Build a Smarter World

MDM9x07&MDM9628 SGMII API MANUAL

www.quectel.com

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

This document applies to MDM9628 and MDM9X07 platforms.

History

Revision	Date	Author	Description
1.0	2017-11-01	Mike	Initial

Contents

1. Introduction

SGMII is a interface between PHY and MAC which similar to GMII and RGMII, yet GMII and RGMII are parallel, and need to follow the clock, PCB layout is relatively troublesome, and not suitable for backplane applications. The SGMII is serial, no need to provide an additional clock, MAC and PHY need CDR to recover the clock. In addition SGMII is 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

2. Specification

SGMII will not enabled by default, the interface is as follows.

1. **int ql_sgmii_enable(void);**

   Start SGMII function, call the function will load SGMII driver; After the driver loading successfully, you can see the eth0 network port in the console, as shown in Figure 1:

```
~ # 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

~ #
```

Figure 1

After the network port start successfully, the qti program will capture the startup event and notify the QCMAP_ConnectionManager process to load the eth0 device to the bridge0, as shown in Figure 2:

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

Figure 2

Note:

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

2. **int ql_sgmii_disable(void);**
   Close the SGMII function, call this function, eth0 network port will removed from the bridge0, and the driver will be uninstalled.

3. **int ql_sgmii_speed_set(ql_sgmii_speed_e speed);**
   Set the network port speed, support 10MHZ / 100MHZ / 1000MHZ, the default is adaptive, the macro is defined as follows:

```
typedef enum {
    QL_SGMII_SPEED_AUTO = 0,    /* automatic adapt */
    QL_SGMII_SPEED_10MHZ,        /* 10MHZ */
    QL_SGMII_SPEED_100MHZ,     /* 100MHZ */
    QL_SGMII_SPEED_1000MHZ     /* 1000MHZ */
} ql_sgmii_speed_e;
```

   If 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 speed of the network port.

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 is defined as follows:

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

   Note: Network port rate cannot be set as QL_SGMII_SPEED_AUTO by calling this function. For the gigabit rate AR8033 chip, it only supports full duplex.

6. **int ql_sgmii_duplex_get(ql_sgmii_duplex_e *duplex);**
   Obtain the duplex mode by the network interface

7. **int ql_sgmii_speed_duplex_set(ql_sgmii_speed_e speed, ql_sgmii_duplex_e duplex);**
   Set the network port speed and duplex mode.

8. **int ql_smgii_info_get(struct ql_sgmii_info *info);**
   Obtain the current status of the network interface, including the number of data packets sent and received, the data size, the current running rate, and the current running duplex mode.

Note: Before calling interface 3 ~ 8, you need to call ql_sgmii_enable ();

3. Example

You can refer to example: 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: Currently Configuration save function is not supported, will support it in the future.