Developing with Contiki-NG in Code Composer Studio

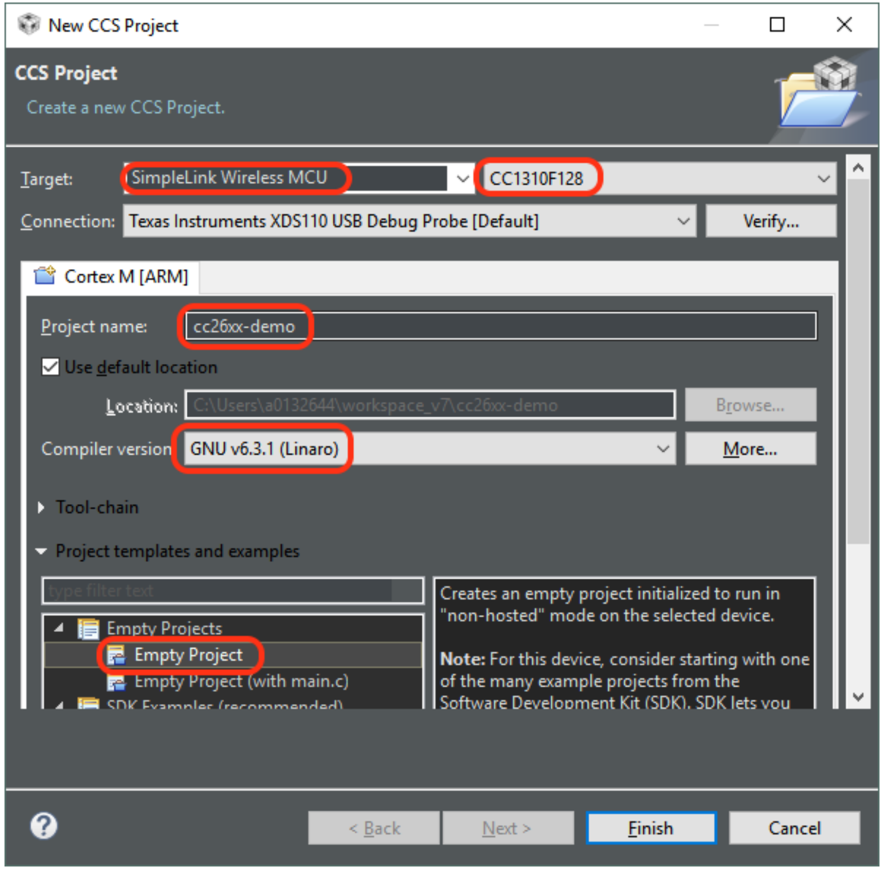
Convenient step-debugging for all platforms

Prerequisites

- Setup the Software Development Environment for Contiki-NG
  - http://processors.wiki.ti.com/index.php/Contiki_setting_up_sw
- In short:
  - Clone the Contiki-NG repository
    - *git clone https://github.com/contiki-ng/contiki-ng.git*
  - Checkout the CC13x0/CC26x0 driverlib submodules from the cloned repository
    - *cd contiki-ng/arch/cpu/cc26xx-cc13xx/lib && git submodule update --init -- .*

1. Create an empty CCS project

- Create a new CCS project
  - File -> New -> CCS Project
- Make sure correct Target device is selected
- Name the project to whatever your liking
- Make sure GNU compiler is selected, as well as the Empty Project template



New CCS Project

CCS Project

Create a new CCS Project.

Target: SimpleLink Wireless MCU | CC1310F128

Connection: Texas Instruments XDS110 USB Debug Probe [Default]

Cortex M [ARM]

Project name: cc26xx-demo

[✓] Use default location

Location: C:\Users\a0132644\workspace_v7\cc26xx-demo

Compiler version: GNU v6.3.1 (Linaro)

Tool-chain

Project templates and examples

Empty Projects
- Empty Project
- Empty Project (with main.c)

SDK Examples (recommended)

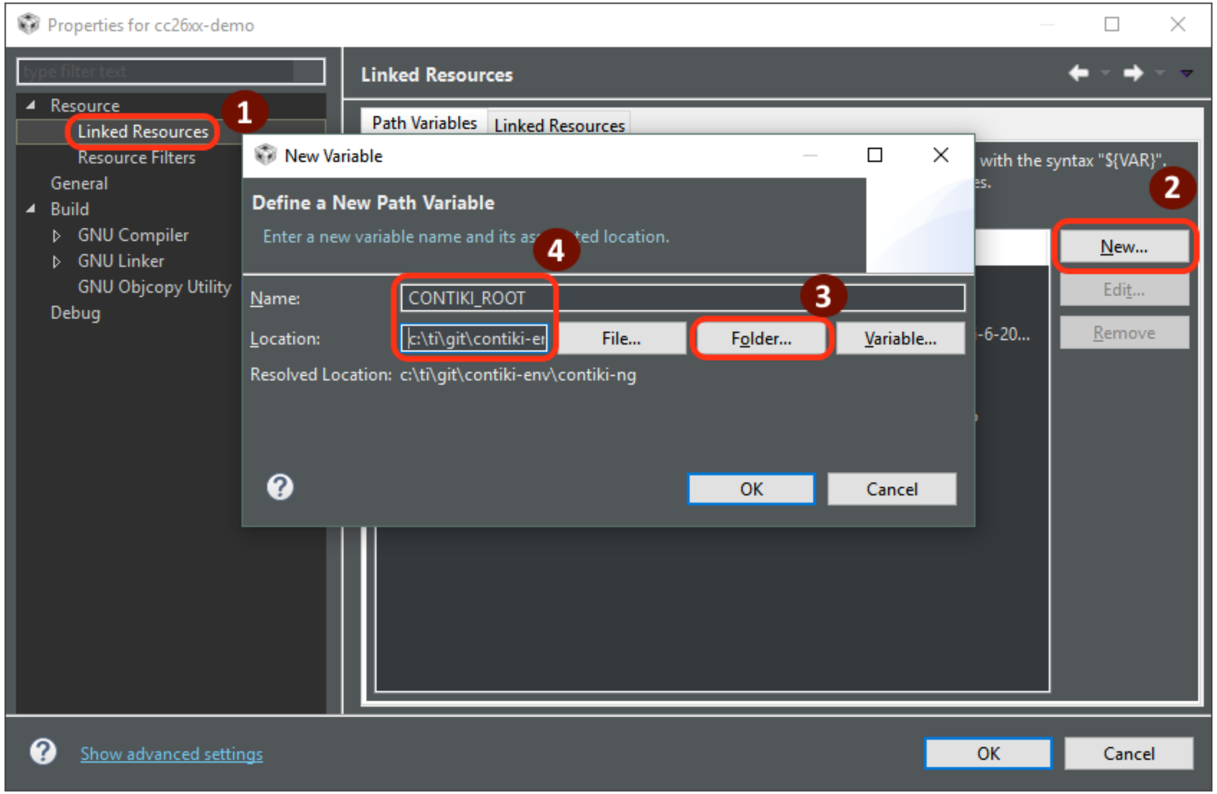
Creates an empty project initialized to run in "non-hosted" mode on the selected device.

Note: For this device, consider starting with one of the many example projects from the Software Development Kit (SDK). SDK lets you

Texas Instruments

2. Add a path variable for Contiki

- Add path variable in project preferences
- Just for Convenience
- Allows us to refer to the Contiki source folder later
- Makes it possible to switch to another contiki folder without changing the project


Properties for cc26xx-demo

Linked Resources

Path Variables | Linked Resources

New Variable

Define a New Path Variable

Enter a new variable name and its associated location.

Name: CONTIKI_ROOT

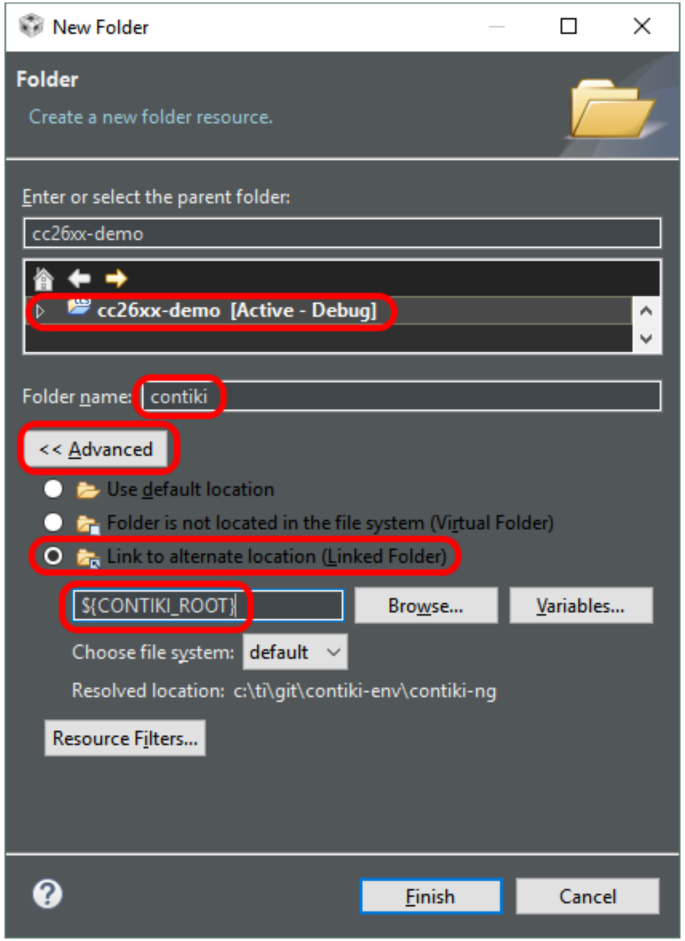
Location: c:\ti\git\contiki-en

Resolved Location: c:\ti\git\contiki-env\contiki-ng

TEXAS INSTRUMENTS

3. Add Contiki sources

- Add new folder
  - Right click project -> New -> Folder
- Click advanced settings
- Add a link based on the CONTIKI_ROOT variable
- This allows to browse Contiki files whithout copying them into the project folder.
- CCS may find .cfg files in the Contiki source tree and asks whether it should build them with XDCTools. **Click «No».**



New Folder

Folder

Create a new folder resource.

Enter or select the parent folder: cc26xx-demo

cc26xx-demo [Active - Debug]

Folder name: contiki

<< Advanced

- Use default location
- Folder is not located in the file system (Virtual Folder)
- [✓ Link to alternate location (Linked Folder)]

${CONTIKI_ROOT}

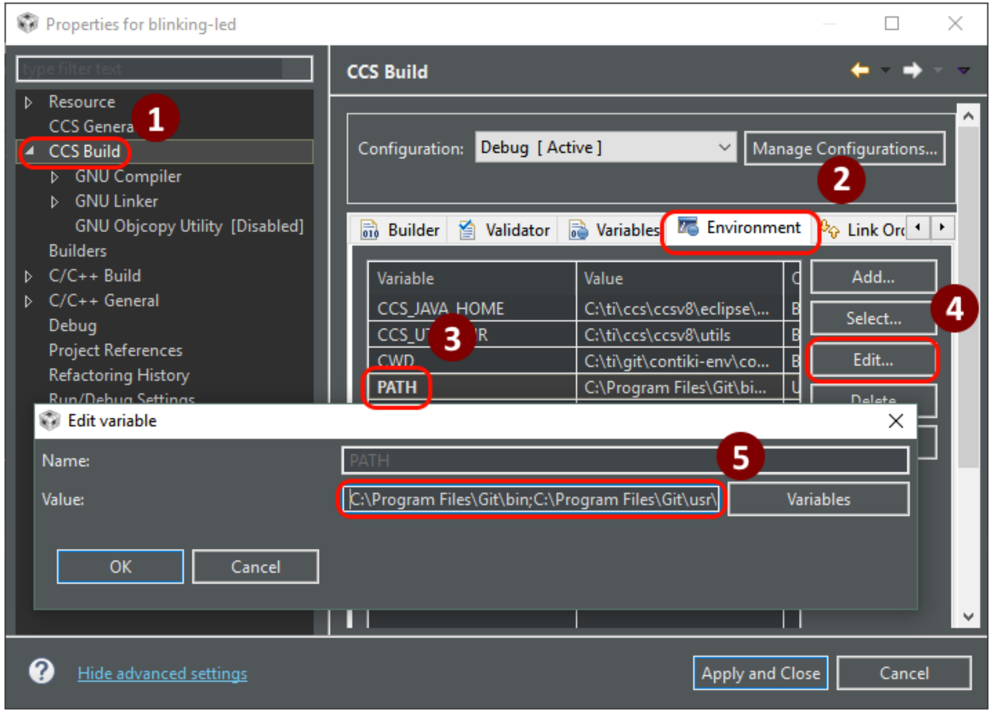
Choose file system: default

Resolved location: c:\ti\git\contiki-env\contiki-ng

Texas Instruments

4. Adjust PATH Environment Variable

- The Contiki build system needs *git*, *make* and some other shell tools to be in the PATH environment variable
- Git Bash provides all the necessary GNU tools, and is a commonly used git distribution for Windows
- Be sure both **/bin** and **/usr/bin** paths are added



Variable	Value
CCS_JAVA_HOME	C:\ti\ccs\ccsv8\eclipse\...
CCS_UTILS_DIR	C:\ti\ccs\ccsv8\utils
CWD	C:\ti\git\contiki-env\co...
PATH	C:\Program Files\Git\bi...

Name: PATH

Value: C:\Program Files\Git\bin;C:\Program Files\Git\usr\

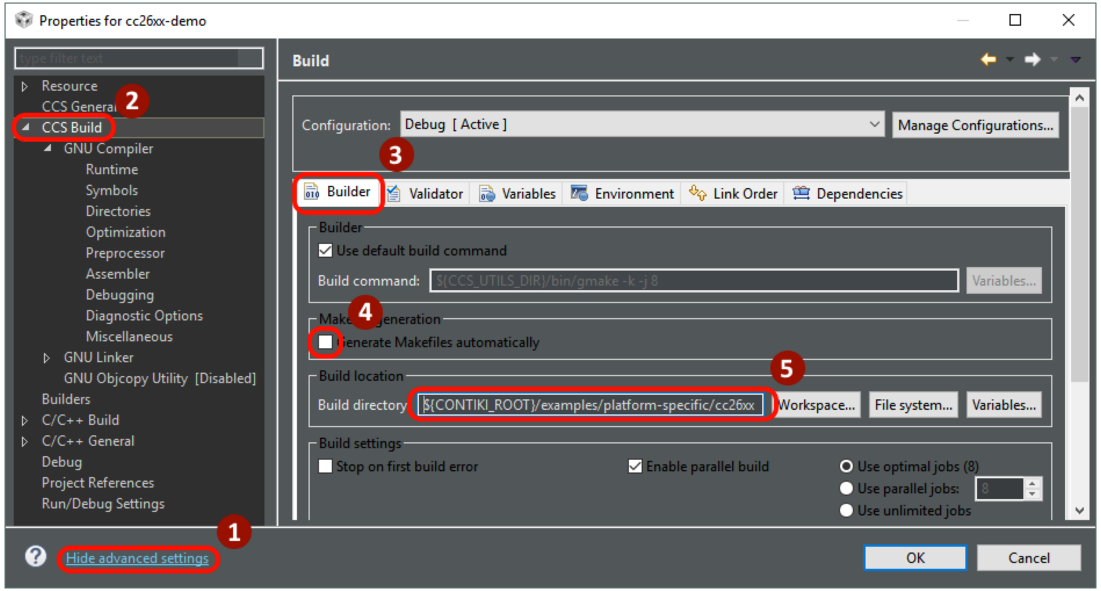
Example: Prepend the following

```
C:\Program Files\Git\bin;C:\Program Files\Git\usr\bin;
```

5. Adjust Builder settings

- Make sure you are viewing advanced settings
- Do not let CCS generate makefiles
- For building an example, use the example's source directory as build directory

Note: When creating your own applications, put the application sources and the Makefile into your project directory, not in the Contiki directory.


Properties for cc26xx-demo

Build

Configuration: Debug [Active]

Builder

[✓] Use default build command

Build command: ${CCS_UTILS_DIR}/bin/gmake -k -j 8

Makefile generation

[ ] Generate Makefiles automatically

Build location

Build directory: ${CONTIKI_ROOT}/examples/platform-specific/cc26xx

Build settings

[ ] Stop on first build error

[✓] Enable parallel build

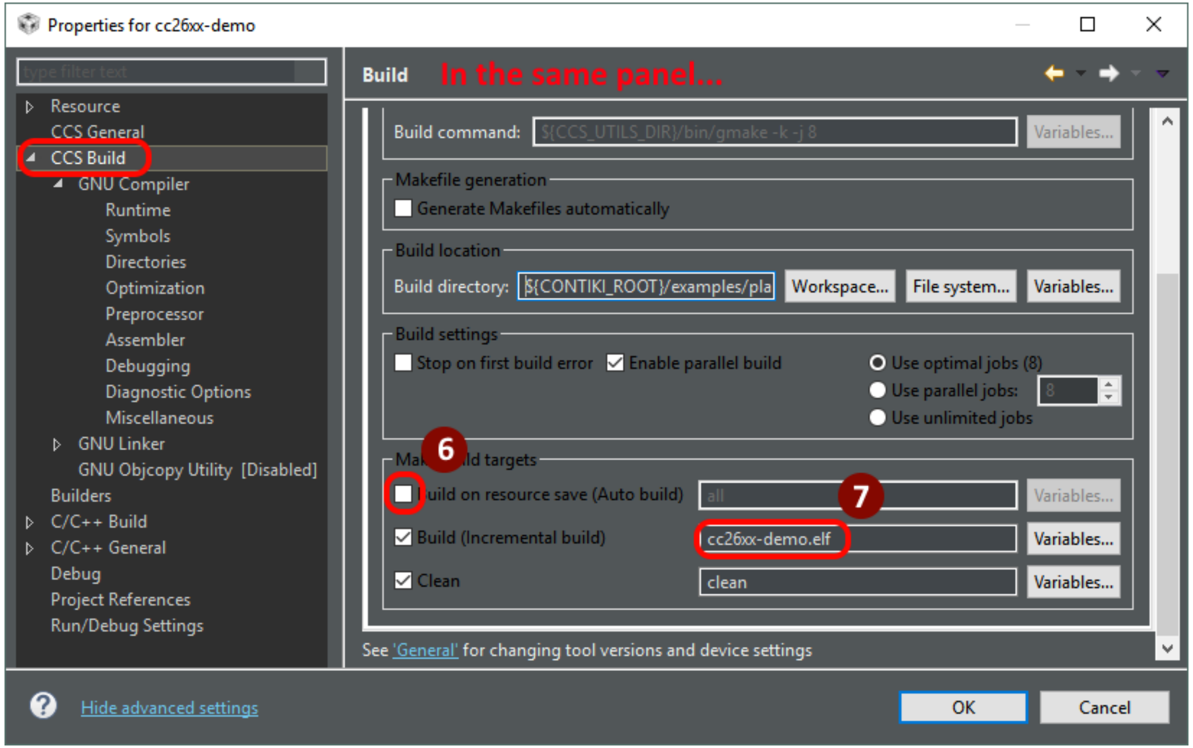
● Use optimal jobs (8)

○ Use parallel jobs: 8

○ Use unlimited jobs

6. Modify the build target

- In the same panel, uncheck build on save
- It is sufficient to build the executable (*.elf)
- No need to do hex conversion, which requires the *srecord* command



Properties for cc26xx-demo

type filter text

- Resource
- CCS General
- CCS Build
  - GNU Compiler
    - Runtime
    - Symbols
    - Directories
    - Optimization
    - Preprocessor
    - Assembler
    - Debugging
    - Diagnostic Options
    - Miscellaneous
  - GNU Linker
  - GNU Objcopy Utility [Disabled]
- Builders
- C/C++ Build
- C/C++ General
- Debug
- Project References
- Run/Debug Settings

Build In the same panel...

Build command: ${CCS_UTILS_DIR}/bin/gmake -k -j 8 [Variables...]

Makefile generation
- [ ] Generate Makefiles automatically

Build location
- Build directory: ${CONTIKI_ROOT}/examples/pla [Workspace...] [File system...] [Variables...]

Build settings
- [ ] Stop on first build error
- [✓] Enable parallel build
- (●) Use optimal jobs (8)
- ( ) Use parallel jobs: 8
- ( ) Use unlimited jobs

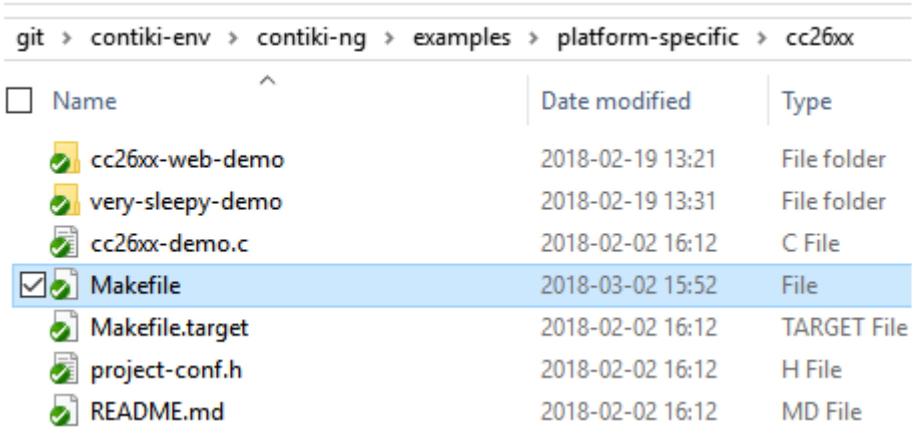
Make build targets 6
- [ ] Build on resource save (Auto build): all 7 [Variables...]
- [✓] Build (Incremental build): cc26xx-demo.elf [Variables...]
- [✓] Clean: clean [Variables...]

See 'General' for changing tool versions and device settings

Hide advanced settings [OK] [Cancel]

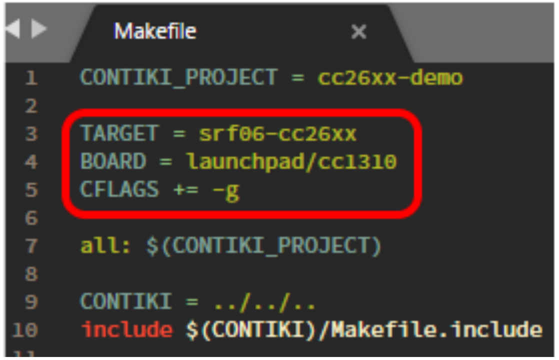
7. Add board and target variables, debug symbols

- You can set those variables either in the Makefile or in the CCS environment just like PATH
- TARGET=srf06-cc26xx for CC13x0 and CC26x0 devices
- BOARD=launchpad/cc1310 for the CC1310 launchpad
- Add CFLAGS += -g to enable debug symbols



git › contiki-env › contiki-ng › examples › platform-specific › cc26xx

Name	Date modified	Type
cc26xx-web-demo	2018-02-19 13:21	File folder
very-sleepy-demo	2018-02-19 13:31	File folder
cc26xx-demo.c	2018-02-02 16:12	C File
Makefile	2018-03-02 15:52	File
Makefile.target	2018-02-02 16:12	TARGET File
project-conf.h	2018-02-02 16:12	H File
README.md	2018-02-02 16:12	MD File

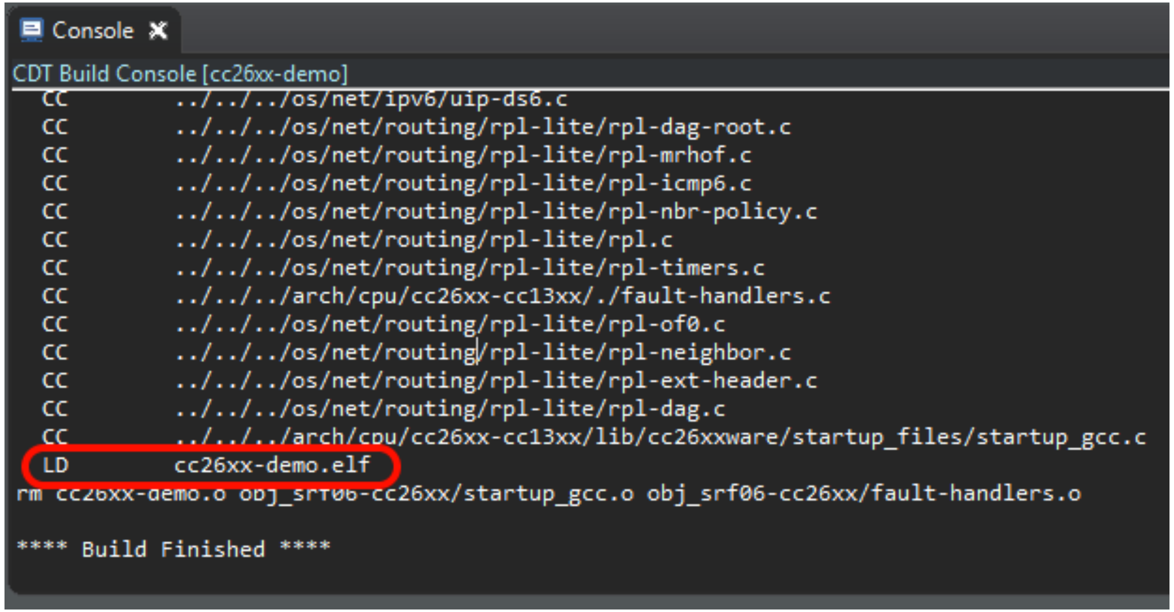


Makefile

```
CONTIKI_PROJECT = cc26xx-demo

TARGET = srf06-cc26xx
BOARD = launchpad/cc1310
CFLAGS += -g

all: $(CONTIKI_PROJECT)

CONTIKI = ../../..
include $(CONTIKI)/Makefile.include
```

8. Build the project

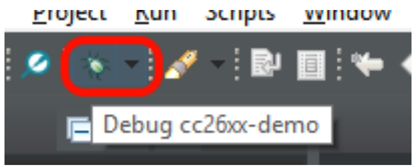
- CCS should now be able to build the *.elf file
- If something goes wrong, it is usually due to tools not being found. Check the PATH environment variable in that case.

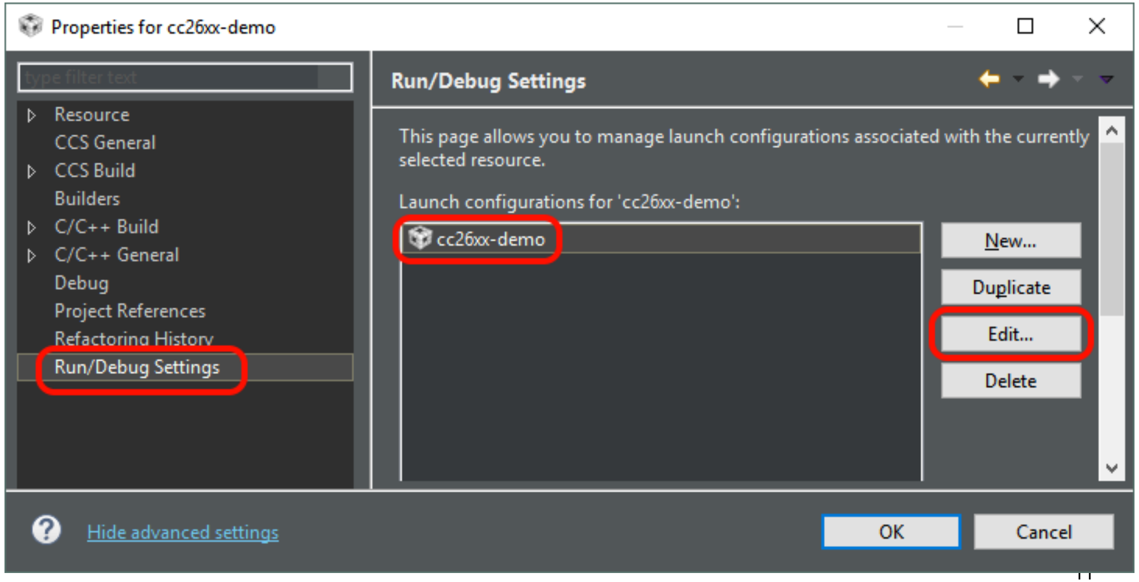


```
CDT Build Console [cc26xx-demo]
  CC          ../../../os/net/ipv6/uip-ds6.c
  CC          ../../../os/net/routing/rpl-lite/rpl-dag-root.c
  CC          ../../../os/net/routing/rpl-lite/rpl-mrhof.c
  CC          ../../../os/net/routing/rpl-lite/rpl-icmp6.c
  CC          ../../../os/net/routing/rpl-lite/rpl-nbr-policy.c
  CC          ../../../os/net/routing/rpl-lite/rpl.c
  CC          ../../../os/net/routing/rpl-lite/rpl-timers.c
  CC          ../../../arch/cpu/cc26xx-cc13xx/./fault-handlers.c
  CC          ../../../os/net/routing/rpl-lite/rpl-of0.c
  CC          ../../../os/net/routing/rpl-lite/rpl-neighbor.c
  CC          ../../../os/net/routing/rpl-lite/rpl-ext-header.c
  CC          ../../../os/net/routing/rpl-lite/rpl-dag.c
  CC          ../../../arch/cpu/cc26xx-cc13xx/lib/cc26xxware/startup_files/startup_gcc.c
  LD          cc26xx-demo.elf
rm cc26xx-demo.o obj_srf06-cc26xx/startup_gcc.o obj_srf06-cc26xx/fault-handlers.o

**** Build Finished ****
```

9. Create a default debug session

- Start a debug session and let CCS create a default debug configuration for the XDS110 debug interface
- This is expected to fail because the executable filepath guessed by CCS is wrong
- Open the newly created launch session for that project




Properties for cc26xx-demo

Run/Debug Settings

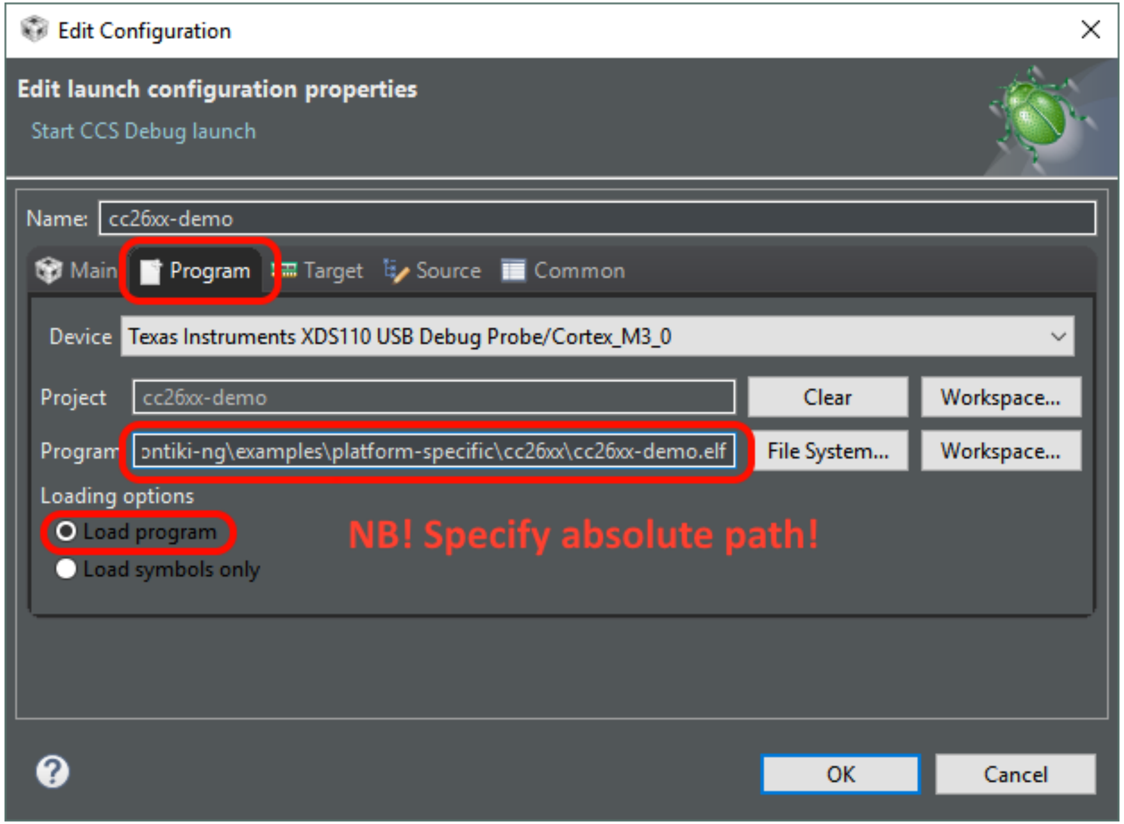
This page allows you to manage launch configurations associated with the currently selected resource.

Launch configurations for 'cc26xx-demo':

cc26xx-demo

10. Set the correct executable path

- CCS tries to deduce the executable filepath from a magic variable
- But we are building it with an external makefile
- The executable path needs to be hard-coded (unless we find a simpler solution)


Edit Configuration

Edit launch configuration properties

Start CCS Debug launch

Name: cc26xx-demo

Device: Texas Instruments XDS110 USB Debug Probe/Cortex_M3_0

Project: cc26xx-demo

Program: ontiki-ng\examples\platform-specific\cc26xx\cc26xx-demo.elf

Loading options

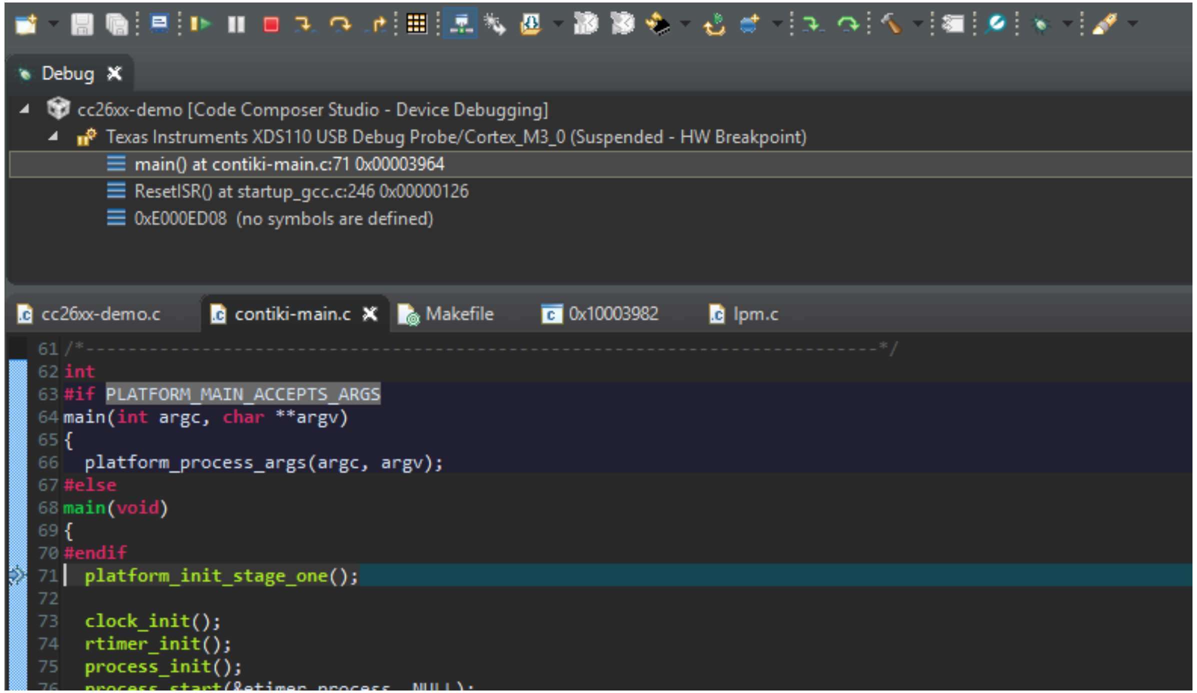
[✓ Load program] Load symbols only

NB! Specify absolute path!

TEXAS INSTRUMENTS

11. Start debugging

- Now that the launch configuration points to the correct file, start the debug session again
- You are now able to step through the source code



Debug

```
cc26xx-demo [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/Cortex_M3_0 (Suspended - HW Breakpoint)
    main() at contiki-main.c:71 0x00003964
    ResetISR() at startup_gcc.c:246 0x00000126
    0xE000ED08  (no symbols are defined)
```

cc26xx-demo.c | contiki-main.c | Makefile | 0x10003982 | lpm.c

```c
/*---------------------------------------------------------------------------*/
int
#if PLATFORM_MAIN_ACCEPTS_ARGS
main(int argc, char **argv)
{
  platform_process_args(argc, argv);
#else
main(void)
{
#endif
  platform_init_stage_one();

  clock_init();
  rtimer_init();
  process_init();
  process_start(&etimer_process, NULL);
```

Texas Instruments