EC2X&AG35-QuecOpen

SGMII API MANUAL

LTE Module Series

Rev. EC2X&AG35-QuecOpen_ SGMII API MANUAL _V1.0

Date: 2018-04-07

Status: temporary

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

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:

http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:

http://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

History

Revision	Date	Author	Description
1.0	2018-04-07	Mike ZHOU	Initial

Contents

1 Introduction

This document introduces SGMII functions and API.

2 SGMII Introductions

SGMII is the interface between PHY and MAC which similar with GMII and RGMII, yet GMII and RGMII are parallel, and need follow-up clock, PCB layout is relatively troublesome, and not suitable for backplane applications. The SGMII is serial, no need to provide additional clock, MAC and PHY need CDR to recover the clock. In addition SGMII is with 8B / 10b encoded, the rate is 1.25G.

Currently, the EC20 and AG35 only support the AR8033 PHY chip, 10BASE-Te / 100BASE-Te / 1000BASE-Te IEEE802.3 compliant.

3 API Interface Introduction

Linux does not start GSMII function by default, interfaces provided are as followings.

(1) int ql_sgmii_enable(void)

Enable SGMII function, calling the function to load the SGMII driver. After successful, eth0 network port can be seen under the console, as shown in following picture.

```
~ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:80:48:BA:D1:30
          inet addr:169.254.4.1  Bcast:169.254.4.255  Mask:255.255.255.0
          inet6 addr: fe80::280:48ff:feba:d130/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:398 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40956 (39.9 KiB)  TX bytes:1405 (1.3 KiB)
          Interrupt:48

~ #
```

After the network port start successfully, qti program will capture the startup event and notify the QCMAP_ConnectionManager process to load the eth0 device onto bridge0, as shown in following picture.

```
~ # brctl show
bridge name     bridge id               STP enabled     interfaces
bridge0         8000.fa85eccde650       no              eth0
~ #
```

NOTE

The default MAC address of the eth0 interface is 00: 80: 48: BA: D1: 30. If the client need to change this MAC address, can through the CLI (for example: ifconfig eth0 hw ether 00: 80: 48: BA: d1: 30), also can modify the address settings in the SGMII startup script (/etc/init.d/start_emac_le).

(2) int ql_sgmii_disable(void);

Disable SGMII function, calling this function, eth0 network port will be removed from bridge0 and SGMII driver will be uninstalled.

(3) int ql_sgmii_speed_set(ql_sgmii_speed_e speed);

Set the network port rate, support 10MHZ / 100MHZ / 1000MHZ, the default is adaptive, the macro definition is as followings.

```
typedef enum {
```

```
        QL_SGMII_SPEED_AUTO = 0,     //Adaptive
        QL_SGMII_SPEED_10MHZ,        //10MHZ
        QL_SGMII_SPEED_100MHZ,       //100MHZ
        QL_SGMII_SPEED_1000MHZ     // 1000MHZ
} ql_sgmii_speed_e;
```

If the setting is QL_SGMII_SPEED_AUTO, the two network interfaces will negotiate the rate through adaptation.

(4) int ql_sgmii_speed_get(ql_sgmii_speed_e *speed);

Get the current network port rate.

(5) int ql_sgmii_duplex_set(ql_sgmii_duplex_e duplex)

Set the network port duplex mode, support half-duplex and full-duplex, the macro definition is as following.

```
typedef enum {
        QL_SGMII_DUPLEX_FULL = 0,        //half-duplex
        QL_SGMII_DUPLEX_HALF              // full-duplex
} ql_sgmii_duplex_e;
```

NOTE

By calling this function, network port rate cannot be set as QL_SGMII_SPEED_AUTO. For the gigabit rate AR8033 chip, it only supports full duplex.

(6) int ql_sgmii_duplex_get(ql_sgmii_duplex_e *duplex);

Get the duplex mode set by network port.

(7) int ql_sgmii_speed_duplex_set(ql_sgmii_speed_e speed, ql_sgmii_duplex_e duplex);

Get the rate and duplex mode of network port.

(8) int ql_smgii_info_get(struct ql_sgmii_info *info);

Get the current status of the network interface, including the number of data packets sent and received, the data size, the running rate and running duplex mode.

NOTE

Before call interface (3)-(8), please call ql_sgmii_enable() at first.

4 Example

Please refer to **example/sgmii/example_sgmii.c**.

```
int main(int argc, char **argv)
{
        ql_sgmii_enable();
        ql_sgmii_speed_duplex_set(QL_SGMII_SPEED_100MHZ, QL_SGMII_DUPLEX_FULL);
        return 0;
}
```

NOTE

Configuration saving is not provided at present, will provided it later.