Leader of Microcontroller Technology
Global Top Smart MCU Innovation Company

CodeGen8 (8-Bit MCU Code Generator)

User Manual

Version 1.0.0
Jun. 28, 2019

For additional information or inquiry, please contact ABOV Semiconductor or visit its website at www.abov.co.kr.

Revision History

Date	Version	Description
Jun. 28, 2019	1.0.0	Document created.

Contents

Chapter 1. Getting started

1.1 System requirements

This section describes the hardware and software system requirements.

1.1.1 Software requirements

This software is compatible with the following operating systems. Both 32-bit and 64-bit versions are available.

- Microsoft Windows NT
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7
- Microsoft Windows 8 & 8.1
- Microsoft Windows 10

Disk space

For the full installation of the software, up to 20 MB of hard disk space is required.

1.1.2 Hardware requirements

This software can be run on a basic PC and does not require powerful specifications. The following are minimum hardware requirements for installing and running this software.

- Pentium PC

Performance is based on the following factor:

- Processor performance

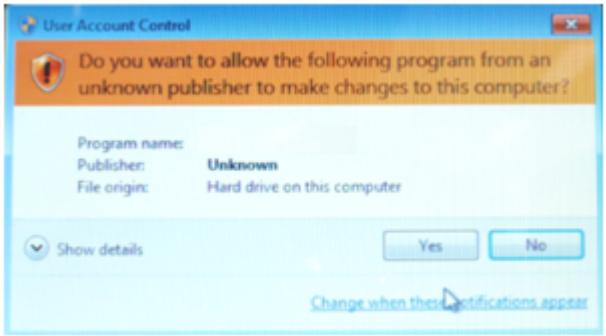
1.2 Software installation

The installer can be downloaded from the ABOV website. You should ideally have the latest version of the software because ABOV constantly adds newer devices and features. The installer package can be executed on any Microsoft operating system.

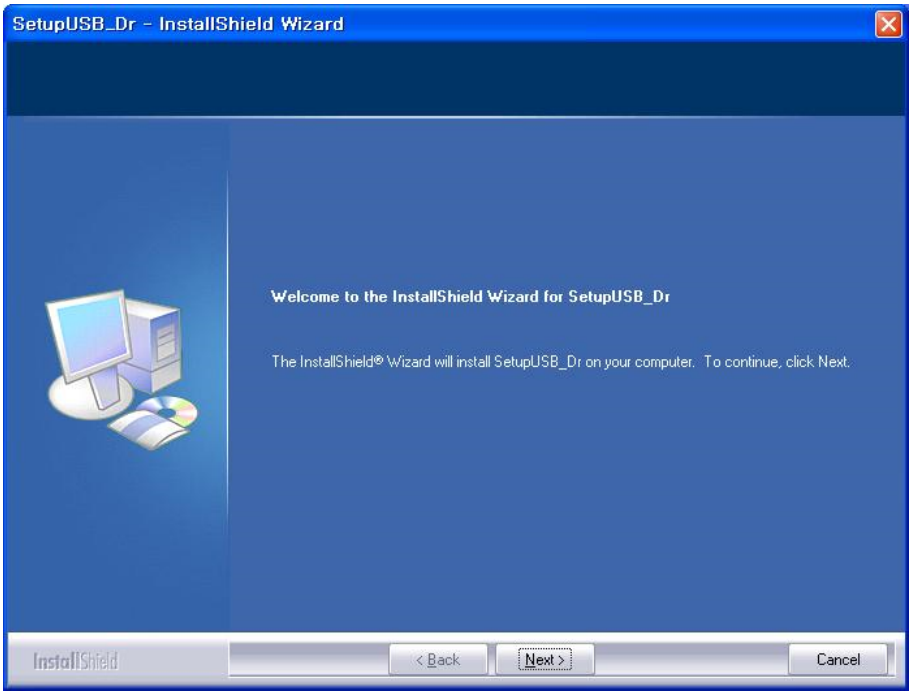
Perform the following:

1. Execute the installer.

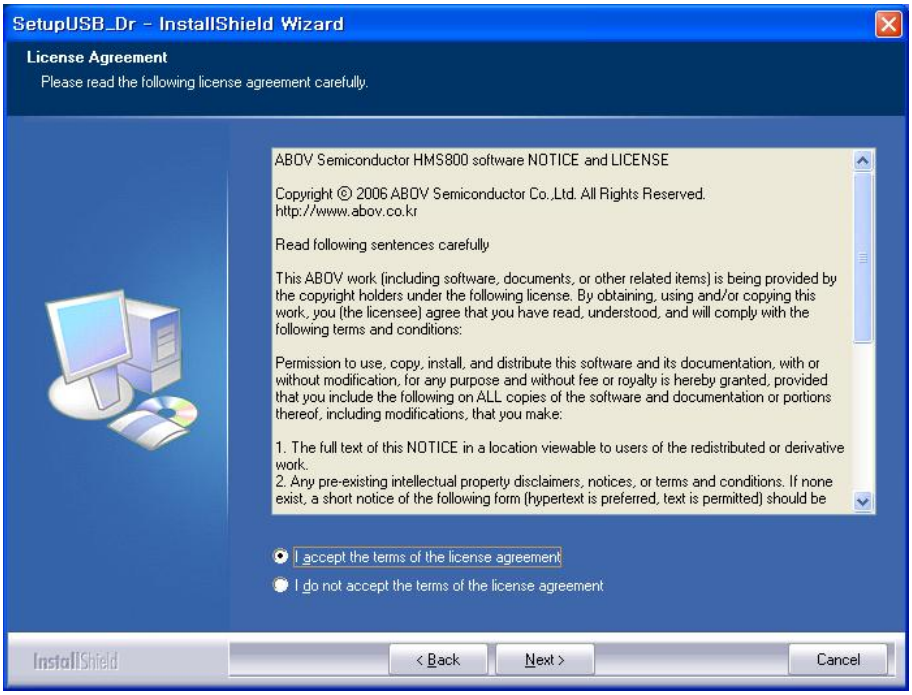
2. Once the warning message below pops up, click **Yes**.



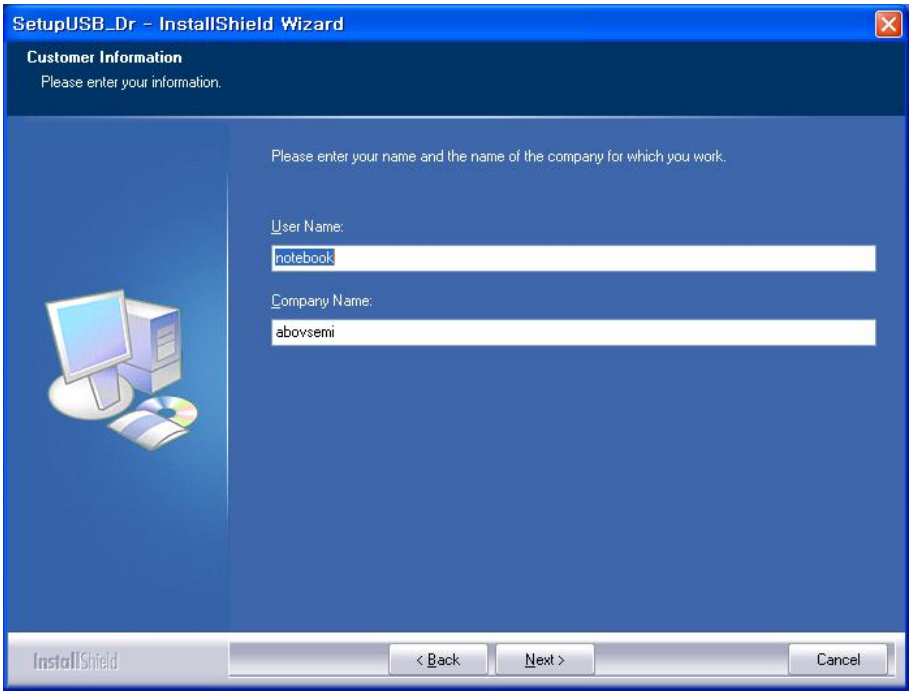
3. You will see the following dialog box. Click **Next**.



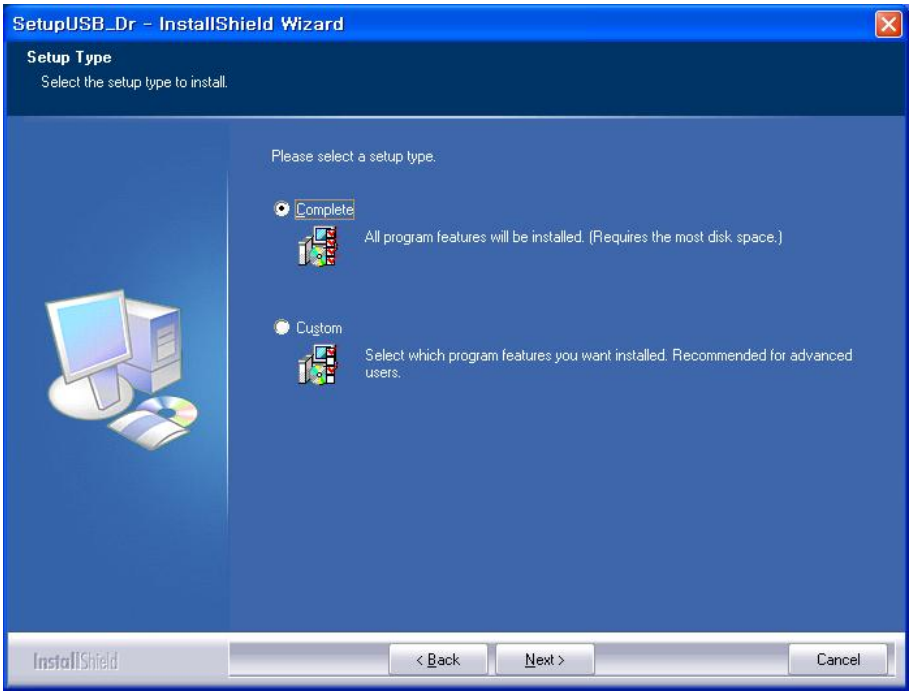
4. Select **I accept the items of the license agreement** on the License agreement dialog box and click **Next**.



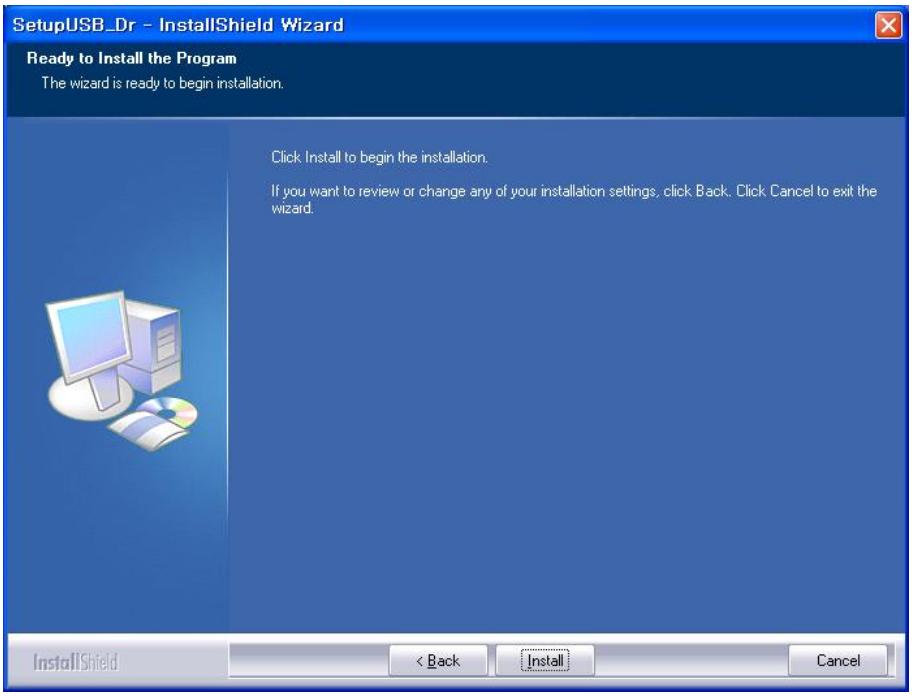
5. Enter the username and company name and click **Next**.



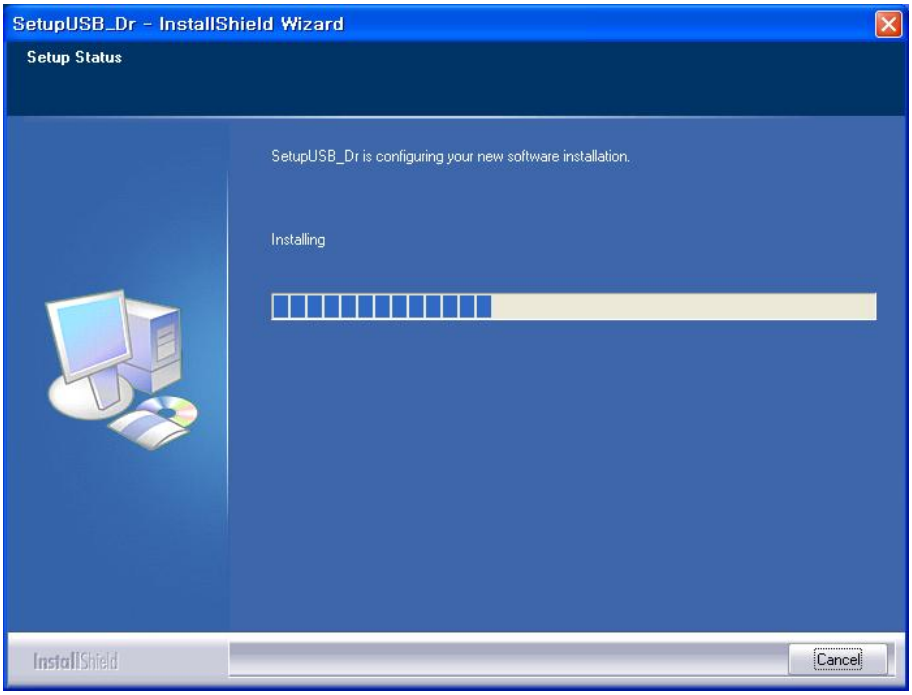
6. Select **Complete** and click **Next**.



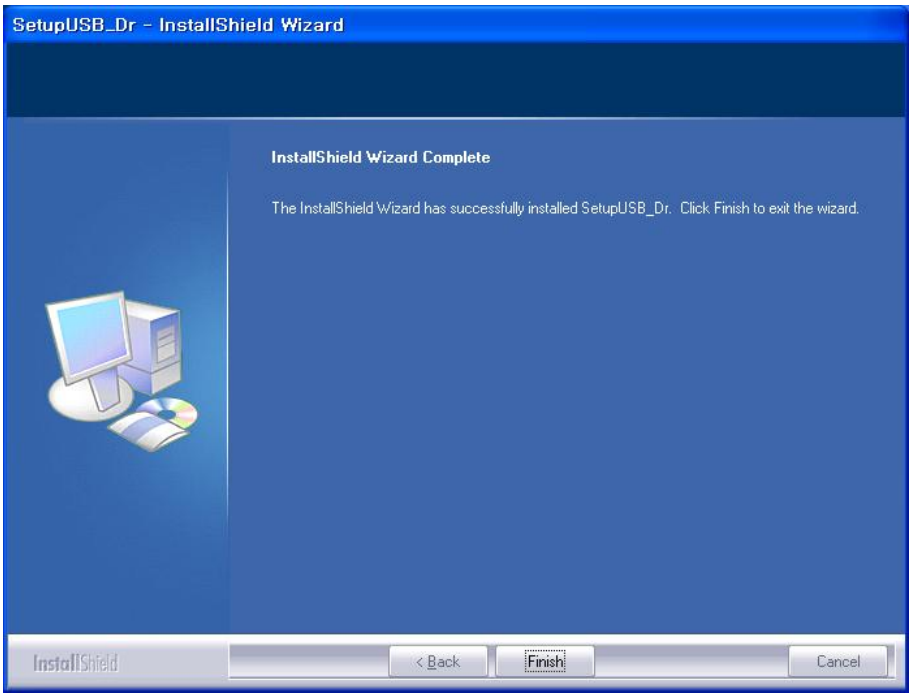
7. Click **Install**.



8. Wait until all the application components are installed.



9. Click **Finish** to complete the installation.



Chapter 2. Structure of CodeGen8 folders

2.1 Structure of CodeGen8 project folder

CodeGen8 includes CodeGen8 project files in the `Project` folder, which is located in the CodeGen8 installation folder. All CodeGen8 project files are either `\*.VPP` or `\*.VPD` files:

- `\*.VPP` files contain basic information (device name, package type, pin count, etc); for example,

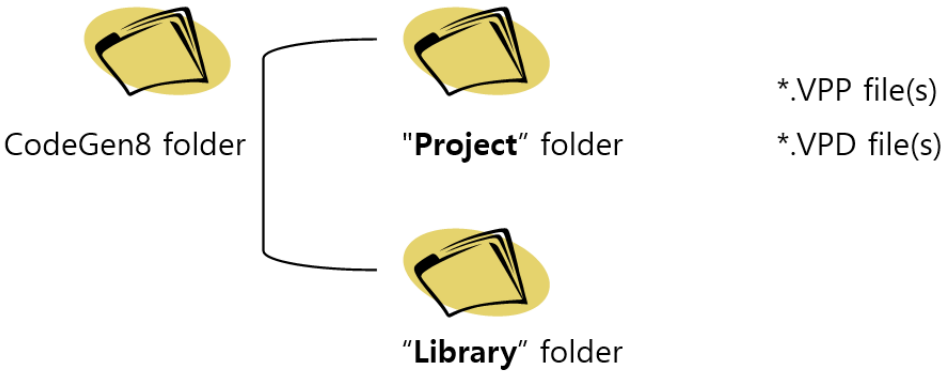
```
ABOV-CodeGen8-MC9x A1.03 20130419
MC96F6332 SOP 28
```

- `\*.VPD` files contain the target device's peripheral property settings; for example,

```
B_isSingle 0
PORT001 -1 0 0 0 0
BUZ 0 2659.574463
:
:
EXTINT10 0 0 0
WDT 0 0 120
```

Caution: Do not modify `\*.VPP` or `\*.VPD` files.

Folder assignment:



2.2 Structure of CodeGen8 library folder

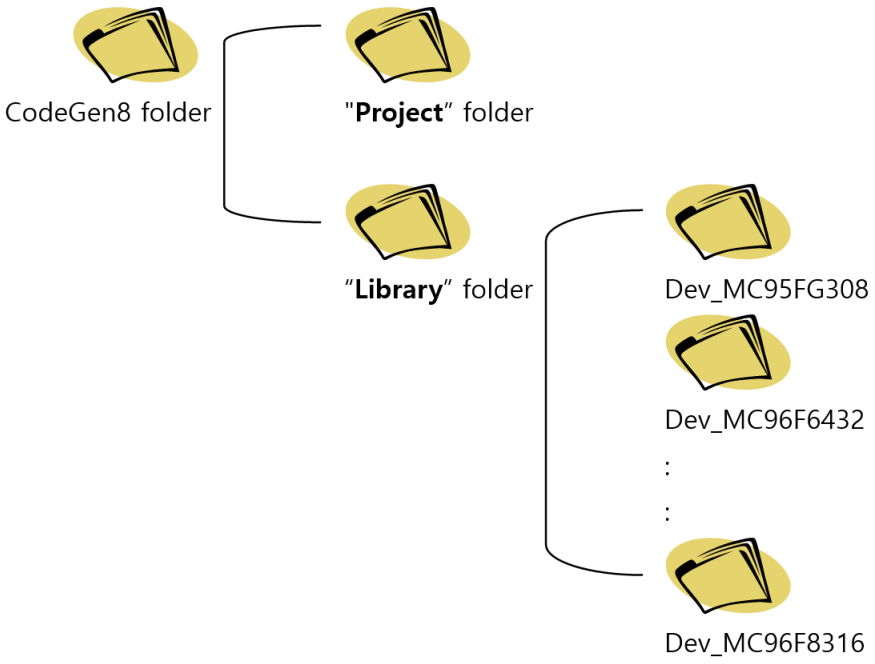
CodeGen8 includes each device's sample files in the device's specific library folder under the common `Library` folder, which is located in the CodeGen8 installation folder. These device-specific library folders contain the following:

- Device package definition file
- Device header file
- Sample source program files

CodeGen8 generates a standard basic source program. To see details, refer to the sample source files.

Caution:	Do not delete or modify any of the files in the `Library` folders.

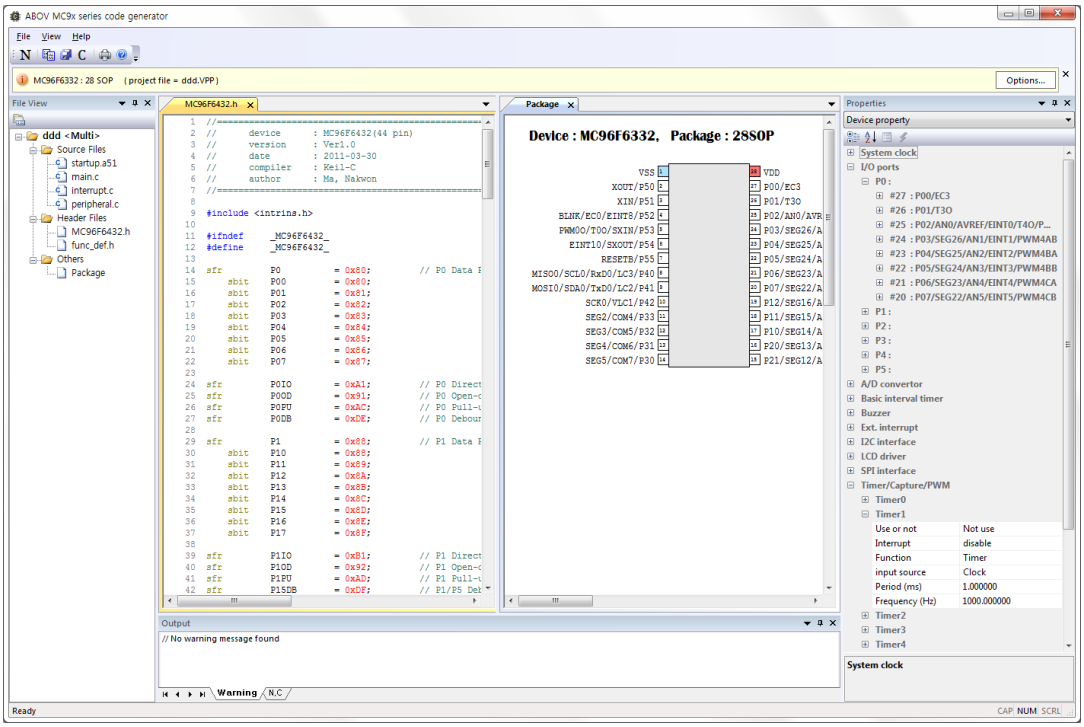
Folder assignment:



Chapter 3. Using CodeGen8

3.1 Features of CodeGen8

CodeGen8 supports the M8051-based MC9x/A9x series developed by ABOV Semiconductor Co., Ltd.



CodeGen8 screenshot

3.1.1 Detailed features

This software application aims at helping both beginners and veterans dramatically save time spent on 8-bit MCU programming. It provides the following functions:

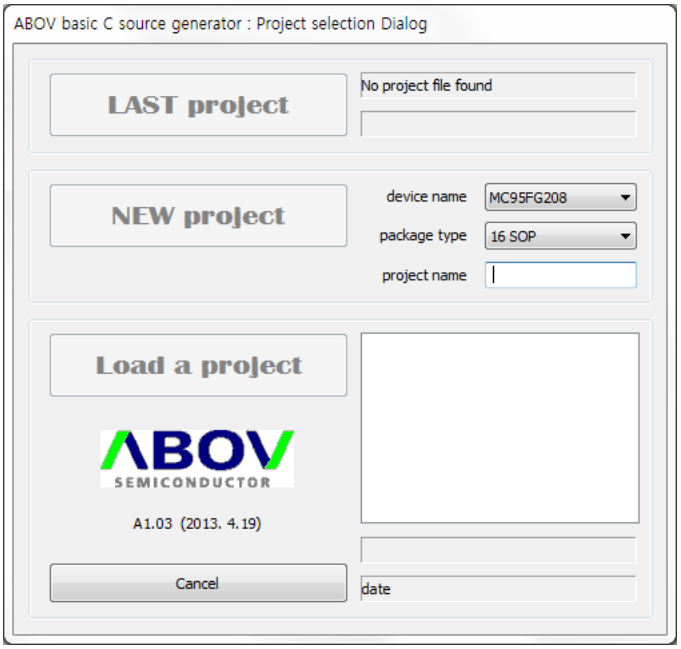
- Generates C source files for the target device simultaneously when the user sets a peripheral.
- Supports the KEIL project format by:
  - Generating `\*.uv2` project files. They support KEIL uVision 4 and 5.
  - Generating each device's header file (e.g. `MC95FG308.h`).
  - Generating the `startup.a51` assembly code file.
  - Generating a single source file (`main.c`) or multiple source files (`main.c`, `interrupt.c`, and `peripheral.c`).

- Displays the following:
  - Device header file(s)
  - Device source file(s)
  - Package view
- Provides a peripheral setting pane.
- Provides a package view that shows pin assignment and allows the user to set port functions.
- Manages CodeGen8 project files automatically.
- Generated source file contents are changed as soon as you click for device setting changes.

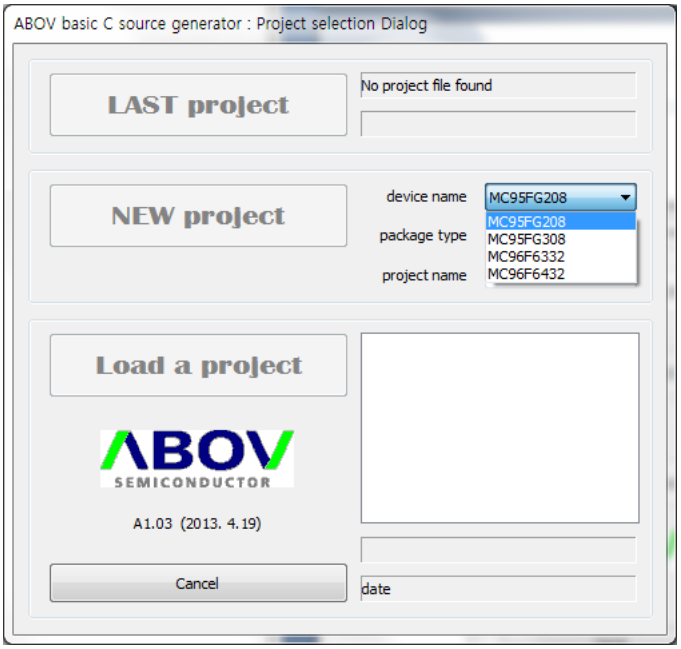
3.1.2 Getting started with CodeGen8

Perform the following:

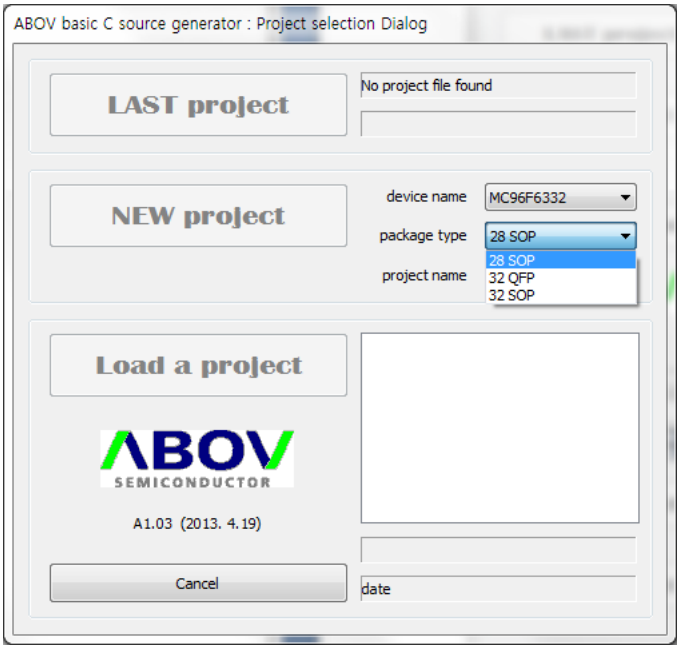
1. Execute CodeGen8 to open the following dialog box; among the controls, only **device name**, **package type**, and **Cancel** will be active.



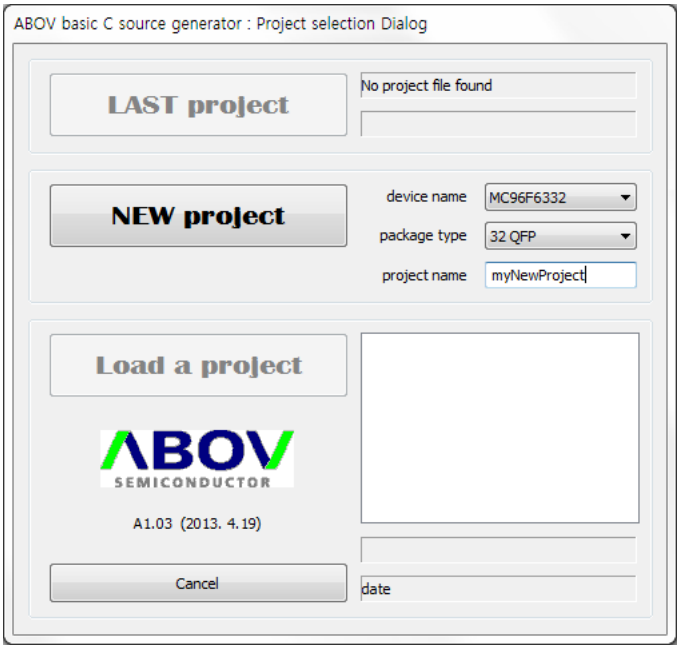
2. Select a target device.



3. Select a device package.

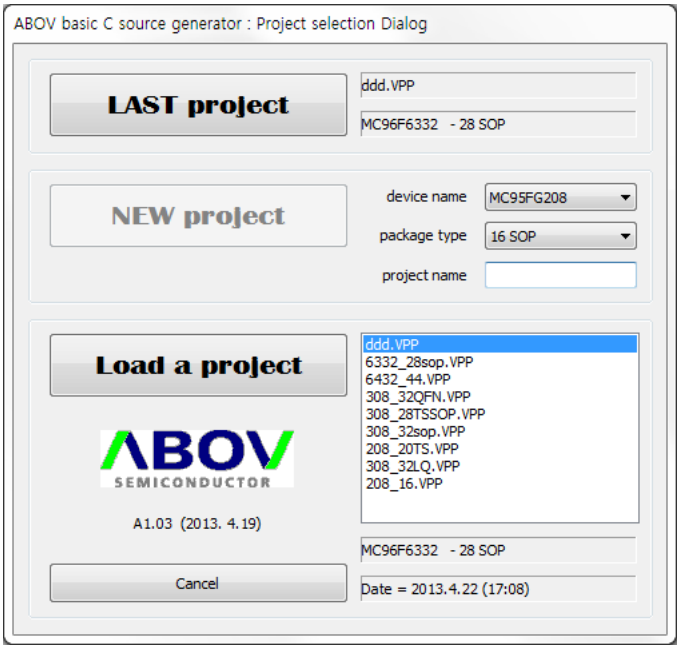


4. Type the project name, and the **New project** button will be enabled.



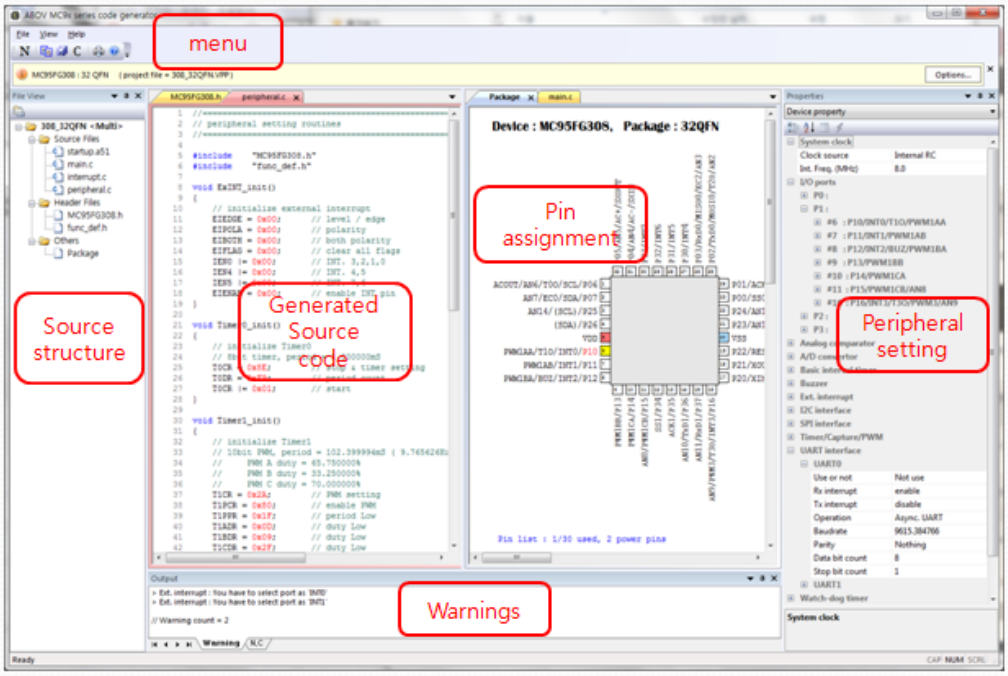
5. Click on **New project**.

6. If you have created more than one CodeGen8 project, the dialog box will list them as shown below:



- **LAST project:** Load the last project.
- **NEW project:** Create a new project.
- **Load a project:** Load a project from the list of previous projects.

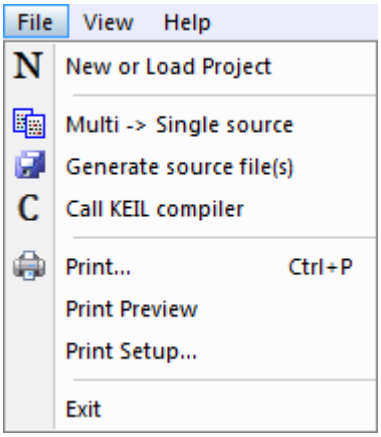
7. On the main screen below, configure your device.



3.2 Menu descriptions

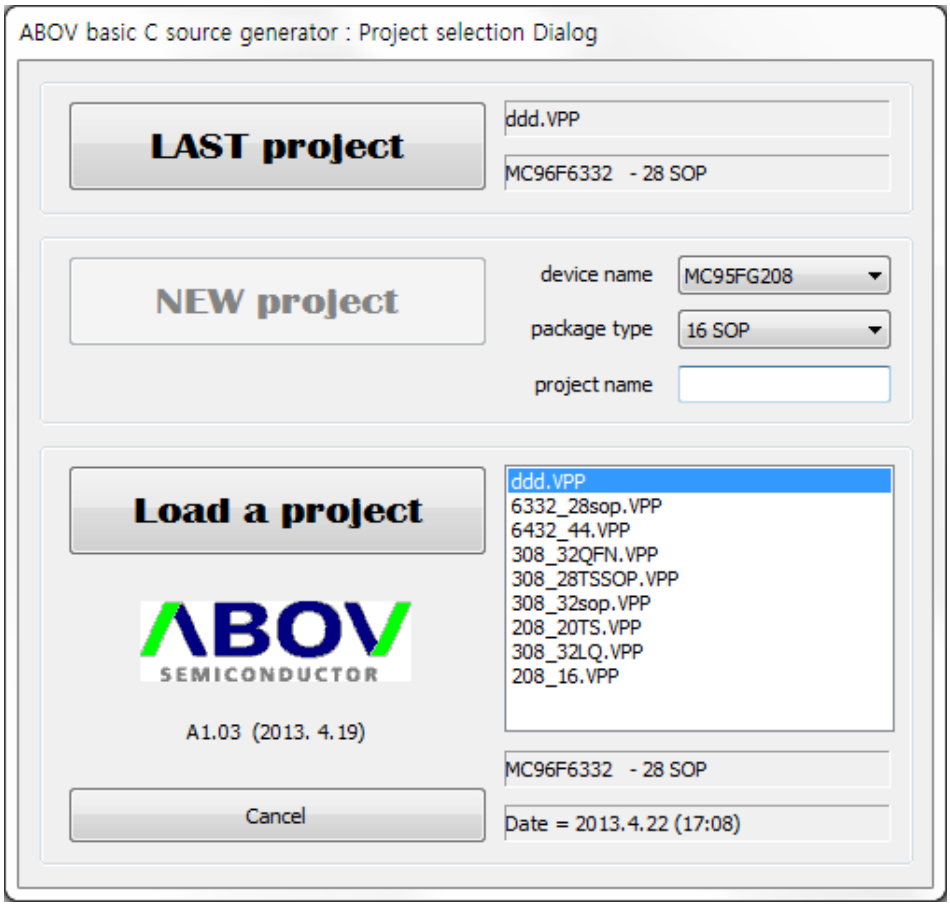
3.2.1 File

The **File** menu includes the following items:



New or Load Project

This menu item closes the currently open CodeGen8 project and prompts the user to create a new project or load a project from the list of previous projects.



Multi -> Single source / Single -> Multi source

Single-source project:

CodeGen8 generates a single source file named `main.c`. This file contains the main function routine, interrupt vector functions, and peripheral functions. It is automatically closed when a multisource file (e.g. `interrupt.c`) is opened.

Multisource project:

CodeGen8 generates `main.c`, `interrupt.c`, and `peripheral.c` source files.

- The `main.c` file contains the main function routine only.
- The `interrupt.c` file contains interrupt vector functions.
- The `peripheral.c` file contains each peripheral's initializing functions.

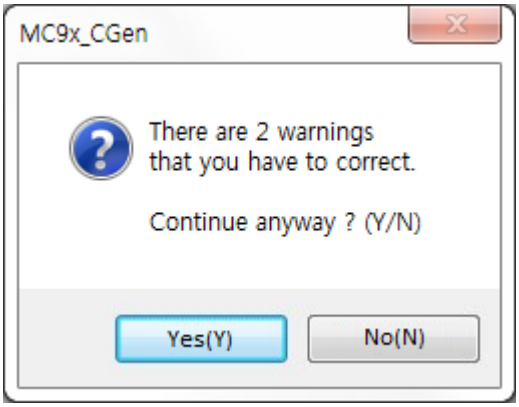
You can see the file structure in the **File View** pane.

Generate source file(s)

CodeGen8 saves the following files:

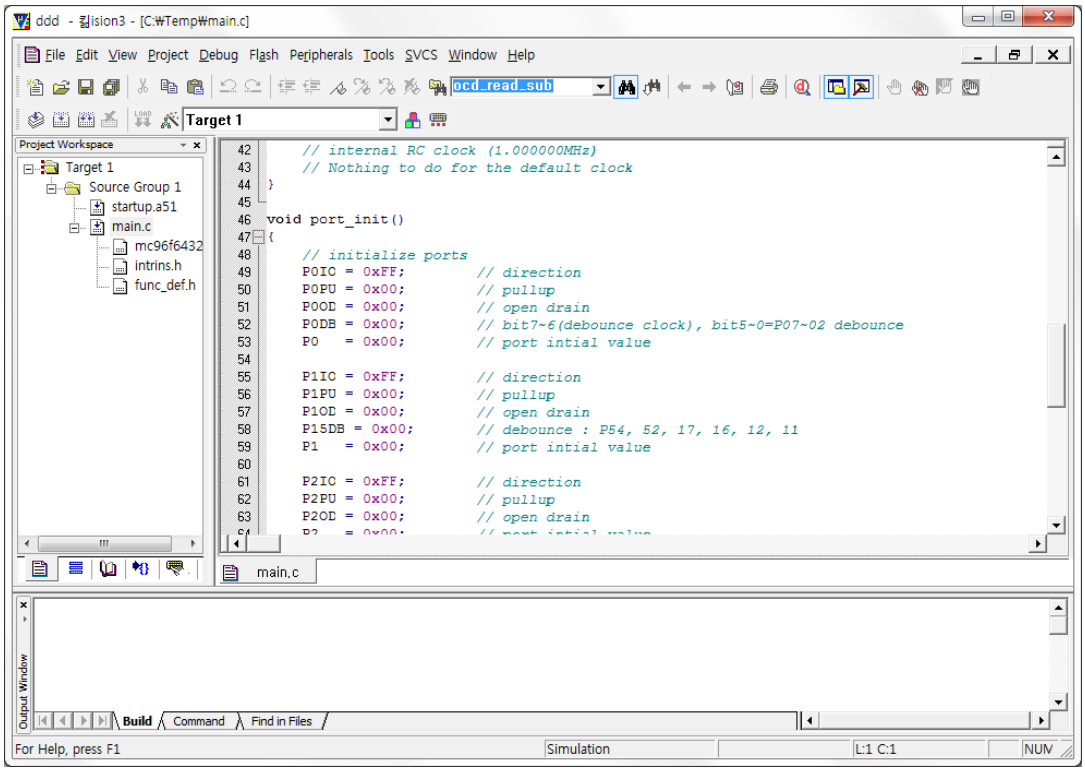
- Header files
  - Device header file, for example, `MC95FG308.h`
  - Function definition file, for example, `func_def.h`
- Source files
  - Assembly file that includes the initialization code, for example, `startup.a51`
  - C source program file, for example, `main.c`, `interrupt.c`, or `peripheral.c`
- KEIL project files
  - It contains the target device, file management, etc., for example, `projectName.uv2`.

If there is a warning appearing because of errors in the device settings, it prompts to choose whether to continue or not. You can see the warning in the **Output** pane.



Call KEIL compiler

This menu item saves the current header file(s), source file(s), and KEIL project file, and calls the KEIL compiler as shown below:



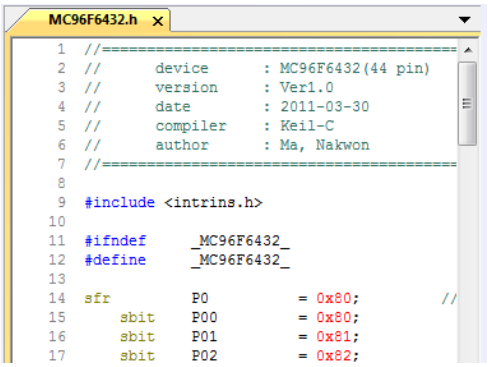
If there is a warning appearing because of errors in the device settings, it prompts to choose whether to continue or not. You can see the warning in the **Output** pane.

Print…

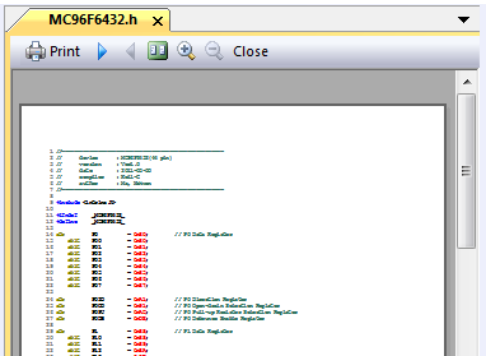
It prints the current child window. The child windows include the view of each source code (header, C) and a package view.

Print Preview

This feature allows you to see exactly how your pages will look when they are printed out. Using this function, you can save paper right up until final printing. **Print Preview** is easy to use, and you can print directly from this screen by clicking on the **Print** button or printer icon.



Normal view



Print Preview

Print Setup...

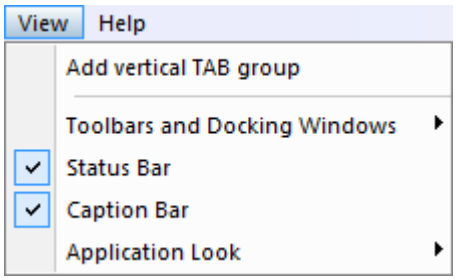
This command displays the standard **Print Setup** dialog. You can choose a printer and select other options.

Exit

This button helps quit from CodeGen8 immediately.

3.2.2 View

The **View** menu controls the display of the CodeGen8 software frame and child windows.

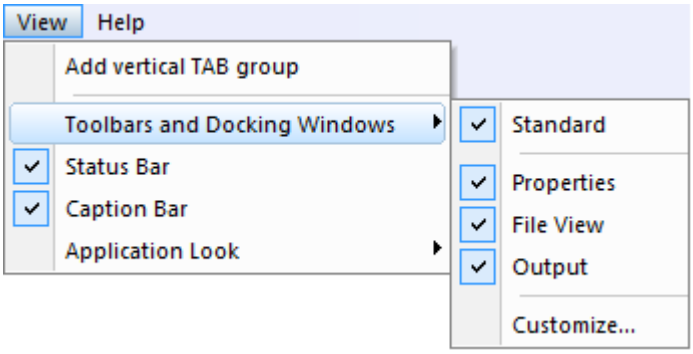


Add vertical TAB group

It provides the functionality for tab control. The tab control displays a dockable window with flat or three-dimensional tabs at its top or bottom. The tabs can display text and images and can change colors when active.

Toolbars and Dockable Windows

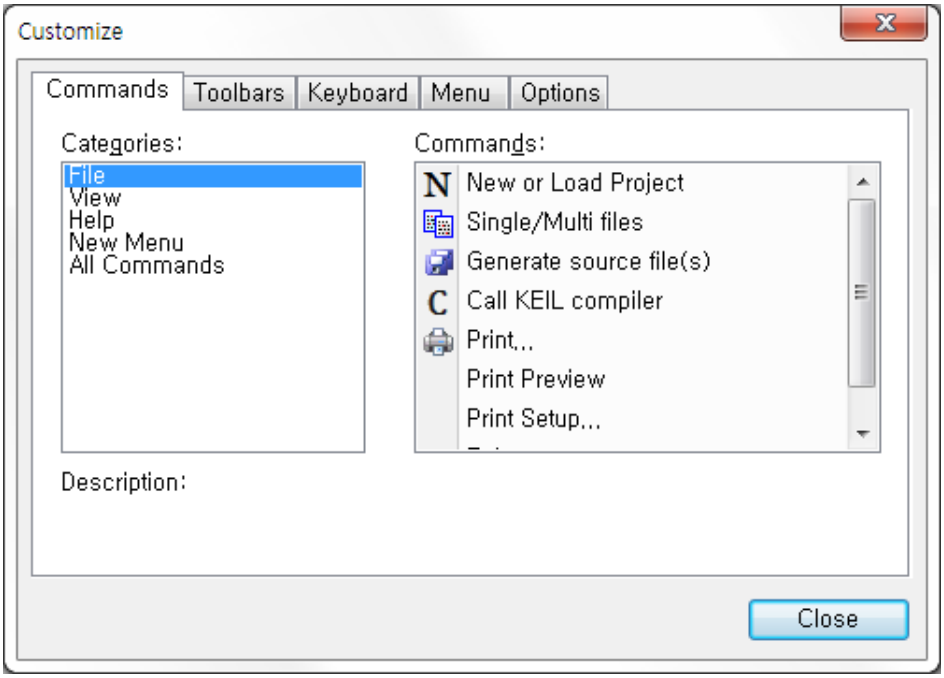
This menu item shows or hides different child views.



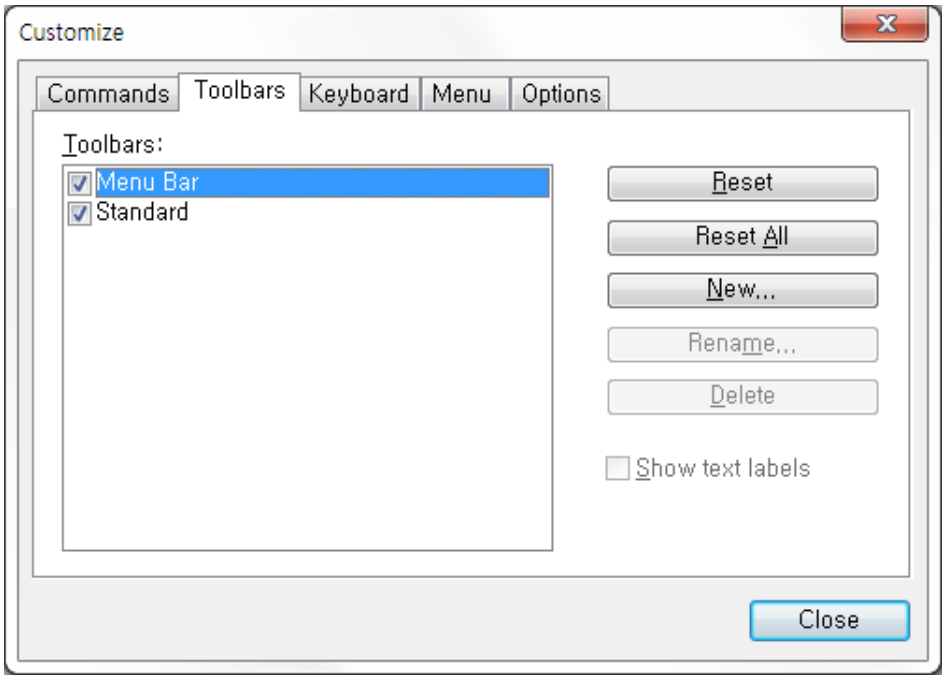
Customize

It allows users to change the debugger software GUI environment to suit their preferences regarding **Commands**, **Toolbars**, **Keyboard**, **Menu**, and **Options**.

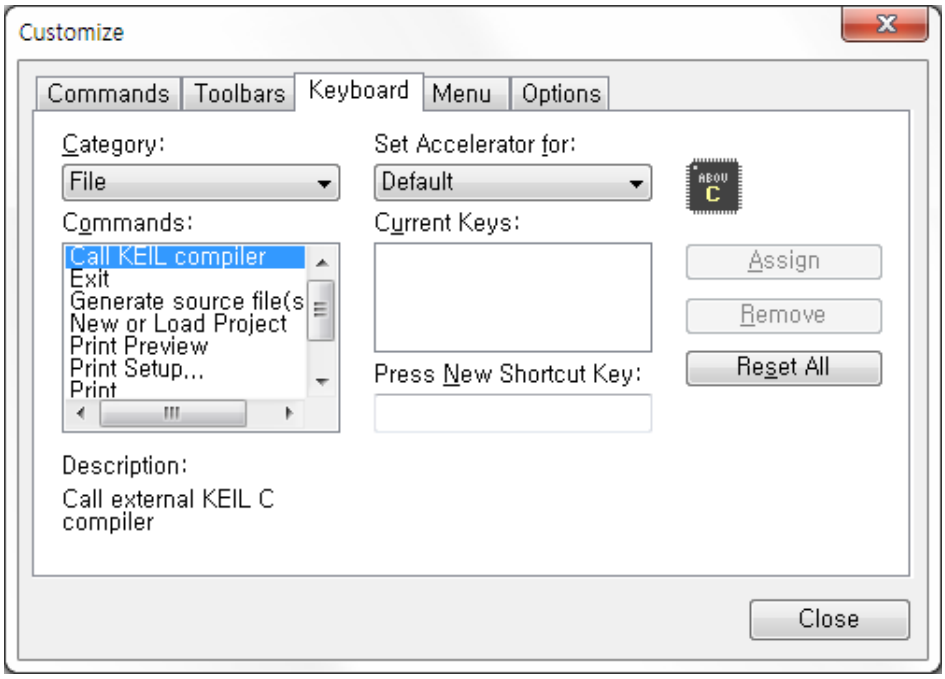
The **Commands** tab provides choices to modify the composition of each menu's sub-items.



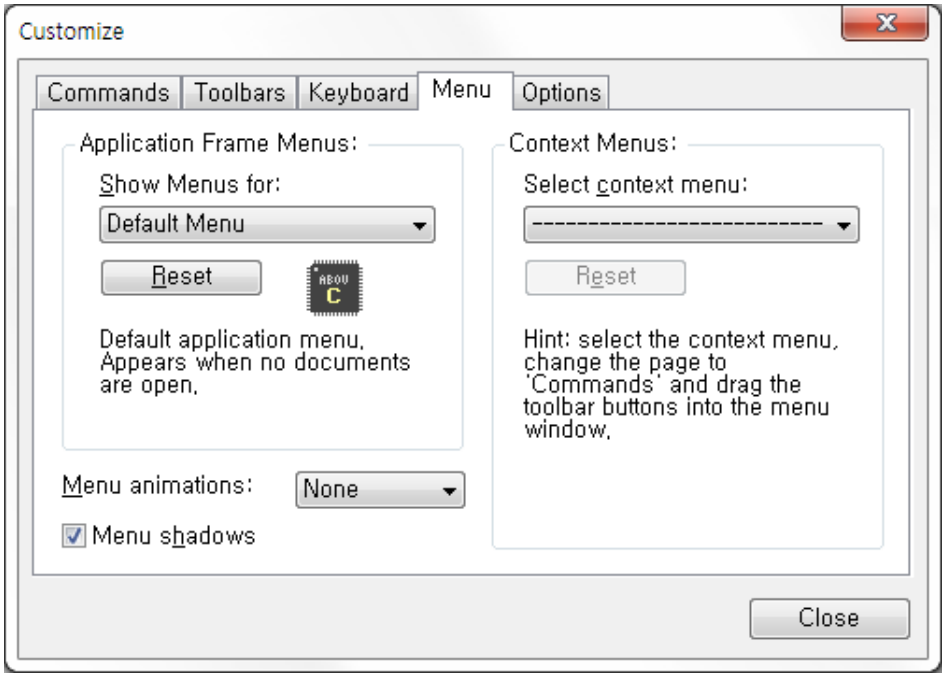
The **Toolbars** tab allows you to switch between toolbar styles.



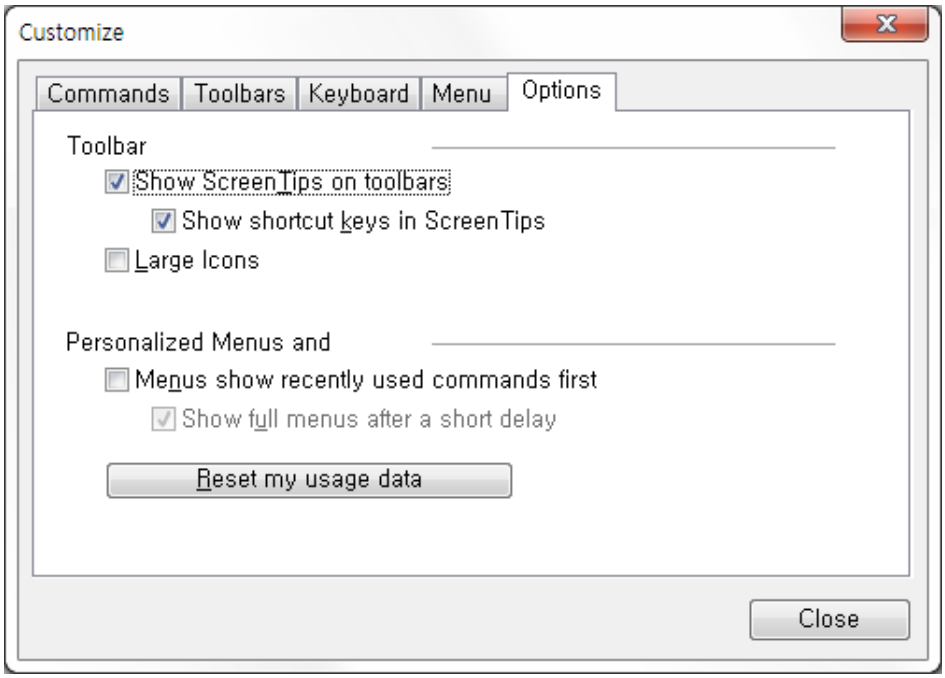
The **Keyboard** tab allows you to define shortcut keys. You can reset them or restore them to the default settings.



The **Menu** tab is to define the menu style.



The **Options** tab provides settings regarding toolbar tip display, icon size, etc.

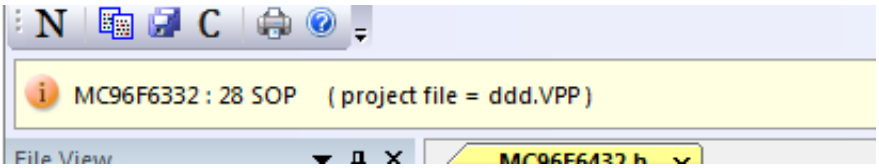


Status Bar

It turns on or off the Status bar, which displays information on the current state of CodeGen8.

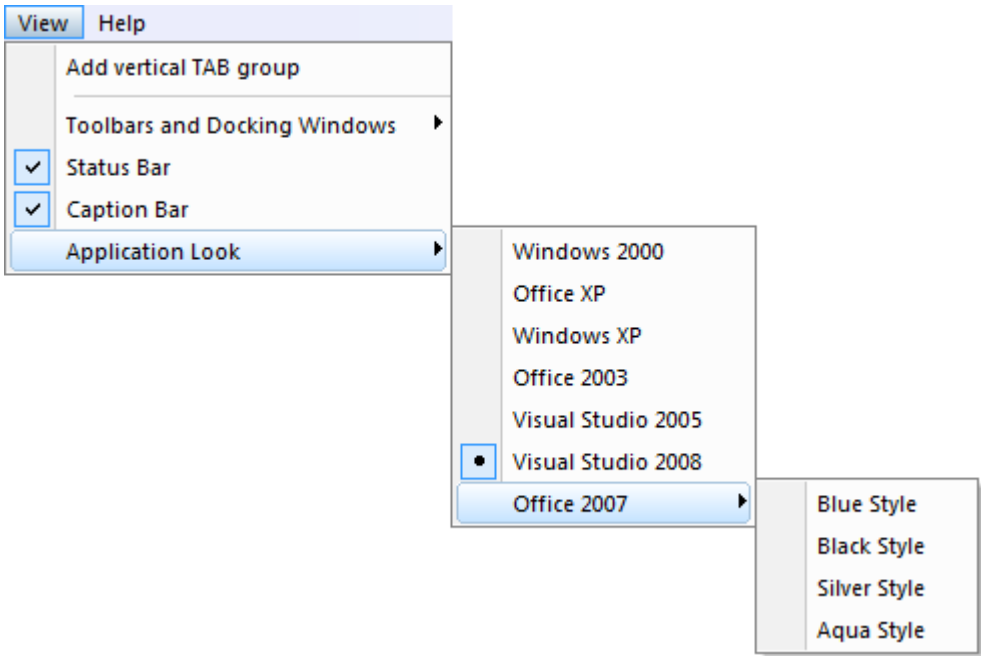
Caption Bar

It turns on or off the Caption bar, which displays the device name, package type, and project name.

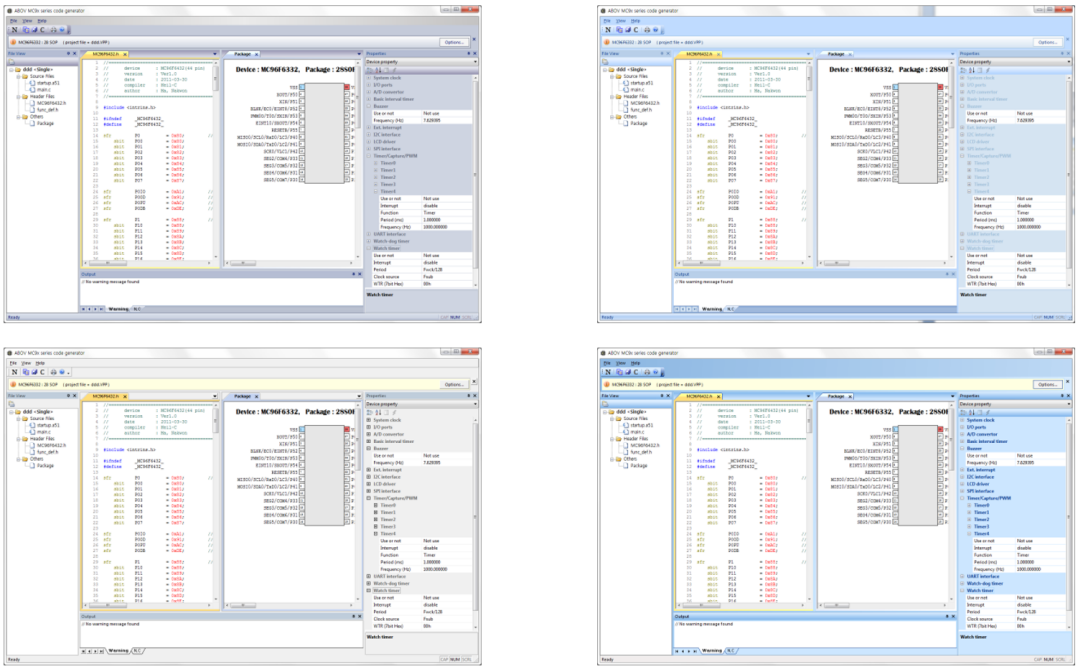


Application Look

It changes CodeGen8's GUI look at once.



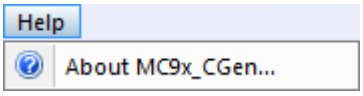
GUI selection

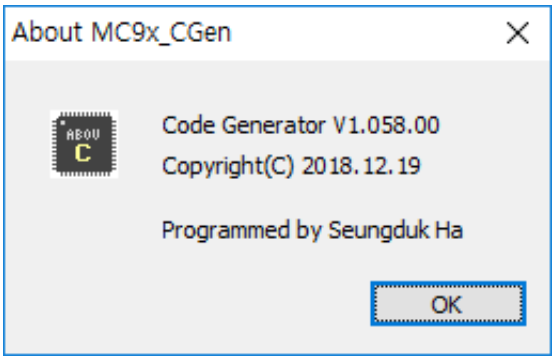


Different GUI styles

3.2.3 Help

It shows the CodeGen8 version and copyright, and the programmer's name.





3.2.4 Toolbar

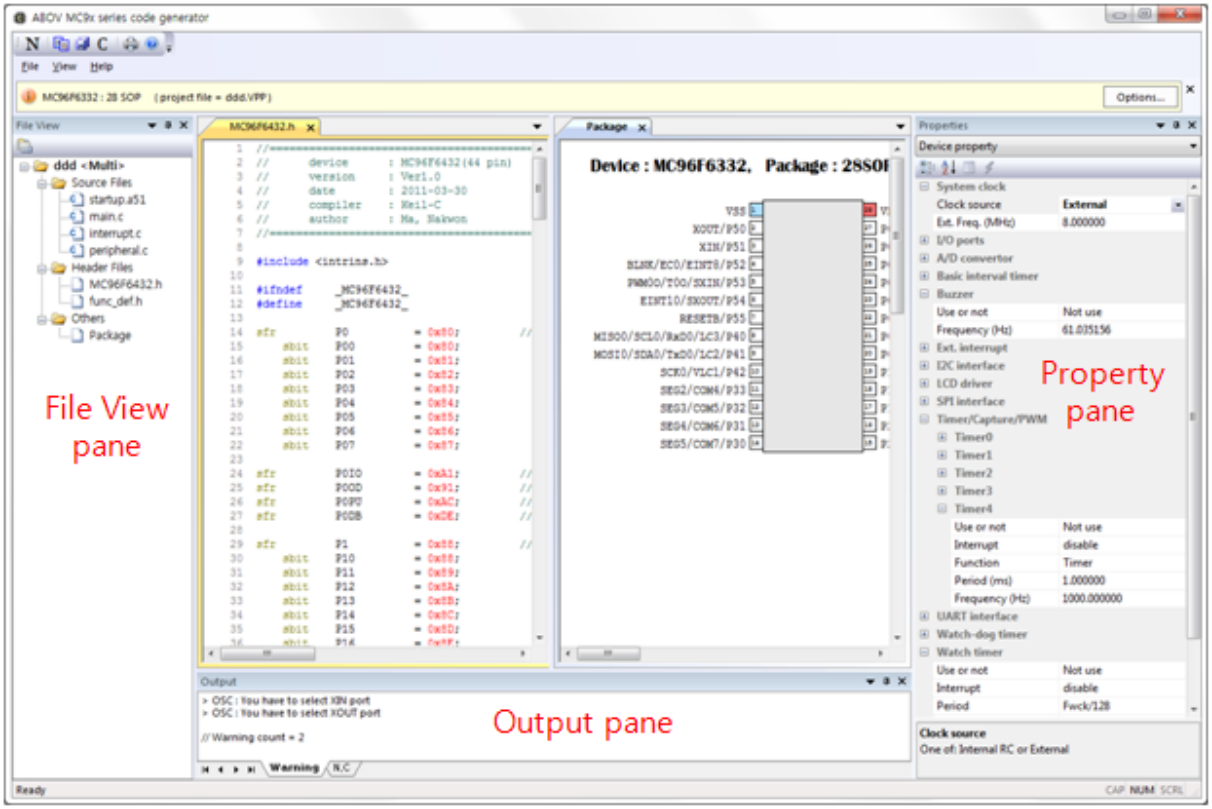
This provides an easy and single-click access to the most frequently used commands.

3.3 Control panes

CodeGen8 provides three control panes for developers. The information displayed on those panes include project status, device settings, and warnings. All panes except the child windows are dockable.

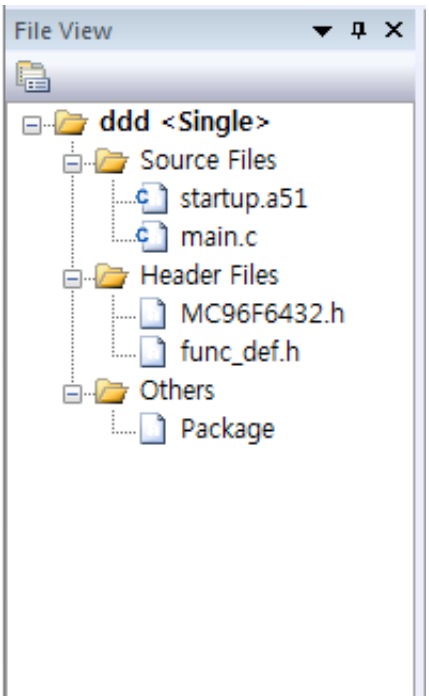
What is docking

Docking is manipulating a window to align it with the edge of another window or to move it into another window.

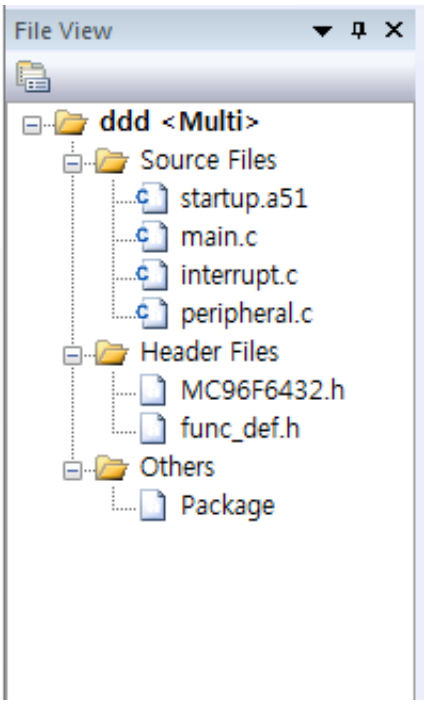


3.3.1 File View pane

It shows the file components of the current project. Single-source projects generate `main.c`, while multisource projects generate another two C files (`interrupt.c` and `peripheral.c`). The rest files (`startup.a51`, `func_def.h`, and device header) are common. You can open any file by double-clicking on the file name.



Example: Single-source project

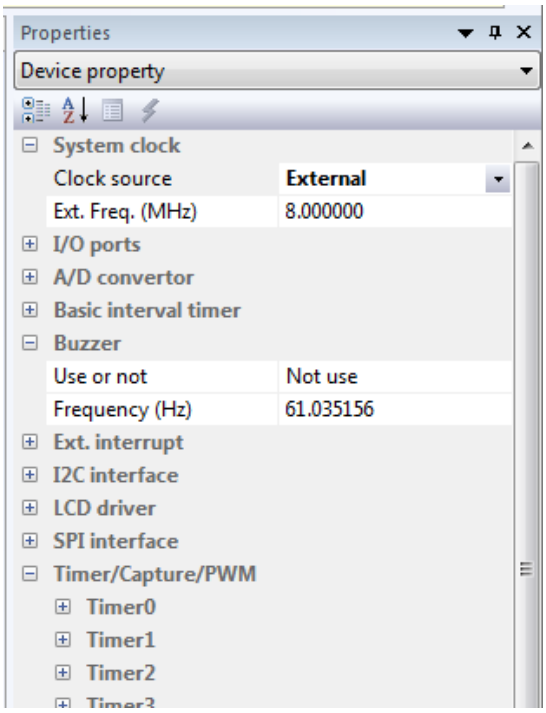


Example: Multisource project

3.3.2 Properties pane

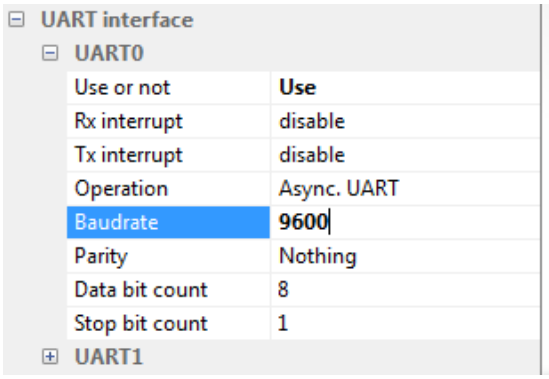
It shows the peripherals of the current project. This is the most important component of CodeGen8.

- This lists specifications of each peripheral.
- The peripheral settings are viewable and editable.
- If you change any of the settings, the source program is generated automatically.
- Some peripherals support only one function; an attempt to enable multiple functions pops up a warning message.
- If a peripheral setting conflicts with other peripherals, the **Output** pane displays it.

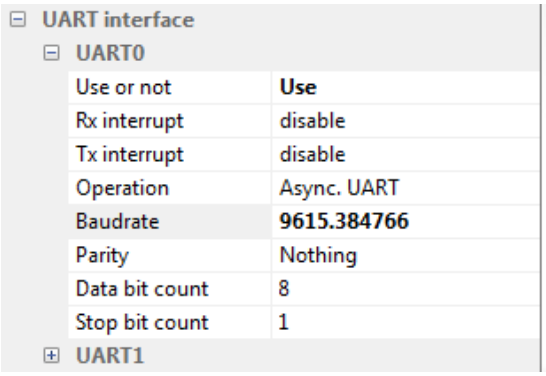


Case 1

Suppose that the system clock is 4 MHz, and you want to use UART at 9600 bps. But a baud rate of exactly 9600 bps cannot be rendered at 4 MHz. CodeGen8 calculates and displays the nearest bps and generates the source program.



Your input: 9600 bps



CodeGen8 calculation: 9615.38 bps

Case 2

Suppose that you are using MC95FG308, whose Timer0 and Timer1 are 8-bit timers. If you want to use Timer0 as a 16-bit timer, Timer1 must be extended to the upper 8-bit timer of Timer0. This means, if Timer0 is set to 16 bits, you do not need to set Timer1. CodeGen8 disables editing of Timer1.

Timer/Capture/PWM	
Timer0	
Use or not	Use
Interrupt	disable
Function	Timer
bit count	8bit
input source	Clock
Period (ms)	1.000000
Frequency (Hz)	1000.000000
Timer1	
Use or not	Use
Interrupt	enable
Function	PWM
input source	Clock
Period (ms)	1.000000
Frequency (Hz)	1000.000000
PWM polarity	Low first
PWM duty A (%)	66.000000
PWM duty B (%)	33.299999
PWM duty C (%)	70.000000

Timer0 and Timer1 are separated

Timer/Capture/PWM	
Timer0	
Use or not	Use
Interrupt	disable
Function	Timer
bit count	16bit
input source	Clock
Period (ms)	1.000000
Frequency (Hz)	1000.000000
Timer1	
Use or not	Not use
Interrupt	enable
Function	PWM
input source	Clock
Period (ms)	1.000000
Frequency (Hz)	1000.000000
PWM polarity	Low first
PWM duty A (%)	66.000000
PWM duty B (%)	33.299999
PWM duty C (%)	70.000000

Timer0 and Timer1 are merged

You do not need to know all the peripheral specifications because it is straightforward to set all the peripherals. If you want to make more detailed settings, then you need to understand all the peripheral specifications.

3.3.3 Output pane

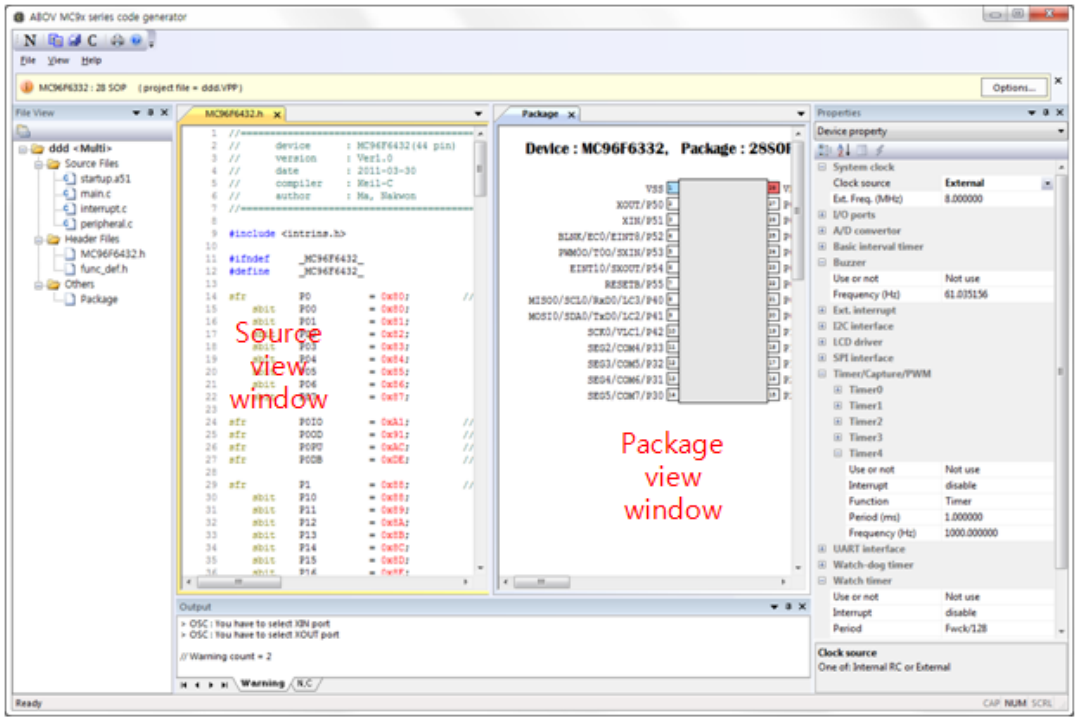
It shows warning messages regarding peripheral settings or conflicts between them.

```
Output
> OSC : You have to select XIN port
> OSC : You have to select XOUT port

// Warning count = 2
Warning  N.C
```

You must clear the warning messages by changing the device's peripheral settings. Else, it generates a C source program omitting some peripheral settings.

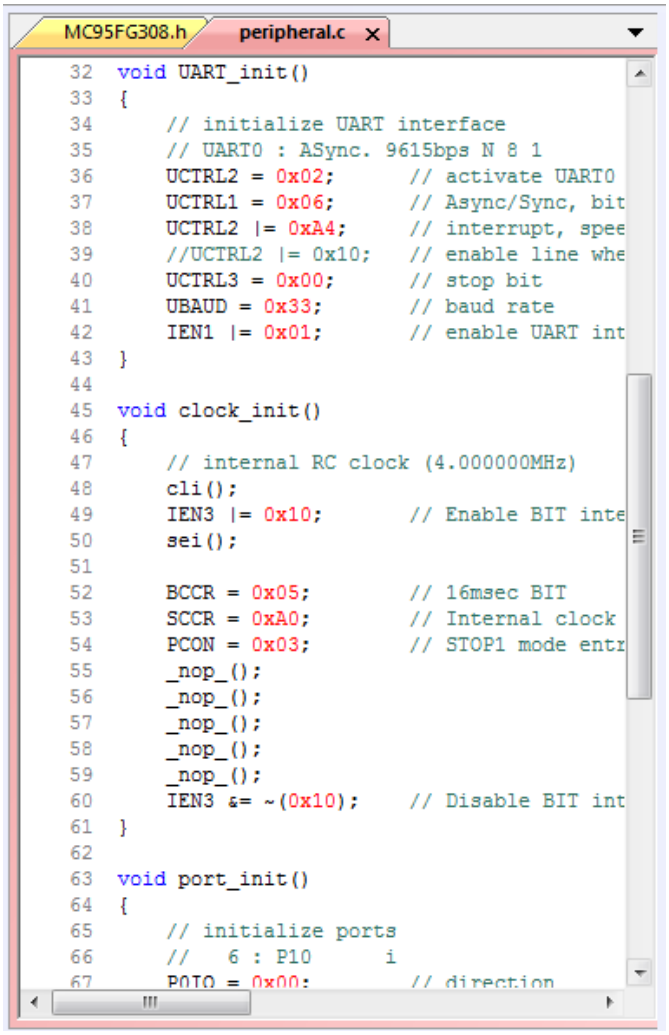
3.4 Child windows

The child windows operate differently from the previously mentioned panes. They include a source file view, virtual source program view, and device package view. They all are simpler views and do not provide editing functions.



3.4.1 Source program view

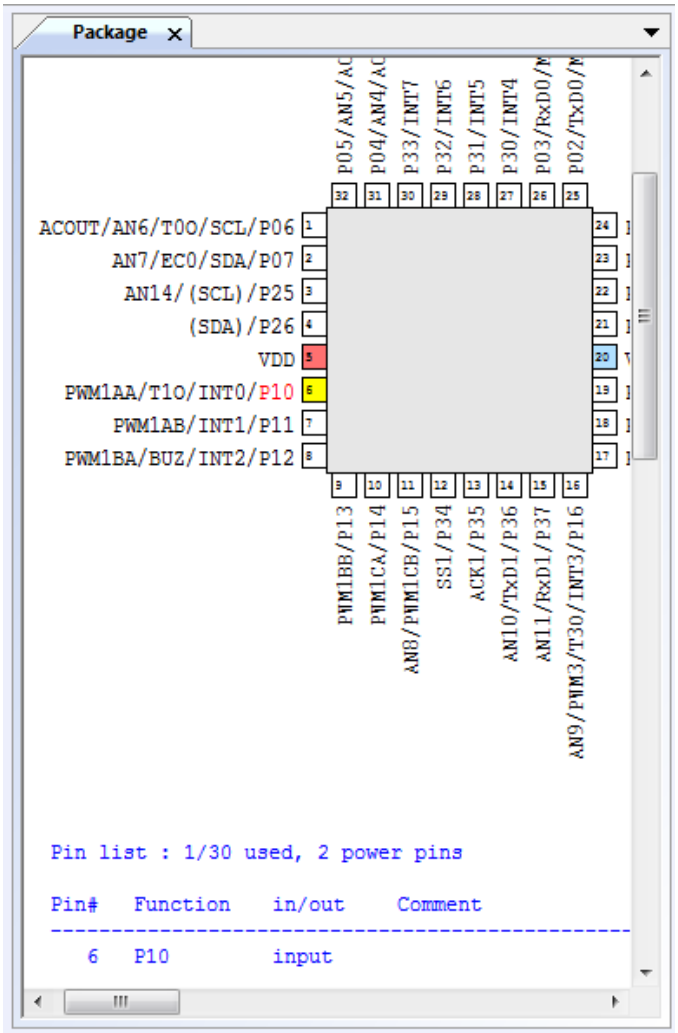
It shows text files or virtually generated source code files with line numbers.



It does not support file editing but allows the user to copy the text to the clipboard. Tab size displayed in the view is fixed at 4. C or assembly keywords and comments are colored for an enhanced visibility.

3.4.2 Package view

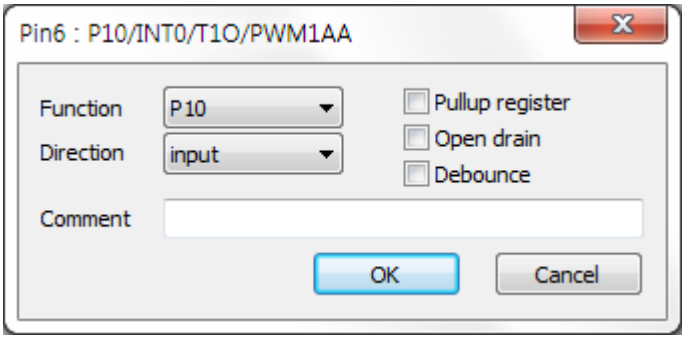
It shows the current target device's package shape and its pin assignment.



You can easily read the status of each pin based on its color:

Pin color	Meaning
Red	This pin is a power source pin.
Blue	This pin is a ground pin.
White	This pin is not assigned a function yet.
Yellow	This pin is assigned a specific function. Assigned function is to change the text color to red. Assigned pin list is displayed under the package shape.

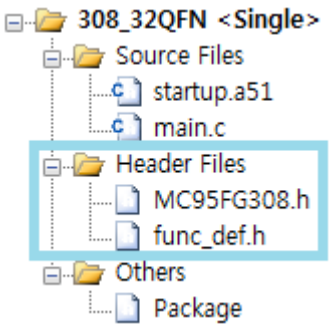
You can set a port function both in the **Properties** pane and from this window. By double-clicking a pin, you can edit port properties using the following dialog box:



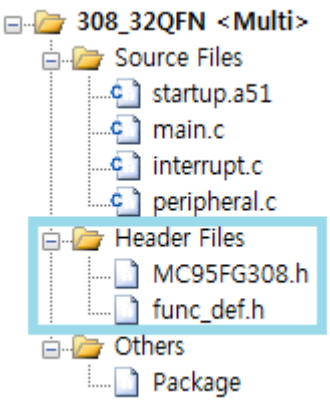
Chapter 4. Output files

4.1 Header files

CodeGen8 generates two header files as shown below. They are not affected by whether they belong to single-source or multisource projects.



Example: Single-source project



Example: Multisource project

4.1.1 Device header

The device header file contains the peripheral definitions. This is the real text file located in the device's `Library` folder. You do not need to modify this file when you are working in a KEIL environment.

The following is an example of `MC95FG308.h`:

```
#include <intrins.h>

#ifndef _MC95FG308_
#define _MC95FG308_

//=====================================================
// PORT Control Register
//=====================================================
// PORT0
sfr         P0          = 0x80;         // P0 Data Register
    sbit    P00         = 0x80;
    sbit    P01         = 0x81;
    sbit    P02         = 0x82;
    sbit    P03         = 0x83;
    sbit    P04         = 0x84;
    sbit    P05         = 0x85;
    sbit    P06         = 0x86;
    sbit    P07         = 0x87;
sfr         P0DA        = 0x80;         // P0 Data Register
    sbit    P0DA0       = 0x80;
    sbit    P0DA1       = 0x81;
    sbit    P0DA2       = 0x82;
    sbit    P0DA3       = 0x83;
    sbit    P0DA4       = 0x84;
    sbit    P0DA5       = 0x85;
    sbit    P0DA6       = 0x86;
    sbit    P0DA7       = 0x87;
sfr         P0IO        = 0x89;         // P0 Direction Register
#define     P0PU        *(volatile unsigned char xdata *) 0x2F00    // P0 Pull-up Register
#define     P0OD        *(volatile unsigned char xdata *) 0x2F0C    // P0 Open Drain Register
#define     P0DB        *(volatile unsigned char xdata *) 0x2F18    // P0 DEBOUNCE Register
// PORT1
sfr         P1          = 0x88;         // P1 Data Register
    sbit    P10         = 0x88;
    sbit    P11         = 0x89;
```

4.1.2 func_def.h

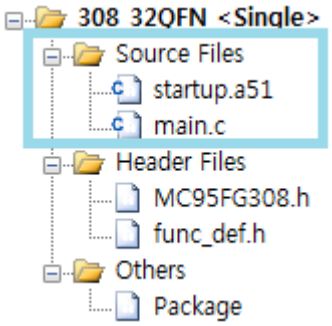
This file contains function definitions. This is a virtual text file that does not exist in any folder. The function name, function body, and comments are assigned by CodeGen8 automatically. You can recognize each function's operation easily with the function name and comments. You do not need to modify this file when you are working in a KEIL environment.

The following is an example of `MC95FG308.h`:

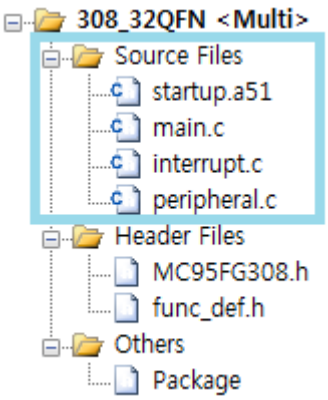
```
//=====================================================
// Function and global variables definition
//=====================================================

void port_init();                   // initialize ports
void clock_init();                  // initialize operation clock
void BIT_init();                    // initialize Basic interval timer
void BUZ_init();                    // initialize Buzzer
void BUZ_OnOff(unsigned char On);    // Buzzer ON(On=1) / OFF(On=0)
void ExINT_init();                  // initialize external interrupt
void Timer0_init();                 // initialize Timer0
void UART_init();                   // initialize UART interface

```

4.2 Source files

These files contain the skeleton of the real source code. The function name, function body, and comments are assigned by CodeGen8 automatically.



Example: Single-source project



Example: Multisource project

4.2.1 startup.a51

It contains device memory and stack initialization code. This is a real file located in the device's `Library` folder. You do not need to modify this file when you are working in a KEIL environment.

The following is an example of `startup.a51`.

```
83  ?C_C51STARTUP   SEGMENT   CODE
84  ?STACK          SEGMENT   IDATA
85
86                  RSEG    ?STACK
87                  DS      1
88
89  EXTRN CODE (?C_START)
90
91                  PUBLIC  ?C_STARTUP
92
93                  CSEG    AT      0
94  ?C_STARTUP:     LJMP    STARTUP1
95
96                  RSEG    ?C_C51STARTUP
97
98  STARTUP1:
99
100 IF IDATALEN <> 0
101                 MOV     R0,#IDATALEN - 1
102                 CLR     A
103 IDATALOOP:      MOV     @R0,A
104                 DJNZ    R0,IDATALOOP
105 ENDIF
106
107 IF XDATALEN <> 0
108                 MOV     DPTR,#XDATASTART
109                 MOV     R7,#LOW (XDATALEN)
110   IF (LOW (XDATALEN)) <> 0
111                 MOV     R6,#(HIGH (XDATALEN)) +1
112   ELSE
113                 MOV     R6,#HIGH (XDATALEN)
114   ENDIF
115                 CLR     A
116 XDATALOOP:      MOVX    @DPTR,A
```

4.2.2 main.c

The `main.c` file is the most important in C programming. Every C program starts with the main function and ends with a null statement. The following are the characteristics of the main function:

- Any C program can have only one main function.
- This function is called by the reset vector.
- In MiCOM programming, the main function must not be terminated to prevent malfunction.

CodeGen8 generates the `main.c` file very clearly and simply. It contains initialization codes only. CodeGen8 generates slightly different `main.c` files in single-source and multisource projects. The `main.c` file in a single-source project contains the main function, interrupt vector functions, and initialization functions.

```
11  void main()
12  {
13      cli();              // disable INT. during peripheral setting
14      port_init();        // initialize ports
15      clock_init();       // initialize operation clock
16      ExINT_init();       // initialize external interrupt
17      sei();              // enable INT.
18
19      // TODO: add your main code here
20
21      while(1);
22  }
23
24  //=======================================================
25  // interrupt routines
26  //=======================================================
27
28  void INT_BIT() interrupt 22
29  {
30      // BIT interrupt
31      // TODO: add your code here
32  }
33
34  //=======================================================
35  // peripheral setting routines
36  //=======================================================
37
38  void ExINT_init()
39  {
40      // initialize external interrupt
41      EIEDGE = 0x00;      // level / edge
42      EIPOLA = 0x00;      // polarity
43      EIBOTH = 0x00;      // both polarity
44      EIFLAG = 0x00;      // clear all flags
```

Example: Single-source project

The `main.c` file in a multisource project contains the main function routine only.

You will see that the file contents are neatly separated.

```
1  //=====================================================
2  // Main program routine
3  // - Device name  : MC95FG308
4  // - Package type : 32QFN
5  //=====================================================
6
7  // - Generated    : Fri, Apr 26, 2013 (13:07:51)
8  #include    "MC95FG308.h"
9  #include    "func_def.h"
10
11 void main()
12 {
13     cli();              // disable INT. during peripheral setting
14     port_init();        // initialize ports
15     clock_init();       // initialize operation clock
16     BIT_init();         // initialize Basic interval timer
17     BUZ_init();         // initialize Buzzer
18     ExINT_init();       // initialize external interrupt
19     Timer0_init();      // initialize Timer0
20     UART_init();        // initialize UART interface
21     sei();              // enable INT.
22
23     // TODO: add your main code here
24
25     while(1);
26 }
27
```

Example: Multisource project

You can switch between a single- and a multisource project at any time.

4.2.3 interrupt.c

This file is optionally generated when you select a multisource project. This file contains the interrupt vector functions only. Vector functions are automatically sorted by their interrupt numbers. Each vector contains a comment that shows the interrupt source.

You will see that the file contents are neatly separated.

```
1  //=====================================================
2  // interrupt routines
3  //=====================================================
4
5  #include    "MC95FG308.h"
6  #include    "func_def.h"
7
8  void INT_USART0_Rx() interrupt 6
9  {
10     // USART0 Rx interrupt
11     // TODO: add your code here
12 }
13
14 void INT_I2C() interrupt 9
15 {
16     // I2C interrupt
17     // TODO: add your code here
18 }
19
20 void INT_BIT() interrupt 22
21 {
22     // BIT interrupt
23     // TODO: add your code here
24 }
25
```

Example: `interrupt.c`

4.2.4 peripheral.c

This file is optionally generated when you select a multisource project. This file contains MCU peripheral functions only. All functions are automatically sorted by their names. Each function contains the initialization code and comments.

You will see that the file contents are neatly separated.

```
21  void BUZ_init()
22  {
23      // initialize Buzzer
24      // Frequency (Hz) = 61.035156
25      BUZCR = 0x04;       // clock source
26      BUZDR = 0xFF;       // count value
27  }
28
29  void ExINT_init()
30  {
31      // initialize external interrupt
32      EIEDGE = 0x00;      // level / edge
33      EIPOLA = 0x00;      // polarity
34      EIBOTH = 0x00;      // both polarity
35      EIFLAG = 0x00;      // clear all flags
36      IEN0 |= 0x00;       // INT. 3,2,1,0
37      IEN4 |= 0x00;       // INT. 4,5
38      IEN5 |= 0x00;       // INT. 7,6
39      EIENAB = 0x00;      // enable INT pin
40  }
41
42  void I2C_init()
43  {
44      // initialize I2C interface
45      // Master : Frequency(Hz) = 3906.250000
46      // High duty = 49.902344(%)
47      I2CSCLLR = 0x80;    // low count
48      I2CSCLHR = 0x7F;    // high count
49      I2CMR = 0x7C;       // setting
50      IEN1 |= 0x80;       // Enable I2C interrupt
51  }
```

Example: `peripheral.c`