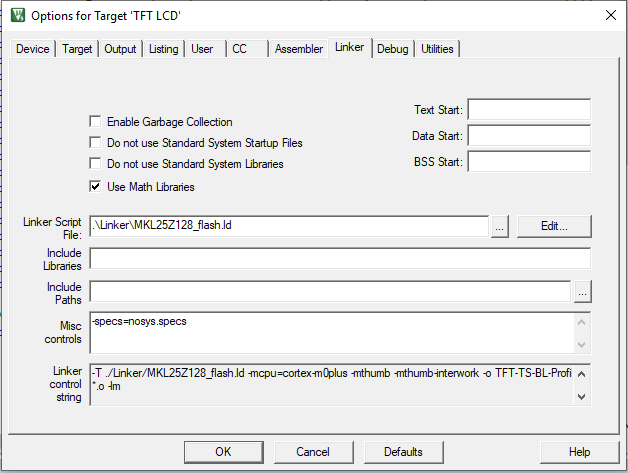
* Get GNU Arm Embedded Toolchain
  + <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm>
  + Download from <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads> - get Win32-bit version …SHA2.exe, signed for Win 7 and later)
  + Install it. On last page (before “Finish”) select “Add path to environment variable”
* Configure MDK-ARM to use it
  + <http://www.keil.com/arm/gnu.asp>
  + <http://www.keil.com/support/man/docs/uv4/uv4_dg_prjfoldext.htm>
    - Folder: …-update\ (not bin etc.)
    - Add include file paths to Options/CC/Include Paths
    - Disable multi-line comment warning (allow C99)
    - Documentation at [file:///C:/Program%20Files%20(x86)/GNU%20Tools%20ARM%20Embedded/8%202019-q3-update/share/doc/gcc-arm-none-eabi/html/gcc/index.html](file:///C:\Program%20Files%20(x86)\GNU%20Tools%20ARM%20Embedded\8%202019-q3-update\share\doc\gcc-arm-none-eabi\html\gcc\index.html)
* Change source code (C:\Users\Alex\Documents\Teaching\ESA\ESA-Fall-2019\Code\ESA-19\GNU\LCDs-Profiler-RTX5-CMSISv2) from Keil armcc5 to gnu syntax
  + #pragma noinline -> ?. [file:///C:/Program%20Files%20(x86)/GNU%20Tools%20ARM%20Embedded/8%202019-q3-update/share/doc/gcc-arm-none-eabi/html/gcc/Inline.html#Inline](file:///C:\Program%20Files%20(x86)\GNU%20Tools%20ARM%20Embedded\8%202019-q3-update\share\doc\gcc-arm-none-eabi\html\gcc\Inline.html#Inline)
  + \_\_align(4) int var -> int var \_\_attribute\_\_ ((aligned (4)) <https://gcc.gnu.org/onlinedocs/gcc-5.2.0/gcc/Variable-Attributes.html#Variable-Attributes>. Commented out: lucida\_8x13.c, lucida\_12x19.c
  + \_\_current\_sp() -> <https://gcc.gnu.org/onlinedocs/gcc-5.2.0/gcc/Return-Address.html#Return-Address>. \_builtin\_frame\_address()? Getsp? Commented out in timers.c
* Looking for start-up code for GNU
  + <https://github.com/ErichStyger/mcuoneclipse/tree/master/Examples/Eclipse/FRDM-KL25Z/FRDM-KL25Z_GNU_Arm_Eclipse/startup> and <http://mcuoneclipse.com/2013/12/23/diy-free-toolchain-for-kinetis-part-7-gnu-arm-eclipse-plugins/>
* And the linker file too (.ld)
* Can CMSIS, RTXv5 and GCC work together?
  + Can build CMSIS components with GCC, but not Keil Middleware. <http://www.keil.com/support/docs/4130.htm>
  + <https://sergeev.io/notes/cortex_cmsis/>
* Sbrk missing… <https://answers.launchpad.net/gcc-arm-embedded/+question/246883>



* Download button disabled. Had to update utilities tab to use debugger settings to enable it
* Code sort of works
  + Starting tone plays very slowly, repeating cycle.
  + Where do I set the stack size?
* Tool installation information: <https://www.nxp.com/document/guide/get-started-with-the-frdm-kl25z:NGS-FRDM-KL25Z>, select Use Arm GCC button at bottom

# Use Arm GCC

#### **Running a demo using Arm**

**1. Set Up Toolchain**

This section contains the steps to install the necessary components required to build and run a KSDK demo application with the Arm GCC toolchain, as supported by the Kinetis SDK. There are many ways to use Arm GCC tools, but this example focuses on a Windows environment. Though not discussed here, GCC tools can also be used with both Linux OS and Mac OSX.

**Install GCC Arm Embedded Toolchain**

Download and run the installer from [launchpad.net/gcc-arm-embedded](https://launchpad.net/gcc-arm-embedded). This is the actual toolchain (i.e., compiler, linker, etc.). The GCC toolchain should correspond to the latest supported version, as described in the *Kinetis SDK Release Notes.*

**Install MinGW**

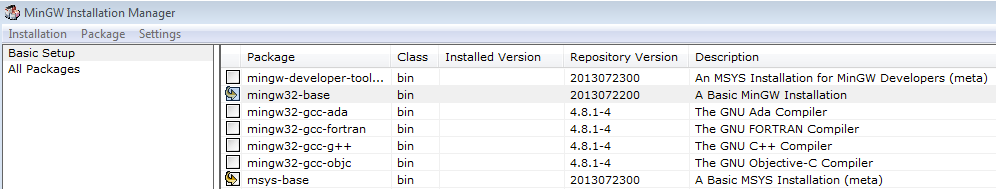
The Minimalist GNU for Windows (MinGW) development tools provide a set of tools that are not dependent on third party C-Runtime DLLs (such as Cygwin). The build environment used by the KSDK does not utilize the MinGW build tools, but does leverage the base install of both MinGW and MSYS. MSYS provides a basic shell with a Unix-like interface and tools.

1. Download the latest MinGW mingw-get-setup installer from [sourceforge.net/projects/mingw/files/Installer/](http://sourceforge.net/projects/mingw/files/Installer/).
2. Run the installer. The recommended installation path is C:\MinGW, however, you may install to any location.

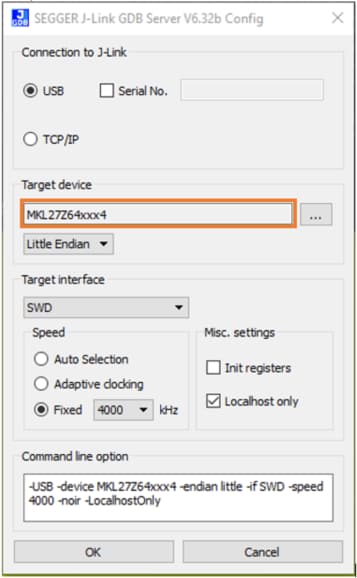
**NOTE**

The installation path cannot contain any spaces.

1. Ensure that the "mingw32-base" and "msys-base" are selected under Basic Setup.



1. Click "Apply Changes" in the "Installation" menu and follow the remaining instructions to complete the installation.



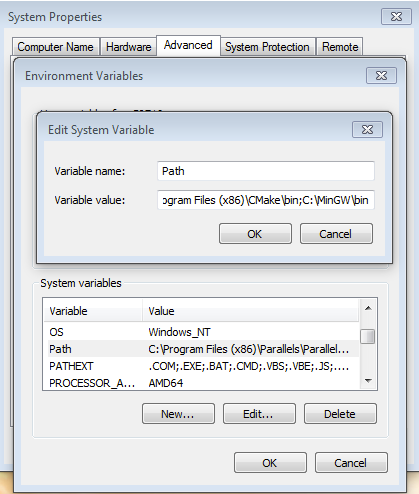
1. Add the appropriate item to the Windows operating system Path environment variable. It can be found under *Control Panel -> System and Security -> System -> Advanced System Settings* in the "Environment Variables..." section. The path is:

*\bin*

Assuming the default installation path, *C:\MinGW*, an example is shown below. If the path is not set correctly, the toolchain does not work.

**NOTE**

If you have "C:\MinGW\msys\x.x\bin" in your PATH variable (as required by KSDK 1.0.0), remove it to ensure that the new GCC build system works correctly.

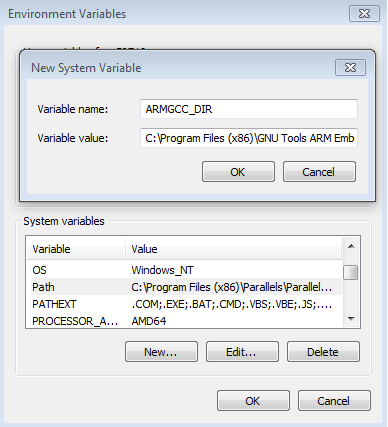


**Add a New Environment Variable for ARMGCC\_DIR**

Create a new *system* environment variable and name it ARMGCC\_DIR. The value of this variable should point to the Arm GCC Embedded tool chain installation path, which, for this example, is:

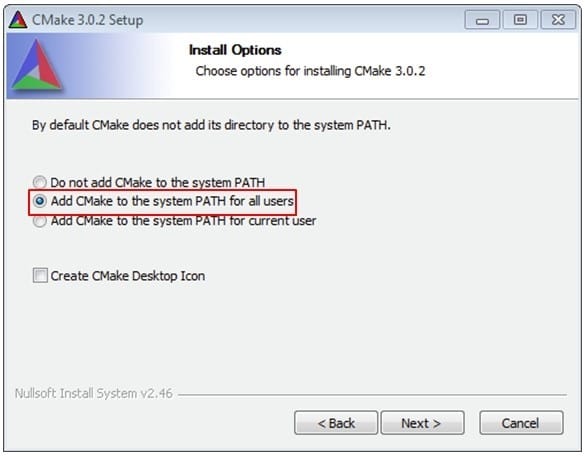
*C:\Program Files (x86)\GNU Tools Arm Embedded\4.9 2015q3*

Reference the installation folder of the GNU Arm GCC Embedded tools for the exact path name of your installation.



**Install CMake**

* 1. Download CMake 3.0.x from [www.cmake.org/cmake/resources/software.html](http://www.cmake.org/download/).
  2. Install CMake, ensuring that the option "Add CMake to system PATH" is selected when installing. It's up to the user to select whether it's installed into the PATH for all users or just the current user. In this example, the assumption is that it's installed for all users.

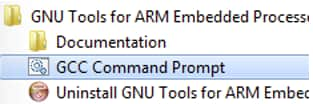


* 1. Follow the remaining instructions of the installer.
  2. You may need to reboot your system for the PATH changes to take effect.

**2. Build an Example Application**

To build an example application, follow these steps.

1. 1. If not already running, open a GCC Arm Embedded tool chain command window. To launch the window, from the Windows operating system Start menu, go to “Programs -> GNU Tools Arm Embedded ” and select “GCC Command Prompt”.



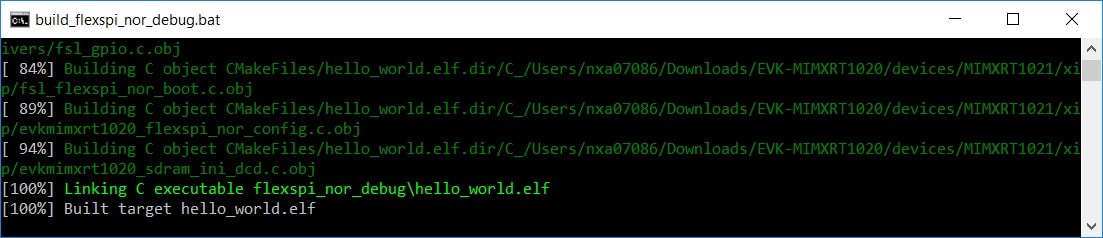
1. Change the directory to the example application project directory, which has a path like this:

*/boards////armgcc*

For this guide, the exact path is:

*/boards/frdmke15z/demo\_apps/hello\_world/armgcc*

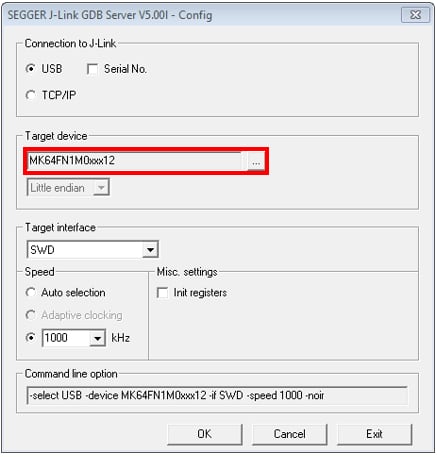
1. Type “build\_debug.bat” on the command line or double click on the "build\_debug.bat" file in Windows operating system Explorer to perform the build. The output is shown in this figure:



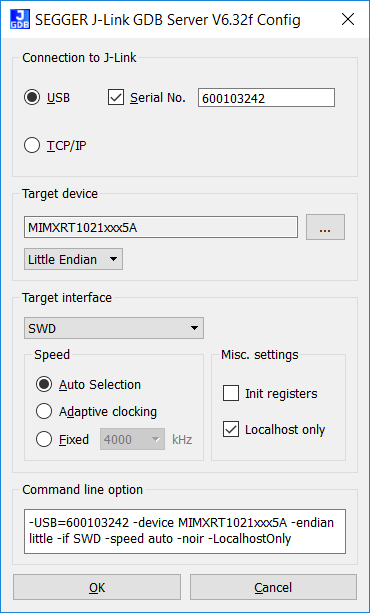
**3. Run an Example Application**

The GCC tools require a J-Link debug interface. To update the OpenSDA firmware on your board to the latest J-Link app, visit [www.nxp.com/opensda](https://www.nxp.com/design/microcontrollers-developer-resources/ides-for-kinetis-mcus/opensda-serial-and-debug-adapter:OPENSDA). After installing the J-Link OpenSDA application, download the J-Link driver and software package from [www.segger.com/downloads.html](https://www.segger.com/downloads.html).

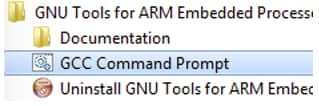
1. Connect the development platform to your PC via USB cable between the "SDAUSB" USB port on the board and the PC USB connector.
2. Open the terminal application on the PC (such as PuTTY or TeraTerm) and connect to the debug COM port you determined earlier. Configure the terminal with these settings:
   * 15200 baud rate
   * No parity
   * 8 data bits
   * 1 stop bit
3. Open the J-Link GDB Server application. Assuming the J-Link software is installed, the application can be launched by going to the Windows operating system Start menu and selecting "Programs -> SEGGER -> J-Link J-Link GDB Server".
4. Modify the settings as shown below. The target device selection chosen for this example is the “MK64FN1M0xxx12” and use the SWD interface.



1. After it is connected, the screen should resemble this figure:



1. If not already running, open a GCC Arm Embedded tool chain command window. To launch the window, from the Windows operating system Start menu, go to "Programs -> GNU Tools Arm Embedded " and select "GCC Command Prompt".



1. Change to the directory that contains the demo application output. The output can be found in using one of these paths, depending on the build target selected:

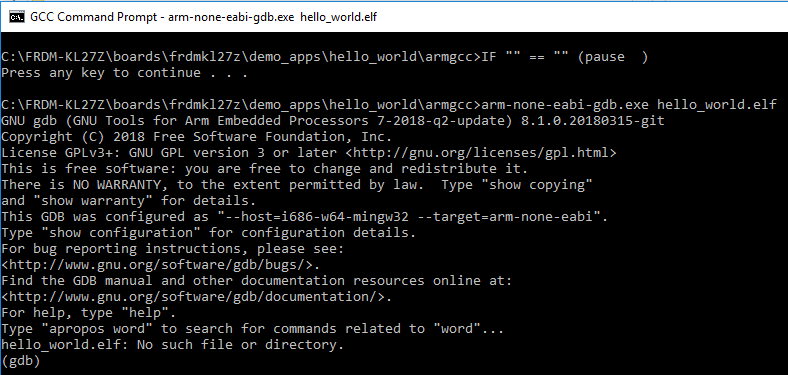
*/boards////armgcc/debug*

*/boards////armgcc/release*

For this guide, the path is:

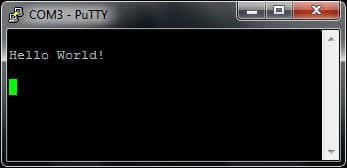
*/boards/frdmke15z/demo\_apps/hello\_world/armgcc/debug*

1. Run the command "arm-none-eabi-gdb.exe .elf". For this example, it is "arm-none-eabi-gdb.exe hello\_world.elf".



1. Run these commands:
   * "target remote localhost:2331"
   * "monitor reset"
   * "monitor halt"
   * "load"
   * "monitor reset"
2. The application is now downloaded and halted at the reset vector. Execute the "monitor go" command to start the example application.

The hello\_world application is now running and a banner is displayed in the terminal window.



**ABOUT NXP**

* [Investors](https://investors.nxp.com/)
* [Press, News, Blogs](https://media.nxp.com/)
* [Careers](https://www.nxp.com/about/careers-at-nxp:CAREERS)

**RESOURCES**

* [Mobile Apps](https://www.nxp.com/about/nxp-mobile-applications:NXP-APPS)
* [Contact Us](https://www.nxp.com/pages/contact-us:CONTACTUS)

**FOLLOW US**

GCC