Micrel KSZ8841M-16 Step-by-Step Programmer's Guide

Version 1.1

09/06/2007

Revision History

Revision	Date	Summary of Changes
1.1	09/06/2007	First release.

Table of Contents

1 Overview

This document provides step-by-step procedure as to which registers and values need to be initialized, how to transmit data to the device, and how to receive data from the device for the KSZ8841M-16 generic bus interface.

Please refer to KSZ8841MQL-DS datasheet for detail register information.

In order to set a bit in a register, such as step 12 in Initialization, read the register first and modify the target bit only and write it back.

2 KSZ8841M Generic Bus Interface Device Initialization Steps

Steps Sequence	Read\write	Register Name[bit]	Value	Description
0		BAR [15-0] Bank 0, 0ffset 0x00	0x0300	Just to make sure that your host bus controller has assigned the low 16bit base address to the KSZ8841M is same as the device default base address 0x0300.
1	Read	SIDER [15-4] Bank 32, 0ffset 0x00	0x880	Read the device chip ID, make sure it is correct ID (0x8810 for KSZ8841M), otherwise there are some errors on the host bus interface.
2	Write	MARL[15-0] Bank 2, 0ffset 0x00	0x89AB	Write QMU MAC address (low). MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
3	Write	MARM[15-0] Bank 2, 0ffset 0x02	0x4567	Write QMU MAC address (Medium). MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
4	Write	MARH[15-0] Bank 2, 0ffset 0x04	0x0123	Write QMU MAC address (High). MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
5	Write	MACAR1[15-0] Bank 39, 0ffset 0x00	0x0123	Write Switch MAC address 1. MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
6	Write	MACAR2[15-0] Bank 39, 0ffset 0x02	0x4567	Write Switch MAC address 2. MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
7	Write	MACAR3[15-0] Bank 39, 0ffset 0x04	0x89AB	Write Switch MAC address 3. MAC address is generally expressed in the form of 01:23:45:67:89:AB. (we use this MAC as an example).
8	Write	TXCR [15-0] Bank 16, 0ffset 0x00	0x000E	Enable QMU Transmit flow control / Transmit padding / Transmit CRC.
9	Write	RXCR [15-0] Bank 16, 0ffset 0x04	0x04E8	Enable QMU Receive flow control / Receive all broadcast frames / Receive all multicast frames / Receive unicast frames / Receive strip the CRC .
10	Write	TXFDPR[15-0] Bank 17, 0ffset 0x04	0x4000	Enable QMU Transmit Frame Data Pointer Auto Increment.
11	Write	RXFDPR[15-0] Bank 17, 0ffset 0x06	0x4000	Enable QMU Receive Frame Data Pointer Auto Increment.

12	Write	QRFCR[12] Bank 0 0ffset 0x04 bit 12	1	[1] Configure QMU Receive High Water Mark to 2KBytes to avoid loss packets (big packet size) under flow control.
13	Write	P1CR4[13] Bank 49, 0ffset 0x02, bit 13	1	Restart Port 1 auto-negotiation.
14	Write	ISR [15-0] Bank 18, 0ffset 0x02,	0xFFFF	Clear the interrupts status.
15	Write	IER [15-0] Bank 18, 0ffset 0x00,	0x2000	Enable Receive Interrupt if your host processor can handle the interrupt, otherwise do not need to do this step.
16	Write	TXCR [0] Bank 16, 0ffset 0x00, bit 0	1	Enable QMU Transmit.
17	Write	RXCR [0] Bank 16, 0ffset 0x04, bit 0	1	Enable QMU Receive.

[1] This feature is not available with KSZ8841M reversion A3 or below.

3 KSZ8841M Generic Bus Interface Transmit Steps

Steps Sequence	Read\write	Register Name[bit]	Value	Description
0	Transmit data frame from the upper layer to KSZ8841M-16 device by a complete packet frame data base. For every complete packet frame data transmit to KSZ8841M-16, process the following the step 1 to 8. There are two variables are needed from the upper layer to transmit a data packet frame. (1). Packet data pointer (**pTxData**). It points to the host CPU system memory space contains the complete Ethernet packet data. (2). Packet length (**txPacketLength**). The Ethernet packet data length not includes CRC.			
1	Read	TXMIR [12-0] Bank 16, 0ffset 0x08	>= (**txPacketLength**+4)	Read value from TXMIR to check if QMU TXQ has enough amount of memory for the Ethernet packet data. Compare the read value with (**txPacketLength**+4), if less than (**txPacketLength**+4), **Exit**.
2	Write	QDRL[15-0] Bank 17, 0ffset 0x08	0x0000	Write 0x0000 to the "control word" of the frame header through a pair of the device registers QDRL.
3	Write	QDRH[15-0] Bank 17, 0ffset 0x0A	(**txPacketLength**)	Write (**txPacketLength**) to the "byte count" of the frame header through a pair of the device registers QDRH.

4	`UINT16 *pTxData;`
	
	if (**txPacketLength** > 0)
	goto Step 5;
	else
	goto Step 8;
				Write 2-byte of Ethernet packet data pointer by **`pTxData`** to the QMU TXQ through a pair of the device registers QDRL, and 2-byte of Ethernet packet data to the QDRH alternately until finished the full packet length (**txPacketLength**).

5	Write	QDRL[15-0] Bank 17, 0ffset 0x08	**`*pTxData++`**	Write 2-byte of Ethernet packet data pointer by **`pTxData`** to the QMU TXUQ through a pair of the device registers QDRL. Increase **`pTxData`** pointer by 2.
6	Write	QDRH[15-0] Bank 17, 0ffset 0x0A	**`*pTxData++`**	Write 2-byte of Ethernet packet data pointer by **pTxData** to the QMU TXQ through a pair of the device registers QDRH. Increase **`pTxData`** `by 2.`
7	**TxPacketLength = txPacketLength – 4;** goto Step 4.			Subtract **TxPacketLength** by 4.
8	Write	TXQCR[15-0] Bank 17, 0ffset 0x00	0x0001	Issue the ENQUEUE transmits command for the device to transmit the Ethernet packet to the Network. **Exit.**

KSZ8841M Generic Bus Interface Receive Steps

Steps Sequence	Read\write	Register Name[bit]	Value	Description
0	There are two methods to receive a complete Ethernet packet from KSZ8841M-16 device to upper layer either as a result of polling or servicing an interrupt. (1). By polling, set a timer routine to periodically execute step 1. (2). By servicing an interrupt, when interrupt occurs, execute step 1. Allocate a system memory space (address by **`pRxData`**) which big enough to hold an Ethernet packet frame.			
1	Read	ISR [13] Bank 18, 0ffset 0x02, bit 13	1	Read value from ISR to check if RXIS Receive Interrupt is set. If not set, **Exit.**
2	Write	ISR [13] Bank 18, 0ffset 0x02, bit 13	1	Acknowledge (clear) RXIS Receive Interrupt bit.
3	Read	RXMIR[12-0] Bank 16, 0ffset 0x0A	> 0	Read value from RXMIR to check if QMU RXQ still has more packet data to be read. If read value <= 0, **Exit**.
4	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**rxStatus**	Read 2-byte of "status word" from QDRL to check if this is a good frame. if **rxStatus**'s bit_15 is 0, goto step **10**; if **rxStatus**'s bit_0, bit_1, bit_2 are 1, go to step **10**
5	Read	QDRH[15-0] Bank 17, 0ffset 0x0A	**rxPacketLength**	Read 2-byte of "byte count" from QDRH to get this received packet Length. Subtract the read value by 4 byte CRC, and store into **rxPacketLength** variable. if **rxPacketLength** <= 0, go to step **10**;

6	`UINT16 *pRxData;`
	
	if (**rxPacketLength** > 0)
	goto Step 7;
	else
	goto Step **10**;
				Read 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through a pair of the device registers QDRL, and 2-byte of Ethernet packet data to the QDRH alternately until finished the full packet length (**rxPacketLength)**.

7	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**`*pRxData ++`**	Read 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the

				QMU RXQ through the device registers QDRL. Increase **pRxData** pointer by 2.
8	Read	QDRH[15-0] Bank 17, 0ffset 0x0A	* **pRxData ++**	Read 2-byte of Ethernet packet to system memory pointer by **pRxData** from the QMU RXQ through the device registers QDRH. Increase **pRxData** pointer by 2.
9	**rxPacketLength = rxPacketLength – 4;** goto Step 6.			Subtract **rxPacketLength** by 4.
10	Write	QRFCR[12] Bank 0 0ffset 0x04 bit 12	0	Reset QMU Receive High Water Mark to 3Kbytes.
11	Write	RXQCR[0] Bank 17, 0ffset 0x02 bit 0	1	Issue the RELEASE frame command for the device to release this frame buffer memory space from QMU RXQ.
12	Write	QRFCR[12] Bank 0 0ffset 0x04 bit 12	1	Set QMU Receive High Water Mark to 2Kbytes.
13	goto step **3** .			Loop again to see if there is more data frame in the QMU RXQ to be read.

4 KSZ8841M Generic Bus Interface Receive Steps for the Turbo Mode [2]

Steps Sequence	Read\write	Register Name[bit]	Value	Description
0	There are two methods to receive a complete Ethernet packet from KSZ8841M-16 device to upper layer either as a result of polling or servicing an interrupt. (1). By polling, set a timer routine to periodically execute step 1. (2). By servicing an interrupt, when interrupt occurs, execute step 1. Allocate a system memory space (address by **`pRxData`**) which big enough to hold an Ethernet packet frame.			
1	Read	ISR [13] Bank 18, 0ffset 0x02, bit 13	1	Read value from ISR to check if RXIS Receive Interrupt is set. If not set, **Exit.**
2	Write	ISR [13] Bank 18, 0ffset 0x02, bit 13	1	Acknowledge (clear) RXIS Receive Interrupt bit.
3	Read	RXMIR[12-0] Bank 16, 0ffset 0x0A	> 0	Read value from RXMIR to check if QMU RXQ still has more packet data to be read. If read value <= 0, **Exit**.
4	Write	RXQCR[1] Bank 17, 0ffset 0x02 bit 1	1	Start the "Turbo mode" [3].
5	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**pDummy**	Dummy read 2-byte from the QMU RXQ through the device registers QDRL.
6	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**rxStatus**	Read 2-byte of "status word" from QDRL to check if this is a good frame. if **rxStatus**'s bit_15 is 0, go to step **16**; if **rxStatus**'s bit_0, bit_1, bit_2 are 1, go to step **16**
7	Read	QDRH[15-0] Bank 17, 0ffset 0x0A	**rxPacketLength**	Read 2-byte of "byte count" from QDRH to get this received packet Length. Subtract the read value by 4 byte CRC, and

[2] It increases KS8841M receiving performance with "**Turbo mode**" operation. The "**Turbo Mode**" only could be applied to the KSZ8841M reversion A6 and it is an optional to the user.
[3] Read other than QMU data registers (QDRL, QDRH) is **NOT** allowed when "**Turbo Mode**" is started.

				store into **rxPacketLength** variable. if **rxPacketLength** <= 0, go to step **16**;
8	**rxPacketLength** --; **rxPacketLength** &= ~0x3;			Leave at most 4 bytes for preparing to stop "Turbo mode" at the end of reading

9	`UINT16 *pRxData;`
	
	if (**rxPacketLength** > 0)
	goto Step 10;
	else
	goto Step **13**;
				Read 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through a pair of the device registers QDRL, and 2-byte of Ethernet packet data to the QDRH alternately until finished the full packet length, but at most 4 byte left **(rxPacketLength)**.

10	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**`*pRxData ++`**	Read 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through the device registers QDRL. Increase **`pRxData`** pointer by 2.
11	Read	QDRH[15-0] Bank 17, 0ffset 0x0A	**`* pRxData ++`**	Read 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through the device registers QDRH. Increase **`pTxData`** pointer by 2.
12	**rxPacketLength = rxPacketLength – 4;** goto Step 9.			Subtract **rxPacketLength** by 4.
13	Read	QDRL[15-0] Bank 17, 0ffset 0x08	**`*pRxData ++`**	Read second last 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through the device registers QDRL. Increase **`pRxData`** pointer by 2.
14	Write	RXQCR[2] Bank 17, 0ffset 0x02 bit 2	1	Let the device knows this is last read during the "Turbo mode".
15	Read	QDRH[15-0] Bank 17, 0ffset 0x0A	**`*pRxData`**	Read last 2-byte of Ethernet packet to system memory pointer by **`pRxData`** from the QMU RXQ through the device registers QDRH.
16	Write	RXQCR[2-1] Bank 17, 0ffset 0x02 bit 2,1	Reset bit 2, bit 1 to '0'.	Stop the "Turbo mode".
17	Write	QRFCR[12]	0	Reset QMU Receive High Water Mark

		Bank 0 0ffset 0x04 bit 12		to 3Kbytes.
18	Write	RXQCR[0] Bank 17, 0ffset 0x02 bit 0	1	Issue the RELEASE frame command for the device to release this frame buffer memory space from QMU RXQ.
19	Write	QRFCR[12] Bank 0 0ffset 0x04 bit 12	1	Set QMU Receive High Water Mark to 2Kbytes.
30	goto step **3** .			Loop again to see if there is more data frame in the QMU RXQ to be read.