INTEGRATING TASKING 8051 TOOLS INTO THE SILICON LABS IDE

1. Introduction

This application note describes how to integrate the Tasking 8051 Tools into the Silicon Laboratories Integrated Development Environment (IDE). Integration provides an efficient development environment with compose, edit, build, download and debug operations integrated in the same program.

2. Key Points

- The Intel OMF-51 absolute object file generated by the Tasking 8051 tools enables source-level debug from the Silicon Labs IDE.
- After Tasking Tools are integrated into the IDE they are called by simply pressing the 'Assemble/Compile Current File' button or the 'Build/Make Project' button.
- See the included software, AN126SW, for an example using the Tasking tools.
- Information in this application note applies to Version 3.20 and later of the Silicon Labs IDE and V7.2 of the Tasking 8051 tools.

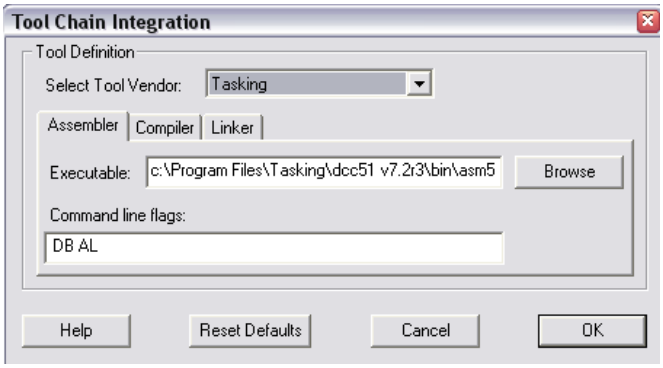
3. Create a Project in the Silicon Labs IDE

A project is necessary in order to link assembly files created by the compiler and build an absolute 'OMF-51' output file. Follow these steps to create a project:

1. Under the 'Project' menu, select 'Add Files to Project . . .' Select the 'C' source files that you want to add and click 'Open.' Continue adding files until all project files have been added.
2. To add files to the build process, right-click on the file name in the 'Project Window' and select 'Add *filename* to build.'
3. Under the 'Project' menu, select 'Save Project As . . .' Enter a project workspace name and click 'Save.'

4. Configure the Tool Chain Integration Dialog

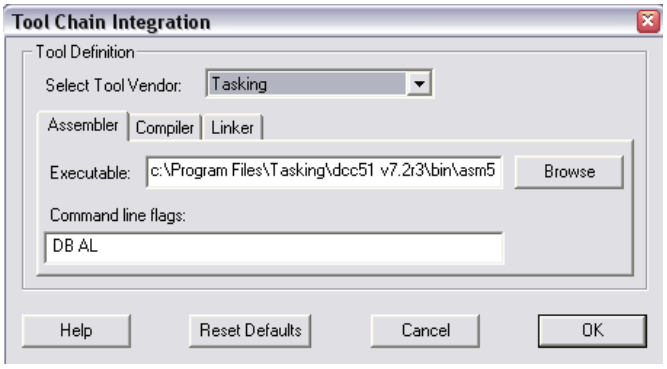
Under the 'Project' menu select 'Tool Chain Integration' to bring up the dialog box shown below. First, select 'Tasking' from the 'Select Tool Vendor' drop down list. Next, define the Tasking assembler, compiler, and linker as shown in the following sections.



4.1. Assembler Definition

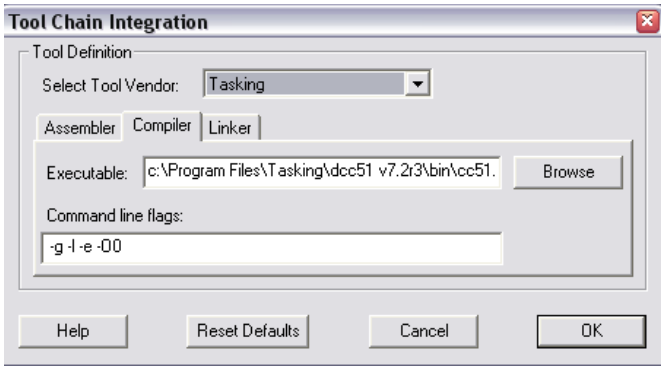
1. Under the 'Assembler' tab, if the assembler executable is not already defined, click the browse button next to the 'Executable:' text box and locate the assembler executable. The default location for the Tasking assembler is:

   "C:\Program Files\TASKING\dcc51 v7.2r3\bin\asm51.exe."
2. Enter any additional command line flags directly in the 'Command Line Flags' box.
3. See the following figure for the 'Assembler' tab with the default Tasking settings.



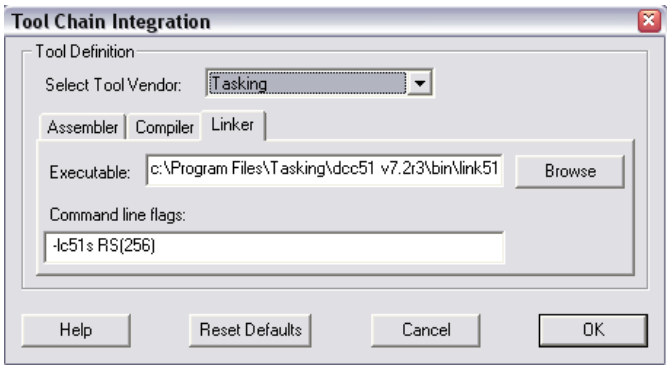
4.2. Compiler Definition

1. Under the ‘Compiler’ tab, if the compiler executable is not already defined, click the browse button next to the ‘Executable:’ text box, and locate the compiler executable. The default location for the Tasking compiler is “C:\Program Files\TASKING\dcc51 v7.2r3\bin\cc51.exe.”
2. Enter any additional command line flags directly in the ‘Command Line Flags’ box.
3. See the following figure for the ‘Compiler’ tab with the default Tasking settings.



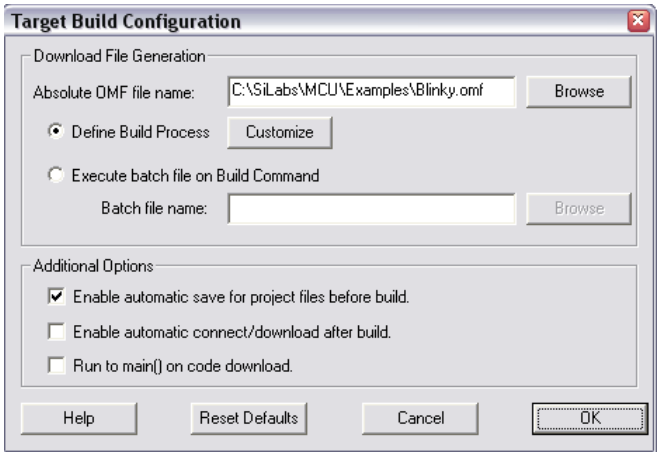
4.3. Linker Definition

1. Under the ‘Linker’ tab, if the linker executable is not already defined, click the browse button next to the ‘Executable:’ text box, and locate the linker executable. The default location for the Tasking linker is “C:\Program Files\TASKING\dcc51 v7.2r3\bin\link51.exe”.
2. Enter any additional command line flags directly in the ‘Command line flags’ box.
3. See the following figure for the ‘Linker’ tab with the default Tasking settings.



5. Target Build Configuration

Under the ‘Project’ menu select ‘Target Build Configuration’ to bring up the dialog box shown below.

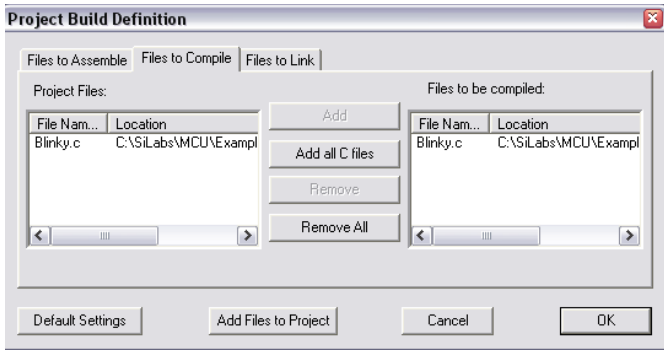


5.1. Output Filename

To customize a default filename or to create a new filename, click the browse button next to the ‘Absolute OMF file name:’ edit box. Select a path and enter an output filename with a ‘.omf’ extension (ex. blinky.omf).

5.2. Project Build Definition

Click the ‘Customize’ button to bring up the ‘Project Build Definition’ window shown below. This window allows selection of the files to be included in the build process. Although default assemble and compile selections will be made, ensure that all files have been correctly included in the build process. Under each tab, add Files to assemble or compile by selecting the desired file and clicking the ‘Add’ button. Files are removed in the same manner.



5.3. Additional Options

1. If the ‘Enable automatic save for project files before build’ box is checked, then all files included in the project will be automatically saved when the ‘Build/Make project’ button is pressed.
2. If the ‘Enable automatic connect/download after build’ box is checked, then the project will be automatically downloaded to the target board when the ‘Build/Make project’ button is pressed.
3. If the “Run to main() on code download” box is checked, the target board will halt at the first line in main() when the “Download code” button is pressed.

6. Building the Project

See the included software, AN126SW, for an example file created for use with the Tasking compiler, Blinky.c.

1. After saving all files that have been edited, the previous revisions will be saved in backup files. Backups are saved as the name of the file with the extension #1, #2, #3, and so on up to the number of backups (N) created and available, with ‘#1’ being the most recent and ‘N’ being the least recent.
2. Click the ‘Assemble/Compile current file’ button to compile just the current file.
3. Click the ‘Build/Make project’ button to compile and link all the files in the project.
4. Review the errors and warnings generated during the build process located in the ‘Build’ tab of the Output window (typically found at the bottom of the screen). Carefully viewing the entire length of the build report is recommended, due to the fact that the Tasking linker will continue to link even when there are compiler errors. Double-clicking on an error that is associated with a line number will automatically move the cursor to the proper line number in the source file that generated the error.

7. Source File Example

```
//-----------------------------------------------------------------------------
// Blinky.c
//-----------------------------------------------------------------------------
// Copyright 2002 Cygnal Integrated Products, Inc.
//
// AUTH: HF
// DATE: 10 DEC 02
//
// This program flashes the green LED on the C8051F020 target board about five times
// a second using the interrupt handler for Timer3.
//
// Target: C8051F02x
//
// Tool chain: Tasking 'C' Compiler
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include "regc51f02x.sfr"                  // SFR declarations

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------
#define SYSCLK 2000000                     // approximate SYSCLK frequency in kHz

_sfrbit LED _atbit(P1, 6);                 // green LED: '1' = ON; '0' = OFF;

//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------
void PORT_Init (void);
void Timer3_Init (int counts);
_interrupt(14) void Timer3_ISR (void);


//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main (void)
{
   // disable watchdog timer
   WDTCN = 0xde;
   WDTCN = 0xad;

   PORT_Init ();
   Timer3_Init (SYSCLK / 12 / 10);// Init Timer3 to generate interrupts
                                  // at a ~10Hz rate.
   EA = 1;                        // enable global interrupts

   while (1);                     // spin forever
}
```

```
//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
   XBR2     = 0x40;    // Enable crossbar and weak pull-ups
   P1MDOUT |= 0x40;    // enable P1.6 (LED) as push-pull output
}

//-----------------------------------------------------------------------------
// Timer3_Init
//-----------------------------------------------------------------------------
//
// Configure Timer3 to auto-reload and generate an interrupt at interval
// specified by <counts> using SYSCLK/12 as its time base.
//
void Timer3_Init (int counts)
{
   TMR3CN = 0x00;                          // Stop Timer3; Clear TF3;
                                           // use SYSCLK/12 as timebase

   TMR3RLL  = (-counts);// Init reload values
   TMR3RLH  = (-counts >> 8);// Init reload values

   TMR3L    = 0xff;                        // set to reload immediately
   TMR3H    = 0xff;

   EIE2    |= 0x01;                        // enable Timer3 interrupts
   TMR3CN  |= 0x04;                        // start Timer3
}

//-----------------------------------------------------------------------------
// Interrupt Service Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Timer3_ISR
//-----------------------------------------------------------------------------
// This routine changes the state of the LED whenever Timer3 overflows.
//
_interrupt(14) void Timer3_ISR(void)
{
   TMR3CN &= ~(0x80);                      // clear TF3
   LED = ~LED;                             // change state of LED
}
```

8. Include File Example

Tasking provides include files for several Silicon Labs device families with the installation of their tools. For the C8051F020 device you would use the regc51f02x.sfr file, located by default in the “C:\Program Files\TASKING\dcc51 v7.2r3\include” directory.

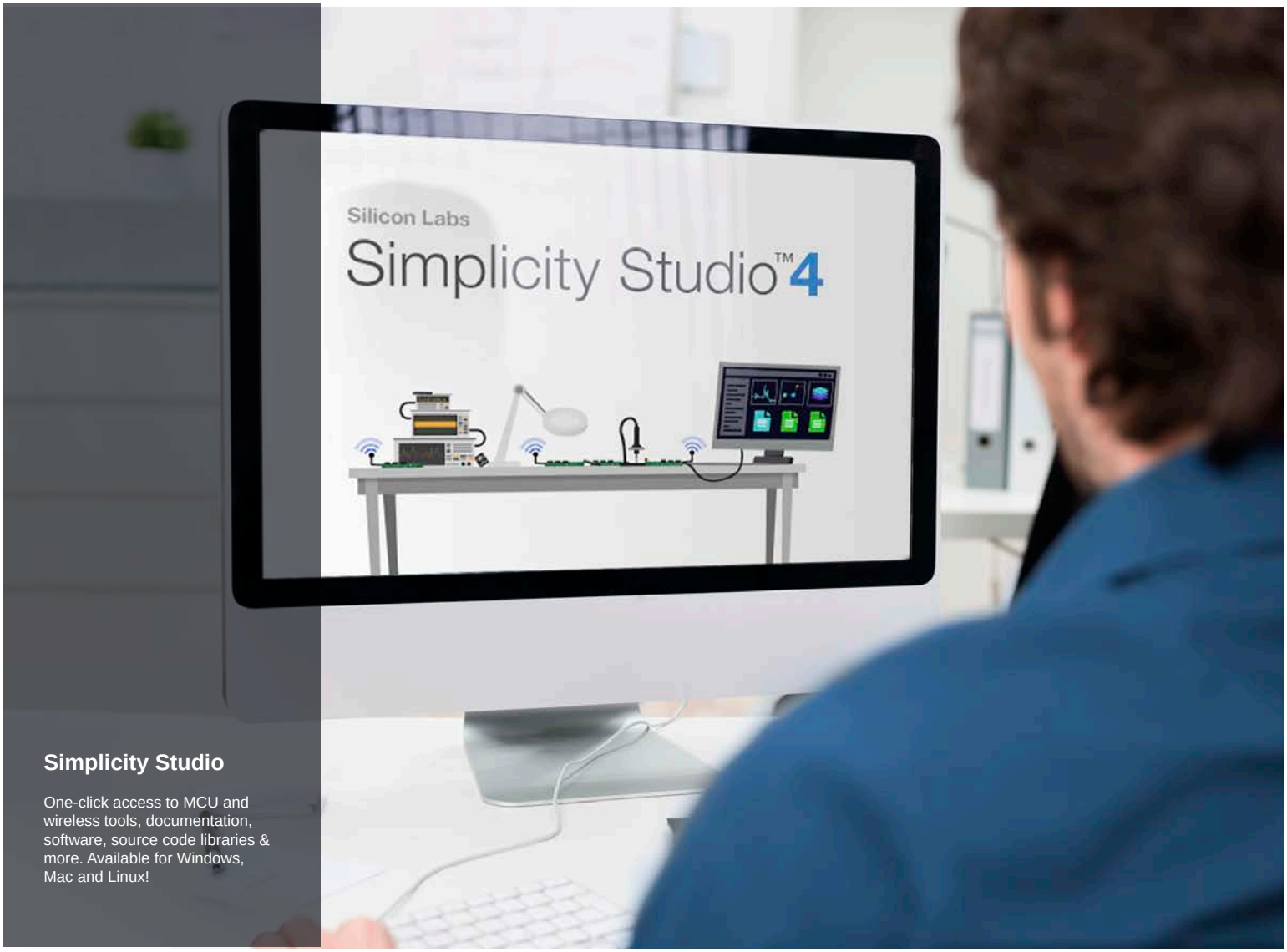
Document Change List

Revision 2.3 to 2.4

- Updated document path from C:\Program Files\TASKING\dcc51 v7.1\ to C:\Program Files\TASKING\dcc51 v7.2r3\

Revision 2.2 to Revision 2.3

- Introduction updated.
- Example path updated from C:\Cygnal\Examples to C:\Silabs\MCU\Examples.
- Target Build Configuration and Project Build Definition windows screenshots updated to reflect the new examples path.
- Tool Chain Integration windows screenshots updated for new Tasking settings.
- Key Points updated to include Silicon Labs and Tasking tools version information.
- Default Tasking tools directory updated to support v7.1.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com