

# BEGV641A

## USER MANUAL

LCD Embedded System,  
Atmel ATmega644p MCU,  
Graphic 240x128 STN LCD,  
LED/white backlight,  
RS232/RS422/RS485, I<sup>2</sup>C /SPI  
64KB in-system programmable Flash  
No Operating System required



# CONTENTS

<b>Precaution</b>		P3
<b>Packing Contents</b>		P4
<b>Chapter 1 Introduction</b>		P5
1-1 Features		P6
1-2 Board Layout		P6
1-3 Block Diagram		P7
1-4 Mechanical Dimension		P7
1-5 Board Specifications		P8
1-6 Ordering Information		P8
<b>Chapter 2 Installation</b>		P9
2-1 Jumpers		P10
	2-1-1 Contrast adjust	P10
	2-1-2 Frame ground	P10
	2-1-3 RS-422/485 VDD/ground	P10
	2-1-4 Screw hole ground	P11
2-2 Connectors		P12
	2-2-1 Connector & Pin Definition	P12
	2-2-2 Pin vs. Function Diagram	P15
<b>Chapter 3 MCU Port Mapping</b>		P18
3-1 MCU Pin Configuration		P19
3-2 MCU Port Mapping		P20
	3-2-1 LCD Controller	P20
	3-2-2 Touch Panel	P20
	3-2-3 RS-232/RS-422/RS-485	P20
	3-2-4 Enable Backlight	P20
	3-2-5 EEPROM/I <sup>2</sup> C	P20
	3-2-6 2-wire serial port	P20
	3-2-7 SPI	P21
	3-2-8 General I/O	P21
<b>Chapter 4 Software Development Tool &amp; Utility</b>		P22
4-1 ATMEL ATmega644p Tool		P23
	4-1-1 Download software from AVR studio website	P23
	4-1-2 Additional tool for C language	P31
4-2 Execute AVR studio 4.16 on designer PC		P35
4-3 In-system programmer AVR ISP mk II		P43
4-4 Bolymin Free Software Utilities		P61
	4-4-1 Website links	P61
	4-4-2 Introduction of BOLYMIN software utilities	P62
	4-4-3 Software utilities function description	P70
<b>Appendix A: LCD Controller Specification</b>		
<b>Appendix B: EEPROM Specification</b>		
<b>Appendix C: ATMEL ATmega644p MCU Specification</b>		

## Precaution

### FCC



WARNING



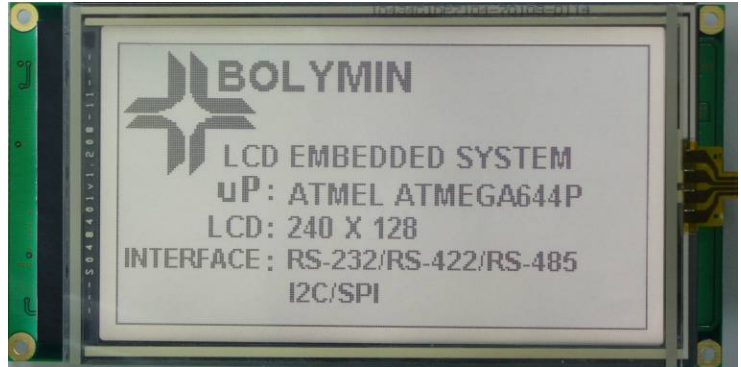
CAUTION

This device is designed to meet the requirement in part 15 of the FCC rules. Operation is subject to conditions ruled under FCC part 15.

Please check packing content upon receiving BEGV641A parcel, make sure that all materials and options are packed inside parcel according to your order.

#### **Packing Contents Check-List**

- ☐ BEGV641A LCD Embedded module
- ☐ Touch panel
- ☐ Software Utility Disc
- ☐ ISP Cable (option)
- ☐ ATMEL software development tool



## Chapter 1 Introduction

### Abstract

This chapter is to offer you basic information regarding BEGV641A, to help you incorporate BEGV641A into your system.

Contents include:

- 1-1 Features
- 1-2 Board Layout
- 1-3 Block Diagram
- 1-4 Mechanical Dimension
- 1-5 Board Specifications
- 1-6 Ordering information

## 1-1 Features

This BEGV641A is designed based on ATmega644p microprocessor, which requires no operating system to run on. Together with a 240x128 STN LCD and LED backlight built-in, this all-in-one LCD embedded system BEGV641A help designer enhance a compact design with cost saving, space saving, and design phase saving.

Armed with RS232, RS422/485, I<sup>2</sup>C and SPI interface port, this BEGV641A communicates many devices and peripherals. The BEGV641A is therefore suitable to sit as a industrial control panel for factory automation equipment, electronics instrument, HMI (human-machine interface), office automation equipment, medical equipment, parking system, ticketing system.. and so on.

There are five LCD colors among choices: STN/gray, STN/yellow-green, STN/blue, FSTN/gray, and FSTN/black. All comes with LED/white backlight.

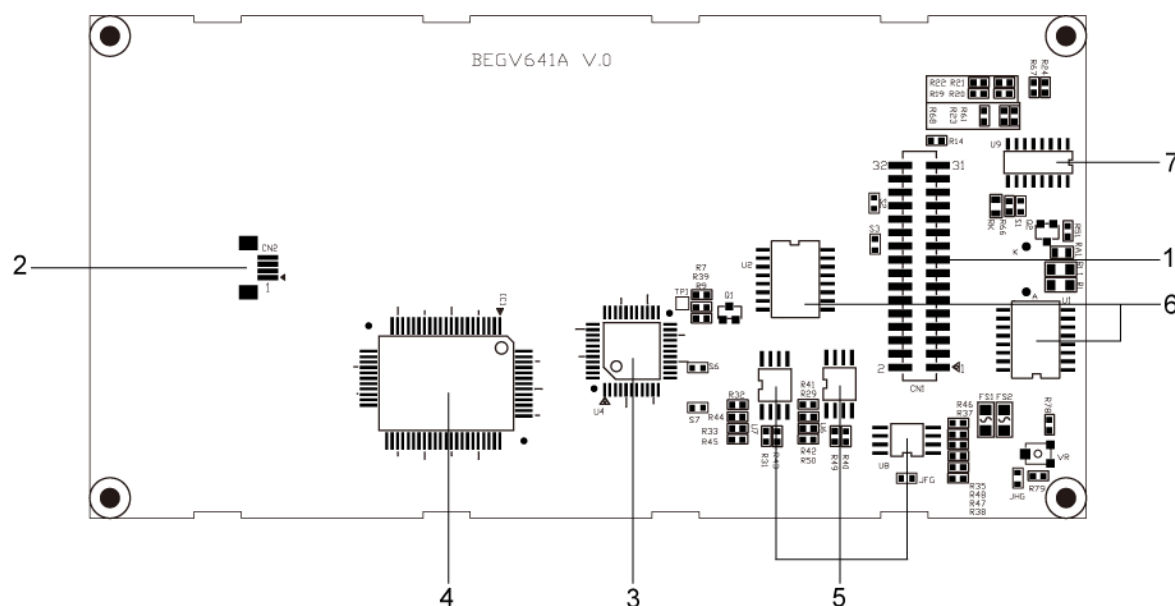
64KB in-system self-programmable Flash offers sufficient ROM size for designer to develop software, further to control LCD, touch panel, and interfaces.

Designer may simply design this BEGV641A into your application as you are designing a ATMEL CPU board, without worrying LCD module and other interfaces, since they are all on one board.

## 1-2 Board Layout

This layout shows the location of each important IC, connector and jumper. Please refer to chapter 2 for further information on jumper and connector.

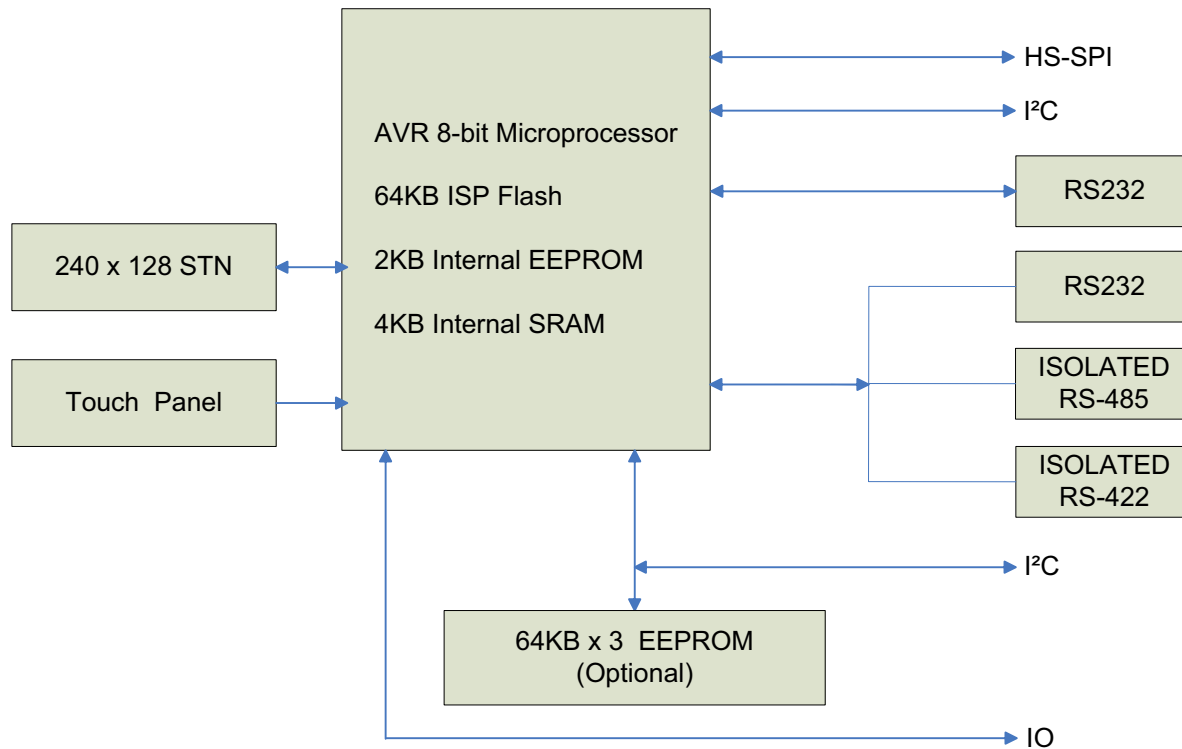
(Drawing 1.2)



- |                           |                   |
|---------------------------|-------------------|
| <b>1</b> :CN1             | <b>7</b> : RS-232 |
| <b>2</b> :CN2             |                   |
| <b>3</b> : Microprocessor |                   |
| <b>4</b> :LCD controller  |                   |
| <b>5</b> :EEPROM          |                   |
| <b>6</b> :RS-485/422      |                   |

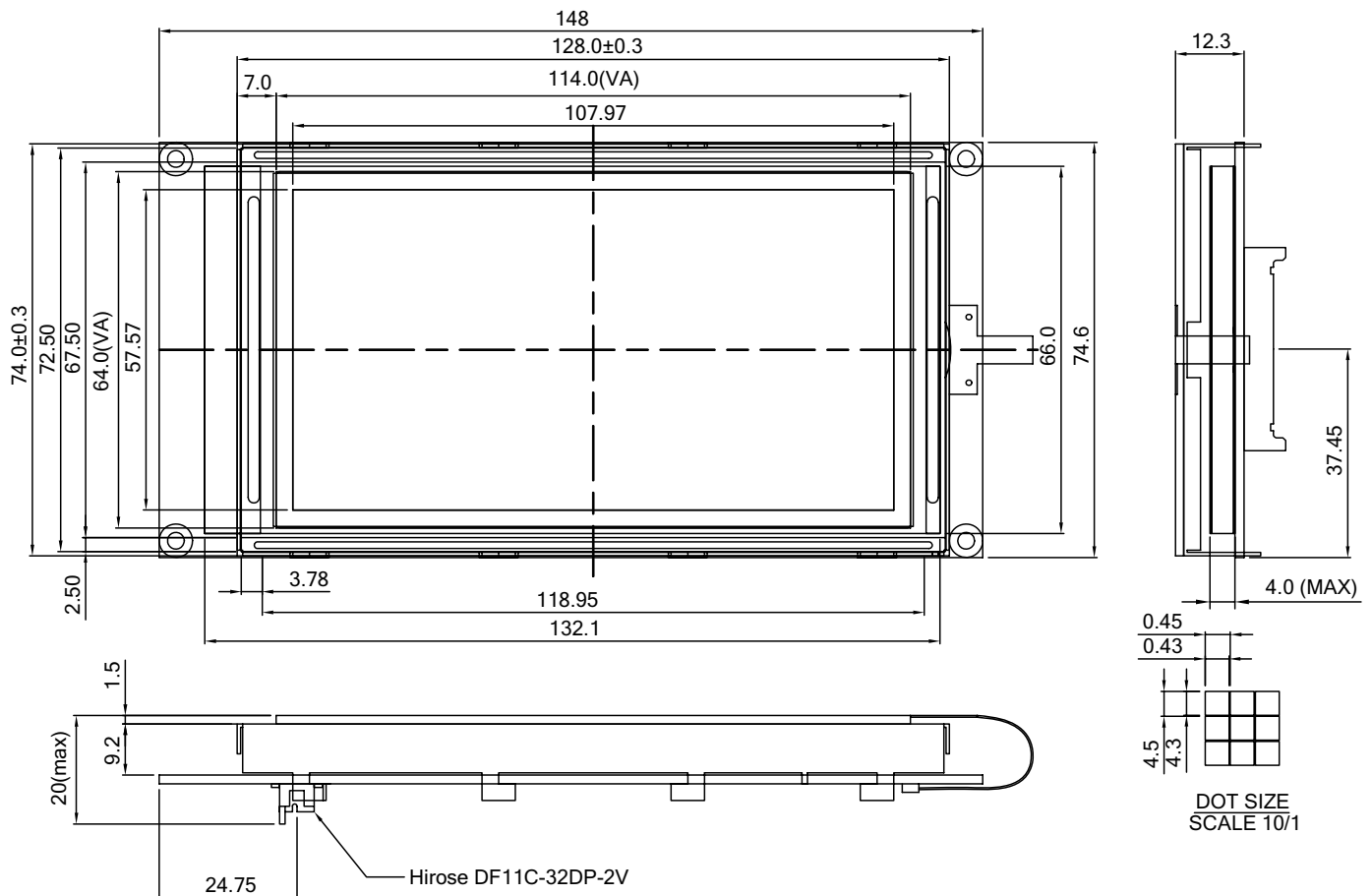
### 1-3 Block Diagram

(Drawing 1.3)



### 1-4 Mechanical Dimension

(Drawing 1.4)



## 1-5 Board Specifications

(Table 1.5)

MCU	High-performance, Low-power AVR <sup>®</sup> 8-bit microprocessor ATMEL ATmega644P
Memory	64K Bytes In-System Self-Programmable Flash 2K Bytes Internal EEPROM 4K Bytes Internal SRAM 3 x 64K Bytes External EEPROM(optional)
Display	Support 8-bit single-scan resolution 240 x 128 monochrome STN LCD, with edge LED white backlight only
Touch Panel(optional)	Support four-wired resistive touch panel
Serial Ports	Support 1 x RS232 port, and 1 x RS232/RS422(isolated)/RS485(isolated) co-shared port Support 1 x full-duplex, three-wired synchronous data transfer SPI port Support 1 x two-wired serial interface to 250 KHz data transfer speed

## 1-6 Ordering Information

(Table 1.6)

Part No.	Description	RS232-A	RS232-B	RS422	RS485
BEGV641A	Dual RS232	☆	☆		
BEGV641A1	One RS232	☆			
BEGV641A2	One RS232/One RS422	☆		☆	
BEGV641A3	One RS232/One RS485	☆			☆

Display: FSTN/Gray LCD,LED/White Backlight(Default)



## Chapter 2 Installation

### Abstract

This chapter is to offer designer fundamental information of BEGV641A jumpers and connectors, in order to help designer configure correct setting and connection between BEGV641A and system application.

Contents include:

2-1 Jumpers

2-2 Connectors

## 2-1 Jumpers

This section is to indicate location and function of each jumper on BEGV641A, which user can arrange according to the needs of different application desired. Be careful when setting jumper, user maybe need tool such as needle-nose pliers to help setting. Please note, jumpers not described here are intended to keep as factory default setting. Please consult Bolymin before trying to change default setting.

The table listed below describes location and function of each available jumper.

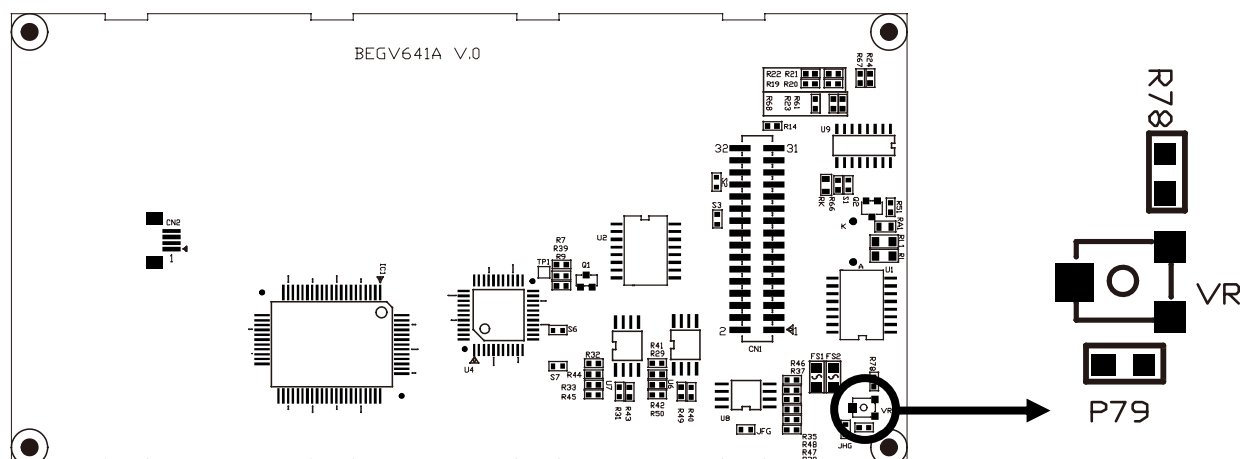
**Jumpers: (Table 2.1)**

Label	Function
VR	LCD contrast control
JFG	Frame ground
JG	RS-422/RS-485 VDD/ground
JHG	Screw hole ground

Detail location and function of each jumper is illustrated below.

### 2-1-1 Contrast Adjust

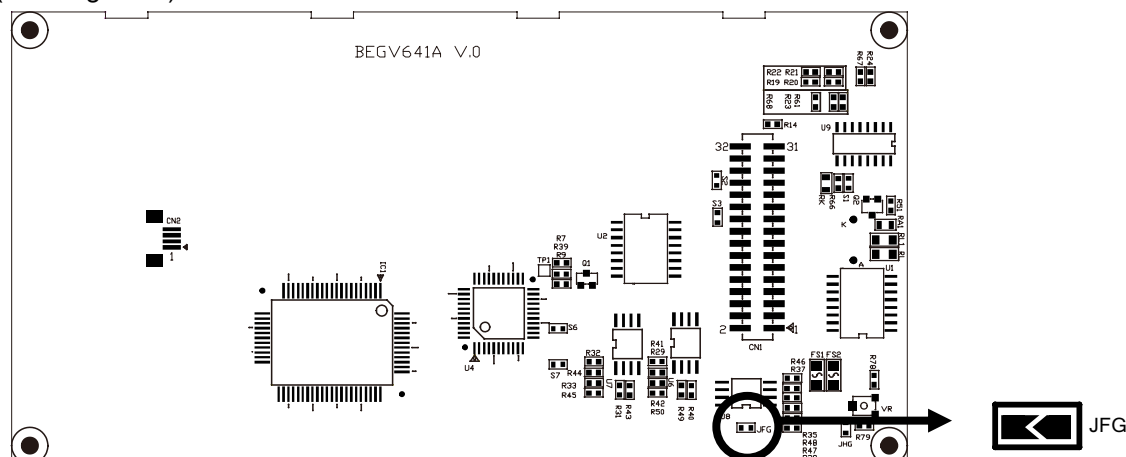
(Drawing 2.1.1)



VR	Contrast adjust	
10K ohm	Inside	default
N/A	Outside	

### 2-1-2 Frame Ground

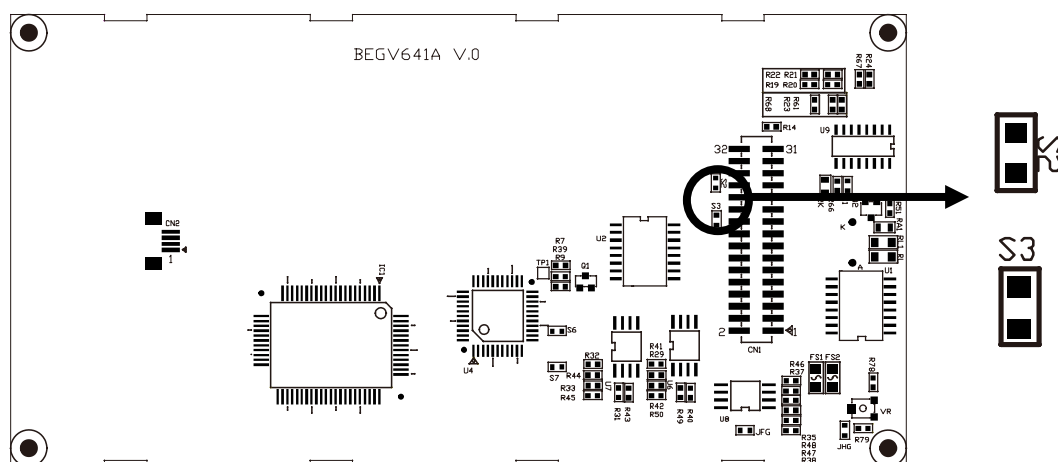
(Drawing 2.1.2)



JFG	Frame ground	
short	Connect metal frame with GND	
open	Not connected metal frame with ground	default

### 2-1-3 RS-422/RS-485 VDD/Ground

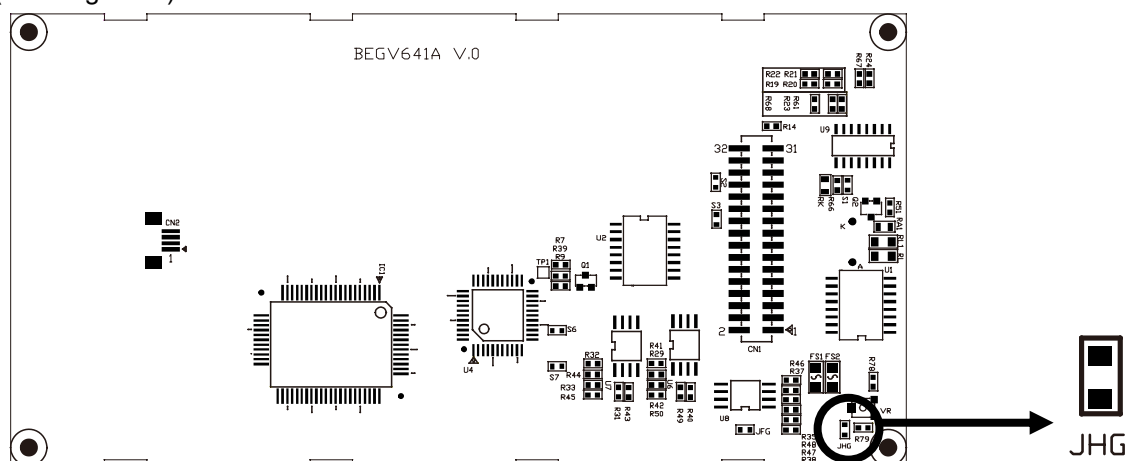
(Drawing 2.1.3)



S1	S2	RS-422/RS-485 VDD/Ground	
short	short	internal VDD/ground for isolated RS-422/485	
open	open	External VDD/ground for isolated RS-422/485	default

### 2-1-4 Screw hole ground

(Drawing 2.1.4)



JHG	Frame ground	
short	Connect screw hole with GND	
open	Not connected screw hole with ground	default

## 2-2 Connectors

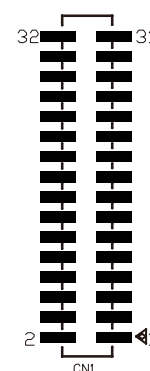
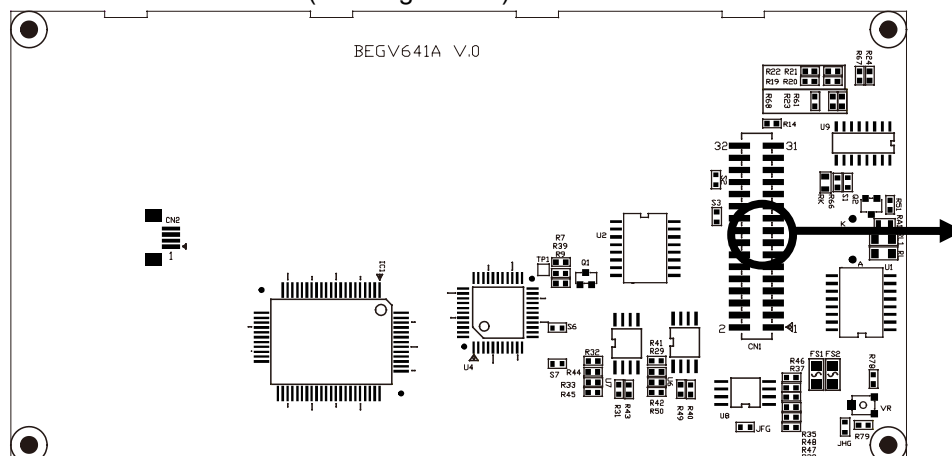
Connectors are the key link between BEGV641A and external devices. Detail locations and functions of available connectors are tabled and illustrated below.

Connectors: (Table 2.2)

Label	Pin No.	Function
CN1	32	Connector

### 2-2-1 Connectors & Pin Definition

#### ● Connector CN1 (Drawing 2.2cn1)



#### Pin Types

I	=Input
O	=Output
Bi	=Input / Output (Bi-Directional)
U	=User defined
P	=Power

#### ● Pin Definition: BEGV641A-Dual RS232 (Table 2.2.1a)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	NC
17	SCK	18	NC
19	RX0	20	NC
21	TX0	22	NC
23	RX1	24	NC
25	TX1	26	NC
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

● **Pin Definition: BEGV641A1-One RS232** (Table 2.2.1b)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	NC
17	SCK	18	NC
19	RX0	20	NC
21	TX0	22	NC
23	NC	24	NC
25	NC	26	NC
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

● **Pin Definition: BEGV641A2-One RS232/One RS422** (Table 2.2.1c)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	422RP
13	MOSI	14	422RN
15	MISO	16	422TP
17	SCK	18	422TN
19	RX0	20	VDD2
21	TX0	22	VDD2
23	NC	24	GND2
25	NC	26	GND2
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

● **Pin Definition: BEGV641A3-One RS232/One RS485** (Table 2.2.1d)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	485P
17	SCK	18	485N
19	RX0	20	VDD2
21	TX0	22	VDD2
23	NC	24	GND2
25	NC	26	GND2
27	SDA	28	EN485
29	SCL	30	NC
31	IOA	32	NC

### 2-2-1-1 Power & Ground (Table 2.2.1.1)

Signal	Type	Pin No.	Description
GND	P	1	Logic power supply (ground)
VDD	P	3	Logic power supply (+5V)
VDD2	P	20	External Power for isolated RS-422/485(+5V)
VDD2	P	22	External Power for isolated RS-422/485(+5V)
GND2	P	24	External Ground for isolated RS-422/485(ground2)
GND2	P	26	External Ground for isolated RS-422/485(ground2)

### 2-2-1-2 Serial I/O (Table 2.2.1.2)

Signal	Type	Pin No.	Description
/SS	U	11	SPI Chip select
MOSI	U	13	MOSI is the master data output line, when SPI on module is configured as a master. When SPI is configured as a slave. This pin reverse the role.
MISO	U	15	MISO is the master data input line, when SPI is configured master. When SPI is configured as a slave. This pin reverse the role.
SCK	U	17	SPI clock
RX0	I	19	Receiver of first RS232 with driver
TX0	O	21	Transmitter of first RS232 with driver
RX1	I	23	Receiver of second RS232 with driver
TX1	O	25	Transmitter of second RS232 with driver
SDA	U	27	Data of 2-wire serial interface, it can be programmed as IO.
SCL	U	29	Clock of 2-wire serial interface, it can be programmed as IO.
EEPDA	Bi	2	Data of 2-wire serial interface for additional EEPROM update.
EEPSCL	Bi	4	Clock of 2-wire serial interface for additional EEPROM update
EEPWP	I	6	Write protect of additional EEPROM
422RP	I	12	no inverting receiver of RS422
422RN	I	14	inverting receiver of RS422
422TP/485P	Bi	16	When it configured as RS422 it act as no inverting transmitter, When is configured as RS 485 it acts as positive differential IO.
422TN/485N	Bi	18	When it configured as RS422 it acts as inverting transmitter, When is configured as RS 485 it acts as negative differential IO.
EN485	O	28	Enable RS-485

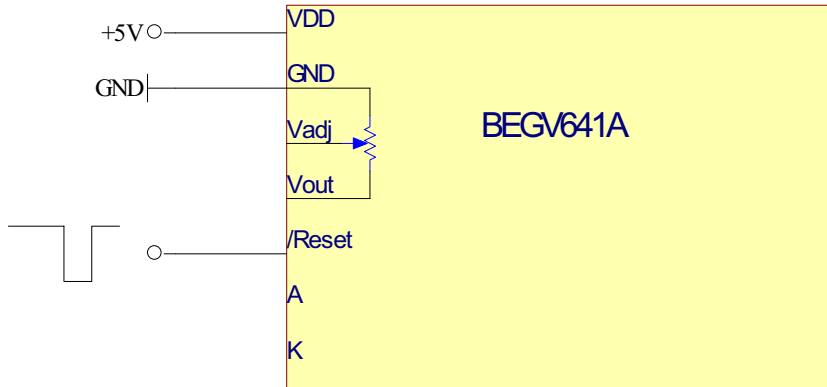
### 2-2-1-3 General I/O (Table 2.2.1.3)

Signal	Type	Pin No.	Description
/Reset	I	5	Auxiliary moment reset for external input
IOA	U	31	I/O port (ATmega644P portA.4)
IOB	U	8	I/O port (ATmega644P portA.7)

## 2-2-2 Pin vs. Function Diagram

### 2-2-2-1 Power/LCD/Backlight

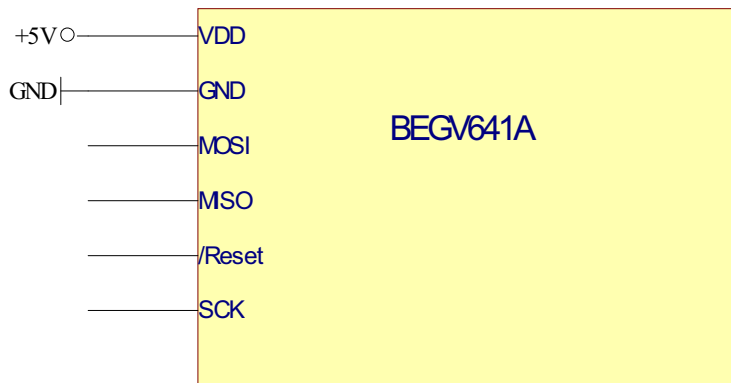
Diagram of system power supply, and contrast adjustment.



### 2-2-2-2 In-System Programming

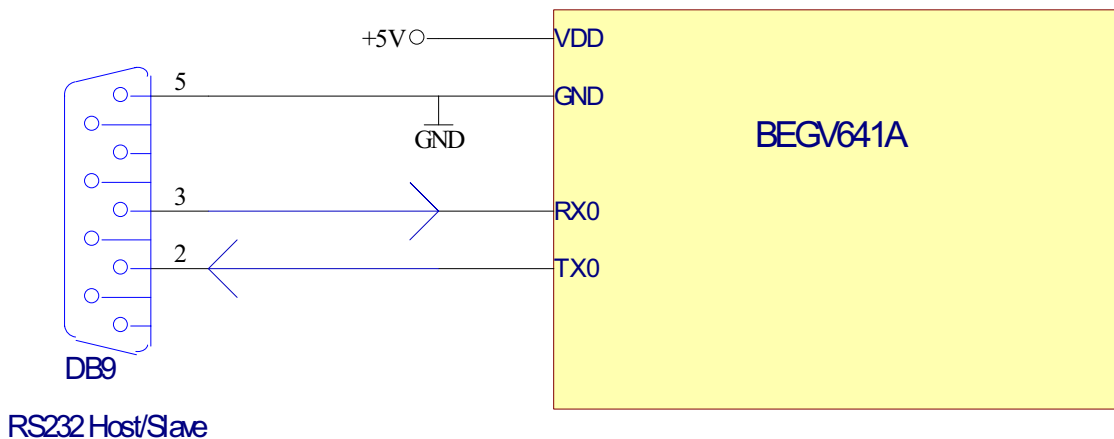
BEGV641A offers in-system software burn-in function, which allows designer to write software into ATmega644p via ATMEL AVR ISP MKII writer without taking ATmega644p IC off board. Please visit following website for more information on AVR ISP MKII writer.

[http://www.atmel.com/dyn/products/tools\\_card.asp?tool\\_id=3808](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3808)



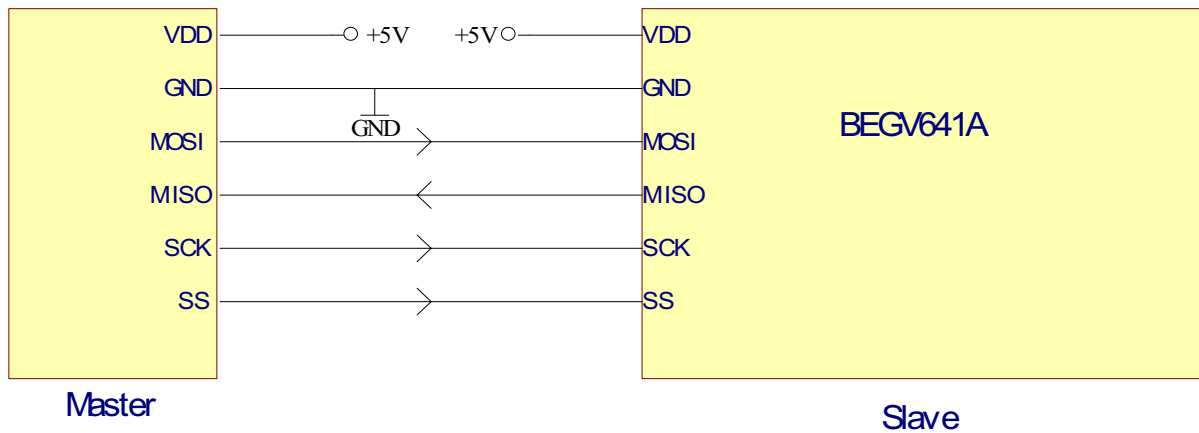
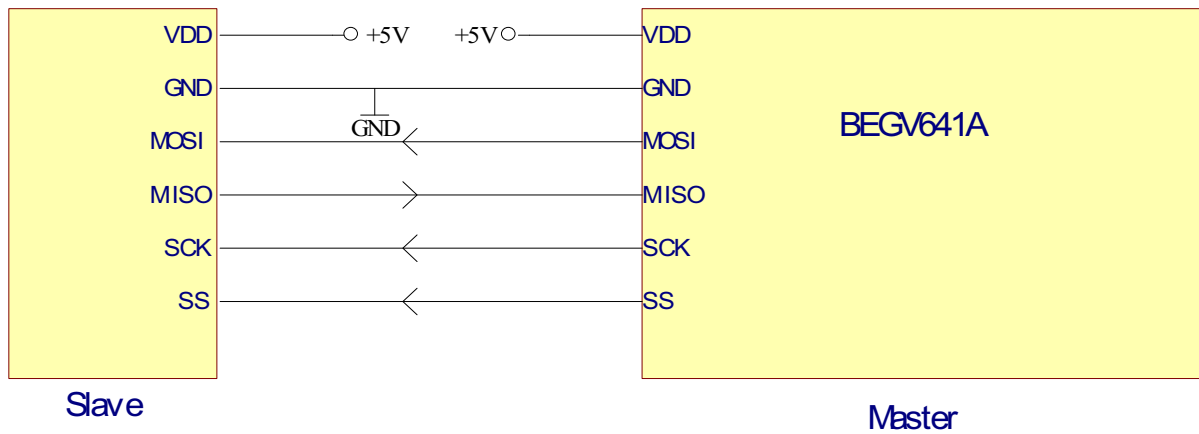
### 2-2-2-3 RS-232

BEGV641A offers RS-232 port to contact PC or other RS232 device directly without ICL232.



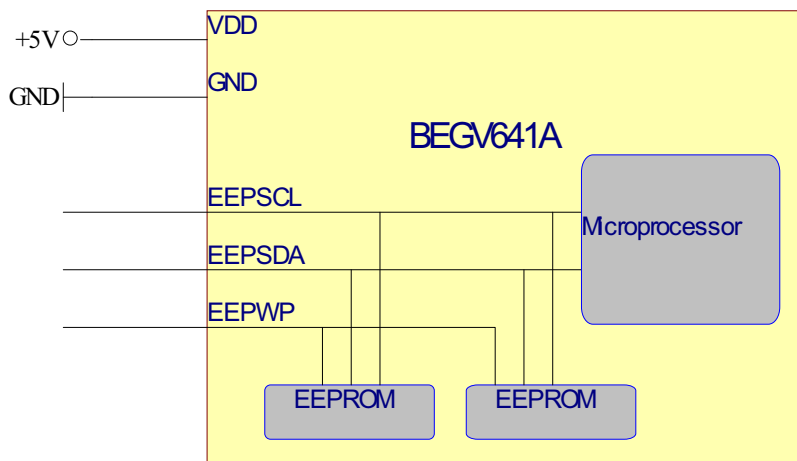
## 2-2-2-4 SPI

BEGV641A offers SPI port. Designer has to define this port as SPI by software, or to use Bolymin SPI driver (free utility).



## 2-2-2-5 I<sup>2</sup>C/EEPROM

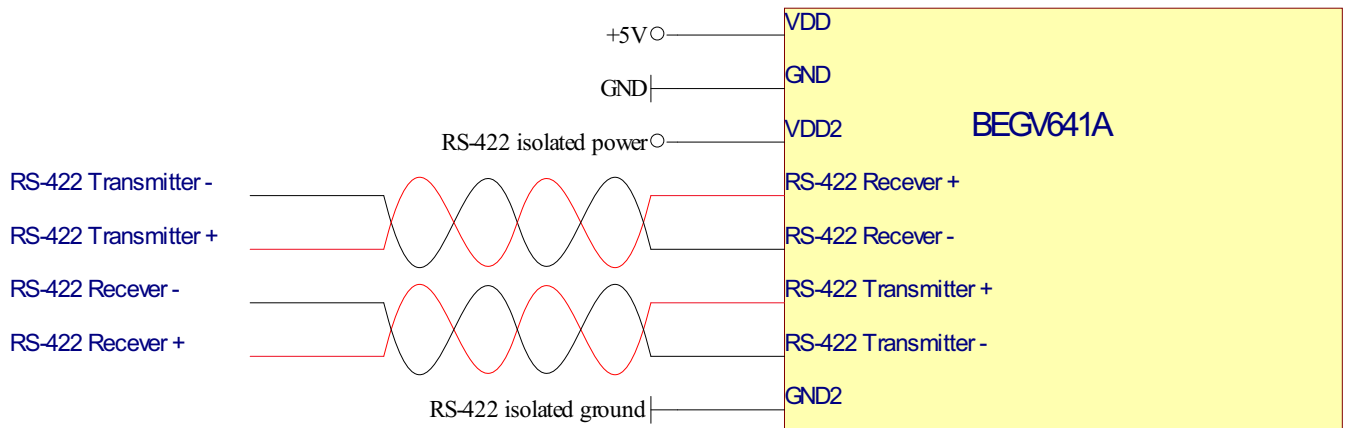
BEGV641A offers I<sup>2</sup>C port. Via this I<sup>2</sup>C port, designer may control 64Kbytes x 3 EEPROM in-system and external I<sup>2</sup>C devices.





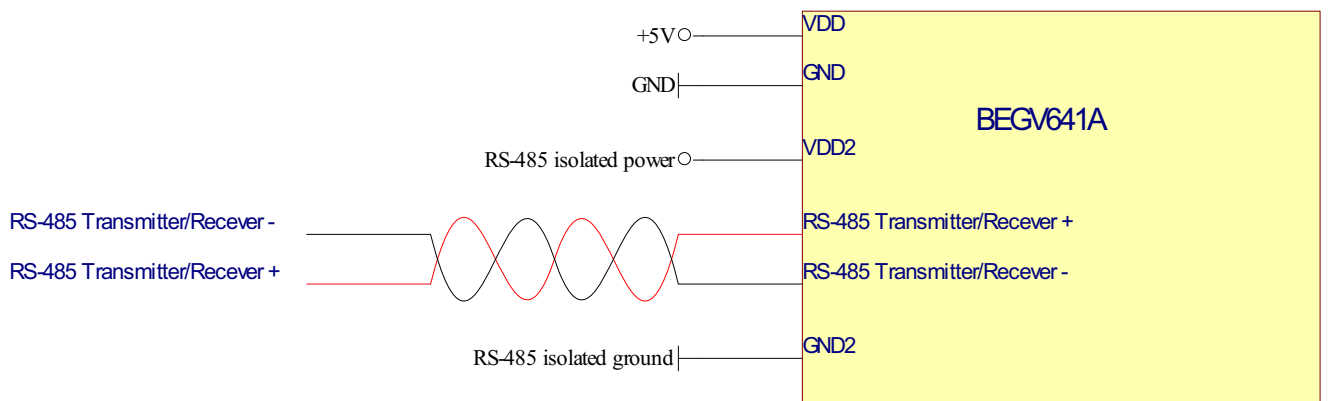
### 2-2-2-6 RS-422

BEGV641A offers 1 x RS-422(isolated) port.



### 2-2-2-7 RS-485

BEGV641A offers 1 x RS-485(isolated) port.



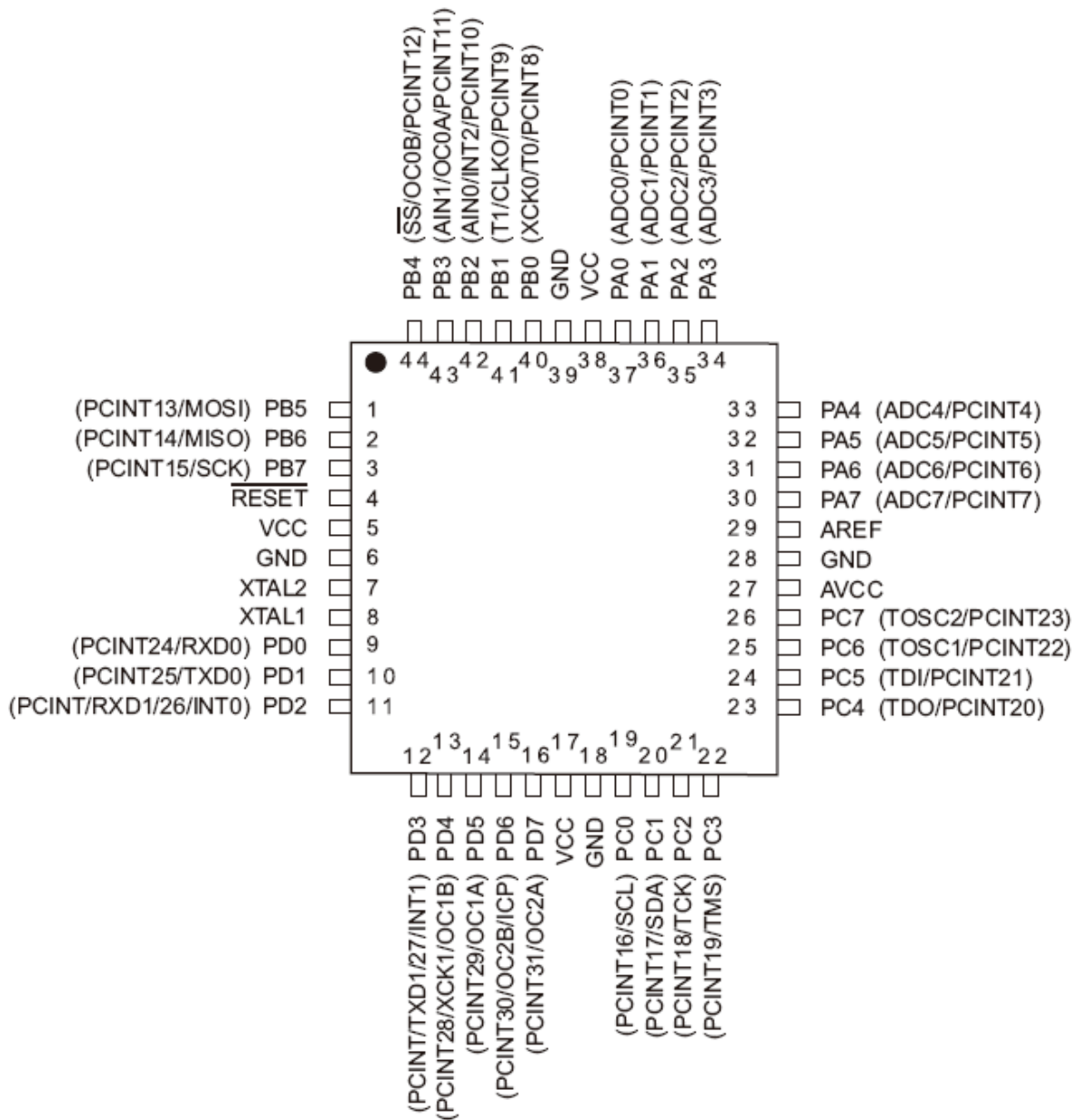
## Chapter 3 MCU port mapping

### Abstract

This chapter explains ATmega644p MCU pin configuration and port mapping toward key elements such as LCD, Touch Panel, RS-232, RS-422, RS-485, LED Backlight, EEPROM/I<sup>2</sup>C, 2-wire serial port, SPI, and General I/O.

### 3-1 MCU Pin Configuration

(Drawing 3.1, ATmega644p MCU)



## 3-2 MCU Port Mapping

### 3-2-1 LCD Controller

(Table 3.2a)

MCU ATmega644P	LCD Controller
PORTB.0	DB0(LSB)
PORTB.1	DB1
PORTB.2	DB2
PORTB.3	DB3
PORTD.4	DB4
PORTD.5	DB5
PORTD.6	DB6
PORTD.7	DB7(MSB)
PORTC.7	LCD /RESET
PORTC.5	LCD RD
PORTC.4	LCD WR
PORTC.6	LCD C/D

### 3-2-2 Touch Panel

(Table 3.2b)

MCU ATmega644P	Touch Panel
PORTA.0	X1
PORTA.1	Y1
PORTA.2	X2
PORTA.3	Y2

### 3-2-3 RS-232/RS-422/RS-485

(Table 3.2c)

MCU ATmega644P	RS-232/422/485
PORTD.0	RX0
PORTD.1	TX0
PORTD.2	RX1
PORTD.3	TX1

(Table 3.2d)

MCU ATmega644P	RS-485
PORTA.5	Enable RS-485

### 3-2-4 Enable Backlight

(Table 3.2e)

MCU ATmega644P	LED backlight
PORTA.6	Enable Backlight

### 3-2-5 EEPROM/I<sup>2</sup>C

(Table 3.2f)

MCU ATmega644P	EEPROM/ I <sup>2</sup> C
PORTC.2	EEPROM SDA
PORTC.3	EEPROM SCL

### 3-2-6 2-wire serial port

(Table 3.2g)

MCU ATmega644P	2-wire serial port
PORTC.1	SDA
PORTC.0	SCL

### 3-2-7 SPI

(Table 3.2h)

MCU ATmega644P	SPI
PORTB.4	/SS
PORTB.5	MOSI
PORTB.6	MISO
PORTB.7	SCK

### 3-2-8 General I/O

(Table 3.2i)

MCU ATmega644P	General I/O
PORTA.4	IOA
PORTA.7	IOB

## Chapter 4 Software Development Tool & Utility

### Abstract

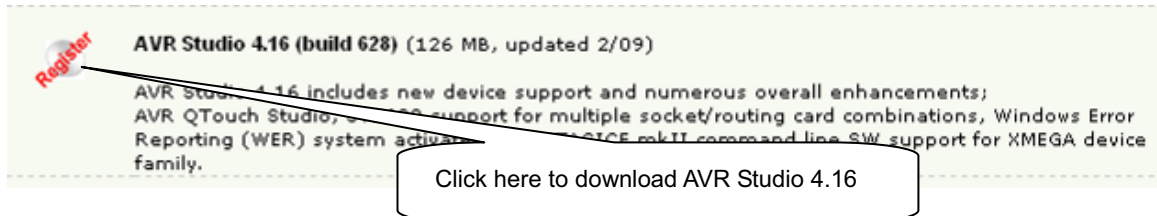
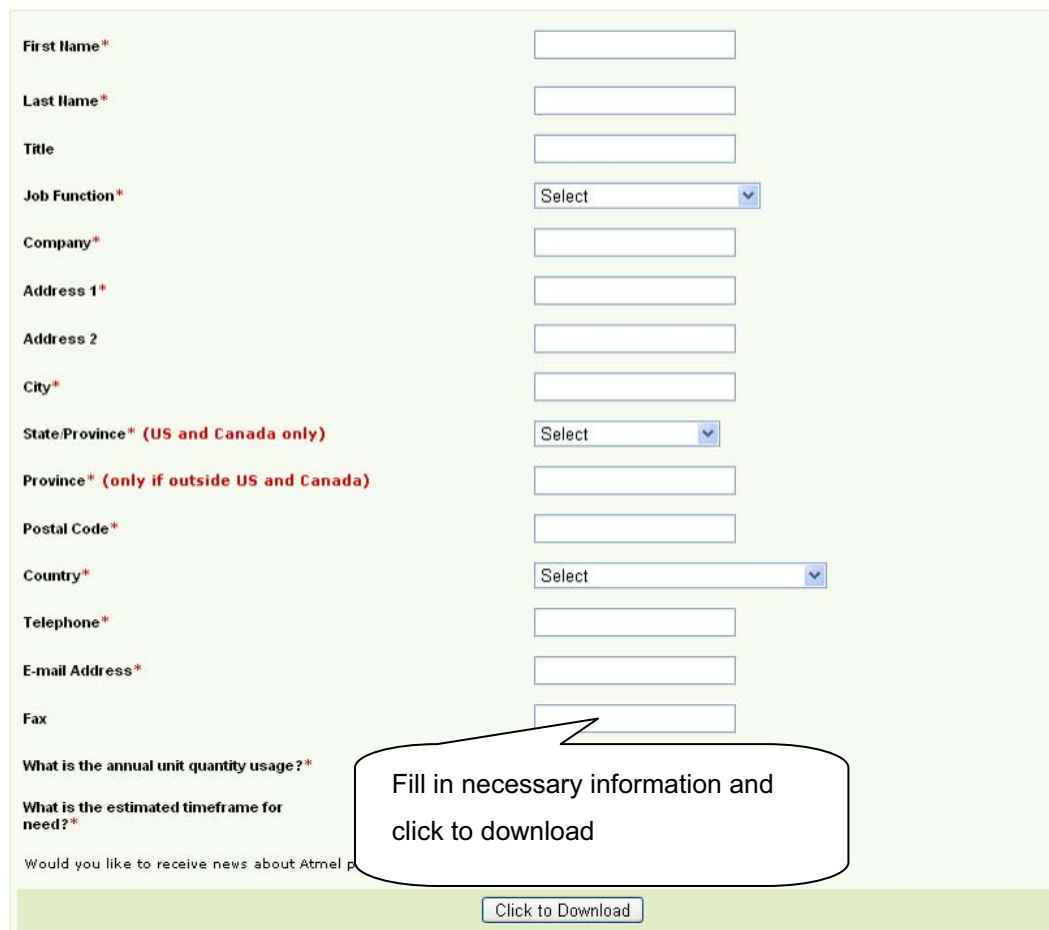
This chapter explains ATmega644p MCU software development tool AVR ISP mkII writer and Bolymin free software utilities.

## 4-1 ATMEL ATmega644p Software Development Tool

Designers may download software development tool from AVR Studio website  
[http://www.atmel.com/dyn/products/tools\\_card.asp?tool\\_id=2725](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725), or from BOLYMIN utility disk.

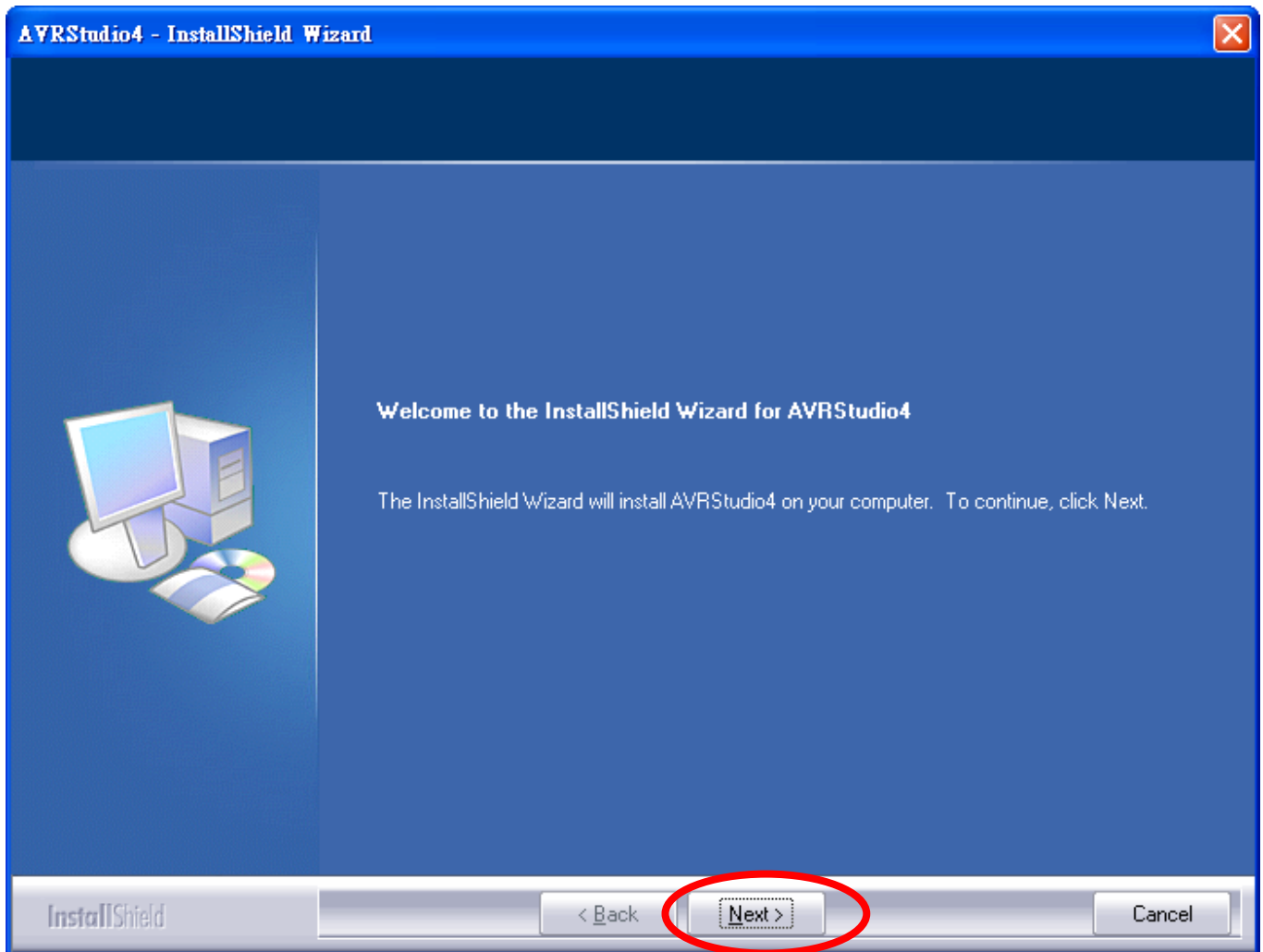
### 4-1-1 Download software from AVR Studio website

step1: Download design software

Step2: install AVR studio on designer PC







# AVRStudio4 - InstallShield Wizard

## License Agreement

Please read the following license agreement carefully.



Welcome to AVR Studio from Atmel Corporation.

AVR Studio is a Development Tool for the entire AVR family of microcontrollers, including tinyAVR, megaAVR, picoPower, and XMEGA AVR devices. The AVR Studio is free of charge and may be freely copied and distributed in its original form.

AVR Studio enables the user to fully control execution of programs on the AVR In-Circuit Emulator or on the included AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly and C/C++ programs assembled with the Atmel Corporation's included AVR Assembler or tools from 3rd party vendors.

AVR Studio runs under Microsoft Windows 98, Windows NT, Microsoft Windows 2000, Windows XP and Windows Vista.

AVR Studio is continuously developing. In order to get latest upgrades of AVR Studio, please visit our web site

[www.atmel.com](http://www.atmel.com)

and check out the AVR page.

☒ I accept the terms of the license agreement

☐ I do not accept the terms of the license agreement

Print

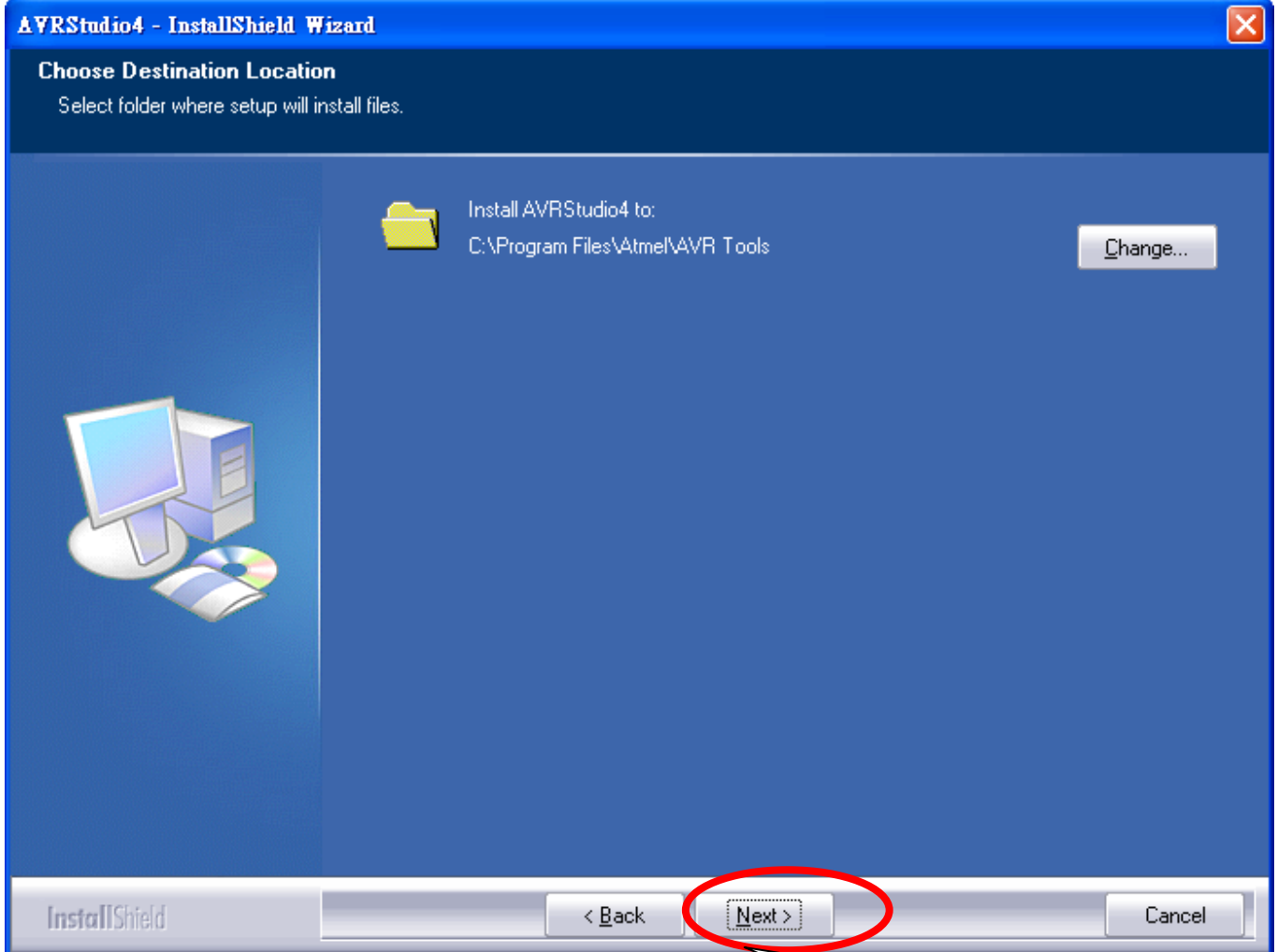
InstallShield

< Back

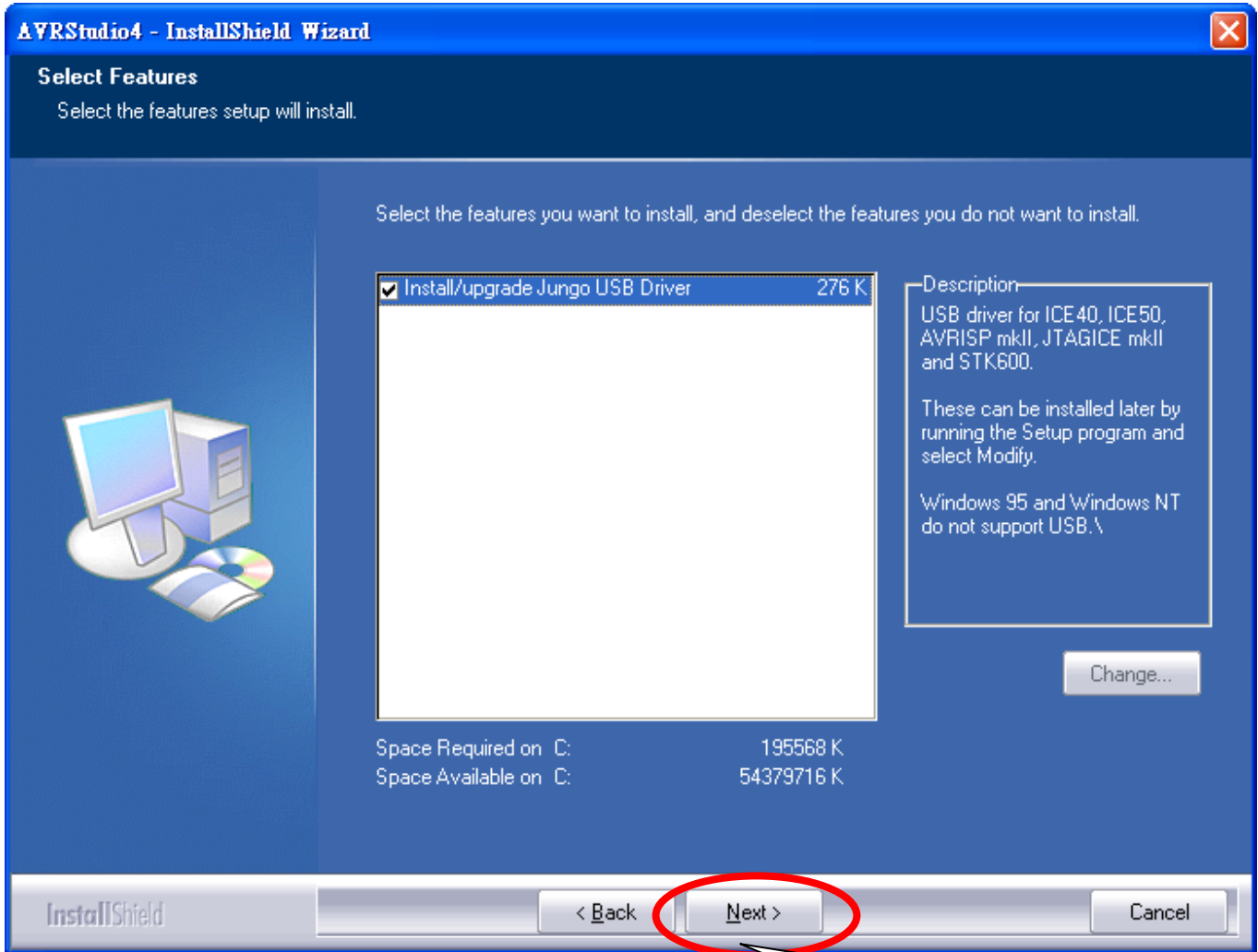
Next >

Cancel

Accept terms to continue



Select folder to install and continue



Select features to install and continue

**AVRStudio4 - InstallShield Wizard****Ready to Install the Program**

The wizard is ready to begin installation.



Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

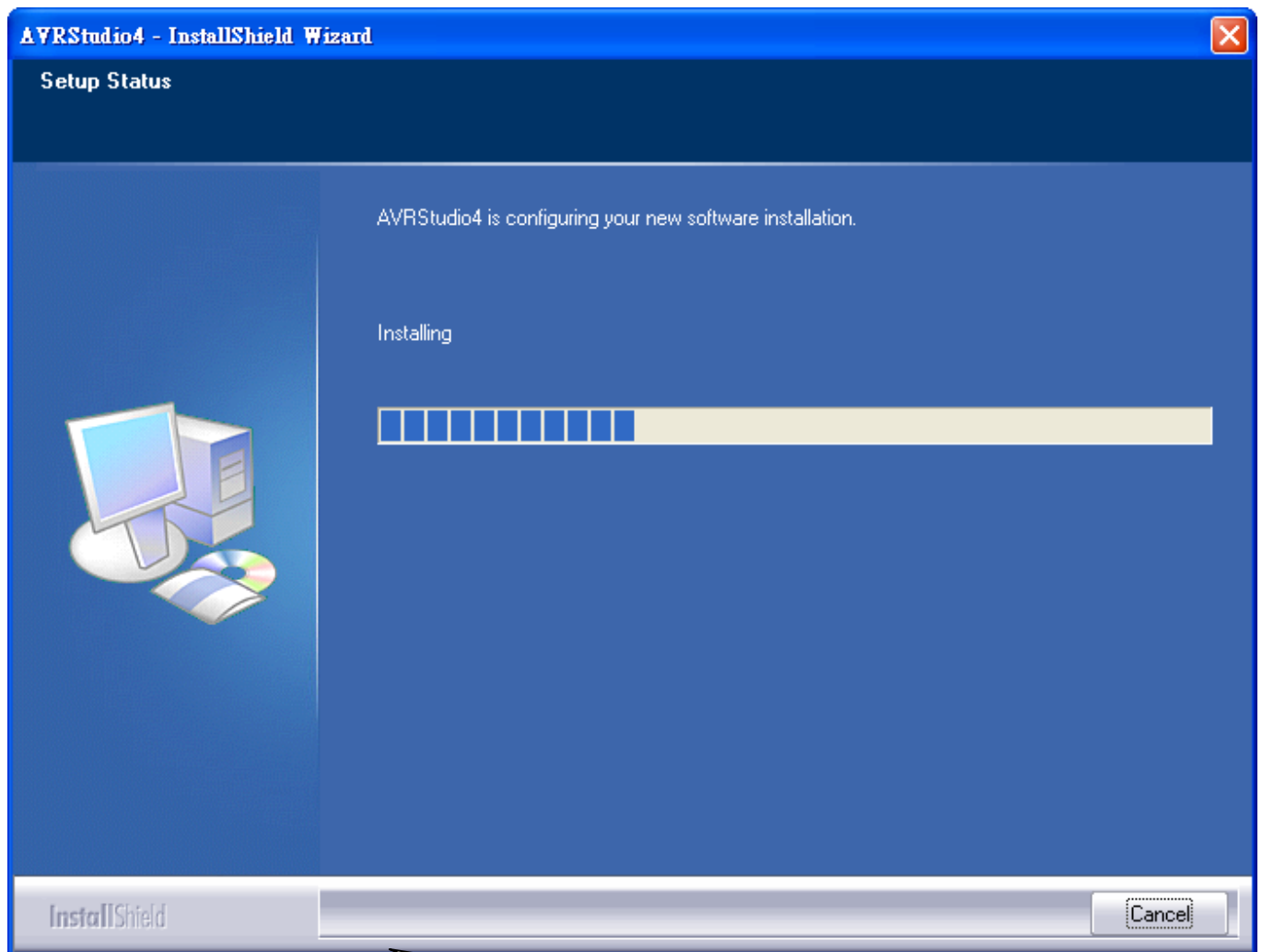
InstallShield

&lt; Back

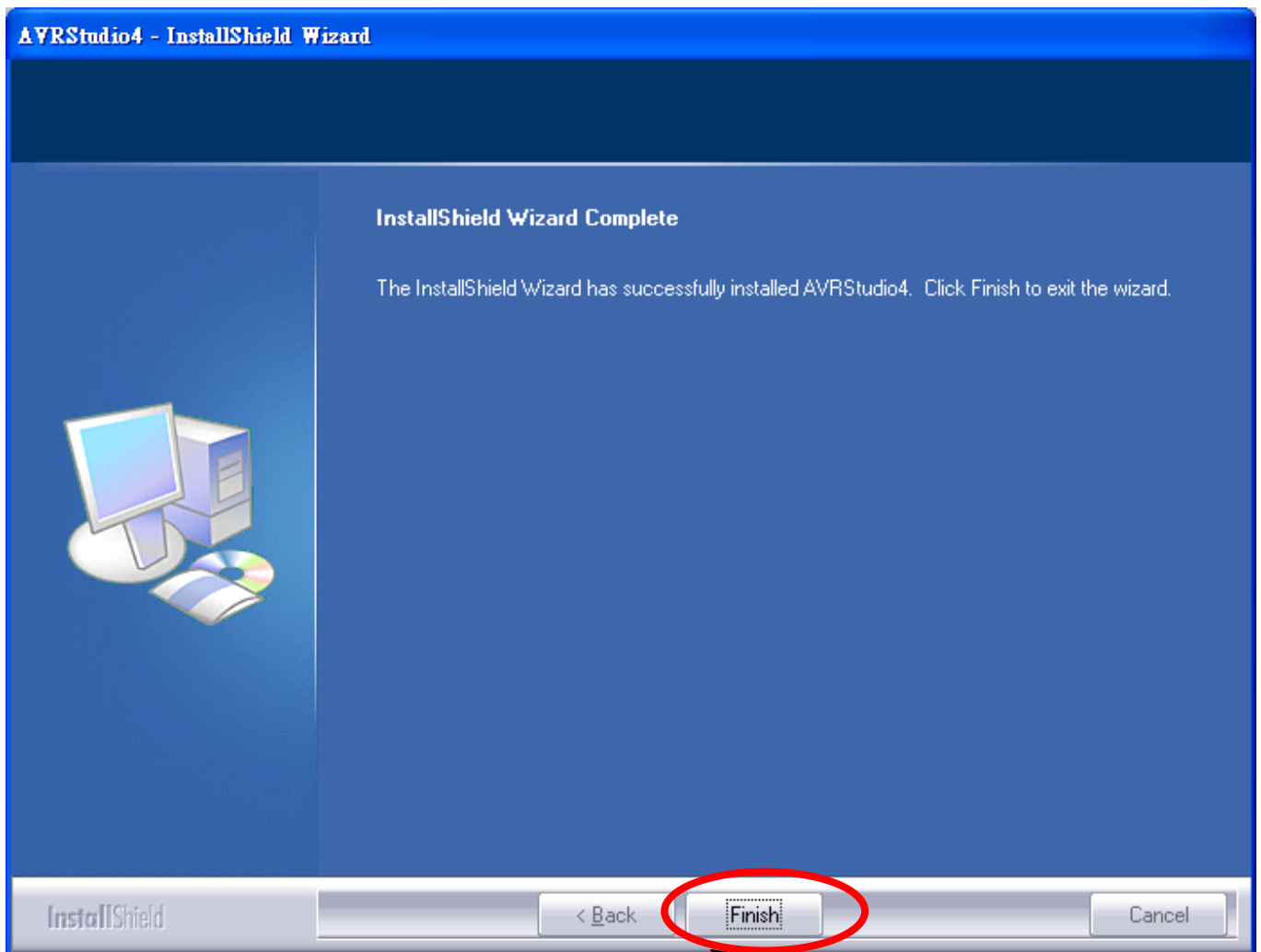
Install

Cancel

Click **install** to begin installation



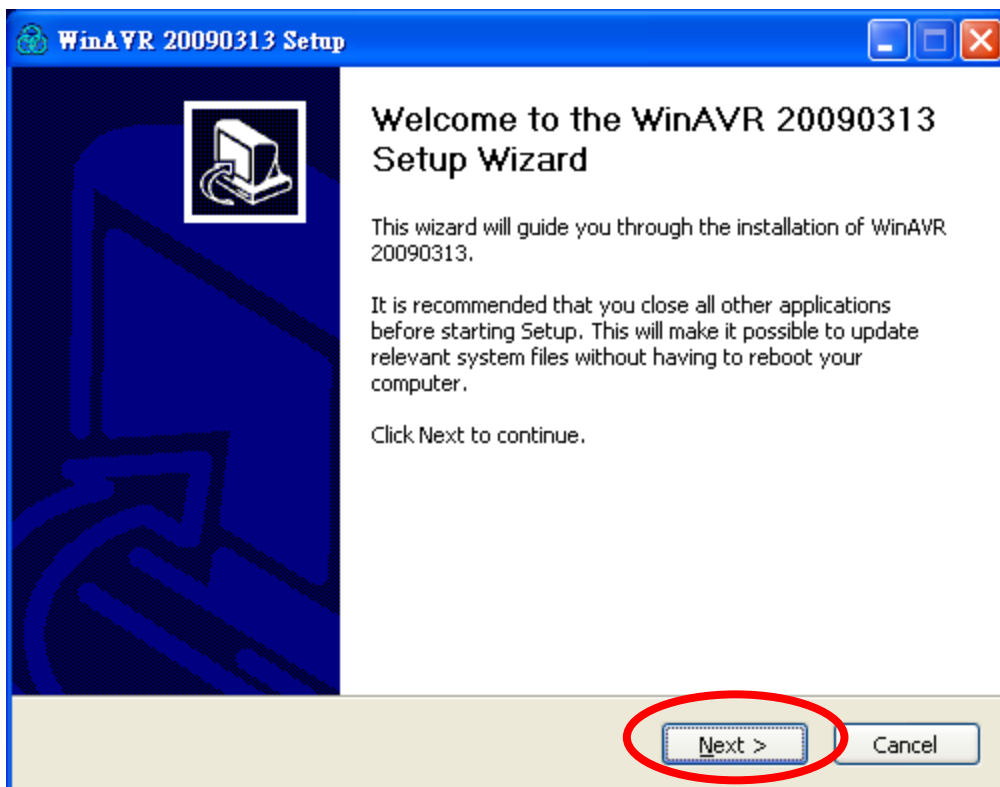
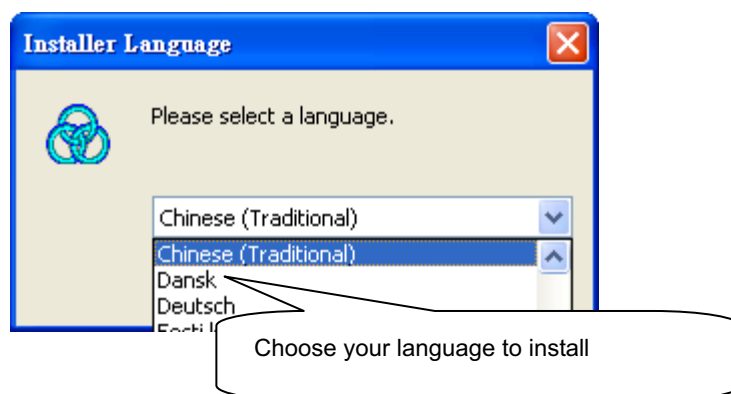
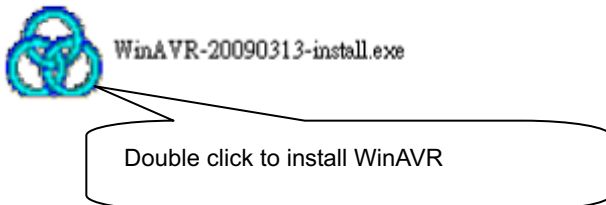
Wait for seconds for installation

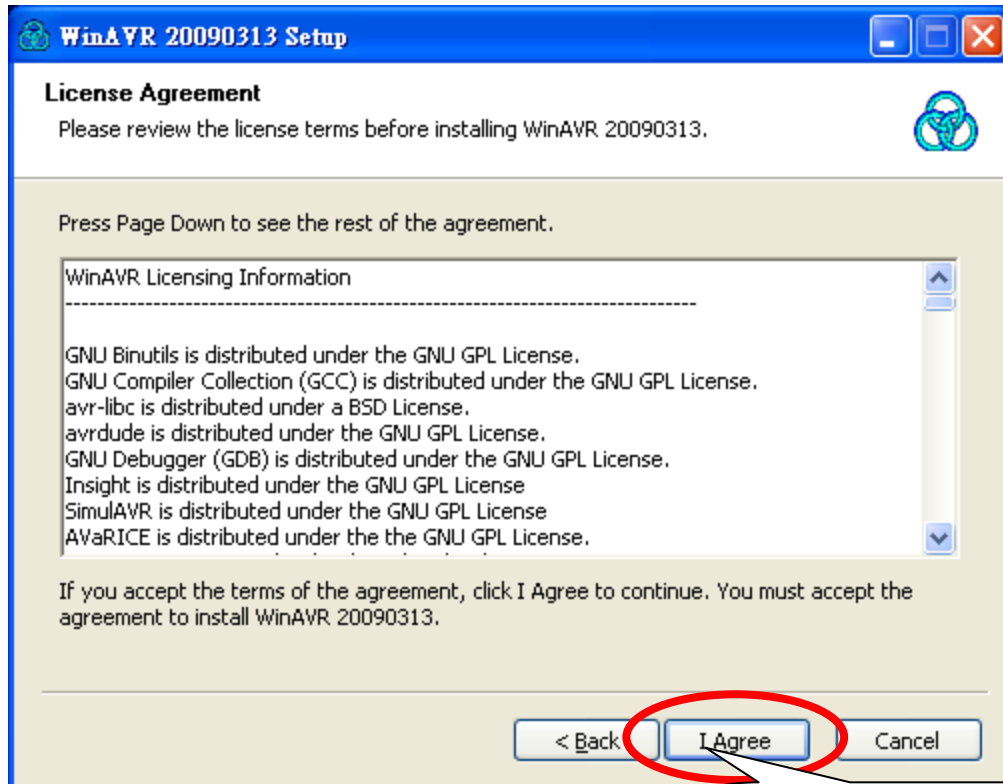


Click **finish** to exit the installation

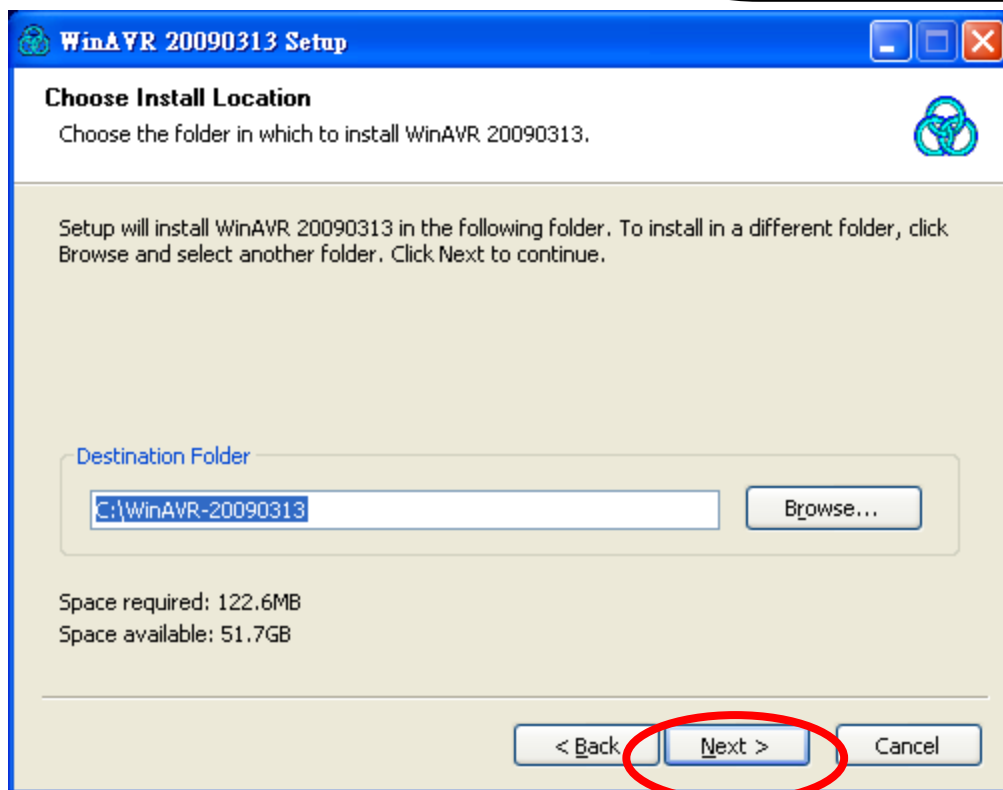
#### 4-1-2 Additional tool for C language

For C language designers, additionally please download and install AVR gcc from [http://sourceforge.net/project/downloading.php?group\\_id=68108&filename=WinAVR-20090313-install.exe&a=6759369](http://sourceforge.net/project/downloading.php?group_id=68108&filename=WinAVR-20090313-install.exe&a=6759369)





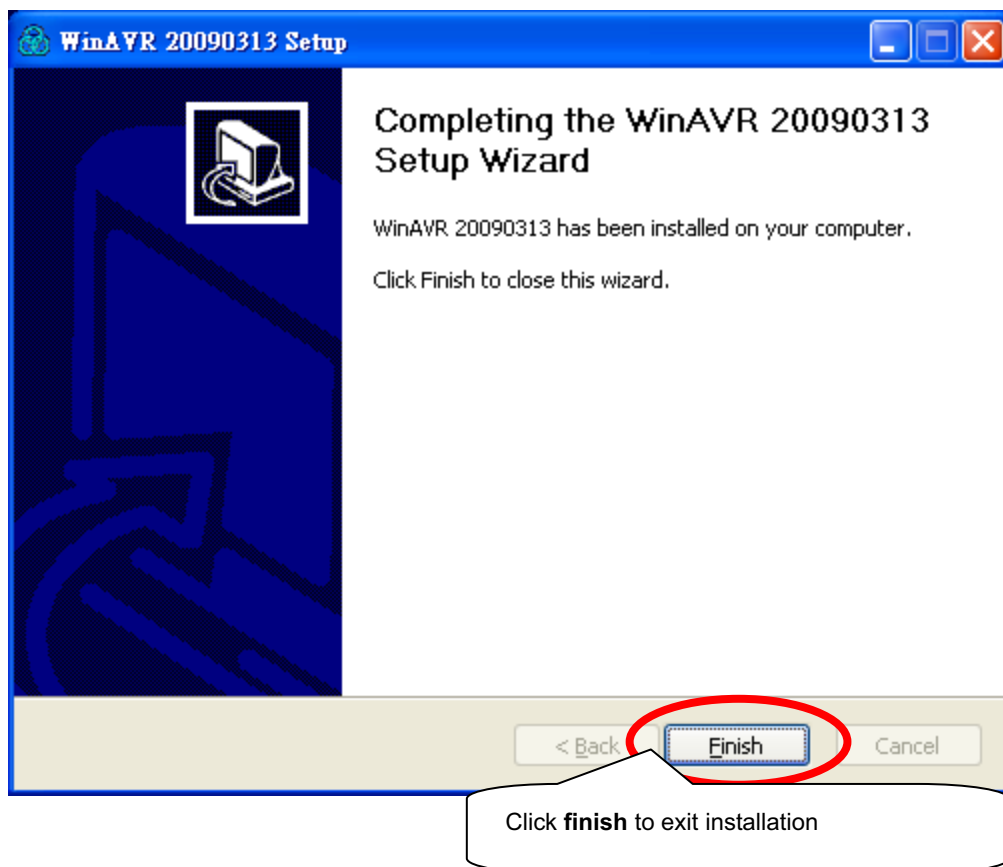
Click I agree to continue



Choose folder to continue



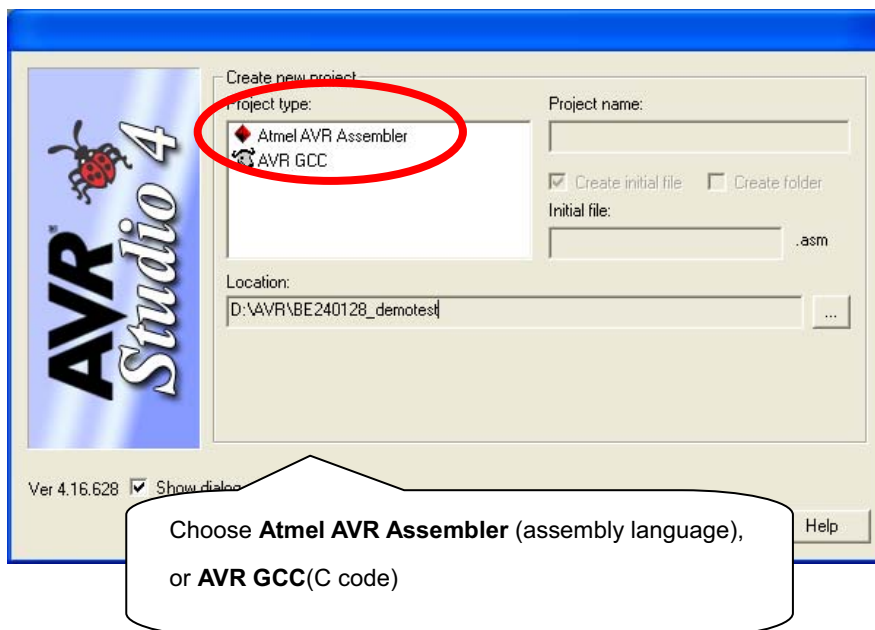
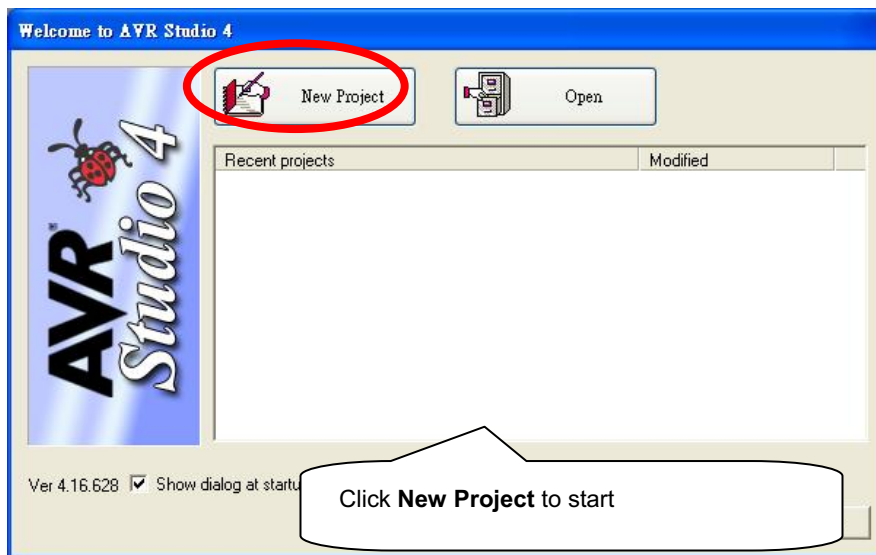
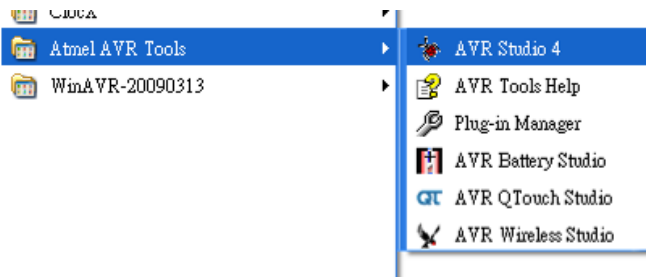


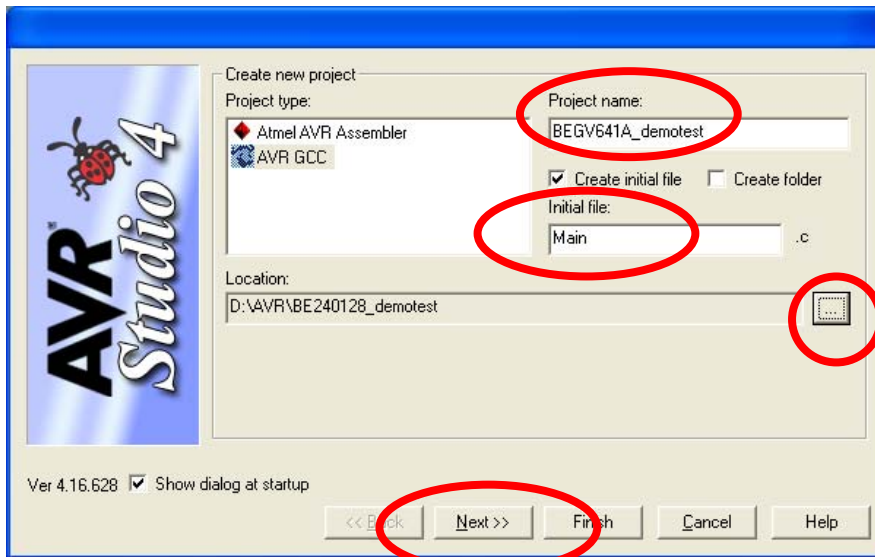


**\*\* AVR Studio and AVR gcc software are also available on Bolymin utility disk or Bolymin website.**

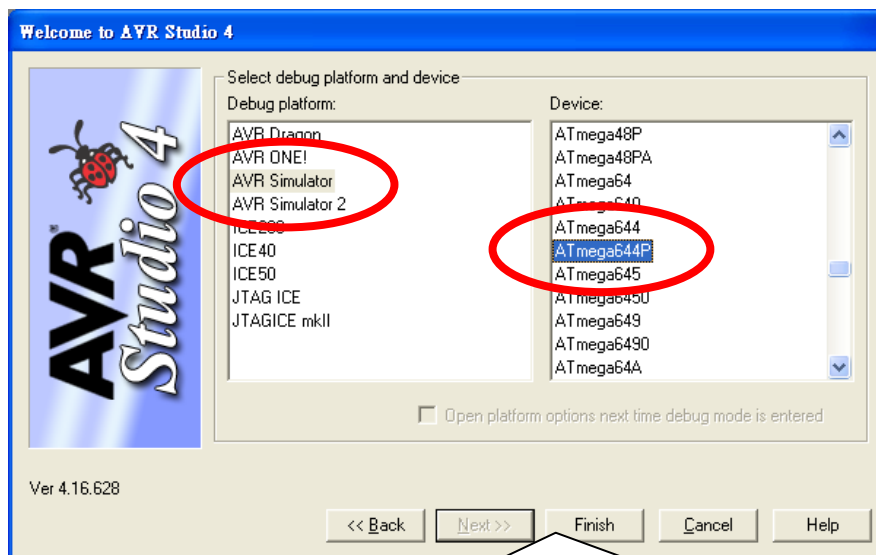
## 4-2 Execute AVR Studio 4.16 on designer PC

Start→All programs→Atmel AVR Tools→AVR Studio 4

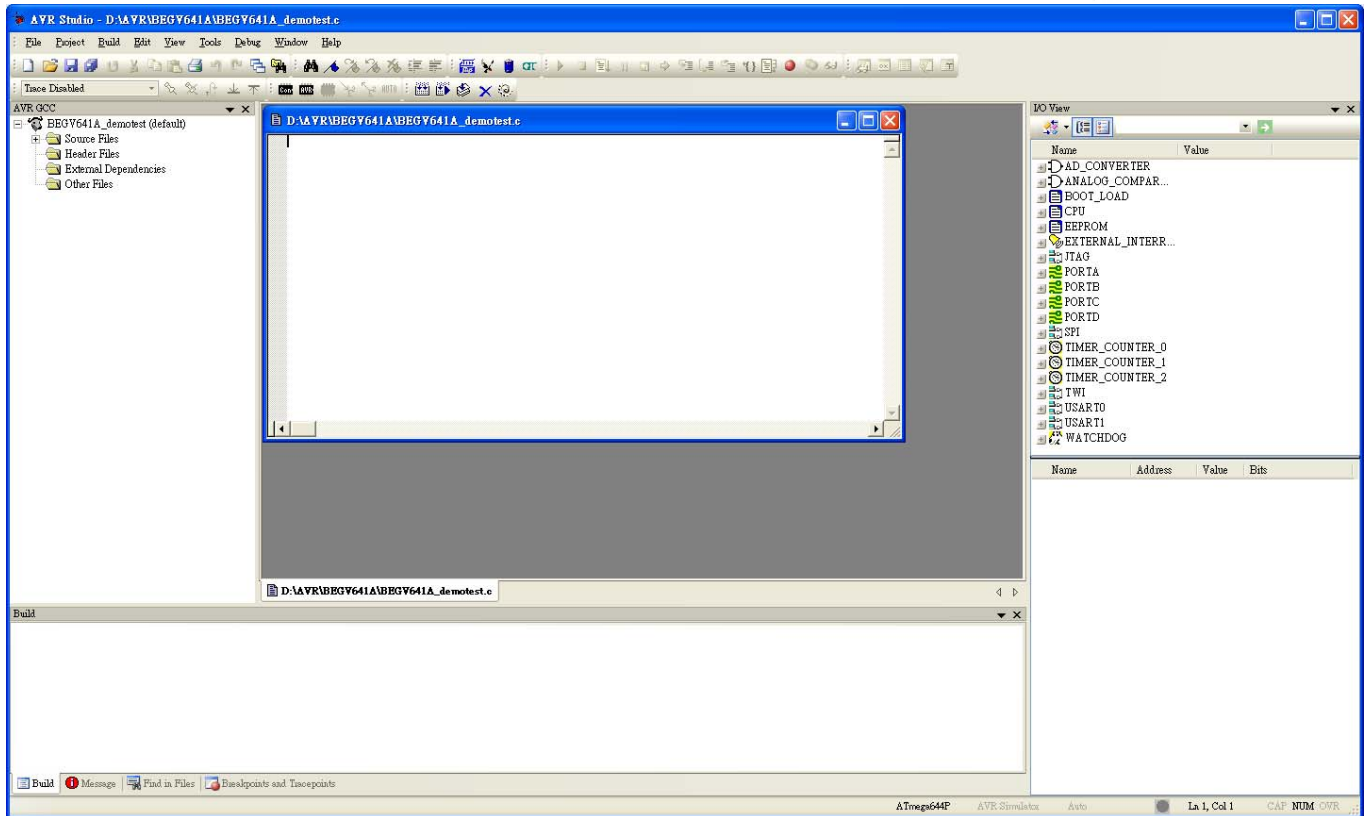




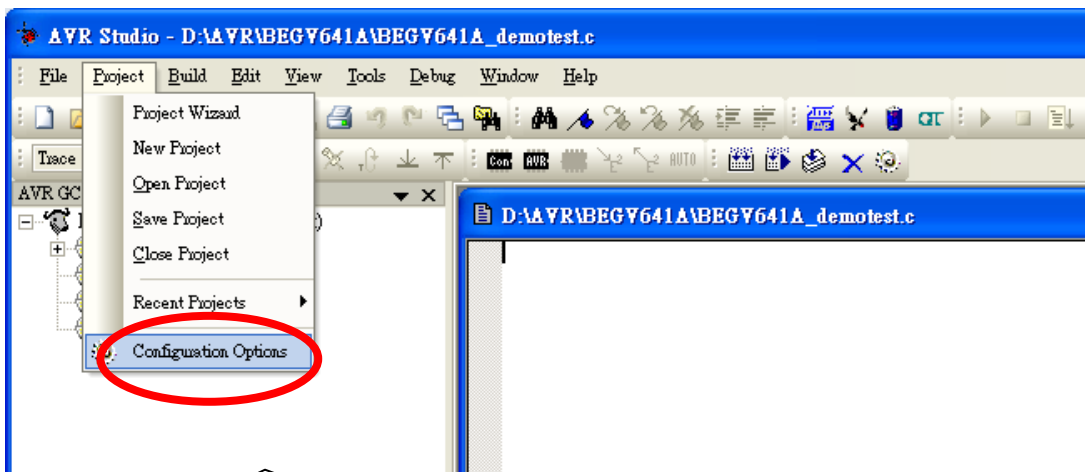
Choose **AVR GCC** here (All Bolymin utility drivers are designed with C code), and key in Project Name and Initial file.



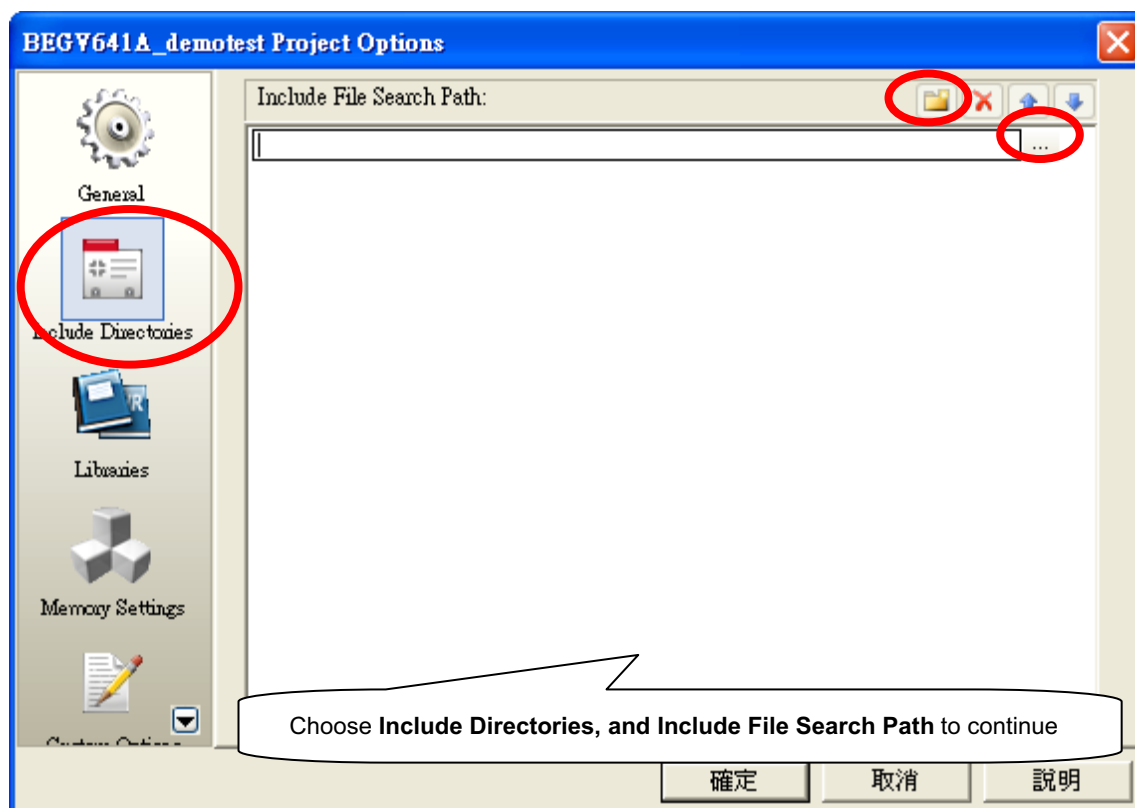
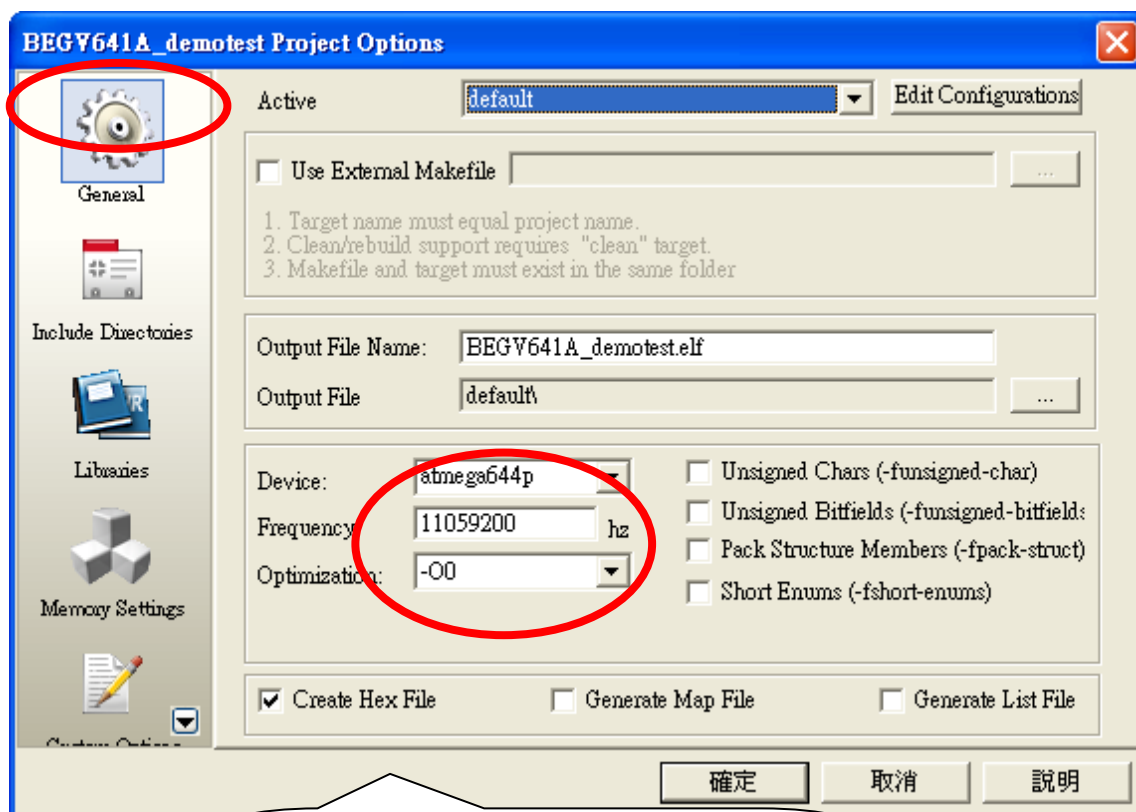
Choose **AVR Simulator**, **ATmega644P**, and click **Finish** to continue

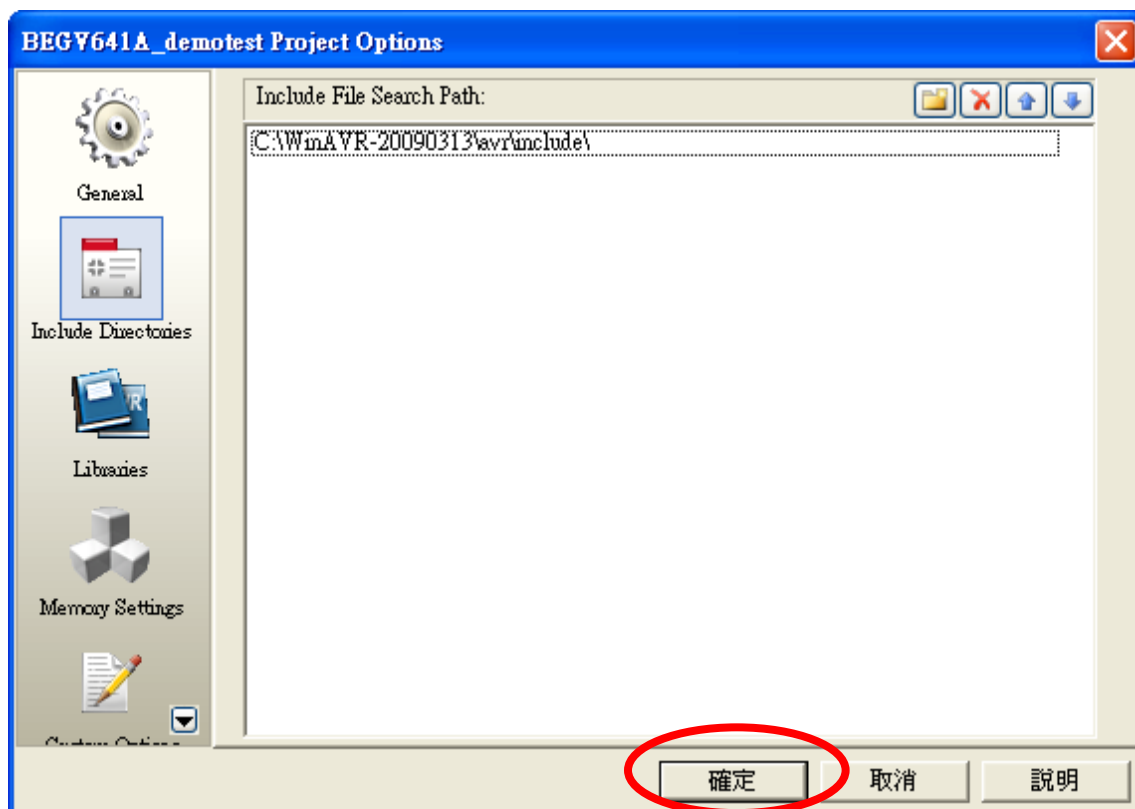
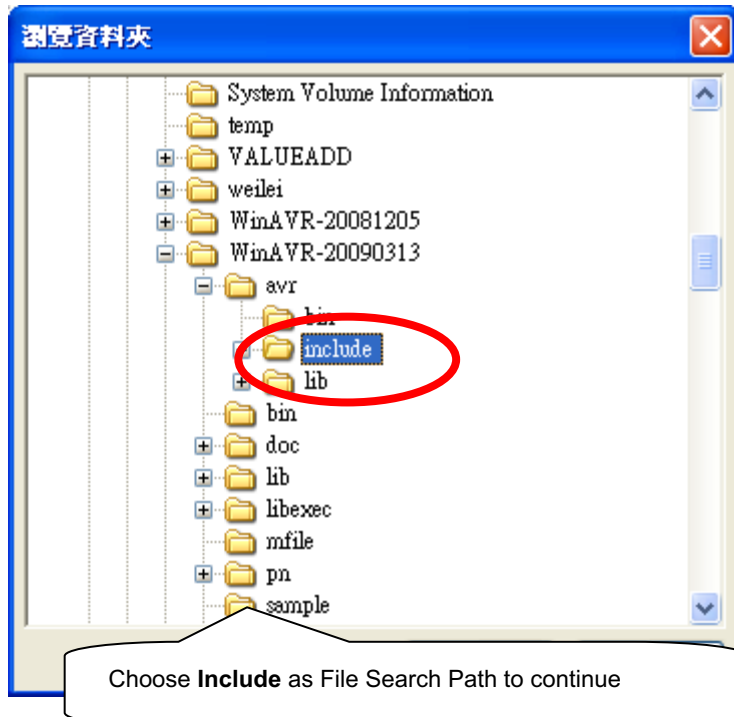


Here you can see software design screen, yet no hurry to start software design yet.

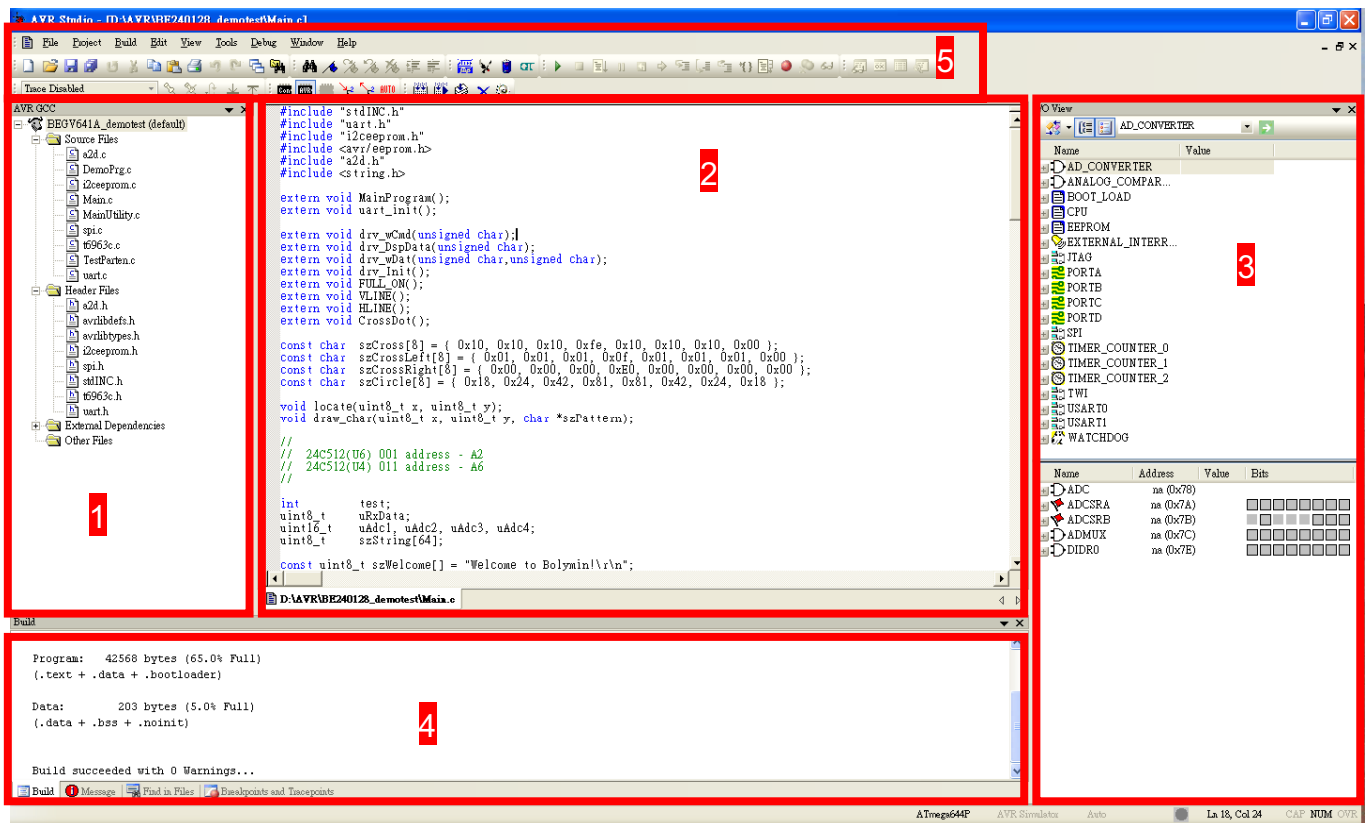


Choose **Project** → **Configuration Options** to continue

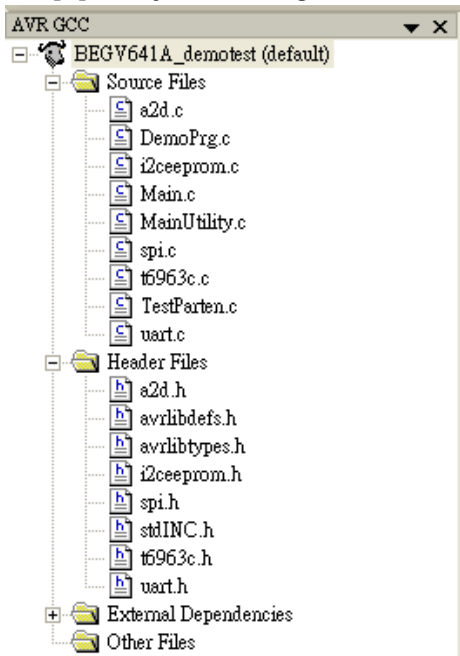




After executing AVR Studio 4.16 on PC, designer can see key information on 5 windows:

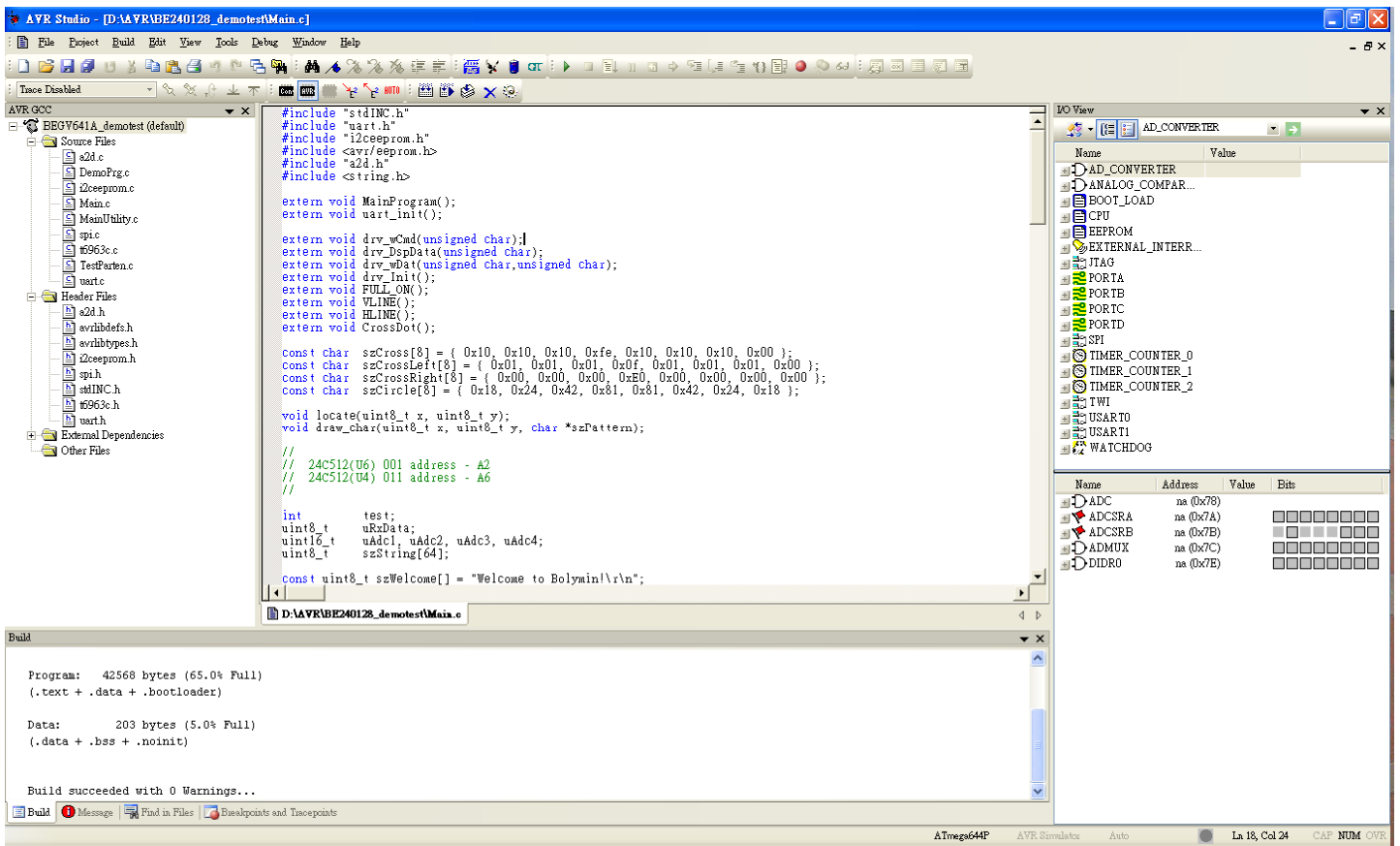


### [1]. Project management window





## [2]. Program editing window



The screenshot displays the AVR Studio IDE interface. The main window shows the source code for `BEKV641A_demoTest(Main.c)`. The code includes standard headers, defines, and functions for a microcontroller program. The project explorer on the left shows the file structure. The I/O View on the right lists the hardware components used in the project. The Build window at the bottom shows the compilation results.

**Source Code:**

```
#include "stdINC.h"
#include "uart.h"
#include "i2ceeprom.h"
#include <avr/eeprom.h>
#include "a2d.h"
#include <string.h>

extern void MainProgram();
extern void uart_init();

extern void drv_wCmd(unsigned char);
extern void drv_DspData(unsigned char);
extern void drv_wDat(unsigned char,unsigned char);
extern void drv_Init();
extern void FULL_ON();
extern void VLINE();
extern void HLINE();
extern void CrossDot();

const char szCross[8] = { 0x10, 0x10, 0x10, 0xfe, 0x10, 0x10, 0x10, 0x00 };
const char szCrossLeft[8] = { 0x01, 0x01, 0x01, 0x0f, 0x01, 0x01, 0x01, 0x00 };
const char szCrossRight[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
const char szCircle[8] = { 0x18, 0x24, 0x42, 0x81, 0x81, 0x42, 0x24, 0x18 };

void locate(uint8_t x, uint8_t y);
void draw_char(uint8_t x, uint8_t y, char *szPattern);

// 24C512(U6) 001 address - A2
// 24C512(U4) 011 address - A6

int test;
uint8_t uRxDat;
uint16_t uAdc1, uAdc2, uAdc3, uAdc4;
uint8_t szString[64];

const uint8_t szWelcome[] = "Welcome to Bolymin!\r\n";
```

**Build Output:**

```
Program: 42568 bytes (65.0% Full)
(.text + .data + .bootloader)

Data: 203 bytes (5.0% Full)
(.data + .bss + .noinit)

Build succeeded with 0 Warnings...
```

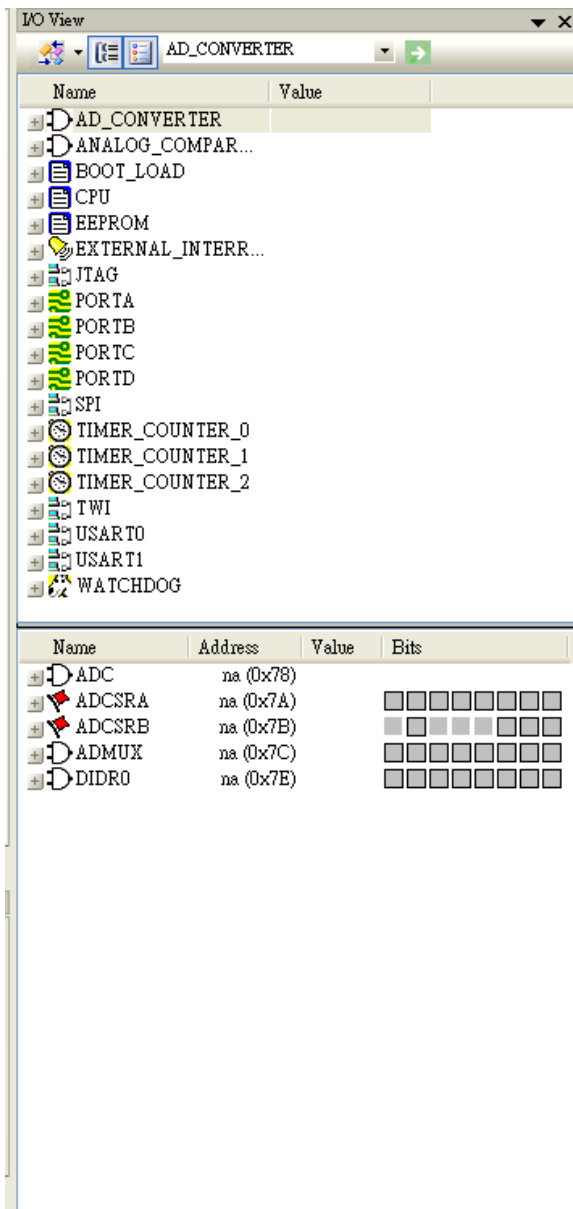
**I/O View:**

Name	Value
AD_CONVERTER	
ANALOG_COMPAR...	
BOOT_LOAD	
CPU	
EEPROM	
EXTERNAL_INTERR...	
JTAG	
PORTA	
PORTB	
PORTC	
PORTD	
SPI	
TIMER_COUNTER_0	
TIMER_COUNTER_1	
TIMER_COUNTER_2	
TWI	
USART0	
USART1	
WATCHDOG	

**Build Details:**

Name	Address	Value	Bits
ADC	na (0x78)		
ADCSRA	na (0x7A)		
ADCSRB	na (0x7B)		
ADMUX	na (0x7C)		
ADIDRO	na (0x7E)		

### [3]. I/O view window



### [4]. Message window



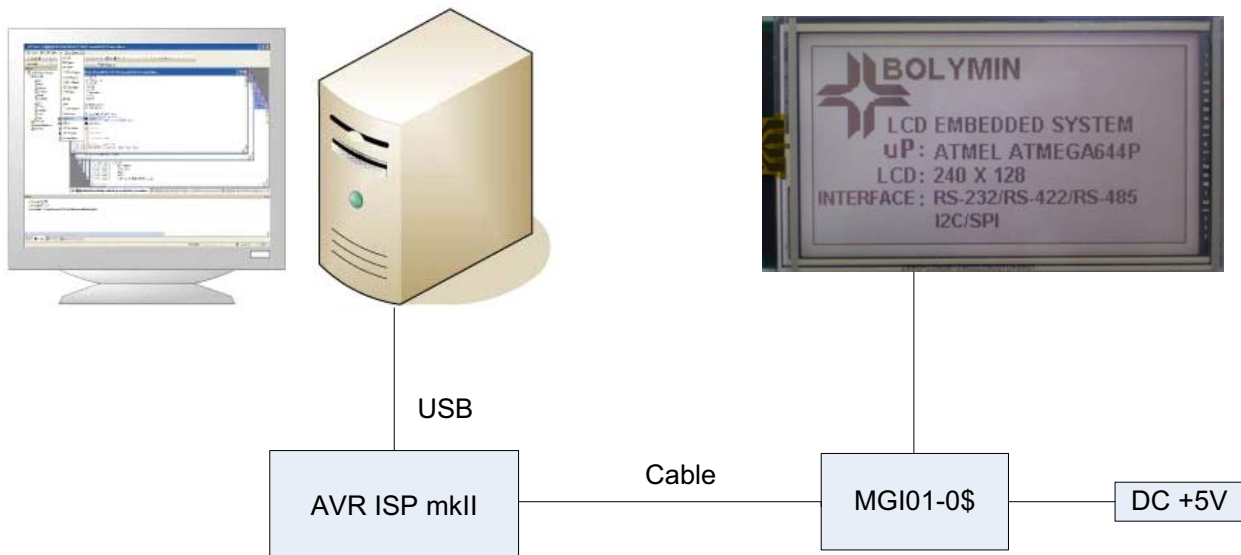
### [5]. Toolbars



### 4-3 In-System Programmer AVR ISP mkII

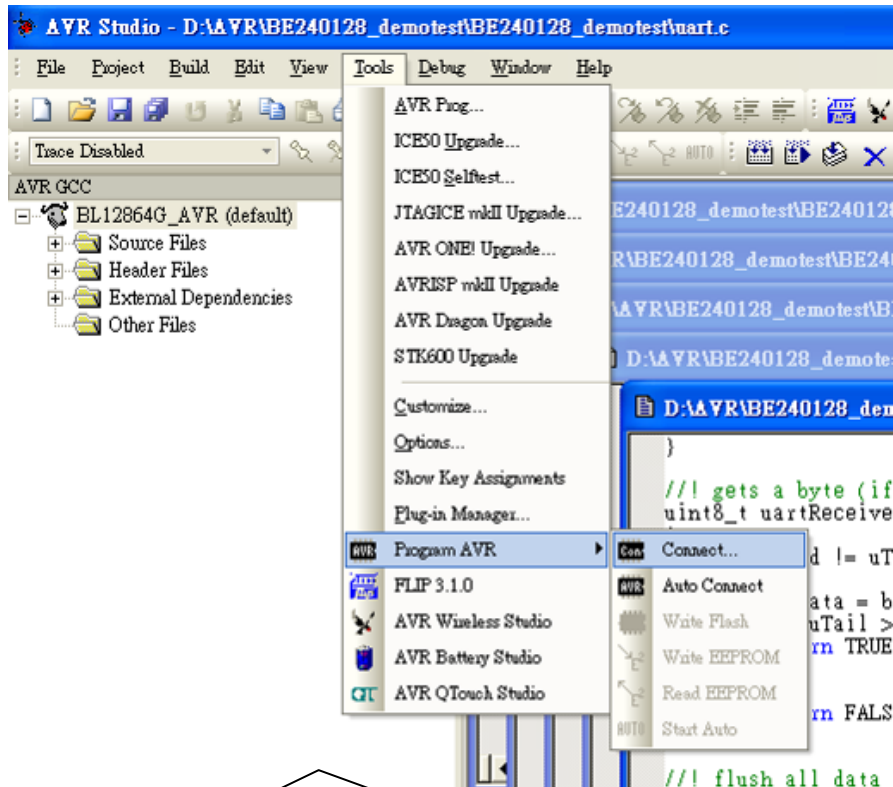


\*\* Inside mkII box(1), designer will get DVD(2), mkII device(3), and USB cable(4). Please be sure that all parts are packed inside.

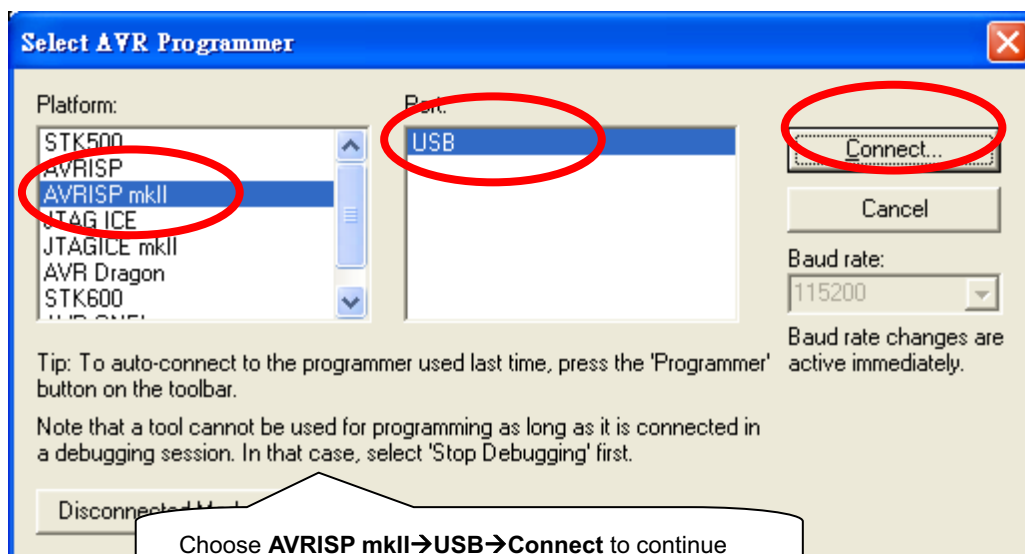


Steps:

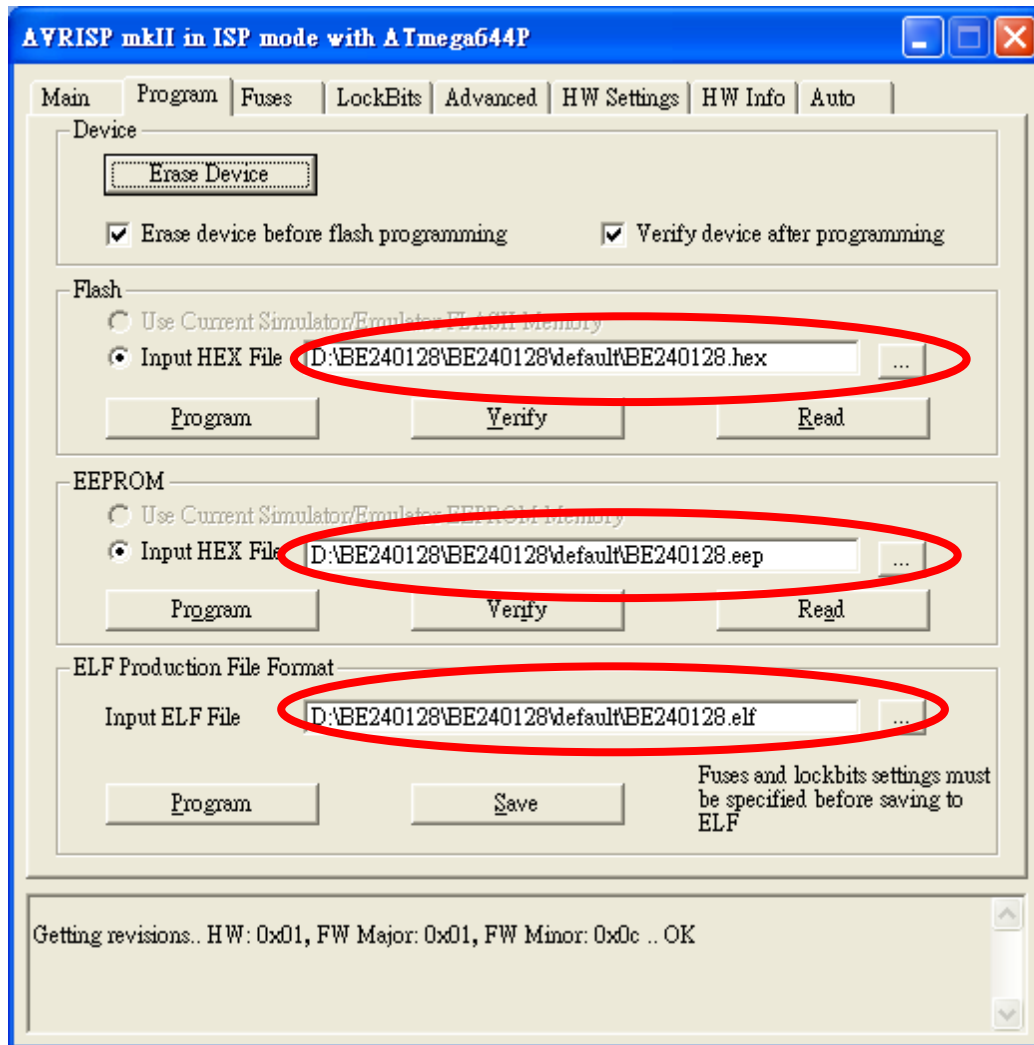
1. Get mkII connected to PC



Choose **Tools→Program AVR→Connect** to continue



Choose **AVRISP mkII→USB→Connect** to continue

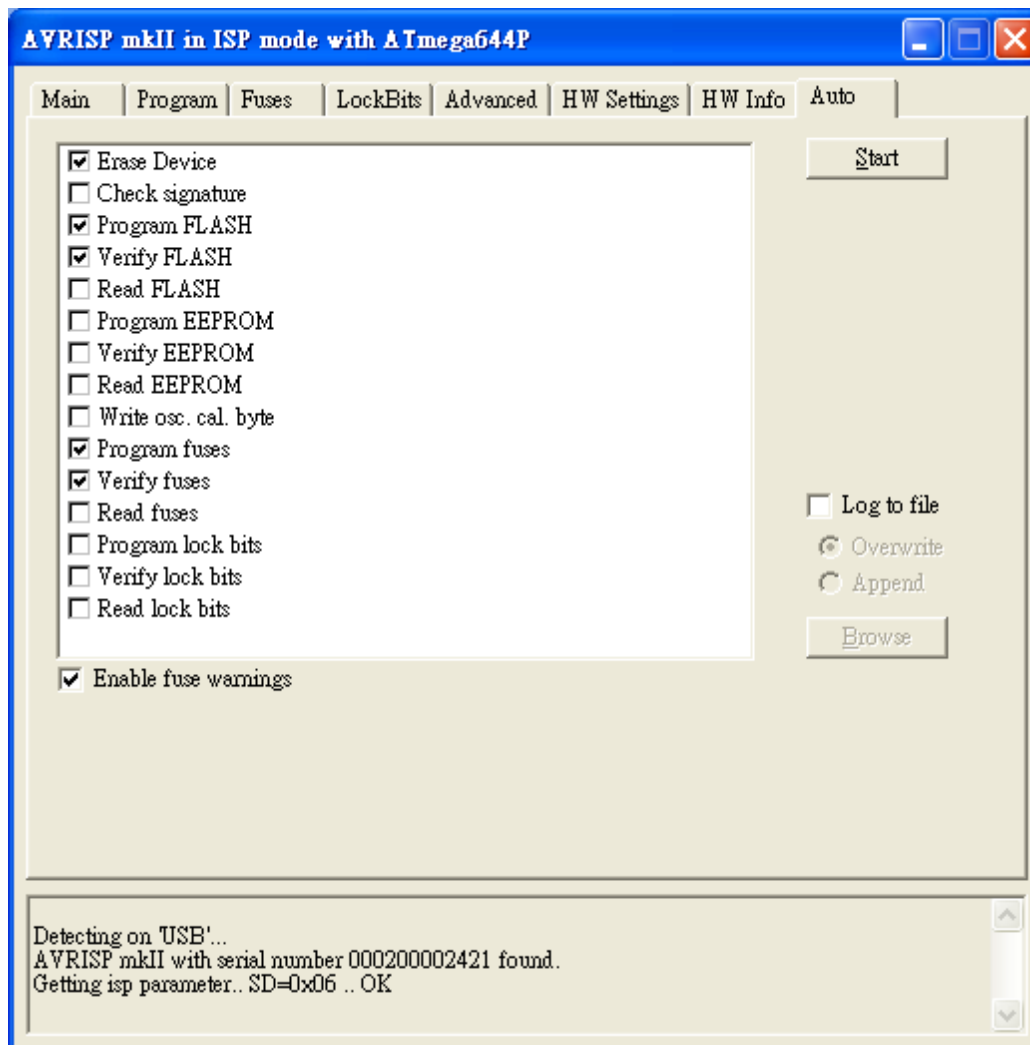


Choose Hex file for ATmega644p flash → **Program**

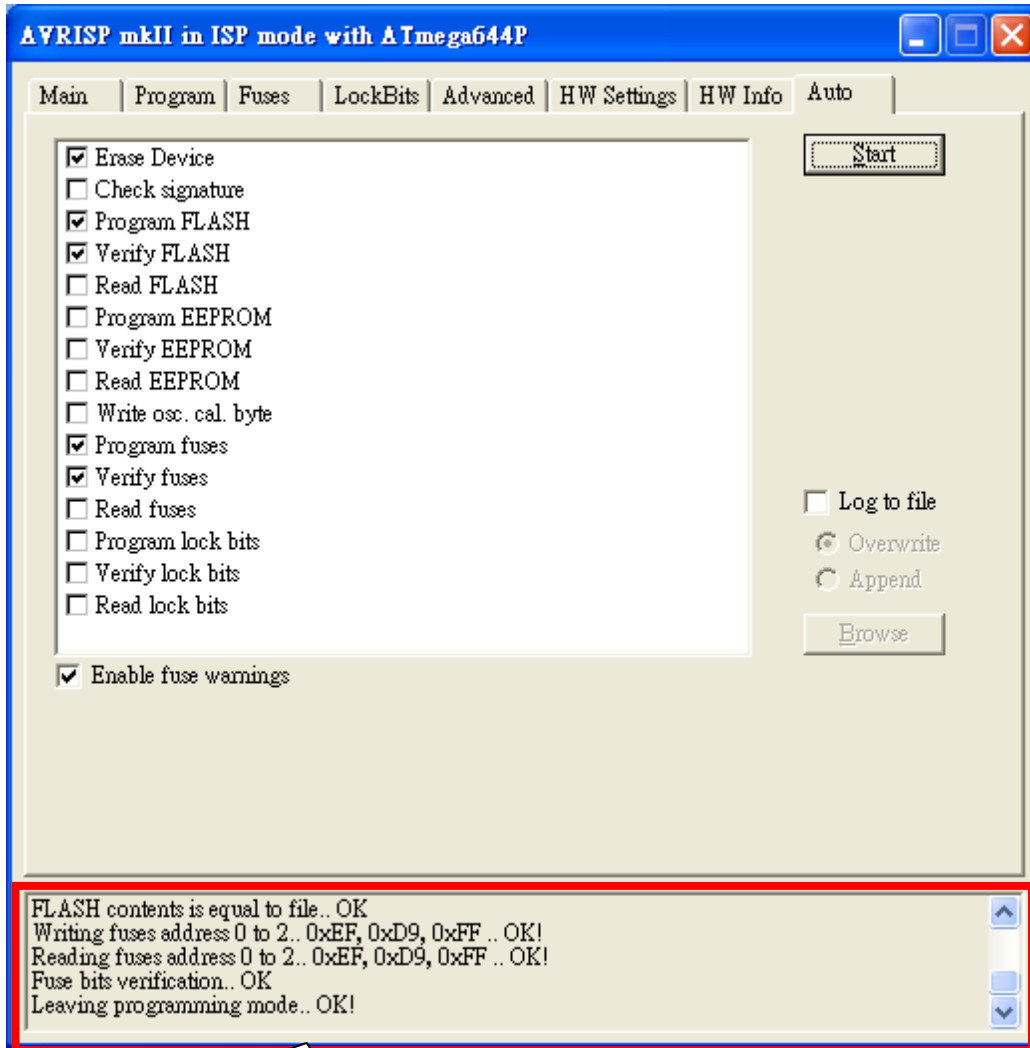
Choose Hex file for ATmega644p EEPROM→**Program**

Choose ELF file for fuses and lockbits→ **Program**

Start software burning

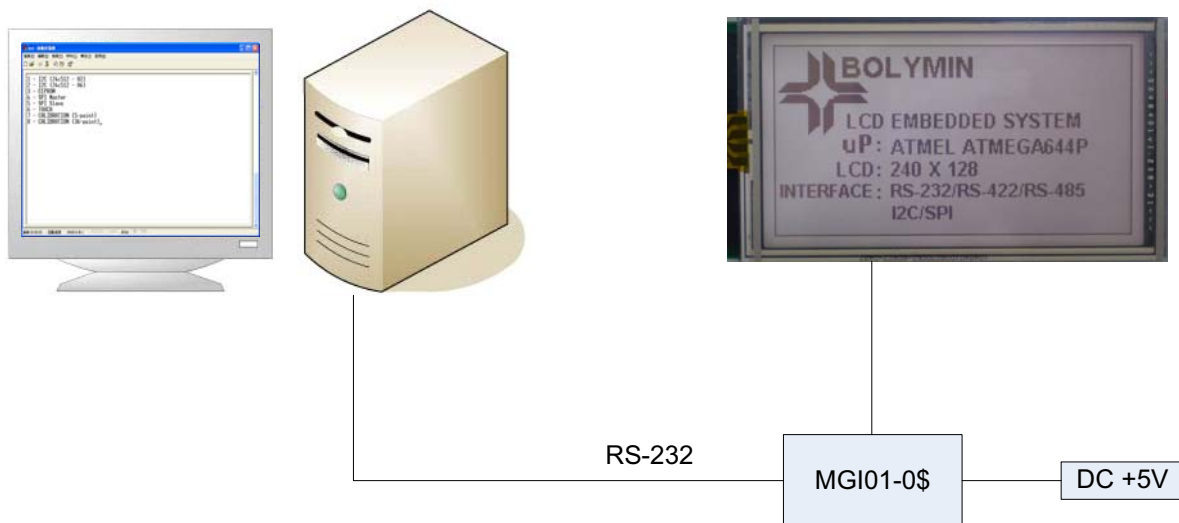


Choose **Auto** and necessary configure  
and click **Start** to program



Good job! Software burning is done!

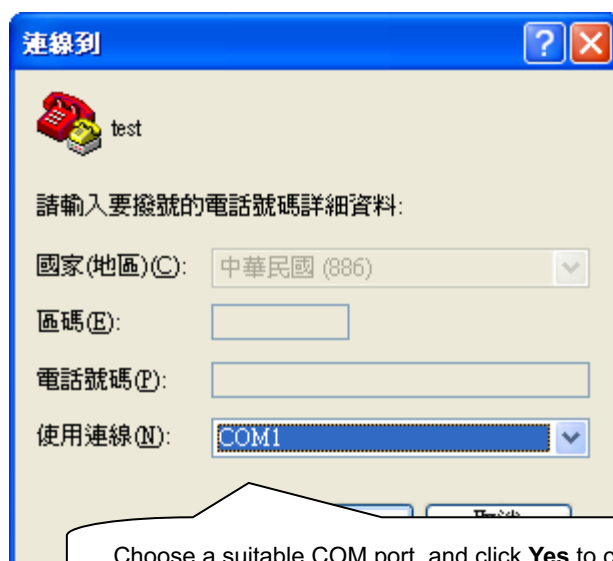
## Product Function Verify



On PC: Start→All programs→Telecommunication→Hyper Terminal



Please define a name and choose a icon for this connection, and click **Yes** to continue.



Choose a suitable COM port, and click **Yes** to continue.



**COM1 內容** [?] [X]

**連接埠設定**

每秒傳輸位元(B): 115200 [v]

資料位元(D): 8 [v]

同位檢查(P): 無 [v]

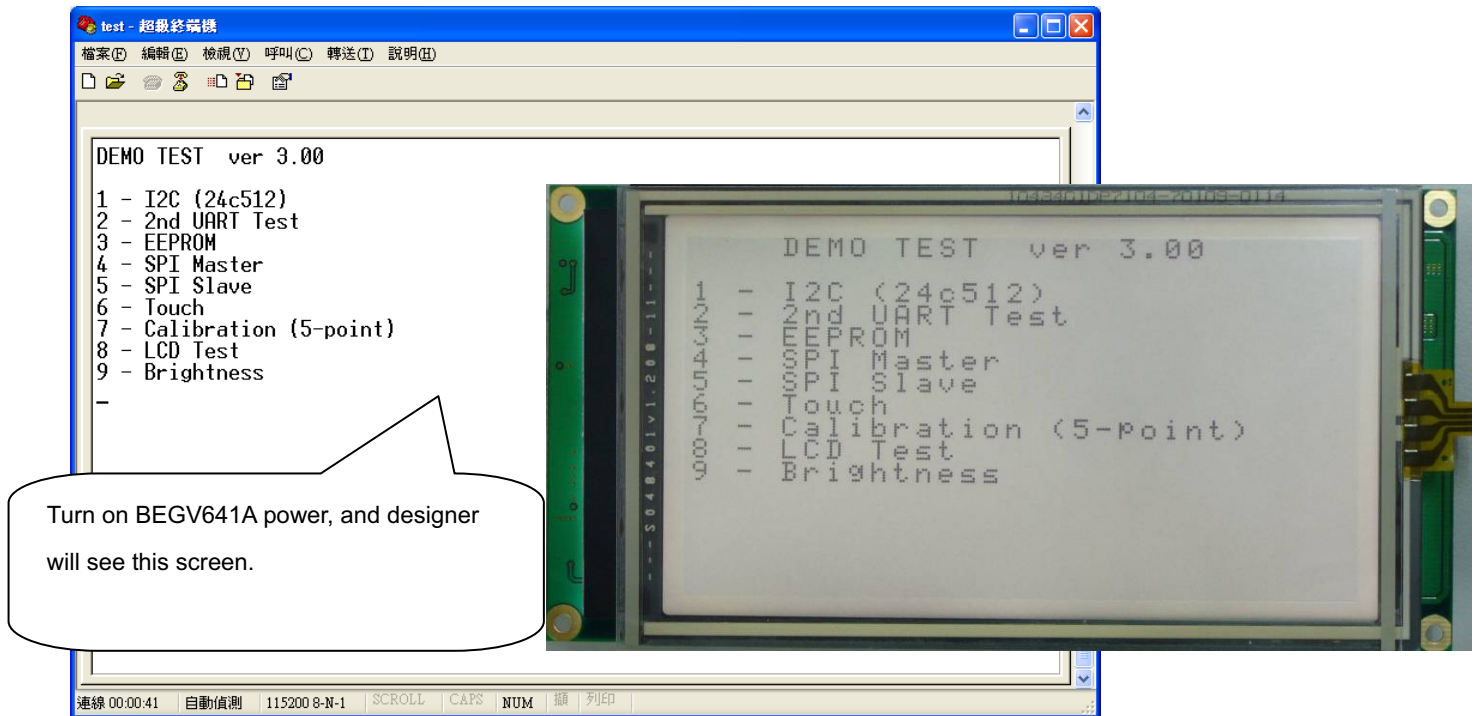
停止位元(S): 1 [v]

流量控制(F): 無 [v]

還原成預設值(R)

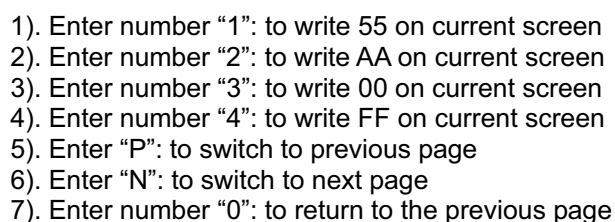
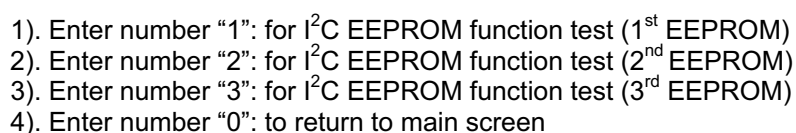
確定 取消 套用(A)

Key in COM port setting:115200/8/No/1/No,  
and click **Yes** to continue.

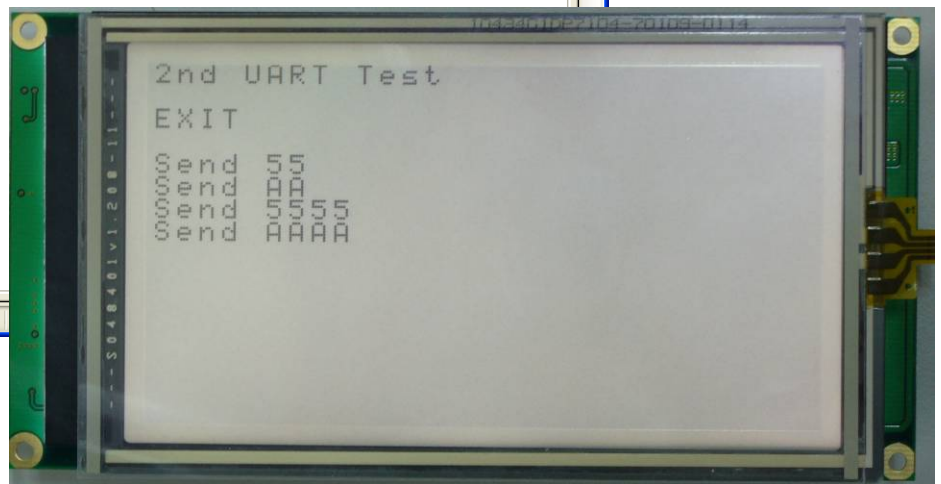
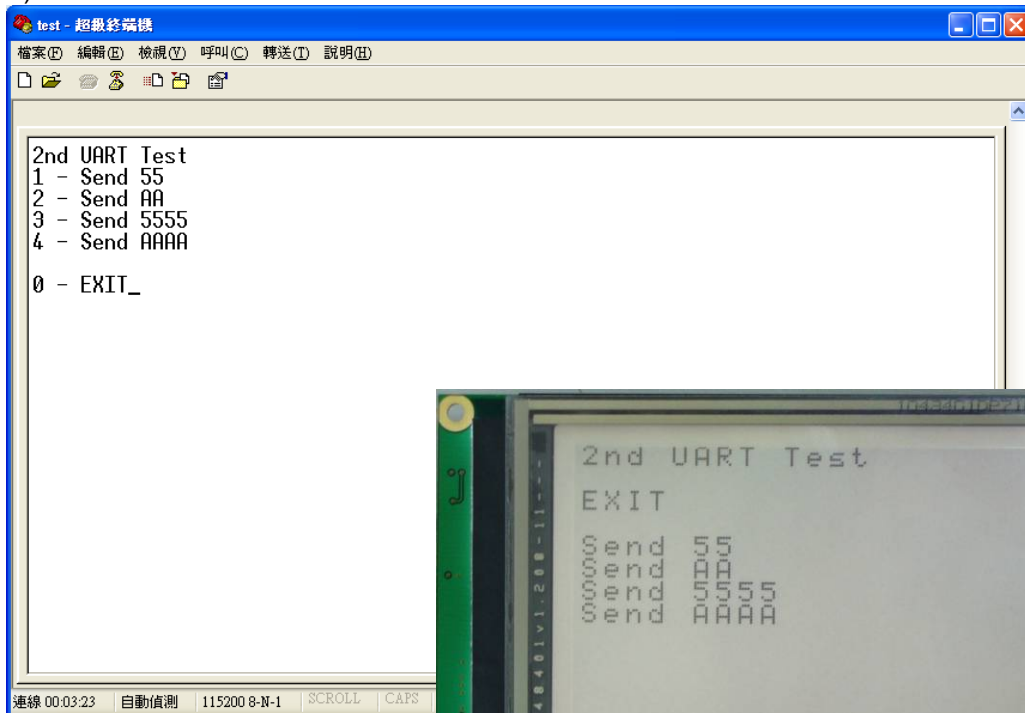


After seeing above screen, designer can operate on PC & BEGV641A:

- A). Enter number "1": for I<sup>2</sup>C EEPROM function test (1<sup>st</sup> 2<sup>nd</sup> 3<sup>rd</sup> EEPROM)
- B). Enter number "2": for 2<sup>nd</sup> UART function test
- C). Enter number "3": for ATmega644P internal EEPROM function test
- D). Enter number "4": for 4-wire SPI Master function test
- E). Enter number "5": for 4-wire SPI Slave function test
- F). Enter number "6": to touch panel function test
- G). Enter number "7": for touch panel calibration (5 point)
- H). Enter number "8": for LCD function test
- I). Enter number "9": for backlight brightness adjustment function test



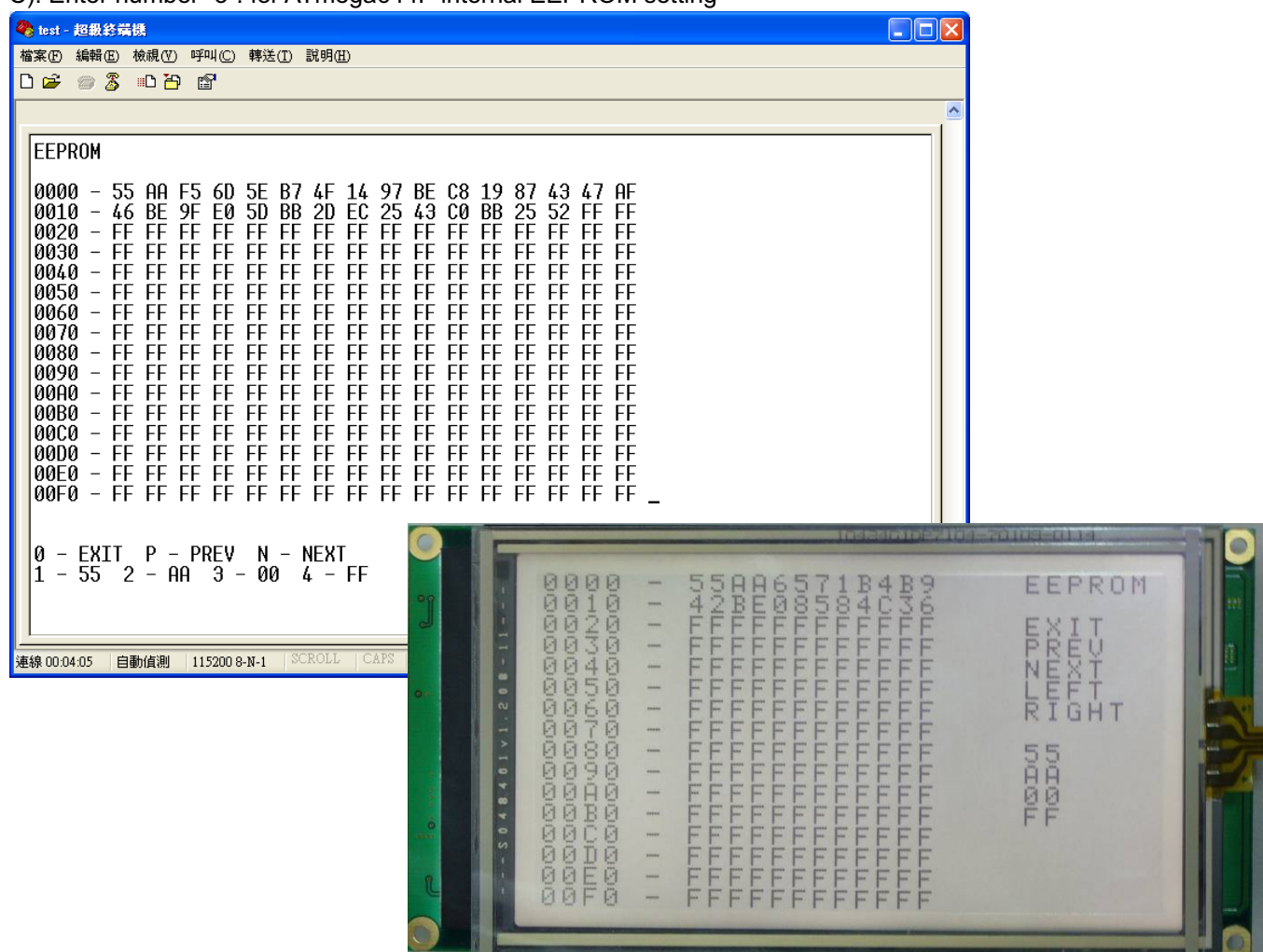
B). Enter number "2": for 2<sup>nd</sup> UART function test



- 1). Enter number "1": to send 1 byte (0x55)
- 2). Enter number "2": to send 1 byte (0xAA)
- 3). Enter number "3": to send 1 word (0x5555)
- 4). Enter number "4": to send 1 byte (0xAAAA)
- 5). Enter number "0": to return to main screen

Auto receive byte or word

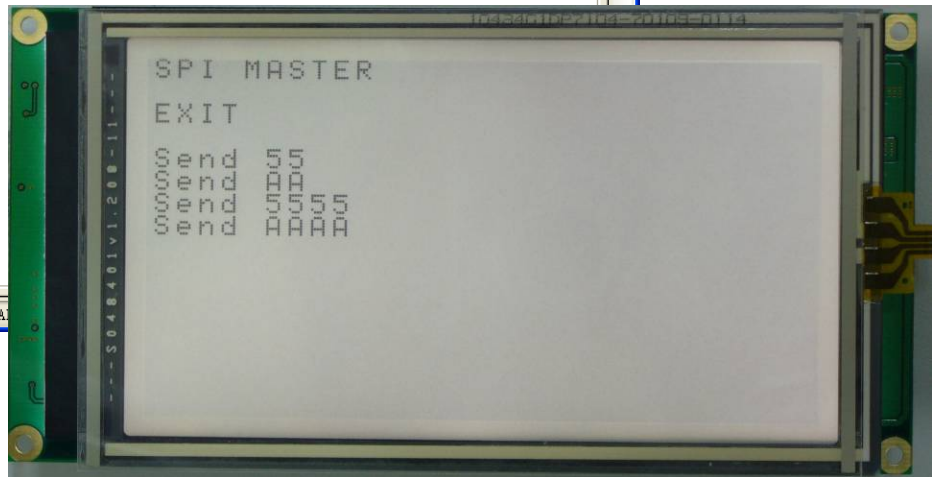
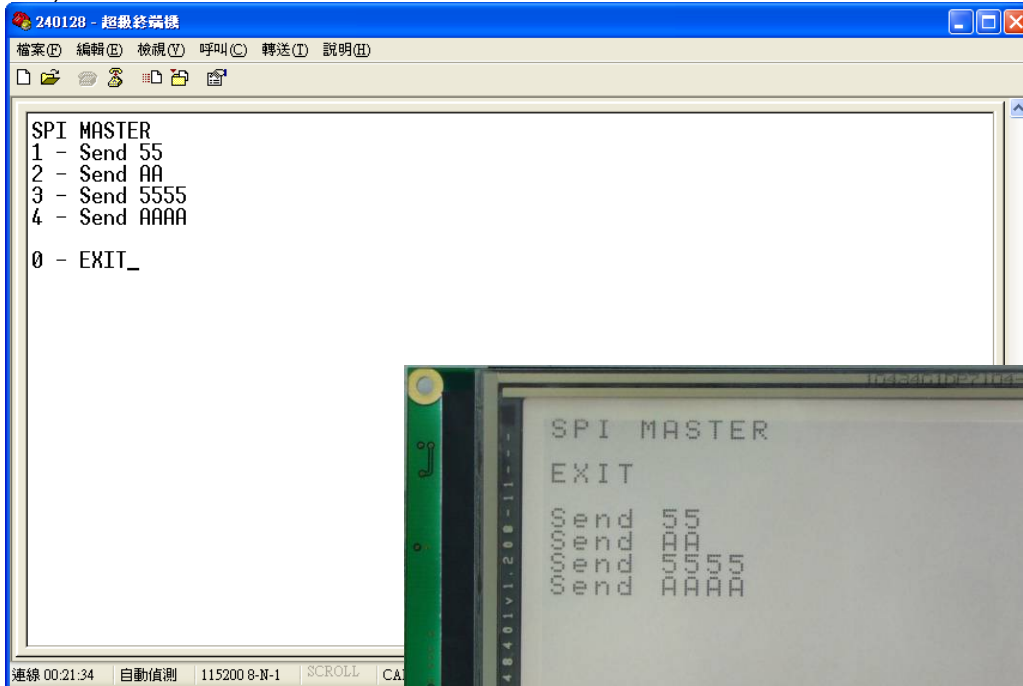
C). Enter number “3”: for ATmega644P internal EEPROM setting



\*\* Address 0x0000~0x001D is for touch panel calibration data

- 1). Enter number “1”: to write 55 on current screen
- 2). Enter number “2”: to write AA on current screen
- 3). Enter number “3”: to write 00 on current screen
- 4). Enter number “4”: to write FF on current screen
- 5). Enter “P”: to switch to previous page
- 6). Enter “N”: to switch to next page
- 7). Enter number “0”: to return to main screen

D). Enter number “4”: for 4-wire SPI Master function test



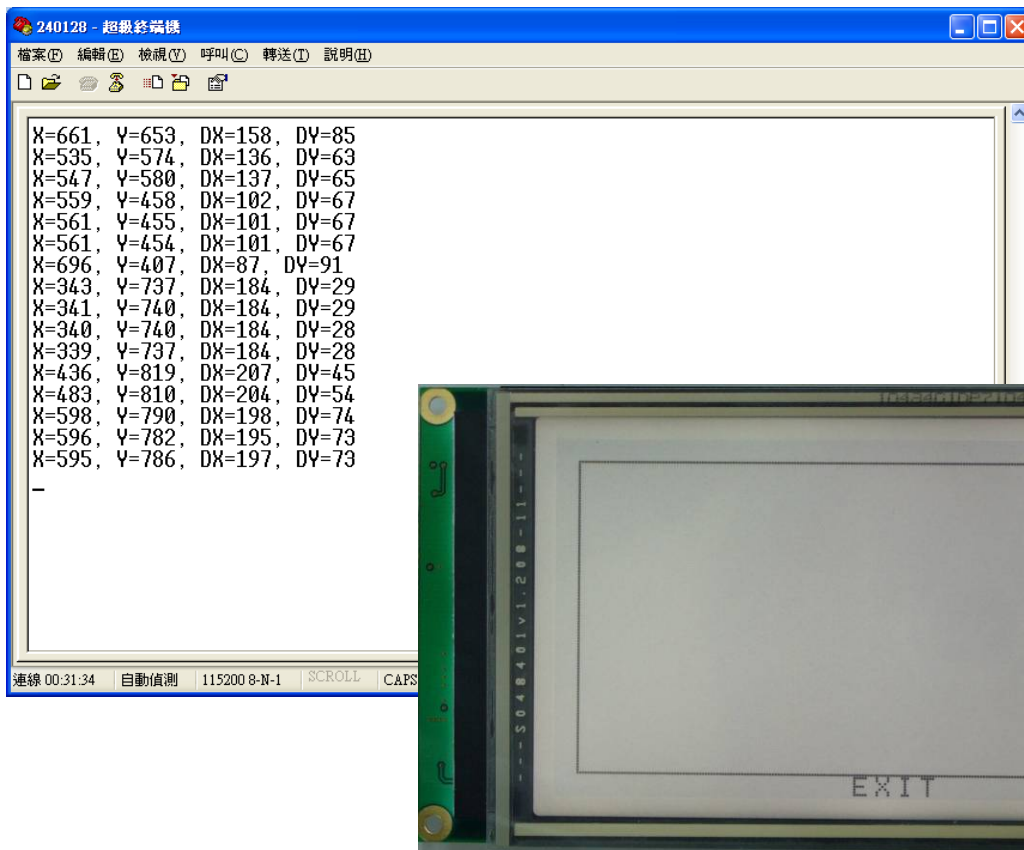
- 1). Enter number “1”: to send 1 byte (0x55)
- 2). Enter number “2”: to send 1 byte (0xAA)
- 3). Enter number “3”: to send 1 word (0x5555)
- 4). Enter number “4”: to send 1 byte (0xAAAA)
- 5). Enter number “0”: to return to main screen

E). Enter number "5": for 4-wire SPI Slave function test



- 1). Enter number "1": to receive 1 byte
- 2). Enter number "2": to receive 1 word
- 3). Enter number "0": to return to main screen

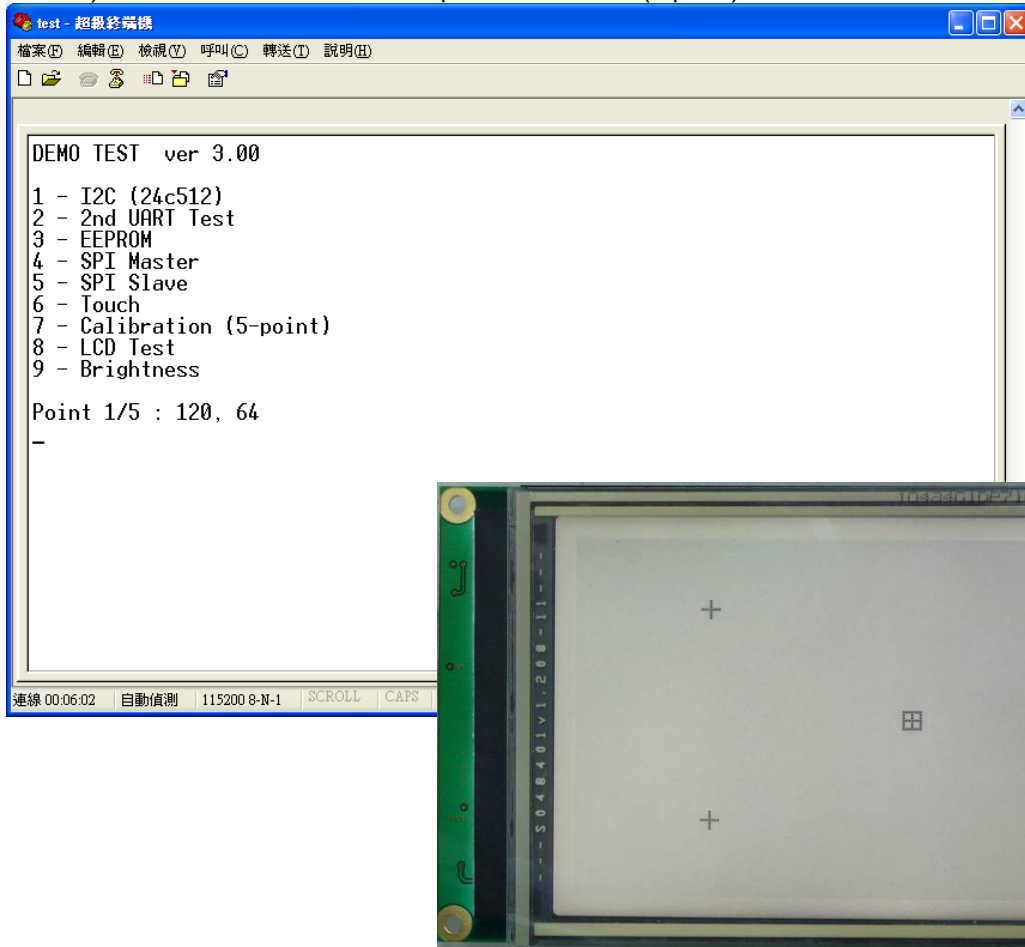
F). Enter number “6”: to read from touch panel






- 1). Using light pen to touch a point on touch panel
- 2). X, Y values are touch panel coordinates
- 3). DX, DY values are LCD coordinates (DX=0/DY=0 if touch panel is not calibrated.)
- 4). Enter number “0”: to return to main screen



G). Enter number "7": for touch panel calibration (5 point)

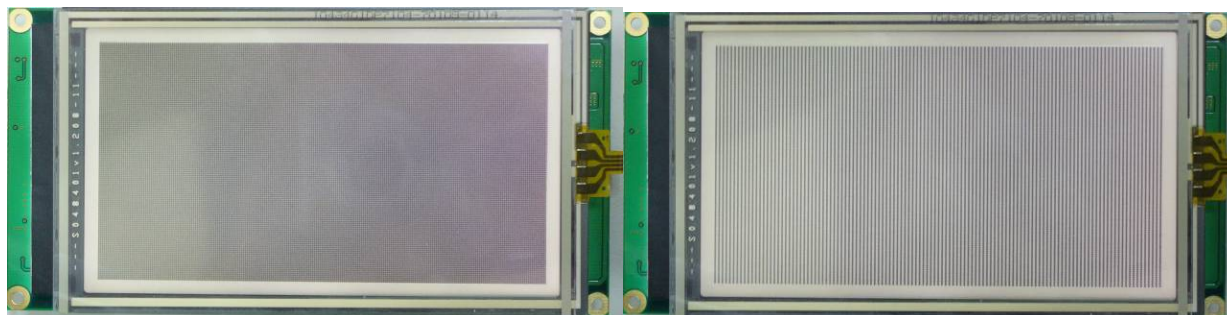


- 1). On the LCD, designer will see  displays. Please use light pen to touch center point of until it turns to  be . Such, one point is calibrated.
- 2). Please repeat above process to calibrate 5 points.
- 3). Touch panel calibration is finished.

H). Enter number “8”: for LCD function test

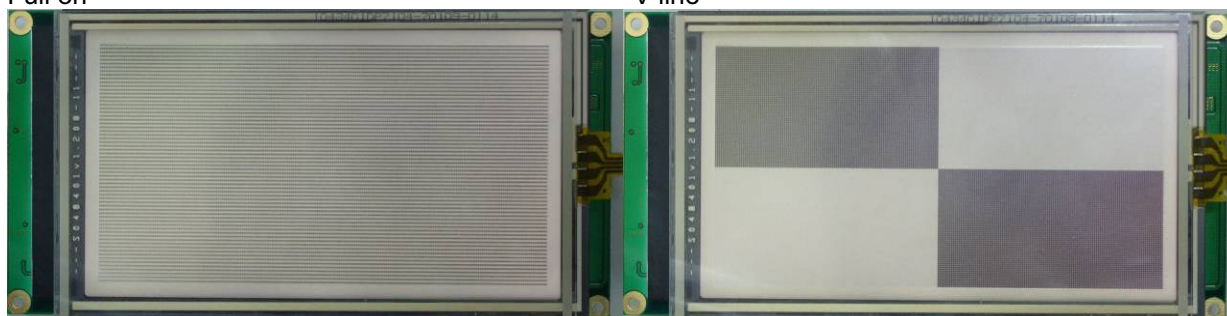


- 1). Enter number “1”: to LCD show full on
- 2). Enter number “2”: to LCD show vertical line
- 3). Enter number “3”: to LCD show horizontal line
- 4). Enter number “4”: to LCD show half
- 5). Enter number “5”: to LCD show cross dot
- 6). Enter number “6”: to LCD show character
- 7). Enter number “7”: to LCD show picture 1
- 8). Enter number “8”: to LCD show picture 2
- 9). Enter number “0”: to return to main screen



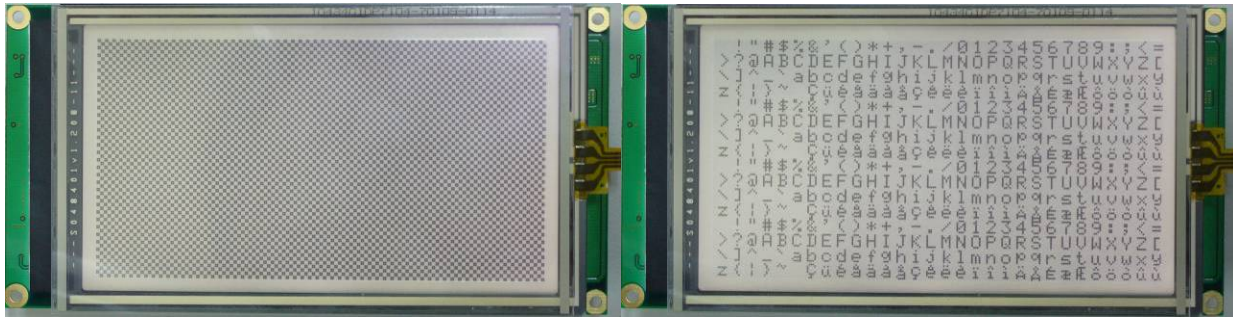
Full on

V line



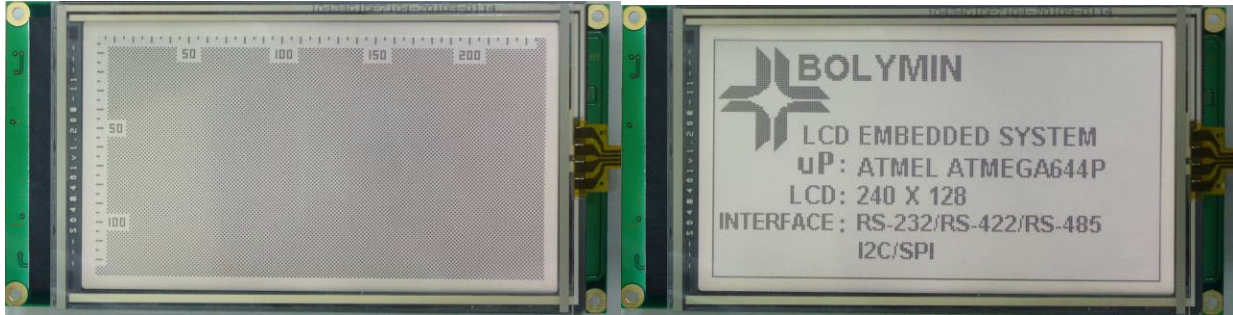
H line

Half



Cross dot

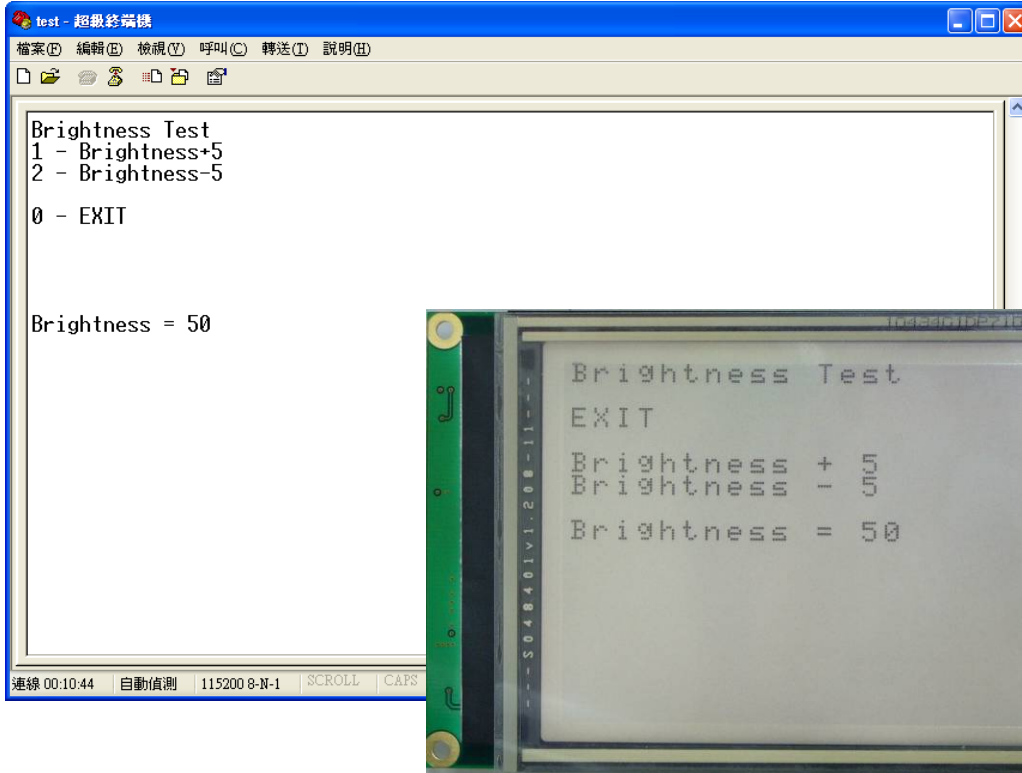
Character



Picture1

Picture2

I). Enter number "9": for backlight brightness adjustment function test



- 1). Enter number "1": to backlight brightness + 5
- 2). Enter number "2": to backlight brightness - 5
- 3). Enter number "0": to return to main screen

## 4-4 Bolymin Free Software Utilities

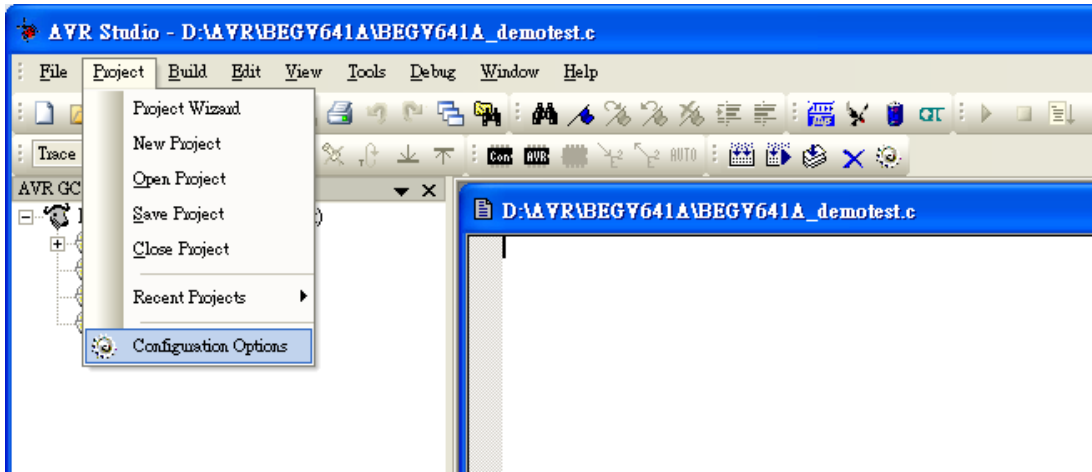
### 4-4-1 Website Links

- Touch Panel Driver  
[www.bolymin.com.tw/embedded/utility/BEGV641A/tpdriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/tpdriver.zip)
- UART Driver (RS-232/485/422)  
[www.bolymin.com.tw/embedded/utility/BEGV641A/uartdriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/uartdriver.zip)
- SPI Driver  
[www.bolymin.com.tw/embedded/utility/BEGV641A/spidriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/spidriver.zip)
- I<sup>2</sup>C Driver  
[www.bolymin.com.tw/embedded/utility/BEGV641A/i2cdriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/i2cdriver.zip)
- Backlight Driver  
[www.bolymin.com.tw/embedded/utility/BEGV641A/backlightdriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/backlightdriver.zip)
- LCD Driver  
[www.bolymin.com.tw/embedded/utility/BEGV641A/lcddriver.zip](http://www.bolymin.com.tw/embedded/utility/BEGV641A/lcddriver.zip)

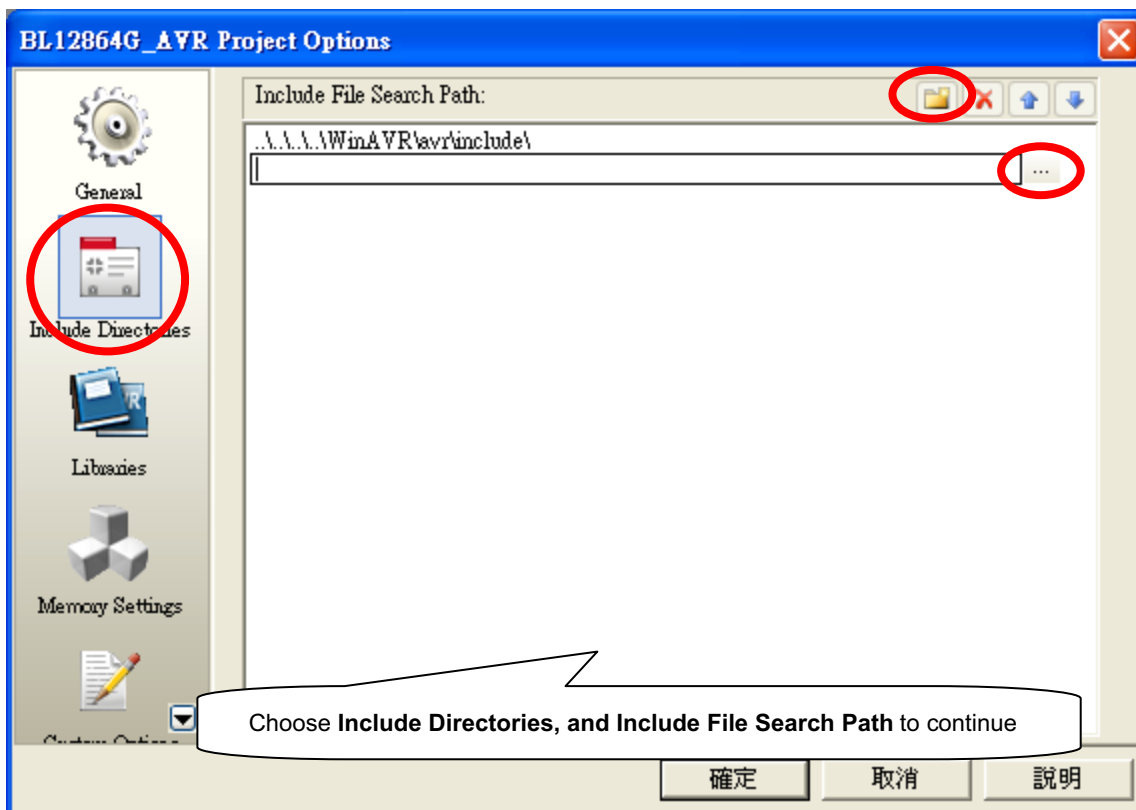
#### 4-4-2 Introduction of Bolymin software utilities

It is recommended to use Bolymin software utilities in order to speed up project development phase. However, designer may develop your own software utilities, if you find Bolymin utilities is not convenient to use.

On following paragraphs, we explain the way to add Bolymin software utilities into designer PC.



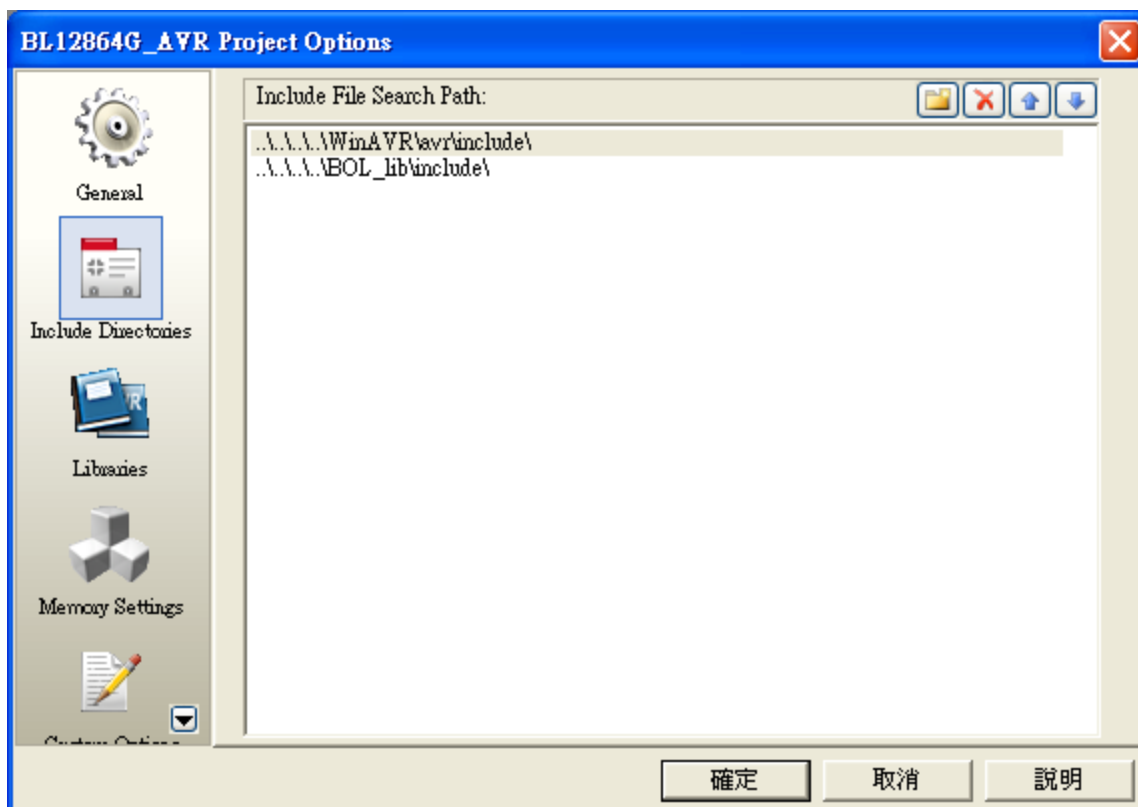
Note: Under AVR Studio4, Project→Configuration Options



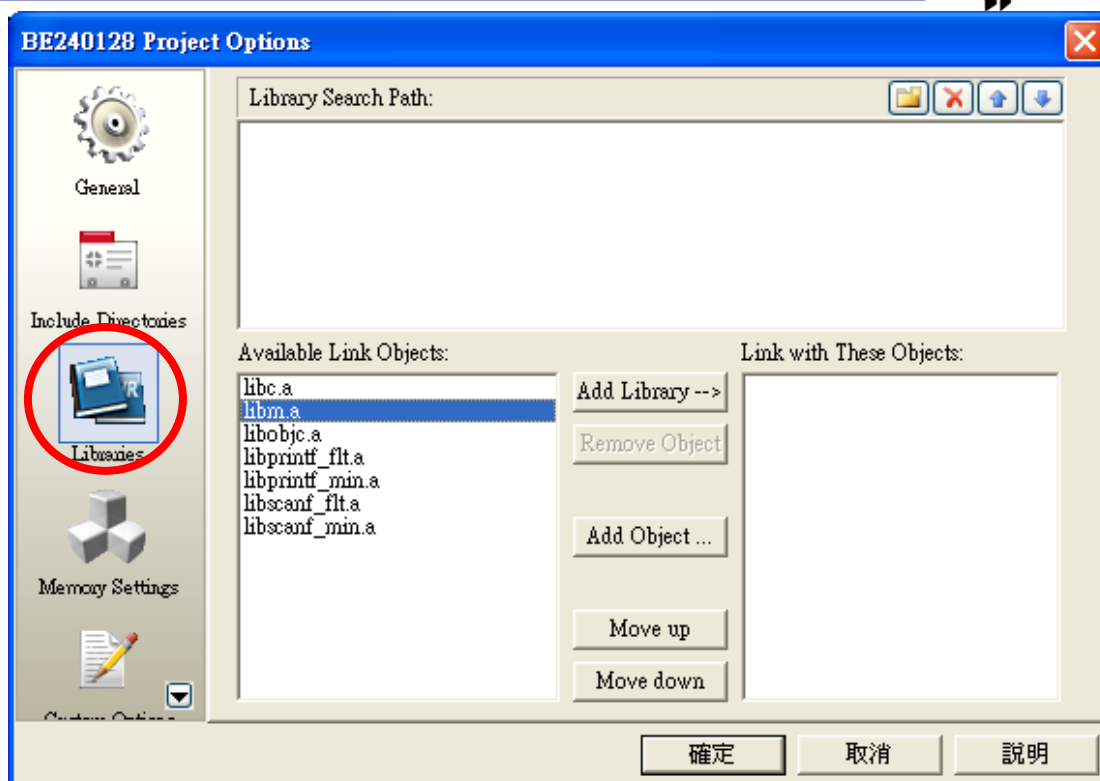
Note: Adding into **Header files**



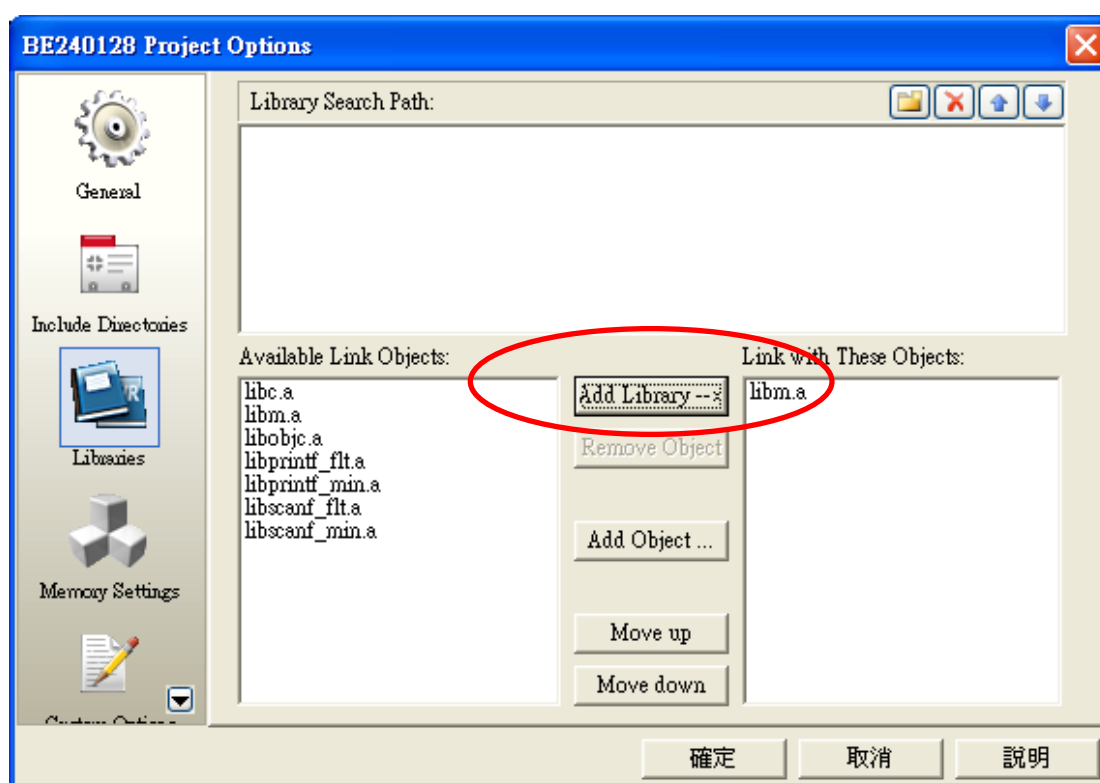
Choose **include** as File Search Path to continue



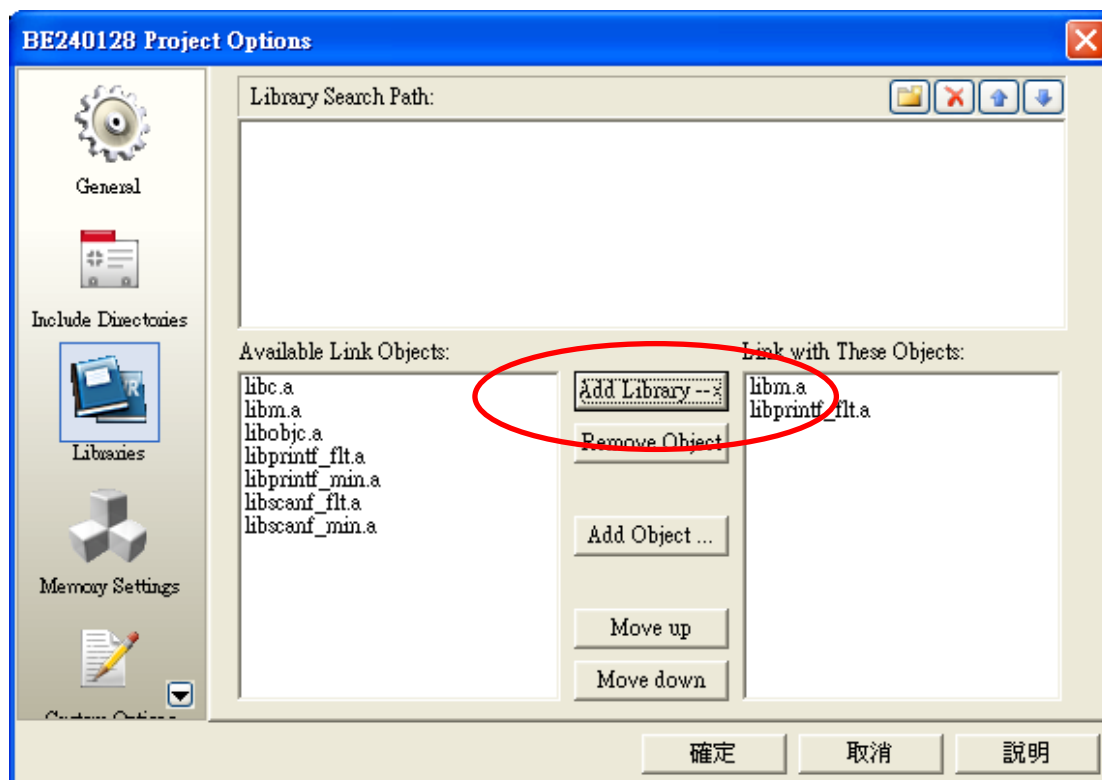
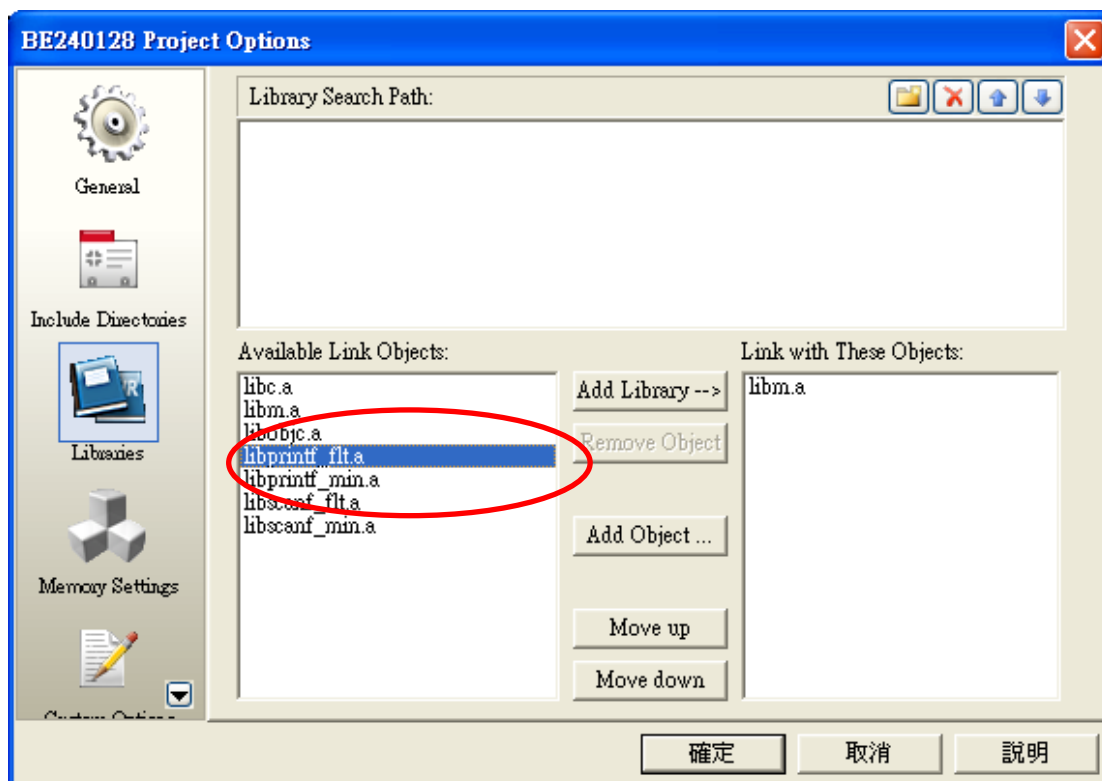


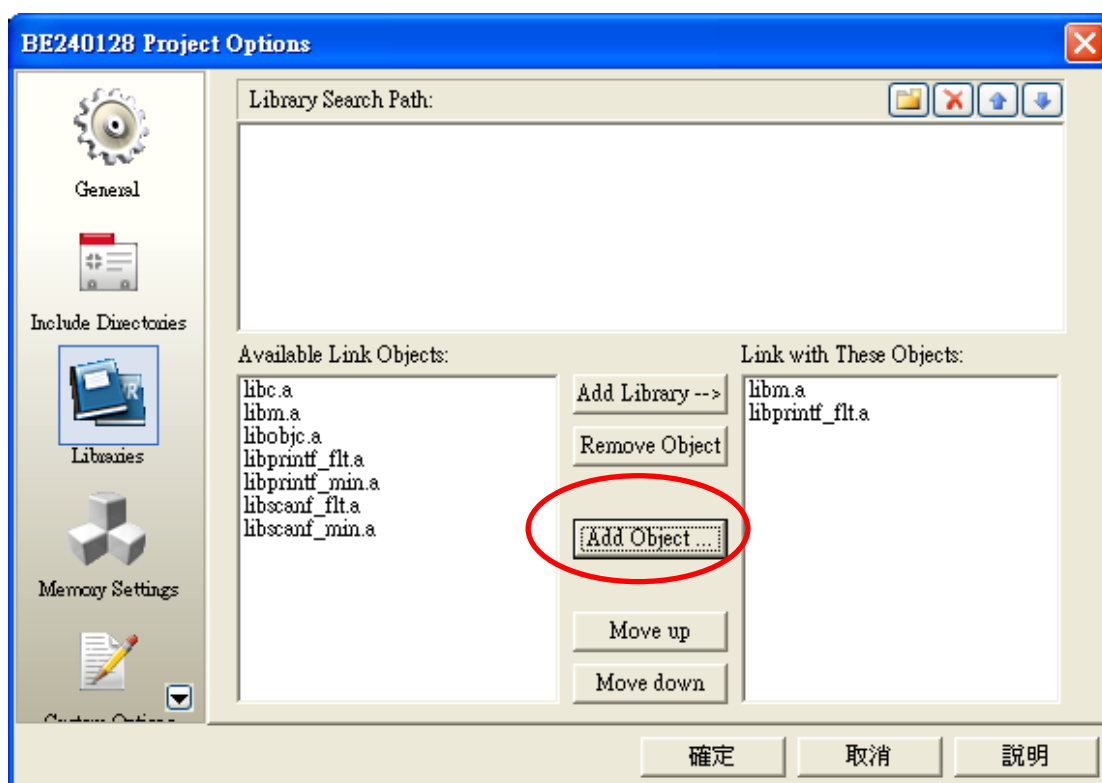


Note: For touch panel calibration, Floating-point operations is necessary, so please add libm.a and libprintf\_flt.a in WinAVR.

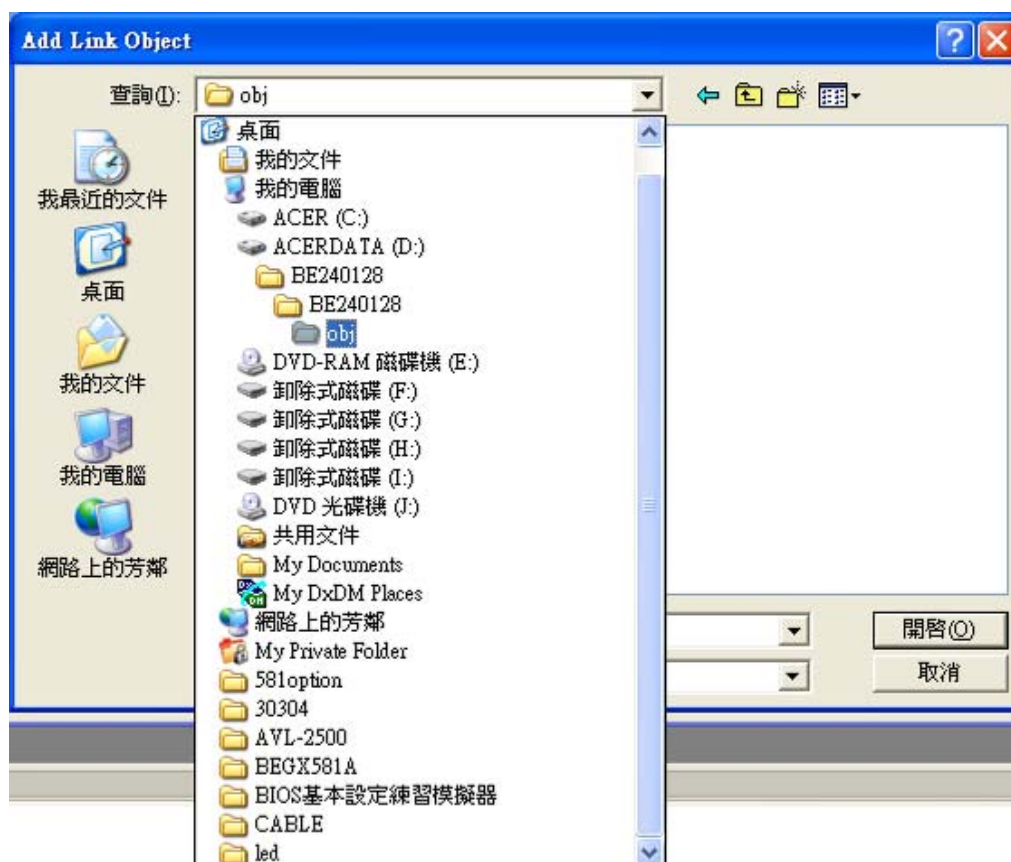


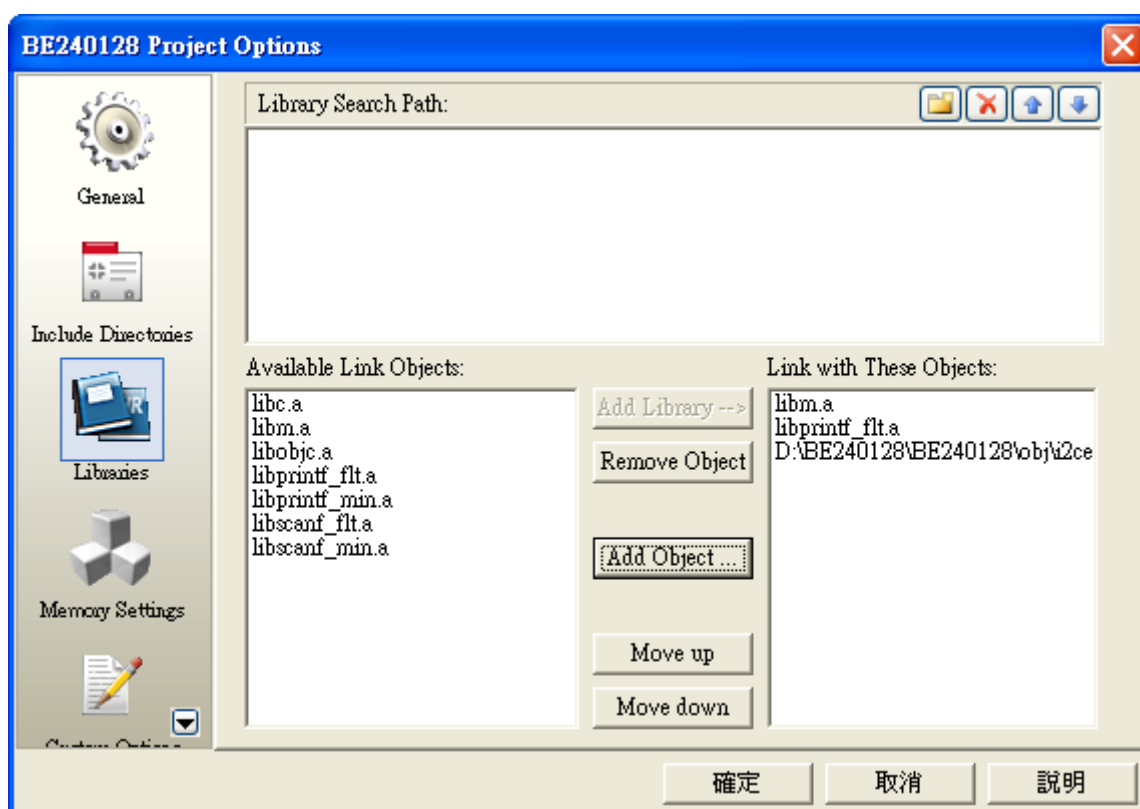
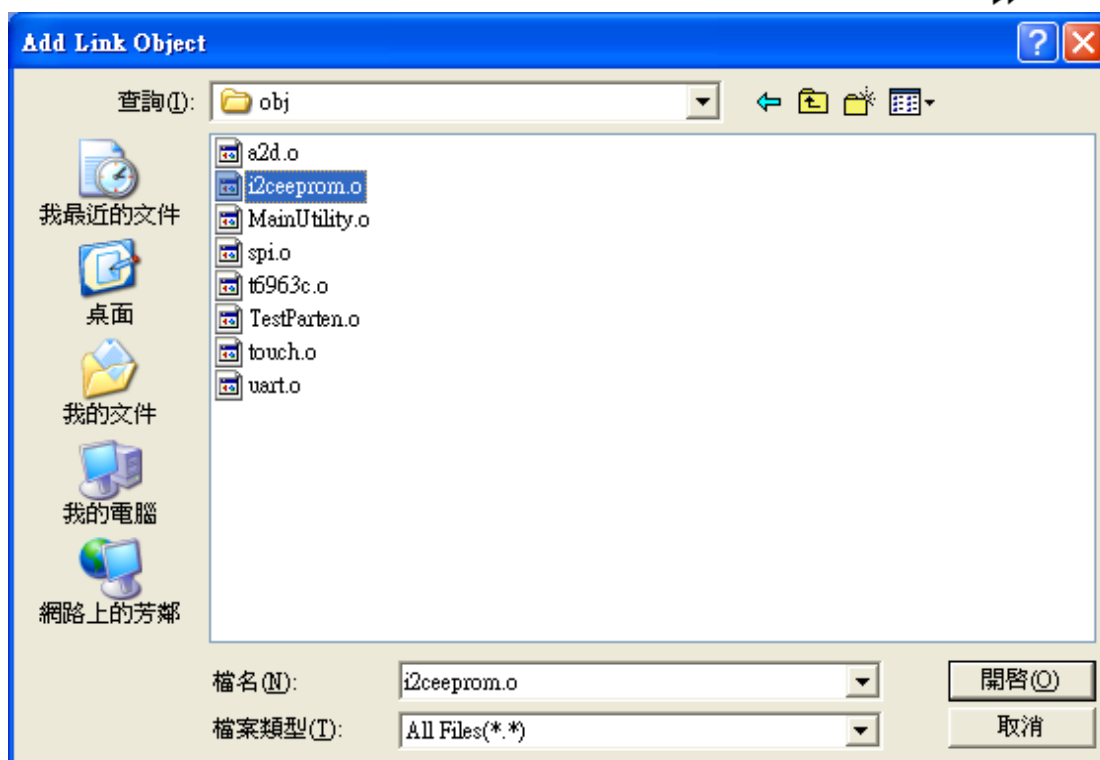


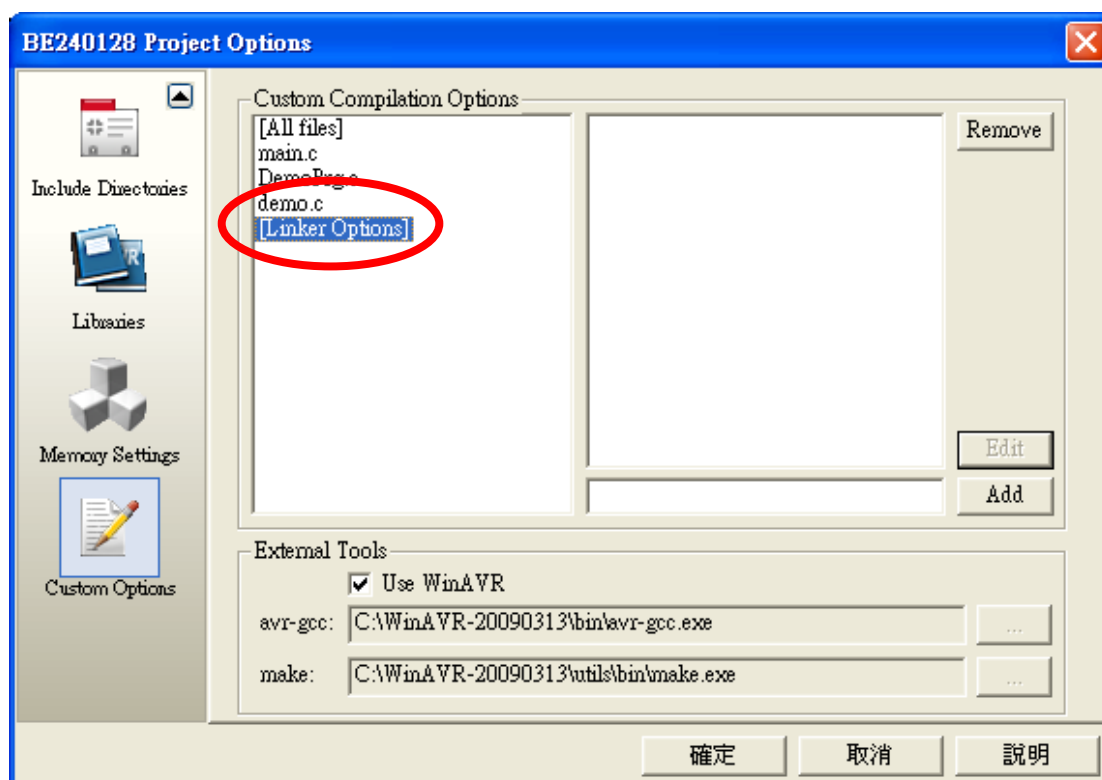
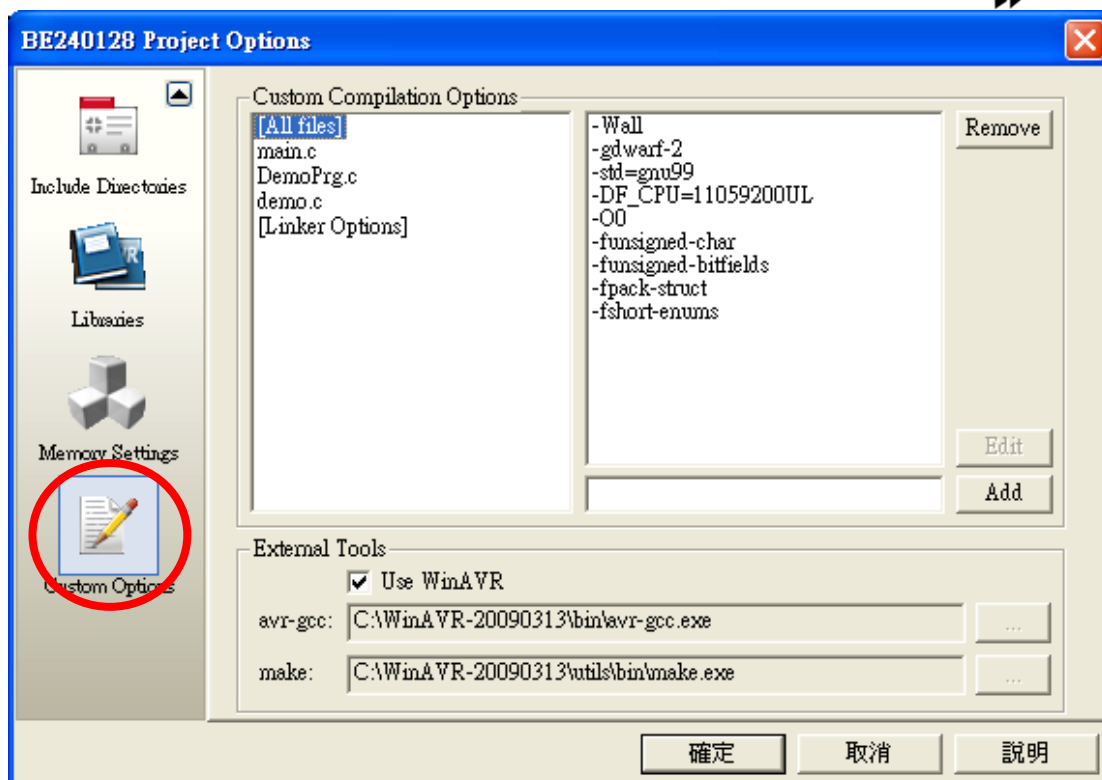




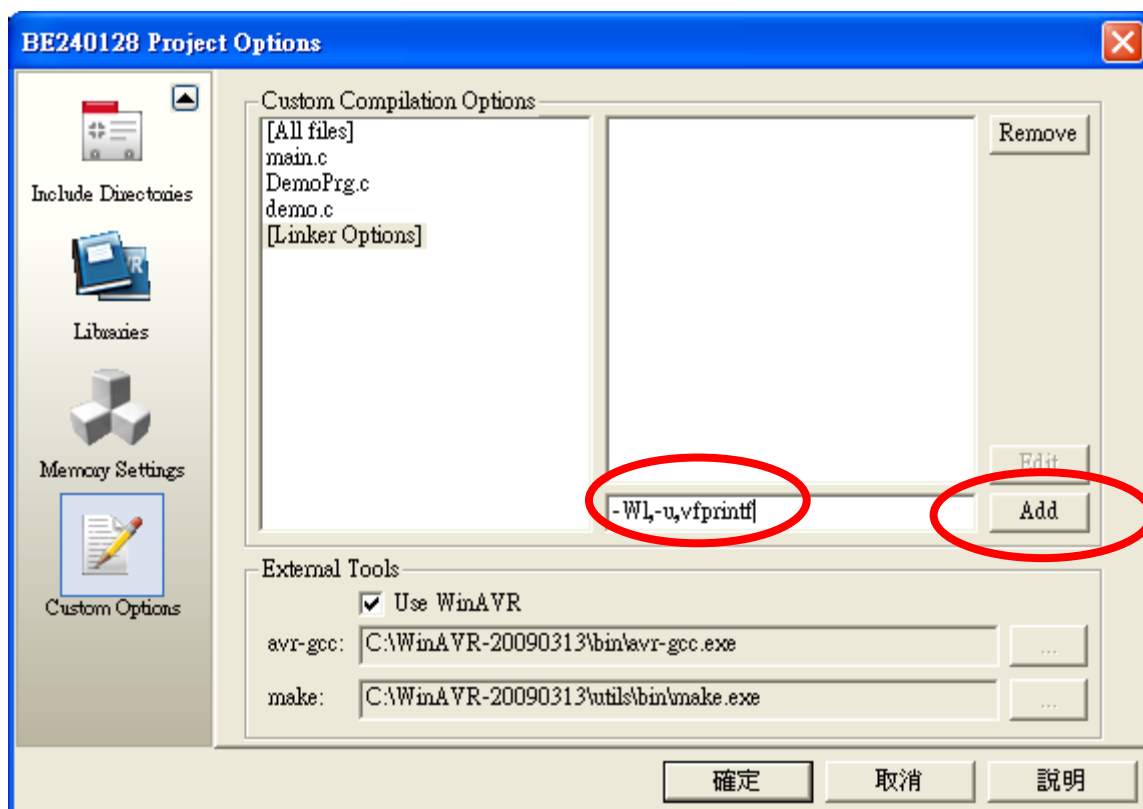
Note: Adding into Obj



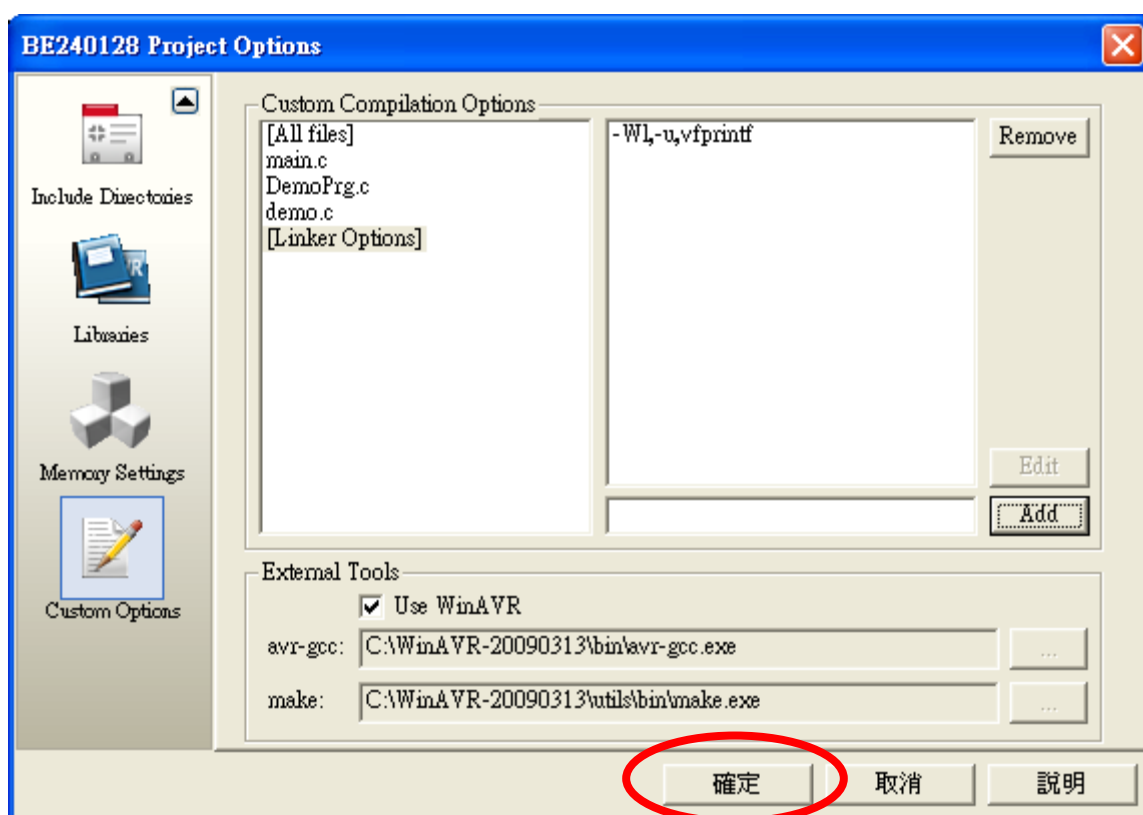




Note: choose [Linker Options]



Note: key in -Wl,-u,vfprintf, and then **Add**



### 4-4-3 Software Utilities Function Description

#### 4-4-3-1 UART function

Header file : uart.h

Object file : uart.o

#### uartInit Function: Initial UART.

<b>Syntax</b>	<pre>void uartInit(     uint8_t    byPort,     uint32_t   uBaudrate,     uint8_t    byParity,     uint8_t    uDatabit,     uint8_t    uStopbit,     uint8_t    nTxMode );</pre>	
<b>Parameters</b>	<div>byPort</div> <div>nBaudrat</div> <div>byParity</div> <div>uDatabit</div> <div>uStopbit</div> <div>nTxMode</div>	<div>UART_PORT0 – 1st Uart port</div> <div>UART_PORT1 – 2nd Uart port</div> <div>Baud rate, ex: 9600.</div> <div>Parity Check, 'N' – None, 'E' – EVEN, 'O' – ODD.</div> <div>data bit , 5 ~ 8.</div> <div>data bit , 1 ~ 2.</div> <div>Transmission Mode.</div> <div>0 or FALSE – RS232.</div> <div>1 or TRUE – RS485 or RS422.</div>
<b>Return value</b>	None.	

#### uartSetBaudRate Function: to set up baud rate for assigned UART port

<b>Syntax</b>	<pre>void uartSetBaudRate(     uint8_t    byPort,     uint32_t   uBaudrate );</pre>	
<b>Parameters</b>	<div>byPort</div> <div>nBaudrate</div>	<div>UART_PORT0 – 1st Uart port</div> <div>UART_PORT1 – 2nd Uart port</div> <div>Baud Rate, ex: 9600.</div>
<b>Return value</b>	None.	

#### uartSendByte Function: To send 1 byte from assigned UART port

<b>Syntax</b>	<pre>void uartSendByte(     uint8_t    byPort,     uint8_t    txData );</pre>	
<b>Parameters</b>	<div>byPort</div> <div>txData</div>	<div>UART_PORT0 – 1st Uart port</div> <div>UART_PORT1 – 2nd Uart port</div> <div>byte to be sent.</div>
<b>Return value</b>	None.	

**uartDisablePort Function:** to stop operation of assigned Uart port

<b>Syntax</b>	void uartDisablePort ( uint8_t byPort );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
<b>Return value</b>	None.

**uartSendString Function:** to send 1 string from assigned UART port

<b>Syntax</b>	void uartSendString( uint8_t     byPort uint8_t*    str );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port str                Index of string to be sent, ending with "0".
<b>Return value</b>	None.

**uartSendBuffer Function:** to send buffer from assigned UART port

<b>Syntax</b>	void uartSendBuffer( uint8_t     byPort, uint8_t*    buffer, uint16_t    nBytes );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port buffer            index of buffer to be sent nBytes            bytes of buffer to be sent
<b>Return value</b>	None.

**uartReceiveByte Function:** to read 1 byte data from assigned UART port

<b>Syntax</b>	uint8_t uartReceiveByte( uint8_t     byPort, uint8_t*    rxData );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port rxData            index of data to be received
<b>Return value</b>	TRUE – rxData is true data FALSE – UART port no data

**uartReceiveBufferIsEmpty Function:** to check if there is data in assigned UART port

<b>Syntax</b>	void uartReceiveBufferIsEmpty( uint8_t     byPort );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
<b>Return value</b>	TRUE – There is data on assigned UART port FALSE – There is no data on assigned UART port

**uartFlushReceiveBuffer Function:** to clear receiving buffer on assigned UART port

<b>Syntax</b>	void uartFlushReceiveBuffer( uint8_t    byPort );
<b>Parameters</b>	byPort            UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
<b>Return value</b>	None.

**uartEnableTx Function:** to Enable or Disable UART transmitter. (When UART port is applied on RS485 or RS422 , transmitter must be set to Disable, and to Enable transmitter only when sending data.)

<b>Syntax</b>	void uartEnableTx( uint8_t    bEnable );
<b>Parameters</b>	bEnable           TRUE – Enable transmitter FALSE – Disable transmitter.
<b>Return value</b>	None.



#### 4-4-3-2 I<sup>2</sup>C function

Header file : i2ceeprom.h

object file : i2ceeprom.o

**i2cInitial Function:** Initial I2C functions. User should call this function before using I2C functions.

<b>Syntax</b>	void i2cInitial( );
<b>Parameters</b>	None.
<b>Return value</b>	None.

**i2cReadByte Function:** to read 1 byte data from I<sup>2</sup>C

<b>Syntax</b>	uint8_t i2cReadByte( uint8_t uDevAddr, uint16_t nAddr );
<b>Parameters</b>	uDevAddr I <sup>2</sup> C device address. (address of three 24c512 on board are: A2 <sub>hex</sub> , A4 <sub>hex</sub> , A6 <sub>hex</sub> . nAddr address to write in .
<b>Return value</b>	Data reading from I <sup>2</sup> C

**i2cWriteByte Function:** to write 1 byte data from I<sup>2</sup>C

<b>Syntax</b>	void i2cWriteByte( uint8_t uDevAddr, uint16_t nAddr, uint8_t byData );
<b>Parameters</b>	uDevAddr I <sup>2</sup> C device address.(address of three 24c512 on board are: A2 <sub>hex</sub> , A4 <sub>hex</sub> , A6 <sub>hex</sub> . nAddr address to write in . byData data to write in .
<b>Return value</b>	None.

**i2cSetSpeed Function:** to set baud rate for I2C

<b>Syntax</b>	void i2cSetSpeed( uint16_t nSpeed, );
<b>Parameters</b>	uSpeed I <sup>2</sup> C baud rate , 0 - 100K, 1 - 250K.
<b>Return value</b>	None.

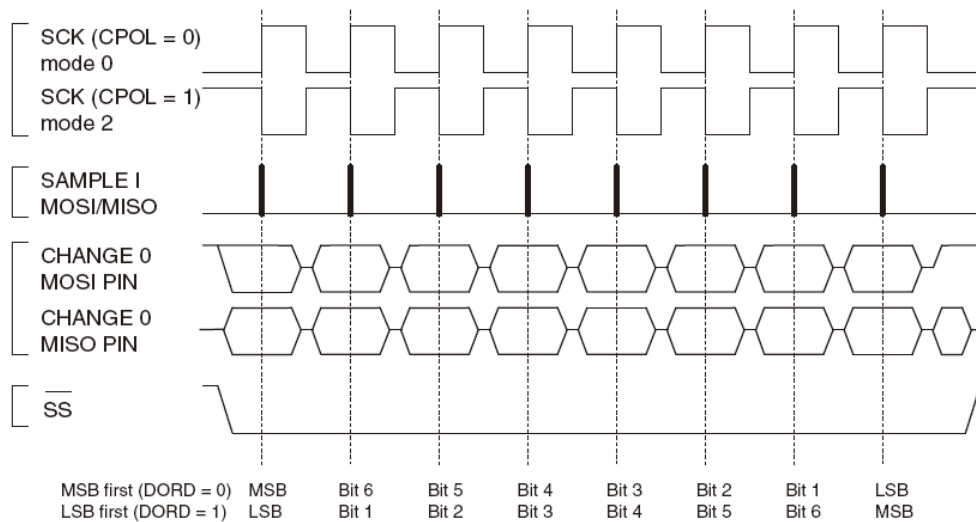
### 4-3-3-3 SPI function

Header file : spi.h  
object file : spi.o

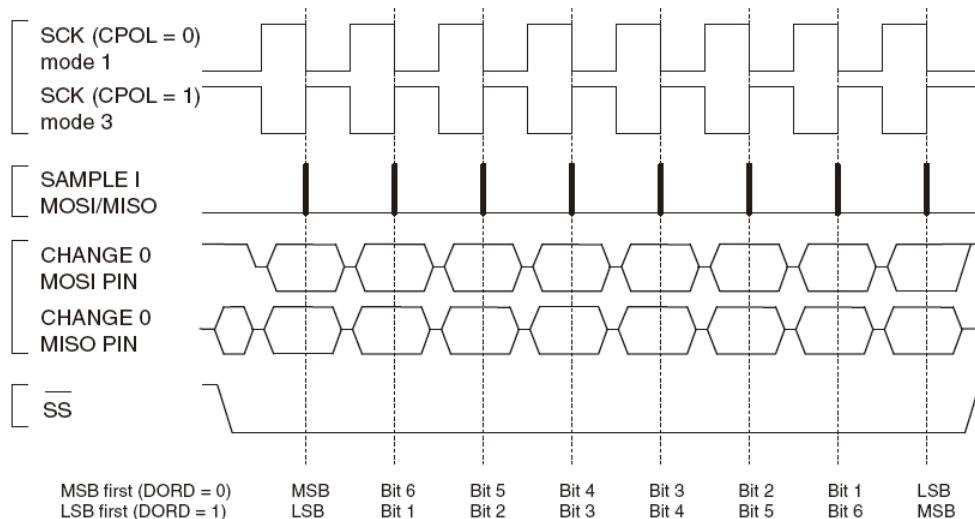
**spiMaster Function:** Initial SPI to master mode.

<b>Syntax</b>	void spiMaster( uint8_t mode );
<b>Parameters</b>	mode      SPI mode. 0 – CPOL=0, CPHA=0 Sample (Rising) Setup (Falling) 1 – CPOL=0, CPHA=1 Setup (Rising) Sample (Falling) 2 – CPOL=1, CPHA=0 Sample (Falling) Setup (Rising) 3 – CPOL=1, CPHA=1 Setup (Falling) Sample (Rising)
<b>Return value</b>	None.

#### SPI Transfer Format with CPHA = 0



#### SPI Transfer Format with CPHA = 1



**spiSlave Function:** Initial SPI to slave mode.

<b>Syntax</b>	void spiSlave( uint8_t mode, );
<b>Parameters</b>	mode           SPI mode. 0 – CPOL=0, CPHA=0 Sample (Rising) Setup (Falling) 1 – CPOL=0, CPHA=1 Setup (Rising) Sample (Falling) 2 – CPOL=1, CPHA=0 Sample (Falling) Setup (Rising) 3 – CPOL=1, CPHA=1 Setup (Falling) Sample (Rising)
<b>Return value</b>	None.

**spiSetBtrate Function:** to set SPI baud rate

<b>Syntax</b>	void spiSetBtrate( uint8_t   spr );
<b>Parameters</b>	spr            SPI baud rate. system OSC is 11.0592MHz. 0 – OSC / 4. 1 – OSC / 16. 2 – OSC / 64. 3 – OSC / 128.
<b>Return value</b>	None.

**spiSetDataOrder Function:** to set SPI order when sending data, LSB first or MSB first.

<b>Syntax</b>	void spiSetDataOrder( uint8_t order );
<b>Parameters</b>	order           LSB first or MSB first. DATA_LSB_FIRST–LSB of the data word will be transmitted first. DATA_MSB_FIRST–MSB of the data word will be transmitted first.(Default)
<b>Return value</b>	None.

**spiSendByte Function:** to send 1 byte data from SPI port

<b>Syntax</b>	void spiSendByte( uint8_t   data );
<b>Parameters</b>	data           byte to be sent
<b>Return value</b>	None.

**spiRecvByte Function:** to receive 1 byte data from SPI port

<b>Syntax</b>	uint8_t spiRecvByte( );
<b>Parameters</b>	None.
<b>Return value</b>	To read 1 byte data from SPI

**spiTransferByte Function:** to send and read 1 byte data from SPI port

<b>Syntax</b>	uint8_t spiTransferByte( uint8_t data );
<b>Parameters</b>	data                      byte to be sent
<b>Return value</b>	1 byte data reading from SPI

**spiTransferWord Function:** to send and read 1 word from SPI port

<b>Syntax</b>	uint16_t spiTransferByte( uint16_t data );
<b>Parameters</b>	data                      word to be sent
<b>Return value</b>	1 word data reading from SPI

#### 4-3-3-4 E<sup>2</sup>PROM function

E<sup>2</sup>PROM Function is built inside WinAVR, so users need only to include eeprom.h in program to call E<sup>2</sup>PROM Function.

For exmaple :

```
#include <avr/eeprom.h>
```

**eeprom\_write\_byte Function:** to write 1 byte to MCU E<sup>2</sup>PROM.

<b>Syntax</b>	void eeprom_write_byte ( uint8_t* address, uint8_t value );
<b>Parameters</b>	address      E <sup>2</sup> PROM address to write in, range 0 ~ 0x7FF value        E <sup>2</sup> PROM data to write in
<b>Return value</b>	None.

**eeprom\_read\_byte Function:** This function read one byte from EEPROM.

<b>Syntax</b>	uint8_t eeprom_read_byte ( uint8_t* address );
<b>Parameters</b>	address      E <sup>2</sup> PROM address to read, range 0 ~ 0x7FF
<b>Return value</b>	Data reading from E <sup>2</sup> PROM

**eeprom\_write\_block Function:** to write block data to E2PROM.

<b>Syntax</b>	void eeprom_write_block ( void*            pointer_ram, const void*    pointer_eeprom, size_t        n );
<b>Parameters</b>	pointer_ram    index of block data to write in . pointer_eeprom E <sup>2</sup> PROM address to write in, range 0 ~ 0x7FF. n              length of E <sup>2</sup> PROM data to write in
<b>Return value</b>	None.

**eeprom\_read\_block Function:** to read block data from E2PROM

<b>Syntax</b>	uint8_t eeprom_read_block ( void*            pointer_ram, const void*    pointer_eeprom, size_t        n );
<b>Parameters</b>	pointer_ram    index of block data to read . pointer_eeprom address of E <sup>2</sup> PROM to read, rnage 0 ~ 0x7FF. n              length of E <sup>2</sup> PROM data to read
<b>Return value</b>	Data block read to .

#### 4-3-3-5 Touch function

Header file : touch.h, a2d.h

object file : touch.o, a2d.o

**touchInit Function:** Initial Touch panel.

<b>Syntax</b>	void touchInit( );
<b>Parameters</b>	None.
<b>Return value</b>	None.

**touchGet Function:** to read touch data from touch panel

<b>Syntax</b>	uint8_t touchGet( int * pX, int * pY );
<b>Parameters</b>	pX to read X Coordinate from touch data pY to read Y Coordinate from touch data
<b>Return value</b>	TRUE data of pX and pY is true FALSE data of pX and pY is false

**touchDrawCalPoint Function:** to draw Calibration cross Coordinate on LCD

<b>Syntax</b>	uint8_t touchDrawCalPoint ( POINT* ptCal, int n );
<b>Parameters</b>	ptCal Calibration Coordinate. n Calibration Coordinate No.
<b>Return value</b>	None.

**setCalibrationMatrix Function:** to set Calibration calculation matrix

<b>Syntax</b>	void setCalibrationMatrix( POINT * ptDisplay, POINT * ptTouch, int n );
<b>Parameters</b>	ptDisplay LCD reference Coordinate for calibration. ptTouch Touch Coordinate for calibration n Coordinate No. for calibration
<b>Return value</b>	None.

**getDisplayPoint Function:** to change Touch Coordinate into LCD

<b>Syntax</b>	void getDisplayPoint( int x, int y, int * pX, int * pY );
<b>Parameters</b>	x Touch X Coordinate. y Touch Y Coordinate. pX LCD X Coordinate changed from Touch X Coordinate pY LCD Y Coordinate changed from Touch Y Coordinate
<b>Return value</b>	None.

#### 4-3-3-6 LCD control function

Header file : t6963c.h

object file : t6963c.o

**LcdInit Function:** Initialize all parameters of LCD display. User should call this function before use functions of LCD display.

<b>Syntax</b>	<code>void LcdInit (</code> <code>);</code>
<b>Parameters</b>	None.
<b>Return value</b>	None.

**LcdDisplayClr Function:** Clear screen (include graphic and text layer).

<b>Syntax</b>	<code>LcdDisplayClr(</code> <code>);</code>
<b>Parameters</b>	None.
<b>Return value</b>	None.

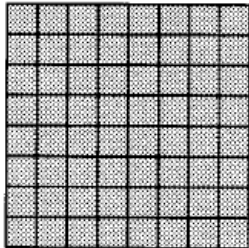
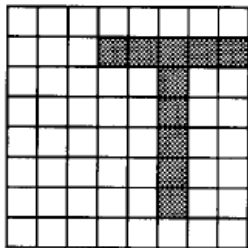
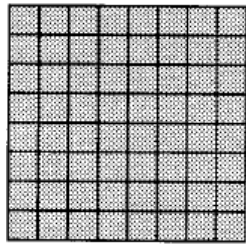
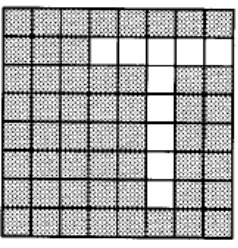
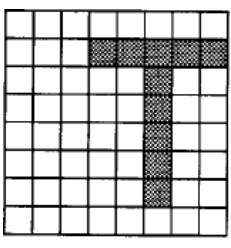
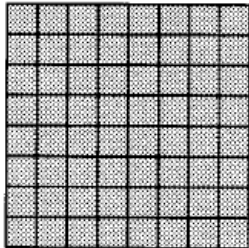
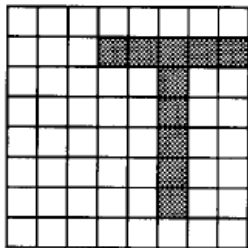
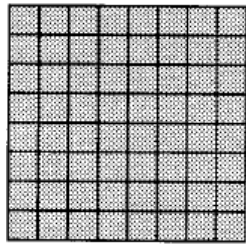
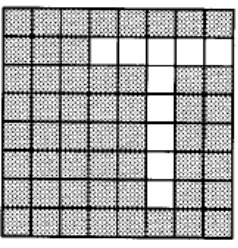
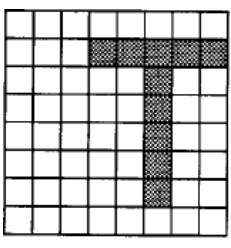
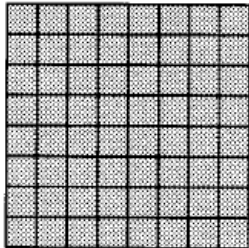
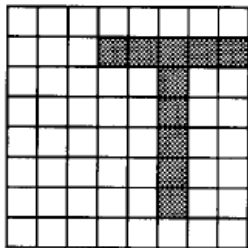
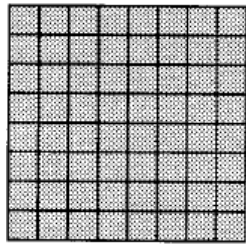
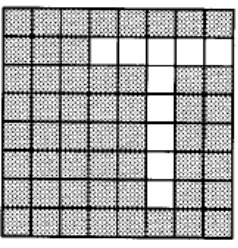
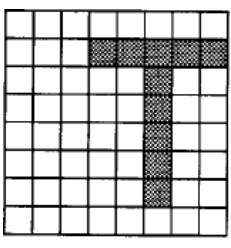
**LcdSetCursorPos Function:** Set the position of cursor. (UNIT=character=8\*8 pixel)

<b>Syntax</b>	<code>void LcdSetCursorPos(</code> <code>uint8_t x,</code> <code>uint8_t y</code> <code>);</code>
<b>Parameters</b>	x X coordinate of cursor y Y coordinate of cursor
<b>Return value</b>	None.

**LcdSwitchDisplay Function:** ON/OFF the display of cursor, text and graphic layer.

<b>Syntax</b>	<code>void LcdSwitchDisplay (</code> <code>uint8_t display_switch</code> <code>);</code>
<b>Parameters</b>	display_switch Display switch flag DS_DISPLAY_OFF Turn off all display. DS_TEXT_ON Turn ON the display of text layer. DS_GRAPHIC_ON Turn ON the display of graphic layer. DS_CURSOR_ON Turn ON the display of cursor. DS_BLINK_ON Turn ON cursor blink. <b>This flag is available while cursor is ON.</b>
<b>Return value</b>	None.
<b>Example</b>	<code>// Turn OFF all display.</code> <code>LcdSwitchDisplay(DS_DISPLAY_OFF);</code>  <code>// .Turn ON the display of text and graphic layer. Turn OFF cursor.</code> <code>LcdSwitchDisplay(DS_TEXT_ON  DS_GRAPHIC_ON);</code> <code>// .Turn ON the display of text layer and cursor without blink. Turn OFF graphic layer.</code> <code>LcdSwitchDisplay(DS_TEXT_ON  DS_CURSOR_ON);</code>  <code>// .Turn ON the display of text layer, graphic layer and cursor with blink.</code> <code>LcdSwitchDisplay(DS_TEXT_ON  DS_GRAPHIC_ON  DS_CURSOR_ON </code> <code>DS_BLINK_ON);</code>

**LcdSetDispMode Function:** Set the display mode between graphic layer and text layer.

Syntax	<pre>void LcdSetDispMode (     uint8_t    new_mode );</pre>																
Parameters	<div>New_mode<div>New display mode between graphic layer and text layer. Value : DM_OR_MODE, DM_XOR_MODE, DM_AND_MODE</div></div> <table><tr><th colspan="2">Content on Graphic layer</th><th colspan="2">Content on Text layer</th></tr><tr><td colspan="2"></td><td colspan="2"></td></tr><tr><th>DM_OR_MODE</th><th>DM_XOR_MODE</th><th colspan="2">DM_AND_MODE</th></tr><tr><td></td><td></td><td colspan="2"></td></tr></table>	Content on Graphic layer		Content on Text layer						DM_OR_MODE	DM_XOR_MODE	DM_AND_MODE					
Content on Graphic layer		Content on Text layer															
																	
DM_OR_MODE	DM_XOR_MODE	DM_AND_MODE															
																	
Return value	None																

**LcdDraw Function:** Draw input binary picture on specified area of graphic area.

<b>Syntax</b>	<pre>void LcdDraw (     uint8_t    x_start,     uint8_t    y_start,     uint8_t    x_end,     uint8_t    y_end,     uint8_t*   pic_data,     uint8_t    mode );</pre>
<b>Parameters</b>	<p>x_start      X coordinate of the top-left point of input picture. (UNIT=pixel)</p> <p>y_start      Y coordinate of the top-left point of input picture. (UNIT=pixel)</p> <p>x_end        X coordinate of the bottom-right point of input picture. (UNIT=pixel)</p> <p>y_end        Y coordinate of the bottom-right point of input picture. (UNIT=pixel)</p> <p>pic_data     Bit map data will be drawn. Input 0 will reverse pixels of specified area.</p> <p>mode         DRAW_NORMAL: Draw the picture normally. DRAW_REVERSE : Reverse the picture and then draw the picture.</p>
<b>Return value</b>	None.



**LcdFillByte Function:** Fill input byte value on specified area of graphic layer.

<b>Syntax</b>	<pre>void LcdFillByte (                 uint8_t    x_start,                 uint8_t    y_start,                 uint8_t    x_end,                 uint8_t    y_end,                 uint8_t    data,                 uint8_t    mode                 );</pre>	
<b>Parameters</b>	<div> <div>x_start</div> <div>Y coordinate of the top-left point of specified area. (UNIT=pixel)</div> </div> <div> <div>y_start</div> <div>X coordinate of the top-left point of specified area. (UNIT=pixel)</div> </div> <div> <div>x_end</div> <div>X coordinate of the bottom-right point of specified area. (UNIT=pixel)</div> </div> <div> <div>y_end</div> <div>Y coordinate of the bottom-right point of specified area. (UNIT=pixel)</div> </div> <div> <div>pic_data</div> <div>Byte value will be filled.</div> </div> <div> <div>Mode</div> <div>DRAW_NORMAL: Fill input value normally. DRAW_REVERSE : Reverse the input value and then fill it on the specified area</div> </div>	
<b>Return value</b>	None.	

**LcdPrintString Function:** Print input string to specified location of text layer.

<b>Syntax</b>	<pre>void LcdPrintString (                 uint8_t    x_start,                 uint8_t    y_start,                 char *      string,                 uint8_t    str_count                 );</pre>	
<b>Parameters</b>	<div> <div>x_start</div> <div>X coordinate of start location that input string will be printed. (UNIT= character=8*8 pixel)</div> </div> <div> <div>y_start</div> <div>Y coordinate of start location that input string will be printed. (UNIT= character=8*8 pixel)</div> </div> <div> <div>string</div> <div>string will be printed to LCD</div> </div> <div> <div>str_count</div> <div>character count of input string</div> </div>	
<b>Return value</b>	None.	

**LcdDrawBit Function:** ON/OFF the pixel on specified location of graphic layer.

<b>Syntax</b>	<pre>void LcdDrawBit (                 uint8_t    x,                 uint8_t    y,                 char        bit_value                 );</pre>	
<b>Parameters</b>	<div> <div>x</div> <div>X coordinate of the location will be drawn. (UNIT=pixel)</div> </div> <div> <div>y</div> <div>Y coordinate of the location will be drawn. (UNIT=pixel)</div> </div> <div> <div>bit_value</div> <div>1 : ON the pixel 0 : OFF the pixel</div> </div>	
<b>Return value</b>	None.	

**LcdDrawRect Function:** Draw rectangle by single line on graphic layer.

<b>Syntax</b>	<pre>void LcdDrawRect (                 uint8_t  x_start,                 uint8_t  y_start,                 uint8_t  x_end,                 uint8_t  y_end,                 );</pre>	
<b>Parameters</b>	<div>x_start</div> <div>y_start</div> <div>x_end</div> <div>y_end</div>	<div>X coordinate of the top-left point of rectangle. (UNIT=pixel)</div> <div>Y coordinate of the top-left point of rectangle. (UNIT=pixel)</div> <div>X coordinate of the bottom-right point of rectangle. (UNIT=pixel)</div> <div>Y coordinate of the bottom-right point of rectangle. (UNIT=pixel)</div>
<b>Return value</b>	None	

#### 4-3-3-7 Backlight PWM control function

Header file : bklight\_pwm.h

object file : bklight\_pwm.o

**Note:** Backlight PWM control used **TIMER0** and **INTERRUPT0**

**bklPWM\_Init Function:** Initialize all parameters of backlight PWM control function. User should call this function before use backlight PWM control functions.

<b>Syntax</b>	void bklPWM_Init ( );
<b>Parameters</b>	None.
<b>Return value</b>	None.

**bklSetBrightness Function:** Set current brightness value of backlight.

<b>Syntax</b>	void bklSetBrightness ( uint8_t    brightness );
<b>Parameters</b>	Brightness    New brightness value 0            – OFF backlight 1~100       – Control the brightness of backlight
<b>Return value</b>	None.

**bklGetBrightness Function:** Get current brightness value of backlight

<b>Syntax</b>	uint8_t bklGetBrightness ( );
<b>Parameters</b>	None.
<b>Return value</b>	Current brightness value of backlight. (0 ~ 100)

## **Appendix A: LCD Controller Specification**

## **Appendix B: EEPROM Specification**

## **Appendix C: ATMEL ATmega644p MCU Specification**

Please download this specification from following ATMEL link:

[http://www.atmel.com/dyn/resources/prod\\_documents/doc8011.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc8011.pdf)

**<END of BEGV641A User Manual>**