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

Ośrodek Automatyki
Elektrycznej
Pracownia Cyfrowych Systemów Wizyjnych

0774 A

Główny wykonawca

Wykonawcy dr inż. Bohdan Kontrymowicz
mgr inż. Dariusz Okrasa
mgr Karol Najar

Konsultant

Nr zlecenia RP-63

Rodzina układów wizyjnych opartych na kamerach linijkowych CCD dla robotów przemysłowych.
Zadanie nr 2.2
Uruchomienie modelu, oprogramowanie podstawowe i badania funkcjonalne układu wizyjnego do analizy jednej linii obrazowej.

Zleceniodawca CPBR 7.1 "Roboty przemysłowe"

Pracę rozpoczęto dnia 02.05.1988 **zakończono dnia** 30.11.1988

Kierownik Pracowni
mgr inż. D.Okrasa

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

Kierownik Ośrodka
dr inż. B.Kontrymowicz

Praca zawiera:

stron	5
rysunków	2
fotografii	-
tabel	-
tablic	-
załączników	3

Nr rejestr. 6180

Rozdzielnik - ilość egz:

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

Analiza deskryptorowa

ROBOT PRZEMYSŁOWY, SYSTEM WIZYJNY, CZUJNIK

Układy wizyjne dla robotów
Kamery linijkowe CCD
Czujniki wizyjne
Czujniki odległości

Analiza dokumentacyjna

W sprawozdaniu przedstawiono oprogramowanie testujące i testy funkcjonalne dla wielokanałowego układu wizyjnego do analizy jednej linii obrazowej.

Tytuły poprzednich sprawozdań

UR-02.04.01	Zastosowanie robota IRb-6 do kontroli wymiarów
1a.	Wykonanie analizy możliwości zastosowania robotów przemysłowych do kontroli wymiarów w oparciu o literaturę. Nr rejstr. 4375
1b.	Założenia techniczne i koncepcje rozwiązań technicznych. Nr rejestr. 4452
2.	Projekt modelu głowicy pomiarowej oraz projekt modelu przetwarzającego wyniki pomiarów. Nr rejestr. 5395
RP-63	Rodzina układów wizyjnych opartych na kamerach linijkowych CCD dla robotów przemysłowych.
1.1	Opracowanie dokumentacji modeli rodziny kamer linijkowych CCD. Nr rejestr. 5744
1.2	Opracowanie oprogramowania i badania modelu urządzenia do bezdotykowej kontroli wymiarów na stanowisku laboratoryjnym. Nr rejestr. 5810
1.3	Wykonanie i przebadanie modeli rodziny kamer linijkowych CCD. Przebadanie modelu do bezdotykowej kontroli wymiarów za pomocą robota przem. na stanowisku z IRp-6. Nr rejestr. 5926
2.1	Opracowanie dokumentacji i montaż modelu wielokanałowego układu wizyjnego do analizy jednej linii obrazowej (1-D). Nr rejestr. 6034

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

PIAP-252/93-6000

SPIS TEŚCI

1. WSTĘP

Zadaniem projektowanego układu jest obsługa trzech kamer liniijkowych CCD. Umożliwia on zapamiętanie, analizę i przetworzenie informacji uzyskanej z obrazu oraz przesłanie wyników przetwarzania do urządzeń współpracujących. Dokładny opis budowy, przeznaczenia i funkcji układu zamieszczony jest w sprawozdaniu z realizacji zadania 2.1, nr. rejestracyjny PIAP 6034.

Według projektu pakiet był wyposażony w dwa interfejsy do wymiany informacji z otoczeniem: interfejs szeregowy i interfejs równoległy. Jednym z zadań interfejsu równoległego było zapewnienie komunikacji z innymi pakietami umieszczonymi w kasecie poprzez magistralę kasety.

Jednakże na skutek tego, że obecnie proponowany protokół wymiany informacji pomiędzy układem sterowania robota przemysłowego, a inteligentnymi czujnikami zewnętrznymi robota zakłada wykorzystanie wyłącznie łącza szeregowego, zrezygnowano ze współpracy pakietu bezpośrednio z magistralą kasety.

Pakiet wykorzystuje z magistrali jedynie napięcia zasilające, w przypadku współpracy z robotem komunikacja z układem sterowania odbywa się łączem szeregowym, a port równoległy może być wykorzystywany do sterowania urządzeniami zewnętrznymi. Możliwości umieszczenia na pakiecie pamięci ROM i RAM o pojemności do 16k każda, sprawiają, że dodatkowe pakiety pamięci nie są potrzebne.

2. REALIZACJA ZADANIA.

W ramach realizacji etapu wykonane zostały następujące prace:

-uruchomiono model wielokanałowego układu wizyjnego do analizy jednej linii obrazowej,

-wykonano i uruchomiono oprogramowanie testujące dla w/wym. układu,

-przeprowadzono testy funkcjonowania pakietu z oprogramowaniem.

Prace uruchomieniowe prowadzono z wykorzystaniem emulatora procesora INTEL 8031 współpracującego z komputerem IBM PC/AT.

2.1. Uruchomienie układu.

W trakcie uruchamiania pakietu wprowadzono zmiany w konstrukcji pakietu w odniesieniu do projektu zawartego w sprawozdaniu z zadania 2.1.

2.1.1. Połączono wyjścia portu "P.1" procesora z wejściami "D" przerzutników U36, U37 i U38 a mianowicie:

U1/1 z U36/2 i U36/12 */

U1/2 z U37/2 i U37/12

U1/3 z U38/2 i U38/12

Pozwoliło to na obsługę tylko określonej kamery z jednoczesnym zablokowaniem pozostałych.

2.1.2. Pominięto kondensatory C6, C7 i C8, które służyły jako kondensatory całkujące sygnały z kamer i okazały się niepotrzebne.

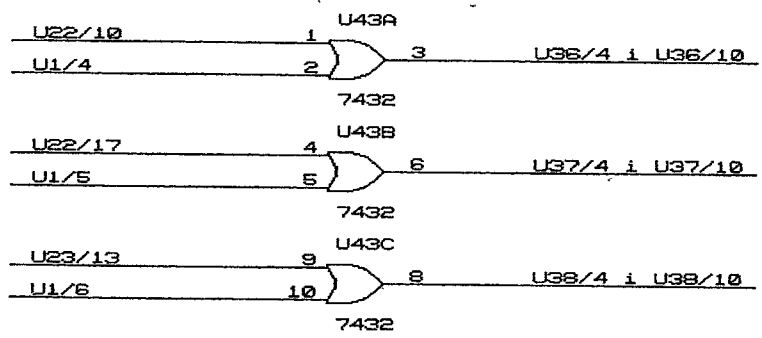
2.1.3. Pominięto inwerter umieszczony pomiędzy U32/10, a U1/10 i połączono te końcówki bezpośrednio, w celu uzyskania prawidłowej logiki sygnału trasmisji szeregowej.

---

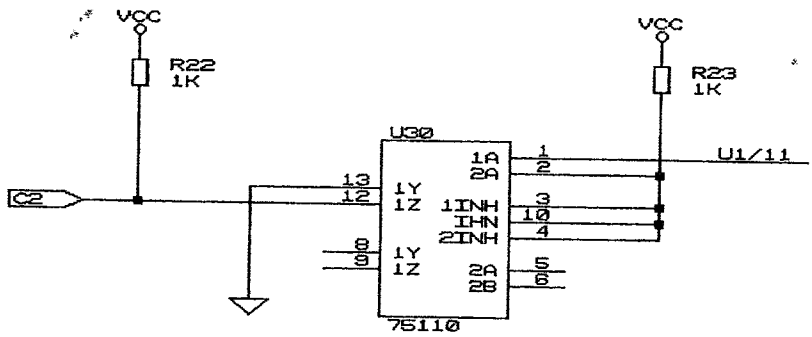
*/ - Zapis Ux/y oznacza y-tą końcówkę układu x.

2.1.4. Zastąpiono układ U10 UCY7432 układem UCY7402 w celu uzyskania właściwej polaryzacji sygnału strobującego układy 8283.

2.1.3. Dodano układ U43 UCY 7432 (dla uzyskania efektu jak w punkcie 2.1.1) połączony w następujący sposób:



2.1.4. Zamieniono miejscami wyjścia nr.12 i 13 układu U30 (dla uzyskania efektu jak w punkcie 2.1.3) tak, że powstał układ:



2.2. Oprogramowanie.

Program testujący napisany niemal w całości w języku C, został skompilowany kroskompilatorem ICC (IAR C Compiler for Microprocessor Development v. 2.00) oraz zlinkowany linkerem tej samej firmy (IAR).

Uruchomienie odbyło się przy użyciu emulatora NEW MICE II produkcji firmy MICROTEK.

Algorytm programu został zamieszczony w Załączniku 1, a listing programu w Załączniku 2.

2.1.1. Założenia programu.

Żadanie postawione przed programem testującym pakiet obsługi trzech kamer linijkowych CCD składało się z następujących elementów:

-realizacja algorytmu "synchronizacji" każdej kamery,
-realizacja algorytmu obsługi kamer w oparciu o układ przerwań,
-realizacja komunikacji z otoczeniem za pomocą sprzęgu szeregowego i równoległego.

Autor oprogramowania uznał za właściwe ująć w/w funkcje w jeden program realizujący powyższe, a działanie poszczególnych urządzeń traktować jako weryfikację poprawnego działania.

2.1.2. Stuktura programu.

Program został podzielony na moduły programowe realizujące poszczególne zadania stawiane przed pracującym sprzętem. Za sukces autor uważa realizację całości prac w języku C i nie korzystanie z możliwości łączenia z fragmentami napisanymi w asemblerze. W celu zbierania doświadczeń zaimplementowany został fragment napisany w asemblerze 8031 zawierający deklarację bufora przeznaczonego na komunikaty z/do portu szeregowego. Niezbędne okazało się wstawienie do programu STARTUP adresów programów obsługi poszczególnych zgłoszeń przerwań.

st ini — jest pakietem inicjującym większość parametrów wykorzystywanych w programie oraz sprzęcie takich jak ustawienie zawartości timerów, inicjalizacji struktury informacyjno-sterującej pst i zbudowanego wektora tych struktur pstve, inicjalizacji portu szeregowego i równoległego,

obslu — jest pakietem dokonującym synchronizacji kamery. Bada stany rejestru ALARM, zapamiętuje w tablicy "ważne" parametry, zapamiętuje zmiany stanów timerów i przygotowuje reakcję systemu na te zmiany,

int serial — jest pakietem umożliwiającym transmisję z/do portu szeregowego, pobiera znaki z zadeklarowanego bufora i interpretuje je. Komunikaty, o których należy powiadamiać system zgłasza do obsługi,

wex — moduł przeznaczony do wpisywania wartości (obslu) do komórek przypisanych danej kamerze (8000-8002),

rex — moduł przeznaczony do zczytywania wartości (obslu) z komórek przypisanych danej kamerze (A000-A002, C000-C002),

parat — ustawia wartości portu równoległego,

wr scr — kształtuje ekran w porcie szeregowym,

wr pre — przygotowuje do wydruku informacje o przekroczeniach w torach kamer,

new al in — moduł czytania rejestru alarmów i właściwego wypełniania struktur informacyjnych.

2.3. Badania funkcjonalne.

Badania funkcjonalne objęły:

-sprawdzenie automatycznej regulacji czasu naświetlania elementu CCD niezależnie dla każdej z kamer,

-wykrywanie i sygnalizacja sytuacji, w której poziom sygnału wizyjnego przekracza poziom alarmowy,

-określanie i pamiętanie numerów punktów obrazu, w których poziom sygnału wizyjnego przekroczy wartość progową,

-przetwarzanie danych zawartych w pamięci,

-wysyłanie wyników do urządzeń zewnętrznych poprzez posiadane interfejsy.

Wszystkie powyższe funkcje są zrealizowane przykładowo w programie testującym, a poprawność działania została sprawdzona w trakcie wykonywania programu.

Do komunikacji z pakietem wykorzystano komputer IBM PC/XT wykonujący program MSKERMIT. Po ukazaniu się na ekranie monitora komunikatu:

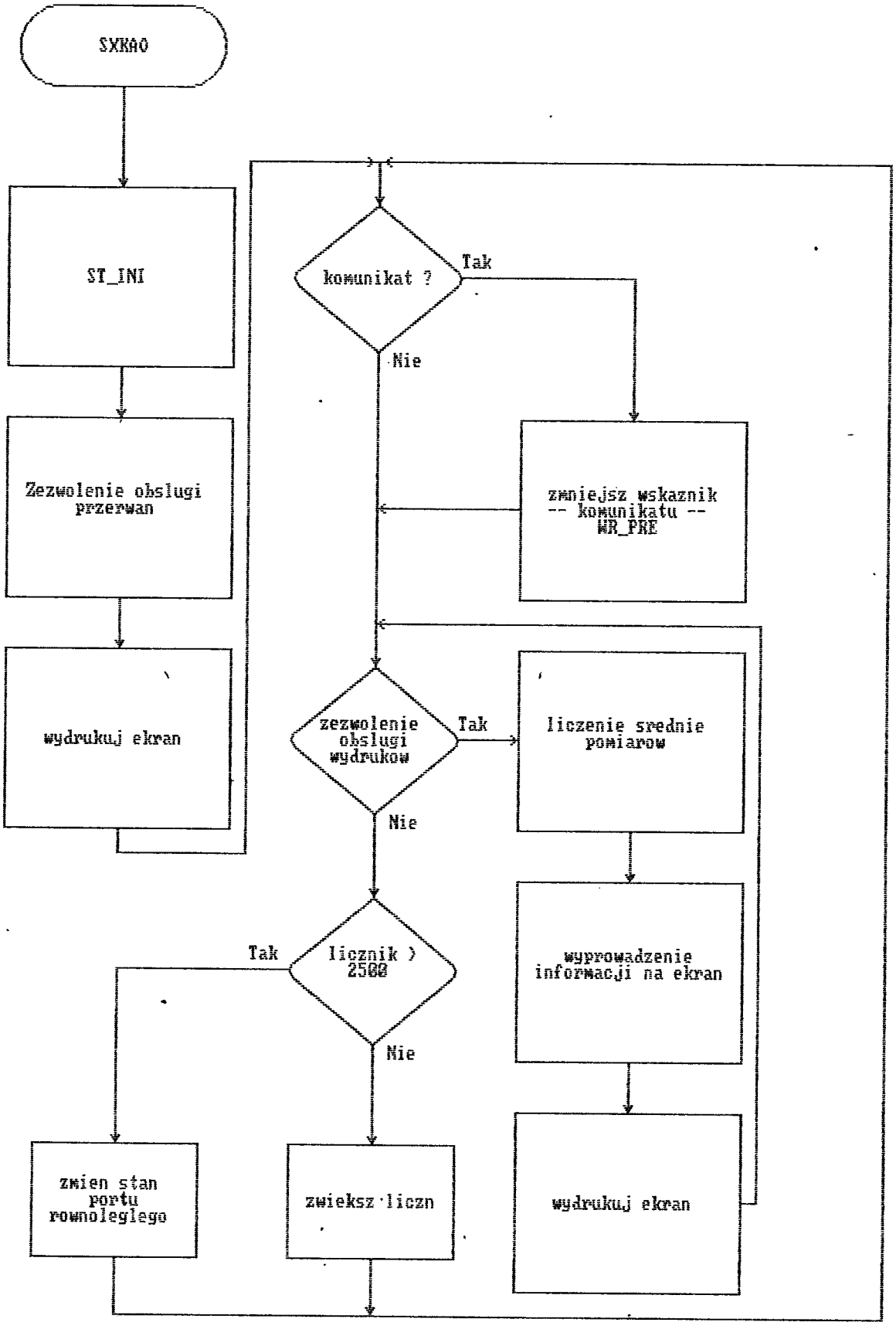
```
0 <- kamera A
1 <- kamera B
2 <- kamera C

( ? )
```

i wybraniu numeru kamery pakiet wysyła zwrotnie numer wybranej kamery, a następnie wyniki pomiarów oraz komunikaty. Przykłady wyglądu ekranu monitora w czasie wykonywania przez pakiet programu testującego zamieszczono w Załączniku 3.

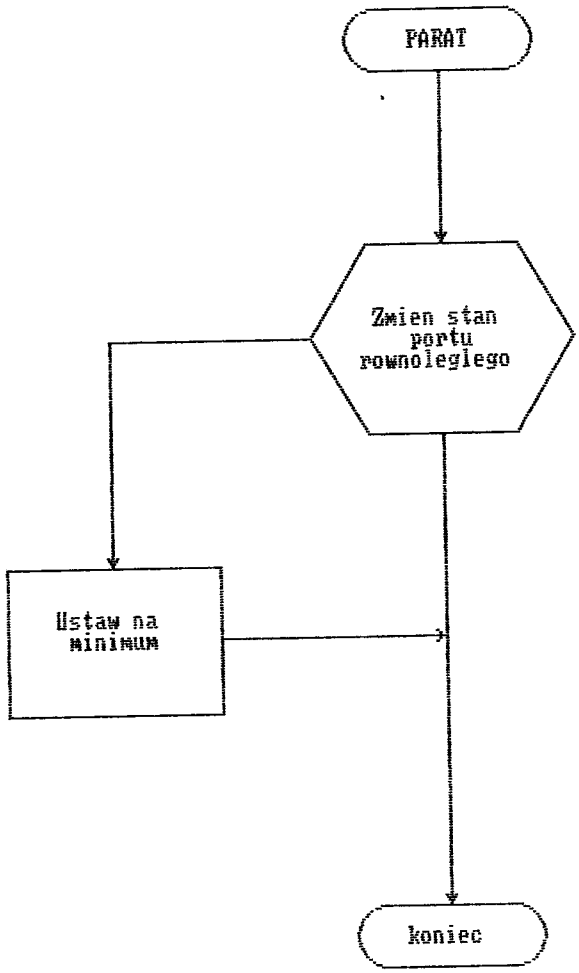
Załącznik 1.

Algorytm programu testującego.

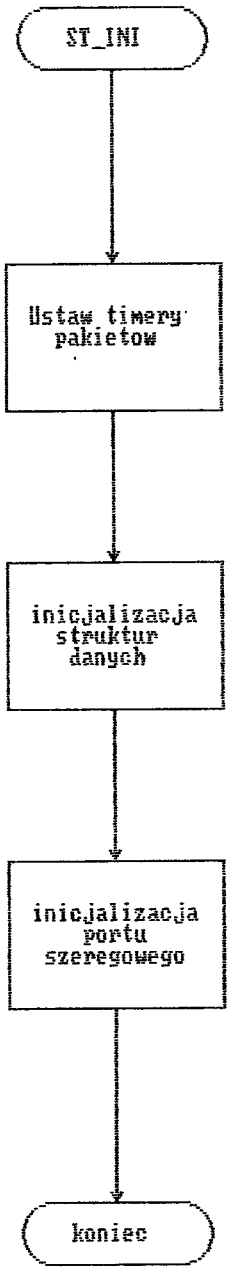
MODUL SXKA0


SXKA0
ST_INI
Zezwolenie obslugi przerwan
wydrukuj ekran
komunikat ?
Tak
Nie
zmniejsz wskaznik -- komunikatu -- WR_PRE
zezwolenie obslugi wydrukow
Tak
liczenie srednie pomiarow
Nie
licznik) 2500
Tak
Nie
wyprowadzenie informacji na ekran
zmien stan portu rownoleglego
zwieksz liczn
wydrukuj ekran

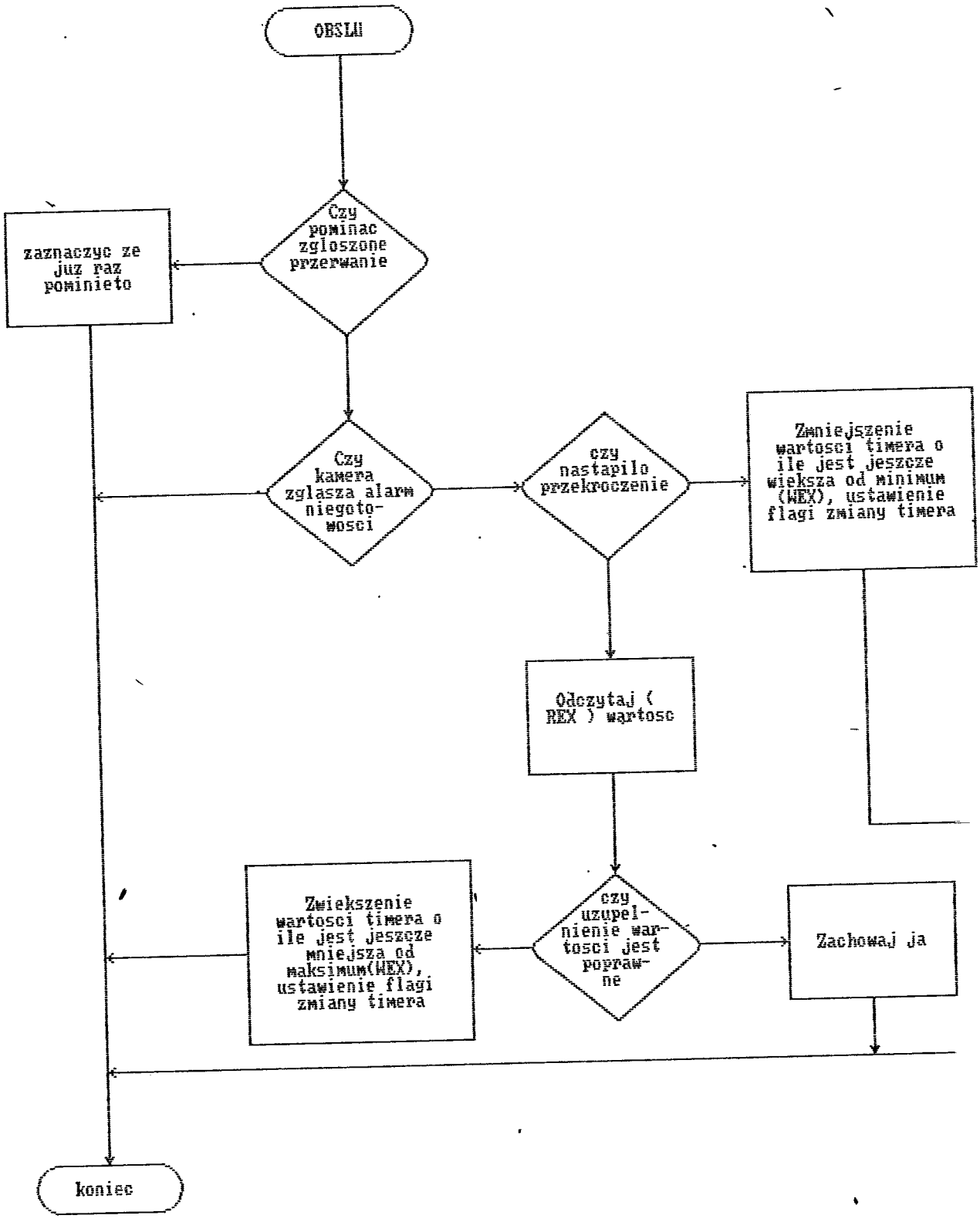
MODUL PARAT


PARAT
Zmien stan portu rownoleglego
Ustaw na minimum
koniec

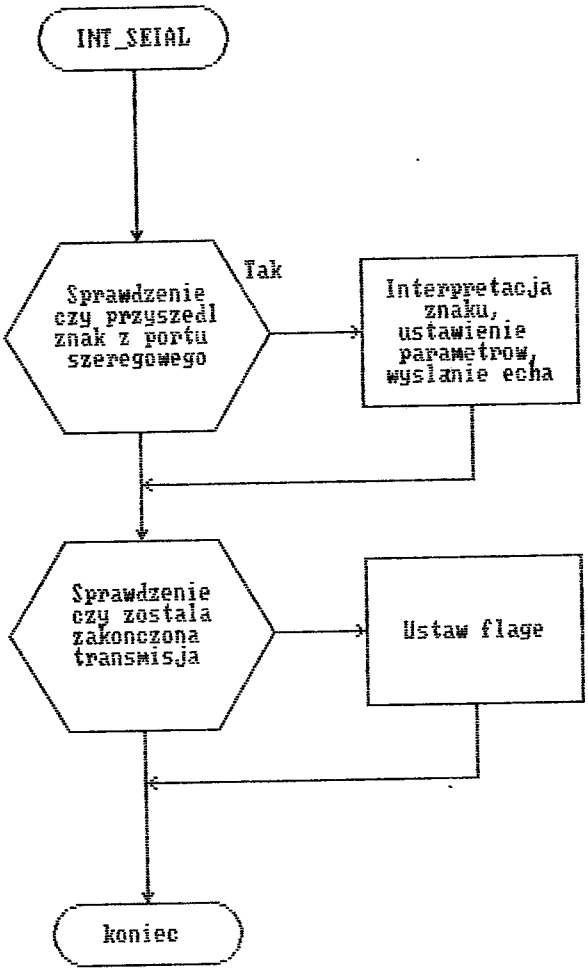
MODUL ST_INI


ST_INI
Ustaw timery pakietow
inicjalizacja struktur danych
inicjalizacja portu szeregowego
koniec

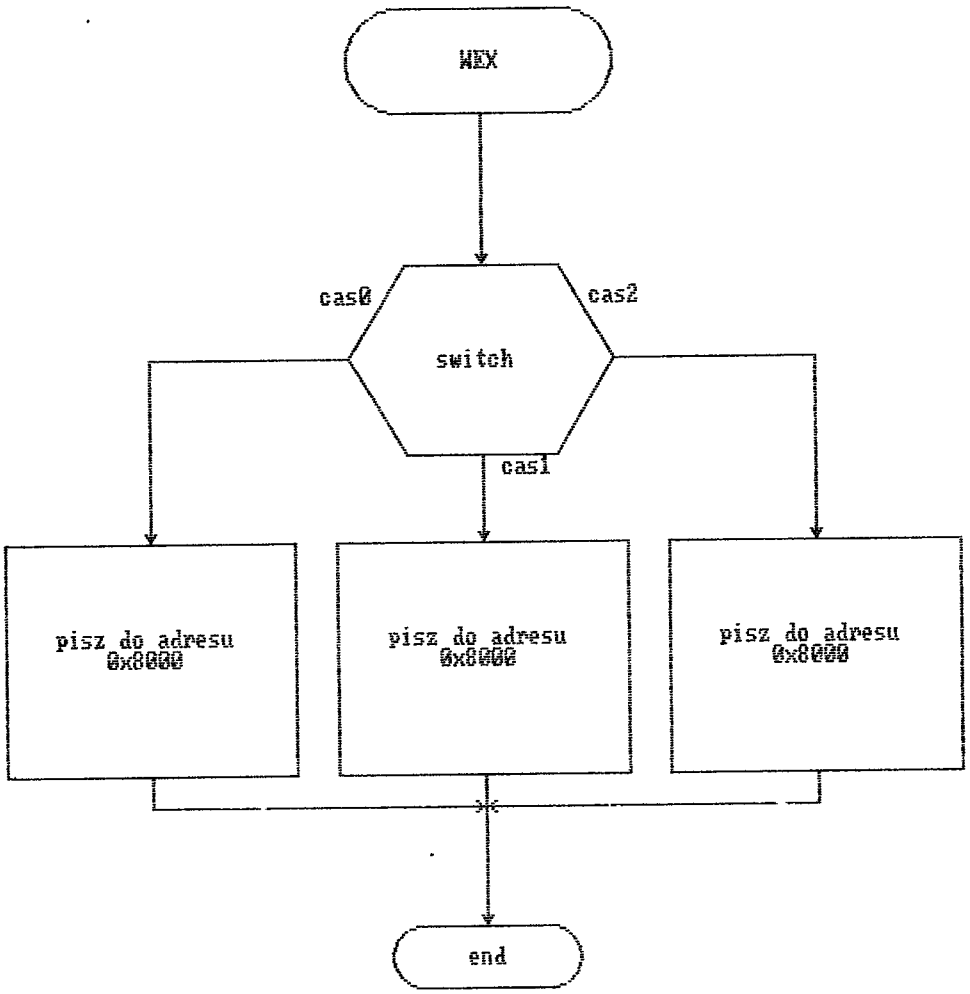
MODUL OBSLU


OBSLU
Czy pominac zgloszone przerwanie
zaznaczyc ze juz raz pominieto
Czy kamera zglasza alarm niegoto-wosci
czy nastapilo przekroczenie
Zmniejszenie wartosci timera o ile jest jeszcze wieksza od minimum (MEX), ustawienie flagi zmiany timera
Odczytaj (REX) wartosc
czy uzupel-nienie war-tosci jest poprawne
Zwiekszenie wartosci timera o ile jest jeszcze mniejsza od maksimum(MEX), ustawienie flagi zmiany timera
Zachowaj ja
koniec

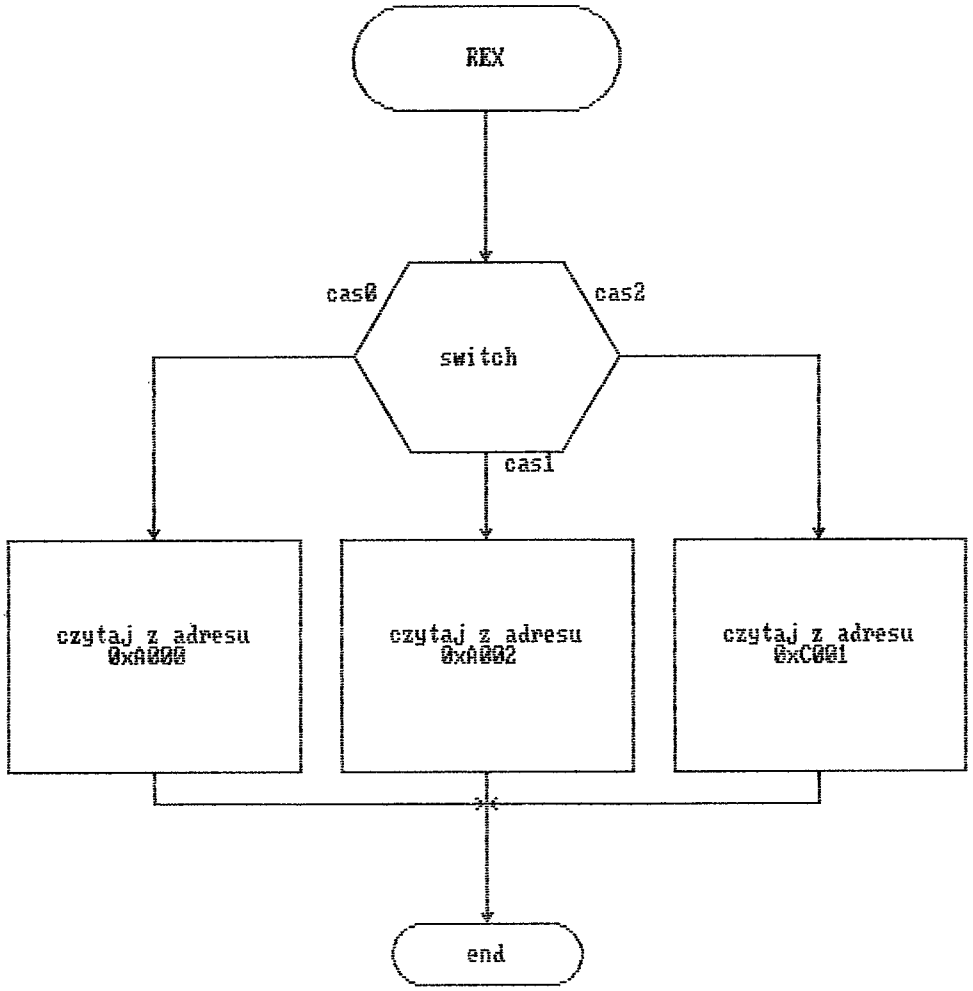
MODUL INT_SERIAL



MODUL WEX


WEX
switch
cas0
cas1
cas2
pisz do adresu 0x8000
pisz do adresu 0x8000
pisz do adresu 0x8000
end

MODUL REX


REX
cas0
cas2
switch
casi
czytaj z adresu
0xA000
czytaj z adresu
0xA002
czytaj z adresu
0xC001
end

Załącznik 2.

Program testujący.

```
;/*******************************************************
;  Przemyslowy   Instytut   Automatyki  i   Pomiarow
;                   MERA - PIAP
;
;     Osrodek    Automatyzacji    Elektrycznej
;
;              02-222   Warszawa
;           Aleje Jerozolimskie 202
;
;
;    Autor :          Karol Najar
;    Warszawa         1988.12.14
;
;    ╔═══════════════════════╗
;    ╟                       ╢
;    ╟  MODULE  R_BUF        ╢
;    ╟                       ╢
;    ╚═══════════════════════╝
;
;   Deklaracja bufora komunikacyjnego
;
;
;*******************************************************/

RSEG DATA
PUBLIC bufer
bufer  equ  $
       ds 50
END
```

```
/*************************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════╗
    ‖                   ‖
    ‖  MODULE  ST_INI   ‖
    ‖                   ‖
    ╚═══════════════════╝

Procedura poczatkowej inicjalizacji zmiennych i stalych
w programie i srodowisku procesora.


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


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


extern PST pstve[3];
extern char o_exi, b_int, o_kla ;

void st_ini(void)
{
int i,inn ;

write_XDATA ( 0x8003 , 0x34 );
write_XDATA ( 0x8003 , 0x74 );
write_XDATA ( 0x8003 , 0xB4 );
write_XDATA ( 0xA003 , 0x34 );
write_XDATA ( 0xA003 , 0x74 );
write_XDATA ( 0xA003 , 0xB4 );
write_XDATA ( 0xC003 , 0x34 );
write_XDATA ( 0xC003 , 0x74 );
write_XDATA ( 0xC003 , 0xB4 );

write_XDATA ( 0x8000 , 0x00 );
write_XDATA ( 0x8000 , 0x04 );
write_XDATA ( 0x8001 , 0x00 );
write_XDATA ( 0x8001 , 0x04 );
write_XDATA ( 0x8002 , 0x00 );
write_XDATA ( 0x8002 , 0x04 );

write_XDATA ( 0xA000 , 0x10 );
write_XDATA ( 0xA000 , 0x01 );

write_XDATA ( 0xA001 , 0x10 );
write_XDATA ( 0xA001 , 0x01 );

write_XDATA ( 0xA002 , 0x10 );
write_XDATA ( 0xA002 , 0x01 );

write_XDATA ( 0xC000 , 0x10 );
write_XDATA ( 0xC000 , 0x01 );

write_XDATA ( 0xC001 , 0x10 );
write_XDATA ( 0xC001 , 0x01 );

write_XDATA ( 0xC002 , 0x10 );
write_XDATA ( 0xC002 , 0x01 );

pstve[0].licz = 1024 ;
pstve[1].licz = 1024 ;
pstve[2].licz = 1024 ;
pstve[0].lix = 1 ;
pstve[1].lix = 1 ;
pstve[2].lix = 1 ;

for ( inn=0 ; inn < 3 ; inn++ ) {
        for ( i=0 ; i< 13 ; i++ ) {
            pstve[inn].wart[i] = 0 ;
           }
 }

output ( TMOD , 0x20 ); /* bylo 20 */ /* 22 */
output ( TCON , 0x40 ); /* bylo 40 */ /* 50 */
output ( SCON , 0x70 );
output ( TH1  , 0xF5 );    /* f3 bylo dla zegara 12 MHz */
output ( TH0  , 0xFF );    /* max  */
set_bit   ( EXi_bit ) ;
clear_bit ( ET2_bit ) ;
clear_bit ( ET1_bit ) ;
clear_bit ( ES_bit ) ;
clear_bit ( ET0_bit ) ;
clear_bit ( EX0_bit ) ;
o_exi = 0   ;
b_int = 1   ;
}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

            02-222   Warszawa
          Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════════╗
    ‖                       ‖
    ‖  MODULE  KAMER.H      ‖
    ‖                       ‖
    ╚═══════════════════════╝

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

typedef struct pst {
 int licz  ;
 char  lix ;
 char  fz ;
 char  fn ;
 char  fp ;
 int wart[13] ;
 ) PST ;
```

```
/*************************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

             02-222   Warszawa
          Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════════╗
    ‖                       ‖
    ‖  MODULE   WR_SCR      ‖
    ‖                       ‖
    ╚═══════════════════════╝

Obsluga ekranu .


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


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


extern  char bufer[] ;
extern char *buf, *bufr ;
extern char flar  ;
extern char flaw  ;
extern PST pstve[3] ;
extern char o_exi, b_int, o_kla ;


void wr_scr(void)
{
       bufr = &bufer ;
       sprintf (  bufer, "\r\n 0 <- kamera A " );
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
   while (flaw) ;
       sprintf (  bufer, "\r\n 1 <- kamera B " );
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
    while (flaw) ;
       sprintf (  bufer, "\r\n 2 <- kamera C " );
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
    while (flaw) ;
       sprintf (  bufer, "\r\n\r\n[ ? ]" ); /*  3 <- kamera A & B & C */
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
    while (flaw) ;
 }
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

            02-222   Warszawa
          Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ┌──────────────────────┐
    │                      │
    │  MODULE  WR_PRE      │
    │                      │
    └──────────────────────┘

Obsluga komunikatow o kamerach.

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


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


extern  char bufer[] ;
extern char *buf, *bufr, flaw ;


void wr_pre(alf,bet)
char alf,bet ;
{
if ( bet ) {
       sprintf (  bufer, "\r\n U W A G A   za silne SwiatLo  kamery %d", alf );
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
   while (flaw) ;
}

else
{
       sprintf (  bufer, "\r\n U W A G A   za sLabe SwiatLo  kamery %d", alf );
       buf = &bufer ;
       flaw = 1 ;
       output ( SBUF , *buf++ );
   while (flaw) ;
}


}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


     Autor :          Karol Najar
     Warszawa         1988.12.14

     ╔═══════════════════════════╗
     ‖                           ‖
     ‖  MODULE  MAKSIM           ‖
     ‖                           ‖
     ╚═══════════════════════════╝

Wyszukiwanie najwiekszych wartosci z dobrych pomiarow

     ╔═══════════════════════════╗
     ‖                           ‖
     ‖  MODULE  MINIM            ‖
     ‖                           ‖
     ╚═══════════════════════════╝

Wyszukiwanie najmniejszych wartosci z dobrych pomiarow

     ╔═══════════════════════════╗
     ‖                           ‖
     ‖  MODULE  SXKAD            ‖
     ‖                           ‖
     ╚═══════════════════════════╝

Petla glowna


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


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

extern void st_ini(void), parat(void), wr_pre(char,char) ;
extern void wr_scr(void), ccam(int)  ;

extern  char bufer[] ;
extern char *buf, *bufr ;
extern char flar  ;
extern char flaw  ;

char n_kam, ext ;
char b_int ;
char o_exi, o_kla ;
int  liczn ;
static double suma, mini, maxi ;


PST pstve[3]   ;

int minim(n_kam)
char n_kam ;
{
int i;
int tem = 256 ;
   for ( i=0 ; i<11 ; i++ ) {
       if ( tem > pstve[n_kam].wart[i] )  tem = pstve[n_kam].wart[i] ;
   }
   return( tem ) ;
}


int maksim(n_kam)
char n_kam ;
{
int i;
int tem = 0 ;
   for ( i=0 ; i<11 ; i++ ) {
       if ( tem < pstve[n_kam].wart[i] )  tem = pstve[n_kam].wart[i] ;
   }
   return( tem ) ;
}


main ()

{
int i ;
clear_bit ( EA_bit ) ;
st_ini() ;
set_bit ( ES_bit )  ;
set_bit ( EA_bit ) ;
wr_scr();
for (;;) {
```

```
  if (ext) {
    ext--   ;
    wr_pre(n_kam,ext);
    ext = 0 ;
  }
  while (o_exi) {
      suma = 0 ;
      for ( i=1 ; i<11 ; i++ ){
          suma += pstve[n_kam].wart[i] ;
          }
      sprintf( bufer, "\r\nwybrana kamera %d, srednia wynosi = %7.1f", n_kam, suma/10 ) ;
      buf = &bufer`;
      flaw = 1 ;
      output ( SBUF , *buf++ );
      while (flaw) ;
      sprintf( bufer, "\r\nminimum wartosci %d, maximum wynosi = %d", minim(n_kam) , maksim(n_kam) ) ;
      buf = &bufer ;
      flaw = 1 ;
      output ( SBUF , *buf++ );
      while (flaw) ;
      wr_scr();
      o_exi = 0 ;
      } /* end while */
      ccam(o_kla) ;
      if ( liczn > 2500 ) {
          parat ();
          liczn = 0 ;
      }
      else { liczn++ ; }
}      /* end for    */
}
```

```
/*************************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════════╗
    ‖                       ‖
    ‖  MODULE   PARAT       ‖
    ‖                       ‖
    ╚═══════════════════════╝

    Obsluga portu rownoleglego .

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


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

extern int liczn ;

/*
void para(void)
{
 liczn++ ;
}
*/

void parat(void)
{
   static char paa ;
   switch ( paa ) {
       case 0:   paa = 1 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 1:   paa = 2 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 2:   paa = 4 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 4:   paa = 8 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 8:   paa = 16 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 16:  paa = 32 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 32:  paa = 64 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 64:  paa = 128 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       case 128: paa = 1 ;
                 write_XDATA ( 0x6000 , paa ) ;
                 break ;
       default:  paa = 1;
   }
}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                   MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


     Autor :          Karol Najar
     Warszawa         1988.12.14

     ╔═══════════════════════════╗
     ‖                           ‖
     ‖  MODULE   NEW_AL_IN       ‖
     ‖                           ‖
     ╚═══════════════════════════╝

Procedura ustawiajaca wskazniki przekroczenia
oraz niegotowosci kamery wedlug odczytanego
rejestru alarmow.
Informacje te sa przechowywane w strukturze
opisujacej stany oraz wartosci pomiarow kamer
wspolpracujacych.

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


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


extern PST pstve[3]   ;
void new_al_in (void)
{
unsigned char alarm ;

   alarm = read_XDATA ( 0xE000 ) ;
( alarm&0x80  ) ? ( pstve[2].fn = 1 ) : ( pstve[2].fn = 0 ) ;
( alarm&0x10  ) ? ( pstve[2].fp = 1 ) : ( pstve[2].fp = 0 ) ;
( alarm&0x40  ) ? ( pstve[1].fn = 1 ) : ( pstve[1].fn = 0 ) ;
( alarm&0x08  ) ? ( pstve[1].fp = 1 ) : ( pstve[1].fp = 0 ) ;
( alarm&0x20  ) ? ( pstve[0].fn = 1 ) : ( pstve[0].fn = 0 ) ;
( alarm&0x04  ) ? ( pstve[0].fp = 1 ) : ( pstve[0].fp = 0 ) ;
}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

             02-222   Warszawa
          Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════════════╗
    ╟                           ╢
    ╟    MODULE   WEX           ╢
    ╟                           ╢
    ╚═══════════════════════════╝
Modul WEX pozwala wpisywac wartosci do odpowiednich licznikow sterujacych
kamerami
    ╔═══════════════════════════╗
    ╟                           ╢
    ╟    MODULE   REX           ╢
    ╟                           ╢
    ╚═══════════════════════════╝
Modul REX pozwala czytac wartosci z odpowiednich licznikow sterujacych
kamerami
    ╔═══════════════════════════╗
    ╟                           ╢
    ╟  MODULE   OBSLU           ╢
    ╟                           ╢
    ╚═══════════════════════════╝
Procedura OBSLU pozwala na realizacje algorytmu synchronizacjii
kamer.

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


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


extern PST pstvel[3] ;
extern char bufer[], *buf, flaw ;
extern char ext ;



void wex ( vec, aar )
    char vec ;
    unsigned int aar ;

{
 unsigned char tym1, tym2 ;
 tym1 = ( char ) aar % 256 ;
 tym2 = ( char ) ( aar >> 8 ) ;
 switch ( vec ) {

 case 0 : write_XDATA ( 0x8000 ,tym1 ) ;
          write_XDATA ( 0x8000 ,tym2 ) ;
     break ;
 case 1 : write_XDATA ( 0x8001 ,tym1 ) ;
          write_XDATA ( 0x8001 ,tym2 ) ;
     break ;
 case 2 : write_XDATA ( 0x8002 ,tym1 ) ;
          write_XDATA ( 0x8002 ,tym2 ) ;
     break ;
   }
 }


char rex ( vec )
 char vec ;
{
 char tym ;

 switch ( vec ) {
 case 0 : tym = read_XDATA ( 0xA000 ) ; /* A000 */
     return ( tym ) ;
     break ;
 case 1 : tym = read_XDATA ( 0xA002 ) ; /* A002 */
     return ( tym ) ;
     break ;
 case 2 : tym = read_XDATA ( 0xC001 ) ; /* C001 */
     return ( tym ) ;
     break ;
   }
 }

void obslu (alf)
int alf    ;
{


unsigned int  tem  ;
PST *p_str ;
unsigned int res_chan ;
```

```
p_str = &pstve[alf]     ;
 if (p_str->fz) {
       p_str->fz = 0  ;
      }
  else {
      if ( p_str->fn == 0 ) {
          if ( p_str->fp != 0 ) {
              tem = p_str->licz >> 2 ;
              tem = p_str->licz - tem ;
              if ( tem > 300 ) {          /* bylo 450 */
                    wex( alf , tem ) ;    /* zmiana zawartosci licznika */
                    p_str->licz = tem ;   /* zapamietanie wartosci */
                    p_str->fz = 1     ;   /* wskaznik zmiany licznika */
                    }
              else {              /* przygotowanie komunikatu */
                  ext=2 ;
                  p_str->lix=11   ;
                  }
          }       /*     if fp != 0       */
          else {
              /* detekcja nie_przekroczenia */
              res_chan  = rex( alf ) ;
              res_chan += rex( alf ) * 256 ;
              if ( ( 0x0110 - res_chan) < 256 ) {
                  p_str->wart[p_str->lix] = ( 0x0110 - res_chan ) ;
                  p_str->lix++                ;
              }
              else {
                  tem  = p_str->licz >> 2 ;
                  tem +=  p_str->licz       ;
                  if ( tem < 10000 ) {
                       wex ( alf,tem  ) ;     /* zmiana zawartosci licznika */
                       p_str->licz =tem ;     /* zapamietanie wartosci */
                       p_str->fz =1     ;     /* wskaznik zmiany licznika */
                       }
              else {              /* przygotowanie komunikatu */
                  ext=1 ;
                  p_str->lix=11   ;
                  }
              }
          }
      }
   }
}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

            02-222   Warszawa
          Aleje Jerozolimskie 202


     Autor :          Karol Najar
     Warszawa         1988.12.14

     ┌──────────────────────┐
     │                      │
     │  MODULE INT_SERIAL   │
     │                      │
     └──────────────────────┘

Procedura obslugi portu transmisjii szeregowej
parametry transmisyjne sa ustawiane w module
ST_INI.
Procedura ponizsza jest wolana w momencie
przerwania z portu szeregowego t.zn, ze w momencie
gdy znak zostal odebrany badz zostalo zakonczone
nadawanie.

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


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



extern char n_kam ;          /* wskaznik blokada_przerwan inicjalizacja w module sxkao */
extern char b_int ;          /* wskazniki obsluga_zakonczenia_synchronizacji_kamery */
extern char o_exi, o_kla ;   /* obsluga_reakcji_na_wcisniecie_klucza_klawiatury
                                inicjalizacja w module sxkao. */

char *buf, *bufr ;       /* pointery pomocniczych buforow transmisji */
char flar  ;             /* flaga zakonczenia czytania */
char flaw  ;             /* flaga zakonczenia pisania  */



void int_serial (void)
     {
     if (read_bit ( RI_bit ))  {                /* detekcja odebranego znaku */
             *bufr = input ( SBUF )&0x7f ;      /* przepisanie znaku do bufora
                                                   pomocniczego, wraz z usunieciem
                                                   bitu parzystosci. */
             clear_bit ( RI_bit ) ;             /* zakonczenie czytania */
             switch ( *bufr - '0' ) {
              case 0: output ( SBUF , *bufr ); /* echo kontrolne */
                      n_kam = 0 ;      /* parametr */
                      o_kla = 0 ;      /* parametr */
                      break ;
              case 1: output ( SBUF , *bufr ); /* echo kontrolne */
                      n_kam = 1 ;      /* parametr */
                      o_kla = 1 ;      /* parametr */
                      break ;
              case 2: output ( SBUF , *bufr ); /* echo kontrolne */
                      n_kam = 2 ;      /* parametr */
                      o_kla = 2 ;      /* parametr */
                      break ;
              case 3: output ( SBUF , *bufr ); /* echo kontrolne */
                      n_kam = 0 ;      /* parametr */
                      o_kla = 3 ;      /* parametr */
                      break ;
              default :
                      n_kam = 0 ;      /* parametr */
                      o_kla = 4 ;      /* parametr blokady synchronizacji kamer */
               } /* switch */
             }
    if (read_bit ( TI_bit ))  {
         clear_bit ( TI_bit ) ;
         if ( *buf == 0 ) flaw = 0 ;  /* zakonczenie transmisji */
         if (flaw) output ( SBUF , *buf++ );   /* transmisja szeregowa */
         }
      }
```

```
/*************************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                   MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


     Autor :          Karol Najar
     Warszawa         1988.12.14

     ╔═══════════════════════╗
     ‖                       ‖
     ‖   MODULE  INTE_VE     ‖
     ‖                       ‖
     ╚═══════════════════════╝

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


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


extern PST pstve[3] ;
extern void new_al_in(void);   /* obsluga struktury stanow kamer */
extern void obslu(int) ;       /* procedura synchronizacjii kamer */
extern char o_exi, b_int, o_kla, n_kam, mul_k ;    /* parametry */


void inte_ve (void)

 {

  new_al_in ()  ;   /* wywolanie obslugi struktury stanow kamer */
  if ( b_int == 0 ) {
  if (  pstve[n_kam].lix < 11 ) {
     obslu (n_kam) ;
     } else {
          pstve[ n_kam ].lix = 0 ;
          if ( o_kla != 3 ) { o_kla = 4; }
             else  { mul_k = 1 ;}
          o_exi++ ;
          b_int++ ;
       }       /* end else   */
    } /* else b_int = 0 ; bo inicjacja nastepuje poprzez wybor z klawiatury */
}
```

```
/*********************************************************
  Przemyslowy   Instytut   Automatyki  i   Pomiarow
                  MERA - PIAP

     Osrodek    Automatyzacji    Elektrycznej

              02-222   Warszawa
           Aleje Jerozolimskie 202


    Autor :          Karol Najar
    Warszawa         1988.12.14

    ╔═══════════════════════╗
    ‖                       ‖
    ‖    MODULE   CCAM      ‖
    ‖                       ‖
    ╚═══════════════════════╝

*********************************************************/
#include <stdio.h>
#include <io51.h>
#include <kamer.h>


extern char n_kam ;         /* wskaznik numer_kamery  inicjalizacja w module sxkao */
extern char b_int ;         /* wskaznik blokada_przerwan inicjalizacja w module sxkao */
extern char o_exi, o_kla ; /* wskazniki obsluga_zakonczenia_synchronizacji_kamery
                              obsluga_reakcji_na_wcisniecie_klucza_klawiatury
                              inicjalizacja w module sxkao. */
char mul_k ;                /* wskaznik wiele_kamer inicjalizacja w module ccam */


void ccam( o_kla )
int o_kla ;
{
          switch ( o_kla ) {
     case 0:  output ( P1, 0x31 ); /* ustawienie wyboru kamery A   */
              n_kam = 0   ;
              b_int = 0   ;    /* blokada procedury synchronizacji kamery */
              break ;
     case 1:  output ( P1, 0x2A ); /* ustawienie wyboru kamery B   */
              n_kam = 1   ;
              b_int = 0   ;    /* blokada procedury synchronizacji kamery */
              break ;
     case 2:  output ( P1, 0x1C ); /* ustawienie wyboru kamery C   */
              n_kam = 2   ;
              b_int = 0   ;    /* blokada procedury synchronizacji kamery */
              break ;
     case 3:
          switch ( n_kam ) {
          case 0 : if (mul_k) {
                       n_kam = 2 ;
                       mul_k = 0 ;
                     }
                   else {
                   output ( P1, 0x31 ); /* wybor kamery A */
                   b_int = 0 ;
                   }
                   break ;
          case 1 : if (mul_k) {
                       n_kam = 0 ;
                       mul_k = 0 ;
                     }
                   else {
                   output ( P1, 0x2A ); /* wybor kamery B */
                   b_int = 0 ;
                   }
                   break ;
          case 2 : if (mul_k) {
                       n_kam = 1 ;
                       mul_k = 0 ;
                     }
                   else {
                   output ( P1, 0x1C ); /* wybor kamery C */
                   b_int = 0 ;
                   o_kla = 3   ; /* blokada ponownego liczenia */
                   }
                   break ;
          default :
                  output ( P1, 0x38 ); /* wybor kamery N */
                  o_kla = 4   ;
                  n_kam =(char)-1  ;
          }    /* end switch */
     default : b_int = 1 ;
               output ( P1, 0x38 ); /* wybor kamery N */
              n_kam = 0 ;
   }    /* end switch */
}
```

Załącznik 3.

Przykłady wyglądu ekranu monitora przy wykonywaniu programu testującego.

```
[ ? ]0
wybrana kamera 0, srednia wynosi =    70.2
minimum wartosci 70, maximum wynosi = 71
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]1
wybrana kamera 1, srednia wynosi =   147.9
minimum wartosci 147, maximum wynosi = 148
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]2
wybrana kamera 2, srednia wynosi =   126.7
minimum wartosci 127, maximum wynosi = 127
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]
Esc chr: ^], Port: 1, Speed: 2400, Parity: None, Echo: Rem, Type
```

```
[ ? ]0
wybrana kamera 0, srednia wynosi =    60.9
minimum wartosci 60, maximum wynosi = 61
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]1
 U W A G A   za silne SwiatLo  kamery 1
wybrana kamera 1, srednia wynosi =   147.9
minimum wartosci 147, maximum wynosi = 148
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]2
 U W A G A   za sLabe SwiatLo  kamery 2
wybrana kamera 2, srednia wynosi =     6.0
minimum wartosci 6, maximum wynosi = 6
 0 <- kamera A
 1 <- kamera B
 2 <- kamera C

[ ? ]
Esc chr: ^], Port: 1, Speed: 2400, Parity: None, Echo: Rem, Type
```