Evaluation board for NXP LPC2103

USE GNU ARM UNDER KEIL IDE

INTRODUCTION

Evaluation version of RealView limits to 16KB of code size. However, the GNU ARM tools (compiler, assembler, and so on) that are provided at KEIL's web site are not limited or restricted in any way except that one must not use it for commercial product. For students, hobbyists, or amateur engineers, using free open source tools provide an attractive alternative.

There are quite a number of resources on this topic. To list a few of those:

1. **Yet another GNU ARM toolchain** at www.yagarto.de. This package uses free Eclipse as the Integrated Development Environment (IDE).
2. **Using Open Source Tools for AT91SAM7S Cross Development, Revision C** which is found at the Atmel web site http://www.atmel.com/dyn/products/allappnotes.asp?family_id=605
3. **GNU-Based Software Development on AT91SAM Microcontrollers** found at the same Atmel web site above.

It is not difficult to get completely free IDE and an ARM toolchain installed except that it just takes more time. However, because it is basically a D.I.Y. option, sometimes it would be rather difficult to get accurate and responsive product support.

There are commercial IDEs for GNU ARM toolchain, of course. Examples are:

1. Sourcery G++ provided by CODESOURCERY at http://www.codesourcery.com
2. CrossWorks for ARM by Rowley Associates. There is an option for personal license at only US$150. Web site at http://www.rowley.co.uk/arm/index.htm
3. RapidiTTy Builder from TTE Systems.
   Web site at http://www.tte-systems.com/products/builder
   There are a lot of useful tutorials available here, edited by the world-famous author Michael J. Pont (author of "Embedded C" and "Patterns for Time-Triggered Embedded Systems"). Besides, the lite version of RapidiTTy is available for download as well.

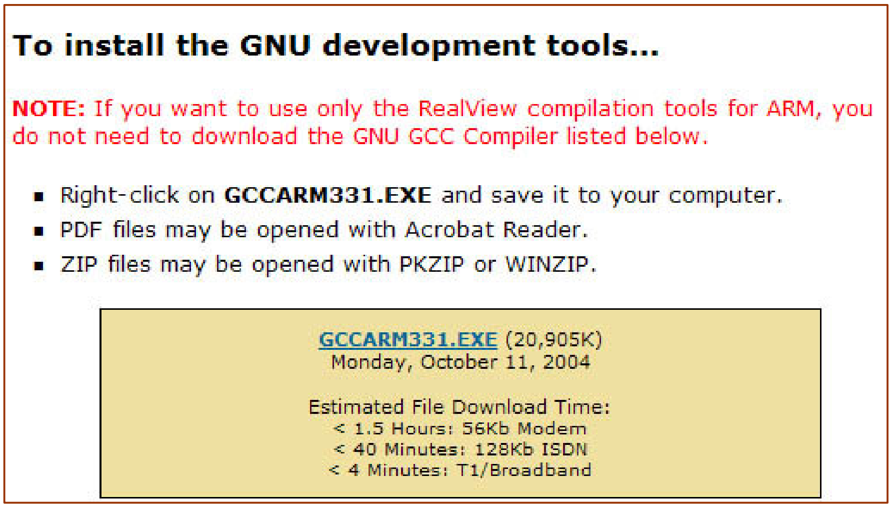
Because I have got used to µVision3 in the past, it is fairly natural to try GNU in the same IDE without having to learn a new tool. This article describes the procedure of how the GNU ARM tools can be integrated with KEIL's µVision3. A simple example based on the previous one for RealView has been provided but now, it is complied and linked by GNU ARM tools.

SOFTWARE

It is assumed that you have got the MDK322a from KEIL web site prior to installation of the GNU package. If not, please refer to a user guide (Doc 02) below for reference.

http://www.techtoys.com.hk/ARM_boards/LPC2103_Eval_1A/LPC2103_Eval_1A.htm

Under the same web site for MDK322a download, there is the GNU package.



To install the GNU development tools...

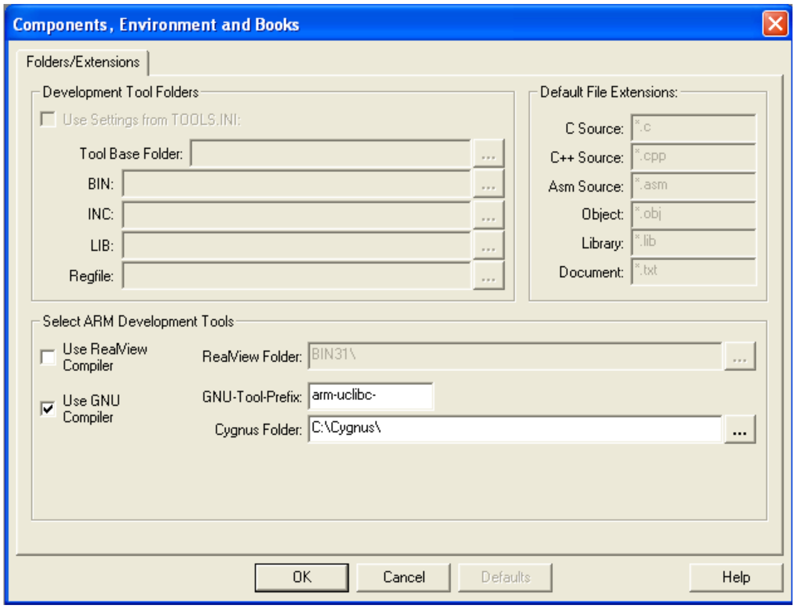
NOTE: If you want to use only the RealView compilation tools for ARM, you do not need to download the GNU GCC Compiler listed below.

- Right-click on **GCCARM331.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

GCCARM331.EXE (20,905K)
Monday, October 11, 2004

Estimated File Download Time:
< 1.5 Hours: 56Kb Modem
< 40 Minutes: 128Kb ISDN
< 4 Minutes: T1/Broadband

Having downloaded and installed GCCARM331.EXE, there will be the Cygnus folder under C:\Cygnus. Launch µVision3, click under the main menu ***Project→Manage→Components, Environment, Books*** to bring up the ARM Development Tools selection manual. Check ***Use GNU Compiler*** and fill-in ***arm-uclibc-*** in the GNU-Tool-Prefix text box and the path for GNU toolchain as ***C:\Cygnus\*** as below. Click ***OK*** to exit.



Components, Environment and Books

Folders/Extensions

Development Tool Folders

☐ Use Settings from TOOLS.INI:

Tool Base Folder:

BIN:

INC:

LIB:

Regfile:

Default File Extensions:

C Source: *.c

C++ Source: *.cpp

Asm Source: *.asm

Object: *.obj

Library: *.lib

Document: *.txt

Select ARM Development Tools

☐ Use RealView Compiler

RealView Folder: BIN31\

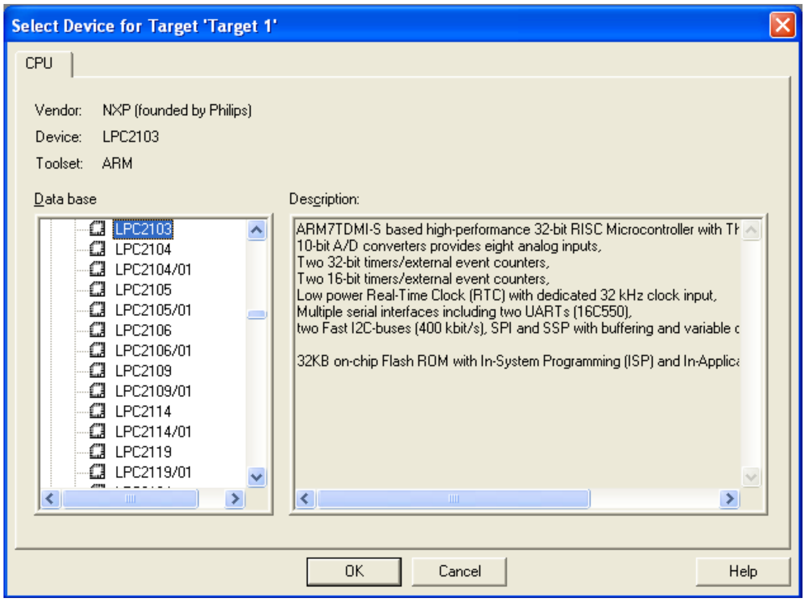
☑ Use GNU Compiler

GNU-Tool-Prefix: arm-uclibc-

Cygnus Folder: C:\Cygnus\

Create a new project from the main menu ***Project→ New µVision Project…***
The path of my project is D:\ARM\LPC2103\GNU\BlinkyLED\BlinkyLED.uv2.

From ***Device Manual***, select LPC2103.



Vendor: NXP (founded by Philips)
Device: LPC2103
Toolset: ARM

LPC2103
LPC2104
LPC2104/01
LPC2105
LPC2105/01
LPC2106
LPC2106/01
LPC2109
LPC2109/01
LPC2114
LPC2114/01
LPC2119
LPC2119/01

ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with Th
10-bit A/D converters provides eight analog inputs,
Two 32-bit timers/external event counters,
Two 16-bit timers/external event counters,
Low power Real-Time Clock (RTC) with dedicated 32 kHz clock input,
Multiple serial interfaces including two UARTs (16C550),
two Fast I2C-buses (400 kbit/s), SPI and SSP with buffering and variable c

32KB on-chip Flash ROM with In-System Programming (ISP) and In-Applica

Click ***Yes*** in answering to the message box asking **"Copy Philips LPC2100 Startup Code to Project Folder and Add File to Project?"**

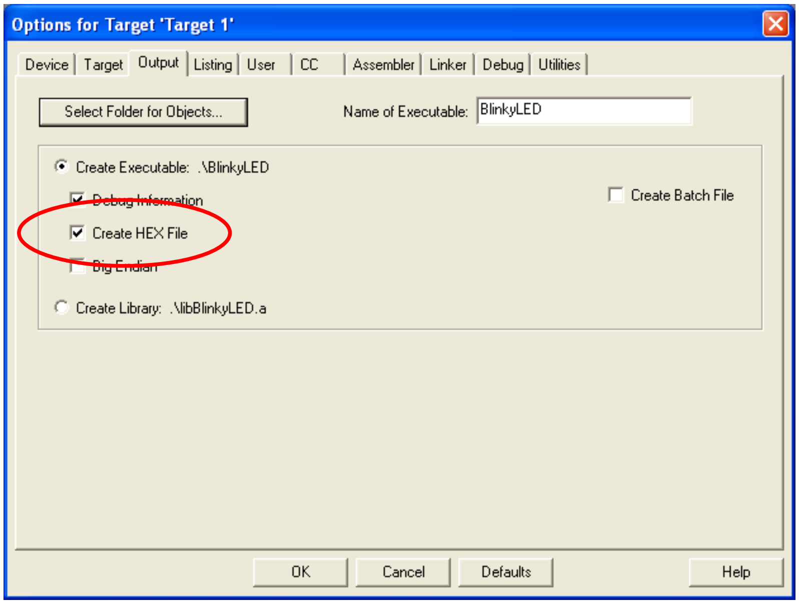
From the project workspace at the left, double-click to open the file Startup.s. Search for the key words .equ Top_Stack, 0x40004000, and then change the hex number 0x40004000 to 0x40002000 for LPC2103 microcontroller.

You need to do this procedure every time a new project is created because for some unknown reason, the original Startup.s file copied from Keil IDE has specified a Top_Stack of 0x40004000 which doesn't work with LPC2103. It is possible to answer No to the same question as above and manually copy the amended Startup.s file to the new project folder to solve this problem.

Right click on the project folder ***Target 1→Options for Target 'Target 1'*** to bring up the configuration tool.

Important configurations:

1. ***Please look at Device, Target, Output, and Linker Tabs***. Under the ***Device Tab***, it states that NXP LPC2103 microcontroller has been chosen.
2. ***Target Tab*** shows the Xtal frequency value. Because we are using an external crystal of 12MHz, we need to input 12.0 under Xtal text box.
3. Check the check-box ***Create HEX File*** under the ***Output Tab***.



Name of Executable: BlinkyLED

● Create Executable: .\BlinkyLED

☑ Debug Information

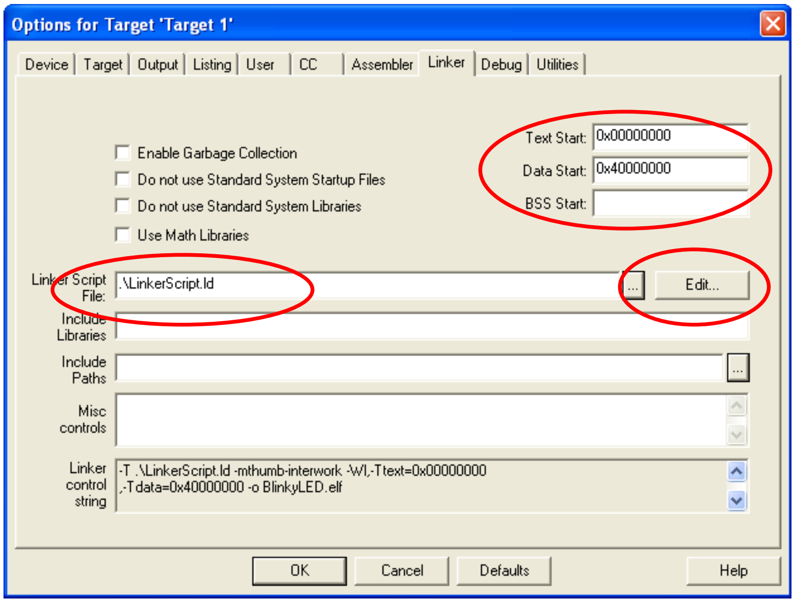
☑ Create HEX File

□ Big Endian

○ Create Library: .\libBlinkyLED.a

□ Create Batch File

4. Copy the linker script file LinkerScript.ld from C:\Keil\ARM\GNU\ to your project directory. This file provides information on how the project would be linked together.
5. Under the ***Linker Tab***, use the following parameters for ***Text Start***, ***Data Start***, and add linker script LinkerScript.ld from the project directory. Don't exit for now.



□ Enable Garbage Collection

□ Do not use Standard System Startup Files

□ Do not use Standard System Libraries

□ Use Math Libraries

Text Start: 0x00000000

Data Start: 0x40000000

BSS Start:

Linker Script File: .\LinkerScript.ld

Include Libraries:

Include Paths:

Misc controls:

Linker control string:

```
-T .\LinkerScript.ld -mthumb-interwork -Wl,-Ttext=0x00000000
,-Tdata=0x40000000 -o BlinkyLED.elf
```

6. Under ***Linker Tab***, click ***Edit*** for Linker Script file. Under memory definition, change the code (FLASH) size to 0x00008000 and data (RAM) size to 0x00002000 matching a FLASH space of 32KB and RAM of 8KB for LPC2103 microcontroller.

   ```
   MEMORY
   {
           CODE (rx) : ORIGIN = 0x00000000, LENGTH = 0x00008000
           DATA (rw) : ORIGIN = 0x40000000, LENGTH = 0x00002000
   }
   ```

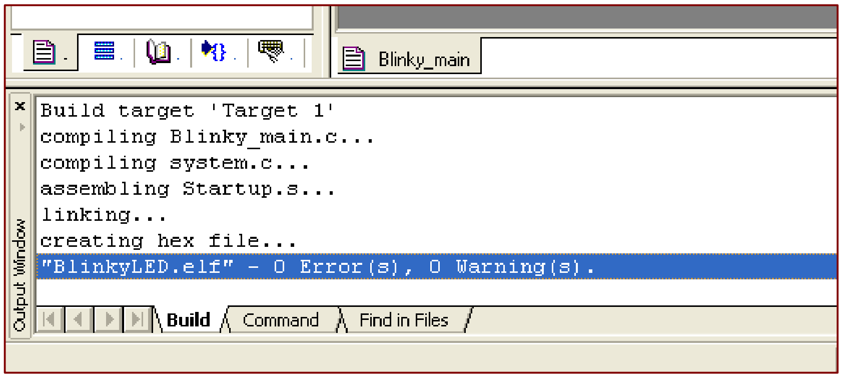
7. There remains to add source code to the project. We may use the same source code that has been developed for RealView complier for this GNU project. There is no modification required. It is available for download under Doc 04 at the following hyperlink.

   http://www.techtoys.com.hk/ARM_boards/LPC2103_Eval_1A/LPC2103_Eval_1A.htm

   Copy those two folders "System" and "main" to the project folder of the GNU project.

   It is important to note that no space is allowed for GNU toolchain. That means if the file name of a source file was like, Blinky LED.c, the project would not be compiled.

8. Finally add the files Blink_main.c and system.c to the project by right click on ***Source Group 1 folder → Add Files to Group 'Source Group1'.*** Compile and link by ***Rebuild All Target Files***. Make sure there is no error from the Output Window as below



```
Build target 'Target 1'
compiling Blinky_main.c...
compiling system.c...
assembling Startup.s...
linking...
creating hex file...
"BlinkyLED.elf" - 0 Error(s), 0 Warning(s).
```

Follow the same procedure to download the hex code generated. You may use LPC2000 FLASH UTITLIY or FlashMagic for this job. Details have been given under the user guide for LPC2103 Evaluation Board so I am not repeating here.

RESOURCES

1. Discussion forum at http://www.keil.com/forum/docs/thread9017.asp