AVR Tool Guide

(Version 2.2)

WIZnet Co., Ltd.

Marketing & Sales: sales@wiznet.co.kr

Technical Support: support@wiznet.co.kr

Table of contents

Table of Figures

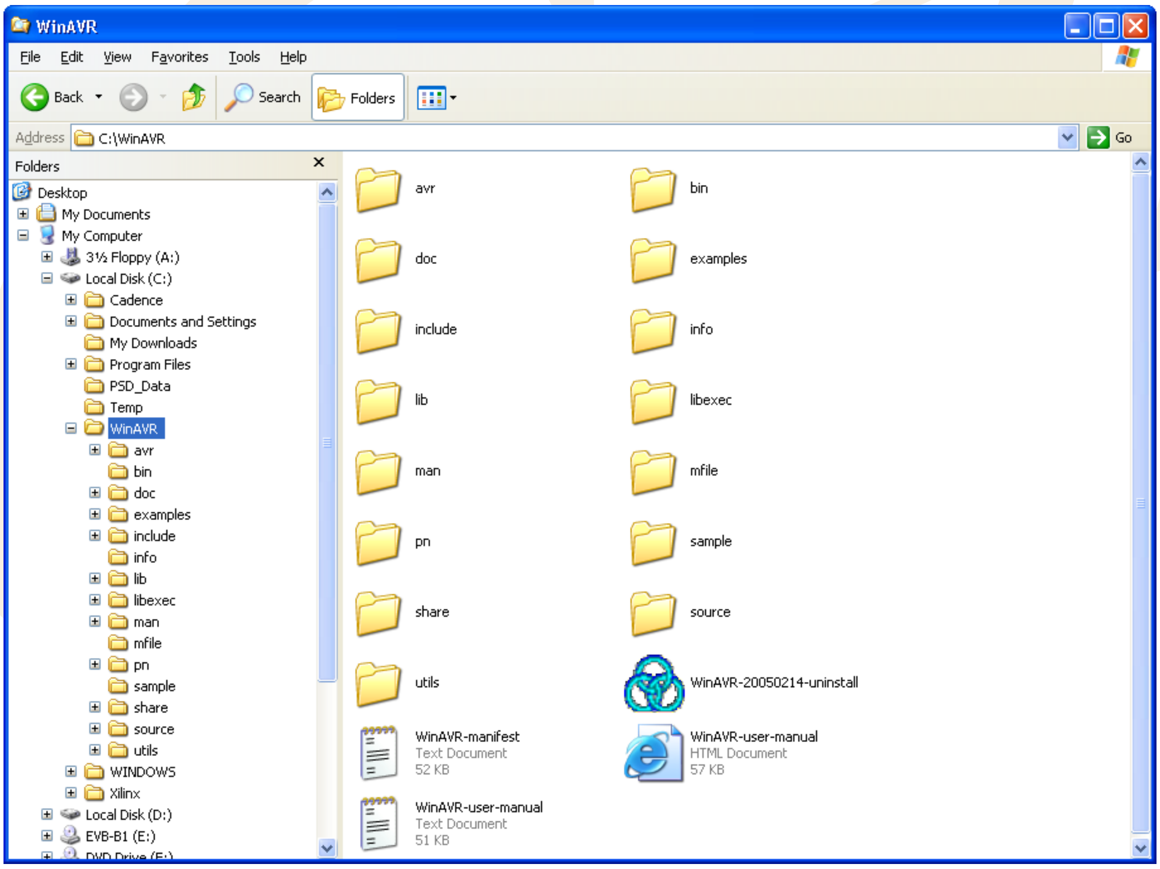
1. WinAVR

WinAVR is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. It includes the GNU GCC compiler. The latest version of WinAVR is available from http://sourceforge.net/projects/winavr.

1.1. Install

1) Run "WinAVR-20050214-install.exe" from CD.
2) You can probably leave most of the settings to their default values.
3) For convenience, choose the default install location "c:\WinAVR".

Figure 1-1. WinAVR Directory.



A. The c:\WinAVR\bin directory contains the software development toolset proper. This includes GNU binutils, GNU GCC, and objtool.

B. The c:\WinAVR\utils\bin contains many miscellaneous Unix or GNU programs that are built for Windows. This includes sh (bash) and make among a host of other things.

C. c:\WinAVR\avr\include contains AVR-GCC header files.

D. c:\WinAVR\avr\lib contains AVR-GCC library files.

1.2. Make a compile

1.2.1. Makefile

There is one program that brings all of this together. This program is GNU make. The make program reads and interprets a makefile. A makefile is a text file that you write that lists and controls how something is made. It is most often used to control how software is made.

Each of these programs is Command Line Interface (CLI) tools. They are controlled by parameters or switches that are added to the command line. Or, in the case of make, by text files that are written and used as input.

Most commercial software development toolsets have an Integrated Development Environment (IDE). This consists of a graphical user-interface (GUI) that contains a programming editor and graphical front-ends to compiler, assembler, linker, standard C library, and librarian programs. These front-ends consist of dialog boxes which allow you to set build options and a way of creating a list of files that are in a "project". These graphical front-ends hide and encapsulate the real command-line compiler, assembler, linker, and standard library that are in the background of any software development toolset.

The template makefile, included C:\WinAVR\sample, should look like this in our version.

Figure 1-2. Contents of makefile.

```
# Hey Emacs, this is a -*- makefile -*-
#----------------------------------------------------------------------------
# WinAVR Makefile Template written by Eric B. Weddington, J?g Wunsch, et al.
#
# Released to the Public Domain
……………………………………
# MCU name
MCU = atmega128
# This should be edited to reflect which AVR mcu you are using.
# Output format. (can be srec, ihex, binary)
FORMAT = ihex
# Target file name (without extension).
TARGET = main
# This is the name of your target.
# List C source files here. (C dependencies are automatically generated.)
SRC = $(TARGET).c
# This is list of 'C' Source files.
…
# List assembly only source file dependencies here:
```

1.2.2. Test of GCC Compile

Let's get try to test gcctest1 which turns on/off LED through PORTG. Figure 1-3 is a source file for gcctest1.

Figure 1-3. gcctest1.c

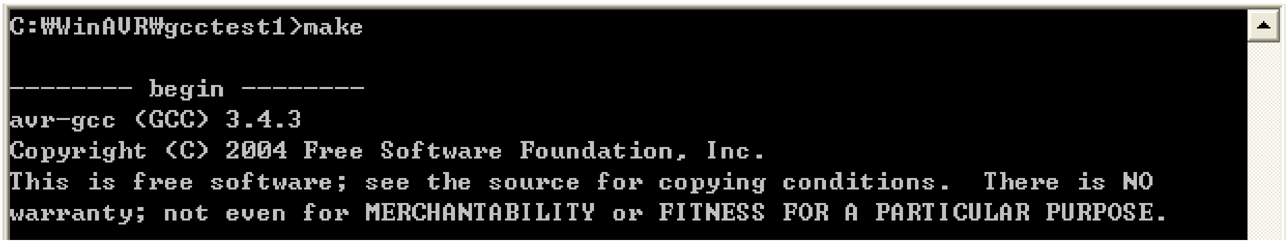
```
#include <avr\io.h>
#define LED1_ON 0xf7                    /* 1111 0111 : PORTG3 on */
#define LED2_ON 0xef                    /* 1110 1111 : PORTG4 on */
int main( void )
{
    unsigned char led;
    unsigned int i, j, k;
    DDRG = 0xff;                        /* use all pins on PortG for output */

    for (;;){
        PORTG = LED1_ON;                /* PORTG3 to zero : LED on */
        for (i=0; i<1000; i++){         /* outer delay loop */
            for (j=0; j<1000; j++)      /* inner delay loop */
            k++;                        /* just do something – could also be a NOP */
        }
        PORTG = LED2_ON;                /* PORTG4 to zero : LED on */

        for (i=0; i<1000; i++)          /* outer delay loop */
          for(j=0; j<1000;j++)          /* inner delay loop */
            k++;                        /* just do something - could also be a NOP */
     }
}
```

1) You should make makefile. Copy template makefile to the directory that gcctest1.c is, and edit TARGET name to gcctest1 as follow.

```
# Target file name (without extension).
#       TARGET = main
        TARGET = gcctest1
```

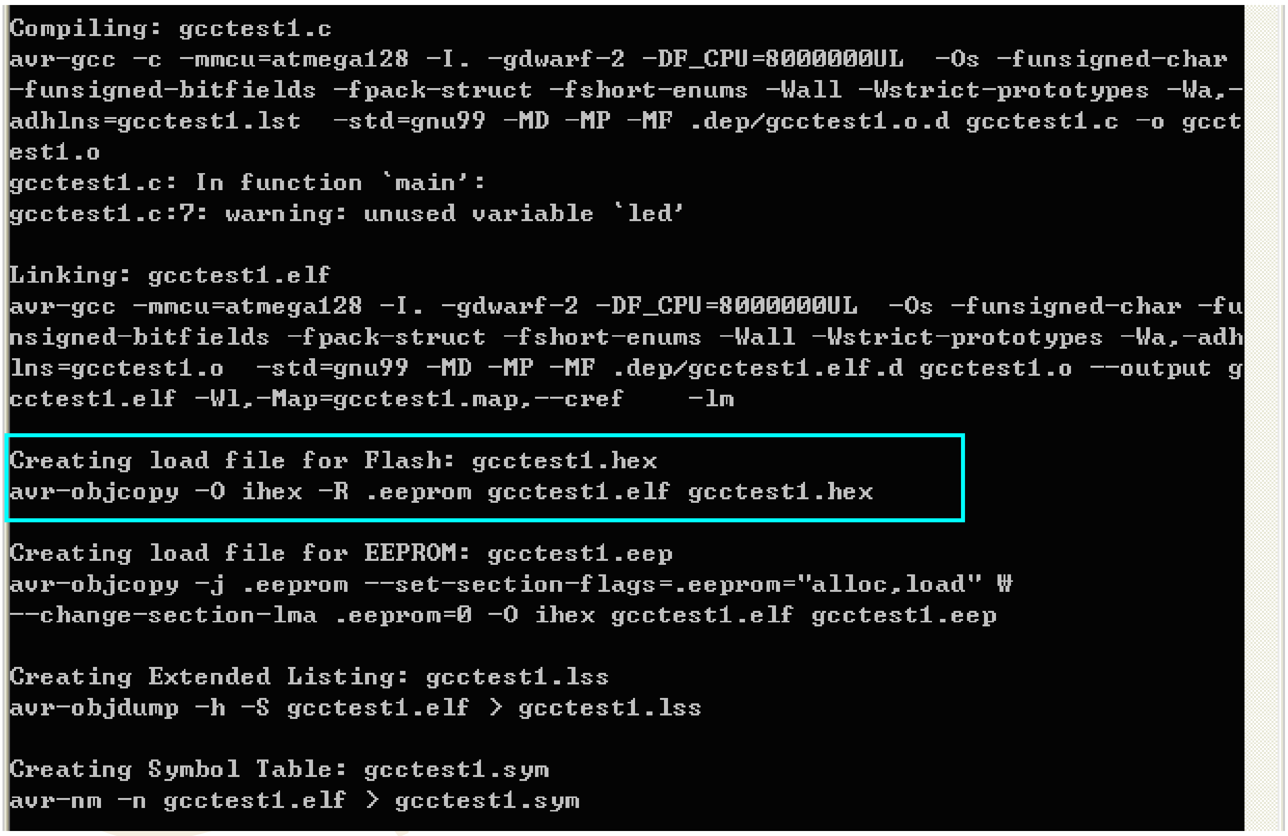
2) Let's make a compile gcctest1.c.
   Move gcctest1 directory, and run "make".

Figure 1-4. make execution screen.



```
C:\WinAVR\gcctest1>make

-------- begin --------
avr-gcc (GCC) 3.4.3
Copyright (C) 2004 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

⋮



```
Compiling: gcctest1.c
avr-gcc -c -mmcu=atmega128 -I. -gdwarf-2 -DF_CPU=8000000UL  -Os -funsigned-char
-funsigned-bitfields -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-
adhlns=gcctest1.lst  -std=gnu99 -MD -MP -MF .dep/gcctest1.o.d gcctest1.c -o gcct
est1.o
gcctest1.c: In function `main':
gcctest1.c:7: warning: unused variable `led'

Linking: gcctest1.elf
avr-gcc -mmcu=atmega128 -I. -gdwarf-2 -DF_CPU=8000000UL  -Os -funsigned-char -fu
nsigned-bitfields -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-adh
lns=gcctest1.o  -std=gnu99 -MD -MP -MF .dep/gcctest1.elf.d gcctest1.o --output g
cctest1.elf -Wl,-Map=gcctest1.map,--cref    -lm

Creating load file for Flash: gcctest1.hex
avr-objcopy -O ihex -R .eeprom gcctest1.elf gcctest1.hex

Creating load file for EEPROM: gcctest1.eep
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 -O ihex gcctest1.elf gcctest1.eep

Creating Extended Listing: gcctest1.lss
avr-objdump -h -S gcctest1.elf > gcctest1.lss

Creating Symbol Table: gcctest1.sym
avr-nm -n gcctest1.elf > gcctest1.sym
```

3) It will make "gcctest1.hex", if you right.

You can get more information from http://www.gnu.org/software/make/ or WinAVR user manual.

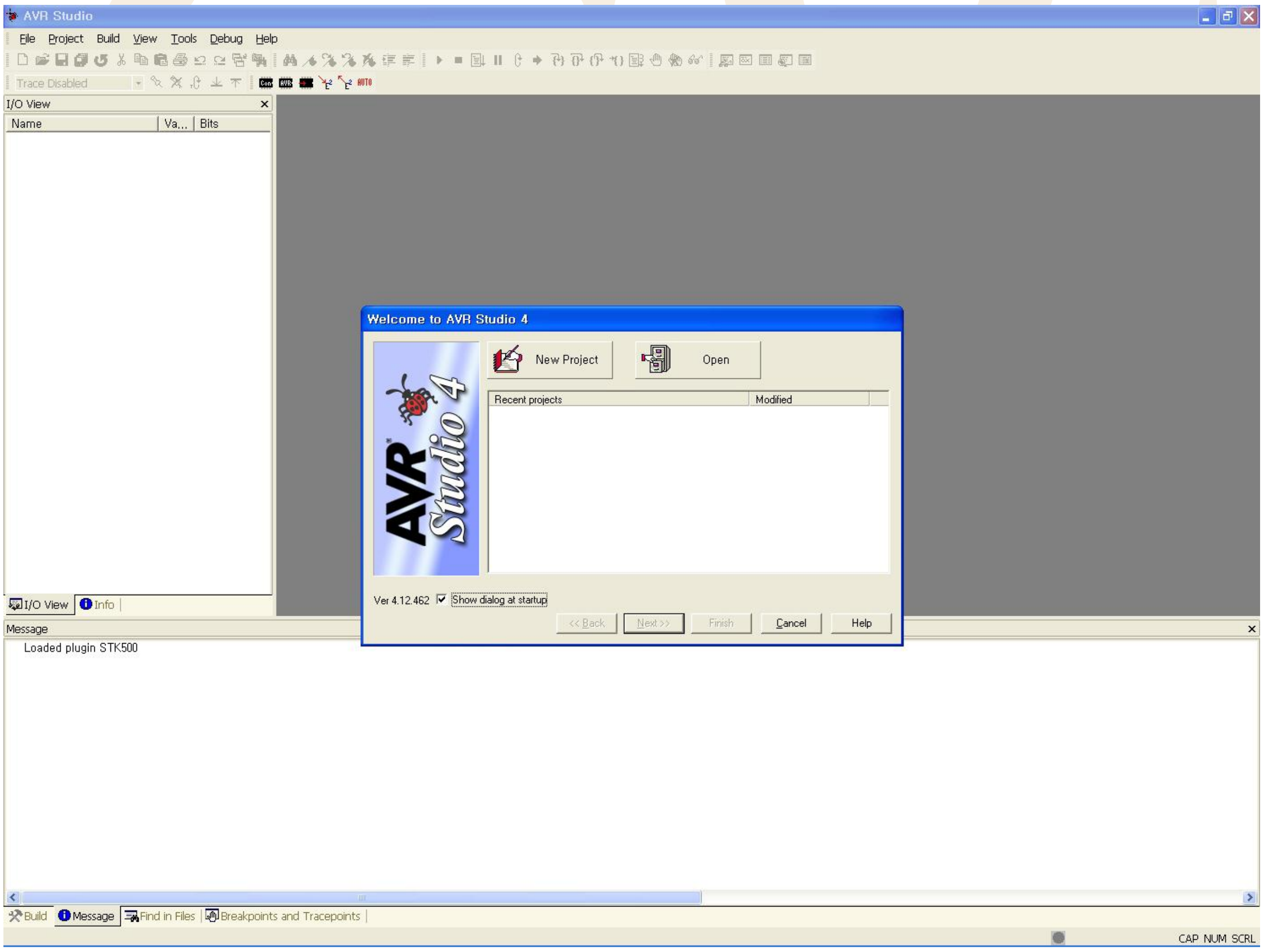
2. AVR Studio

AVR Studio is an Integrated Development Environment (IDE) for writing and debugging AVR applications in Windows 9x/Me/NT/2000/XP environments.   AVR Studio provides a project management tool, source file editor, chip simulator and In-circuit emulator interface for the powerful AVR 8-bit RISC family of microcontrollers. The latest version of AVR Studio is available from http://www.atmel.com .

2.1. Install

1) Start installation through the execution file, “aStudio4b401.exe” in CD
2) Fulfill installation following the direction came out in screen
3) After AVRStudio install process finishing, Start Service Pack install through the execution file, “aStudio411b412SP1.exe” in CD
4) Run this program, start>program>Atmel AVR Tool>AVR Studio 4.

Figure 2-1. AVRStudio’s main screen.

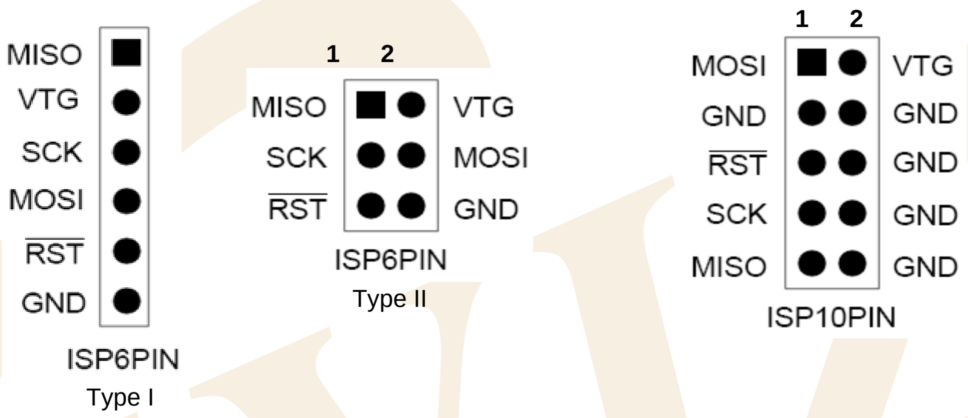


Loaded plugin STK500

2.2. Programming the AVR using AVR ISP Tools

The AVR ISP Tools from Atmel Corporation is an In-System Programmer covering all AVR 8-bit RISC Micro Controllers. The programmer connects to a PC through a standard RS232 serial interface or USB interface and draws the necessary power from the target board eliminating the need for an additional power supply. Figure 2-2 is specification of AVR ISP of Pin mapping. One of every three pin specification is used by WIZnet AVR modules.

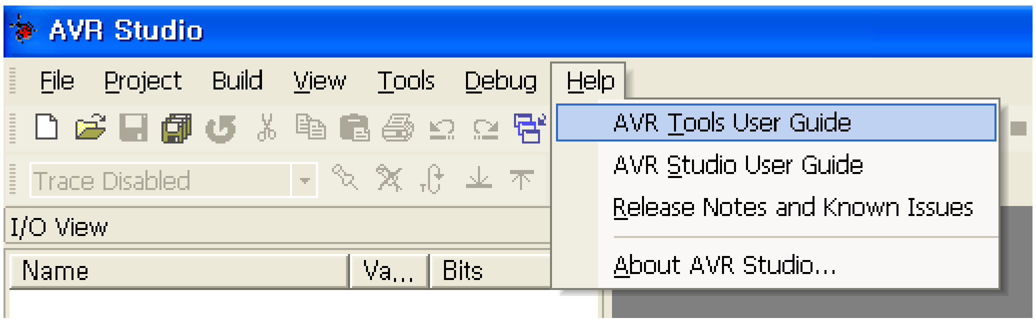
Figure 2-2. Specification of AVR ISP Pin Mapping



SIGNAL	PIN #		I/O	Description
	ISP6PIN	ISP10PIN		
VTG	2	2	-	Power is delivered to the AVRISP
GND	6	3,4,6,8,10	-	Ground
MOSI	4	1	Input	Commands and data from AVRISP to the target board
MISO	1	9	Output	Data from the target to AVRISP
SCK	3	7	Input	Serial Clock, Controlled by AVRISP
/RST	5	5	Input	Reset. Controlled by AVRISP

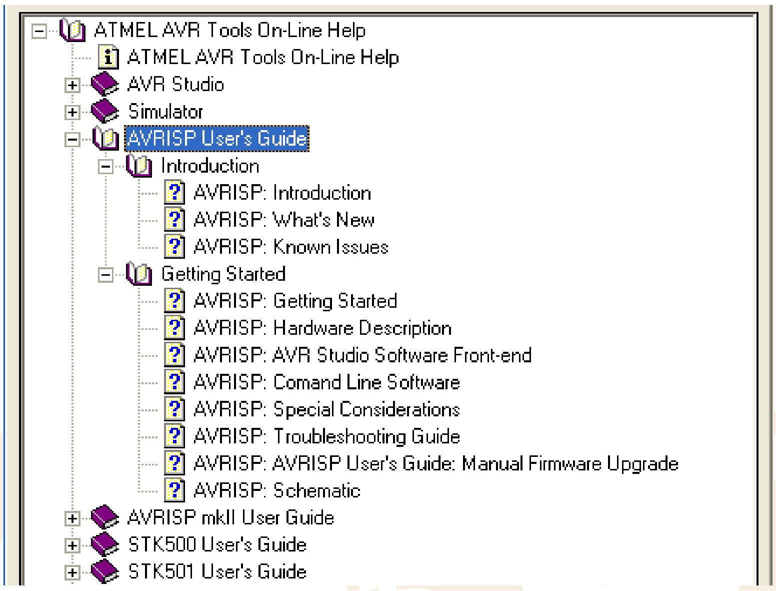
How to Programming using ISP tools refer to 'Help' of AVRStudio.

Figure 2-3. Help Menu of AVRStudio.



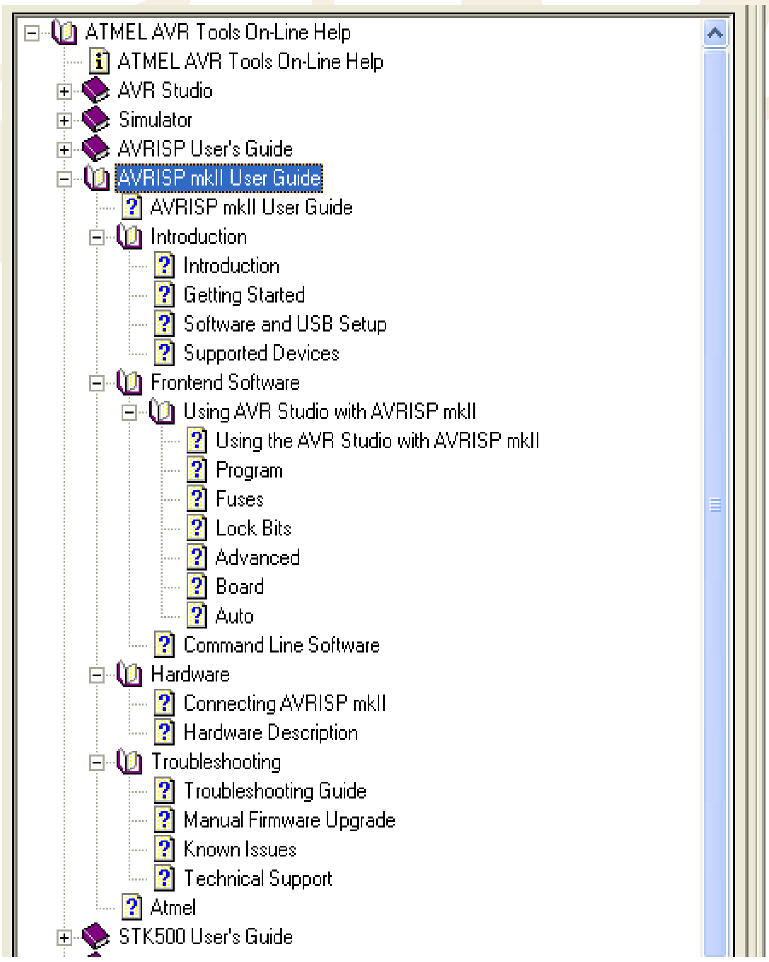
If you click [Help>>AVR Tools User Guide], then show the followed Figures.

Figure 2-4. On-line Help of AVRISP.



ATMEL AVR Tools On-Line Help
- ATMEL AVR Tools On-Line Help
- AVR Studio
- Simulator
- AVRISP User's Guide
  - Introduction
    - AVRISP: Introduction
    - AVRISP: What's New
    - AVRISP: Known Issues
  - Getting Started
    - AVRISP: Getting Started
    - AVRISP: Hardware Description
    - AVRISP: AVR Studio Software Front-end
    - AVRISP: Comand Line Software
    - AVRISP: Special Considerations
    - AVRISP: Troubleshooting Guide
    - AVRISP: AVRISP User's Guide: Manual Firmware Upgrade
    - AVRISP: Schematic
- AVRISP mkII User Guide
- STK500 User's Guide
- STK501 User's Guide

Figure 2-5. The On-line Help For AVRISP mkII.



ATMEL AVR Tools On-Line Help
- ATMEL AVR Tools On-Line Help
- AVR Studio
- Simulator
- AVRISP User's Guide
- AVRISP mkII User Guide
  - AVRISP mkII User Guide
  - Introduction
    - Introduction
    - Getting Started
    - Software and USB Setup
    - Supported Devices
  - Frontend Software
    - Using AVR Studio with AVRISP mkII
      - Using the AVR Studio with AVRISP mkII
      - Program
      - Fuses
      - Lock Bits
      - Advanced
      - Board
      - Auto
    - Command Line Software
  - Hardware
    - Connecting AVRISP mkII
    - Hardware Description
  - Troubleshooting
    - Troubleshooting Guide
    - Manual Firmware Upgrade
    - Known Issues
    - Technical Support
  - Atmel
- STK500 User's Guide

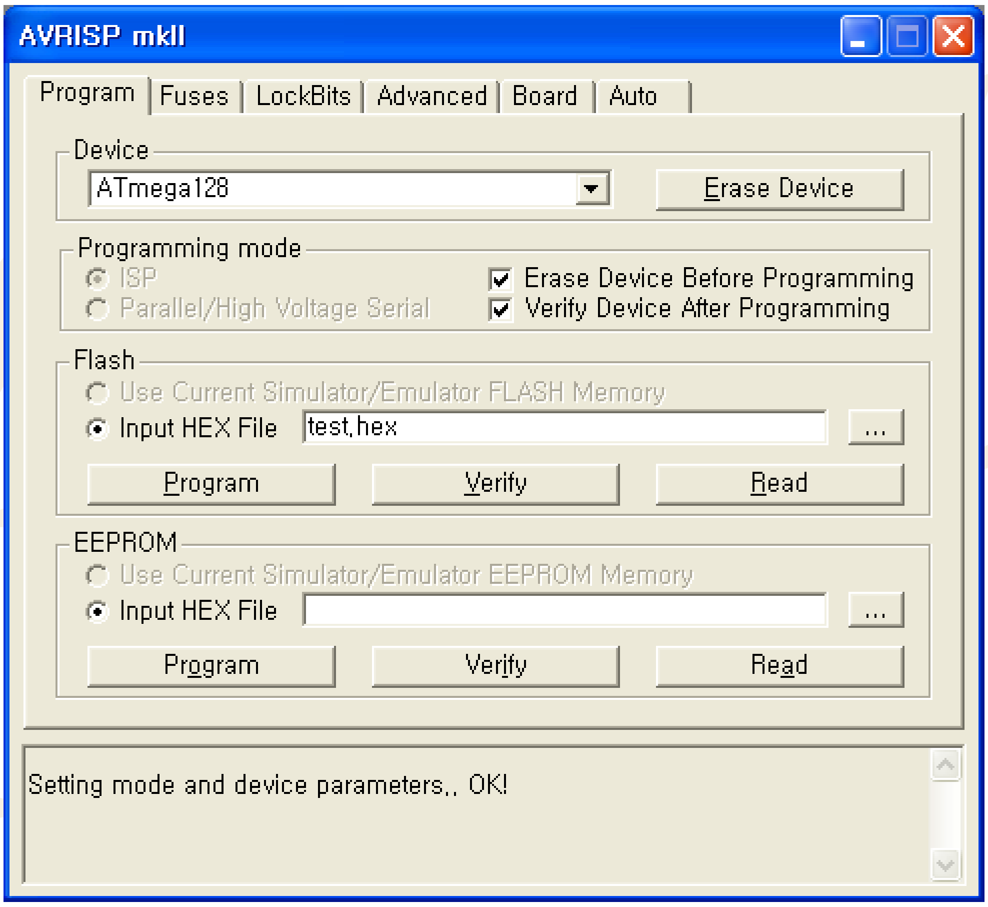
2.2.1. Programming AVR modules of WIZnet

<Notice>

When do you want to program EVB-B1, You must remove the jumper Cap of JP5 on MB-EVB-X1 before it is programmed.

1) Select Device & Programming

After you select 'ATmega128' in 'Decive' Window and browse your file, Click [Program] in 'Flash' Window.



AVRISP mkII

Device: ATmega128

Programming mode: [✓ ISP] Parallel/High Voltage Serial

[✓] Erase Device Before Programming

[✓] Verify Device After Programming

Flash: [✓ Input HEX File] test.hex

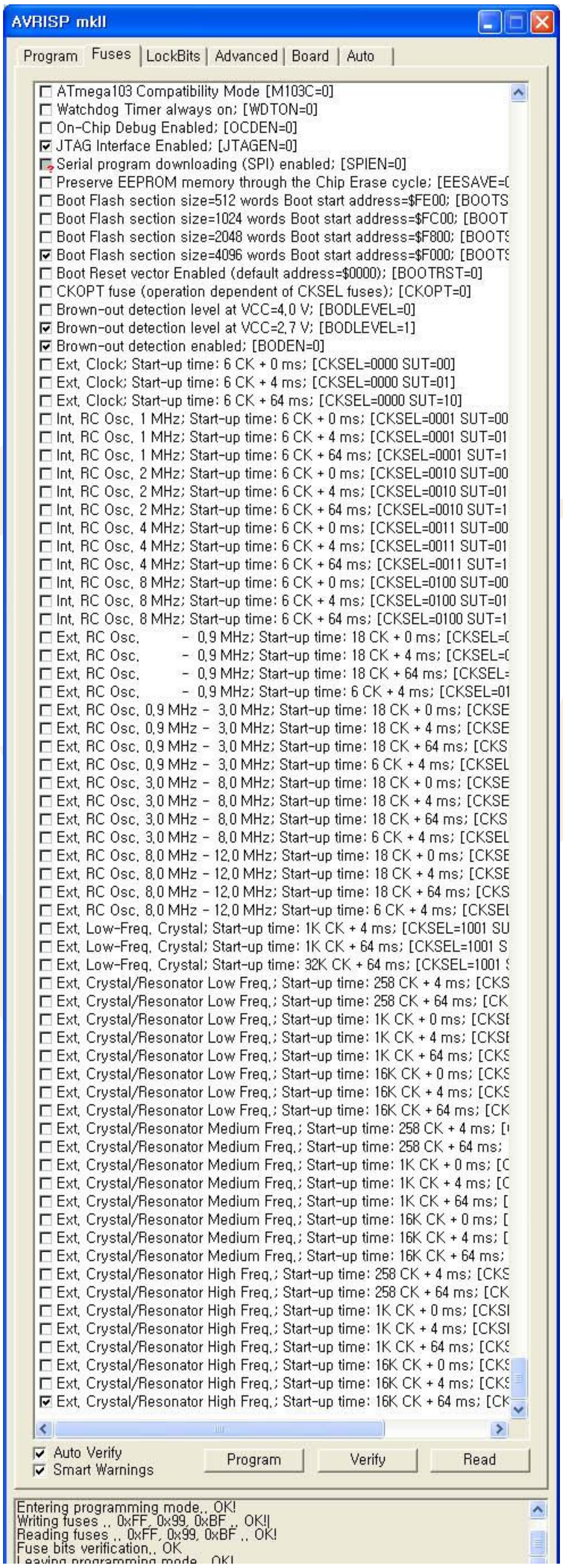
EEPROM: [✓ Input HEX File]

Setting mode and device parameters.. OK!

Program
Fuses
LockBits
Advanced
Board
Auto
Erase Device
Use Current Simulator/Emulator FLASH Memory
Program
Verify
Read
Use Current Simulator/Emulator EEPROM Memory
Program
Verify
Read

2) Fuse Bits Programming

Check Fuse bits like as the followed figures, and then click [Program].



☐ ATmega103 Compatibility Mode [M103C=0]
☐ Watchdog Timer always on; [WDTON=0]
☐ On-Chip Debug Enabled; [OCDEN=0]
☑ JTAG Interface Enabled; [JTAGEN=0]
☑ Serial program downloading (SPI) enabled; [SPIEN=0]
☐ Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=(
☐ Boot Flash section size=512 words Boot start address=$FE00; [BOOTS
☐ Boot Flash section size=1024 words Boot start address=$FC00; [BOOT
☐ Boot Flash section size=2048 words Boot start address=$F800; [BOOTS
☑ Boot Flash section size=4096 words Boot start address=$F000; [BOOTS
☐ Boot Reset vector Enabled (default address=$0000); [BOOTRST=0]
☐ CKOPT fuse (operation dependent of CKSEL fuses); [CKOPT=0]
☐ Brown-out detection level at VCC=4.0 V; [BODLEVEL=0]
☑ Brown-out detection level at VCC=2.7 V; [BODLEVEL=1]
☑ Brown-out detection enabled; [BODEN=0]
☐ Ext. Clock; Start-up time: 6 CK + 0 ms; [CKSEL=0000 SUT=00]
☐ Ext. Clock; Start-up time: 6 CK + 4 ms; [CKSEL=0000 SUT=01]
☐ Ext. Clock; Start-up time: 6 CK + 64 ms; [CKSEL=0000 SUT=10]
☐ Int. RC Osc. 1 MHz; Start-up time: 6 CK + 0 ms; [CKSEL=0001 SUT=00
☐ Int. RC Osc. 1 MHz; Start-up time: 6 CK + 4 ms; [CKSEL=0001 SUT=01
☐ Int. RC Osc. 1 MHz; Start-up time: 6 CK + 64 ms; [CKSEL=0001 SUT=1
☐ Int. RC Osc. 2 MHz; Start-up time: 6 CK + 0 ms; [CKSEL=0010 SUT=00
☐ Int. RC Osc. 2 MHz; Start-up time: 6 CK + 4 ms; [CKSEL=0010 SUT=01
☐ Int. RC Osc. 2 MHz; Start-up time: 6 CK + 64 ms; [CKSEL=0010 SUT=1
☐ Int. RC Osc. 4 MHz; Start-up time: 6 CK + 0 ms; [CKSEL=0011 SUT=00
☐ Int. RC Osc. 4 MHz; Start-up time: 6 CK + 4 ms; [CKSEL=0011 SUT=01
☐ Int. RC Osc. 4 MHz; Start-up time: 6 CK + 64 ms; [CKSEL=0011 SUT=1
☐ Int. RC Osc. 8 MHz; Start-up time: 6 CK + 0 ms; [CKSEL=0100 SUT=00
☐ Int. RC Osc. 8 MHz; Start-up time: 6 CK + 4 ms; [CKSEL=0100 SUT=01
☐ Int. RC Osc. 8 MHz; Start-up time: 6 CK + 64 ms; [CKSEL=0100 SUT=1
☐ Ext. RC Osc. - 0.9 MHz; Start-up time: 18 CK + 0 ms; [CKSEL=(
☐ Ext. RC Osc. - 0.9 MHz; Start-up time: 18 CK + 4 ms; [CKSEL=(
☐ Ext. RC Osc. - 0.9 MHz; Start-up time: 18 CK + 64 ms; [CKSEL=
☐ Ext. RC Osc. - 0.9 MHz; Start-up time: 6 CK + 4 ms; [CKSEL=01
☐ Ext. RC Osc. 0.9 MHz - 3.0 MHz; Start-up time: 18 CK + 0 ms; [CKSE
☐ Ext. RC Osc. 0.9 MHz - 3.0 MHz; Start-up time: 18 CK + 4 ms; [CKSE
☐ Ext. RC Osc. 0.9 MHz - 3.0 MHz; Start-up time: 18 CK + 64 ms; [CKS
☐ Ext. RC Osc. 0.9 MHz - 3.0 MHz; Start-up time: 6 CK + 4 ms; [CKSEL
☐ Ext. RC Osc. 3.0 MHz - 8.0 MHz; Start-up time: 18 CK + 0 ms; [CKSE
☐ Ext. RC Osc. 3.0 MHz - 8.0 MHz; Start-up time: 18 CK + 4 ms; [CKSE
☐ Ext. RC Osc. 3.0 MHz - 8.0 MHz; Start-up time: 18 CK + 64 ms; [CKS
☐ Ext. RC Osc. 3.0 MHz - 8.0 MHz; Start-up time: 6 CK + 4 ms; [CKSEL
☐ Ext. RC Osc. 8.0 MHz - 12.0 MHz; Start-up time: 18 CK + 0 ms; [CKSE
☐ Ext. RC Osc. 8.0 MHz - 12.0 MHz; Start-up time: 18 CK + 4 ms; [CKSE
☐ Ext. RC Osc. 8.0 MHz - 12.0 MHz; Start-up time: 18 CK + 64 ms; [CKS
☐ Ext. RC Osc. 8.0 MHz - 12.0 MHz; Start-up time: 6 CK + 4 ms; [CKSEL
☐ Ext. Low-Freq. Crystal; Start-up time: 1K CK + 4 ms; [CKSEL=1001 SU
☐ Ext. Low-Freq. Crystal; Start-up time: 1K CK + 64 ms; [CKSEL=1001 S
☐ Ext. Low-Freq. Crystal; Start-up time: 32K CK + 64 ms; [CKSEL=1001 S
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 258 CK + 4 ms; [CKS
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 258 CK + 64 ms; [CK
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 1K CK + 0 ms; [CKSE
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 1K CK + 4 ms; [CKSE
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 1K CK + 64 ms; [CKS
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 16K CK + 0 ms; [CKS
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 16K CK + 4 ms; [CKS
☐ Ext. Crystal/Resonator Low Freq.; Start-up time: 16K CK + 64 ms; [CK
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 258 CK + 4 ms; [(
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 258 CK + 64 ms;
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 1K CK + 0 ms; [C
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 1K CK + 4 ms; [C
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 1K CK + 64 ms; [
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 16K CK + 0 ms; [
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 16K CK + 4 ms; [
☐ Ext. Crystal/Resonator Medium Freq.; Start-up time: 16K CK + 64 ms;
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 258 CK + 4 ms; [CKS
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 258 CK + 64 ms; [CK
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 1K CK + 0 ms; [CKSI
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 1K CK + 4 ms; [CKSI
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 1K CK + 64 ms; [CKS
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 16K CK + 0 ms; [CKS
☐ Ext. Crystal/Resonator High Freq.; Start-up time: 16K CK + 4 ms; [CKS
☑ Ext. Crystal/Resonator High Freq.; Start-up time: 16K CK + 64 ms; [CK

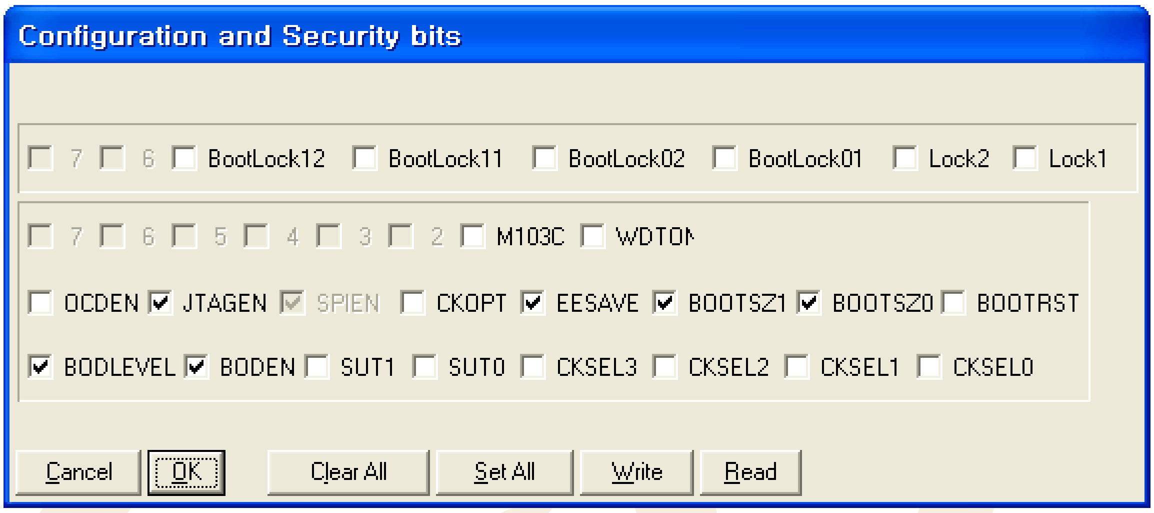
☑ Auto Verify
☑ Smart Warnings

```
Entering programming mode.. OK!
Writing fuses .. 0xFF, 0x99, 0xBF .. OK!
Reading fuses .. 0xFF, 0x99, 0xBF .. OK!
Fuse bits verification.. OK
Leaving programming mode.. OK!
```

3) Lockbits & Other Setting

These are set by default.

4) “Ponyprog2000” users check configure bits like as the followed figure and then Click [Write].


Configuration and Security bits

☐ 7 ☐ 6 ☐ BootLock12 ☐ BootLock11 ☐ BootLock02 ☐ BootLock01 ☐ Lock2 ☐ Lock1

☐ 7 ☐ 6 ☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ M103C ☐ WDTON

☐ OCDEN ☑ JTAGEN ☑ SPIEN ☐ CKOPT ☑ EESAVE ☑ BOOTSZ1 ☑ BOOTSZ0 ☐ BOOTRST

☑ BODLEVEL ☑ BODEN ☐ SUT1 ☐ SUT0 ☐ CKSEL3 ☐ CKSEL2 ☐ CKSEL1 ☐ CKSEL0

You can get more information from ‘Help of Ponyprog2000’.

3. WinCUPL

WinCUPL(Universal Compiler for Programmable Logic) is a logic compiler that can be used to create very sophisticated logic designs for SPLD and CPLD. This tool makes possible for engineers to design their logic and creates JEDEC (Joint Electronic Device Engineering Council Standard) file. Therefore, you can do mapping in Device using ROM writer.

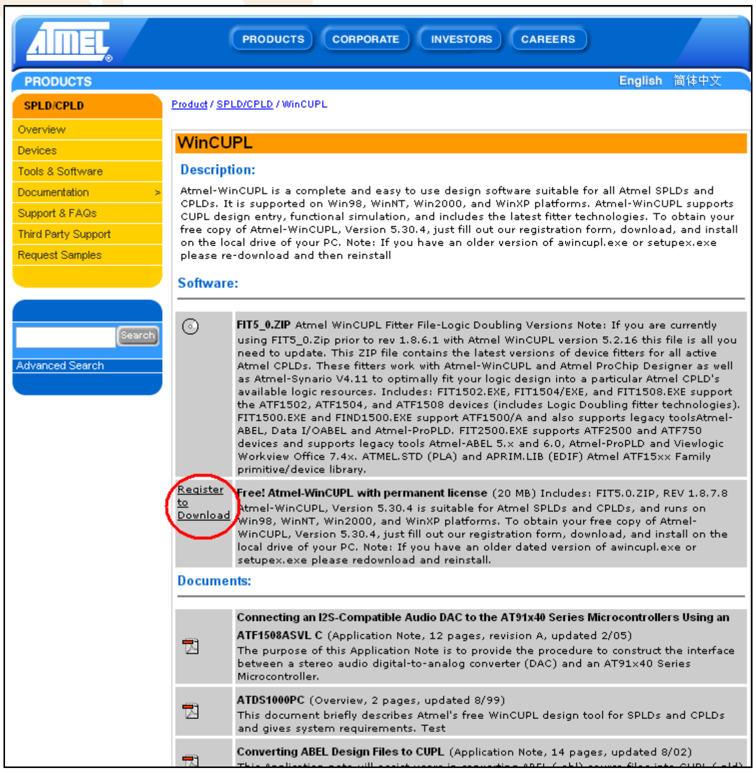
The WinCUPL package includes the following tools:

WinCUPL A powerful front end and user interface for all of the WinCUPL tools including the compiler. For more details on the features of WinCUPL, see WinCUPL Features.
CUPL Compiler Logic descriptions written in the CUPL language are compiled, and can be assigned to specific logic devices (PLDs). Upon compilation, the CUPL compiler searches its libraries and creates a file which can be downloaded to a device programmer. From this point, the PLD can be programmed.
Simulator Designs can be simulated with CSIM before they are put into production. CSIM compares the expected values to actual values calculated during CUPL operation. Both the simulation inputs and the results of the simulation can be graphically viewed and modified with WinSim.
WinSim Simulation input and results are set and displayed by WinSim in waveform.

3.1. How to Install

1) Go to http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2759
2) Click “register to download” in this web page

Figure 3-1. Atmel’s Web site for download WinCUPL.



WinCUPL

Description:

Atmel-WinCUPL is a complete and easy to use design software suitable for all Atmel SPLDs and CPLDs. It is supported on Win98, WinNT, Win2000, and WinXP platforms. Atmel-WinCUPL supports CUPL design entry, functional simulation, and includes the latest fitter technologies. To obtain your free copy of Atmel-WinCUPL, Version 5.30.4, just fill out our registration form, download, and install on the local drive of your PC. Note: If you have an older version of awincupl.exe or setupex.exe please re-download and then reinstall

Software:

FIT5_0.ZIP Atmel WinCUPL Fitter File-Logic Doubling Versions Note: If you are currently using FIT5_0.Zip prior to rev 1.8.6.1 with Atmel WinCUPL version 5.2.16 this file is all you need to update. This ZIP file contains the latest versions of device fitters for all active Atmel CPLDs. These fitters work with Atmel-WinCUPL and Atmel ProChip Designer as well as Atmel-Synario V4.11 to optimally fit your logic design into a particular Atmel CPLD's available logic resources. Includes: FIT1502.EXE, FIT1504/EXE, and FIT1508.EXE support the ATF1502, ATF1504, and ATF1508 devices (includes Logic Doubling fitter technologies). FIT1500.EXE and FIND1500.EXE support ATF1500/A and also supports legacy toolsAtmel-ABEL, Data I/OABEL and Atmel-ProPLD. FIT2500.EXE supports ATF2500 and ATF750 devices and supports legacy tools Atmel-ABEL 5.x and 6.0, Atmel-ProPLD and Viewlogic Workview Office 7.4x. ATMEL.STD (PLA) and APRIM.LIB (EDIF) Atmel ATF15xx Family primitive/device library.

Register to Download

Free! Atmel-WinCUPL with permanent license (20 MB) Includes: FIT5.0.ZIP, REV 1.8.7.8 Atmel-WinCUPL, Version 5.30.4 is suitable for Atmel SPLDs and CPLDs, and runs on Win98, WinNT, Win2000, and WinXP platforms. To obtain your free copy of Atmel-WinCUPL, Version 5.30.4, just fill out our registration form, download, and install on the local drive of your PC. Note: If you have an older dated version of awincupl.exe or setupex.exe please redownload and reinstall.

Documents:

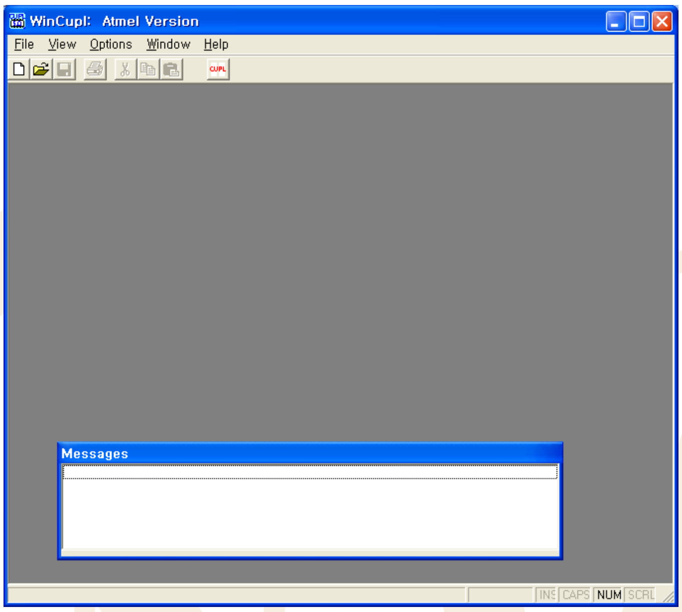
Connecting an I2S-Compatible Audio DAC to the AT91x40 Series Microcontrollers Using an ATF1508ASVL C (Application Note, 12 pages, revision A, updated 2/05)
The purpose of this Application Note is to provide the procedure to construct the interface between a stereo audio digital-to-analog converter (DAC) and an AT91x40 Series Microcontroller.

ATDS1000PC (Overview, 2 pages, updated 8/99)
This document briefly describes Atmel's free WinCUPL design tool for SPLDs and CPLDs and gives system requirements. Test

Converting ABEL Design Files to CUPL (Application Note, 14 pages, updated 8/02)

3) Please register for downloading and get Serial Number. After this preceding work, you can process download
4) Execute the downloaded file, "awincupl.exe"
5) Install following InstallShield Wizard
6) After restarting, execute Start>Program>Atmel WinCupl>WinCupl

Figure 3-2. WinCUPL's Main Screen.



3.2. Designing with the CUPL Language

This chapter introduces CUPL operators for Design and shows you an example about Design processing.

3.2.1. Usage of Language Syntax

Basic logic and arithmetic operator and functions used in Boolean equation design are like below.

1) Logic operators

Following table shows the way of expression and precedence in order to use NOT, AND, OR, and XOR as logical operators

Operator	Examples	Description	Precedence
!	!A	NOT	1
&	A & B	AND	2
#	A # B	OR	3
$	A $ B	XOR	4

2) Arithmetic operators and functions

Below table shows the way of expression, examples and precedence about frequently used 6 operators.

Operator	Examples	Description	Precedence
**	2**3	Exponentiation	1
*	2*l	Multiplication	2
/	4/2	Division	2
%	9%8	Modulus	2
+	2+4	Addition	3
-	4-l	Subtraction	3

One arithmetic function is available to use in arithmetic expressions being used in $repeat and $macro commands. The following table shows the arithmetic function and its bases.

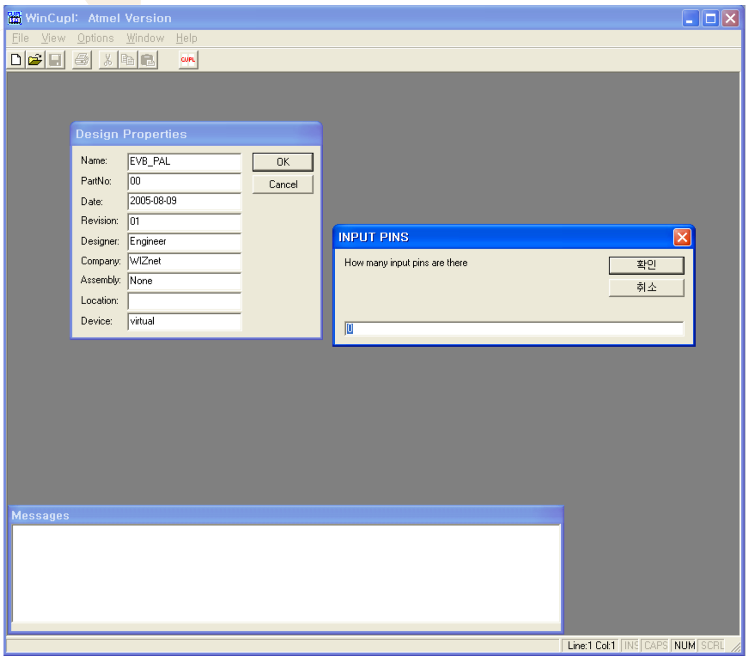
Function	Base
LOG2	Binary
LOG8	Octal
LOG16	Hexadecimal
LOG	Decimal

3.2.2. Start Designing

Now, we introduce how to design PLD through a simple example. You can execute PLD including your waiting function by following processes

1) After executing of WinCupl, Click File>New>Project
2) If you write above-mentioned contents in Design Properties and click OK button, INPUT PIN window appears

Figure 3-3. INPUT PIN screen



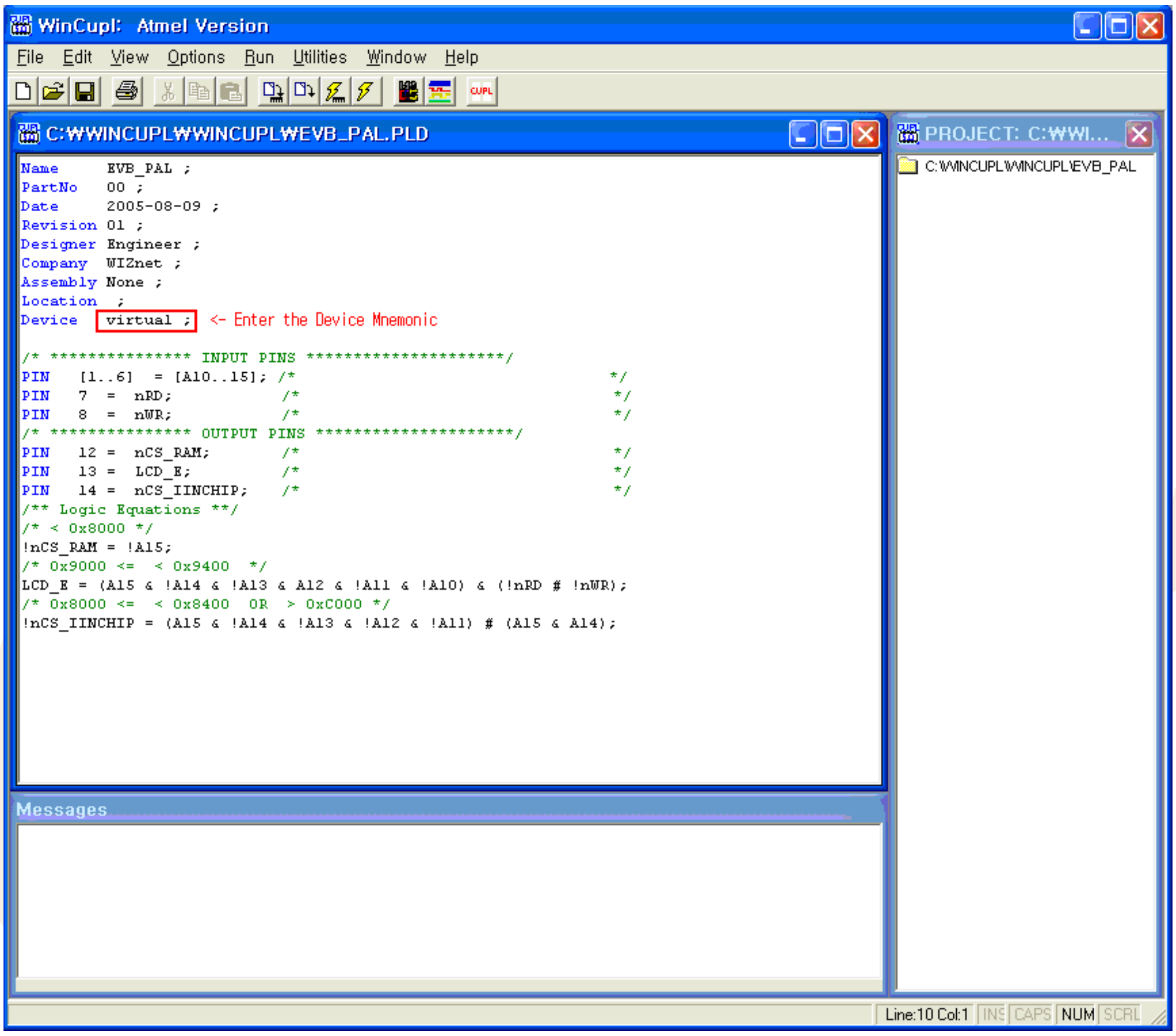
Name: EVB_PAL
PartNo: 00
Date: 2005-08-09
Revision: 01
Designer: Engineer
Company: WIZnet
Assembly: None
Location:
Device: virtual

How many input pins are there

0

3) Put in INPUT PIN Number and click OK button. And then, put in information of OUTPUT PIN, PINNODESS etc in the same way. (* Pin assignment needs to be done if the designer already knows the device he want to use)
4) In Design Window that has the created Form, do coding within user's needs

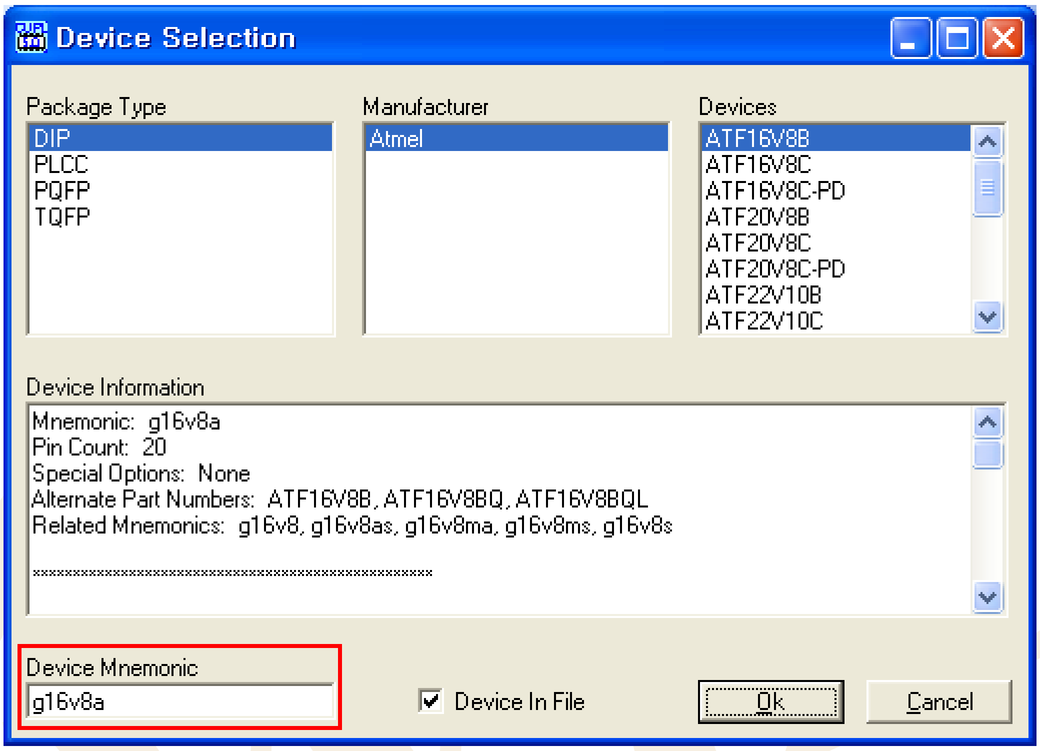
Figure 3-4. Sample source.


WinCupl: Atmel Version
File Edit View Options Run Utilities Window Help
C:₩WINCUPL₩WINCUPL₩EVB_PAL.PLD
PROJECT: C:₩WI...
C:\WINCUPL\WINCUPL\EVB_PAL
Messages
Line:10 Col:1 INS CAPS NUM SCRL

```
Name      EVB_PAL ;
PartNo    00 ;
Date      2005-08-09 ;
Revision  01 ;
Designer  Engineer ;
Company   WIZnet ;
Assembly  None ;
Location   ;
Device    virtual ;  <- Enter the Device Mnemonic

/* *************** INPUT PINS *********************/
PIN   [1..6]  = [A10..15]; /*                          */
PIN    7  =  nRD;           /*                          */
PIN    8  =  nWR;           /*                          */
/* *************** OUTPUT PINS *********************/
PIN   12 =  nCS_RAM;        /*                          */
PIN   13 =  LCD_E;          /*                          */
PIN   14 =  nCS_IINCHIP;    /*                          */
/** Logic Equations **/
/* < 0x8000 */
!nCS_RAM = !A15;
/* 0x9000 <=  < 0x9400  */
LCD_E = (A15 & !A14 & !A13 & A12 & !A11 & !A10) & (!nRD # !nWR);
/* 0x8000 <=  < 0x8400  OR  > 0xC000 */
!nCS_IINCHIP = (A15 & !A14 & !A13 & !A12 & !A11) # (A15 & A14);
```

5) Select Device that you will use in Options > Devices menu.screen. After Device Selection, you should put the shown “Device Mnemonic” information in your Coding page.
Refer to the left and lower side of screen as the below figure.

Figure 3-5. Device Selection.



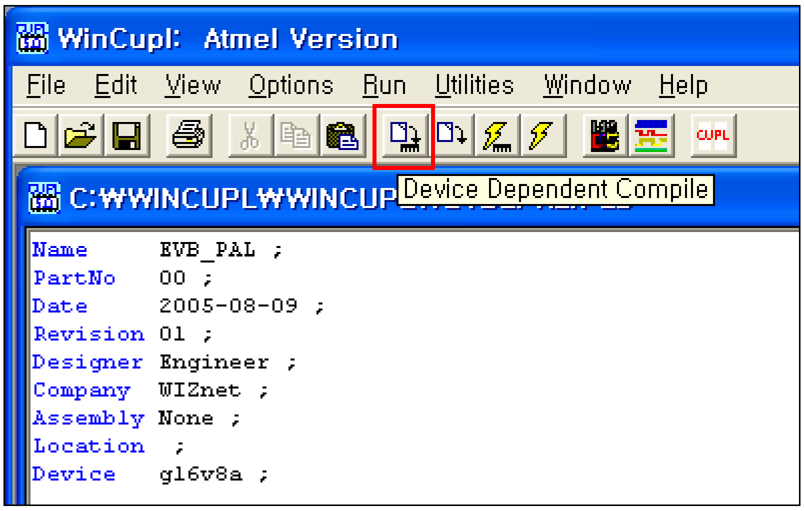
Mnemonic: g16v8a
Pin Count: 20
Special Options: None
Alternate Part Numbers: ATF16V8B, ATF16V8BQ, ATF16V8BQL
Related Mnemonics: g16v8, g16v8as, g16v8ma, g16v8ms, g16v8s

Device Mnemonic: g16v8a

3.2.3. Compiling

1) After completion of Coding process, please select your wanting Compile item through Run Menu or the indicated icon.

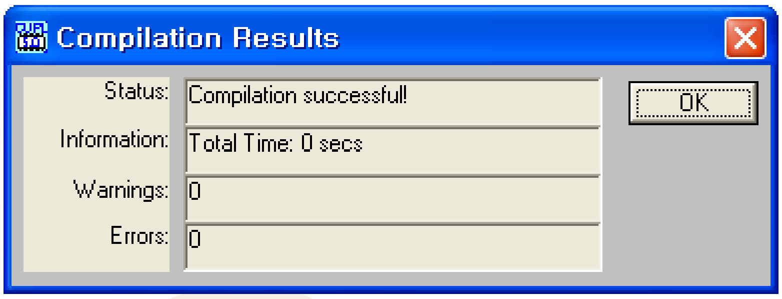
Figure 3-6. Device Dependent Compile.



```
Name      EVB_PAL ;
PartNo    00 ;
Date      2005-08-09 ;
Revision  01 ;
Designer  Engineer ;
Company   WIZnet ;
Assembly  None ;
Location   ;
Device    g16v8a ;
```

2) If Compile process is completed, Compilation Results screen appears in your monitor as shown in following figure.

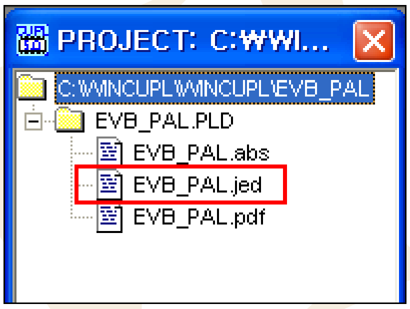
Figure 3-7. Compilation Results.



Status: Compilation successful!
Information: Total Time: 0 secs
Warnings: 0
Errors: 0

3) You can confirm the created JEDEC file through Compile process.
However, if you wrote the Device information as Virtual condition, you would fail in JEDEC file creating. Therefore, this process requires delicate care

Figure 3-8. Created JEDEC file Screen



C:\WINCUPL\WINCUPL\EVB_PAL
EVB_PAL.PLD
EVB_PAL.abs
EVB_PAL.jed
EVB_PAL.pdf

4) You can perform Writing process of created JEDEC file in Device using Rom writer.

You can get more information from http://www.atmel.com or WinCUPL Users Manual.