Library Installation and Usage Guide for Boardcon EM2440-III

By Team Echo (Gordon Leung, Lily Wang, Rodrick Yu)

This document guides the user through:

1. Setting up the host and target environments to easily configure and target libraries
2. Configuring and cross-compiling libraries in general as well as several specific libraries
3. Configuring the Kernel to use the V4L and ALSA libraries
4. Compiling Makefile and Qt projects that target cross-compiled libraries

Table of Contents

Introduction

This document will show step-by-step how to cross-compile libraries. In doing so, we aim to do two things:

1. Work around the issue of install paths that are hardcoded into the library files
2. Use `pkg-config` to retrieve compiler and linker flags during project compilation

This guide assumes that the libraries will be installed on `~/cmpt433/private/library` on the host and `/mnt/remote` on the target. If different directories are being used, adjust those paths in the guide accordingly.

Prepare the Host and Target

1. Create the library folder on the host
   ```
   > mkdir ~/cmpt433/private/library
   > sudo ln -s ~/cmpt433/private/library /mnt/remote
   ```

2. Add the following to `~/.profile` on the host
   ```
   PKG_CONFIG_PATH="/mnt/remote/lib/pkgconfig:$PKG_CONFIG_PATH"
   export PKG_CONFIG_PATH
   ```

3. Log-out and log back in to take effect.

   - Check if the environment variable is set:
     ```
     > echo $PKG_CONFIG_PATH
     ```
     Make sure that this returns `/mnt/remote/lib/pkgconfig`

4. Add the following to `/etc/profile` on the target
   ```
   LD_LIBRARY_PATH="/mnt/remote/lib:$LD_LIBRARY_PATH"
   export LD_LIBRARY_PATH
   ```

   - If you have used Brian's QT INSTALL GUIDE, make sure this line is below `source /etc/qt_env.sh` to prevent that script from clobbering the library path.

Install Libraries to Host

General Install Procedures

This should work for many simple libraries with little to no modification.

1. Download and decompress the library source
   For `tar` files: `tar -xvf [file-name]`
   For `zip` files: `unzip [file-name]`

2. Change to the library's source directory
   ```
   > cd [source-directory-name]
   ```

3. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

4. Build the library
   ```
   > make
   ```

5. Install the library
   ```
   > make install
   ```

6. Troubleshooting

   - If you get this error while configuring: `configure: error: unrecognized option:` then the general install procedure will not work due to different configuration options. Read the options available to try to resolve configure errors:
     ```
     > ./configure --help
     ```

   - If you get this warning while compiling: `library search path "/usr/local/lib" is unsafe for cross-compilation` this is likely due to one or more of the following:

     i. The library is dependent on another library that you did not cross-compile yet. Check what dependencies are needed:
     ```
     > ./configure --help
     ```

     Under `Some influential environment variables` you will see environment variables for specific libraries that you can manually set.

     For example:
     ```
     GLIB_CFLAGS C compiler flags for GLIB, overriding pkg-config
     GLIB_LIBS   linker flags for GLIB, overriding pkg-config
     GLIB_ONLY_CFLAGS
                 C compiler flags for GLIB_ONLY, overriding pkg-config
     GLIB_ONLY_LIBS
                 linker flags for GLIB_ONLY, overriding pkg-config
     ```

     This would indicate that `GLIB` is a dependent library that needs to be cross-compiled first.

     ii. You did not properly set the `pkg-config` path environment variable (see the PREPARE THE HOST AND TARGET section)

iii. The library configuration ignores the `pkg-config` path for a dependent library. You will have to manually set the `CFLAGS` and `LDFLAGS`.

For example, if `GLIB` is a dependent library:
Get the `CFLAGS` and `LDFLAGS` for `GLIB`:
```
> pkg-config --cflags --libs glib-2.0
```

This returns:
```
-I/mnt/remote/include/glib-2.0 -I/mnt/remote/lib/glib-2.0/include
-L/mnt/remote/lib -lglib-2.0
```

Then in the configure flags,
Add to `CFLAGS` and `CPPFLAGS`:
```
-I/mnt/remote/include/glib-2.0 -I/mnt/remote/lib/glib-2.0/include
```

Add to `LDFLAGS`:
```
-L/mnt/remote/lib -lglib-2.0
```

iv. A dependent library does not have a `pkg-config` entry.
In the configure flags,
Add to `CFLAGS` and `CPPFLAGS`:
```
-I/mnt/remote/include
```

Add to `LDFLAGS`:
```
-L/mnt/remote/lib
```

- If you get this error while installing: `- install: cannot … : Permission denied` then the `PREFIX` path has not been properly set.

  Check the configure options to see if the `PREFIX` path can be set using a different flag.

- If the library compiles but results in an `illegal instruction` upon use, then you may have compiled the library for the wrong target.

  Check the library file (usually `libraryname.so`) under `~/cmpt433/private/library/lib` to see if it's been properly cross-compiled:
  ```
  > readelf [library-filename] –hA
  ```

  The result should include the following:
  ```
  Machine:                           ARM
  Tag_CPU_name: “4T”
  Tag_CPU_arch: v4T
  ```

If these values are different, then the libraries are not targeted correctly. Check the configure options to see if the `CFLAGS`, `CPPFLAGS`, and the `LDFLAGS` can be set in a different way.

Install Procedures for Specific Libraries

The following is a list of libraries that we've successfully compiled. For general troubleshooting, see the TROUBLESHOOTING section under GENERAL INSTALL PROCEDURES.

ALSA LIB

1. Download alsa-lib from: ftp://ftp.alsa-project.org/pub/lib/alsa-lib-1.0.24.1.tar.bz2

2. Decompress the library source
   ```
   > tar -xvf alsa-lib-1.0.24.1.tar.bz2
   ```

3. Change to the library's source directory
   ```
   > cd alsa-lib-1.0.24.1
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

FFMPEG

1. Download ffmpeg from: http://ffmpeg.org/releases/ffmpeg-0.8.7.tar.bz2

2. Decompress the library source
   ```
   > tar -xvf ffmpeg-0.8.7.tar.bz2
   ```

3. Change to the library's source directory
   ```
   > cd ffmpeg-0.8.7
   ```

4. Configure the library for the target
   ```
   > ./configure --cross-prefix=arm-linux- --enable-cross-compile --arch=arm --prefix=/mnt/remote --disable-asm --target-os=linux --disable-static --disable-armv6 --disable-armv5te --disable-armv6t2 --disable-neon --disable-armvfp --disable-ffplay --host-cc=gcc --disable-mmx --enable-shared --disable-optimizations --enable-gpl --extra-cflags="-msoft-float -march=armv4t -mtune=arm920t" --extra-ldflags='-march=armv4t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

LIBJPEG

1. Download libjpeg from:
   http://sourceforge.net/projects/libjpeg/files/latest/download?_test=goal

2. Decompress the library source
   ```
   > unzip jpegsr6.zip
   ```

3. Change to the library's source directory
   ```
   > cd jpeg-6b
   ```

4. Install dos2unix
   ```
   > sudo apt-get install dos2unix
   ```

5. Fix the configure file
   ```
   > dos2unix configure
   ```

6. Configure the library for the target
   ```
   > ./configure CC='arm-linux-gcc' CFLAGS='-march=armv4t -mtune=arm920t -
   mcpu=arm920t -Wall' --prefix=/mnt/remote --target=armv4t
   ```

7. Build the library
   ```
   > make
   ```

8. Install the library
   ```
   > make install-lib
   ```

9. Troubleshooting

   - If you get this error:
     ```
     /usr/bin/install: cannot create regular file
     `/mnt/remote/include/jconfig.h': No such file or directory
     ```

     Create the missing folders
     ```
     > mkdir /mnt/remote/include
     > mkdir /mnt/remote/lib
     ```

LIBV4L

1. Install Dependencies: LIBJPEG

2. Download libv4l from: http://freecode.com/projects/libv4l

3. Decompress the library source
   ```
   > tar –xvf v4l-utils-0.8.5.tar.bz2
   ```

4. Change to the library's source directory
   `> cd v4l-utils-0.8.5`
5. Modify the file `Make.rules`
   - Open the file with `gedit`
     `> gedit Make.rules`
   - Change the line `CFLAGS := -g -O1`
     to `CFLAGS := -g -O1 -march=armv4t -mtune=arm920t -mcpu=arm920t`
   - Change the line `PREFIX = /usr/local`
     to `PREFIX = /mnt/remote`
6. Modify the file `lib/libv4lconvert/Makefile`
   - Open the file with `gedit`
     `> gedit Make.rules`
   - Change the line `LIBS_libv4lconvert = -lrt -lm -ljpeg`
     to `LIBS_libv4lconvert = -lrt -lm -L/mnt/remote/lib -ljpeg`
7. Build the library
   `> make CC='arm-linux-gcc' CXX='arm-linux-g++'`
8. Install the library
   `> make install`
9. Troubleshooting
   - If you get this error while compiling:
     `Assembler messages - bad instruction`
     This is okay since all of the required files are compiled at this point.
   - If you get this error while installing:
     `install: cannot remove `/etc/rc_maps.cfg': Permission denied`
     This is okay since all of the required files are installed at this point.

LIBFFI

1. Download libffi from: ftp://sourceware.org/pub/libffi/libffi-3.0.10.tar.gz
2. Decompress the library source
   `> tar -xvf libffi-3.0.10.tar.gz`
3. Change to the library's source directory
   `> cd libffi-3.0.10`

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

ZLIB

1. Download zlib from: http://zlib.net/zlib-1.2.5.tar.gz

2. Decompress the library source
   ```
   > tar -xvf zlib-1.2.5.tar.gz
   ```

3. Change to the library's source directory
   ```
   > cd zlib-1.2.5
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote
   ```

5. Modify the generated `Makefile`

   - Open the file with `gedit`
     ```
     > gedit Makefile
     ```

   - Modify the variables to match the following
     ```
     CC=arm-linux-gcc
     CFLAGS=-O0 -D_LARGEFILE64_SOURCE=1 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t
     SFLAGS=-O0 -fPIC -D_LARGEFILE64_SOURCE=1 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t
     LDFLAGS= -L. libz.a -march=armv4t -mcpu=arm920t
     TEST_LDFLAGS=-L. libz.a -march=armv4t -mcpu=arm920t
     LDSHARED=arm-linux-gcc -shared -Wl,-soname,libz.so.1,--version-script,zlib.map
     CPP=arm-linux-gcc -E
     AR=arm-linux-ar rc
     RANLIB=arm-linux-ranlib
     ```

6. Build the library
   ```
   > make
   ```

7. Install the library
   ```
   > make install
   ```

EXPAT

1. Download expat from: http://sourceforge.net/projects/expat/

2. Decompress the library source
   ```
   > tar -xvf expat-2.0.1.tar.gz
   ```

3. Change to the library's source directory
   ```
   > cd expat-2.0.1
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

DBUS

1. Install Dependencies: EXPAT

2. Download glib from: http://dbus.freedesktop.org/releases/dbus/dbus-1.4.16.tar.gz

3. Decompress the library source
   ```
   > tar -xvf dbus-1.4.16.tar.gz
   ```

4. Change to the library's source directory
   ```
   > cd dbus-1.4.16
   ```

5. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t -I/mnt/remote/include' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t -I/mnt/remote/include' 'LDFLAGS=-march=armv4t -mcpu=arm920t -L/mnt/remote/lib'
   ```

6. Build the library
   ```
   > make
   ```

7. Install the library
   ```
   > make install
   ```

GLIB

1. Install Dependencies: LIBFFI, ZLIB, DBUS

2. Download glib from:
   http://ftp.gnome.org/pub/gnome/sources/glib/2.30/glib-2.30.2.tar.bz2

3. Decompress the library source
   ```
   > tar -xvf glib-2.30.2.tar.bz2
   ```

4. Change to the library's source directory
   ```
   > cd glib-2.30.2
   ```

5. Create a new file named `arm-linux-cache` with the following content
   ```
   glib_cv_have_qsort_r=no
   glib_cv_stack_grows=no
   glib_cv_uscore=no
   ac_cv_func_posix_getgrgid_r=yes
   ac_cv_func_posix_getpwuid_r=yes
   ```

6. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux --cache=arm-linux-cache 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

7. Build the library
   ```
   > make
   ```

8. Install the library
   ```
   > make install
   ```

LIBXML

1. Download libxml from: ftp://xmlsoft.org/libxml2/libxml2-git-snapshot.tar.gz

2. Decompress the library source
   ```
   > tar -xvf libxml2-git-snapshot.tar.gz
   ```

3. Change to the library's source directory
   ```
   > cd libxml2-git-snapshot
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

GSTREAMER

1. Install Dependencies: GLIB, LIBXML

2. Download gstreamer from:
   http://gstreamer.freedesktop.org/src/gstreamer/gstreamer-0.10.35.tar.bz2

3. Decompress the library source
   ```
   > tar -xvf gstreamer-0.10.35.tar.bz2
   ```

4. Change to the library's source directory
   ```
   > cd gstreamer-0.10.35
   ```

5. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t -I/mnt/remote/include/libxml2' 'LDFLAGS=-march=armv4t -mcpu=arm920t -L/mnt/remote/lib -lxml2' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t -I/mnt/remote/include/libxml2'
   ```

6. Build the library
   ```
   > make
   ```

7. Install the library
   ```
   > make install
   ```

LIBOGG

1. Download libogg from: http://downloads.xiph.org/releases/ogg/libogg-1.3.0.tar.gz

2. Decompress the library source
   ```
   > tar -xvf libogg-1.3.0.tar.gz
   ```

3. Change to the library's source directory
   ```
   > cd libogg-1.3.0
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

LIBVORBIS

1. Download libvorbis from: http://downloads.xiph.org/releases/vorbis/libvorbis-1.3.2.tar.gz

2. Decompress the library source
   ```
   > tar -xvf libvorbis-1.3.2.tar.gz
   ```

3. Change to the library's source directory
   ```
   > cd libvorbis-1.3.2
   ```

4. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t'
   ```

5. Build the library
   ```
   > make
   ```

6. Install the library
   ```
   > make install
   ```

GST-PLUGIN-BASE

1. Install Dependencies: ALSA, GSTREAMER, LIBOGG, LIBVORBIS, (Optionally: PANGO, LIBTHEORA, and others…)

2. Download gst-plugin-base from:
   http://gstreamer.freedesktop.org/src/gst-plugins-base/gst-plugins-base-0.10.35.tar.bz2

3. Decompress the library source
   ```
   > tar -xvf gst-plugins-base-0.10.35.tar.bz2
   ```

4. Change to the library's source directory
   ```
   > cd gst-plugins-base-0.10.35
   ```

5. Configure the library for the target
   ```
   > ./configure --prefix=/mnt/remote --host=arm-linux 'CFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' 'LDFLAGS=-march=armv4t -mcpu=arm920t' 'CPPFLAGS=-O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t' --disable-pango --disable-examples --without-x --without-ft
   ```

6. Build the library
   ```
   > make
   ```

7. Install the library
   ```
   > make install
   ```

8. Troubleshoot

   - If you get this error while compiling: `libgstinterfaces-0.10.so.0, needed by ./.libs/libgstaudio-0.10.so, not found`

     Open the file `gst-libs/gst/audio/Makefile.am`
     ```
     > gedit gst-libs/gst/audio/Makefile.am
     ```

Modify the variable `testchannels_LDADD` to match the following:

```
testchannels_LDADD = $(builddir)/libgstaudio-$(GST_MAJORMINOR).la
$(builddir)/../interfaces/libgstinterfaces-$(GST_MAJORMINOR).la
$(GST_LIBS)
```

Install Libraries to Target

Copy the /lib and /share directories to the public directory.

```
> cp -r ~/cmpt433/private/library/lib ~/cmpt433/public
> cp -r ~/cmpt433/private/library/share ~/cmpt433/public
```

Remember to do this every time new libraries have been installed.

Configure the Kernel for LIBV4L and ALSA LIB

Enter the Kernel configuration.

```
> make menuconfig
```

V4L support

```
-Device drivers

--Multimedia devices

---Video for Linux [*]

---Video capture adapters [*]

----Autoselect pertinent encoders/decoders and other helper... [*]

----V4L USB devices [*]

-----USB Video Class [*]

-----UVC input events device support [*]
```

ALSA /w Webcam Microphone support

```
-Device drivers

--Sound card support [*]

---Advanced Linux Sound Architecture [*]

----USB sound devices [*]

-----USB Audio/MIDI driver [*]
```

Link Libraries in a Project using PKG-CONFIG

In a standard Makefile Project

Add the following to the `Makefile`

```
LIBS = [list of library names]
CFLAGS = $(shell pkg-config --cflags $(LIBS))
LDFLAGS = $(shell pkg-config --libs $(LIBS))
```

Then add `$(CFLAGS)` and `$(LDFLAGS)` to the compile statement.

For example:

```
LIBS = gstreamer-0.10 zlib
CFLAGS = -O0 -msoft-float -march=armv4t -mtune=arm920t -mcpu=arm920t $(shell
pkg-config --cflags $(LIBS))
LDFLAGS = -march=armv4t -mcpu=arm920t $(shell pkg-config --libs $(LIBS))

helloworld:
      arm-linux-gcc -Wall $(CFLAGS) helloworld.c -o helloworld $(LIBS)
```

The resulting compilation looks like:

```
arm-linux-gcc -Wall -O0 -msoft-float -march=armv4t -mtune=arm920t -
mcpu=arm920t -pthread -I/mnt/remote/include/gstreamer-0.10 -
I/mnt/remote/include/glib-2.0 -I/mnt/remote/lib/glib-2.0/include -
I/mnt/remote/include/libxml2 -I/mnt/remote/include   helloworld.c -o
helloworld -march=armv4t -mcpu=arm920t -pthread -L/mnt/remote/lib -
lgstreamer-0.10 -lgobject-2.0 -lgmodule-2.0 -lxml2 -lgthread-2.0 -lrt -lglib-
2.0 –lz
```

In a Qt Project

Add this to the `.pro` file:

```
unix: CONFIG += link_pkgconfig
unix: PKGCONFIG += [list of library names]
```

For example:

```
TEMPLATE = app
TARGET =
DEPENDPATH += .
INCLUDEPATH += .

# Input
HEADERS += mainwindow.h
FORMS += mainwindow.ui
SOURCES += mainwindow.cpp
unix: CONFIG += link_pkgconfig
unix: PKGCONFIG += gstreamer-0.10 zlib
```

CAUTION: These environment variables will be removed when the `.pro` file is regenerated.

PKG-CONFIG Library name reference

ALSA LIB

alsa

FFMPEG

libavcodec	libavdevice	libavfilter
libavformat	libavutil	libpostproc
libswscale		

LIBV4L

libv4l1	libv4l2	libv4lconvert

LIBFFI

libffi

ZLIB

zlib

DBUS

dbus-1

GLIB

gio-2.0	gio-unix-2.0	glib-2.0
gmodule-2.0	gmodule-export-2.0	gmodule-no-export-2.0
gobject-2.0 gthread-2.0		

LIBXML

libxml-2.0

GSTREAMER

gstreamer-0.10	gstreamer-net-0.10	gstreamer-app-0.10
gstreamer-netbuffer-0.10	gstreamer-audio-0.10	gstreamer-pbutils-0.10
gstreamer-base-0.10	gstreamer-cdda-0.10	gstreamer-plugins-base-0.10
gstreamer-riff-0.10	gstreamer-check-0.10	gstreamer-floatcast-0.10
gstreamer-rtsp-0.10	gstreamer-rtp-0.10	gstreamer-controller-0.10
gstreamer-sdp-0.10	gstreamer-fft-0.10	gstreamer-dataprotocol-0.10
gstreamer-tag-0.10	gstreamer-video-0.10	gstreamer-interfaces-0.10

LIBOGG

ogg

LIBVORBIS

vorbis	vorbisenc	vorbisfile