

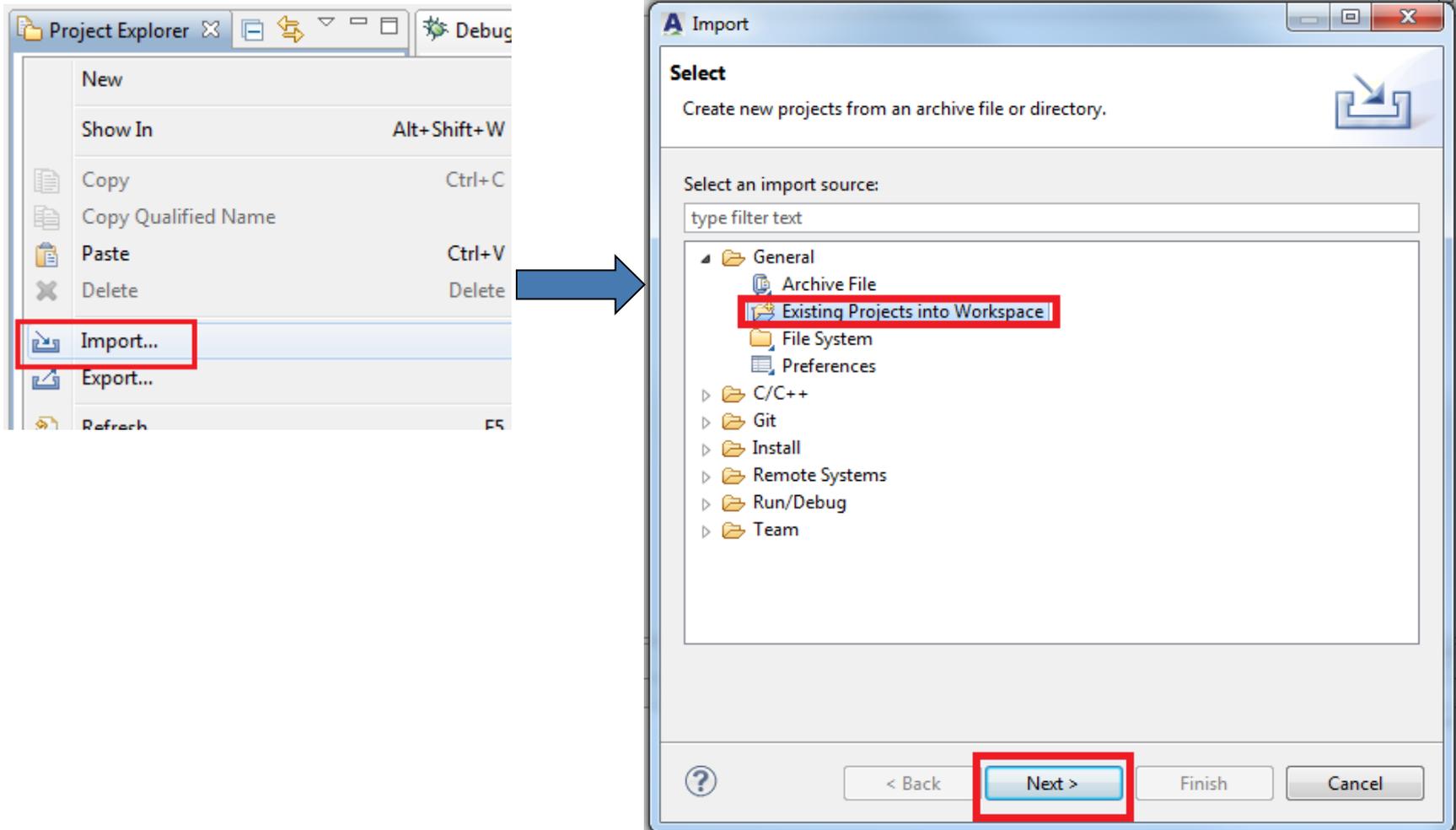
# Code Coverage - Simulator

Driving Innovations™



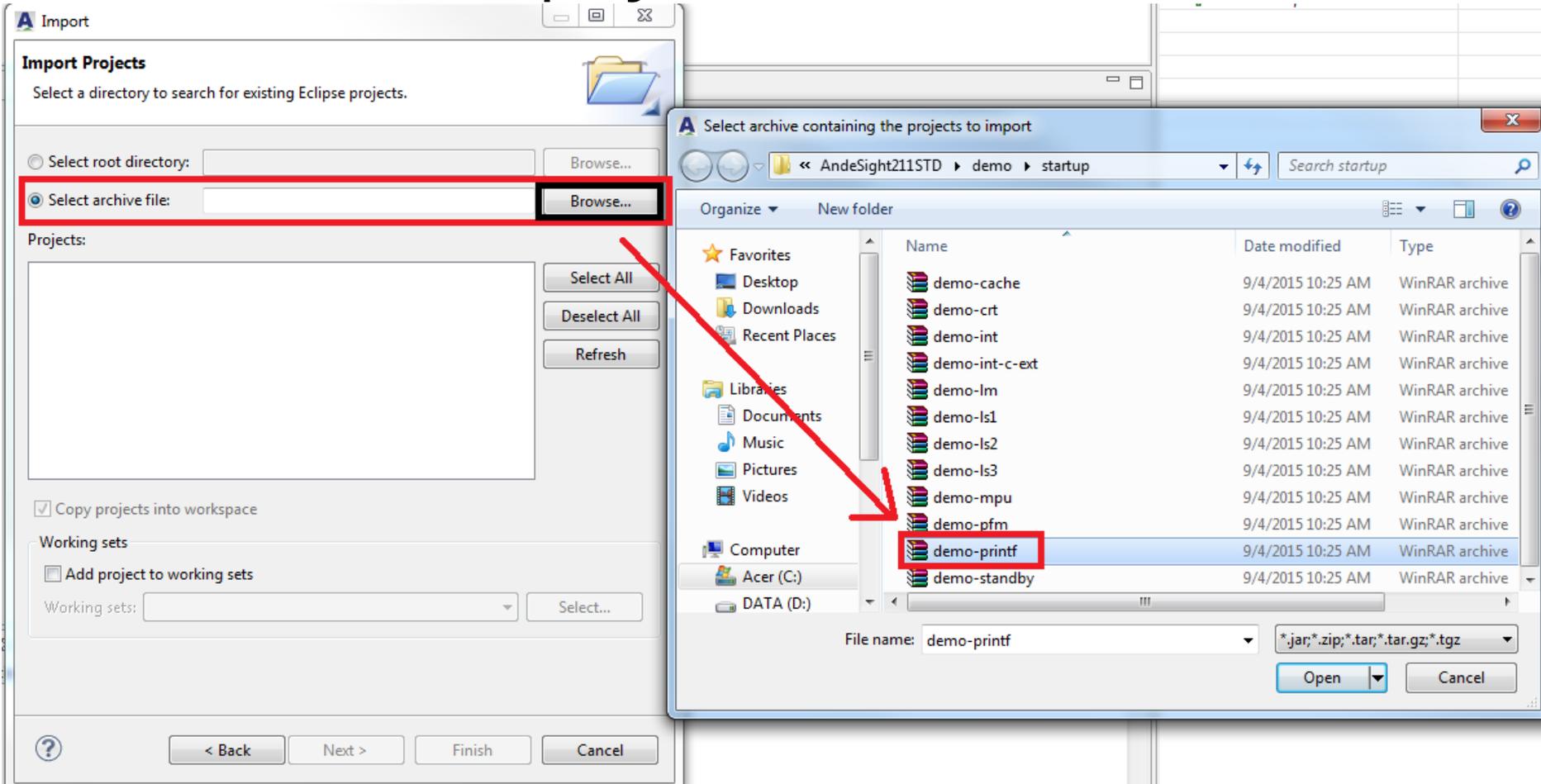
# Import

❖ Right click on Project Explorer view.



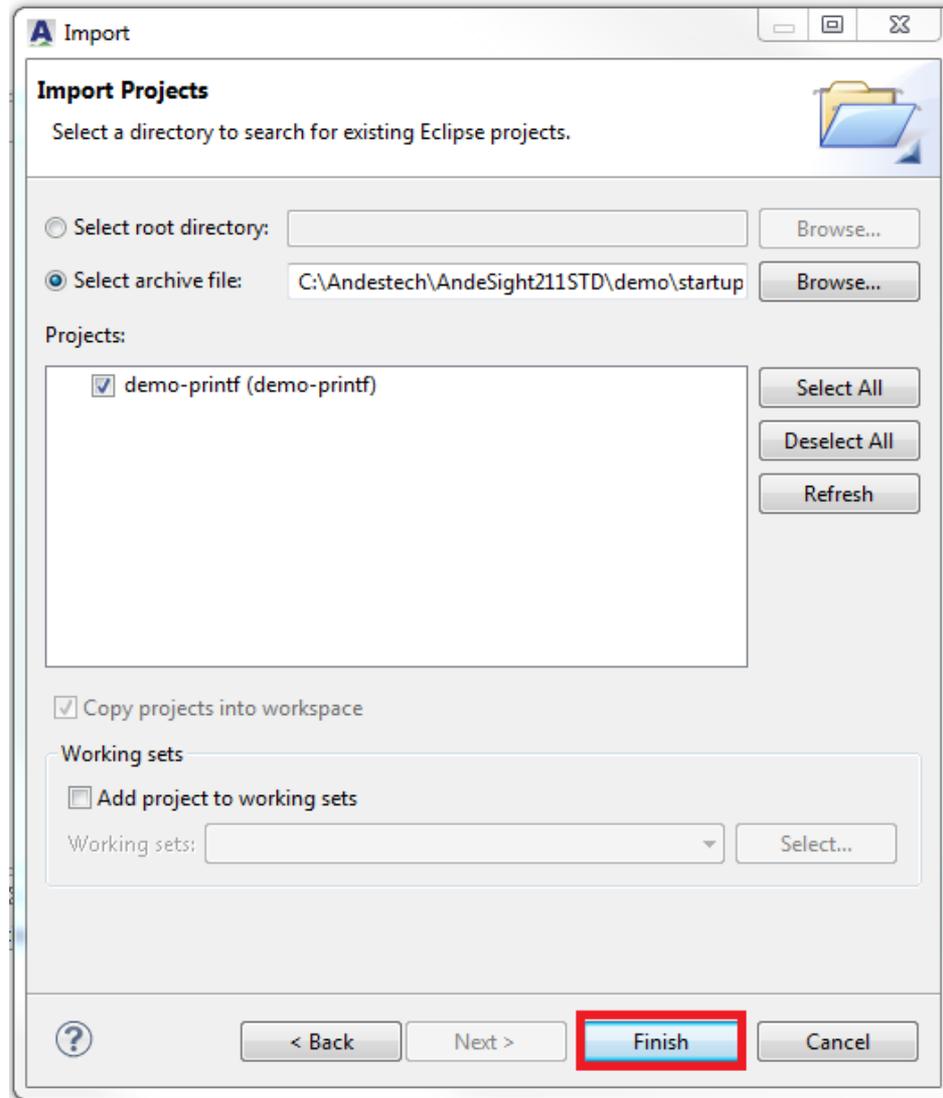
# Import Project

❖ Select the demo project.



# Import Project

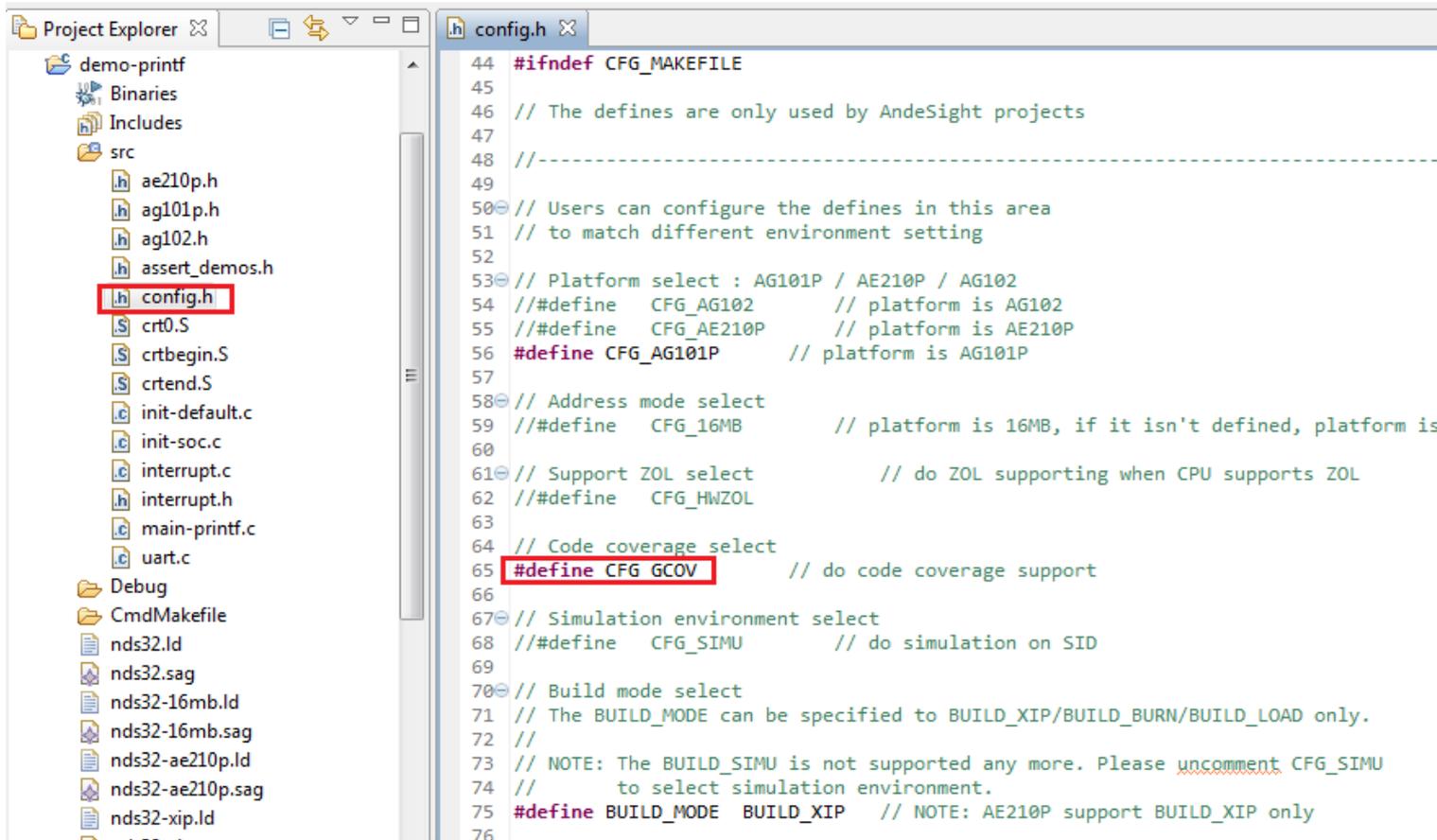
❖ Click “Finish”.



# config.h

❖ Please modify the config.h.

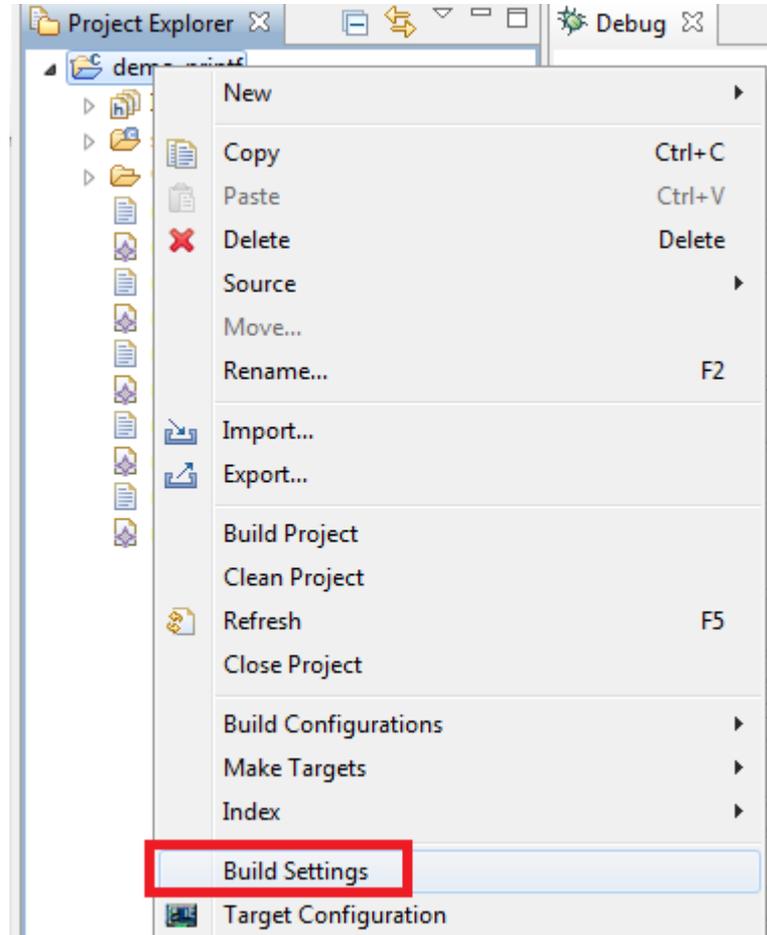
■ Please uncomment the "#define CFG\_GCOV".



```
44 #ifndef CFG_MAKEFILE
45
46 // The defines are only used by AndeSight projects
47
48 //-----
49
50 // Users can configure the defines in this area
51 // to match different environment setting
52
53 // Platform select : AG101P / AE210P / AG102
54 //#define  CFG_AG102      // platform is AG102
55 //#define  CFG_AE210P    // platform is AE210P
56 #define  CFG_AG101P     // platform is AG101P
57
58 // Address mode select
59 //#define  CFG_16MB      // platform is 16MB, if it isn't defined, platform is
60
61 // Support ZOL select          // do ZOL supporting when CPU supports ZOL
62 //#define  CFG_HWZOL
63
64 // Code coverage select
65 #define  CFG_GCOV       // do code coverage support
66
67 // Simulation environment select
68 //#define  CFG_SIMU     // do simulation on SID
69
70 // Build mode select
71 // The BUILD_MODE can be specified to BUILD_XIP/BUILD_BURN/BUILD_LOAD only.
72 //
73 // NOTE: The BUILD_SIMU is not supported any more. Please uncomment CFG_SIMU
74 //       to select simulation environment.
75 #define  BUILD_MODE    BUILD_XIP // NOTE: AE210P support BUILD_XIP only
76
```

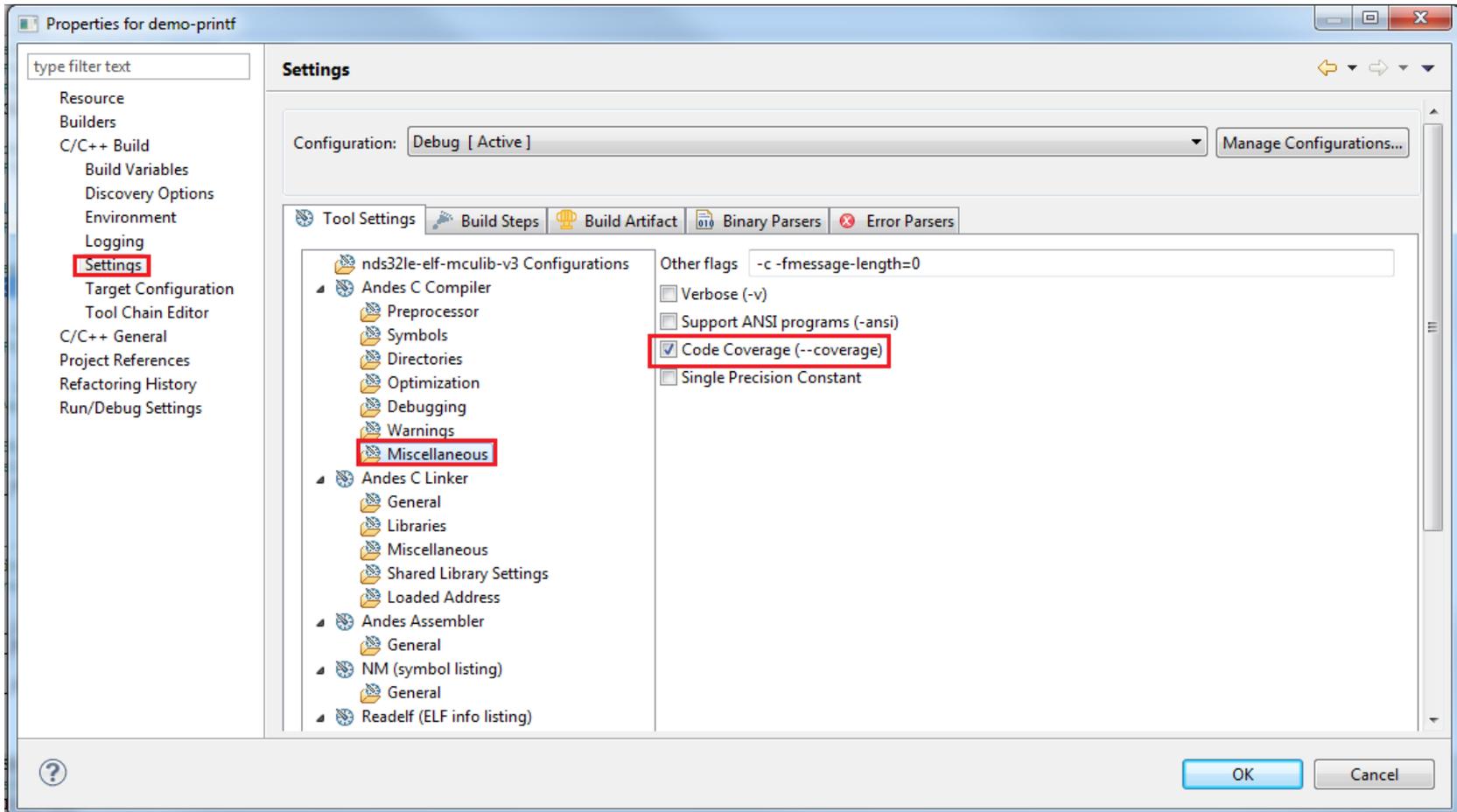
# Build Settings

❖ Right click on your project in Project Explorer view.



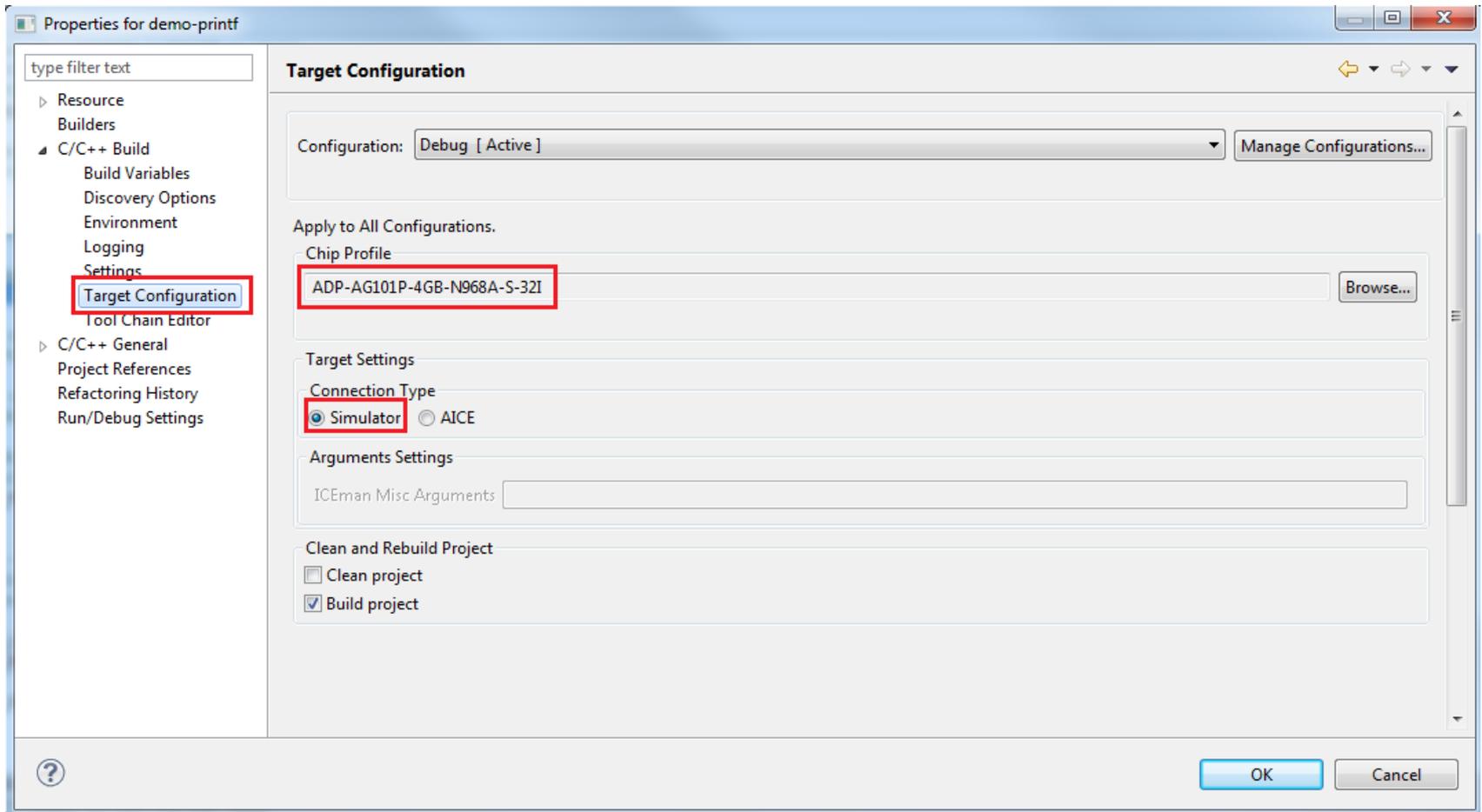
# Code Coverage

❖ Check the “Code Coverage” option.



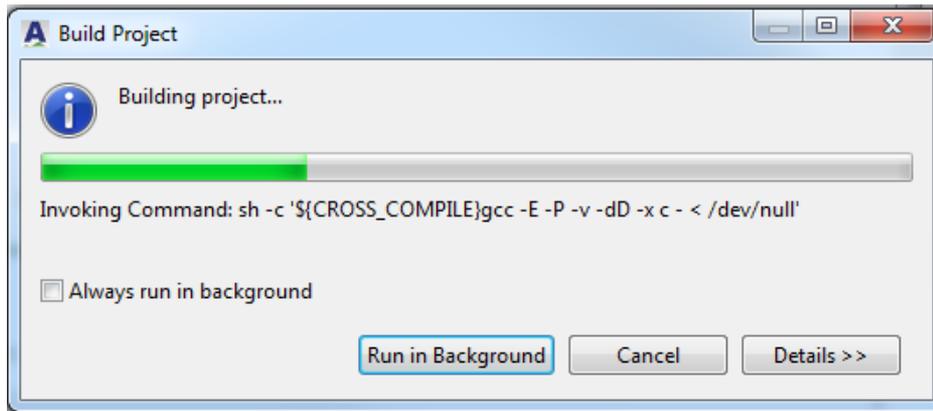
# Target Configuration

- ❖ Choose the corresponding Chip Profile.



# Build Project

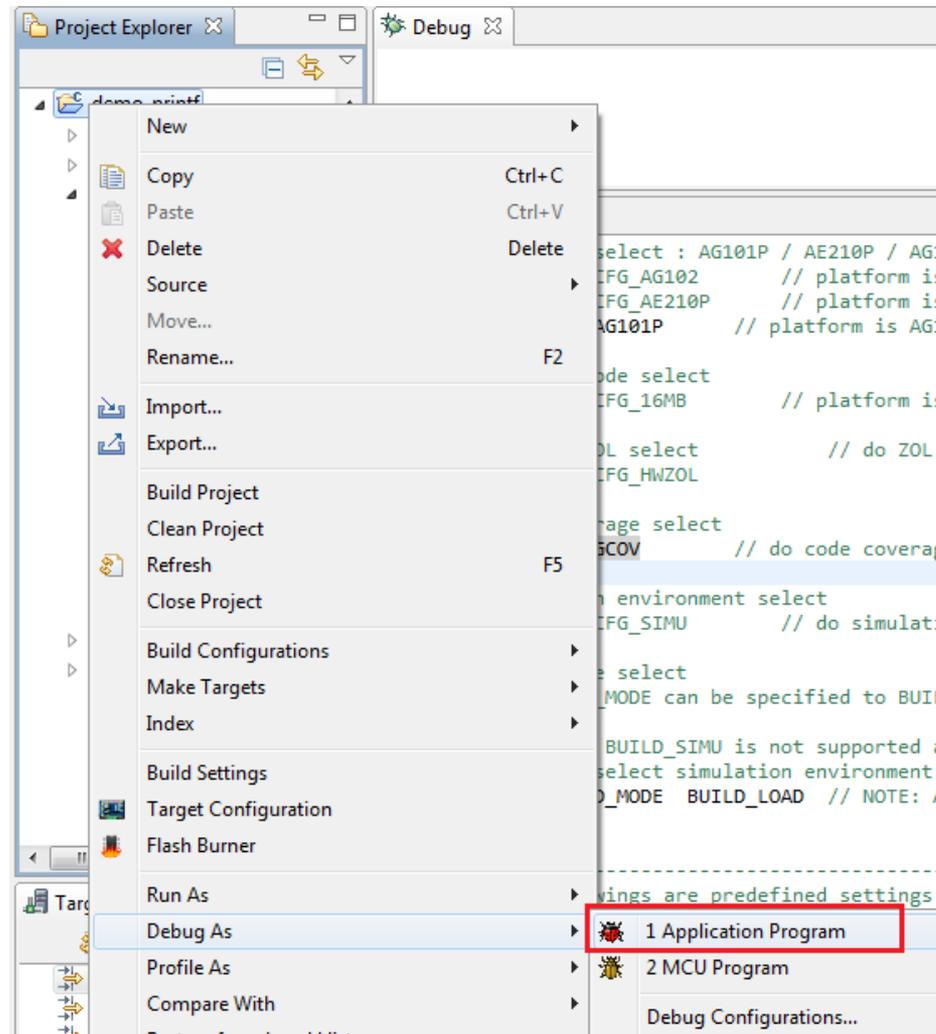
- ❖ AndeSight™ will build the project automatically after you have done the target configuration.



- ❖ Or, you can build the project by:
  - Click  on AndeSight™ toolbar to commence the build process.
  - Right-click the project folder to trigger the pull-down menu and select "Build Project".

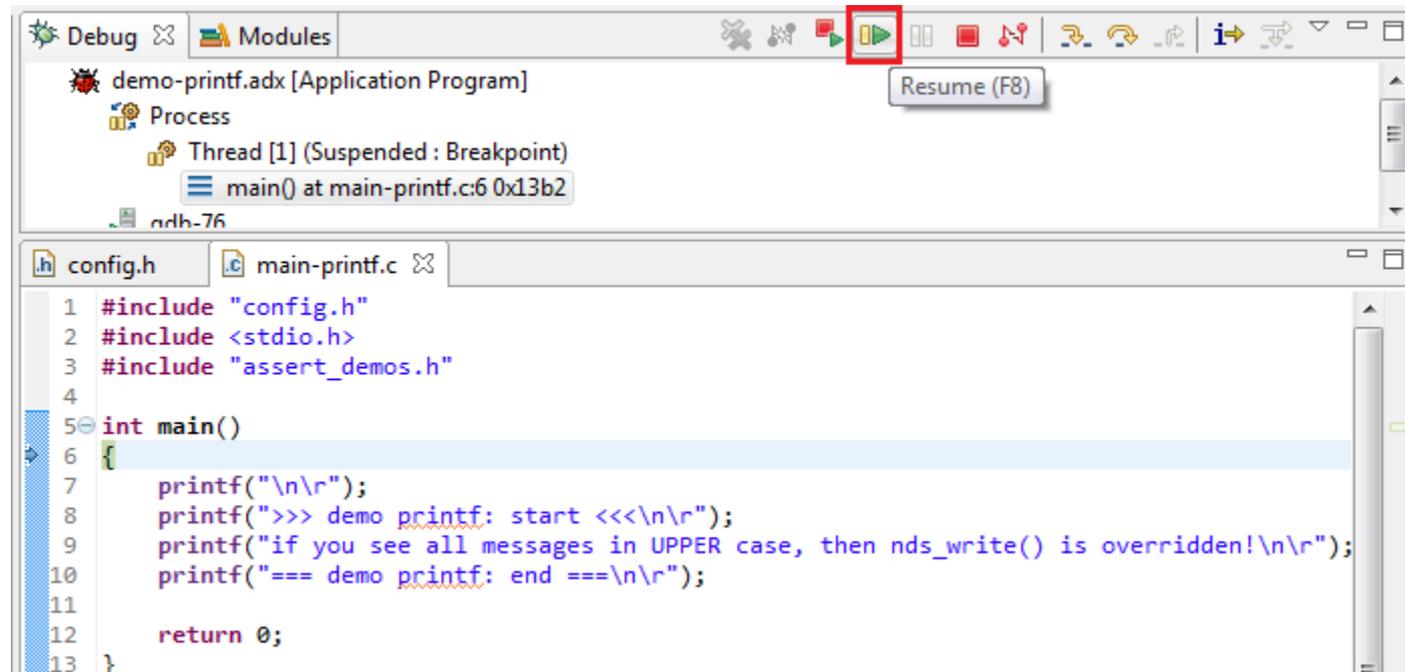
# Debug

❖ Right click on your project in Project Explorer view.



# Debug

- ❖ It will stop at main(), please click Resume to let your demo project free run.



# Debug

- ❖ Upon the completion of program execution, the gcov view containing the coverage data is brought to AndeSight automatically.

The screenshot displays the AndeSight IDE interface. The main window shows the source code of a C program named `demo-printf.adx`. The code includes headers `config.h`, `stdio.h`, and `assert_demos.h`. The `main` function prints several messages and returns 0. The program has been executed, as indicated by the status bar and the console output.

On the right side, the `gcov` window is open, showing a coverage report for the program. The report includes a table with the following data:

Name	Total Lines	Instrumented ...	Executed Lines	Coverage
Summary	1036	210	26	12.38%
init-default	276	38	3	7.89%
init-soc.c	71	3	3	100.0%
interrupt.c	311	134	0	0.0%
main-print	47	14	13	92.86%
stdio.h	216	1	0	0.0%
stdlib.h	67	1	0	0.0%
uart.c	48	19	7	36.84%

# Debug

- ❖ You may investigate the line coverage in source code by double-clicking an interested file or function in the gcov view. A code editor would show the source code with coverage highlighting.

The screenshot shows the gcov tool interface. The top window displays a table of coverage data for various files and functions. A red arrow points from the 'djpeg\_main' entry in the table to a code editor window showing the source code for 'djpeg.c' with the corresponding lines highlighted in green.

Name	Total Lines	Instrument...	Executed L...	Coverage %
Summary	16334	5445	1752	32.18%
djpeg.c	933	263	79	30.04%
djpeg_main	56	43	43	76.79%
endian_check	4	4	4	100.0%
fatal_error	2	0	0	
jpeg_getc	7	0	0	
main	14	14	14	

```
16 573 int djpeg_main (int argc, char **argv)
16 574 {
575 int fidx = argc;
576
577 struct jpeg_decompress_struct cinfo;
578 struct jpeg_error_mgr jerr;
579 #ifndef PROGRESS_REPORT
580 struct cdjpeg_progress_mgr progress;
581 #endif
```

# Debug

- ❖ Since the target is simulator, the result would be printed in Uart3 view.

