# Stellaris® Development and Evaluation Kits for Keil<sup>™</sup> RealView® MDK

The Stellaris Development and Evaluation Kits provide a low-cost way to start designing with Stellaris microcontrollers using Keil's RealView Microcontroller Development Kit (MDK) for ARM controllers. The boards can function as either a complete evaluation target or as a debugger interface to any external Stellaris device.

#### Requirements

- You have a PC with a USB interface, running Microsoft® Windows 2000, XP, or Vista
- You have the Stellaris Evaluation Kit Documentation and Software CD or the standalone ARM/Keil CD found in the Development Kit



**CAUTION:** There is a known electrical issue with the FT2232 device that is used in the on-board In Circuit Debug Interface (ICDI). Some USB hubs can cause the device to misbehave, with symptoms ranging from failed enumeration to corrupt data transfers. If you experience trouble when using the on-board ICDI, try connecting the USB cable directly to one of the USB ports on your PC or laptop.

### Keil<sup>™</sup> RealView® MDK

This quickstart shows you how to install the evaluation version of the Keil RealView MDK, and then how to use it to build and run an example application on your Stellaris Evaluation Board.

### Step 1: Install the RealView MDK Tools

- 1. Insert the Evaluation Kit Documentation and Software CD or the standalone ARM/Keil CD into the CD-ROM drive of your computer. If Autoplay is enabled on your PC, the index.htm file automatically opens in your default web browser. If not, use Windows Explorer to open it manually.
- 2. With the Evaluation Kit CD, click the Tools button and then the Keil Logo to download a zip file containing the installer. If you prefer to run the setup executable from the CD, use Windows Explorer to view the files on the CD and double-click the MDKxxx.exe file located in the Tools\Keil\ directory.

With the standalone CD, follow the installer dialog.

#### Step 2: Install the StellarisWare® Package

A full set of C-based peripheral drivers is provided, covering all peripherals and functionality of the Stellaris devices. The StellarisWare package includes various example applications with project files for all major tool vendors that support Stellaris, including Keil. To install StellarisWare, follow these steps:



1. Navigate to the *Tools* tab on the Evaluation Kit Documentation and Software CD, or to the *Software* tab on the Development Kit Documentation and Software CD.

**NOTE:** If you are navigating the CD using Windows Explorer (or a similar application), go to the Tools/StellarisWare or Software/StellarisWare directories.

- Click on the 'Install' link next in the StellarisWare section (under Tools or Software) of the CD and run the StellarisWare installer. If you prefer to manually install StellarisWare, the installer is a self-extracting zip file that is located in the Tools/StellarisWare directory. You can use a zip file extraction utility such as WinZip to manually extract the contents.
- 3. To view the StellarisWare documentation, navigate to the installation directory and click on the *Stellaris Peripheral Driver Library User's Guide* PDF.

#### Step 3: Start the Keil µVision IDE and Load the Hello Project

1. Start the Keil  $\mu$ Vision IDE by double-clicking the icon on your desktop or by selecting it from the Windows Start Menu.





2. From the Project menu, select "Open Project."

🕎 µVision3					
<u>File E</u> dit <u>V</u> iew	<u>P</u> roj	ect Debug Flash Peripherals Tools SVCS Window !			
🐴 🚘 🗖 🗗		New Project			
		Import µVision1 Project			
I 🍩 🕮 🏧 🛣		Open Project			
Project Workspace		<u>Close</u> Project			
	♣	🐣 Components, Environment, Books			
		Select Device for Target 'Development Board'			
		Remove Item			
		Options for Target 'Development Board'			

3. Use the dialog box to navigate to the Hello program in the directory appropriate for your board. From the location where you installed StellarisWare, the Hello project is located in:

StellarisWare\boards\{board\_name}\hello

The Keil tools also install StellarisWare as part of the default installation, but the version may be older than what is currently available in the development or evaluation kit. You can find StellarisWare in the Keil tree by looking in:

C:\Keil\ARM\Boards\Luminary\{board\_name}

**Important:** For the most recent version of the StellarisWare example projects, use the standalone version found on the Evaluation Kit Documentation and Software CD or check <u>www.ti.com/stellaris</u> for the latest software updates.

4. Select the hello.uvproj project file and click Open. The project opens in the IDE.





5. You can view source files in the project by double-clicking on a filename in the Project Workspace pane on the left. For example, double-click on hello.c, and the source file opens in the editor.





#### Step 4: Build the Hello Project

1. Select "Rebuild all target files" from the Project menu, or click on the "Rebuild all" button (icon).

🌄 hello 🕞 µVisi	on3					
<u>File E</u> dit <u>V</u> iew	Proje	ect Debug Flash Peripherals Tools SVCS Window Help				
1 🖀 🚘 🗖 🗗		New Project				
		Import µVision1 Project				
I 🌣 🏛 🎬 🖌		Open Project				
Project Workspace		<u>C</u> lose Project				
🖃 🛅 Developm			<u>_</u>			
🖃 📇 Starti	₽	Components, Environment, Books				
Final Association		Select Device for Target 'Development Board'				
🚊 🛄 H		R <u>e</u> move File 'hello.c' fo				
🕀 🖓 🗄 🗄		Options for File 'hello.c'				
E 📩 L			h			
Endar Elbrar		<u>B</u> uild target	F7 Mi			
🖃 🔄 Docu		<u>R</u> ebuild all target files				
🛄 r	٩	Translate C:\FIRMWARE\DriverLib\examples\power\power.c				
	X	Stop b <u>u</u> ild				
			B;			

2. All of the source files are compiled and linked. The activity can be seen in the Build window at the bottom of the  $\mu$ Vision IDE. The process completes with an application named hello.axf built with no errors and no warnings.





#### Step 5: Load the Hello Program into the Flash Memory

**Important:** The FTDI driver interface in the kit is slower than a standard ULINK interface, and is intended to be used for tool evaluation. Upgrading to a ULINK for your custom design allows for much faster download speeds. The ULINK *cannot* be used with the EKK-LM3S811, but is compatible with all other evaluation boards.

1. Select "Download" from the Flash menu, or click on the "Download" button (icon).



- 2. The process takes a few seconds. You see a progress bar at the bottom of the IDE window as the device is programmed. When it is finished, you should see in the Build window that the device was erased, programmed, and verified OK.
- 3. The Hello application is now programmed into the flash memory of the Stellaris microcontroller on the Evaluation Board.



#### Step 6: Debug and Run the Hello Program

1. Select "Start/Stop Debug Session" from the Debug menu or click the "Debug" button (icon).



2. The IDE switches to debugging mode. The processor registers show up in a window on the left, the debugger command window is visible at the bottom, and the main window shows the source code being debugged. The debugger automatically stops at main.

🕎 hello - µVision3 - [C:\Keil\ARM\RV30	Boards\Luminary\DK-LM3S301\hello\hello.c]	
Eile Edit View Project Debug Flash Pe	ipherals Iools SVCS Window Help	_ & ×
1 🏠 😅 🖬 🍠   X 🖻 🛍   으 으   1	■幸ん%%%%% ● ● ● ● ● ● ● ● ● ● ● ● ●	
i 💒 i 💷 🛛 🗗 🗗 🖓 🗮 🔀	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!</td <td></td>	
Project Workspace 🗸 🛪	R1 int	
Register Value	62 main (void)	<u> </u>
- Core	<> 63 □ (	
R0 0x20000138	64 //	
R1 0x0000000	65 // Set the clocking to run directly from the crystal.	
R2 0x20000060		
R3 0x20000060	co systeticiockset(siscil_sistitu_1 siscil_ost_ost siscil_ost_main	
R4 Ux20000010	siscin_rial_onic),	
RC 0v000000	70 //	
BZ 0x0000000	71 // Init the PDC, and then the LCD, then write the LEDs.	
R8 0x0000000	72 //	
R9 0x200000a0	73 PDCInit();	
R10 0x000008d8	74 PDCLCDInit();	
R11 0x0000000	75 PDCLCDBacklighton();	
R12 0x20000040		
R13 (SP) Ux20000160	78 // Hello/	
B15 (PC) 0v00000433	79 //	
PSB 0x4100000	80 PDCLCDSetPos(0, 0);	
	81 PDCLCDWrite("Hello World!", 12);	
	82	
🖻 Internal	83 //	
Mode Thread	84 // Ex2t.	
Privilege Privileged	80 // 96 DiacEvit(0):	-
Stack MSP		•
	readme.txt i hello.c	
Load "C:\\Keil\\ARM\\RV30	Name	
3		
₽ >		
ASSIGN BreakDisable		
Build Command / Fir	Memory #1 / Memory #2 / Memory #3 / Memory #4 /	als $\langle Watch \#1 \rangle W$
Ready	ULINK Cortex-M3 Debugger  t1: 0.00000000 sec  L:63 C:1  NUM	I R/W



3. From here, you can examine and modify memory, program variables and processor registers, set breakpoints, single step, and all other typical debugging activities. To run the program, select "Run" from the Debug menu, or click on the "Run" button (icon).



4. The application starts running, and you should see the text "Hello World!" output to the OLED display of the evaluation board.

#### Step 7: Build and Run Additional Example Programs

There are several additional example project folders under the folder:

```
StellarisWare\boards\{board_name}
```

These additional example programs are discussed in the *StellarisWare Peripheral Driver Library User's Guide* (found on the Evaluation Kit Documentation and Software CD). You can open, build, and run any of these in the same way by going back to Step 4 and opening a different project, as long as it fits within the 32 KB code limit of the evaluation tools. All projects larger than 32 KB have binaries available that can be downloaded with the LMFLASH utility.

### **Creating a New Project**

Once you have gone through the StellarisWare example applications, you may want to create your own project to start development. While you can always start with an existing, simple project, sometimes you may want to start fresh.

To create a new project, in project menu, select Project > New uVision Project...



Vision3		
Eile Edit View	Project Debug Flash Peripherals Tools SVCS Window Help New µVision Project New Project Workspace Import µVision 1 Project	
Project Workspace	Open Project Close Project	

You then get a dialog box asking you where you would like to save your new project. While you can put the new project anywhere, if you plan to use the StellarisWare framework, you can create the project within the StellarisWare tree. You can put it in either the existing StellarisWare\boards directory, or create a new item in the boards directory that corresponds to either your specific board or development. In this case, we'll create a new directory called "my\_board" in StellarisWare boards and create a new project called "my\_project." The StellarisWare tree now looks like:

C:\StellarisWare\boards\my\_board\my\_project

Create New Pro	oject				? 🔀
Save <u>i</u> n:	my_project		•	🗢 🗈 💣 📰 •	
My Recent Documents Desktop					
My Documents					
My Computer					
My Network Places	File <u>n</u> ame: Save as <u>t</u> ype:	my_project Project Files (*.uvproj)		•	<u>S</u> ave Cancel

Once the project file (.uvproj) is saved, a dialog window pops up asking you to select the device that you are using. Select the appropriate device under the Luminary Micro or Texas Instruments list.



**Important:** A common question is "what if my device isn't in the list?" That's OK, don't worry. The device selection is mainly used to configure the memory sizes (linker). If you select a similar device to the one you have, it will work just fine.

Select Device for Target 'Target 1'	×
CPU Vendor: Actel Device: Toolset: Data base Description:	
Luminary Micro LM3S101 LM3S102 LM3S1120 LM3S1133 LM3S1138 LM3S1138 LM3S1165 LM3S1165 LM3S1165 LM3S1165 LM3S1165 LM3S122 LM3S1322 LM3S1332 LM3S1332 LM3S1435 LM3S1439 LM3S1512 LM3S1538 VINCOUNTRY	
OK Cancel	Help

The last thing that the tool asks is whether you want to add startup code to the project. You can click yes or no, but just know that you do need to have startup code whether it's the one provided by Keil or one that you get from somewhere else. For this example, we'll add the startup code provided by the tool.



Your empty project will look like this:





The startup code adds the vector table for you, so all you need to do is create a C file with a main function. For this example, create a new file (from File > New...) and add a simple main function with a while(1) loop; then save it as a \*.c file in your project directory. Once you have saved it, right click on the "Source Group 1" folder and select "Add Files to Group 'Source Group 1'." When the dialog box pops up to find the file, browse to your project directory and select the main.c file you just created.



This process gives you the most basic project that will run on a Stellaris microcontroller. The next steps include configuring your project for your specific board, adding the hooks for StellarisWare, and setting up the debug and flash programming environment.

#### Configuring for your hardware:

To set up the project for your specific board, go to Project > Options for Target 'Target 1.' The following dialog box appears:

Options for Target 'Target 1'					
Device Target Output Listing User C/C++ Asm Linker Debug Utilities					
Luminary Micro LM3S6965					
<u>≯</u> tal (MHz): <mark>5</mark>	0	_ Code Generation			
Operating system: None		Use Cross-Module Optimization			
		Use MicroLIB 🔽 Big Endian			
		Use Link-Time Code Generation			
Read/Only Memory Areas		Read/Write Memory Areas			
default off-chip Start Size	Startup	default off-chip Start Size Nolnit			
ROM1:	0	RAM1:			
ROM2:	0	□ RAM2: □			
ROM3:		RAM3:			
on-chip		on-chip			
IROM1: 0x0 0x40000	•	IRAM1: 0x20000000 0x10000 □			
IROM2:		□ IRAM2: □			
ок	Can	ancel Defaults Help			

Here you need to set up the Xtal frequency for your specific board. This step is important because the value of the Xtal field is used to calculate flash programming timing later on. Also check the ROM/RAM sizes to make sure that it corresponds to the device you have.

#### Setting up Debug:

In the same project options dialog shown above, select the Debug tab. The default configuration for a new project is to use the simulator, which is not going to allow you to debug real hardware. Choose the "Use" radio on the right side with the "ULINK Cortex Debugger" next to it (this selection is the default in the drop-down box). If you're using a Stellaris evaluation or development board as your in-circuit debug interface (ICDI), choose the "Luminary Eval Board" option from the drop-down box.



Options for Target 'Target 1'	
Device   Target   Output   Listing   User   C/C++   Asm	Linker Debug Utilities
Limit Speed to Real-Time	
Initialization File:	Initialization File:
Restore Debug Session Settings	Restore Debug Session Settings
CPU DLL: Parameter: SARMCM3.DLL	Driver DLL: Parameter:
Dialog DLL: Parameter: DLM.DLL -pLM3S6965	Dialog DLL: Parameter: TLM.DLL pLM3S6965
ОК Са	ncel Defaults Help

#### Setting up Flash Programming:

Just as you did for debug, you must select the appropriate debug interface for flash programming. Browse to the Utilities tab of the project options and select the "Luminary Eval Board" option from the drop-down box.



ptions for Tar	get 'Target 1'	×
Device   Target	Output Listing User C/C++ Asm Linker Debug Utilities	
Configure Flash	Menu Command	
Use Targe	t Driver for Flash Programming	
	Luminary Eval Board Settings IV Update Target before Debugging	
Init File:	Edit	
C Use Extern	al Tool for Flash Programming	
Command:		
Arguments:		
	🗖 Run Independent	
	OK Cancel Defaults Help	

With these settings complete, you will be able to download and debug your simple application.

#### Adding StellarisWare hooks:

The easiest way to see how to configure your project to use the StellarisWare drivers and utilities is to look at the examples. The basic things you'll need to do are:

- Add the driverlib.lib file to your project. This gives the application the entire driver source at compile/link time. To actually use any of the functions, you'll need to include the appropriate header files. To add driverlib.lib to your project, right click on the "Source Group 1" folder and select "Add Files to Group 'Source Group 1'." Browse to StellarisWare\driverlib\rvmdk and select the driverlib.lib file. Note, you need to tell the file browser to look for \*.lib file types, so change the "Files of type" drop-down from "C Source file (\*.c)" to "Library file (\*.lib)" or "All files".
- 2. In your project options, select the C/C++ tab. Here you need to tell the project that you're using the ARM compiler, so you should define "rvmdk." This definition is used in StellarisWare to compile the Keil specific sections correctly. You should also define the part you're using as well as add the top level StellarisWare directory to your include path.



Options for Target Target 1'	×
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Preprocessor Symbols	
Define: wmdk PART_LM3S6965	
Undefine:	
Language / Code Generation	
Strict ANSI C Warnings:	
Optimization: Level 0 (-00)	
Optimize for Time     Ime     Ime	
Split Load and Store Multiple	
One ELF Section per Function	
Paths	
Misc Controls	
control -DPART_LM3S6965 -o "t.o" -omf_browse "t.of" -depend "t.d"	
string	
OK Cancel Defaults Help	

3. In the linker section, you must tell the application where the entry point is, so add " entry Reset\_Handler" to the Miscellaneous controls box. If your startup code changes, you should adjust this accordingly.

The default linker configuration is to use an automatically generated linker/scatter file, which is fine for most people. If you want to use your own scatter file, you should uncheck the "Use Memory Layout from Target Dialog" and point the tool to a scatter file.



Options for Target 'Target 1'	
Device   Target   Output   Listing   User   C/C++   Asm	Linker Debug Utilities
<ul> <li>✓ Use Memory Layout from Target Dialog</li> <li>✓ Make RW Sections Position Independent</li> <li>✓ Make RO Sections Position Independent</li> <li>✓ Don't Search Standard Libraries</li> <li>✓ Report 'might fail' Conditions as Errors</li> </ul>	E/O Base:         0x0000000           R/W Base         0x20000000           disable Warnings:
Scatter File	Edit
Miscentry Reset_Handler	
Linker control string	t.sct"
ОК	Cancel Defaults Help

These steps get the basic StellarisWare hooks into your project. The best way to make sure you've done everything correctly is to go over some of the examples that we provide in the StellarisWare package.

### Conclusion

You have now installed the Keil RealView Microcontroller Development Kit and used it to build, load, and run a demonstration application on your Stellaris Evaluation Board. You have also learned how to create a new project from scratch. From here, you can experiment with the debugger or start creating your own application from the ground up or using the Hello program as an example.



### References

The following references are included on the Stellaris Evaluation Kit Documentation and Software CD and are also available for download at <u>www.ti.com/stellaris</u>:

- Stellaris Evaluation Kit User's Manual
- StellarisWare Software, Order Number SW-LM3S
- StellarisWare Peripheral Driver Library User's Guide, Order Number SW-DRL-UG

In addition, the following website may be useful:

RealView MDK website at <a href="http://www.keil.com/arm/rvmdkkit.asp">http://www.keil.com/arm/rvmdkkit.asp</a>

Copyright © 2006–2010 Texas Instruments, Inc. All rights reserved. Stellaris and StellarisWare are registered trademarks of Texas Instruments. ARM and Thumb are registered trademarks, and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as the property of others.

Texas Instruments 108 Wild Basin Rd., Suite 350 Austin, TX 78746 http://www.ti.com/stellaris



