INTEGRATING IAR 8051 TOOLS INTO THE SILICON LABORATORIES IDE

1. Introduction

This application note describes how to integrate the IAR 8051 Tools into the Silicon Laboratories IDE (Integrated Development Environment). 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 IAR 8051 tools enables source-level debug from the Silicon Laboratories IDE.
- Once IAR 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, AN236SW, for an example using the IAR tools.
- Information in this application note applies to Version 2.90 and later of the Silicon Labs IDE and Version 4.05 and later of the IAR 8051 Tools.

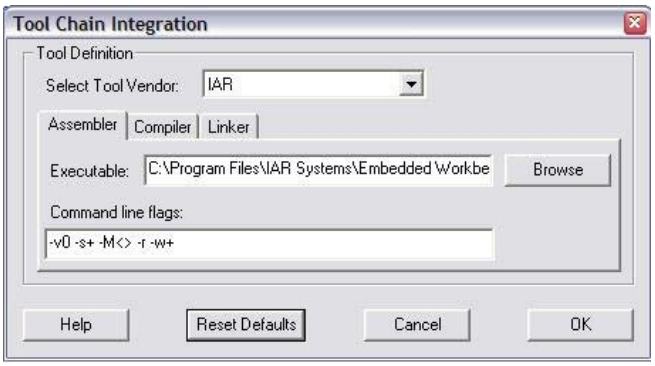
3. Create a Project in the Silicon Laboratories IDE

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 work space 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 “IAR” from the “Select Tool Vendor” drop-down list. Next, define the IAR assembler, compiler, and linker as shown in the following sections.



The IDE will first look in the registry for the installed path to the IAR Embedded Workbench's assembler, compiler, and linker; otherwise, the default locations are as follows:

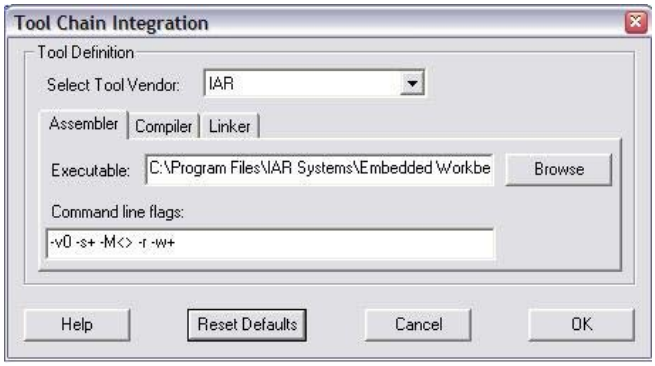
assembler: “C:\Program Files\IAR Systems\Embedded Workbench 4.05\8051\bin\a8051.exe”

compiler: “C:\Program Files\IAR Systems\Embedded Workbench 4.05\8051\bin\icc8051.exe”

linker: “C:\Program Files\IAR Systems\Embedded Workbench 4.05\common\bin\xlink.exe”

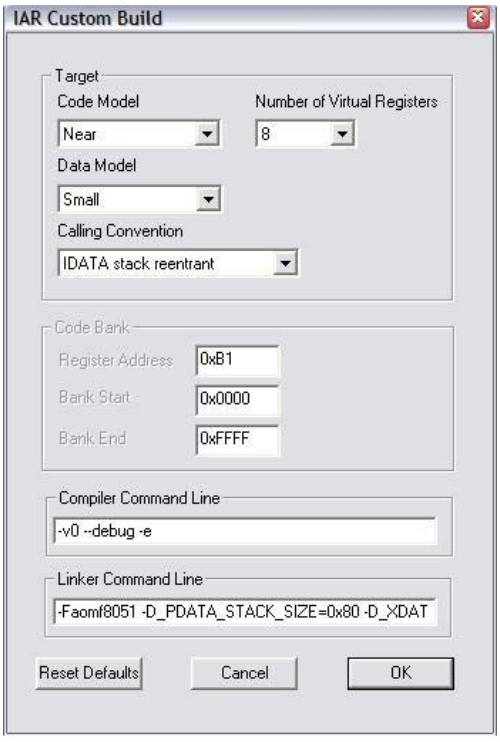
4.1. Assembler Definition

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.
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 IAR settings.

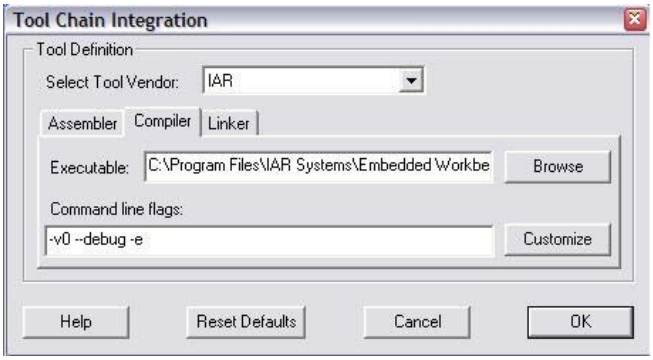


4.2. Compiler Definition

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.
2. Enter any additional command line flags directly in the “Command Line Flags” box.
3. On the IAR Compiler tab, there is a Customize button. The user can select this button for the IAR Custom Build dialog to appear. The user can select options from this dialog to help customize the build of their project. The compiler and linker command line will update as the selections are made.

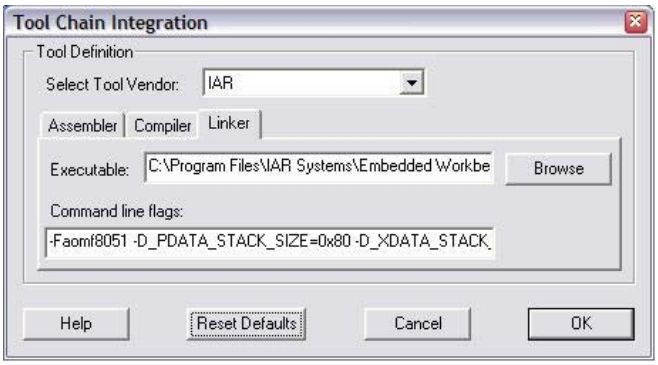


4. See the following figure for the “Compiler” tab with the default IAR settings.



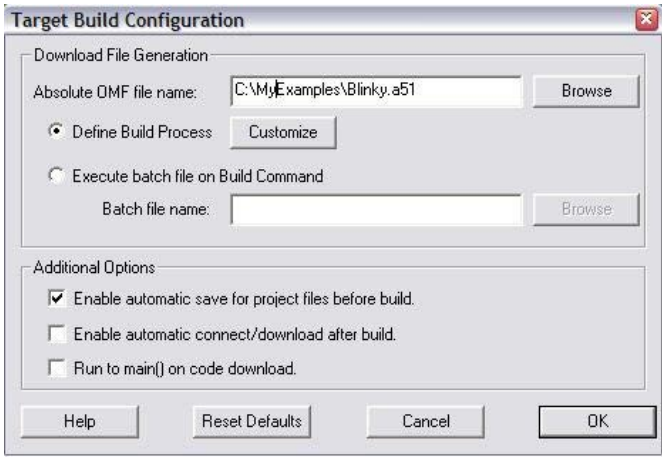
4.3. Linker Definition

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.
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 IAR settings.



5. Target Build Configuration

Under the “Project” menu, select “Target Build Configuration” to bring up the dialog box shown below.

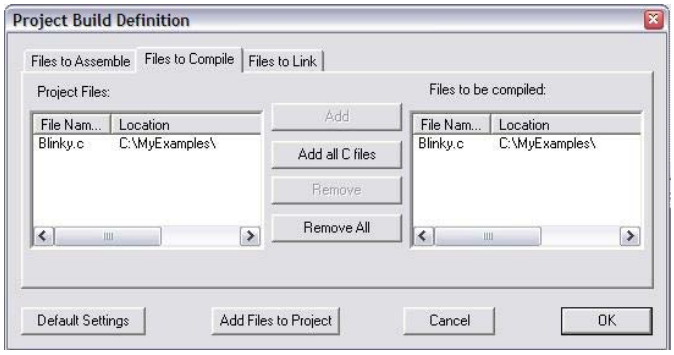


5.1. Output Filename

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 “.a51” extension (e.g., blinky.a51).

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, 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, 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

Blinky.c is the one example included with the IDE. The header files are IAR header files and can be found in the "..\IAR Systems\Embedded Workbench 4.05\8051\inc" folder.

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. “#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). 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. IAR Considerations

The considerations for IAR are as follows:

- Silicon Laboratories header files will not compile if they are included in the project. The project must use IAR header files for the specific target board.
- The compiler generates an “r51” extension object file. (ex. Blink.r51).

Once you have updated your Embedded Workbench tools to Version 7.10 or greater, you will need to override the configuration file named lnk51ew.xcl located on your computer in the folder "..\IAR Systems\Embedded Workbench 4.05\8051\config”. The updated file is included with the AN236SW software example.

8. Source File Example

```
//-----------------------------------------------------------------------------
// Blinky.c
//-----------------------------------------------------------------------------
// Copyright 2005 Silicon Laboratories, Inc.
//
// AUTH: SH
// DATE: 18 MAY 05
//
// This program flashes the green LED on the C8051F124 target board
// using the interrupt handler for Timer3.
// Target: C8051F12x
//
// Tool chain: IAR 'C' Compiler
//


//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <ioC8051F124.h> //IAR include file

//-----------------------------------------------------------------------------
// 16-bit SFR Definitions for 'F12x
//-----------------------------------------------------------------------------
unsigned char RCAP3   = 0xCA;                 // Timer3 reload value
unsigned char TMR3     = 0xCC;                 // Timer3 counter

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------
#define SYSCLK 3062500                    // approximate SYSCLK frequency in Hz

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

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

   SFRPAGE = 0x0F;                    // Switch to configuration page
   PORT_Init ();

   SFRPAGE = 0x01;                    // Switch to Timer 3 page
   Timer3_Init (SYSCLK / 12 / 10);    // Init Timer3 to generate interrupts
                                     // at a 10 Hz rate.
   IE =  0x90;

   SFRPAGE = 0x00;                   // Page to sit in for now
```

```
   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
   RCAP3   = -counts;                 // Init reload values
   TMR3    = 0xff;                    // set to reload immediately
   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.
//
// NOTE: The SFRPAGE register will automatically be switched to the Timer 3 Page
// When an interrupt occurs.  SFRPAGE will return to its previous setting on exit
// from this routine.
#pragma vector=0x73
__interrupt void Timer3_ISR (void)
{
   TMR3CN &= ~(0x80);                  // clear TF3
   P1 = ~P1;                           // change state of LED
}
```

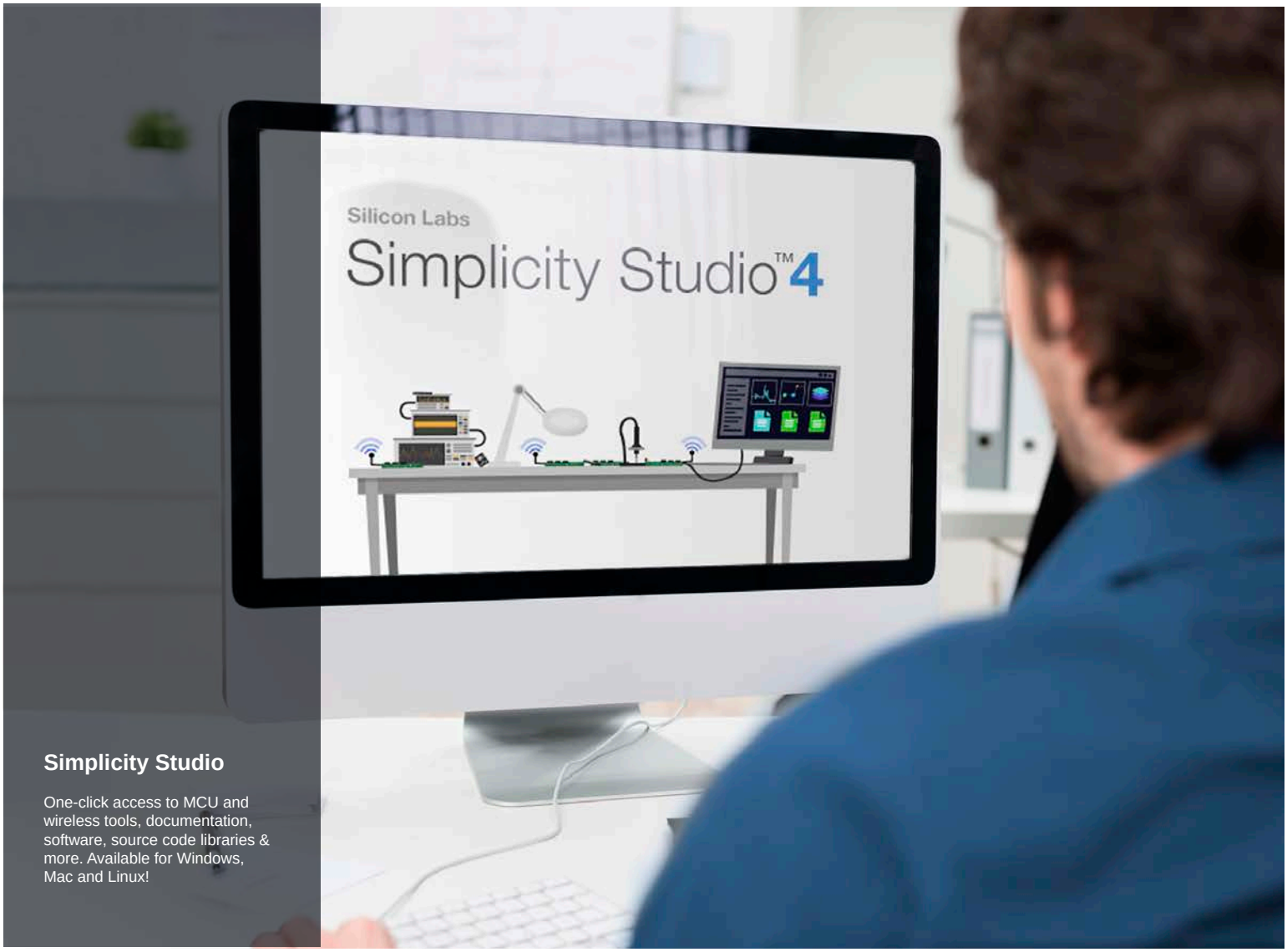
Document Change List

Revision 0.1 to Revision 0.2

- Updated "7. IAR Considerations" on page 4.

Revision 0.2 to Revision 0.3

- Updated paths to IAR tools to support Version 4.05.
- Updated supported version of the IDE to Version 2.90.
- Added bullet on front page to specify supported toolset versions.



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