PERFECT WIRELESS EXPERIENCE

FG132

QMI User Guide

V1.0

Disclaimer

Any action you take in the course of using this document is at your own risk, and Fibocom shall not be liable for any damages or losses under any circumstances. Due to product version upgrade or other reasons, Fibocom reserves the right to modify any information in this document at any time without prior notice and any responsibility. Unless otherwise agreed, all statements, information and suggestions in this document do not constitute any express or implied guarantee.

This document may include the third-party information covering products, services, software, data, and so on. Fibocom does not control and assumes no responsibility for the third-party content, including but not limited to the accuracy, compatibility, reliability, availability, legitimacy, appropriateness, performance, non-infringement, and status update, unless otherwise specified in this document. Fibocom does not provide any guarantee or authorization for the third-party content mentioned or referenced in this document. If you need a third-party license, obtain it in an authorized or legal way, unless otherwise specified in this document.

Copyright Notice

Unless specially authorized by Fibocom, the recipient of the documents shall keep the documents and information received confidential, and shall not use them for any purpose other than the implementation and development of this project. Without the written permission of Fibocom, no unit or individual shall extract or copy part or all of the contents of this document without authorization, or transmit them in any form. Fibocom has the right to investigate legal liabilities for any offense and tort in connection with violation of confidentiality obligations, or unauthorized use or malicious use of the said documents and information in other illegal forms.

Trademark Statement

Fibocom The trademark is registered and owned by Fibocom Wireless Inc.

Other trademarks, product names, service names and company names appearing in this document are owned by their respective owners.

Contact Information

Website: https://www.fibocom.com

Address: 10/F-14/F, Block A, Building 6, Shenzhen International Innovation Valley, Dashi First Road, Xili Community, Xili Subdistrict, Nanshan District, Shenzhen

Tel: 0755-26733555

Contents

Change History

V1.0 (2024-08-22)	Initial version.

Applicable Model

No.	Applicable Model	Description
1	FG132	None

1 Introduction

QMI (Qualcomm Messaging Interface) is a service interface of the Qualcomm platform based on the C/S architecture. The QMI interface supports:

- Synchronous and asynchronous interfaces.
- Communication between multiple processors.
- Good scalability.
- Concurrent running of multiple clients and multiple servers, and each server can support multiple clients.
- The server supports different versions.

Compared with AT command interfaces based on serial text processing, the QMI interface is more friendly to application programming.

The following figure shows the software block diagram of the QMI.

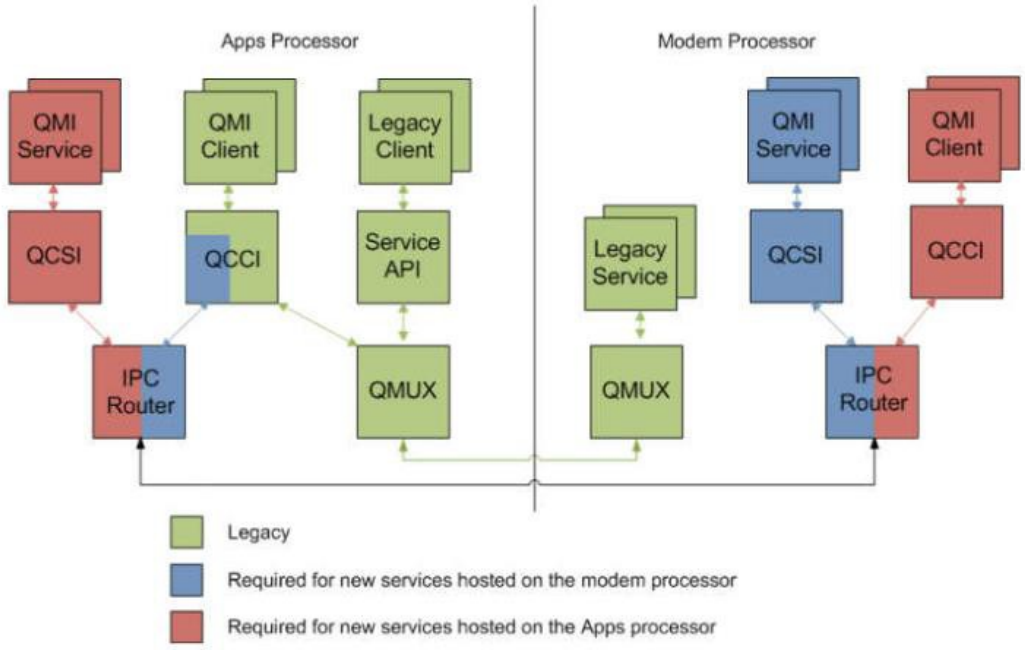

Figure 1. Software block diagram of the QMI

This document introduces how FG132 OpenSDK-based applications use the QMI interface.

2 Compilation

The FG132 OpenSDK compilation system is based on OpenWRT. If the application needs to use the QMI interface, refer to the following steps to configure the application:

1. Add the path of the QMI interface header file, reference path, and QMI component dependencies to the corresponding package makefile.

```
owrt_workspace/owrt/package/fibocom/sample-uim-qmi/
└── Makefile
```

```
10 LOCAL_SRC := $(FPKG_CURDIR)
11
12 include $(INCLUDE_DIR)/package.mk
13
14 define Package/sample-uim-qmi
15  SECTION:=APP
16  CATEGORY:=Fibocom
17  TITLE:=sample-uim-qmi
18  SUBMENU:=app
19  #USERID:=radio=1001:radio=1001 radio=1001:diag=2901 radio=1001:inet=3003 radio=1001:net_admin=3005 radio=1001:system=1000 radio=1001:wifi=1010 radio=10
20  DEPENDS:=+common +libqmi +qmi +qmiservices +qmiidl +qmi-framework +libqmi-client-qmux
21 endef
22
23 define Package/src/Description
24  data/fibo-test
25 endef
26
27 TARGET_CFLAGS += -g -I../inc -I$(TOPDIR)/src/vendor/qcom/proprietary/qcril-qmi-services-headers -I$(STAGING_DIR)/usr/include/qmi
28
29 TARGET_LDFLAGS += -L$(STAGING_DIR)/usr/lib
30
31 define Build/Prepare
32 #$(warning "**********Host prepare **********")
33     mkdir -p $(PKG_BUILD_DIR)/
34     $(CP) $(LOCAL_SRC)/* $(PKG_BUILD_DIR)/
35 endef
36
37 define Build/InstallDev
38     $(INSTALL_DIR) $(STAGING_DIR)
39 endef
40
41 define Build/Compile
42 #$(warning build "get FPKG_CURDIR")
```

2. Compile the application makefile and add QMI library dependencies.

```
owrt_workspace/vendor/fibocom/opensource/sample-qmi/uim/
├── inc
│   └── fibo_test.h
├── Makefile
└── src
    └── uim_qmi_test.c
```

```
#Makefile for OWRT system

INCLUDE_DIR += ./inc -I./api -I$(FPKG_ROOTDIR)/common
CFLAGS += -I$(INCLUDE_DIR) \
        -Wundef \
        -Wstrict-prototypes \
        -Wno-trigraphs \
        -g -O0 \
        -fno-short-enums \
        -fpic

LDFLAGS += $(TARGET_LDFLAGS) -lqmi \
                    -lqmiidl   \
                    -lqmi_cci  \
                    -lqmi_csi  \
                    -lqmi_common_so  \
                    -lqmiservices  \
                    -lqmi_client_qmux  \
                    -lqmi_client_helper \

RM := rm -rf

SRC_DIR := ./src
TARGET_APP := $(FPKG_TARGET)

FPKG_SRCS = $(wildcard $(SRC_DIR)/*.c)
FPKG_OBJS := $(notdir $(FPKG_SRCS:%.c=%.o))

$(TARGET_APP): $(FPKG_OBJS)
    $(CC) $(FPKG_OBJS) $(CFLAGS) $(LDFLAGS) -o $@
    mkdir -p bin
    mv $(TARGET_APP) bin/
```

3 Introduction to Commonly Used QMI

3.1 Introduction to Key APIs

Following are commonly used interfaces for clients, declared in qmi_client.h.

3.1.1 qmi_client_ind_cb()

The QCCI infrastructure calls this function when an indication is received. This callback function is registered at initialization, and ind_buf is the encoded QMI message that must be decoded by the client using the qmi_client_message_decode function.

```
typedef void (*qmi_client_ind_cb)
(
 qmi_client_type                user_handle,
 unsigned int                   msg_id,
 void                           *ind_buf,
 unsigned int                   ind_buf_len,
 void                           *ind_cb_data
);
```

in	user_handle	Handle used by the infrastructure to identify different clients
in	msg_id	Message ID
in	ind_buf	Pointer to original/undecoded instructions
in	ind_buf_len	Length of indication data
in	ind_cb_data	User data

3.1.2 qmi_client_init_instance ()

Initializes a connection to a service with a specific instance ID.

```
qmi_client_error_type qmi_client_init_instance
(
 qmi_idl_service_object_type service_obj,
 qmi_service_instance        instance_id,
 qmi_client_ind_cb           ind_cb,
 void                        *ind_cb_data,
```

```
 qmi_client_os_params        *os_params,
 uint32_t                    timeout,
 qmi_client_type             *user_handle
);
```

in	service_obj	Service object
in	instance_id	Instance ID of the service_info entry
in	ind_cb	Indication callback function
in	ind_cb_data	User data of indication callback
in	os_params	Operating system-specific parameters. Can be a pointer to an event object or a signal mask and task control block (TCB)
in	timeout	Timeout (Unit: ms)
out	user_handle	Handle used by the infrastructure to identify different clients

3.1.3 qmi_client_send_msg_sync()

Sends synchronous QMI service message. It provides encoding/decoding functions, and the user obtains the decoded data in the provided response structure.

```
extern qmi_client_error_type
qmi_client_send_msg_sync
(
 qmi_client_type    user_handle,
 unsigned int       msg_id,
 void               *req_c_struct,
 unsigned int       req_c_struct_len,
 void               *resp_c_struct,
 unsigned int       resp_c_struct_len,
 unsigned int       timeout_msecs
);
```

in	user_handle	Handle used by the infrastructure to identify different clients
in	msg_id	Message ID
in	req_c_struct	Pointer to request
in	req_c_struct_len	Length of request
in	resp_c_struct	Pointer to response storage location

in	resp_c_struct_len	Length of response buffer
in	timeout_msecs	Timeout in ms 0 = No timeout

3.1.4 qmi_client_release()

Releases the connection to the service.

```
extern qmi_client_error_type
qmi_client_release
(
 qmi_client_type     user_handle
);
```

in	user_handle	The handle used by the infrastructure to identify clients.

3.2 Introduction to Common Server Components

3.2.1 Network Access Service (NAS)

QMI_NAS provides commands related to network access for applications running on the host PC:

- Signal strength
- Network registration and attachment
- Service system
- Network scanning
- Home, preferred, and forbidden networks

Header file: network_access_service_v01.h

Get the service object function macro: nas_get_service_object_v01( )

QMI message command prefix: QMI_NAS_

3.2.2 Wireless Messaging Service (WMS)

QMI_WMS provides commands related to wireless message for applications running on the host PC, including:

- Sending original data
- Writing, reading, and deleting data in device memory
- Modifying labels
- Setting and reading routes
- Reading and setting the SMS center (SMSC) address

Header file: wireless_messaging_service_v01.h

Get the service object function macro: wms_get_service_object_v01()

QMI message command prefix: QMI_WMS_

3.2.3 Device Management Service (DMS)

QMI_DMS provides commands related to device management for applications running on bound devices, such as terminal equipment (TE):

- Device identification (manufacturer, model, firmware version, phone number, serial number)
- Device functions (data service type, SIM, data rate)
- Device power status (battery power, power supply)

Header file: device_management_service_v01.h

Get the service object function macro: dms_get_service_object_v01( )

QMI message command prefix: QMI_DMS_

3.2.4 Wireless Data Service (WDS)

The QMI_WDS provides a set of commands for interfacing with the wireless mobile station, providing IP connectivity and related value-added services. QMI_WDS provides the following applications running on the host PC with commands related to IP data services on the wireless radio network:

- Data call setup and release
- Network registration and attachment
- Packet transmission statistics
- Data carrying ratio
- Data session profile management

Header file: wireless_data_service_v01.h

Get the service object function macro: wds_get_service_object_v01( )

QMI message command prefix: QMI_WDS_

3.2.5 Voice Service (VOICE)

QMI_VOICE provides commands related to voice service for applications running on the host PC:

- Call initiation
- Call termination
- Incoming call response
- Flashing
- Dual Tone Multi-Frequency (DTMF)
- Supplementary services

Header file: voice_service_v02.h

Get the service object function macro: voice_get_service_object_v02( )

QMI message command prefix: QMI_NAS_

3.2.6 User Identity Module (UIM)

The UIM module is used to access the cards available on the device. The implementation supports SIM and USIM cards for GSM/WCDMA/LTE/NR devices, as well as UIM and CSIM cards for CDMA devices. The QMI_UIM module supports accessing multiple functions of the card.

- Accessing files on the card
  - Reading transparency and record
  - Writing transparency and record
  - Getting the file attribute
- Password operations
  - Enabling and disabling
  - Validation
  - Unlocking
  - Changing
- Other tasks
  - Supporting file refresh operation
  - Supporting personalization
  - Starting/closing the card
  - Verification
  - Selecting configuration applications from the card
  - Getting the modem configuration
  - Sending the original APDU to the card
  - SIM access profile

Header file: user_identity_module_v01.h

Get the service object function macro: uim_get_service_object_v01( )

QMI message command prefix: QMI_UIM_

3.2.7 IP Multimedia Subsystem Settings (IMSS)

The QMI_IMSS service provides setup services for its control points. These services include interfaces for reading configuration parameters and writing configuration parameters. This service can be extended to support additional configuration parameters in the future.

The control point can also register an indication to be sent from the service when any configuration parameter on the modem changes.

User-level applications use QMI_IMSS to access this feature on MSM devices.

Header file: ip_multimedia_subsystem_settings_v01.h

Get the service object function macro: imss_get_service_object_v01( )

QMI message command prefix: QMI_IMS_SETTINGS_

4 Example

Following is an example of using the QMI-related interfaces.

For example, the QMI NAS service provides network-related interfaces. Taking obtaining signal strength as an example, you can call the QMI_NAS_GET_SIG_INFO interface. The sample code is as follows.

```
static qmi_client_type fibo_qmi_nas_handle = NULL;
int32_t network_client_init(void)
{
      qmi_client_error_type qmi_error;
    qmi_idl_service_object_type nas_qmi_idl_service_object;
qmi_cci_os_signal_type qmi_os_param;
nas_qmi_idl_service_object = nas_get_service_object_v01();
if (nas_qmi_idl_service_object == NULL)
{
            return -1;
}
qmi_error = qmi_client_init_instance(nas_qmi_idl_service_object,
                                        QMI_CLIENT_INSTANCE_ANY,
                                        NULL,
                                        NULL,
                                        &qmi_os_param,
                                        3000,
                                        &fibo_qmi_nas_handle);
    if(qmi_error != QMI_NO_ERR)
    {
        FFW_LOG_INFO("Failed to init Network error code: %d", qmi_error);
        if(fibo_qmi_nas_handle != NULL)
        {
            qmi_client_release(fibo_qmi_nas_handle); //Release the handle in case of
  a failure.
            fibo_qmi_nas_handle = NULL;
                    return -1;
        }
    }
return 0;
}
```

```
//
int32_t ffw_pal_radio_get_signal()
{
nas_get_sig_info_req_msg_v01 nas_network_get_sig_info_req_msg;
nas_get_sig_info_resp_msg_v01 nas_network_get_sig_info_resp_msg;
qmi_client_error_type  qmi_err1 = QMI_NO_ERR;

if(fibo_qmi_nas_handle == NULL)
{
fibo_network_client_init();
}
qmi_err1 = qmi_client_send_msg_sync(fibo_qmi_nas_handle,
QMI_NAS_GET_SIG_INFO_REQ_MSG_V01,
                                    (void *)&nas_network_get_sig_info_req_msg,
sizeof(nas_get_sig_info_req_msg_v01),
                                    (void *)&nas_network_get_sig_info_resp_msg,
sizeof(nas_get_sig_info_resp_msg_v01),
                                    FIBO_QMI_NAS_TIMEOUT_VAL);
if (qmi_err1 ==QMI_NO_ERR)
{
if(nas_network_get_sig_info_resp_msg.resp.result == QMI_RESULT_SUCCESS_V01)
{
      FFW_LOG_INFO("success");
  }
}
else
  {
       FFW_LOG_INFO("check respone faild");
         return -1;
  }
    return 0;
}
```

Where ffw_pal_radio_get_signal() is an entry function to obtain signal strength, the request nas_get_sig_info_req_msg_v01 and response nas_get_sig_info_resp_msg_v01 are defined in network_access_service_v01.h. fibo_qmi_nas_handle is used to identify the handles of different clients. If there is no conflict, it is recommended that one application registers one handle for each QMI service.

network_client_init calls qmi_client_init_instance to initialize the connection. After the initialization is successful, use qmi_client_send_msg_sync to send a request to the client. Then, you can obtain the signal strength from rsp in the response.

The software version carries a more complete example **fibo_open_sdk\owrt_workspace\vendor\fibocom\opensource\sample-qmi**.

It takes SIM card status acquisition as an example to detail functions such as initializing client, status reporting, and status query.

5 Other Considerations

- It is recommended to start one QMI service for one application and create a client at startup to avoid frequent resource release.
- Do not call QMI within QMI active reporting activities, or perform other blocking operations such as loop or sleep.