6817 mp

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

Ośrodek Automatyzacji Procesów Produkcji (OAP) BE 10

440

Główny wykonawca mgr inż. Stanisław Wółtański

Wykonawcy: mgr inż. Kazimierz Maliszewski, mgr inż. Stanisław Wółtański

Konsultant

Nr zlecenia S1275

Temat: "Opracowanie dokumentacji generacyjnej systemu operacyjnego SIRTOS".

Etap nr 1: "Wykonanie dokumentacji generacyjnej dla PROM-wej wersji systemu SIRTOS przeznaczonej do posadowienia na j.c. MV-52, zrealizowanej przy wykorzystaniu narzędzi programowych zakupionych przez PIAP".

Zleceniodawca Praca finansowana ze środków statutowych Instytutu

Pracę rozpoczęto dnia 1992.01.15 **zakończono dnia** 1992.03.31

Kierownik Ośrodka
dr inż. M.Wrzesień

Z-ca Dyrektora d/s Badawczo-Rozwojowych
dr inż. J.Jabłkowski

Praca zawiera:

stron	9
rysunków	—
fotografii	—
tabel	—
tablic	—
załączników	4

Nr rejestr. 6817

Rozdzielnik - ilość egz: 3

Egz. 1 BOINTE
Egz. 2 OAP-6
Egz. 3 ZSS
Egz. 4 (OAP-6 /KM)
Egz. 5
Egz. 6

Analiza deskryptorowa :
SYSTEM OPERACYJNEGO CZASU RZECZYWISTEGO
MONITOR OPERATORSKI
STEROWANIE PROCESEM PRZEMYSŁOWYM
URZĄDZENIA AUTOMATYCZNEJ REGULACJI I STEROWANIA:
KSAP + STEROWNIK + MIKROPROCESOR

Analiza dokumentacyjna

Praca zawiera opis generacji wersji PROM-owej systemu czasu rzeczywistego SIRTOS wraz z monitorem operatorskim zawierającym dezasembler instrukcji procesora Intel 80186 jak również aplikacji przygotowywanych do pracy pod tym systemem dla jednostki centralnej MV-52. Wykorzystano zakupione przez PIAP oprogramowanie narzędziowe firm: Microsoft oraz System Software Inc.

Tytuły poprzednich sprawozdań

I. Opracowanie i uruchomienie systemu operacyjnego dla sterownika grupy robotów i elastycznych systemów produkcyjnych:
Zadanie nr 1: Wykonanie wersjiPROM-wej systemu SIRTOS dla jednostki MM86. Sprawozdanie MERA-PIAP, nr rejestr. 6649, 1991
Zadanie nr 2: Wykonanie wersji PROM-wej systemu SIRTOS (obejmującej koprocesor arytmetyczny) dla jednostki AMS-M17-A8. Rozbudowa systemu SIRTOS do pracy z dużym modelem pamięci i instalacja na pakiecie jednostki centralnej PIAP. Sprawozdanie MERA-PIAP, nr rejestr. 6792,1991

II. W ramach zlecenia RP16.1 (CPBR 7.1) wykonano następujące prace:
Wersja systemu SIRTOS do sterowania grupy robotów i ESP.
Etap 1. (Zad. 4.2.1) Opracowanie i uruchomienie monitora operatorskiego czasu rzeczywistego systemu R-T SIRTOS. Dołączenie monitora wraz z deasemblerem do systemu w małym modelu pamięci. Sprawozdanie MERA-PIAP, nr rejestr. 6497, 1990
Etap 2. (Zad. 4.2.2) Generacja systemu SIRTOS dla sterownika grupy robotów i ESP na podstawie dokumentacji techniczno-ruchowej sterownika. Sprawozdanie MERA-PIAP, nr rejestr. 6519, 1990

III. Praca własna OAP
Dezasembler instrukcji procesorów Intel 8086/88/186/188. Sprawozdanie MERA- PIAP, nr rejestr. 6548, 1990

UKD

PIAP 41/88 10000

2

Spis treści

1. WPROWADZENIE I UWAGI OGOLNE

W ramach kontynuacji zlecenia K109.1 p.t.:"Opracowanie i uruchomienie systemu operacyjnego dla sterownika grupy robotów i elastycznych systemów produkcyjnych" podjęto w OAP pracę p.t.:"Wykonanie dokumentacji generacyjnej dla PROM-owej wersji systemu SIRTOS przeznaczonej do posadowienia na j.c. MV-52, zrealizowanej przy wykorzystaniu narzędzi programowych zakupionych przez PIAP. W chwili rozpoczecia tematu (styczeń 1992 roku), jedynym legalnie zakupionym narzędziem był pakiet oprogramowania firmy Microsoft C PDS (Professional Development System) wersja 6.00A. Niestety, pakiet ten nie zawiera programu umożliwiającego po kompilacji i konsolidacji modułów źródłowych oprogramowania dokonania tzw. "romowania" kodu wynikowego, tzn. umieszczenia oprogramowania w pamięci stałej EPROM sterownika. Rozwiązaniem umożliwiającym dokonanie zamierzonego celu okazało się połączenie oprogramowania firmy Microsoft (komplilator języka C, makroasembler, programy pomocnicze typu NMAKE) z zakupionym również przez PIAP oprogramowaniem firmy Systems & Software, Inc.'s, zawierającym n.in. LINKER, LOCATOR oraz programy XOH86 i PROM86 do wygenerowania kodu wynikowego zrozumiałego dla programatora pamięci EPROM. Oprogramowanie to, zgodnie z zapewnieniem producenta, powinno umożliwić wygenerowanie romowalnego kodu dla procesorów Intel 8086/88/186/286 -do mikroprocesurowych sterowników- przy użyciu oprogramowania Microsoft-C wersji 5.0 i późniejszych.

2. UWAGI NA TEMAT EFEKTYWNOSCI UZYTEGO OPROGRAMOWANIA NARZĘDZIOWEGO

W wyniku przeprowadzonego rozpoznania teoretycznego, na podstawie istniejącej dokumentacji oraz po wykonaniu prób na komputerze IBM PC/AT i sterowniku MV-52, wykonawc/ oceniają, że wymienione wyżej pakiety oprogramowania narzędziowego, zakupione przez PIAP, są przydatne w pracach nad przygotowywaniem i rozwijaniem własnego oprogramowania systemowego oraz użytkowego, mającego być w przyszłości przedmiotem oferty handlowej Instytutu. Sądzimy jednak, że zastosowanie jednolitego pakietu jednej firmy, umożliwiającego wygenerowanie romowalnego kodu będzie bardziej efektywne. W następnych punktach pracy zostały bliżej omówione i sprecyzowane trudności wynikajace z użytkowania powyższych pakietów firmowego oprogramowania narzędziowego jak również zostały zasygnalizowane problemy powstające w trakcie przygotowywania i testowania oprogramowania.

3. KOMPILACJA MODUłOW ZRODłOWYCH

Proces kompilacji przy wykorzystaniu oprogramowania Microsoft'a może przebiegać etapowo lub grupowo. W pierwszym przypadku poszczególne moduły źródłowe w języku C lub asemblerze są kompilowane oddzielnie. Poniżej podano typowe komendy kompilacyjne (symbol %1 oznacza nazwę modułu).

```
cl /c /Gs /Od /Zd /GO %1.c

cl /c /Gs /Od /Zd /Fa /G1 /FPi87 %1.c
```

```
ml /c /Zd %1.asm
```

Istotną rzeczą jest wcześniejsze ustawienie odpowiedniej ścieżki w katalogu systemu DOS. np.:

```
PATH=c:\DOS;c:\MSC\BIN;c:\MSC\BINB;c:\LNKLOC\BIN;c:\MSC\LIB;c:\UTILITIE;
```

Drugim sposobem jest kompilacja grupowa. Z uwagi na podaną wyżej wadę dostępnego pakietu Microsoft'a wersja 6.00A (tzn. niemożność wygenerowania romowalnego kodu) przydatność tej metody jest ograniczona. Poniżej podano zawartość zbioru 'makefile' o nazwie 'map' generatora NMAKE dla przykładowej aplikacji przygotowywanej do pracy pod systemem SIRTOS.

```
s=sirtos          /* definicje makrosów wykorzystanych w liniach
                     sterujących generatora NMAKE */
r=startup
l=srt
a=usr
R=startup.obj
L=srt.obj
A=usr.obj
v=vi
w=usersfs
u=user
p=uinit
i=uints
V=vi.obj
W=usersfs.obj
U=user.obj
P=uinit.obj
I=uints.obj
k=tkbd
e=tmon
m=tmmc
b=tbgt
c=ram
d=tads
q=flo
K=tkbd.obj
E=tmon.obj
M=tmmc.obj
B=tbgt.obj
C=ram.obj
D=tads.obj
Q=flo.obj

F=/c /Gs /Od /FPi87
G=/c

$s.exe: $V $R $L $A $W $P $I $U $K $E $M $B $C $D $Q
        LINK @rfile     /* linker użyty jedynie dla potrzeb
                           diagnostyki */
        @echo SIRTOS OK!
        @echo map makefile end

$L: gcc.h gsc.h gse.h        guc.h gue.h                exm.h seg.h
        $(CC) $F $1.c
$V: gcc.h                                       gfe.h exm.h seg.h
        cl $F $v.c
$P: gcc.h gsc.h gse.h ees.h        gue.h gic.h gfe.h exm.h seg.h
        cl $F $p.c
$I: gcc.h gsc.h gse.h ees.h guc.h gue.h gic.h        exm.h seg.h
        cl $F $i.c
$W: gcc.h                    guc.h       gic.h       exm.h
```

```
        cl $F $w.c
$K: gcc.h                ees.h guc.h gue.h
        cl $F $k.c
$E: gcc.h                ees.h guc.h gue.h gic.h
        cl $F $e.c
$M: gcc.h                ees.h guc.h gue.h                                mmc.h
        cl $F $m.c
$B: gcc.h                ees.h       gue.h       gfe.h       seg.h
        cl $F $b.c
$D: gcc.h                                  gic.h             seg.h
        cl $F $d.c
$C: gcc.h                ees.h       gue.h                    seg.h
        cl $F $c.c
$O: gcc.h                                  gue.h
        cl $F $q.c

$A:
        ml $G $a.asm
$R:
        ml $G $r.asm
$U:
        ml $G $u.asm
```

gdzie zawartość zbioru 'rfile' jest następująca:

```
startup vi srt usr usersfs uinit uints user tmmc tkbd tmon tbgt ram tads flo
sirtos.exe ;
```

Należy zwrócić uwagę, że korzystanie z kompilatora i linkera poprzez generator 'nmake' wymaga minimum 640 KB pamięci operacyjnej IBM PC.

4. KONSOLIDACJA MODUŁÓW RELOKOWALNYCH

Faktyczny przebieg procesu konsolidacji modułów relokowalnych, zarówno przy generacji systemu SIRTOS jak i aplikacji wymaga użycia programu konsolidatora 'xlink86' firmy Systems & Software, Inc. Poniżej podano typowe odwołania do tego programu.

```
xlink86    @lnk.cmd
```

gdzie zbiór 'lnk.cmd' dla przykładowej aplikacji do systemu SIRTOS ma postać:

```
startup,vi,srt,usr,usersfs,uinit,uints,user,tmmc,tkbd,tmon,&
tbgt,ram,tads,flo,fli,slibc7.ssi
to sirtos.lnk&
NA(sirtos_apl)
```

Utworzony podczas procesu linkowania zbiór 'sirtos.mpl' ma postać jak w załączniku B.

5. GENERACJA KODU WYNIKOWEGO

Do przygotowania programów aplikacyjnych w postaci wynikowej służy program LOCATOR'a o nazwie 'xloc86. Przykładowe użycie tego programu podaje poniższa linia:

```
xloc86      @locs.cmd
```

gdzie zbiór komend 'locs.cmd' wygląda jak niżej:

```
sirtos.lnk to sirtos.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, DATA_END, &
               BSS, BSS_END, STACK, &
               CODE, CODE_END)) &
RESERVE(20000H to 0EFFFFH) &
addresses(classes(FAR_DATA_BEG(200h),CODE(8000H))) &
NOIC &
NOLI &
NOCM
```

Należy zwrócić uwagę na pewne niezgodności pomiędzy działaniem oprogramowania firmy Systems & Software a dostarczoną wraz z nim dokumentacją. Przykładowo argumenty EXCEPT dla komend PUBLICS i NOPUBLICS programu 'xlink86' nie dają rezultatu; z kolei w trakcie tworzenia oprogramowania wyświetlane są niektóre ostrzeżenia ('warningi') nie odnotowane w dokumentacji. Utworzony podczas procesu binaryzacji kodu zbiór 'sirtos.mp2' ma postać jak w załączniku B.

6. UMIESZCZANIE OPROGRAMOWANIA W PAMIĘCI EPROM

Przed zapisem uruchomionego i przetestowanego oprogramowania w pamięci stałej typu EPROM, należy utworzyć zbiory wymagane przez programatory tej pamięci. Można do tego celu wykorzystać programy 'prom86' i/lub 'xoh86' . Przykładowe wywołania tych programów podają poniższe linie odwołań.

```
xoh86      sirtos.abs to sirtos.apl

prom86 sirtos.abs to sirtos.apl ad(008000h,00BFFFh) initdata hex

prom86 sirtos.abs to sirtos.bin ad(0F0000h,0FFFFFh) initdata split

prom86 s.abs to s.apl ad(0E0000h,0E47FFh) initdata hex noeof
```

Wymienione w powyższych punktach etapy linkowania, lokowania i romowania oprogramowania można połączyć w jeden przebieg, wykorzystując zbiór komend umieszczony w jednym pliku komend analogicznie jak w przykładzie:

```
nmake -d -c -i -f map
del sirtos.exe        /* wynik LINK-a nie będzie wykorzystany */
xlink86      @lnk.cmd
del sirtos.mp1        /* oszczędność pamięci na dysku */
xloc86       @locs.cmd
del sirtos.lnk        /* j.w. */
prom86 sirtos.abs to sirtos.apl ad(008000h,00C7FFh) initdata hex
del sirtos.abs
copy rh+sirtos.apl a
```

Ostatnia linia zawiera uzupełnienie zbioru wynikowego 'sirtos.apl' w postaci 'Intel-hex' o rekord definiujący wartość segmentu kodu (Extended Address Record -Type '02') wymaganego

przez program transmisji programu z IBM PC do pamięci RAM sterownika, w celu jego uruchamiania i testowania. Przykładowa postać takiego rekordu wygląda jak niżej (dla CS=800H):

:020000020800F4

Należy podkreślić, że niektóre programatory pamięci EPROM nie akceptują typu rekordu '02' w formacie Intel'a; użycie programu 'prom86' zamiast 'xoh86' jest wówczas wskazane.

Wykorzystując programator Sunshine Eprom Programmer V 6.1 Model EW-904B do zaprogramowania kości typu 27512, w celu przygotowania właściwych zbiorów 's52.l' i 's52.h'- zawierających monitor OAP, SIRTOS oraz dwie przykładowe aplikacje - mogą być użyte następujące linie sterujące:

```
prom86 sirtos.abs to apl.hex ad(0E0000h,0E37FFh) initdata hex noeof one
prom86 sirtos.abs to ap18.hex ad(0E8000h,0EB7FFh) initdata hex noeof one
prom86 sirtos.abs to sys.hex ad(0F0000h,0F57FFh) initdata hex noeof one

copy rhf000+sys.hex s
   (gdzie zbiór 'rh.f000' zawiera rekord ":02000002F0000C")
copy rhe000+apl.hex a                          (":02000002E0001C")
copy rhe800+ap18.hex a8                        (":02000002E80014")

copy a+a8+s+monitor.hex mv52.hex
\eprom\hexbin2 mv52.hex mv52.bin i E000
\eprom\split2 mv52.bin s52.l s52.h
```

7. WYKORZYSTANIE OPERACJI ZMIENNOPRZECINKOWYCH

Przy uruchamianiu zadań wykorzystujących w języku C operacje zmiennego przecinka powstały niżej zasygnalizowane problemy.

a) Nieprawidłowe wykonanie funkcji testowych 'fp1' i 'fp2' (źródła w załączniku A) na j.c. MV-52, jeżeli programy są kompilowane z opcją /G1 oraz /FPi87, t.j. do wykonania na procesorze 80186+8087. W tej postaci instrukcje koprocesora nie są poprzedzane instrukcjami WAIT, co powoduje zamazywanie, w rejestrach koprocesora, jednych danych przez drugie. Natomiast w trybie pracy krokowej taki program działał prawidłowo.

b) Niemożliwość śledzenia na j.c. MV-52 pracy krokowej instrukcji z opcją /G0 oraz /FPi87. W tym trybie instrukcje koprocesora są przez kompilator poprzedzane instrukcjami WAIT. Stwierdzono, że śledzona w pracy krokowej instrukcja WAIT blokuje wyświetlanie następnej instrukcji (fakt ten został zasygnalizowany np. w "iAPX 86/88, 186/188 User's Manual Hardware Reference", 1985, str. 3-14. Następna instrukcja (która nie musi być nawet instrukcją koprocesora) jest wykonywana, co stwierdzono po wyniku, ale nie jest wyświetlana. Dzieje się tak, ponieważ instrukcja WAIT blokuje wszystkie przerwania (również pracy krokowej) na okres wykonania następnej instrukcji. W istniejącej wersji dezasemblera monitora operatorskiego systemu SIRTOS rozwiązano ten problem w ten sposób, że przed wykonaniem instrukcji WAIT wyświetlana jest również następna instrukcja.

8. PODSUMOWANIE I WNIOSKI

1. Przeanalizowano źródłowy zbiór "emcem.asm". Zawiera on procedury instalacji, wywołania i dołączenia handlera (programu obsługi) przerwań koprocesora arytmetycznego. Procedury te mogą służyć jako wzór do podobnych rozwiązań, ale nie mogą być bezpośrednio użyte, gdyż zakładają pracę w systemie DOS oraz wykorzystanie w systemie jego ekstrakodów; system operacyjny SIRTOS wykonuje np. podobną inicjację sterowników przerwań. Podłączenie obsługi błędów koprocesora pozostawia się użytkownikowi. Błędy koprocesora są wstępnie maskowane w koprocesorze przez system. Program użytkowy może je odmaskować, ale wówczas powinna być dołączona stosowna procedura obsługi tego przerwania. W wypadku wystąpienia odmaskowanego błędu w zadaniu, które nie zadeklarowało jego handlera SIRTOS wykonuje minimalną obsługę zastępczą, polegającą na wypisaniu odpowiedniego komunikatu i wywołaniu monitora operatorskiego.

2. Na podstawie znanej dokumentacji i pracy z pakietem Microsoft C stwierdzono, że deklaracja typu procesora głównego jako 80186/188 powoduje, że kompilator generuje instrukcje koprocesora arytmetycznego nie poprzedzone instrukcjami WAIT (a więc w postaci nieodpowiedniej dla koprocesora 8087). Postać zawierającą WAIT'y można uzyskać tylko przez deklarację procesora głównego jako 8086/88. Taki program wykonuje się na procesorze 80186/188, ale nie wykorzystuje jego specyficznych instrukcji, które przyśpieszają pracę.

3. Wersja narzędziowa systemu oprogramowania Microsoft C 6.00A została wygenerowana i zainstalowana na komputerze IBM PC/AT. Jest ona kompletna w tym sensie, że zawiera procedury, których brak uprzednio sygnalizowano.

4. Pod systemem wymienionym w uwadze 3 wygenerowano s.o. SIRTOS z dezasemblerem, który w czasie śledzenia programu w trybie pracy krokowej wyświetla w jednym kroku zarówno instrukcję WAIT, jak i następującą po niej instrukcję (która jest wykonywana natychmiast po zakończeniu WAIT'a).

W celu usprawnienia wymienionych w sprawozdaniu operacji oraz nadania wyników pracy charakteru handlowego wykonawcy proponują:

a. rozpoznanie możliwości wykorzystania oryginalnego oprogramowania uruchomieniowo-rozwojowego firmy INTEL zakupionego ostatnio (kwiecień 1992 rok) przez PIAP;

b. zakupienie, po uprzednim rozpoznaniu, nowszej wersji pakietu Microsoft'a, dającego możliwość przygotowywania programów 'romowalnych'.

Załącznik A: funkcje 'fp1' oraz 'fp2' - używające instrukcji koprocesora 8087, wykorzystane w zadaniach ADS oraz BGT

```
/*************************************************************************
                        The file "flo.c"  Task: ADS, BGT
                       Functions: memcpy, cdot, fp1, fp2
*************************************************************************/

#include "gcc.h"
#include "que.h"
/*************************************************************************/
static char space [] = "                " ;
static char bu [] = { ON, "\33Y(WTest koprocesora"
                          "\33Y*VADS              BGT" } ;
char buf []= { ON, "\33Ywk                " } ;
char buf2 []= { ON, "\33Ywk               " } ;
static float x, y, z ;
static double u, v ;
/*************************************************************************/
void memcpy ( char *s, char *t, unsigned int c )
    {
    s+= c ;
    t+= c ;
    do {
       *--t = *--s ;
    } while (--c ) ;
    return ;                                                  /* end memcpy */
}
/*************************************************************************/
void cdot ( long liczba, int mode, char *str )
    {                           /* dekodowanie liczby > 0 na kod ASCII */
    char *st ;                  /* liczba -dekodowana liczba */
                                /* mode:typ liczby INT = 0 albo FLOAT = 1,2,3 */
    if ( liczba < 0 ) {         /* str -wskaznik na pole jednosci */
        *( str- 1 ) = '-' ;     /* wpisanie "-" przed polem jednosci */
        liczba = -liczba ;
    }
    *str = '0' ;                /* wpisanie zera przed kropka dziesietna */
    st = str+ 1 ;               /* wskaznik na kropke */
    str+= mode ;
    do {
        *str-- = liczba% 10 +'0' ;
    } while ( liczba/= 10 ) ;   /* wpisywanie kolejnych cyfr dziesietnych */
    if ( mode ) {
        memcpy( st, st+ 1, mode ) ;
        *st = '.' ;             /* wpisanie kropki dziesietnej */
        if ( *( st+ 1 ) == ' ' )
            *( st + 1 ) = '0' ; /* wpisanie zera za kropka dziesietna */
        if ( *( st+ 2 ) == ' ' )
            *( st + 2 ) = '0' ; /* wpisanie drugiego zera za kropka */
    }
    return ;                                                  /* end cdot */
}
/*************************************************************************/
void fp1 ()
    {
    int i ;

    SENDM ( &buc_1, bu ) ;                  /* Test koprocesora */
    x = .1 ;
    y = x ;
    for ( i = 0 ; i < 10 ; ++i ) {
        WAITOUT ( buf, 1, 2 ) ;             /* synchronizacja z pisaniem */
        z = 10.* y ;
        z = z* z ;
        if ( i == 1 )
            WAITOUT ( buf, 1, 1 ) ;
        *( buf+ 3 ) = 12+ ' '+i ;           /* wiersz */
        *( buf+ 4 ) = 52+ ' ' ;             /* kolumna */
        memcpy( space, buf+ 5, 11 ) ;       /* spacjowanie bufora */
        cdot( i+ 1, 0, buf+ 6 ) ;           /* zdekodowanie 'i' */
        cdot( z, 3, buf+ 12 ) ;             /* zdekodowanie 'z' */
        SENDM ( &buc_1, buf ) ;             /* wypisanie komunikatu */
        y+= x ;
    }
    return ;                                                  /* end fp1 */
}
/*************************************************************************/
void fp2 ()
    {
    int i ;

    SENDM ( &buc_1, bu ) ;                  /* Test koprocesora */
    u = .2 ;
    for ( i = 0 ; i < 10 ; ++i ) {
        WAITOUT ( buf2, 1, 1 ) ;            /* synchronizacja z pisaniem */
        v = 20.* u ;
```

Załącznik B

```
REX 8086/8087/8088/80186 OBJECT CODE LOCATER, V5.1d                    TIME:  13:05:15 06-02-1992
INPUT FILES: SIRTOS.LNK
OUTPUT FILE: SIRTOS.ABS
CONTROLS SPECIFIED IN INVOCATION COMMAND:
  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, DATA_END,
                 BSS, BSS_END, STACK,
                  CODE, CODE_END))
  RESERVE(20000H TO 0DFFFFH)
  ADDRESSES(CLASSES(FAR_DATA_BEG(200H),CODE(0F0000H)))
  NOIC
  NOLI
  NOCM

SYMBOL TABLE OF MODULE  SIRTOS_SYS
BASE    OFFSET TYPE SYMBOL
F000H   415FH  PUB  AL_DAT8
F000H   41AFH  PUB  AL_M8
F000H   413AH  PUB  AX_R16
F000H   43AAH  PUB  CBOP1
F000H   454BH  PUB  CFRD
0020H   0484H  PUB  CJNP
F000H   4107H  PUB  COMMA
F000H   4191H  PUB  DX_AL
F000H   3E10H  PUB  EXESER
0000H   0000H  PUB  FICR8Q
0000H   0000H  PUB  FIER8Q
F000H   3D9EH  PUB  FILL8N
F000H   3E9AH  PUB  FILLMA
F000H   3D03H  PUB  FILLN
0000H   0000H  PUB  FISR8Q
0000H   0000H  PUB  FJAR8Q
0000H   0000H  PUB  FJSR8Q
F000H   3DE7H  PUB  GETWS
F000H   3A4EH  PUB  GWSI
F000H   4532H  PUB  JFRD
F000H   41C5H  PUB  M16_AX
F000H   3F99H  PUB  MODIFS
F000H   41DCH  PUB  P83
F000H   421AH  PUB  P8E
F000H   463CH  PUB  PC0
F000H   41E8H  PUB  PC6
F000H   4485H  PUB  PD8S
F000H   44DEH  PUB  PD9S
F000H   4474H  PUB  PDES
F000H   45D8H  PUB  PENTER
F000H   4259H  PUB  PF6
F000H   4288H  PUB  PFF
F000H   436EH  PUB  PINT
F000H   45ECH  PUB  PLEAVE
F000H   42BAH  PUB  PUSHR16
F000H   4154H  PUB  R8_DAT8
F000H   3F51H  PUB  REGB
F000H   4133H  PUB  RIRM32
F000H   411CH  PUB  RM8_R8
F000H   3A65H  PUB  SBLIMD
F000H   43B1H  PUB  SGSI
F000H   3982H  PUB  SPAC
F000H   4176H  PUB  TXALC
F000H   3A1CH  PUB  WRCS
0000H   0000H  PUB  __fltused
0020H   00DAH  PUB  _adrinfs_
0020H   0000H  PUB  _bdata
0020H   0000H  PUB  _bfdata
F000H   34AFH  PUB  _brp
F000H   0EE5H  PUB  _changestack
F000H   3603H  PUB  _cpy
F000H   4C0AH  PUB  _dec2
F000H   0F99H  PUB  _dicli
0020H   0630H  PUB  _edata
0020H   0000H  PUB  _efdata
F000H   4DADH  PUB  _eic
0020H   0222H  PUB  _en
F000H   1033H  PUB  _eoi
F4E0H   0000H  PUB  _etext
F000H   08F3H  PUB  _getbuf
F000H   4C9CH  PUB  _getch_wait
F000H   4BCAH  PUB  _hex4
0020H   0224H  PUB  _hnlsq_
F000H   4D75H  PUB  _inbyte
F000H   011AH  PUB  _init_done
F000H   0E62H  PUB  _initstack
0020H   05FCH  PUB  _ins_
0020H   00CCH  PUB  _int_

BASE    OFFSET TYPE SYMBOL
F000H   4183H  PUB  AL_DX
F000H   416DH  PUB  AX_DAT16
F000H   3A9CH  PUB  BX0AP1
F000H   40C8H  PUB  CDAT
F000H   457CH  PUB  CJ
F000H   4576H  PUB  CNRD
F000H   3D8EH  PUB  CONVAL
F000H   43D1H  PUB  ESDI
0000H   0000H  PUB  FIAR8Q
0000H   0000H  PUB  FIDR8Q
F000H   396EH  PUB  FILL25
F000H   3991H  PUB  FILLH8S
F000H   3E51H  PUB  FILLMAE
F000H   3D68H  PUB  FILLNW
0000H   0000H  PUB  FIWR8Q
0000H   0000H  PUB  FJCR8Q
F000H   3A28H  PUB  GBA
F000H   3A40H  PUB  GWA
F000H   3A07H  PUB  HARDNL
F000H   4551H  PUB  JNRD
F000H   41C0H  PUB  M8_AL
F000H   41CAH  PUB  P8082
F000H   41F3H  PUB  P8C
F000H   4228H  PUB  P8F
F000H   4644H  PUB  PC1
F000H   4234H  PUB  PD0
F000H   4460H  PUB  PD9M
F000H   450EH  PUB  PDBS
F000H   4520H  PUB  PDFS
F000H   441AH  PUB  PESC
F000H   4272H  PUB  PFE
F000H   4630H  PUB  PIMUL116
F000H   412CH  PUB  PLEA
F000H   4611H  PUB  PPUSHA
F000H   43FDH  PUB  PXLAT
F000H   410CH  PUB  R8_RM8
F000H   3AA6H  PUB  REGNAV
F000H   3FDDH  PUB  RM8
F000H   4628H  PUB  R_PRM16
F000H   3F3DH  PUB  SEGPR
F000H   43E3H  PUB  SGSIBW
F000H   0000H  PUB  START
F000H   39AFH  PUB  WRBUF
0000H   0001H  PUB  __acrtused
0020H   00CAH  PUB  _act_
0020H   00D0H  PUB  _as_
0020H   0000H  PUB  _bfbss
0020H   0000H  PUB  _bhbss
0020H   00C8H  PUB  _bstack_
0020H   00D4H  PUB  _control_
F000H   4C3DH  PUB  _dec
F000H   4CFCH  PUB  _delay
F000H   4DA7H  PUB  _dlay
0020H   0000H  PUB  _efbss
0020H   0000H  PUB  _ehbss
F000H   0F82H  PUB  _eisti
0020H   0630H  PUB  _end
F000H   1042H  PUB  _eoi_186
F000H   0F6AH  PUB  _gadrinf
F000H   12E2H  PUB  _getch
F000H   4B58H  PUB  _hex2
0020H   05EAH  PUB  _hextab
F000H   0FA0H  PUB  _icini
F000H   0001H  PUB  _init_begin
F000H   11ECH  PUB  _inithz
F000H   4DCFH  PUB  _inr
F000H   0D49H  PUB  _insta
0020H   00CEH  PUB  _inter_
```

11

Załącznik B

```
REX 8086/8087/8088/80186 OBJECT CODE LINKER, V5.1e                    TIME:  11:16:28 05-06-1992
INPUT FILES: STARTUP.obj
             VI.obj
             SRT.obj
             USR.obj
             USERSFS.obj
             UINIT.obj
             UINTS.obj
             USER.obj
             TMMC.obj
             TKBD.obj
             TMON.obj
             TBGT.obj
             RAM.obj
             TADS.obj
             FLO.obj
OUTPUT FILE: SIRTOS.LNK
CONTROLS SPECIFIED IN INVOCATION COMMAND:
   NA(SIRTOS_APL)
LINK MAP OF MODULE SIRTOS_APL
LOGICAL SEGMENTS INCLUDED :
LENGTH ADDRESS ALIGN  SEGMENT          CLASS            OVERLAY
  0000H ------    G    BEGFDATA        FAR_DATA_BEG
  0000H ------    G    FAR_DATA_START  FAR_DATA
  0000H ------    G    ENDFDATA        FAR_DATA_END
  0000H ------    G    BEGFBSS         FAR_BSS_BEG
  0000H ------    G    FAR_BSS_START   FAR_BSS
  0000H ------    G    ENDFBSS         FAR_BSS_END
  0000H ------    G    BEGHBSS         HUGE_BSS_BEG
  0000H ------    G    HUGE_BSS_START  HUGE_BSS
  0000H ------    G    ENDHBSS         HUGE_BSS_END
  0008H ------    G    NULL            DATA_BEG
  0C51H ------    W     DATA           DATA
  0028H ------    W    CONST           CONST
  0000H ------    G    ENDDATA         DATA_END
  0088H ------    W     BSS            BSS
  0000H ------    W    ENDBSS          BSS_END
  1000H ------    G    STACK           STACK
  1FDBH ------    W     TEXT           CODE
  0000H ------    G    C_ETEXT         CODE_END
  0035H ------    W    C_COMMON        BSS
INPUT MODULES INCLUDES :
STARTUP.obj(startup)
VI.obj(vi)
SRT.obj(srt)
USR.obj(usr)
USERSFS.obj(usersfs)
UINIT.obj(uinit)
UINTS.obj(uints)
USER.obj(user)
TMMC.obj(tmmc)
TKBD.obj(tkbd)
TMON.obj(tmon)
TBGT.obj(tbgt)
RAM.obj(ram)
TADS.obj(tads)
FLO.obj(flo)
WARNING 14 Group enlarged
   GROUP:  DGROUP
```

Załącznik B

```
RSX 8086/8087/8089/80186 OBJECT CODE LOCATER, V5.1d                    TIME:  08:50:09 06-03-1992
INPUT FILES: SIRTOS.LNK
OUTPUT FILE: SIRTOS.ABS
CONTROLS SPECIFIED IN INVOCATION COMMAND:
  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, DATA_END,
                 BSS, BSS_END, STACK,
                 CODE, CODE_END))
  RESERVE(20000H TO 0EFFFFH)
  ADDRESSES(CLASSES(FAR_DATA_BEG(200H),CODE(8000H)))
  NOIC
  NOLI
  NOCM

SYMBOL TABLE OF MODULE  SIRTOS_APL
BASE    OFFSET TYPE SYMBOL          BASE    OFFSET TYPE SYMBOL
0000H   0000H  PUB  FIARQQ          0000H   0000H  PUB  FICRQQ
0000H   0000H  PUB  FIDRQQ          0000H   0000H  PUB  FIERQQ
0000H   0000H  PUB  FISRQQ          0000H   0000H  PUB  FIWRQQ
0000H   0000H  PUB  FJARQQ          0000H   0000H  PUB  FJCRQQ
0000H   0000H  PUB  FJSRQQ          0800H   0000H  PUB  START
0800H   0479H  PUB  _GETBUF        0800H   03B2H  PUB  _KILL
0800H   0451H  PUB  _PAUSE         0800H   0401H  PUB  _PERIOD
0800H   0465H  PUB  _PUTBUF        0800H   048DH  PUB  _SENDM
0800H   043DH  PUB  _SIGNAL        0800H   03EDH  PUB  _STARTC
0800H   03C5H  PUB  _STOP          0800H   03D9H  PUB  _STRT
0800H   0415H  PUB  _WAIT          0800H   04A1H  PUB  _WAITM
0800H   0429H  PUB  _WAITOUT       0800H   25A4H  PUB  __aNNaldiv
0800H   265EH  PUB  __aNftol       0800H   25C4H  PUB  __aNldiv
0800H   2504H  PUB  __aNlrem       0000H   0001H  PUB  __acrtused
0020H   00C8H  PUB  __bstack       0000H   0000H  PUB  __fltused
0800H   265EH  PUB  __ftol         0800H   0D52H  PUB  _abo
0020H   00CAH  PUB  _act_          0020H   0FE2H  PUB  _adprt
0020H   0FF4H  PUB  _adrd          0020H   0FE0H  PUB  _adrk
0020H   0D62H  PUB  _akur          0800H   05ACH  PUB  _api
0800H   07C8H  PUB  _apinit        0020H   00D0H  PUB  _as_
0020H   0000H  PUB  _bdata         0020H   0000H  PUB  _bfbss
0020H   0000H  PUB  _bfdata        0020H   0000H  PUB  _bhbss
0020H   0E3CH  PUB  _bu1           0020H   0E4EH  PUB  _bu2
0020H   0E9AH  PUB  _bu3           0020H   0EACH  PUB  _bu4
0020H   0D84H  PUB  _buc_1        0020H   0D7CH  PUB  _buc_2
0020H   0D68H  PUB  _buf          0020H   0FC8H  PUB  _bufwe
0800H   1FDAH  PUB  _cdot          0020H   07C2H  PUB  _clday
0020H   0FE6H  PUB  _clh           0020H   0FECH  PUB  _clm
0020H   07C4H  PUB  _clmonth       0020H   0FF6H  PUB  _cls
0020H   07C6H  PUB  _clyear        0800H   0F66H  PUB  _coa
0020H   00D4H  PUB  _control_      0800H   1D18H  PUB  _coram
0800H   05D4H  PUB  _cpy           0020H   0FF2H  PUB  _cycle
0020H   07F4H  PUB  _day_name      0020H   07C0H  PUB  _day_number
0020H   07E6H  PUB  _day_tab       0800H   06D3H  PUB  _dec
0800H   06A8H  PUB  _dec2          0800H   1140H  PUB  _delay
0800H   04EEH  PUB  _di            0800H   05C3H  PUB  _dis
0800H   0D12H  PUB  _divz          0020H   0D60H  PUB  _dkom
0800H   11EBH  PUB  _dlay          0800H   0E79H  PUB  _dma0
0800H   0E99H  PUB  _dma1          0020H   0FDEH  PUB  _dsval
0020H   0F10H  PUB  _edata         0800H   0FA6H  PUB  _ee
0020H   0000H  PUB  _efbss         0020H   0000H  PUB  _efdata
0020H   0000H  PUB  _ehbss         0800H   04DAH  PUB  _ei
0800H   05C5H  PUB  _eis           0020H   0FFAH  PUB  _end
0800H   0D92H  PUB  _esp           0A6AH   0000H  PUB  _etext
0800H   01F9H  PUB  _everys_       0020H   0FEAH  PUB  _first
0800H   2090H  PUB  _fp1           0800H   219AH  PUB  _fp2
0800H   235EH  PUB  _fp3           0800F   2441H  PUB  _fp4
0800H   073AH  PUB  _getch_wait    0800H   02CEH  PUB  _getcha
0020H   0FC6H  PUB  _halfds        0800H   05F6H  PUB  _hex2
0800H   0668H  PUB  _hex4          0020H   05EAH  PUB  _hextab
0800H   0516H  PUB  _icinit        0800H   0FF2H  PUB  _im0
0800H   1012H  PUB  _im1           0800H   1032H  PUB  _im2
0800H   1052H  PUB  _im3           0800H   1072H  PUB  _im4
0800H   1092H  PUB  _im5           0800H   10B2H  PUB  _im6
0800H   10D2H  PUB  _im7           0800H   11B9H  PUB  _inbyte
0800H   052AH  PUB  _initHZ        0800H   0001H  PUB  _init_begin
0800H   011AH  PUB  _init_done     0800H   11FFH  PUB  _inr
0020H   05FCH  PUB  _ins_          0800H   0502H  PUB  _instal
0800H   0EB9H  PUB  _int0          0800H   0ED9H  PUB  _int1
0800H   0EF9H  PUB  _int2          0800H   0F19H  PUB  _int3
0020H   00CCH  PUB  _int_          0020H   00CEH  PUB  _inter_
0800H   0D32H  PUB  _intb          0800H   1211H  PUB  _inv
0800H   11C4H  PUB  _inword        0800H   1168H  PUB  _irb
0020H   0D92H  PUB  _kas           0800H   1A5AH  PUB  _kbord
0020H   0FE4H  PUB  _keyboard      0020H   00D2H  PUB  _ki_
0020H   0FF0H  PUB  _last          0020H   00D6H  PUB  _level_
0800H   012AH  PUB  _main          0800H   1FA0H  PUB  _memcpy
0800H   0566H  PUB  _mint          0800H   1234H  PUB  _mmc
```

13

Załącznik C

C COMPILER OPTIONS

-MEMORY MODEL-
/AS small model (default)
/AC compact model
/AM medium model
/AL large model
/AH huge model
/AT tiny model (.COM files)

-OPTIMIZATION-

/O enable optimization (same as /Ot)
/Oa ignore aliasing
/Od disable optimizations
/Oe enable registers allocations
/Og enable global optimization
/Oi enable intrinsic functions
/Ol enable loop optimizations
/On disable "unsafe" optimizations
/Op enable precision optimizations
/Or disable in_line return
/Os optimize for space
/Ot optimize for speed (default)
/Ow assume aliasing in function calls
/Ox max. optimization (/Oegilt /Gs)

-CODE GENERATION-

/G0 8086 instructions (default)
/G1 186 instructions
/G2 286 instructions
/Gm put strings in constant segment
/Gc Pascal style function calls
/Gr _fastcall type calls
/Gs no stack checking
/Gt[number] data size threshold
/Gw Windows entry sequence

-OUTPUT FILES-

/Fa[assembly listing file]
/Fb[bound executable file]
/Fc[mixed source/object listing file]
/Fe<executable file>
/Fl[object listing file]
/Fm[map file]
/Fo<object file>
/Fr[source browser info file]
/FR[extended source browser info file]
/Fs[source listing file]

-PREPROCESSOR-

/C don't strip comments
/D<name>[=text] define macro
/E preprocess to stdout
/EP same as /E but no #line
/I<name> add #include path
/P preprocess to file
/U<name> remove predefined macro
/u remove all predefined macros
/X ignore "standard places"

-LANGUAGE-

/Za disable extensions
/Zd line number information
/Ze enable extensions (default)
/Zg generate declarations
/Zi symbolic debugging information
/Zl remove default library info
/Zp[n] pack structs on n-byte boundary
/Zs syntax check only

-FLOATING POINT-

/FPa calls with altmath
/FPc calls with emulator
/FPc87 calls with 8087 library
/FPi inline with emulator (default)
/FPi87 inline with 8087

Załącznik D: Podział pamięci systemu SIRTOS i aplikacji w wersji PROM-owej dla j.c. MV-52

Adres (hex)	Model pamięci w systemie uruchomieniowym	Model pamięci w systemie docelowym
0000:0000 RAM 0000:01FF	wektor przerwań monitora operatorskiego, SIRTOS-a i aplikacji	wektor przerwań monitora operatorskiego SIRTOS-a i aplikacji
0020:0000 RAM	dane zainicjowane systemowe i użytkowe użytkowe dane niezainicjowane stos (podstawowy 4 kB)	dane zainicjowane systemowe i użytkowe użytkowe dane niezainicjowane stos (podstawowy 4 kB)
0FC0:0000 0FC0:03FF 0020:FFFF	obszar danych i stosu monitora OAP	
CS:0000 RAM CS:FFFF	kod zadań użytkowych aplikacji A lub B w RAM kopia użytkowych danych zainicjowanych (CS = 0x800)	
E000:0000 ROM E000:37FF		kod zadań użytkowych aplikacji B w ROM, kopia użytkowych danych zainicjowanych
E800:0000 ROM E800:37FF		kod zadań użytkowych aplikacji A w ROM, kopia użytkowych danych zainicjowanych
F000:0000 ROM F000:57FF	jądro systemu SIRTOS kopia systemowych danych zainicjowanych	jądro systemu SIRTOS kopia systemowych danych zainicjowanych
F800:0000 ROM F800:7F6F	kod monitora operatorskiego OAP	kod monitora operatorskiego OAP
FFF7:0000 FFF7:008F	inicjacja jednostki MV52 i badanie sł.stanu, skok do F800:0 lub E000:0	inicjacja jednostki MV52 i badanie sł.stanu, skok do E000:0000
FFFF:0000 FFFF:000F	argument długiego skoku do FFF7:0000	argument długiego skoku do FFF7:0000