Tutorial

Configuring CodeLite and/or NetBeans

for developing and debugging on STM32xx target systems – for free!!

Brief

I wanted a **professional, free,** alternative to the IAR[1], Keil[1] or Visual Studio[1] (+VisualGDB[1]) development environments for the STM32 ARM processors.

This tutorial offers two options – one for CodeLite (a light-weight IDE) and another for NetBeans (a full-featured IDE).

I suggest you go for the NetBeans option. It's quick and easy to configure for new projects and it's a very professional environment. That said, I still love CodeLite as well. Also, I'm pretty sure you could now modify other IDEs to work on the STM32…

Physical equipment needed

#	Equipment	Notes
1	PC (with Windows or Linux – your choice)	However, this tutorial describes the procedure for Windows. If you're a Linux person, I'm pretty sure you can do the conversion yourself
2	ST-Link/V2	Either the onboard ST-Link on the Nucleo or Discovery boards or a separate ST-Link/V2 dongle
3	STM32-based target board	*Example: "STM32F4Discovery"*

Software Required

(In the order they must be installed)

#	Software	Link	Notes
1	GNU ARM Embedded Toolchain	https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads *Version 7-2018-q2-update or later*	**Hint:** When installing, choose a path without spaces.
2	Open source ST-Link (v1.5+)	https://github.com/texane/stlink *Committed 3 August 2018 or later*	(Instructions below)
3	STM32CubeMX	https://www.st.com/en/development-tools/stm32cubemx.html *Version 4.26.1 or later*	Helps you create initialization code for your device
4	ST-Link utilities	STSW-LINK009 (Driver for Windows 10) STSW-LINK004 (STM32 ST-Link Utility) STSW-LINK007 (Firmware upgrade)	Also available on Linux

		Version 2.0.0 of the driver *Version 4.2.0 of the utility* *Version 2.31.21 of the firmware*	
5	CodeLite IDE	https://codelite.org/ *Version 12.0.6 or later*	You pick one
	NetBeans IDE	https://netbeans.org/ *Version 8.2 or later*	
5	GCC	http://www.mingw.org/ (not MinGW-64) *Version 6.3.0 or later*	(No need to tell the Linux guys)
6	CMAKE	https://cmake.org/download/ *Version 3.12.0 or later*	Ditto
7	Git	https://git-scm.com/ *Version 2.18.0 or later*	Needed for GitHub

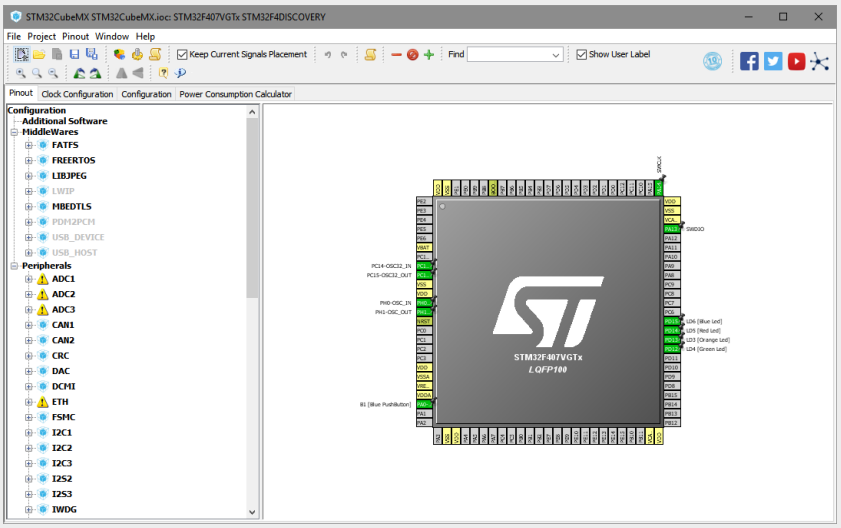
Preparation work (done once)

#	Instructions	
1	Download and install • Git • GCC • CMAKE • GNU ARM Embedded Toolchain • Add the '/bin' folder to the PATH	

2	Create a working folder
	`> git clone https://github.com/texane/stlink.git .` (← see the 'space'-'dot')
	`> mkdir build`
	`> cd build`
	`> cmake ..` (← see the 'space'-'dot-dot')
	`> make`
	
	This will create
	• `Debug\`**`st-flash.exe`**
	• `Debug\`**`st-info.exe`**
	• `src\gdbserver\Debug\`**`st-util.exe`**
	• `3thparty\libusb-1.0.22\MinGW32\dll\`**`libusb-1.0.dll`**	

	Then copy: • **`st-util.exe (and the other EXE files)`** • **`libusb-1.0.dll`** to some folder in your PATH…	
3	Download and install STM32CubeMX for your device (example STM32CubeF4) ***HINT**: Create a login account to make life easy for yourself.*	
4	Download and install the ST-Link utilities (using the same account)	
5	Download and install either CodeLite or NetBeans for C/C++ ***HINT**: Don't change the default configuration just yet*	
6	**NOTE:** You don't have to add the GNY GCC compiler to your IDE – it will use the "Makefile" anyway	

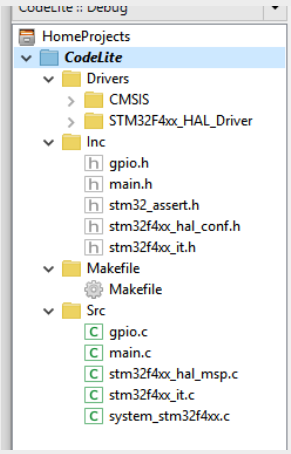
Project work: STM32CubeMX

#	Instructions	Notes
1	Use STM32CubeMX to define your peripherals.  	This detail is beyond the scope of this tutorial. See the ST documentation.
2	Select "**Makefile**" for the toolchain/IDE	***HINT**: I like:* • "Copy only the necessary library file" • "Generate '.c/.h' files"
3	"Project" → "Generate Code"	
4	***HINT**: AT this point you can test your project by calling "make" on the command line.*	`> make` It should build with no errors or warnings.

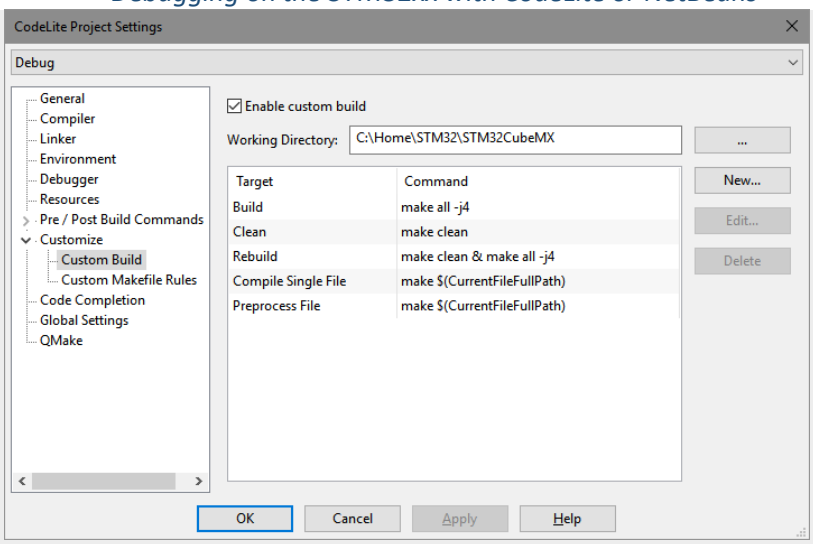
So far, so good. But now we want to do some serious work …

Option 1: Setting up a CodeLite project with ST-Link

#	Instructions	Notes
1	Create a "Simple GCC" project	Select the default GCC compiler – we won't use it anyway
2	Remove and delete "main.c"	We'll use the one from STM32CubeMX

3	Right-Click on the project name:
	• "Import files from directory"
	  ○ Select the Drivers, Inc and Src folders from the STM32Cube project
	  ○ Also add the "Makefile"
	 
		So you can edit and set breakpoints…

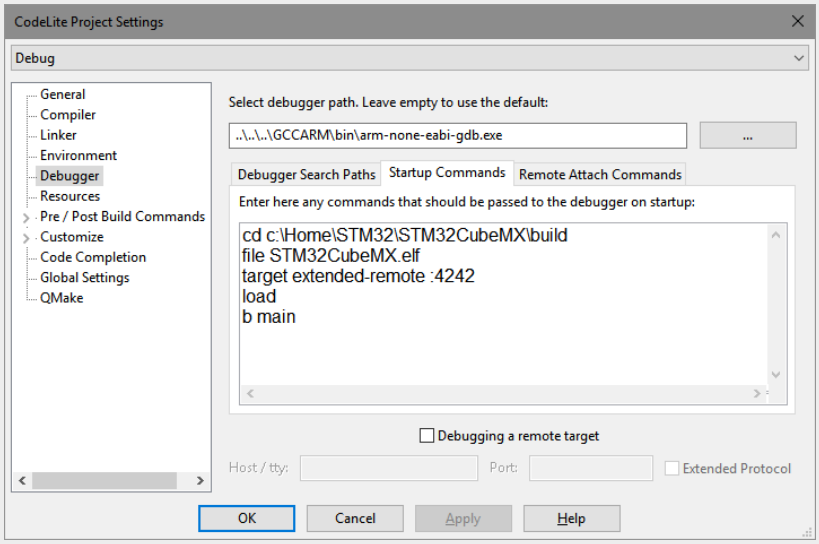
4	Right-Click on the project name: • "Settings" → "Customize --> Custom Build": • [X] Enable Custom builds: <table><tr><td>Working Directory</td><td>*<folder where Makefile is>*</td></tr><tr><td>Build</td><td>`make all -j4`</td></tr><tr><td>Clean</td><td>`make clean`</td></tr><tr><td>Rebuild</td><td>`make clean & make all -j4`</td></tr><tr><td>Compile single</td><td>`make $(CurrentFileFullPath)`</td></tr><tr><td>Preprocess file</td><td>`make $(CurrentFileFullPath)`</td></tr></table>	



5 Right-Click on the project name:

- "Settings" → "Debugger"

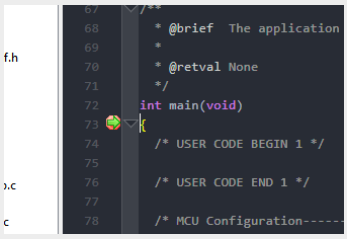
Selected debugger	*<....\bin\arm-none-eabi-gdb.exe>*
Debug search paths	{blank}
Startup Commands	cd *<path where ELF file is>* file *<name of ELF file>* target extended-remote :4242 load b main
Remote attach cmd	{blank}



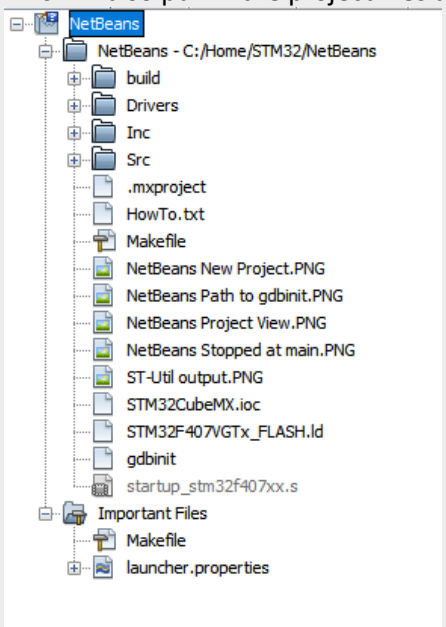
Development Cycle: CodeLite

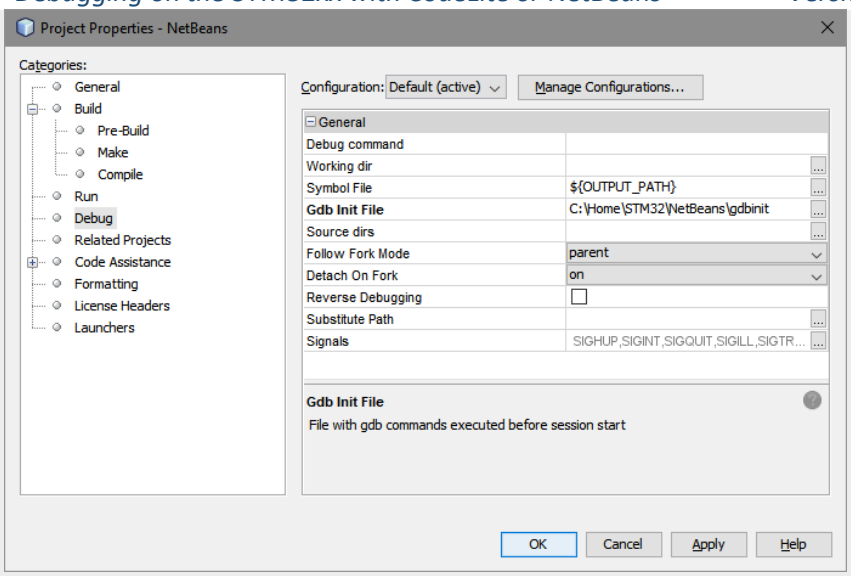
#	Instructions	Notes

1	Start the "st-util" program in a separate window…
	`C:\Home\STM32>call st-util`
	`st-util 1.4.0-47-gae717b9`
	`2018-08-09T02:07:04 INFO common.c: Loading device parameters....`
	`2018-08-09T02:07:04 INFO common.c: Device connected is: F4 device, id 0x10016413`
	`2018-08-09T02:07:04 INFO common.c: SRAM size: 0x30000 bytes (192 KiB), Flash: 0x100000 bytes (1024 KiB) in pages of 16384 bytes`
	`2018-08-09T02:07:04 INFO gdb-server.c: Chip ID is 00000413, Core ID is 2ba01477.`
	`2018-08-09T02:07:04 INFO gdb-server.c: Listening at *:4242...`
		← Don't forget – only done once

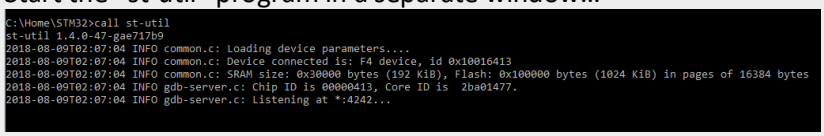
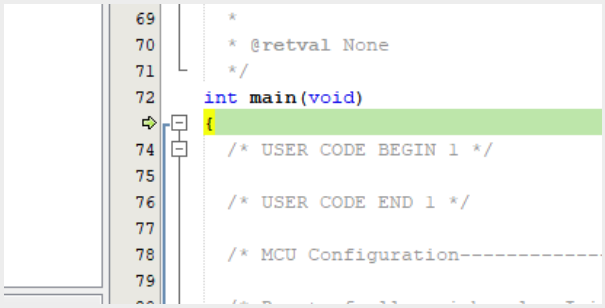
2	Develop as normal • Edit • Build / Rebuild / Clean, etc • Press F5  	And the rest is up to you !!!

Option 2: Setting up a NetBeans project with ST-Link

#	Instructions	Notes
1	Select "File"→"New Project" • "C/C++ Project with Existing Sources" • Specify the folder where the STM32CubeMX **Makefile** is • Select the standard GCC tool chain *(won't be used anyway)* • Keep the Automatic Mode Press "Finish" and it will do a build… **(should give 0 errors, 0 warnings)** This will also pull in the project files automatically  	
2	Create a file "**gdbinit**" with the following content: • `cd build` • `file <name of the ELF file>` • `target extended-remote :4242` • `load` • `b main` and save it in the same folder as the Makefile.	
3	"Project" → "Properties" → "Debug" Gdb Init File: `<full reference to gdbinit>`	



Development Cycle: NetBeans

#	Instructions	Notes
1	Start the "st-util" program in a separate window… 	← Don't forget – only done once
2	Develop as normal • Edit • Build / Rebuild / Clean, etc • Press Shift-F5 	And the rest is up to you !!!

Notes

[1] Keil, IAR, Visual Studio and WindowsGDB are excellent software available from:

- http://www.keil.com/
- https://www.iar.com/
- https://visualstudio.microsoft.com/

Disclaimer

These instructions come with no guarantee whatsoever. Use at your own risk.