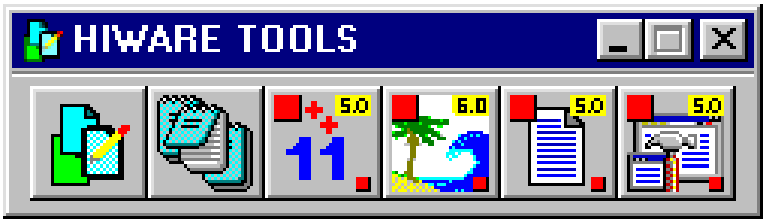
Pakiet HI-CROSS firmy Hiware

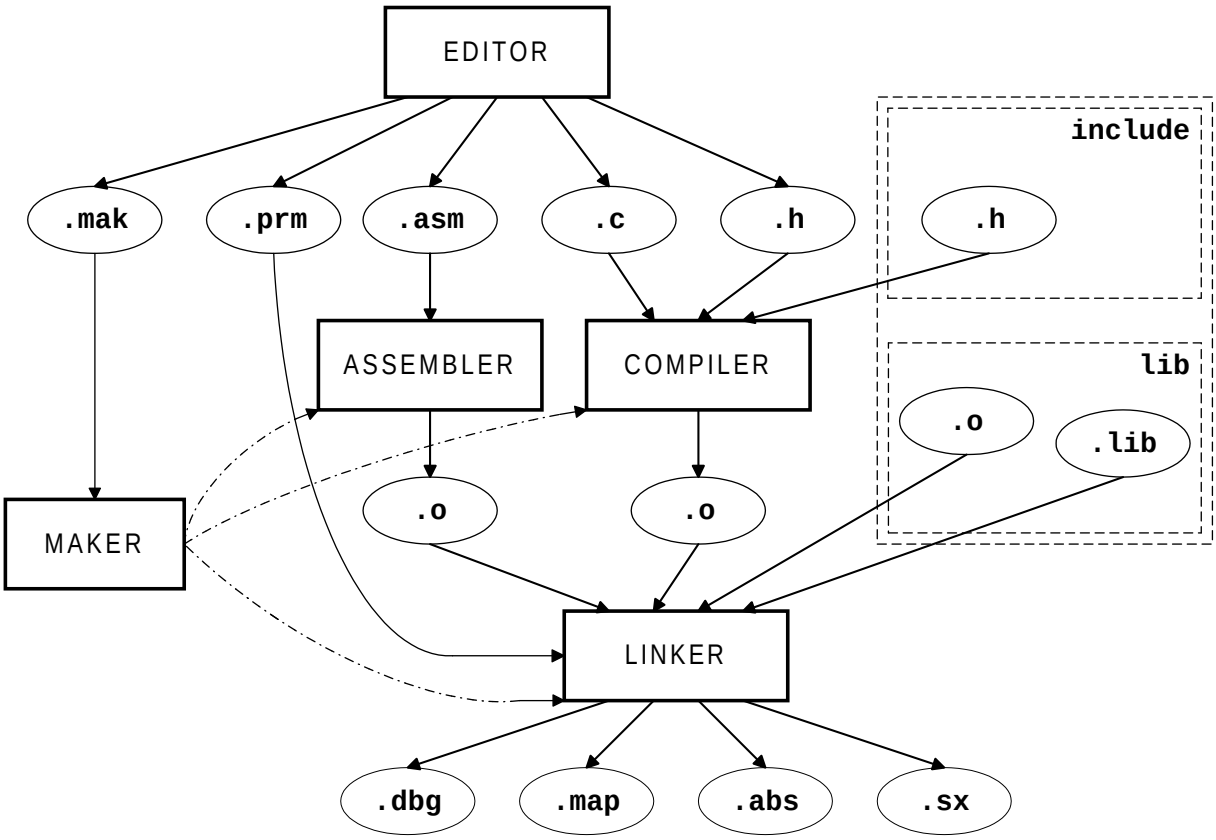


HiTools Setup - konfigurowanie projektu
Editor - tworzenie plików tekstowych
Compiler - kompilacja programów w C
Debugger - uruchamianie i testowanie programów
Decoder - deasemblacja plików **.o*
Maker - sterowanie realizacją projektu

Inne narzędzia pakietu:

Assembler
Burner - programowanie pamięci stałych (EPROM, FLASH)
Libmaker - tworzenie i obsługa bibliotek funkcji

Tworzenie aplikacji




```
/*****************************************
  Demo dla HICROSS
  (c) MW <mw.ict.pwr.wroc.pl>
 *****************************************/
#include <string.h>
const char Napis[] = "Hello!";
void main(void) {
  char buf[80];
  strcpy(buf, Napis);
}
```

Maker (1/2)

Składnia pliku Makefile

MakeFile	=	Entry | Directive.
Entry	=	Macro | Update | Rule.
Macro	=	Name ”=”|”+=”|”=+” Line NL.
Update	=	Name ”:” [Name [”,”] Name] NL Command.
Command	=	WhiteSpace WhiteSpace Line NL.
Rule	=	”.” Suffix [”.” Suffix] ”:” NL Command.
Directive	=	INCLUDE Name NL.
WhiteSpace	=	” ”|”\t”.
NL	=	”\n”.
Line	=	<dowolny znak poza nie cytowanym końcem linii>.
Name	=	<dowolna poprawna nazwa pliku>.
Suffix	=	Letter [Letter] [Letter].
Letter	=	<dowolna litera od ”A” do ”Z” lub od ”a” do ”z”>.

Komendy wbudowane

```
copy file1 file2
del file1 file2... fileN
cd directory
echo text
puts outputfile text
fc file1 file2
fctext file1 file2
rehash
ren file1 file2
```

Maker (2/2)

Makra dynamiczne

$*	nazwa bazowa (bez rozszerzenia i kropki) pliku wynikowego
$@	pełna nazwa pliku wynikowego
$<	pełna lista plików źródłowych
$?	lista plików źródłowych nowszych od pliku wynikowego

Przykładowy plik hello.mak

```
## Makefile dla hello
## (c) MW <mw@ict.pwr.wroc.pl>

FLAGS = -Cc -W2 -Ob

hello.abs : hello.prm hello.o
    $(LINK) -B hello.prm

.c.o:
    $(COMP) $(FLAGS) $<
```

Plik parametrów dla *Linker*-a (*hello.prm*)

```
/*************************************
 Parametry dla linkera
 (c) MW <mw@ict.pwr.wroc.pl>
 *************************************/
LINK hello.abs

NAMES hello.o start11s.o ansis.lib END

SECTIONS
  MY_RAM   = READ_WRITE  0x0000 TO 0x00FF;
  MY_ROM   = READ_ONLY   0xF800 TO 0xFEFF;

PLACEMENT
  DEFAULT_ROM, STRINGS, ROM_VAR INTO  MY_ROM;
  DEFAULT_RAM                   INTO  MY_RAM;
END

STACKSIZE  0x60

VECTOR 0 _Startup   /* wektor restartu */
```

Mapa pamięci dla *hello.abs* (*hello.map*)

```
PROGRAM "hello.abs"

************************************************************
TARGET SECTION
------------------------------------------------------------
Processor   : Motorola HC11
Memory Model: SMALL
File Format : HIWARE
Linker      : HI-CROSS+ SmartLinker V-5.0.9, Sep  8 1999

************************************************************
FILE SECTION
------------------------------------------------------------
  hello.o                                    SMALL  ANSI-C
  start11s.o                                 SMALL  ANSI-C
  string.o             (ansis.lib         ) SMALL  ANSI-C

************************************************************
STARTUP SECTION
------------------------------------------------------------
Entry point: 0xF800
Linker generated code (at 0xF800) before calling __Startup:
SEI
JMP 0xF87D
_startupData is allocated at F804 and uses 30 Bytes
```

```
extern struct _tagStartup{
    unsigned flags                              0
    _PFunc   main                            F824 (_main)
    unsigned dataPage                           0
    long     stackOffset                       60
    int      nofZeroOuts                        0
    _Range   pZeroOut ->                      NONE
    long     toCopyDownBeg                    F822
    _PFunc   mInits   ->                      NONE
    void *   libInits ->                      NONE
} _startupData;

*************************************************************
SEGMENT-ALLOCATION SECTION
-------------------------------------------------------------
Segmentname            Size Type    From       To  Name

ROM_VAR                   7  R      F8BD     F8C3  MY_ROM
FUNCS                    99  R      F824     F8BC  MY_ROM
COPY                      2  R      F822     F823  MY_ROM
STARTUP                  1E  R      F804     F821  MY_ROM
_PRESTART                 4  R      F800     F803  MY_ROM
SSTACK                   60 R/W       0       5F  MY_RAM

ROM size:       C4 (dec:      196)
RAM size:       60 (dec:       96)
```

```
*************************************************************
OBJECT-ALLOCATION SECTION
-------------------------------------------------------------
Type:     Name:           Address:  Size:
VECTOR:
          &_Startup           FFFE      2


MODULE:       -- hello.o --
- PROCEDURES:
          main                F824     1A  (  0) FUNCS

- VARIABLES:
          Napis               F8BD      7  (  1) ROM_VAR


MODULE:       -- start11s.o --
- PROCEDURES:
          Init                F83E     3F  (  1) FUNCS
          _Startup            F87D     16  (  1) FUNCS

- VARIABLES:
          _startupData        F804     1E  (  6) STARTUP


MODULE:       -- string.o --
- PROCEDURES:
          strcpy              F893     2A  (  1) FUNCS
```

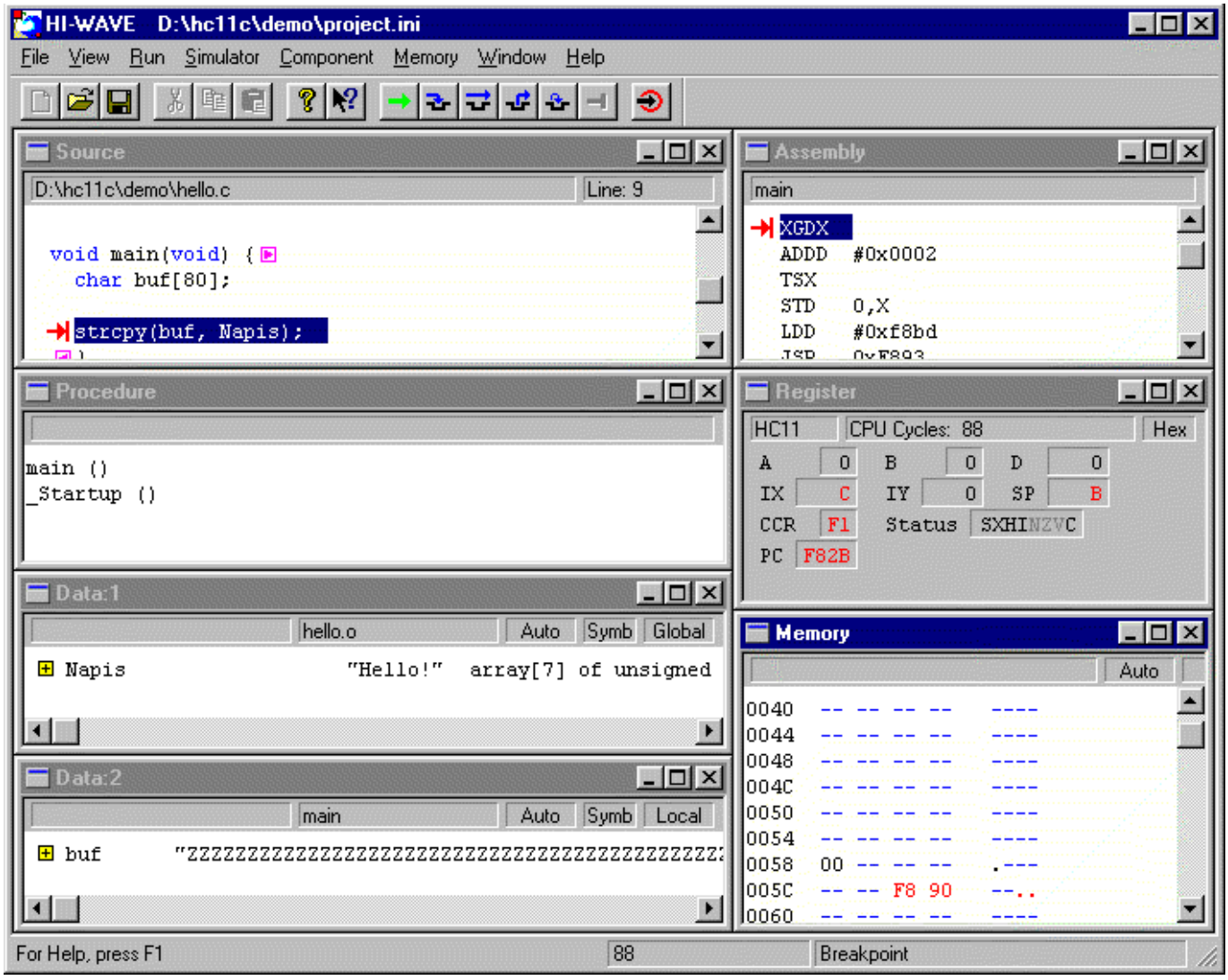
```
*******************************************************
SEGMENT USE IN OBJECT-ALLOCATION SECTION
-------------------------------------------------------
SEGMENT " FUNCS"
          main                  F824     1A  (  0) FUNCS
          Init                  F83E     3F  (  1) FUNCS
          _Startup              F87D     16  (  1) FUNCS
          strcpy                F893     2A  (  1) FUNCS

SEGMENT " ROM_VAR"
          Napis                 F8BD      7  (  1) ROM_VAR

*******************************************************
UNUSED-OBJECTFILE-OBJECTS SECTION
-------------------------------------------------------
NOT USED VARIABLES
hello.o:

*******************************************************
COPYDOWN SECTION
-------------------------------------------------------
*******************************************************
OBJECT-DEPENDENCIES SECTION
-------------------------------------------------------
main                USES
          strcpy                F893     2A  FUNCS
          Napis                 F8BD      7  ROM_VAR
Init                USES
          _startupData          F804     1E  STARTUP
```

```
_Startup            USES
          Init                  F83E     3F  FUNCS
          _startupData          F804     1E  STARTUP

************************************************************
OBJECT LIST SORTED BY ADDRESS
------------------------------------------------------------
          _startupData          F804     1E  ( 6) VAR  STARTUP
          main                  F824     1A  ( 0) CODE FUNCS
          Init                  F83E     3F  ( 1) CODE FUNCS
          _Startup              F87D     16  ( 1) CODE FUNCS
          strcpy                F893     2A  ( 1) CODE FUNCS
          Napis                 F8BD      7  ( 1) VAR  ROM_VAR

************************************************************
STATISTICS SECTION
------------------------------------------------------------
Resources needed during link section:
-------------------------------------
heap memory requested: 669871 bytes
files opened and reopend: 46

ExeFile:
--------
number of blocks to be downloaded: 5
total size of all blocks to be downloaded: 198
```

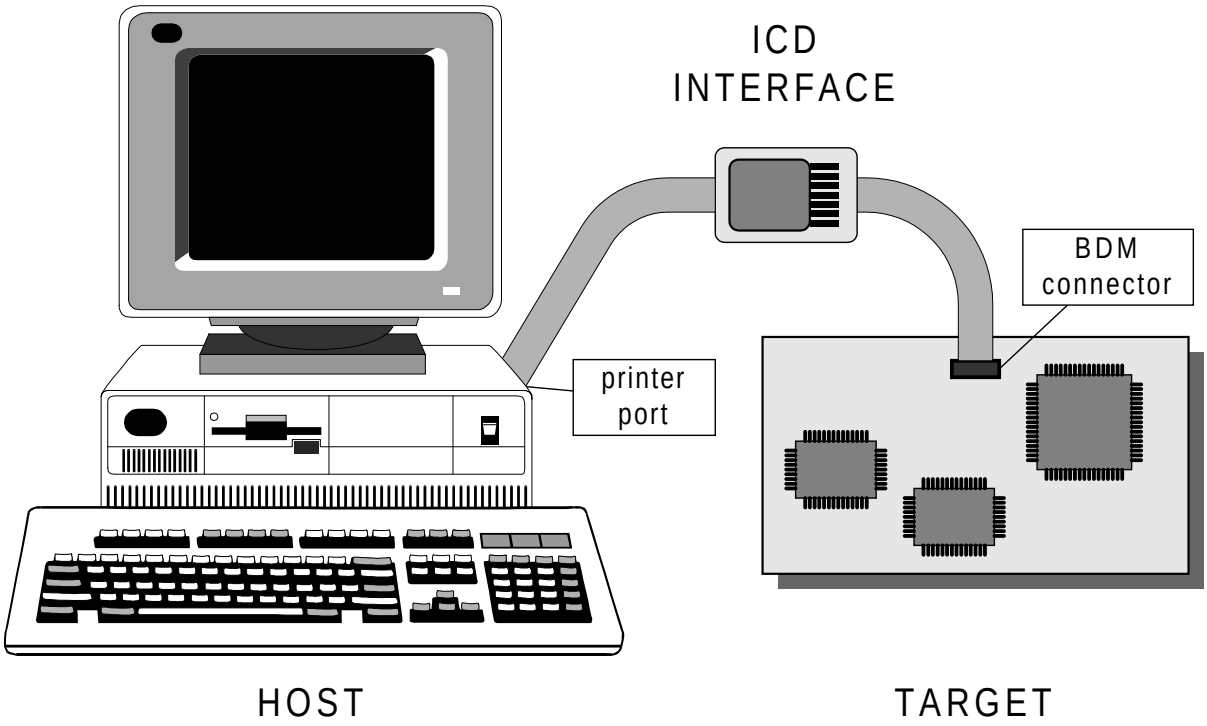
Wynik pracy *Decoder*-a dla *hello.o* (*hello.lst*)

```
Decoding File: 'D:\hc11c\demo\hello.o'

   10:  void main(void) {
   11:    char buf[80];
   12:

00000000 30          TSX
00000001 8F          XGDX
00000002 C3FFAE      ADDD  #0xffae
00000005 8F          XGDX
00000006 35          TXS
   13:    strcpy(buf, Napis);
00000007 8F          XGDX
00000008 C30002      ADDD  #0x0002
0000000B 30          TSX
0000000C ED00        STD   0,X
0000000E CC0000      LDD   @Napis
00000011 BD0000      JSR   strcpy
   14:  }
00000014 30          TSX
00000015 C652        LDAB  #0x52
00000017 3A          ABX
00000018 35          TXS
00000019 39          RTS
```

Debugger HI-WAVE



Interfejs ICD dla HI-WAVE



Skrypt startowy dla interfejsu ICD

```
// Ten plik zawiera komendy dla HI-WAVE wykonywane przy
// uruchamianiu interfejsu ICD dla BDM.
// Inicjalizuje on niektore rejestry w MC68332 aby umozliwic
// dostep do pamieci.
// Dodatkowo zawiera komendy pozwalajace przechwycic wszystkie
// przerwania przez ustawienie w tablicy wektorow wskaznika na
// instrukcje "BGND"
// dla plytki WAN332 (MW)

ww   0xFFFFFA20   0x0006   // SYPCR, System protection
ww   0xFFFFFA04   0x7F00   // SYNCR, Synthesizer control
ww   0xFFFFFA00   0x624A   // MCR,   Module configuration

// ustawianie chip select-ow
ww   0xFFFFFA44   0x3FAB   // CSPAR0, Chip sel. pin assignment
ww   0xFFFFFA46   0x02BE   // CSPAR1, Chip sel. pin assignment
ww   0xF7FFFA48   0x0806   // CSBART, Boot 256k @ 0x080000
ww   0xF7FFFA4A   0x68f0   // CSORBT, Boot options

wl   0xFFFFFA6C   0x00067870   // cs8 : RAM
wl   0xFFFFFA70   0xEFF8D830   // cs9 : LCD

rs           A6   0        // inicjal. listy stosu (0 => A6)

catchtraps    2   255      // przechwycenie przerwan
```