EC2x&EG9x&EG25-G Series
QuecOpen Log System Application Note

LTE Standard Module Series

Rev. EC2x&EG9x&EG25-G_Series_QuecOpen_Log_System_Application_Note_V1.0

Date: 2020-07-07

Status: Released

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai, 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

Revision History

Version	Date	Author	Description
1.0	2020-07-07	Tyler KUANG/ Tinker SUN	Initial

Contents

Table Index

1 Introduction

Quectel LTE Standard EC2x&EG9x&EG25-G series modules support QuecOpen® solution. This document mainly introduces the use, viewing and configuration of log system as well as related application exception record, and matter needing attention of these modules.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x series	EC25 series
	EC21 series
	EC20 R2.1
EG9x series	EG95 series
	EG91 series
EG25-G	EG25-G

2 Log System

2.1. Overview

Android Logcat solution is adopted in Quectel EC2x&EG9x&EG25-G series QuecOpen® solution. Logcat contains 4 ring buffers that can be read and written (MAIN, RADIO, EVENTS and SYSTEM), and 6 log levels (VERBOSE, DEBUG, INFO, WARN, ERROR and FATAL). The 4 ring buffers are used as below:

MAIN	Used by customers' application.
RADIO	Deprecated.
EVENTS	Used to record system events.
SYSTEM	Used to record system key service program log.

Logcat commands can be used to view log and to filter log based on tag and log levels. For more details, please refer to https://developer.android.com/studio/command-line/logcat.

2.2. Use Log

Both liblog system encapsulated by Android and qlsyslog system encapsulated by QuecOpen® solution can be adopted for log system. This chapter introduces how to use qlsyslog system.

Head file: *qlsyslog/ql_sys_log.h*
Library: *libql_sys_log.so*

The definition of log interface is located in head file *qlsyslog/ql_sys_log.h*. Please refer to the following sample code when calling log interface, and add *-lql_sys_log* to link options in *Makefile*.

```
#include"qlsyslog/ql_sys_log.h"

#define LOG_TAG "fortest"

int main(int argc, char *argv)
{
    QLOGV(LOG_TAG, "I am QL_SYS_LOG_VERBOSE");
    QLOGD(LOG_TAG, "I am QL_SYS_LOG_DEBUG");
```

```
    QLOGI(LOG_TAG, "I am QL_SYS_LOG_INFO");
    QLOGW(LOG_TAG, "I am QL_SYS_LOG_WARN");
    QLOGE(LOG_TAG, "I am QL_SYS_LOG_ERROR");
    QLOGF(LOG_TAG, "I am QL_SYS_LOG_FATAL");

    return 0;
}
```

2.3. View Log

View log with Logcat tool. For more details about how to use the tool, please refer to https://developer.android.com/studio/command-line/logcat.

This chapter takes the sample code in ***Chapter 2.2*** as an example to briefly introduce 3 ways of viewing and filtering logs with Logcat tool.

1. Filter logs by tag.

```
/data # logcat -s fortest
--------- beginning of system
--------- beginning of main
07-17 12:24:11.472  2039  2039 V fortest : I am QL_SYS_LOG_VERBOSE
07-17 12:24:11.472  2039  2039 D fortest : I am QL_SYS_LOG_DEBUG
07-17 12:24:11.472  2039  2039 I fortest : I am QL_SYS_LOG_INFO
07-17 12:24:11.472  2039  2039 W fortest : I am QL_SYS_LOG_WARN
07-17 12:24:11.472  2039  2039 E fortest : I am QL_SYS_LOG_ERROR
07-17 12:24:11.472  2039  2039 F fortest : I am QL_SYS_LOG_FATAL
```

2. Filter logs by log level.

```
/data # logcat -s fortest:w
--------- beginning of system
--------- beginning of main
07-17 12:24:11.472  2039  2039 W fortest : I am QL_SYS_LOG_WARN
07-17 12:24:11.472  2039  2039 E fortest : I am QL_SYS_LOG_ERROR
07-17 12:24:11.472  2039  2039 F fortest : I am QL_SYS_LOG_FATAL
```

3. Filter logs by buffer.

```
/data # logcat -b main
07-17 12:24:11.472  2039  2039 V fortest : I am QL_SYS_LOG_VERBOSE
07-17 12:24:11.472  2039  2039 D fortest : I am QL_SYS_LOG_DEBUG
07-17 12:24:11.472  2039  2039 I fortest : I am QL_SYS_LOG_INFO
07-17 12:24:11.472  2039  2039 W fortest : I am QL_SYS_LOG_WARN
07-17 12:24:11.472  2039  2039 E fortest : I am QL_SYS_LOG_ERROR
07-17 12:24:11.472  2039  2039 F fortest : I am QL_SYS_LOG_FATAL
```

2.4. Configure Log

The qlsyslog system will filter kernel log and application log based on filtering policy and store the logs to the position specified by log_file. The configuration file is located in */data/qllog.json* and in JSON format. The key configuration items are listed below:

Table 2: Log System Configuration Items

Items	Option	Description
log_file	Must	Log files storage position. During the operation of the system, the log files may be frequently written. Therefore, the log cannot be saved in a critical system partition.
rotate_file_size	Must	Size limitation of a single log file. Unit: KB.
rotate_file_count	Must	The maximum number of log files.
log_format	Optional	Output format of log files. Options: default and csv.
kernel_priority	Optional	Kernel log level. Namely, printk level. Options: m (emergency messages), a (alert messages), c (critical conditions), e (error conditions), w (warnings), n (notice), i (informational messages), d (debugging messages) and * (all levels).
buffer_list.{i}.name	Must	Buffer name. Options: main, system and events.
buffer_list.{i}. filter_list.{i}.tag	Optional	The tags of logs that need to be saved. All tags will be used by default if this item is not specified.
buffer_list.{i}.filter_list. {i}.priority	Optional	The levels of logs that need to be saved. Options: v (VERBOSE), d (DEBUG), i (INFO), w (WARN), e (ERROR), f (FATAL) and * (all levels).

3 Application Exception Record

3.1. Monitor Application Exception

Application can record exception through monitoring signal, and the sample code is as follows. In the exception handling program, *ql_sys_log_signal(qlsyslog/ql_sys_log.h)* will record as more exception information of the application as possible.

```
#include <signal.h>
#include <stdlib.h>
#include"qlsyslog/ql_sys_log.h"

#define LOG_TAG "fortest"

static void handle_signal (int sig_num, siginfo_t *info, void *ptr)
{
    ql_sys_log_signal(QL_SYS_LOG_ID_MAIN, QL_SYS_LOG_FATAL, LOG_TAG, sig_num, info, ptr);
    exit(-1);
}

int main(int argc, char *argv)
{
    struct sigaction sa = {0};
    sa.sa_sigaction = handle_signal;
    sa.sa_flags = SA_SIGINFO;
    sigaction (SIGTERM, &sa, NULL);
    sigaction (SIGSEGV, &sa, NULL);
    sigaction (SIGABRT, &sa, NULL);
    sigaction (SIGINT, &sa, NULL);
    sigaction (SIGBUS, &sa, NULL);

    QLOGI(LOG_TAG, "bootup");

    /* other code */

    return 0;
}
```

3.2. Obtain Abnormal Call Stack

When an exception occurs in an application, the exception code can be located through the detailed application call stack. *ql_sys_log_signal* can trace back the function call stack, but it also has certain limitations.

- **Method**

Delete the optimization options -O1, -O2, -O3 and -fomit-frame-pointer in application compilation options, and add the compilation option -fasynchronous-unwind-tables –rdynamic.

- **Limitation**

Some libraries had added optimization options when compiling. If an exception occurs in library functions, the detailed call stack might not be traceable.

4 Matter Needing Attention

During the operation of the system, a massive amount of logs may be generated and the log files may be frequently written. Storing log files in the flash will shorten the flash service life (there is a limit to the maximum writing times of flash technical parameters).

In the research & development and testing stage, the log file can be saved in flash to facilitate debugging. Whereas in the subsequent stage, it is recommended to save the log files to the temporary file system as follows:

```
~ # cat /data/qllog.json
{
        /* log output format : log type, time, pid, tid, tag, msg */
        /* log type : k->kernel, e->events, s->system, m->main */
        /* (mandatory) log file path */
        "log_file":"/tmp/qllog/ql_log.txt",

        /* (optional) rotate log every kbytes, default is unlimit */
        "rotate_file_size":  1024,
        /* (optional) max number of rotated logs, default is 4 */
        "rotate_file_count": 2,

        /* (optional) log format, default or csv */
        "log_format":"default",
```

Figure 1: Save Log Files to Temporary File System

5 Appendix A References

Table 3: Terms and Abbreviations

Abbreviation	Description
KB	Kilobyte
LTE	Long Term Evolution