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

Ośrodek Automatyki Elektrycznej

Pracownia Oprogramowania Wieloprocesorowych Systemow Automatyki i Robotow

Główny wykonawca mgr. inż. Tomasz Wański

Wykonawcy mgr inż. Stanislaw Wóltański

074

Konsultant

Nr zlecenia UR 010403

Opracowanie sekwencyjnego mikroprocesorowego układu sterowania SP grupą robotow przemysłowych.
Etap 11. Opracowanie generatora programow sterujacych dla różnych konfiguracji i zestawow robotow.

Zleceniodawca problem węzłowy 06.6

Pracę rozpoczęto dnia 19.08.1985 **zakończono dnia** 31.10.85

Kierownik Pracowni	Z-ca Dyrektora d/s Automatyki	Kierownik Ośrodka
mgr inż. A. Aderek	dr inż. T. Gałązka	prof. dr inż. T. Missala

Praca zawiera:

stron 6
rysunków 1
fotografii
tabel
tablic
załączników 2

Nr rejestr. 5483

Rozdzielnik - ilość egz: 3

Egz. 1 BOINTE
Egz. 2 OAE
Egz. 3 OAE-83
Egz. 4
Egz. 5
Egz. 6

Załączniki 1 i 2 są poufne.

Analiza deskryptorowa ROBOT+KOMPUTER+MIKROPROCESOR+PROGRAMOWANIE +JĘZYK PROGRAMOWANIA+GENERACJA OPROGRAMOWANIA

Analiza dokumentacyjna

Praca przedstawia sposób generacji programów sterujących dla różnych konfiguracji i zestawów robotów prostych PR-02, przy pomocy minikomputera SM-4. Opisano poszczególne części systemu generacji oraz przedstawiono zasady wprowadzania parametrów generacji z terminala komputera.

Tytuły poprzednich sprawozdań

Etap 3. Opracowanie struktury programu sterującego. Opracowanie programu asemblera /nr rej. 5179/.

Etap 7. Opracowanie programu sterującego dla prototypowego zestawu robota /nr rej. 5273/.

681.322.004.14 Komputery - Zastosowania

335.45:62/69].002.1/.2 Roboty przemysłowe

UKD

PIAP-252/83-6000

2

S P I S T R E S C I

1. Przeznaczenie programu

2. Budowa generatora i przebieg generacji

3. Wprowadzanie parametrow generacji z terminala

4. Sygnalizacja bledow

Zalaczniki:
---

1. Tabulogram konwersacyjnego programu GPR
   (wraz z definicja stalych globalnych)
2. Listing zbioru komend GENPR.CMD
   (wraz z pomocniczym zbiorem DIVINT.CMD)

1. Przeznaczenie programu

Praca zawiera opis s y s t e m u g e n e r a c j i programow sterujacych dla zestawow prostych robotow PR-02. System ten umozliwia przygotowanie postaci wynikowej programow (wymaganej przez programator pamieci stalej EPROM) przy uzyciu komputera SM-4.

Aktualna wersja generatora pozwala modyfikowac zrodlowa postac programu sterujacego (napisanego w jezyku asembler mikrokomputera Intel 8080) w takim zakresie, ze otrzymany program obsluguje co najwyzej 8 robotow, z ktorych kazdy moze zawierac co najwyzej 8 modulow dwupolozeniowych. Moduly te moga byc sterowane z kontrola sygnalow sprzezenia zwrotnego (informujacych o realizacji zadanego polozenia przez dany modul) lub bez takiej kontroli.

Zautomatyzowanie procesu generacji programow sterujacych zestawem robotow, pozwala kreowac nowe wersje tych programow przez osoby nie znajace szczegolow ich budowy.

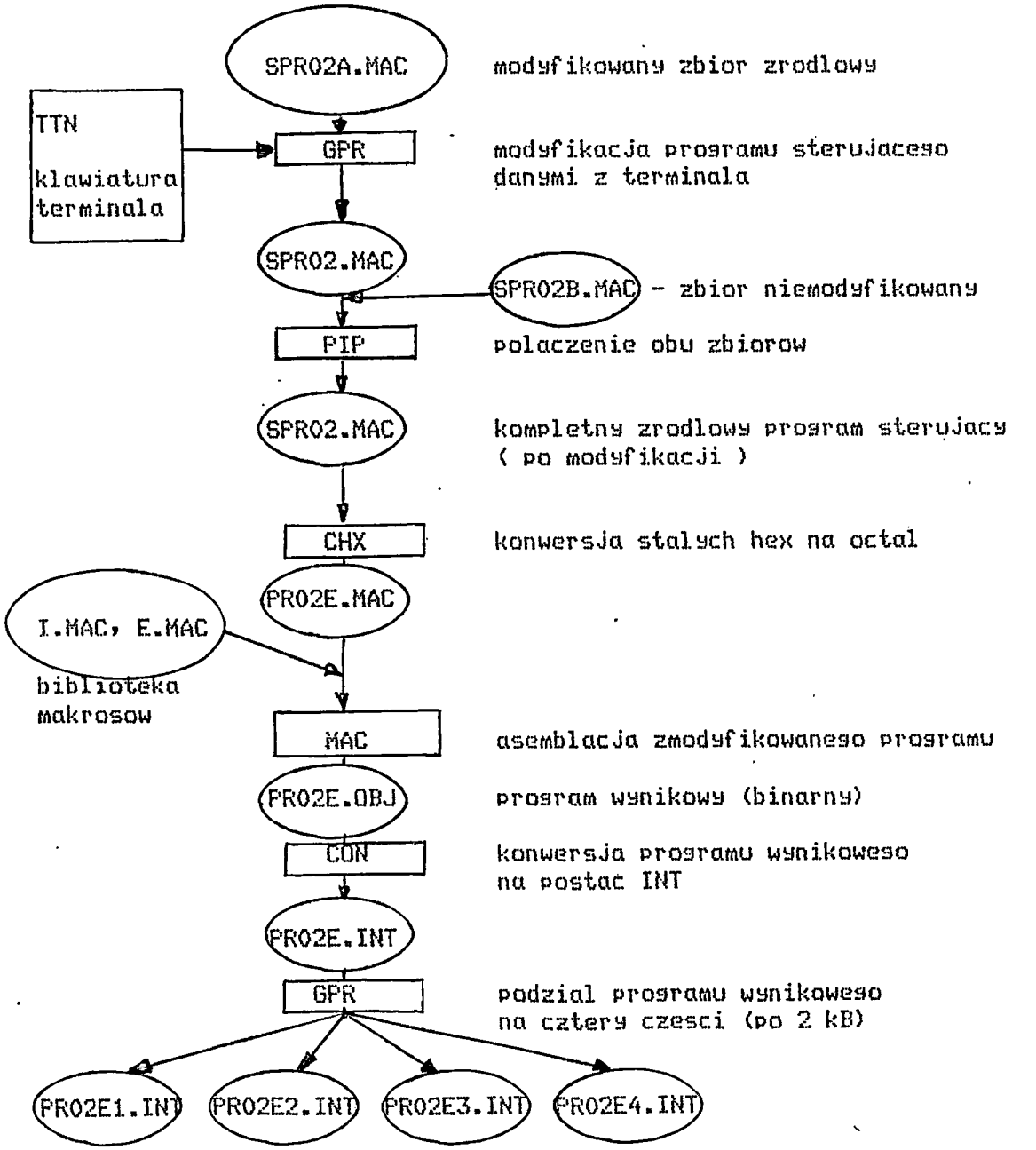
W przypadku zainteresowania klientow zestawami robotow o wiekszej liczbie modulow (rowniez bardziej zlozonych, np.: trojpolozeniowych), generator ten moze byc odpowiednio rozbudowany.

2. Budowa generatora i przebieg generacji

Sposób generacji programow sterujacych polega na wykonaniu ciagu komend monitora systemu operacyjnego RSX-11M (DOS-RW) uruchamianych automatycznie po zainicjowaniu pliku rozkazowego (command file) o nazwie GENPR.

Modyfikacje programu zrodlowego sa wykonywane przez operatora z klawiatury terminala, przy pomocy konwersacyjnego programu GPR. Program ten, napisany w jezyku C, przygotowuje rowniez wynikowa postac programu sterujacego, pamietanego w standardowym kodzie heksadecymalnym INT (znaki w kodzie ASCII, zgrupowane w rekordy), dostosowujac ja do wymagan programatora pamieci 2716, wykorzystywanej w sterowniku PR-02 SP.
Program sterujacy miesci sie w czterech jednostkach tej pamieci (czyli nie przekracza 8 kB).

Rysunek 1 obrazuje przeplyw strumienia danych podczas generacji.



Rys. 1. Przetwarzanie strumienia danych w procesie generacji programow sterujacych.

Otrzymane w wyniku procesu przedstawionego na rysunku 1 cztery zbiory z zakodowanym heksadecymalnie programem wynikowym mozna, w najprostszym przypadku, wyperforowac na tasmie papierowej (wykorzystujac program systemowy FLX), a nastepnie kolejno wczytywac do programatora pamieci EPROM.

Jezeli dysponujemy dodatkowo systemem uruchomieniowo -rozwojowym typu RTB3-0, mozemy bezposrednio przeslac zbior PRO2E.INT do pamieci operacyjnej tego mikrokomputera (gdy jest on podlaczony jako terminal minikomputera SM).
W tym celu nalezy wykorzystac program transmisji danych TRANS, pracujacy pod systemem CP/M. Nastepnie uzywajac dyrektyw MONITOR-a (system RTIS) mozna zaprogramowac kolejne jednostki pamieci 2716.

Operator terminala komputera SM-4, wykonujacy generacji nowej wersji programu sterujacego dla zestawu robotow PR-02 powinien dysponowac nastepujacymi zbiorami danych:

SPRO2A.MAC - modyfikowana czesc programu sterujacego (postac zrodlowa),
SPRO2B.MAC - czesc programu sterujacego nie wymagajaca modyfikacji,
GPR.TSK - konwersacyjny program modyfikujacy postac zrodlowa oraz dokonujacy podzialu postaci wynikowej programu sterujacego,
GENPR.CMD - zbior dyrektyw generatora (command file),
DIVINT.CMD - pomocniczy zbior dyrektyw (perforacja tasm papierowych),
CONHEX.TSK - program konwersji programu sterujacego przed kompilacja skrosna,
HOX.TSK - program konwersji po kompilacji skrosnej,
I.MAC - makrosy asemblacji skrosnej,
E.MAC - dyrektywa konca programu sterujacego.

Efektem dzialania systemu generacji jest zbior wynikowy PRO2E.INT (moze byc podzielony na cztery czesci), ktory jest zbiorem wejsciowym dla programatora pamieci stalych. Wszystkie potrzebne do generacji zbiory zajmuja okolo 350 blokow pamieci dyskowej (mieszcza sie na jednej dyskietce).

3. Wprowadzanie parametrow generacji z terminala

Operator inicjuje prace generatora dyrektywa MCR-a:

@GENPR <CR>.

W toku wykonywania sie dyrektyw ze zbioru GENPR, operator odpowiada na zadawane przez system pytania, wpisujac w przypadku odpowiedzi

TAK - litere Y <CR>,
NIE - litere N <CR> lub tylko <CR>.

Po wczytaniu do pamieci dyskowej (z dyskietki o nazwie SWPR02) zbiorow, wymienionych w poprzednim punkcie i stanowiacych calosc systemu generujacego nowy program sterujacy, operator wprowadza dane zwiazane z nowa konfiguracja grupy robotow prostych. Wprowadzanie danych z klawiatury terminala nalezy dokonywac zgodnie z podanym zawsze przykladem i dopiero po wydruku informacji "Wprowadz dane:".

Postepujac zgodnie z zaleceniami wyswietlanymi na monitorze terminala operator wprowadza kolejno nastepujace dane:

liczba robotow,
liczba wejsc obiektowych (maksymalnie dla jednego robota),
liczba wyjsc obiektowych " - " " - " " - " " - " ;

Przyklad:
3,5,8. ,

a nastepnie dla kolejnych robotow:

liczba modulow,
adres pakietu we/wy obiektowych (tylko mlodszy bajt adresu),
adres pakietu wyjsc sterujacych modulami (" - " - " - " - " - "),
adres pakietu wejsc sprzezenia zwrotnego (" - " - " - " - " - "),
0 albo 1, w zaleznosci od tego czy we/wy obiektowe sa podlaczone do pakietu wykorzystywanego przez poprzedni robot (zero, gdy nie sa);

Przyklad:
6,0E6H,0F2H,0F0H,0 ,

liczba modulow ze sprzezeniem

Przyklad:
3.

Na podstawie powyzszych danych system okresla maski sygnalow sprzezenia zwrotnego, a nastepnie tworzy jeden zbior, zawierajacy kompletny zrodlowy program sterujacy po modyfikacji.

4. Sygnalizacja bledow

W trakcie generacji moga wystapic bledy dwoch kategorii:

a.) blad operatora,
na przyklad: brak kropki dziesietnej, znaku H dla liczb heksadecymalnych, za duzo lub za malo danych, zla wartosc parametru (liczba modulow ze sprzezeniem wieksza od liczby wszystkich modulow), itp;

b.) bledy zwiazane z obsluga pamieci zewnetrznej,
na przyklad: nie mozna otworzyc zbioru wejsciowego (SPR02A.MAC, PR02E.INT), brak dostatecznej ilosci wolnych blokow na kasecie wymiennej, itp.

Bledy sa sygnalizowane na monitorze terminala komunikatem:

Sorry, there was an error;
Error on input file;
Cannot "file name" open (create).

Wszystkie zmodyfikowane zrodlowe linie programu sterujacego sa wyswietlane na monitorze terminala.

W przypadku wystapienia bledow kompilacji programu sterujacego nalezy ponowic sama kompilacje (gdy przyczyna bledu bylo np. zbytnie obciazenie SM-a przez uzytkownikow) albo rozpoczac generacje od nowa - inicjujac zbior komend GENPR.CMD (gdy byla pomylka operatora).

Załącznik nr 1


```
/* GENERATOR GPR programow sterujacych dla zestawow robotow PRO2 *
 * Zbior GSPRO2.C zawiera funkcje: min, strcpy, getline, index, omit,*
 *                                 afile, modline, main              *
 * Zbior GRPD.H zawiera definicje stalych i zmiennych globalnych     */
;
#include "stdio.h"
#include "gprd.h"
;
        min(a,b)
        int a, b;
        {
                return (a <= b? a: b);
        }
        strcpy(s,t)                     /* copy string t to s */
        char *s, *t;
        {
        int ti ;
                ti = t;
                while (*s++ = *t++);
                return (t - ti);
        }
;
        getline(s,lim)                  /* get line from stdin into s */
        char *s;                        /* without NL */
        int lim;
        {
        int c, is;
                is = s;
                while (--lim > 0 && (c = getchar()) != EOF && c != NL)
                        *s++ = c;
                *s = EOS;
                return(s-is);           /* return length  */
        }
        index(s,t)                      /* return index t in s, -1 if none */
        char s[], t[];
        {
        int i,j,k;
                for (i = 0; s[i] != EOS; i++) {
                 for (j = i,k = 0; t[k] != EOS && (s[j] & KL)==t[k]; j++,k++);
                 if (t[k] == EOS) return(i);
                };
                return (-i);
        }
;
        omit(partline)                  /* omit all chars to the first space*/
        char partline[];                /* tab or NL in the input line*/
        {
        char *plp;
                plp = partline;
                while (isspace (*plp) == FALSE) plp++;
                return(plp-partline);   /*return the number of omitted chars*/
        }
        FILE *
        afile (fn, ta)                  /* active input and output files */
        char fn[];
        int ta;
        {
        char *fnp;
        FILE *fp;
                fnp = fn;
                if ((fp = fopen (fnp, ta? "w": "r")) == NULL) {
                        printf ("%s Cannot %s\n",fn, ta? "create": "open");
                        exit (4);
                }
                return(fp);
```

```
modline(mod, plus, cond)        /* modyfication of line */
        char mod[];             /* containing mod plus 'plus' */
        int plus, cond;         /* -from char definded by cond */
{                               /* (number or colon) */
int i, c;
char  *sp, *ttp, *lp;
static char chs;
static int lms [LM];
char line [MAX+1], tt [MAX+1];
extern int ii, ch;
extern char *masz[];
extern FILE *fin, *fo;
;
while ((i = fgets (line,MAX,fin)) != NULL)
        {
        if ((c = *mod) == NULL) goto thisline;
        if ((i=index (line, mod)) < 0)
                fputs (line,fo);
                else {
                        if ((c = *(mod + 1)) == USD) {
;                               fputs (line,fo);
                                return(0);
                                }
                        if (cond < 2)
                        for (i = 1; i < plus+1; i++) {
                                fputs (line,fo);
                                fgets (line,MAX,fin);
                                };
                        goto thisline;
                };
        };                                      /* end of while */
        puts("Error on input file");
        return (1);
thisline:
        if (ch != COLON) goto firstrun;
        if (ii < 3) fputs (line, fo);
        fputs (ENDINT, fo);
        return(0);
firstrun:
         for (ttp = tt, lp = line, c = *lp;
                cond? c != COMMA: isdigit (c) == FALSE; ttp++) {
                  if ((*ttp = c) == NULL) break;
                        c = *++lp;
                };
        if (c == COMMA) {
                          lp++;
                        *ttp++ = c;
                        };
        if (cond == 2) {
                        while (plus--).
                                if (*--ttp == SP) ttp++ ;
                        sp = masz [lms [ii]];
                        i = strcpy (ttp, sp);
                        ttp += --i;
                        if (*ttp != EOS) ttp++;
                        lp += min (omit (lp), i);
                        goto intermod;
                        };
```

```
                if (*mod == NULL && plus > 0) {
                                *ttp++ = ch;
                                *(lp++ + plus) = chs;
                                lms [ii] = chs - 48;
                                goto intermod;
                                };
                puts("Wprowadz dane:");
                puts("-------------");
                i = getline (ttp,MAX);
                ch = *ttp;
                ttp += i;
                lp += min (omit (lp), i);
                if (*(ttp-1) == DOT) lp--;
                chs = ch;
intermod:
                i = strcpy (ttp,lp);
outline:
        puts ("*Zmodyfikowany wiersz zrodlowy w programie sterujacym:");
        printf ("%s",tt);
out:
                fputs (tt,fo);
                return(0);                      /* all right */
        }
```

```
/*********************   MAIN   PROGRAM.  ********************************/
        main(argc,argv)
        int argc;
        char *argv[];
        {
        int lr, lm, s;
        char *weh;
        extern int ii, ch;
        extern char *etopr[], *etmask[], *int2k[], *wyhi[];
        extern FILE *fin, *fo;
        FILE *afile();
;
        if (argc > 1) s = *argv[1];
        switch(s) {
case('1'):
         puts("\n_____________________________________________________________");
        puts("\n*GENERACJA programu sterujacego dla zestawu robotow PRO2*\n");
          puts("-------------------------------------------------------\n");
         puts("Zestaw moze zawierac co najwyzej 8 robotow, z ktorych kazdy");
         puts("moze skladac sie  z  co najwyzej 8 modulow ze sprzezeniem");
         puts("lub bez sprzezenia.");
        fin = afile (WE, 0);
        fo  = afile (WY, 1);
;
         puts("\n* Podaj parametry (oddzielajac je przecinkiem):");
         puts("liczba robotow, liczba wejsc, liczba wyjsc obiektowych");
         puts("                  (maksymalne dla jednego robota) ");
         puts("\nUWAGI:");
       puts(" 1. Dane z klawiatury wprowadzamy dopiero po 'Wprowadz dane:'.");
       puts(" 2. Gdy liczba > 7 wowczas konieczna jest kropka.") ;
         puts("\nPrzyklad :\n3,5,8.\n");
        if (modline (MODL, 3, 0)) exit (2);
         lr = ch - 48;
        if ( lr > LRM) goto err;
;
   for (ii = 0; ii < lr; ii++) {
        puts("\n");
 printf("*********************   ROBOT %d   **********************\n", ii+1);
         puts("liczba modulow, adres pakietu we/wy obiektowych,");
         puts("adres pakietu wyjsc sterujacych,");
         puts("adres pakietu wejsc sprzezenia zwrotnego,");
         puts("0 albo 1 (gdy nie ma lub jest kompresja we/wy)");
        puts("\nUWAGA: podajemy tylko mlodszy bajt adresu (heksadecymalnie)");
         puts("\nPrzyklad :\n6,0E6H,0F2H,0F0H,0\n");
        if (modline (etopr [ii], 0, 1)) exit (2);
        if ((lm = ch) > LM+48) goto err;
         puts ("\nLiczba modulow ze sprzezeniem:");
        if (modline (NULL, 0, 0)) exit (2);
        if ((ch = lm - ch) < 0) goto err;
         ch += 48;
        puts ("\nModuly bez sprzezenia:");
        if (modline (NULL, 4, 0)) exit (2);
        };
        puts ("\n****   MASKI  sygnalow  sprzezenia  zwrotnego:  ****");
  for (ii = 0; ii < lr; ii++)
        if (modline (etmask [ii], 5, 2)) exit (2);
        ch = 0;
        modline (EOFA, 0, 0);
   break;
```

```
case('2'):
        weh = WEHE;
divide:
        for (ii = 0, ch = COLON, fin = afile (weh, 0);
                ii < 4; ii++) {
                        fo  = afile (wshi [ii], 1);
                        if (modline (int2k [ii], 0, 0)) exit (2);
                        fclose (fo);
        };
   break;
case('3'):
        weh = WEHD;
        goto divide;
   break;
default:
err:     puts("Sorry, there was an error");
        exit (2);
        };
        exit (1);
        }
/********************  END  OF  MAIN  ***************************************/
```

```
#define MODL ";I.3"
#define KL '\177'
#define EOFA ";*****"
#define ENDINT ":00000001FF"
#define NL '\n'
#define SP ' '
#define DOT '.'
#define USD '$'
#define COMMA ','
#define COLON ':'
#define MAX 80
#define LRM 8
#define LM 8
#define WE "SPR02A.MAC"
#define WY "SPR02.MAC"
#define WEHE "PR02E.INT"
#define WEHD "PR02D.INT"
;
char     *masz []  = {
                "0, 0", "0C0H, 0", "0F0H, 0", "0FCH, 0", "0FFH, 0",
                "0FFH, 0C0H", "0FFH, 0F0H", "0FFH, 0FCH", "0FFH, 0FFH"
                },
         *etopr []  = {
                "TOPR1:", "TOPR2:", "TOPR3:", "TOPR4:",
                "TOPR5:", "TOPR6:", "TOPR7:", "TOPR8:"
                },
         *etmask [] = {
                "TMASK1:", "TMASK2:", "TMASK3:", "TMASK4:",
                "TMASK5:", "TMASK6:", "TMASK7:", "TMASK8:"
                },
        *int2k []  = {
                ":1007F0", ":100FF0", ":1017F0", ":02AE33"
                },
         *wyhi []  = {
                "PR02E1.INT", "PR02E2.INT", "PR02E3.INT", "PR02E4.INT"
                };
int ii, ch;
FILE *fin, *fo;
```

Załącznik nr 2


```
.SETS UC "[25,2]"
.SETS DX "SWPR02"
.SETS S  "SPR02.MAC"
.SETS SA "SPR02A.MAC"
.SETS SB "SPR02B.MAC"
.SETS G  "GENPR"
.SETF A
.ONERR 999
;
.ASK INS CZY POTRZEBUJESZ OBJASNIEN
.IFF INS .GOTO 100
;
!* POSTEPUJ ZGODNIE Z UWAGAMI WYSWIETLANYMI NA MONITORZE.
!  SORRY, NO MORE HELP YET.
;
.100:.ENABLE SUBSTITUTION
SET /UIC= 'UC'
.ASK DK CZY PRACUJESZ NA DYSKU DK2:
.IFF DK .GOTO 110
ASN DK2:=SY:
.GOTO 120
.110:
ASN DK:=SY:
.120:
.ASKN %N PODAJ NUMER TERMINALA NA KTORYM PRACUJESZ
.IF %N GT 7 .GOTO 999
.OPEN GLOPR.CMD
.ENABLE DATA
.ENABLE GLOBAL
.SETS %T "GPR"
.SETS %F "PR02E"
.SETN %N '%N'
.DISABLE DATA
.CLOSE
@GLOPR
;
.ASK K CZY WCZYTAC ZBIORY GENERATORA Z DYSKIETKI DX1:'DX'
.IFF K .GOTO 121
.IFINS CHX REM CHX
.IFINS CON REM CON
MOU DX1:'DX'
PIP =DX1: *.MAC, *.TSK, *.CMD, *.INT
PIP *.*/PU
.121:
.IFNINS CHX INS CONHEX/CKP=YES
.IFNINS CON INS HOX/CKP=YES
.ASK M CZY MODYFIKOWAC PROGRAM STERUJACY
.IFT M .GOTO 125
PIP 'SA'='SA'
PIP 'S' = 'SA'/RE
.GOTO 126
.125:
;
!* PO ZGLOSZENIU SIE TT'%N'> WPISZ CYFRE 1
RUN '%T'
.WAIT TT'%N'
.126:
PIP 'S' = 'SB'/AP
.ASK C CZY KOMPILOWAC NOWA WERSJE PROGRAMU STERUJACEGO
.IFF C .GOTO 129
;
```

```
!* UWAGA: GENERACJA PROGRAMU STERUJACEGO POTRWA OKOLO 15 MIN.
;
.127:
CHX '*F'.MAC='S'
MAC '*F',NL:=I,'*F',E
.DELAY 5S
;
.ASK B CZY BYLY BLEDY KOMPILACJI
;
.IFF B .GOTO 128
.ASK B CZY PONOWIC PROBE KOMPILACJI
.IFT B .GOTO 127
.128:
CON '*F'.INT='*F'.OBJ
PIP '*F'.OBJ;*, '*F'.MAC;*/DE
.129:
PIP 'S';*/DE
;
@DIVINT
.GOTO 300
.999:
! SORRY, ERRORS !
;
.300:
.IFF C .GOTO 450
.ASK A CZY ZAPISAC NOWY PROGRAM '*F'.INT NA DX1:'DX'
.IFF A .GOTO 450
MOU DX1:'DX'
PIP DX1: '*F'.INT;*/DE
PIP DX1:='*F'.INT
.450:
.IFT K .OR .IFT A DMO DX1:
.ASK Z CZY SKASOWAC ZBIORY SYSTEMU GENERACJI (OPROCZ GENPR.CMD) NA KASECIE
.IFF Z .GOTO 700
PIP 'G'. = 'G'.CMD/RE
PIP *.CMD;*, *.MAC;*, *.TSK;*/DE
PIP 'G'.CMD = 'G'./RE
.700:
PIP *.*/PU
```

```
.ASK ET3  CZY PERFOROWAC TASME WYNIKOWA Z PROGRAMEM STERUJACYM
.IFT ET3 .GOTO 130
.ASK ET4  CZY PRZESLAC PROGRAM DO RTDS-a
.IFF ET4 .GOTO 500
!* POPROS OAE-83 O POMOC!
.GOTO 500
.130:
@GLOPR
.ENABLE SUBSTITUTION
;
!* PO ZGLOSZENIU SIE TT'*N') WPISZ CYFRE 4, GDY BUROR PROGRAMATORA
!  MA 2KB  PAMIECI ALBO CYFRE  7, GDY MA 1KB.
RUN '*T'
.140:
.WAIT TT'*N'
!* UWAGA: BEDA PERFOROWANE TASMY DLA PROGRAMATORA PAMIECI 2716 (4 ALBO 7)
;
.OPEN NULL.INT
.CLOSE
ALL PP:
.SETN I 1
.IFNINS FLX INS *FLX
.210:
.ASK T1  CZY PERFOROWAC TASME NR 'I'
.IFF T1 .GOTO 220
FLX PP:= '*F''I'.INT/RS
FLX PP:= NULL.INT/RS
.220:
.INC I
.IF I LE 7 .GOTO 210
PIP NULL.INT;*/DE
DEA PP:
.ASK ET5  CZY SKASOWAC POWSTALE PODCZAS PODZIALU ZBIORY *.INT
.IFF ET5 .GOTO 500
.SETN I 1
.230:
PIP '*F''I'.INT/DE
.INC I
.IF I LE 7 .GOTO 230
.500:
```