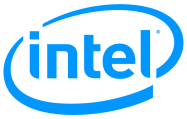
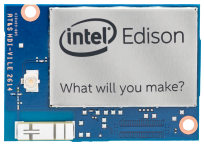
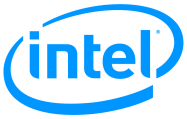




Intel® Edison Tutorial: Liquid-DSP – Installation Guide



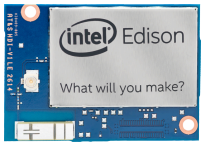


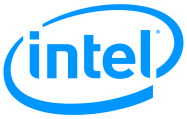
Table of Contents

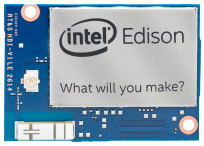
Introduction **3**

Prerequisite Tutorials **3**

List of Required Materials and Equipment **3**

Library Installation Procedure **4**





Introduction

liquid-dsp is a free and open source digital signal processing (DSP) library. It aims to provide support for building software defined radio applications on embedded platforms, without excessive external dependencies.

The official website is shown below.

www.liquidsdr.org

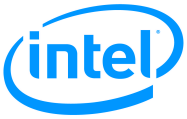
Prerequisite Tutorials

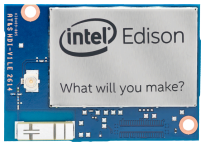
Users should ensure they are familiar with the documents listed below before proceeding.

1. Intel Edison Tutorial – Introduction, Linux Operating System Shell Access
2. Intel Edison Tutorial – Introduction to Linux
3. Intel Edison Tutorial – Introduction to OPKG
4. Intel Edison Tutorial – Introduction to Vim

List of Required Materials and Equipment

1. 1x Intel Edison Compute Module.
2. 2 x USB 2.0 A-Male to Micro-B Cable (micro USB cable).
3. 1x powered USB hub **OR** 1x external power supply.
4. 1x Personal Computer.





Library Installation Procedure

Follow the steps below in order to correctly configure and install the liquid-dsp library.

1. Access the shell on the Intel Edison. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction, Shell Access and SFTP***.
2. Ensure that the Intel Edison has the Vim editor installed. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction to Vim***.
3. Ensure that the Intel Edison has Git. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction to OPKG***.
4. Ensure that the Intel Edison has internet access.
5. Navigate to the home directory.

   $ cd

6. Clone the liquid-dsp library by issuing the command shown below.

   $ git clone git://github.com/jgaeddert/liquid-dsp.git

```
root@edison:~# git clone git://github.com/jgaeddert/liquid-dsp.git
Cloning into 'liquid-dsp'...
remote: Counting objects: 49397, done.
remote: Total 49397 (delta 0), reused 0 (delta 0), pack-reused 49397
Receiving objects: 100% (49397/49397), 20.36 MiB | 2.19 MiB/s, done.
Resolving deltas: 100% (30037/30037), done.
Checking connectivity... done.
root@edison:~# ls
liquid-dsp
```

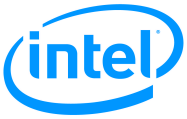
Figure 1: Successful cloning of liquid-dsp repo.

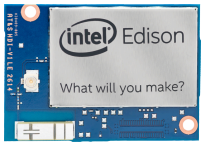
7. Enter the liquid-dsp directory and issue the following command.

   $ cd liquid-dsp
   $ git checkout v1.2.0

```
Note: checking out 'v1.2.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 875147b... build: consolidating .gitignore files
```

Figure 2: Successfully checking out v1.2.0 from the git repo.





8. Issue the commands below

   $ ./reconf
   ./reconf may take a few seconds before completing. This command should not produce output on stdout.

9. **$ ./configure**
   ./configure may take a few seconds to complete. It should output some text to stdout.

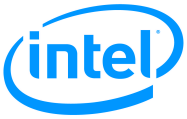
```
checking for size_t... yes
checking for uint32_t... yes
checking for uint8_t... yes
checking size of int... 4
checking size of unsigned int... 4
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
configure: creating ./config.status
config.status: creating makefile
config.status: creating doc/makefile
config.status: creating config.h
root@edison:~/liquid-dsp#
```

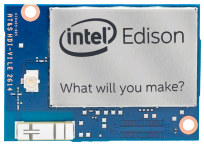
Figure 3: Truncated output from ./configure to stdout.

10. **$ make**
    make will take a few minutes to complete. It should output some text to stdout.

```
ar r liboptim.a src/optim/src/chromosome.o src/optim/src/gasearch.o src/optim/src/gradsearch.o src/optim
/src/optim.common.o src/optim/src/quasinewton_search.o src/optim/src/rosenbrock.o
ar: creating liboptim.a
ar r libquantization.a src/quantization/src/compand.o src/quantization/src/quantizercf.o src/quantizatio
n/src/quantizerf.o src/quantization/src/quantizer.inline.o
ar: creating libquantization.a
ar r librandom.a src/random/src/rand.o src/random/src/randn.o src/random/src/randexp.o src/random/src/ra
ndweib.o src/random/src/randgamma.o src/random/src/randnakm.o src/random/src/randricek.o src/random/src/
scramble.o
ar: creating librandom.a
ar r libsequence.a src/sequence/src/bsequence.o src/sequence/src/msequence.o
ar: creating libsequence.a
ar r libutility.a src/utility/src/bshift_array.o src/utility/src/count_bits.o src/utility/src/msb_index.
o src/utility/src/pack_bytes.o src/utility/src/shift_array.o
ar: creating libutility.a
gcc -lm -lc  -shared -Xlinker -soname=libliquid.so -o libliquid.so -Wl,-whole-archive libagc.a libaudio.
a libbuffer.a libdotprod.a libequalization.a libfec.a libfft.a libfilter.a libframing.a libmath.a libmat
rix.a libmodem.a libmulticarrier.a libnco.a liboptim.a libquantization.a librandom.a libsequence.a libut
ility.a -Wl,-no-whole-archive
```

Figure 4: Truncated output from make to stdout.





11. **$ make install**
    make install should complete very quickly. It should output some text to stdout.

```
root@edison:~/liquid-dsp# make install
installing...
mkdir -p /usr/local/lib
install -m 644 -p libliquid.so libliquid.a /usr/local/lib
mkdir -p /usr/local/include
mkdir -p /usr/local/include/liquid
install -m 644 -p include/liquid.h /usr/local/include/liquid

------------------------------------------------------------
  liquid-dsp was successfully installed.

  On some machines (e.g. Linux) you should rebind your
  libraries by running 'ldconfig' to make the shared
  object available.  You might also need to modify your
  LD_LIBRARY_PATH environment variable to include the
  directory /usr/local
------------------------------------------------------------

root@edison:~/liquid-dsp#
```

Figure 5: Output from "make install" to stdout.

If the output on stdout matches the above screenshot, the liquid-dsp library has been successfully installed.

12. To ensure the installation went correctly, issue the command below.

    $ make check
    make check should take a few minutes to complete. It should output some text to stdout.

```
================================
 WARNINGS : 34
================================
 PASSED ALL 42373 CHECKS
================================
```

Figure 6: Truncated output from "make check" to stdout.

If the output on stdout matches the above screenshot, the liquid-dsp library has been successfully installed. It is safe to ignore the 17 warnings that may appear.

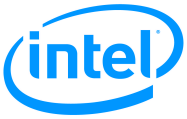
13. In order to enable compilation of C-code applications using the **–lliquid** flag, one must run the **ldconfig** command in order to make the shared object available.

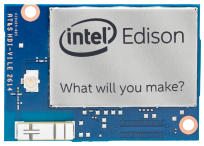
    Open the file **ld.so.conf** located in the **/etc/** and add the line **/usr/local/lib** to the file. Save and terminate the Vim editor application.

    $ vi /etc/ld.so.conf

```
/usr/local/lib
~
```

Figure 7: Contents of /etc/ld.so.conf after adding the required line.






```
root@edison:~/liquid-dsp# cat /etc/ld.so.conf
/usr/local/lib
root@edison:~/liquid-dsp#
```

Figure 8: Output from "cat /etc/ld.so.conf" command.

14. Issue the command below to create the necessary links and to cache the most recent shared libraries found in file **"/etc/ld.so.conf"**.

    $ ldconfig

15. Now that the shared library is available, let's attempt to compile a program. Issue the commands below in order to copy one of the existing files into a separate directory.

    $ cd
    $ mkdir dsp_test
    $ cp liquid-dsp/examples/fft_example.c dsp_test/

```
root@edison:~/liquid-dsp# cd
root@edison:~# mkdir dsp_test
root@edison:~# cp liquid-dsp/examples/fft_example.c dsp_test
root@edison:~#
```

Figure 9: Copying the fft_example.c file into a separate working directory.

16. Issue the commands below in order to edit the file.

    $ cd ~/dsp_test
    $ vi fft_example.c

```
root@edison:~# cd ~/dsp_test/
root@edison:~/dsp_test# vi fft_example.c
```

Figure 10: Editing the fft_example.c file

17. Edit the line **#include "liquid.h"** to be **#include <liquid/liquid.h>**

```
#include <stdio.h>
#include "liquid.h"
```

Figure 11: Relevant lines before editing.

```
#include <stdio.h>
#include <liquid/liquid.h>
```

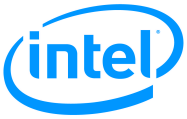
Figure 12: Relevant lines after editing.

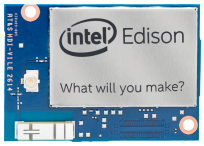
18. Save the changes and terminate the Vim editor application.

19. Compile the C-code source file into an executable binary.

    $ gcc –o fft_example fft_example.c –lm –lc –lliquid

    NOTE: there is an 'L' before 'LIQUID'. The flag is '-LLIQUID' (all in lower case).





20. Execute the binary file by issuing the following command.

$ ./fft_example

```
root@edison:~/dsp_test# gcc -o fft_example fft_example.c -lm -lc -lliquid
root@edison:~/dsp_test# ./fft_example
fft plan [forward], n=16, Cooley-Tukey
16, Cooley-Tukey mixed radix, Q=8, P=2
  8, DFT
  2, DFT
rmse =   4.5637e-07
done.
root@edison:~/dsp_test#
```

Figure 13: Truncated output with successful compilation and execution of fft_example.