2C3):(adr=IT2A1C2);
      while(i++<=POJ_PAM)
       {
         wyjwew=(inportw(adr)>>3)&0x1f;
    wyjzew=inportw(adr)&0x1f;
    p=cmpbit(wyjwew,wyjzew);
    if(p+1)
      {
        printf("\n\tNajstarszy bledny bit: 2exp%d sygnalu",p);
        if(!j)                                    /*przypadek 0 bledow*/
          {
            printf(" SIN,\t\nUsterka w zespole E1,B2,B5.\n");
            s=1;
          }
        else
          {
            printf(" COS,\n\tUsterka w zespole E4,B5,D5.\n");
            t=1;
          }
        printf("\tPrawidlowa wartosc sygnalu wyjsciowego: 0x%x\n",wyjwew);
        printf("\tWartosc zmierzona sygnalu wyjsciowego : 0x%x\n",wyjzew);
        if(s*t)
           printf("\tMozliwa usterka w zespole (1,2,3)/F1,F2,F3,F4\n");
           printf("\nWcisniecie 'p' przerywa test\n");
             if(getch()=='p')
           return(-1);
      }
    outportw(IT2B1C1,0x78);                       /* BLOK */
    outportw(IT2B1C1,0x70);                       /* GEN + BLOK */
       }
      i=0;
      j++;
    }
   outportw(IT2B1C1,0x7c);
   if(!(s!t))
     {
       scr_clr();
       printf("\tTESTOWANIE PAKIETU MZ-70\n");
       printf("\nCZESC CYFROWA PAKIETU MZ-70\t\t\t\t\t\tOK!\n");
       cr();
       return(0);
```

```
      }
      blad();
      return(-1);
 }

anal()
 {
   static char *koma[]=
     {
   /*0*/   "Przekroczona dopuszczalna rozrica pomiedzy amplitudami",
        /*1*/       "Niewlasciwa wartosc amplitudy przebiegu",
        /*2*/       "Przekroczona dopuszczalna wartosc skladowej stalej przebieg
u",
       };
   struct
     {
       char *koma1;
       int data;
     } static sterowanie[]=
       {

           /*0*/     "SIN+ i SIN-", 0x247c,
   /*1*/     "SIN+ i COS+", 0x307c,          /*COS+/*/
   /*2*/     "SIN+ i COS-", 0x2c7c,
   /*3*/     "COS+ i COS-", 0x6c7c,
   /*4*/     "COS+ i SIN-", 0x647c,
   /*5*/     "COS- i SIN-", 0x847c,          /*COS-/*/
   /*6*/     "SIN+       ", 0x287c,
   /*7*/     "SIN-       ", 0x447c,
   /*8*/     "COS+       ", 0x687c,
   /*9*/     "COS-       ", 0x4c7c,
   /*10*/    "COS~       ", 0xa07c,
   /*11*/    "SIN~       ", 0x147c,
      };
    int nrkoma,x,wy_anal,nrdata,b,mask,j,k;                  /* 0,1,2, wg koma[]
 */
    wy_anal=b=nrkoma=0;        /* sygnal wyjsciowy z trzech komparatorow IT2
; OK=0x3 */
    nrdata=-1;                 /* wybor rodzaju testu dla czesci analogowej
*/
    outportw(IT2B1C1,0x7c);
    printf("\nAMPLITUDY SYGNALOW SIN i COS:\n\n");
    delayp(30000);                    /*ustabilizowanie sie sin i cos*/
    while(nrdata++<11)                        /* nrdata=0x24dc,...,0xfbdc */
      {
         outportw(IT2B1C1,sterowanie[nrdata].data);
    delayp(10000);           /* opoznienie przy badaniu SIN i COS */
         wy_anal=inportw(IT2B2B4);
         if(nrdata<6)
           nrkoma=0,mask=0x2;                 /* badanie roznicy amplitud */
    else
    if(nrdata<10)
        nrkoma=1,mask=0x4;             /* badanie amplitud */
    else
        nrkoma=2,mask=0x1;             /* badanie skladowej stalej */
```

```
    if(!(wy_anal&mask))
      {
        b++;
        printf("%s",koma[nrkoma]);
             printf(" %s\n",sterowanie[nrdata].koma1);
      }
    else
      printf("%s\t\t\t\t\t\t\t\tOK!\n",sterowanie[nrdata].koma1);
      }
    outportw(IT2B1C1,0x7c);
    if(!b)                                  /* ocena testu czesci analogowej
*/
      {
        printf("\nCZESC ANALOGOWA PAKIETU MZ-70\t\t\t\t\tOK!\n");
  cr();
        return(0);
      }
    else
  blad();
        return(-1);
}
                                        /*koniec*/
```

```
/*1988-11-29*/
                    /* URMZ70 */
            /*URUCHAMIANIE PAKIETU MZ-70; CZESC ANALOGOWA */
                 /* opracowal dr inz Marian Wrzesien dnia 1988.10.01 */
# include <fun.h>
# include <adr.h>
main(){urmz70();}
urmz70()
  {
    printf("URUCHAMIANIE PAKIETU MZ-70\nczesc analogowa (a)\nczesc cyfrowa
(c)\n");
    printf("Wybierz opcje.\n");
    (getch()=='a')?uranal():urcyf();
  }
    static char *text[]=
      {
        "1AB,2AB",
   "3AB,4AB",
   "URUCHAMIANIE PAKIETU MZ-70; CZESC ANALOGOWA",
   "URUCHAMIANIE PAKIETU MZ-70; CZESC CYFROWA",
   "Normalne dzialanie pakietu.",
   "Wartosci wyjsciowe analogowe zerowe.",
   "Wartosci wyjsciowe analogowe dodatnie.",
   "Wartosci wyjsciowe analogowe ujemne.",
   "Praca krokowa.",
   "Koniec uruchamiania czesci cyfrowej pakietu MZ70.",
      };
    uranal()
   {
    int i=0;
    char k;
    scr_clr();
    outportw(IT2B1C1,0x7c);                    /*pakiet generuje SIN i COS */
    printf("\n%s\n\n",text[2]);
  while(i<2)
   {
    printf("Dolacz woltomierz pradu stalego do zaciskow %s gniazda 1ELN.\n"
,text[i]);
    printf("Po kolejnym wciskaniu CR wyrownuj wartosci bezwzgledne wskazywa
nych napiec,\n");
    printf("przy ustawionym napieciu ujemnym,\n");
    printf("przy pomocy potencjometru P%d\n",i+3);
    printf("Roznica nie moze przekraczac ¦10mV¦.\n");

    printf("Po osiagnieciu celu wcisnij 'p'.\n");
    while(1)
      {
   outportw(IT2B1C1,0x38);          /* SC MIN z pakietu IT2 */
       if(getch()=='p')
      break;
   outportw(IT2B1C1,0x58);          /* SC MAX z pakietu IT2 */
   if(getch()=='p')
      break;
      }
    scr_clr();
```

```
     printf("%s\n\n",text[2]);
     printf("Ustaw potencjometrem P%d wartosc napiecia 9V\n",i+1);
     cr();
     i++;
     scr_clr();
     printf("%s\n\n",text[2]);
     if(i==2)
       {
         printf("Czy powtarzasz strojenie? (t/n)\n");
         if(k=getch()!='t')
         break;
    i=0;
       }
}
printf("\n\n\n\nStrojenie zakonczone.Powtorz testowanie!!!\n");
cr();
return(0);
}
   urcyf()
    {
      char x,n,b,p,m,k;
      scr_clr();
      do
     {
      scr_clr();
      printf("\n%s\n",text[3]);
      printf("Wybor opcji:\n");
      printf("\tn - %s\n",text[4]);
      printf("\t0 - %s\n",text[5]);
      printf("\t+ - %s\n",text[6]);
      printf("\t- - %s\n",text[7]);
      printf("\tk - %s\n",text[8]);
      printf("\tt - %s\n",text[9]);
      switch(x)
    {
      case 'n':
            outportw(IT2B1C1,0x7c);
            printf("\n%s\n",text[4]);
            break;
      case '0':
            outportw(IT2B1C1,0x68);
            outportw(IT2B1C1,0x60);
            printf("\n%s\n",text[5]);
            break;
      case '+':
            outportw(IT2B1C1,0x58);
            printf("\n%s\n",text[6]);
            break;
      case '-':
            outportw(IT2B1C1,0x38);
            printf("\n%s\n",text[7]);
            break;
      case 'k':
            printf("\n%s\n",text[8]);
            printf("Kazde nacisniecie karetki zmienia sygnal cyfrowy o jeden krok
.\n"
```

```
);
            printf("Wcisniecie 'p' konczy prace krokowa.\n");
            do
              {
                outportw(IT2B1C1,0x78);
                outportw(IT2B1C1,0x70);
              }while(getch()!='p');
            printf("\nKoniec pracy krokowej.Wybierz opcje.\n");
            break;
     case 't':
                    printf("\n%s\n",text[9]);
            cr();
            return(0);
   }
     }while(x=getch());
    }
                                        /*koniec*/
```

```
/*1988-11-29*/
   /* ADRESY WYSTAWIANE NA LINIACH ADR... MAGISTRALI SA NEGACJAMI BITOW
      OKRESLONYCH ADRESAMI W FUNKCJACH OUTPORT(BW) I INPORT(BW)
      POSTAC OGOLNA ADRESU PAKIETU MA-70:
      XX(4*DAT) ORAZ XX(4*DAT+2) DLA CZYTANIA I PISANIA,
      PRZY CZYM DAT=0x00 DO 0x0F.
      DAT OKRESLA SIE: outportb(0xfdd8,0xX0-0xXf)          */

                    /* DEKMA70.C */
   /* TESTOWANIE DEKODERA ADRESOW PAKIETU MA70 */
   /* opracowal dr inz. Marian Wrzesien dnia 1988-10-05 */

# include <fun.h>
# include <adr.h>
main(){dekma70();}                                    /******/
dekma70()
 {
  int xackpt(),xa,p,k,ZA,kp;
  unsigned long adr=0x0;
  ZA=xa=p=k=kp=0;                          /* ZA=ZA0,ZA1,ZA2,ZA3 */
  printf("\nDEKODER ADRESOW:");
  while(ZA<=0xf)
    {
      outportb(IT3B3WY,ZA);
      adr=0;
      while(adr<=0xff)
       {
        if((xa=xackpt('1',adr)!=0)&&(adr!=4*ZA)&&(adr!=4*ZA+2))
    {
              k=1;
      printf("\nNiewlasciwe generowanie sygnalu XACK/ przy adresie:adr=0x%x (
odczyt)",adr);
    }
   if(xa==0&&(p=((adr==4*ZA)||(adr==4*ZA+2))!=0))
    {
      k=1;
      printf("\nBrak sygnalu XACK/ przy adresie: adr=0x%x (odczyt)",adr);
    }
        if((p!=0)&&(xackpt('o',adr)!=1))
    {
      k=1;
      printf("\nBrak sygnalu XACK/ przy adresie: adr=0x%x (zapis)",adr);
    }
         if(k)
      {
        kp=1;
        printf("\nprzy stanie sygnalu ZA=0x%x",ZA);
        printf("\n* p * przerywa test\n\t\t\t\t\t\t\t\t* CR *\n");
        if(getch()=='p')
          return(-1);
       }
   k=0;
   adr++;
       }
      ZA++;
```

```
    }
      scr_clr();
      printf("DEKODER ADRESOW");
  if(kp)
    {
      blad();
      return(-1);
    }
  printf("\t\t\t\t\t\t\tOK!");
  cr();
  return(0);
}
                              /*koniec*/
```

```
                    /* DKMA70WE.C */
          /*TESTOWANIE DEKODERA ADRESOW WEWNETRZNYCH */
    /* opracowal dr inz. Marian Wrzesien dnia 1988.08.22 */
/* Uwaga!Jesli CSPROM/ pojawi sie razem z CSRAM/ w zakresie RAM, to nie da
sie/*
/*        tego wykryc */

# include <fun.h>
# include <adr.h>
# define init 0x0
# define notinit 0x20
# define prominh 0x20
# define hold 0x10
main(){dkma70we();}                                          /******/
dkma70we()
  {
    int fhold();
    int sreadypt(),xa,i,k,p;
    unsigned long adr,dat;
    xa=i=k=p=0;
    fhold();        /* kontrola HOLD */
    scr_clr();
    printf("DEKODER ADRESOW WEWNETRZNYCH\n");
    printf("Sygnaly WE0-WE5, WY0-WY4:");
    adr=0xf000;
    while(adr<=0xf0ff)              /*zakres zmian wplywajacych na dekoder */
     {
      (adr>=0xf000&&adr<=0xf005)?(i=2):(i=1);
      if((xa=sreadypt('bi',adr)==1)&&(i==1))
   {
         k=1;
     printf("\nNiewlasciwe generowanie sygnalu SREADY przy adresie: adr=0x%x
(odczyt)\n",adr);
   }
      if(xa==0&&i==2)
   {
         k=1;
     printf("\nBrak sygnalu SREADY przy adresie: adr=0x%x (odczyt)\n",adr);
   }
      if((i==2)&&(sreadypt('bo',adr)!=1)&&adr!=0xf005)
   {
         k=1;
     printf("\nBrak sygnalu SREADY przy adresie: adr=0x%x (zapis)\n",adr);
   }
      adr++;
      if(k)
        {
         blad();
         return(-1);
    }
     }
      printf("\t\t\t\t\t\tOK!\n");

    k=0;
    printf("\nSYGNAL CSRAM/ ");
```

```
    dat=0x0;
    while(dat<=0x27ff)              /* zakres zmian dla RAM i ROM */
     {
       adr=0x80000|dat;             /* tylko takie zmiany istotne!*/
       (adr>=0x82000&&adr<=0x823ff)?(i=2):(i=1);
       if((xa=sreadypt('br',adr)==1)&&(i==1))
   {
     k=1;
     printf("\nNiewlasciwe generowanie sygnalu CSRAM/ przy adresie: adr=0x%x
(odczyt)\n",adr);
   }
       if(xa==0&&i==2)
   {
     k=1;
     printf("\nBrak sygnalu CSRAM/ przy adresie: adr=0x%x (odczyt)\n",adr);
   }
       if((i==2)&&(sreadypt('bw',adr)!=1))
   {
     k=1;
     printf("\nBrak sygnalu CSRAM/ przy adresie: adr=0x%x (zapis)\n",adr);
   }
       dat++;
       if(k)
    {
      blad();
      return(-1);
    }
       }
    if(!k)
      printf("\t\t\t\t\t\t\t\tOK!\n");
    k=0;
    printf("\nSYGNAL CSPROM/");
    dat=0x0;
    outportb(IT3B3WY,(prominh|hold));              /* PROMINH/ =1 */
    while(dat<=0x07ff)
       {
   adr=0x80000|dat;
   if(!(sreadypt('br',adr)))
    {
      k=1;
      printf("\nBrak sygnalu CSPROM/ przy adresie: adr=0x%x (odczyt)\n",adr);
    }
   dat++;
   if(k)
    {
      blad();
      return(-1);
    }
       }
    printf("\t\t\t\t\t\t\t\tOK!\n");
    cr();
 return(0);
}
                              /*koniec*/
```

```
                        /* EPROM.C */
                    /*TESTOWANIE PAMIECI EPROM  */
    /* opracowal dr inz. Marian Wrzesien dnia 1988.10.06 */
# define SUM 0xb301
# define SUM1 0x7
# define prominh 0x20
# define hold 0x10
# include <adr.h>
# include <fun.h>
main(){eprom();}                                    /******/
eprom()
  {
    int xackpt(),xa,i,k=0,dat;
    unsigned long suma1,suma,a,ai;
    unsigned long adr;
    fhold();
    outportb(IT3BZWY,(prominh|hold));
    scr_clr();
    printf("PAMIEC EPROM PAKIETU MA-70");
   suma1=0x0;
   suma=0x0;
   adr=0x80000;
        while(adr<=0x807ff)
         {
      /*printf("EPROM=0x%x\n",mreadb(adr)&0xff);*/
      a=mreadb(adr);
      suma+=(a&0xff);
      if(suma>=0xffff)
            {
        suma-=0xffff;
        suma1++;
       }
      adr++;
     }
       if(suma!=SUM||suma1!=SUM1)
    {
          printf("\nSuma zawartosci komorek pamieci EPROM niezgodna z suma
 wzorcowa\n");
      printf("Suma=0x%x",suma1);
      printf("%x",suma);
      blad();
      return(-1);
     }
   printf("\t\t\t\t\t\t\tOK!");
        cr();
        return(0);
}
                              /*koniec*/
```

```
/*1988-11-30*/
                    /* MP1MP2.C */
   /* TESTOWANIE PRZELACZNIKOW MP1,MP2 PAKIETU MA70 */
      /* opracowal dr inz.Marian Wrzesien dnia 1988.08.13 */
# include <adr.h>
# include <wewyma70.h>
# include <fun.h>
main(){mp1mp2();}                                    /******/
int fhold();
mp1mp2()
  {
    int i,j,k,l,m,n,a[8],b[8],mp1,mp2;
    unsigned int datwyj,dattest;
    fhold();
    outportb(IT1B1,0x9b);
    outportb(IT1B3,0x8b);
    outportb(IT1B3PA,0x0);
    mp1=mp2=0;
    n=0;
    while(n<2)                      /* przelacznik MP1 oraz przelacznik MP2 */
     {
       k=0;
       while(k<2)           /* wedrujaca '1' oraz wedrujace '0' */
        {
          scr_clr();
             i=0;
     while(i<8)             /* 10 miejsc w liczbie datzad  */
      {
        scr_clr();
        if(!n)
          printf("\t\tUKLAD WYBORU PARAMETROW - MP1.\n\n");
        else
        printf("\t\tPRZELACZNIK PRZERWAN - MP2.\n\n");
        printf("przesuniecie segmentu przelacznika MP%d do dolu = 1\n",n+1);
        printf("przesuniecie segmentu przelacznika MP%d do gory = 0\n\n",n+1)
;
             printf("\t\t\t    Nr.MP%d:  1  2  3  4  5  6  7  8\n",n+1);
        printf("   Ustaw przelacznik MP%d w pozycji:",n+1);
        dattest=(1-k)*pelz1(i) + (k*~pelz1(i))&0xff;
        j=0;
        while(j<8)
         {
           a[j]=(dattest&pelz1(j))>>j;
           printf("  %d",a[j]);
           j++;
         }
        cr();
        if(!n)
          datwyj = inportb(WE5);
        else
         datwyj=inportb(IT1B1PC);
        if(dattest!=datwyj)
         {
           printf("\n Cdczytane polozenie segmentow MP:%d",n+1);
           m=0;
```

```
            while(m<8)
              {
                b[m]=(datwyj & pelz1(m))>>m;
                printf("  %d",b[m]);
                m++;
              }
            printf("\n");
            blad();
            mp1+=(1-n);
            mp2+=n;
          }
        i++;
          }
        k++;
      }
    n++;
  }
    scr_clr();
    printf("\nUKLAD WYBORU PARAMETROW");
    mp1?printf("\t\t\t\t\t^BLAD^"):printf("\t\t\t\t\t  OK!");
    printf("\nPRZELACZNIK PRZERWAN");
    mp2?printf("\t\t\t\t\t^BLAD^"):printf("\t\t\t\t\t  OK!");
    cr();
    return((mp1|mp2)?-1:0);
}
                          /*koniec*/
```

```
                    /* PWS.C */
   /* TESTOWANIE UKLADU KONTROLI POLOZENIA WALU SILNIKA */
   /*opracowal dr inz.Marian Wrzesien dnia 1988.08.23 */
# include <wewyma70.h>
# include <fun.h>
# include <adr.h>
# define CZASPOM 6000
# define intpws() (inportb(WE1)&0x80)
main(){pws();}                                          /******/
pws()
  {
    int i,n,datwyj,p,k,b,j,czasp;
    fhold();
    scr_clr();
    printf("UKLAD KONTROLI POLOZENIA WALU SILNIKA");
    outportw(IT2B1C1,0x7e);                  /* wysterowanie MZ-70 (generacja) + s
ilnik w p
rawo */
    outportb(IT1B2WY,0x1);          /* wlaczenie przekazu CP i SINREF (A6 it1) */
    delayp(30000);
    if(intpws())          /* kontrola INT  kosci D36 */
      {
        printf("\nSygnal INT (D36) nie zeruje sie\n");
      cr();
        return(-1);
      }
    outportw(IT2B1C1,0x6c);                  /* silnik stoi; generator blok */
    inportb(WE0);                    /* ten odczyt kasuje INT (D36) */
    if(!(intpws()))                          /* sygnal INT nie zmienia sie przy wp
isie */
      {
        printf("\nSygnal INT (D36) nie przyjmuje stanu 'H'\n");
      cr();
        return(-1);
      }
    outportw(IT2B1C1,0x7e);                  /* SIN,COS + silnik w prawo */
    delay();
    n=k=i=0;
    p=(inportb(WE1)<<8 | inportb(WE0))&0x3ff;
    while(n<2)
     {
      czasp=CZASPOM*(1+n);
      while(i<czasp)
       {
    while(intpws())
      ;
    datwyj=(inportb(WE1)<<8 | inportb(WE0))& 0x3ff;
         if(((k=p-datwyj)>1||k<-1) && k!=1023 && k!=-1023) /* k-rozn.biez.*
/
          {
            outportw(IT2B1C1,0x7c);                  /* SIN,COS + silnik stop */
            printf("\nBlad w rejestrach D24 i D36,\n");
            printf("wartosc prawidlowa = 0x%x\n",p);
            printf("wartosc odczytana  = 0x%x\n",datwyj);
       printf("test przerwany\n");
       cr();
       return(-1);
```

```
          }
        p=datwyj;
        i++;
       }
      i=0;
      n++;
      outportw(IT2B1C1,0x7d);                    /*wyster. MZ-70(gen.)+ silnik lewo */
     }
    outportw(IT2B1C1,0x7c);                      /* SIN,COS + silnik stop */
    printf("\t\t\t\t\tOK!");
    cr();
    return(0);
}
                              /*koniec*/
```

```
/*1988-12-01*/
                /* RAM.C */
          /*TESTOWANIE PAMIECI RAM  */
    /* opracowal dr inz. Marian Wrzesien dnia 1988.10.06 */
# define ADRAMBEG (unsigned long)0x82000
# define ADRAMEND (unsigned long)0x823ff
# include <fun.h>
main(){ram();}                                                  /******/
ram()
 {
  int fhold();
  int datwe,datwy,i;
  unsigned long adr;
  fhold();
  scr_clr();
  printf("PAMIECI RAM FAKIETU MA-70");
  i=0;
  while(i<4)
   {
     switch(i)
      {
           case 0: datwe=0x0;
              break;
      case 1: datwe=0xff;
              break;
      case 2: datwe=0xaa;
              break;
      case 3: datwe=0x55;
              break;
      }
     adr=ADRAMBEG;
     while(adr<=ADRAMEND)
      {
        mwriteb(adr,datwe);
        if((datwy=mreadb(adr)&0xff)!=datwe)
          {
            printf("\nBlad zapis/odczyt przy adresie=0x%x.\n",adr);
            printf("Wartosc wpisywana=0x%x\n",datwe);
            printf("Wartosc odczytana=0x%x\n",datwy);
       blad();
            return(-1);
          }
        adr++;
       }
     i++;
   }
 printf("\t\t\t\t\t\tOK!");
 cr();
 return(0);
}
                              /*koniec*/
```

41

```
                    /* URDAC.C */
   /* STROJENIE PRZETWORNIKA CYFROWO-ANALOGOWEGO */
          /* Opracowal dr inz. Marian Wrzesien dnia 10.08.1988 */
# include <wewyma70.h>
# include <adr.h>
# include <fun.h>
main(){urdac();}                                    /******/
urdac()
   {
     fhold();
     while(1)
   {
          scr_clr();
          printf("STROJENIE PRZETWORNIKA CYFROWO-ANALOGOWEGO\n");
     printf("Dolacz multimetr MUC2000 do punktu TP5 pakietu MA-70.\n");
     printf("Masa multimetru dolaczona do .O. analogowego !!!\n");
     printf("Reguluj napiecie 0V (zakres pomiarowy=200mV) potencjometrem P1 (
1k).\n");
     printf("Reguluj napiecie   10V (zakres pomiarowy=20V) potencjometrem P2
(10 k).\n\n");
     outportb(WY3,0x0);
     outportb(WY4,0x0);
     outportb(IT3B5PA,0x0);
     outportb(IT3B5PC,0x30);
     printf("Napiecie wyjsciowe=-10V\n");
     cr();
     outportb(WY4,0x8);
     outportb(IT3B5PC,0x38);
     printf("Napiecie wyjsciowe=0v\n");
     cr();
     outportb(WY3,0xff);
     outportb(WY4,0xf);
     outportb(IT3B5PA,0xff);
     outportb(IT3B5PC,0x3f);
     printf("Napiecie wyjsciowe=+10V\n\n");
          printf("* p * przerywa strojenie\n\t\t\t\t\t\t\t\t* CR *");
     if(getch()=='p')
       return(0);
   }
   }
                              /*koniec*/
```

```
/*1988-12-01*/
                    /* WEWY.C */
        /*TESTOWANIE ZESPOLU UKLADOW WEJSC/WYJSC */
    /* opracowal dr inz. Marian Wrzesien dnia 1988.08.18 */

# define KOC IT2B1C1
# define mar ((inportb(WE4)>>6)&0x1)
# define mar1 (((inportw(WE1C))&0x1)&((inportw(WE1C)>>1)&0x1))
# define mar5 (((inportb(WE4)>>5)&0x1)&((inportb(WE4)>>4)&0x1)&((inportb(WE
4)>>2)&0x1))
# include <wewyma70.h>
# include <adr.h>
# include <fun.h>
unsigned long pelz1();
int cmpbit();
int fhold();
main(){while(1){wewy();}}                                    /*******/
    static int p1,p2,p3,p4,p5,p6;
wewy()
  {
   p1=p2=p3=p4=p5=p6=0;
   outportw(KOC,0x0);
   rpr_rzp();
   ukladwej();
   outportb(IT1B1PB,0x0);
   rss();
   outportb(IT1B1PB,0x0);
   outportb(IT1B3PA,0x0);
   outportb(WY2,0x0);
   scr_clr();
   printf("OBWODY WE/WY");
   if(p1|p2|p3|p4|p5)
      {
         blad();
    return(-1);
      }
    else
      {
         printf("\t\t\t\t\t\t\t\t\tOK!\n");
    cr();
    return(0);
       }
 }
 rpr_rzp()
  {
   int p,magist;
   int k,m=0;
   int i,n=0;
   int zesp1=19;
   int zesp2=32;
   unsigned int dattest,datwyj;
   fhold();
   scr_clr();
   outportb(IT1B1,0x99);                          /* PB out; PA,PC in */
   outportb(IT1B1PB,0x0);
```

```
printf("REJESTR POLOZENIA RZECZYWISTEGO -");
while(m<2)
 {
  if(m)
    printf("REJESTR ZADANEGO PRZYROSTU      - ");
  k=0;
  while(k<2)
   {
     i=0;
     while(i<16)
      {
        dattest = (1-k)*pelz1(i) + k*(~pelz1(i));        /*sygnal testujacy */
        if(!m)
     {
       outportb(WY0,dattest);                          /* wpis do D32 */
       outportb(WY1,dattest>>8);                       /* wpis do D31 */
       datwyj = inportw(WE0C);                         /* odczyt z D31+D32 */
      }
    else
     {
        outportw(WY0C,dattest);
        datwyj =inportb(WE3)<<8|inportb(WE2);
     }
    p = cmpbit(dattest,datwyj);           /* jesli ok funkcja zwraca -1*/
    if(p+1)
      {
        magist=p;
        if(p<4)
          ;
        else
        if(p<8)
          zesp1-=1;
        else
          if(p<12)
            {
              magist=p-8;
              zesp1+=2;
              zesp2-=1;
             }
          else
           {
              magist=p-8;
              zesp1+=1;
              zesp2-=1;
            }
     printf("\n\n\tBledny przekaz po linii:\n");
     printf("\tsystemowa magistrala danych - sygnal DAT %d,\n",p);
     printf("\tbrama we/wy - obwod D %d,\n ",zesp1);
     printf("\tbrama rejestrow testowanego zespolu - obwod D %d,\n",zesp2);
     printf("\tsygnal DB %d wewnetrznej magistrali danych.\n",magist);
     printf("\tBlad powstal przy wyslaniu danej:  0x%x\n",dattest);
     printf("\tOdczytana wartosc sygnalu:  0x%x\n",datwyj);
     p1=1;
     blad();
     goto rzp;
```

```
          }
        i++;
      }
          k++;
        }
       printf("\t\tOK!\n");
       rzp:
       m++;
       zesp1=19;
       zesp2=34;
      }
        return(0);
   }
ukladwej()
 {
    int maska1,maska8;
    int n,i,j,t,glob;
    int a[10];
    int p,k,k8,k9;
    unsigned int m=0,datwyj,dattest,datzad,dat6;
    glob=t=p=k=k8=k9=i=j=n=0;
    fhold();
    printf("UKLADY WEJSCIA :");
    while(glob<2)
     {
       printf("\n");
       t=0;
       while(((p!=2||n!=1)&&glob==0)||((p==2||n==1)&&glob==1))
        {
          if(!glob)
       printf("Wloz obwod D6 typu 74174 w podstawke i zewrzyj zwore Z1.");
          else
       printf("Wyjmij obwod D6 typu 74174 z podstawki i rozewrzyj zwore Z1.");

          cr();
          n=0;
          inportw(WEOC);                                  /* reset D22 zwora */
            n+=mar;
          outportw(WYIC,0x20);                            /* sterowanie D22  "1
"    */
            n+=mar;
          outportw(WYIC,0x0);                             /* sterowanie D22  "0
"       */
            n+=mar;
          p=0;
          outportw(WYIC,0xe);                             /* sterowanie D6 "1"
*/
            p+=mar5;
          outportw(WYIC,0x0);                             /* sterowanie D6 "0"
*/
            p+=mar5;
          outportw(WYIC,0xe);
            p+=mar5;
       t++;                                    /*dwukrotna szansa przy kontroli*/
       if(t==2)                                /*wsuniecia obwodu scalonego */
         {
           printf("Mozliwe uszkodzenie ukladow D22,D15,D6,D16 lub nie wykonane i
nstrukcje\n");
          printf("test UKLADOW WEJSCIA przerwany!\n");
```

```
          p3=1;
          blad();
          return(0);
        }
         }
        datzad=0;
        printf("test");
        while(datzad<=0x3ff)
         {
           if((datzad%15)==0)
                printf(".");
      i=0;
      while(i<10)                                  /* 10 miejsc w liczbie datzad  */
       {
         a[i]=(datzad&pelz1(i))>>i;
         i++;
       }
           dattest=(datzad & (a[1]?0xff:0xf7) & (a[8]?0xff:0xfe) & 0xfd)|a[9]
<<1;
      if(glob==0)                                          /* dla osi fi    */
        {
         dat6=a[6]<<5 | (a[3]&a[1])<<4 | a[5]<<3 |     /*a[6] ZAK.ZAPIS*/
              a[4]<<2 | a[2]<<1 | (a[0]&a[8]);         /*a[5] REDPRED  */
         outportw(WY1C,dat6);                          /*a[4] SEARCH   */
        }                                              /*a[3] ZEZWOL   */
      else                                             /*a[2] STOP     */
       {                                               /*a[0] SYNCHR   */
         dat6=(~((a[0]&a[8])<<5 | a[5]<<4 | a[4]<<3 | a[2]<<2 |
                 (a[3]&a[1])<<1 | a[6] ))&0x3f;
         outportb(IT1B1,0x99);
         outportb(IT1B1PB,dat6);
        }
      a[7]?inportb(WE3):outportw(WY0C,0x0);            /*a[7] WRITTEN */
      if(k9!=a[9])
        {
          outportw(K0C,(a[9]?0x200:0x0));              /*a[9] SYNCSW */
          delay();
          k9=a[9];
        }
      outport(IT3B3WY,(a[1]?0x50:0x10));               /*a[1] CZUJNIK */
      if(k8!=a[8])
        {
          outport(WY2,(a[8]?0x0:0x2));                 /*a[8] OSZSYNCHR */
          k8=a[8];
        }
      datwyj = inportb(WE4);
      p=cmpbit(dattest,datwyj);
      if(p+1)
        {
          printf("\ndattest = 0x%x",dattest);
          printf("\twyjscie = 0x%x\n",datwyj);
          printf("Jest blad na bicie BIT %d",p);
          printf(" * CR *, lub * p * w celu przerwania testu\n");
          if(getch()=='p')
            {
```

```
            outportb(IT1B1PB,0x0);
            p2=1;                                   /* kontrola poprawnosci*/
            return(0);
          }
      }
    datzad++;
       }
      glob++;
    }
   printf("\nUKLADY WEJSCIA\t\t\t\t\tOK!\n");
   outportb(IT1B1PB,0x0);
   return(0);
 }
 rss()
  {
   int i,p,n,glob,k,t;
   unsigned int dattest,datwyj,sterowanie,maska,maska1,datzad;
   int a[5];
   fhold();
   outportb(IT1B1,0x99);              /*zaprogr.B1 it1 PA in, PB out */
   outportb(IT1B1PB,0x0);             /* wszystkie wyjscia B1 = 'H' oc*/
   outportw(KOC,0x0);
   printf("REJESTR STANU STEROWNIKA POLOZENIA :");
   outportb(IT1B3,0x8b);              /* zaprogramowanie B3 it1 - sygnaly.....*/
   maska=0x407f;                      /* WRITTEN,BUDZIK ==1*/
   t=p=n=glob=k=0;
   while(glob<2)                      /* INPOS/,ERROR/,ROBZSYNCHR/ ==1 */
    {                                         /* (nie ma blokady ukladu D5) */
      printf("\n");
      t=0;
     while(((p!=2||n!=1)&&glob==0)||       /* wsuniecie kosci i zwarcie Z1 */
         ((p==2||n==1)&&glob==1))
        {
         if(glob==0)
           printf("Wloz obwod D5 typu 74125 w podstawke i zewrzyj zwore Z1.
");
         if(glob==1)
      printf("Wyjmij obwod D5 typu 74125 z podstawki i rozewrzyj zwore Z1.");

         cr();
         n=0;
         inportw(WEOC);                       /* reset D22 ZAKAZ ZAPISU */
           n+=mar;
         outportw(WY1C,0x20);                 /* sterowanie D22  "1"  */
           n+=mar;
         outportw(WY1C,0x0);                  /* sterowanie D22  "0"  */
           n+=mar;
         p=0;
         outportb(WY2,0x81);                  /* sterowanie D17 "1" */
    delay();
           p+=mar1;
         outportb(WY2,0x0);                   /* sterowanie D17 "0" */
    delay();
           p+=mar1;
         outportb(WY2,0x81);
```

```
    delay();
          p+=mar1;
    t++;                                /*dwukrotna szansa przy kontroli*/
    if(t==2)                            /*wsuniecia obwodu scalonego */
      {
       printf("Mozliwe uszkodzenie ukladow:D4,D5,D17, lub nie wykonane instr
ukcje\n");
       printf("Test REJESTRU STANU STEROWNIKA POLOZENIA przerwany!\n");
       p4=1;
       blad();
       return(0);
      }
      }

   datzad=0;
   while(datzad<=0x7f)
    {
      i=0;
      while(i<5)                        /* 5 miejsc w liczbie datzad  */
       {
         a[i]=(datzad&pelz1(i))>>i;
         i++;
       }
      dattest =( (a[4]<<14 | a[2]<<6 | a[1]<<5 | a[3]<<4 |
            a[0]<<3 | a[1]<<2 | a[3]<<1 | a[0] ) & maska) ;
      sterowanie= a[3]<<7 | a[2]<<2 | a[1]<<1 | a[0] ;
      outportb(WY2,sterowanie);
      a[4]?inportb(WE3):outportw(WYOC,0x0);                 /*a[4] WRITTEN */
      datwyj = (inportw(WE1C)) & maska ;           /* maska dla braku obwodu*/
      p=cmpbit(dattest,datwyj);
      if(p+1)
        {
     k=1;
          printf("\ndattest = 0x%x",dattest);
          printf("\twyjscie = 0x%x\n",datwyj);
          printf("Jest blad na bicie BIT %d\n",p);
             printf("* CR *, lub * p *  w celu przerwania testu\n");
     if(getch()=='p')
        {
          p6=1;
          return(0);
        }
   }
      datzad++;
    }
  glob++;
  maska=0x78;                                /* maska przy braku obwodu D5 */
 }
if(!k)
 {
   printf("\nREJESTR STANU STEROWNIKA POLOZENIA\t\tOK!");
   cr();
   return(0);
  }
else
```

```
  {
    p5=1;
    blad();
    return(0);
   }
}
                              /*koniec*/
```

```
/*1988-12-01*/
                    /* DAC.C */
      /*TESTOWANIE PRZETWORNIKA CYFROWO-ANALOGOWEGO */
   /* opracowal dr inz Marian Wrzesien dnia 1988.08.24 */
# include <adr.h>
# include <fun.h>
# include <wewyma70.h>
/** define notinit 0x20*/
main'){dac();}
dac()
  {
   int i,j,k,m,n,p,kompar,x;
   unsigned int datzad,datit3,datma;
   fhold();
   scr_clr();
   printf("PRZETWORNIK CYFROWO-ANALOGOWY");
   j=k=0;
   n=2;                                        /* liczba rozniacych schodkow */
   while(j<2)
     {
       m=1;
       datzad=0;
       while(m>=-1)
         {
            printf("\ntest");
            while(datzad>=0 && datzad<=0xfff)
              {
                if(datzad<0x800)
             {
               datit3=datzad+n;
               datma =datzad;
             }
                 else
             {
               datit3=datzad;
               datma =datzad-n;
             }

            outportb(WY3,datma&0xff);              /* sterowanie AD na pakiecie
MA-70 */
            outportb(WY4,datma>>8);                /* j.w. */
            outportb(IT3B5PA,datit3&0xff);         /* sterowanie AD na pakiecie
it3
   */
            outportb(IT3B5PC,((datit3>>8)|0x30));  /* j.w. pozostaja sygnaly X
   ack req i Waitreq*/
            delayp(10);
            kompar=inportb(IT3B5PB)&0x80;
            if(kompar!=0x80 && n!=0)
              {
                printf("\nNapiecie wyjsciowe DAC zbyt duze przy sygnale steru
   jacym =0x%x\n",datzad);
                printf("Wcisnij * CR * lub * p * w celu przerwania testu\n");
                k=1;
                if(getch()=='p')
                   return(-1);
              }
```

```
            if(kompar!=0x0 && n==0)
              {
                printf("\nNapiecie wyjsciowe DAC zbyt male przy sygnale steru
jacym =0x%x\n",datzad);
                printf("* p * przerywa test\r\t\t\t\t\t\t\t\t* CR *");
                k=1;
                if(getch()=='p')
                   return(-1);
              }
            if(!(datzad%70))
              printf(".");
            datzad+=m;
              }
            datzad=0xfff;
            m-=2;
          }
        j++;
        n=0;
      }
   if(!k)
     {
        scr_clr();
        printf("PRZETWORNIK ANALOGOWO-CYFROWY\t\t\t\t\t\tOK!");
   cr();
        return(0);
      }
  }
                                        /*koniec*/
```

```
                /* TESTOWANIE PLYTKI P3 PANELU PROGRAMOWANIA*/          /* 1988-01-26
 */

     # define klawisz (scanf("%d",&x))
     # define druk(nrkoma) (printf("\t%s\n\n",koma[nrkoma]))
     # define ster_it2(a,d)  (outportw(adr[a],data[d]))
        p3()
            {
               static int adr[]=
                  {
                     0xfbc0
                  };
               static int data[14]=
                  {
                     0x00fe,                                              /* 0
Nic nie swieci */
                     0x00f6,                                              /* 1
I kolumny */
                  0x00ee,                                              /* 2 II kolum
ny */
                  0x00de,                                              /* 3 III kolu
mny */
                  0x00be,                                              /* 4 IV kolum
ny */
                  0x007e,                                              /* 5 V kolumn
y */

                  0x00fd,                                              /* 6 I wiersz
e */
                  0x00fb,                                              /* 7 II wiers
ze */
                  0x00f7,                                              /* 8 III wier
sze */
                  0x00ef,                                              /* 9 IV wiers
ze */
                  0x00df,                                              /* 10 V wiers
ze */
                  0x00bf,                                              /* 11 VI wier
sze */
                  0x007f,                                              /* 12 VII wie
rsze */
                  0x0000,                                              /* 13 wszystk
ie swieca */
                  };
               static char *koma[]=
                  {
                  "         TESTOWANIE STATYCZNE",
                        "   TESTOWANIE IMPULSOWE",
                        "   TESTOWANIE DYNAMICZNE",
                        "Sprawdzic swiecenie",
                        " kolumn wyswietlaczy",
                        " wierszy wyswietlaczy",
                        " po sprawdzeniu wcisnij CR",
                        " wcisnij CR",
                        "TESTOWANIE PLYTKI P3 PANELU PROGRAMOWANIA",
                 "OCENA WYNIKOW TESTU NA PODSTAWIE OBSERWACJI WYSWIETLACZY PLYTKI P
     3",
                 "START...WSZYSTKIE DIODY WYSWIETLACZY SWIECA",
                 "KONIEC TESTU PLYTKI P3 PANELU PROGRAMOWANIA",
                  };
               int st0,st1,st2,st3,st4,n,N,x;
```

52

```
ale pomocnicze */

                scr_clr();
                ster_it2(0,0);                                    /* zg
aszenie diod wyswietlaczy */
                druk(8);
                   czas(30000);
                   czas(30000);
                druk(9);
                   czas(30000);
```

```
                druk(10);
                ster_it2(0,13);
                   czas(30000);
                printf("%s\n",koma[6]);
                klawisz;
                st4=-1;
                N=3;
                while(st4++<2)                                  /* po
trojna petla 1:S
     TAT.,IMP.,DYN. */
                    {
                       st0=st2=st3=0;
                       st1=5;
                       scr_clr();
                       druk(st4);                               /* koma[st4]:
 test.STAT. lub  IMP. lu
     b DYN. */
                       printf("\t%s\n",koma[7]);
                       ster_it2(0,0);
                       klawisz;
                          if(st4<2)
                             {
                                while(st3++<2)                  /* po
dwojna petla 1:test. kolumn
      2:test. wiersz */
                                   {
                                      while(st0++<st1)          /* pe
tla wybierania data[
     st0] */
                                         {
                                            printf("\t\t%s",koma[3]);
                                            printf(" %d",(st0-st2));
                                            printf("%s\n\t",koma[3+st3]);
                                            printf("%s\n",koma[6]);
                                               {
                                                  for(n=-N;n<N;n++)     /* cz
as trwani
     a sterowania impulsowego */
                                                     {
                                                        ster_it2(0,0);  /* ni
c nie
     swieci */
                                                        czas(100);
                                                        ster_it2(0,st0);  /* swieci
     kolumna lub wiersz = f(st0) */
                                                        czas(100);
                                                     }
                                               }
                                            klawisz;
                                         }
                                      printf("\n");
                                      st1=12;                   /* ro
zszerzenie zakresu data[st0
     ] do st0=12 */
                                      st2=5;                    /* us
tawienie nr liczby porzadkow
     j wiersza na = 1 */
                                      st0-=1;                   /* po
prawka na odliczanie data[s
     t0] */
                                   }
                                N=100;
```

54

```
                    }
                  else
```



```
                    for(n=0;n<150;n++)                          /* liczba obi
egow wtest. DYN.
      */
                       {
                          st0=st3=0;
                          st1=12;
                          while(st3++<2)                              /* po
dwojna petla dla wiers
     zy i kolumn */
                             {
                                while(st0++<st1)                      /* pe
tla wyboru adre
     su koma[] */
                                   {
                                      ster_it2(0,st0);          /* swieci kol
ej
     na kolumna i nastepny wiersz */
                                      czas(1000-6*n);
                                      printf("%d\n",st0);             /* ws
kaznik
     kontrolny - normalnie Ctrl Y */
                                   }
                                st0-=1;                               /* po
prawka na odliczanie da
     ta[st0] */
                             }
                       }
                    ster_it2(0,0);                                  /* wy
gaszenie diod wyswietlaczy
     */
                 }
               druk(11);                                            /* ko
niec testu */
          czas(30000);
               return(0);
           }
        czas(t)                                                   /* funkcja wp
rowadzajaca opoznienie czasowe */
          int t;
           {
              int q;
              for(q=-t;q<t;q++)
              ;
              return(0);
           }
                                    /* KONIEC */
```

opr. Zbigniew Stonak

```
#include tes.h
#include ascii.h
#include tstglb.h


main()
{
int k;
while((k = menugl()) ){
     switch(k){
          case 1:mem_line=0;
                 ma70();
                 mem_line=0;
                 break;
          case 2:mz70();
                 break;
          case 3:mem_line=0;
                 mc42();
                 mem_line=2;
                 break;
          case 4:mem_line=0;
                 mw31();
                 mem_line=3;
                 break;
          case 5:mw32();
                 break;
          case 6:ml50();
                 break;
          case 7:mi50();
                 break;
          case 8:mem_line=0;
                 p_prog();
                 mem_line=7;
                 break;
          case 9:pr_ur();
                 break;
    }




  }
}


p_prog(){
int k;
while((k = menuprog()) ){
     switch(k){
          case 1:p1();
                 break;
          case 2:p2();
                 break;
          case 3:p3();
                 break;
          case 4:return 0;
             break;
     }
}
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include proext.h

int menugl()
{
/* ************************ menu ***************************************
* */
static char * naglowek[2] ={
        "* menu  *",
"                           WYBOR  PAKIETU ",
};
static char * text[]={
         "MA70",
         "MZ70",
         "MC42",
         "MW31",
         "MW32",
         "ML50",
         "MI50",
         "PANEL PROGRAMOWANIA",
         "PROGRAM URUCHOMIENIOWY",
};
char * podpis = "                                                    * spa
cja * del * cr *";
int line_n = 9;
static int line_l[9]={4,4,4,4,4,4,4,19,22};
/* ************************ menu ***************************************
* */

return ( menu(naglowek,text,podpis,line_n,line_l) );
}
```



```
#include ascii.h
#include proext.h

int menuprog()
{
/* ************************ menu ***************************************
* */
static char * naglowek[2] ={
        "* menu 1 *",
"                           TEST PANELU PROGRAMOWANIA",
};
static char * text[]={
         "plytka p1",
         "plytka p2",
         "plytka p3",
         "glowne menu",
};
char * podpis = "                                                    * spa
cja * del * cr *";
int line_n = 4;
static int line_l[4]={9,9,9,11};
/* ************************ menu ***************************************
* */
return ( menu(naglowek,text,podpis,line_n,line_l) );
}
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include proext.h
menumw31()
{
int k;

/* ************************ menu ***************************************
* */
static char * naglowek[2] ={
        "* menu I *",
"                           TEST PAKIETU MW31",
};
static char * text[]={
         "test calkowity",
         "dekoder adresow",
         "blok kontroli magistrali",
         "uklad przerwan",
         "uklad kontroli zasilania",
         "program uruchomieniowy",
         "glowne menu",
};
char * podpis = "                                                    * spacja *
del * cr *";
int line_n = 7;
static int line_l[7]={14,15,24,14,24,22,11};
/* ************************** menu ***************************************
* */

return(menu(naglowek,text,podpis,line_n,line_l));
}
```

```
#include "ascii.h"
#include "proext.h"
static autotest();

mw31()
{
int k ;
inic_it2();
i_it1_mw();
while(k = menumw31()){
/*-------------------------------------------------------------------------- */
switch(k){
case 1:autotest();
       break;
case 2: dekmw31();
        mmap2();
       break;
case 3: bkm();/*test1*/
       break;
case 4:up() ;/*test2*/
       intmw31();
       break;
case 5:ukz();/*test3*/
       break;
case 6:;/*pr ur */
       break;
case 7:return 0;
       break;
default: printf("bledna wartosc zwracana przez menu2");
       cr();
       break;
  }
}/* end while */
/*-------------------------------------------------------------------------- */
}/* end mw31 */

/* ************************************************************** */

static autotest()
{
if(dekmw31());
  if(mmap2());
    if(bkm());
      if(up());
        if(intmw31());
          if(ukz()){
             cr();
             return;
          }
cr();
return 0;
}
/* ************************************************************** */
```

```
/*                              opracowal:  Zbigniew Stanczak
  */

#define ADRR OXFBFE
#define ADRZ OXFB06
#define ZER (int)0X0
#include ascii.h
#include proe:t.h
menumc42()
{
int k;


/* ************************ menu ****************************************
* */
static char * naglowek[2] ={
        "* menu I *",
"                           TEST PAKIETU MC42",
};
static char * text[]={
        "test calkowity",
        "dekoder adresow i przelacznika AP",
        "zapis do pakietu mc42",
        "sygnal OUTON/",
        "odczyt z pakietu mc42",
        "zespol detektora przeciazen",
        "glowne menu",
};
char * podpis = "                                                    * spacja *
del * cr *";
int line_n = 7;
static int line_l[7]={14,33,21,13,21,27,11};
/* ************************ menu ****************************************
* */

return(menu(naglowek,text,podpis,line_n,line_l));
}
```

```
/*                              opracowal:  Zbignew Stanczak
  */

#define ADRR 0XFBFE
#define ADRZ 0XFB06
#define ZER (int)0X0
#include ascii.h
#include proext.h
int dekmc42(),zapis(),outon(),odczyt(),przec();
static void autotest();
mc42()
{
int k;

while(k = menumc42()){
     switch(k){
          case 1:autotest();
                 break;
          case 2:dekmc42();
                 break;
          case 3:zapis();
                 break;
          case 4:outon();
                 break;
          case 5:odczyt();
                 break;
          case 6:przec();
                 break;
          case 7:return 0;
                 break;
          default:return -1;
    }
}
return 0;
}

static void autotest(){
if(dekmc42() == 0)
     if(zapis() == 0)
       if(outon() == 0)
         if(odczyt() == 0)
           if(przec() == 0){
          cr();
             }
cr();
return ;
}
```

```
/*                              opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include proext.h

int menu2ma7()
{
/* ************************ menu ***************************************
* */
static char * naglowek[2] ={
        "* menu 2 *",
"                       PROGRAMY URUCHOMIENIOWE MA70",
};
static char * text[]={
        "strojenie przetwornika c/a",
        "uruchamianie dekodera adresow",
        "uruchamianie dekodera adresow zewnetrznych",
        "uruchamianie zespolu ukladow we/wy",
        "poprzednie menu",
};
char * podpis = "                                                    * spacj
a * del * cr *";
int line_n = 5;
static int line_1[5]={26,29,42,34,15};
/* ************************ menu ***************************************
* */

return ( menu(naglowek,text,podpis,line_n,line_1) );
}
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include proext.h

int menu1ma7()
{
/* ************************ menu ***************************************
* */
static char * naglowek[2] ={
        "* menu 1 *",
"                          TEST PAKIETU MA70",
};
static char * text[]={
        "test calkowity",
        "dekoder adresow",
        "dekoder adresow wewnetrznych",
        "pamieci EPROM",
        "pamieci RAM",
        "przetwornik cyfrowo-analogowy",
        "uklad kontroli polozenia walu silnika",
        "zespol ukladow wejsc-wyjsc",
        "przelaczniki wyboru parametrow i przerwan",
        "identyfikacja typu pakietu",
        "programy uruchomieniowe MA70",
        "glowne menu",
};
char * podpis = "                                                    * spa
cja * del * cr *";
int line_n = 12;
static int line_1[12]={14,15,28,13,11,29,37,26,41,26,28,11};
/* ************************ menu ***************************************
* */

return ( menu(naglowek,text,podpis,line_n,line_1) );
}
```

opr. Zbigniew Stańczyk

```
#include tes.h
#include ascii.h
#include proext.h

static void autotest();
void ur_ma70();

ma70()
{
int k;
while((k = menu1ma7()) ){
    switch(k){
        case 1:autotest();
               break;
        case 2:dekma70();
               break;
        case 3:dkma70we();
               break;
        case 4:eprom();
               break;
        case 5:ram();
               break;
        case 6:dac();
               break;
        case 7:pws();
               break;
        case 8:wewy();
               break;
        case 9:mp1mp2();
               break;
        case 10:itp();
               break;
        case 11:mem_line=0;
               ur_ma70();
              mem_line = 10;
               break;
        case 12:return 0;
               break;
        default:return -1;
   }
 }
}


static void autotest(){
  if(dekma70() == 0)
    if(dkma70we() == 0);
      if(eprom() == 0);
        if(ram() == 0);
          if(dac() == 0);
            if(pws() == 0);
              if(wewy() == 0);
                if(mp1mp2() == 0);
cr();
return ;
```

```
}

void ur_ma70(){
int k;
while((k = menu2ma7()) ){
     switch(k){
          case 1:urdac();
                 break;
          case 2:udma70();
                 break;
          case 3:udma70we();
                 break;
          case 4:uwewy();
                 break;
          case 5:return ;
                 break;
    }
  }
}
```

```
/*                          opracowal:  Zbigniew Stanczak

  */

#define start_line  6
#define start_col  8
#include ascii.h
#include proext.h

menu(naglowek,text,podpis,line_n,line_l)
char *text[],*naglowek[],*podpis;
int line_n;
int * line_l;
{
static mem_line ;
int i,k;
#ifdef MERA
#define znak 0x61
stout(0x1b67);
#else
#define znak 0x2a
#endif
line_n -= 1;
echo_on_off = 0;
scr_clr();
scr_curs(2,30);
printf("%s",naglowek[0]);
scr_curs(3,0);
printf("%s",naglowek[1]);
for(i=0;i<=line_n;i++){
   scr_curs(start_line+i,start_col);
   printf("%s",text[i]);
}
scr_curs(23,0);
printf("%s",podpis);
lin_num = start_line + mem_line;
col_num = 0;
scr_curs(lin_num,col_num);
for(i=0;i<=80;++i)((i<(start_col-1))||(i>(*(line_l+lin_num-start_
line)+star
t_col)))?stout(znak):scr_curr();
scr_curs(lin_num,col_num);
while( (k=scr_getc()) != crr )
        {
     if((k != (int)speace )&&(k != (int)bs)) continue;
        for(i=0;i<=80;++i)((i<(start_col-1))||(i>(*(line_l+lin_nu
m-start_li
ne)+start_col)))?stout(speace):scr_curr();
        if(k == (int)speace )lin_num = ( lin_num == start_line +
line_n )?
 start_line: lin_num+1;
        else lin_num = ( lin_num == start_line )?  start_line + l
ine_n: lin
_num-1;
        scr_curs(lin_num,col_num=0);
        for(i=0;i<=80;++i)((i<(start_col-1))||(i>(*(line_l+lin_nu
m-start_li
ne)+start_col)))?stout(znak):scr_curr();
        scr_curs(lin_num,col_num=0);
     }
mem_line = lin_num - start_line;
```

```
return(mem_line +1);
}
```

```
/*                          opracowal:  Zbignew Stanczak
  */

#include tes.h
#include mc42.h
#include ascii.h
unsigned long adr1,adr2;

dekmc42(){
unsigned long adr;
int k,j,bl=0,i=0;
unsigned int * p;
unsigned int * ptr;
 cli();
 scr_clr();
 printf("\n\tTEST DEKODERA ADRESOW\n");
 if(xackpt('c',UL 0xfff0)){
   printf("%s",wy0);
   blad();
   cr();
   return -1;
 }
 printf("%s\n",wyd);
 printf("%s\n",wydru);
 cr();
 scr_clr();
 printf("%s\n",inf1);
 cr();
 scr_clr();
 adr1= UL ADRR;
 adr2= UL ADRZ;
 if(!xackpt('o',UL ADRR))if(!xackpt('o',UL ADRZ)){
                            adr1= UL ADRZ;
                            adr2= UL ADRR;
                            printf("%s",wy2);
                            blad();
                            cr();
                            return -1;
                           }
printf("\ntest pozytywny dekodera\t\t\t\t\t\t\t\t\tOK!\n");
printf("\n\tCzy wykonac test calkowity dekodera t/n\n");
printf("\tprzy roznych stanach przelacznika AP\n");
if(getch()== 't'){
 for(k=0;k<32;k++){
  scr_clr();
  printf("\nustaw przelacznik AP w pozycji:\n");
  for(j=0;j<5;j++){
     printf("\n\tAP%d\t",j);
     p=ltob(UL k,5);
     if(*(p+4-j))printf("<-\n");
     else printf("->\n");
  }
  printf("\n( <- zwarcie  -> rozwarcie)\n");
  if(cr()==esk)break;
  for(i=0,adr = 01;adr< UL 256;++adr){
```

```
    if(xackpt('i',adr)){
      printf("adres pojawienia sie syg. XACK/\t");
      ++i;
      printf("%x hex\n",(unsigned int)adr);
      if((adr != UL adrmc42[k])&&(adr != UL (adrmc42[k]+1))){
        bl=1;
        blad();
        /*if(cr()==esk)return -1;*/
      }
      if(i>6)
        if(cr() == esk){
          i=0;
          return -1;
        }
    }
   }
   if(cr()==esk)break;
  }
if(bl>0){
  printf("%s\n",wy3);
  printf("uszkodzony dekoder adresow\n");
  blad();
  cr();
  return -1;
}
scr_clr();
printf("\ndekoder adresow\t\t\t\t\t\t\t\tOK!\n");
cr();
}/*if*/
return 0;
}/*main*/
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include tes.h
#include mc42.h

odczyt(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
unsigned int *p;
unsigned long adr1,adr2;
inic_it2();
koc();
scr_clr();
 if(xackpt('i',UL ADRR))adr1= UL ADRR;
   else if(xackpt('i',UL ADRZ))adr1= UL ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= UL ADRR;
                       else if(k=xackpt('i',UL ADRZ))adr1= UL ADRZ;
            scr_clr();
        }
printf("\n\tTEST ODCZYTU Z PAKIETU\n");
for(i=0;i<16;i++){
    dwe=(unsigned int)pelz1(i);
    outportw((unsigned int)ADRTES,~dwe);
    delayp(5000);
    dwy=inportw((unsigned int)adr1);
    if(dwe != dwy){
      ++lk;
      bl=1;
      p = ltob( UL dwe,16);
      printf("\ndana wejsciowa na laczu obiektowym\t");
      for(k=0;k<16;k++)printf("%d",*(p+15-k));
      p = ltob( UL dwy,16);
      printf("\ndana odcytana z pakietu\t\t\t");
      for(k=0;k<16;k++)printf("%d",*(p+15-k));
      blad();
    }
    if(lk > 4){
      lk=0;
      if(cr() == esk)break;
    }
}
/*  lk=0;
    outportw( UI adr1,~dwe);
    dwy= inportw( UI ADRTES);
    if(dwy '= dewy){
    }*/
if(bl>0){
  printf("\nbald w obwodzie odczytu  DAT(N)-WE(N+1)");
  blad();
  cr();
  return -1;
}
```

```
printf("\nodczyt z pakiet\t\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

```
/*                          opracowal:  Zbignew Stanczak
  */

#include ascii.h
#include tes.h
#include mc42.h
#include adr.h


zapis(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
unsigned int *p;
unsigned long adr1,adr2;
inic_it2();
koc();
scr_clr();
 if(xackpt('i',UL ADRR))adr1= UL ADRR;
   else if(xackpt('i',UL ADRZ))adr1= UL ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= UL ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= UL ADRZ;
            scr_clr();
        }
outportb(IT1D6WY,0x10);
delayp(1000);
printf("\n\tTEST ZAPISU DO PAKIETU\n");
for(i=0;i<16;i++){
     dwe=(unsigned int)pelz1(i);
     outportw( UI adr1,~dwe);
     delayp(5000);
     dwy=inportw( UI ADRTES);
     if(dwe != dwy){
       ++lk;
       bl=1;
       p = ltob( UL dwe,16);
       printf("\ndana zapisywana do pakietu(negacja)\t");
       for(k=0;k<16;k++)printf("%d",*(p+15-k));
       p = ltob( UL dwy,16);
       printf("\ndana na laczu obiektowym \t\t");
       for(k=0;k<16;k++)printf("%d",*(p+15-k));
       blad();
     }
     if(lk > 4){
       lk=0;
       if(cr() == esk)break;
     }
}
lk=0;
/*    outportw( UI adr1,~dwe);
     dwy= inportw( UI ADRTES);
     if(dwy != dewy){
     }*/
if(bl>0){
  printf("\nbald w obwodzie zapisu  DAT(N)-WY(N+1)");
```

```
    blad();
    cr();
    return -1;
}
printf("\nzapis do pakiet\t\t\t\t\t\t\t\tOK'\n");
cr();
return 0;
}
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include tes.h
#include mc42.h
#include adr.h

outon(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
unsigned int *p;
unsigned long adr1,adr2;
inic_it2();
koc();
scr_clr();
 if(xackpt('i',UL ADRR))adr1= UL ADRR;
   else if(xackpt('i',UL ADRZ))adr1= UL ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= UL ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= UL ADRZ;
            scr_clr();
        }
dwe= 0x0000;
outportb(IT1D6WY,0x10); /* zezwolenie (praca normalna) syg. OUTON */
delayp(10000);
printf("\n\tSYGNAL OUTON\n");

outportw( UI adr1,~dwe); /* 0xffff */
delayp(5000);
dwy=inportw( UI ADRTES); /* 0x0000 */
if(dwy != 0){
  bl=1;
  p = ltob( UL dwe,16);
  printf("\ndana zapisywana do pakietu(negacja)\t");
  for(k=0;k<16;k++)printf("%d",*(p+15-k));
  p = ltob( UL dwy,16);
  printf("\ndana na laczu obiektowym\t\t");
  for(k=0;k<16;k++)printf("%d",*(p+15-k));
  blad();
  printf("\nblad zapisu,brak testowalnosci sygnalu OUTON/\n");
  cr();
  return -1;
}

outportb(IT1D6WY,0x0); /* zabr. syg. OUTON */
delayp(10000);

outportw(UI adr1,~dwe);
delayp(5000);
dwy=inportw(UI ADRTES);
if(dwy == 0){
  bl=1;
  p = ltob( UL dwe,16);
  printf("\ndana zapisywana do pakietu(negacja)\t");
```

```
  for(k=0;k<16;k++)printf("%d",*(p+15-k));
  p = ltob( UL dwy,16);
  printf("\ndana na laczu obiektowym \t\t");
  for(k=0;k<16;k++)printf("%d",*(p+15-k));
}

outportb(IT1D6WY,0x10); /* zezwolenie (praca normalna) syg. OUTON */
delayp(10000);

if(bl>0){
  printf("\nnie sprawne blokowanie sygnalem OUTON/\n");
  blad();
  cr();
  return -1;
}
printf("\nsygnal OUTON\t\t\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

```
/*                          opracowal:  Zbigniew Stanczak
  */

#include ascii.h
#include tes.h
#include mc42.h
#include adr.h
#define int2() ((inportb(0xfedc) & 0x4)>>2)
przec(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0,blm=0,bld=0;
unsigned int *p;
unsigned long adr1,adr2;
inic_it2();
outportb(IT1B1,0x9b);
koc();
scr_clr();
 if(xackpt('i',UL ADRR))adr1= UL ADRR;
   else if(xackpt('i',UL ADRZ))adr1= UL ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= UL ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= UL ADRZ;
            scr_clr();
        }
outportb(IT1D6WY,0x10);
delayp(1000);
printf("\n\tTEST DETEKTORA PRZECIAZENIA\n");
outportw(UI adr1,0xffff);
delayp(5000);
if(int2()==0){
  printf("\nsygnal INT2/ aktywny w stanie poczatkowym\n");
  blad();
  printf("Czy dokonywana bedzie regulacja sygnalu INT2 ? (t/n)");
  if(getch()=='t'){
     scr_ldel();
     printf("nalezy obracac potencjometr P1 w lewo");
     while(!int2());
     stout(bel);
  }
  else  bl=1;
  scr_ldel();
}
else
 for(i=0;i<16;i++){
    koc();
    dwe=(unsigned int)pelz1(i);
    outportw( UI adr1,~dwe);
    delayp(5000);
    dwy=inportw(UI ADRTES);
    if(dwe != dwy){
      bl=1;
      lk += 2;
      p = ltob( UL dwe,16);
      printf("\ndana zapisywana do pakietu(negacja)\t");
      for(k=0;k<16;k++)printf("%d",*(p+15-k));
```

```
      p = ltob( UL dwy,16);
      printf("\ndana na laczu obiektowym \t\t");
      for(k=0;k<16;k++)printf("%d",*(p+15-k));
      blad();
      printf("\nbrak testowalnosci detektore przeciazenia na pozycji %d\n",
++i);
      if(cr()==esk)break;
      continue;
    }
    if(lk > 4){
      lk=0;
      if(cr() == esk)break;
    }
/*    outportb(0xfede,0x9b);
    delayp(500);*/

    kocm();
    outportw( UI adr1,~dwe);
    delayp(5000);

    if(int2() == 0){
      lk++;
      printf("\nza niska greanica przeciazenia na pozycji %d\n",++i);
      blad();
      if(blm==0){
        printf("Czy dokonywana bedzie regulacja sygnalu INT2 ? (t/n)");
        if(getch()=='t'){
          scr_ldel();
          printf("nalezy obracac potencjometr P1 w lewo");
          while(!int2());
          stout(bel);
        }
        else  blm=1;
        scr_ldel();
      }
     }
    kocd();
    outportw( UI adr1,~dwe);
    delayp(5000);
    if(int2() != 0){
      lk++;
      printf("\nbrak dzialania detektora przeciazenia na pozycji %d\n",++i)
;
      blad();
      if(bld==0){
        printf("Czy dokonywana bedzie regulacja sygnalu INT2 ? (t/n)");
        if(getch()=='t'){
          scr_ldel();
          printf("nalezy obracac potencjometr P1 w prawo");
          while(int2());
          stout(bel);
        }
        else  bld=1;
        scr_ldel();
      }
```

```
      }
    if(lk>4){
      lk=0;
      if(cr()==esk)break;
    }
}
koc();
outportw(UI adr1,0xffff);
delayp(5000);
if((bl|bld|blm)>0){
  printf("\nbald zespolu detektora przeciazenia INT2/ \n");
  blad();
  cr();
  return -1;
}
printf("\ndetektor przeciazenia\t\t\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

opr. Zbigniew Stelcel

```
#include ascii.h
#include "adrmw3i.h"
bkm(){
scr_clr();
if(!xackpt('i',IOADR11))
printf("\n\tUstaw zwore D2 w pozycii rozwartej(polozenie dolne)");
while(!xackpt('i',IOADR11));
scr_clr();
outportb(IT1D6WY,notinit);
outportb(0xd9,0xffff);
outportw(IOADR11,0);
inportw(IOADR11);
pokew(RWADR11Q0,0xffff);
if(!(peekw(RWADR11Q7)&DAT7)){
                    scr_clr();
                    printf("\nUKLAD KONTROLI MAGISTRALI\t\t\t\t\t\t\t\tOK!");
                    cr();
                    return 0;
}
else{
   scr_clr();
   printf("\nUszkodzenie ukladu kontroli magistrali\t\t\t\t\t^blad^\n");
   printf("\nRozpoczac program uruchomieniowy ukladu kontroli magistrali ? T\
/N");
   if(getch() == 't'){
                    scr_clr();
                    printf("\nW odpowiedniej kolejnosci podawane sa sygnaly");
                    printf("\nsterujace na przerzutniki D9(3),D9(11),D8(3),D8(11)"),
                    printf("\noraz C1(1)-reset przygotowywujacy zespol przerzutnikow.")
                    printf("\nNa uklad podawane sa odpowiednie dane(DAT0/-DAT15/).")-
                    printf("\nW cyklu generowany jest kazdy z sygnalow MRDC/ IORC/ I"
C/.");
                    printf("\nZ tym ze sygnal MRDC/ umozliwia odczyt wyjscia przerzutni
(8),");
                    printf("\nstan tego wyjscia jest wyswietlany ponizej 0 - wartosc pr
owa");
                    printf("\n\n\n\t\t\t");
                    while(1){
                       outportb(0xd9,0xffff);
                       outportw(IOADR11,0);
                       inportw(IOADR11);
                       pokew(RWADR11Q0,0xffff);
                       if(peekw(RWADR11Q7)&DAT7)stout(0x31);
                       else stout(0x30);
                       stout(0x8);
                       if(sgetch())if(inportb(0xd8)== 0x1b)break;
                    }
                    return -1;
         }
  }
}
```

```
#include "proext.h"
#include "adrmw31.h"
#include "ascii.h"
/* ************************************************************** */

/* ************************************************************** */
static char KOMOO[]=
{
"\n\t\tpodczas testowania I/O instrukcja out"
};
/* ************************************************************** */
static char KOMOI[]=
{
"\n\t\tpodczas testowania I/O instrukcja in"
};
/* ************************************************************** */
komOr()
{
printf("\n\t\tpodczas testowania MR/MW instrukcja read");
printf("\n\t\t na jednym z czterech adresow ");
printf("\n\t\t zaleznie od stanu zwory D2 : ");
}
/* ************************************************************** */
komOw()
{
printf("\n\t\tpodczas testowania MR/MW instrukcja write");
printf("\n\t\t na jednym z czterech adresow ");
printf("\n\t\t zaleznie od stanu zwory D2 : ");
}
/* ************************************************************** */
komio()
{
printf("\n\t\t na jednym z czterech adresow ");
printf("\n\t\t zaleznie od stanu zwory D2 : ");
printf("\n\t\t\t\t\t%xhex",IOADR11);
printf("\n\t\t\t\t\t%xhex",IOADR00);
printf("\n\t\t\t\t\t%xhex",IOADR10);
printf("\n\t\t\t\t\t%xhex",IOADR01);
}
/* ************************************************************** */
komrO()
{
printf("\n\t\t\t\t%xEhex",RWADR11Q0>>4);
printf("\n\t\t\t\t\t%xEhex",RWADR00Q0>>4);
printf("\n\t\t\t\t\t%xEhex",RWADR10Q0>>4);
printf("\n\t\t\t\t\t%xEhex",RWADR01Q0>>4);
}
/* ************************************************************** */
komr5()
{
printf("\n\t\t\t\t%x4hex",RWADR11Q5>>4);
printf("\n\t\t\t\t\t%x4hex",RWADR00Q5>>4);
printf("\n\t\t\t\t\t%x4hex",RWADR10Q5>>4);
printf("\n\t\t\t\t\t%x4hex",RWADR01Q5>>4);
}
```

```
/* ********************************************************************** */
komr6()
{
printf("\n\t\t\t\t%x2hex",RWADR11Q6>>4);
printf("\n\t\t\t\t%x2hex",RWADR00Q6>>4);
printf("\n\t\t\t\t%x2hex",RWADR10Q6>>4);
printf("\n\t\t\t\t%x2hex",RWADR01Q6>>4);
}
/* ********************************************************************** */
komr7()
{
printf("\n\t\t\t\t%xOhex",RWADR11Q7>>4);
printf("\n\t\t\t\t%xOhex",RWADR00Q7>>4);
printf("\n\t\t\t\t%xOhex",RWADR10Q7>>4);
printf("\n\t\t\t\t%xOhex",RWADR01Q7>>4);
}
/* ********************************************************************** */
/* ********************************************************************** */
kom1()
{
printf("\n\t\tBrak sygnalu KOUT/ uszkodzony C5(1,2,13,12)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tE6(4,6,8,2,10,12)\n");
printf("\t\t\tD6(11,8)\n");
printf("\t\t\tD5(3,6,8)\n");
printf("\t\t\tD4(3,6)\n");
printf("\t\t\tE7(Q1)\n");
printf("\t\t\tE5(Q0)\n");
printf("\t\t\tC5(8)\n");
printf("\t\t\tE3(8)\n");}

/* ********************************************************************** */
kom2()
{
printf("\n\t\tBrak sygnalu KIN/ uszkodzony C5(6)");
}
/* ********************************************************************** */
kom3()
{
printf("\n\t\tBrak sygnalu CZYT.A/ uszkodzony E8(Q0)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tE7(Q0) \n");
printf("\t\t\tE12(8)\n");
printf("\t\t\tD3(1-2)\n");
printf("\t\t\tC3(6)\n");
printf("\t\t\tC4(8)\n");
printf("\t\t\tE2(4)\n");
}
/* ********************************************************************** */
kom4()
{
printf("\n\t\tBrak sygnalu UST.2/ uszkodzony E8(Q5)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tD3(12)\n");
printf("\t\t\tC3(9)\n");
```

```
}

/* ********************************************************************** */
kom5()
{
printf("\n\t\tBrak sygnalu UST.1/ uszkodzony E8(Q6)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tD3(10)\n");
printf("\t\t\tC2(6)\n");
}
/* ********************************************************************** */
kom6()
{
printf("\n\t\tBrak sygnalu CZYT.S/ uszkodzony E8(Q7)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tD3(9)\n");
printf("\t\t\tC2(8)\n");
}
/* ********************************************************************** */
kom7()
{
printf("\n\t\tBrak sygnalu WPISK/ uszkodzony C3(3)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tC4(12-8) \n");
}
/* ********************************************************************** */
kom8()
{
printf("\n\t\tBrak sygnalu ZER.2/ uszkodzony C3(11)");
}
/* ********************************************************************** */
kom9()
{
printf("\n\t\tBrak sygnalu ZER.1/ uszkodzony C2(3)");
}
/* ********************************************************************** */
kom10()
{
printf("\n\t\tBrak sygnalu CS uszkodzony C2(11)");
}
/* ********************************************************************** */
int dekmw31()
{
int k,zs;

for(zs=0;zs<=3;++zs){
scr_clr();
printf("\nUstaw zwore D2 w polozeniu:\tXX??\n");
printf("(XX-polozenie dowolne,");
printf(" ??-polozenia: 00, 01, 10, 11)\n");
cr();
 if((k = dekadrp()) == 0){
   printf("\t\tTest(pozytywny) dekodera adresow\t\t\tOK!");
   printf("\n\t\t(przy aktualnym stanie zwory D2)\n");
   printf("\nCzy rozpoczac test calkowity dekodera adresow(w przestreni 1 M
```

```
b) t/n\n");
    if(getch() == 't'){
     scr_clr();
     printf("\n\t\tTEST CALKOWITY DEKODERA ADRESOW\n");
     if((k = dekadrn()) != 0) break;
printf("\nTest\t\t\t\t\t\t\t\t\tOK!");
cr();
   }
  }
 else break;
}
if(k > 0){
    printf("\n\t\t\tUszkodzony dekoder adresow !\n");
    if(k != 11)printf("\n\t\t\tuszkodzenie wykryto  ");
    switch(k){
            case 1:
        printf("%s",&KOMOO);
        komio();
        kom1();
                        break;
            case 2:
        printf("%s",&KOMOI);
        komio();
        kom2();
                        break;
            case 3:
        komOr();
        komrO();
        kom3();
                    break;
            case 4:
        komOr();
        komr5();
        kom4();
                    break;
            case 5:
        komOr();
        komr6();
        kom5();
                    break;
            case 6:
        komOr();
        komr7();
        kom6();
            break;
            case 7:
        komOw();
        komrO();
        kom7();
                    break;
           case 8:
        komOw();
        komr5();
        kom8();
                    break;
```

```
            case 9:
        kom0w();
        komr6();
        kom9();
                    break;
            case 10:
        kom0w();
        komr7();
        kom10();
                    break;
            case 11:
                    break;
            default:
                    printf("\nBledny stan sygnalu XACK/ na pakiecie IT1 \n");
                    break;
    }
cr();
return -1;
}
scr_clr();
printf("\nTEST CALKOWITY DEKODERA ADRESOW\t\t\t\t\t\tOK!");
cr();
return 0;
}
/* ************************************************************** */

/* ************************************************************** */

int dekadrp()
{
int k=0,i;
                if(xackpt('o',(unsigned long)0xfffff))return -1;

            if(!xackpt('o',IOADR11))
            if(!xackpt('o',IOADR00))
                    if(!xackpt('o',IOADR10))
                    if(!xackpt('o',IOADR01))return 1;

            if(!xackpt('i',IOADR11))
            if(!xackpt('i',IOADR00))
                    if(!xackpt('i',IOADR10))
                    if(!xackpt('i',IOADR01))return 2;

            if(!xackpt('r',RWADR11Q0))
            if(!xackpt('r',RWADR00Q0))
                    if(!xackpt('r',RWADR01Q0))
                    if(!xackpt('r',RWADR10Q0))return 3;

            if(!xackpt('r',RWADR11Q5))
                    if(!xackpt('r',RWADR00Q5))
                    if(!xackpt('r',RWADR01Q5))
                    if(!xackpt('r',RWADR10Q5))return 4;

            if(!xackpt('r',RWADR11Q6))
            if(!xackpt('r',RWADR00Q6))
```

```
                if(!xackpt('r',RWADR01Q6))
                if(!xackpt('r',RWADR10Q6))return 5;

        if(!xackpt('r',RWADR11Q7))
        if(!xackpt('r',RWADR00Q7))
                if(!xackpt('r',RWADR01Q7))
                if(!xackpt('r',RWADR10Q7))return 6;

        if(!xackpt('w',RWADR11Q0))
        if(!xackpt('w',RWADR00Q0))
                if(!xackpt('w',RWADR01Q0))
                if(!xackpt('w',RWADR10Q0))return 7;

        if(!xackpt('w',RWADR11Q5))
        if(!xackpt('w',RWADR00Q5))
                if(!xackpt('w',RWADR01Q5))
                if(!xackpt('w',RWADR10Q5))return 8;

        if(!xackpt('w',RWADR11Q6))
        if(!xackpt('w',RWADR00Q6))
                if(!xackpt('w',RWADR01Q6))
                if(!xackpt('w',RWADR10Q6))return 9;

        if(!xackpt('w',RWADR11Q7))
        if(!xackpt('w',RWADR00Q7))
                if(!xackpt('w',RWADR01Q7))
                if(!xackpt('w',RWADR10Q7))return 10;
return 0;
}
/* ************************************************************ */
dekadrn()
{
unsigned long adr=0;
int zw,ii,ki,j=0,k=0,bld = 0;
ki= 0;
for(k=0;k<=3;k++){

 while(adr++ <= 0xffff){
                if(!(adr& 0xfff)){
                        if(j++>80){
                                stout(0xa);
                                stout(0xd);
                                j= 0;
                        }
                        stout(0x2e);
                }
                if(xackpt('i',adr)){
                        ki++;
                        printf("\nadres pojawienia sie sygnalu XACK/   .'
                        printf("%xhex\t przy instr. in\n",adr&0xffff);
                        j=0;
                        if((adr != IOADR11)&&(adr != IOADR00)&&
                          (adr != IOADR11N)&&(adr != IOADR10N)&&
                          (adr != IOADR10)&&(adr != IOADR01)){
                           printf("*******************************
```

```
*****************************\n");
                                        bld = 1;
                                    }
                                    if(ki >7){
                    printf("\n\tCzy testowac dalej t/n\n");
                    if(getch()=='t')ki = 0;
                     else return 11;
        }
                    }
/*                  if(xackpt('r',adr)){
                                    ki++;
                                    printf("\nadres pojawienia sie sygnalu XACK/ %x
                                    printf("%xhex\t przy instrukcji read\n",adr[
                                    j=0;
                                   if((adr != RWADR11Q0)&&(adr != RWADR00Q0)&&
                                    (adr != RWADR10Q0)&&(adr != RWADR01Q0)&&
                                    (adr != RWADR11Q5)&&(adr != RWADR00Q5)&&
                                    (adr != RWADR10Q5)&&(adr != RWADR01Q5)&&
                                    (adr != RWADR11Q6)&&(adr != RWADR00Q6)&&
                                    (adr != RWADR10Q6)&&(adr != RWADR01Q6)&&
                                    (adr != RWADR11Q7)&&(adr != RWADR00Q7)&&
                                    (adr != RWADR10Q7)&&(adr != RWADR01Q7)){
printf("********************************** ^ BLAD ^ ************************
***********\n");
                                    bld = 1;
                                   }
                    }
                   if(ki >7){
                       printf("\n\tCzy testowac dalej t/n\n");
                       j = 0;
                       if(getch()=='t')ki = 0;
                       else return 11;
                   }
*/
   }/*end while */
if(k == 0){
   if(xackpt('i',IOADR11)) adr = RWADR11Q0,zw=11;
   if(xackpt('i',IOADR00)) adr = RWADR00Q0,zw=00;
   if(xackpt('i',IOADR10)) adr = RWADR10Q0,zw=10;
   if(xackpt('i',IOADR01)) adr = RWADR01Q0,zw=01;
}
if(k == 1){
   if(zw == 11)adr =RWADR11Q5;
   if(zw == 00)adr =RWADR00Q5;
   if(zw == 10)adr =RWADR10Q5;
   if(zw == 01)adr =RWADR01Q5;
}
if(k == 2){
   if(zw == 11)adr = RWADR11Q6;
   if(zw == 00)adr = RWADR00Q6;
   if(zw == 10)adr = RWADR10Q6;
   if(zw == 01)adr = RWADR01Q6;
}
if(k == 3){
   if(zw == 11)adr = RWADR11Q7;
```

```
    if(zw == 00)adr = RWADR00Q7;
    if(zw == 10)adr = RWADR10Q7;
    if(zw == 01)adr = RWADR01Q7;
}
        for(ii=0;ii<16;ii++){
                                adr = adr|((unsigned long)ii<<16);
                    if(xackpt('r',adr)){
                                        ki++;
                                        printf("\nadres pojawienia sie sygnalu XACK/ "%
                                        printf("%xhex\t przy instrukcji read\n",adr&
                                        j = 0;
                                 if((adr != RWADR11Q0)&&(adr != RWADR00Q0)&&
                                        (adr != RWADR10Q0)&&(adr != RWADR01Q0)&&
                                        (adr != RWADR11Q5)&&(adr != RWADR00Q5)&&
                                        (adr != RWADR10Q5)&&(adr != RWADR01Q5)&&
                                        (adr != RWADR11Q6)&&(adr != RWADR00Q6)&&
                                        (adr != RWADR10Q6)&&(adr != RWADR01Q6)&&
                                        (adr != RWADR11Q7)&&(adr != RWADR00Q7)&&
                                        (adr != RWADR10Q7)&&(adr != RWADR01Q7)){
printf("********************************** ^ BLAD ^ ***********************
**********\n");
                                         bld = 1;
                                 }
                                if(ki >7){
                                        printf("\n\tCzy testowac dalej t/n\n");
                                        j = 0;
                                        if(getch()=='t')ki = 0;
                                        else return 11;
                                }
                        }
    }/*petla for */
 }/* petla for <= 3 */
if(bld >0)return 11;
else return 0;
}
```

```
extern char enter_bufor[30];          /* bufor klawiatury funkcji scr_getc()
*/
extern int echo_on_off;               /* echo funkcji scr_getc() */
extern int lin_num;                   /* nr linii kursora */
extern int col_num;                   /* nr kolumny kursora */
extern int mem_line;
extern int rob_type;                  /* nr typu robota */
extern int test_type;                 /* nr testu */
extern int sek_num;                   /* nr przebiegu uruchomieniowego */
extern unsigned int adr_usz;             /* adres uszkodzenia( == -1 ok)  */
extern char it1_e6;                   /* stany wyjsc */
extern char it1_b2;                   /* stany wyjsc */
extern char it3_b3;                   /* stany wyjsc */
extern char it3_b5_pa;                /* stany wyjsc */
extern char it3_b5_pc;                /* stany wyjsc */
extern int buf_btmo;                 /* stan btmo */
extern int buf_mc42;                 /* stan przeciazenia mc42 : 00-OK 01-BLA
D */
extern int buf_welcfi;
extern void btmo();
extern void mc();
extern void int39();
```

```
extern char enter_bufor[30];          /* bufor klawiatury funkcji scr_getc()
*/
extern int echo_on_off;               /* echo funkcji scr_getc() */
extern void btmo();                      /* przerwanie brak XACK */
extern void mc();                     /* przerwanie przeciazenie MC42  */
extern void int39();                  /* przerwanie zwory Z8 (int 39) */
int lin_num;                    /* nr linii kursora */
int col_num;                    /* nr kolumny kursora */
int mem_line;
int rob_type;                   /* nr typu robota */
int test_type;                  /* nr testu */
int sek_num;                    /* nr przebiegu uruchomieniowego */
int buf_welcfi;                  /* stan zwory Z8 (przerw.39)ITP  */
unsigned int adr_usz;              /* adres przy ktorym wystopilo */
                          /* uszkodzenie ( < 0)nie ma uszkodzenia */
char it1_e6;                    /* stany wyjsc */
char it1_b2;                    /* stany wyjsc */
char it3_b3;                    /* stany wyjsc */
char it3_b5_pa;                 /* stany wyjsc */
char it3_b5_pc;                 /* stany wyjsc */
int buf_btmo;                  /* stan btmo */
int buf_mc42;                  /* stan przeciazenia mc42 : 00-OK 01-BLAD */
```

```
#define  bel   0x07
#define  bs    0x08
#define  tab   0x09
#define  lf    0x0a
#define  crr  0x0d
#define  esk   0x1b
#define  del   0x7f
#define  nul   0x00
#define  speace   0x20
#define TRUE    0xffff
#define FALSE   0
#define cr() (printf("\n\t\t\t\t\t\t\t\t\t* CR *\r"),getch())
#define blad() (printf("\n\t\t\t\t\t\t\t\t\t\t\t^BLAD^\n"))
#define OUT    outport
#define IN     inport
```

```
                    /* WEWY.H */
        /*TESTOWANIE ZESPOLU UKLADOW WEJSC/WYJSC */
   /* opracowal dr inz. Marian Wrzesien dnia 1988.11.26 */

# define WY0C 0x0
# define WY1C 0x2

# define WE0C 0x0
# define WE1C 0x2

# define WY0 0xf000
# define WY1 0xf001
# define WY2 0xf002
# define WY3 0xf003
# define WY4 0xf004

# define WE0 0xf000
# define WE1 0xf001
# define WE2 0xf002
# define WE3 0xf003
# define WE4 0xf004
# define WE5 0xf005
```

```
/*****************************************************************/
#define IOADR11    0x1717  /* adres pak. mw31 I/O zwora D2 (4-5) rozwarte "1"
i (3-6) rozwarte "1" */
#define IOADR00 0X1414  /* zwora D2 (4-5) zwarte (3-6) zwarte */
#define IOADR01 0X1616  /* (4-5) zwarte (3-6) rozwarte */
#define IOADR10 0X1515  /* (4-5) rozwarte (3-6) zwarte */

#ifdef ZAP

#define RWADR11Q0 (unsigned long)0X8E8EE /* adres pak mw31 MEMW/R sygnal na
 Q0 zwora D2 (4-5) rozw. (3-6) rozw. */
#define RWADR00Q0 (unsigned long)0X8EBEE /* syg. Q0   (4-5) ZWAR. (3-6) ZWA
R. */
#define RWADR01Q0 (unsigned long)0X8E9EE /* syg. Q0         ZWAR.        ROZ
W. */
#define RWADR10Q0 (unsigned long)0X8EAEE /* syg. Q0         ROZW.        ZWA
R. */

#define RWADR11Q5 (unsigned long)0x8E8E4 /* syg. Q5    (4-5) ROZW. (3-6) RO
ZW. */
#define RWADR00Q5 (unsigned long)0x8EBE4
#define RWADR01Q5 (unsigned long)0x8E9E4
#define RWADR10Q5 (unsigned long)0x8EAE4

#define RWADR11Q6 (unsigned long)0x8E8E2
#define RWADR00Q6 (unsigned long)0x8EBE2
#define RWADR01Q6 (unsigned long)0x8E9E2
#define RWADR10Q6 (unsigned long)0x8EAE2

#define RWADR11Q7 (unsigned long)0x8E8E0
#define RWADR00Q7 (unsigned long)0x8EBE0
#define RWADR01Q7 (unsigned long)0x8E9E0
#define RWADR10Q7 (unsigned long)0x8EAE0

#else

#define RWADR11Q0 (unsigned long)0XE8EE  /* adres pak mw31 MEMW/R sygnal na
 Q0 zwora D2 (4-5) rozw. (3-6) rozw. */
#define RWADR00Q0 (unsigned long)0XEBEE  /* syg. Q0   (4-5) ZWAR. (3-6) ZWA
R. */
#define RWADR01Q0 (unsigned long)0XE9EE /* syg. Q0         ZWAR.        ROZW
. */
#define RWADR10Q0 (unsigned long)0XEAEE /* syg. Q0         ROZW.        ZWAR
. */

#define RWADR11Q5 (unsigned long)0xE8E4 /* syg. Q5    (4-5) ROZW. (3-6) ROZ
W. */
#define RWADR00Q5 (unsigned long)0xEBE4
#define RWADR01Q5 (unsigned long)0xE9E4
#define RWADR10Q5 (unsigned long)0xEAE4

#define RWADR11Q6 (unsigned long)0xE8E2
#define RWADR00Q6 (unsigned long)0xEBE2
#define RWADR01Q6 (unsigned long)0xE9E2
```

```
#define RWADR10Q6 (unsigned long)0xEAE2

#define RWADR11Q7 (unsigned long)0xE8E0
#define RWADR00Q7 (unsigned long)0xEBE0
#define RWADR01Q7 (unsigned long)0xE9E0
#define RWADR10Q7 (unsigned long)0xEAE0

#endif

#define DAT0    0x1
#define DAT1    0x2
#define DAT2    0x4
#define DAT3    0x8
#define DAT4    0x10
#define DAT5    0x20
#define DAT6    0x40
#define DAT7    0x80
#define DAT8    0x100
#define DAT9    0x200
#define DAT10   0x400
#define DAT11   0x800
#define DAT12   0x1000
#define DAT13   0x2000
#define DAT14   0x4000
#define DAT15   0x8000
```

```
                    /*  <adr.h> */

# define IT1B1PA Oxfed8            /* 8255 */
# define IT1B1PB Oxfeda
# define IT1B1PC Oxfedc
# define IT1B1   Oxfede

# define IT1D6WY Oxfed0            /* 8212 */

# define IT1B3PA Oxfec8            /* 8255 */
# define IT1B3PB Oxfeca
# define IT1B3PC Oxfecc
# define IT1B3   Oxfece

# define IT1B2WY Oxfec0            /* 8212 */

# define IT3B6PA Oxfdc0            /* 8255 */
# define IT3B6PB Oxfdc2
# define IT3B6PC Oxfdc4
# define IT3B6   Oxfdc6

# define IT3B5PA Oxfdc8            /* 8255 */
# define IT3B5PB Oxfdca
# define IT3B5PC Oxfdcc
# define IT3B5   Oxfdce

# define IT3A6D1 Oxfdd0            /* 8251 */
# define IT3A6C1 Oxfdd2
# define IT3A6D2 Oxfdd4
# define IT3A6C2 Oxfdd6

# define IT3B3WY Oxfdd8            /* 8212 */

# define IT2B1C1 Oxfbd0            /* out,out 8212 */
# define IT2A1C2 Oxfbd0            /* in,in   8212 */
# define IT2B2B4 Oxfbc0            /* in,out  8212 */
# define IT2B3C4 Oxfbc8            /* out,out 8212 */
# define IT2A2C3 Oxfbc8            /* in,in   8212 */

# define MC42    Oxfbf6            /*         8212 */
```

```
            /* <fun.h> */

# define cr() (printf("\n\n\t\t\t\t\t\t\t\t\t* CR *\n"),getch())
# define blad() (printf("\n\t\t\t\t\t\t\t\t\t^BLAD^\n"),getch())
# define wy1(data) (outportw(MC42,data))    /* sterowanie zewn. z it2 */
# define wy2(data) (outportw(IT2B3C4,data))          /* WYL.B*/
# define wy3(data) (outportb(IT1B2WY,data))
# define wy4(data) (outportw(MW31dd,data))
# define czyt (c=mreadw(CZYT_A))
# define wyd (printf("stan=0x%x\n",mreadw(CZYT_A)))
# define in3 ((inportw(MC42)&uokmc42)>>8)
# define in4 (inportw(MC42)&utrprmc42)
# define in5 (inportb(IT1B1PA))                          /* odczyt zanegowany w IT1 */
# define pfin (in5&0x4)
# define pfsn (in5&0x8)
# define mpro (in5&0x2)
# define reset (in5&0x1)
# define outon (in5&0x40)
# define it2(data) (outportw(0xfbc8,data))
# define DR() (printf("Test przerwany,dokonaj naprawy.\
                       \n\n\t\t\t\t\t\t\t* CR *\n"),k=1)
# define map2 ((inportb(IT1B3PC)>>5)&0x1)
# define INIT (outportb(IT1D6WY,0x0))
# define NOTINIT (outportb(IT1D6WY,0x20))
# define UST1(adrmw31) (mreadw(adrmw31))
# define ZER1(adrmw31) (mwritew(adrmw31,0x0))
# define UST2(adrmw31) (mreadw(adrmw31))
# define ZER2(adrmw31) (mwritew(adrmw31,0x0))
# define CZYT_S(adrmw31) (mreadw(adrmw31))
# define CZYT_A(adrmw31) (mreadw(adrmw31))
# define odczyt (od=inportb(IT1B1PC))
# define INT2 ((od>>2)&0x1)
# define INT3 ((od>>3)&0x1)
# define INT4 ((od>>4)&0x1)
# define prz_1 (outportw(0xfbf6,0x40))
# define endprz_1 (outportw(0xfbf6,0x0))
# define wyl_al (outportw(IT2B3C4,0x13))
# define notwyl_al (outportw(IT2B3C4,0x17))
```