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

0711

A

Główny wykonawca dr inż. Marian Wrzesień

Wykonawcy mgr inż. Zbigniew Stańczak

Konsultant

Nr zlecenia RP-58.2
Zad. 3.2

Oprogramowanie użytkowe dla pakietów P1, P2 panelu programowania oraz MW-32. Weryfikacja oprogramowania użytkowego podczas testowania serii próbnej układów sterowania IRp-6/60. Wprowadzenie programów użytkowych testera do pamięci EPROM. Zweryfikowana dokumentacja dla wykonań jednostkowych urządzenia. Dokumentacja techniczno-ruchowa.

Zleceniodawca CPBR 7.1 "Roboty przemysłowe"

Pracę rozpoczęto dnia 01.12.1988 **zakończono dnia** 30.03.1989

Kierownik Zespołu
dr inż. P. Jabłoński

Z-a Dyr. d/s Automatyki
doc. dr inż. T. Gałązka

Kierownik Ośrodka
dr inż. B. Kontrymowicz

Praca zawiera:		**Rozdzielnik - ilość egz:**	
stron	2	**Egz. 1**	BOINTE
rysunków	–	**Egz. 2**	PPDiM
fotografii	–	**Egz. 3**	– OAE
tabel	–	**Egz. 4**	OAE
tablic	–	**Egz. 5**	~~ZAP~~
załączników	1 (146 s.)	**Egz. 6**	–
Nr rejestr.	6230		

A

Analiza deskryptorowa

URZADZENIA AUTOMATYCZNEJ REGULACJI I STEROWANIA: ROBOTY PRZEMYSLOWE, TESTOWANIE.

Analiza dokumentacyjna

Sprawozdanie zawiera wykaz prac wykonanych w zadaniu 3.2. tematu Rp-58.2. Do sprawozdania załączono wersję źródłową oprogramowania urządzenia do uruchomienia i testowania pakietów układu sterowania robotów przemysłowych IRp-6/60.

Tytuły poprzednich sprawozdań

Zlecenie UR-01.03.01 K:
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).
Zadanie 3.1: Opracowanie i wykonanie urządzeń dla uruchomienia i testowania układów sterowania IRp-6/60 (nr spr. 6189).

UKD 338.45:62/69].002.1/2 Roboty przemysłowe, systemy sterowania
681.3.02
PIAP 41/88 10000

SPIS TRESCI.

1. Wstep.

Niniejsze sprawozdanie stanowi podsumowanie prac realizowanych w temacie RP-58.2 celu 58, pt. "Urzadzenia do testowania i diagnostyki układów sterowania oraz podzespołów układu sterowania robotów przemysłowych IRp. Zawiera ono wykaz prac wykonanych w ramach tematu RP-58.2 oraz oprogramowanie pełne urzadzenia do uruchamiania i testowania pakietów układów sterowania robotów przemysłowych.

2. Wykaz prac wykonanych w ramach zadania 3.2 tematu RP-58.2.

W ramach zadania 3.2 wykonano :
- badania czystości patentowej - sprawozdanie nr rej. 6259,
- projekt dokumentacji techniczno-ruchowej - sprawozdanie nr rej. 6260,
- dokumentacje techniczna - nr arch. 4668,
- oprogramowanie do testowania i uruchamiania pakietu MW-32 oraz panelu programowania.

3. Oprogramowanie pełne dla urzadzenia do uruchamiania i testowania pakietów układu sterowania robotów przemysłowych.

Oprogramowanie pełne zawiera następujące programy
- program dla wyboru pakietu testowanego,
- program uruchomieniowy pr_ar,
- pakiet programów do testowania pakietu MC-42,
- pakiet programów do testowania pakietu MW-31,
- pakiet programów do testowania pakietu MW-32,
- pakiet programów do testowania pakietu MZ-70,
- pakiet programów do testowania pakietu ML-50,
- pakiet programów do testowania pakietu MI-50,
- pakiet programów do testowania pakietu MA 70,
- pakiet programów do testowania panelu programowar a,
- funkcje biblioteczne,
- zbiory typu header.

4. Uwagi dotyczace wdrozenia urzadzenia do uruchamiania i testowania pakietów w PPDiM.

- Zgodnie z wczesniejszymi ustaleniami zostanie przeprowadzone szkolenie pracowników PPDiM, zapoznajace ich z :
  = budowa i działaniem urzadzenia,
  = oprogramowaniem do testowania kazdego z wyzej wymienionych pakietów oraz sposobem przeprowadzania testów,
  = programem uruchomieniowym, umozliwiajacym konstruowanie własnych podprogramów wspomagajacych uruchamianie pakietów,
  W czasie szkolenia zostana zebrane uwagi, które beda uwzglednione w koncowej wersji DTR.
- W przypadku wykrycia niedogodnosci (w czasie uzytkowania urzadzenia) w programach do testowania pakietów, zostanie wprowadzona korekta do oprogramowania.
- W przypadku zainteresowania PPDiM produkcja urzadzenia do uruchamiania i testowania pakietów, OAE bedzie sprawowało nadzór autorski nad wykonaniem tego urzadzenia.

Załącznik 1

OPROGRAMOWANIE

URZĄDZENIA DO URUCHAMIANIA I TESTOWANIA PANELI UKŁADU STEROWANIA ROBOTÓW PRZEMYSŁOWYCH

WARSZAWA 1989

PROGRAM

DLA WYBORU PARAMETRU TECTOWANEGO

```
/*                      opracowal:  Zbigniew Stanczak                      */

#include <ascii.h>
#include <proext.h>
#include <proto.h>

int menugl()
{
/* ************************** menu ******************************************** */
static char * naglowek[2] ={
        "* menu  *",
"                       WYBOR  PAKIETU ",
};
static char * text[]={
        "PROGRAMY NARZEDZIOWE",
        "MC42",
        "MW31",
        "MW32",
        "MZ70",
        "ML50",
        "MT50",
        "MA70",
        "PANEL PROGRAMOWANIA",
};
char * podpis = "                                                    * spacja * del * cr *";
int line_n = 9;
static int line_1[9]={20,4,4,4,4,4,4,4,4,19};
/* ************************** menu ******************************************** */

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

```
#include <ascii.h>
#include <tstglb.h>
#include <proto.h>
/*-------------------------------------------------------------- */
main(){
int skok = 0;
ster_prz();
initmm86();
stout(esk);
stout('6');
setintad(3,0xff00,0x97);/* MON86 */
setintad(10,0xd000,0x0);/* menugl */
setintad(11,0xd100,0x0);/* mc42 */
setintad(12,0xd400,0x0);/* mw31 */
setintad(13,0xd900,0x0);/* mw32 */
setintad(14,0xde00,0x0);/* mz70 */
setintad(15,0xe000,0x0);/* ml50 */
setintad(16,0xe100,0x0);/* mi50 */
setintad(17,0xe200,0x0);/* ma70 */
setintad(18,0xe700,0x0);/* panel prog.*/
setintad(19,0xec00,0x0);/* pr_ur */
for(1;1;1){
skok = menugl();
/*-------------------------------------------------------------- */
switch(skok){
case 1:mem_line=0;
       wybprur();
       mem_line=0;
       break;


case 2:/* mc42 */
#asm
        int 11
#endasm
       break;


case 3:/* mw31 */
#asm
        int 12
#endasm
       break;


case 4:/* mw32 */
#asm
        int 13
#endasm
       break;


case 5:/* mz70 */
#asm
        int 14
#endasm
       break;


case 6:/* ml50 */
#asm
        int 15
#endasm
       break;


case 7:/* mi50 */
#asm
        int 16
#endasm
       break;


case 8: /* ma70 */
#asm
        int 17
#endasm
       break;


case 9:/* panel programowania */
#asm
        int 18
#endasm
       break;
default: printf("bledna wartosc zwracana przez menu");
       getch();
       break;
}
/*-------------------------------------------------------------- */
}/* for */
}/* main */

wybprur()
{
int k ;
while(k = menuprur()){
/*-------------------------------------------------------------- */
switch(k){
case 1:/* pr_ur */
#asm
```

```
        int 19
#endasm
        break;
case 2:/* MON86 */
scr_clr();
delayp(1000);
#asm
        int 3
#endasm
        break;
default:return 0;
  }
}
}
int menuprur()
{
/* ************************** menu ******************************************** */
static char * naglowek[2] ={
        "* menu 1 *",
"                        WYBOR PROGRAMOW NARZEDZIOWYCH"
};
static char * text[]={
        "projektowalny program uruchomieniowy",
        "program MON86",
        "glowne menu",
};
char * podpis = "                                                    * spacja * del * cr *";
int line_n = 3;
static int line_l[3]={36,13,11};
/* ************************** menu ******************************************** */

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

/* initmm86.asm */

```
finclude lmacros.h
extrn delay_:near
;  ustawienie 8253 8251(4800) na pakiecie MM86
procdef initmm86

mov     ax,36h
out     0cfh,al
mov     al,16       ;4800       2400 - 31
out     0c9h,al
mov     al,0
out     0c9h,al

mov     al,76h
out     0cfh,al
mov     ax,9441h
out     0cbh,al
mov     al,148
out     0cbh,al
mov     cx,10
call    delay_


mov     al,82h ;rodzaj pracy
out     0dah,al

push    cx
mov     cx,10
call    delay_
mov     al,40h  ;internal reset
out     0dah,al
call    delay_
mov     al,0fah
out     0dah,al
call    delay_
mov     al,17h
out     0dah,al
call    delay_
pop     cx

pret
pend initmm86
finish
end
```

PROGRAM

URUCHOMIENIOWY

PR_UR

/* program uruchomieniowy PR_UR.C */

```
#include <ascii.h>
#include <proto.h>
#include <tstgib.h>
#define NULL ((void *)0)
#define MAX 50
typedef struct{
        long adr;
        int dana;
}KK;
typedef struct{
        char ro,byte;
        long adr;
        int dana;
        KK * p;
}KO;

KO TAB[MAX];
KK TABK[MAX];
static int krok,x,a,p,mind;
static void hlp(void);
static void raport(void);
static void lraport(int index);

pr_ur(){
int c,in=0,xack;
    printf("\n o (outport) i (inport) w (write) r (read) b (byte) * (sekwencja) k (koniec)");
    printf("\n\n\t\tADRES(hex)\t\t\tDANA(hex)\n");
        mind=0;
        TAB[in].byte=TAB[in].ro=0;
        TAB[in].adr=01;
        TAB[in].dana=0;
        TAB[in].p=NULL;

    while((c=getch()) != 'k'){
        if(c==esk)return 0;
        if(c=='b') TAB[in].byte=1;
        if(c=='*') TAB[in].p=&TABK[in];
        if((c=='o')||(c=='i')||(c=='w')||(c=='r')) TAB[in].ro=c;
        if((TAB[in].ro!=0)&&(c==crr)){
          if(TAB[in].p == NULL){
                printf("\nPodaj adres (hex)\t");
                scanf("%lx",(int *)&TAB[in].adr);
                if((TAB[in].ro == 'w')||(TAB[in].ro == 'o')){
                        printf("\nPodaj dana (hex)\t");
                        scanf("%x",&TAB[in].dana);
                        scr_curu();
                        stout(crr);
                }
          }
          else {

                printf("\nPodaj adres poczatkowy (hex)\t");
                scanf("%lx",(int *)&TAB[in].adr);
                printf("\nPodaj adres koncowy (hex)\t");
                scanf("%lx",(int *)&TAB[in].p->adr);
                if((TAB[in].ro == 'w')||(TAB[in].ro == 'o')){
                        printf("\nPodaj dana poczatkowa (hex)\t");
                        scanf("%x",&TAB[in].dana);
                        printf("\nPodaj dana koncowa (hex)\t");
                        scanf("%x",&TAB[in].p->dana);
                        scr_curu();
                        scr_curu();
                        stout(crr);
                }
                scr_curu();
                stout(crr);
          }
          scr_eos();
          lraport(in);
          ++in;
          TAB[in].byte=TAB[in].ro=0;
          TAB[in].adr=01;
          TAB[in].dana=0;
          TAB[in].p=NULL;
        }/* end crr */
    }/* end while */
    mind=--in;
    printf("\nCzy wykonywac petle krokowo\t\t\t\t\t\t*(t/n)*");
    c=getch();
    scr_idel();
    krok=0;
    if(c=='t'){
      krok=*krok;
      printf("\t\tPraca krokowa");
    }
    c=getch();
    while((c=scr_pool())!=esk){
    int cc,dd;
    long ad;
      for(in=0;in<=mind;in++){
        long war;
        int war2;
        ad=TAB[in].adr;
        dd=TAB[in].dana;
        if(!p)printf("\n%c%c\t",TAB[in].ro,TAB[in].byte?'b':' ');
        war=(TAB[in].p==NULL)?TAB[in].adr:TAB[in].p->adr;
        war2=(TAB[in].p==NULL)?TAB[in].dana:TAB[in].p->dana;
        for(ad=TAB[in].adr;ad <= war;ad++){
        for(dd=TAB[in].dana;dd <= war2;dd++){
         if(!p&&(TAB[in].p!=NULL))printf("\n\t");
         if(!x){
           switch(TAB[in].ro){
```

```
                case 'i':if(TAB[in].byte)cc=0xff&inportb((int)ad);
                         else cc=inportw((int)ad);
                        if(p){stout(crr);stout(lf);}
                        if(!m)printf("%x",cc);
                         break;
                case 'r':if(TAB[in].byte)cc=0xff&mreadb(ad);
                         else cc=mreadw(ad);
                        if(p){stout(crr);stout(lf);}
                        if(!m)printf("%x",cc);
                         break;
                case 'c':if(TAB[in].byte)outportb((int)ad,(char)dd);
                         else outportw((int)ad,dd);
                         break;
                case 'w':if(TAB[in].byte)mwriteb(ad,(char)dd);
                         else mwritew(ad,dd);
                         break;
              }
            }else{
              switch(TAB[in].ro){
                case 'i':if(TAB[in].byte){xack=xackpt('bi',ad,&cc);cc&=0xff;}
                         else xack=xackpt('i',ad,&cc);
                        if(p){stout(crr);stout(lf);}
                        if(!m)printf("%x",cc);
                         break;
                case 'r':if(TAB[in].byte){xack=xackpt('br',ad,&cc);cc&=0xff;}
                         else xack=xackpt('r',ad,&cc);
                        if(p){stout(crr);stout(lf);}
                        if(!m)printf("%x",cc);
                         break;
                case 'o':if(TAB[in].byte)xack=xackpt('bo',ad,&dd);
                         else xack=xackpt('o',TAB[in].adr,&dd);
                         break;
                case 'w':if(TAB[in].byte)xack=xackpt('bw',ad,&dd);
                         else xack=xackpt('w',ad,&dd);
                         break;
              }
              if(0x1&xack)printf("\t%ACK%");
            }
            if(krok!=0)c=getch();
            else c=scr_pool();
               if(c!=0){
                    if(c=='h'){ hlp();continue;}
                    if(c=='r'){ raport();continue;}
                    if(c=='x')x=~x;
                    if(c=='m')m=~m;
                    if(c=='p')p=~p;
                    if(c=='k')krok=~krok;
                    if(c==esk)return 0;
               }
            if(!TAB[in].byte)++dd;
           }/*end for dd */
            if(!TAB[in].byte)++ad;
           }/*end for ad */
/*         if(krok!=0)c=getch();
           else c=scr_pool();
              if(c!=0){
                   if(c=='h')hlp();
                   if(c=='r')raport();
                   if(c=='x')x=~x;
                   if(c=='m')m=~m;
                   if(c=='p')p=~p;
                   if(c=='k')krok=~krok;
                   if(c==esk)return 0;
              }
*/
        }/* end for */
      }/* end while */

}/* end pr_ur */

static void hlp(void){
 printf("\nwcisniecie: p - informacje pomocnicze w trakcie wyk. petli");
 printf("\n            k - zmiana pracy krokowej na automatyczna i odwrotnie");
 printf("\n            m - wstrzymanie i wznowienie wydruku danych na monitor");
 printf("\n            x - informacja o pojawianiu sie sygnalu /ACK/ pak.testowanego");
 printf("\n            r - raport o projekcie programu uruchomieniowego\n");
}
static void raport(void){
int ind;
printf("\n#################################################################################");
printf("\n\t\tADRES(hex)\t\t\tDANA(hex)\n");
for(ind=0;ind <= mind;ind++){stout(crr);stout(lf);lraport(ind);}
printf("\n#################################################################################");
}
static void lraport(int ind){
 printf("%c%c\t\t",TAB[ind].ro,TAB[ind].byte?'b':' ');
 printf("%5lx\t",TAB[ind].adr);
 if(TAB[ind].p!=NULL)printf("(%5lx)",TAB[ind].p->adr);
 if((TAB[ind].ro=='w')||(TAB[ind].ro=='o')){
   printf("\t\t\t%4x\t",TAB[ind].dana);
   if(TAB[ind].p!=NULL)printf("(%4x)",TAB[ind].p->dana);
 }
}
```

PAKIET PROGRAMÓW

DO TESTOWANIA PAKIETU MS 42

```
/*                      opracowal:  Zbigniew Stanczak                       */

#include <ascii.h>
#include <proext.h>
#include <proto.h>
int menumc42()
{
/* ************************** 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:  Zbigniew Stanczak                    */
#include <proto.h>
#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;

/* ************************* menu ********************************************* */
static char * naglowek[2] ={
        "* menu 1 *",
"                       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_1[7]={14,33,21,13,21,27,11};
/* ************************* menu ********************************************* */

while((k = menu(naglowek,text,podpis,line_n,line_1))/* != (int)8*/ ){
    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){
             printf("calkowity test mc42\t\t\t\t\t\t\tOK!");
             cr();
             return ;
          }
printf("calkowity test mc42");
blad();
cr();
return ;
}
```

```
#include <tstglb.h>
main(){
mc42();
#asm
int 10
#endasm
}
```

```
/*                      opracował:  Zbigniew Stanczak                    */
#include <ascii.h>
#include <proto.h>
#include <mc42.h>
#include <adr.h>

outon(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
char *p;
unsigned adr1,adr2;
inic_it2();
koc(7);
scr_clr();
 if(xackpt('i',UL ADRR))adr1= ADRR;
   else if(xackpt('i',UL ADRZ))adr1= ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= ADRZ;
            scr_clr();
        }
dwe= 0x0000;
outportb(IT1D&WY,(char)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;
  printf("\ndana zapisywana do pakietu (negacja)\t");
  printf("%s",ltob((long)dwe,(int)16));
  printf("\ndana na laczu obiektowym\t\t");
  printf("%s",ltob((long)dwy,(int)16));
  blad();
  printf("\nblad zapisu, brak testowalnosci sygnalu OUTON/\n");
  cr();
  return -1;
}

outportb(IT1D&WY,(char)0x0); /* zabr. syg. OUTON */
delayp(10000);

outportw(UI adr1,*dwe);
delayp(5000);
dwy=inportw(UI ADRTES);
if(dwy == 0){
  bl=1;
  printf("\ndana zapisywana do pakietu (negacja)\t");
  printf("%s",ltob((long)dwe,(int)16));
  printf("\ndana na laczu obiektowym \t\t");
  printf("%s",ltob((long)dwy,(int)16));
}

outportb(IT1D&WY,(char)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\tOK!\n");
cr();
return 0;
}
```

```
/*                      opracowal:  Zbigniew Stanczak                    */
#include <mc42.h>
#include <ascii.h>
#include <proto.h>
unsigned long adr1,adr2;

dekmc42(){
unsigned adr;
int k,j,blc=0,bl=0,i=0;
char * p;
 cli();
 scr_clr();
 printf("\n\tTEST DEKODERA ADRESOW\n");
 if(kackpt('o',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= UI ADRR;
 adr2= UI ADRZ;
 if(!kackpt('o',UL ADRR))if(!kackpt('o',UL ADRZ)){
                            adr1= UI ADRZ;
                            adr2= UI ADRR;
                            printf("%s",wy2);
                            blad();
                            cr();
                            return -1;
                          }
printf("\ntest pozytywny dekodera\t\t\t\t\t\t\t\tOK!\n");
printf("\n\tCzy wykonac test calkowity dekodera\n");
printf("\tprzy roznych stanach przelacznika AP ?\t\t\t\t*(t/n)*\n");
if(getch()== 't'){
 for(k=0;k<32;k++){
  scr_clr();
  printf("\nustaw przelacznik AP w pozycji:\n");
  p=itoa(UL k,(int)5);
  for(j=0;j<5;j++){
     printf("\n\tAP%d\t",j);
     if(*p++)-'0')printf("<-\n");
     else printf("->\n");
  }
  printf("\n( <- zwarcie      -> rozwarcie)\n");
  if(cr()==esk)break;
  for(bl=i=0,adr = 0l;adr< 256;++adr){
   if(kackpt('i',UL adr)){
     printf("adres pojawienia sie syg. XACK/\t");
     ++i;
     printf("%x hex\n",UI adr);
     if((adr != adrmc42[k])&&(adr != (adrmc42[k]+1))){
       bl=blc=1;
       blad();
       stout(bel);
     }
     else bl=-1;
     if(i>5)
       if(cr() == esk){
         i=0;
         return -1;
       }
   }
  }
  if((bl==0)&&(k!=3)&&(k !=11)&&(k != 19)&&(k != 27)){
     blc=1;
     blad();
     stout(bel);
  }
  if(cr()==esk)break;
 }
if(blc > 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\tOK!\n");
cr();
}/*if*/
return 0;
}/*main*/
```

```
/*                      opracował:  Zbigniew Stanczak                   */

#include <ascii.h>
#include <proto.h>
#include <mc42.h>

odczyt(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
char *p;
unsigned adr1,adr2;
inic_it2();
koc(7);
scr_clr();
 if(xackpt('i',UL ADRR))adr1= ADRR;
   else if(xackpt('i',UL ADRZ))adr1= ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= ADRR;
                          else if(k=xackpt('i',UL ADRZ))adr1= 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;
      printf("\ndana wejsciowa na laczu obiektowym\t");
      printf("%s",ltob(UL dwe,(int)16));
      printf("\ndana odczytana z pakietu\t\t\t");
      printf("%s",ltob(UL dwy,(int)16));
      blad();
    }
    if(lk > 4){
      lk=0;
      if(cr() == esk)break;
    }
}
if(bl)0){
  printf("\nblad w obwodzie odczytu  DAT(N)-WE(N+1)");
  blad();
  cr();
  return -1;
}
printf("\nodczyt z pakietu\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

```
/*                      opracowal:  Zbigniew Stanczak                    */

#include <ascii.h>
#include <proto.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;
char *p;
unsigned adr1,adr2;
inic_it2();
outportb(IT191,(char)0x9b);
koc();
scr_clr();
 if(xackpt('i',UL ADRR))adr1= ADRR;
   else if(xackpt('i',UL ADRZ))adr1= ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= ADRZ;
            scr_clr();
        }
outportb(IT1D6WY,(char)0x10);
delayp(1000);
printf("\n\t\tTEST DETEKTORA PRZECIAZEN\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;
      printf("\ndana zapisywana do pakietu (negacja)\t");
      printf("%s",ltob(UL dwe,(int)16));
      printf("\ndana na laczu obiektowym \t\t");
      printf("%s",ltob(UL dwy,(int)16));
      blad();
      printf("\nbrak testowalnosci detektora przeciazen na pozycji %d\n",i+1);
      if(cr()==esk)break;
      continue;
    }
    if(lk > 4){
      lk=0;
      if(cr() == esk)break;
    }

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

    if(int2() == 0){
      lk++;
      printf("\nza niska granica przeciazenia na pozycji %d\n",i+1);
      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 przeciazen 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("\nblad zespolu detektora przeciazen INT2/ \n");
  blad();
  cr();
  return -1;
}
printf("\ndetektor przeciazen\t\t\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

```
/*                      opracowal:  Zbigniew Stanczak                    */

#include <ascii.h>
#include <proto.h>
#include <mc42.h>
#include <adr.h>


zapis(){
unsigned int k=0,i,dwy,dwe,lk=0,bl=0;
char *p;
unsigned adr1,adr2;
inic_it2();
koc(7);
scr_clr();
 if(xackpt('i',UL ADRR))adr1= ADRR;
   else if(xackpt('i',UL ADRZ))adr1= ADRZ;
        else{
            printf("\nustaw przelacznik AP w polozeniu skrajnym\n");
            while(!k)if(k=xackpt('i',UL ADRR))adr1= ADRR;
                        else if(k=xackpt('i',UL ADRZ))adr1= ADRZ;
            scr_clr();
        }
outportb(IT1D&WY,(char)0x10);
delayp(1000);
printf("\n\tTEST ZAPISU DO PAKIETU\n");
for(i=0;i<16;i++){
    dwe=(unsigned int)pelz!(i);
    outportw(UI adr1,dwe);
    delayp(5000);
    dwy=inportw(UI ADRTES);
    if(dwe != dwy){
      ++lk;
      bl=1;
      printf("\ndana zapisywana do pakietu (negacja)\t");
      printf("%s",ltob(UL dwe,(int)16));
      printf("\ndana na laczu obiektowym \t\t");
      printf("%s",ltob(UL dwy,(int)16));
      blad();
    }
    if(lk > 4){
      lk=0;
      if(cr() == esk)break;
    }
}
lk=0;
if(bl>0){
  printf("\nblad w obwodzie zapisu  DAT(N)-WY(N+1)");
  blad();
  cr();
  return -1;
}
printf("\nzapis do pakietu\t\t\t\t\t\t\tOK!\n");
cr();
return 0;
}
```

PAKIET PROGRAMOW

DO TESTOWANIA PAKIETU MV 31

```
#include <ascii.h>
#include <proext.h>
#include <protc.h>

int menumw31()
{
/* ************************* menu ********************************************** */
static char * naglowek[2] ={
        "* menu I *",
"                          TEST PAKIETU MW31"
};
static char * text[]={
        "test calkowity",
        "dekoder adresow",
        "blok kontroli przesylu po magistrali",
        "uklad przerwan",
        "uklad kontroli zasilania",
        "glowne menu"
};
char * podpis = "                                                    * spacja * del * cr *";
int line_n = 6;
static int line_1[6]={14,15,36,14,24,11};
/* ************************* menu ********************************************** */

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

```
#include <ascii.h>
#include <tstglb.h>
#include <profo.h>
void autotest(void);

main(){
int k ;
/*inic_it2();
i_it1_mw();*/
while(k = menumw31()){
/*------------------------------------------------------------ */
switch(k){
case 1:autotest();
       break;
case 2: dekmw31();
        mmap2();
       break;
case 3: bkm();
       break;
case 4:up() ;
      intmw31();
       break;
case 5:ukz();
       break;

default:
{
#asm
        int 10
#endasm
};
       break;
  }
}/* end while */
/*------------------------------------------------------------ */
}/* end mw31 */

/* ********************************************************************** */
void autotest(void)
{
if(dekmw31()==0)
  if(mmap2()==0)
     if(bkm()==0)
       if(up()==0)
         if(intmw31()==0)
           if(ukz()==0){
             printf("\ncalkowity test pakietu\t\t\t\t\t\t\t\tOK!");
             cr();
             return ;
           }
 printf("\ncalkowity test pakietu");
 blad();
 cr();
 return;
}
```

```
                        /* DEKMW31.C*/
                /* TESTOWANIE DEKODERA ADRESOW */
        /* Opracowal mgr inz. Z.Stanczak dnia 01-12-1988 */
#include <adr.h>
#include <proext.h>
#include <adrmw31.h>
#include <ascii.h>
#include <proto.h>
/* ********************************************************************** */
static char KOM0O[]=
{
"\n\t\tpodczas testowania I/O instrukcja out"
};
/* ********************************************************************** */
static char KOM0I[]=
{
"\n\t\tpodczas testowania I/O instrukcja in"
};
/* ********************************************************************** */
kom0r()
{
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 : ");
}
/* ********************************************************************** */
kom0w()
{
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);
}
/* ********************************************************************** */
komr0()
{
printf("\n\t\t\t\t\t%lxhex",RWADR11Q0);
printf("\n\t\t\t\t\t%lxhex",RWADR00Q0);
printf("\n\t\t\t\t\t%lxhex",RWADR10Q0);
printf("\n\t\t\t\t\t%lxhex",RWADR01Q0);
}
/* ********************************************************************** */
komr5()
{
printf("\n\t\t\t\t\t%lxhex",RWADR11Q5);
printf("\n\t\t\t\t\t%lxhex",RWADR00Q5);
printf("\n\t\t\t\t\t%lxhex",RWADR10Q5);
printf("\n\t\t\t\t\t%lxhex",RWADR01Q5);
}
/* ********************************************************************** */
komr6()
{
printf("\n\t\t\t\t\t%lxhex",RWADR11Q6);
printf("\n\t\t\t\t\t%lxhex",RWADR00Q6);
printf("\n\t\t\t\t\t%lxhex",RWADR10Q6);
printf("\n\t\t\t\t\t%lxhex",RWADR01Q6);
}
/* ********************************************************************** */
komr7()
{
printf("\n\t\t\t\t\t%lxhex",RWADR11Q7);
printf("\n\t\t\t\t\t%lxhex",RWADR00Q7);
printf("\n\t\t\t\t\t%lxhex",RWADR10Q7);
printf("\n\t\t\t\t\t%lxhex",RWADR01Q7);
}
/* ********************************************************************** */
/* ********************************************************************** */
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(9)\n");
```

```
printf("\t\t\tD2(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(05)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tD3(12)\n");
printf("\t\t\tC3(8)\n");
}

/* ********************************************************************** */
kom5()
{
printf("\n\t\tBrak sygnalu UST.1/ uszkodzony E8(06)");
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(07)");
printf("\n\t\tlub uszkodzone uklady :\n");
printf("\t\t\tD3(8)\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;
/* scr_clr();
 printf("\nCzy testowany pakiet ma negator na linii ADR19/ - E12(11)  %(t/n")%);
 if(getch() == 't'){
 }
*/
 scr_clr();
 if((k = dekadrp()) == 0){
                printf("Test pozytywny dekodera adresow\t\t\t\t\tOK!");
                printf("\n(przy aktualnym stanie zwory D2)\n");
                printf("\nCzy rozpoczac test calkowity dekodera adresow(w przestreni 1 Mb)\t\t%(t/n)
                if(getch() == 't'){
                                scr_clr();
                                printf("\n\tTest calkowity dekodera adresow");
                                printf("\n\t(przy aktualnym stanie zwory D2)\n");
                                if((k = dekadrn()) == 0){

                                                printf("\n\t\tTest calkowity dekodera adresow\i
                                                getch();
                                                return 0;
                                 }
                }
                else return 0;
 }
 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",&KOM00);
       komio();
       kom1();
                        break;
                case 2:
       printf("%s",&KOM0I);
       komio();
       kom2();
                        break;
                case 3:
       kom0r();
       komr0();
       kom3();
                        break;
                case 4:
       kom0r();
       komr5();
```

```
        kom4();
                        break;
                case 5:
        kom0r();
        komr6();
        kom5();
                        break;
                case 6:
        kom0r();
        komr7();
        kom6();
                break;
                case 7:
        kom0w();
        komr0();
        kom7();
                        break;
                case 8:
        kom0w();
        komr5();
        kom8();
                        break;
                case 9:
        kom0w();
        komr6();
        kom9();
                        break;
                case 10:
        kom0w();
        komr7();
        kom10();
                        break;
                case 11:
                        break;
                default:
                        printf("Bledny stan sygnalu XACK/ na pakiecie IT1 \n");
                        break;
        }
 getch();
 return -1;
}

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

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

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

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

                if(!xackpt('i',(long)IOADR11))
                if(!xackpt('i',(long)IOADR00))
                if(!xackpt('i',(long)IOADR10))
                if(!xackpt('i',(long)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',RWADR01Q5))
                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,blad = 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/ %lx",adr);
                                printf("hex\t przy instrukcji in\n");
                                j=0;
                                if((((int)adr != IOADR11)&&((int)adr != IOADR00)&&
                                  ((int)adr != IOADR10)&&((int)adr != IOADR01)){
                                   printf("************************************ ^ BLAD ^ *****
                                   blad = 1;
                                }
                                if(ki >5){
                                     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/ %lx",adr);
                                printf("hex\t przy instrukcji read\n");
                                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");
                                blad = 1;
                               }
                        }
                       if(ki >5){
                          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',(long)IOADR11))   adr = RWADR11Q0,zw=11;
        if(xackpt('i',(long)IOADR00)) adr = RWADR00Q0,zw=00;
        if(xackpt('i',(long)IOADR10)) adr = RWADR10Q0,zw=10;
        if(xackpt('i',(long)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/ %lx",adr);
                                printf("hex\t przy instrukcji read\n");
                                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");
                                blad = 1;
                               }
                               if(ki >5){
                                  printf("\n\tCzy testowac dalej t/n\n");
```

```
                                        j = 0;
                                        if(getch()=='t')ki = 0;
                                        else return 11;
                                    }
                            }
        }/*petla for */
 }/* petla for <= 3 */
if(blad >0)return 11;
else return 0;
}
                                     /*koniec*/
```

```
/*1998-12-01*/
                        /* 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>
# include <proto.h>
mmap2()
  {
    unsigned int k;
    scr_clr();
    printf("\nSYGNAL MMAP2");
    outportb(IT1B3,(char)0x8b); /* zaprogramowanie bramy IT1B3 */
    outportb(IT1D6WY,(char)ZER);        /* reset C1 IT1 */
    outportb(IT1D6WY,(char)NOTZER);     /* wycofanie resetu */
    if(map2)
      {
        printf("\nUsterka w zespole odczytujacym testera!");
        blad();
        return(-1);
      }
    mreadw(0x0e8ee1);
    mreadw(0x0ebee1);
    mreadw(0x0e9ee1);
    mreadw(0x0eaee1);
    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*/
```

```
                        /*    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 0x9000
# 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>
# include <proto.h>
  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]=
   {
     "przerwanie 3",
     "przerwanie 2",
     "przerwanie 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,(char)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\t\t\t\t\t\t\t*(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\t* CR *\n");  /* !! to ma byc !! */
     getch();
     j=1;
     while(j<9)
      {
        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;
            break;
           case 7:                                  /* Kout * 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*/
```

```
                        /*      UKZ.C   */
        /*TESTOWANIE UKLADU KONTROLI ZASILAN PAKIETU MW-31 */
        /*opracował dr inż Marian Wrzesień dnia 1988-09-21  */
# define p_reset 0xff0f                         /* maska P_RESET/ */
# define lock 0xff1d                            /* maska LOCK/ */
# define uokmc42 0x100                          /* maska na odczyt w MC42 */
# define utrpmc42 0x1                           /*      j.w.              */
# include <fun.h>
# include <adr.h>
# include <adrmw31.h>
# include <proto.h>
ukz()
 {
  int k,p,m;
  k=p=m=0;
  scr_clr();
  printf("\n\t\t\tUKLAD KONTROLI ZASILAN\n");
  outportb(IT1B1,(char)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\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                  */

      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 potencjometram: P3,...,P7\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 UOK5VB:\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.UTR.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);
  if(pfsn)(wy2(v[5]&lock));                     /* proba ustawienia PFSN/ */
  pfsn?(m=1,printf("\n\nSygnal PFSN/ nie da sie ustawic sygnalem LOCK/\n")):(m=0);
  if(mpro)(wy2(v[5]&p_reset));else delayp(1000);        /* 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 LOCK/=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 napiecia w UOK\n"));
  wy2(v[5]&p_reset);                            /* reset B2(8) */
  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*/
```

```
                        /* BKM.C */
        /*TESTOWANIE BLOKU KONTROLI PRZESYLU PO MAGISTRALI */
        /* Opracowal mgr inz. Z. Stanczak dnia  01-12-1988 */
#include <adr.h>
#include <adrmw31.h>
#include <proto.h>
bkm(){
scr_clr();
if(!xackpt('i',(long)IOADR11))
printf("\n\tUstaw zwore D2 w pozycii rozwartej(polozenie dolne)");
while(!xackpt('i',(long)IOADR11));
scr_clr();
outportb(IT1D6WY,(char)0x20);
outportb(0xd9,(char)0xffff);
outportw(IOADR11,0);
inportw(IOADR11);
pokew(RWADR1100,0xffff);
if(!(peekw(RWADR1107)&DAT7)){
        scr_clr();
        printf("\nUKLAD KONTROLI MAGISTRALI\t\t\t\t\t\tOK!");
        getch();
        return 0;
}
else{
        scr_clr();
        printf("\n\t\t\tUszkodzenie ukladu kontroli magistrali \n");
        printf("\nRozpoczac program uruchomieniowy ukladu kontroli magistrali ?\t\t\t\t\t#(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/ MWTC/ IOWC/.");
        printf("\nZ tym ze sygnal MRDC/ umozliwia odczyt wyjscia przerzutnika D8(8),");
        printf("\nstan tego wyjscia jest wyswietlany ponizej 0 - wartosc prawidlowa");
        printf("\n\n\n\t\t\t");
                while(1){
                   outportb(0xd9,(char)0xffff);
                   outportw(IOADR11,0);
                   inportw(IOADR11);
                   pokew(RWADR1100,0xffff);
                   if(peekw(RWADR1107)&DAT7)stout(0x31);
                   else stout(0x30);
                   stout(0x8);
                   if(sgetch())if(inportb(0xd8)== 0x1b)break;
                }
                return -1;
        }
 }
}
                                /*koniec*/
```

```
                                /* INT.C */
                    /* TESTOWANIE OBWODU PRZERWAN SYSTEMOWYCH */
             /* opracowal dr inz. Marian Wrzesien dnia 1988.11.7 */
# include <fun.h>
# include <adr.h>
# include <adrmw31.h>
# include <proto.h>
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,(char)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 k;
  }
                        /*koniec*/
```

PAKIET PROGRAMOW

DO TESTOWANIA PAKIETU MW 32

```
#include <ascii.h>
#include <proext.h>
#include <proto.h>

int menumw32()
{
/* ************************** menu ******************************************** */
static char * naglowek[2] ={
        "* menu 1 *",
"                           TEST PAKIETU MW32",
};
static char * text[]={
        "test calkowity",
        "dekoder adresow",
        "uklad kontroli alarmow i przerwan",
        "uklad kontroli zasilania",
        "blok kontroli przesylu po magistrali",
        "uklad RESET",
        "uklad MONITOR/PROGRAM",
        "rejestr 8 lampek programowych",
        "glowne menu",
};
char * podpis = "                                                    * spacja * del * cr *";
int line_n = 9;
static int line_l[9]={14,15,33,24,36,11,21,29,11};
/* ************************** menu ******************************************** */
return ( menu(naglowek,text,podpis,line_n,line_l) );
}
```

```
#include <ascii.h>
#include <tstglb.h>

static void autotest(void);
main()
{
int k ;
while(k = menumw32()){
/*---------------------------------------------------------------- */
switch(k){
case 1:autotest();
       break;
case 2: dekadr();
       break;
case 3: alprzer();
       break;
case 4:ukz();
       break;
case 5:magistr();
       break;
case 6:preset();
       break;
case 7:monitor();
       break;
case 8:rejBlam();
       break;
default:
    {
     #asm
      int 10
     #endasm
    }
  }
}/* end while */
/*---------------------------------------------------------------- */
}/* end mw31 */

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

static void autotest()
{
int bl=0;
if(dekadr()==0){
if(alprzer())bl=1;
if(ukz())bl=1;
if(magistr())bl=1;
if(preset())bl=1;
if(monitor())bl=1;
if(rejBlam())bl=1;
if(bl==0){
        printf("\ntest calkowity pakietu\t\t\t\t\t\t\tOK'");
        cr();
        return;
}
}
printf("\ntest calkowity pakietu ");
blad();
cr();
return;
}
```

```
/*1988-02-24*/                    /*ALPRZER.C*/

/*Testowanie torow sygnalowych :
                    obiektowe we : TEMP, WENT, OTW, 24, 24I, 24II, WYL.AL, ALARM,
                    obiektowe wy : AL.SW, AL.DZW,
                sygnaly stanu C7 : OTW/, BUDZIK,
                sygnaly stanu C6 : INTER/,
              sygnaly wewnetrzne : AL.MAG,
                      magistrala : RESET/,
                                                                        */

# include <mw32.h>
# include <proto.h>
# define inter (wralarm(datsuszasob))
# define notinter (wralarm(0x0))
# define ty(nr) (printf("\t\t%s\n\n\n",intbud[nr]))
# define tx(nr1,nr2) (printf("%s\t%s\n\n\n",intbud[nr1],intbud[nr2]))
# define tz(nr) (printf("%s\n",intbud[nr]))
   struct
     {
        int  data;
        char *sygnal;
        int  maskac7;
     }  static sterowanie[]=
          {
                0x0100, "TEMP",                 MASKA7,
                0x0001, "DYM",                  MASKA6,
                0x0400, "WENT",                 MASKA5,
                0X2000, "ZAS.OB",               MASKA2,
                0x0200, "ZAS.OB.I",             MASKA3,
                0x0002, "ZAS.OB.II",            MASKA4,
                0x0800, "OTW",                  MASKA1,
                0x1000, "WYL.AL",               0xFF,
                0x2202, "SUM.ZAS.OB",           0xFF,
                0x0   , "ZERO",                 0xFF,
          };
   static char *text[]=
     {
/*0*/   "zly sygnal w slowie stanu C6",
/*1*/   "prawidlowy sygnal w slowie stanu C6",
/*2*/   "nieuzasadnione generowanie przerwania INT",
/*3*/   "zly stan zerowy INT=0x",
/*4*/   "brak sygnalu INT",
     };
   static char *intbud[]=
     {
/*0*/   "       ***      OTW = 'L'      ***",
/*1*/   "***     ZAS.OB = 'L'   BU-WYL = 'H'     ***",
/*2*/   "***     OTW = 'H'      BU-WYL = 'H'     ***",
/*3*/   "***     OTW = 'H'      BU-ZAL = 'L'     ***",
/*4*/   "***     ZAS.OB = 'H'   BUDZIK/ = 'L'    ***",
/*5*/   "***     OTW = 'L'      BU-WYL = 'H'     ***",
/*6*/   "***   OTW = 'L'  BU-WYL = 'H'  BUDZIK/ = 'L'  ***",
/*7*/   "stan C7-OTW/",
/*8*/   "stan C7-BUD",
/*9*/   "stan C6-INTER/",
/*10*/  "OK !",
/*11*/  "blad !",
/*12*/  "Ustaw przelacznik 'BU' w polozeniu dolnym",
/*13*/  "Wyzeruj lampki kontrolne przyciskiem 'ZER.AL'",
/*14*/  "Oczekuje na przelaczenie 'BU'",
/*15*/  "Sprawdz zapalenie sie lampki 'SY'",
     };
static int a1=0,a2=0,a3=0,a4=0,a5=0,a6=0,a7=0,a8=0,a9=0,a10=0;
static int r1=0,r2=0,r3=0,r4=0,r5=0,r6=0,r7=0,r8=0,r9=0,r10=0;
alprzer()
 {
   scr_clr();
   while(1)
     {
        printf("UWAGA!!\nNiniejszy test wymaga zwarcia krosow K2 i K4\n");
        beep();
        delayp(5000);
        scr_curs((char)0,(char)0);
        scr_clr();
        delayp(5000);
        if(scr_pool())
          break;
     }
   cr();
   wewy();
   if(cont())goto koniec;
   interbud();
   if(cont())goto koniec;
   almag();
   if(cont())goto koniec;
   przer();
   if(cont())goto koniec;
   toralarm();
   uralprzer();
   return(-1*(r1|r2|r3|r4|r5));
   koniec:
   return(-1);
 }

 uralprzer()
   {
     urwewy();
     uralmag();
     urprzer();
   }
 wewy()
```

```
{
  int i=0,j=0,k=0;
  scr_clr();
  printf("\nUKLADY WEWY DWUSTANOWYCH");
  outportb(adrwpisl,(char)0xff);
  for(i=0;i<7;i++)
   {
     wralarm(sterowanie[i].data);
     delayp(1000);
     if(stanc7 & sterowanie[i].maskac7)
       {
         printf("\nBledny stan synalu %s w slowie stanu C7",sterowanie[i].sygnal);
         r1=1;
       }
     wralarm(0x0);
     delayp(100);
   }

  printf("\n\nTest lampek alarmu: TE, DY, WE, 24 \n");
  for(i=0;i<3;i++)
    {
       printf("\nTest dla sygnalow: TEMP,DYM,WENT,24 (%s)\n",sterowanie[3+i].sygnal);
       printf("Wyzeruj lampki kontrolne przyciskiem ZER.AL.\n");
       printf("Lampki powinny sie zapalac w ww. kolejnosci\n");
       cr();
      for(j=0;j<4;j++)
       {
              beep();
              delayp(400);
             wralarm(sterowanie[((j-3)?(k=j):(k=8))].data);
             delayp(100);
             wralarm(sterowanie[((j-3)?(k=9):(k=8-(j+i)))].data);
             delayp(1000);

       }
    }
  return(r1);
}
               /*URUCHAMIANIE WEWY*/

urwewy()
  {
      int j=0,k=-1;
       scr_clr();
       printf("Uruchamiasz tory WEWY ?  (t/n)\n");
       if(getch()!='t')
         return();
       scr_clr();
       printf("URUCHAMIANIE TOROW WEWY\n");

       while(1)
         {
           if(scr_pool() ! k==-1)
               {
                  if(k<5)
                    ++k;
                  else
                      break;
                  printf("Uruchamiany tor sygnalu %s\n",sterowanie[k].sygnal);
               }
           (k<3)?wralarm(sterowanie[k].data):wralarm(sterowanie[8].data);
           delayp(10000);
           (k<3)?wralarm(sterowanie[9].data):
                 (wralarm(sterowanie[9].data-sterowanie[k].data));
           delayp(10000);
         }
    return(0);
  }

               /*SPRAWDZENIE TORU INTER/ i BUDZIK */

interbud()
 {
   int i=0,j=0;
   scr_clr();
   printf("\nTor sygnalowy: INTER/,BUDZIK,RESET/\n\n\n");
   wralarm((char)0x0);
   delayp(100);
   ty(0);
   stanc7OTW? tx(7,10) : (tx(7,11),a1=0);
   tz(12);
   cr();
   if(stanc7BUD)
     tx(8,11),(a2=1);
   else
     tx(8,10),inter;
   zerp;
   notinter;
   cr();
   scr_curs((char)4,(char)0);
   scr_eos();
   ty(1);
   stanc6INTER? (tx(9,11),a3=1) : tx(9,10);
   wralarm(datsumzasob);
   zerp;
   zer1;
   cr();
   scr_curs((char)4,(char)0);
   scr_eos();
   ty(2);
   tz(13);
   tz(14);
```

```
    if(reset)
       printf("\nN:euzasadniony RESET/\n\n"),a8=1;
    while(!reset)
     {
      if(scr_pool())
          {
            i=1;
            printf("\nbrak RESET/\n\n"),a9=1;
            break;
          }
       if(j++>15000)
         beep(),j=0;
     }
    delayp(100);
    if(!i)
        printf("\nRESET/\tOK '\n\n");
    tz(15);
    cr();
    scr_curs((char)4,(char)0);
    scr_eos();
    ty(3);
    stanc7BUD? tx(8,10) : (tx(8,11),a4=1);
    cr();
    scr_curs((char)4,(char)0);
    scr_eos();
    ty(4);
    zerp;
    budzik;
    while(!stanc6INTER)
      {
        if(scr_pool())
           {
             a5=1;
             tx(9,11);
             break;
           }
        if(j++>15000)
          beep(),j=0;
      }
    if(!a5)
      tx(9,10);
    tz(12);
    wralarm(datotw);
    delayp(100);
    cr();
    scr_curs((char)4,(char)0);
    scr_eos();
    ty(5);
    stanc7BUD? tx(8,10) : (tx(8,11),a6=1);
    cr();
    scr_curs((char)4,(char)0);
    scr_eos();
    ty(6);
    budzik;
    delayp(5);
    stanc7BUD? (tx(8,11),a7=1) : tx(8,10);
    cr();
    r2=a1|a2|a3|a4|a5|a6|a7;
    return(r2);
}
                          /*sygnal AL.MAG*/

almag()
  {
    int i=0;
    scr_clr();
    printf("TOR SYGNALU AL.MAG.\n");
    for(i=0;i<2;i++)
      {
         printf("Wyzeruj lampki alarmow przyciskiem ZER.AL.\n");
         printf("Obserwuj zapalenie sie lampki MA przy sygnale dzw.\n");
         cr();
         beep();
         i?almag1:almag2;
      }
    return(0);
}
                                  /*URUCHAMIANIE AL.MAG */

uralmag()
  {
    int i=0,j=0;
    scr_clr();
    printf("Uruchamiasz tor AL.MAG ? (t/n)\n");
    if(getch()!='t')
       return(0);
    scr_clr();
    printf("URUCHAMIANIE TORU AL.MAG\n");
    while(1)
      {
         i?(i=0):(i=1);
        if(scr_pool())
          break;
        zer1;
        i?almag1:almag2;
        delayp(10000);
      }
    return(0);
  }
                                  /*UKLAD PRZERWAN*/

przer()
```

```
{
  int i=0,j=0,mas=0,mas1=0;
  scr_clr();
  printf("\nUKLAD PRZERWAN\n");
  outportb(adrwpisl,(char)0xff);
  outportb(IT1B1,(char)0x9b);
  outportb(IT1D5WY,(char)0x20);
  for(i=0;i<8;i++)
   {
      inter;
      delayp(100);
      (i>3)?(mas=0xf0):(mas=0x0f);
      (i>3)?(mas1=0x2):(mas1=0x1);
      cr();
      scr_curs((char)3,(char)0);
      scr_eos();
      printf("\nZewrzyj kolki %d+%d krosu K3",i+1,16-i);
      beep();
      cr();
      (i>3)?zerp:zeri;
      if((*INT & mas))
        {
          printf("%s %x\n",text[3],(INT & mas1));
          (stanc6 & mas1)?(printf("%s\n",text[0]),(a1=1)):printf("%s\n",text[1]);
        }
      (i>3)?notinter:usti;
      delayp(100);
      for(j=0;j<8;j++)
      if( INT & (0x1<<j))
       {
        if(i==j)
          {
            printf("%s%d\n",text[4],i);
            (stanc6 & mas1)?printf("%s\n",text[1]):(printf("%s\n",text[0]),(a2=1));
          }
        }
       else
         (i!=j)?(printf("%s%d\n",text[2],j),(a3=1)):printf("INT%d\t\t\t\t\t\tOK !\n",i);
     }
  r4=a1!a2!a3;
  cr();
  return(r4);
}

urprzer()
 {
    scr_clr();
    printf("Uruchamiasz uklad przerwan ? (t/n)\n");
    if(getch()!='t')
      return(0);
    scr_clr();
    printf("URUCHAMIANIE UKLADU PRZERWAN\n");
    tz(12);
    cr();
    while(1)
      {
       if(scr_pool())
         break;
       zerp;
       inter;
       delayp(10000);
       notinter;
       delayp(1000);
      }
  return(0);
}

                                /*LOGIKA ALARMU*/

toralarm()
 {
   int i=0;
   static char *arm[]=
     {
       "ALARM/ = 'L'",
       "ALMAG1 = 'L'",
       "ALMAG2 = 'L'",
       "OTW/   = 'H'",
     };
  static int ald[]={0,0,0,0};
  static int als[]={0,0,0,0};
  scr_clr();
  printf("\nTor sygnalowy: ALMAG,BUDZIK,ALARM,AL.DZW, AL.SW\n");
       a1=a2=0;
       tz(12);                                          /* B5/9=0*/
       beep();
       cr();
       zeri;                                            /* B5/1=0*/
       wylal;                                           /*zerowanie B1/9 */
       delayp(100);
       wralarm(0x0);                                    /*wycofanie zerowania*/
       delayp(10000);

       if(al_dzw)                                       /*stan zerowy = 'H'*/
         printf("WYL.AL nie zeruje sygnalu AL.DZW\n"),a1=1;
       if(al_sw)                                        /*stan zerowy = 'L'*/
         printf("AL.SW  nie zeruje sie\n"),a2=1;
       for(i=0;i<4;i++)
        {
          wylal;
```

```
            delayp(100);
            wralarm(0x0);
            delayp(100);
            zer1;
            delayp(100);
            switch(i)
                {
                        case 0 : alarm;
                                 break;
                        case 1 : almag1;
                                 break;
                        case 2 : almag2;
                                 break;
                        case 3 : otw;
                                 break;
                }
          delayp(100);
          if(!al_dzw)
            printf("AL.DZW nie generuje sie przy %s\n",arm[i]),ald[i]=1;
          else
            printf("AL.DZW przy %s \t\t\t\t\tOK !\n",arm[i]);
          if(!al_sw)
            printf("AL.SW nie generuje sie przy %s\n",arm[i]),als[i]=1;
          else
            printf("AL.SW przy %s \t\t\t\t\tOK !\n",arm[i]);
        }


        wralarm(0x0);
        delayp(100);
        r5=al|a2|ald[0]|ald[1]|ald[2]|ald[3]|als[0]|als[1]|als[2]|als[3];
        r5? blad() : printf("OK !\n");
      return(r5);
   }

cont()
  {
     char i=0;
     do
        {
          scr_curs((char)0,(char)0);
          scr_eos();
          printf("Kontynuujesz test ? (t/n)\n");
        }
     while((i=getch())!='n' && i!='N' && i!='t' && i!='T');
     if(i=='n' || i=='N')
       return(1);
     return(0);
  }
```

```
/*1989-02-07*/
                        /*DEKADR.C*/
            /*TESTOWANIE DEKODERA ADRESOW PAKIETU MW32*/
         /*opracowal dr inz. Marian Wrzesien dnia 1989-02-07*/

/*      Testowanie sygnalow :
  sygnaly wewnetrzne dekodera : CZYTR/, CZYTDL/, CZYTDH/, CZYT.S/,CZYT.A/,
                                CZYT.Z/, INPUTMB, INPUTSB,ALMAGIN1,ALMAGIN2,
                                USTI/, ZERI/, ZERP/, ZER1/, WPIS.L/, BUDZIK/,
                                OUTPUTMB, OUTPUTSB, ALMAGOUT1, ALMAGOUT2.
                                (przy zalaczonym i wylaczonym MPRO/)
                                STRO/
              magistrala wy : XACK/, MMAP2/.                                  */


# include <mw32.h>
# include <proto.h>

dekadr()
 {
   int a1=0,a2=0,a3=0,a4=0,a5=0,a6=0,a7=0;
   int i=0,j=0,xa=0,k=0,m=0,n=0;
   unsigned long int adr=0;
   static char *text[]=
     {
        "nieuzasadnione generowanie sygnalu XACK/ przy adresie: 0x",
        "brak sygnalu XACK/ przy adresie: 0x",
        "test ograniczony",
        "MPRO='L'",
        "MPRO='H'",
        "brak sygnalu MMAP2/ przy adresie: 0x",
     };
    static unsigned int tabadrs[10];
    static unsigned long int tabkom[40];
    struct
      {
        unsigned long int adr;
        char *in;
        char *out;
      } static tab[]=
        {
              adrczytr,       "CZYTR/ - D7/7",        "USTI/ - D6/7",
              adrczytdl,      "CZYTDL/ - D7/9",       "ZERI/ - D6/9",
              adrczytdh,      "CZZYTDH/- D7/10",      "ZERP/ - D6/10",
              adrczyts,       "CZYT.S/ - D7/11",      "ZER1/ - D6/11",
              adrczyta,       "CZYT.A/ - D7/12",      "WPIS.L/- D6/12",
              adrczytz,       "CZYT.Z/ - D7/13",      "BUDZIK/ - D6/13",
              adrczytmb,      "INPUTMB - C2/11",      "OUTPUTMB -C2/11",
              adrczytsb,      "INPUTSB - C2/8",       "OUTPUTSB - C2/8",
              adralmagczyt1,  "ALMAGIN1 - C2/6",      "ALMAGOUT1 - C2/6",
              adralmagczyt2,  "ALMAGIN2 - C2/3",      "ALMAGOUT2 - C2/3",
        };
        static unsigned long int adrmmap2[]=
          {
              0x00e900L,      /* 0x00e900 ÷ 0x00e9ff  gorna */
              0x00ea00L,      /* 0x00ea00 ÷ 0x00eaff  srodkowa */
              0x00eb00L,      /* 0x00eb00 ÷ 0x00ebff  dolna */
          };
    static int a[3]={0,0,0};


    scr_clr();
    printf("\n\tDEKODER ADRESOW");
    outportb(adrwpisl,(char)0xff);                      /*gaszenie diod 0÷7*/
    outportb(IT1B1,(char)0x9b);                         /*in,in,in*/
    outportb(IT1D6WY,(char)0x20);                       /*NOTINIT oraz ACL*/

    for(i=0;i<6;i++)                                    /*budowa tablicy dla...*/
        tabadrs[i]=((tab[i].adr) & 0x00ff);             /*zbioru adresow */
    for(k=i=0;i<6;i++)
      for(j=0;j<6;j++)
        tabkom[k++]=(((tabadrs[j])<<8) | (tabadrs[i]));
    for(i=6;i<10;i++)
      tabkom[i+30]=tab[i].adr;                          /*zbior 40 adresow */

                        /*TEST OGRANICZONY*/

    for(k=0;k<2;k++)
      {
        a1=a2=a3=a4=0;
        k?wrweukz(datmpro):wrweukz(datnotmpro);         /*z MPRO i bez */
        delayp(100);
        k?printf("\n%s %s",text[2],text[4]):printf("\n%s %s",text[2],text[3]);
        for(i=0;i<40;i++)
          {
            adr=tabkom[i];
            if(!k && !xackpt('bi',adr))                 /*nie ma XACK*/
              {
                 printf("\n%s%x\n",text[1],adr);        /*brak XACK*/
                 blad();
                 a1=1;
              }
            if(k && i<36 && xackpt('bi',adr))           /*jest XACK/*/
              {
                 printf("\n%s%x\n",text[0],adr);        /*nieuzasadn.XACK/*/
                 blad();
                 a2=1;
              }
            if(k && i>35 && !xackpt('bi',adr))          /*XACK/ ma byc z MPRO*/
              {
                 printf("\n%s%x\n",text[1],adr);
                 blad();
```

47

```
                a3=1;
              }
          }
            if(!(a1|a2|a3))
                printf("\t\t\t\t\t\tOK !\n");
      }
                                /*TEST PELNY*/

     a4=0;
     printf("\nTest pelny");
     wrweukz(datnotmpro);                                  /* bez MPRO */
     delayp(100);
     for(adr=0;adr<=0xffff;adr+=2)
       {
         if(xackpt('i',adr))                               /*jest XACK*/
           {
              k=0;
              for(i=0;i<40;i++)
                ((tabkom[i])==adr)?(++k):(k=k);
              if(!k)
                {
                   printf("\n%s%x\n",text[0],adr);
                   getch();
                   a4=1;
                }
           }
       }

     if(!a4)
       printf("\t\t\t\t\t\t\tOK !\n");
                          /* TESTOWANIE SYGNALU MMAP2 */
   printf("\nMMAP2/");
   for(i=0;i<3;i++)
     {
      adr=adrmmap2[i];
      (mmap2pt('r',adr))?(a[i]=1):(a[i]=0);
     }
   if(!(a[0]|a[1]|a[2]))
      printf("\t\tBrak sygnalu MMAP2/\n"),a5=1;
   else
    {
     a[0]?(adr=adrmmap2[0]):(a[1]?(adr=adrmmap2[1]):(a[2]?(adr=adrmmap2[2]):(adr=0x0)));
     printf("\tadres MMAP2/:\t0x00%xXX\t\t\t\tOK !\n",((adr & 0xff00))>>8));
    }
    cr();
    scr_clr();
    printf("\nOBSERWACJA SYGNALOW WYJSCIOWYCH DEKODERA ADRESOW\n");
    cr();
    for(i=0;i<10;i++)
      {
        printf("Sygnal %s oraz sygnal %s\n",tab[i].in,tab[i].out);
        while(!scr_pool())
          {
            outportb((int)(tab[i].adr),(char)0x0);
            inportb((int)(tab[i].adr));
          }
      }
    return(-1*(a1|a2|a3|a4|a5));
}

                                /*KONIEC*/
```

```
/*1989-02-07*/
                        /*MAGISTR*/
        /*TESTOWANIE POPRAWNOSCI  PRZESYLU PO MAGISTRALI KASETY*/
          /*opracowal dr inz. Marian Wrzesien dnia 1989-02-07*/

/*Testowane sygnaly :
        sterowanie : IOR/, IOW/, MEMR/, MEMW/, XACK/,
             DATA : DAT0/ ÷ DAT15/,
        wewnetrzne : ALMA6/,
*/

# include <mw32.h>
# include <proto.h>
magistr()
 {
   unsigned int data=0,wy=0;
   int i=0,k=0,j=0,m=0,n=0;
   int a1=0,a2=0,a3=0,a4=0,a5=0;
   static char *text[]=
     {
        "zly przekaz przy sterowaniu",
        "brak sygnalu XACK/ przy adresie",
     };
    static int dat[]={0x0,0xffff,0x5555,0xaaaa};
    static int e3[] = {0xf1,0xf1,0xe4,0xe4,0xf2,0xf2,0xe8,0xe8};
    static char *st[]= {"IN_mb","IN_sb","MR1","MR2","OUT_mb","OUT_sb",
                        "MW1","MW2"};

    scr_clr();
    printf("\n\tMAGISTRALA KASETY\n");
    outportb(adrwpis1,(char)0xff);                      /*gaszenie diod 0÷7*/
    outportb(IT1B1,(char)0x9b);                         /*in,in,in*/
    outportb(IT1D6WY,(char)0x20);                       /*NOTINIT oraz ACL*/
    for(i=0;i<2;i++)
      for(j=0;j<4;j++)
       {
         switch (j)
          {
            case 0:     data=0x0000;
                        break;
            case 1:     data=0xffff;
                        break;
            case 2:     data=0x5555;
                        break;
            case 3:     data=0xaaaa;
                        break;
          }
        i?wrmagmb(data):wrmagsb(data);
        wy=((stane6<<8) | stane5);
        if(data != wy)
          printf("\n\t0x%x: %s %-6s danej 0x%lx",wy,text[0],st[i+4],dat[j]),a1=1;
        i?wrmagpam1(data):wrmagpam2(data);
        wy=((stane6<<8) | stane5);
        if(data != wy)
          printf("\n\t0x%x: %s %-6s danej 0x%lx",wy,text[0],st[i+6],dat[j]),a2=1;
       }
     if(!a1)
       printf("\nKONTROLA I/O\t\t\t\t\t\t\tOK !\n");
     if(!a2)
       printf("\nKONTROLA MEMORY\t\t\t\t\t\t\tOK !\n");
    printf("\nREJESTR ROZKAZOW");
    for(i=j=0;i<8;i++)
      {
         switch(i)
           {
                 case 0 : rmagmb;
                          break;
                 case 1 : rmagsb;
                          break;
                 case 2 : rmagpam1;
                          break;
                 case 3 : rmagpam2;
                          break;
                 case 4 : wrmagmb((char)0x0);
                          break;
                 case 5 : wrmagsb((char)0x0);
                          break;
                 case 6 : wrmagpam1((unsigned char)0x0);
                          break;
                 case 7 : wrmagpam2((unsigned char)0x0);
                          break;
           }
         if(stane3!=e3[i])
           printf("\t bledny stane3  rowny: 0x%x przy sterowaniu %s\n",stane3,st[i]),a3=1;
      }
     if(!a3)
          printf("\t\t\t\t\t\tOK !\n");
     if(!(a1|a2|a3))
      {
         cr();
         scr_curs((char)2,(char)0);
         scr_eos();
         scr_curs((char)1,(char)64);
         printf("OK !\n");
         cr();
      }
      else
       blad();
      return(-1*(a1|a2|a3));

   }

                                        /*KONIEC*/
```

```
/*1989-02-12*/
                                /* MMAP2.C */
   /*FUNKCJA POMOCNICZA SPRAWDZAJACA POPRAWNOSC WYSTAPIENIA SYGNALU MMAP2*/
          /*opracowal dr inz Marian Wrzesien dnia 1989-02-12 */

# define ZER     0x0
# define NOTZER  0x40
# define notinit 0x20
# define map2 ((inportb(IT1B3PC)>>5)&0x1)
# include <adr.h>
# include <proto.h>
int mmap2pt(n,ad)                               /*zwraca 1 jesli jest MMAP2/ */
unsigned long ad;
unsigned int n;
  {
        outportb(IT1B3,(char)0x8b);                     /* zaprogramowanie bramy IT1B3 */
        outportb(IT1D6WY,(char)(ZER|notinit));  /* reset C1 IT1 */
        outportb(IT1D6WY,(char)(NOTZER|notinit));       /* wycofanie resetu */
        switch(n){
                case 'w':mwritew(ad,(unsigned int)0);
                         break;
                case 'r':mreadw(ad);
                         break;
                case 'bw':mwriteb(ad,(unsigned char)0);
                         break;
                case 'br':mreadb(ad);
                         break;
                default: return -1;
        }
return (map2);
}
                        /*koniec*/
```

```
/*1989-02-07*/
                        /*MONITOR*/
        /*TESTOWANIE PRZELACZNIKA 'MONITOR' PLYTY CZOLOWEJ PAKIETU MW32*/
             /*opracowal dr inz Marian Wrzesien dnia 1989-02-07*/


# include <mw32.h>
# include <proto.h>
# define dat_otw 0x800
monitor()
 {
   int i=0,j=0;
   int a1,a2,a3,a4,a5,a6;
    static char *text[]=
      {
        "nieuzasadniony poziom 'L' sygnalu stanu C6 przy podaniu reset ukladu",
        "Ustaw przelacznik MON w polozeniu",
        "nieuzasadniony poziom 'H' sygnalu stanu C6 przy podaniu set ukladu",
        "nieuzasadniony poziom 'L' sygnalu stanu C6 przy podaniu OTW",
        "nieuzasadniony poziom 'H' sygnalu stanu OTW-C7 przy braku sterowania",
        "nieuzasadniony poziom 'L' sygnalu stanu OTW-C7 przy sterowaniu alarmu",
      };
    scr_clr();
    printf("\nUKLAD MONITOR\n");
    outportb(adrwpisl,(char)0xff);                    /*gaszenie diod 0÷7*/
    outportb(IT1B1,(char)0x9b);                  /*in,in,in*/
    outportb(IT1D6WY,(char)0x20);                     /*NOTINIT oraz ACL*/
    i=j=0;

    wralarm(0x0);                                /*otw=1; otw/=0*/
    delayp(100);
    (stanc7 & 0x2)?(a1=0):(printf("%s\n",text[4]),a1=1,blad());
    scr_curs((char)5,(char)0);
    printf("%s DOLNYM\n",text[1]);
    cr();
    (stanc6 & 0x8)?(a2=0):(printf("%s\n",text[0]),a2=1,blad());

    scr_curs((char)5,(char)0);
    scr_eos();
    printf("%s GORNYM\n",text[1]);
    cr();
    (stanc6 & 0x8)?(printf("%s\n",text[2]),a3=1,blad()):(a3=0);

    wralarm(dat_otw);
    delayp(100);
    (stanc7 & 0x2)?(printf("%s\n",text[5]),a4=1,blad()):(a4=0);
    (stanc6 & 0x8)?(a5=0):(printf("%s\n",text[3]),a5=1,blad());
    scr_curs((char)3,(char)0);
    scr_eos();
    scr_curs((char)1,(char)65);
    if(a1|a2|a3|a4|a5)
      {
        blad();
        return(-1);
      }
     else
       {
         printf("OK !\n");
         cr();
         return(0);
       }

}
                                        /*KONIEC*/
```

```
/*1989-02-07*/
                        /*PRESET.C*/
        /*TESTOWANIE PRZYCISKU 'RESET' NA PLYCIE CZOLOWEJ PAKIETU MW32*/
             /*opracowal dr inz Marian Wrzesien dnia 1989-02-07*/


/*Testowane tory sygnalow :
      przyciski sterujace : P_RESET,
           magistrala wy : RESET/,'OUTON/,                              */

# include <mw32.h>
# include <proto.h>
preset()
 {
   int i=0,j=0;
    \
    scr_clr();
    printf("\nUKLAD P_RESET\n");
    outportb(adrwpisl,(char)0xff);                    /*gaszenie diod 0÷7*/
    outportb(IT1B1,(char)0x9b);              /*in,in,in*/
    outportb(IT1D&WY,(char)0x20);                     /*NOTINIT oraz ACL*/
   i=j=0;
   printf("\n\nWcisnij przycisk RESET na plycie czolowej pakietu MW-32\n");
   printf("Oczekuje na sygnaly: RESET/ i OUTON/\n\n");
   scr_curs((char)15,(char)0);
   printf("Brak komentarza po wcisnieciu przycisku RESET ?:wcisnij CR \n");
   delayp(1000);
   while(!reset)
     if(scr_pool())
         {
            beep();
            printf("Brak sygnalu RESET/ %\n"),i=1;
            break;
         }
   while(!outon)
     if(scr_pool())
         {
            printf("Brak sygnalu OUTON/ % %\n"),j=1;
            beep();
            delayp(100);
            beep();
            break;
         }
   delayp(100);
   if(!(i|j))
     {
       scr_curs((char)2,(char)0);
       scr_eos();
       scr_curs((char)1,(char)44);
       printf("RESET/ i OUTON/ \t\tOK !\n");
       cr();
     }
   else
    blad();
   return(-1*(i|j));
}
                                  /*KONIEC*/
```

```
/*1989-02-07*/
                                /*REJBLAM*/
          /*TESTOWANIE REJESTRU 8 LAMPEK SYGNALOWYCH */
        /*opracowal dr inz Marian Wrzesien dnia 1989-02-07*/


# include <mw32.h>
# include <proto.h>
rejBlam()
 {
   int i=0,j=0;

   printf("\n\tOceny testu dokonuje sie poprzez obserwacje lampek kontrolnych\n");
   printf("Diody naprzemian zapalaja sie i gasna\n");
   beep();
   cr();
   for(i=0;i<3;i++)
    {
      outportb(adrwpisl,(char)0x0);
      delayp(5000);
      outportb(adrwpisl,(char)0xff);
      delayp(5000);
    }
  printf("Diody gasna kolejno\n");
  beep();
  cr();
  for(i=0;i<3;i++)
    for(j=0;j<8;j++,delayp(5000))
         outportb(adrwpisl,(char)(0x1<<j));
  outportb(adrwpisl,(char)0xff);
  printf("Diody zapalaja sie kolejno\n");
  beep();
  cr();
  for(i=0;i<3;i++)
    for(j=0;j<8;j++,delayp(5000))
        outportb(adrwpisl,(char)(~(0x1<<j)));

  outportb(adrwpisl,(char)0xff);
  printf("\n\n\n\t\t\tDo widzenia mily uzytkowniku\n");
  cr();
  return(0);
}
                                /*KONIEC*/
```

```
/*1988-12-07*/
                        /*      UKZ.C   */
       /*TESTOWANIE UKLADU KONTROLI ZASILAN PAKIETU MW-32 */
       /*opracowal dr inz Marian Wrzesien dnia 1988-12-07  */

/*Testowanie torow sygnalowych :
                magistrala we :  +5V, +15V, +12V, -12V, -15V, -5V, +5VB, AC,
                   wewnetrzne :  DCL/, ACL/,
                magistrala wy :  PFIN/, PFSN/, RESET/, MPRO/, OUTON/,
                 obiektowe wy :  ZAKL.PAM., STYCZNIK/,                  */

# include <mw32.h>
# include <proto.h>
# define stop (printf("\n\t\t\t%s",text[6]))
ukz()
  {

 struct
   {
     int stan;
     char * sterc8;
     int ster;
   } static sterowanie[]=
      {
        /*0*/   0x0,   "ZERO"  , 0x1f  ,
        /*1*/   0x1,   "ACL/"  , 0x80  ,                /*IT1D&WY*/
        /*2*/   0x2,   "DCL/"  , 0x0f1f,                /*PT8*/
        /*3*/   0x4,   "+12VL/", 0x091f,                /*PT9*/
        /*4*/   0x8,   "+15VL/", 0x0b1f,                /*PT10*/
        /*5*/   0x10,  "-5VL/" , 0x101f,                /*PT3*/
        /*6*/   0x20,  "-12VL/", 0x901f,                /*PT3*/
        /*7*/   0x40,  "-15VL/", 0x501f,                /*PT2*/
        /*8*/   0x80,  "+5VBL/", 0x0d1f,                /*PT1*/
      };
   static char *text[]=
    {
     /*0*/       "brak sygnalu",
     /*1*/       "Nieuzasadnione generowanie sygnalu",
     /*2*/       "przy prawidlowych wartosciach kontrolowanych napiec",
     /*3*/       "przy zaniku sygnalu AC",
     /*4*/       "w slowie stanu brak sygnalu",
     /*5*/       "przy powrocie sygnalu AC",
     /*6*/       "test przerwany",
     /*7*/       "Blednе  slowo  stanu rowne: 0x",
     /*8*/       "Wlasciwe slowo stanu rowne: 0x",
     /*9*/       "Po powrocie napiec stalych nieuzasadnione generowanie sygnalu",
    };

    unsigned int ap;
    int a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,i,j,k;
    int k1,k2,k3,k4,k5,k6,k7,k8;
    zero_ai;
    scr_clr();
    printf("\nUKLAD KONTROLI ZASILANIA\n");
    outportb(adrwpisl,(char)0xff);                  /*gaszenie diod 0÷7*/
    outportb(IT1B1,(char)0x9b);                  /*in,in,in*/
    outportb(IT1D&WY,(char)0x20);                   /*NOTINIT*/

    printf("STAN ZEROWY");
    wrweukz(sterowanie[0].ster);                 /* STAN ZEROWY UKZ */
    zer1;
    delayp(100);
    if(stanc8)
      printf("\nSlowo stanu: jest 0x%x - powinno byc:0x0",stanc8),a1=1;
    stycznik?(printf("\n%s STYCZNIK/",text[1]),a2=1):(a2=0); /*norm='L'*/
    zaklpam ?(a3=0):(printf("\n%s ZAKL.PAM/\n",text[1]),a3=1); /*norm='H'*/
    mpro?    (printf("\n%s MPRO/",text[1]),a4=1):(a4=0);
    reset?   (printf("\n%s RESET/",text[1]),a5=1):(a5=0);
    pfin?    (printf("\n%s PFIN/",text[1]),a6=1):(a6=0);
    pfsn?    (printf("\n%s PFSN/",text[1]),a7=1):(a7=0);
    outon?   (printf("\n%s OUTON/",text[1]),a8=1):(a8=0);
    if(a1|a2|a3|a4|a5|a6|a7|a8)
      {
        printf("\n%s",text[2]);
        stop;
        blad();
        return(-1);
      }
    else
      printf("\t\t\t\t\t\tOK !\n\n");
                        /*SPADEK NAPIECIA AC*/
    printf("DETEKTOR AC");
    wrweacl((char)(sterowanie[1].ster|0x20));
    i=j=0;
    while(!reset && i<1000)
      i++;
    while(!outon && j<1000)
      j++;
    stanc8OUTON?(printf("\n%s C6-OUTON/\n",text[4]),a9=1):(a9=0);
    delayp(100);
    (i<990)?(printf("\n%s RESET/\n",text[1]),a5=1):(a5=0);
    (j<990)?(printf("\n%s OUTON/\n",text[1]),a8=1):(a8=0);
    pfin?(a6=0):(printf("\n%s PFIN/\n",text[0]),a6=1);
    pfsn?(a7=0):(printf("\n%s PFSN/\n",text[0]),a7=1);

    if(stanc8!=sterowanie[1].stan)
      {
        printf("\n%s",text[7]);
        printf("%x\n",stanc8);
        printf("%s",text[8]);
        printf("%x\n",sterowanie[1].stan);
        a1=1;
```

```
  }
if(a1|a5|a6|a7|a8|a9)
  {
     printf("\n%s",text[3]);
     stop;
     blad1();
     return(-1);
  }
i=j=0;
wrweacl((char)(0x0|0x20));
while(!reset && i<1000)
  i++;
while(!outon &&j<1000)
  j++;
(i>990)?(printf("\n%s RESET/",text[0]),a5=1):(a5=0);
(j>990)?(printf("\n%s OUTON/",text[0]),a8=1):(a8=0);
zer1;
pfin? (printf("%s PFIN/\n",text[1]),a6=1):(a6=0);
pfsn? (printf("%s PFSN/\n",text[1]),a7=1):(a7=0);
stanc6OUTON?(a9=0):(printf("%s C6-OUTON/\n",text[4]),(a9=1));
if(a5|a6|a7|a8|a9)
  {
     printf("\n%s",text[5]);
     stop;
     blad1();
     return(-1);
  }
else
  printf("\t\tPFIN/,PFSN/,RESET/,OUTON/\t\tOK !\n\n");


delayp(10000);
printf("DETEKTOR DC");
for(k=2;k<=8;k++)
  {
     zer1;
     wrweukz(sterowanie[k].ster);
         i=j=0;
         while(!reset && i<1000)
           i++;
         while(!outon && j<1000)
           j++;
         delayp(100);
         (i<990)?(printf("\n%s RESET/",text[1]),a5=1):(a5=0);
         (j<990)?(printf("\n%s OUTON/",text[1]),a8=1):(a8=0);
         stanc6OUTON? (printf("\n%s C6-OUTON",text[1]),a9=1):(a9=0);
     delayp(100);
     if(k<8)
       stycznik? (a2=0):(printf("\n%s STYCZNIK/",text[0]),a2=1);
     if(k==2)
       {
         mpro? (a4=0):(printf("\n%s MPRO/",text[0]),a4=1);
         wrweukz(sterowanie[0].ster);                /*wycofanie MPRO/*/
         delayp(10000);
       }
     if((k1=stanc8)!=sterowanie[k].stan)
       {
         printf("\n%s",text[7]);
         printf("%x\n",k1);
         printf("%s",text[8]);
         printf("%x\n",sterowanie[k].stan);
         a1=1;
       }
     delayp(100);
     if(k==8)
        zaklpam?  (printf("\n%s ZAKL.PAM/",text[0]),a3=1):(a3=0);
     else
       zaklpam?  (a3=0):(printf("\n%s ZAKL.PAM",text[1]),a3=1);
     if(a1|a2|a3|a4|a5|a8|a9)
       {
          printf("\nprzy spadku napiecia %s",sterowanie[k].sterc8);
          stop;
          blad1();
          return(-1);
       }
  }

                                    /* POWROT NAPIEC STALYCH */

zer1;
for(k=2;k<=8;k++)
  {
     i=j=0;
     wrweukz(sterowanie[k].ster);
     delayp(100);
     wrweukz(sterowanie[0].ster);
     while(!reset && i<10000)
       i++;
     while(!outon && j<10000)
       j++;
     if(k==2)
       {
         (i>9990)?(printf("\n%s RESET/",text[0]),a5=1):(a5=0);
         (j>9990)?(printf("\n%s OUTON/",text[0]),a8=1):(a8=0);
         stanc6OUTON? (a9=0):(printf("\n%s C6-OUTON",text[0]),a9=1);
       }
     else
       {
         (i<9990)?(printf("\n%s RESET/",text[1]),a5=1):(a5=0);
         (j<9990)?(printf("\n%s OUTON/",text[1]),a8=1):(a8=0);
         stanc6OUTON?  (printf("\n%s C6-OUTON/",text[1]),a9=1):(a9=0);
       }
     mpro?    (printf("\n%s MPRO/",text[1]),a4=1):(a4=0);
```

```
        delayp(100);
        stycznik?(printf("\n%s STYCZNIK/",text[1]),a2=1):(a2=0);
        if(k<8)
          zaklpam? (a3=0):(printf("\n%s ZAKL.PAM/",text[1]),a3=1);
        else
          zaklpam? (printf("\n%s ZAKL.PAM/",text[0]),a3=1):(a3=0);
        if(a1|a2|a3|a4|a5|a8|a9)
          {
            printf("\npo powrocie napiecia %s",sterowanie[k].sterc8);
            stop;
            blad!);
            return(-1);
          }
      }
  printf("\tRESET/,OUTON/,MPRO/,STYCZNIK/,ZAKL.PAM/\t\tOK !\n");
cr();
scr_curs((char)2,(char)0);
scr_eos();
scr_curs((char)1,(char)65);
printf("OK !\n");
cr();
return(0);
}

                                /*KONIEC*/
```

PAKIET PROGRAMÓW

DO TESTOWANIA PAKIETU MZ-70

```
 #include <proto.h>
main(){
scr_clr();
if(mz70()!=0)
  {
    scr_clr();
    printf("Czy rozpoczac program uruchomieniowy MZ70\t\t\t*(t/n)*");
       if(getch()=='t')
         {
           scr_clr();
           urmz70();
         }
  }
  {
    #asm
        int 10
    #endasm
  }
}
```

```
/*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>
# include <proto.h>
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, bl, i, j, p, s, x , t ;
   b=bl=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\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(wejscia),\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)>>8)&0xff;
         wyjzew=inportw(adr)&0xff;
         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 roznica pomiedzy amplitudami",
      /*1*/   "Niewlasciwa wartosc amplitudy przebiegu",
      /*2*/   "Przekroczona dopuszczalna wartosc skladowej stalej przebiegu",
    };
  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>
# include <proto.h>
urmz70()
  {
    printf("URUCHAMIANIE PAKIETU MZ-70\nczesc analogowa (a)\nczesc cyfrowa (c)\n");
    printf("Wybierz opcje.\n");
    if(getch()=='a')uranal();else urcyf();
    return 0;
  }
    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 !ELN.\n",text[i]);
    printf("Po kolejnym wciskaniu CR wyrownuj wartosci bezwzgledne wskazywanych 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(IT2B1C1,0x30);
                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':

                return(0);
    }
 }while((x=getch())!='t');
}
                                        /*koniec*/
```

PAKIET PROGRAMÓW

DO TESTOWANIA PAKIETU ML 50

```
;
;  Testy uruchomieniowe pakietu ML50 02.01.89
;
;               Wersja dla AZTECa
;
LF EQU 10
CR EQU 13
IT1B3   EQU     0FECEH
IT1D&WY EQU     0FED0H
IT1B3PB EQU     0FECAH
;
        PUBLIC  ml50_
;
DATASEG SEGMENT
MEM_SP  DW 1 DUP(?)             ;pole na pamietanie SP
DATASEG ENDS
;
CODESEG     SEGMENT
        ASSUME  CS:CODESEG,ES:CODESEG,DS:DATASEG
;
ml50    PROC
POCZA:  PUSH    ES
        PUSH    DS
        PUSH    BP
        MOV     AX,DATASEG
        MOV     DS,AX
        MOV     MEM_SP,SP       ;pamietanie SP
ML500:  MOV     AX,DATASEG
        MOV     DS,AX
        MOV     SP,MEM_SP       ;wyrownanie stosu
        IN      AL,0D9H         ;zerowanie BTMD
        MOV     AX,CODESEG
        MOV     ES,AX
        MOV     BX,OFFSET TEKST1
ML501:  CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,' '
        JNZ     ML502
        POP     BP
        POP     DS              ;koniec testow
        POP     ES
        RET
ML502:  CMP     AL,'1'
        JZ      REPET           ;wybrane testy repetycyjne
        CMP     AL,'2'
        JZ      DIAGNO          ;wybrane testy diagnostyczne
        MOV     BX,OFFSET TEKST2
        JMP     ML501
;
;Testy repetycyjne
;
REPET:  MOV     BX,OFFSET TEKST3
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,' '
        JZ      ML500
        CMP     AL,'5'
        JNC     REPET           ;blad-podana liczba wieksza od 4
        CMP     AL,'1'
        JB      REPET           ;blad-podana liczba mniejsza od 1
        MOV     DL,AL
        CALL    ADRES           ;adres komorki DS:BX
        MOV     CX,BX
        CMP     DL,'2'
        JC      REPODC          ;test odczytu 8-mio lub 16-bitow
        MOV     BX,OFFSET TEKST5
        CALL    OUTTXT
        CALL    GETADR          ;pobranie danych do wysylania
        MOV     AX,BX
        MOV     BX,CX
        CMP     DL,'3'
        JZ      RZA08           ;zapis 8-mio bitowy
RZA16:  MOV     [BX],AX         ;zapis 16-bitowy
        CALL    CZY_STOP
        JMP     RZA16
RZA08:  MOV     [BX],AL
        CALL    CZY_STOP
        JMP     RZA08
REPODC: CMP     DL,'1'
        JZ      ROD08           ;odczyt 8-mio bitowy
ROD16:  MOV     AX,[BX]         ;odczyt 16-bitowy
        CALL    CZY_STOP
        JMP     ROD16
ROD08:  MOV     AL,[BX]         ;odczyt 8-bitowy
        CALL    CZY_STOP
        JMP     ROD08
;
;Testy diagnostyczne
;
DIAGNO: MOV     BX,OFFSET TEKST4
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,' '
        JZ      POCZA
        CMP     AL,'1'
        JZ      TPROM
        CMP     AL,'2'
        JZ      TPROM
        CMP     AL,'3'
        JZ      TUPLYW
        CMP     AL,'5'
        JZ      TXXXX
```

64

```
        CMP     AL,'4'
        JNZ     DIAGNC          ;blad
        JMP     TINRAM
POCZ1:  JMP     ML500
TXXXX:  JMP     TXACK
;
;Test PROM
;
TPROM:  MOV     DL,AL
        PUSH    DS
        MOV     BP,100H
        CALL    ADRES
        POP     ES
        CMP     DL,'1'
        JZ      TPROM8          ;odczyt 8-mio bitowy
        MOV     CX,4000H
TPROM1: CALL    GENERJ2
        CMP     AX,[BX]
        JNZ     TPROM2          ;blad
TPROM3: INC     BX
        INC     BX
        LOOP    TPROM1
TPROM4: MOV     AX,CODESEG
        MOV     ES,AX
        MOV     BX,OFFSET TEKST7        ;koniec testu
        CALL    OUTTXT
        JMP     ML500
TPROM8: MOV     CX,8000H
TPROM9: CALL    GENERUJ
        CMP     AL,[BX]
        JNZ     TPROMA
TPROMB: INC     BX
        LOOP    TPROM9
        JMP     TPROM4
TPROM2: CALL    BLAD16
        JMP     TPROM3
TPROMA: CALL    BLAD08
        JMP     TPROMB
;
;Test uplywnosci
;
TUPLYW: MOV     BX,OFFSET TEKST8
        CALL    OUTTXT
        CALL    GETADR          ;pobranie opoznienia
        MOV     SI,BX
        CALL    ADRES
        PUSH    BX
        MOV     BX,OFFSET TEKST9
        CALL    OUTTXT          ;zapis jedynek
        MOV     DL,-1
TUPLY1: POP     BX
        PUSH    BX
        MOV     CX,2000H        ;zapisujemy caly 8 K RAM
TUPLY2: MOV     [BX],DL
        CMP     DL,[BX]
        JZ      TUPLY9
        CALL    BLADUP          ;blad przy zapisie
TUPLY8: INC     BX
        LOOP    TUPLY2
        MOV     BX,OFFSET TEKSTA
        CALL    OUTTXT          ;opoznienie
        MOV     CX,SI
TUPLY3: CALL    SEKUNDA
        LOOP    TUPLY3
        POP     BX
        PUSH    BX
        MOV     CX,2000H        ;sprawdzanie
TUPLY4: CMP     DL,[BX]
        JZ      TUPLY9
        CALL    BLADUP          ;blad po opoznieniu
TUPLY9: INC     BX
        LOOP    TUPLY4
        POP     BX
        INC     DL
        JNZ     TUPLY5
        PUSH    BX
        MOV     BX,OFFSET TEKSTB
        CALL    OUTTXT          ;zapis zer
        JMP     TUPLY1
TUPLY5: JMP     TPROM4
;
;Test informacyjny RAM
TINRAM: CALL    ADRES           ;adres poczatku testowanego obszaru
        SUB     DL,DL           ;zerowanie calego obszaru 8k
        MOV     BP,BX           ;adres poczatku kolejnego bloku 2k
        MOV     SI,BX           ;adres obrabianej komorki
        PUSH    BX
        PUSH    BX
        MOV     BX,OFFSET RAMT2
        CALL    OUTTXT
        POP     BX
        MOV     CX,2000H
ZEROW:  MOV     [BX],DL
        CMP     DL,[BX]         ;czy informacja wpisala sie poprawnie
        JZ      RAM0            ;tak
        CALL    BLADUP          ;nie - blad przy zapisie
RAM0:   INC     BX
        LOOP    ZEROW
        MOV     DX,8000H        ;pierwsza wartosc plywajacej jedynki
        MOV     CX,4            ;petla na 4*2k testowanego RAMu
RAM2:   CMP     CL,4
        JNE     NIE0
```

```
        MOV     BX,OFFSET RAMTA ;testujemy 0-7FF
NIE0:   CMP     CL,3
        JNE     NIE1
        MOV     BX,OFFSET RAMTB ;testujemy 800-FFF
NIE1:   CMP     CL,2
        JNE     NIE2
        MOV     BX,OFFSET RAMTC ;testujemy 1000-17FF
NIE2:   CMP     CL,1
        JNE     NIE3
        MOV     BX,OFFSET RAMTD ;testujemy 1800-1FFF
NIE3:   CALL    OUTTXT
        PUSH    CX
        MOV     CX,400H         ;petla 2k pamieci
RAM3:   PUSH    CX
RAM4:   MOV     [SI],DX         ;zapis plywajacej jedynki
        MOV     AX,[SI]
        CMP     DX,AX           ;czy sie wpisalo
        JZ      RAMOK           ;tak
        PUSH    AX
        MOV     BX,OFFSET RAMT8
        CALL    OUTTXT
        MOV     AX,SI
        CALL    OUTWOR          ;wydruk adresu blednej komorki
        MOV     BX,OFFSET RAMT9
        CALL    OUTTXT          ;zapisano
        MOV     AX,DX
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT6
        CALL    OUTTXT          ;odczytano
        POP     AX
        CALL    OUTWOR
        CALL    CZY_KONIEC
RAMOK:  MOV     DI,BP
        MOV     CX,400H
RAM5:   SUB     AX,AX
        PUSH    ES
        PUSH    DS
        POP     ES
REPE    SCASW
        POP     ES
        JNE     RAM6            ;natrafiono na blad
RAMW:   ROR     DX,1            ;zmiana maski
        JNC     RAM4            ;jeszcze zapis tego samego bajtu
        MOV     [SI],WORD PTR 0 ;odtworzenie komorki ostatnio testowanej
        INC     SI
        INC     SI
        CALL    CZY_STOP
        POP     CX
        LOOP    RAM3            ;koniec petli przesuwania jedynki dla 2K
        ADD     BP,800H         ;przejscie do nastepnego obszaru 2k
        POP     CX
        DEC     CX
        JZ      RAM01           ;koniec petli obszarow 2K
        JMP     RAM2
RAM01:  JMP     RAM9            ;koniec plywajacej jedynki - do plywajacego zera
;Obsluga bledu wykrytego w stringu:
RAM6:   DEC     DI              ;DI-2 - adres w ktorym byl blad
        DEC     DI
        CMP     DI,SI           ;SI - adres komorki obrabianej
        JNZ     RAM7            ;to jest blad rzeczywiscie
RAMX:   INC     DI              ;blad dotyczy komorki w ktorej jest plywajaca jedynka
        INC     DI
        AND     CX,CX
        JZ      RAMW
        JMP     RAM5
RAM7:   MOV     BX,OFFSET RAMT4 ;po wpisie
        CALL    OUTTXT
        MOV     AX,DX           ;kod plywajacej jedynki
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT5 ;do komorki
        CALL    OUTTXT
        MOV     AX,SI           ;adres komorki gdzie byla plywajaca jedynka
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT6 ;odczytano
        CALL    OUTTXT
        MOV     AX,[DI]         ;to co odczytano w zlej komorce
        CALL    OUTWOR
        MOV     [DI],WORD PTR 0 ;odtworzenie zera po wykryciu bledu
        MOV     BX,OFFSET RAMT7 ;w komorce
        CALL    OUTTXT
        MOV     AX,DI           ;adres zlej komorki
        CALL    OUTWOR
        CALL    CZY_KONIEC      ;czy kontynuowac po bledzie
        JMP     RAMX
;plywajace zero
RAM9:   MOV     DL,-1           ;FF-owanie calego obszaru 8k
        MOV     BX,OFFSET RAMT3
        CALL    OUTTXT
        POP     BX
        MOV     BP,BX           ;adres poczatku kolejnego bloku 2k
        MOV     SI,BX           ;adres obrabianej komorki
        MOV     CX,2000H
FFOWA:  MOV     [BX],DL
        CMP     DL,[BX]         ;czy informacja wpisala sie poprawnie
        JZ      RAMXQ           ;tak
        CALL    BLADUP          ;nie - blad przy zapisie
RAMXQ:  INC     BX
        LOOP    FFOWA
        MOV     DX,7FFF         ;pierwsza wartosc plywajacego zera
        MOV     CX,4            ;petla na 4*2k testowanego RAMu
RAMB:   CMP     CL,4
        JNE     NIE4
```

```
        MOV     BX,OFFSET RAMTA ;testujemy 0-7FF
NIE4:   CMP     CL,3
        JNE     NIE5
        MOV     BX,OFFSET RAMTB ;testujemy 800-FFF
NIE5:   CMP     CL,2
        JNE     NIE6
        MOV     BX,OFFSET RAMTC ;testujemy 1000-17FF
NIE6:   CMP     CL,1
        JNE     NIE7
        MOV     BX,OFFSET RAMTD ;testujemy 1800-1FFF
NIE7:   CALL    OUTTXT
        PUSH    CX
        MOV     CX,400H         ;petla 2k pamieci
RAMC:   PUSH    CX
RAMD:   MOV     [SI],DX         ;zapis plywajacej jedynki
        MOV     AX,[SI]
        CMP     DX,AX           ;czy sie wpisalo
        JZ      RAMOK2          ;tak
        PUSH    AX
        MOV     BX,OFFSET RAMT8
        CALL    OUTTXT
        MOV     AX,SI
        CALL    OUTWOR          ;wydruk adresu blednej komorki
        MOV     BX,OFFSET RAMT9
        CALL    OUTTXT          ;zapisano
        MOV     AX,DX
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT6
        CALL    OUTTXT          ;odczytano
        POP     AX
        CALL    OUTWOR
        CALL    CZY_KONIEC
RAMOK2: MOV     DI,BP
        MOV     CX,400H
RAME:   MOV     AX,-1
        PUSH    ES
        PUSH    DS
        POP     ES
REPE    SCASW
        POP     ES
        JNE     RAMF            ;natrafiono na blad
RAMU:   ROR     DX,1            ;zmiana maski
        JC      RAMD            ;jeszcze zapis tego samego bajtu
        MOV     [SI],WORD PTR -1;odtworzenie komorki ostatnio testowanej
        INC     SI
        INC     SI
        CALL    CZY_STOP
        POP     CX
        LOOP    RAMC            ;koniec petli przesuwania zera dla 2K
        ADD     BP,800H         ;przejscie do nastepnego obszaru 2k
        POP     CX
        DEC     CX
        JZ      RAMQ2           ;koniec petli obszarow 2K
        JMP     RAMB
RAMQ2:  JMP     TPROM4          ;koniec calego testu
;Obsluga bledu dla plywajacego zera:
RAMF:   DEC     DI              ;DI-2 - adres w ktorym byl blad
        DEC     DI
        CMP     DI,SI           ;SI - adres komorki obrabianej
        JNZ     RAM6            ;to jest blad rzeczywiscie
RAMQ:   INC     DI              ;blad dotyczy komorki w ktorej jest plywajace zero
        INC     DI
        AND     CX,CX
        JZ      RAMU
        JMP     RAME
RAM6:   MOV     BX,OFFSET RAMT4 ;po wpisie
        CALL    OUTTXT
        MOV     AX,DX           ;kod plywajacego zera
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT5 ;do komorki
        CALL    OUTTXT
        MOV     AX,SI           ;adres komorki gdzie bylo plywajace zero
        CALL    OUTWOR
        MOV     BX,OFFSET RAMT6 ;odczytano
        CALL    OUTTXT
        MOV     AX,[DI]         ;to co odczytano w zlej komorce
        CALL    OUTWOR
        MOV     [DI],WORD PTR -1;odtworzenie jedynki po wykryciu bledu
        MOV     BX,OFFSET RAMT7 ;w komorce
        CALL    OUTTXT
        MOV     AX,DI           ;adres zlej komorki
        CALL    OUTWOR
        CALL    CZY_KONIEC      ;czy kontynuowac po bledzie
        JMP     RAMQ
;
;Test sygnalu XACK
TXACK:  MOV     BX,OFFSET TEKSTD
        CALL    OUTTXT
        CALL    ADRES           ;pobranie adresu RAMu do DS:BX
        PUSH    BX
;komunikacja 8-bitowa
        PUSH    BX
        MOV     BX,OFFSET TEKSTF
        CALL    OUTTXT
        POP     BX
        MOV     CX,2000H
TXAC1:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset CI IT1
```

```
        MOV     AL,40H
        OUT     DX,AL           ;wycofanie resetu
        MOV     AL,[BX]         ;odczyt RAMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXAC2           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKSTC ;brak XACK dla odczytu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXAC2:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset C1 IT1
        MOV     AL,40H
        OUT     DX,AL           ;wycofanie resetu
        MOV     [BX],AL         ;zapis RAMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXAC3           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKSTH ;brak XACK dla zapisu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXAC3:  INC     BX
        LOOP    TXAC1
;komunikacja 16-bitowa
        MOV     BX,OFFSET TEKST6
        CALL    OUTTXT
        POP     BX
        MOV     CX,1000H
TXAC4:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset C1 IT1
        MOV     AL,40H
        OUT     DX,AL           ;wycofanie resetu
        MOV     AX,[BX]         ;odczyt RAMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXAC5           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKSTC ;brak XACK dla odczytu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXAC5:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset C1 IT1
        MOV     AL,40H
        OUT     DX,AL           ;wycofanie resetu
        MOV     [BX],AX         ;zapis RAMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXAC6           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKSTH ;brak XACK dla zapisu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXAC6:  INC     BX
        INC     BX
        LOOP    TXAC4
;test PROMu
        MOV     BX,OFFSET TEKSTE
        CALL    OUTTXT
        CALL    ADRES           ;pobranie adresu PROMu do DS:BX
        PUSH    BX
;komunikacja 8-bitowa
        PUSH    BX
        MOV     BX,OFFSET TEKSTF
        CALL    OUTTXT
        POP     BX
        MOV     CX,8000H
TXAC7:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset C1 IT1
        MOV     AL,40H
```

```
        OUT     DX,AL           ;wycofanie resetu
        MOV     AL,[BX]         ;odczyt PROMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXAC8           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKST0 ;brak XACK dla odczytu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXAC8:  INC     BX
        LOOP    TXAC7
;komunikacja 16-bitowa
        MOV     BX,OFFSET TEKST6
        CALL    OUTTXT
        POP     BX
        MOV     CX,4000H
TXAC9:  MOV     DX,IT1B3
        MOV     AL,8BH
        OUT     DX,AL           ;zaprogramowanie bramy IT1B3
        MOV     DX,IT1D6WY
        MOV     AL,0
        OUT     DX,AL           ;zaprogramowanie reset CI IT1
        MOV     AL,40H
        OUT     DX,AL           ;wycofanie resetu
        MOV     AX,[BX]         ;odczyt PROMu
        MOV     DX,IT1B3PB
        IN      AL,DX
        TEST    AL,20H
        JNZ     TXACA           ;jest XACK - OK
        PUSH    BX
        MOV     BX,OFFSET TEKST0 ;brak XACK dla odczytu
        CALL    OUTTXT
        POP     BX
        MOV     AX,BX
        CALL    OUTADR
        CALL    CZY_KONIEC
TXACA:  INC     BX
        INC     BX
        LOOP    TXAC9
        JMP     ML500
ml50_   ENDP
;
;PODPROGRAMY
;
;Pobranie adresu z klawiatury do DS:BX
;gdy brak segmentu to DS=0 - niszczy AX
ADRES   PROC    NEAR
        MOV     BX,OFFSET TEKST4
        CALL    OUTTXT
        SUB     AX,AX
        MOV     DS,AX
        CALL    GETADR
        CMP     AL,':'
        JZ      ADRES1
        RET
ADRES1: MOV     DS,BX
        CALL    GETADR
        CMP     AL,':'
        JZ      ADRES
        RET
ADRES   ENDP
;Badanie czy wcisnieto CTRL/Z - gdy tak to skok do ML500
CZY_STOP        PROC    NEAR
        PUSH    AX
        IN      AL,0DAH
        TEST    AL,2
        JNZ     CZYST1          ;jest znak
        POP     AX
        RET
CZYST1: IN      AL,0D8H
        CMP     AL,1AH          ;czy CTRL/Z
        POP     AX
        JZ      CZYST2          ;tak
        RET
CZYST2: POP     AX              ;wyrownanie stosu
        JMP     ML500
CZY_STOP        ENDP
;Druk zawartosci rej. BL w postaci binarnej:
BINA08  PROC    NEAR
        PUSH    CX
        PUSH    AX
        MOV     CX,8
BINAR1: MOV     AL,30H
        SHL     BL,1
        ADC     AL,0
        CALL    OUTCHR
        LOOP    BINAR1
        CALL    SPACJA
        POP     AX
        POP     CX
        RET
BINA08  ENDP
;Druk zawartosci rej. BX w postaci binarnej
BINA16  PROC    NEAR
        XCHG    BH,BL
        CALL    BINA08
        XCHG    BH,BL
        CALL    BINA08
```

```
        CALL    SPACJA
        RET
BINA16  ENDP
;Pobranie kolejnego bajtu z generatora pseudolosowego
GENERUJ PROC    NEAR
        PUSH    CX
        PUSH    BX
        MOV     BX,BP
        MOV     AL,0
        MOV     CX,8
GENER1: TEST    BL,1
        JZ      GENER2
        STC
GENER2: RCL     BH,1
        MOV     BL,0
        JC      GENER9
        TEST    BH,10H
        JZ      GENER4
GENER3: INC     BX
        STC
GENER4: RCR     AL,1
        LOOP    GENER1
        MOV     BP,BX
        POP     BX
        POP     CX
        RET
GENER9: TEST    BH,10H
        JNZ     GENER4
        JMP     GENER3
GENERUJ ENDP
;Pobranie kolejnego slowa z generatora pseudolosowego
GENERJ2 PROC    NEAR
        CALL    GENERUJ
        MOV     AH,AL
        CALL    GENERUJ
        XCHG    AL,AH
        RET
GENERJ2 ENDP
;Wykrycie konca lub kontynuacji testu po bledzie
;kropka konczy test, kazdy inny znak kontynuuje
CZY_KONIEC      PROC    NEAR
        PUSH    AX
        CALL    GETCHR
        CMP     AL,'.'
        POP     AX
        JZ      CZYKON
        RET
CZYKON: JMP     ML500
CZY_KONIEC      ENDP
;Wydruk informacji o bledzie dla testow 8-bitowych
;adres w BX, dane poprawne w AL, dane aktualne wg [BX]
;niszczy AL
BLAD08  PROC    NEAR
        CALL    LFCROT
        XCHG    AX,BX
        CALL    OUTADR          ;druk adresu przeklamanej komorki
        CALL    BINA08          ;druk danych poprawnych
        PUSH    AX
        MOV     BX,AX
        MOV     BL,[BX]
        CALL    BINA08          ;druk danych zpamieci
        POP     BX
        CALL    CZY_KONIEC
        RET
BLAD08  ENDP
;Wydruk informacji o bledzie dla testow 16-bitowych
;adres w BX, dane poprawne w AX, dane aktualne wg [BX]
;niszczy AX
BLAD16  PROC    NEAR
        CALL    LFCROT
        XCHG    AX,BX
        CALL    OUTADR          ;druk adresu przeklamanej komorki
        CALL    BINA16          ;druk danych poprawnych
        PUSH    AX
        MOV     BX,AX
        MOV     BX,[BX]
        CALL    BINA16          ;druk danych z pamieci
        POP     BX
        CALL    CZY_KONIEC
        RET
BLAD16  ENDP
;Wydruk informacji obledzie dla testu uplywnosci
BLADUP  PROC    NEAR
        MOV     AL,DL
        CALL    BLAD08
        RET
BLADUP  ENDP
;Opoznienie 1 sek.
SEKUNDA PROC    NEAR
        PUSH    CX
        MOV     CX,10000
        CALL    DELAY
        POP     CX
        RET
SEKUNDA ENDP
;Wydruk zawartosci AX ze spacja na koncu
OUTADR  PROC    NEAR
        PUSH    ES
        PUSH    AX
        MOV     AX,CODESEG
        MOV     ES,AX
        POP     AX
```

```
        CALL    OUTWOR
        CALL    SPACJA
        POP     ES
        RET
OUTADR  ENDP
;
;PP POBIERZ ZNAK Z KLAWIATURY DAJAC ECHO
;GDY KOD ASCII ZNAKU WIEKSZY OD 1FH
GEZNAK  PROC NEAR
GEZNA1: IN      AL,0DAH
        TEST    AL,2
        JZ      GEZNA1          ;gdy nie RxRdy
        IN      AL,0D8H
        AND     AL,7FH
        CMP     AL,7FH
        JE      GEZNA2          ;GDY DEL
        CMP     AL,20H
        JB      GEZNA2
        CALL    OUTCHR
GEZNA2: RET
SEZNAK  ENDP
;
;PP POBIERZ ZNAK Z KLAWIATURY ZERUJAC BIT 5
;(zamiana liter malych na DUZE) dajac echo gdy kod ASCII znaku wiekszy od 1FH
GETCHR  PROC    NEAR
        CALL    GEZNAK
        CMP     AL,'a'
        JB      GETCH1          ;gdy kod znaku mniejszy od 'a' bez zerowania bitu
        CMP     AL,'z'
        JA      GETCH1          ;gdy kod znaku wiekszy od 'z' bez zerowania bitu
        AND     AL,0DFH         ;zerowanie bitu D5 dla malych liter
GETCH1: RET
SETCHR  ENDP
;
;PP WYSLIJ ZNAK Z AL DO USART'A
OUTCHR  PROC    NEAR
        PUSH    AX
OUTCH1: IN      AL,0DAH
        TEST    AL,1
        JZ      OUTCH1
        POP     AX
        OUT     0D8H,AL
        RET
OUTCHR  ENDP
;
;PP WYSLIJ BAJT Z AL JAKO 2 ZNAKI W KODZIE ASCII DO USART'A
OUTBYT  PROC    NEAR
OUTBYC: PUSH    ES
        PUSH    BX
        PUSH    CX
        PUSH    AX
        PUSH    AX
        MOV     BX,CODESEG
        MOV     ES,BX
        MOV     BL,AL
        MOV     BH,0
        MOV     CL,4
        SHR     BL,CL
        MOV     CX,2
OUTBY1: AND     BL,0FH
        MOV     AL,ES:ASCII [BX]
        CALL    OUTCHR
OUTBY2: POP     AX
        MOV     BL,AL
        LOOP    OUTBY1
        POP     CX
        POP     BX
        POP     ES
        RET
OUTBYT  ENDP
;
;PP WYSLIJ SLOWO Z AX DO USART'A JAKO 4 ZNAKI W KODZIE ASCII
OUTWOR  PROC    NEAR
        PUSH    AX
        MOV     AL,AH
        CALL    OUTBYT
        POP     AX
        CALL    OUTBYT
        RET
OUTWOR  ENDP
;
;PP WYSLIJ DO USART'A LF I CR
LFCROT  PROC    NEAR
        PUSH    AX
        MOV     AL,0AH
        CALL    OUTCHR
        MOV     AL,0DH
        CALL    OUTCHR
        POP     AX
        RET
LFCROT  ENDP
;
;PP ZAMIANY ZNAKU Z AL W KODZIE ASCII NA ZNAK HEX W AL
;GDY OK TO CY=0, GDY BLEDNY ZNAK CY=1.
;ZACHOWUJE REJESTRY
CHRHEX  PROC    NEAR
        PUSH    BX
        PUSH    CX
        PUSH    ES
        MOV     BX,CODESEG
        MOV     ES,BX
        MOV     BX,0
```

```
        MOV     CX,16
CHRHE1: CMP     AL,ES:ASCII[BX]
        JE      CHRHE3
        INC     BX
        LOOP    CHRHE1
        STC
CHRHE2: POP     ES
        POP     CX
        POP     BX
        RET
CHRHE3: MOV     AL,BL
        JMP     CHRHE2
CHRHEX  ENDP
;
;PP WYDRUKU TEKSTU DO ZNAKU 0 WZGLEDEM ES
;ADRES POCZATKU TEKSTU W BX
OUTTXT  PROC    NEAR
OUTTX1: MOV     AL,ES:[BX]
        AND     AL,AL
        JZ      OUTTX2
        CALL    OUTCHR
        INC     BX
        JMP     OUTTX1
OUTTX2: RET
OUTTXT  ENDP
;
;PP pobrania adresu (do 4 znaki Hex) do BX
;Znak konczacy w AL. Ilosc wprowadzonych cyfr w DH
GETADR  PROC    NEAR
        CALL    GETCHR
GETAD0: PUSH    CX
        MOV     BX,0
        MOV     DH,0
GETAD1: PUSH    AX
        CALL    CHRHEX
        JC      GETAD2          ;znak nie hex
        INC     DH              ;licznik wprowadzonych cyfr hex
        MOV     CL,4
        SHL     BX,CL           ;kolejna cyfre wprowadzamy do wyniku
        OR      BL,AL
        POP     AX
        CALL    GETCHR          ;pobieramy kolejny znak do AL
        JMP     GETAD1
GETAD2: POP     AX
        POP     CX
        RET
GETADR  ENDP
;
;PP DRUKOWANIA SPACJI
SPACJA  PROC    NEAR
        PUSH    AX
        MOV     AL,20H
        CALL    OUTCHR
        POP     AX
        RET
SPACJA  ENDP
;
;PP OPOZNIENIA PROGRAMOWEGO - JEDNOSTKA 100 mikrosek.
;ILOSC JEDNOSTEK W CX. ZACHOWUJE REJESTRY
DELAY   PROC    NEAR
        PUSH    CX
DELAY2: PUSH    CX
        MOV     CL,68H
        SAL     CX,CL
        POP     CX
        LOOP    DELAY2
        POP     CX
        RET
DELAY   ENDP
;
ASCII   DB      '0123456789ABCDEF'
;
;
;=============================================================
;
TEKST1  DB      0AH,0AH,0DH,'TESTY URUCHOMIENIOWE PAKIETU ML50'
TEKST2  DB      0AH,0DH,'1 - TESTY REPETYCYJNE'
        DB      0AH,0DH,'2 - TESTY DIAGNOSTYCZNE'
        DB      0AH,0DH,'. - KONIEC TESTOW'
        DB      0AH,0DH,':',0
TEKST3  DB      0AH,0DH,'1 - ODCZYT 8-mio BITOWY'
        DB      0AH,0DH,'2 - ODCZYT 16-to BITOWY'
        DB      0AH,0DH,'3 - ZAPIS 8-mio BITOWY'
        DB      0AH,0DH,'4 - ZAPIS 16-to BITOWY'
        DB      0AH,0DH,'. - KONIEC TESTOW'
        DB      0AH,0DH,':',0
TEKST4  DB      0AH,0DH,'PODAJ ADRES:',0
TEKST5  DB      0AH,0DH,'PODAJ DANE:',0
TEKST6  DB      0AH,0DH,'1 - TEST PROM (8-bitowy)'
        DB      0AH,0DH,'2 - TEST PROM (16-bitowy)'
        DB      0AH,0DH,'3 - TEST UPLYWNOSCI RAM'
        DB      0AH,0DH,'4 - TEST INFORMACJI RAM'
        DB      0AH,0DH,'5 - TEST SYGNALU XACK'
        DB      0AH,0DH,'. - KONIEC TESTOW'
        DB      0AH,0DH,':',0
TEKST7  DB      0AH,0DH,7,'KONIEC TESTU'
        DB      0
TEKST8  DB      0AH,0DH,'PODAJ OPOZNIENIE (W SEK.):',0
TEKST9  DB      0AH,0DH,'Zapis jedynek',0
TEKSTA  DB      0AH,0DH,'Opoznienie',0
TEKSTB  DB      0AH,0DH,'Zapis zer',0
TEKSTC  DB      0AH,0DH,'BRAK XACK DLA ODCZYTU Z ADRESU: ',0
```

```
TEKSTD  DB      0AH,0DH,' RAM:',0
TEKSTE  DB      0AH,0DH,' PROM:',0
TEKSTF  DB      0AH,0DH,'KOMUNIKACJA 8-BITOWA',0
TEKSTG  DB      0AH,0DH,'KOMUNIKACJA 16-BITOWA',0
TEKSTH  DB      0AH,0DH,'BRAK XACK DLA ZAPISU POD ADRES: ',0
;
RAMT2   DB      LF,CR,'Plywajaca jedynka',0
RAMT3   DB      'F,CR,'Plywajace zero',0
RAMT4   DB      LF,CR,'Po wpisie ',0
RAMT5   DB      ' do komorki ',0
RAMT6   DB      ' odczytano ',0
RAMT7   DB      ' z komorki ',0
RAMT8   DB      LF,CR,'Blad dla adresu ',0
RAMT9   DB      ' zapisano ',0
RAMTA   DB      LF,CR,7,'BLOK 1',0
RAMTB   DB      'F,CR,7,'BLOK 2',0
RAMTC   DB      LF,CR,7,'BLOK 3',0
RAMTD   DB      !F,CR,7,'BLOK 4',0
;
CODESEG ENDS
        END
```

PAKIET PROGRAMOW

DO TESTOWANIA PAKIETU MI-50

```
;
;  Testy uruchomieniowe pakietu MI50 15.01.88
;
;            Wersja dla AZTECa
;
      PUBLIC  mi50_
;
LF EQU 10
CR EQU 13
;
DATASEG    SEGMENT
ADRESP  DW      10 DUP(?)               ;pole na adresy pakietu
NRPRZE  DB      1 DUP(?)                ;pole ne numer przewijaka
MEM_SP  DW      1 DUP(?)                ;pole na pamietanie SP
DATASEG    ENDS
;
;
CODESEG     SEGMENT PUBLIC
        ASSUME  CS:CODESEG, DS:DATASEG, ES:CODESEG
;
mi50_   PROC
        PUSH    ES
        PUSH    DS
        MOV     AX,CODESEG
        MOV     ES,AX
        MOV     AX,DATASEG
        MOV     DS,AX
        MOV     MEM_SP,SP
        MOV     BYTE PTR ADRESP,20H
        MOV     BX,OFFSET INFO1
        CALL    OUTTXT
;    ustawienie sygnalu INIT
;
        PUSH    DX
        PUSH    AX
        MOV     AX,020h
        MOV     DX,0fed0h
        OUT     DX,AL
        POP     AX
        POP     DX
MI500:  MOV     BX,OFFSET INFO4
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,CR
        JNE     MI504
MI499:  MOV     AL,BYTE PTR ADRESP      ;okreslenie adresow rejestrow pakietu
        MOV     BYTE PTR ADRESP[1],AL   ;w kolejnych slowach pola ADRESP
        ADD     AL,2
        MOV     AH,AL
        MOV     ADRESP[2],AX
        ADD     AX,202H
        MOV     ADRESP[4],AX
        ADD     AX,202H
        MOV     ADRESP[6],AX
        ADD     AX,202H
        MOV     ADRESP[8],AX
;
MI501:  MOV     SP,MEM_SP
        MOV     BX,OFFSET INFO2
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,','
        JNZ     MI502
MI50A:  POP     DS                      ;wyjscie z testow
        POP     ES
        RET
MI504:  CMP     AL,'.'
        JE      MI50A                   ;kropka - koniec testu
        CALL    GETADD
        CMP     DH,2
        JC      MI500                   ;za malo cyfr w adresie
        AND     BL,0F0H
        MOV     BYTE PTR ADRESP,BL
        JMP     MI499
MI502:  CMP     AL,'1'
        JZ      MI510                   ;testy rejestrow
        CMP     AL,'2'
        JNZ     MI501
        JMP     PKTEST                  ;testy wspolpracujace z PK
;
; Testy rejestrow pakietu MI50
;
MI510:  MOV     BX,OFFSET INFO3
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,','
        JZ      MI501
        CMP     AL,'4'
        JNC     MI510                   ;podana liczba wieksza od 4
        CMP     AL,'1'
        JB      MI510                   ;podana liczba mniejsza od 1
        MOV     CL,AL
        CMP     AL,'4'
        JC      MIREJ                   ;testy 1,2,3
;
;Repetycyjne testy rejestrow pakietu
;
MIREJ:  MOV     BX,OFFSET INFO5
        CALL    OUTTXT
        CALL    GETCHR
        CMP     AL,'.'
```

```
        JNE     MIRE0
        JMP     MI510
MIRE0:  CMP     AL,'6'
        JNC     MIREJ
        DEC     AL
        CALL    CHRHEX
        JC      MIREJ
        ADD     AL,AL           ;4 mlodsze bity zawieraja adres rejestru na pakiecie (0,2,4,6,8)
        CMP     CL,'1'
        JNZ     MIRE1
        CMP     AL,6
        JC      MIRE1
        MOV     BX,OFFSET INF06
        CALL    OUTTXT
        JMP     MIREJ
MIRE1:  OR      AL,BYTE PTR ADRESP
        MOV     DL,AL
        MOV     DH,AL           ;w DX pelny adres rejestru
        CMP     CL,'1'
        JZ      MIRE3           ;test odczytu
MIRE9:  MOV     BX,OFFSET INF07 ;pobranie informacji do wysylania
        CALL    OUTTXT
        CALL    GETADR
        CMP     AL,'.'
        JZ      MIREJ           ;kropka - koniec testu
        CMP     DH,0
        JE      MIRE9           ;nie bylo zadnej cyfry
        MOV     AL,BL
;
MIRE2:  OUT     DX,AL
        CALL    CZY_STOP
        CMP     CL,'2'
        JZ      MIRE2           ;test informacji stalej
        NOT     AL              ;test informacji naprzemiennej
        JMP     MIRE2
;
MIRE3:  IN      AL,DX
        CALL    CZY_STOP
        JMP     MIRE3
;
;Inicjacja USART'a na pakiecie
;internal reset+zapis+odczyt+eh+er
USART   PROC    NEAR
        PUSH    AX
        PUSH    DX
        CALL    OPOZ10
        MOV     AL,02H
        MOV     DX,ADRESP[2]
        OUT     DX,AL
        CALL    OPOZ10
        MOV     AL,40H
        OUT     DX,AL
        CALL    OPOZ10
        MOV     AL,9CH
        OUT     DX,AL
        CALL    OPOZ10
        MOV     AL,0AAH
        OUT     DX,AL
        CALL    OPOZ10
        MOV     AL,95H
        OUT     DX,AL
        CALL    OPOZ10
        POP     DX
        POP     AX
        RET
USART   ENDP
;Badanie czy wcisnieto CTRL/Z
;gdy tak to skok do MI501
CZY_STOP        PROC    NEAR
        PUSH    AX
        IN      AL,0DAH
        TEST    AL,2
        JNZ     CZYST1          ;jest znak
        POP     AX
        RET
CZYST1: IN      AL,0DBH
        CMP     AL,1AH          ;czy CTRL/Z
        POP     AX
        JZ      CZYST2          ;tak
        RET
CZYST2: POP     AX
        CALL    USART           ;zerowanie USART'a
        MOV     DX,ADRESP[4]
        XOR     AL,AL           ;zdjecie sygnalow WCD i RST (zewolenie czytania
        OUT     DX,AL           ;i pisania)
        JMP     MI501
CZY_STOP        ENDP
;
;Opoznienie 10 milisekund
OPOZ10  PROC    NEAR
        PUSH    CX
        MOV     CX,0BB6H
OPOZ11: NOP
        LOOP    OPOZ11
        POP     CX
        RET
OPOZ10  ENDP
;
;Pobranie bajtu z klawiatury do BL
;gdy OK to CY=0, gdy blad to CY=1
;gdy kropka jako pierwszy znak to Z=1
GETBYT  PROC    NEAR
```

```
        PUSH    AX
        CALL    GETCHR
GETBY0: CMP     AL,'.'
        JZ      GETBY1          ;kropka konczy
        CALL    CHRHEX
        JC      GETBY1          ;blad
        PUSH    CX
        MOV     CL,4
        SHL     AL,CL
        POP     CX
        MOV     BL,AL
        CALL    GETCHR
        CALL    CHRHEX
        JC      GETBY1          ;blad
        OR      BL,AL
        MOV     AL,1            ;ustawienie flagi Z=0 CY=0
        AND     AL,AL
GETBY1: POP     AX
        RET
GETBYT  ENDP
;
;Teksty stale
;
ASCII   DB      '0123456789ABCDEF'
OK      DB      0AH,0DH,'OK',0
INF01   DB      0AH,0DH,0AH,'TESTY URUCHOMIENIOWE PAKIETU MI50'
        DB      0
INF02   DB      0AH,0DH,'1 - TESTY REJESTROW PAKIETU'
        DB      0AH,0DH,'2 - TESTY WSPOLPRACY Z PK'
        DB      0AH,0DH,'. - KONIEC TESTOW'
        DB      0AH,0DH,':',0
INF03   DB      0AH,0DH,'1 - ODCZYT REJESTRU'
        DB      0AH,0DH,'2 - ZAPIS DO REJESTRU INFORMACJI STALEJ'
        DB      0AH,0DH,'3 - ZAPIS DO REJESTRU INFORMACJI NAPRZEMIENNEJ'
        DB      0AH,0DH,':',0
INF04   DB      0AH,0DH,'PODAJ ADRES PAKIETU - 2 CYFRY HEX [20]: ',0
INF05   DB      0AH,0DH,'1 - REJESTR DANYCH'
        DB      0AH,0DH,'2 - S.S. USART-A'
        DB      0AH,0DH,'3 - S.S. PK'
        DB      0AH,0DH,'4 - REJESTR LICZNIKA'
        DB      0AH,0DH,'5 - REJESTR ZEZWOLEN'
        DB      0AH,0DH,':',0
INF06   DB      '   ODCZYT REJESTRU NIEDOZWOLONY',0
INF07   DB      0AH,0DH,'PODAJ INFORMACJE (2 CYFRY HEX): ',0
;
;  Testy MI50 wspolpracujace z PK  15.01.88
;
PKTEST: MOV     BX,OFFSET INF0D
        CALL    OUTTXT
        CALL    GETCHR          ;pobranie nr przewijaka
        CMP     AL,'.'
        JNZ     MI521
        JMP     MI501           ;kropka konczy test
MI521:  SUB     AL,30H
        CMP     AL,1
        JC      PKTEST          ;nr przewijaka < 1
        CMP     AL,3
        JNC     PKTEST          ;nr przewijaka > 2
        NOT     AL
        AND     AL,3
        MOV     NRPRZE,AL       ;nr przewijaka w konwencji pakietu w NRPRZE
MI522:  MOV     CL,NRPRZE
        MOV     BX,OFFSET INF0C
        CALL    OUTTXT
        CALL    GETCHR          ;wybor testu
        CMP     AL,'.'
        JNZ     MI520
        JMP     MI501           ;kropka konczy testy wspolpracy z PK
MI520:  CMP     AL,'1'
        JNZ     MI528
        JMP     PK10
MI528:  CMP     AL,'2'
        JNZ     MI523
        JMP     PK20
MI523:  CMP     AL,'3'
        JNZ     MI524
        JMP     PK30
MI524:  CMP     AL,'4'
        JNZ     MI525
        JMP     PK40
MI525:  CMP     AL,'5'
        JNZ     MI526
        JMP     PK50
MI526:  CMP     AL,'6'
        JNZ     MI527
        JMP     PK60
MI527:  CMP     AL,'7'
        JNZ     MI522           ;zly wybor testu
;
;Zliczanie blokow
PK70:   MOV     BX,OFFSET INF0I
        CALL    OUTTXT          ;kierunek przewijania
        CALL    GETCHR
        CMP     AL,'.'
        JNE     PK72
        JMP     MI522
PK72:   AND     AL,0DFH         ;zamiana malych liter na duze
        MOV     BL,4
        CMP     AL,'P'
        JE      PK71            ;przewijanie do przodu
        CMP     AL,'T'
        JNE     PK70            ;ani do przodu ani do tylu
```

```
        MOV     BL,9
PK71:   PUSH    BX
PK73:   MOV     BX,OFFSET INF0L
        CALL    OUTTXT
        CALL    GETBYT
        JC      PK73            ;blad
        JNZ     PK74
        POP     BX
        JMP     MI522           ;kropka konczy test
PK74:   MOV     CH,BL           ;dane do CL
        POP     BX
        CALL    PRZYDZ
        JZ      PK75            ;pamiec przydzielona - OK
        JMP     PK14
PK75:   MOV     DX,ADRESP[6]
        MOV     AL,CH
        OUT     DX,AL           ;ilosc zliczonych blokow
        MOV     DX,ADRESP[8]
        MOV     AL,1
        OUT     DX,AL           ;zezwolenie na zliczanie blokow
        MOV     AL,CL
        OR      AL,BL
        OR      AL,10H
        MOV     DX,ADRESP[4]
        OUT     DX,AL
PK76:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,40H
        JNZ     PK77            ;zliczono zadana liczbe blokow
        TEST    AL,20H
        JZ      PK76            ;nie koniec tasmy
        JMP     PK18
PK77:   SUB     AL,AL
        OUT     DX,AL           ;zatrzymanie tasmy, zwolnienie przewijaka
        MOV     BX,OFFSET INF0M
        CALL    OUTTXT
        JMP     PK70
;
;Repetycyjny zapis blokow: SYNC,ZNAK,SYNC
PK10:   MOV     BX,OFFSET INF07
        CALL    OUTTXT
        CALL    GETBYT          ;pobranie danych
        JC      PK10
        JNZ     PK1A
        JMP     MI522           ;kropka konczy test PK10
PK1A:   CALL    PRZYDZ
        JZ      PK15            ;pamiec przydzielona - OK
PK14:   MOV     BX,OFFSET INF0E ;PK nie przydzielona
PK16:   CALL    OUTTXT
        JMP     MI522
PK15:   IN      AL,DX           ;w DX adres rej. sterujacego po PRZYDZ
        TEST    AL,10H
        JNZ     PK11            ;zapis dozwolony - OK
PK17:   MOV     BX,OFFSET INF0F ;zapis niedozwolony
        JMP     PK16
PK11:   MOV     AL,CL
        OR      AL,0C4H         ;odczyt,zapis,ruch w przod,sel
        OUT     DX,AL
        CALL    SEKUND
        JNC     PK13
PK18:   MOV     DX,ADRESP[4]
        XOR     AL,AL
        OUT     DX,AL           ;zwolnienie przewijaka
        MOV     BX,OFFSET INF0J ;koniec tasmy
        CALL    OUTTXT
        JMP     MI522
PK13:   CALL    USART
        MOV     DX,ADRESP[8]
        MOV     AL,0
        OUT     DX,AL           ;zezwolenie nadawania do rej. zezwolen
        MOV     DX,ADRESP[4]
PK12:   CALL    CZY_END
        MOV     AL,0AAH
        CALL    OUT_PK          ;pierwszy znak SYNC
        CALL    CZY_END
        MOV     AL,BL
        CALL    OUT_PK          ;bajt danych
        CALL    CZY_END
        MOV     AL,0AAH
        CALL    OUT_PK          ;drugi znak SYNC
        IN      AL,DX
        TEST    AL,20H
        JZ      PK12            ;nie koniec tasmy
PK19:   CALL    USART
        JMP     PK18
;PP wyslania bajtu danych do PK z AL
;w DX adres rej. stanu USART'a
;w BP adres rej. danych USART'a
OUT_PK  PROC    NEAR
        PUSH    DX
        PUSH    AX
        MOV     DX,ADRESP[2]
OUTPK1: IN      AL,DX
        TEST    AL,1
        JZ      OUTPK1
        MOV     DX,ADRESP
        POP     AX
        OUT     DX,AL
        POP     DX
        RET
OUT_PK  ENDP
;
```

```
;Repetycyjny odczyt blokow
PK20:   CALL    PRZYDZ
        JZ      PK21            ;pamiec przydzielona - OK
        JMP     PK14
PK21:   CALL    USART
        MOV     AL,CL
        OR      AL,44H
        OUT     DX,AL           ;w przod + odczyt + sel
        MOV     DX,ADRESP[2]
        MOV     CX,ADRESP
PK22:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,2
        JZ      PK22            ;nie ma RxRdy
        XCHG    CX,DX
        IN      AL,DX           ;odczyt znaku
        XCHG    CX,DX
        JMP     PK22
;
;Zapis kolejnych blokow danych
PK30:   CALL    PRZYDZ
        JZ      PK31            ;pamiec przydzielona - OK
        JMP     PK14
PK31:   IN      AL,DX
        TEST    AL,10H
        JNZ     PK37
        JMP     PK17            ;zapis niedozwolony
PK37:   MOV     AL,CL
        OR      AL,0C4H         ;odczy + zapis + ruch w przod + sel.
        OUT     DX,AL
        CALL    SEKUND
        JNC     PK35
        JMP     PK19            ;koniec tasmy
PK35:   CALL    USART
        MOV     SI,ADRESP[0]
        MOV     BP,ADRESP[2]
        MOV     DX,ADRESP[4]
        MOV     BL,1            ;dane pierwszego zapisywanego bloku
;petla zapisywania blokow:
PK32:   XCHG    DX,SI
        MOV     AL,0
        OUT     DX,AL           ;zezwolenie nadawania do rej.zezwolen
        XCHG    DX,SI
        MOV     CX,100h       ;w CX dlugosc zapisywanego bloku
        CALL    CZY_END
        MOV     AL,0AAH
        CALL    OUT_PK          ;pierwszy znak SYNC
PK33:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,20H
        JZ      PK38
        JMP     PK19            ;koniec tasmy
PK38:   MOV     AL,BL
        CALL    OUT_PK          ;kolejny bajt danych
        LOOP    PK33            ;nie wszystkie bajty w bloku
;
        CALL    CZY_END
        MOV     AL,0AAH
        CALL    OUT_PK          ;SYNC konczace blck
        XCHG    BP,DX
PK34:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,4
        JZ      PK34            ;nie Empty
        MOV     AL,40h
        OUT     DX,AL
        XCHG    DX,BP
        XCHG    SI,DX
        MOV     AL,4
        OUT     DX,AL           ;zakaz nadawania do rej. zezwolen
        XCHG    SI,DX
        CALL    USART
        CALL    SEKUND          ;przerwa miedzy-blokowa
        JNC     PK36
        JMP     PK19            ;koniec tasmy
PK36:   CALL    CZY_END
        INC     BL
        CMP     BL,0AAH
        JNE     PK32
        INC     BL              ;dane wysylane nie moga byc rowne AA
        JMP     PK32
;
;Odczyt kolejnych blokow danych
PK40:   CALL    PRZYDZ
        JZ      PK41            ;PK przydzielona - OK
        JMP     PK14
PK41:   CALL    USART
        MOV     AL,CL
        OR      AL,44H          ;w przod, odczyt
        OUT     DX,AL
        MOV     DX,ADRESP[2]
        MOV     CX,ADRESP
PK47:   MOV     BP,1
PK42:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,40H          ;SYNC
        JZ      PK42            ;czekamy na pierwsze SYNC
        TEST    AL,2            ;RxRDY
        JZ      PK43
        XCHG    CX,DX
        IN      AL,DX           ;odczyt SYNC
        XCHG    CX,DX
```

```
PK43:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,40H
        JNZ     PK46            ;koniec bloku - drugi SYNC - za wczesnie
        TEST    AL,2
        JZ      PK43            ;nie RxRdy
        XCHG    CX,DX
        IN      AL,DX           ;odczyt bajtu danych
        XCHG    CX,DX
        MOV     BL,AL           ;pierwszy odczytany bajt w BL
PK44:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,40H
        JNZ     PK46            ;koniec bloku - drugi SYNC
        TEST    AL,2
        JZ      PK44            ;nie RxRdy
        XCHG    CX,DX
        IN      AL,DX           ;odczyt bajtu danych
        XCHG    CX,DX
        CMP     AL,BL           ;kolejny bajt rozny od poprzednio odczytanego
        JZ      PK45
        MOV     BL,0AAH
PK45:   INC     BP              ;zwiekszamy licznik odczytanych bajtow
        JMP     PK44
PK46:   TEST    AL,2
        JZ      PK48            ;nie RxRdy
        XCHG    CX,DX
        IN      AL,DX
        XCHG    CX,DX
PK48:   CALL    USART
        CMP     BL,0AAH
        JZ      PK49            ;blad (zle dane)
        CMP     BP,100h
        JNZ     PK4A            ;blad (zla dlugosc)
        CALL    CRLFOUT         ;blok odczytany OK
        MOV     AL,BL
        CALL    OUTBYT          ;wydruk bajtu danych z odczytanego bloku
        JMP     PK47
PK49:   MOV     BX,OFFSET INFOG ;blad danych
PK4B:   CALL    OUTTXT
        JMP     PK47
PK4A:   MOV     BX,OFFSET INFOH ;zla dlugosc
        JMP     PK4B

;
;Przewijanie tasmy
PK50:   MOV     BX,OFFSET INFOI
        CALL    OUTTXT          ;pytanie o kierunek przewijania
        CALL    GETCHR
        CMP     AL,'.'
        JNE     PK54
        JMP     MI522           ;kropka konczy testy
PK54:   AND     AL,0DFH         ;zamiana malych liter na duze
        MOV     BL,4
        CMP     AL,'P'
        JE      PK51            ;przewijanie do przodu
        CMP     AL,'T'
        JNE     PK50            ;blad - ani do przodu ani do tylu
        MOV     BL,8            ;przewijanie do tylu
PK51:   CALL    PRZYDZ
        JZ      PK53            ;PK przydzielona - OK
        JMP     PK14
PK53:   MOV     AL,CL
        OR      AL,BL
        OR      AL,10H          ;szybki ruch
        MOV     DX,ADRESP[4]
        OUT     DX,AL
PK52:   CALL    CZY_END
        IN      AL,DX
        TEST    AL,20H
        JZ      PK52            ;nie koniec tasmy
        JMP     PK18

;
;Ustawianie tasmy w polozeniu poczatkowym
PK60:   CALL    REVIND
        JNC     PK61            ;OK
        MOV     BX,OFFSET INFOK
        CALL    OUTTXT          ;blad PK
PK61:   MOV     DX,ADRESP[4]
        XOR     AL,AL
        OUT     DX,AL           ;zwolnienie przewijaka
        JMP     MI522

;
;REVIND
;NR PRZEWIJAKA W CL
;
REVIND  PROC    NEAR
        PUSH    DX
        PUSH    AX
        CALL    PRZYDZ
        JNZ     REVIN0          ;gdy blad PK lub zly przewijak
        MOV     AL,CL
        OR      AL,20H
        MOV     DX,ADRESP[4]
        OUT     DX,AL           ;sygnal RVD
        MOV     AL,CL
        OUT     DX,AL           ;zdjecie sygnalu
        PUSH    CX
        MOV     CX,-1           ;w CX licznik time-outu
READY0: IN      AL,DX
        TEST    AL,8
        JNZ     REVIN2          ;gdy READY
        CALL    OPOZ10
```

```
        LOOP    READY0
        POP     CX              ;time-out
REVIN0: STC
REVIN1: POP     AX
        POP     DX
        RET
REVIN2: POP     CX
        MOV     AL,CL
        OUT     DX,AL
        OR      AL,4
        OUT     DX,AL           ;ruch w przod
BET001: IN      AL,DX
        TEST    AL,20H
        JNZ     BET001          ;czekamy na koniec rozbiegowki
BET002: IN      AL,DX
        TEST    AL,20H
        JZ      BET002          ;czekamy na BET
BET003: IN      AL,DX
        TEST    AL,20H
        JNZ     BET003          ;czekamy na koniec BET
        MOV     AL,CL
        OUT     DX,AL           ;zatrzymanie tasmy
REVIN4: CLC
        JMP     REVIN1
REVIND  ENDP
;
;PP przydzielenia PK wg CL. Gdy OK to Z=1
;Niszczy AL,CH,DX
PRZYDZ  PROC    NEAR
        MOV     AL,CL
        MOV     DX,ADRESP[4]
        OUT     DX,AL           ;sygnal SEL
        OR      AL,8
        MOV     CH,AL
        IN      AL,DX
        AND     AL,0BH
        CMP     AL,CH
        JE      PRZYD1          ;PK gotowa
        CALL    SEKUN1
        IN      AL,DX
        AND     AL,0BH
        CMP     AL,CH
PRZYD1: RET
PRZYDZ  ENDP
;Badanie czy wcisniete CTRL/Z
;gdy tak to zatrzymanie PK inicjacja USARTa i skok do MI522
CZY_END PROC    NEAR                                    ;CZY_END
        PUSH    AX
        IN      AL,0DAH
        TEST    AL,2
        JNZ     CZYEN1
        POP     AX
        RET
CZYEN1: IN      AL,0D8H
        CMP     AL,1AH
        POP     AX
        JZ      CZYEN2
        RET
CZYEN2: PUSH    DX
        MOV     DX,ADRESP[4]
        SUB     AL,AL
        OUT     DX,AL
        CALL    USART
        POP     DX
        POP     AX
        JMP     MI522
CZY_END ENDP
;
;PP opoznienia 1 sek ze sprawdzaniem czy byl BET
;Gdy BET to CY=1
;Wykorzystuje licznik MI50
SEKUND  PROC    NEAR
        PUSH    DX
        PUSH    AX
        MOV     AL,100
        MOV     DX,ADRESP[6]
        OUT     DX,AL           ;ustawienie licznika
        MOV     AL,2
        MOV     DX,ADRESP[8]
        OUT     DX,AL           ;zliczanie czasu do raj. zezwolen
        MOV     DX,ADRESP[4]
SEKUN0: IN      AL,DX
        TEST    AL,40H
        JNZ     SEKUN2          ;minal czas
        TEST    AL,20H
        JZ      SEKUN0          ;gdy nie BET
        STC                     ;BET
SEKUN2: POP     AX
        POP     DX
        RET
SEKUND  ENDP
;
;PP opoznienia 1 sek. Wykorzystuje licznik MI50
SEKUN1  PROC    NEAR
        PUSH    DX
        PUSH    AX
        MOV     AL,100
        MOV     DX,ADRESP[6]
        OUT     DX,AL           ;ustawienie licznika
        MOV     AL,2
        MOV     DX,ADRESP[8]
        OUT     DX,AL           ;zliczanie czasu do raj. zezwolen
```

```
        MOV     DX,ADRESP[4]
SEKUN3: IN      AL,DX
        TEST    AL,40H
        JZ      SEKUN3          ;nie minal czas
        POP     AX
        POP     DX
        RET
SEKUN1  ENDP
;
INFOC   DB      LF,CR,'1 - REPETYCYJNY ZAPIS BLOKOW'
        DB      LF,CR,'2 - REPETYCYJNY ODCZYT BLOKOW'
        DB      LF,CR,'3 - ZAPIS KOLEJNYCH BLOKOW DANYCH'
        DB      LF,CR,'4 - ODCZYT KOLEJNYCH BLOKOW DANYCH'
        DB      LF,CR,'5 - PRZEWIJANIE TASMY'
        DB      LF,CR,'6 - USTAWIANIE TASMY NA POCZATKU'
        DB      LF,CR,'7 - ZLICZANIE BLOKOW'
        DB      LF,CR,'. - KONIEC TESTU'
        DB      LF,CR,':',0
INFOD   DB      LF,CR,'PODAJ NR PRZEWIJAKA (1/2):',0
INFOE   DB      LF,CR,'PK NIE PRZYDZIELONA',7,0
INFOF   DB      LF,CR,'ZAPIS NIEDOZWOLONY',7,0
INFOG   DB      LF,CR,'BLAD DANYCH',7,0
INFOH   DB      LF,CR,'ZLA DLUGOSC',7,0
INFOI   DB      LF,CR,'PODAJ KIERUNEK PRZEWIJANIA (P/T):',0
INFOJ   DB      LF,CR,'KONIEC TASMY',7,0
INFOK   DB      LF,CR,'BLAD PK',7,0
INFOL   DB      LF,CR,'PODAJ ILOSC BLOKOW DO ZLICZENIA (1-FF):',0
INFOM   DB      LF,CR,'OK',0
;
;  Podprogramy testow ML50 i MI50
;
;PP POBIERZ ZNAK Z KLAWIATURY DAJAC ECHO
;GDY KOD ASCII ZNAKU WIEKSZY OD 1FH
GEZNAK  PROC NEAR
GEZNA1: IN      AL,0DAH
        TEST    AL,2
        JZ      GEZNA1          ;gdy nie RxRdy
        IN      AL,0D8H
        AND     AL,7FH
        CMP     AL,7FH
        JE      GEZNA2          ;GDY DEL
        CMP     AL,20H
        JB      GEZNA2
        CALL    OUTCHR
GEZNA2: RET
GEZNAK  ENDP
;
;PP POBIERZ ZNAK Z KLAWIATURY ZERUJAC BIT 5
;(zamiana liter malych na DUZE) dajac echo gdy kod ASCII znaku wiekszy od 1FH
GETCHR  PROC    NEAR
        CALL    GEZNAK
        CMP     AL,'a'
        JB      GETCH1          ;gdy kod znaku mniejszy od 'a' bez zerowania bitu
        CMP     AL,'z'
        JA      GETCH1          ;gdy kod znaku wiekszy od 'z' bez zerowania bitu
        AND     AL,0DFH         ;zerowanie bitu D5 dla malych liter
GETCH1: RET
GETCHR  ENDP
;
;PP WYSLIJ ZNAK Z AL DO USART'A
OUTCHR  PROC    NEAR
        PUSH    AX
OUTCH1: IN      AL,0DAH
        TEST    AL,1
        JZ      OUTCH1
        POP     AX
        OUT     0D8H,AL
        RET
OUTCHR  ENDP
;
;PP WYSLIJ BAJT Z AL JAKO 2 ZNAKI W KODZIE ASCII DO USART'A
OUTBYT  PROC    NEAR                                            /$ 'MZOJ (
OUTBY0: PUSH    ES
        PUSH    BX
        PUSH    CX
        PUSH    AX
        PUSH    AX
        MOV     BX,CODESEG
        MOV     ES,BX
        MOV     BL,AL
        MOV     BH,0
        MOV     CL,4
        SHR     BL,CL
        MOV     CX,2
OUTBY1: AND     BL,0FH
        MOV     AL,ES:ASCII [BX]
        CALL    OUTCHR
OUTBY2: POP     AX
        MOV     BL,AL
        LOOP    OUTBY1
        POP     CX
        POP     BX
        POP     ES
        RET                                                     /$ 'MZOJ (
OUTBYT  ENDP
;
;PP WYSLIJ SLOWO Z AX DO USART'A JAKO 4 ZNAKI W KODZIE ASCII
OUTWOR  PROC    NEAR
        PUSH    AX
        MOV     AL,AH
        CALL    OUTBYT
```

```
        POP     AX
        CALL    OUTBYT
        RET
OUTWOR  ENDP
;
;PP WYSLIJ DO USART'A LF I CR
LFCROT  PROC    NEAR
        PUSH    AX
        MOV     AL,0AH
        CALL    OUTCHR
        MOV     AL,0DH
        CALL    OUTCHR
        POP     AX
        RET
LFCROT  ENDP
;
;PP ZAMIANY ZNAKU Z AL W KODZIE ASCII NA ZNAK HEX W AL
;GDY OK TO CY=0, GDY BLEDNY ZNAK CY=1.
;ZACHOWUJE REJESTRY
CHRHEX  PROC    NEAR
        PUSH    BX
        PUSH    CX
        PUSH    ES
        MOV     BX,CODESEG
        MOV     ES,BX
        MOV     BX,0
        MOV     CX,16
CHRHE1: CMP     AL,ES:ASCII[BX]
        JE      CHRHE3
        INC     BX
        LOOP    CHRHE1
        STC
CHRHE2: POP     ES
        POP     CX
        POP     BX
        RET
CHRHE3: MOV     AL,BL
        JMP     CHRHE2
CHRHEX  ENDP
;
;PP WYDRUKU TEKSTU DO ZNAKU 0 WZGLEDEM ES
;ADRES POCZATKU TEKSTU W BX
OUTTXT  PROC    NEAR
OUTTX1: MOV     AL,ES:[BX]
        AND     AL,AL
        JZ      OUTTX2
        CALL    OUTCHR
        INC     BX
        JMP     OUTTX1
OUTTX2: RET
OUTTXT  ENDP
;
;PP pobrania adresu (do 4 znaki Hex) do BX
;Znak konczacy w AL, Ilosc wprowadzonych cyfr w DH
GETADR  PROC    NEAR
        CALL    GETCHR
GETAD0: PUSH    CX
        MOV     BX,0
        MOV     DH,0
GETAD1: PUSH    AX
        CALL    CHRHEX
        JC      GETAD2          ;znak nie hex
        INC     DH              ;licznik wprowadzonych cyfr hex
        MOV     CL,4
        SHL     BX,CL           ;kolejna cyfre wprowadzamy do wyniku
        OR      BL,AL
        POP     AX
        CALL    GETCHR          ;pobieramy kolejny znak do AL
        JMP     GETAD1
GETAD2: POP     AX
        POP     CX
        RET
GETADR  ENDP
;
;PP DRUKOWANIA SPACJI
SPACJA  PROC    NEAR
        PUSH    AX
        MOV     AL,20H
        CALL    OUTCHR
        POP     AX
        RET
SPACJA  ENDP
;
;PP OPOZNIENIA PROGRAMOWEGO - JEDNOSTKA 100 mikrosek.
;ILOSC JEDNOSTEK W CX. ZACHOWUJE REJESTRY
DELAY   PROC    NEAR
        PUSH    CX
DELAY2: PUSH    CX
        MOV     CL,48H
        SAL     CH,CL
        POP     CX
        LOOP    DELAY2
        POP     CX
        RET
DELAY   ENDP
;
mi50    ENDP
CODESEG ENDS
        END
```

PAKIET PROGRAMOW

DO TESTOWANIA PAKIETU MA-70

```
/*                      opracowal:  Zbigniew Stanczak                    */

#include <ascii.h>
#include <proext.h>
#include <proto.h>

int menuma7()
{
/* ************************* menu ********************************************** */
static char * naglowek[2] ={
        "* menu I *",
"                       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 = "                                                    * spacja * del * cr *";
int line_n = 12;
static int line_1[12]={14,15,28,13,11,28,37,26,41,26,28,11};
/* ************************* menu ********************************************** */

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

```
/*                      opracował:  Zbigniew Stanczak                    */
#include <proto.h>
#include <asci1.h>
#include <tstglb.h>
static void autotest(),ur_ma70();

main(){
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:
{
#asm
int 10
#endasm
}
               break;
        default:return -1;
   }/* end switch */
 }/* end while */
}/* end main */


static void autotest(){
scr_clr();
  if(dekma70() == 0)
    if(dkma70we() == 0);
       if(eprom() == 0);
          if(ram() == 0);
            if(dac() == 0);
               if(pws() == 0);
                  if(wewy() == 0);
                     if(mp1mp2() == 0){
                        printf("\ntest calkowity\t\t\t\t\t\t\t\tOk!\n");
                        cr();
                        return ;
                     }
printf("\nuszkodzony pakiet ma70");
blad();
cr();
return ;
}

static void ur_ma70(){
int k;
while((k = menu2ma7()) ){
    switch(k){
        case 1:urdac();
               break;
        case 2:udma70();
               break;
        case 3:urdacit3();
               break;
        case 4:return ;
               break;
   }
 }
}
```

```
 /*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>
# include <proto.h>
dekma70()
 {
  int xackpt(),xa,p,k,ZA,kp;
  unsigned long adr=0x0;
  ZA=xa=p=k=kp=0;                                  /* ZA=ZA0,ZA1,ZA2,ZA3 */
  outportb(IT1B1,(char)0x99);
  outportb(IT1B1PA,(char)0x0);
scr_clr();
  printf("\nDEKODER ADRESOW:");
  while(ZA<=0xf)
    {
      outportb(IT3B3WY,(char)ZA);
      adr=0;
      while(adr<=0xff)
       {
        if((xa=xackpt('i',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\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>
# include <proto.h>
# define init 0x0
# define notinit 0x20
# define prominh 0x20
# define hold 0x10
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,(char)(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\tOK!\n");
```

```
    err();
  return(0);
}
                                /*koniec*/
```

```
/*1989-03.23*/
                                /* EPROM.C */
                        /*TESTOWANIE PAMIECI EPROM  */
        /* opracowal dr inz. Marian Wrzesien dnia 1988.10.06 */
# define prominh 0x20
# define hold 0x10
# include <adr.h>
# include <fun.h>
# include <proto.h>
# define adrw 0x807ff1
eprom()
  {
    int dana=0,i=0;
    unsigned long adr,suma;
 fhold();
 outportb(IT3B3WY,(char)(prominh!hold));
 scr_clr();
 printf("PAMIEC EPROM PAKIETU MA70");
 suma=0x01;
 adr=0x800001;
 while((unsigned long)adr<=(unsigned long)adrw)
   {
     dana =  (0xff & mreadb(adr));
     if(dana !=0xff)
     suma += dana;
     adr++;
   }
 printf("\nSuma zawartosci pamieci EPROM jest rowna: 0x%lx\n",suma);
 cr();
 return(0);
}
```

```
/*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>
# include <proto.h>
ram()
 {
  int fhold();
  int datwe,datwy,i;
  unsigned long adr;
  fhold();
  scr_clr();
  printf("PAMIECI RAM PAKIETU 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,(char)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*/
```

```
/*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>
# include <prgto.h>
static int n=2;
static int datzad=0x800;
dac()
  {
   int i,j,k,m,p,kompar,x;
   int datit3,datma;
   fhold();
   scr_clr();
   printf("PRZETWORNIK CYFROWO-ANALOGOWY");
   printf("\nNapiecie wyjsciowe DAC przy wysterowaniu 0x800");
   if(pomiar()==0)
     printf("\tOK !\n");
   if(getch()==0x1b)
     return();
   j=k=0;
   n=2;
   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,(char)(datma&0xff));      /* sterowanie AD na pakiecie MA-70 */
                 outportb(WY4,(char)(datma>>8));        /* j.w. */
                 outportb(IT3B5PA,(char)(datit3&0xff));         /* sterowanie AD na pakiecie it3 */
                 outportb(IT3B5PC,(char)((datit3>>8)|0x30));    /* j.w. pozostaja sygnaly Xack req i Wa
                 delayp(10);
                 kompar=inportb(IT3B5PB)&0x80;
                 if(kompar!=0x80 && n!=0)
                   {
                      printf("\nNapiecie wyjsciowe DAC zbyt duze przy sygnale sterujacym =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 sterujacym =0x%x\n",datza
                      printf("* p * przerywa test\n\t\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*/
pomiar()
  {
     int k=0;
     int p;
     int kompar;
     int datma,datit3;
    for(p=10;p>=0;p--)
    {
      if(datzad<0x800)
        {
          datit3=datzad+p;
          datma =datzad;
        }
      else
        {
          datit3=datzad;
          datma =datzad-p;
        }
```

```
    outportb(WY3,(char)(datma&0xff));
    outportb(WY4,(char)(datma>>8));
    outportb(IT3B5PA,(char)(datit3&0xff));
    outportb(IT3B5PC,(char)((datit3>>8)|0x30));
    delayp(10);
    kompar=inportb(IT3B5PB)&0x80;
    if(kompar!=0x80)
      {
        printf("\nOdchylka nie przekracza %d*4.8828 mV\n",p+1);
        k=1;
        break;
      }
    if(kompar!=0 && p==0)
      {
        printf(" zbyt niskie.\n");
        k=1;
        break;
      }
  }
 return(k);
}
```

```
                        /* 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>
# include <proto.h>
# define CZASPOM 6000
# define intpws() (inportb(WE1)&0x80)
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) + silnik w prawo */
    outportb(IT1B2WY,(char)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 wpisie */
      {
        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*/
                    /* 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(WE4)>>2)&0x1))
# include <wewyma70.h>
# include <adr.h>
# include <fun.h>
# include <proto.h>
   static int p1,p2,p3,p4,p5,p6;
wewy()
  {
   p1=p2=p3=p4=p5=p6=0;
   outportw(KOC,0x0);
   rpr_rzp();
   uklad_wej();
   outportb(IT1B1PB,(char)0x0);
   rss();
   outportb(IT1B1PB,(char)0x0);
   outportb(IT1B3PA,(char)0x0);
   outportb(WY2,(char)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\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,(char)0x99);                          /* PB out; PA,PC in */
   outportb(IT1B1PB,(char)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,(char)dattest);                     /* wpis do D32 */
               outportb(WY1,(char)(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(WE0C);                                  /* reset D22 zwora */
          n+=mar;
        outportw(WY1C,0x20);                            /* sterowanie D22  "1"  */
          n+=mar;
        outportw(WY1C,0x0);                             /* sterowanie D22  "0"  */
          n+=mar;
        p=0;
        outportw(WY1C,0xe);                             /* sterowanie D6 "1" */
          p+=mar5;
        outportw(WY1C,0x0);                             /* sterowanie D6 "0" */
          p+=mar5;
        outportw(WY1C,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 instrukcje\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[9]);        /*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,(char)0x99);
           outportb(IT1B1PB,(char)dat6);
          }
        if(a[7])
          inportb(WE3);
        else
          outportw(WY0C,0x0);           /*a[7] WRITTEN */
        if(k9!=a[9])
          {
            outportw(K0C,(a[9]?0x200:0x0));             /*a[9] SYNCSW */
            delay();
            k9=a[9];
          }
        outport(IT3B2WY,(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,(char)0x0);
                p2=1;                                   /* kontrola poprawnosci*/
                return(0);
              }
          }
        datzad++;
      }
    glob++;
  }
  printf("\nUKLADY WEJSCIA\t\t\t\t\tOK!\n");
  outportb(IT1B1PB,(char)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,(char)0x99);           /*zaprogr.B1 it1 PA in, PB out */
  outportb(IT1B1PB,(char)0x0);          /* wszystkie wyjscia B1 = 'H' oc*/
  outportw(KOC,0x0);
  printf("REJESTR STANU STEROWNIKA POLOZENIA :");
  outportb(IT1B3,(char)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(WE0C);                  /* 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,(char)0x81);               /* sterowanie D17 "1" */
        delay();
          p+=mar1;
        outportb(WY2,(char)0x0);                /* sterowanie D17 "0" */
        delay();
          p+=mar1;
        outportb(WY2,(char)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 instrukcje\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,(char)sterowanie);
     if(a[4])
       inportb(WE3);
     else
       outportw(WY0C,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')
                    {
                      pé=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*/
```

```
/*1988-11-30*/
                        /* MP1MP2.C */
      /* TESTOWANIE PRZELACZNIKOW MP1,MP2 PAKIETU MA70 */
     /* opracowal dr inz.Marian Wrzesien dnia 1988.08.17 */
# include <adr.h>
# include <wewyma70.h>
# include <fun.h>
# include <proto.h>
mp1mp2()
  {
    int i,j,k,l,m,n,a[8],b[8],mp1,mp2;
    unsigned int datwyj,dattest;
    fhold();
    outportb(IT1B1,(char)0x9b);
    outportb(IT1B3,(char)0x8b);
    outportb(IT1B3PA,(char)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<9)
              {
                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 Odczytane polozenie segmentow MP%d:",n+1);
                m=0;
                while(m<8)
                  {
                    b[m]=(datwyj & pelz1(m))>>m;
                    printf("  %3",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*/
```

```
                        /* ITP.C */
        /* TESTOWANIE-IDENTYFIKACJA TYPU PAKIETU */
        /* opracowal dr inz. Marian Wrzesien dnia 1988.07.30 */
# include <fun.h>
# include <adr.h>
# include <wewyma70.h>
# include <proto.h>
# 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(WE4)>>2)&0x1))
# include <proext.h>
itp()
  {
    int n=0,p=0,t=0,s=0,m=0,k=0,r=0,j=0;
    static int a[8]=0;
    scr_clr();
    fhoId();
    j=0;
    while(1)
    {
      if(j)
        printf("\tIDENTYFIKACJA TYPU PAKIETU MA-70\n\n");
      n=0;
      inportw(WE0C);                          /* reset D22 zwora */
      delayp(10);
      n+=mar;
      outportw(WY1C,0x20);                    /* sterowanie D22  "1"  */
      n+=mar;
      outportw(WY1C,0x0);                     /* sterowanie D22  "0"  */
      n+=mar;

      p=0;
      outportw(WY1C,0xe);                     /* sterowanie D6 "1" */
      p+=mar5;
      outportw(WY1C,0x0);                     /* sterowanie D6 "0" */
      p+=mar5;
      outportw(WY1C,0xe);
      p+=mar5;

      outportb(IT1B1,(char)0x99);       /* Ctrl B1 it1 Pa,Pb,Pc na in */
      outportb(IT1B3,(char)0x8b);       /* Ctrl B3 it1 it1 nie blokuje */
      delay();
      outportb(IT1B3PA,(char)0x0);      /* B3 it1 Pa */

      t=0;
      outportb(WY2,(char)0x81);                 /*sterowanie D17  '1' */
      delay();
      t+=mar1;
      outportb(WY2,(char)0x0);
      delay();
      t+=mar1;
      outportb(WY2,(char)0x81);
      delay();
      t+=mar1;
      inportb(WE3);                     /*ustawienie D22 WRITTEN */

      s=inportb(IT1B1PC);               /*odczyt stanu przelacznika MP2 */

      k=inportb(WE5);                   /*odczyt stanu przelacznika MP1 */
      if(j)
        {
          if(n==1)
            printf("1. Zwora Z1 zwarta\n");
          else
            printf("1. Zwora Z1 rozwarta\n");
          if(p==2)
            printf("2. Obwod scalony D6 typu 74174 umieszczony w pakiecie\n");
          else
            printf("2. Brak obwodu scalonego D6 typu 74174 w pakiecie\n");
          if(t==2)
            printf("3. Obwod scalony D5 typu 74125 umieszczony w pakiecie\n");
          else
            printf("3. Brak obwodu scalonego D5 typu 74125 w pakiecie\n");
          printf("\tNr.segmentu przelacznika MP1,MP2: 1  2  3  4  5  6  7  8\n");
          printf("4. Stan przelacznika MP2 (0 - zwarcie): ");
          m=0;
          while(m<8)
            {
              a[m]=(s&pelz1(m))>>m;
              printf("  %d",a[m]);
              m++;
            }

          printf("\n5. Stan przelacznika MP1 (poziom  TTL): ");
          m=0;
          while(m<8)
            {
               a[m]=(k&pelz1(m))>>m;
               printf("  %d",a[m]);
               m++;
            }
          r=9&-18*k;
          if((n==1&&p==2&&t==2&&s!=0xff)&&(k==0x5||k==0x2))
            {
              printf("\n\n\n\t\tPrzy zwartej zworze Z8\n");
              printf("Pakiet przeznaczony dla osi FI i typu robota: IRp-%d\n",r);
            }
          else
          if((n==3&&p==2&&t!=2&&s==0xff)&&(k==5||k==2))
            {
              printf("\n\n\n\t\tPrzy rozwartej zworze Z8\n");
              printf("Pakiet przeznaczony dla osi roznej od FI i typu robota: IRp-%d\n",r);
```

```
            }
          else
            printf("\n\n\tTyp pakietu: NIEOKRESLONY\n");
          printf("\nESC\t przerywa testowanie\n\t\t\t\t\t\t\t\t\t * CR *\n");
          if(getch()==0x1b)
             return(0);
          scr_clr();
        }
      j=1;
    }
}
                                    /*koniec*/
```

```
/*                      opracował:  Zbigrew Stanczak                    */

#include <ascii.h>
#include <proext.h>
#include <proto.h>

int menu2ma7()
{
/* ************************* menu ********************************************** */
static char * naglowek[2] ={
        "* menu 2 *",
"                       PROGRAMY URUCHOMIENIOWE MA70",
};
static char * text[]={
        "strojenie przetwornika c/a",
        "uruchamianie dekodera adresow",
        "strojenie przerwornika c/a testera",
        "poprzednie menu",
};
char * podpis = "                                                  * spacja * del * cr *";
int line_n = 4;
static int line_1[4]={26,29,34,15};
/* ************************* menu ********************************************** */

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

```
                        /* URDAC.C */
        /* STROJENIE PRZETWORNIKA CYFROWO-ANALOGOWEGO */
        /* Opracowal dr inz. Marian Wrzesien dnia 10.09.1999 */
# include <wewyma70.h>
# include <adr.h>
# include <fun.h>
# include <proto.h>
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 .0. 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,(char)0x0);
          outportb(WY4,(char)0x0);
          outportb(IT3B5PA,(char)0x0);
          outportb(IT3B5PC,(char)0x30);
          printf("Napiecie wyjsciowe=-10V\n");
          cr();
          outportb(WY4,(char)0x8);
          outportb(IT3B5PC,(char)0x38);
          printf("Napiecie wyjsciowe=0V\n");
          cr();
          outportb(WY3,(char)0xff);
          outportb(WY4,(char)0xf);
          outportb(IT3B5PA,(char)0xff);
          outportb(IT3B5PC,(char)0x3f);
          printf("Napiecie wyjsciowe=+10V\n\n");
          printf("# p # przerywa strojenie\n\t\t\t\t\t\t\t\t\t\t# CR #");
          if(getch()=='p')
            return(0);
        }
   }
                                /*koniec*/
```

```
                                /*UDMA70.C */
                    /* URUCHAMIANIE DEKODERA MA70 */
          /*Opracował dr inż. Marian Wrzesień dnia 1999-03-24 */

# include <fun.h>
# include <proto.h>
# define IT3B3WY 0xfdd8
udma70()
  {
     int dat1=0x0,cc=0,i=1;
     int adr1,adr2;
     while(((cc=scr_pool())!=0x1b) &&( dat1<=0xf))
       {
         if(cc | i)
           {
            scr_clr();
            printf("Obserwuj sygnały WY 07 i WY 05 przy:\n");
            printf("adr1=0x%x,\t\tadr2=0x%x\n",(int)(adr1=4*dat1),(int)(adr2=4*dat1+2));
            dat1++;
            i=0;
           }
            outportb(IT3B3WY,(char)dat1);
            adr1=4*dat1;
            adr2=adr1+2;
            outportw(adr1,0xff);        /*WY 07 */
            outportw(adr2,0xff);        /*WY 05 */
            inportw(adr1);
            inportw(adr2);

       }
     cr();
   }
```

```
                        /* URDACIT3.C */
        /* STROJENIE PRZETWORNIKA CYFROWO-ANALOGOWEGO */
        /* Opracowal dr inz. Marian Wrzesien dnia 10.08.1988 */
# include <wewyma70.h>
# include <adr.h>
# include <fun.h>
# include <proto.h>
urdacit3()
 {
  int k=0;
  fhold();
   while(k<2)
    {
     while(1)
        {
          scr_clr();
          printf("STROJENIE OBWODOW PRZETWORNIKA CYFROWO-ANALOGOWEGO PAKIETU IT3\n\n");
          printf("Dolacz multimetr MUC2000 do punktu WY pakietu IT3.\n");
          printf("Masa multimetru dolaczona do .O. analogowego (obok WY)!!!\n");
          printf("Reguluj napiecie 0V (zakres pomiarowy=200m) potencjometrem P1 (1k).\n");
          printf("Reguluj napiecie   10V (zakres pomiarowy=20V) potencjometrem P2 (10 k).\n\n");
          outportb(WY3,(char)0x0);
          outportb(WY4,(char)0x0);
          outportb(IT3B5PA,(char)0x0);
          outportb(IT3B5PC,(char)0x30);
          if(!k)
            printf("Napiecie wyjsciowe=-10V\n");
          else
            printf("Doprowadz do minimum Uwy A4 potencjometrem P5 (100k)\n");
          cr();
          outportb(WY4,(char)0x8);
          outportb(IT3B5PC,(char)0x38);
          if(!k)
             printf("Napiecie wyjsciowe=0v\n");
          else
             printf("Doprowadz do zera Uwy A4 potencjometrem P5 (100k)\n");
          cr();
          outportb(WY3,(char)0xff);
          outportb(WY4,(char)0xf);
          outportb(IT3B5PA,(char)0xff);
          outportb(IT3B5PC,(char)0x3f);
          if(!k)
            printf("Napiecie wyjsciowe=+10V\n\n");
          else
            printf("Doprowadz do minimum Uwy A4 potencjometrem P5 (100k)\n");
          if(!k)
            printf("\n* p * przerywa strojenie DAC IT3\n\t\t\t\t\t\t\t\t\t* CR *");
          else
            printf("\n* p * przerywa strojenie komparatora DAC-IT3\n\t\t\t\t\t\t\t\t\t* CR *");
          if(getch()=='p')
            break;
        }
      k++;
     }
   scr_clr();
   printf("STROJENIE OBWODOW PRZETWORNIKA CYFROWO-ANALOGOWEGO PAKIETU IT3\n");
   printf("\n\"staw napiecie odniesienia komparatora +50mV potencjometrem P4");
   cr();
   printf("\nKoniec strojenia obwodu DAC na pakiecie IT3");

  cr();
  return(0);
}                               /*koniec*/
```

/* ZERO DAC. */

```
main()
   {
     outportb(0xfdce,0x82);
     outportb(0xfdd8,0x10);
printf("Wcisnij reset MA-70");
     getch();
     while(1)
         {
           outportb(0xf003,0x0);
           outportb(0xf004,0x0);
           outportb(0xfdc8,0x0);
           outportb(0xfdcc,0x30);
           printf("Napiecie wyjsciowe=-10V\n");
         getch();
           outportb(0xf004,0x8);
           outportb(0xfdcc,0x38);
           printf("Napiecie wyjsciowe=0v\n");
         getch();
           outportb(0xf003,0xff);
           outportb(0xf004,0xf);
           outportb(0xfdc8,0xff);
           outportb(0xfdcc,0x3f);
           printf("Napiecie wyjsciowe=+10V\n");
         getch();
         }
   }
```

PAKIET PROGRAMOW

DO TESTOWANIA PANELI PROGRAMOWANIA

```
#include (ascii.h)
#include (proext.h)
#include (proto.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 = "                                                    * spacja * 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) );
}
```

```
#define PPIO 0xf000
#include <proto.h>
#include <ascii.h>
#include <tstglb.h>

main()
{
int k;
outportb(PPIO+0x7c,(char)0);/*zgaszenie wyswietlaczy */
while((k = menuprog()) ){
    switch(k){
        case 1:mem_line=0;
               p1();
               mem_line=0;
               break;
        case 2:mem_line=0;
               p2();
               mem_line=1;
               break;
        case 3:mem_line=0;
               p3();
               mem_line=2;
               break;
        case 4:
{
#asm
int 10
#endasm
}
                break;
    }
}
}
```

```
/*                      opracowal:  Zbigniew Stanczak                      */

#include <ascii.h>
#include <proext.h>
#include <proto.h>

int menup1()
{
/* ************************** menu ********************************************* */
static char * naglowek[2] ={
        "* menu 2 *",
"                       TEST PAKIETU P1",
};
static char * text[]={
        "test calkowity",
        "pamieci EPROM",
        "pamieci RAM",
        "interfejs szeregowy (V24)",
        "blok sterowania wyswietlaczy LED",
        "blok sterowania przyciskow",
        "blok sterowania joystick'a",
        "poprzednie menu",
};
char * podpis = "                                                    * spacja * del * cr *";
int line_n = 8;
static int line_l[8]={14,13,11,25,32,26,26,15};
/* ************************** menu ********************************************* */

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

```
#include <ascii.h>
#include <proext.h>
#include <proto.h>

static void autotest();
p1(){
int k;
while(k = menup1()){
    switch(k){
        case 1:autotest();
               break;
        case 2:eprom();
               break;
        case 3:ram();
               break;
        case 4:piso();
               break;
        case 5:led();
               break;
        case 6:przyc();
               break;
        case 7:joi();
               break;
        case 8:return 0;
               break;
        default:return -1;
   }
 }
}


static void autotest(){
  eprom();
    if(ram() == 0)
       if(piso() == 0)
          if(led() == 0)
            if(przyc() == 0)
               if(joi() == 0){
                  scr_clr();
                  printf("test calkowity plytki P1\t\t\t\t\t\tOK!");
                  cr();
                  return;
               }
printf("test calkowity plytki P1");
blad();
cr();
return ;
}
```

111

```
                        /*TESTOWANIE PAMIECI EPROM  */
# include <proto.h>
# define prominh 0x20
# define hold 0x10
# include <adr.h>
# include <fun.h>
eprom()
  {
    int dana=0,i=0;
    unsigned long adr,adrw,suma,suma1=0l;
 phold();
 outportb(IT3B3HY,(char)(prominh|hold));
 scr_clr();
 printf("PAMIEC EPROM PLYTKI P1");
 suma=0x0l;
 adr=0x800001;
 for(i=0;i<3;++i){
    adrw = 0x1fffl+adr;
    while((unsigned long)adr<=(unsigned long)adrw){
           dana=0xff & areadb(adr);
           suma1 +=dana;
           adr++;
    }
    suma +=suma1;
    printf("\nSuma zawartosci komorek pamieci EPROM (U11%d) wynosi:",2+i);
    printf("\t\t%lx",suma1);
    suma1 = 0l;
 }
 printf("\nCalkowita suma zawartosci komorek pamieci EPROM wynosi:");
 printf("\t\t%lx",suma);
 cr();
 return(0);
}
```

```
            /*TESTOWANIE PAMIECI RAM  */
# define ADRAMBEG (unsigned long)0x86000
# define ADRAMEND (unsigned long)0x867ff
# include <ascii.h>
# include <proto.h>
ram()
 {
  int datwe,datwy,i,bl=0;
  unsigned long adr;
  phold();
  scr_clr();
  printf("PAMIECI RAM PANELU PROGRAMOWANIA");
  i=0;
  while(i<4)
   {
     switch(i)
      {
           case 0: datwe=0x0;
printf("\nzapis/odczyt danej 0 hex\n");
                   break;
           case 1: datwe=0xff;
printf("zapis/odczyt danej ff hex\n");
                   break;
           case 2: datwe=0xaa;
printf("zapis/odczyt danej aa hex\n");
                   break;
           case 3: datwe=0x55;
printf("zapis/odczyt danej 55 hex\n");
                   break;
      }
     adr=ADRAMBEG;
     while(adr<=ADRAMEND)
      {
        mwriteb(adr,(char)datwe);
        if((datwy=(mreadb(adr)&0xff))!=datwe)
          {
            bl=1;
            printf("\nBlad zapisu/odczytu przy adresie=0x%x.\n",adr);
            printf("Wartosc wpisywana=0x%x\n",datwe);
            printf("Wartosc odczytana=0x%x\n",datwy);
            blad();
            if(cr()==esk)break;
          }
        adr++;
      }
     i++;
   }
printf("\npamiec ram");
if(bl==0){
 printf("\t\t\t\t\t\t\t\t\tOK!");
 cr();
 return(0);
}
blad();
cr();
return -1;
}
```

```
# include <asci.h>
# include <adr.h>
# include <proto.h>
# define init 0x0
# define notinit 0x20
# define PPIO 0xf000
piso()
 {
  unsigned int k=0,j=0,bl[2];
  unsigned char i=0;
  bl[0]=bl[1]=0;

  phold();
  scr_clr();
  printf("INTERFEJS SZEREGOWY PANELU PROGRAMOWANIA");

k=inportb(IT3A&D1);
k=inportb(PPIO!0x8a);

  outportb(IT3A&C1,(char)0xce);
  outportb(IT3A&C1,(char)0x27);    /* 8251 */

  outportb(PPIO!0x8f,(char)0x36);  /* 8253 */
  outportb(PPIO!0x8f,(char)0x76);
  outportb(PPIO!0x8f,(char)0xb6);
  outportb(PPIO!0x8c,(char)0x19);
  outportb(PPIO!0x8c,(char)0x00);

  outportb(PPIO!0x8b,(char)0xce);  /* 8251 */
  outportb(PPIO!0x8b,(char)0x27);

k=inportb(IT3A6D1);
k=inportb(PPIO!0x8a);
printf("\nzapis szeregowy z panelu do testera");
for(i=0;i<0xff;i++){
   j=0;
   while(!(inportb(PPIO!0x8b)&0x1))if(!(++j)){
                                       bl[0]=1;
                                       printf("\nbrak sygnalu TxRDY");
                                       cr();
                                       break;
                                    }
   if(bl[0]==1)break;
   outportb(PPIO!0x8a,i);
   j=0;
   while(!(inportb(IT3A&C1)&0x2))if(!(++j)){
                                       bl[0]=1;
                                       printf("\ntester nie odbiera informacji szeregowej");
                                       cr();
                                       break;
                                    }
   if(bl[0]==1)break;
   k= inportb(IT3A&D1);
   if((unsigned char)k != i){
        bl[0]=1;
        printf("\ndana zapisana rownolegle do USART'u panelu %x",(int)i);
        printf("\ndana odebrana szeregowo od USART'u panelu %x",k);
        blad();
        if(cr()==esk)break;
   }
 }
if(bl[0]==0)printf("\t\t\t\t\t OK!");
printf("\nodbior szeregowy z testera do panelu ");
for(i=0;i<0xff;i++){
j=0;
   while(!(inportb(IT3A&C1)&0x1))delay();if(!(++j)){
                                       bl[1]=1;

                                       printf("\ntester nie moze wyslac informacji szeregowej");
                                       cr();
                                       break;
                                    }
  if(bl[1]==1)break;

  outportb(IT3A&D1,i);
  j=0;
  while(!(inportb(PPIO!0x8b)&0x2))if(!(++j)){
                                       bl[1]=1;
                                       printf("\nbrak sygnalu RxRDY");
                                       cr();
                                       break;
                                    }
  if(bl[1]==1)break;
  k= inportb(PPIO!0x8a);
  if((unsigned char)k != i){
       bl[1]=1;
       printf("\ndana przesłana szeregowo do USART'u panelu %x",(int)i);
       printf("\ndana odczytana rownolegle od USART'u panelu %x",k);
       blad();
       if(cr()==esk)break;
  }
 }
if(bl[1]==0)printf("\t\t\t\t\t OK!");
if((bl[0]|bl[1])==0){
   printf("\ninterfejs szeregowy\t\t\t\t\t\t\t\tOK!");
   cr();
   return 0;
}
 printf("\ninterfejs szeregowy");
 blad();
 cr();
 return -1;
}
```

```
                        /* hold.c */
            /*WPROWADZANIE W STAN HOLD */
        /* opracował dr inż. Marian Wrzesień dnia 1988.08.22 */

# include <adr.h>
# include <ascii.h>
# include <proto.h>
# define init 0x0
# define notinit 0x20
# define prominh 0x20
# define hold 0x10
int phold()
 {
    int i=0,k;

while(1){
        ++i;
                outportb(IT3B5,(char)0x82);
        outportb(IT3B5PC,(char)0x78);
        outportb(IT3B3WY,(char)(0x80|hold)); /* reset it3B3A6  (785) */
delayp(100);
        outportb(IT3B3WY,(char)(hold));
        outportb(IT1D6WY,(char)(init));
delayp(1000);
        outportb(IT1D6WY,(char)(notinit));
        delayp(100);
        if(!(inportb(IT3B5PB)&0x2))return 0;
        if(i>=10)
         {
          scr_clr();
          stout(bel);
          printf("\nSygnal INIT/ nie wprowadza procesora  w stan HOLD\n");
          cr();
          return -1;
         }

}
}
```

```
# include <adr.h>
#:nclude <prate.h>
# define PPIO 0xf000
# include <asc:i.h>
led(){
   int i, j;
   phold();
   scr_clr();
 printf("TEST WYSWIETLACZY LED\n");
/*************************************************************** */
 outportb(PPIO!0x83,(char)0x8a);/* sterowanie U121 */
 outportb(PPIO!0x87,(char)0x92);/* ster U122       */
/*************************************************************** */
for(i=0;i<2;i++){
printf("Wszystkie LED'y swieca");
stout(bel);
 outportb(PPIO!0x84,(char)0x00);
 outportb(PPIO!0x80,(char)0x00);
 cr();
scr_ldel();
printf("Wszystkie LED'y zgaszone  ");
 outportb(PPIO!0x84,(char)0xff);
 outportb(PPIO!0x80,(char)0xff);
 cr();
 scr_ldel();
}
/*************************************************************** */
       return(0);
}
```

```
# include <dos.h>
# define PPIO (int)0xf000
# include <ascii.h>
extern echo_on_off;
static char * klawisz[7][8]={
      {"  ","A2","A3","B1","B2","B3","E2","E3"},
      {"E1","F1","F2","C4","  ","  ","  ","  "},
      {"C4","C5","E4","E5","  ","  ","  ","  "},
      {"E8","A7","A8","A9","B7","B8","B9","C7"},
      {"C8","C9","E7","E9","E10","E11","  ","  "},
      {"B10","B11","A10","A4","A5","A6","B6","  "},
      {"A11","C10","C11","  ","  ","  ","  ","  "}
};
przyc(){
   int k=0,j=0;
   char i=0;
   echo_on_off = 0;
   phold();
   scr_clr();
 printf("TEST PRZYCISKOW \n");
prog8255();
printf("\nWcisnij dowolny przycisk panelu \n i sprawdz zgodnosc wyswietlonej");
printf(" wartosci \nz wartoscia podana na rysunku widoku elementow plytki P3\n\n");
while(!scr_pool()){
  outportb(PPIO!0x94,(char)0xff);

 for(i=1;i<=7;i++){
     outportb(PPIO!0x86,(char)i);/* nr. grupy */
     k=~(inportb(PPIO!0x85)!0xff00);
prog8255();
delay();
     if(k){
       stout(bel);
       scr_ldel();
       if(k==-1){printf("\t\t\t\t(reset)");prog8255();break;}
       for(j=0;j<8;j++)if((k)>>j)%1)break;
       printf("\t\t\t\t%s",klawisz[i-1][j]);
       outportb(PPIO!0x94,(char)(~4));
       delayp(1000);
    }
  }
}
return(0);
}
prog8255(){
/************************************************************************ */
 outportb(PPIO!0x83,(char)0x8a);/* sterowanie U121 */
 outportb(PPIO!0x87,(char)0x82);/* ster U122       */
 outportb(PPIO!0x94,(char)0xff);
 outportb(PPIO!0x80,(char)0xff);
/************************************************************************ */
}
```

```
# include <adr.h>
# include <fun.h>
# include <wewyma70.h>
# include <proto.h>
# define PPIO 0xf000
joi(){
   unsigned jj,j,i,k=0,kk,bl=0;
   unsigned int dana;
   phold();
   scr_clr();
 printf("TEST JOYSTICK'a");
outportb(PPIO|0x7c,(char)0);
/************************************************************************ */
 outportb(PPIO|0x83,(char)0x8a);
/************************************************************************ */
for(i=0x800;i<=0xfff;i+=0x7ff){
dana = i;
jj=0;
 for(j=1;!(j&8);j<<=1){
   outportb(IT3B5PA,(char)dana);                /* sterowanie AD na pakiecie it3 */
   outportb(IT3B5PC,(char)((dana>>8)|0x70));    /* j.w. pozostaja sygnaly Xack req i Waitreq*/
   delayp(0xfff);
   outportb(PPIO|0x82,(char)(0x4|jj));
   outportb(PPIO|0x82,(char)(0xc|jj));
   outportb(PPIO|0x82,(char)(0x4|jj));
   delayp(100);
   kk=inportb(PPIO|0x81);
   if(k==0){
     if( kk > (unsigned)10) bl|=j;
   }
   else{
        if(jj==0)if(kk < (unsigned)200) bl|=j;
        if(jj==1)if((kk < 150)||(kk > 180))bl|=j;
        if(jj==2)if((kk < 190)||(kk > 200))bl|=j;
   }
  ++jj;
 }
 k++;
}
if(bl==0){
  printf("\njoystick\t\t\t\t\t\t\t\tOK!");
  cr();
  return 0;
}

printf("\nuszkodzony zespol joystick'a ");
if(bl&1)printf("JSTC1 ");
if(bl&2)printf("JSTC2 ");
if(bl&4)printf("JSTC3 ");
printf("U121(PB)");
blad();
return -1;
/************************************************************************ */
}
```

```
/*                      opracowal:  Zbigniew Stanczak                      */

#include <ascii.h>
#include <proext.h>
#include <proto.h>
int menup2()
{
/* ************************ menu ******************************************** */
static char * naglowek[2] ={
        "* menu 2 *",
"                       TEST PAKIETU P2",
};
static char * text[]={
        "test calkowity",
        "blok pamieci i sterowania wyswietlaniem",
        "sygnal BUDZIK/",
        "poprzednie menu",
};
char * podpis = "                                                  * spacja * del * cr *";
int line_n = 4;
static int line_l[4]={14,39,14,15};
/* ************************** menu ******************************************** */

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

```
#include <ascii.h>
#include <proext.h>
#include <proto.h>

static void autotest();

p2()
{
int k;
while((k = menup2()) ){
    switch(k){
        case 1:autotest();
               break;
        case 2:wys();
               break;
        case 3:budz();
               break;
        case 4:return 0;
               break;
        default:return -1;
   }
 }
}


static void autotest(){
                  wys();
                  budz();
cr();
return ;
}
```

```
# include <adr.h>
# include <fun.h>
# include <wewyma70.h>
# include <proto.h>
# define PPIO 0xf000
wys(){
unsigned j;
   phold();
   scr_clr();
printf("TEST BLOKU STERUJACEGO I PAMIECI WYSWIETLACZY\n");
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
outportb(PPIO!0x7c,(char)0);
printf("wszystkie wyswietlacze zgaszone");
printf("\n(zakaz wyswietlania)");
cr();
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
outportb(PPIO!0x7c,(char)0);
for(j=0;j(1344;j++)outportb(PPIO!0x7d,(char)0x00);
scr_clr();
printf("\nwszystkie wyswietlacze zapalone");
printf("\n(zezwolenie wyswietlania)");
printf("\nram wyswietlaczy wypelniony zerami");
outportb(PPIO!0x7e,(char)0xff);
cr();
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
outportb(PPIO!0x7c,(char)0);
for(j=0;j(1344;j++)outportb(PPIO!0x7d,(char)0xff);
scr_clr();
printf("\nwszystkie wyswietlacze zgaszone");
printf("\n(zezwolenie wyswietlania)");
printf("\nram wyswietlaczy wypelniony jedynkami");
outportb(PPIO!0x7e,(char)0xff);
cr();
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
outportb(PPIO!0x7c,(char)0);
for(j=0;j(1344;j++)outportb(PPIO!0x7d,(char)0x00);
scr_clr();
printf("\nwszystkie wyswietlacze zapalone");
printf("\n(zezwolenie wyswietlania)");
printf("\nram wyswietlaczy wypelniony zerami");
outportb(PPIO!0x7e,(char)0xff);
cr();
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
scr_clr();
printf("wszystkie wyswietlacze zgaszone");
printf("\n(zakaz wyswietlania)");
printf("\nram wyswietlaczy wypelniony zerami");
outportb(PPIO!0x7c,(char)0);
cr();
/*################################################################## */
return 0;
}
```

```
# include <adr.h>
# include <proto.h>
# include <proext.h>
# define PPIO 0xf000
# include <ascii.h>
budz(){
   int i=0,j;
   char tim=0;
   phold();
   scr_clr();
 printF("TEST SYGNALU BUDZIK/\n");
/************************************************************** */
 outportb(PPIO|0x83,(char)0x8a);/* sterowanie U121 */
 outportb(PPIO|0x87,(char)0x92);/* ster U122       */
/************************************************************** */
outportb(PPIO|0x94,(char)0x00);
outportb(PPIO|0x80,(char)0x00);
printf("\nPoprawny sygnal BUDZIK/ - LED'y zgaszone");
outportb(PPIO|0xfe,(char)0);
 cr();
/************************************************************** */
/************************************************************** */
 outportb(PPIO|0x83,(char)0x8a);/* sterowanie U121 */
 outportb(PPIO|0x87,(char)0x92);/* ster U122       */
/************************************************************** */
outportb(PPIO|0x94,(char)0x00);
outportb(PPIO|0x20,(char)0x00);
scr_clr();
printf("Podtrzymywanie sygnalu BUDZIK/");
printf("\nPodaj czas odstepu miedzy sygnalami BUDZIK/ \n\t\t[x100ms]= ");
scanf("%d",&j);
if(j==0)j=4;
printf("\n\nLED'y zapalone - prawidlowy czas powtarzania sygnalu BUDZIK/");
printf("\nLED'y zgaszone - za  dlugi  czas powtarzania sygnalu BUDZIK/");
stout(bel);
printf("\n\t\t\t\t\t\t\t\t* CR *");
outportb(PPIC|0xfe,(char)0);
while(++tim){delayp(j);outportb(PPIO|0xff,(char)0);}
while(!scr_pool())outportb(PPIO|0xff,(char)0);
return(0);
}
```

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

#include <proto.h>
#include <ascii.h>
# 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 kolumny */
               0x00de,                                                  /* 3 III kolumny */
               0x00be,                                                  /* 4 IV kolumny */
               0x007e,                                                  /* 5 V kolumny */

               0x00fd,                                                  /* 6 I wiersze */
               0x00fb,                                                  /* 7 II wiersze */
               0x00f7,                                                  /* 8 III wiersze */
               0x00ef,                                                  /* 9 IV wiersze */
               0x00df,                                                  /* 10 V wiersze */
               0x00bf,                                                  /* 11 VI wiersze */
               0x007f,                                                  /* 12 VII wiersze */
               0x0000,                                                  /* 13 wszystkie swieca */
            };
         static char *koma[]=
            {
               "       TESTOWANIE STATYCZNE",
               "       TESTOWANIE IMPULSOWE",
               "       TESTOWANIE DYNAMICZNE",
               "Sprawdzic swiecenie",
               " kolumn wyswietlaczy",
               " wierszy wyswietlaczy",
               " po sprawdzeniu ",
               " ",
               "TESTOWANIE PLYTKI P3 PANELU PROGRAMOWANIA",
               "ocena wynikow testu na podstawie obserwacji wyswietlaczy plytki p3",
               "start...wszystkie diody wyswietlaczy swieca",
               "koniec testu plytki p3 panelu programowania",
            };
         int st0,st1,st2,st3,st4,n,N,x;                                 /* stale pomocnicze */
         scr_clr();
printf("Na czas testu plytki P3 polacz zlacze Z302 kablem KZ-302 z testerem!");
         cr();
         scr_clr();
          ster_it2(0,0);                                                /* zgaszenie diod wyswietlaczy
          druk(9);
          druk(9);
          druk(10);
          ster_it2(0,13);
             delayp(3000);
          cr();
          st4=-1;
          N=3;
          while(st4++<2)                                                /* potrojna petla 1:STAT.,IMP.,
             {
                st0=st2=st3=0;
                st1=5;
                scr_clr();
                druk(st4);                                              /* koma[st4]: test.STAT. lub  I
                ster_it2(0,0);
                cr();
                   if(st4<2)
                      {
                         while(st3++<2)                                 /* podwojna petla 1:test. kolu
                            {
                               while(st0++<st1)                         /* petla wybierania data[st0] =
                                  {
                                     printf("\t\t%s",koma[3]);
                                     printf(" %d",(st0-st2));
                                     printf("%s\n\t",koma[3+st3]);
                                        {
                                           for(n=-N;n<N;n++)            /* delayp trwania sterowania i
                                              {
                                                 ster_it2(0,0);         /* nic nie swieci */
                                                    delayp(100);
                                                 ster_it2(0,st0);       /* swieci kolumna lub wiersz =
                                                    delayp(100);
                                              }
                                        }
                                     cr();
                                  }
                               printf("\n");
                               st1=12;                                  /* rozszerzenie zakresu data[st
                               st2=5;                                   /* ustawienie nr liczby porzad
                               st0-=1;                                  /* poprawka na odliczanie data
                            }
                         N=100;
                      }
                   else
                         for(n=0;n<150;n++)                             /* liczba obiegow wtest. DYN. =
                            {
                               st0=st3=0;
                               st1=12;
                               while(st3++<2)                           /* podwojna petla dla wierszy i
                                  {
```

```
                        while(st0++<st1)                    /* petla wyboru adresu komu[] */
                           {
                             ster_it2(0,st0);               /* swieci kolejna kolumna i na
                                delayp(1000-6*n);
                           }
                        st0-=1;                             /* poprawka na odliczanie data
                      }
                  }
               ster_it2(0,0);                               /* wygaszenie diod wyswietlacz
            }
         druk(11);                                          /* koniec testu */
         cr();
         return(0);
      }
                              /* KONIEC */
```

FUNKCJE

BIBLIOTECZNE

```
#include "ascii.h"
extern int buf_btmo;
void btmo()
{

/*stout(0x2a);*/
 if(buf_btmo == 0){
                buf_btmo = 0xff;
                printf("\n\tBrak sytgnalu XACK w trakcie wykonywania programu testujacego");
                printf("\n\tKoniec testowania");
                eoci();
 }
    buf_btmo = 0xfe;
/*     stout(0x21);*/
    eoci();
}
/*************************************************************************/
```

```
cmpbit(x,y)  /* x!=y zwraca nr pierwszeych najstarszeych nierownych bitow */
             /* x==y zwraca -1 */
unsigned int x,y;
 {
   int p;
     if(x!=y)
     {
       p=16;
       while(p>=0)
        {
          if((x>>p) != (y>>p))
             return(p);
          p--;
        }
      }
    else
       return(-1);
 }
```

```
unsigned long htol(b)
unsigned char  *b;
{
unsigned char *p,i;
unsigned long k;
unsigned long sum;
sum = i = 0;
k = 1;
p = b;
while(*p) != 0xd)p++;
while((--p) >= b){
                switch(*p)
                      {
                      case 'A':
                      case 'a': i = 10;
                                break;
                      case 'B':
                      case 'b': i = 11;
                                break;
                      case 'C':
                      case 'c': i = 12;
                                break;
                      case 'D':
                      case 'd': i = 13;
                                break;
                      case 'E':
                      case 'e': i = 14;
                                break;
                      case 'F':
                      case 'f': i = 15;
                                break;
                      default: if((*p < 0x30)||(*p > 0x39)) return 01;
                               i = *p - 0x30;
                               break;
                      }
              sum = sum + (unsigned long)(i * k);
              k *= 16;
               }
return sum;
}
```

```
itoa(n,s)
char s[];
int n;
{
int i=0;
int sgn;
if( (sgn=n)<0 ) n = -n;
do      {
        s[i++]=n%10+'0';
        } while((n/=10)>0);
if( sgn<0 )  s[i++]='-';
s[i] = 0;
reverse(s);
}
```

```
itoha(n,s)
char s[];
unsigned int n;
{
int i=0;
int zn;
do      {
        zn = n%16;
        s[i++] = (zn<10)? (zn+'0') : ('A'+(zn-10));
        }  while((n/=16)>0);
s[i] = 0;
reverse(s);
}
```

```
#include "ascii.h"
extern int buf_mc42;
void mc()
{
stout(0x21);
stout(0x21);
outportw(0xfbf6,0xffff);          /* zerowanie pakietu mc42 */
printf("\n\t\t\tPrzeciazenie pakietu MC42 !\n");
buf_mc42 = 0x1;
     eoci();
}
/*************************************************************************/
```

```
unsigned long abstolptr();
/*-------------------------------------------------------------------------*/
char mreadb(adr)
unsigned long adr;
{
        if(adr > 0xffff)adr = abstolptr(adr);
        return peekb(adr);
}
```

```
unsigned long abstolptr();
/*-------------------------------------------------------------------------*/
int mreadw(adr)
unsigned long adr;
{
                        if(adr > 0xffff)adr = abstolptr(adr);
                                        return peekw(adr);

}
```

```
unsigned long abstolptr();
/*-------------------------------------------------------------------*/
void mwriteb(adr,data)
unsigned long adr;
unsigned char data;
{
                if(adr >0xffff)adr  = abstolptr(adr);
                pokeb(adr,data);
}
```

```
unsigned long abstolptr();
/*-------------------------------------------------------------------------*/
void mwritew(adr,data)
unsigned long adr;
unsigned int data;
{
        if(adr >0xffff)adr = abstolptr(adr);
        pokew(adr,data);
}
```

```
large =1
include !macros.h
;               parametry to ptr na funkcje btmo
;               oraz ptr na funkcje obslugujaca przerw innt2
procdef adr_prz,<<bt,ptr>,<inn,ptr>>
pushf
cli
PUSH AX
PUSH ES
MOV AX,0H
MOV ES,AX
MOV AX,CS
ldptr dx,bt,es
ldptr bx,inn,es
;MOV DX,OFFSET BTMO
;MOV BX,OFFSET INNT2
                MOV ES:[84H],DL
                MOV ES:[85H],DH
                MOV ES:[86H],AL
                MOV ES:[87H],AH
                MOV ES:[0BCH],BL
                MOV ES:[0BDH],BH
                MOV ES:[0BEH],AL
                MOV ES:[0BFH],AH
POP ES
POP AX
popf
pret
pend adr_prz
finish
end
```

```
large=1
include lmacros.h
;    zeruje flage przerwan
procdef cli
cli
pret
pend cli
```

ZBIOR HEADER

*.h

/*1988-12-07*/ MW32.H

/* UKŁAD.C */

/*TESTOWANIE UKŁADU KONTROLI ZASILAN PAKIETU MW-32 */
/*opracował dr inż Marian Wrzesień dnia 1988-12-07 */

```
# define adrusti        0x4040
# define adrczytr       0x4040
# define adrzeri        0x4242
# define adrczytdl      0x4242
# define adrzerp        0x4444
# define adrczytdh      0x4444
# define adrzer1        0x4646
# define adrczyts       0x4646
# define adrwpisl       0x4848
# define adrczyta       0x4848
# define adrbudzik      0x4a4a
# define adrczytz       0x4a4a
# define adrczytmb      0xb0b0
# define adrpiszmb      0xb0b0
# define adrczytsb      0x4f4f
# define adrpiszsb      0x4f4f
# define adralmagczyt1  0xb3b3
# define adralmagpisz1  0xb3b3
# define adralmagczyt2  0x4c4c
# define adralmagpisz2  0x4c4c
# define adrczytpam1    0x5fffL
# define adrczytpam2    0xaffffL
# define adrpiszpam1    0x5fffL
# define adrpiszpam2    0xaffffL
# define MASKA0         0x1
# define MASKA1         0x2
# define MASKA2         0x4
# define MASKA3         0x8
# define MASKA4         0x10
# define MASKA5         0x20
# define MASKA6         0x40
# define MASKA7         0x80
# define MASKA8         0X100
# define MASKA9         0x200
# define MASKA10        0x400
# define MASKA11        0x800
# define MASKA12        0x1000
# define MASKA13        0x2000
# define MASKA14        0x4000
# define MASKA15        0x8000
# define datsumzasob    0x2202
# define datsumal       0x2707
# define dattemp        0x100
# define datdym         0x1
# define datwent        0x400
# define datzasob       0x2000
# define datzasob1      0x200
# define datzasob2      0x2
# define datotw         0x800
# define datwylal       0x1000
# define datmpro        0x1f
# define datnotmpro     0x1f
# define MC42           0xfbf6
# define IT1B1          0xfede
# define IT1B1PA        0xfed8
# define IT1B1PC        0xfedc
# define IT1B1WY        0xfed0
# define IT2B3CC4       0xfbc8
# define IT1B3          0xfece
# define IT1B3PC        0xfecc
# define stycznik       (wyob & MASKA10)
# define zaklpam        (wyob & MASKA15)
# define al_dzw         (wyob & MASKA0)
# define al_sw          (wyob & MASKA9)
# define zer1           (outportb(adrzer1,(char)0x0))
# define zerp           (outportb(adrzerp,(char)0x0))
# define usti           (outportb(adrusti,(char)0x0))
# define zeri           (outportb(adrzeri,(char)0x0))
# define budzik         (outportb(adrbudzik,(char)0x0))
# define almag1         (outportb(adralmagpisz1,(char)0x0))
# define almag2         (outportb(adralmagpisz2,(char)0x0))
# define stanc6         (inportb(adrczyts))
# define stanc7         (inportb(adrczyta))
# define stanc8         (inportb(adrczytz))
# define stane4         (inportb(adrczytr))
# define stane5         (inportb(adrczytdl))
# define stane6         (inportb(adrczytdh))
# define wyob           (inportw(MC42))
# define wyukz          (inportb(IT1B1PA))
# define INT            (inportb(IT1B1PC))
# define reset          (wyukz & MASKA0)
# define mpro           (wyukz & MASKA1)
# define pfin           (wyukz & MASKA2)
# define pfsn           (wyukz & MASKA3)
# define buton          (wyukz & MASKA5)
# define stanc6BUTON    (stanc6 & MASKA2)
# define stanc7OTW      (stanc7 & MASKA1)
# define stanc7BUD      (stanc7 & MASKA0)
# define stanc6INTER    (stanc6 & MASKA1)
# define wrweukz(data)  (outportw(IT2B3CC4,data))
# define wrweacl(data)  (outportb(IT1B1WY,data))
# define wralarm(data)  (outportw(MC42,data))
# define wrmagmb(data)  (outportw(adrpiszmb,data))
# define wrmagsb(data)  (outportw(adrpiszsb,data))
# define alarm          (wralarm(datsumal))
# define wylal          (wralarm(datwylal))
# define otw            (wralarm(datotw))
# define wrmagpam1(data) (mWritew(adrpiszpam1,data))
```

```
# define wrmagpam2(data) (mwritew(adrrejzpam2,data))
# define rmagmt          (inportb(adrczytmb))
# define rmagst          (inportb(adrczytsb))
# define rmagpam1        (mreadw(adrczytpam1))
# define rmagpam2        (mreadw(adrczytpam2))
# define zero_ai         (a1=a2=a3=a4=a5=a6=a7=a8=a9=a10=0)
# define cr()            (printf("\n\t\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^GLAD^\n"),getch())
# define beep()          (printf("%c",0x07))
```

/* proto.h */

```
int main(void);
int pr_ur(void);
char petla(void);
int help(void);
static char * fcvt(int *ap,int base,register char *cp,int len);
int format(register int (*putsub)(),register char *fmt,char *argp);
int aputc(int c);
void btmp(void);
int cmpbit(unsigned int x,unsigned int y);
unsigned long htol(unsigned char *b);
int itoa(int n,char s[]);
int itoha(unsigned int n,char s[]);
void mc(void);
int inportb(int adr);
int inportw(int adr);
void outportb(int adr,char dana);
void outportw(int adr,int dana);
char mreadb(unsigned long adr);
int mreadw(unsigned long adr);
void mwriteb(unsigned long adr,unsigned char data);
void mwritew(unsigned long adr,unsigned int data);
unsigned long pelzl(unsigned int i);
int scanf(char *fmt,int *args);
int scanfmt(int (*getsub)(),register char *fmt,register int **args);
int skipblank(void);
int scr_cdel(void);
int scr_clr(void);
int scr_curd(void);
int scr_curl(void);
int scr_curr(void);
int scr_curs(char lin,char col);
int scr_curu(void);
int scr_eol(void);
int scr_eos(void);
int scr_getc(void);
int scr_home(void);
int scr_ldel(void);
int scr_putc(char c);
char sgetch(void);
int xackpt(unsigned int n,unsigned long ad);
int menu(char *naglowek[],char *text[],char *podpis,int line_n,int *line_l);
char *ltob(unsigned long dana,unsigned n);
int ltoha(unsigned long n,char s[]);
int reverse(char s[]);
int piso(void);
int fhold(void);
int mmap2pt(unsigned int n,unsigned long ad);


void interrup(int nr_przerwania);
int stin(char * tekst);
int stout(int znak);
int getch(void);
int scr_pool(void);
void outtxt(char *);/*wskznik do tekstu w SI */
char outch(char znak);/* znak w AL */
void setint(int inter_nr,void(*probsluq)());
void setintad(int inter_nr,unsigned int seg,unsigned int offset);
void i_it1_mw(void);
void inic_it2(void);
void cli(void);
void sti(void);
void ipret(void);
void ster_prz(void);/* inicjalizuje ster. przerwan dla BTMC i INNT1(mc42) */
void delay(void);
void eoci(void);
void delayp(int cunt);
void adr_prz(void(*btmc)(),void(*int2)());
```

```
/*                    opracował:  Zbigniew Stanczak      mc42.h      */

typedef unsigned long UL ;
#define UL   (UL)
typedef unsigned int UI ;
#define UI  (UI)
typedef long L ;
#define L   (L)
static char * wy0 = " bledny stan sygnalu XACK";
static char * wyd = "\n\t\tProsze nastawic przelacznik AP w ";
static char * wydru = "\t\tpolozenie skrajne";
static char * wy2 = "blad w obwodzie Aa9--XACK lub Ac9--XACK";
static char * wy3 = "blad w obwodzie przelacznika AP lub XACK,U4(6,5),U3(8)";
static char * inf1 = "sygnal XACK/po zainstalowaniu\t\t\t\t\t\tOK!";
#define ADRR UI 0xfbfe
#define ADR2 UI 0xfb06
#define ADRTES UI 0xfbf6 /* adres pakietu mc42 testera */
#define kocm() outportw(0xfbd0,0x1fc)
#define kocd() outportw(0xfbd0,0x2fc)
#define koc()  outportw(0xfbd0,0xfc)
static int adrmc42[] = { 0x5,
                         0x85,
                         0x45,
                         0x0,
                         0x25,
                         0xa5,
                         0x65,
                         0xe5,
                         0x15,
                         0x95,
                         0x55,
                         0x0,
                         0x35,
                         0xb5,
                         0x75,
                         0xf5,
                         0xd,
                         0x8d,
                         0x4d,
                         0x0,
                         0x2d,
                         0xad,
                         0x6d,
                         0xed,
                         0x1d,
                         0x9d,
                         0x5d,
                         0x0,
                         0x3d,
                         0xbd,
                         0x7d,
                         0xfd,
};
```

```
                /* <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^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,(char)data))
# define wy4(data) (outportw(IOADR11,data))
# define czyt (c=CZYT_A(RWADR11Q0))
# define wyd (printf("stan=0x%x\n",CZYT_A(RWADR11Q0)))
# define in3 ((inportw(MC42)&wokmc42)>>8)
# define in4 (inportw(MC42)&utrprmc42)
# define in5 ((int)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\t* CR *\n"),k=1)
# define map2 ((inportb(IT1B3PC)>>5)&0x1)
# define INIT (outportb(IT1D&WY,(char)0x0))
# define NOTINIT (outportb(IT1D&WY,(char)0x20))
# define UST1(adrmw31) (mreadw((long)adrmw31))
# define ZER1(adrmw31) (mwritew((long)adrmw31,0x0))
# define UST2(adrmw31) (mreadw((long)adrmw31))
# define ZER2(adrmw31) (mwritew((long)adrmw31,0x0))
# define CZYT_S(adrmw31) (mreadw((long)adrmw31))
# define CZYT_A(adrmw31) (mreadw((long)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))
```

/*adrmw31.h */

```
/*****************************************************************/
#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
#define RWADR00Q0 (unsigned long)0X9EBEE /* syg. Q0   (4-5) ZWAR. (3-6) ZWAR. */
#define RWADR01Q0 (unsigned long)0X8E9EE /* syg. Q0         ZWAR.       ROZW. */
#define RWADR10Q0 (unsigned long)0X8EAEE /* syg. Q0         ROZW.       ZWAR. */

#define RWADR11Q5 (unsigned long)0x8E8E4 /* syg. Q5    (4-5) ROZW. (3-6) ROZW. */
#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
#define RWADR00Q0 (unsigned long)0XEBEE  /* syg. Q0   (4-5) ZWAR. (3-6) ZWAR. */
#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) ROZW. */
#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 0xfed8         /* 8255 */
# define IT1B1PB 0xfeda
# define IT1B1PC 0xfedc
# define IT1B1   0xfede

# define IT1D6WY 0xfed0         /* 8212 */

# define IT1B3PA 0xfec8         /* 8255 */
# define IT1B3PB 0xfeca
# define IT1B3PC 0xfecc
# define IT1B3   0xfece

# define IT1B2WY 0xfec0         /* 8212 */

# define IT3B6PA 0xfdc0         /* 8255 */
# define IT3B6PB 0xfdc2
# define IT3B6PC 0xfdc4
# define IT3B6   0xfdc6

# define IT3B5PA 0xfdc8         /* 8255 */
# define IT3B5PB 0xfdca
# define IT3B5PC 0xfdcc
# define IT3B5   0xfdce

# define IT3A6D1 0xfdd0         /* 8251 */
# define IT3A6C1 0xfdd2
# define IT3A6D2 0xfdd4
# define IT3A6C2 0xfdd6

# define IT3B3WY 0xfdd8         /* 8212 */

# define IT2B1C1 0xfbd0         /* out,out 8212 */
# define IT2A1C2 0xfbd0         /* in,in   8212 */
# define IT2B2B4 0xfbc0         /* in,out  8212 */
# define IT2B3C4 0xfbc8         /* out,out 8212 */
# define IT2A2C3 0xfbc8         /* in,in   8212 */

# define MC42    0xfbf6         /*         8212 */
```

/* proext.h */

```
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 sek_num;                 /* nr przebiegu uruchomieniowego */
extern int buf_btmo;                /* stan btmo */
extern int buf_mc42;                /* stan przeciazenia mc42 : 00-OK 01-BLAD */
extern int buf_welcfi;
extern void btmo();
extern void int39();
extern void mc();
typedef struct{
        char sb;/* szczyt bufora */
        char pb;/* aktualna pozycja bufora */
        int *p;/* ptr. bufora */
}STDIN;
```

/* tstglb.h */

```
extern char enter_bufor[30];        /* bufor klawiatury funkcji scr_getc() */
int echo_on_off;                    /* echo funkcji scr_getc() */
extern void btmo();                     /* przerwanie brak XACK */
extern void mc();                     /* przerwanie przeciazenie MC42  */
int lin_num;                        /* nr linii kursora */
int col_num;                        /* nr kolumny kursora */
int mep_line;
int sek_num;                        /* nr przebiegu uruchomieniowego */
int buf_btmo;                       /* stan btmo */
int buf_welcfi;
int buf_mc42;                       /* stan przeciazenia mc42 : 00-OK 01-BLAD */
typedef struct{
        char sb;/* szczyt bufora */
        char pb;/* aktualna pozycja bufora */
        int *p;/* ptr. bufora */
}STDIN;
```

/* ascii.h */

```
#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 #\n"),getch())
#define blad() (printf("\n\t\t\t\t\t\t\t\t\t\t^BLAD^\n"))
#define OUT    outport
#define IN     inport
```

/* lmacros.h */

```
        nlist
; Copyright (C) 1985 by Manx Software Systems, Inc.
; :ts=8
        ifndef  MODEL
MODEL   equ     0
        endif
        if      MODEL and 1
        largecode
FARPROC equ 1
FPTRSIZE equ    4
        else
FPTRSIZE equ    2
        endif
        if      MODEL and 2
LONGPTR equ 1
        endif

;this macro to be used on returning
;restores bp and registers
pret    macro
if havbp
        pop bp
endif
        ret
        endm

internal macro  pname
        public  pname
pname   proc
        endm

intrdef macro   pname
        public  pname
ifdef FARPROC
        pname   label   far
else
        pname   label   near
endif
        endm

procdef macro   pname, args
        public pname&_
ifdef   FARPROC
        _arg    = 6
        pname&_ proc    far
else
        _arg    = 4
        pname&_ proc    near
endif
ifnb <args>
        push bp
        mov bp,sp
        havbp = 1
        decll <args>
else
        havbp = 0
endif
        endm

entrdef macro   pname, args
        public pname&_
ifdef   FARPROC
        _arg    = 6
        pname&_:
else
        _arg    = 4
        pname&_:
endif
ifnb <args>
if      havbp
        push    bp
        mov     bp,sp
else
        error must declare main proc with args, if entry has args
endif
        decll <args>
endif
        endm

;this macro equates 'aname' to arg on stack
decl    macro   aname, type
;;'byte' or anything else
havtyp  = 0
ifidn   <type>,<byte>
        aname   equ     byte ptr _arg[bp]
        _arg = _arg + 2
        havtyp = 1
endif
ifidn   <type>,<dword>
        aname   equ dword ptr _arg[bp]
        _arg = _arg + 4
        havtyp = 1
endif
ifidn <type>,<cdouble>
        aname   equ qword ptr _arg[bp]
        _arg = _arg + 8
        havtyp = 1
endif
ifidn <type>, <ptr>
        ifdef LONGPTR
                aname   equ dword ptr _arg[bp]
                _arg = _arg + 4
```

```
        else
                aname equ       word ptr _arg[bp]
                _arg = _arg + 2
        endif
        havtyp = 1
endif
ifidn <type>, <fptr>
        ifdef FARPROC
                aname   equ dword ptr _arg[bp]
                _arg = _arg + 4
        else
                aname equ       word ptr _arg[bp]
                _arg = _arg + 2
        endif
        havtyp = 1
endif
ifidn <type>, <word>
        aname equ       word ptr _arg[bp]
        _arg = _arg + 2
        havtyp = 1
endif
ife     havtyp
        error -- type is unknown.
endif
        endm

;this macro loads an arg pointer into DEST, with optional SEGment
ldptr   macro   dest, argname, seg
ifdef   LONGPTR
        ifnb <seg>              ;;get segment if specified
                ifidn <seg>,<es>
                        les     dest,argname
                else
                        ifidn <seg>,<ds>
                                lds dest,argname
                        else
                                mov dest, word ptr argname
                                mov seg, word ptr argname[2]
                        endif
                endif
        else
                ifidn <dest>,<si>               ;;si gets seg in ds
                        lds     si, argname
                else
                        ifidn <dest>,<di>       ;;or, es:di
                                les     di, argname
                        else
                                garbage error: no seg for long pointer
                        endif
                endif
        endif
else
        mov dest, word ptr argname      ;;get the pointer
ENDIF
        ENDM

decll   macro   list
        IRP     i,<list>
        decl i
        ENDM
        ENDM

pend    macro   pname
pname&_ endp
        endm

retptrm macro   src,seg
mov     ax, word ptr src
ifdef   LONGPTR
        mov     dx, word ptr src+2
endif
        endm

retptrr macro   src,seg
mov     ax,src
ifdef LONGPTR
        ifnb <seg>
                mov     dx, seg
        endif
endif
        endm

retnull macro
ifdef LONGPTR
        sub     dx,dx
endif
        sub     ax,ax
        endm

pushds  macro
        ifdef LONGPTR
        push    ds
        endif
        endm

popds   macro
        ifdef LONGPTR
        pop     ds
        endif
        endm

finish  macro
```

```
codeseg ends
        endm

        list
codeseg segment byte public 'code'
        assume  cs:codeseg
```