TODO

See also the following links:

- https://github.com/scintill/android_frameworks_opt_telephony_ril_ofono
- https://redmine.replicant.us/issues/1958
- https://redmine.replicant.us/issues/1813
- Replicant and oFono based Java RIL Presentation: https://redmine.replicant.us/projects/replicant/wiki/ContributorsMeetingJuly2019#Presentations
- https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/clk/qcom/gcc-mdm9615.c?h=v5.2.7

QMI-RIL

This page covers development efforts to create a Radio Interface Layer (RIL) for modems that use the Qualcomm MSM Interface (QMI) protocol. The work is currently carried out for the Galaxy S 3 4G (I9305) which has a Qualcomm MDM9615 modem. The Galaxy Note 2 4G (t0lte) uses the same modem and this device could be supported as well in Replicant when the QMI-RIL is ready. We will hopefully identify further candidates that have an isolated modem using the QMI protocol and for which Replicant support could be added. Among the non-Android devices, the iPhone 5 is one that uses the MDM9615 modem as well.

relevant thread on libqmi mailing list

Modem boot

Background

When the MDM9615 modem is activated, it first boots into a Download mode. There, firmware, bootloaders and EFS files are uploaded to the modem via a serial interface. The used protocol is called SAHARA and it is a proprietary protocol developed by Qualcomm. The protocol covers various functionalities. We likely only need to implement a very small subset that is responsible for transferring firmware and other files to the modem and synchronising changes to EFS files with the filesystem. Being able to retrieve RAM dumps from the modem would also be helpful for debugging. In case of a fatal error, the Linux kernel already requests RAM dumps from the modem, so only the file transfer part needs to be implemented.

Qualcomm provides the proprietary tool kickstart for uploading the files to the modem. It is run by the daemon qcks. After the modem is booted, qcks spawns efsks which is responsible for EFS sync.

Logcat and dmesg output of the upload: ks_logcat, ks_dmesg

libopenpst, their sahara tool, and Linaro's QDL code can be used as a reference to implement the file upload and as documentation about the protocol. The kernel itself can be checked for the different boot modes.

Implementation

An initial version of the modem boot part is completed and a tool called modem-boot is available as part of the qmi-ril repo: https://git.replicant.us/contrib/wiewo/qmi-ril/
It uploads the needed files in the Download mode and reboots the modem. The files are uploaded from a directory that qcks uses to prepare the files. This preparation step (reading data from partitions and preparing them in a certain way before the upload) needs to be implemented. Otherwise, updates to the EFS partitions aren't used at the next boot of the modem.
During regular operation of the modem, EFS data is received. The data is not yet written to the EFS partition. This step needs to be implemented as well.
See the commit messages for more details.

QMI protocol

libqmi is a library that implements the QMI protocol and it will be used for implementing the RIL. The source code for the command-line tool qmicli, which is part of libqmi, and ModemManager's code is helpful for figuring out how to use libqmi.

libqmi needs the cdc-wdm and qmi_wwan kernel drivers for communicating with the modem. These are backported to the smdk4412 kernel on the qmi branch. The qmi_wwan driver replaces the RMNET usb driver that the blobs use for a network interface to the modem. Some missing code in the backported qmi_wwan driver was ported from the RMNET driver. The commit messages offer more details.

libqmi provides the command-line tool qmicli to communicate with the modem. It was already successfully tested on the S 3 4G. Using the modem boot tool described above and the qmi branch of the kernel, qmicli is usable with only free software.

The RIL header documents the Android side of the interface and the commands that need to be implemented. The Samsung-RIL page also offers some documentation in that regard.

Status of QMI-RIL

An initial version of QMI-RIL is available in the qmi-ril repo See this commit message for the status details.
The device-specific repos for i9305 need to use these two branches to make the RIL work:
https://code.fossencdi.org/device_samsung_i9305.git/log/?h=qmi-wip
https://git.replicant.us/contrib/wiewo/device_samsung_smdk4412-qcom-common/log/?h=replicant-6.0

Cross-compiling libqmi for Android on ARM

These instructions can be used to build libqmi and include it in an image for the Galaxy S 3 4G. See the build instructions for the Galaxy S 3 4G for building the image.

Android.mk files need to be written to integrate libqmi and its dependencies into a regular device build. Some of the source code is auto-generated and some dependencies like GLib have a lot of source files, so writing the Android Makefiles will be some work. For GLib, these commits could be used for creating the Android.mk files:
https://github.com/scintill/android_external_glib/commit/9bc8d813979140b8abdad77619aba20f08b19c6f
https://github.com/scintill/android_external_glib/commit/ae860f678520471da44823f500f302a1a27c1be9
They are for GLib 2.32, but libqmi requires at least GLib 2.36, so they need to be ported to that version. It looks like that with these Makefiles, only libiconv is required as an additional dependency for GLib. For libiconv, the Makefiles from this commit could be helpful:
https://github.com/tguillem/android-libiconv/commit/8be7e8a7670abf251d6198606098a1908ec1033c

Based on these instructions

Dependencies

```
apt-get install groff libltdl-dev pkg-config gtk-doc-tools
```

Getting the source code

```
git clone https://code.fossencdi.org/external_libqmi.git external/libqmi
git clone https://git.savannah.gnu.org/git/libiconv.git external/libiconv -b v1.15
git clone https://github.com/libffi/libffi external/libffi -b v3.2.1
git clone https://code.fossencdi.org/external_gettext.git external/gettext
git clone https://github.com/GNOME/glib external/glib -b 2.52.3
```

Configure and build

The attached script qmi_build_envsetup.sh makes it more convenient to configure the environment. You need to set the Replicant source tree root folder as the REPLICANT_BASE variable at the top of the script. Then you can source it:

```
. path/to/qmi_build_envsetup.sh
```

It's recommended to not run this and the following commands in the same shell you use for building a Replicant image as the environment variables break the Replicant build. Run the commands after you built an image for the I9305.

libiconv

```
./autogen.sh
./configure --build=${BUILD_SYSTEM} --host=arm-eabi --prefix=${PREFIX} --disable-rpath
make install -j8
```

If autoconf-2.69 and autoheader-2.69 are not found, remove the -2.69 suffix from the AUTOCONF and AUTOHEADER variables in Makefile.devel, libcharset/Makefile.devel and preload/Makefile.devel.

libffi

```
./autogen.sh
```

```
sed -e '/^includesdir/ s/$(libdir).*$/$(includedir)/' -i include/Makefile.in
sed -e '/^includedir/ s/=.*$/=@includedir@/' -e 's/^Cflags: -I${includedir}/Cflags:/' -i libffi.pc
.in
./configure --build=${BUILD_SYSTEM} --host=arm-eabi --prefix=${PREFIX} --enable-static
make install -j8
```

gettext (probably not needed)

```
./autogen.sh
./configure --build=${BUILD_SYSTEM} --host=arm-eabi --prefix=${PREFIX} --disable-rpath --disable-l
ibasprintf --disable-java --disable-native-java --disable-openmp --disable-curses
make install -j8
```

glib

First, the file android.cache with the following content needs to be created:

```
glib_cv_long_long_format=ll
glib_cv_stack_grows=no
glib_cv_sane_realloc=yes
glib_cv_have_strlcpy=no
glib_cv_va_val_copy=yes
glib_cv_rtldglobal_broken=no
glib_cv_uscore=no
glib_cv_monotonic_clock=no
ac_cv_func_nonposix_getpwuid_r=no
ac_cv_func_posix_getpwuid_r=no
ac_cv_func_posix_getgrgid_r=no
glib_cv_use_pid_surrogate=yes
ac_cv_func_printf_unix98=no
ac_cv_func_vsnprintf_c99=yes
ac_cv_func_realloc_0_nonnull=yes
ac_cv_func_realloc_works=yes
```

Make it read-only:

```
chmod 444 android.cache
```

Then configure and build:

```
./autogen.sh --build=${BUILD_SYSTEM} --host=${TOOLCHAIN_PREFIX} --prefix=${PREFIX} --disable-depen
dency-tracking --cache-file=android.cache --enable-included-printf --enable-static --with-pcre=no
--enable-libmount=no
make install -j8
```

libqmi

```
./autogen.sh --build=${BUILD_SYSTEM} --host=${TOOLCHAIN_PREFIX} --prefix=${PREFIX} --enable-mbim-q
mux=false --enable-firmware-update=false --enable-qmi-username=radio --without-udev --disable-mm-r
untime-check  --with-udev-base-dir=${PREFIX}/etc/udev
make install -j8
```

Files

ks_logcat	7.44 KB	06/23/2017	Wolfgang Wiedmeyer
ks_dmesg	18.2 KB	06/23/2017	Wolfgang Wiedmeyer
qmi_build_envsetup.sh	1.25 KB	06/24/2017	Wolfgang Wiedmeyer