RTL9047AA, RTL9044AA

RTL9044AB, RTL9043AA

REALTEK INTERNAL USE ONLY

# SINGLE-CHIP AUTOMOTIVE ETHERNET SWITCH CONTROLLER WITH 100BASE-T1 TRANSCEIVER

## SDK API PROGRAMMING GUIDE

**(CONFIDENTIAL: Development Partners Only)**

**Rev. 0.79**
**5 April 2017**
**Track ID: JATR-8275-15**

**Realtek Semiconductor Corp.**
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047
www.realtek.com

Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

## COPYRIGHT

©2017 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

## DISCLAIMER

Realtek provides this document 'as is', without warranty of any kind. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

## TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

## LICENSE

This product is covered by one or more of the following patents: US5,307,459, US5,434,872, US5,732,094, US6,570,884, US6,115,776, and US6,327,625.

## USING THIS DOCUMENT

This document is intended for the software engineer's reference and provides detailed programming information.

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

## REVISION HISTORY

| Revision | Release Date | Summary |
|---|---|---|
| 0.72 | 2015/11/13 | Preliminary release. |
| 0.73 | 2015/12/10 | Add Section 4.17, page 55 |
| 0.74 | 2016/03/23 | Revise section 3.5, page 6 |
| 0.75 | 2016/03/25 | Add section 4.17 for RTCT |
| 0.76 | 2016/06/13 | Revised Section 4.1, page 7 |
| 0.77 | 2016/07/26 | Revised Section 1, page 1 |
| 0.78 | 2016/09/09 | Add section 4.20 for ACL |
| 0.79 | 2017/01/26 | Add section 4.22, Page 60<br>Add section 4.23, Page 61 |

Realtek confidential files
The document is authorized to RICH_POWER_PHY_TWN
2018-09-25 14:07:49

Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

# Table of Contents



Realtek Confidential files
For document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

Realtek confidential files
The document authorized to
RICH_POWER_PIN_TWN
2018-09-25 14:07:49

# List of Tables



Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

# List of Figures



Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

# 1. General Description

The RTL9047AA / RTL9044AA / RTL9044AB / RTL9043AA SDK contains general APIs that users can use to perform configuration.

The RTL9047AA / RTL9044AA / RTL9044AB / RTL9043AA SDKs are fully interchangeable. Through the rest of this document we use 'RTL904Xxx' to refer to the four products

Section 2, page 3 describes the organization of the RTL904Xxx SDK

Section 3, page 4 is a porting guide to porting SDK on a new platform

Section 4, page 7 is an introduction to function modules in the SDK

**Table 1. Product List**

| Product Name | Package | Description |
|---|---|---|
| RTL9047AA | BGA, 217 ball count | Full interface with the same package size. For systems that require one GMII, or two SGMII ports.<br>Contains:<br>4*100BASE-T1 PHY: Port index 0~3<br>1*C0: 100BASE-T1 and SGMII<br>1*E0: FE PHY, SGMII, and RGMII/MII/RMII<br>1*E1: GMII/RGMII/MII/RMII |
| RTL9044AA | BGA, 217 ball count | Contains:<br>1*100BASE-T1 PHY: Port index 0<br>1*C0: 100BASE-T1 and SGMII<br>1*E0: FE PHY, SGMII, and RGMII/MII/RMII<br>1*E1: GMII/RGMII/MII/RMII |
| RTL9044AB | BGA, 217 ball count | Contains:<br>2*100BASE-T1 PHY: Port index 0~1<br>1*C0: 100BASE-T1 and SGMII<br>1*E0: FE PHY, SGMII, and RGMII/MII/RMII |
| RTL9043AA | BGA, 217 ball count | Contains:<br>2*100BASE-T1 PHY: Port index 0~1<br>1*E0: FE PHY, SGMII and RGMII/MII/RMII |

Realtek confidential files
The document authorized to
RICH_POWER only_IWN
2018-09-05 14:07:49

## 1.1. Port Index Definition

The physical ports of the RTL904Xxx are referred to as Port0 to Port7. The MAC Circuits for each port are referred to as MAC0 to MAC7 respectively. The external interfaces and connection for all the RTL904Xxx ports are listed as follows:

**Table 2. Port Index**

| Port Index | Interface | Note |
|---|---|---|
| 0 | 100BASE-T1 PHY | Short name is P0 |
| 1 | 100BASE-T1 PHY | Short name is P1 |
| 2 | 100BASE-T1 PHY | Short name is P2 |
| 3 | 100BASE-T1 PHY | Short name is P3 |
| 4 | 100BASE-T1 PHY and SGMII | Combo port 0. Configurable. 100BASE-T1 PHY and SGMII are individual pins. These two interfaces share the same MAC layer. This port may connect to a CPU or cascade switch.<br>Short name is C0. |
| 5 | FE PHY, SGMII, and RGMII/MII/RMII (share pin) | External port 0. Configurable. FE PHY, SGMII and RGMII/MII/RMII are individual pins. RGMII, RMII, and MII share pins. This port may connect to a CPU or cascade switch.<br>These three interfaces share the same MAC layer.<br>Short name is E0. |
| 6 | GMII/RGMII/MII/RMII (share pin) | External port 1. Configurable. GMII/RGMII/MII/RMII are shared pins. This port may connect to a CPU or cascade switch.<br>Short name is E1. |
| 7 | Internal 8051 | This port connects to the internal CPU(8051). Short name is P7. |

Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

# 2. SDK Structure

Realtek provides an RTK layer to abstract the functions between different chips. The main header files should be included in your program. They are:

| | |
|---|---|
| rtk_api_ext.h: | RTK Function Prototype |
| rtk_error.h: | RTK error list |
| rtk_api.h: | RTK Data structure |

The other files are:

**SDK/src/asicdrv**

This is a low-level asic driver, which should not be used directly. These functions will be called by RTK APIs that are implemented in rtk_api.c, to provide an easier, more uniform way of use.

**SDK/src/asicdrv/basic**

This is a low-level asic driver to access RTL904Xxx registers.

**SDK/src/rtk_i2c.c**

This is an I2C sample code used where some functions are customer-implemented when porting to a new hardware platform.

**rtk_mdc.c**

This is an MDC/MDIO sample code used where some functions are customer-implemented when porting to a new hardware platform.

**rtk_spi.c**

This is an SPI sample code used where some functions are customer-implemented when porting to a new hardware platform.

Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

# 3. Porting Issues

## 3.1. Environment Setting

In order to compile the SDK successfully, put the following directories into the development path:

'include', 'include/asicdrv' and 'include/asicdrv/basic'.

## 3.2. Register Access Interface Porting

An external CPU can access registers of the RTL9040 via I2C, MDC/MDI, and SPI interfaces, which is decided by strapping setting. In order to port to new platforms successfully, the low-level GPIO signal function should be rewritten. Users should follow the steps in the following sections to provide correct GPIO control function.

## 3.3. I2C Interface Porting

The I2C interface includes SDA/SCL. This is a serial management interface; SCK transmits the clock signal and SDA transmits the data signal. Refer to the RTL904Xxx_Series_DataSheet for more information about protocol and timing characteristics.

The sample code is at src/rtk_i2c.c. The sample code uses two GPIO pins to emulate SDA/SCL signals. RTL9040_I2C_READ & RTL9040_I2C_WRITE is used to read/write the registers. In code below we use the RTL9040 as our sample model.

First, choose two GPIO pins. Specify one as SCL, the other as SDA;

The code body embraced by the macro SAMPLE_CODE should be rewritten when porting to a new platform.

The APIs needing rewritten are:

```
RTL9040_I2C_init();
_gpio_sda_mode_set();
_SCL_SDA_set();
_SCL_set();
_SDA_get();
_SCL_wait_half_CLK();
```

Function RTL9040_I2C_init() is used to initialize the GPIOs that are used as a I2C interface. Usually this function only configures GPIOs in the master IC to output mode. _gpio_sda_mode_set() is used to configure GPIO SDA to input or output mode.

Functions _SCL_SDA_set(), _SCL_set() and _SDA_get() are used to get/set data on SDA/SCL GPIO lines. _SCL_SDA_set() is used to output SCL and SDA signals together, and is used when writing addresses or data to the RTL9040. _SCL_set() is used to output SCL signal alone, this API will be used to output SCL signal when read data from RTL9040. _SDA_get() is used to read the data at SDA.

Realtek confidential files
the document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49

Function _SCL_wait_half_CLK(), a very common function, is used to add some hold time to keep the SCL & SDA signals stable. Note that different interfaces require different delay cycles (see the RTL904Xxx_Series_DataSheet for more information).

## 3.4. MDC/MDIO Interface Porting

The MDC/MDIO(SMI) interface includes MDC/MDIO. It is a serial management interface. MDC transmits the clock signal and MDIO transmits the data signal (same as I2C). Refer to the RTL904Xxx_Series_DataSheet for more information about protocol and timing characteristics.

The sample code is at src/rtk_mdc.c. The sample code uses two GPIO pins to emulate MDC/MDIO signals. RTL9040_SMI_READ & RTL9040_SMI_WRITE is used to read/write the registers of the RTL9040.

First, choose two GPIO pins and specify one as MDC, the other as MDIO;

Second, the code body embraced by macro SAMPLE_CODE should be rewritten when porting on new platform.

The APIs needing rewritten are:

RTL9040_MDCMDIO_init();

_gpio_mdio_mode_set();

_MDC_MDIO_set();

_MDC_set();

_MDIO_get();

_MDC_wait_half_CLK();

Function RTL9040_MDCMDIO_init() is used to initialize the GPIOs that are used as the MDC/MDIO(SMI) interface. Normally this function only configures GPIOs in the master IC to output mode. _gpio_mdio_mode_set() is used to configure GPIO MDIO to input or output mode.

Functions _MDC_MDIO_set(), _MDC_set() and _MDIO_get() are used to get/set data on MDC/MDIO GPIO lines. _MDC_MDIO_set() is used to output MDC and MDIO signals together, and is used when writing address or data to the RTL9040. _MDC_set() is used to output only MDC signals. This API is used to output MDC signals when reading data from the RTL9040. _MDIO_get() is used to read the data at MDIO.

The common function _MDC_wait_half_CLK() is used to add hold time to keep MDC and MDIO signals stable. Note that different interfaces require different delay cycles (see the RTL9047AA_RTL9047AB_RTL9044AA_RTL9043AA_DataSheet for more information).

Realtek confidential files
This document authorized to RTCT_POWER_PHY_TWN
2018-09-25 14:07:49

## 3.5. SPI Interface Porting

The SPI interface includes CS, CLK, SI, and SO. It is a serial management interface. CS is used to indicate this SPI interface is in use, and CLK transmits clock signals to slave devices. SI and SO are the data signals. SI is used to get input data while SO is the output data signal. Refer to the RTL904Xxx_Series_DataSheet for more information about protocol and timing characteristics.

The sample code is at src/rtk_spi.c. The sample code uses four GPIO pins to emulate SPI signals. RTL9040_SPI_READ & RTL9040_SPI_WRITE is used to read/write the registers of the RTL9040.

First, choose four GPIO pins and specify as CS, CLK, SI, and SO;

Second, the code body embraced by macro SAMPLE_CODE should be rewritten when porting on new platform.

The APIs needing rewritten are:

RTL9040_SPI_init();

REGIFSPI_CS_H()

REGIFSPI_CS_L()

REGIFSPI_CLK_H()

REGIFSPI_CLK_L()}

REGIFSPI_SI_H()

REGIFSPI_SI_L()

REGIFSPI_SO_STS()

spi_waveform_wait();

Function RTL9040_SPI_init() is used to initialize the GPIOs that are used as CS, CLK, SI, and SO interfaces. Normally this function only configures GPIOs in the master IC to output mode, except for SI which is configured to input mode.

Functions REGIFSPI_CS_H(), REGIFSPI_CS_L(), REGIFSPI_CLK_H(), REGIFSPI_CLK_L(), REGIFSPI_SI_H(), REGIFSPI_SI_L(),and REGIFSPI_SO_STS() are used to get/set data on GPIO lines. REGIFSPI_CLK_H() and REGIFSPI_CLK_L() are used to output a clock signal, and REGIFSPI_CS_H() and REGIFSPI_CS_L() are used to indicate that this SPI interface is in use when writing address or data to the RTL9040. REGIFSPI_SI_H() and REGIFSPI_SI_L() are used to output only data signals. This API is used to output data signals when writing data to the RTL9040. REGIFSPI_SO_STS() is used to read output data signals from RTL9040.

The common function spi_waveform_wait () is used to add hold time to keep the 4 signals of SPI stable. Note that different interfaces require different delay cycles (see the RTL9047AA_RTL9047AB_ RTL9044AA_RTL9043AA_DataSheet for more information).

Realtek confidential files
RTL_POWER_PHY_TWN
2018-03-25 14:07:49

# 4. Function Overview

## 4.1. VLAN

The RTL904Xxx support a 1024-entry VLAN table. Users should use API rtk_vlan_init to initialize the VLAN before using it. The initialization will enable VLAN functions and set up a default VLAN with VID 1 that contains all ports. Each port's PVID will also be set to the default VLAN.

After initialization, users can add and set VLAN 0~4095 through API rtk_vlan_set, and configure its member set and untag set. In order to get the member set information and untag a set on a specified VLAN, rtk_vlan_get can be used. For Port-Based VLAN there are two APIs provided (rtk_vlan_portPvid_set and rtk_vlan_portPvid_get) to set and get the PVID of the ports.

Users can turn on the ingress filter function with rtk_vlan_portIgrFilterEnable_set. The accepted frame types for ports can be set through rtk_vlan_portAcceptFrameType_set.

The following are explanations and sample codes for VLAN APIs.

### 4.1.1. Initialization

**int32 rtk_vlan_init(void)**

This API should be called before using VLAN. It enables VLAN functions and sets up a default VLAN with VID 1 that contains all ports. It also sets each port's PVID to the default VLAN. After VLAN initialization, the switch uses 4K VLAN mapping for tagged frames while using Port-Based VLAN mapping for untagged frames. After calling rtk_vlan_init, user can change the default VLAN arrangement through the following introduced API rtk_vlan_set and rtk_vlan_portPvid_set.

Example:

```
/* initialize VLAN */
rtk_vlan_init ();
/* all the ports are in the default VLAN 1 after VLAN is initialized. Modify it as
follows
VLAN1 member: port0, port1, port2;
VLAN2 member: port3, port4, port5 */
rtk_portmask_t mbrmsk, untagmsk;
rtk_vlan_t VLAN1, VLAN2;
VLAN1 = 100;
VLAN2 = 200;
mbrmsk.bits[0]=0x07;      /* port 0~ port 2*/
untagmsk.bits[0]=0x3F;
rtk_vlan_set(VLAN1, mbrmsk, untagmsk, 0);
mbrmsk.bits[0]=0x38;      /* port 3~ port 5*/
untagmsk.bits[0]=0x3F;
rtk_vlan_set(VLAN2, , mbrmsk, untagmsk, 0);
```

Realtek Confidential Files
The document is authorized to
RICH_POWER_PINTWN
2018-09-25 14:07:49

```
/* set PVID for each port */
rtk_vlan_portPvid_set(0, VLAN1, 0);
rtk_vlan_portPvid_set(1, VLAN1, 0);
rtk_vlan_portPvid_set(2, VLAN1, 0);
rtk_vlan_portPvid_set(3, VLAN2, 0);
rtk_vlan_portPvid_set(4, VLAN2, 0);
rtk_vlan_portPvid_set(5, VLAN2, 0);
```

### 4.1.2. Add/Modify/Destroy a VLAN

**int32 rtk_vlan_set(rtk_vlan_t vid, rtk_portmask_t mbrmsk, rtk_portmask_t untagmsk, rtk_fid_t fid)**

typedef uint32 rtk_vlan_t;

typedef struct rtk_portmask_s

```
{
  uint32  bits[RTK_TOTAL_NUM_OF_WORD_FOR_1BIT_PORT_LIST];
}  rtk_portmask_t;
```

typedef uint32  rtk_fid_t;

This API can set the member set, untagged set, and filtering database to the specified VLAN. The mbrmask's bit N means port N. For example, mbrmask.bits[0] = 23=0x17=0b010111 means port 0, 1, 2, 4 in the member set. A VLAN entry can be cleared by specifying its member set, untagging, and setting to zero. For this chip, FID range is 0~14, and is used as a spanning tree instance ID.

If a VLAN is set to SVL (Shared VLAN Learning) mode, FID is also used as a filtering database ID. However, if a VLAN is set to IVL (Independent VLAN Learning) mode, FID is only used as a spanning tree instance ID.
After VLAN initialization, VLAN is configured to IVL mode. API rtk_vlan_ivlsvlMode_set can be used to change the mode.

```
Example:
/* set port 0,1,2,3 as member set and port 2,3 as untag set for VLAN 1000 with fid 0 */
rtk_vlan_t vid;
rtk_portmask_t mbrmsk, untagmsk;
rtk_fid_t fid;

vid = 1000;
mbrmsk.bits[0]=0x0F;
untagmsk.bits[0]=0x0C;
fid = 0;
rtk_vlan_set(vid, mbrmsk, untagmsk, fid);
```

Realtek confidential files
The document authorized to
RICH_POWER_PTY_TWN
2018-09-25 14:07:19

```
/* clear member set and untag set for VLAN 1000 */
vid = 1000;
mbrmsk.bits[0]=0;
untagmsk.bits[0]=0;
fid = 0;

rtk_vlan_set(vid, mbrmsk, untagmsk, fid);
```

**int32 rtk_vlan_get(rtk_vlan_t vid, rtk_portmask_t *pMbrmsk, rtk_portmask_t *pUntagmsk, rtk_fid_t *pFid)**

```
typedef uint32  rtk_vlan_t;
typedef struct rtk_portmask_s
{
  uint32 bits[RTK_TOTAL_NUM_OF_WORD_FOR_1BIT_PORT_LIST];
} rtk_portmask_t;
Typedef  uint32  rtk_fid_t;
```

The API can get the information of member set, untagged set, and filtering database on the specified VLAN. The information is retrieved in ***mbrmask***. The ***mbrmask***'s bit N means port N. For example, ***mbrmask.bits[0]*** = 23 = 0x17 = 0b010111 means port 0,1,2,4 in the member set.

Example:

```
/* get the member set and untagged set on VLAN 1000 */
rtk_vlan_t vid;
rtk_portmask_t mbrmsk, untagmsk;
rtk_fid_t fid;

vid = 1000;
rtk_vlan_get(vid, & mbrmsk, & untagmsk, &fid);
```

**int32 rtk_vlan_destroy(rtk_vlan_t vid)**

```
typedef uint32  rtk_vlan_t;
```

This API can delete an existing VLAN with the specified VID from the VLAN table.

Example:

```
/* delete the VLAN with VID 100 */
rtk_vlan_destroy (100);
```

### 4.1.3. Get VLANs by Port

The way to find out the VLANs which a port belongs to is to scan the whole VLAN table, and check the member port of each VLAN.

Realtek confidential files
The document authorized to
RICH_POWER_PHY_TWN
2018-09-25 14:07:49