7101

PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81
ZESPÓŁ AUTOMATYKI ELEKTRONICZNEJ
Pracownia Elektronicznych Testerów

4.10 BE10

Główny wykonawca mgr inż. Jarosław Kowalski

Wykonawcy mgr inż. Jarosław Kowalski
mgr inż. Tadeusz Goszczyński

Konsultant

Nr zlecenia S 1438

Opracowanie mikroprocesorowego kalibratora dla elektronicznej symulacji czujników temperatury

Etap 3:
Dokumentacja oprogramowania

Zleceniodawca Praca statutowa PIAP

Pracę rozpoczęto dnia 2.01.94 **zakończono dnia** 30.06.94

Kierownik Pracowni	Z-ca Dyrektora ds. Badawczo Rozwojowych	Kierownik Zespołu
mgr inż. T. Goszczyński	dr inż. J. Jabłkowski	doc. dr inż. J. Korytkow

---

Praca zawiera:		Rozdzielnik - ilość egz:	
stron	73	Egz. 1	BOINTE
rysunków	4	Egz. 2	ZAE-1
fotografii	—	Egz. 3	ZAE-1
tabel		Egz. 4	ZAE-3
tablic		Egz. 5	
załączników		Egz. 6	

Nr rejestr. 7101

Analiza deskryptorowa

AUTOMATYKA PRZEMYSŁOWA, DOKUMENTACJA OPROGRAMOWANIA

Analiza dokumentacyjna

Dokumentacja zawiera:
specyfikację, opisy programów, instrukcję użytkownika oraz teksty programów w języku źródłowym.

Tytuły poprzednich sprawozdań

UKD

PIAP 41/88 10000

2

Spis treści

strona

1. Specyfikacja

Nazwa	Uwagi
emstup.asm	program inicjalizacji procesora
kalmain.c	program główny
kaltg.c	podprogramy obliczeniowe
kaldisp.c	podprogramy wyświetlania danych`
kconst.h	stałe dla wszystkich programów
kvaria.h	zmienne globalne dla programu głównego
kexvaria.h	zmienne globalne dla innych programów
kaldisp.h	deklaracje do programu kaldisp.c
exdos.h	deklaracjie funkcji we/wy
stroj.h	stałe do strojenia
kaltg.h	deklaracje do programu kaltg.h
ter_el.hl	tabele termoelementów
emkompil.bat	programy narzędziowe
kompiluj.bat	
linkuj.bat	
lokuj.bat	
loc1.cmd	

2. Opis programu

2.1 Informacje ogólne

Program przystosowany jest do pracy w systemie mikroprocesorowym opartym o procesor NEC - V30 zastosowany w systemie BUSMAT2 firmy GURU w module GSM-NCPUV30.

Wymagania sprzętowe: moduł GSM-NCPUV30 wyposażony w układy pamięci RAM 64 kB, oraz pamięci EPROM 64 kB.

Program inicjalizacji procesora emstup.asm napisany został w asemblerze dla procesora Intel 8086 a pozostałe programy w języku "C".

2.2 Przeznaczenie funkcjonalne

Programy przeznaczone są do implementacji w kalibratorze temperatury KAL-400 i umożliwiają procesorowi sterowanie poprzez magistralę poszczególnych układów kalibratora umieszczonych na oddzielnych pakietach .
Programy umożliwiają:

- ustalenie odpowiedniej konfiguracji połączeń wewnętrznych kalibratora (wysterowanie odpowiednich przekaźników) w celu dopasowania do sygnałów symulowanych i mierzonych
- wysterowanie przetwornika cyfrowo-analogowego podczas symulacji sygnałów
- wysterowanie symulatora rezystancji
- wysterowanie kluczy analogowych i przetwornika analogowo-cyfrowego podczas pomiaru sygnałów
- odczyt sygnałów z klawiatury
- wyświetlanie danych na module wyświetlacza LCD

4

- wykonanie przeliczeń wartości symulowanych, mierzonych i błędów

2.3 Opis struktury logicznej

2.3.1 Opis programu głównego

Schemat blokowy programu głównego znajdującego się w zbiorze kalmain.c przedstawiony jest na rys. 1.
Po załączeniu zasilania wykonywany jest program inicjalizacji sprzętu emstup() w którym ustawiane są rejestry pomocnicze na pakiecie procesora, inicjalizowana jest tabela wektorów przerwań oraz inicjalizowane są obszary RAM, w których znajdują się zmienne. Następnie w programie ini() inicjalizowany jest panel wyświetlacza LCD, wypisywana jest winietka, odblokowane jest przerwanie zegarowe 20msek. (do czytania klawiatury), inicjalizowane są zmienne przez wartości odczytane z części pamięci RAM z podtrzymaniem bateryjnym .
W kalibratorze wykorzystywane są dwa przerwania sprzętowe: zegarowe 20msek. do obsługi klawiatury oraz przerwanie od gotowości przetwornika analogowo-cyfrowego.
Wyświetlanie znaków na module LCD następuje przy pomocy odpowiednich podprogramów zamiany liczb na znaki oraz ustawiania pozycji i wyświetlania (podprogramy w kaldisp.c).
Po wyświetleniu dolnej i górnej linii ekranu z danymi zapamiętanymi w rejestrze 00 przed wyłączeniem zasilania następuje wejście w program zmian konfiguracji (programowania parametrów) zmiany(), który jest wykonywany do momentu wciśnięcia przycisku START. Przy zakończeniu programu zmiany() bieżące nastawy zapamiętane są w rejestrze 00.
Następnie wykonywany jest cyklicznie program pomiary(), w którym mierzona jest wartość sygnału na zaciskach pomiarowych oraz możliwa jest zmiana nastawy. Po wykonaniu pomiaru (jeżeli jest to możliwe) obliczany jest błąd wyjścia w stosunku do wejścia.
Po wciśnięciu przycisku stop następuje zatrzymanie pomiarów i ponowne wejście w program zmian konfiguracji .

2.3.2 Opis programu zmian konfiguracji

Schemat blokowy programu zmian konfiguracji zmiany() znajdującego się w zbiorze kaldisp.c przedstawiony jest na rys. 2.
W programie odczytywana jest klawiatura (podprogram klawt1() wykonywany w przerwaniu co 20msek.) i w zależności od położenia kursora (migającej części linii) wykonywane są odpowiednie podprogramy powodujące zmiany parametrów. Wciśnięcie przycisku PRAWO powoduje zmianę położenia kursora i umożliwia zmianę kolejnego parametru przyrostowo poprzez wciskanie przycisków PLUS i MINUS.
Zmiana niektórych parametrów powoduje automatyczne przeliczenia i zmiany innych, zależnych parametrów (podprogramy parametry_pomiaru(), parametry_symulacji(), zadaj() w kaltg.c).
Po wciśnięciu przycisku STO można bieżące dane zapamiętać w rejestrze pod numerem wybranym przez naciskanie PLUS i MINUS (podprogram memory_plus() w kaldisp.c).
Przycisk RCL umożliwia przeglądanie rejestrów z zapamiętanymi uprzednio parametrami i wynikami.
Zmiana parametrów możliwa jest do naciśnięcia przycisku START (główna pętla while() w zmiany()).

2.3.3 Opis programu pomiarów

Schemat blokowy programu pomiarów znajdującego się w zbiorze kalmain.c przedstawiony jest na rys. 3.

Pomiary wykonywane są w przerwaniu od przetwornika a/c (przerwanie oznacza gotowość) w czasie ok.1 sek wyznaczonym przez 30 000 pustych pętli zliczanych przez licznik opóźnienia. W tym czasie są odczytywane i sumowane wartości z przetwornika a/c po 4 razy i liczona jest średnia. Następuje też zmiana położenia kluczy analogowych żeby zmierzyć temperaturę zimnego końca termopary (podprogram przerwania ac_glowny() w kaltg.c). Zliczanie opóźnienia może być przerwane przez naciśnięcie przycisku w celu zmiany nastawy lub zapisania albo odczytu rejestru. Po czasie opóźnienia lub reakcji na wciśnięty przycisk następuje obliczenie wyniku i aktualizacja wyświetlenia na LCD. Zatrzymanie pomiarów następuje po wciśnięciu przycisku STOP (wyniki są zamrożone na wyświetlaczu do momentu jakiejkolwiek zmiany lub ponownego startu).

2.3.4 Opis programu przeglądania wyników

Schemat blokowy programu przeglądania wyników znajdującego się w zbiorze kaldisp.c przedstawiony jest na rys. 4.
Wejście w ten podprogram następuje po wciśnięciu przycisku RCL (podprogram memory_recall() w kaldisp.c). Bieżący numer rejestru ustawiany jest na ostatnio używany lub na 1 jeśli zostało włączone zasilanie. Wyświetlona zostaje zawartość rejestru o bieżącym numerze natomiast nie jest zmieniona nastawa i rzeczywiste parametry. Przy pomocy przycisków PLUS i MINUS można zmieniać numery rejestru, który jest wyświetlany. Program pracuje w pętli (pętla while(1) w podprogramie memory_recall()) do momentu naciśnięcia przycisku STOP - zmiana parametrów na te z rejestru i wejście w programowanie parametrów; przycisku START - zmiana parametrów na te z rejestru i wejście w pomiary; przycisku RCL - powrót do stanu sprzed wejścia w podprogram memory_recall() bez zmiany nastaw i parametrów

2.4 Wykorzystywane środki techniczne

Programy tworzone są przy pomocy komputera PC. Do przetworzenia programów źródłowych wykorzystywane są programy:

- Microsoft C compiler V5.0
- Microsoft Macro Assembler V5.0
- Link&Locate ++ V5.1e

W celu umieszczenia oprogramowania w układach pamięci EPROM typu 27C256 w module procesora GSM-NCPUV30 należy wykonać następujące czynności:

- kompilować program inicjalizacji procesora emstup.asm wywołując program narzędziowy
  -emkompil
- kompilować programy źródłowe dla KAL-400 wywołując program narzędziowy
  - kompiluj kalmain
  - kompiluj kaldisp
  - kompiluj kaltg
- linkować wywołując program narzędziowy
  - linkuj
- lokować wywołując program narzędziowy
  - lokuj
- utworzyć zbiory binarne dla układów EPROM wywołując program narzędziowy
  - kosci

W wyniku powstają zbiory binarne zawierające dane dla programatora układów eprom . Wywołany jest program obsługi programatora EW911 firmy MOMIK. Należy zaprogramować 2 układy pamięci EPROM typu 27C256 umieszczając w nich zbiory proj.evn oraz proj.odd a

następnie umieścić układy w odpowiednich podstawkach w module procesora GSM-NCPUV30.

2.5 Wywołanie programu

Wywołanie programu obsługi kalibratora temperatury KAL-400 następuje automatycznie po włączeniu zasilania kalibratora. Na LCD wyświetlona jest najpierw winietka programu, a następnie kalibrator wchodzi w tryb pracy - programowanie.

2.6 Dane wejściowe

- sygnały wprowadzane z klawiatury kalibratora
- sygnały analogowe doprowadzone do zacisków pomiarowych kalibratora

2.7 Dane wyjściowe

- sygnały wyprowadzone na zaciski symulacji kalibratora
- dane wyświetlane na LCD (rodzaje i zakresy symulacji, rodzaje pomiaru, wartości sygnałów symulowanych i mierzonych, błąd przetwornika)
- dane zapamiętane w pamięci podtrzymywanej bateryjnie (zestawy danych które były wyświetlane na LCD)

3. Instrukcja użytkownika

Obsługa kalibratora następuje poprzez zaprogramowanie parametrów sygnałów nastawianych przy pomocy klawiatury kalibratora i zgodne z tym odpowiednie wykorzystanie zacisków na płycie czołowej. Wszystkie parametry i wartości pokazywane są na wyświetlaczu LCD kalibratora.
Na LCD wyświetlone są 2 linie:

Mem	Rodz. symul.	Zakres symulacji	Wartość nastawy	
01	Pt100	+150..+250°C	224 °C	74.0%
	0..20mA	14.82 mA	+0.10%	
	Rodz. pomiaru	Wartość	Błąd	

Górna linia odpowiada części symulującej kalibratora, a dolna pomiarowej. Oprócz trybu odczytu danych wartości pokazane na wyświetlaczu odpowiadają zawsze wartościom symulowanym. Sygnały na zaciskach, które nie odpowiadają aktualnemu rodzajowi symulacji mają wartość nieokreśloną. Mierzone są sygnały tylko z tych zacisków, które odpowiadają zaprogramowanemu rodzajowi pomiaru.
Na zaciskach oznaczonych "symulacja R" (4 zaciski) symulowana jest rezystancja i czujniki typu Pt ... , na zaciskach "symulacja mA" symulowany jest prąd stały, a na zaciskach "symulacja mV" symulowane jest napięcie stałe i sygnały termoelementów.

Na zaciskach oznaczonych "pomiar mA" mierzony jest prąd, a na zaciskach "pomiar mV, V" mierzone jest napięcie stałe oraz temperatura przy pomocy termoelementu. Do kompensacji zimnego końca termoelementu (CJC) mierzona jest temperatura w pobliżu tych zacisków.

Pomiar temperatury przy pomocy czujników rezystancyjnych wymaga czteroprzewodowego podłączenia czujnika: zasilenia czujnika prądem z zacisków "symulacja mA" i pomiaru spadku

7

napięcia na czujniku poprzez zaciski "pomiar mV, V" (mierzony jest spadek napięcia na czujniku wywołany przepływem symulowanego prądu i odpowiednio przeliczany).

Kalibrator może pracować w 3 trybach:
- programowania
- pomiaru
- zapamiętywania i odczytu danych

3.1 Tryb programowania

Po wciśnięciu przycisku **STOP** (lub po włączeniu zasilania) następuje przejście w tryb programowania rodzaju wyjścia oraz rodzaju wejścia. Programowanie następuje po wybraniu odpowiedniego pola przez naciskanie przycisku → (wartość lub tekst w wybranym polu miga) i przyrostowe wybieranie nastawy w danym polu przy pomocy przycisków + − oraz przycisku przyspieszenia zmian >> (przycisk ten działa tylko w przypadku ustawiania zakresu pomiarowego i zadawania wartości symulowanej).

W kolejnych polach można wybrać:
- **Mem** - numer miejsca w pamięci od 01 do 19 (tylko po uprzednim naciśnięciu przycisku **RCL** lub **STO**)

- **Rodzaj symulacji** - jeden z następujących sygnałów, które mogą być symulowane przez kalibrator:
  - Pt100 - rezystancja
  - Pt500 - rezystancja
  - Pt1000 - rezystancja
  - Rezystancja
  - Termoelement typu J, K, T, S, B - napięcie
  - 0..20 mA
  - 4..20 mA
  - 0..100 mV
  - bez nastawy

- **Zakres symulacji** (zależny od nastawionego rodzaju symulacji - nastawiana górna i dolna granica zakresu)
  Wartości nastawiane zmieniają się o 1 przy pojedynczym naciśnięciu przycisków + − , z kilkukrotnie większy skokiem przy ciągłym naciskaniu przycisków + − lub ze skokiem ok. 10% maksymalnego zakresu przy użyciu przycisku przyspieszenia zmian >> (kierunek zmiany jest zależny od tego czy wcześniej był naciśnięty przycisk + czy przycisk −).
  - dla temperatury (Pt i termoelementy) nastawiana dolna i górna temperatura (co 1 °C)
    Pt100 -140 ... +850 °C
    Pt500 -200 ... +850 °C
    Pt1000 -200 ... +850 °C
    Termoelement typu J -210 ... +1200 °C
    Termoelement typu K -270 ... +1370 °C
    Termoelement typu T -270 ... +1370 °C
    Termoelement typu S -270 ... +1370 °C
    Termoelement typu B -270 ... +1370 °C
  - dla rezystancji 40 .. 4000om (co 1om)
  - dla mA 0..20mA (co 1 mA)

8

- dla mV -100 ... +100mV (co 1 mV)

- **Nastawa symulacji**
  Nastawiana jest wartość w jednostkach odpowiadających wybranemu rodzajowi symulacji (pomiędzy górną i dolną granicą zakresu) z maksymalną możliwą rozdzielczością nastawy przy pojedynczym naciśnięciu przycisków + − , z kilkukrotnie większy skokiem przy ciągłym naciskaniu przycisków + − lub ze skokiem ok. 10% zakresu przy użyciu przycisku przyspieszenia zmian >> (kierunek zmiany jest zależny od tego czy wcześniej był naciśnięty przycisk + czy przycisk −).

- **Rodzaj pomiaru** (rodzaj sygnału mierzonego):
  Sygnały standardowe (wynik pomiaru w odpowiednich jednostkach):
  - 0..20 mA
  - 4..20 mA
  - 0..100 mV
  - 0..5V
  - 0..10V
  - ±100mV
  - ±5V
  - ±10V

  Temperatury (wynik pomiaru w °C):
  - Termoelement typu J, K, T, S, B
  - Termoelement typu J, K, T, S, B + CJC (kompensacja zimnego końca)
  - Pt100 (wejście 10V z wykorzystaniem wyjścia 10mA dokł. 0.2%)
  - Pt500 (wejście 10V z wykorzystaniem wyjścia 2mA dokł. 1%)
  - Pt1000 (wejście 10V z wykorzystaniem wyjścia 1mA dokł. 2%)
  - bez pomiaru

Po zmianie rodzaju symulacji następuje automatyczna zmiana początku i końca zakresu symulacji na wartości graniczne dla danego rodzaju symulacji i ustawienie wartości zadanej na początek zakresu.
Zmiana zakresu symulacji powoduje automatyczną zmianę wartości zadanej na początek zakresu.
Przeliczenia wartości zadawanej rezystancji z temperatury i temperatury ze zmierzonej rezystancji dokonywane jest dla czujników typu Pt na podstawie wzorów zgodnie z PN-83/M-53852 (odpowiednie wzory dla temperatur mniejszych i większych od 0 °C) natomiast przeliczenia dla czujników termoelektrycznych na podstawie tabel zgodnych z tabelami umieszczonymi w normach PN 81/M-53854.02 do 06 przy pomocy interpolacji liniowej.

Pomiar temperatury przy pomocy czujników rezystancyjnych wymaga ustawienia symulacji prądu na odpowiednią wartość i odpowiedniego zasilenia nim czujnika (mierzony jest spadek napięcia na czujniku wywołany przepływem symulowanego prądu). Po wybraniu rodzaju pomiaru "Pt ..." następuje automatyczne ustawienie rodzaju i wartości symulacji.

3.2 Tryb pomiaru

Po naciśnięciu **START** następuje przejście do trybu pomiaru (z jednoczesną symulacją).
W tym trybie można zmieniać wartość zadawanej wielkości wyjściowej przy użyciu przycisków + − oraz przycisku przyspieszenia zmian >>. Do momentu wciśnięcia przycisku **STOP** prowadzone są ciągłe pomiary sygnału wejściowego i jeśli to możliwe obliczany jest błąd.

Błąd jest liczony tylko dla układu symulacja - pomiar takiego jak występuje w przetwornikach. Błąd liczony jest jako różnica wartości zmierzonej w % zakresu wyjściowego przetwornika oraz wartości zadanej (symulowanej) w % zakresu wejściowego przetwornika.

3.3 Tryb zapamiętywania i odczytu danych

Po naciśnięciu przycisku **STO** w trybie pomiaru lub programowania można przypisać dane do miejsca pamięci o wybranym numerze . Zapamiętane zostaną wszystkie wartości widoczne na wyświetlaczu. Ponowne naciśnięcie przycisku **STO** powoduje rezygnację z zapisania danych do pamięci, a przyciśnięcie **START** lub **STOP** powoduje zapisanie nowych wartości do pamięci na miejsce dotychczasowych i wyjście z tego trybu do trybu pomiaru lub programowania rodzajów we/wy i zakresów.

Zmiana wartości i jednocześnie zmiana na zapamiętany zestaw rodzajów we/wy i zakresów poprzez wybór miejsca w pamięci następuje po naciśnięciu **RCL** i programowanie numeru przez naciskanie przycisków + − .Można w ten sposób przeglądać zawartość poszczególnych rejestrów pamięci (rzeczywiste wartości wyjściowe i parametry nie zmieniają się w tym czasie) , a następnie przez ponowne naciśnięcie **RCL** powrócić do poprzedniego trybu pracy bez zmian lub nacisnąć **START** lub **STOP** co spowoduje przejście do pomiaru wg nowego zestawu parametrów i wartości lub przejście do programowania z nowym zestawem parametrów.

Wciśnięcie klawisza **RCL** lub **START** powoduje zapisanie bieżących danych w pamięci pod numerem 00 (w celu umożliwienia powrotu do poprzednich danych)

Po włączeniu do sieci kalibrator ustawia się automatycznie na ostatnio używane parametry i numer miejsca w pamięci 00 .

4. Tekst programu w języku źródłowym

```
;to jest emstup.asm dla kal400
        ;
              NAME MSC_EM_START_UP_CODE       ; emstup.asm, version 5.2.
        ; This file contains sample start-up code for building a Microsoft C
        ; application for an embedded environment.
        ; It also supports the floating point emulator and math functions supplied
        ; with the Microsoft C optimizing compiler package.
        ; It assumes the embedded system does not contain an 8087 coprocessor.
        ;
        ; You need to modify this file to adapt it to your embedded environment.
        ; The places for you to make changes are marked with comment lines filled
        ; with asterisks (*****).
        ;
        ; The primary function of the start-up code is to set up the run-time
        ; environment before passing control to C function main().
        ; The start-up code performs the following functions:
        ; 1) Initialize hardware and check RAM.
        ; 2) Copy initializers from ROM to RAM to setup initialized program variables
        ;    to proper initial values.
        ; 3) Zero all uninitialized program variables.
        ; 4) Initialize interrupt vector table, if necessary.
        ; 5) Setup data segment.
        ; 6) Setup stack segment.
        ; 7) Initialize the floating point emulator.
        ; 8) Pass control to C function main().
        ;
        ; This assembly file is called emstup.asm.  You should use the Microsoft MASM
        ; assembler, version 4 and up.  For example,
        ;       masm /Mx emstup,emstup,emstup,nul
        ; The /Mx option is needed to preserve lower-case in public and external names.
        ;
              PUBLIC  __acrtused
        __acrtused      EQU     1
        ; The symbol __acrtused has to be in lower-case.  Starting from version 4,
        ; when the Microsoft C optimizing compiler compiles a C file, it places an
        ; external reference to this symbol in the object file.  The public definition
        ; of __acrtused is contained in an object module called crt0.  This module is
        ; placed in the Microsoft C run-time library file.  This module contains the
        ; start-up code for the DOS environment.  So when you link your C object files
        ; with the run-time library file, the linker will extract the start-up object
        ; module from the run-time library file to satisfy the external references.
        ; As a result, the start-up code for DOS environment will be included in the
        ; linker output.
        ; If you do not make use of any function supplied in the run-time library
        ; file, you do not link with the library file at all.  Then you have to
        ; include a public definition of __acrtused in this file to resolve all the
        ; external references in the C object files.
        ; Even though you are building an application for an embedded environment,
        ; sometimes you may want to link with the run-time library file.  The reason
        ; is that the run-time library file contains both DOS-dependent functions,
        ; such as printf(), and DOS-independent functions, such as strcpy().  If you
        ; want to make use of certain DOS-independent functions that are contained in
        ; the run-time library file, you would link with the library file.  If the
        ; link map shows that the linked module contains the crt0 module, you know
        ; that you have linked in some DOS-dependent functions from the run-time
        ; library file.
        ; Make sure you specify this start-up file as the first object file
        ; in the linker command line, then the other object files and place the
        ; modified combined run-time library file as the last file.  An object file
        ; called eminit.obj has to be added to the combined run-time library which
        ; is then processed by the utility cv2omf to produce a modified combined
        ; library file in OMF format.  A batch file called emsetup.bat is provided
        ; to process the library file.
        ; Supposed the object file eminit.obj is added to the combined library file
        ; llibce.lib when is then processed by cv2omf to generate a library file
        ; called llibce.ssi.  The following shows a typical linker command line:
```

```
;       xlink86 emstup.obj,main.obj,util.obj,llibce.ssi to proj.lnk
;
; The following memory map shows a typical run-time environment of an embedded
; application developed using the Microsoft C optimizing compiler.
; It will help you understand the start-up code presented in this file.
; The naming convention of class names presented here conforms with the
; Microsoft C optimizing compiler, version 4 and up.  The names enclosed in
; single quotes are class names of segments.
;
;  High address
;  -        ---------------
;  |        |             |
;  | FFFF:0 --------------- <-- Bootstrap code (JMP FAR PTR START)
;  |        :             :
;  |        --------------- <--
;  |        | [FAR_DATA]  | ^  This area of ROM contains initializers that
;  |        --------------- |  are used by start-up routine to initialize
;  |        | [CONST]     | |  segments with these class names in RAM at
;  |        --------------- |  power-up.
;  |        | [DATA]      | |  (The INITDATA control causes the PROM86 v5.2
;  R        --------------- |  utility to place initializers between address
;  O        | [DATA_BEG]  | v  labels _bdata and _edata and between _bfdata
;  M        --------------- <-- and _efdata in this area of ROM)
;  |        | 'CODE_END'  | <---- This class contains only one segment of zero
;  | _etext ---------------      length. Initializers are located at _etext.
;  |        | 'CODE'      | <-- Text segments with class name CODE.
;  | _start ---------------
;  - START  :             :
;           :             :
;  -        :             :
;  |        --------------- <---- End of DGROUP (size of DGROUP <= 64K bytes)
;  |        | 'STACK'     | <-- This class contains the stack.
;  |        ---------------
;  |        | 'BSS_END'   |
;  | _end   ---------------
;  |        | 'BSS'       | <-- This class contains uninitialized data.
;  |        ---------------
;  |        | 'DATA_END'  |
;  | _edata --------------- <----
;  |        | 'CONST'     | ^ These three classes of segments are to be
;  R        --------------- |  initialized with initializers in ROM
;  A        | 'DATA'      | |  by start-up routine at power-up.
;  M        --------------- |
;  |        | 'DATA_BEG'  | v
;  | _bdata --------------- <---- Start of DGROUP
;  |        |'HUGE_BSS_END'|
;  | _ehbss ---------------
;  |        | 'HUGE_BSS'  | <-- This class contains uninitialized data.
;  |        ---------------
;  |        |'HUGE_BSS_BEG'|
;  | _bhbss ---------------
;  |        | 'FAR_BSS_END'|
;  | _efbss ---------------
;  |        | 'FAR_BSS'   | <-- This class contains uninitialized data.
;  |        ---------------
;  |        | 'FAR_BSS_BEG'|
;  | _bfbss ---------------
;  |        |'FAR_DATA_END'|
;  | _efdata---------------
;  |        | 'FAR_DATA'  | <-- This class contains initialized data.
;  |        ---------------
;  |        |'FAR_DATA_BEG'|
;  | _bfdata---------------
;  |        :             :
;  |        ---------------
;  |        |             | <-- The interrupt vector table should be
```


12

```
;  -    0:0 ---------------     initialized by start-up routine at power-up.
;  low address
;
; NOTE:
;       It is important to maintain the order of SEGMENTS/ENDS pairs as shown
;       below.
;
;       You can find out what segments are generated in the object file by the
;       C compiler by linking the object file by itself with linker XLINK86.
;       The map file generated by the linker will show you all the segments
;       in the object file. For example, if the object file is myprog.obj,
;       enter the following command:
;              xlink86 myprog.obj to myprog.lnk
;       The linker will generate a relocatable object file called myprog.lnk
;       and a map file called myprog.mpi.  The map file shows a segment map.
;
;       Text segments all have class name CODE.  They contain program code
;       which is machine instructions generated by the compiler.
;
;       Data segments with class names DATA_BEG, DATA, CONST, BSS and STACK
;       belong to a group named DGROUP.  Since these segments belong to
;       a group, it follows that the total memory space occupied by them
;       cannot exceed 64K bytes.
;
;       The object files may contain data segments with class names
;       FAR_DATA, FAR_BSS and HUGE_BSS.
;       These classes of segments are generated by the Microsoft C
;       optimizing compiler to support 'far' and 'huge' data objects.
;       Check your C manual for details.
;
;       Data segments with class names FAR_BSS and HUGE_BSS contain
;       'far' and 'huge' uninitialized data, respectively. These segments
;       do not belong to any group.  They should be located before
;       the group DGROUP in the RAM space of your target system.  The
;       advantage of locating these segments before DGROUP is that you may
;       use the memory space from the end of STACK segment to the end of
;       RAM as heap to implement your own memory allocation scheme.
;
;       For 'huge' data objects, multiple segments may be generated by the
;       C compiler to hold a single data object that is greater than 64K
;       bytes in size.  These segments must be located together in proper
;       order.
;
;       Data segments with class name FAR_DATA contain 'far' and 'huge'
;       initialized data.  They should also be located before the group
;       DGROUP in the RAM space of your target system.
;
;       When you prepare data file for programming eprom, the initializers
;       for segments with class names DATA_BEG, DATA, CONST and FAR_DATA
;       have to be placed in ROM addresses behind the CODE_END class.
;       The INITDATA control of the PROM86 v5.2 utility performs this
;       operation for you.  At power-up, the start-up routine will copy these
;       initializers from ROM to initialize the appropriate variables in RAM.
;
;       In order to make use of the INITDATA control of PROM86 operation,
;       you must preserve the following public labels in this start-up file:
;       _bfdata - beginning of initializers in FAR_DATA class
;       _efdata - end of initializers in FAR_DATA class
;       _bdata  - beginning of initializers in DGROUP group
;       _edata  - end of initializers in DGROUP group
;       _etext  - end of code
;
;       In addition to the above labels, the routine in this start-up file
;       uses the following public labels:
;       _end    - end of uninitialized data in BSS class
;       _bfbss  - beginning of uninitialized data in FAR_BSS class
```

```
;       _efbss  - end of uninitialized data in FAR_BSS class
;       _bhbss  - beginning of uninitialized data in HUGE_BSS class
;       _ehbss  - end of uninitialized data in HUGE_BSS class

; ****************************************************************************
; Choose the memory model by setting the appropriate label to 1.
; ****************************************************************************
; You may use this start-up code for any memory model.
SMALL   EQU     1       ; 1 => small model
COMPACT EQU     0       ; 1 => compact model
MEDIUM  EQU     0       ; 1 => medium model
LARGE   EQU     0       ; 1 => large model, default setting
HUGE    EQU     0       ; 1 => huge model

; ****************************************************************************
; Choose the target CPU
; CPU86 EQU 1 IF target CPU is 8086 or 8088.
; CPU86 EQU 0 IF target CPU is 80186 or 80188.
; ****************************************************************************
CPU86   EQU     1       ; default setting

; ****************************************************************************
; Specify stack size.
; ****************************************************************************
STACK_SIZE      EQU     400H ; Reserve 1K word of stack or any size you want.

BEGFDATA SEGMENT PARA PUBLIC 'FAR_DATA_BEG'
        PUBLIC _bfdata
_bfdata LABEL BYTE      ; This label marks the beginning of initialized data
                        ; in FAR_DATA class.
BEGFDATA ENDS
;
FAR_DATA_START SEGMENT PARA PUBLIC 'FAR_DATA'
FAR_DATA_START ENDS
; By default, the XLOC86 locator places segments with the
; same class name together.
; The purpose of the FAR_DATA_START segment is to cause the locator
; to locate all segments with class name FAR_DATA that contain
; initialized variables between the BEGFDATA and ENDFDATA segments.
ENDFDATA SEGMENT PARA PUBLIC 'FAR_DATA_END'
        PUBLIC _efdata
_efdata LABEL BYTE      ; This label marks the end of initialized data
                        ; in FAR_DATA class.
ENDFDATA ENDS

BEGFBSS SEGMENT PARA PUBLIC 'FAR_BSS_BEG'
        PUBLIC _bfbss
_bfbss LABEL BYTE       ; This label marks the beginning of uninitialized
                        ; data in FAR_BSS class.
BEGFBSS ENDS
;
FAR_BSS_START SEGMENT PARA PUBLIC 'FAR_BSS'
FAR_BSS_START ENDS
; By default, the XLOC86 locator places segments with the same class name
; together.
; The purpose of the FAR_BSS_START segment is to cause the locator to locate
; all segments with class name FAR_BSS between the BEGFBSS and ENDFBSS
; segments.
; Segments with class name FAR_BSS contain uninitialized variables.
ENDFBSS SEGMENT PARA PUBLIC 'FAR_BSS_END'
        PUBLIC _efbss
_efbss LABEL BYTE       ; This label marks the end of uninitialized data
                        ; in FAR_BSS class.
ENDFBSS ENDS

BEGHBSS SEGMENT PARA PUBLIC 'HUGE_BSS_BEG'
```

```
        PUBLIC _bhbss
_bhbss LABEL BYTE       ; This label marks the beginning of uninitialized
                        ; data in HUGE_BSS class.
BEGHBSS ENDS
HUGE_BSS_START SEGMENT PARA PUBLIC 'HUGE_BSS'
HUGE_BSS_START ENDS
; By default, the XLOC86 locator places segments with the same class name
; together.
; The purpose of the HUGE_BSS_START segment is to cause the locator to locate
; all segments with class name HUGE_BSS between the BEGHBSS and ENDHBSS
; segments.
; Segments with class name HUGE_BSS contain uninitialized variables.
ENDHBSS SEGMENT PARA PUBLIC 'HUGE_BSS_END'
        PUBLIC _ehbss
_ehbss LABEL BYTE       ; This label marks the end of uninitialized data
                        ; in HUGE_BSS class.
ENDHBSS ENDS

DGROUP GROUP NULL,_DATA,CDATA,CONST,HDR,MSG,PAD,EPAD,ENDDATA,_BSS,ENDBSS,STACK

NULL SEGMENT PARA PUBLIC 'DATA_BEG'
; This segment contains 8 bytes of zeros.  If a (DS:0) null pointer assignment
; occurs, these byte locations will be overwritten.  You may implement your
; own routine to check for null pointer assignment.
        PUBLIC _bdata   ; This label marks the beginning of initialized data.
_bdata  LABEL BYTE
        DB 8 DUP (0)
NULL ENDS

_DATA SEGMENT WORD PUBLIC 'DATA'
; Segment with class name DATA contains initialized variables.
_DATA ENDS

CDATA SEGMENT PARA PUBLIC 'DATA' ; MUST BE PARAGRAPH ALIGNED
; Segment contains data for initializing floating point emulator.
CDATA ENDS

CONST SEGMENT WORD PUBLIC 'CONST'
; Segment with class name CONST contains constants.
CONST ENDS

HDR SEGMENT BYTE PUBLIC 'MSG'   ; HEADER SEGMENT OF ERROR MESSAGE STRINGS
HDR ENDS

MSG SEGMENT BYTE PUBLIC 'MSG'   ; ERROR MESSAGE STRINGS
MSG ENDS

PAD SEGMENT BYTE COMMON 'MSG'   ; ERROR MESSAGE PADDING MARKER
PAD ENDS

EPAD SEGMENT BYTE COMMON 'MSG'  ; END OF PADDING MARKER
EPAD ENDS

ENDDATA SEGMENT PARA PUBLIC 'DATA_END'
        PUBLIC _edata
_edata  LABEL BYTE      ; This label marks the end of initialized data.
ENDDATA ENDS

_BSS    SEGMENT WORD PUBLIC 'BSS'
; Segment with class name BSS contains uninitialized variables.
_BSS    ENDS

ENDBSS  SEGMENT WORD PUBLIC 'BSS_END'
        PUBLIC _end
_end    LABEL BYTE      ; This label marks the end of uninitialized data.
ENDBSS  ENDS
```

```
STACK   SEGMENT PARA STACK 'STACK'
        DW STACK_SIZE DUP (?)
stack_top LABEL WORD
STACK   ENDS

;*****************************************************************************
; Place EXTRN statements for interrupt handling routines outside
; of all SEGMENT/ENDS pairs.
;*****************************************************************************
; Interrupt handling routine example for interrupt vector table initialization.
;
;       przerwania  dla TEC300
;
;       EXTRN _int_woda:FAR       ;INT0 INT1 INT2  40,41,42,45,46
;       EXTRN _int_ciep:FAR  rezerwa INT3  43 dla wejscia imp. ciepla
        EXTRN _int_zegar:FAR      ;INT5       45
        EXTRN _int_ac:FAR         ;INT4       44
        EXTRN _int_puste:FAR      ;INT7       47
;
        EXTRN   __eminit:FAR    ;floating point emulator initialization

IF SMALL OR COMPACT
_TEXT SEGMENT WORD PUBLIC 'CODE'
        EXTRN   _main:NEAR      ;C main()
;       EXTRN   _blad_procedure1:NEAR     ;sygnalizacja bladu/0
;       EXTRN   _blad_procedure2:NEAR     ;sygnalizacja bladu math
;       EXTRN   _blad_procedure3:NEAR     ;sygnalizacja bladu int21

ASSUME  CS:_TEXT
ENDIF
IF MEDIUM OR LARGE OR HUGE
        EXTRN   _main:FAR       ;C main()
EMSTUP_TEXT SEGMENT WORD PUBLIC 'CODE'
ASSUME  CS:EMSTUP_TEXT
ENDIF
        ASSUME DS:DGROUP, SS:DGROUP
IF CPU86
        PUBLIC START
START:
ENDIF
        PUBLIC _start
_start:
        CLI
;*****************************************************************************
; Place code here to do hardware initialization and RAM check
;*****************************************************************************
;  inicjalizacja  sterownika przerwan 8259
;
        MOV AX,13H
        OUT 0,AL
        MOV AX,40
        OUT 2,AL
        MOV AX,1DH   ; nested , NOT AEOI
        OUT 2,AL
        MOV AX,0FFH
        OUT 2,AL
;  outp(0, 0x13)   /* ICW1 - EDGE TRIGGERD  */
;  outp(2, 40  )   /* ICW2 - nr przerwania 1-go (decymalnie) */
;  outp(2, 0x0F)   /* ICW4 - SPECIAL MASK + AUTO  END_OF_INTERRUPT */
;  outp(2, 0xFF)   /* OCW1 - WSZYSTKIE PRZERWANIA ZABLOKOWANE (przed czy po OCW3)   */
;  WYCIETY 0x68)   /* OCW3 - SPECIAL MASK MODE */
;                  /* moze potrzebny tylko do zmian trybu w czasie pracy   */

; Perform variable initialization.  Initializers are copied from ROM to RAM.
;
```

```
; Your program will contain segments with class names DATA_BEG, DATA and CONST
; which contain initialized variables.  Since these variables may be modified
; at run-time, the segments containing them have to be located in RAM space.
; When the main function of your C program takes control, it expects
; all variables in these classes of segments to be initialized.
; For embedded systems, the initializers for these variables have to be placed
; in ROM space which are copied to the segments in RAM space by the start-up
; routine at power-up.
; When you use PROM86 v5.2 utility to prepare an output file for burning an
; eprom, you need to specify the INITDATA control.  The INITDATA control
; causes PROM86 v5.2 to place all initializers at address _etext in the output
; file.  The start-up routine assumes these initializers are placed behind
; program code in ROM and copy them to RAM at power-up.
        PUBLIC _init_begin
_init_begin:
        CLD
;256H - ilosc bajtow na zmienne zapamietane w CMOS RAM =
;offset zmiennej glupoty
;zmienne zadeklarowane w varia.h jako inicjowane zeby byly umieszczone
;na poczatku segmentu DATA - 8 bajtow za _bdata. 130 dla IEC300
;wynika z mapy mp2, 256 dla KAL400
;8 bajtow segmentu BEG_DATA i 20 bajtow na zmienne mimo ze maja dlugosc
;129 bajtow
;te zmienne to pam_nr_typu i tablica typy_stale[20][6]
;przy wiekszej ilosci typow bedzie np.typy_stale[99][6] i trzeba sprawdzic
;na mapie mp2 ile zajmuja
;transfer Count zmniejszony o 130 i offset ladowany do DI i SI zwiekszony
;o 130 czyli tyle ile zajmuja zmienne ktore nie beda inicjalizowane ani
;na wartosc ani na 0
;       Transfer Count
        MOV AX,OFFSET DGROUP:_edata     ; Transfer counter
        SUB AX,256H
        CMP AX,0
        JZ no_init_data
        MOV CX,AX
;       Destination
        MOV AX,SEG _bdata
        MOV ES,AX           ; Destination ES:[DI]
        MOV DI,256H         ; Start of initialized variable area in RAM
;       Source
        MOV AX,SEG _etext   ; Source DS:[SI]
        MOV DS,AX           ; Start of initializer storage in ROM
        MOV SI,256H
;       Begin BYTE transfer
REP     MOVSB               ; Begin byte transfer from ROM to RAM
no_init_data:
;   Clear uninitialized data area in DGROUP group
        MOV CX,OFFSET DGROUP:_end ; End of 'BSS' class in RAM
        MOV DI,OFFSET DGROUP:_edata ; Start of 'BSS' class in RAM
        SUB CX,DI           ; Size of 'BSS' class in bytes
        JCXZ no_uninit_data
        MOV AX,0            ; Initialize to 0
REP     STOSB
no_uninit_data:
;   Initialize FAR_DATA data in RAM with initializers stored in ROM

;       Transfer Count
        MOV AX,SEG _bfdata

        MOV CX,SEG _efdata
        SUB CX,AX           ; Compute size of FAR_DATA segments in paragraphs
        JCXZ loopend        ; No FAR_DATA class
        MOV DX,CX           ; Saves transfer count in paragraphs
;       Destination
        MOV ES,AX           ; Destination ES:[DI]
        MOV DI,0            ; Start of FAR_DATA class in RAM
```

```
;       Source
        MOV AX,SEG _etext    ; Source DS:[SI]
        MOV DS,AX            ; Start of FAR_DATA initializer storage in ROM
        MOV SI,OFFSET DGROUP:_edata     ; _edata is paragraph aligned
;       Normalize Source Pointer
        MOV AX,SI            ; Process base of source pointer
        MOV CL,4
        SHR AX,CL            ; Divide by 16
        MOV BX,AX
        MOV AX,DS
        ADD AX,BX
        MOV DS,AX            ; Adjust base of source pointer
        MOV SI,0             ; Offset of source pointer is zero
        MOV AX,DX            ; Restore transfer count in paragraphs
loopbegin:
        CMP AX,1000H         ; More than 64K bytes to transfer?
        JBE lastxfer         ; No
        MOV CX,8000H         ; Prepare to transfer 8000H words
        SUB AX,1000H         ;
        JMP SHORT xferbegin
lastxfer:
        MOV CL,3
        SHL AX,CL            ; Number of WORDs = paragraph * 8
        MOV CX,AX            ; Set up transfer count in terms of WORDs
        MOV AX,0             ; No more to transfer
xferbegin:
REP     MOVSW                ; Transfer WORDs from ROM to RAM
        CMP AX,0             ; Any more data to transfer?
        JE loopend           ; No
        ; Adjust Source and Destination pointers
        MOV BX,AX            ; Saves transfer count
        MOV AX,DS
        ADD AX,1000H
        MOV DS,AX
        MOV AX,ES
        ADD AX,1000H
        MOV ES,AX
        MOV SI,0
        MOV DI,0
        MOV AX,BX            ; Restores transfer count
        JMP loopbegin
loopend:

;   Clear uninitialized data area in FAR_BSS class
;       Transfer Count
        MOV AX,SEG _bfbss
        MOV CX,SEG _efbss
        SUB CX,AX            ; Compute size of FAR_BSS segments in paragraphs
        JCXZ loopfend        ; No FAR_BSS class
;       Destination
        MOV ES,AX            ; Destination ES:[DI]
        MOV DI,0             ; Start of FAR_BSS class in RAM
;       Transfer Count
        MOV AX,CX
loopfbegin:
        CMP AX,1000H         ; More than 64K bytes to initialize?
        JBE lastfxfer        ; No
        MOV CX,8000H         ; Prepare to transfer 8000H words
        SUB AX,1000H         ;
        MOV BX,AX            ; Saves transfer count
        JMP SHORT xferfbegin
lastfxfer:
        MOV CL,3
        SHL AX,CL            ; Number of WORDs = paragraph * 8
        MOV CX,AX            ; Set up transfer count in terms of WORDs
        MOV AX,0             ; No more to transfer
```

```
        MOV BX,AX           ; Saves transfer count
xferfbegin:
        MOV AX,0
REP     STOSW               ; Initialize WORDs to zero
        MOV AX,BX           ; Restore transfer count
        CMP AX,0            ; Any more data to transfer?
        JE loopfend         ; No
        ; Adjust Destination pointers
        MOV AX,ES
        ADD AX,1000H
        MOV ES,AX
        MOV DI,0
        MOV AX,BX           ; Restore transfer count
        JMP loopfbegin
loopfend:

;   Clear uninitialized data area in HUGE_BSS class
;       Transfer Count
        MOV AX,SEG _bhbss
        MOV CX,SEG _ehbss
        SUB CX,AX           ; Compute size of HUGE_BSS segments in paragraphs
        JCXZ loophend       ; No HUGE_BSS class
;       Destination
        MOV ES,AX           ; Destination ES:[DI]
        MOV DI,0            ; Start of HUGE_BSS class in RAM
;       Transfer Count
        MOV AX,CX
loophbegin:
        CMP AX,1000H        ; More than 64K bytes to initialize?
        JBE lasthxfer       ; No
        MOV CX,8000H        ; Prepare to transfer 8000H words
        SUB AX,1000H        ;
        MOV BX,AX           ; Saves transfer count
        JMP SHORT xferhbegin
lasthxfer:
        MOV CL,3
        SHL AX,CL           ; Number of WORDs = paragraph * 8
        MOV CX,AX           ; Set up transfer count in terms of WORDs
        MOV AX,0            ; No more to transfer
        MOV BX,AX           ; Saves transfer count
xferhbegin:
        MOV AX,0
REP     STOSW               ; Initialize WORDs to zero
        MOV AX,BX           ; Restore transfer count
        CMP AX,0            ; Any more data to transfer?
        JE loophend         ; No
        ; Adjust Destination pointers
        MOV AX,ES
        ADD AX,1000H
        MOV ES,AX
        MOV DI,0
        MOV AX,BX           ; Restore transfer count
        JMP loophbegin
loophend:
        PUBLIC _init_done
_init_done:
; ***************************************************************************
; Initialize the interrupt vector table here
; ***************************************************************************
; Interrupt types 0 to 4 are dedicated internal interrupts.
; Type 0 - Divide-error
; Type 1 - Single-step
; Type 2 - Non-maskable interrupt
; Type 3 - 1-byte INT instruction or Breakpoint
; Type 4 - Overflow
; Interrupt types 5 to 31 are reserved internal interrupts.
```

```
; Interrupt types 32 to 255 are available for use.
;
; For example:
;   The interrupt handling routine for vector 32 decimal is assumed to
;   be the C function: void interrupt far int_hdlr().
;   The statement declaring code label _int_hdlr as an external far procedure
;   has to be placed outside of all SEGMENT/ENDS pairs.
;   See the sample EXTRN statement above.  It is behind the GROUP statement.
;   Below is the sample code to initialize an entry in the vector table:
;
;    przerwanie puste
;
        TYPE40 EQU 40
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE40*4,AX   ;Offset portion of vector 40
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE40*4+2,AX ;Base portion of vector 40
;    przerwanie puste
;
        TYPE41 EQU 41
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE41*4,AX   ;Offset portion of vector 41
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE41*4+2,AX ;Base portion of vector 41
;    przerwanie wodomierz bez filtra dla TEC300
;
        TYPE42 EQU 42
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE42*4,AX   ;Offset portion of vector 42
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE42*4+2,AX ;Base portion of vector 42
;    przerwanie zarezerwowane dla wejscia imp. ciepla
;
        TYPE43 EQU 43
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE43*4,AX   ;Offset portion of vector 43
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE43*4+2,AX ;Base portion of vector 43
;    przerwanie ac dla TEC300
;
        TYPE44 EQU 44
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_ac
        MOV ES:WORD PTR TYPE44*4,AX   ;Offset portion of vector 44
        MOV AX,SEG _int_ac
        MOV ES:WORD PTR TYPE44*4+2,AX ;Base portion of vector 44
;    przerwanie zegarowe dla tec300
        TYPE45 EQU 45
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_zegar
        MOV ES:WORD PTR TYPE45*4,AX
        MOV AX,SEG _int_zegar
        MOV ES:WORD PTR TYPE45*4+2,AX
;   nie uzywane 46 i 47
        TYPE46 EQU 46
        MOV AX,0
```

```
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE46*4,AX
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE46*4+2,AX
;   tu wejscie przy zakloceniu 8259 - za krotkie przerwanie itp.
        TYPE47 EQU 47
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET _int_puste
        MOV ES:WORD PTR TYPE47*4,AX
        MOV AX,SEG _int_puste
        MOV ES:WORD PTR TYPE47*4+2,AX
;
; Setup vector for INT 0
;
        TYPE0 EQU 0
        MOV AX,0
        MOV ES,AX       ;Reference base of interrupt vector table
        MOV AX,OFFSET __cintDIV
        MOV ES:WORD PTR TYPE0*4,AX   ;Offset portion of vector 0
        MOV AX,SEG __cintDIV
        MOV ES:WORD PTR TYPE0*4+2,AX ;Base portion of vector 0

; Setup data and stack segment here

        MOV AX,DGROUP
        MOV DS,AX           ; Setup data segment
        ASSUME DS:DGROUP
        MOV SS,AX           ; Setup stack pointer
        MOV SP,OFFSET DGROUP:STACK_TOP
        ASSUME SS:DGROUP
        STI                 ; Enable interrupt now
; CALL FLOATING POINT EMULATOR INITIALIZATION FUNCTION

        CALL __eminit

; CALL C main() FUNCTION

        CALL _main
        NOP

; C PROGRAM EXIT
; DEFAULT ACTION IS TO HALT THE PROCESSOR.
        PUBLIC _exit, __exit
_exit   LABEL FAR
        NOP
__exit  LABEL FAR
        NOP
; TRAP FOR MISSING FLOATING-POINT SOFTWARE
; THE FLOATING POINT INITIALIZATION ROUTINE (FPMATH) WILL CALL THIS ROUTINE
; WHEN ONE OF THE FOLLOWING CONDITIONS OCCURS:
; (1) 8087 FLOATING POINT LIBRARY (87.LIB) IS LINKED IN BUT NO 8087 COPROCESSOR
;     IS PRESENT, THAT IS, FLOATING POINT EMULATOR LIBRARY IS NOT LINKED.
; (2) FLOATING POINT I/O CONVERSIONS ARE DONE, BUT NO FLOATING-POINT VARIABLES
;     OR EXPRESSIONS ARE USED IN THE PROGRAM.
; DEFAULT ACTION IS TO HALT THE PROCESSOR.
        PUBLIC __fptrap
__fptrap LABEL FAR
        NOP
        HLT

; ***  ERROR HANDLING ***
;
; ENTRY POINT TO HANDLE FLOATING POINT ERROR SIGNAL,
; SUCH AS OVERFLOW, DIVISION BY ZERO, OR INVALID OPERATION.
```

```
; DEFAULT ACTION IS TO HALT THE PROCESSOR.
        PUBLIC __fpsignal
__fpsignal PROC FAR
;       CALL _blad_procedure2
        HLT
        RET
__fpsignal ENDP
;
; INT 21H CALLED. REGISTER AH CONTAINS THE FUNCTION CODE.
; IF FUNCTION CODE IS 00H, 25H, 35H OR 4CH, THE INTERRUPT HANDLER
; PROVIDED IN EMINIT.OBJ WILL PROCESS THE CALL.
; FOR ALL OTHER FUNCTION CODES, THE INTERRUPT HANDLER WILL JUMP HERE.
; DEFAULT ACTION IS TO HALT THE PROCESSOR.
        PUBLIC __doscalled
__doscalled LABEL FAR
;
; IF YOU WANT TO IGNORE THE DOS CALL,
; REPLACE THE
;       CALL _blad_procedure3
        HLT
; INSTRUCTION WITH:
;       EXTRN __ignored:FAR
;       JMP FAR PTR __ignored
; THIS WILL CAUSE THE INTERRUPT HANDLER TO IGNORE THE DOS CALL.
; __ignore IS AN ENTRY POINT BACK TO THE INTERRUPT HANDLER.
;
;
; VARIABLE _errno:
;
; FOR CERTAIN C RUN-TIME FUNCTIONS, WHEN AN ERROR CONDITION OCCURS WITHIN THE
; FUNCTION, AN ERROR CODE WILL BE PLACED IN THE _errno GLOBAL VARIABLE.
; IF A FUNCTION SETS THE _errno VARIABLE UPON ERROR, ITS REFERENCE PAGE WILL
; EXPLICITLY MENTION THE _errno VARIABLE.
; ALL OF THE ERROR CODES ARE DESCRIBED IN APPENDIX A: ERROR MESSAGES OF THE
; MICROSOFT C RUN-TIME LIBRARY REFERENCE MANUAL.
; THE VALUES OF THESE ERROR CODES ARE LISTED IN THE errno.h INCLUDE FILE.
;
;
; FUNCTION matherr:
;
; YOU MAY SUPPLY YOUR OWN VERSION OF matherr FUNCTION IN YOUR C PROGRAM.
; IF YOU DO, YOU CAN OBTAIN A VALUE FROM THE type FIELD OF THE exception
; DATA STRUCTURE WHICH CORRESPONDS TO THE MATH ERROR CODE LISTED IN THE
; math.h INCLUDE FILE.  THE MICROSOFT C RUN-TIME LIBRARY REFERENCE
; MANUAL CONTAINS A DETAIL DESCRIPTION OF THE matherr FUCNTION AND THE
; MATH ERROR CODES.
;
; IF YOU DO NOT LINK IN YOUR OWN matherr FUNCTION, THE matherr FUNCTION
; INCLUDED IN THE MICROSOFT C RUN-TIME LIBRARY WILL BE LINKED IN.
; THE FUNCTION SIMPLY RETURNS A ZERO VALUE.
;
; IF YOU LINK IN YOUR OWN VERSION OF THE matherr FUNCTION, IT MAY PERFORM
; SPECIAL ERROR HANDLING.  IF CORRECTIVE ACTION IS TAKEN AND THE
; THE RETURN VALUE SHOULD BE NONZERO.
;
;
; PROCEDURE __NMSG_WRITE:
;
; THE __NMSG_WRITE PROCEDURE WILL BE CALLED WHEN AN ERROR CONDITION OCCURS
; WITHIN IN A MATH FUNCTION AND CERTAIN C RUN-TIME FUNCTIONS.
; IT SEARCHES THE MSG SEGMENT FOR THE ADDRESS OF A MESSAGE STRING CORRESPONDING
; TO THE ERROR CONDITION.
; IF A MESSAGE STRING IS FOUND, DS:DX = STRING ADDRESS, CX = STRING LENGTH.
; YOU MAY PROCESS OR IGNORE THE ERROR MESSAGE STRING.
;
        PUBLIC __NMSG_WRITE
```

```
IF SMALL OR COMPACT
__NMSG_WRITE PROC NEAR
ENDIF
IF MEDIUM OR LARGE OR HUGE
__NMSG_WRITE PROC FAR
ENDIF
        PUSH BP
        MOV BP,SP
        PUSH    DS
        POP     ES
IF SMALL OR COMPACT
        MOV DX,WORD PTR [BP+4]  ; DX = MESSAGE NUMBER
ENDIF
IF MEDIUM OR LARGE OR HUGE
        MOV DX,WORD PTR [BP+6]  ; DX = MESSAGE NUMBER
ENDIF
        ASSUME DS:DGROUP
        MOV     SI,OFFSET DGROUP:MSG  ; start of near messages
TLOOP:
        LODSW                   ; AX = CURRENT MESSAGE NUMBER
        CMP     AX,DX
        JE      FOUND           ; FOUND ERROR MESSAGE STRING
        INC     AX
        XCHG    AX,SI
        JZ      NOTFOUND        ; AT END AND ERROR MESSAGE STRING NOT FOUND
        XCHG    DI,AX
        XOR     AX,AX
        MOV     CX,-1
        REPNE   SCASB           ; SKIP UNTIL 0H
        MOV     SI,DI
        JMP     TLOOP           ; TRY NEXT ENTRY

FOUND:
        XCHG    AX,SI           ; SI = OFFSET TO STRING ADDRESS
        XCHG    DX,AX           ; DS:DX = STRING ADDRESS
        MOV     DI,DX           ; DETERMINE LENGTH OF MESSAGE STRING
        XOR     AX,AX           ; STRING IS TERMINATED WITH BYTE 0H
        MOV     CX,-1
        REPNE   SCASB           ; ES = DS ALREADY
        NOT     CX
        DEC     CX              ; CX = STRING LENGTH
; MAY INCLUDE USER-DEFINED CODE HERE TO OUTPUT ERROR MESSAGE
; DS:DX = STRING ADDRESS, CX = STRING LENGTH
NOTFOUND:
        MOV SP,BP
        POP BP
        RET
__NMSG_WRITE ENDP
;
;
; PROCEDURE __cintDIV:
;
        PUBLIC __cintDIV
__cintDIV PROC FAR      ; DIVIDE BY 0 INTERRUPT HANDLER
;       CALL _blad_procedure1
        HLT             ; USER-DEFINED ERROR RECOVERY ROUTINE
__cintDIV ENDP

IF SMALL OR COMPACT
_TEXT   ENDS
ENDIF
IF MEDIUM OR LARGE OR HUGE
EMSTUP_TEXT     ENDS
ENDIF

C_ETEXT SEGMENT PARA PUBLIC 'CODE_END'
```

```
        PUBLIC _etext
_etext  LABEL BYTE      ; This label marks the end of program code.
; If you specify the INITDATA control in the PROM86 v5.2 utility, it will
; cause PROM86 v5.2 to place all initializers in ROM starting at this
; location.  The start-up routine assumes these initializers are placed
; behind program code in ROM and copy them to RAM at power-up.
C_ETEXT ENDS

IF CPU86 EQ 0   ; Includes only if CPU is 80186 or 80188
; ****************************************************************************
; This sample segment expands the upper memory chip select and performs a jump
; to the initialization code.
; This segment must be located in the top 1K to insure that the ROM remains
; selected in the 80186 system at power-up and reset.
; For more information on setting the chip select of 80186, refer to:
; Application Notes AP-186, "Introduction to the 80186" in Microsystem
; Components Handbook, Microprocessors Volume I, Intel Corp, 1986.
; ****************************************************************************
UMCS_REG EQU 0FFA0H     ; Address of UMCS register
UMCS_VAL EQU 0F038H     ; 64K, no wait states
INIT_HW SEGMENT AT 0FFF0H       ; Absolute address FFF0H:0000H
        ASSUME CS:INIT_HW
        PUBLIC START
START:
        MOV DX,UMCS_REG
        MOV AX,UMCS_VAL
        OUT DX,AX
        JMP FAR PTR _start
INIT_HW ENDS
ENDIF

; ****************************************************************************
; For embedded system applications, your application is burned into one
; or more eprom.  In order to boot your system at power-up or reboot at
; hardware reset, you must place an opcode equivalent to a far jump
; instruction at absolute address FFFF0H.  The CPU will execute
; this instruction upon a hardware reset or power-up.  The instruction
; jumps to the system initialization routine.  This piece of code is called
; bootstrap code.
;
; There are two ways to include the bootstrap code in your application:
;
; (1) Specify the BOOTSTRAP control in the XLOC86 locator command line,
; such as:
; xloc86 proj.lnk to proj.abs bootstrap noinitcode
;
; If you look at the locator's map file (proj.mp2 in this example), an
; UNNAME segment is generated by the locator at address 0FFFF0H which
; contains the bootstrap code.  The bootstrap code is equivalent to a
; far jump instruction to the label START.
;
; or
;
; (2) Specify an absolute segment in this source file, such as:
 R_STER EQU 12H
 ADR_STER EQU 01H

 BOOTSTRAP SEGMENT AT 0FFFFH
        CLI
        MOV AX,R_STER           ;init rejestr ster NCPUV30
        OUT ADR_STER,AL
        JMP FAR PTR START
 BOOTSTRAP ENDS
;
; NOTE:
;       The bootstrap code is not needed when you are debugging with a
```


24

```
;       debugger, such as SoftProbe II/TX target debugger or an in-circuit
;       emulator.  You add this code when you are ready to burn eproms.
;
;*******************************************************************************
;
        END START       ; Make sure you include the START symbol here.
```

```
//*  kalmain.c   */
/* przerwania:   4 - ac
   5 - klaw   reszta - puste */
      #include "exdos.h"
      #include <math.h>
      #include "kconst.h"
      #include "kvaria.h"
      #include "kaldisp.h"
      #include "kaltg.h"
/* czestotliwosc migania kursora */

#pragma intrinsic(sqrt,inp,outp)
/* stale do sterowania przerwaniami */
#define        EN_AC4      0xFF - 16
#define        DIS_AC4     16
#define        EN_KL5      0xFF - 32
#define        DIS_KL5     32


/*-------------------------------------------------------------*/
void  ini(void)
{
  int i,j;

    /* tutaj sa odblokowane przerwania procesora ale zablokowane 59 */
    stan_przerwan=0xff;   /* 1 blokuje */

    ini_8255();
    initlcd();
/*
    if((pam_nr_mem >= IL_MEM_MAX)||(pam_nr_mem<0))
      pam_nr_mem=0;

    nr_mem= pam_nr_mem;
*/
    nr_mem=1;
    wczytaj_mem(0);

    if(rodzaj_pom >= IL_RODZ_POM)
     rodzaj_pom=0;
    if(rodzaj_sym >= IL_RODZ_SYM)
     rodzaj_sym=0;
    parametry_pomiaru();
    parametry_symulacji();

    if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
     obliczona_nastawa(p_sym);
     dolne_r=nastx;
     obliczona_nastawa(k_sym);
     gorne_r=nastx;
     obliczona_nastawa(nastawa);
     rez=nastx;
     }
     else {
      obliczona_nastawa(p_sym);
      dolne_mV=nastx;
      obliczona_nastawa(k_sym);
      gorne_mV=nastx;
      obliczona_nastawa(nastawa);
      mV=nastx;
     }
    zadaj();
    ac_start();
    display_restart();

 outp(2,stan_przerwan=(stan_przerwan & EN_KL5));    /* odblokuj tez klawiature */
```

```
}

/*-------------------------------------------------------------*/
int   pomiary()
{
unsigned int i,j;

       zadaj();
       disp_gorna();
       disp_dolna();

       /*============================*/
       while( STop == 0 )    {   /* GLOWNA  PETLA  POMIAROWA  */

       /* tutaj wykonywane pomiary w przerwaniu */
        stan_przerwan=(stan_przerwan & EN_AC4);
        outp(2,stan_przerwan);  /* odblokuj ac */
        for(i=0;i<30000;i++){   /* martwa petla miedzy wyswietleniami */
         if(PLus||MInus||STop||RCl||STo||FAst)
          break;
        }
        stan_przerwan=(stan_przerwan|DIS_AC4);
        outp(2,stan_przerwan);   /*   OCW3 blokuj ac */
        if(STo){
           memory_plus();
        }
        if(RCl){
           if(memory_recall()>=1){
            if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
             obliczona_nastawa(p_sym);
             dolne_r=nastx;
             obliczona_nastawa(k_sym);
             gorne_r=nastx;
             obliczona_nastawa(nastawa);
             rez=nastx;
            }
            else {
             obliczona_nastawa(p_sym);
             dolne_mV=nastx;
             obliczona_nastawa(k_sym);
             gorne_mV=nastx;
             obliczona_nastawa(nastawa);
             mV=nastx;
            }

            zadaj();
           }
           disp_t(0);
        }

        if( PLus || MInus || FAst )  {
         if(rodzaj_pom > 18){  /* jesli pomiar Pt to nie ma zmian nastawy */
           PLus=0; MInus=0; FAst=0;
         }
         else
         {
         /* zmiana nastawy */
         outp(2,stan_przerwan=(stan_przerwan | DIS_KL5));  /* zablokuj*/
         zmien_nastawe();
         if(APlus==0) PLus=0;
         if(AMinus==0) MInus=0;
         FAst=0;
         zadaj();
         outp(2,stan_przerwan=(stan_przerwan & EN_KL5));    /* odblokuj */
         }
```

```
        }
        oblicz_pomiar();   /* aktualizacja wynikow */
        disp_wynik();
       } /*   while STop */

     /*================== odblokowanie dodane zeby po STOP juz dzialal AC */
/*       stan_przerwan=(stan_przerwan & EN_AC4);
       outp(2,stan_przerwan);
*/
       /*==================*/
       return(0);
}

/*---------------------------------------------------------*/
/* przerwanie puste */
void interrupt far int_puste()    /* INT7 */
{
        _enable();
        outp(0,0x20);   /* EOI  */
}
/*---------------------------------------------------------*/
/* przerwanie zegarowe */
void interrupt far int_zegar()    /* INT5 */
{
 _enable();     /* umozliwienie starszych przerwan */
 klawt1();
        outp(0,0x20);   /* EOI  */
}

/*-------------------------------------------------*/
main()
{
  ini();
  while(1){
             zapal_stop();
             disp_gorna();
             disp_dolna();
             /* tu bylo STOP */
             zmiany();
             zapal_start();
             /* po START */
             while(STop==0)
             {
               pomiary();
             }
          }

  return(0);
}
```

```
/* kaltg.c  30.06.94   14-19.04.94 */

      #include <math.h>
      #include "kconst.h"
      #include "kexvaria.h"
      #include "ter_el.h"
      #include "exdos.h"
      #include "stroj.h"


#define ROZDZ_CA        5000.0/4095.0
#define ROZDZ_R400      0.0125
#define ROZDZ_R4000      0.125

#define   AC_ADRES      0x0004
#define   IL_WEJ        2
#define   EN_AC4        0xFF-16
#define   BASE_55       0x0014          /* 8255 */
#define   KLAWIATURA    BASE_55         /* Port A */
#define   ZRMLODY       BASE_55 + 1     /* Port B */
#define   ZRSTARY       0x0018          /* 374 */
#define   CYFRA_ANALOG  AC_ADRES + 4

void  klawt1();
int ac_glowny(void);
void interrupt far int_ac();      /* INT4 */
void ac_start();
double zmien_procenty(int,double,double,double);
int zmien_proc(void);
int zmien_int(void);

int  zadaj_termoelement(void);
int zadaj_ptXXX(void);
double Pt_ujemna(double);
int zadaj_rezystancje(void);
int  zadaj_prad(void);
int oblicz_pomiar(void);
int parametry_pomiaru(void);
int parametry_symulacji(void);
int zadaj(void);



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

#define    MASKA_WYLPOM 143  /* 255-16-32-64   &  zeruje  */
#define    MASKA_PRAD   16
#define    MASKA_mV     32
#define    MASKA_VOLT   64
#define    MASKA_SYMmV 128
#define    MASKA_ZNAK    2
#define    az   0.00390802
#define    bz  -0.0000005802
#define    cz   -4.2735e-12

double CnaBit=0.0125; /* 50C = 4000bit = 1000mV_we */

/* STROJ  double  wyn_stroj, temp_stroj; */

int znak=1;

int ilosc_ac, il_pom;
unsigned char ac_ml[IL_WEJ], ac_st[IL_WEJ];
unsigned int ac[IL_WEJ] ;
double sum_ac[IL_WEJ], sred_ac[IL_WEJ];
```

```
unsigned char ac_ster=0, ac_poz=0;

int zad_int;
double bit;

unsigned char  ml_bajt_rez, st_bajt_rez;
double rz;

double sym_wsp_elektr;

double pom_wsp_elektr;   /* BurrBrown 100/5000; 10/5000; 20/5000 */
double zero_wyvolt,koniec_wyvolt;  /* dane zakresu przetwornika */

int tab_temp[200];
long int tab_wart[200];
unsigned char  st_bajt_ca,  ml_bajt_ca;


/*-------------------------------------------------*/

void  klawt1()
{
       klaw = ~inp(KLAWIATURA); /* ~ zmienia kazdy bit 0 na 1 i 1 na 0*/
       /* pierwszy odczyt klaw=0 i ustawiaja sie Z_flagi=1 */
       AStop = klaw & 2;
       if(AStop && ZStop) { ZStop=0; STop=1;}   /* AStop chce wpisac STop=1 */
       if(AStop==0)       ZStop=1;   /* ZStop bylo STop=0 mozna wpisac nowy STop=1 */
       AStart= klaw & 1;
       if(AStart && ZStart) { ZStart=0;  STart=1;}
       if(AStart==0)        ZStart=1;
       ARcl = klaw &32;
       if(ARcl && ZRcl)  { ZRcl=0; RCl=1; }
       if(ARcl==0)        ZRcl=1;
       ASto= klaw &16;
       if(ASto && ZSto) { ZSto=0;  STo=1;}
       if(ASto==0)        ZSto=1;

       APlus = klaw & 4 ;
       if(APlus && znak==1)       ++czas_przyrostu;
       if(APlus && ZPlus)    { ZPlus=0;  Plus=1; znak=+1; } /* ostatni plus czy minus */
       if(!APlus && !ZPlus) { DOdalem=0; czas_przyrostu=0; } /* bo APlus = 0  gdy minus wcisniety */
       if(APlus==0)     ZPlus=1;

       AMinus= klaw & 8 ;
       if(AMinus && znak==-1)       ++czas_przyrostu;
       if(AMinus && ZMinus) { ZMinus=0;  MInus=1; znak = -1; }
       if(!AMinus && !ZMinus) { DOdalem=0; czas_przyrostu=0;} /* koniec wciskania */
       if(AMinus==0)        ZMinus=1;

       AFast  = klaw & 64;
       if(AFast && ZFast)  { ZFast=0; FAst=1; }
       if(AFast==0)   {   ZFast=1; }
       APrawo= klaw & 128;
       if(APrawo && ZPrawo) { ZPrawo=0;  PRawo=1;}
       if(APrawo==0)        ZPrawo=1;
}

/*************************************************************/
/* UWAGA  zmieniam ac_ster
bylo przedtem ze    tylko wysyla ac_ster | ac_poz  + 8 */
/* ac_ster ustawiany jest w funkcjach ustawiania przekaznikoow i znaku WY */
/*-----------------------------------------------------------*/
int ac_glowny(void)
  {
    if (++ilosc_ac>5) ilosc_ac=0;
     if(ilosc_ac==5){    /* odczyt po 5 przetw. - uspok. po klucz */
```

```
            ac_ster= ac_ster | (ac_poz+4);
      outp( AC_ADRES, ac_ster );  /* STOP  */
      ac_ml[ac_poz]= inp(AC_ADRES +4 +2 );  /*czytaj mlodszy bajt*/
      ac_st[ac_poz]= inp(AC_ADRES +4 +1 );  /*czytaj startszy bajt   */
      if((ac_st[ac_poz] & 0x10) == 0x10)  /* przekroczenie */
/*TG*/   goto start_next;       /* dac komunikat ? */
      ac_st[ac_poz] = ac_st[ac_poz] & 0x3F;
      ac[ac_poz] = ac_ml[ac_poz] + ( ac_st[ac_poz] & 0x0F ) * 256;
      if((ac_st[ac_poz] & 0x20) == 0x20)
       ac[ac_poz] += 4096;      /* dodati */
      else
       ac[ac_poz] = 4096 - ac[ac_poz] ;      /* znak ujemny */
      ++il_pom;
      sum_ac[ac_poz]+= (double)ac[ac_poz];
      if(il_pom>3) {
       il_pom=0;
       sred_ac[ac_poz] =sum_ac[ac_poz]/4.0;
       sum_ac[ac_poz]=0.0;
      }
start_next:       if(++ac_poz >= IL_WEJ)   ac_poz = 0;        /* przyrost pozycji kluczy */
              ac_ster=( ac_ster |(ac_poz+4)) | 8;
       outp( AC_ADRES, ac_ster );  /* ZMIANA KLUCZY I START */
            /* po start AC robi autozero a napiecie WE sie ustala */
      }
      else
      {
          ac_ster= ac_ster |(ac_poz+4);
       outp( AC_ADRES, ac_ster );  /* STOP  */
          ac_ster=( ac_ster |(ac_poz+4)) | 8;
       outp( AC_ADRES, ac_ster ); /* START ze starymi kluczami */
      }
      return(0);
}
/*-------------------------------------------------------*/

void interrupt far int_ac()      /* INT4 */
{
/*      _enable(); */
       ac_glowny();
       outp(0,0x20);   /* EOI  */
}
/*-------------------------------------------------------*/

void ac_start()
{
      ac_poz = 0;    /* ustawienie kluczy */

      outp( AC_ADRES,( ac_ster|(ac_poz+4))| 8 );  /* +8 START */
      outp(2,stan_przerwan=(stan_przerwan & EN_AC4));      /* odblokuj tylko ac */
}
/*******************************************************/

int zmien_proc(void)
{               /* odnawianie wartosci powinno byc co 0.5 sek */
                /* wstawic gdzies procent=0;  */

  if(FAst==1) {
   procent = procent + (double)znak * 10.0 ;
   FAst=0;
  }
  if(czas_przyrostu>1 && czas_przyrostu<9) { /* jednorazowe - 5 przerwan /sek  */
   if(!DOdalem) {
    procent+= (double)znak * bit ;  /* jeden bit */
    DOdalem=1;
   }
  }
```


31

```
   if( czas_przyrostu>8 && czas_przyrostu<15 ) { /* 1.5 sekundowe  */
    procent = procent + (double)znak * 10.0 * bit ;
   }
   if(czas_przyrostu>14  && czas_przyrostu<150) {
    procent = procent + (double)znak * 100.0 * bit ;
   }
    if(procent>100.0) procent=100.0;
    if(procent<0.0) procent=0.0;
return(0);
}
/*****************************************************************/

int zmien_nastawe(void)
{
 double wart, krok;

  if(FAst==1) {  /* wartosc + 10% */
    if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
     rez= rez+(double)znak * 0.1 * (gorne_r-dolne_r);
     wart=rez;
    }
    else
    {
    mV= mV+ (double)znak * 0.1 * (gorne_mV-dolne_mV);
     wart=mV;
    }
    FAst=0;
  }
  else
  {
   if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
    wart=rez;
    if(rez<400.0)
     krok=ROZDZ_R400;
    else
     krok=ROZDZ_R4000;
    }
    else {
     wart=mV;
     krok=ROZDZ_CA;
    }

   if(czas_przyrostu>1 && czas_przyrostu<12) { /* jednorazowe - 5 przerwan /sek  */
    if(!DOdalem) {
     wart=  wart + (double)znak * 1.1 * krok ;  /* wiecej niz jeden bit */
     DOdalem=1;
    }
   }
   if( czas_przyrostu>11 ) { /* 1.5 sekundowe  */
    wart = wart + (double)znak * 5.2 * krok ;
   }
/*   if(czas_przyrostu>19  && czas_przyrostu<150) {
    wart = wart + (double)znak * 20.2 * krok ;
   }
*/
  } /* koniec warunku na FAST */

  if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
    rez=wart;
    if(rez<dolne_r) rez=dolne_r;
    if(rez>gorne_r) rez=gorne_r;
  }
  else {
    mV=wart;
    if(mV<dolne_mV) mV=dolne_mV;
    if(mV>gorne_mV) mV=gorne_mV;
```

```
  }
  return(0);
}
/*************************************************************/
void obliczona_nastawa(temp)
double temp;
{
 double wartosc;
 int z=0;
  if(typ_zadaj==PLATYNOWE){
   if(temp>=0.0)
    wartosc = rz *(1 + az*temp + bz * temp*temp);
   else
    wartosc = rz *(1 + az*temp + bz*temp*temp +cz*(temp-100.0)*temp*temp*temp);
  }
  if(typ_zadaj==REZYSTANCJA){
  wartosc=temp;
  }
  if(typ_zadaj==TERMOELEMENT)
  {
   while((double)tab_temp[z++] < temp)
    ;   /* przy wyjsciu z+1*/
   if((double)tab_temp[z-1]==temp)
    wartosc = (double)tab_wart[z-1]/1000.0;
   else
   /*  35.45 *100 =3545.0 ;   reszta =545.0       */
   if(tab_temp[z-1]!=tab_temp[z-2])
    wartosc = (double)tab_wart[z-2]/1000.0 + (temp-(double)tab_temp[z-2])/((double)tab_temp[z-1]-(double)tab_temp[z-2]) *
            ((double)tab_wart[z-1]/1000.0 - (double)tab_wart[z-2]/1000.0);

  }
  if((typ_zadaj==PRAD)||(typ_zadaj==mVOLOTY))
  {
    wartosc = temp/sym_wsp_elektr;
  }
  nastx=wartosc;
}
/*************************************************************/

int zmien_int(void)
{               /* odnawianie wartosci powinno byc co 0.5 sek */
                /* wstawic gdzies zad_int=0;  */
   if(FAst==1) {
    zad_int = zad_int + znak * 100 ;
    FAst=0;
   }
   if(czas_przyrostu<3) { /* jednorazowe - 5 przerwan /sek  */
    zad_int = zad_int + znak  ;  /* jeden bit */
   }
   if( czas_przyrostu>2 && czas_przyrostu<8 ) { /* 1.5 sekundowe  */
    zad_int = zad_int + znak * 5 ;
   }
   if(czas_przyrostu>7  && czas_przyrostu<15 ) {
    zad_int = zad_int + znak * 10 ;
   }
return(0);
}
/*************************************************************/

int zmien_double_a1(zmienna,minimum,maksimum)
                /* odnawianie wartosci powinno byc co 0.5 sek */
double *zmienna;
double minimum,maksimum;
{
   if(FAst==1) {
    *zmienna +=  (double)znak * 100.0 ;
```

```
    FAst=0;
   }
   if(czas_przyrostu>1 && czas_przyrostu<9) { /* jednorazowe - 5 przerwan /sek  */
    if(!OOdalem) {
     *zmienna +=   (double)znak  ;  /* jeden bit */
     OOdalem=1;
    }
   }
   if( czas_przyrostu>8 && czas_przyrostu<15) { /* 1.5 sekundowe  */
    *zmienna +=   (double)znak * 5.0 ;
   }
   if(czas_przyrostu>14 && czas_przyrostu<150) {
    *zmienna +=   (double)znak * 10.0 ;
   }

   if(*zmienna>maksimum)
    *zmienna=maksimum;
   if(*zmienna<minimum)
    *zmienna=minimum;

return(0);
}

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

int zadaj_ptXXX(void)  /* zwraca naprawde zadana temp w ~C */
{
double dz;
unsigned int   bity_rez;
double temp, zakr_rez=400.0;
double czasowe;

/* ffffffffffJK
   temp = p_sym + (k_sym -p_sym)*procent/100.0;
   if(temp>=0.0)
    rez = rz *(1 + az*temp + bz * temp*temp);
   else
    rez = rz *(1 + az*temp + bz*temp*temp +cz*(temp-100.0)*temp*temp*temp);
 ffffffffffJK  */

   if (rez>400.0)
   zakr_rez=4000.0;
   czasowe=rez * 32767.0 / zakr_rez ;
   bity_rez= (unsigned int)czasowe;
   st_bajt_rez = (unsigned char) (( bity_rez & 0x7f00 ) / 256 ) ;
         /* zeruje mlodszy bajt_rez przesuwa starszy na mlodszy */
   if(zakr_rez==4000.0)
    st_bajt_rez = st_bajt_rez | 0x80 ;    /* ustawienie bitu 16 na 1 */
   ml_bajt_rez = (unsigned char) bity_rez;

   outp(ZRSTARY,st_bajt_rez);
   outp(ZRMLODY,ml_bajt_rez);

   rez=(double)bity_rez * zakr_rez / 32767.0; /* rez nastawiona naprawde */

   if( rez/rz >= 1.0 ){
    dz = az * rz *az *rz - 4.0* bz *rz *(rz - rez); /*delta row kwadrat */
    if(dz>0.0)
     nastawa = (sqrt(dz) - az *rz)/( 2.0* bz * rz); /* temp nastawiona naprawde */
    else
     nastawa = +850;
   }
   else
    nastawa=Pt_ujemna(rez);
```

```
  if (k_sym != p_sym)
   nastawa_proc = 100.0 * (nastawa -  p_sym ) / (k_sym -p_sym);
  else
/*TG*/   return(1);
return(0);
}

/*************************************ffffffffffffffff  JK */
/* obliczanie temperatury ujemnej z bledem 0.1st = 0.04 om */
double Pt_ujemna(rezyst)
double rezyst;
{
double dz, rwylicz,temp,roznica,bl_gran_rez;
int i;
  /* wstepne obliczenie temperatury z rownania kwadratowego */
  dz = az * rz *az *rz - 4.0* bz *rz *(rz - rezyst); /*delta row kwadrat */
  if (dz>=0.0)
   temp = (sqrt(dz) - az *rz)/( 2.0* bz * rz);
  else{
   temp=-200;
   return(temp);
  }

  bl_gran_rez=0.04*rz/100.0;

  /* iteracyjne dopasowanie do rownania dla temp. ujemnych */
  i=0;
  while(i<100){     /* ograniczenie ilosci iteracji do wyjscia */
   i++;
   rwylicz= rz *(1 + az*temp + bz*temp*temp +cz*(temp-100.0)*temp*temp*temp);
   roznica=rezyst-rwylicz;
   if (roznica <= 0){
    if (roznica>(-1.0)*bl_gran_rez)
     return(temp);
    else
     temp=temp-0.1;
   }
   if (roznica > 0){
    if (roznica<bl_gran_rez)
     return(temp);
    else
     temp=temp+0.1;
   }
  }
}
/*****************************************************************/

int zadaj_rezystancje(void)
{

unsigned int   bity_rez;
double   zakr_rez=400.0;
double czasowe;

/*    fffffffffJK
  rez = p_sym + (k_sym -p_sym)*procent/100.0;
 fffffffffJK  */

  if (rez>400.0)
  zakr_rez=4000.0;
  czasowe=rez * 32767.0 / zakr_rez ;
  bity_rez= (unsigned int)czasowe;
  st_bajt_rez = (unsigned char) (( bity_rez & 0x7f00 ) / 256 ) ;
        /* zeruje mlodszy bajt_rez przesuwa starszy na mlodszy */
  if(zakr_rez==4000.0)
   st_bajt_rez = st_bajt_rez | 0x80 ;    /* ustawienie bitu 16 na 1 */
```


35

```
   ml_bajt_rez = (unsigned char) bity_rez;

   outp(ZRSTARY,st_bajt_rez);
   outp(ZRMLODY,ml_bajt_rez);

   rez=(double)bity_rez * zakr_rez / 32767.0; /* rez nastawiona naprawde */
   nastawa=rez;
   if (k_sym != p_sym)
    nastawa_proc = 100.0 * (nastawa -  p_sym ) / (k_sym -p_sym);
/*TG*/   else  return(1);
return(0);
}


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

int  zadaj_termoelement(void)
 {
double temp, czasowe;
unsigned int   bity_mV;
int z=0;
/*ff   temp = p_sym + (k_sym - p_sym)*procent/100.0;
   while((double)tab_temp[z++] < temp)
    ;  ff*/           /* przy wyjsciu z+1   temp>=pt100 */
/*ff   if((double)tab_temp[z-1]==temp)
    mV = (double)tab_wart[z-1]/1000.0;
   else   ff*/
              /*  35.45 *100 =3545.0 ;   reszta =545.0       */
/*ff  if(tab_temp[z-1]!=tab_temp[z-2])
   mV = (double)tab_wart[z-2]/1000.0 + (temp-(double)tab_temp[z-2])/((double)tab_temp[z-1]-(double)tab_temp[z-2]) *
                    ((double)tab_wart[z-1]/1000.0 - (double)tab_wart[z-2]/1000.0);
ff*/

   if(mV>=0)
    czasowe = mV * 40.95 ;   /* 4095/100mV */
   else
    czasowe = -(mV * 40.95) ;
   bity_mV = (unsigned int)czasowe;
   st_bajt_ca = (unsigned char) (( bity_mV & 0x0f00 ) / 256 ) ;
         /* zeruje mlodszy bajt_ca przesuwa starszy na mlodszy */

   ml_bajt_ca = (unsigned char) bity_mV;
   outp(CYFRA_ANALOG+1,ml_bajt_ca);
   outp(CYFRA_ANALOG+2,st_bajt_ca);

   if(mV>=0)  {
    mV = (double)bity_mV  / 40.95;   /* mV nastawiona naprawde */
    ac_ster= ac_ster & ~MASKA_ZNAK;
   }
   else {
     mV =-(double)bity_mV  / 40.95;   /* mV nastawiona naprawde */
     ac_ster= ac_ster | MASKA_ZNAK;
   }
   outp( AC_ADRES, ac_ster );
   z=0;
   while((double)tab_wart[z++]/1000.0 < mV)
   ;              /* przy wyjsciu z+1   temp>=pt100 */
   if((double)tab_wart[z-1]/1000.0==mV)
    nastawa = (double)tab_temp[z-1];      /* naprawde nastawiona temp */
   else
    /*TG*/  if(tab_wart[z-1]!=tab_wart[z-2])
    nastawa = (double)tab_temp[z-2] + (mV-(double)tab_wart[z-2]/1000.0)/((double)tab_wart[z-1]/1000.0-(double)tab_wart[z-2]/1000.0)
                   * ((double)tab_temp[z-1] - (double)tab_temp[z-2]);
     if (k_sym!=p_sym)
    nastawa_proc = 100.0 * (nastawa -  p_sym ) / (k_sym -p_sym);
```

```
/* printf("\n nastawa = %lf  nastawa_proc = %lf", nastawa,nastawa_proc);*/
return(0);
}

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

int  zadaj_prad(void)
{

double temp,  czasowe;
unsigned int   bity_mV;

/*ff   temp = p_sym + (k_sym - p_sym)*procent/100.0;
    if(sym_wsp_elektr==0) {
     return(1);
    }
   mV = temp/sym_wsp_elektr;   ff*/          /* 5000 = 20mA */

    if(mV>=0)
    czasowe = mV * 4095.0/5000.0 ;
   else
    czasowe = -(mV * 4095.0/5000.0) ;
   bity_mV = (unsigned int)czasowe;
   st_bajt_ca = (unsigned char) (( bity_mV & 0x0f00 ) / 256 ) ;
         /* zeruje mlodszy bajt_ca przesuwa starszy na mlodszy */
   ml_bajt_ca = (unsigned char) bity_mV;
   outp(CYFRA_ANALOG+1,ml_bajt_ca);
   outp(CYFRA_ANALOG+2,st_bajt_ca);

   if(mV>=0)  {
    mV = (double)bity_mV * 5000.0 / 4095.0;   /* mV nastawiona naprawde */
    ac_ster= ac_ster & ~MASKA_ZNAK;
   }
   else {
    mV =-(double)bity_mV * 5000.0 / 4095.0;   /* mV nastawiona naprawde */
    ac_ster= ac_ster | MASKA_ZNAK;
   }
   outp( AC_ADRES, ac_ster );
   nastawa =  mV * sym_wsp_elektr;
   if (k_sym!=p_sym)
   nastawa_proc = 100.0 * (nastawa -  p_sym ) / (k_sym -p_sym);

/* printf("\n nastawa = %lf  nastawa_proc = %lf", nastawa,nastawa_proc); */
return(0);
}

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

int oblicz_pomiar(void)
{
double  pom_wyvolt, t_otocz;
double mVo, temp,rezy, proc_wy;

double dz;
int z;

/*mV */ pom_wyvolt = 1000.0 * (ac[1] / 819.1 - 5.0);  /* bity * 10.0V/0x1fff  - 5.0V  */
       pom_wyvolt-=ZERO_STROJ*50.0;  /* odczyt na zakresie 100mV */

   if(typ_pomiaru==PLATYNOWE) {
    if(nastawa > 0.0){                  /* prad */
     rezy= (pom_wyvolt * pom_wsp_elektr) / nastawa;
     if (rezy<0)  /* jesli zamiana zaciskow podczas pomiaru */
      rezy=rezy*(-1);
     if( rezy >= (100.0*(rz/100.0)) ){    /* dla temp>0 */
      /* delta */
```


37

```
      dz = az * rz *az *rz - 4.0* bz *rz *(rz - rezy);
      if(dz>=0.0){
       pomiar = (sqrt(dz) - az *rz)/( 2.0* bz * rz);
      }
      else
       pomiar=+850;
     }
     else {                          /* dla temp<0 */
      pomiar=Pt_ujemna(rezy);
     }
    }
   }

   if(typ_pomiaru ==TERMOELEMENT) {
/*STROJ*/     t_otocz =(double)ac[0] * CnaBit + TEMP_KAL -TEMP_OTOCZ; ;
    mVo = pom_wyvolt * pom_wsp_elektr;
    z=0;
    while((double)tab_wart[z++]/1000.0 < mVo){
     if (tab_wart[z]==13000)   /* sprawdzenie czy nie wyszlo za tabele */
      break;
    }
    /* przy wyjsciu z+1   temp>=mVo */
    if((double)tab_wart[z-1]/1000.0==mVo)
     temp = (double)tab_temp[z-1];
    else
/*TG*/      if(tab_wart[z-1]!=tab_wart[z-2])
     temp = (double)tab_temp[z-2] + (mVo-(double)tab_wart[z-2]/1000.0)/((double)tab_wart[z-1]/1000.0-(double)tab_wart[z-2]/1000.0)
                 * ((double)tab_temp[z-1] - (double)tab_temp[z-2]);
    if(rodzaj_pom>13)
     pomiar = temp;
    else {          /* termopary z kompensacja */
     pomiar = temp + t_otocz;
     /* termop.B do 50 st nie ma kompensacji */
     if ((rodzaj_pom==10)&&(t_otocz<=50.0))
      pomiar = temp;
    }

   }
   if(typ_pomiaru== ELEKTRYCZNE) {
    pomiar = pom_wyvolt * pom_wsp_elektr;
/*TG*/ if(koniec_wyvolt!=zero_wyvolt)
    proc_wy = ( pom_wyvolt*pom_wsp_elektr - zero_wyvolt)*100.0/(koniec_wyvolt-zero_wyvolt);
    blad = proc_wy - nastawa_proc;
   }
/* printf("\n pomiar= %lf",pomiar); */
   return(0);
}
/*****************************************************/

int parametry_pomiaru(void)
{
unsigned char z=0;
     ac_ster = ac_ster & MASKA_WYLPOM;   /* wylacz wszystkie pom */
       switch (rodzaj_pom) {
        case 1 :
        case 2 :
         typ_pomiaru=ELEKTRYCZNE;
         pom_wsp_elektr=0.004;
         ac_ster = ac_ster | MASKA_PRAD;
         outp( AC_ADRES, ac_ster );
         zero_wyvolt=0.0;
         koniec_wyvolt=20.0;
         if(rodzaj_pom==2)
          zero_wyvolt=4.0;
         break;
        case 3 :
```

```
case 6 :
 typ_pomiaru=ELEKTRYCZNE;
 pom_wsp_elektr=0.02;
 ac_ster = ac_ster | MASKA_mV;
 outp( AC_ADRES, ac_ster );
 zero_wyvolt=0.0;
 koniec_wyvolt=100.0;
 if(rodzaj_pom==6)
  zero_wyvolt=-100.0;
 break;
case 4 :
case 5 :
case 7 :
case 8 :
 typ_pomiaru=ELEKTRYCZNE;
 pom_wsp_elektr=0.002;  /* 10V/5000mV */
 ac_ster = ac_ster | MASKA_VOLT;
 outp( AC_ADRES, ac_ster );
 zero_wyvolt=  0.0;
 koniec_wyvolt=5.0;
 if(rodzaj_pom==5)
  koniec_wyvolt=  10.0;
 if(rodzaj_pom==7)
  zero_wyvolt=    -5.0;
 if(rodzaj_pom==8) {
  zero_wyvolt=   -10.0;
  koniec_wyvolt= +10.0;
 }
 break;
case 9 :
case 14:
   typ_pomiaru=TERMOELEMENT;
   pom_wsp_elektr=0.02;
   ac_ster = ac_ster | MASKA_mV;
   outp( AC_ADRES, ac_ster );
   do {
      tab_wart[z] = s_w[z];
      tab_temp[z] = s_t[z++];
      } while(s_w[z] <130000);
   break;
case 10 :
case 15 :
   typ_pomiaru=TERMOELEMENT;
   pom_wsp_elektr=0.02;
   ac_ster = ac_ster | MASKA_mV;
   outp( AC_ADRES, ac_ster );
   do {
      tab_wart[z] = b_w[z];
      tab_temp[z] = b_t[z++];
      } while(b_w[z] <130000);
   break;
case 11 :
case 16 :
   typ_pomiaru=TERMOELEMENT;
   pom_wsp_elektr=0.02;
   ac_ster = ac_ster | MASKA_mV;
   outp( AC_ADRES, ac_ster );
   do {
      tab_wart[z] = k_w[z];
      tab_temp[z] = k_t[z++];
      } while(k_w[z] <130000);
   break;
case 12 :
case 17 :
   typ_pomiaru=TERMOELEMENT;
   pom_wsp_elektr=0.02;
```

```
            ac_ster = ac_ster | MASKA_mV;
            outp( AC_ADRES, ac_ster );
            do {
               tab_wart[z] = j_w[z];
               tab_temp[z] = j_t[z++];
               } while(j_w[z] <130000);
            break;
          case 13 :
          case 18 :
            typ_pomiaru=TERMOELEMENT;
            pom_wsp_elektr=0.02;
            ac_ster = ac_ster | MASKA_mV;
            outp( AC_ADRES, ac_ster );
            do {
               tab_wart[z] = t_w[z];
               tab_temp[z] = t_t[z++];
               } while(t_w[z] <130000);
            break;
          case 19 :
           typ_pomiaru=PLATYNOWE;
           pom_wsp_elektr=2.0;  /* pomiar na zakresie 10V ale wynik w mV */
           ac_ster = ac_ster | MASKA_VOLT;
           outp( AC_ADRES, ac_ster );
           rz=100.0;
           break;
          case 20 :
           typ_pomiaru=PLATYNOWE;
           pom_wsp_elektr=2.0;
           ac_ster = ac_ster | MASKA_VOLT;
           outp( AC_ADRES, ac_ster );
           rz=500.0;
           break;
          case 21 :
           typ_pomiaru=PLATYNOWE;
           pom_wsp_elektr=2.0;
           ac_ster = ac_ster | MASKA_VOLT;
           outp( AC_ADRES, ac_ster );
           rz=1000.0;
           break;

          default:
           break;
         }
return(0);
}
/*******************************************************/
int parametry_symulacji(void)
{
unsigned char z=0;

         switch (rodzaj_sym) {
          case 1 :
           typ_zadaj=PLATYNOWE;
           rz=100.0;
           bit=0.00305;  /*  100/32768 */
           break;
          case 2 :
           typ_zadaj=PLATYNOWE;
           rz=500.0;
           bit=0.00305;
           break;
          case 3 :
           typ_zadaj=PLATYNOWE;
           rz=1000.0;
           bit=0.00305;
           break;
```

```
case 4:
   typ_zadaj=TERMOELEMENT;
   ac_ster = ac_ster | MASKA_SYMmV;
   outp( AC_ADRES, ac_ster );
   bit=0.0244;
   do {
      tab_wart[z] = s_w[z];
      tab_temp[z] = s_t[z++];
      } while(s_w[z] <130000);
   break;
case 5 :
   typ_zadaj=TERMOELEMENT;
   ac_ster = ac_ster | MASKA_SYMmV;
   outp( AC_ADRES, ac_ster );
   bit=0.0244;
   do {
      tab_wart[z] = b_w[z];
      tab_temp[z] = b_t[z++];
      } while(b_w[z] <130000);
   break;
case 6 :
   typ_zadaj=TERMOELEMENT;
   ac_ster = ac_ster | MASKA_SYMmV;
   outp( AC_ADRES, ac_ster );
   bit=0.0244;
   do {
      tab_wart[z] = k_w[z];
      tab_temp[z] = k_t[z++];
      } while(k_w[z] <130000);
   break;
case 7 :
   typ_zadaj=TERMOELEMENT;
   ac_ster = ac_ster | MASKA_SYMmV;
   outp( AC_ADRES, ac_ster );
   bit=0.0244;
   do {
      tab_wart[z] = j_w[z];
      tab_temp[z] = j_t[z++];
      } while(j_w[z] <130000);
   break;
case 8 :
   typ_zadaj=TERMOELEMENT;
   ac_ster = ac_ster | MASKA_SYMmV;
   outp( AC_ADRES, ac_ster );
   bit=0.0244;
   do {
      tab_wart[z] = t_w[z];
      tab_temp[z] = t_t[z++];
      } while(t_w[z] <130000);
   break;
case 9 :
 typ_zadaj=REZYSTANCJA;
 bit=0.00305;
 break;
case 10 :  /* mA */
 typ_zadaj=PRAD;
 sym_wsp_elektr=0.004;
 bit=0.0244;
 ac_ster = ac_ster & ~MASKA_SYMmV;
 outp( AC_ADRES, ac_ster );
 break;
case 11 :   /* 100 mV */
 typ_zadaj=mVOLOTY;
 sym_wsp_elektr=0.02;
 bit=0.0244;
 ac_ster = ac_ster | MASKA_SYMmV;
```

```
            outp( AC_ADRES, ac_ster );
            break;

          default:
            break;
        }
return(0);
}
/*****************************************************/
int zadaj(void)
{
   if(typ_zadaj==TERMOELEMENT)
    zadaj_termoelement();
   if(typ_zadaj==PLATYNOWE)
    zadaj_ptXXX();
   if(typ_zadaj==REZYSTANCJA)
    zadaj_rezystancje();
   if(typ_zadaj==PRAD)
    zadaj_prad();
   if(typ_zadaj==mVOLOTY)
    zadaj_prad();

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

```
/* kaldisp.c   podprogramy sterowania LCD dla KALT400   */

      #include "exdos.h"
      #include <math.h>
      #include "kconst.h"
      #include "kexvaria.h"
      #include "kaldisp.h"

#pragma intrinsic(sqrt,inp,outp)
extern   char* ltoa();
extern   char* itoa();
extern   char* strcat();
extern   char* strcpy();
extern   int strlen();

/*---------- ADRESY ------------*/
/* FD dodac przy wspolpracy z PC */

#define   BASE_55       0x0014          /* 8255 */
#define   LCD_DANE      0x000C          /* 374 na CYF */
#define   LCD_ENABLE    0x0010          /* tylko chip select */
#define   LCD_RS        BASE_55 + 2     /* Port C */
#define   KLAWIATURA    BASE_55         /* Port A */
#define   CONTROL_55    BASE_55 + 3     /* Ctrl Word 8255 */
#define   LED           LCD_RS
#define   ZRMLODY       BASE_55 + 1     /* Port B */
#define   ZRSTARY       0x0018          /* 374 */

/*----------- DANE ------------*/

#define   CW_8255       0x98            /* Awe, Bwy, Clwy, CHwe */
#define   LED_START     0x01
#define   LED_STOP      0x02
#define   RS            0x04

#define   MASKA_LED     0xFC
#define   MASKA_RS      0xFB

#define         EN_KL5      0xFF - 32
#define         DIS_KL5     32

#define         DIS_AC4     16

/* port 374 na CYF  bity danych LCD */
#define         CLEAR_DISPLAY       0x01   /* clear display data */
#define         FUNCTION_SET        0x38   /* 8bit, 2lines, 5X7dots, */
#define         ENTRY_MODE          0x06   /* auto RAM increment  */
#define         SHIFT_CURSOR_RIGHT  0x14
#define         DISP_ON_CURSOR_OFF  0x0C
#define         DISP_ON_CURSOR_ON   0x0E
#define         SET_CG_ADDRESS      0x40   /* wpisywanie wzorow wlasnych znakow */
#define         SET_DD_ADDRESS      0x80   /* ustaw miejsce na LCD  */
#define         LINIA1              0x00
#define         LINIA2              0x40
/* miejsce: SET_DD_ADDRESS + LINIAx (+pozycja w linii 1..40)     */


/*-----Zmienne */
char rodz_sym[IL_RODZ_SYM][12]={"----------","Pt100  [£C]","Pt500  [£C]","Pt1000 [£C]",
                    "T.C. S [£C]","T.C. B [£C]","T.C. K [£C]","T.C. J [£C]","T.C. T [£C]",
                    "R [Ohm]    ","I [mA]     ","U [mV]     "};

char rodz_pom[IL_RODZ_POM][11]={"----------","0..20mA   ","4..20mA   ",
                    "0..100mV  ","0..5V     ","0..10V    ",
                    "+-100mV   ","+-5V      ","+-10V     ",
                    "T.C. S+CJC","T.C. B+CJC","T.C. K+CJC","T.C. J+CJC",
```

```
                    "T.C. T+CJC",
                    "T.C. S    ","T.C. B    ","T.C. K    ","T.C. J    ",
                    "T.C. T    ",
                    "Pt100     ","Pt500     ","Pt1000    " };

char jedn[10][3]={"  ","mA","mA","mV","V ","V ","mV","V ","V ","£C"};

/* zakres poczatkowy 40 st C tylko dla T.C. B   */
/* UWAGA: symulacja Pt100 od -140st bo ponizaj rezyst < 40 oM*/

double zakresp[IL_RODZ_SYM]=   {0,-140,-200,-200,
                    -50,40,-270,-210,-270,
                    40, 0, -100};
double zakresk[IL_RODZ_SYM]=   { 0, 850, 850, 850,
                     1760, 1820, 1370,1200, 400,
                     4000, 20, 100};
double minz[IL_RODZ_SYM]=  { 0, 100, 100, 100,
                     200, 200, 200, 200, 100,
                     360, 2, 10};

/* pozycje pol programowanych migajacych */
int dl_pola[]={2,11,4,4,13,11};
int pocz_pola[]={0,3,15,21,27,3};
int f_poz;                    /* pozycja aktualnie migajaca */

char bufres[]="KAL-400 V1.94 TG/JK PIAP/ZAE - Warszawa";
char bufczek[]="Czekaj - pomiary wstepne...";
char bufpom2[]=" Wylacz zasilanie i sprawdz polaczenia !";
char bufnac[]="NACISNIJ -> DALEJ   ";

/* procedury wewnetrzne */
char* s_double(double,char*,int,int);
char* zero_s_int(int,char*,int);

/* procedury uzywane tez gdzie indziej (kaldisp.h)*/
/*
void zgas_led(void);
void zapal_start(void);
void zapal_stop(void);
void ini_8255(void);
void waitms(int);
void initlcd(void);
void ustaw_kursor(int,int);
void display_znak(char);
void lcdcom(void);
void lcddat(void);
void disp_gorna();
void disp_dolna();
void disp_wynik();
void czysc_blad();
void zmiany();
int memory_plus();
int memory_recall();
int zapisz_mem(int);
int wczytaj_mem(int);
void disp_t(int);
void disp_spacje(int);
void pulapka_double(int,double);
void pulapka_lint(int,unsigned long int);
void display_restart(void);
*/

/************** PROCEDURY PODSTAWOWE **************/
/*---------------------------------*/
void zgas_led()
{
```

```
  outp(LED,((unsigned char)inp(LED))&MASKA_LED);
}
/*----------------------------------*/
void zapal_start()
{
  outp(LED,(((unsigned char)inp(LED))&MASKA_LED)|LED_START);
}
/*----------------------------------*/
void zapal_stop()
{
  outp(LED,(((unsigned char)inp(LED))&MASKA_LED)|LED_STOP);
}
/*----------------------------------*/
void ini_8255()
{
  outp(CONTROL_55,CW_8255);
  outp(LED,LED_STOP);       /* port C tylko led_stop reszta 0 */
  outp(ZRHLODY,0);          /* port B  wszystkie 0 */
  outp(LCD_DANE,0);         /* port A  wszystkie 0 */
}


/*-------------------------------------- */
void lcdcom()
{
 /*  Displaycommand     RS=0 */
   outp(LCD_RS,((unsigned char)inp(LCD_RS))&MASKA_RS);
   outp(LCD_ENABLE,0);   /* chip select dla LCD */
   outp(LCD_DANE,0);
}
/*-------------------------------------- */
void lcddat()
{
  /* Displaydata        RS=1 */
   outp(LCD_RS,(((unsigned char)inp(LCD_RS))&MASKA_RS)|RS);
   outp(LCD_ENABLE,0);
   outp(LCD_RS,((unsigned char)inp(LCD_RS))&MASKA_RS);
   outp(LCD_DANE,0);
}
/*-------------------------------------- */
/* wait x msec */
void waitms(x)
int x;
{
 long int i;
 int j;
 for(j=0;j<x;j++)
  for(i=0;i<100;i++)   /* ok 1 msek */
   ;
}
/*-------------------------------------- */
/* init display */
void initlcd()
{
 int i;
 char c;

 outp(LCD_DANE,FUNCTION_SET);       /* clear display */
 lcdcom();
 for(i=0;i<4;i++)
  waitms(25);
 c=FUNCTION_SET;
 outp(LCD_DANE,c);        /* 8bit, 2lines, 5x7dots  */
 lcdcom();
 waitms(25);     /* wait 25 msec */
 outp(LCD_DANE,FUNCTION_SET);       /* 8bit, 2lines, 5x7dots  */
```

```
 lcdcom();
  waitms(25);    /* wait 25 msec */

  outp(LCD_DANE,FUNCTION_SET);       /* 8bit, 2lines, 5x7dots  */
 lcdcom();
 waitms(25);     /* wait 25 msec */
 outp(LCD_DANE,ENTRY_MODE);
 lcdcom();
 waitms(25);     /* wait 25 msec */

 outp(LCD_DANE,SHIFT_CURSOR_RIGHT);
 lcdcom();
 waitms(25);     /* wait 25 msec */

 outp(LCD_DANE,DISP_ON_CURSOR_OFF);
 lcdcom();
 waitms(25);     /* wait 25 msec */

 outp(LCD_DANE,CLEAR_DISPLAY);
 lcdcom();
 waitms(25);     /* wait 25 msec */

}
/*---------------------------------------- */
/* ustaw kursor w linii na pozycji */
/* linia 1 lub 2 ,pozycja 0..39 */
void ustaw_kursor(linia,pozycja)
int linia;
int pozycja;
{
 if(linia==1)
  outp(LCD_DANE,SET_DD_ADDRESS+LINIA1+(unsigned char)pozycja);
 else
  outp(LCD_DANE,SET_DD_ADDRESS+LINIA2+(unsigned char)pozycja);
  lcdcom();
  waitms(2);     /* wait 2 msec */
}
/*---------------------------------------- */
/* wyswietl znak na lcd */
void display_znak(znak)
char znak;
{
 if(znak == 'f')    /* wyswietlenie oznaczenia stopni zamiast f */
  outp(LCD_DANE,0xdf);
 else
  outp(LCD_DANE,(unsigned char)znak);
 lcddat();
 waitms(1);      /* wait 1 msec */
}
/*---------------------------------------- */

/************** PROCEDURY WYSWIETLANIA LICZB **************/
  /*-------------------------------------------------*/
/* funkcja s_double zamienia liczba typu double na string
   o dlugosci miejsca
   po kropce na po_kropce miejsc
   uzupelnia spacjami poczatek
   jesli za dluga to obcina koniec
   NIE korzysta z DOS (tylko ltoa, strcat, strlen)    */

char* s_double(liczba,bufor,miejsca,po_kropce)
 double liczba;
 char* bufor;
 int miejsca, po_kropce;
 {
  int i,j;
```

```
char buf1[40];
char buf2[40];
long int x;
double y;

if(liczba==0.0)
{
 if(po_kropce==0){
  for(i=0;i<(miejsca-1);i++)
   bufor[i]=' ';
 bufor[i++]='0';
 bufor[i]=0;
 return(bufor);
 }

 j=miejsca-po_kropce-2;
 i=0;
 for(i=0;i<j;i++)
  bufor[i]=' ';
 bufor[i++]='0';
 bufor[i++]='.';
 for (j=0;j<po_kropce;j++)
  bufor[i++]='0';
 bufor[i]=0;
 return(bufor);
}

i=0;                        /* okreslenie znaku */
if(liczba<0){
 buf2[i++]='-';
 liczba=liczba*(-1.0);
}
buf2[i]=0;

if(liczba<1.0)
 y=liczba+1.0;
else
 y=liczba;

 for (i=0;i<(po_kropce+1);i++)
  y=y*10.0;

 if(y>=2147483646.0)        /* max long int -1 */
 {  /* za duza liczba */
   for(i=0;i<miejsca;i++)
    bufor[i]='9';
   bufor[i]=0;
   return(bufor);
 }
 if(liczba<1.0)
  x=(long int) y;
 else
  x=(long int) (y+1);        /* zeby uniknac zaokroglenia */
 x=x/10;
 ltoa(x,buf1,10);
 if(liczba<1.0)
  buf1[0]='0';
 strcat(buf2,buf1);
/* w buf2 liczba ze znakiem bez kropki */

if(po_kropce>0){
 j=strlen(buf2)-po_kropce;
 for(i=0;i<j;i++)
  buf1[i]=buf2[i];
 buf1[i]='.';
 while(buf2[i]!=0){
```

```
    buf1[i+1]=buf2[i];
    i++;
   }
   buf1[i+1]=0;
  }
  else
   strcpy(buf1,buf2);

/* w buf1 liczba ze znakiem i kropka */

   bufor[0]=0;
   j=miejsca-strlen(buf1);
   for(i=0;i<j;i++)          /* uzupelnienie spacjami */
    bufor[i]=' ';
   bufor[i]=0;

   strcat(bufor,buf1);

   if(strlen(bufor)>miejsca)
    bufor[miejsca]=0;
   return(bufor);
 }
/*----------------------------------*/
/*--------------------------------------------------*/
/* funkcja zero_s_int zamienia liczba typu int na string
   o dlugosci miejsca
   uzupelnia zerami poczatek
   jesli za dluga to obcina koniec
   NIE korzysta z DOS (tylko itoa, strcat, strlen)    */

char* zero_s_int(liczba,bufor,miejsca)
 int liczba;
 char* bufor;
 int miejsca;
 {
  int i,j;
  char buf1[40];
  if(liczba==0){
   j=miejsca;
   for(i=0;i<j;i++)
     bufor[i]='0';
   bufor[i]=0;
   return(bufor);
  }

  itoa(liczba,buf1,10);
  if(liczba>0)
  j=miejsca-strlen(buf1);
  if(liczba<0) j--;
  i=0;
  if (j >0)
   for(i=0;i<j;i++)
    if(liczba<0)
     bufor[i]=' ';
    else
     bufor[i]='0';
  bufor[i]=0;
  strcat(bufor,buf1);
  if(strlen(bufor)>miejsca)
   bufor[miejsca]=0;
  return(bufor);
 }
/************** PROCEDURY WYSWIETLANIA MASEK **************/
/*--------------------------------------------------*/
void disp_gorna()
{
```

```
 char buf[50];
 char buf1[20];
 int i;
  if(rodzaj_sym>0){
   zero_s_int(nr_mem,buf1,2);
   strcpy(buf,buf1);
   strcat(buf," ");
   strcat(buf,rodz_sym[rodzaj_sym]);
   strcat(buf," ");
   s_double(p_sym,buf1,4,0);
   strcat(buf,buf1);
   strcat(buf,"  ");
   s_double(k_sym,buf1,4,0);
   strcat(buf,buf1);
   strcat(buf,"  ");
   s_double(nastawa,buf1,6,2); /* 6 miejsc, 1 po przec */
   strcat(buf,buf1);
   strcat(buf," ");
   s_double(nastawa_proc,buf1,5,2);
   strcat(buf,buf1);
   strcat(buf,"%");
  }
  else
    strcpy(buf,"  ----------                          ");

   buf[40]=0;   /* konczenie bufora na wszelki wypadek */
   i=0;
   ustaw_kursor(1,0);
   while(buf[i]!=0)
    display_znak(buf[i++]);
}
/*-------------------------------------------------------*/
void disp_dolna()
{
 char buf[50];
 char buf1[20];
 int i;
  if(rodzaj_pom>0){
   strcpy(buf,"   ");
   strcat(buf,rodz_pom[rodzaj_pom]);
   strcat(buf,"  ");
   s_double(pomiar,buf1,6,2);
   strcat(buf,buf1);
   if(rodzaj_pom<10)
     strcat(buf,jedn[rodzaj_pom]);
   else
     strcat(buf,jedn[9]);
   if((rodzaj_pom<9)&&(rodzaj_sym>0)){        /* tylko dla ew.przetwornikow blad */
    strcat(buf,"    ");
    s_double(blad,buf1,6,2);
    strcat(buf,buf1);
    strcat(buf,"%");
   }
   else
    strcat(buf,"            ");
  }
  else
    strcpy(buf,"  ----------                          ");
  buf[40]=0;
  i=0;
  ustaw_kursor(2,0);
  while(buf[i]!=0)
  display_znak(buf[i++]);

}
/*-------------------------------------------------------*/
```

```
void disp_wynik()
{
 char buf[50];
 char buf1[20];
 int i;
 /* nastawa  i procenty w gornej linii */
 if(rodzaj_sym>0){
   s_double(nastawa,buf1,6,2); /* 6 miejsc, 1 po przec */
   strcpy(buf,buf1);
   strcat(buf," ");
   s_double(nastawa_proc,buf1,5,2);
   strcat(buf,buf1);
   strcat(buf,"%");
   buf[14]=0;   /* konczenie bufora na wszelki wypadek */
   i=0;
   ustaw_kursor(1,pocz_pola[4]);
   while(buf[i]!=0)
    display_znak(buf[i++]);
 }

/* pomiar i blad w dolnej linii */
 if(rodzaj_pom>0){
   s_double(pomiar,buf1,6,2);
   strcpy(buf,buf1);
   if(rodzaj_pom<10)
     strcat(buf,jedn[rodzaj_pom]);
   else
     strcat(buf,jedn[9]);
   if((rodzaj_pom<9)&&(rodzaj_sym>0)){  /* tylko dla ew.przetwornikow blad */
    strcat(buf,"    ");
    s_double(blad,buf1,6,2);
    strcat(buf,buf1);
    strcat(buf,"%");
   }
  buf[25]=0;
  i=0;
  ustaw_kursor(2,15);
  while(buf[i]!=0)
  display_znak(buf[i++]);
 }
}
/*-------------------------------------------------------*/
void czysc_blad()
{
 char buf[50];
 int i;
  strcpy(buf,"           ");
  ustaw_kursor(2,24);
  i=0;
  while(buf[i]!=0)
  display_znak(buf[i++]);

}
/*-------------------------------------------------------*/
void    zmiany()
{
 int licznik;
 int spacje;
 int ZMiana=0;
double stara;

       stan_przerwan=(stan_przerwan|DIS_AC4);
       outp(2,stan_przerwan);   /*   OCW3 blokuj ac */

       licznik=MAX1;
       spacje=1;
```

```
STart=0;
STop=0;
if(rodzaj_pom>18)
 f_poz=5;

while(STart==0)  {
 /* klawt();*/
  if( PRawo ) {
   if( PRawo && (rodzaj_pom <= 18)){   /* zmiana polozenia */
    czysc_blad();
    disp_t(f_poz);
    if((f_poz==1)&&(rodzaj_sym==0))
     f_poz=5;
    else
     if( ++ f_poz > IL_POL_EKRANU )
         f_poz = 1;    /* ++ pozycja na ekranie - w kolko */
    licznik=0;
    spacje=1;
   }
   PRawo = 0;
  }

  if(STo){
   disp_t(f_poz);
   memory_plus();
   disp_t(0);
  }
  if(RCl){
   disp_t(f_poz);
   if(memory_recall()>=1){
    if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
     obliczona_nastawa(p_sym);
     dolne_r=nastx;
     obliczona_nastawa(k_sym);
     gorne_r=nastx;
     obliczona_nastawa(nastawa);
     rez=nastx;
    }
    else {
     obliczona_nastawa(p_sym);
     dolne_mV=nastx;
     obliczona_nastawa(k_sym);
     gorne_mV=nastx;
     obliczona_nastawa(nastawa);
     mV=nastx;
    }

    zadaj();
   }
   disp_t(0);
  }

  if( Plus || MInus || FAst ) {
     licznik=0;
     spacje=1;
     switch ( f_poz )  {

        case 1:
         FAst=0;
         if( Plus ) {
          if( ++ rodzaj_sym>(IL_RODZ_SYM-1))
           rodzaj_sym--;
          else
           ZMiana=1;
         Plus = 0;
         }
```

```
                if(MInus){
                 if( -- rodzaj_sym<0)
                   rodzaj_sym++;
                 else
                  ZMiana=1;
                 MInus = 0;
                }
                if(ZMiana){
                 ZMiana=0;
                 parametry_symulacji();
                 p_sym=zakresp[rodzaj_sym];
                 k_sym=zakresk[rodzaj_sym];
                 if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
                   obliczona_nastawa(p_sym);
                   dolne_r=nastx;
                   obliczona_nastawa(k_sym);
                   gorne_r=nastx;
                   rez=dolne_r;
                 }
                 else {
                   obliczona_nastawa(p_sym);
                   dolne_mV=nastx;
                   obliczona_nastawa(k_sym);
                   gorne_mV=nastx;
                   mV=dolne_mV;
                 }
               /*  procent=0; */
                 zadaj();
                 disp_gorna();
                 czysc_blad();
                }
                     break;
               case 2:
               /* poczatek symulacji musi byc wiekszy od poczatku dla danego
                  rodzaju, a zakres musi byc <= rozpietosc zakresu
                  dla danego rodzaju */
                stara=p_sym;
                outp(2,stan_przerwan=(stan_przerwan | DIS_KL5));  /* zablokuj*/
zmien_double_ol(&p_sym,zakresp[rodzaj_sym],k_sym-minz[rodzaj_sym]);
                if(p_sym != stara){
                 if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
                   obliczona_nastawa(p_sym);
                   dolne_r=nastx;
                   rez=dolne_r;
                 }
                 else {
                   obliczona_nastawa(p_sym);
                   dolne_mV=nastx;
                   mV=dolne_mV;
                 }
               /*  procent=0;*/
                 zadaj();
                 disp_gorna();
                }
                outp(2,stan_przerwan=(stan_przerwan & EN_KL5)); /* odblokuj*/
                if(APlus==0) Plus=0;
                if(AMinus==0) MInus=0;
                FAst=0;
                break;
               case 3:
               /* koniec symulacji musi byc mniejszy od konca dla danego
                  rodzaju itd j.w.*/
                stara=k_sym;
               outp(2,stan_przerwan=(stan_przerwan | DIS_KL5));  /* zablokuj*/
zmien_double_ol(&k_sym,p_sym+minz[rodzaj_sym],zakresk[rodzaj_sym]);
                if(k_sym != stara){
```


52

```
 if((typ_zadaj==PLATYNOWE)||(typ_zadaj==REZYSTANCJA)){
   obliczona_nastawa(k_sym);
   gorne_r=nastx;
   rez=dolne_r;
 }
 else {
   obliczona_nastawa(k_sym);
   gorne_mV=nastx;
   mV=dolne_mV;
 }
/* procent=0;*/
 zadaj();
 disp_gorna();
}
outp(2,stan_przerwan=(stan_przerwan & EN_KL5)); /* odblokuj*/

if(APlus==0) Plus=0;
if(AMinus==0) MInus=0;
FAst=0;
break;
case 4:
/* zmiana nastawy */
outp(2,stan_przerwan=(stan_przerwan | DIS_KL5));  /* zablokuj*/
  /*zmien_proc();*/
  zmien_nastawe();
  if(APlus==0) Plus=0;
  if(AMinus==0) MInus=0;
  FAst=0;
  zadaj();
outp(2,stan_przerwan=(stan_przerwan & EN_KL5));    /* odblokuj */
 break;

case 5:
 FAst=0;
 if( Plus ){
  if( ++ rodzaj_pom>(IL_RODZ_POM-1))
   rodzaj_pom--;
  else
   ZMiana=1;
  Plus = 0;
 }
 if(MInus){
  if( -- rodzaj_pom<0)
   rodzaj_pom++;
  else
   ZMiana=1;
  MInus = 0;
 }
 if(ZMiana){
  ZMiana=0;
  parametry_pomiaru();
  pomiar=0;
  disp_dolna();
  czysc_blad();
  if(rodzaj_pom>18) /* platynowe */{
   rodzaj_sym=10; /*mA*/
   parametry_symulacji();
   p_sym=zakresp[rodzaj_sym];
   k_sym=zakresk[rodzaj_sym];
   obliczona_nastawa(p_sym);
   dolne_mV=nastx;
   obliczona_nastawa(k_sym);
   gorne_mV=nastx;

   if(rodzaj_pom==19) /* Pt100 */
    mV=2500;                    /* 10mA */
```

```
                if(rodzaj_pom==20) /* Pt500 */
                 mV=500;                    /* 2mA */
                if(rodzaj_pom==21) /* Pt1000 */
                 mV=250;                    /* 1 mA */
                zadaj();
                disp_gorna();
               }
              }
              break;
               default:
               break;

            } /* switch */
           } /* PLus||MInus */


           if(licznik==0){
            licznik=MAX1;
            if(spacje){     /* ostatnio wyswietlano spacje */
             disp_t(f_poz);
             spacje=0;
            }
            else{
             disp_spacje(f_poz);
             spacje=1;
            }
           }
           licznik--;
          } /* end while  */
          zapisz_mem(0); /* zapisanie biezacych na miejscu 0*/
}


/*-------------------------------------*/
/* wyswietl pole typu z miganiem - do programowania */
void  disp_t(f_p)
int f_p;
{
 int i;
 char buf[40];
 char buf1[20];

 switch ( f_p )  {
  case 0:
   ustaw_kursor(1,pocz_pola[0]);
   zero_s_int(nr_mem,buf,2);
   ustaw_kursor(1,pocz_pola[0]);
   for(i=0;buf[i]!=0;i++)
    display_znak(buf[i]);
   break;
  case 1:
   ustaw_kursor(1,pocz_pola[1]);
   for(i=0;rodz_sym[rodzaj_sym][i]!=0;i++)
    display_znak(rodz_sym[rodzaj_sym][i]);
   break;
  case 2:
   ustaw_kursor(1,pocz_pola[2]);
   s_double(p_sym,buf,4,0);
   for(i=0;buf[i]!=0;i++)
    display_znak(buf[i]);
   break;
  case 3:
   ustaw_kursor(1,pocz_pola[3]);
   s_double(k_sym,buf,4,0);
   for(i=0;buf[i]!=0;i++)
    display_znak(buf[i]);
```

```
   break;
  case 4:
   ustaw_kursor(1,pocz_pola[4]);
   s_double(nastawa,buf1,6,2); /* 6 miejsc, 1 po przec*/
   strcpy(buf,buf1);
   strcat(buf," ");
   s_double(nastawa_proc,buf1,5,2);
   strcat(buf,buf1);
   strcat(buf,"%");
   for(i=0;buf[i]!=0;i++)
    display_znak(buf[i]);
   break;
  case 5:
   ustaw_kursor(2,pocz_pola[5]);
   for(i=0;rodz_pom[rodzaj_pom][i]!=0;i++)
    display_znak(rodz_pom[rodzaj_pom][i]);
   break;
  default:
   break;

 } /* switch */

/* tutaj zmienne kontrolne 5 znakow  ffffffffffffffffff */

/*   ustaw_kursor(2,34);
   s_double(mV,buf,5,1); */
/*   zero_s_int(czas_przyrostu,buf,2);*/
/*   buf[5]=0;
      for(i=0;buf[i]!=0;i++)
    display_znak(buf[i]);
*/
/* fffffffffffffffffffffffffffffffffffffffffffffffff */



}


/*-------------------------------------*/
void disp_spacje(f_p)
int f_p;
{
 int i;

 if (f_p>4)
  ustaw_kursor(2,pocz_pola[f_p]);
 else
  ustaw_kursor(1,pocz_pola[f_p]);

 for(i=0;i<dl_pola[f_p];i++)
    display_znak(' ');

}

/*-----------------------------------------------------------*/
/* wywolanie calosci z pamieci o okreslonym numerze */
int   memory_recall()
/*  zwraca 0 gdy anulowane przez ponowne nacisniecie RCl
    zwraca 1 gdy koniec przez start, 2 gdy przez stop */
{
    int licznik;
    int spacje;

   licznik=MAX1;
   spacje=1;
```

```
        RCl=0;
        STart=0;
        STop=0;
        zapisz_mem(0); /* zapisanie biezacych na miejscu 0*/
        /* wczytanie i wyswietlenie z biezacego numeru */
        wczytaj_mem(nr_mem);
        disp_gorna();
        disp_dolna();
        while(1)     {   /* WYBOR  REJESTRU PAMIECI  */
         /*    klawt();  */
         /* nacisniecie START powoduje start pomiarow
            STOP programowanie, RCl bez zmian */
          if(STart){
            return(1);
           }
          if(STop){
            return(2);
           }
          if(RCl){  /* powrot do poprzedniego wyswietlenia */
            RCl=0;
            wczytaj_mem(0);
            disp_gorna();
            disp_dolna();
            return(0);
           }
          if(PLus)  {
            PLus=0;
            if(nr_mem<(IL_MEM_MAX-1)){
                ++ nr_mem;
                wczytaj_mem(nr_mem);
                disp_gorna();
                disp_dolna();
                licznik=MAX1; spacje=1;
            }
          }
          if(MInus) {
             MInus=0;
             if(nr_mem>1){
              --nr_mem;
              wczytaj_mem(nr_mem);
              disp_gorna();
              disp_dolna();
              licznik=MAX1; spacje=1;
             }
          }

         if(licznik==0){
           licznik=MAX1;
           if(spacje){     /* ostatnio wyswietlano spacje */
            disp_t(0);
            spacje=0;
           }
           else{
            disp_spacje(0);
            spacje=1;
           }
          }
          licznik--;

        } /* while 1 */
}
/*-------------------------------------------------------------*/
/* zapamietanie calosci do pamieci o okreslonym numerze */
int   memory_plus()
/*  zwraca 0 gdy anulowane przez ponowne nacisniecie STO
    zwraca 1 gdy koniec przez start, 2 gdy przez stop */
```

```
{
   int licznik;
   int spacje;

  licznik=MAX1;
  spacje=1;

        STo=0;
        STart=0;
        STop=0;
        if (nr_mem==0)
         nr_mem=1;
        while(1)      {   /* WYBOR  REJESTRU PAMIECI  */
         /*    klawt();  */
         /* nacisniecie START powoduje start pomiarow
            STOP programowanie, STO bez zmian */
           if(STart){
             zapisz_mem(nr_mem);
             return(1);
            }
           if(STop){
             zapisz_mem(nr_mem);
             return(2);
            }
           if(STo){  /* powrot do poprzedniego wyswietlenia */
             STo=0;
             return(0);
            }
           if(Plus)  {
             Plus=0;
             if(nr_mem<(IL_MEM_MAX-1)){
                 ++ nr_mem;
                 licznik=MAX1; spacje=1;
             }
           }
           if(Minus) {
              Minus=0;
              if(nr_mem>1){
               --nr_mem;
               licznik=MAX1; spacje=1;
              }
           }

          if(licznik==0){
            licznik=MAX1;
            if(spacje){     /* ostatnio wyswietlano spacje */
             disp_t(0);
             spacje=0;
            }
            else{
             disp_spacje(0);
             spacje=1;
            }
          }
          licznik--;

        } /* while 1 */
}
/*-----------------------------------------*/
int zapisz_mem(numer)
 int numer;
 {
  rejestr[numer].rsym = rodzaj_sym;
  rejestr[numer].psym = (float)p_sym;
  rejestr[numer].ksym = (float)k_sym;
  rejestr[numer].nast = (float)nastawa;
```

```
  rejestr[numer].nastp = (float)nastawa_proc;
  rejestr[numer].rpom = rodzaj_pom;
  rejestr[numer].pom = (float)pomiar;
  rejestr[numer].bl = (float)blad;
 }
/*----------------------------------------*/
int wczytaj_mem(numer)
 int numer;
 {
       rodzaj_sym = rejestr[numer].rsym;
       p_sym = (double)rejestr[numer].psym;
       k_sym = (double)rejestr[numer].ksym;
       nastawa = (double)rejestr[numer].nast;
       nastawa_proc = (double)rejestr[numer].nastp;
       rodzaj_pom = rejestr[numer].rpom;
       pomiar = (double)rejestr[numer].pom;
       blad = (double)rejestr[numer].bl;
 }

 /*--------------------------------------*/
 /* wyczyszczenie ekranu i wyswietlenie buf oraz zatrzymanie programu
    z blokada przerwan */

void display_error(buf)
char* buf;
{
 int i;
 _disable();
 outp(LCD_DANE,CLEAR_DISPLAY);
 lcdcom();
 waitms(25);
 for(i=0;buf[i]!=0;i++)
  display_znak(buf[i]);
 ustaw_kursor(2,0);
 for(i=0;bufpom2[i]!=0;i++)
  display_znak(bufpom2[i]);
 while(1)
 ;
}
/*--------------------------------------*/
/*-DISPLAY RESTART - STRONA TYTULOWA ---------------------*/
void display_restart()
{
   int i;

    for(i=0;bufres[i]!=0;i++)
     display_znak(bufres[i]);
    ustaw_kursor(2,10);
    for(i=0;bufczek[i]!=0;i++)
     display_znak(bufczek[i]);

    for(i=0;i<50;i++)   /* czekanie tu ac po raz pierwszy liczy */
     waitms(25);
    /* czyszczenie ekranu */
    outp(LCD_DANE,CLEAR_DISPLAY);
    lcdcom();
    waitms(25);
/* ffffffffffffffffffffffffffff test */
    czas_przyrostu=5;

    f_poz=1;  /* pozycja migajaca */
}
/*--------------------------------------*/
/* Pulapka z wyswietleniem numeru pulapki oraz                    */
/* liczby double  uaktualnianej w petli pulapki np przez przerwanie */
/* wyjscie z pulapki przez  nacisniecie ->                        */
```


58

```
void pulapka_double(numer,liczba)
int numer;
double liczba;
{
 char buf1[20];
 char buf[40];

  int i;

  outp(LCD_DANE,CLEAR_DISPLAY);
  lcdcom();
  waitms(25);    /* wait 25 msec */
   ustaw_kursor(2,10);
  for(i=0;bufnac[i]!=0;i++)
   display_znak(bufnac[i]);

 STo=0;
 while(STo==0){      /* czekaj na nacisniecie -> */
  ustaw_kursor(1,0);
  zero_s_int(numer,buf1,2);
  strcpy(buf," PULAPKA nr: ");
  strcat(buf,buf1);
  strcat(buf," !!  double: ");
  s_double(liczba,buf1,13,1);
  strcat(buf,buf1);

  for(i=0;buf[i]!=0;i++)
   display_znak(buf[i]);

  klawt1();
 }
 STo=0;
}
/*--------------------------------------*/
/* Pulapka z wyswietleniem numeru pulapki oraz                    */
/* liczby long int  uaktualnianej w petli pulapki np przez przerwanie */
/* wyjscie z pulapki przez  nacisniecie ->                        */

void pulapka_lint(numer,liczba)
int numer;
unsigned long int liczba;
{
 char buf1[20];
 char buf[40];

  int i;
  unsigned long int a;
  long int b;

  outp(LCD_DANE,CLEAR_DISPLAY);
  lcdcom();
  waitms(25);    /* wait 25 msec */
   ustaw_kursor(2,10);
  for(i=0;bufnac[i]!=0;i++)
   display_znak(bufnac[i]);

 STo=0;
 while(STo==0){      /* czekaj na nacisniecie -> */
  ustaw_kursor(1,0);
  zero_s_int(numer,buf1,2);
  strcpy(buf," PULAPKA nr: ");
  strcat(buf,buf1);
  strcat(buf," !!  lint st: ");
  a=(liczba & 0xffff0000);
  b=(long int)(a>>16);
```

```
  ltoa(b,buf1,16);
  strcat(buf,buf1);
  strcat(buf," ml: ");
  b=(long int)(liczba-a);
  ltoa(b,buf1,16);
  strcat(buf,buf1);

  for(i=0;buf[i]!=0;i++)
   display_znak(buf[i]);

  klawt1();
 }
 STo=0;
}

/*================================*/
```

```
/* kconst.h    stale do kal400 wykorzystywane w roznych modulach */

#define MAX1            4000  /* do migania */
#define IL_MEM_MAX      20    /* ilosc rejestrow pamieci */
#define IL_RODZ_SYM     12    /* ilosc rodzajow symulacji */
#define IL_RODZ_POM     22    /* ilosc rodzajow pomiarow */
#define IL_POL_EKRANU   5     /* ilosc zmienianych pol ekranu */


/* deklaracja typu rejestr pamieci */
typedef struct p{
   int rsym;
   float psym;
   float ksym;
   float nast;
   float nastp;
   int rpom;
   float pom;
   float bl;
}REJ_PAM;

#define  TERMOELEMENT    1
#define    PLATYNOWE     2
#define    REZYSTANCJA   3
#define    PRAD          4
#define    mVOLOTY       5
#define    ELEKTRYCZNE   6
```

```
/* kvaria.h   zmienne dla KAL400   */
/* zmienne pamietane: od poczatku do glupoty[] */
unsigned char pam_nr_mem = 0;
REJ_PAM rejestr[IL_MEM_MAX+1]=  {0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,

                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,

                                 0,0,0,0,0,0,0,0};

char glupoty[20]={"123456789012345678 9"};


unsigned char   klaw;
unsigned char   STop, STart, RCl, STo, PLus, MInus, FAst, PRawo, ZApis, DOdalem;
unsigned char   AStop, AStart, ARcl, ASto, APlus, AMinus, AFast, APrawo;
unsigned char   ZStop, ZStart, ZRcl, ZSto, ZPlus, ZMinus, ZFast, ZPrawo;
unsigned char   nr_mem;
unsigned char  stan_przerwan, rodzaj_pom, rodzaj_sym;
unsigned char typ_zadaj, typ_pomiaru;
int czas_przyrostu;

double nastawa, nastawa_proc, p_sym, k_sym, pomiar, blad;
double dolne_r, gorne_r, dolne_mV, gorne_mV;
double rez, mV;
/* REJ_PAM rejestr[IL_MEM_MAX+1];  */
double procent;
double nastx;
```

```
/* kexvaria.h   14.01.92   deklaracje zmiennych globalnych z externami */
extern REJ_PAM rejestr[IL_MEM_MAX+1];
extern unsigned char klaw;
extern unsigned char STop, STart, RCl, STo, PLus, MInus, FAst, PRawo, ZApis, DOdalem;
extern unsigned char AStop, AStart, ARcl, ASto, APlus, AMinus, AFast, APrawo;
extern unsigned char ZStop, ZStart, ZRcl, ZSto, ZPlus, ZMinus, ZFast, ZPrawo;
extern unsigned char stan_przerwan, rodzaj_pom, rodzaj_sym;
extern double nastawa, nastawa_proc, p_sym, k_sym, pomiar, blad;
extern unsigned char   nr_mem;
extern double procent;
extern int czas_przyrostu;
extern double dolne_r, gorne_r, dolne_mV, gorne_mV;
extern unsigned char typ_zadaj, typ_pomiaru;
extern double rez, mV;
double nastx;
```


63

```
/*  kaldisp.h   podprogramy do LCD */
/* zwiazane z plytka cyfrowa */


void zgas_led(void);
void zapal_start(void);
void zapal_stop(void);
void ini_8255(void);


void waitms(int);
void initlcd(void);
void ustaw_kursor(int,int);
void display_znak(char);
void lcdcom(void);
void lcddat(void);
void disp_gorna();
void disp_dolna();
void disp_wynik();
int memory_plus();
int memory_recall();
void disp_t(int);
void disp_spacje(int);
void pulapka_double(int,double);
void pulapka_lint(int,unsigned long int);
void display_restart(void);
```

```
/* exdos.h  funkcje kompilatora bez odwolan do DOS */
/* zam £ include <> */

extern int inp();
extern int outp();
extern double sqrt();
extern void _enable();
extern void _disable();
```

```
                    /* stroj.h    30.06.94 */
£define ZERO_STROJ  +0.08      /* odczyt na zakresie 100mV */
£define TEMP_KAL    25.0
£define TEMP_OTOCZ   -3.0
```

```
/* kaltg.h  podprogramy TG do KAL400 */

void klawt1(void);
int zadaj(void);
int oblicz_pomiar(void);
int ac_glowny(void);
void interrupt far int_ac();      /* INT4 */
void ac_start();
int zmien_proc(void);
int zmien_double_o1(double*,double,double);
int parametry_pomiaru(void);
int parametry_symulacji(void);
void obliczona_nastawa(double);
```

```
int s_t[] = {
-50,-40,-30,-20,-10,
0,10,20,30,40,50,60,70,
80, 90, 100, 110, 120,130,140,150,
160,170,180, 190, 200,210,220,230,
240,250,260, 270, 280,290,300,310,
320,330,340, 350, 360,370,380,390,
400,410,420, 430, 440,450,460,470,
480,490,500, 510, 520,530,540,550,
560,570,580, 590, 600,610,620,630,
640,650,660, 670, 680,690,700,710,
720,730,740, 750, 760,770,780,790,
800,810,820, 830, 840,850,860,870,
880,890,900, 910, 920,930,940,950,
960,970,980, 990, 1000, 1010, 1020, 1030,
1040,1050,1060, 1070, 1080,1090,1100,1110,
1120,1130,1140, 1150, 1160,1170,1180,1190,
1200,1210,1220, 1230, 1240,1250,1260,1270,
1280,1290,1300, 1310, 1320,1330,1340,1350,
1360,1370,1380, 1390, 1400,1410,1420,1430,
1440,1450,1460, 1470, 1480,1490,1500,1510,
1520,1530,1540, 1550, 1560,1570,1580,1590,
1600,1610,1620, 1630, 1640,1650,1660,1670,
1680,1690,1700, 1710, 1720,1730,1740,1750,
1760,
30000 };

long int s_w[] = {
 -236,-194,-150,-103,-53,
 0,55,113, 173,235,299, 365,432,
 502,573,645, 719,795,872, 950,1029,
 1109,1190,1273, 1356,1448,1534, 1611,1698,
 1785,1873,1962, 2051,2141,2232, 2323,2414,
 2506,2599,2692, 2786,2880,2974, 3069,3164,
 3260,3356,3452, 3549,3645,3743, 3840,3938,
 4036,4135,4234, 4333,4432,4532, 4632,4732,
 4832,4933,5034, 5136,5237,5339, 5442,5544,
 5648,5751,5855, 5960,6064,6169, 6274,6380,
 6486,6592,6699, 6805,6913,7020, 7128,7236,
 7345,7454,7563, 7672,7782,7892, 8003,8114,
 8225,8336,8448,8560,8673, 8786, 8899,9012,
 9126,9240,9355,9470,9585, 9700, 9816,9932,
 10048,10165,10282,10400,10517,10635,10754,10872,
 10991,11110,11229,11348,11467,11587,11797,11827,
 11947,12067,12188,12308,12429,12550,12671,12792,
 12913,13034,13155,13276,13397,13519,13640,13761,
 13883,14004,14125,14247,14368,14489,14610,14731,
 14852,14973,15094,15215,15336,15456,15576,15697,
 15817,15937,16057,16176,16296,16415,16534,16653,
 16771,16890,17008,17125,17243,17360,17477,17594,
 17711,17826,17942,18056,18170,18282,18394,18504,
 18612,
130000 };


int b_t[] = {
40,50,60,70,
80, 90, 100, 110, 120,130,140,150,
160,170,180, 190, 200,210,220,230,
240,250,260, 270, 280,290,300,310,
320,330,340, 350, 360,370,380,390,
400,410,420, 430, 440,450,460,470,
480,490,500, 510, 520,530,540,550,
560,570,580, 590, 600,610,620,630,
```

```
640,650,660, 670, 680,690,700,710,
720,730,740, 750, 760,770,780,790,
800,810,820, 830, 840,850,860,870,
880,890,900, 910, 920,930,940,950,
960,970,980, 990, 1000, 1010, 1020, 1030,
1040,1050,1060, 1070, 1080,1090,1100,1110,
1120,1130,1140, 1150, 1160,1170,1180,1190,
1200,1210,1220, 1230, 1240,1250,1260,1270,
1280,1290,1300, 1310, 1320,1330,1340,1350,
1360,1370,1380, 1390, 1400,1410,1420,1430,
1440,1450,1460, 1470, 1480,1490,1500,1510,
1520,1530,1540, 1550, 1560,1570,1580,1590,
1600,1610,1620, 1630, 1640,1650,1660,1670,
1680,1690,1700, 1710, 1720,1730,1740,1750,
1760,1770,1780, 1790, 1800,1810,1820,
30000 };

long int b_w[] = {
  0, 2,  6, 11,
  17, 25, 33,  43, 53, 65,  78, 92,
 107,123,140, 159,178,199, 220,243,
  266, 291, 317,  317, 344, 372,  401, 431,
  494, 527, 561,  596, 632, 669,  707, 746,
  786, 827, 870,  913, 957,1002, 1048,1095,
 1143,1192,1241, 1292,1344,1397, 1450,1505 ,
 1560,1617,1674, 1732,1791,1851, 1912,1974,
 2036,2100,2164, 2230,2296,2363, 2430,2499,
 2569,2639,2710, 2782,2855,2928, 3003,3078,
 3154,3231,3308, 3387,3466,3546, 3626,3708,
 3790,3873,3957,4041,4126, 4212,4298,4386 ,
 4474,4562,4652,4742,4833, 4924,5016,5109 ,
 5202,5297,5391,5487,5583, 5680,5777,5875,
 5973,6073,6172,6273,6374, 6475,6577,6680,
 6783,6887,6991,7096,7202, 7308,7414,7521,
 7628,7736,7845,7953,8063, 8172,8283,8393,
 8504,8616,8727,8839,8952, 9065,9178,9291,
 9405,9519,9634,9748,9863, 9979,10094,10210,
10325,10441,10558,10674,10790,10907,11024,11141,
11257,11374,11491,11608,11725,11842,11959,12076,
12193,12310,12426,12543,12659,12776,12892,13008,
13124,13239,13354,13470,13585,13699,13814,
130000 };

int k_t[] = {
-270, -260, -250, -240, -230, -220, -210, -200,
-190, -180, -170, -160, -150, -140, -130, -120,
-110, -100, -90,  -80,  -70,  -60,  -50,  -40,
-30,  -20,  -10, 0, 10, 20, 30, 40,
50,60,  70,  80,  90,  100,  110,  120,
130, 140, 150, 160, 170, 180,  190,  200,
210, 220, 230, 240, 250, 260,  270,  280,
290, 300, 310, 320, 330, 340,  350,  360,
370, 380, 390, 400, 410, 420,  430,  440,
450, 460, 470, 480, 490, 500,  510,  520,
530, 540, 550, 560, 570, 580,  590,  600,
610, 620, 630, 640, 650, 660,  670,  680,
690, 700, 710, 720, 730, 740,  750,  760,
770, 780, 790, 800, 810, 820,  830,  840,
850, 860, 870, 880, 890, 900,  910,  920,
930, 940, 950, 960, 970, 980,  990,  1000,
1010, 1020, 1030, 1040, 1050, 1060,  1070,  1080,
1090, 1100, 1110, 1120, 1130, 1140,  1150,  1160,
1170, 1180, 1190, 1200, 1210, 1220,  1230,  1240,
1250, 1260, 1270, 1280, 1290, 1300,  1310,  1320,
1330, 1340, 1350, 1360, 1370,
30000 };
```

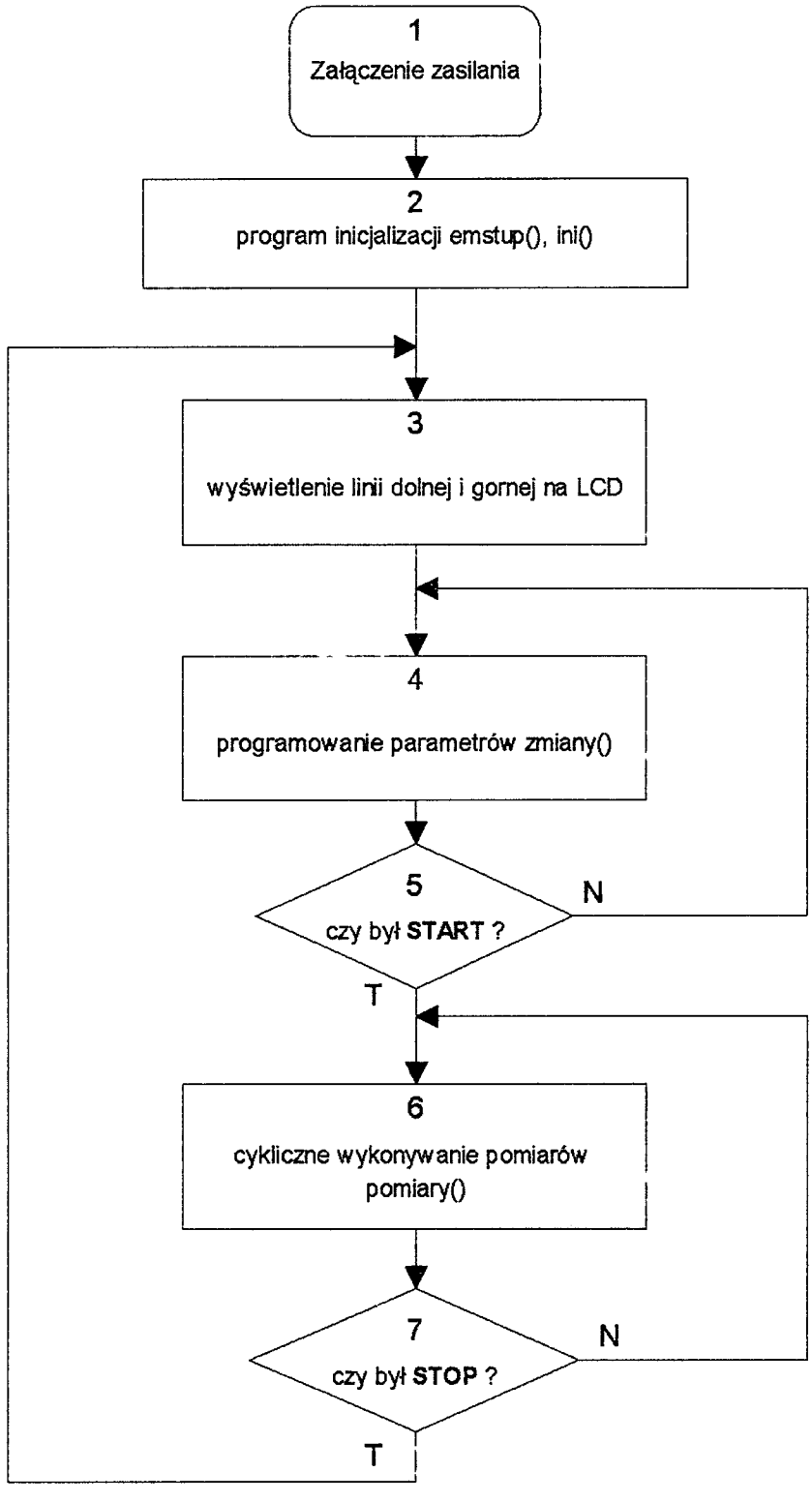
```
long int k_w[] = {
-6458,-6441,-6404,-6344,-6262,-6158,-6035,-5891,
-5730,-5550,-5354,-5141,-4912,-4669,-4410,-4138,
-3852,-3553,-3242,-2920,-2586,-2243,-1889,-1527,
-1156,-777,-392, 0, 397, 798, 1203,
 1611,2022,2436,2850,3266,3681,4095,4508,
 4919,5327,5733,6137,6539,6939,7338,7737,
 8137,8537,8938,9341,9745,10151,10560,10969,
11381,11793,12207,12623,13039,13456,13874,14292,
14712,15132,15552,15974,16395,16818,17241,17664,
18088,18513,18938,19363,19788,20214,20640,21066,
21493,21919,22346,22772,23198,23624,24050,24476,
24902,25327,25751,26176,26599,27022,27445,27867,
28288,28709,29128,29547,29965,30383,30799,31214,
31629,32042,32455,32866,33277,33686,34095,34502,
34909,35314,35718,36121,36524,36925,37325,37724,
38122,38519,38915,39310,39703,40096,40488,40879,
41269,41657,42045,42432,42817,43202,43585,43968,
44349,44729,45108,45486,45863,46238,46612,46985,
47356,47726,48095,48462,48828,49192,49555,49916,
50276,50633,50990,51344,51697,52049,52398,52747,
53093,53439,53782,54125,54466,54807,
130000 };

int j_t[] = {
-210, -200,-190, -180, -170, -160, -150,-140,
-130, -120,-110, -100, -90,  -80,  -70,  -60,
-50,  -40,-30,  -20,  -10, 0, 10, 20, 30, 40,
50, 60,  70,  80,  90,  100,  110,  120,
130, 140, 150, 160, 170, 180,  190,  200,
210, 220, 230, 240, 250, 260,  270,  280,
290, 300, 310, 320, 330, 340,  350,  360,
370, 380, 390, 400, 410, 420,  430,  440,
450, 460, 470, 480, 490, 500,  510,  520,
530, 540, 550, 560, 570, 580,  590,  600,
610, 620, 630, 640, 650, 660,  670,  680,
690, 700, 710, 720, 730, 740,  750,  760,
770, 780, 790, 800, 810, 820,  830,  840,
850, 860, 870, 880, 890, 900,  910,  920,
930, 940, 950, 960, 970, 980,  990,  1000,
1010, 1020, 1030, 1040, 1050, 1060,  1070,  1080,
1090, 1100, 1110, 1120, 1130, 1140,  1150,  1160,
1170, 1180, 1190, 1200,
30000 };

long int j_w[] = {
-8096,-7890,-7659,-7402,-7122,-6821,-6499,-6159,
-5801,-5426,-5036,-4632,-4215,-3785,-3344,-2892,
-2431,-1960,-1481,-995,-0501, 0, 507,1019,
 1536,2058,2585,3115,3649,4186,4725,5268,
 5812,6359,6907,7457,8008,8560,9113,9667,
 10222,10777,11332,11887,12442,12998,13553,14108,
 14663,15217,15771,16325,16879,17432,17984,18537,
 19089,19640,20192,20743,21295,21846,22397,22949,
 23501,24054,24607,25161,25716,26272,26829,27388,
 27949,28511,29075,29642,30210,30782,31356,31933,
 32513,33096,33683,34273,34867,35464,36066,36671,
 37280,37893,38510,39130,39754,40382,41013,41647,
 42283,42922,
 43563,44207,44852,45498,46144,46790,47434,48076,
 48716,49354,49989,50621,51249,51875,52496,53115,
 53729,54341,54948,55553,56155,56753,57349,57942,
 58533,59121,59708,60293,60876,61459,62039,62619,
 63199,63777,64355,64933,65510,66087,66664,67240,
 67815,68390,68964,69536,
```

```
130000 };

int t_t[] = {
-270, -260, -250, -240, -230, -220, -210, -200,
-190, -180, -170, -160, -150, -140, -130, -120,
-110, -100, -90,  -80,  -70,  -60,  -50,  -40,
-30,  -20,  -10, 0, 10, 20, 30, 40,
50,60,  70,  80,  90,  100,  110,  120,
130, 140, 150, 160, 170, 180,  190,  200,
210, 220, 230, 240, 250, 260,  270,  280,
290, 300, 310, 320, 330, 340,  350,  360,
370, 380, 390, 400,
30000 };

long int t_w[] = {
-6258,-6232,-6181,-6105,-6007,-5889,-5753,-5603,
-5439,-5261,-5069,-4865,-4648,-4419,-4177,-3923,
-3656,-3378,-3089,-2788,-2475,-2152,-1819,-1475,
-1121,-757,-383,0,391,789,1196,
 1611,2035,2467,2908,3357,3813,4277,4749,
 5227,5712,6204,6702,7207,7718,8235,8757,
 9286,9820,10360,10905,11456,12011,12572,13137,
13707,14281,14860,15443,16030,16621,17217,17816,
18420,19027,19638,20252,20869,
130000 };
```

```
rem
        rem emkompil.bat
        rem Kompilacja startupu  emstup.asm (use floating point emulator)
        masm /Mx emstup;




        rem
        rem   kompiluj.bat
        rem   kompilacja %1.c z funkcjami matematycznymi small model
        rem   (z optymalizacja   cl /c /AS /Gs /Gm /Fc /FPi /Zd %1.c)
        cl /c /AS /Gs /Gm /Fc /FPi /Od /Zd %1.c




        rem
        rem linkuj.bat
        rem linkowanie zbiorow do proj.lnk dla KAL400
        xlink86 emstup,kalmain,kaltg,kaldisp,slibce.ssi to proj.lnk




        rem
        rem  lokuj.bat
        rem  lokowanie proj.lnk do proj.abs dla NCPUV30
        xloc86  @loc1.cmd
        rem




        proj.lnk to proj.abs &
        order(classes( FAR_DATA_BEG, FAR_DATA, FAR_DATA_END, &
                       FAR_BSS_BEG, FAR_BSS, FAR_BSS_END, &
                       HUGE_BSS_BEG, HUGE_BSS, HUGE_BSS_END, &
                       DATA_BEG, DATA, CONST, MSG, DATA_END, &
                       BSS, BSS_END, STACK, &
                       CODE, CODE_END)) &
        RESERVE(10000H to 0EFFFFH) &
        addresses(classes(FAR_DATA_BEG(400H), CODE(0F0000H))) &
        NOINITCODE
```

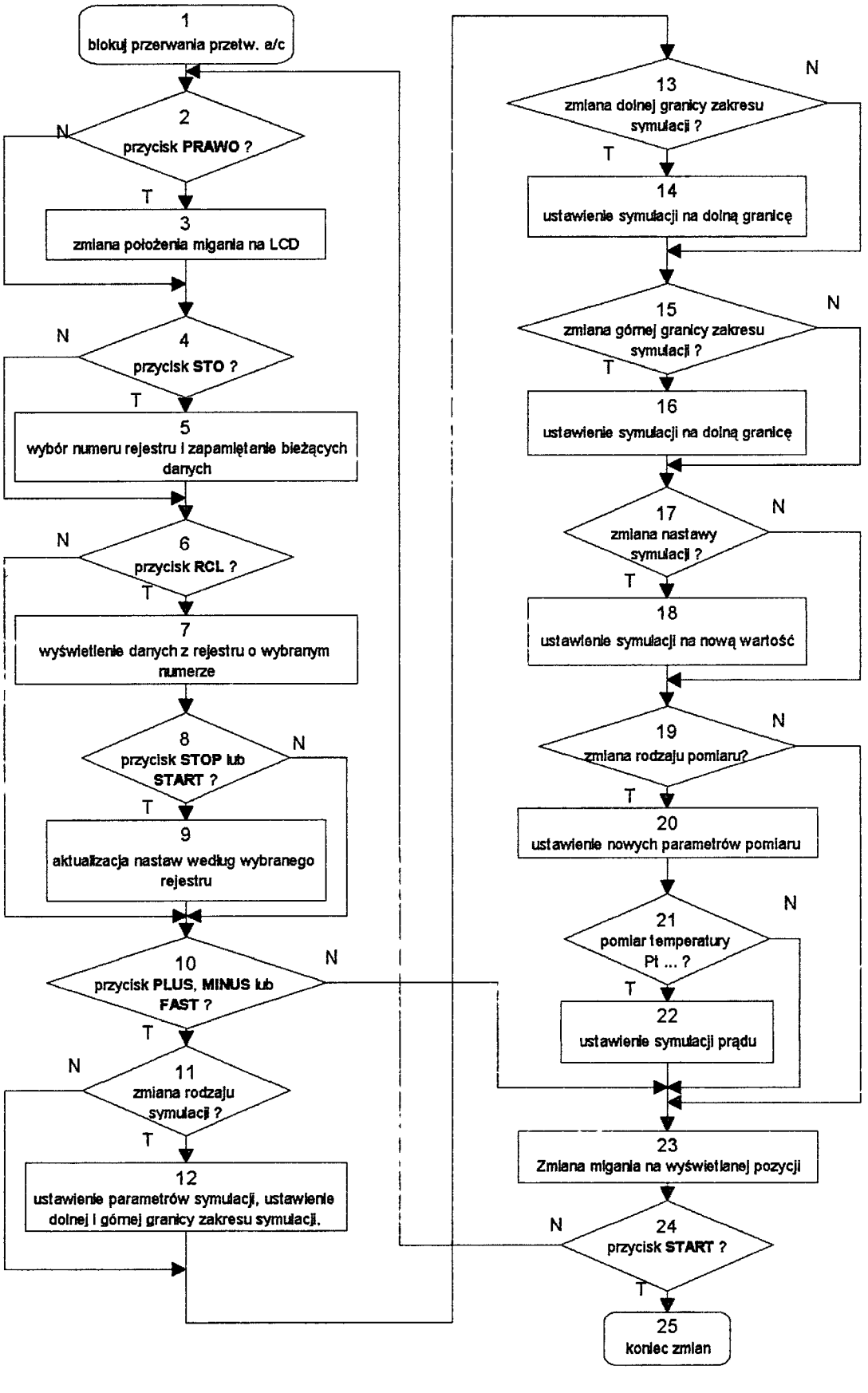
5. Rysunki

Rys. 1 **Schemat blokowy programu głównego**

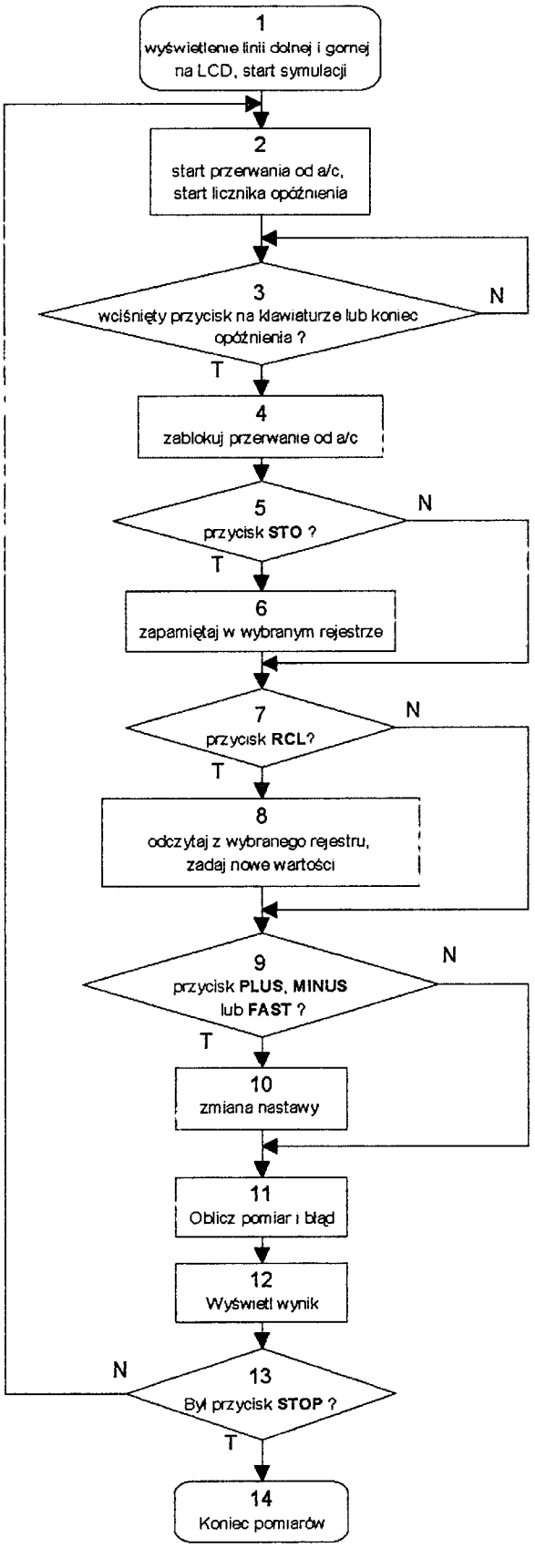


72

Rys. 2 **Schemat blokowy programu zmian parametrów**



Rys. 3 **Schemat blokowy programu pomiarów**



Rys. 4 **Schemat blokowy programu przeglądania wyników**