0711 A

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

Ośrodek Automatyki Elektrycznej

Pracownia Systemów Wizyjnych

Główny wykonawca dr inż. Bohdan Kontrymowicz

Wykonawcy mgr inż. Andrzej Zasucha
mgr inż. Dariusz Okrasa
mgr inż. Marek Szymański

Konsultant

Nr zlecenia RP-61

System wizyjny dla robotów IRp

Zadanie 3.1 - Dokumentacja, wykonanie, oprogramowanie i badanie modelu systemu wizyjnego 2-D.

Zleceniodawca CPBR 7.1

Pracę rozpoczęto dnia 01.05.88 **zakończono dnia** 30.11.89

Kierownik Pracowni dr inż. J. Frontczak

Z-ca Dyrektora d/s Automat. i Pomiar. doc. dr inż. T. Gałązka

Kierownik Ośrodka dr inż. B. Kontrymowicz

Praca zawiera:

stron 6
rysunków
fotografii
tabel
tablic
załączników 10

Nr rejestr. 6370

Rozdzielnik - ilość egz:

Egz. 1 BOINTE
Egz. 2 OAE
Egz. 3 OAE
Egz. 4 OAE
Egz. 5
Egz. 6

Analiza deskryptorowa

SYSTEMY WIZYJNE + ANALIZA SCENY
ROBOTY PRZEMYSLOWE + ROZPOZNAWANIE OBRAZOW

ROBOT PRZEMYSŁOWY, URZĄDZENIE WIZYJNE, BADANIE MODELU MATERIALNEGO

Analiza dokumentacyjna

W sprawozdaniu zamieszczono dokumentację modelu systemu wizyjnego 2-D dla robta IRp. Opisano badanie funkcjonalne modelu. Sprawozdanie zawiera też wnioski wynikające z fazy uruchamiania modelu dotyczące dalszej realizacji pracy.

Tytuły poprzednich sprawozdań

Zadanie 1.1- Prace studialne
Nr rejestr.5836
Zadanie 1.2- Opracowanie założeń i wybór algorytmów
Nr rejestr.5918
Zadanie 2.1- Projekt sprzętu i oprogramowania systemu wizyjnego 2-D i badanie algorytmów przetwarzania wizji
Nr rejestr.6035

338.45:62/69].002.1/.2 - Roboty przemysłowe
621.397.6 Urządzenia wizyjne

UKD

PIAP 41/88 10000

2

SPIS TRESCI

1. WSTĘP.

Zadaniem systemu wizyjnego 2-D dla robota przemysłowego jest rozpoznanie obiektu znajdującego się w polu widzenia kamery tzn.przyporządkowanie go do jednej z klas oraz określenie parametrów chwytu czyli położenia punktu Tp we współrzędnych kamery oraz kąta obrotu wokół tego punktu w stosunku do orientacji wyznaczonego wzorca.Wykonany sprzęt oraz oprogramowanie systemu wizyjnego pozwala na realizację tego zadania zgodnie z założeniami dla scen o najmniejszym stopniu skomplikowania tzn. jeżeli w polu widzenia kamery znajdują się pojedyńcze detale.

Dokładny opis przeznaczenia, budowy i funkcji systemu zamieszczony jest w sprawozdaniu z realizacji zadania 2.1.(nr rejestracyjny PIAP 6034).

Ze względu na fakt, że w chwili obecnej robot nie jest przygotowany do współpracy z układem sensorycznym, funkcje robota (z punktu widzenia systemu wizyjnego) są symulowane na komputerze IBM AT.Dane pomiędzy systemem wizyjnym a IBM są przesyłane łączem szeregowym zgodnie z protokołem wymiany informacji między robotem a układami sensorycznymi.

2.REALIZACJA ZADANIA.

W ramach realizacji etapu wykonane zostały następujące prace:

- uruchomiono model systemu wizyjnego 2-D dla robota przemysłowego,
- wykonano i uruchomiono oprogramowanie systemu,
- przeprowadzono badania funkcjonalne systemu z oprogramowaniem.

Prace uruchomieniowe przeprowadzono z wykorzystaniem komputera IBM jako symulatora robota oraz dla potrzeb uruchomienia oprogramowania dla pakietu MM16 oraz emulatora procesora INTEL 8031.

2.1 Uruchomienie układu.

W trakcie uruchamiania oprogramowania dla pakietu jednostki centralnej MM16 okazało się, że nie jest konieczne korzystanie z dodatkowego pakietu pomięci MW30- w założeniach projektowych przewidywano wykorzystanie tego pakietu.
Konieczne natomiast stało się wykorzystanie pakietu transmisji szeregowej MI24 jako łącza szeregowego pomiędzy MM16 a IBM.Wymiana informacji między tymi urządzeniami sensowna jest bowiem jedynie przez łącze szeregowe, natomiast to łącze w pakiecie MM16 nie jest oprogramowane. Dodatkowo,nie można było wykorzystać wszystkich przewidzianych styków na platerze magistrali kasety przeznaczonych do połączeń indywidualnych, jako że część z nich wykorzystuje pakiet MM16.
Pakiety systemu wizyjnego:binaryzacji, filtracji i preprocesora zrealizowane zgodnie z założeniami.

2.2. Oprogramowanie systemu wizyjnego.

Oprogramowanie systemu wizyjnego składa się z następujących programów;
-"hist",
-"prog",
-"mm16",
-"s1".

Programy "hist" oraz "prog" służą do określania poziomu komparacji na podstawie rozkładu jasności obrazu w celu zbinaryzowania sygnału wideo.Program "prog" wpisuje dane początkowo do liczników i rejestratorów, natomiast program "hist"- inicjowany przerwaniami- oblicza wartość progu komparycji każdorazowo dla nowego obrazu.

Programy te zostały skompilowane kroskompilatorem ICC (IAR C Compiler for Microprocesor Development v.2.00) oraz skonsolidowane linkerem tej firmy.Uruchomienie odbyło się przy użyciu emulatora NEW MICE II produkcji firmy Mikrotek.Kod wynikowy został wpisany do pamięci EPROM układu mikroprocesora 8031.

Program mm16 rezydujący w pamięci pakietu MM16 na podstawie danych przygotowanych sprzętowo oblicza konieczne do klasyfikacji i określenia obrotu parametry detali, a następnie po odpowiednim żądaniu wysyła do IBM'a.Program ten ściśle odpowiada funkcjom procedur "klasyfikacji"oraz "orientacja" przedstawionym w sposób szczegółowy w sprawozdaniu z realizacji zadania 2.1.

Należy nadmienić, że w celu skrócenia czasu wykonywania tych zadań, opracowano procedury dla kooprocesora arytmetycznego 8087 na obliczanie funkcji pierwiastka kwadratowego oraz tangensa.

Program s1 dla komputera IBM AT symuluje funkcje robota odpowiednio do funkcji przedstawionych w "Założeniach na komunikację robota przemysłowego z układami sensorycznymi". Dodatkowo wyniki pewnych funkcji wizualizowane są na ekranie monitora. W szczególności możliwe jest przedstawienie położenia środka ciężkości detalu, jego kontur, numer klasy detalu oraz kąt odbioru

2.3. Badania funkcjonale.

W celu przeprowadzenia badań funkcjonalnych wycięto z czarnej kalki kilka figur geometrycznych, które przyklejone na papier kolejno umieszczano w polu widzenia kamery w odległości ostrego widzenia.

Badania funkcjonalne objęły:

a) sprawdzenie poprawności obliczania środka ciężkości detali oraz określenie błędu tych obliczeń,
   - w tym celu przed kamerą przesuwano detale obserwując jednocześnie na ekranie odpowiednie przesuwanie się środka ciężkości. Stwierdzono, że środek ciężkości jest liczony poprawnie dla całego ekranu (256x256) z dokładnością dwóch pikseli, wystarczającą dla postawionego zadania.

b) sprawdzenie poprawności rozpoznawania detali,
   - w tym celu dla kilku detali dokonano klasyfikacji w trybie "uczenie". Następnie "znane" systemowi wizyjnemu detale umieszczano w polu widzenia kamery obserwując jednocześnie podawany przez system na monitorze numer klasy. Stwierdzono poprawne rozpoznawanie detali przez system wizyjny.

c) sprawdzenie poprawności liczenia kąta obrotu oraz określenie dokładności obliczeń,
   - w tym celu dokonywano wielokrotnych obliczeń kąta obrotu dla różnych detali i ich różnych położeń. Stwierdzono, że system spełnia ustalone założenia i liczy kąty z błędem nie przekraczającym 5 stopni.

d) sprawdzenie czasu trwania rozpoznawania oraz liczenia kątów,
   - w tym celu rozpoczęcie i zakończenie tej funkcji sygnalizowane było programowo ustawionymi sygnałami, które były rejestrowane przez analizator stanów logicznych. Dla różnych detali, przy różnej liczbie klas czasy były różne, stwierdzono jednak, że nie przekraczają ustalonych w założeniach 64 ms.

3. WNIOSKI DOTYCZACE DALSZEJ REALIZACJI PRACY

Opracowując założenia na system wizyjny dla robotów IRp przyjęto wykorzystać istniejący już w systemie INTELDIGIT-PROWAY pakiet jednostki centralnej MM-16 oparty na procesorze 8086 z koprocesorem arytmetycznym 8087. Takie podejście zdawało się oszczędzać czas potrzebny na opracowanie systemu wizyjnego i było argumentem przeważającym mimo oczywistej nadmiarowości układowej.

W praktycznej realizacji tej koncepcji napotkano na znaczne trudności w fazie uruchamiania modelu.

Nieznany wcześniej fakt wykorzystywania przez jednostkę centralną MM-16 linii zarezerwowanych dla potrzeb użytkownika, spowodował konieczność przesyłania niektórych sygnałów pomiędzy pakietami systemu wizyjnego dodatkowym kablem. Takie rozwiązanie jest mało eleganckie technicznie i podatne na zakłócenia.

Następna trudność wiązała się z wykorzystaniem magistrali rezydentnej. Przesyłane po niej dane łatwo się zakłócały. W rezultacie prace nad uruchomieniem modelu urządzenia trwały znacznie dłużej, niż planowano. Ponadto zdobyte w tej fazie doświadczenia pozwalają wątpić, czy prototyp systemu wizyjnego wykonany według tej koncepcji przejdzie badania kompatybilności elektromagnetycznej (KEM).

W tej sytuacji wykonawcy proponują wykonać daleko idące modyfikacje konstrukcyjne, a mianowicie:

- zrezygnować ze stosowania jednostki centralnej MM-16 na rzecz opracowania własnej specjalizowanej jednostki centralnej opartej na procesorze 80186, zajmującej tylko część płytki formatu stosowanego w systemie INTELDIGIT-PROWAY;
- cały system wizyjny składający się z kilku płytek elektronicznych skonstruować w postaci jednego bloku montowanego w kasecie INTELDIGIT-PROWAY, z jedną wspólną

płytą czołową i połączeniem z magistralą kasety wyłącznie dla pobierania napięć zasilających;

Proponowana zmodyfikowana koncepcja ma następujące zalety:
- zmniejszy się objętość systemu wizyjnego przynajmniej o dwa stanowiska w kasecie zajmując łącznie najprawdopodobniej 3 stanowiska: w efekcie system powinien być dla użytkownika relatywnie tańszy,
- zostanie zastosowany procesor 80186 zawierający w sobie układy, które na pakiecie MM-16 występują w otoczeniu procesora 8086 jako osobne układy scalone; większy stopień scalenia pozwoli nie tylko zmniejszyć objętość układu, lecz również zwiększy odporność na zakłócenia,
- zmniejszając ilość elementów stosowanych w systemie wizyjnym poprzez stosowanie układów o wyższym stopniu scalenia wzrasta atrakcyjność systemu, szczególnie w przypadku ewentualnej realizacji wersji autonomicznej przewidywanej dla innych zastosowań niż z robotem IRp,
- w MERA-PIAP istnieją już doświadczenia w zakresie stosowania procesora 80186, co daje rękojmię sprawnego wykonania proponowanej modyfikacji.

ZAŁĄCZNIK NR 1.

PROGRAM MM16.C

```
#include "stdio.h"
int i,BIT,ADRKLASY,NRKLASY,SK;
long SUMA;
void srod_ciez(void);
void przep(int);
void rozpoz(void);
void par(int);
void pobkatow(int);
void pobieranie(int);
void wyslanie(int,int);
void danepocz(void);
void wyslac(void);
void odczyt(void);
void rot(void);

main()
{
for(i=0;i<=1;i++)
   {
       outportb(0x002e,0x36);      /* inicjalizacja TIMER-a           */
       outportb(0x0028,33);        /* podzial czestotliwosci przez 33 */
       outportb(0x0028,00);

       outportb(0x0032,0x70);      /* RESET ukladu 8251 */

       outportb(0x0032,0x0fe);     /* inicjalizacja 8251 - :16,e,8,2 */
       outportb(0x0032,0x37);      /* COMMAND - ustawia RTS=0,DTR=0 */
   }


BIT = peekb(0x0C000,0x00);    /* odblokowuje zliczanie */

danepocz();

  do
     pobieranie(0x0e00);          /* pobiera przesylke i wpisuje ja od adresu 0x0600 */
  while(peekb(0x0e01,0x00)!=1);      /* czeka az kod przesylki jest rowny 1 */

i=0;
ADRKLASY=0x0a00;
NRKLASY=1;
do
   {
      odczyt();
      srod_ciez();         /* oblicza wsp. srodka ciez. i wpisuje je pod adr. 0F02 i 0F03 */
      pobieranie(0x0e00);
      switch(peekb(0x0e01,0x00))
        {
          case 3: i=1;             /* Dezaktywacja ukl.wizyjnego  */
                  break;           /* Przesylka o kodzie k=3 oznacza NACT SENS.*/
          case 2:
          case 4: rozpoz();
                  wyslanie(7,0x4ff);
                  break;

          case 10: wyslac();                    /* Wysyla wspolrzedne konturu */
                   przep(ADRKLASY);
```

```
                    pokeb(ADRKLASY,0x00,NRKLASY);
                    par(ADRKLASY);          /* Zapamietuje obwod i pole oraz */
                                            /* oblicza P5, P6,P7 i zapisuje je od adresu 0x3805 */
                    pobieranie(ADRKLASY+7); /* pobiera katy i zapisuje je od adresu 0x3808 */

                    if(ADRKLASY<=0x0d00)
                       {
                         ADRKLASY=ADRKLASY+0x100;
                         NRKLASY++;
                       }
                    break;
          }
       if(peekb(0x0e01,0x00)==0x0b)
            {
             if(peekb(0x0504,0x00)==0)    /* Jezeli w kom.0504 jest 0 to oznacza */
               {                          /* to ze obiekt jest niezidentyfikowany */
                  pokeb(0x0e33,0x00,0);   /* i nie ma sensu badac obrotu. Zatem */
                                          /* do komorki 0632 wpisujemy 0 */
                  SUMA=peekb(0x0e31,0x00)+peekb(0x0e32,0x00)+peekb(0x0e33,0x00)+peekb(0x0e34,0x00);
                  SK=256-(SUMA & 0x0ff);
                  pokeb(0x0e35,0x00,SK);              /*   suma kontrolna SK */
               }
             else
               {
                  rot();
               }
             wyslanie(6,0x0e30);
            }
     }
while(i==0);
#asm
int 3
#endasm
}



void danepocz(void)
{
                                    /* Przesylka kata obrotu          */
pokeb(0xe30,0x00,0x3a);             /* poczatek przesylki tzn. znak : */
pokeb(0xe31,0x00,0x03);             /* liczba przesylanych bajtow     */
pokeb(0xe32,0x00,16);               /* kod przesylki                  */

pokeb(0x4ff,0x00,0x3a);       /* poczatek przesylki tzn. znak : */
pokeb(0x500,0x00,0x04);       /* liczba bajtow rowna 4 */
pokeb(0x501,0x00,0x21);       /* kod przesylki */
                              /* komorka 0x0502 zarezerwowana dla wsp. srodka ciez. X     */
                              /*   #     0x0503     #       #   #     #     #   Y     */
                              /*   #     0x0504     #       #   numeru klasy - NRKLASY */
                              /*   #     0x0505     #       #  sumy kontrolnej SK      */

     pokeb(0x50a,0x00,0x00);
     pokeb(0x50d,0x00,0x00);
}
```

```
void pobieranie(int adr)
{
 int RRDY,i,m,LICZBA,DANA,SK;
 long SUMA;
 void error(void);

  do
   {                                /*                                  */
    outportb(0x0032,0x26);          /* COMMAND - ustawia RTS=0,DTR=0 */
    do
       RRDY=inportb(0x0032) & 0x02;
    while(RRDY==0);                 /*                                  */
    outportb(0x0032,0x00);          /* COMMAND - ustawia RTS=1,DTR=1 */
   }
  while(inportb(0x0030)!=0x3A);     /* Czeka na poczatek przesylki tzn. na : */


  do
   {                                /*                                  */
    outportb(0x0032,0x26);          /* COMMAND - ustawia RTS=0,DTR=0 */
    do
       RRDY=inportb(0x0032) & 0x02;
    while(RRDY==0);                 /*                                  */
    outportb(0x0032,0x00);          /* COMMAND - ustawia RTS=1,DTR=1 */
    LICZBA=inportb(0x0030);
   }
  while(LICZBA==0x3A);              /* Po znaku : powinien przyjsc bajt */
                                    /* oznaczajacy liczbe przesylanych bajtow */
  pokeb(0x400,0x00,LICZBA);

  i=0;
  SUMA=LICZBA;
  m=1;

  do
   {                                /*                                  */
    outportb(0x0032,0x26);          /* COMMAND - ustawia RTS=0,DTR=0 */
    do
       RRDY=inportb(0x0032) & 0x02;
    while(RRDY==0);                 /*                                  */
    outportb(0x0032,0x00);          /* COMMAND - ustawia RTS=1,DTR=1 */
    DANA=inportb(0x0030);
    if(DANA==0x3A)
       error();
    pokeb(adr+m,0x00,DANA);
    pokeb(0x401+m,0x00,DANA);
    SUMA=SUMA+DANA;
    m++;
    i++;
   }
while(i<LICZBA);

                                   /*                                  */
outportb(0x0032,0x26);             /* COMMAND - ustawia RTS=0,DTR=0 */
do
   RRDY=inportb(0x0032) & 0x02;
while(RRDY==0);                    /*                                  */
outportb(0x0032,0x00);             /* COMMAND - ustawia RTS=1,DTR=1 */
```

```
  SK=inportb(0x0030);
  pokeb(0x410,0x00,SK);
  if(((SUMA+SK) & 0x0ff) !=0)
      error();
}


void error(void)
  {
  }



void wyslanie(int l,int adres)
{
int m,dana,DSR,TRDY,STATUS;
                          /*                          ___  ___   */
outportb(0x32,0x27);      /* COMMAND - ustawia RTS=0,DTR=0 */
for(m=0;m<l;m++)
   {
      do
          {
              STATUS=inportb(0x32);
              DSR=STATUS & 0x80;
              TRDY=STATUS & 0x01;
          }
      while ((TRDY==0) || (DSR==0));
      dana=peekb(adres+m,0x00);            /* Pobiera bajt z pod adresu="adres+m"*/
      outportb(0x0030,dana);               /* i wysyla go na port szeregowy */
   }                      /*                          ___  ___   */
outportb(0x0032,0x00);    /* COMMAND - ustawia RTS=1,DTR=1 */
}



void wyslac(void)
{
int BIT,KON,adr,i,r,dana,DSR,TRDY,STATUS;
adr=0x3701;
   KON = (peekw(0x506,0x00) * 4) + 4;
      do{
         BIT = peekb(0x4000,0x00);
         }
      while(( BIT & 1 ) == 0 );
      do{
         BIT = peekb(0x4000,0x00);
         }
      while(( BIT & 1 ) == 1 );
      BIT = peekb(0x8000,0x1000); /* blokuje zliczanie */
      for( i = 0; i < KON; i++ ){         /* czyta punkty konturu z RAM'u      */
         BIT = peekw(0x7ffd-2*i,0x1000);  /* systemu do pamieci MM16 od adresu */
         pokew(0x3700+2*i,0x00,*BIT);      /* 0x3000 */
      }

      do{
         BIT = peekb(0x4000,0x00);
         }
```

```
while(( BIT & 1 ) == 1 );

BIT = peekb(0x0C000,0x00);    /* odblokowuje zliczanie */

outportb(0x32,0x27);          /* COMMAND - ustawia RTS=0,DTR=0 */
for( i = 0; i < KON; i++ ){
      do
      {
        STATUS=inportb(0x32);
        DSR=STATUS & 0x80;
        TRDY=STATUS & 0x01;
      }
    while ((TRDY==0) || (DSR==0) );
    dana=peekb(adr+2*i,0x00);            /* Pobiera bajt z pod adresu="adres+m"*/
    outportb(0x0030,dana);             /* i wysyla go na port szeregowy */
}
      do
      {
        STATUS=inportb(0x32);
        DSR=STATUS & 0x80;
        TRDY=STATUS & 0x01;
      }
      while ((TRDY==0) || (DSR==0) );
      outportb(0x0030,0x00);           /* wysyla dwa zera */

      do
      {
        STATUS=inportb(0x32);
        DSR=STATUS & 0x80;
        TRDY=STATUS & 0x01;
      }
      while ((TRDY==0) || (DSR==0) );
      outportb(0x0030,0x00);           /* wysyla dwa zera */

adr=0x3700;
for( i = 0; i < KON; i++ ){
      do
      {
        STATUS=inportb(0x32);
        DSR=STATUS & 0x80;
        TRDY=STATUS & 0x01;
      }
    while ((TRDY==0) || (DSR==0) );
    dana=peekb(adr+2*i,0x00);            /* Pobiera bajt z pod adresu="adres+m"*/
    outportb(0x0030,dana);             /* i wysyla go na port szeregowy */
}

      do
      {
        STATUS=inportb(0x32);
        DSR=STATUS & 0x80;
        TRDY=STATUS & 0x01;
      }
      while ((TRDY==0) || (DSR==0) );
```

```
            outportb(0x0030,0x00);

            do
            {
              STATUS=inportb(0x32);
              DSR=STATUS & 0x80;
              TRDY=STATUS & 0x01;
            }
            while ((TRDY==0) || (DSR==0) );
            outportb(0x0030,0x00);
          outportb(0x0032,0x00);        /* COMMAND - ustawia RTS=1,DTR=1 */
}


/*  Z komorek 0F08 i 0F09 pobiera 2-bajtowe POLE. Z komorek 0F0A,0F0B i 0F0C
    pobiera 3-bajtowe S1. Nastepnie wykonuje dzielenie S1/POLE i otrzymujemy
    wspolrzedna X. Analogicznie oblicza Y. Z metody obliczania wspolrzednych
    wynika, ze naleza one do przedzialu od 0 do 255, a zatem mozna je zapisac
    jednym bajtem. Wspolrzedne srodka ciezkosci X i Y zapisywane sa pod adresami
    0F02 i 0F03.
    Obliczana jest rowniez suma kontrolna bajtow z kom. od 0f00 do 0f04 i wpisywana
    do komorki 0f05   */


void odczyt(void)
{
int i,BIT;

     do{
       BIT = peekb(0x4000,0x00);
       }
     while(( BIT & 1 ) == 1 );
     do{
       BIT = peekb(0x4000,0x00);
       }
     while(( BIT & 1 ) == 0 );
     do{
       BIT = peekb(0x4000,0x00);
       }
     while(( BIT & 1 ) == 1 );

     BIT = peekb(0x8000,0x1000); /* blokuje zliczanie */

     pokew(0x8000,0x00,0x0c7);
     pokew(0x8000,0x00,0x87);
     pokew(0x8006,0x00,0x00);
     pokew(0x8006,0x00,0x00);
     pokew(0x8004,0x00,0x00);
     pokew(0x8004,0x00,0x00);

     pokew(0x00,0x1000,0x0c7);
     pokew(0x00,0x1000,0x87);
     pokew(0x06,0x1000,0x00);
     pokew(0x06,0x1000,0x00);
     pokew(0x04,0x1000,0x00);
     pokew(0x04,0x1000,0x00);

     BIT = peekb(0x0C000,0x00);
```

```
        do{
           BIT = peekb(0x4000,0x00);
           }
        while(( BIT & 1 ) == 0 );
        do{
           BIT = peekb(0x4000,0x00);
           }
        while(( BIT & 1 ) == 1 );

        BIT = peekb(0x8000,0x1000);

        BIT = peekb(0x8006,0x00);
        pokeb(0x506,0x00,BIT);
        BIT = peekb(0x8006,0x00);
        pokeb(0x507,0x00,BIT);
        BIT = peekb(0x8004,0x00);
        pokeb(0x508,0x00,BIT);
        BIT = peekb(0x8004,0x00);
        pokeb(0x509,0x00,BIT);

        BIT = peekb(0x06,0x1000);
        pokeb(0x50b,0x00,BIT);
        BIT = peekb(0x06,0x1000);
        pokeb(0x50c,0x00,BIT);
        BIT = peekb(0x04,0x1000);
        pokeb(0x50e,0x00,BIT);
        BIT = peekb(0x04,0x1000);
        pokeb(0x50f,0x00,BIT);

     BIT = peekb(0x0C000,0x00);
}


void srod_ciez(void)
{
int X,Y,SK;
long S1,S2,POLE,SUMA;

POLE =(long)peekw(0x508,0x00) * 4L;
S1= ((long)peekb(0x50c,0x00)*65536L + (long)peekb(0x50b,0x00)*256L) * 4L;
S2= ((long)peekb(0x50f,0x00)*65536L + (long)peekb(0x50e,0x00)*256L) * 4L;

X=(int)(S1/POLE);
Y=(int)(S2/POLE) + 4;

pokeb(0x502,0x00,Y);
pokeb(0x503,0x00,X);

SUMA=peekb(0x500,0x00)+peekb(0x501,0x00)+peekb(0x502,0x00)+peekb(0x503,0x00)
     +peekb(0x504,0x00);
SK=256-(SUMA & 0x0ff);
pokeb(0x505,0x00,SK);
}



void przep(int adr)
 {
```

```
  int POLE,OBWOD;

  POLE=peekw(0x0508,0x00);       /* Przepisanie bajtow POLA i OBWODU */
  pokew(adr+0x03,0x00,POLE);     /* z obszaru roboczego do tablicy   */
  OBWOD=peekw(0x0506,0x00);      /* wzorcow                          */
  pokew(adr+0x01,0x00,OBWOD);    /* Pole i obwod 4-ry razy mniejsze  */
 }                               /* od rzeczywistych                 */


void par(int adr)
{
int i,KON,XCIEZ,YCIEZ,KONTX,KONTY,P5,P6,P7,ADRX;
long ODLX,ODLY,ODL;

  XCIEZ=peekb(0x503,0x00);
  YCIEZ=peekb(0x502,0x00);
  P5=255;                        /* minimalna odleglosc */
  P6=0;                          /* maksymalna odleglosc*/
  P7=0;                          /* srednia  odleglosc  */
  ADRX=0x3700;
  KON = (peekw(0x506,0x00)*4) + 4;
    for( i = 0; i < KON; i++ ){
      KONTX=peekb(ADRX+2*i,0x00);
      KONTY=peekb(ADRX+1+2*i,0x00);

      ODLX=fabs(KONTX-XCIEZ);
      ODLY=fabs(KONTY-YCIEZ);

      ODL=sq2(ODLX*ODLX+ODLY*ODLY);     /* oblicza pierwiastek*/

      if(ODL>P6)
           P6=ODL;
      if(ODL<P5)
            P5=ODL;
      P7=P7+ODL;
    }

 P7=P7/KON;

 pokeb(adr+0x05,0x00,P5);
 pokeb(adr+0x06,0x00,0);
 pokeb(adr+0x07,0x00,P7);
}

void rozpoz(void)
{
int adres,NR;
long ROZ,WSP1,WSP2,WSP3,WSP;
adres=0x0a00;

do
{
  if(peekb(adres,0x00)!=0)
    {
WSP1=fabs(peekw(0x506,0x00)-peekw(adres+1,0x00)); /* Roznica obwodu obliczonego i obwodem wzorcowym */
WSP2=fabs(peekw(0x508,0x00)-peekw(adres+3,0x00)); /* Roznica pola obliczonego i pola wzorcowym */

     WSP=WSP1*WSP1+WSP2*WSP2;
```

```
      ROZ=sq2(WSP);                                    /* pierwiastek z sumy dwoch kwadratow */

      if(ROZ<=100)
        {
           par(0x0e10);                                  /* Oblicza parametry P5,P6,P7 */

      WSP1=fabs(peekb(0x0e15,0x00)-peekb(adres+5,0x00)); /* Roznica P5 obliczonego i P5 wzorcowego */
      WSP2=fabs(peekb(0x0e16,0x00)-peekb(adres+6,0x00)); /* Roznica P6 obliczonego i P6 wzorcowego */
      WSP3=fabs(peekb(0x0e17,0x00)-peekb(adres+7,0x00)); /* Roznica P7 obliczonego i P7 wzorcowego */

           WSP=WSP1*WSP1+WSP2*WSP2+WSP3*WSP3;

           ROZ=sq2(WSP);                 /* pierwiastek z sumy trzech kwadratow */

           if(ROZ>300)
                 pokeb(0x504,0x00,0);
           else
              {
                 NR=peekb(adres,0x00);
                 pokeb(0x504,0x00,NR);
              }
        }
      else                              /* Jezeli duza roznica to obiekt nierozpoznany */
           pokeb(0x504,0x00,0);         /* i wpisujemy 0 do komorki 0504  */
     }
   adres=adres+0x100;
  }
while((peekb(0x504,0x00)==0) && (adres<=0x0d00));
}


void rot( void )
{

int KON,SK,OBROT,f,adres,ROZ;
int WSPOLX[20],WSPOLY[20],PROMIEN,m,i,j,ILKAT,USTAW,k,SRX,SRY;
int CI,WX[20],WY[20];
long ROTATE,SUMA,MINIM,WYMIAR,A,B,C,KAT[20],KATY[20],KAT1;

    adres=0x0a00+(peekb(0x0504,0x00)-1)*0x100;
    k=0;
    ILKAT=0;
    j=0;
    i=0;

      do{
         BIT = peekb(0x4000,0x00);
         }
      while(( BIT & 1 ) == 0 );
      do{
         BIT = peekb(0x4000,0x00);
         }
      while(( BIT & 1 ) == 1 );
      BIT = peekb(0x8000,0x1000);
      KON = (peekw(0x506,0x00) * 4) + 4;
      for( i = 0; i < KON; i++ ){
         BIT = peekw(0x7ffd-2*i,0x1000);
         pokew(0x3700+2*i,0x00,~BIT);
      }
```

```
    do{
       BIT = peekb(0x4000,0x00);
       }
    while(( BIT & 1 ) == 1 );

    BIT = peekb(0x0C000,0x00);

   WYMIAR=KON-1;
   PROMIEN=peekb(adres+8,0x00);
   SRX=128;
   SRY=128;

 for(i=0;i<WYMIAR;i++)
   {
        A=(double)(peekb(0x3700+2*i,0x00)-SRX);
        B=(double)(peekb(0x3701+2*i,0x00)-SRY);
        C=A*A+B*B;
        C=sq2(C);
        CI = (int)C;
        ROZ= fabs(PROMIEN-CI);
        if(ROZ==0 || ROZ==1)
          {
             WSPOLX[ILKAT]=peekb(0x3700+2*i,0x00);
             WSPOLY[ILKAT]=peekb(0x3701+2*i,0x00);
             ILKAT++;
          }
   }

for(i = 0; i < ILKAT-1; i++ ){
  for( j = i+1; j < ILKAT; j++ ){
       if(((WSPOLX[i] - WSPOLX[j]) < 6 && (WSPOLX[i] - WSPOLX[j]) > -6)
         && ((WSPOLY[i] - WSPOLY[j]) < 6 && (WSPOLY[i] - WSPOLY[j]) > -6))
         {  WSPOLX[j] = WSPOLY[j] = 0; }
  }
}

j = 0;
for( i = 0; i < ILKAT; i++ ){
  if( WSPOLX[i] != 0 ){
       WX[j] = WSPOLX[i];
       WY[j] = WSPOLY[i];
       j++;
  }
}
ILKAT = j;
for( i = 0; i < ILKAT; i++ ){
  WSPOLX[i] = WX[i];
  WSPOLY[i] = WY[i];
}
pokeb(0x701,0x00,ILKAT);
for(i=0;i<ILKAT;i++){
  pokeb(0x800+2*i,0x00,WSPOLX[i]);
  pokeb(0x801+2*i,0x00,WSPOLY[i]);
}

for(i=0;i<ILKAT;i++)
 {
```

```
    if((WSPOLX[i]>=SRX) && (WSPOLY[i]<SRY))
     {
       A=WSPOLX[i]-SRX;
       B=SRY-WSPOLY[i];
       if(A>=B)
           KAT[i]=90 - tan(A,B);
       else
           KAT[i]=tan(B,A);
     }

   if((WSPOLX[i]<SRX) && (WSPOLY[i]<SRY))
     {
       A=SRX-WSPOLX[i];
       B=SRY-WSPOLY[i];
       if(A>=B)
           KAT[i]=90 + tan(A,B);
       else
           KAT[i]=180 - tan(B,A);
     }

   if((WSPOLX[i]<SRX) && (WSPOLY[i]>=SRY))
     {
       A=SRX-WSPOLX[i];
       B=WSPOLY[i]-SRY;
       if(A>=B)
           KAT[i]=270 - tan(A,B);
       else
           KAT[i]=180 + tan(B,A);
     }

   if((WSPOLX[i]>=SRX) && (WSPOLY[i]>=SRY))
     {
       A=WSPOLX[i]-SRX;
       B=WSPOLY[i]-SRY;
       if(A>=B)
           KAT[i]=270 + tan(A,B);
       else
           KAT[i]=360 - tan(B,A);
     }
   }

      /* ustawia kolejnosc katow od najm. do najw. */
   for(i=1;i<ILKAT;i++)
    {
    j=i;
    while ((KAT[j]<KAT[j-1]) & (j>=1))
         {
           KAT1=KAT[j];
           KAT[j]=KAT[j-1];
           KAT[j-1]=KAT1;
           j=j-1;
         }
    }

for( i = 0; i < ILKAT; i++ ){
   pokew(0x705+2*i,0x00,KAT[i]);
}
    pokew(0x703,0x00,ILKAT);
```

```
   /* do tab. KATY od KATY[10] wpisuje katy norm wzorca */
KAT1 = peekb(adres+10,0x00);
for( i = 0; i < ILKAT-1; i++ ){
    KATY[11+i] = KAT1 + peekb(adres+11+i);
    KAT1 = KATY[11+i];
}
KATY[10] = peekb(adres+10,0x00);

for( i = 0; i < ILKAT; i++ ){
   pokew(0x600+2*i,0x00,KATY[10+i]);
}

   /* w pam. od adres+10 katy rozw. wzorca */
KAT1 = peekb(adres+10,0x00) + (360 - peekb(adres+10+ILKAT,0x00));
pokeb(adres+10,0x00,KAT1);         /* obliczony pierwszy kat rozw.wzorca*/

   /* licz. katow rozw. detalu */
  for(i=ILKAT-1;i>=1;i=i-1)
   KATY[i]=KAT[i]-KAT[i-1];
  KATY[0] = 360 - (KAT[ILKAT-1] - KAT[0]);

for( i = 0; i < ILKAT; i++ ){
   pokew(0x610+2*i,0x00,KATY[i]);
}

    k=0;
    MINIM=1000;
    SUMA=0;

    for(m=0;m<ILKAT;m++)
      {
        SUMA=0;
        for(i=0;i<ILKAT;i++){
         KAT1 = fabs(KATY[i] - (long)(peekb(adres+10+i)));
         SUMA=SUMA+KAT1;
        }
        if(SUMA<MINIM)
           {
              MINIM=SUMA;
              USTAW=k;
           }
        k++;
        KAT1=KATY[ILKAT-1];
        for(i=ILKAT-1;i>=1;i=i-1)
          {KATY[i]=KATY[i-1];}
        KATY[0]=KAT1;
      }
    pokeb(0x700,0x00,USTAW);
    for(i=0;i<ILKAT;i++) /* katy detalu przed przes. do tabl. pom. KATY */
        KATY[i] = KAT[i]; /* mozna do KATY[0-ILKAT] bo katy rozw. juz wykorzyst */

    i=0;
    m=USTAW;
    for(i=0;i<ILKAT;i++)
        {
          KAT[m]=KATY[i];
          m++;
          if(m>ILKAT-1)
```

```
            a=0;
        }

  ROTATE=0;
  for(i=0;i<ILKAT;i++){   /* liczy kat obrotu dla wszystkich katow */
   KATY[i] =  KAT[i] - KATY[10+i];
  }
  k = 0;
  for(i=0;i<ILKAT;i++){
    if( KATY[i] < 0 ){
         k = 1;
         goto dalej;
    }
  }
dalej:  for(i=0;i<ILKAT;i++){
          if( k == 0 && KATY[i] > 180 )
             KATY[i] = 360 - KATY[i];
          if( k == 1 && KATY[i] > 180 )
             KATY[i] = KATY[i] - 360;
         }
    for( i = 0; i < ILKAT; i++ ){
         ROTATE = ROTATE + KATY[i];
    }
  OBROT=(int)(ROTATE/ILKAT);
  if( OBROT < 0 ){
      OBROT = (int)fabs(OBROT);
      pokeb(0x0e33,0x00,OBROT);
      pokeb(0x0e34,0x00,0);
  }
  if( OBROT >= 0){
     pokeb(0x0e33,0x00,OBROT);
     pokeb(0x0e34,0x00,1);
  }
  SUMA=peekb(0x0e31,0x00)+peekb(0x0e32,0x00)+peekb(0x0e33,0x00)+peekb(0x0e34,0x00);
  SK=256-(SUMA & 0xFF);
  pokeb(0x0e35,0x00,SK);
  pokeb(adres+10,0x00,KATY[10]);  /* odtworzenie pierwsz. kata wzorcow */
}
```

```
;=============================================================================
;
;                              TAN (X,Y)
;                                                                    7.06.89
;=============================================================================
; OBLICZANIE ARCTANG(X,Y)
;-----------------------------------------------------------------------------
;
MODEL equ 0
include lmacros.h

procdef tan, <<doub,dword>,<deab,dword>>

FINIT
FENI
FWAIT

mov bx,0e00h
mov ax,180
mov [bx],ax

FILD  DWORD PTR deab          ;Y na szczycie stosu kooprocesora,w ST(0)
FILD  DWORD PTR doub          ;X na szczycie w ST(0), Y w ST(1)
FPATAN                        ;arctan(X/Y) w ST(0)

FILD  WORD PTR [BX]           ;180 na szczycie w ST(0), arctan(X/Y) w ST(1)
FMULP ST(1),ST(0)             ;180*arctan(X/Y) na szczycie, w ST(0)

FLDPI                         ;3.14 na szczycie w ST(0)
FDIVP ST(1),ST(0)             ;180*arctan(X/Y)/3.14 na szczycie, w ST(0)

FISTP DWORD PTR [bx]          ;w komorce 0x0e00 umieszcza rezultat tj.
FWAIT                         ;arctan(X/Y) wyrazony w stopniach

mov ax,[bx]                   ;rezultat do AX
mov dx,[bx+2]

pret
pend tan
finish
```

```
;=============================================================================
;
;                              S Q 2 ( X )
;                                                                    19.05.89
;=============================================================================
; OBLICZANIE SQRT(X)
;-----------------------------------------------------------------------------
;
MODEL equ 0
include lmacros.h

procdef sq2, <<doub,dword>>

FINIT
FENI
FWAIT

mov bx,0e00h

FILD  DWORD PTR doub          ;X na szczycie stosu kooprocesora
FSQRT                         ;sqrt(X) na szczycie stosu kooprocesora

FISTP DWORD PTR [bx]          ;sqrt(X) do komorki pam. 0x0e00
FWAIT

mov ax,[bx]                   ;w AX wartosc sqrt(X)
mov dx,[bx+2]

pret
pend sq2
finish
```

Sposob konsolidacji programu mm16d.c i przesylania kodu do pakietu MM16.

```
cc mm16.c
ln -C 1000 -D 500 mm16.o sq2.o tan.o M87.LIB -lc
hex86 -b1000 -z -s8 mm16.exe
mode com1:48,e,7,2
copy mm16.hex com1
```

Sposob konsolidacji programu MM16 w celu otrzymania kodu dla PROM'u.

1) Opcja -c umieszcza segment kodu od adresu 0xf800
2) Opcja -d umieszcza segment danych od adresu 0x40
3) Procedura rom.o jest procedura startu (startup routine) i nalezy ja zmodyfikowac (patz nizej)
4) Poniewaz programowano dwa EPROMY typu 27128 (2*16 kB) zatem program hex86 nalezy wywolac z opcja -s32.

```
cc mm16.c
ln -c f800 -d 40 -o mm16.exe rom.o mm16.o sq2.o tan.o m87.lib -lc
hex86 -s32 mm16.exe
```

U W A G A !

Powstaly kod INTEL-HEX,ktory jest w pliku MM16.HEX nalezy zmodyfikowac w nastepujacy sposob:
od adresu 0x0068 do adresu 0x0071 wpisac bajty 0x90 (NOP).
(Zmienic sumy kontrolne na BA i 87)
Do programowania uzyto programatora SUNSHINE
a) stosujac program HEXBIN2.EXE zamieniono kod typu INTEL-HEX na kod binarny (CODE segment = 0000)
b) wybieramy EPROM typu 27128 o napieciu przepalania 21V
c) powstaly kod binarny ladujemy do bufora programatora
d) zmieniamy TARGET ZONE na od adr.0000 do adr.7fff (!)
e) programujemy oba EPROM-y stosujac opcje even i odd

ZAŁĄCZNIK NR 2.
PROGRAM S1.C

```
/*          PROGRAM S1.C (kompilator - TURBO C)                    */
/*                                                                 */
#include <stdio.h>
#include <graphics.h>
#include <dos.h>
#include <conio.h>
#include <math.h>
void ramka(void);
void krzyzyk(double,double);
void uczenie(void);
void pobrac(long *);
void wyslac(int,int *);
             int pobrac1(int *);
int T,WSPX,klawiat(void),WYS[10],f;
double X,Y,K,M,A,B,A1,B1;
long TAB[10],P;
char bufor[10];
union REGS inregs,outregs;
void main()

{
/*-----------------------------------------------------------------------*/
/*                                                                       */
/*       USTALENIE TRYBU GRAFICZNEGO MONITORA                            */
/*                                                                       */

   int driver=3, mode = 1;                    /* EGA, 600*350, 16 colors */
   initgraph( &driver, &mode, "c:\tc5" );
/*-----------------------------------------------------------------------*/
/*                                                                       */
/*      USTAWIENIE TRYBU PRACY PORTU SZEREGOWEGO COM1                    */
/*                                                                       */
    outportb(0x03fb,0x9f);            /* inicjalizacja COM1:8,E,2, LCR7=1 */
    outportb(0x03f8,0x18);            /* predkosc 4800 bodow              */
    outportb(0x03f9,0x00);

    outportb(0x03fc,0x03);            /* wyslanie do MCR DTR=1 i RTS=1    */
    outportb(0x03fb,0x1f);            /* LCR7=0                           */
/*-----------------------------------------------------------------------*/

                          TAB[0] = TAB[1] = TAB[2] = TAB[3] = 0;
 do
  {
   ramka();                           /* wyswietla ramke   */

   setcolor(BLACK);                   /* zamazuje napis    */
   outtextxy(6,339,"F2-NACT SENS  F3-POZ_A      ");

   setcolor(YELLOW);                  /* wyswietla napis   */
   outtextxy(6,339,"F1-ACT SENS  F5-QUIT");

   inregs.h.ah=0x00;
   T=0;
   do
     {
      int86(0x16,&inregs,&outregs);
      switch(outregs.h.ah)           /* Sprawdza, ktory klawisz dotknieto */
        {
```

```
            case 0x3b:T=1;                          /* Dotknieto F1-ACT SENS */
                    WYS[0]=0x3A; WYS[1]= 1; WYS[2]=1; WYS[3]=254;
                    wyslac(4,WYS);              /* Wysyla do MM16 przesylke
                                                o kodzie k=1 tj.ACT SENS */
                    break;
            case 0x3f:T=2;                              /* Dotknieto F5-QUIT */
                    break;
          }
      }
    while(T==0);  /* Jezeli dotknieto inny klawisz niz F1 lub F5 to powtarza */

    setcolor(BLACK);                            /* zamazuje napis             */
    outtextxy(6,339,"F1-ACT SENS  F5-QUIT");
    do
      {
        if(T==1)
          {
            setcolor(WHITE);                    /* wyswietla napis            */
            outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");
            do
              {
                WYS[0]=0x3A; WYS[1]=1; WYS[2]=2; WYS[3]=253;
                wyslac(4,WYS);              /*  wysyla przesylke 4-bajtowa */

                pobrac(TAB);                /* pobiera 4 bajtowa przesylke */

                X=TAB[2]+192;           /*  normalizacja wspolrzednych */
                Y=TAB[1]+32;

                setlinestyle( 0,0XFFF,1);   /*         linia ciagla, gruba */
                setcolor( BLACK );          /*            zamazuje krzyzyk */
                krzyzyk(A1,B1);
                A1=X;B1=Y;

                setcolor( WHITE );          /* Rysuje krzyzyk w miejscu srodka ciezkosci */
                krzyzyk(X,Y);

                sprintf(bufor,"%d",TAB[3]); /* Wynik rozpoznawania do bufora */
                if(TAB[4]!=P)
                  {
                    setcolor(BLACK);      /* Zamazuje poprzedni wynik rozpoznawania */
                    outtextxy(610,339,bufor);
                    setcolor(WHITE);
                  }
                P=TAB[3];
                outtextxy(610,339,bufor);   /* Wyswietla wynik rozpoznawania */
                f=klawiat();                /*         Sprawdza klawiature */
              }
            while(f==0);                    /* Chodzi w petli dopoki nie dotknieto klawiatury */

inregs.h.ah=0x00;
int86(0x16,&inregs,&outregs);
switch(outregs.h.ah)                    /* Sprawdza ktory klawisz dotknieto */
  {
    case 0x3c: setcolor(BLACK);         /* Jezeli dotknieto klawisz F2 to   */
               outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");
               krzyzyk(X,Y);            /* zamazuje napis i krzyzyk         */
```

```
            WYS[0]=0x3A; WYS[1]=1; WYS[2]=3; WYS[3]=252;
            wyslac(4,WYS);          /* Wysyla przesylke 4-bajtowa dezaktywujaca uklad sensoryczny */

            T=0;
            break;
case 0x3d:  setcolor(BLACK);        /* Jezeli dotknieto F3 to zamazuje napis i     */
            outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");

            WYS[0]=0x3A; WYS[1]=1; WYS[2]=4; WYS[3]=251;
            wyslac(4,WYS);          /* wysyla przesylke 4-bajtowa  POZ_A           */

            setcolor(YELLOW);       /* wyswietla napis                             */
            outtextxy(10,339,"POZ_A  -  POZYCJONOWANIE ROBOTA DO OBIEKTU");

            pobrac(TAB);            /* Odbiera 4-bajtowa przesylke                 */

            X=(TAB[1]*314/256); /* Normalizacja wspolrzednych X                */
            Y=(TAB[2]*633/256); /* Normalizacja wspolrzednych Y                */
            K=(Y-320)/(X-160);      /* Obliczanie wspolczynnikow prostej po jakiej */
            M=320-(K*160);          /* bedzie sie przesuwac krzyzyk                */
            WSPX=(X<160)?1:(-1);
            setcolor( WHITE );      /* Rysuje krzyzyk w miejscu srodka ciezkosci   */
            krzyzyk(Y,X);
            A=Y;B=X;

            do
               {
                   setcolor( BLACK );    /* Zamazuje poprzedni krzyzyk              */
                   krzyzyk(A,B);
                   X=X+WSPX;
                   Y=K*X+M;
                   A=Y;B=X;
                   setcolor( WHITE );    /* Rysuje krzyzyk w miejscu srodka ciezkosci */
                   krzyzyk(Y,X);
                   for(f=0;f<10000;f++); /* Opoznienie                              */
                }
            while(X!=160);

            setcolor(BLACK);            /* Zamazuje napis i krzyzyk          */
            outtextxy(10,339,"POZ_A  -  POZYCJONOWANIE ROBOTA DO OBIEKTU");
            krzyzyk(Y,X);
            setcolor(WHITE);
            setlinestyle(1,0XFFF,1);
            line(310,160,330,160);      /* Rysuje krzyzyk w srodku ekranu     */
            line(320,150,320,170);
            setlinestyle(0,0XFFF,1);

            T=1;
            break;
case 0x3e:  setcolor(BLACK);            /* Jezeli dotknieto klawisz F4 to     */
            outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");
            krzyzyk(A1,B1);             /* zamazuje srodek ciezkosci obiektu  */

            WYS[0]=0x3A; WYS[1]=1; WYS[2]=10; WYS[3]=245;
```

```
                wyslac(4,WYS);              /* Wysyla przesylke 4-bajtowa UCZENIE */

                uczenie();                  /* Procedura uczenia                  */

                T=1;
                break;
        case 0x3f: T=2;                     /* Jezeli dotknieto F5 to koniec programu - QUIT*/
                break;
        case 0x40: setcolor(BLACK);         /* Jezeli dotknieto F6 to zamazuje napis i */
                outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");
                WYS[0]=0x3A; WYS[1]=1; WYS[2]=11; WYS[3]=244;
                wyslac(4,WYS);              /* wysyla przesylke 4-bajtowa -OBROT  */
                pobrac(TAB);                /* Pobiera 2-bajty dotyczace obrotu   */
                setcolor(YELLOW);           /* Wyswietla :                        */
                if(TAB[1]==0)               /* jezeli kat obrotu rowny 0          */

                   outtextxy(6,339,"OBROT ROWNY - ");
                if((TAB[2]==0)&(TAB[1]!=0))  /* jezeli obrot w prawo tj. TAB[2]=0  */
                    {
                       outtextxy(6,339,"OBROT W PRAWO WYNOSI - ");
                       TAB[1]=257-TAB[1];
                     }
                if((TAB[2]==1)&(TAB[1]!=0))   /* jezeli obrot w lewo tj. TAB[2]=1  */
                    outtextxy(6,339,"OBROT W LEWO WYNOSI - ");

                sprintf(bufor,"%d",TAB[1]);
                outtextxy(200,339,bufor);

                while(!kbhit());            /* Czeka na dotkniecie klawiatury */

                setcolor(BLACK);            /* Zamazuje napisy */
                outtextxy(6,339,"OBROT W PRAWO WYNOSI - ");
                outtextxy(6,339,"OBROT W LEWO WYNOSI - ");
                outtextxy(6,339,"OBROT ROWNY - ");
                outtextxy(200,339,bufor);
                T=1;
                break;
              }
            }
          }
         while(T==1);

      }
   while(T==0);

   closegraph();        /* powraca z trybu graficznego w tryb tekstowy */
}




   void ramka()
     {
       setlinestyle( SOLID_LINE, 0XFFF, THICK_WIDTH);
       setcolor( YELLOW );                          /* rysuje ramke */
       line(0,0,0,350);
       line(0,350,640,350);
```

```
    line(640,350,640,0);
    line(640,0,0, 0);
    setlinestyle( SOLID_LINE, 0XFFF, NORM_WIDTH);
    setcolor( RED );
    line(500,335,500,348);
    line(2,320,638,320);
    line(2,335,638,335);
    setcolor ( WHITE );                          /* srodek ekranu */
                     line(192,32,448,32);
                     line(448,32,448,288);
                     line(448,288,192,288);
                     line(192,288,192,32);
    setlinestyle( 1,0XFFF,1);                    /* linia przerywana */
    line(310, 160, 330, 160);
    line(320, 150, 320, 170);
    setcolor(WHITE);
    outtextxy(510,339,"OBIEKT NR");
    setcolor( YELLOW );
    outtextxy(140,324,"Robot.c ver.1.0 1989 MERA-PIAP by OAE-6");
  }


void krzyzyk(double Y,double X)        /* Rysuje krzyzyk "+" o srodku w pkt.X,Y */
  {
    line((Y-5), X,(Y+5), X);
    line(Y, (X-5), Y,(X+5));
  }



 void wyslac(int l,int *tablica)
  {
    int dana[30],LSR,MSR,DSR,BUF,BLAD,m,smiec;

    for(m=0;m<l;m++)
       dana[m]=*(tablica+m);
                                              /*                              */
    outportb(0x03fc,0x03);                    /* Ustawienie w MCR bitu DTR=0 i RTS=0 */
    setcolor(GREEN);
    m=0;
    do
      {
        do
           {
              do
                 {
                    LSR=inportb(0x03fd);      /* Odczyt Rej. Stanu Transmisji LSR    */
                    BUF=LSR & 0x0020;
                    BLAD=LSR & 0x000e;
                    outtextxy(450,339,"O.K.");/* Wyswietla napis O.K. Bez tego moze nie dzialac */
                 }                            /* bo nie beda sie zgadzaly zaleznosci czasowe    */
              while ((BUF==0)||(BLAD!=0));    /* sprawdzenie bitu nr 5 BUFOR NADAJNIKA          */

              MSR=inportb(0x03fe);            /* Odczyt Rej. Stanu Modemu MSR        */

              DSR=MSR & 0x0020;               /* sprawdzenie bitu nr 5 DSR           */
              outtextxy(450,339,"O.K.");
           }
```

```
        while (DSR==0);

        setcolor(BLACK);                    /* zamazuje napis                    */
        outtextxy(450,339,"O.K.");
        setcolor(GREEN);                    /* wyswietla napis                   */
        outtextxy(450,339,"O.K.");
        outportb(0x03f8,dana[m]);           /* wyslanie danej na port COM1       */
        m=m+1;
     }
    while (m<l);
                                            /*                                   */
    outportb(0x03fc,0x02);                  /* Ustawienie w MCR bitu DTR=1       */

    while((inportb(0x03fd) & 0x01)==1)      /* Pobiera smieci z portu szeregowego jezeli sa */
       smiec=inportb(0x03f8);

    setcolor(BLACK);                        /* zamazuje napis                    */
    outtextxy(450,339,"O.K.");
  }

void pobrac(long *dana)
{
 int BUF,i,m,LICZBA,DANA,SK;
 long SUMA;
 void error(void);

 do
   {                                /*                                   */
     outportb(0x03fc,0x03);         /* COMMAND - ustawia RTS=0,DTR=0     */
     do
        BUF=inportb(0x03fd) & 0x0001;
     while(BUF==0);                 /*                                   */
     outportb(0x03fc,0x02);         /* COMMAND - ustawia RTS=0,DTR=1     */
   }
 while(inportb(0x03f8)!=0x3A);      /* Czeka na poczatek przesylki tzn. na : */

 do
   {                                /*                                   */
     outportb(0x03fc,0x03);         /* COMMAND - ustawia RTS=0,DTR=0     */
     do
       BUF=inportb(0x03fd) & 0x0001;
     while(BUF==0);                 /*                                   */
     outportb(0x03fc,0x02);         /* COMMAND - ustawia RTS=0,DTR=1     */
     LICZBA=inportb(0x03f8);
   }
 while(LICZBA==0x3A);               /* Po znaku : powinien przyjsc bajt  */
                                    /* oznaczajacy liczbe przesylanych bajtow */
 i=0;
 SUMA=LICZBA;
 m=0;

 do
   {                                /*                                   */
     outportb(0x03fc,0x03);         /* COMMAND - ustawia RTS=0,DTR=0     */
```

```
        do
          BUF=inportb(0x03fd) & 0x0001;
        while(BUF==0);                  /*                                  */
        outportb(0x03fc,0x02);          /* COMMAND - ustawia RTS=0,DTR=1    */
        *(dana+m)=inportb(0x03f8);
        if(*(dana+m)==0x3A)
           error();
        SUMA=SUMA+(*(dana+m));
        m++;
        i++;
      }
    while(i<LICZBA);

    outportb(0x03fc,0x03);              /* COMMAND - ustawia RTS=0,DTR=0    */
    do
      BUF=inportb(0x03fd) & 0x0001;
    while(BUF==0);                      /*                                  */
    outportb(0x03fc,0x02);              /* COMMAND - ustawia RTS=0,DTR=1    */
    SK=inportb(0x03f8);
    if(((SUMA+SK) & 0xFF)!=0)
        error();
  }


  void error(void)
    {
       printf("BLAD TRANSMISJI\n");
    }



  pobrac1(int *dana)
     {
                        int z,BUF,fl;
                        z = 0;
     fl = 0x0ff;

       outportb(0x03fc,0x03);               /*Ustawienie w MCR bitu DTR=1 i RTS=1                    */

loop:    do
             BUF=inportb(0x03fd) & 0x0001;  /* Buf. Odb. jest bitem nr 0 w Rej. Stanu Transmisji LSR */
          while (BUF==0);                   /*Sprawdzenie bitu nr 0 Bufor Odbiornika                 */

          outportb(0x03fc,0x02);            /*Ustawienie w MCR bitu DTR=0                            */
          *(dana+z)= inportb(0x03f8);        /*Pobranie danej z bufora                                */
          outportb(0x03fc,0x03);            /*Ustawienie w MCR bitu DTR=1 i RTS=1                    */

                              if( (*(dana+z)!=0) || (fl != 0) ){
                                    fl = *(dana+z);
                                    z++;
                                    goto loop;
                              }

       outportb(0x03fc,0x02);               /*Ustawienie w MCR bitu DTR=0                            */
                         return( z );
     }

   /* Funkcja sprawdza czy byla dotknieta klawiatura tzn.czy jest cos w buforze klawiatury.
```

```
/  Jezeli nie byla dotknieta, to funkcja zwraca wartosc 0. W przeciwnym razie
/  zwraca wartosc 0ffh.                                                        */

int klawiat(void)
{
  union REGS inregs,outregs;
  inregs.h.ah=0x0b;
  intdos(&inregs,&outregs);
  return(outregs.h.al);

}
```

```
/* Procedura rysuje okrag o srodku w pkt.320,160, promieniu=promien i w kolorze=kolor */

void okrag(int promien,int kolor)
  {
    int x,y,a;

    for(x=320-promien;x<=320+promien;x++)
      {
          a=abs(promien*promien-(x-320)*(x-320));
          y=sqrt(a)+160;
          putpixel(x,y,kolor);
          y=160-sqrt(a);
          putpixel(x,y,kolor);
      }
  }

void uczenie()
  {
    int T,i,j,m,KONTY[1000],KONTX[1000],WYS[10],PROM,KATINT[50],WSPOLX[50],WSPOLY[50];
                      int ROZMIAR,SUM,DIF,WX[100],WY[100];
    double A,B,SUMA,KX,KY,ROZ1,RADIUS,ODLEG,SINUS,KAT[50],KAT1;
    long SK;
    char p;
    void wyslac(int,int *);
    void okrag(int,int);
    int pobrac1(int *);

    /* W tym miejscu nastepuje pobranie konturu do tab. KONTX i KONTY.
       Jest to przydatne gdy chcemy sprawdzic poprawnosc procedury
       "uczenie" bez wnikania w wartosci punktow konturu przychodzace
       z ukladu wizyjnego. Ponizej wczytywane sa przykladowe wspolrzedne
       prostokata.
                        */

    setcolor(YELLOW);                   /* wyswietla napis                    */
    outtextxy(10,339,"WAIT !");

    ROZMIAR = pobrac1(KONTX);                   /* odczytuje z COM1 wspolrzedne X konturu i zapisuje je w tab. KONT
```

```
ROZMIAR = pobrac1(KONTY);                    /* odczytuje z COM1 wspolrzedne Y konturu i zapisuje je w tab. KT

                                    /* normalizacja wspolrzednych konturu,taka zeby srodek obiektu      */
                                         /* znajdowal sie w srodku pola widzenia kamery                 */
       for( i = 0; i < ROZMIAR; i++ ){
      KONTX[i]=KONTX[i]+192;
      KONTY[i]=KONTY[i]+32;
   }

setcolor(BLACK);                     /* zamazuje napis                                                  */
outtextxy(10,339,"WAIT !");


for(i=0;i<ROZMIAR;i++)               /* rysowanie konturu ,kolor bialy                                  */
                        {
   putpixel(KONTX[i],KONTY[i],15);
}

do
  {
    setcolor(BLACK);                 /* zamazuje napisy                                                 */
    outtextxy(6,339,"F2-NACT SENS  F3-POZ_A  F4-UCZENIE  F5-QUIT  F6-OBROT");
    outtextxy(10,339,"F1 - ACCEPT   F2 - CLEAR");
    sprintf(bufor,"%d",PROM);
    outtextxy(300,339,"PROMIEN ROWNY  ");
    outtextxy(430,339,bufor);

    setcolor(YELLOW);                /* wyswietla napis                                                 */
    outtextxy(10,339,"UCZENIE  PODAJ PROMIEN");
    setlinestyle(0,0xfff,1);
    outtextxy(200,339,"(TRZYCYFROWY)");

    do                               /* wczytanie promienia z klawiatury (instrukcja scanf()            */
      {                              /* nie nadaje sie, bo drukuje wartosc promienia w trybie tekstowym) */
         p=getch();                  /* Promien wczytywany musi byc trzycyfrowy                         */
         PROM=(p-48)*100;
         p=getch();
         PROM=PROM+(p-48)*10;
         p=getch();
         PROM=PROM+(p-48);
      }
    while(PROM>=200);

    cleardevice();                   /* kasuje ekran                                                    */
    ramka();                         /* wyswietla ramke                                                 */

    for(i=0;i<ROZMIAR;i++)           /* rysowanie konturu ,kolor bialy                                  */
       putpixel(KONTX[i],KONTY[i],15);

    okrag(PROM,14);                  /* procedura rysowania okregu,kolor zolty                          */

/* znajdowanie punktow wspolnych okregu o srodku pokrywajacym sie ze srodkiem obiektu i promieniu PROM */
```

```
/* z konturem obiektu. Obliczone wspolrzedne wspisywane sa do tablic WSPOLX i WSPOLY                    */
/* Ilosc punktow wspolnych okresla parametr m.                                                          */

m=0;
for(i=0;i<ROZMIAR;i++)
  {
                          KX = (double)(KONTX[i] - 320);
                          KY = (double)(KONTY[i] - 160);

     SUMA=sqrt(KX*KX + KY*KY);
     SUM=(int)SUMA;
     DIF=abs(SUM-PROM);
     if(DIF==0 || DIF == 1)
       {
         WSPOLX[m]=KONTX[i];
         WSPOLY[m]=KONTY[i];
                                  m++;
       }
  }
for( i = 0; i < m-1; i++ ){
                        for( j = i+1; j < m; j++ ){
                              if( abs(WSPOLX[i] - WSPOLX[j]) < 5 && abs(WSPOLY[i] -WSPOLY[j]) < 5 )
                                    WSPOLX[j] = WSPOLY[j] = 0;
                        }
                  }
                  j = 0;
                  for( i = 0; i < m; i++ ){
                        if( WSPOLX[i] != 0 ){
                              WX[j] = WSPOLX[i];
                              WY[j] = WSPOLY[i];
                              j++;
   }
                  }
                  m = j;
                  for( i = 0; i < m; i++){
                        WSPOLX[i] = WX[i];
                        WSPOLY[i] = WY[i];
                  }
for(i=0;i<m;i++){                       /* rysowanie promieni tj.linii laczacych srodek z punktami wspolnymi*/
   line(WSPOLX[i],WSPOLY[i],320,160);
              }

setcolor(BLACK);                        /* zamazuje napis                                               */
outtextxy(10,339,"UCZENIE  PODAJ PROMIEN");

setcolor(YELLOW);                       /* wyswietla napis                                              */
outtextxy(10,339,"F1 - ACCEPT    F2 - CLEAR");
sprintf(bufor,"%d",PROM);
outtextxy(300,339,"PROMIEN ROWNY  ");
outtextxy(430,339,bufor);

T=0;

do
  {
    inregs.h.ah=0x00;                   /* Sprawdza czy dotknieto klawisz F1 (ACCEPT) czy F2 (CLEAR)    */
    int86(0x16,&inregs,&outregs);
    switch(outregs.h.ah)
      {
```

```
            case 0x3b: T=1;                 /* Dotknieto F1                                    */
                       break;
            case 0x3c: T=2;                 /* Dotknieto F2                                    */
                       break;
          }
      }
    while(T==0);
}
while(T==2);

setcolor(BLACK);                            /* zamazuje napisy                                 */
outtextxy(10,339,"F1 - ACCEPT    F2 - CLEAR");

outtextxy(300,339,"PROMIEN ROWNY  ");
outtextxy(430,339,bufor);


/* obliczanie katow pomiedzy promieniami a polprosta OX i wpisanie ich wartosci do tablicy KAT  */

for(i=0;i<m;i++)
  {
    if((WSPOLX[i]>=320) && (WSPOLY[i]<160))   /* I cwiartka ukladu wspolrzednych                 */
      {
        A=WSPOLX[i]-320;
        B=160-WSPOLY[i];
        if(A<B)
          KAT[i]=atan(B/A);
        else
          KAT[i]=1.57-atan(A/B);
      }

    if((WSPOLX[i]<320) && (WSPOLY[i]<160))    /* II cwiartka ukladu wspolrzednych                */
      {
        A=320-WSPOLX[i];
        B=160-WSPOLY[i];
        if(A<B)
          KAT[i]=3.14-atan(B/A);
        else
          KAT[i]=1.57+atan(A/B);
      }

    if((WSPOLX[i]<320) && (WSPOLY[i]>=160))   /* III cwiartka ukladu wspolrzednych               */
      {
        A=320-WSPOLX[i];
        B=WSPOLY[i]-160;
        if(A<B)
          KAT[i]=3.14+atan(B/A);
        else
          KAT[i]=4.71-atan(A/B);
      }

    if((WSPOLX[i]>=320) && (WSPOLY[i]>=160))  /* IV cwiartka ukladu wspolrzednych                */
      {
        A=WSPOLX[i]-320;
        B=WSPOLY[i]-160;
        if(A<B)
          KAT[i]=6.28-atan(B/A);
        else
```

ZAŁĄCZNIK NR 3.
PROGRAM PROG.C

```
; Program prog.c inicjalizuje liczniki i czeka na przerwanie. ;

#include <stdio.h>
#include <io51.h>

void main()
{

        write_XDATA( 0x5003, 0x38 );   /* initializ. licznikow 8253 */
        write_XDATA( 0x5003, 0x78 );
        write_XDATA( 0x5003, 0x0c8 );
        write_XDATA( 0x6003, 0x38 );
        write_XDATA( 0x6003, 0x78 );
        write_XDATA( 0x6003, 0x0c8 );
        write_XDATA( 0x7003, 0x38 );
        write_XDATA( 0x7003, 0x78 );

        output( 0x0a8, 1 );    /* 1 do rejestru IE, enable interrupt */
loop:  ;
       goto loop;
}
```

```
; Program hist.c sluzy do obliczania histogramu.  ;

#include <stdio.h>
#include <io51.h>

void hist( void )
{
unsigned int p[10],k,l,c,n,i;
       clear_bit( P1_0_bit );      /* zeruje przerwanie */
       output( 0x0a8, 0 );    /* 0 do rejestru IE, disable interrupt */

       p[1] = 0xffff - ( read_XDATA( 0x5000 ) + ( read_XDATA ( 0x5000 ) * 256 ));
       p[2] = 0xffff - ( read_XDATA( 0x5001 ) + ( read_XDATA ( 0x5001 ) * 256 ));
       p[3] = 0xffff - ( read_XDATA( 0x5002 ) + ( read_XDATA ( 0x5002 ) * 256 ));
       p[4] = 0xffff - ( read_XDATA( 0x6000 ) + ( read_XDATA ( 0x6000 ) * 256 ));
       p[5] = 0xffff - ( read_XDATA( 0x6001 ) + ( read_XDATA ( 0x6001 ) * 256 ));
       p[6] = 0xffff - ( read_XDATA( 0x6002 ) + ( read_XDATA ( 0x6002 ) * 256 ));
       p[7] = 0xffff - ( read_XDATA( 0x7000 ) + ( read_XDATA ( 0x7000 ) * 256 ));
       p[8] = 0xffff - ( read_XDATA( 0x7001 ) + ( read_XDATA ( 0x7001 ) * 256 ));

       write_XDATA( 0x5000, 0xff );   /* pisz wartosci poczatkowe do licznikow */
       write_XDATA( 0x5000, 0xff );
       write_XDATA( 0x5001, 0xff );
       write_XDATA( 0x5001, 0xff );
       write_XDATA( 0x5002, 0xff );
       write_XDATA( 0x5002, 0xff );
       write_XDATA( 0x6000, 0xff );
       write_XDATA( 0x6000, 0xff );
       write_XDATA( 0x6001, 0xff );
       write_XDATA( 0x6001, 0xff );
       write_XDATA( 0x6002, 0xff );
       write_XDATA( 0x6002, 0xff );
       write_XDATA( 0x7000, 0xff );
       write_XDATA( 0x7000, 0xff );
       write_XDATA( 0x7001, 0xff );
       write_XDATA( 0x7001, 0xff );

      k = l = c = 0;
      for( i = 1; i < 9; i++ ){
             if( p[i] > c ){
                    c = p[i];
                    k = i;
             }
      }
      l = c = p[k] = 0;
      for( i = 1; i < 9; i++ ){
             if( p[i] > c ){
                    c = p[i];
                    l = i;
             }
      }
      if( k < l ){
             for( i = k+1; i < l; i++ ){
                    if( p[i] < c ){
                           c = p[i];
                           n = i;
                    }
             }
      }
```

```
        if( k > l ){
                for( i = l+1; i < k; i++ ){
                        if( p[i] < c ){
                                c = p[i];
                                n = i;      /* n - prog komparacji */
                        }
                }
        }

        set_bit( P1_1_bit );              /* zezwolenie na zapis progu do 7475 */
        write_XDATA( 0x8000, n );     /* pisze nowy prog komp. do 7475 */
        clear_bit( P1_1_bit );
        output( 0x0a8, 1 );           /* 1 do rejestru IE, enable interrupt */
}
```

ZAŁĄCZNIK NR 5.

PROGRAM CSTARTUP.S03

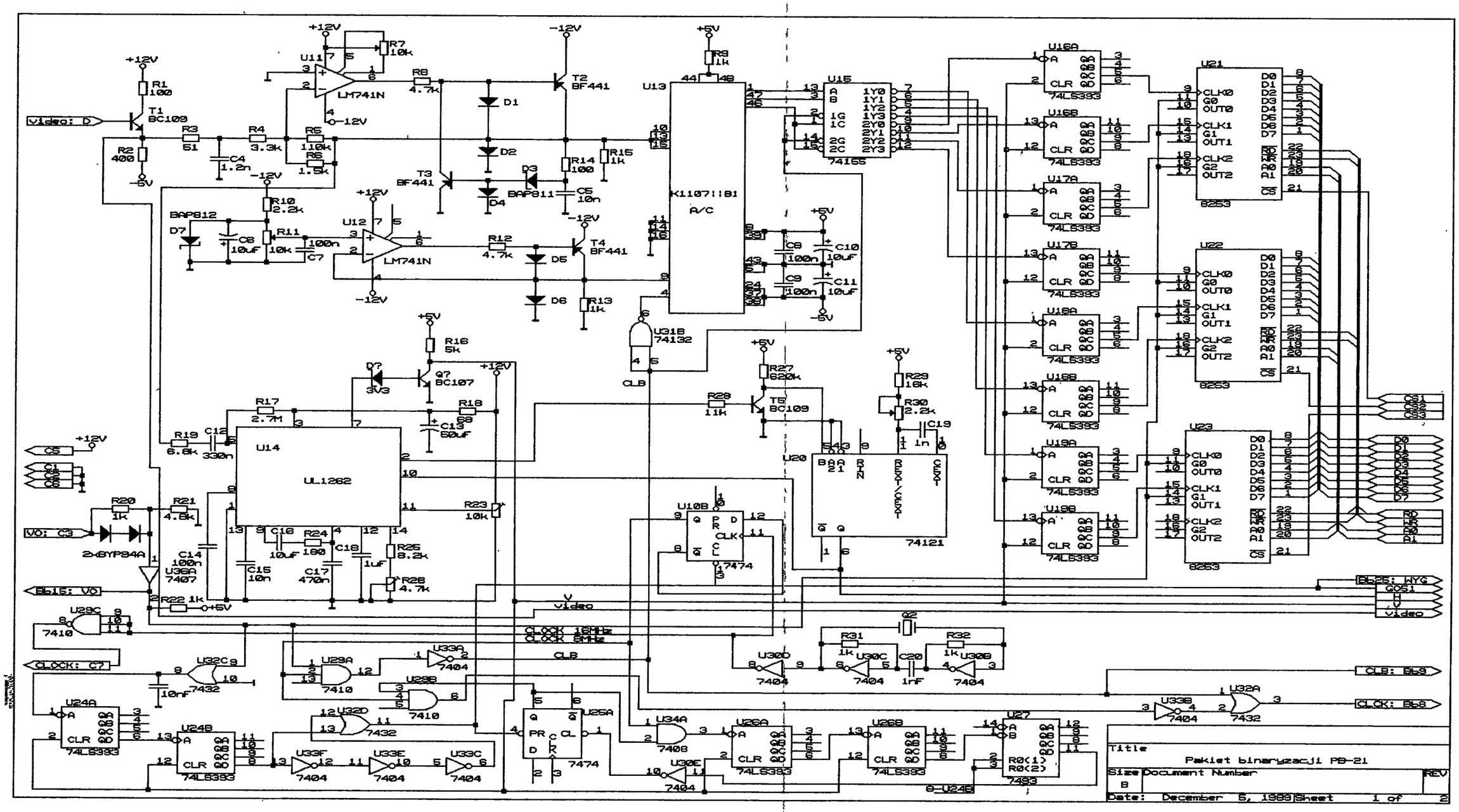

```
-!                -LNK8051.XCL-

        XLINK command file to be used with the 8051 C-compiler V2.xx
        using the -m0 to -m3 options (expanded)
        Usage:  xlink  your_file(s)  -f lnk8051

        First: define CPU  -!

-c8051
-l cros
-x

-!  Assign/allocate a value for SP (change if not reg-bank #0) -!
-Z(IDATA)ISTACK=7

-!  Select register bank [0,8,10 or 18] -!
-D_R=0

-!  Setup all read-only segments (PROM). Usually at zero -!
-Z(CODE)CSTART,RCODE,CODE,CDATA,ZVECT,CONST,CSTR,CCSTR=0

-!  Setup all writable segments which must be mapped to external RAM. -!
-Z(XDATA)DATA,TEMP,IDATA,UDATA,ECSTR,WCSTR,XSTACK=4000

-A prog,hist

-!  Load the 'C' library -!
cl8051

-o aout.hex
-z
```

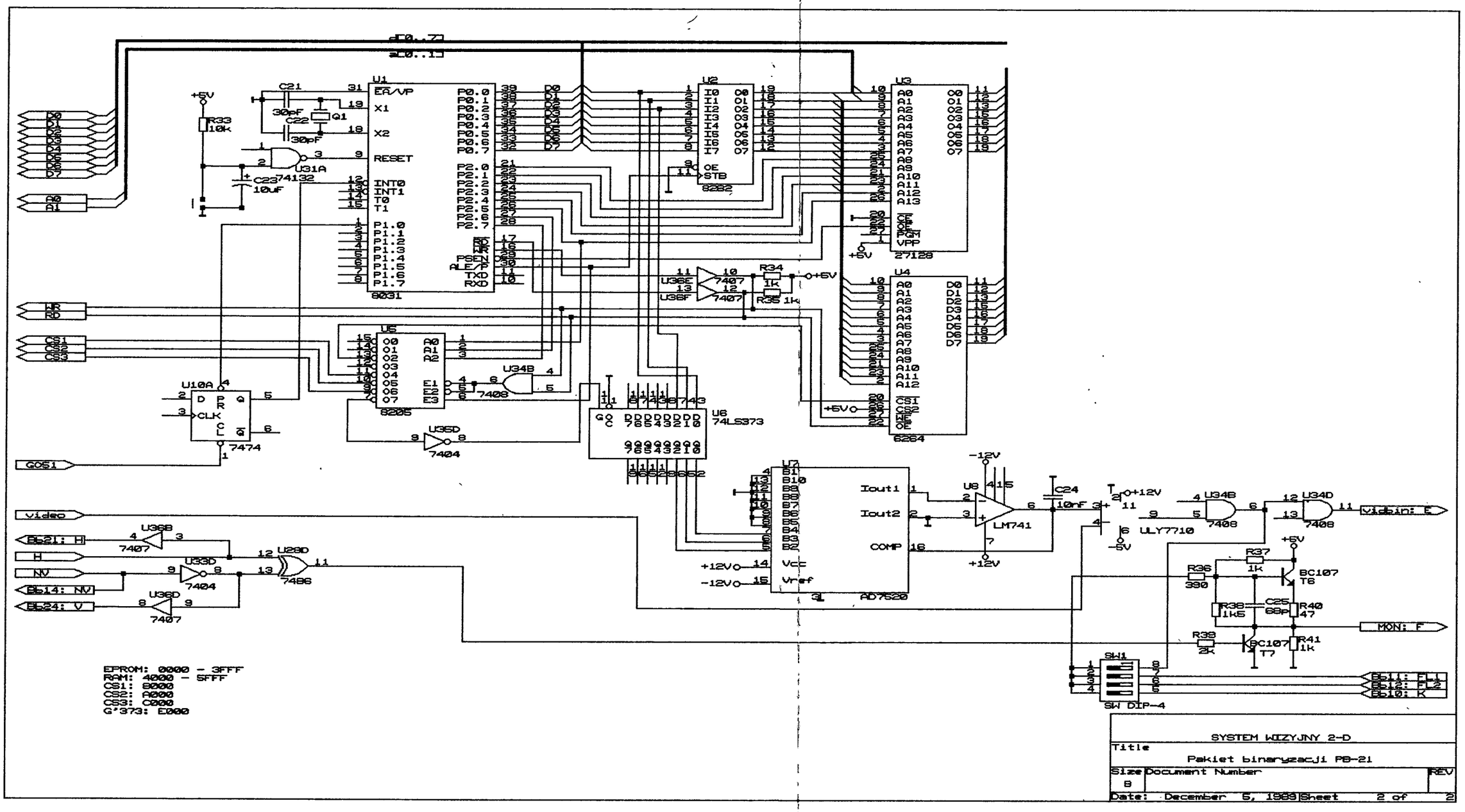
```
-!                -LNK8051.XCL-

        XLINK command file to be used with the 8051 C-compiler V2.xx
        using the -m0 to -m3 options (expanded)
        Usage:  xlink  your_file(s)  -f lnk8051

        First: define CPU  -!

-c8051
-l cros
-x

-!  Assign/allocate a value for SP (change if not reg-bank #0) -!
-Z(IDATA)ISTACK=7

-!  Select register bank [0,8,10 or 18] -!
-D_R=0

-!  Setup all read-only segments (PROM). Usually at zero -!
-Z(CODE)CSTART,RCODE,CODE,CDATA,ZVECT,CONST,CSTR,CCSTR=0

-!  Setup all writable segments which must be mapped to external RAM. -!
-Z(XDATA)DATA,TEMP,IDATA,UDATA,ECSTR,WCSTR,XSTACK=4000

-A prog,hist

-!  Load the 'C' library -!
cl8051

-o aout.hex
-z
```

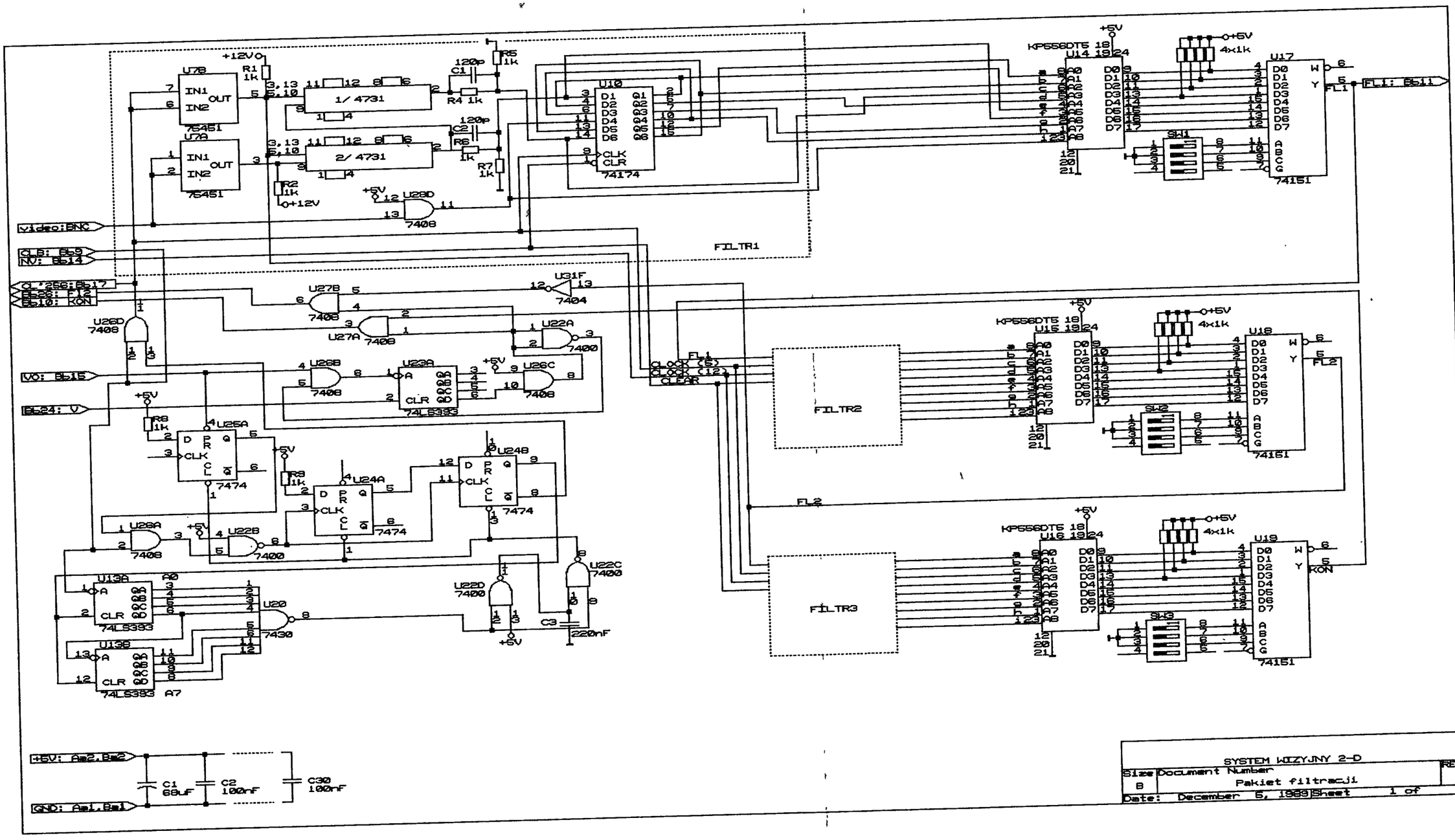
```
;------------------------------------------------;
;                                                ;
;                 CSTARTUP.S03                   ;
;                                                ;
; This module contains the 8051 C startup-routine and ;
; must usually be tailored to suit customer hardware. ;
; Version: 2.00 [A.R. 10/Jun/87]                 ;
; Update :  1988.05.02                           ;
;          by:  Andrzej Zasucha                  ;
;                                                ;
;           Przemyslowy Instytut Automatyki      ;
;                i  Pomiarow                     ;
;                 MERA-PIAP                      ;
;                                                ;
;------------------------------------------------;
        NAME    CSTARTUP

$DEFMM.INC              ; define memory model

        PUBLIC  exit
        EXTERN  _R      ; Register bank (0, 8, 16 or 24)
        EXTERN  main
        EXTERN  hist

;-------------------------------------------------------------;
; Virtual stack segment. It should be mapped into external RAM ;
;-------------------------------------------------------------;
      IF      single_chip = 0

        RSEG    XSTACK
        DS      356             ; Change if needed
xstack_end:
        ENDIF

;-------------------------------------------------------------;
; Internal stack used for LCALL's and temporary storage for   ;
; code generator help-routines (math etc).  Stack can be loca- ;
; ted anywhere in the internal RAM with the exception of      ;
; _R - _R+7 that are reserved for R0-R7. Note that C interrupt ;
; routines require that you increase stack by 25 bytes to     ;
; assure that no overflows occur.                             ;
;-------------------------------------------------------------;
        RSEG    ISTACK
stack_begin:
        IF      single_chip
        DS      150
        ELSE
        DS      120
        ENDIF

        RSEG    CSTART
startup:                        ; Should be at location zero
        LJMP    init_C
        DS      16              ; Space for 8051 vectors

;-------------------------------------------------------------;
; This is the place to insert interrupt vectors/handlers in.  ;
; For locations consult the Intel Microcontroller Handbook.   ;
```

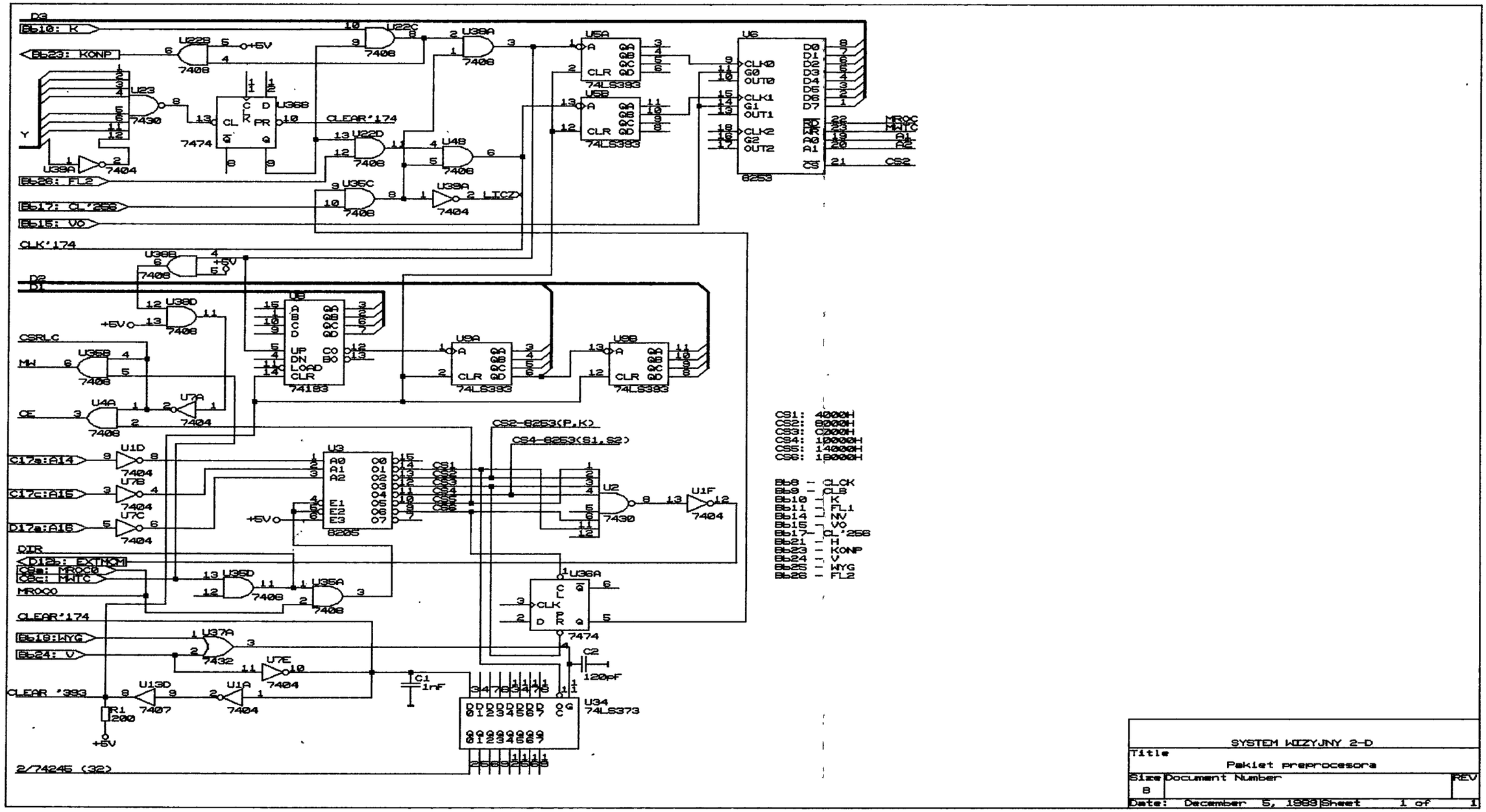
```
;------------------------------------------------------------;
;      CLR    IE.7
       ORG    startup+3
       LJMP   handl

handl:
       PUSH   ACC
       PUSH   B
       PUSH   PSW
       PUSH   DPH
       PUSH   DPL
       PUSH   _R+0
       PUSH   _R+1
       PUSH   _R+2
       PUSH   _R+3
       PUSH   _R+4
       PUSH   _R+5
       MOV    DPTR, #0
       LCALL  hist         ; obsluga przerwania
       POP    _R+5
       POP    _R+4
       POP    _R+3
       POP    _R+2
       POP    _R+1
       POP    _R+0
       POP    DPL
       POP    DPH
       POP    PSW
       POP    B
       POP    ACC
       SETB   IE.7
       RETI
init_C:
;------------------------------------------------------------;
; Ativate the (at link-time) selected register bank.         ;
;------------------------------------------------------------;

       MOV     A,#_R
       MOV     C,ACC.3
       MOV     PSW.3,C
       MOV     C,ACC.4
       MOV     PSW.4,C

       MOV     SP,#stack_begin ; From low to high addresses
;------------------------------------------------------------;
; If you don't want global/static C variables to be initial- ;
; ized at startup you can just remove the next two lines.    ;
; NOTE: Never remove the call in the large models (-m0 or -m1) ;
;------------------------------------------------------------;

       EXTERN  ?SE6_INIT_L17
       LCALL   ?SE6_INIT_L17   ; Initialize segments

;------------------------------------------------------------;
```

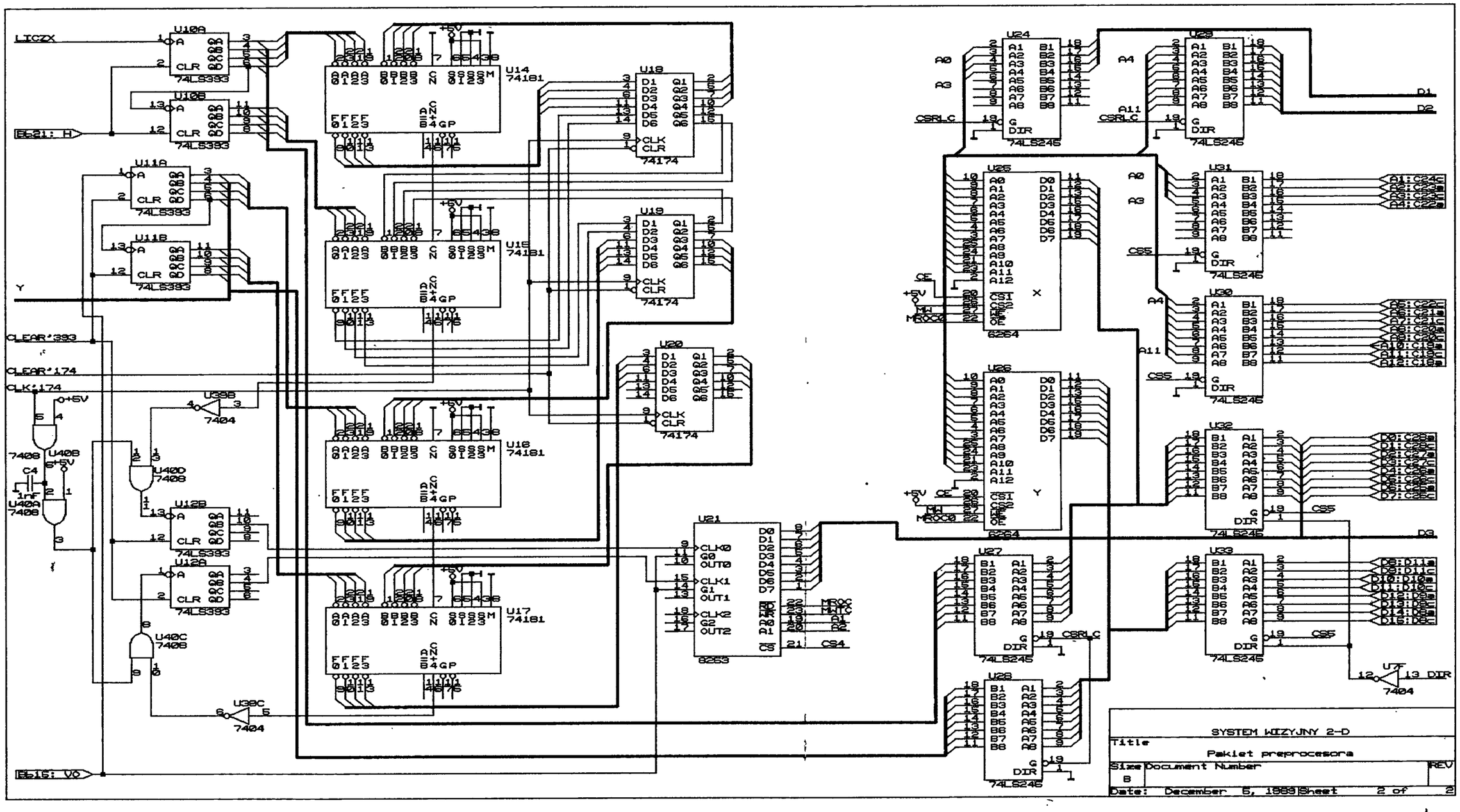
```
; If hardware must be initiated from assembly or if interrupts ;
; should be on when reaching main, this is the place to insert ;
; such code.                                                   ;
;-------------------------------------------------------------;

        IF      single_chip

        MOV     R1,#0
        LCALL   main            ; main()
exit:                           ; someone called exit()

        ELSE

        MOV     R6,#HIGH(xstack_end)
        MOV     R7,#LOW(xstack_end)

        IF      banked_mode

        EXTERN  ?X_CALL_L18

        MOV     DPTR,#main
        LCALL   ?X_CALL_L18             ; main()
        DW      0                       ; No parameter space
local_exit:                             ; someone called exit()

        ELSE

        MOV     DPTR,#0         ; No parameters

        LCALL   main            ; main()
exit:                           ; someone called exit()

        ENDIF

        ENDIF

;-------------------------------------------------------------;
; Now when we are ready with our C program (usually 8051 C    ;
; programs are continouous) we must perform a system-dependent ;
; action.  In this simple case we just stop.                  ;
;-------------------------------------------------------------;

        SJMP    $               ; Forever...

        IF      banked_mode

        RSEG    FLIST
exit:
        DD      local_exit

        ENDIF

        END     startup
```


+12V
-12V
+5V
R1 100
T1 BC109
video: D
R2 400
R3 51
R4 3.3k
R5 110k
R6 1.5k
C4 1.2n
U11 LM741N
R7 10k
R8 4.7k
T2 BF441
T3 BF441
D1
D2
D3 BAP811
D4
C5 10n
R14 100
R15 1k
U13 K1107IB1 A/C
R9 1k
U15 74155
U16A U16B U17A U17B U18A U18B U19A U19B 74LS393
U21 U22 U23 8253
D0 D1 D2 D3 D4 D5 D6 D7
CLK0 GATE0 OUT0 CLK1 GATE1 OUT1 CLK2 GATE2 OUT2 CS A0 A1 RD WR
CS1 CS2 CS3
R10 2.2k
BAP812
D7
C6 10uF
R11 10k
C7 100n
U12 LM741N
R12 4.7k
T4 BF441
D5
D6
R13 1k
C8 100n
C10 10uF
C9 100n
C11 10uF
U31B 74132
CLB
R16 5k
D? 3V3
Q? BC107
R17 2.7M
R18 68
C13 60uF
R27 620k
T5 BC109
R29 11k
R28 16k
R30 2.2k
C19 1n
U20 74121
R19 6.8k
C12 330n
U14 UL1262
C5
C1 C2 C3 C4
V0: C3
R20 1k
R21 4.8k
2xBYP94A
C14 100n
C16 10uF
R24 180
C18 1uF
R25 8.2k
C15 10n
C17 470n
R28 4.7k
R22 1k
Bb15: V0
U10B 7474
R23 10k
V video
CLOCK 16MHz
CLOCK 8MHz
CLB
Q2
R31 1k
R32 1k
U30C 7404
C20 1nF
U30B 7404
U30D 7404
U29C 7410
U32C 7432
10nF
CLOCK: C7
U24A U24B 74LS393
U29A 7410
U33A 7404
U29B 7410
U32D 7432
U33F U33E U33C 7404
U26A 7474
U34A 7408
U30E 7404
U26A U26B 74LS393
U27 7493
R0(1) R0(2)
U33B 7404
U32A 7432
CLB: Bb9
CLOCK: Bb8
Title
Pakiet binaryzacji PB-21
Size B
Document Number
REV
Date: December 5, 1989
Sheet 1 of 2


D[0..7]
A[0..15]
U1
8031
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
WR
RD
PSEN
ALE/P
TXD
RXD
C21
30pF
C22
30pF
Q1
R33
10k
U31A
74132
C23
10uF
+5V
U2
8282
I0
I1
I2
I3
I4
I5
I6
I7
O0
O1
O2
O3
O4
O5
O6
O7
OE
STB
U3
27128
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
VPP
U4
6264
CS1
CS2
WE
OE
U5
8205
E1
E2
E3
U10A
7474
U34B
7408
U35D
7404
U6
74LS373
U7
AD7520
Iout1
Iout2
COMP
Vcc
Vref
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
U8
LM741
C24
10nF
+12V
-12V
-5V
ULY7710
U34D
U36B
7407
U33D
7404
U28D
7486
U36D
7407
U36E
U36F
R34
1K
R35 1K
R37
1k
R36
390
R38
1k5
C25
68p
R40
47
R39
2K
R41
1k
BC107
T6
BC107
T7
SW1
SW DIP-4
video
BL21: H
H
NV
BL14: NV
BL24: V
GOSI
Vidbin: E
MON: F
BL11: FL1
BL12: FL2
BL10: K
EPROM: 0000 - 3FFF
RAM: 4000 - 5FFF
CS1: 8000
CS2: A000
CS3: C000
G'373: E000
SYSTEM WIZYJNY 2-D
Title
Pakiet binaryzacji PB-21
Size
B
Document Number
REV
Date: December 5, 1989
Sheet 2 of 2


+12V
R1 1k
U7B
7
6
IN1
IN2
OUT
5
75451
U7A
1
2
IN1
IN2
OUT
3
75451
3,13 11 12 8 6
5,10
1/ 4731
9
1
4
2
3,13 11 12 8 6
5,10
2/ 4731
9
1
4
2
120p
C1
R4 1k
R5 1k
120p
C2
R6 1k
R7 1k
R2 1k
0+12V
+5V
12
U28D
13
7408
11
video:BNC
CLB: B5,9
NV: B6,14
U10
D1 Q1
D2 Q2
D3 Q3
D4 Q4
D5 Q5
D6 Q6
CLK
CLR
74174
FILTR1
KP556DT5 18
U14 19 24
+5V
0+5V
4x1k
U17
D0 W
D1 Y
D2
D3
D4
D5
D6
D7
A
B
C
G
74151
FL1
FL1: B5,11
SW1
Q1*256:B5,17
B5,25: F12
B5,10: KON
U27B
6
5
7408
4
U27A
3
7408
2
U31F
12
13
7404
U22A
1
2
3
7400
U26D
7408
1
2
3
VO: B5,15
B5,24: V
U28B
4
5
6
7408
U23A
1
A
CLR
2
74LS393
3
4
5
6
+5V
U26C
9
10
8
7408
R8 1k
U25A
4
2
D
PR
CLK
CL
3
5
6
1
7474
+5V
R9 1k
U24A
4
2
D
PR
CLK
CL
3
5
6
1
7474
U24B
12
11
D
PR
CLK
CL
13
9
8
1
7474
U28A
1
2
3
7408
+5V
U22B
4
5
6
7400
U13A
A0
1
A
CLR
2
74LS393
3
4
5
6
U20
1
2
3
4
5
6
7430
8
U13B
13
A
CLR
12
74LS393
A7
11
10
9
8
U22D
7400
U22C
8
9
10
7400
C3
220nF
+5V
FL1
CLEAR
FILTR2
FILTR3
FL2
KP556DT5 18
U15 19 24
U18
SW2
74151
KP556DT5 18
U16 19 24
U19
SW3
74151
KON
+5V: A=2,B=2
GND: A=1,B=1
C1
68uF
C2
100nF
C30
100nF
SYSTEM WIZYJNY 2-D
Size
B
Document Number
Pakiet filtracji
REV
Date: December 5, 1988
Sheet 1 of 1


D3
Bb10: K
Bb23: KONP
U22B
7408
+5V
U22C
U38A
U5A
74LS393
U5B
U6
8253
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
D0
D1
D2
D3
D4
D5
D6
D7
WR
RD
A0
A1
CS
U23
7430
Y
U38A
7404
7474
U36B
CLEAR'174
U22D
U4B
Bb26: FL2
U36C
U39A
LICZ
Bb17: CL'256
Bb15: VO
CLK'174
U38B
D2
D1
U38D
+5V
U8
74193
U9A
U9B
CSRLC
MW
U35B
7408
CE
U4A
U7A
7404
CS2-8253(P,K)
CS4-8253(S1,S2)
U1D
C17a:A14
U7B
C17c:A15
U7C
D17a:A16
U3
8205
A0
A1
A2
E1
E2
E3
O0
O1
O2
O3
O4
O5
O6
O7
CS1
CS2
CS3
CS4
CS5
CS6
U2
7430
U1F
DIR
D12b: EXTMEM
C8a: MROOO
C8c: MWTC
MROOO
U35D
U35A
U36A
7474
CLEAR'174
Bb18:WYG
Bb24: V
U37A
7432
U7E
C2
120pF
C1
1nF
CLEAR '393
U13D
7407
U1A
R1
1200
+5V
U34
74LS373
2/74245 (32)
CS1: 4000H
CS2: 8000H
CS3: C000H
CS4: 10000H
CS5: 14000H
CS6: 18000H
Bb8 - CLCK
Bb9 - CLB
Bb10 - K
Bb11 - FL1
Bb14 - NV
Bb15 - VO
Bb17 - CL'256
Bb21 - H
Bb23 - KONP
Bb24 - V
Bb25 - WYG
Bb26 - FL2
SYSTEM WIZYJNY 2-D
Title
Pakiet preprocesora
Size
B
Document Number
REV
Date: December 5, 1989
Sheet 1 of 1


LICZX
B521: H
Y
CLEAR*393
CLEAR*174
CLK*174
B515: VO
U10A
U10B
U11A
U11B
U12A
U12B
U14
U15
U16
U17
U18
U19
U20
U21
U24
U25
U26
U27
U28
U29
U30
U31
U32
U33
74LS393
74181
74174
74LS245
8253
8264
7404
7408
+5V
SYSTEM WIZYJNY 2-D
Pakiet preprocesora
Size B
Document Number
REV
Date: December 5, 1989
Sheet 2 of 2