AN-140

Programmable Guide and Supporting Software for KSZ88xxM Generic Bus Interface

Introduction

KSZ88xxM software drivers work for KSZ8841M, KSZ8842M, and KSZ8862M devices and are independent from hardware platforms and operating systems. The KSZ88xxM software drivers support the generic bus or non-PCI bus interface.

This application note provides step-by-step procedure as to which registers and values need to be initialized, how to transmit data from host processor to the device, and how to receive data from the device to host processor as well as available Application Programming Interface (API) driver functions for user.

The following topics will be discussed:

- Initialization routine for KSZ88xxM devices
- Driver routine and flowchart for transmit packet from host processor to KSZ88xxM
- Driver routine and flowchart for receive packet from KSZ88xxM to host processor
- Hardware platform supported by KSZ88xxM drivers
- KSZ88xxM driver Application Programming Interface (API) reference

Please refer to individual of KSZ88XXM datasheet for detail register information.

Initialization Routine for KSZ88xxM Devices

The initialization routine is the first routine in the hardware-specific sub-layer that the system calls during initialization. The purpose of this routine is to prepare the KSZ88xxM device and the software driver for immediate use by the system.

Initialization Setting for Switch and PHY Function Block Registers

This section describes the typical register settings for the switch function block with generic bus interface. User can use the default value for most of the switch registers. The following Table 1 describes all registers which need to be set to a value other than the default value and read back to verify.

Register Name [bit]	Description
BAR[15:0]	Base address for KSZ88xxM. It will load from external EEPROM if EEEN=1, otherwise this value is default 0x0300
MARH[15:0] MARM[15:0] MARL[15:0]	Host MAC address. It will load from external EEPROM if EEEN=1, otherwise this value is written by user. MARH[15:0]->MAC[47:32], MARM[15:0]->MAC[31:16], MARL[15:0]->MAC[15:0]
MACAR1[15:0] MACAR2[15:0] MACAR3[15:0]	This MAC address is used for sending PAUSE frame. This value normally is same as MARH, MARM and MARL. MACAR1[15:0]->[47:32], MACAR2[15:0]->[31:16], MACAR3[15:0]->[15:0]
SGCR1[8]	This bit set 1 to enable more aggressive back off algorithm in half-duplex mode to enhance performance
SGCR2[3]	The switch does not drop packets when 16 or more collisions occur if this bit set to 1
SGCR3 [5]	This bit set 1 to enable full-duplex flow control on switch Host port
P1/2CR2[12:11]	The bit 12 set 1 always to enable flow control on the port, regardless of AN result The bit 11 set 1 to enable port's half-duplex back pressure
P1/2CR4[13]	This bit set 1 to restart auto-negotiation
SIDER [15:0]	Bit [15:1] is used to verify family ID, chip ID and revision ID. Bit [0] is set 1 to start the switch (for two port) or operate (for one port) function
QRFCR[12]	This bit set to 1 to configure the watermark at 2KBytes

Table 1. Initialization Setting for Switch and PHY Function Block Registers

Initialization Setting for Transmit Function Block Registers

This section describes the typical register settings for transmitting packets from host processor to KSZ88xxM with generic bus interface. User can use the default value for most of the transmit registers. The following Table 2 describes all registers which need to be set to a value other than the default value and read back to verify.

Register Name [bit]	Description
TXCR[3:0]	Set transmit control function as below: Set bit 3 to enable transmit flow control. Set bit 2 to enable transmit padding. Set bit 1 to enable transmit CRC. Set bit 0 to enable transmit block operation.
TXFDPR[14]	Set bit 14 to enable QMU transmit frame data pointer auto increment.
ISR[15:0]	Write 1 (0xFFFF) to clear all interrupt status bits in Interrupt Status Register.
IER[14]	Set bit 14 to enable transmit interrupt.

Table 2. Initialization Setting for Transmit Function Block Registers

Initialization Setting for Receive Function Block Registers

This section describes the typical register settings for receiving packet from KSZ88xxM to host processor with generic bus interface. User can use the default value for most of the receive registers. The following Table 3 describes all registers which need to be set to a value other than the default value and read back to verify.

Register Name [bit]	Description
RXCR[10][7:5][3][0]	Set receive control function as below: Set bit 10 to enable receive flow control. Set bit 7 to enable receive broadcast frames. Set bit 6 to enable receive multicast frames Set bit 5 to enable receive unicast frames Set bit 3 to enable receive to strip CRC. Set bit 0 to enable receive block operation.
RXFDPR[14]	Set bit 14 to enable QMU receive frame data pointer auto increment.
IER[13]	Set bit 13 to enable receive interrupt.

Table 3. Initialization Setting for Receive Function Block Registers

Driver Routine for Transmit Packet from Host Processor to KSZ88xxM

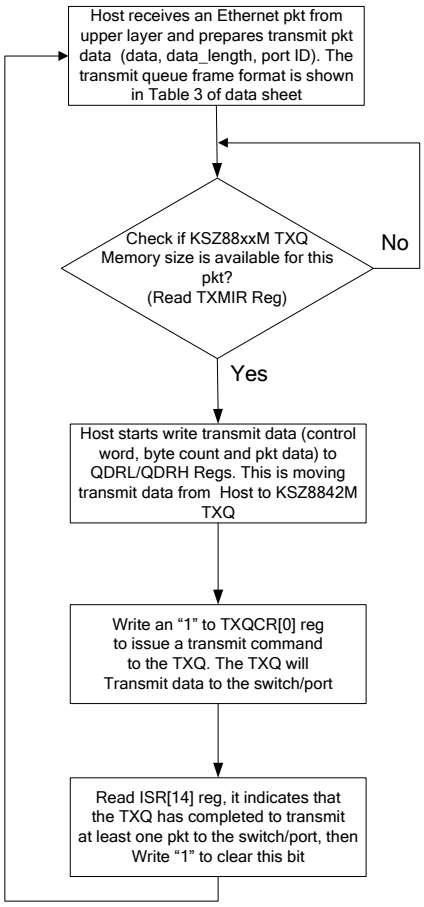
The transmit routine is called by the upper layer to transmit a contiguous block of data through the Ethernet controller. It is user's choice to decide how the transmit routine is implemented. The software may elect to wait until the Ethernet controller is ready to transmit new data and then synchronously send the new frame, or it may append the new frame to a queue in software and transmit the data asynchronously when a transmit completed interrupt signals that the controller is able to accept a new transmit request.

If the Ethernet controller encounters an error while transmitting the frame, it's the user's choice to decide whether the driver should attempt to retransmit the same frame or discard the data.

Table 4 and the flowchart show the steps for transmit packet from host processor to KSZ88xxM.

Sequence	Register Name[bit]	Description
0	Host CPU	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. Process the following from step 1 to 8 for every transmit packet to KSZ88xxM.
1	TXMIR [12:0] Bank 16, 0ffset 0x08	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	QDRL[15:0] Bank 17, 0ffset 0x08	Write the frame ID and destination port to “control word” of the first 16-bit to QDRL register. The internal switch engine forwards the packets according to the control word.
3	QDRH[15:0] Bank 17, 0ffset 0x0A	Write the packet length (**txPacketLength**) to the “byte count” of the second 16-bit to QDRH register.
4	QDRL[15:0] Bank 17, 0ffset 0x08	Write 2-byte of Ethernet packet data pointer by **pTxData** to the QMU TXQ through the QDRL register. Increase **pTxData** pointer by 2.
5	QDRH[15:0] Bank 17, 0ffset 0x0A	Write 2-byte of Ethernet packet data pointer by **pTxData** to the QMU TXQ through the QDRH register. Increase **pTxData** by 2.
6	Host CPU	Subtract **txPacketLength** by 4. If **txPacketLength** > 0 goto step 4, else goto step 7.
7	TXQCR[0] Bank 17, Offset 0x00	Write “1” to TXQCR[0] to issue the ENQUEUE transmits command for the KSZ88xxM to transmit the Ethernet packet to the network.
8	ISR[14] Bank 18, Offset 0x02	When this bit is set, it indicates that the TXQ has transmitted at least a frame to the network and the QMU TXQ is ready for new frames from the host CPU.

Table 4. Transmit Packet from Host Processor to KSZ88xxM



KSZ88xxM Transmit Flowchart

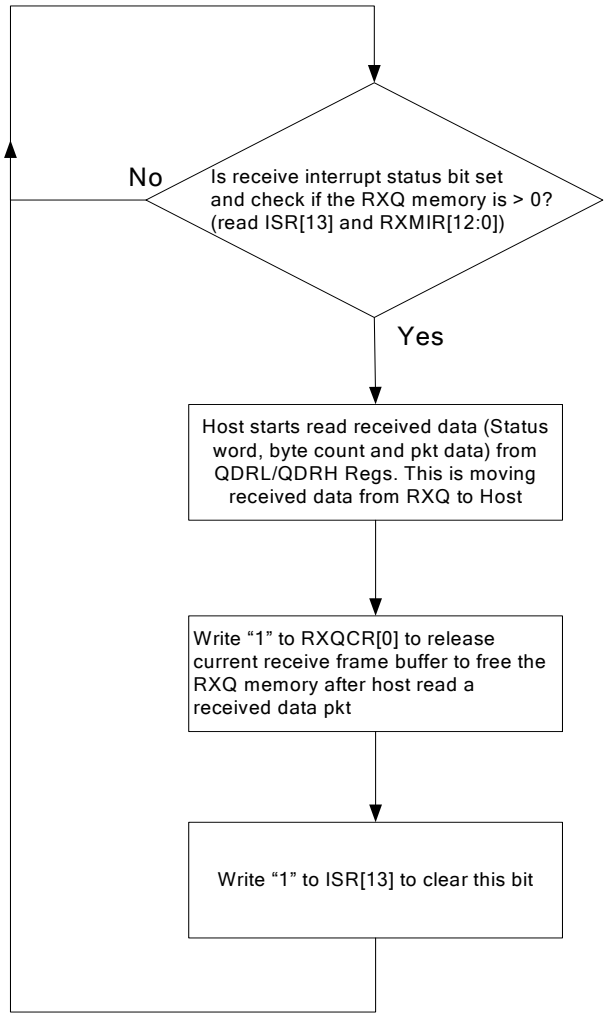
Receive Packet from KSZ88xxM to Host Processor

The software driver receives data packet frames from the KSZ88xxM device either as a result of polling or an interrupt based service. When an interrupt is received, the OS invokes the interrupt service routine that is in the interrupt vector table.

If your system has OS support, to minimize interrupt lockout time, the interrupt service routine should handle at interrupt level only those tasks that require minimum execution time, such as error checking or device status change. The routine should queue all the time-consuming work to transfer the packet from the KSZ88xxM RXQ into system memory at task level.

Sequence	Register Name[bit]	Description
0	Host CPU	There are two methods to receive a complete Ethernet packet from KSZ88xxM 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 to execute step 1. Allocate a system memory space (address by **pRxData**) which big enough to hold an Ethernet packet frame.
1	ISR [13] Bank 18, 0ffset 0x02	Read value from ISR to check if RXIS Receive Interrupt bit is set. If this bit is set, go to next step 2. If this bit is not set, then Exit.
2	ISR [13] Bank 18, 0ffset 0x02	Write "1" to acknowledge and clear RXIS Receive Interrupt bit.
3	RXMIR[12:0] Bank 16, 0ffset 0x0A	Read value from RXMIR to check if QMU RXQ still has more packet data to be read. If read value <= 0, Exit.
4	QDRL[15:0] Bank 17, 0ffset 0x08	Read first 2-byte of "status word" from QDRL to check if this is a good frame. If the bit [15] is 0 in "status word", goto step 9; If one of the bit[0], bit[1] or bit[2] is 1, goto step 9.
5	QDRH[15:0] Bank 17, 0ffset 0x0A	Read next 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, goto step 9. Otherwise goto next step 6.
6	QDRL[15:0] Bank 17, 0ffset 0x08	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.
7	QDRH[15:0] Bank 17, 0ffset 0x0A	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.
8	Host CPU	Subtract **rxPacketLength** by 4. If **rxPacketLength** > 0 goto step 6, else goto step 9.
9	RXQCR[0] Bank 17, 0ffset 0x02	Write "1" to RXQCR[0] to release the current receive frame buffer. Goto step 3 to see if there is more data frame in the QMU RXQ to be read.

Table 5. Receive Packet from KSZ88xxM to Host Processor



KSZ88xxM Receive Flowchart

Hardware Platform Supported by KSZ88xxM Drivers

Other than the platform/OS independent KSZ88xx hardware driver, Micrel also provides the following platform/OS dependent sample drivers as shown in the Table 6.

Buses	Microprocessor	Operating System	Protocol Stack	Compiler Option Definition	Software Driver
8-Bit Generic Bus	Zilog eZ80L92	ZTP 1.3.2	ZTP 1.3.2	_EZ80L92	KS88xx
	Renesas M16C/62P	None	OpenTCP 1.0.4	M16C_62P	KS88xx
16-Bit Generic Bus	Renesas M16C/62P	None	OpenTCP 1.0.4	M16C_62P	KS88xx
32-Bit Generic Bus	Renesas SH7751R	vxWorks 5.5.1 Tornado 2.2.1	vxWorks 5.5.1 Tornado 2.2.1	DEF_VXWORKS	KS88xx
	Renesas SH7760	WinCE 5.0	WinCE 5.0	_WIN32	KS88xx
		Linux 2.4/2.6	Linux 2.4/2.6	DEF_LINUX	KS88xx
		Windows 2000/XP	Windows 2000/XP	_WIN32	KS88xx

Note: All the KS88xx software drivers are available upon request

Table 6. KSZ88xx Driver Platform Support

The KS88xx driver can be built for different sets of drivers:

- KS8841 driver for KSZ8841M generic bus device
- KS8842 driver for KSZ8842M/KSZ8862M generic bus device

The drivers are designed to operate the KSZ88xx devices and demonstrate their hardware features. They share some common code that is called the KS884X library. This library provides an application programming interface to program the KSZ88xx hardware. Because the library code is shared in several platforms, the drivers may not run efficiently. To increase performance, the conditional INLINE can be defined to put some functions inline.

The generic version of the KSZ884X driver is used for generic bus. It uses a list of banks to access registers.

There are some identifiers that must be defined from the compiler options to generate a particular driver for a particular device.

DEF_KS8841	DEF_KS8842	KS_ISA_BUS	KS_ISA	Driver
Yes	No	Yes	Yes	KSZ8841 Generic Bus Driver
No	Yes	Yes	Yes	KSZ8842/8862 Generic Bus Driver

Table 7. Definition to Generate a Particular KSZxx Driver

KSZ88xxM Driver Application Programming Interface (API) Reference

The KSZ88xx driver contains a rich set of API functions that are used to configure all the KSZ88xxM device features. Functions within the API are organized within logical groups and alphabetized within each group. A description of each API group is shown in Table 8.

API Name	Function Description
Device Accesses APIs	Provides functions and macros to read /write device registers.
Device Initialization APIs	For device initializations.
Device Interrupt APIs	To handle the device interrupts.
Device Transmit APIs	For target host CPU transmits the packet from host system memory to device.
Device Receive APIs	For target host CPU receives the packet from device to the host system memory.
Set Device PHY APIs	Provides functions to configure device PHY operation.
Set Device Ports APIs	Provides functions to configure device port operation.
Set Device LinkMD APIs	Provides functions to configure device LinkMD feature to test cable diagnostics capabilities to determine cable length, diagnose faulty cables, and determine distance to fault.
Set Wake-on-LAN APIs	Provides functions to configure device Wake-on-LAN function for KSZ8841/61M only. When device detects a wake up event by receipt of a Magic Packet, it asserts the PMEN pin to low which could connect to host system to put the system into a powered state.
Set Device STP APIs	Provides functions to set Spanning Tree states on the device for KSZ8842/62M only.
Set Device VLAN APIs	Provides functions to configure VLAN function on the device for KSZ8842/62M only.
Set Device Rate Limiting APIs	Provides functions to configure broadcast storm protection and rate-limiting control at the ingress and egress ports on the device KSZ8842/62M only.
Set Device QoS APIs	Provides functions to configure QoS priority control, including port-based priority, 802.1p based priority and DiffServ based priority on the device KSZ8842/62M only.
Set Device Mirror APIs	Provides functions to configure Port Mirroring, Monitoring and Sniffing operations on the device KSZ8842/62M only.
Set Device Table Accesses APIs	Provides functions to read/write device indirect registers including up to 16 group VLAN table, 8 entries of static MAC table, 1K entries of dynamic MAC table and 34 MIB counters per port for KSZ8842/62M only.

Note: Please refer to “KSZ88xx Programming Guide” for more detail.

Table 8. KSZ88xxm Driver API Groups

Conclusion

This application note has highlighted all software drivers and APIs available for the KSZ88xxM Embedded family of Ethernet Controllers.

In order to meet the needs of diversified embedded system design, Micrel provides all software drivers to support the following popular OS platforms:

- Windows CE 5.0 & 6.0
- Linux 2.4
- Linux 2.6
- VxWorks 5.5 (For Renesas SH7551R)

MICREL, INC. 2180 FORTUNE DRIVE SAN JOSE, CA 95131 USA
TEL +1 (408) 944-0800 FAX +1 (408) 474-1000 WEB http:/www.micrel.com