QE TDM Working with QUICC Multi-channel Controller

1. QE TDM QMC Driver Introduction

Time division multiplexing(TDM) is a communication term for multiplexing several channels on the same link. QUICC multi-channel controller(QMC) is a firmware package which uses a unified communication controller(UCC) working in slow mode. The QMC is used to emulates up to 64 time-division serial channels through a time-division-multiplexed(TDM) physical interface. Each of QMC channels can be independently programmed to support either HDLC or transparent protocols. This document introduces TDM QMC driver implementation in Linux Kernel for the processors with QE UCC working in slow mode.

The driver configures QE block to establish independent channels on different QMC devices. Each QMC device, which bounds a specified TDM and a specific UCC, can support up to 64 channels. Programming of the SI and SIRAM for routing of the timeslots with a TDM stream to the appropriate QMC channel. The SI contains a dedicated RAM, which keeps the configurations of TSAs for Tx&Rx. The SI RAM is organized in time slots, 512 for Tx and 512 for Rx. The time slots are divided per TDM line.

Data flow over a QMC channel involves a TDM line and a UCC, working in slow mode. For each channel, Tx data flow consists of data transfer from the external memory to the TDM physical connection. Rx data flow consists of data transfer from the TDM physical connection to the external memory. In both data flows, the major stations are: data buffers, UCC, and the TDM line. Please refer to the following figure for the data flow, the driver requires to configure two levels of routing tables. The first level consists of the SI RAM routing tables, Tx and Rx, which are common to other controllers as well. The tables route data between a TDM line and a specified UCC. The second level of routing consists the QMC time-slot assignment(TSA) tables, Tx and Rx. These tables resides in the MURAM, and responsible to route data between the UCC to a specific channel data buffer.

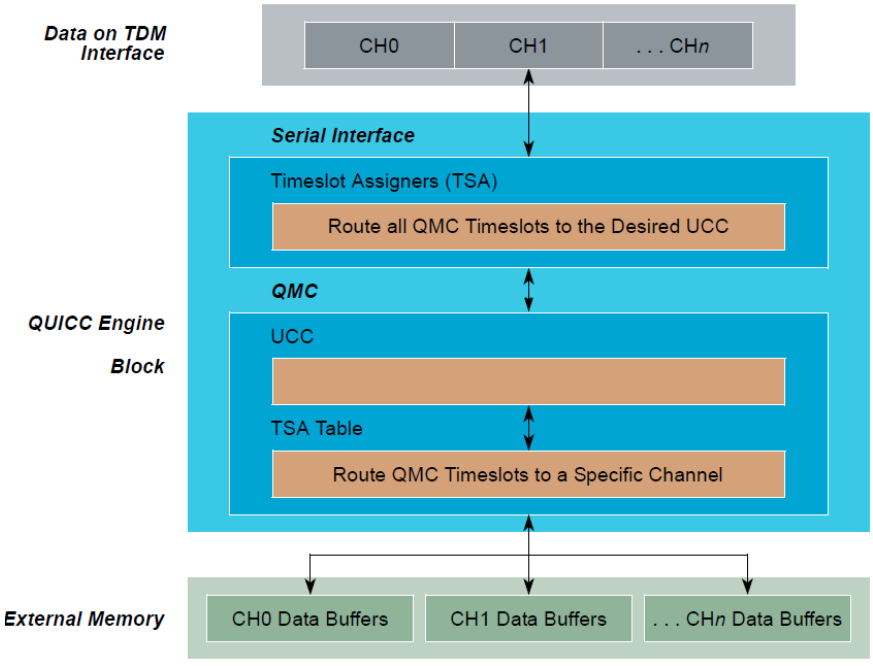


Figure 1.1 QE TDM QMC Driver Data Flow

For each channel, data flow can be made using HDLC or transparent protocols. In HDLC mode while transmitting very two HDLC frames, the QMC inserts a unique idle pattern. This pattern is being removed on reception by the QMC. This pattern identifies the beginning and end of each HDLC frame, which allows the application to deal only with the valid data. In transparent mode, all data is transferred to the application, which should remove the idle bytes and knows how to recognize the beginning and the end of the actual data that was received.

2. Driver Architecture and Components

This section describes some key points in the TDM QMC driver design.

2.1 QMC Driver Memory Allocation

QMC memory structure is defined as the following. QMC memory allocation is implemented in function utdm_init.

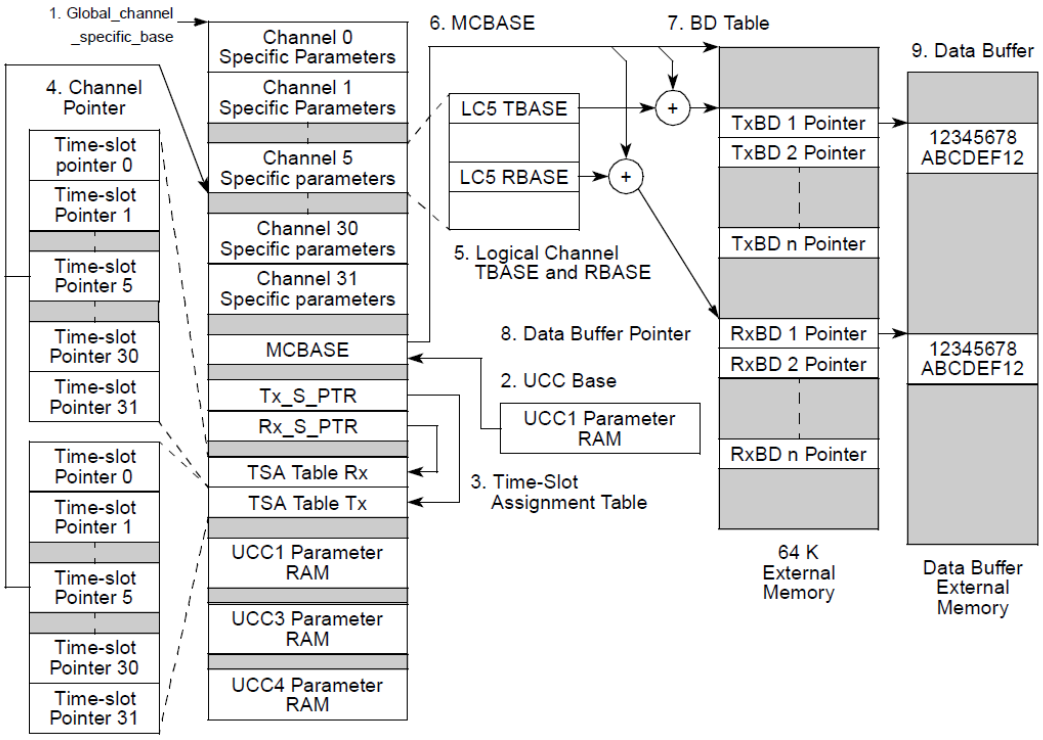


Figure 2.1 QMC Driver Memory structure

UCC based and global multichannel parameters, the UCC base points to the start of the parameter RAM for each UCC. When QMC protocol is enabled on a UCC, its parameter RAM is used to store the global multichannel parameters for all the logical channels.

The time-slot assignment table pointers are within the global multichannel parameters. There are two pointers, Tx_S_PTR for transmit and Rx_S_PTR for receive. The Rx_S_PTR is normally set to UCC Base + 0x20; this is the normal location of the receive time slot assignment table. The Tx_S_PTR is normally set to UCC Base+0x60, this is the normal location of the transmit time-slot assignment table.

TSATRx/TSATTx channel pointers. The channel pointers are 12-bit pointer to the channel-specific parameters in the internal multi-user RAM. The 6 most-significant bits of the address are taken from the time slot assignment table. The 6 least-significant bits are zero, mapping out a 64-byte area for each of the channel-specific parameters.

TBASE and RBASE are within the channel-specific parameters. TBASE is the Tx buffer descriptor base address, and RBASE is the Rx buffer descriptor base address. These 16-bit offsets from MCBASE point to individual logical channel's buffer descriptors located within the buffer descriptor table.

MCBASE is located in the global multichannel parameters. Each UCC has a unique MCBASE value pointing to base of the UCC's buffer descriptor table in external memory. MCBASE normally points to external RAM, but it is permissible to set it up so that some or all BDs are placed within free areas of the MURAM.

A buffer descriptor table for each UCC is located in a 64K area of external memory. This block size is determined by TBASE and RBASE addressing range. For a 32 channel implementation, each logical channel has a maximum of 128 buffers for receive and 128 buffers for transmit.

Data buffer pointer is addressed by a 32 bit pointer within the buffer descriptor, this addresses the data received or transmitted from external memory.

Data buffer in external memory can hold up to 64 Kbytes of data as determined by the data length in the buffer descriptor.

The global multichannel parameters reside in the UCC's parameter RAM page and are common to all logical channels.

The largest portion of the global area is the time-slot assigner tables for the receiver and transmitter section of the UCC. For 32 channel support, there is one table for Tx and one for Rx within the parameter RAM. The multi-user RAM is used for the channel-specific area for all UCCs. It is important that individual time slots are mapped to only one UCC.

2.2 QMC and TDM Devices Initialization

2.2.1 SI RAM entry initialization

Function init_si sets and validates SIRAM tables, configures SI mode register SIxMR. SI contains a dedicated RAM, which keeps the configuration of the four TSAs for Tx&Rx. The SI RAM is organized in time slots, 512 for Tx and 512 for Rx, the time slots are divided per TDM line.

SI RAM is shown as the following.

Access: Read/Write

	0	1	2	3	4	5	6	7	10	11	13	14	15
R W	—	SWTR	SSEL 1	SSEL 2	SSEL 3	SSEL 4	—	CSEL		CNT		BYT	LST
Reset	All zeros												

CSEL: Channel select. In the driver configures this field as the following.

0001: The bit/byte group is routed to UCC5.

1001: The bit/byte group is routed to UCC1.

1010: The bit/byte group is routed to UCC2.

1011: The bit/byte group is routed to UCC3.

1100: The bit/byte group is routed to UCC4.

CNT: Count. Indicates that number of bits/bytes that the routing and strobe select of this entry control.

SI Mode Register(SIxMR) is configured as the following. There are four SIxMR registers, one for each TDM.

Offset SIAMR 0x00; SIBMR 0x02; SICMR 0x04; SIDMR 0x06
SIEMR 0x20; SIFMR 0x22; SIGMR 0x24; SIHMR 0x26 Access: Read/Write

	0–3	4–5	6–7	8	9	10	11	12	13	14–15
R W	SAD	SDM	RFSD	—	CRT	SL	CE	FE	GM	TFSD
Reset	All zeros									

SDM: SI Diagnostic Mode for all four TDMs. 00, normal operation. 01, Automatic echo, 10 internal loopback. 11, Loopback control, this mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines.

CRT: Common receive and transmit pins for all TDMs.

SL: Sync level for all TDMs.

CE: Clock edge for all TDMs.

FE: Frame sync edge for all TDMs.

GM: Grant mode for all TDMs.

2.2.2 UCC Slow Mode QMC Initialization

Function ucc_slow_qmc_init sets UCC to slow type, sets GUMR registers, configures UPSMR normal mode, grant support, breakpoint support, TSA or MNSI mode and sets interrupt mask register at UCC level.

General UCC Mode Registers(GUMR) defines options common to most of the protocols. GUMR_L, contains the low-order 32 bits; GUMR_H, contains the high-order 32 bits. These registers is configured in ucc_tdm_probe function.

2.2.3 QMC Channel Initialization

Function qmc_ch_init initializes channel-specific parameters, initializes TxBDs and corresponding Tx data buffers, Initialize RxBDs, set the offset from the base address of the Rx/Tx BDs block.

The receive buffer descriptor is described as the following.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	—	**W**	**I**	L	F	**CM**	—	**UB**	—	LG	NO	AB	CR	—	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	Rx DATA BUFFER POINTER															
OFFSET + 6																

Notes: Entries in boldface must be initialized by the user.

E: Empty. 0: The data buffer associated with the BD has been filled with received data. 1: The data buffer associated with BD is empty. Configure this field as 1 in the initialization.

I: Interrupt. 0: The RXB bit is not set after this buffer has been used. 1: The RXB or RXF bit in the HDLC interrupt circular table entry is set. Configure this field as 1 in the initialization.

W:Wrap(final BD in table). Configure this field as 1 for the last BD in the RxBD table.

2.2.4 QMC TSA Slot Initialization

The function int qmc_init_tsa_slot is used to initialize TSA slot, it prepare the initialized data, calculate slot address and write data to tsa slot. When the QMC channel is super channel mode, synchronize channel parameters according to the new slot route by invoking

qmc_update_trans_ch_sync. The workflow of the function qmc_update_trans_ch_sync is described as the following.

For Rx slot, updating the smallest slot number if necessary. If this slot is the first byte slot, setting transparent synchronization bit in channel CHAMR register. If the user didn't mark any other byte as first, or if this slot was previously defined as first and undefined now, the smallest slot is the first by default.

For Tx slot, updating the highest slot number if necessary. If this slot is the first byte slot, setting transparent synchronization bit in channel CHAMR register, searching backwards for the new last Tx slot. If this slot is not the first slot, check the current slot is a new last slot as the following.

If the first Tx slot index is lower than the current last Tx slot index, the new last Tx slot can be any of the slots below marked with #:

| # | # | # | # |First| | | | | |Last| # | # | # |

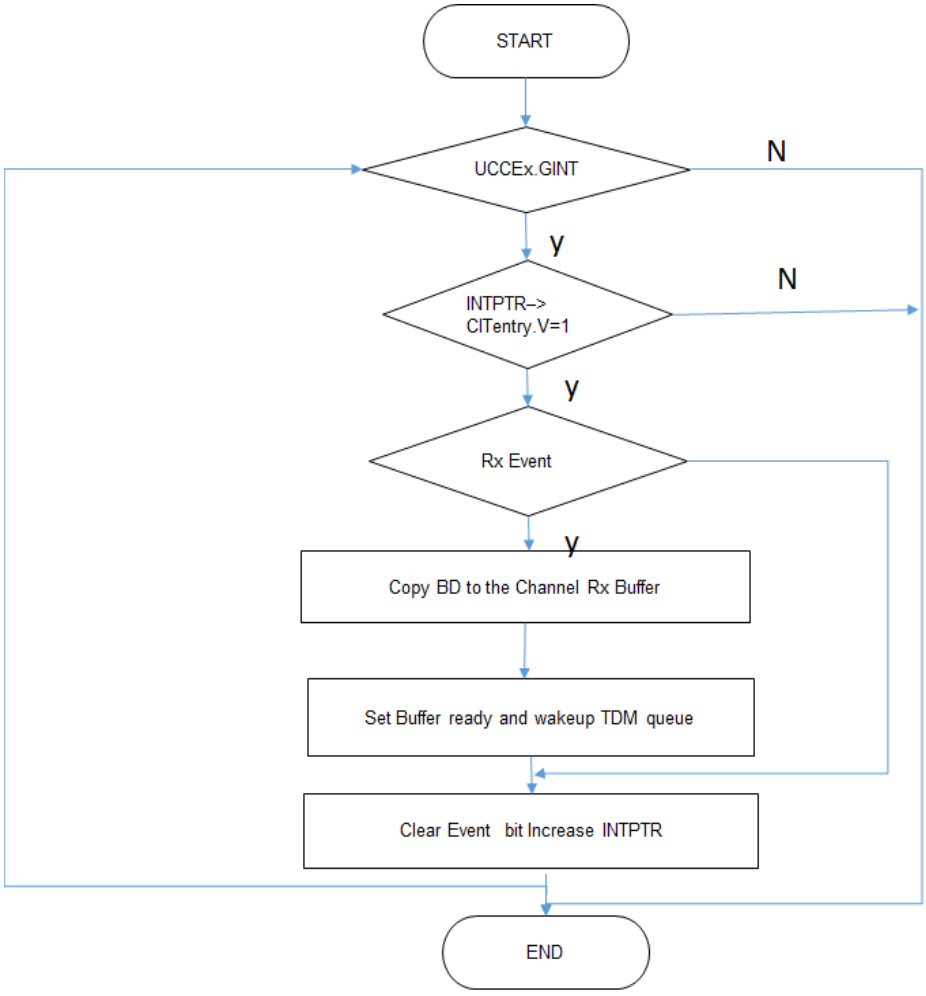
If the first Tx slot index is larger than the current last Tx slot index, the new last Tx slot can be any of the slots below marked with #:

| | | | |Last| # | # | # | # | # |First| | | |

If this slot was previously defined as the first and undefined now, the highest Tx slot is the last by default.

2.3 QMC Channel Interrupt Handling

The function is used to handle interrupts by the QMC, first read the UCCE register. For a new GINT event, always handle all the new entries in the queue, not just a single one. After handling an entry, make the entry is completely cleared out with the exception of Wrap bit. Any entry that does not have the valid bit set must be completely cleared out. The QMC channel interrupt handling flow is as the following.



2.4 QMC and TDM Configuration

2.4.1 Enable and Configure QMC Channel

QMC channel is configured as Transparent or HDLC mode through functions qmc_trns_init and qmc_hdlc_init.

QMC channel HDLC mode configuration:

Initialize QMC channel with the default configuration and configure the type as HDLC.

Set the CRC size.

Enable the CRC transmission on the last BD.

Mask non relevant interrupts.

Turn on RXF interrupt as requested.

Set channel's serial number.

Init the states fields

Save Rx/Tx state parameters according to HDLC mode.

Set MFLR field according to mode.

Set the CHAMR field according to user input and spec.

The transparent channel mode configuration sequence is as the following:

Initialize the QMC channel and configure it as TRNS channel type.

Initialize the ZDSTATE and ZISTATE fields.

Save TSTATE and RSTATE according to TRNS mode.

Set TMRBLR according to channel mode.

Set CHAMR mode register

Set interrupt mask, mask non relevant interrupt

Turn on RXF INTR as requested

Set channel's serial number.

2.4.2 Enable QMC

Configuring general UCC mode registers(GUMR) to enable receiver and transmitter hardware state machine for the UCC.

Offset 0x0 Access: Read/Write

	0–2	3	4–5	6	7	8–12	13	14–15
R/W	—	TCI	—	RINV	TINV	—	TEND	TDCR
Reset	All zeros							

	16–17	18–20	21–23	24–25	26	27	28–31
R/W	RDCR	RENC	TENC	DIAG	ENR	ENT	MODE
Reset	All zeros						

ENR: Enable receive. Enables the receiver hardware state machine for this UCC.

0: Reset the UCC receive. 1: The receiver is enabled.

ENT: Enable transmit. Enable the transmitter hardware sate machine for this UCC.

0: Reset the UCC transmitter. The transmitter is disabled. 1: The transmitter is enabled.

2.4.3 Enable TDM

TDM is enabled through configure SI Global Mode Register High(SIGLMRH).

Offset 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7
R/W	STZD	STZC	STZB	STZA	END	ENC	ENB	ENA
Reset	All zeros							

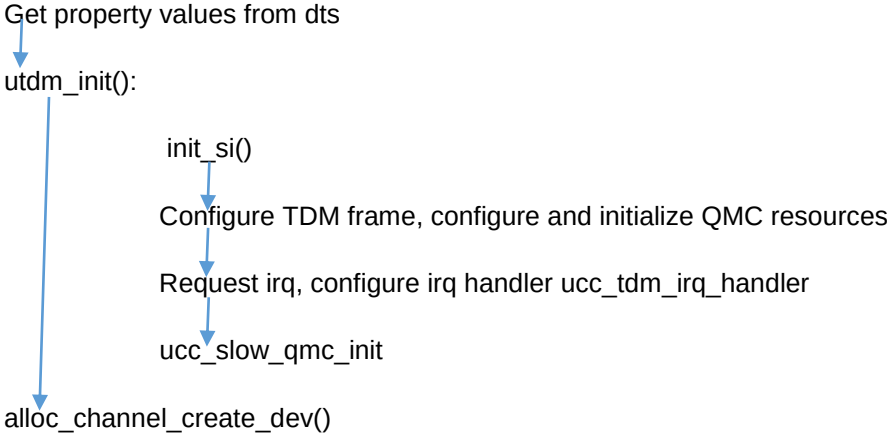
ENx: Enable TDMx.

0 TDMx is disabled.

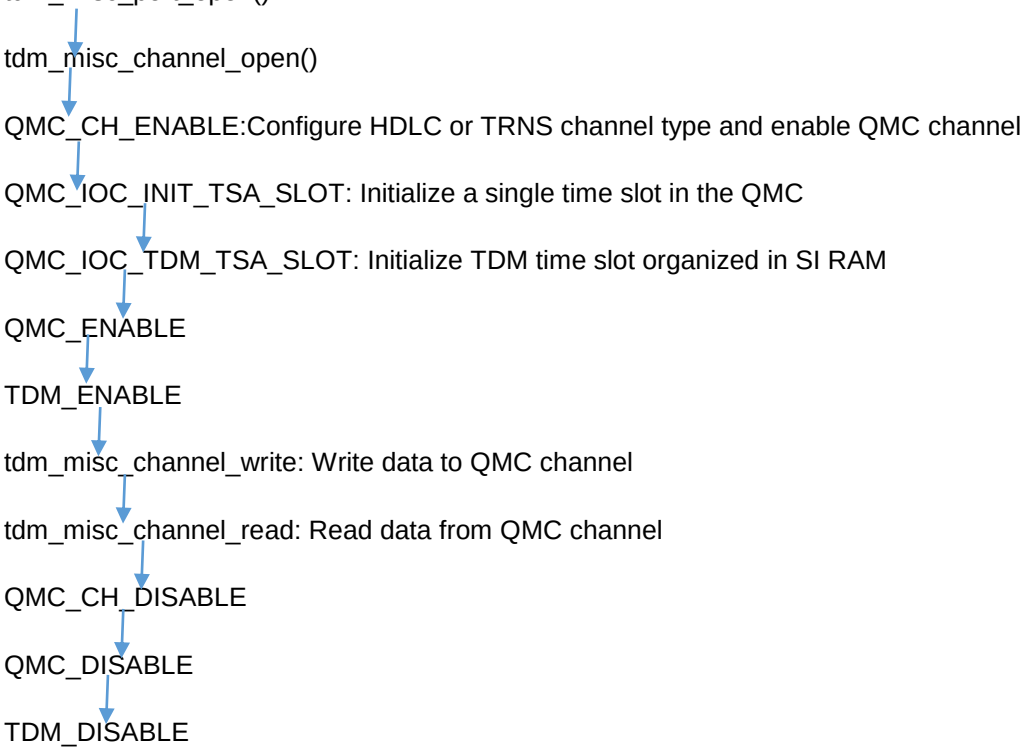
1 All TDMx functions are enabled.

3. QMC TDM Driver Calling Sequence

QMC TDM driver probe function:



API invoking sequence in the application:



4. QMC TDM DTS Definition

TDM device node definition is as the following:

```
tdma: ucc@2000 {
        compatible = "fsl,ucc-tdm";
        cell-index = <0>; /* UCC number */
        rx-clock-name = "clk3"; /* Rx clock source*/
```

```
                tx-clock-name = "clk4"; /* Tx clock source*/
                fsl,rx-sync-clock = "rsync_pin"; /* Rx Sync clock */
                fsl,tx-sync-clock = "tsync_pin";  /* Tx Sync clock */
                fsl,tx-timeslot = <0xfffffffe>;   /* Tx time slot mask */
                fsl,rx-timeslot = <0xfffffffe>;   /* Rx time slot mask */
                pio-handle = <&pio_tdma>;
                fsl,tdm-framer-type = "e1";    /* TDM framer type E1 or T1*/
                fsl,tdm-mode = "internal-loopback"; /* TDM mode internal-loopback or normal */
                fsl,tdm-id = <0>;      /* TDM port number */
                fsl,siram-entry-id = <0>;  /* SIRAM entry ID for UCC */
};
```

5. Configure QMC TDM Driver and Running the Testing Program

Deploy attached QMC TDM driver in the Kernel source code and configure Linux Kernel as the following.

```
Device Drivers  --->
    <*> TDM support  --->
        TDM test  --->
            < > TDM test Module
            < > UCC TDM test Module
            < > TDM Loopback test Module
            <*> UCC TDM QMC Loopback test Module
        TDM Device support  --->
            < > Driver for Freescale TDM controller
            < > UCC TDM driver for Freescale QE engine
            <*> UCC QMC TDM driver for Freescale QE engine
```

Set up Linux Kernel on the target board, and get the test result as the following.

```
QE TDM LOOPBACK TEST:
TDM Driver(ID=1)is attached with Adapterfsl_tdm(ID = 0) drv_count=1
QE TDM LOOPBACK TEST module installed
QMC_CH_ENABLE
QMC_ENABLE
TDM_ENABLE
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
ipip: IPv4 over IPv4 tunneling driver
TCP: cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 10
sit: IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
Iteration: 8
```

TX DATA:
230
231
232
233
234
235
236
237
238
239
23a
23b
23c
23d
23e
23f
240
241
242
243

RX DATA:
230
231
232
233
234
235
236
237
238
239
23a
23b
23c
23d
23e
23f
240
241
242
243

TX and RX buffer MATCH
QE TDM Loopback test completed.
Internal QE TDM loopback test PASSED!
QMC_CH_DISABLE
QMC_DISABLE
TDM_DISABLE