NOT FOR PUBLIC RELEASE

RTL8103E-GR
RTL8103EL-GR

INTEGRATED FAST ETHERNET CONTROLLER FOR PCI EXPRESS APPLICATIONS

eFUSE APPLICATION NOTES
(CONFIDENTIAL: Development Partners Only)

Rev. 1.0
18 December 2008
Track ID: CONF-1076-21

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

COPYRIGHT

DISCLAIMER

Realtek provides this document "as is", without warranty of any kind, neither expressed nor implied, including, but not limited to, the particular purpose. 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.

USING THIS DOCUMENT

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. In that event, please contact your Realtek representative for additional information that may help in the development process.

REVISION HISTORY

Revision	Release Date	Summary
1.0	2008/12/18	First release.

Table of Contents

List of Tables

List of Figures

1. General Description

The RTL8103E(L) features embedded eFUSE One-Time-Programmable (OTP) memory. eFUSE is a technology that allows for dynamic real-time reprogramming of computer chips. Via an eFUSE, Realtek allows for the circuits on our IC to change while it is in operation.

The RTL8103E(L) eFUSE contains state-of-the-art memory space-saving and programming technology. The eFUSE memory stores configuration values, such as MAC ID, SSID, SVID, and other settings according to customer's requirements, and eliminates the need for an external EEPROM. The primary application of this technology is to provide in-chip performance tuning. If certain sub-systems fail, or are taking too long to respond, or are consuming too much power, the chip can instantly change its behavior by 'blowing' an eFUSE. Total number of programming times depends on user usage and command type.

The eFUSE interface permits the RTL8103E(L) to read from, and write data to, eFUSE memory. As a resistor in the circuit, the eFUSE uses a current to burn the resistor out, increasing resistance and allowing an internal comparator to identify whether an eFUSE cell is burnt-out or not through a reference eFUSE cell.

A complete eFUSE auto load consists of GPHY Access, EPHY Access, EMAC Configuration, GMAC IDR Access, GMAC Configuration Access, GMAC IO Access, and End of Auto Load commands. Each command, except End of Auto Load, is composed of Command, Address/Byte, Count/Byte, Enable/Config Code, and Data Byte.

Each byte is written individually via the eFUSE interface into the internal eFUSE register of the GMAC I/O register. The eFUSE interface then writes the value in the data field to the eFUSE Register bits[17:8] in the eFUSE memory. These steps are repeated until the auto load information has been written in the correct place with the correct value. An eFUSE read or write is indicated in the flag field of the eFUSE Register bit[31]. The output value of a programmed cell is '0', and a non-programmed cell is '1'. The controller identifies the latest MAC ID and ignores previously written MAC IDs. E.g., the first MAC ID is programmed as 00-11-22-33-44-55, and there is another MAC ID programmed later as 55-44-33-22-11-00; the RTL8103E(L) will take the later one as the final MAC ID.

Values in eFUSE memory can override fields in PCI configuration space and I/O space following a power-on or software auto-load command. If both an external EEPROM and eFUSE memory are present, the RTL8103E(L) initialization uses default values for the appropriate Configuration and Operational Registers. Software can read and write to the eFUSE via the eFUSE Access Register in I/O space. The interface consists of TEST2, GPI, and GPO for the RTL8103E-GR and EECS, EESK, and EEDI for the RTL8103EL-GR.

2. Design Considerations

Aspects to be considered when programming for eFUSE are listed below and detailed in the following sections.

- eFUSE Interface
- Command Description
  - GPHY Access (If necessary)
  - EPHY Access (If necessary)
  - EMAC Configuration Access (If necessary)
  - GMAC IDR Access
  - GMAC Configuration Access (If necessary)
  - GMAC IO Access (If necessary)
  - End of OTP Auto Load
- eFUSE Interface Timing
- eFUSE Command Format
- Recovery Flow

3. eFUSE Interface

3.1. eFUSE Interface for RTL8103E-GR

Table 1. eFUSE Interface for RTL8103E-GR

Pin Name	Description
TEST2	eFUSE Chip Select
GPI	eFUSE Serial Data Clock
GPO	eFUSE Serial DataIn and DataOut

3.2. eFUSE Interface for RTL8103EL-GR

Table 2. eFUSE Interface for RTL8103EL-GR

Pin Name	Description
EECS	eFUSE Chip Select
LED1/EESK	eFUSE Serial Data Clock
LED2/EEDI/AUX	eFUSE Serial DataIn and DataOut

3.3. Extra Pin Condition for eFUSE Programming

Table 3. Extra Pin Condition for eFUSE Programming

Pin Name	Condition
PERSTB	De-Asserted
REFCLK_P	100MHz Clock
REFCLK_M	100MHz Clock
CKTAL1	25MHz Clock
CKTAL2	25MHz Clock

4. Command Description

There are six commands available for use, however, we recommend you do not use GPHY, EPHY, EMAC, EMAC configuration[1], and GMAC I/O[2] commands to change fields values unless you obtain approval from Realtek, or for recovery purposes due to an incorrect value being programmed, or the manufacturing process causing an error leading to an unsuccessful program.

To recover from an unsuccessful programming attempt, the program builder must read the programmed value for comparison immediately after the program procedure for each byte. The precise recovery method is discussed in section 7 Recovery Flow, page 12.

Note1: This restriction does not include the registers VID, SID, SVID, SSID, and SERIAL NUMBER register.

Note2: The restriction does not include the IDR register.

4.1. GPHY Access (If Necessary)

The GPHY Access command is composed of three Command bits (000b), five GPHY address bits, and two GPHY data bytes, for a total length of three bytes (see Figure 1).

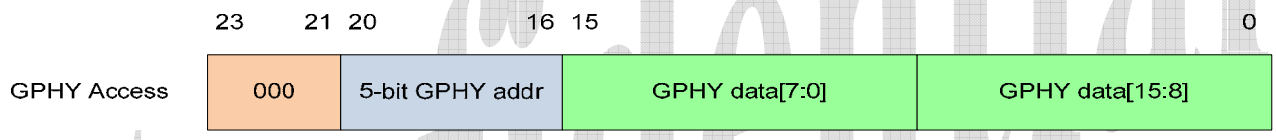

Figure 1. GPHY Command

Example: To write GPHY address 12h with data 3456Fh to eFUSE offset 78h.

Step 1. Write 80007812h to I/O register offset DCh (eFUSE Register) via the eFUSE Interface (Cmd=000b, GPHY addr=10010b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh via the eFUSE Interface.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007956h to I/O register offset DCh via the eFUSE Interface (GPHY data[7:0]=56h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007A34h to I/O register offset DCh via the eFUSE Interface (GPHY data[15:8]=34h).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

4.2. EPHY Access (If Necessary)

The EPHY programming command is composed of three Command bits (001b), five EPHY address bits, and two EPHY data bytes, for a total length of three bytes (see Figure 2).

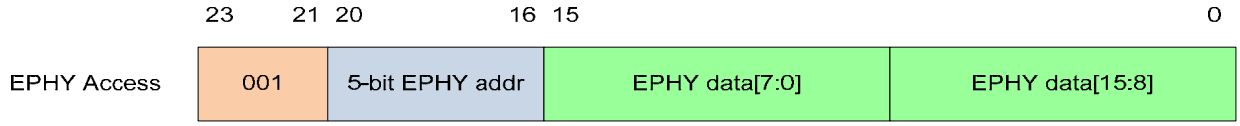

Figure 2. EPHY Command

Example: To write EPHY address 12h with data 3456h to eFUSE offset 78h.

Step 1. Write 80007832h to I/O register offset DCh (The Cmd=001b, 5-bit EPHY addr=10010b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007956h to I/O register offset DCh (EPHY data[7:0]=56h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007A34h to I/O register offset DCh (EPHY data[15:8]=34h).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

4.3. EMAC Configuration Access (If Necessary)

The EMAC Configuration programming command is composed of two Command bits, two Byte Count bits, twelve EMAC Configuration address bits, and one to four bytes of EMAC Configuration data, for a total length of three to six bytes (see Figure 3).

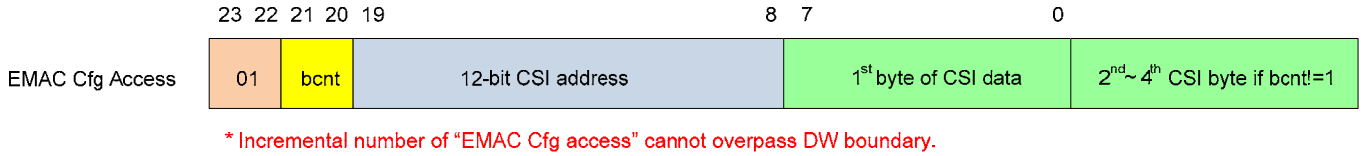

Figure 3. EMAC Configuration Command

Table 4. Byte Count

bcnt	total_data_bcnt
0	4
1	1
2	2
3	3

Example: To write EMAC Configuration address 12h with data 3456789Ah to eFUSE offset 78h.

Step 1. Write 80007840h to I/O register offset DCh (Cmd=01b, bcnt=00b and highest nibble of CSI address[11:8]=0000b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007912h to I/O register offset DCh (the low byte of CSI address[7:0]=12h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007A9Ah to I/O register offset DCh (the 1st byte of CSI data=9Ah).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

Step10. Write 80007B78h to I/O register offset DCh (the 2nd byte of CSI data=78h).

Step11. Repeat Steps 2~5, except the address writes for the read operation changes to 00007B00h.

Step12. Write 80007C56h to I/O register offset DCh (the 3rd byte of CSI data=56h).

Step13. Repeat Steps 2~5, except the address writes for the read operation changes to 00007C00h.

Step14. Write 80007D34h to I/O register offset DCh (the 4th byte of CSI data=34h).

Step15. Repeat Steps 2~5, except the address writes for the read operation changes to 00007D00h.

4.4. GMAC IDR Access

The GMAC IDR programming command is composed of two Command bits, two Byte Count bits, twelve GMAC IDR address bits, and one to four bytes of GMAC IDR Configuration data, for a total length of three to six bytes (see Figure 4).

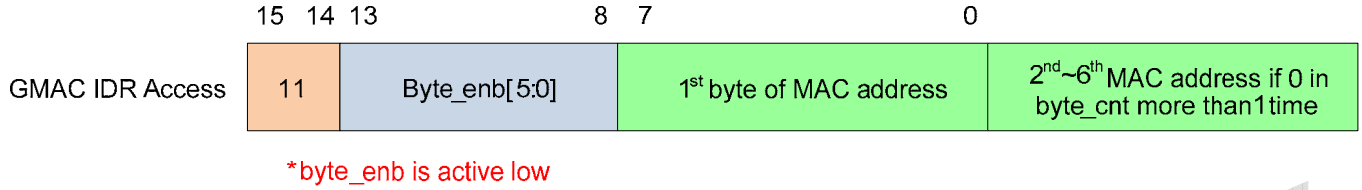

Figure 4. GMAC IDR Command

Example: To write GMAC IDR with data 0000364CE000h to eFUSE offset 78h.

Step 1. Write 800078C0h to I/O register offset DCh (Cmd=11b, Byte_enb[5:0]=000000b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007900h to I/O register offset DCh (the 1st byte of MAC address=00h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007AE0h to I/O register offset DCh (the 2nd byte of MAC address=E0h).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

Step10. Write 80007B4Ch to I/O register offset DCh (the 3rd byte of MAC address=4Ch).

Step11. Repeat Steps 2~5, except the address writes for the read operation changes to 00007B00h.

Step12. Write 80007C36h to I/O register offset DCh (the 4th byte of MAC address=36h).

Step13. Repeat Steps 2~5, except the address writes for the read operation changes to 00007C00h.

Step14. Write 80007D00h to I/O register offset DCh (the 5th byte of MAC address=00h).

Step15. Repeat Steps 2~5, except the address writes for the read operation changes to 00007D00h.

Step16. Write 80007E00h to I/O register offset DCh (the 6th byte of MAC address=00h).

Step17. Repeat Steps 2~5, except the address writes for the read operation changes to 00007E00h.

4.5. GMAC Configuration Access (If Necessary)

The GMAC Configuration programming command is composed of two Command bits, five mac_cfg_code bits, and one to six bytes of GMAC Configuration data, for a total length of three to six bytes (see Figure 5).

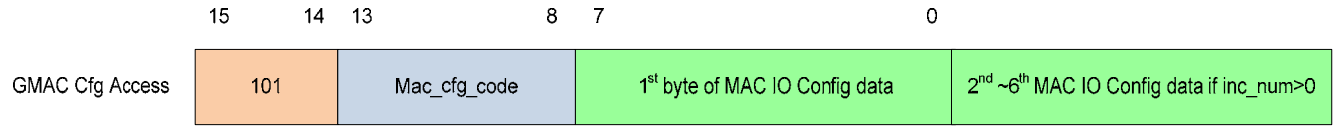

Figure 5. GMAC Configuration Command

Example: To write GMAC Configuration offset 51~53h with data 2C4F00h to eFUSE offset 78h.

Step 1. Write 800078AAh to I/O register offset DCh (Cmd=101b, Mac_cfg_code=01010b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007900h to I/O register offset DCh (the 1st byte of MAC IO Config=00h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007A4Fh to I/O register offset DCh (the 2nd byte of MAC IO Config=4Fh).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

Step10. Write 80007B2Ch to I/O register offset DCh (the 3rd byte of MAC IO Config=2Ch).

Step11. Repeat Steps 2~5, except the address writes for the read operation changes to 00007B00h.

4.6. GMAC IO Access (If Necessary)

The GMAC IO programming command is composed of six Command bits, two Byte Count bits, eight GMAC IO address bits, and one to four bytes of GMAC IO data, for a total length of three to six bytes (see Figure 6).

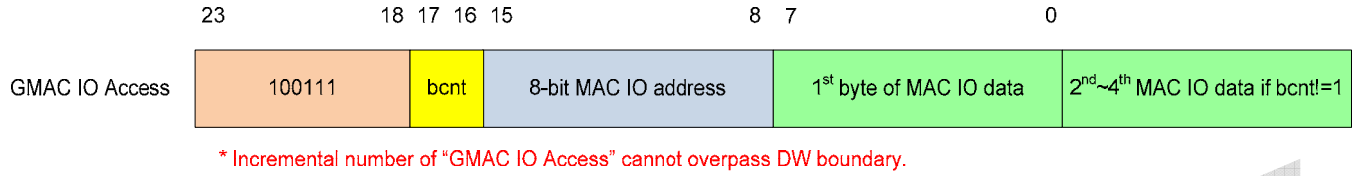

Figure 6. GMAC IO Command

Example: To write GMAC IO offset 38h with data 0Ch to eFUSE offset 78h.

Step 1. Write 8000789Dh to I/O register offset DCh (Cmd=100111b, bcnt=01b).

Step 2. Wait 1ms.

Step 3. Write 00007800h to I/O register offset DCh.

Step 4. Wait 1ms.

Step 5. Read the last byte written to I/O register offset DCh to confirm it is the same as that programmed to be written.

Step 6. Write 80007938h to I/O register offset DCh (the MAC IO address=38h).

Step 7. Repeat Steps 2~5, except the address writes for the read operation changes to 00007900h.

Step 8. Write 80007A0Ch to I/O register offset DCh (the 1st byte of MAC IO data=0Ch).

Step 9. Repeat Steps 2~5, except the address writes for the read operation changes to 00007A00h.

4.7. End of eFUSE Auto Load

The End of Auto Load command value 'FFh' indicates to the RTL8103E(L) that the auto load procedure has completed.

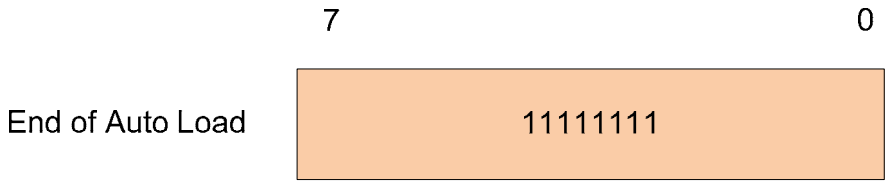

Figure 7. End of Auto Load Command

5. eFUSE Interface Timing

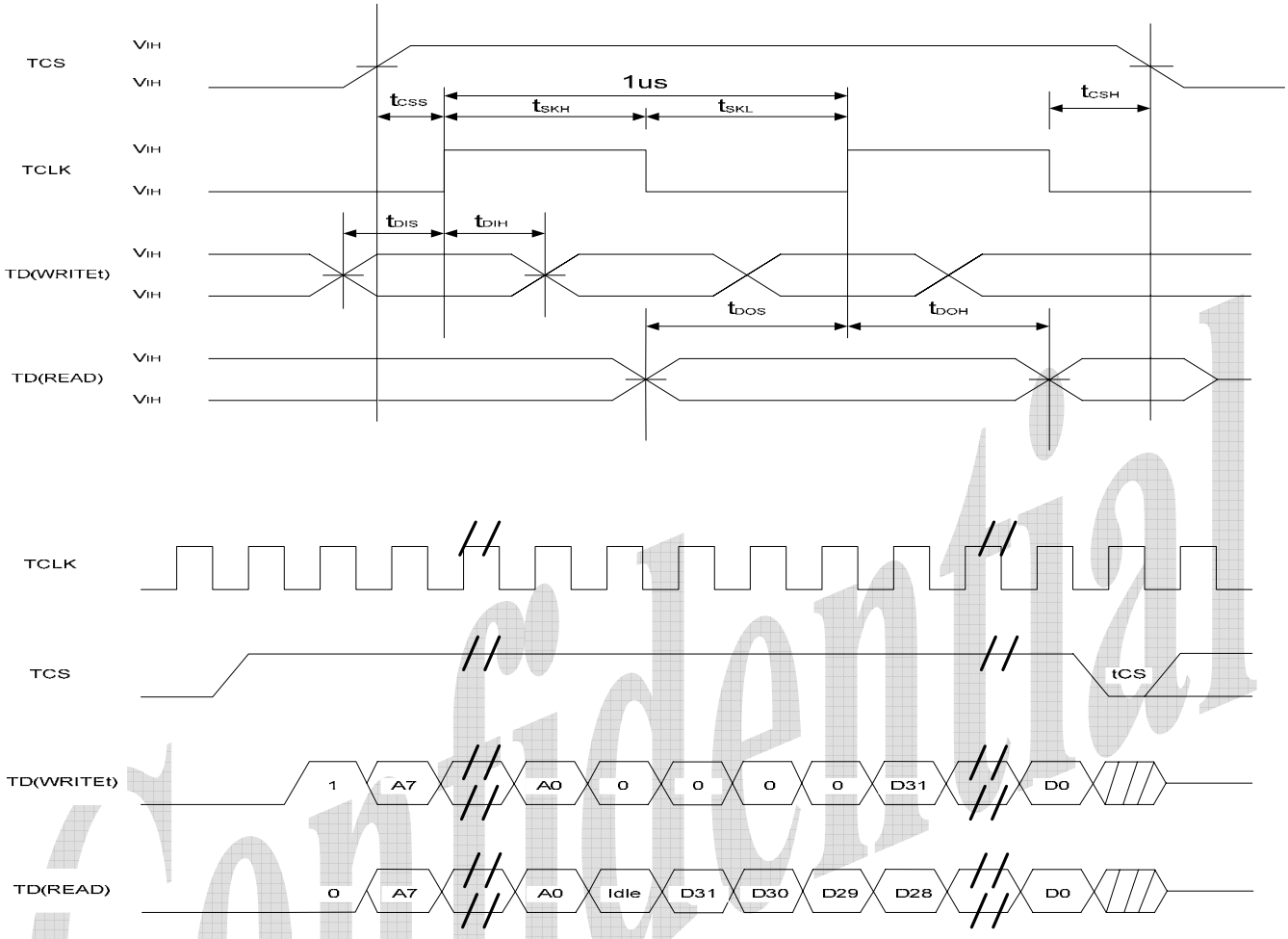

Figure 8. eFUSE Interface Timing

Table 5. eFUSE Interface DC/AC Characteristics

Symbol	Parameter	Condition	Min.	Max.	Unit
tCS	Minimum CS Low Time	-	1000	-	ns
tSKH	SK High Time	-	1000	-	ns
tSKL	SK Low Time	-	1000	-	ns
tCSS	CS Setup Time	-	200	-	ns
tCSH	CS Hold Time	-	0	-	ns
tDIS	DI Setup Time	-	400	-	ns
tDIH	DI Hold Time	-	400	-	ns
tDOS	DO Setup Time	-	2000	-	ns
tDOH	DO Hold Time	-	-	2000	ns
VIH	Voltage Input High	-	2.4	-	V
VIL	Voltage Input Low	-	-	0.8	V
VOH	Voltage Output High	Ioh=-4mA	2.6	-	V
VOL	Voltage Output Low	Iol=4mA	-	0.4	V

6. eFUSE Command Format

The eFUSE Interface protocol utilizes RTL8103E(L) registers to allow reads/writes to the GPHY Access register, EPHY Access register, EMAC Configuration Access register, GMAC Configuration register, GMAC IO register, and eFUSE Access register.

The registers are located in the GMAC I/O Register:

- Offset 60h~63h for GPHY Access
- Offset 64h~67h for EMAC Configuration Data Register
- Offset 68h~6Bh for EMAC Configuration Access
- Offset 80h~83h for EPHY Access
- Offset DCh~DFh for eFUSE Access.

Note: The parameter tables are provided in section 8, page 14.

Example: To write GPHY offset 00h to eFUSE offset 03h via the eFUSE Interface.

'Rvd' indicates Reserved Field.

'Cmd' indicates the Command for the eFUSE Interface; 1 for write and 0 for read.

Table 6. eFUSE Interface Write Format

eFUSE Interface Write Format																							
TCS	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
CLK	Rvd	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…	…	43	44	45	Rvd	
TD	Rvd	Wr	A7	A6	A5	A4	A3	A2	A1	A0	B3	B2	B1	B0	D31	D30	…	…	D2	D1	D0	Rvd	
Data	-	1	DCh (Addr of eFUSE Access)								0h				80000300h (DataField)							Rvd	

Example: To read eFUSE offset 04h via the eFUSE Interface.

Table 7. eFUSE Interface Read Format

eFUSE Interface Read Format																					
TCS	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
CLK	Rvd	1	2	3	4	5	6	7	8	9	10	11	12	13	…	…	40	41	42	Rvd	
TD	Rvd	Rd	A7	A6	A5	A4	A3	A2	A1	A0	Idle	D31	D30	D29	…	…	D2	D1	D0	Rvd	
Data	-	0	DCh								-	820004XXh (DataField)									Rvd

Note: To read from GPHY, EPHY, and EMAC Configuration, write the read command and the address that you want to retrieve from eFUSE memory, and then read the data from the eFUSEACC register.

Note: The EMAC Configuration Register is separated into DataReg and AccessReg, so software must write the data into DataReg before writing:

- The command
- Byte enable
- The related address into AccessReg

7. Recovery Flow

To recover from an unsuccessful programming attempt, the program builder should read the programmed value for each byte for comparison purposes immediately after the program procedure. If a read back value does not match the value intended to be written, then the program builder should determine which field the wrong value belongs to, e.g., the Command Field; Address Field, Byte Count Field, Byte Enable Field, Configuration Code Field.

The basic recovery principle is to write the command field to an invalid command (write the command field to a value not belonging to any command described in the previous section).

To save memory space, another method that can be used is to program the field to the error value returned in the un-wanted Command/Address/ByteCnt/ByteEnb/ConfigCode field.

If the error cannot be recovered by the above methods:

- Re-write the returned error value mapped by the wrong Command/Address/ByteCnt/ByteEnb field
- Write the value that was supposed to be written to the next address in eFUSE memory

The more comprehensive a recovery procedure the program builder writes, the more memory space will be saved. Figure 9, Figure 10, and Figure 11 illustrate the recovery flow.

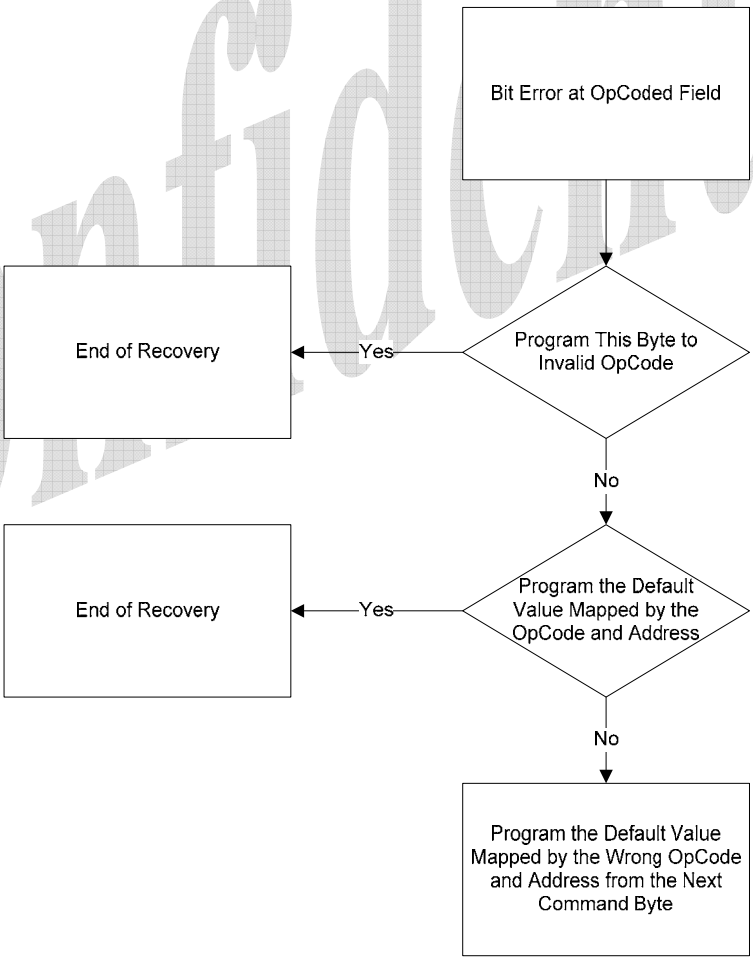

Figure 9. Error Handling of Command Field

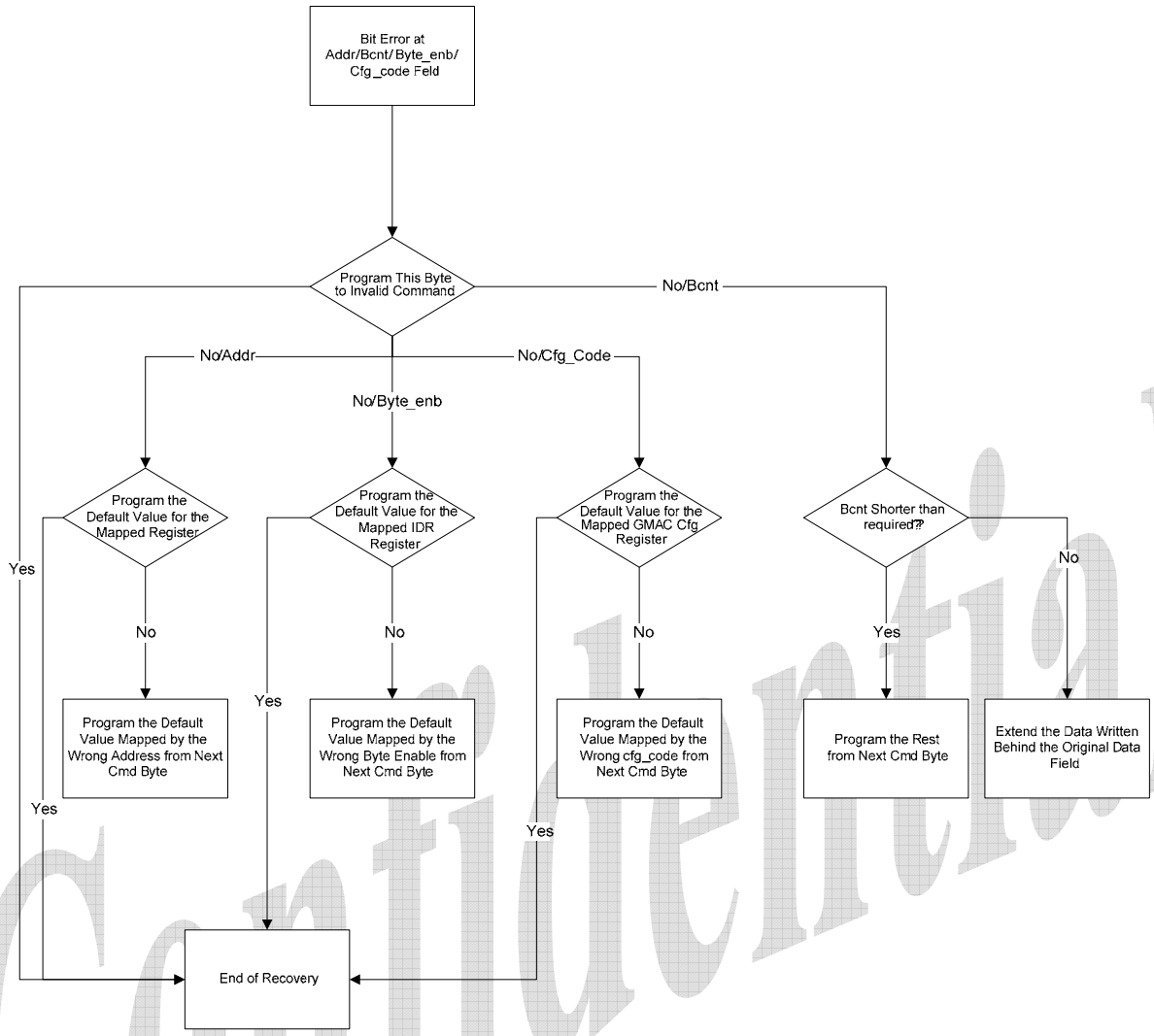

Figure 10. Error Handling of Addr/Bcnt/Byte_Enb/Cfg_Code Field

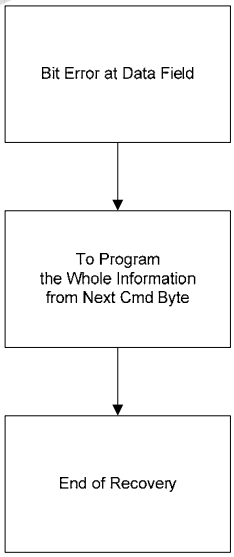

Figure 11. Error Handling of Data Field

8. Appendix A - Register Table

8.1. eFUSEAR: eFUSE Access (Offset 00DCh-00DFh, RW)

Table 8. eFUSEAR: eFUSE Access (Offset 00DCh-00DFh, RW)

Bit	Symbol	RW	Description
31	Flag	RW	Flag bit, used as PCI VPD access method. 1: Write data to the eFUSE register, and turn to 0 automatically whenever the RTL8103E(L) has completed writing to the specified eFUSE register. 0: Read data from the eFUSE register, and indicate 1 automatically whenever the RTL8103E(L) has completed retrieving data from the specified eFUSE register.
30-29	PG_time	RW	eFUSE Program Time. 00: 500µs 01: 510µs 10: 520µs 11: 590µs
28-27	Vpp_vsel	RW	eFUSE VPP Voltage Select: 00: VPP=1.8V 01: VPP=1.9V 01: VPP=2.0V 11: VPP=2.1V
26	OP0	RW	eFUSE OP0 bit. Selects the eFUSE sense amplifier threshold. Default is 0.
25	autoload_n	R	1: Indicates auto-load is done
24-18	-	-	Reserved
17-8	RegAddr9-0	RW	10-bit eFUSE Register Address.
7-0	Data7-0	RW	8-bit eFUSE Register Data.

8.2. PHYAR: PHY Access (Offset 0060h-0063h, RW)

Table 9. PHYAR: PHY Access (Offset 0060h-0063h, RW)

Bit	Symbol	RW	Description
31	Flag	RW	Flag bit, used as PCI VPD access method. 1: Write data to the MII register, and turn to 0 automatically whenever the RTL8103E(L) has completed writing to the specified MII register. 0: Read data from the MII register, and indicate 1 automatically whenever the RTL8103E(L) has completed retrieving data from the specified MII register.
30-21	-	-	Reserved
20-16	RegAddr4-0	RW	5-bit GMII/MII Register Address.
15-0	Data15-0	RW	16-bit GMII/MII Register Data.

8.3. CSIDR: CSI Data (Offset 0064h-0067h, RW)

Table 10. CSIDR: CSI Data (Offset 0064h-0067h, RW)

Bit	Symbol	RW	Description
31-0	DATA 31-0	RW	CSI Data, used as PCI VPD access method. The driver can use this method to access (R/W) all the PCI configuration registers of the RTL8103E(L). CSI is the internal Digital interface between the Gigabit MAC and PCIE MAC. This method provides another method to access the PCI configuration register. *Note: Some registers, such as Vendor ID, denoted as Read-Only by PCI Express configuration Read Request TLP, is R/W able via this method.*

8.4. CSIAR: CSI Access (Offset 0068h-006Bh, RW)

Table 11. CSIAR: CSI Access (Offset 0068h-006Bh, RW)

Bit	Symbol	RW	Description
31	Flag	RW	Flag bit, used as PCI VPD access method. 1: Write data to the PCI configuration register, and turn to 0 automatically whenever the RTL8103E(L) has completed writing to the specified PCI configuration register. 0: Read data from the PCI configuration register, and indicate 1 automatically whenever the RTL8103E(L) has completed retrieving data from the specified PCI configuration register.
30-16	-	-	Reserved
15-12	ByteEn	RW	4-bit CSI Access Register Byte Enable.
11-0	Addr11-0	RW	12-bit CSI Access Register Address (Double-Alignment). The lower 2 bits are always ignored.

8.5. EPHYAR: EPHY Access (Offset 0080h-0083h, RW)

Table 12. EPHYAR: EPHY Access (Offset 0080h-0083h, RW)

Bit	Symbol	RW	Description
31	Flag	RW	Flag bit, used as PCI VPD access method. 1: Write data to the EPHY register, and turn to 0 automatically whenever the RTL8103E(L) has completed writing to the specified EPHY register. 0: Read data from the EPHY register, and indicate 1 automatically whenever the RTL8103E(L) has completed retrieving data from the specified EPHY register.
30-21	-	-	Reserved
20-16	RegAddr4-0	RW	5-bit EPHY Register Address.
15-0	Data15-0	RW	16-bit EPHY Register Data.

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