DOCUMENT NUMBER
S12IQUEV1/D

IQUE
Block Guide
V1.6

Original Release Date: 18 Oct 2001
Revised: 29 Nov 2004

TSPG 8/16 Bit MCU
Freescale Semiconductor, Inc.

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
0.1	18 Oct 2001	18 Oct 2001	Jim Sibigtroth	Initial release
0.2	20 Nov 2001	20 Nov 2001	Johnson Pang	Change from Que to Integrated Queue module Add system and internal block diagram and description for Integrated Queue module Modify the Introduction, Memory/Register definition and section Add 8-bit data transfer operation description Add data transfer protocol diagram and description
0.3	28 Nov 2001	28 Nov 2001	Johnson Pang	Change QUE RAM Interface to QUE Integration Module Add Application Information section
0.4	28 Dec 2001	28 Dec 2001	Johnson Pang	Revise Modes of Operation section Add IQUE control register, including module reset and module enable field. Add Buffer full interrupt feature in each channel. Revise Functional description and Application Information sections.
1.0	17 Jan 2002	17 Jan 2002	Johnson Pang	Revise the mode of operation section Revise the Memory Map section Revise the waveform diagrams in Function Description section
1.1	9 Apr 2002	9 Apr 2002	Johnson Pang	Add Programming Information
1.2	30 Aug 2002	30 Aug 2002	Johnson Pang	Add Double Buffer mode freature descriptions and related registers infromation
1.3	3 June 2003	3 June 2003	Johnson Pang	Change QC12DTR and QC34DTR to QC12DSHR and QC34DSHR respectively. Revise the address offset of QCnEP and QCnCR
1.4	5 Sept 2003	5 Sept 2003	Johnson Pang	Change the default value of QCnREQ register from 00 to 0F Change the double buffer passthrough mode to double buffer mode
1.5	27 Nov 2003	27 Nov 2003	Johnson Pang	Revise the description in Modes of Operation section Revise the Register definition description for Duble buffer operation Revise the Modes of Operation Remove the internal bus operation section
1.6	29 Nov 2004	29 Nov 2004	Wai-On Law	Changed company logo.

Table of Contents

List of Figures

List of Tables

Section 1 Introduction

This block guide describes a multi-channel general purpose Integrated Queue module (IQUE) that is compatible with the IP bus used in the HCS12 MCU families. The module manages the movement of data without any direct intervention from the CPU. A static 16-bit RAM block (QRAM) is embedded in the module. Data is written into or transferred out of the QRAM in FIFO fashion. The basic IQUE design can be adjusted to have from 2 to 8 channels.

Figure 1-1 is a high level block diagram of an MCU that includes the general purpose IQUE module. It shows the CPU, main MCU memory, and IP Bus Interface (IPBI) interconnected by the CPU system bus. Peripheral modules which require queue data transfer have direct data path and handshake control signals connected to IQUE module. The QRAM data can be accessed by the CPU through IP bus data port or directly through S12 Bus Interface with EEPROM address mapping, provided that the EEPROM module is not used in the system. Synchronous time multiplexing is used to control whether the CPU or the QUE controller has control of the QRAM FIFO buffer.

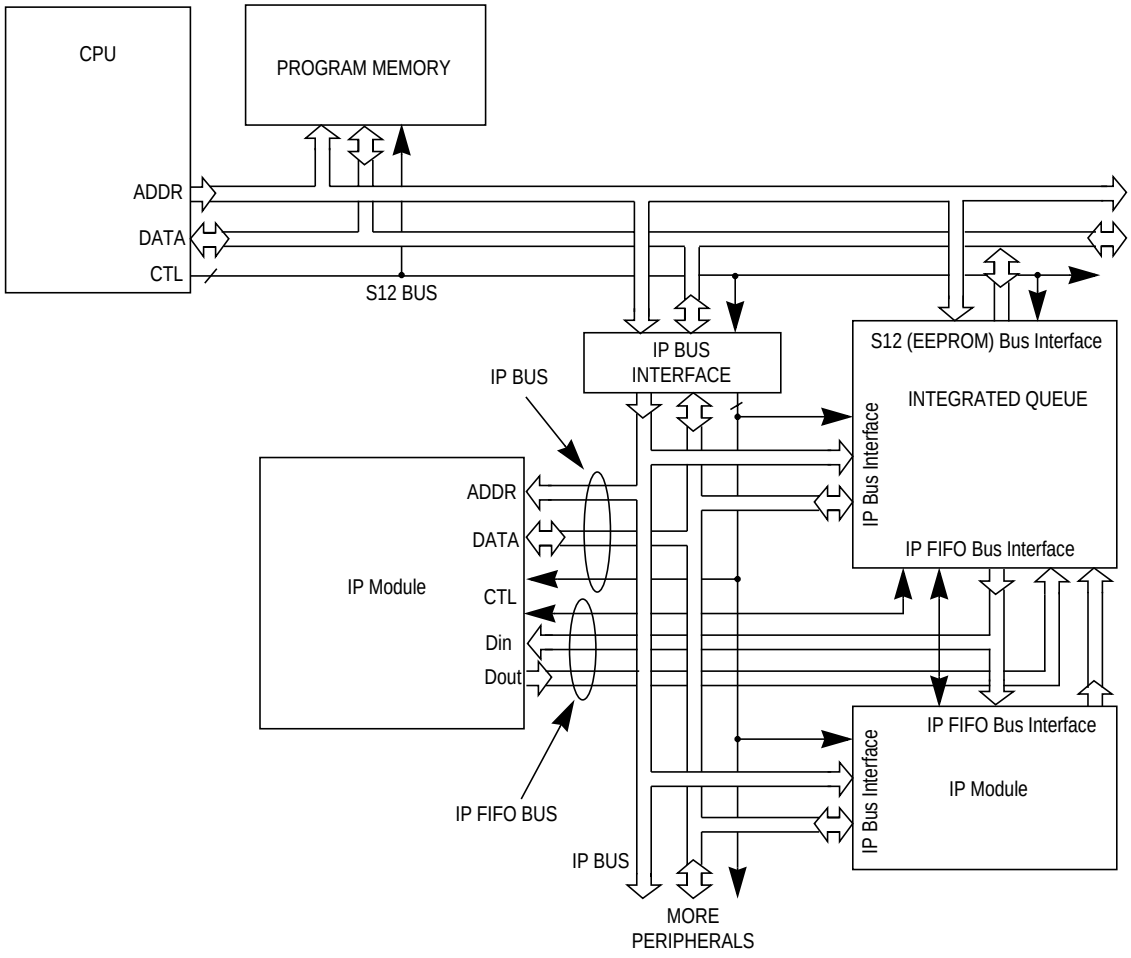


Figure 1-1 Block Diagram of an MCU with IQUE

Figure 1-2 is the block diagram of Integrated Queue module. It consists of three main blocks: QUE controller, QUE Integration Module (QIM) and a 16-bit static RAM block (QRAM). QUE controller has four channels that can move data one byte or one 16-bit word in a single IQUE clock cycle. The multiplex and control logic in the QIM allows QRAM sharing between QUE controller accesses and CPU accesses. The module has three interface ports: IP bus interface (SkyBlue-Line signals), IP FIFO interface (ForestGreen-Line signals) and S12 core interface (EEPROM bus signals).

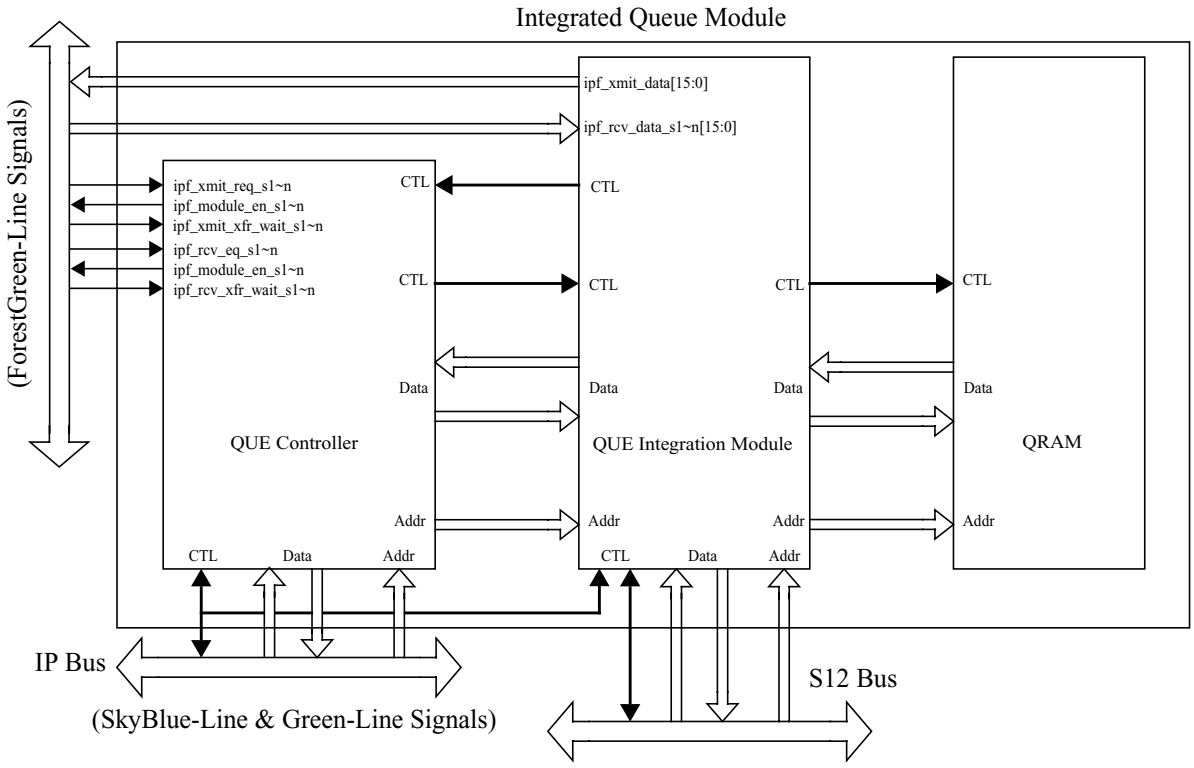


Figure 1-2 IQUE Block Diagram

The IQUE module operating clock frequency is targeted at 60MHz and CPU also operates at 60MHz or its frequency divider. For example, if the clock frequency of IQUE and CPU are 60MHz and 30MHz respectively. There would be four time slots for QRAM access for each CPU bus cycle in this case. One of these slots would be reserved for use by the CPU if selects indicate the CPU needs access to the QRAM during that bus cycle. The remaining slots are available to the QUE Controller. In a typical system, this fast peripheral would use every-other QRAM cycle or two of the four available time slots.

In case the CPU clock is increased to higher than 30MHz(i.e. 60MHz), QIM will assert the ee_hold signal to extend CPU read operation by one bus clock cycle, thus QRAM can be operating at constant speed.

1.1 Overview

IQUE allows automatic block data transfer without CPU intervention, and there is no need to steal CPU bus cycles because queue data transfers take place during time slots where the CPU is not using the QRAM. A fast channel is dedicated for higher data rate peripheral. This channel has guaranteed bandwidth which data transfer rate is not reduced even CPU utilizing the QRAM. Unlike a typical DMA controller, IQUE is specifically designed for use in an MCU in order to simplify some control logic and thus fit in a smaller die area. One side of every queue data transfer is always a data value from an internal MCU peripheral, and the other side of the transfer is always the QRAM. This greatly reduces the size and complexity of pointers, counters, and other control logic so the die area is much less than would be required for a traditional DMA controller.

During an automated queue data transfer, data is written into the QRAM or transferred out of the QRAM in FIFO fashion. The CPU can write information into the QRAM or read information out of the QRAM through registers in the QUE controller that act as FIFO data ports. In addition, the CPU can access information in the QRAM in random fashion with ordinary CPU read and write accesses. When information is accessed through the FIFO data port registers, pointers are automatically updated so the QRAM buffer behaves as a circular or linear FIFO. User can also directly write to the FIFO begin and end pointers.

The number of queue requesting sources are usually more than queue channels. IQUE includes control structure to allow a user to select one of several sources as the source for queue requests for each queue channel. At chip integration time, the system design engineer for a particular chip will assign (connect) peripheral requests and acknowledge signals to the appropriate inputs and outputs of IQUE according to their required data rate and priority. A bandwidth guaranteed queue channel (either channel 1 or 2) is designed for IP module which requires higher data transfer rate (Max. 60M btye/s).

1.2 Features

The IQUE module includes these distinctive features:

- Four channels (each channel programmable for 8-bit or 16-bit transfers)
- Up to 4K bytes of static RAM (QRAM) for queue buffers
- QRAM buffers act as circular or linear FIFOs
  - Independently programmable buffer sizes from 16 to 4K bytes
  - Independently programmable buffer base address within 4K bytes QRAM
  - QRAM buffers can act as software FIFOs when queue channel is disabled
  - Begin and end pointers are writable so software can undo a partial message transfer
- Flexible mapping of peripheral requests to queue channels
- Queue data transfers do not affect real time operation of the CPU
  - Data transfer rate up to one transfer per two QRAM clock cycles per channel (almost 30 million 16-bit transfers per second)

- Status flags for valid FIFO data, FIFO empty and full for each channel
  - each channel can optionally interrupt on valid data, empty or full condition

1.3 Modes of Operation

Queue channels can operate as either Receive queue or Transmit queue mode. Passthrough mode associates a receive channel and a transmit channel to the same QRAM FIFO buffer so data can be transferred from one peripheral to another through the buffer without any CPU intervention. In addition, the FIFO buffers associated with a queue channel can function as software FIFOs when the queue channel is disabled. **Table 1-1**summarize the data transfer operation modes supported by IQUE module.

- Receive queue mode

  When a queue channel is enabled and configured as Receive queue mode, data is automatically written into the FIFO buffer in response to a peripheral request. The user application program (CPU) reads data out of the FIFO buffer through data port register. If CPU do not request for data transfer, other peripheral will occupy the rest of the data transfer bandwidth. In this mode, CPU requests to write data into FIFO buffer through data port register are ignored.

- Transmit queue mode

  When a queue channel is enabled and configured as Transmit queue mode, the user application program (CPU) writes data into the FIFO buffer through data port register. Data is automatically read from the FIFO buffer in response to a peripheral request. If CPU do not request for data transfer, other peripheral will occupy the rest of the data transfer bandwidth. In this mode, CPU requests to read data from FIFO buffer through data port register are ignored.

- Passthrough mode

  Passthrough mode are supported to allow two queue channels to be linked so they share the same FIFO buffer. Data can be transferred into the FIFO buffer through a receive channel and automatically be sent out through a transmit queue channel - all without any CPU intervention other than the initialization instructions that established the passthrough setup. In this mode, CPU requests FIFO buffer read/write through data port registers are ignored. But CPU is allowed to access the QRAM data in random access fashion through S12 EEPROM bus interface.

- Block Base Trasnfer (Single or Double Buffer Tranfer) mode

  Double Buffer mode is supported to allow two queue channels to be linked so they share the same FIFO buffer similar to Passthrough mode with additional Ping-Pong double buffering feature. Unlike circular buffer in the other modes, in double buffer mode, two buffers are accessed in buffer base, i.e. buffer empty, valid and full is determined from entire buffer point of view. Buffer can only be read after the entire buffer is full, or written after entire buffer is empty. After one buffer is full, pointer will automatically switch to next buffer for further writting, provided that the next buffer is empty. When this mode is enabled, the actual total buffer size is equal to twice of the size specified in buffer size field. Buffer to be accessed is toggled automatically. The number of buffers to be transfered is programmable. The buffer empty and full status information is shown in the control and status registers.

- Software FIFO mode

  When a queue channel is disabled, peripheral requests are ignored and the queue channel does not request hardware interrupts. CPU accesses to the FIFO data port still access the QRAM in FIFO fashion through data port register, pointers are still updated normally, and the valid data and full status flags still operate normally. This allows a user to treat the QRAM buffer as a software FIFO.

- Stop and Wait mode

  When the MCU is in stop mode, no queue transfers take place in the queue system, and the operation of IQUE will be suspended immediately. It will be resumed once MCU is out of stop mode. When the MCU is in wait mode, queue transfers can continue to take place. Queue system interrupts can be used to wake the MCU from wait mode. Some peripheral modules have control bits that determine whether or not the peripheral remains active when the MCU is in wait mode. These controls would affect whether or not the peripheral would continue to issue queue requests when the MCU was in wait mode.

Table 1-1 Summary of IQUE data transfer modes

Circular Buffer	Block BaseTransfer	
Software FIFO Tx Que Rx Que Passthrough	Single Buffer	Double Buffer
	Software FIFO Tx Que Rx Que Passthrough	

Section 2 External Signal Description

There are no external signals (external pins) associated with the queue module.

Section 3 Memory Map/Register Definition

Table 3-1 shows the registers associated with the IQUE module.

Table 3-1 Module Memory Map

Address Offset	Use	Access
$00	IQUE Module Control Register(IQUECR)	R/W
$01	Reserved	
$02	QUE Ch. 1 FIFO Data Register (QC1DRH:QC1DRL)	R/W[1]
$04	QUE Ch. 1 Begin Pointer Register (QC1BP)	R/W
$06	QUE Ch. 1 End Pointer Register (QC1EP)	R/W
$08	QUE Ch. 1 Control Register (QC1CR)	R/W
$09	QUE Ch. 1 Status Register (QC1SR)	R[2]
$0A	QUE Ch. 1 Size/Base Register (QC1SZB)	R/W

Table 3-1 Module Memory Map

$0B	QUE Ch. 1 Request Mapping (QC1REQ)	R/W
$0C	QUE Ch. 2 FIFO Data Register (QC2DR)	R/W[1]
$0E	QUE Ch. 2 Begin Pointer Register (QC2BP)	R/W
$10	QUE Ch. 2 End Pointer Register (QC2EP)	R/W
$12	QUE Ch. 2 Control Register (QC2CR)	R/W
$13	QUE Ch. 2 Status Register (QC2SR)	R[2]
$14	QUE Ch. 2 Size/Base Register (QC2SZB)	R/W
$15	QUE Ch. 2 Request Mapping (QC2REQ)	R/W
$16	QUE Ch. 3 FIFO Data Register (QC3DR)	R/W[1]
$18	QUE Ch. 3 Begin Pointer Register (QC3BP)	R/W
$1A	QUE Ch. 3 End Pointer Register (QC3EP)	R/W
$1C	QUE Ch. 3 Control Register (QC3CR)	R/W
$1D	QUE Ch. 3 Status Register (QC3SR)	R[2]
$1E	QUE Ch. 3 Size/Base Register (QC3SZB)	R/W
$1F	QUE Ch. 3 Request Mapping (QC3REQ)	R/W
$20	QUE Ch. 4 FIFO Data Register (QC4DR)	R/W[1]
$22	QUE Ch. 4 Begin Pointer Register (QC4BP)	R/W
$24	QUE Ch. 4 End Pointer Register (QC4EP)	R/W
$26	QUE Ch. 4 Control Register (QC4CR)	R/W
$27	QUE Ch. 4 Status Register (QC4SR)	R[2]
$28	QUE Ch. 4 Size/Base Register (QC4SZB)	R/W
$29	QUE Ch. 4 Request Mapping (QC4REQ)	R/W
$2A	QC12 Double Buffer Control Status Register(QC12DCS)	R/W
$2C	QC34 Double Buffer Control Status Register(QC34DCS)	R/W
$2E	QC Double Buffer Software Handshake Register(QCDSHR)	R/W

NOTES:
1. Normally read-only for a receive queue or write-only for a transmit queue
2. Normally read-only, write (ones) used in clearing sequences.

3.1 Register Descriptions

This section consists of register descriptions in address order. This block guide describes a 4-channel IQUE module, but the module is intended to be expandable from two to eight channels.

3.1.1 IQUE module Control Register (IQUECR)

Address Offset: $00 (IQUECR)

	7	6	5	4	3	2	1	0
R	0	0	0	QC34DBE	QC12DBE	0	0	IQUEEN
W							IQUERST	
RESET:	0	0	0	0	0	0	0	0

= Reserved

Figure 3-1 IQUE Module Control Register (IQUECR)

The IQUE module can be disabled for power saving purpose. When the module is disabled, the write access of all registers are suspended except for IQUECR. QRAM directly accessed by CPU is allowed even the module is disabled.

IQUEEN — IQUE Module Enable

This read/write control bit determines whether the IQUE module is enabled. When this bit change from zero to one, all registers in the module will be reset.

1 = IQUE module is enabled.
0 = IQUE module is disabled.

IQUERST — IQUE Module Reset

This read/write control bit reset all registers in the module.

1 = All registers in IQUE module reset in next clock cycle.
0 = Unaffected.

QC12DBE — QUE channel 1+2 block base single or double buffer mode

This read/write control bit determines whether the QUE channel 1+2 operates in block base single or double buffer mode.

1 = Enable block base buffering mode.
0 = Circular buffer mode.

QC34DBE — QUE channel 3+4 block base single or double buffer mode

This read/write control bit determines whether the QUE channel 3+4 operates in block base single or double buffer mode.

1 = Enable block base buffering mode.
0 = Circular buffer mode.

3.1.2 Queue Channel n FIFO Data Port Registers (QC1DR, QC2DR, QC3DR, QC4DR)

The Queue Channel n FIFO Data Port Registers provides access information in the QRAM buffers in FIFO fashion. When a channel is configured as a receive queue, data is read out from the QRAM through the QUE Channel n FIFO Data Port Register. The current value of data port register is reflecting the data content of QRAM pointed by the current pointer register. After any read/write operation of data port register, pointer value will increment and update the content of data port register. When a queue is configured as a transmit queue, data is written into the QRAM through the Queue Channel n FIFO Data Port Register.

It is normally considered an error to write to the data register of a receive queue or to read the data port register of a transmit queue. These actions could interfere with normal operation of internal FIFO pointers. It is also possible to access the data in the QRAM buffer through reads and writes to 64K bytes addresses in the CPU memory map without affecting FIFO pointers.

Address Offset: $02 (QC2DR@$0C, QC3DR@$16, and QC4DR@$20)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	QnDATA[1]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:

1. When the corresponding queue channel is configured for 8-bit queue transfers, the 8-bit queue data is located in the higher order 8 bits of this register and the lower order 8-bits are unused

Figure 3-2 Queue Channel n FIFO Data Port Register (QCnDR)

When a queue channel is configured for 16-bit queue transfers, it always use 16-bit accesses to read or write the data port registers. If two bytes accesses are used to access data in a 16-bit wide queue channel, FIFO pointers can be updated inappropriately between these accesses. When the corresponding queue channel is configured for 8-bit queue transfers, the 8-bit queue data is located in the higher order 8 bits of this register and the lower order 8 bits are unused.

The values read from these register locations are not meaningful after reset because the FIFOs contain no meaningful information at that time. Accesses to these register locations are redirected to QRAM locations and reset does not affect the contents of any QRAM locations.

3.1.3 Queue Channel n Begin Pointer (QC1BP, QC2BP, QC3BP, QC4BP)

The Queue Channel Begin Pointer registers access the FIFO begin pointer for each queue channel. In the case of an 8-bit queue, each time a byte of data is read out of a queue channel FIFO (due to an automatic transfer from a transmit queue or due to a CPU read of QCnDR), the begin pointer is incremented by one. In the case of a 16-bit queue, each time a word of data is read out of a queue channel FIFO, the begin pointer is incremented by two. When the queue channel begin pointer equals the queue channel end pointer, the FIFO is empty and so the begin pointer cannot be advanced.

Reset forces all begin and end pointers to point at the start of the QRAM. Other conditions, such as changing the QCnSZB register, also force corresponding begin and end pointers to be reset, but in those cases the pointers are reset to the start of the buffer which is determined from the values in QnSML bit in QCnCR and QBASE field in QCnSZB. When QnSML=0, the base address of a queue buffer is formed by using the current value of the QBASE field in QCnSZB for bits 11 through 8, and forcing bits 7 through 0 to zero. When QnSML=1, the base address of a queue buffer is formed by using the current value of the QBASE field in QCnSZB for bits 7 through 4, and forcing bits 11 through 8 and bits 3 through 0 to zero. Bits 15 through 12 of the address are determined by controls the QRAM within the 64K bytes CPU memory map.

Address Offset: $04 (QC2BP@$0E, QC3BP@$18, and QC4BP@$22)

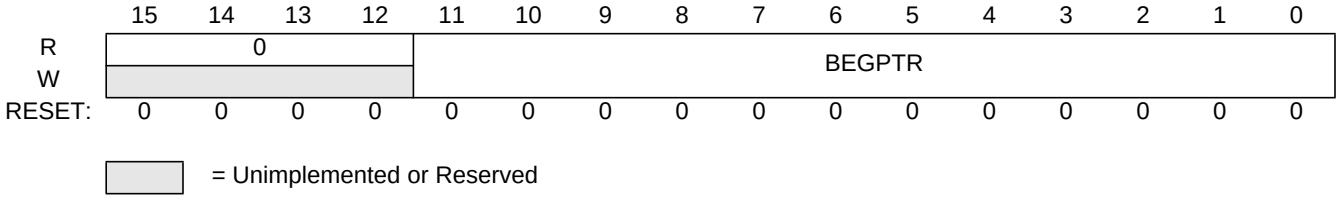


Figure 3-3 Queue Channel n Begin Pointer (QCnBP)

BEGPTR — Begin Pointer for Queue Channel n FIFO

This 12-bit field specifies the next address where data would be read out of a queue channel n FIFO. The upper four bits of the address in the 64K bytes address space are determined by the location of the 4K bytes QRAM within the MCU memory map.

If the increment operation would advance the pointer beyond the end of the space allocated to that queue channel, the pointer is reset to the start of the buffer (the FIFO buffers are circular). If the begin pointer and end pointer are equal before an increment operation, it indicates that the FIFO is empty and so the begin pointer cannot be advanced.

When Passthrough mode is selected, QCnBP registers for even numbered channels are ignored and all queue read transfers use QCnBP for the odd numbered channel in the passthrough channel pair. For example if Q1THRU in QC1CR is set, channels 1 and 2 act as a pair and QC2BP is ignored while QC1BP acts as the begin pointer for both channels 1 and 2 in the passthrough pair.

3.1.4 Queue Channel n End Pointer (QC1EP, QC2EP, QC3EP, QC4EP)

The Queue Channel End Pointer accesses the FIFO end pointer for each queue channel. In the case of an 8-bit queue, each time a byte of data is written into a queue channel FIFO (due to an automatic transfer to a receive queue or due to a CPU write to QCnDR), the end pointer is incremented by one. In the case of a 16-bit queue, each time a word of data is written into a queue channel FIFO, the end pointer is incremented by two. When the queue channel end pointer would equal the queue channel begin pointer after the increment, the FIFO is full and so no new data can be written into the FIFO and the end pointer is not incremented.

Reset forces all begin and end pointers to point at the start of the QRAM. Other conditions such as changing the QCnSZB register also force corresponding begin and end pointers to be reset, but in those cases the pointers are reset to the start of the buffer which is determined from the values in QnSML in QCnCR and QBASE in QCnSZB. When QnSML=0, the base address of a queue buffer is formed by using the current value of the QBASE field in QCnSZB for bits 11 through 8, and forcing bits 7 through 0 to zero. When QnSML=1, the base address of a queue buffer is formed by using the current value of the QBASE field in QCnSZB for bits 7 through 4, and forcing bits 11 through 8 and bits 3 through 0 to zero.

Address Offset: $06 (QC2EP@$10, QC3EP@$1A, and QC4EP@$24)

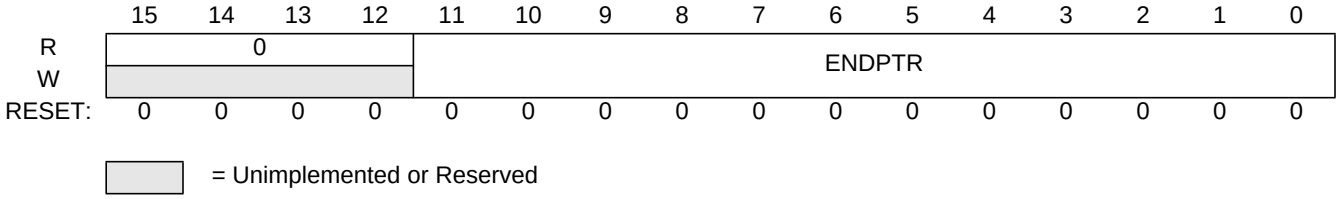


Figure 3-4 Queue Channel n End Pointer (QCnEP)

ENDPTR — End Pointer for Queue Channel n FIFO

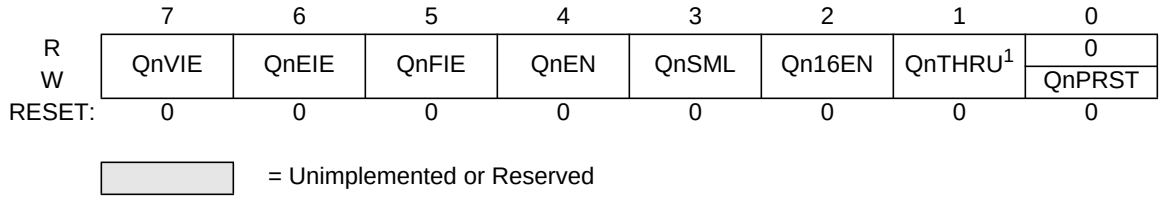
This 12-bit field specifies the next address where data would be written into a queue channel n FIFO. The upper four bits of the address in the 64K bytes address space are determined by the location of the 4K bytes QRAM within the MCU memory map.

If the increment operation would advance the pointer beyond the end of the space allocated to that queue channel, the pointer is reset to the start of the buffer (the FIFO buffers are circular). If the begin pointer and end pointer would be equal after an increment operation, it indicates that the FIFO is full and so no new data can be written into the FIFO and no increment takes place.

When Passthrough mode is selected, QCnEP registers for even numbered channels are ignored and all queue write transfers use QCnEP for the odd numbered channel in the passthrough channel pair. For example if Q1THRU in QC1CR is set, channels 1 and 2 act as a pair and QC2EP is ignored while QC1EP acts as the end pointer for both channels 1 and 2 in the passthrough pair.

3.1.5 Queue Channel n Control Register (QC1CR, QC2CR, QC3CR, QC4CR)

Address Offset: $08 (QC2CR@$12, QC3CR@$1C, and QC4CR@$26)



NOTES:
1. In QCnCR for odd-numbered channels this bit position is used for the QnTHRU bit. In even-numbered channels this bit position is unimplemented and always reads zero.

Figure 3-5 Queue Channel n Control Register (QCnCR)

QnVIE — Queue Channel n Valid Interrupt Enable

This read/write control bit determines whether the QnVF status flag of QCnSR causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when QnEN=QnVIF=1.
0 = Hardware interrupt requests from QnVIF are disabled.

QnEIE — Queue Channel n Empty Interrupt Enable

This read/write control bit determines whether the QnEIF status flag of QCnSR causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when QnEN=1 and QnEIF=1.
0 = Hardware interrupt requests from QnEIF are disabled.

QnFIE — Queue Channel n Full Interrupt Enable

This read/write control bit determines whether the QnFIF status flag of QCnSR causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when QnEN=QnFIF=1.
0 = Hardware interrupt requests from QnFIF are disabled.

QnEN — Queue Channel n Enable

The queue channel enable controls whether interrupts can be generated and whether requests from peripheral modules trigger automatic data transfers into or out of the QRAM. CPU reads or writes to the QCnDR still cause data to be transferred out of and into the QRAM in FIFO fashion so the QRAM associated with a queue channel can be used as a software FIFO.

Disabling QnEN does not affect the begin and end pointers for the associated queue channel so these pointers can still be used after a channel has been disabled.

Changing QnSML, QnTHRU, QnEN, or Qn16EN while a queue channel is enabled or a FIFO is operating is ambiguous so it should be considered illegal. This would confuse the logic controlling pointer updates and might lead to unexpected operation.

1 = Queue channel n is enabled.
0 = Queue channel n is disabled.

QnSML — Queue Channel n Small Buffer Select

This read/write control bit determines the resolution of the QSIZE setting for channel n. Any change to the value of QnSML causes the begin and end pointers (QCnBP and QCnEP) for the associated queue channel to be reset to point at the base address for that queue. The base address for each queue buffer is determined by QnSML and QCnSZB.

1 = Queue channel n is configured for small buffer size (16 to 256 bytes in steps of 16 bytes as set by the QSIZE field in QCnSZB).
0 = Queue channel n is configured for large buffer size (256 to 4K bytes in steps of 256 bytes as set by the QSIZE field in QCnSZB).

Qn16EN — Configure Channel n for 16-bit/8-bit Transfers

This read/write control bit determines the width of queue data transfers for queue channel n. The width selected determines whether the begin and end pointers get incremented by two or one for each queue transfer. Any change to the value of Qn16EN causes the begin and end pointers (QCnBP and QCnEP) for the associated queue channel to be reset to point at the base address for that queue.

1 = 16-bit transfers.
0 = 8-bit transfers.

In 8-bit transfer mode, data aligns to the most significant byte of the data bus (i.e. bit 8 to bit 15)

QnPRST — Queue Channel n Pointers Reset

When QnRST=1, it causes the begin, end pointers (QCnBP and QCnEP) and status registers (QCnSR) for the associated queue channel to be reset to point at the base address for that queue.

1 = Reset both Begin, End Pointer and Control Status Registers.

0 = Unaffected.

Q1THRU (Q3THRU) — Select Passthrough Mode Involving Queue Channel 1+2 (3+4)

These read/write bits are used to select a passthrough mode where two queue channels share a single set of begin and end pointers. In passthrough mode, data is automatically entered into the queue buffer in response to requests from one peripheral module and data is read out of the same queue buffer in response to requests from another peripheral module. In effect, once the queue module and peripherals have been setup, this causes data to be passed directly from one peripheral module to another through the queue FIFO buffer without any CPU intervention.

1 = Channel 1 (3) is configured for passthrough to/from channel 2 (4).

0 = No passthrough mode is selected.

3.1.6 Queue Status Register (QCnSR)

Address Offset: $09 (QC2SR@$13, QC3SR@$1D, QC4SR@$27)

	7	6	5	4	3	2	1	0
R	QnVSF	QnFSF	QnESF	QnVIF	QnFIF	QnEIF		
W				0	0	0		
RESET:	0	0	1	0	0	1	0	0

= Unimplemented or Reserved

Figure 3-6 Queue Channel n Status Register (QCnSR)

This register consists of queue channel status and interrupt flags. The status flags indicate when each queue channel has valid data, is empty or full. These flags become set and latched when the corresponding condition first becomes true. The status flags can be forced to 1 by writing a 1 to the QnVS, QnFS and QnES bits. So that data can be read or write again into the same buffer location.

The interrupt flags also become set and latched when the corresponding condition first becomes true. But these flags can only be cleared by writing 0 to flag bit position for interrupt acknowledgement.

QnVSF — Queue Channel n Valid Data Control and Status Flag

1 = Set when queue channel n FIFO has at least one valid entry.

0 = Queue channel n FIFO is empty.

QnFSF — Queue Channel n Full Status Flag

1 = Set when queue channel n FIFO becomes full.

0 = Queue channel n FIFO is not full.

QnESF — Queue Channel n Empty Status Flag

The reset value of this bit is 1. After module reset or queue channel pointer reset, the Empty status bit is set.

1 = Set when queue channel n FIFO becomes empty.

0 = Queue channel n FIFO is not empty.

QnVIF — Queue Channel n Valid Data Interrupt Flag

This Interrupt Flag can be cleared only. Interrupt will not be generated unless the QnVIE bit in QCnCR register is enabled.

1 = Set when queue channel n FIFO has at least one valid entry.
0 = Queue channel n FIFO is empty.

QnFIF — Queue Channel n Full Interrupt Flag

This Interrupt Flag can be cleared only. Interrupt will not be generated unless the QnFIE bit in QCnCR register is enabled.

1 = Set when queue channel n FIFO becomes full.
0 = Queue channel n FIFO is not full.

QnEIF — Queue Channel n Empty Interrupt Flag

This Interrupt Flag can be cleared only. The reset value of this bit is 1. After module reset or queue channel pointer reset, the Empty status bit is set. Interrupt will not be generated unless the QnEIE bit in QCnCR register is enabled.

1 = Set when queue channel n FIFO becomes empty.
0 = Queue channel n FIFO is not empty.

3.1.7 Buffer Size/Base Address (QC1SZB, QC2SZB, QC3SZB, QC4SZB)

These four read/write registers specify the size and base address for the RAM buffers associated with each queue channel. The queue n small buffer select(QnSML) control bit in QCnCR determines whether QSIZE specifies the number of 16-byte blocks or 256-byte blocks that are associated with queue channel n and determines how the QBASE field is interpreted. When QnSML=0, QSIZE determines the number of 256-byte blocks are assigned to queue channel n and QBASE determines one of 16 possible starting points for the queue channel n FIFO buffer in the 4K bytes QRAM. When QnSML=1, QSIZE determines the number of 16-byte blocks are assigned to queue channel n and QBASE determines one of 16 possible starting points for the queue channel n FIFO buffer in the first 256-byte block of the 4K bytes QRAM.

Any write to QCnSZB causes the begin and end pointers (QCnBP and QCnEP) for the associated queue channel to be reset to point at the base address for that queue. The base address for each queue buffer is determined by QnSML and QCnSZB. When passthrough mode is selected, the size and base address for the QRAM buffer is specified by the first (odd numbered) QCnSZB register and the second (even numbered) QCnSZB register of the passthrough pair is ignored.

The user is responsible for making sure that the base address plus the size does not cause the FIFO buffer to extend past the base address of the QRAM plus 4K bytes. Failure to follow this guideline can cause the FIFO buffer to be mapped to unexpected locations.

Address Offset: $0A (QC2SZB@$14, QC3SZB@$1E, and QC4SZB@$28)

	7	6	5	4	3	2	1	0
R W	QSIZE				QBASE			
RESET:	0	0	0	0	0	0	0	0

Figure 3-7 Buffer Size/Base Address Register (QCnSZB)

QSIZE — Queue Channel n Buffer Size

When QnSML=0, this 4-bit field specifies the number of 256-byte blocks of QRAM that are assigned to the associated queue channel. 0000 corresponds to 1 block of 256 bytes and 1111 corresponds to 16 blocks of 256 bytes (or 4K bytes).

When QnSML=1, this 4-bit field specifies the number of 16-byte blocks of QRAM that are assigned to the associated queue channel. 0000 corresponds to 1 block of 16 bytes and 1111 corresponds to 16 blocks of 16 bytes (or 256 bytes).

QBASE — Queue Channel n Buffer Base Address

This 4-bit field specifies the base (starting) address of the QRAM buffer that is assigned to the associated queue channel. When QnSML=0, these four bits effectively become address bits 8 through 11 of the address used to access the QRAM during FIFO data transfers. When QnSML=1, the address used to access the QRAM during FIFO data transfers has address bits 8 through 11 forced to zero and the four QBASE bits effectively become address bits 4 through 7.

3.1.8 Queue Channel Request Mapping Registers (QCnREQ)

The QCnREQ registers are used to specify which peripheral is associated with each queue channel. Selecting a request source also controls routing of the acknowledge signal from the queue channel to the selected peripheral and the direction of queue transfers is implied. If there are simultaneous requests from more than one slow peripheral, queue channel 1 has the highest priority and channel 4 has the lowest priority.

Address Offset: $0B (QC2REQ@$15, QC3REQ@$1F, QC4REQ@$29)

	7	6	5	4	3	2	1	0
R W					QnREQ			
RESET:	0	0	0	0	1	1	1	1

= Unimplemented or Reserved

Figure 3-8 Queue Channel n Request Mapping Register (QCnREQ)

QnREQ — Select Peripheral Request for Queue Channel n

These bits specify which peripheral function is associated with queue channel n. Depending on the implementation of a specific device derivative, one or more of the high order bits of this field may be eliminated. Any such unused bits will become unimplemented bits that always read zero.

Selecting a peripheral function causes the queue system logic to logically connect the channel request and acknowledge signals to the selected peripheral and sets the direction for queue transfers for this queue channel. *(These settings are determined at the time of chip integration so this column would be filled in at that time.)*

Table 3-2 Queue Channel n Request Mapping

QnREQ	Peripheral Function	Direction (Rx/Tx)
0000	source0[1]	Rx
0001	source1[1]	Tx
0010	source2[1]	Rx
0011	source3[1]	Tx
0100	source4[1]	Rx
0101	source5[1]	Tx
0110	source6[1]	Rx
0111	source7[1]	Tx
1000	source8[1]	Rx
1001	source9[1]	Tx
1010	source10[1]	Rx
1011	source11[1]	Tx
1111	Disabled	Disabled

NOTES:

1. During definition of a specific derivative, peripheral request and acknowledge signals are connected to module inputs and outputs. The setting in QnREQ determines which peripheral is associated with each queue channel. See the associated device user's guide.

3.1.9 Que Channel Double Buffer Control Register (QC12DCR, QC34DCR)

The Que Channel Double Buffer Control Register is activated only when transferring data in block base style. Unlike Circular buffering, the data read/write in Block Base data transfer mode depends on the empty and full status of the entire block. For example, data can only be read out from the buffer once the buffer is full, or data can only be written into the buffer once the buffer is empty. Double Buffer mode enable bit (QC12DBE or QC34DBE) of IQUECR must be set before using this block base single or double buffering feature.

There are two options in Block Base buffering transfer mode, they are Single and Double Block Transfer mode. In single block transfer mode, data can be read out from the buffer once the whole buffer is full. And data can only be written into the buffer once the buffer is empty. So either read or write operation is allowed during the same period.

In double block transfer mode, data written into and read out from IQUE buffer in PING-PONG buffering style. Data read and write operations are allowed at the same time. Once the first buffer is full, data can be written into the second buffer continuously while data can be read out from the first buffer at the same time.

Address Offset: $2A (QC34DCR@$2B)

	7	6	5	4	3	2	1	0
R				SBTE	DBTIE	DBEIE	DBFIE	0
W								DBRST
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-9 Queue Channel 1+2 Double Buffer Control Register (QC12DCR)

SBTE — Single Bock Transfer mode enable

This read/write control bit determines whether the IQUE operates in block based transfer for a single block data transfer. When switching from Double buffer mode to Single buffer, Status and Pointer registers of the associated channel will be reset.

1 = Enable Single Block Transfer mode.
0 = Disable Single Block Transfer mode.

DBTIE — Double Buffer Block Transfer Complete Interrupt Enable

This read/write control bit determines whether the DBTCIF status flag of causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when DBTRANxx is set.
0 = Hardware interrupt requests from DBCTIF are disabled.

DBEIE — Queue Channel Double Buffer Empty Interrupt Enable

This read/write control bit determines whether the DBEIF status flag of causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when both buffers are empty. (i.e. DBEIF = 1)
0 = Hardware interrupt requests from DBEIF are disabled.

DBFIE — Queue Channel Double Buffer Full Interrupt Enable

This read/write control bit determines whether the DBFIF status flag of causes hardware interrupt requests to be generated when the flag is set.

1 = Hardware interrupt requested when both buffers are full. (i.e. DBFIF = 1)
0 = Hardware interrupt requests from DBFIF are disabled.

DBRST — Double Buffer Status and Pointer Register Reset

When DBRST=1, it causes QC12DSR (or QC34DSR) registers to be reset, and the begin and end pointers (QCnBP and QCnEP) for the associated queue channel to be reset to point at the base address for that queue. This bit has to be written a one to enable the reset function.

1 = Reset both Begin, End Pointer and Status registers.
0 = Unaffected.

3.1.10 Que Channel Double Buffer Status Register (QC12DSR, QC34DSR)

This Double Buffer Status register indicates the status of buffers when Double Buffer is enabled. Otherwise, the content of this register will not be changed.

Address Offset: $2C (QC34DSR@$2D)

	7	6	5	4	3	2	1	0
R	DBTCIF	DBEIF	DBFIF				DBSF	
W								
RESET:	0	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-10 Queue Channel 1+2 Double Buffer Status Register (QC12DSR)

DBTCIF — Block Transfer Complete Interrupt Flag

This flag indicates whether the number of buffers to be transfered specified in DBTRANxx field of QCDCT register has completed or not. This flag can be cleared by writing a zero to this bit.

1 = Set when DBTRAN12 (or DBTRAN34) field of QCDCT register = $0000.
0 = Block Transfer Complete not complete.

DBEIF — Double Buffer Empty Interrupt Flag

This Interrupt Flag can be cleared by user or reset. Interrupt will not be generated unless the DBEIE bit in QC12DCR (or QC34DCR) register is enabled. This flag can be cleared by writing a one to this bit.

1 = Set when both buffers are empty.
0 = At lease one buffer is full.

DBFIF — Double Buffer Full Interrupt Flag

This Interrupt Flag can be cleared by user or reset. Interrupt will not be generated unless the DBFIE bit in QC12DCR (or QC34DCR) register is enabled. This flag can be cleared by writing a zero to this bit.

1 = Set when at lease one buffer is full.
0 = Both buffers are empty.

DBSF — Double Buffer Status Flag

This read only buffer status flag indicates the full or empty status of the buffers.

00 = Both buffers are empty.
01 = One buffer is empty, one buffer is full.
10 = Both buffers are full.
11 = reserved

3.1.11 Que Channel Double Buffer Counter Register (QCDCT)

Address Offset: $2E

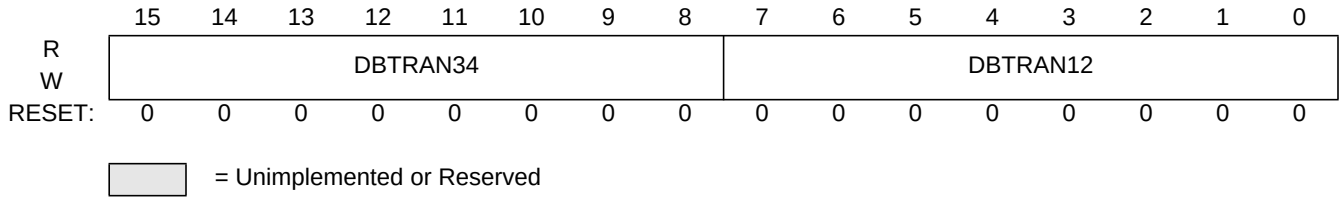


Figure 3-11 Queue Channel Double Buffer Counter Register (QCDCT)

DBTRAN12 — Number of buffers to be transfered for Queue Channel 1+2

This 8-bits read/write register specifies the number buffers to be transfered. The register will count down to zero when transfer complete.

DBTRAN34 — Number of buffers to be transfered for Queue Channel 3+4

This 8-bits read/write register specifies the number buffers to be transfered. The register will count down to zero when transfer complete.

3.1.12 Que Channel Double Buffer Software Handshake Register (QC12DSHR, QC34DSHR)

This Double Buffer Software Handshake register controls the handshake signaling in Double Buffer mode through software. When the Software Handshake control mode of transmit or receive que channel is enabled. Data being transmitted or received have to be acknowledged or qualified to be valid data before changing the status of buffer. Also, data can be flushed or resend before data are qualified to be valid data. When the software handshake mode is disabled. Buffer status registers automatically updated once the status buffer status has changed.

Address Offset: $30 (QC34DSHR@$31)

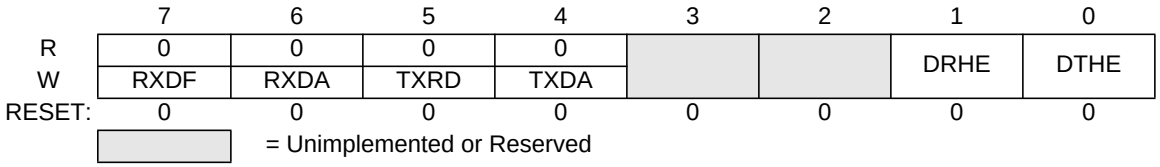


Figure 3-12 Queue Channel 1+2 Double Buffer Software Handshake Register (QC12DSHR)

DTHE — Queue Channel Double Buffer Transmit Handshake Enable

This control bit determines whether Transmitted Data Acknowledge/Resend Transmitted Data handshake control protocol is enable.

1 = Double Buffer Transmit handshake protocol is enabled.

0 = Double Buffer Transmit handshake protocol is disabled.

DRHE — Queue Channel Double Buffer Receive Handshake Enable

This control bit determines whether Received Data Flush/Received Data Valid handshake control protocol is enable.

1 = Double Buffer Receive handshake protocol is enabled.
0 = Double Buffer Receive handshake protocol is disabled.

TXDA — Transmitted Data Acknowledge

This control bit acknowledges the data transmitted from queue channel is consumed. Buffer status registers advance when this bit asserts. This bit will be deserted in next cycle automatically. Transmitted data of last buffer will be flushed and overrided by next write cycle.

1 = Transmitted Data Acknowledged.
0 = Transmitted data of last buffer accessed cannot be overrided and DBFIF will not be set.

TXRD — Resend Transmitted Data

This control bit requests resend the data of last buffer accessed. Buffer status register will keep unchanged and the pointer register will be reset for resending data. If TXDA bit is set, data will be flushed and resend operation is not allowed.

1 = Request resend data of last buffer accessed.
0 = Unaffected.

RXDA — Received Data Acknowledge

This control bit acknowledges the data of last buffer accessed received by queue channel to qualify the received data is valid. Buffer and pointer status advance when this bit asserts. Data is ready for transmit.

1 = Received Data is valid, advance buffer status.
0 = Received data of last buffer accessed cannot be overrided.

RXDF — Received Data Flush

Setting of this control bit discards the data of last buffer accessed received by queue channel.

1 = Received Data Flush. Discard buffer data of last buffer accessed. Data will be resent.
0 = Received data of last buffer accessed cannot be overrided.

Section 4 Functional Description

The IQUE module includes four independent queue channels that are independently programmable for 8-bit or 16-bit transfers. Up to 4K bytes of static RAM is used for FIFO buffers for queue channels. Each queue channel controls the transfer of data bytes or 16-bit words between an MCU on-chip peripheral and the FIFO buffer. Pair of queue channels can be configured for passthrough mode where data is effectively passed from one peripheral to another with the FIFO buffer acting as an elastic buffer between the peripherals.

Peripheral requests are each wired to a request input to the IQUE module. Inside the module these request inputs are wired to one or more channel request input multiplexer. Depending upon the number of potential request sources that could be associated with each queue channel in a specific derivative MCU. At run time a user writes to QnREQ of Request Mapping register fields to specify which peripheral request source is connected to each queue channel. Peripheral sources are typically wired to the channel request input multiplexer of all queue channels unless there are more than 16 request sources. The direction of transfers is implied based on which peripheral function is selected. Selecting a request source also causes the queue channel acknowledge signal to be logically routed to the same peripheral function. The acknowledge signal acknowledges the peripheral for data transfer request and indicates the completion of data transfer.

In receive queue mode, data is automatically written into the FIFO buffer in response to a request from a peripheral and it causes the FIFO end pointer to be incremented. Data is read out of a receive queue by a CPU read of the FIFO data port register and it causes FIFO begin pointer to be incremented. If the source with Rx direction is assigned, channel is enabled and passthrough mode is disabled, IQUE will operate in receive mode automatically. FIFO pointers will not be affected even by a CPU write of the FIFO data port register.

Similarly in transmit queue mode, data is automatically written into the FIFO buffer by a CPU write of the FIFO data port register and it causes the FIFO end pointer to be incremented. Data is read out of the FIFO buffer in response to a request from a peripheral and it causes FIFO begin pointer to be incremented. If source with Tx direction is assigned, channel is enabled and passthrough mode is disabled, IQUE will operate in transmit mode automatically. FIFO pointers will not be affected even by a CPU read out of the FIFO data port register.

In both receive and transmit queue modes, half of the total data transfer bandwidth is assigned to CPU access through data port registers, and one of the channels with the highest priority occupies rest half of the bandwidth even there are more than one requests from peripherals simultaneously. If CPU do not request for data transfer, all data transfer bandwidth will be assigned to two channels associated with peripherals. CPU has higher priority than peripheral so that it can access QRAM FIFO buffer through data port register read/write anytime by suspending one channel which associated with peripheral.

In passthrough mode, channel 1 (or 3) can be linked to channel 2 (or 4). One channel acts as a receive queue while the linked channel acts as a transmit queue. Data received by one channel is automatically transmitted on the linked channel. As long as the receive queue is not full, a new data transfer is triggered each time that channel’s request signal is active. Whenever there is any data in the queue, a new transfer out of the FIFO buffer is triggered each time the linked channel’s request signal is active. The FIFO buffer acts as an elastic buffer between the receive peripheral and the transmit peripheral. In this mode, the size and base address of the FIFO buffer is determined by controls for the odd-numbered channel of the passthrough pair and all control and status for the even-numbered channel is ignored except the setting in Queue Request Mapping register that selects the peripheral function associated with the even-numbered channel of the passthrough pair. The two channels of each passthrough pair are not allowed to be configured as the same data transfer direction (ie. both Rx or both Tx). Otherwise, requests from even-numbered channels are ignored.

In the passthrough mode, requests from CPU through data port register read/write are ignored. CPU read/write the data port register cannot change the value of FIFO pointer and the content of data port registers. However, CPU still can access the QRAM through S12 EEPROM bus interface. The bandwidth

guaranteed channel (channel 1) is provided to allow guaranteed data transfer bandwidth for either one of the channels without degraded by this kind of CPU's QRAM access. If there are simultaneous requests from all four channels, channel 1 has the highest priority to be served.

For example, in a system that has four RAM access time slots per CPU bus cycle (numbered 1, 2, 3, 4), slots 1 and 3 are available for this fast peripheral channel. That means the data transfer rate with using these channels are guaranteed to be 60M byte/s when the IQUE module core clock is equal to 60MHz. If the CPU requests access to the QRAM through S12 bus, slot 2 becomes available for another channel and the CPU uses slot 4 for the QRAM access. If the CPU is not accessing QRAM, both slot 2 and 4 can be used by other channels.

Similar to Passthrough mode, IQUE module supports two pairs of channel in double buffer mode. The actual total buffer size of the two buffers for each channel pair is equal to twice the size specified in the QSIZE field of QCnSZB register. The buffer to be accessed is toggled automatically. The number of buffers to be transferred is programmable in the DBTRAN12 and DBTRAN34 field of QCDCT register for channel 1+2 and 3+4 respectively. The buffer empty and full status information is shown in the control and status registers QC1SR and QC3SR for channel 1+2 and 3+4 respectively. Unlike normal Passthrough mode, this double buffer mode is operating in linear ping-pong buffering instead of circular buffering style. So both of the start and end address used to determine the empty and full status of the buffer is fixed. Unlike circular buffer, in double buffer mode, two buffers are accessed in buffer base, i.e. buffer empty, valid and full is determined based on one buffer. Buffer can only be read after the entire buffer is full, or written after entire buffer is empty. After one buffer is full, pointer will automatically switch to next buffer for further writting, provided that the next buffer is empty.

4.1 QUE Integration Module

QUE Integration Module (QIM) is an interface between QUE Controller, CPU and QRAM to allow accesses of QRAM in a time multiplexed fashion.

QIM will notify QUE Controller about the start of bus cycle to allow time slots allocation. Normally, for all time slots, QUE Controller provides control and address, and selects which channel will be the input to the QRAM. When there is CPU access to the QRAM, QIM will notify the QUE controller that the CPU requests access to the QRAM. QUE Controller will be selected to take control of the QRAM in slot 1,2 and 3, but CPU will be selected to access the QRAM in slot 4.

4.2 QUE Controller

QUE Controller manages the QRAM read and write control and data path switching in the QIM. It consists of a FSM and control status register sets for each channel. The FSM controls the RAM read/write timing and data path switching, and handles the handshake signals between IQUE and other queue request peripherals. All registers are accessible through IP bus interface.

QUE Controller also maintains FIFO begin and end pointers (QCnBP and QCnEP) as well as a Queue Channel n FIFO data register (QCnDR) for each queue channel. This allows the QRAM buffers to be accessed as circular or linear FIFOs. The QnSML bit in QCnCR and the QCnSZB register allow a user to

specify the size and base address for the FIFO buffer for each queue channel in the QRAM. QnSML (queue channel n small buffer select) chooses the resolution for setting the size of the FIFO buffer. When QnSML=1, the 4-bit QSIZE field in QCnSZB sets the number (1 to 16) of 16-byte blocks of QRAM to assign to the FIFO buffer for channel n and the 4-bit QBASE field sets the starting address of the buffer within the first 256 bytes of the queue RAM to one of 16 16-byte boundaries. When QnSML=0, QSIZE sets the number of 256-byte blocks of queue RAM to assign to the FIFO buffer for channel n and QBASE sets the starting address of the buffer within the 4K bytes QRAM to 1-of-16 256-byte boundaries.

Each queue channel has a FIFO valid data status flag (QnVF) and a FIFO full status flag (QnFF). Three other control bits (QnVIE, QnEIE and QnFIE) control whether these flags also cause hardware interrupt requests. After reset all of these flags are cleared and all local enable bits (QnVIE, QnEIE and QnFIE) are clear to disable hardware interrupt requests. Flags are set and latched when the associated condition becomes true. Flags are cleared by reading the status register while the flag is set and then writing a one to the flag that is to be cleared.

IQUE module can be disabled for power saving purpose by setting the IQUEEN bit in IQUECR register. All registers in the module remain unchanged when the module is disabled. They can be reset to default value by hardware reset or software reset through setting the IQUERST bit in IQUECR register. Also, individual Queue Channel pointer register can be reset by setting the QnRST bit in QCnCR register.

4.3 Bus Interface

There are three bus interface ports in IQUE module including IP bus interface, IP FIFO bus interface (ForestGreen-Line signals) and S12 Bus interface. All the ForestGreen-Line handshake and data signals are synchronized with the 60MHz IQUE clock.

Section 5 Initialization/Application Information

5.1 Initialization

After reset, the IQUE remains in an idle state, initialization of registers should be performed before any data transfer operations. All registers will be reset to the default values. An MCU RESET forces all register bits to be cleared. The begin and end FIFO pointers are cleared (which effectively forces the FIFOs to the empty condition). All of the interrupt flags and local enable bits are cleared to disable hardware interrupt requests.

5.2 Application Information

During operation of a typical application, user software will write to control registers in IQUE to decide which request source is connected to each QUE channel at any particular time. These assignments can also be changed during the course of executing an application program so that the IQUE could be used for one peripheral at one time and later the same queue channel might be used for other peripheral. This structure allows up to 16 request sources per queue channel but some requests could be mapped to more than one

queue channel. When every source is connected to all queue channels. This allows complete freedom to assign any combination of peripheral requests to any combination of queue channels.

Receive queue mode

For a receive queue that is not already full, data is automatically written into the FIFO buffer in response to a request from a peripheral. This automatic write also causes the FIFO end pointer (QCnEP) to be incremented. Data is read out of a receive queue by a CPU read of the FIFO data port register (QCnDR). This CPU read of QCnDR also causes the begin pointer (QCnBP) to be incremented. If the increment of either pointer would cause it to point past the last location in the buffer, it is reset to the starting address of the buffer (wraps around). The FIFO buffer is said to be empty if the begin pointer is equal to the end pointer. The FIFO buffer is full if the end pointer is one less than the begin pointer (or if the begin pointer is at the start of the buffer and the end pointer is at the end of the buffer). If the FIFO buffer is full, peripheral requests cannot write more data into the queue. If the FIFO buffer is empty, a CPU read of QCnDR returns the data from the buffer location pointed to by the end pointer but the begin pointer is not incremented.

Transmit queue mode

For a transmit queue, data is written into the FIFO buffer by a CPU write to QCnDR. This CPU write to FIFO data port register also causes the end pointer to be incremented. Data is automatically read out of the FIFO buffer in response to a peripheral request. This automatic read also causes the FIFO begin pointer to be incremented. If the increment of either pointer would cause it to point past the last location in the buffer, it is reset to the starting address of the buffer (wraps around). The FIFO buffer is said to be empty if the begin pointer is equal to the end pointer. The FIFO is full if the end pointer is one less than the begin pointer (or if the begin pointer is at the start of the buffer and the end pointer is at the end of the buffer). If the FIFO is full, a CPU write to QCnDR is ignored and the end pointer is not incremented. If the FIFO is empty, peripheral requests are ignored and no data transfer to the peripheral occurs.

Passthrough mode

When the passthrough mode bit is enabled (QnTHRU=1), pairs of queue channels are configured for a passthrough mode where data enters a queue FIFO RAM buffer in response to requests from one peripheral device and this data is automatically passed through to another peripheral device. In this mode, pairs of channels share a common block of memory and common begin and end pointers. This allows data to be passed without any CPU intervention from one peripheral to another using the queue FIFO RAM as an intermediate elastic buffer. CPU is not allowed to access the QRAM FIFO buffer through read/write data port registers in this mode. CPU can only access the QRAM through S12 EEPROM bus read/write operation.

Software FIFO mode

When a queue channel is disabled (QnEN=0), the QRAM buffer still functions as a FIFO so it can be used by a programmer as a software FIFO. In this mode, data is written into the FIFO by a CPU write to QCnDR and data is read from the FIFO by a CPU read of QCnDR. FIFO writes cause the end pointer to be incremented (wraps around if incrementing past the end of the buffer). FIFO reads cause the begin pointer to be incremented (wraps around if incrementing past the end of the buffer). A FIFO is said to be empty if the begin pointer is equal to the end pointer. The FIFO is full if the end pointer

is one less than the begin pointer (or if the begin pointer is at the start of the buffer and the end pointer is at the end of the buffer). If the FIFO is full, a CPU write to QCnDR is ignored and the end pointer is not incremented. If the FIFO is empty, a CPU read of QCnDR returns the data from the buffer location pointed to by the end pointer but the begin pointer is not incremented.

Programming Information

To use and initialise IQUE module, the following registers have to be programmed.

1. IQUE Module Control Register, IQUECR
2. Que Channel N Control Register, QCnCR
3. Que Channel N Size and Base Address Register, QCnSZB
4. Que Channel N Request Mapping Register, QCnREQ (For Passthrough, Tx Que, Rcv Que mode)

To access QRAM via CPU EE Bus, only need to enable IQUE module, no any other registers have to be programmed. CPU can also access via EE Bus any time in any operation mode.

A. Initialize IQUECR
- bit 0 - IQUEEN, module enable
- bit 1 - IQUERST, module reset
- bit 3 - QC12DBE, QC12 Block Base Transfer mode enable
- bit 4 - QC34DBE, QC34 Block Base Transfer mode enable

IQUEEN bit must be set to 1 to enable the module before read/write of other IQUE module registers.

B. Initialize QCnCR
- bit 0 - QnPRST, Channel n Pointer value reset
- bit 1 - QnTHRU, Channel n/n+1 Pass through mode enable
- bit 2 - Qn16EN, Channel n 16 bit transfer mode enable
- bit 3 - QnSML, Channel n small buffer size select
  - 1 = 16 to 256 bytes in steps of 16 bytes
  - 0 = 256 to 4k bytes in steps of 256 bytes
- bit 4 - QnEN, Channel n enable for Passthru, Tx and Rx que mode
  - Disable this bit for Software FIFO mode
- bit 5 - QnFIE, Channel n buffer full interrupt enable
- bit 6 - QnEIE, Channel n buffer empty interrupt enable
- bit 7 - QnVIE, Channel n buffer has valid data interrupt enable

Table 5-1 Example of operation mode selection

Operation mode		IQUECR	QC12DCR/ QC34DCR	QCnCR	Remarks
Circular Buffer	Software FIFO mode	0000 00X1	0000 0000	XXX0 XX00	-
	Pass through mode			XXX1 XX10	QnREQ set to Rx & Tx
	Rceive Que mode			XXX1 XX00	QnREQ set to Rx
	Transmit Que mode			XXX1 XX00	QnREQ set to Tx
Block Base (Single Buffer)	Software FIFO mode	0000 10X1 (for QC12) 0001 00X1 (for QC34)	0001 XXXX	XXX0 XX00	-
	Pass through mode			XXX1 XX10	QnREQ set to Rx & Tx
	Rceive Que mode			XXX1 XX00	QnREQ set to Rx
	Transmit Que mode			XXX1 XX00	QnREQ set to Tx
Block Base (Double Buffer)	Software FIFO mode	0000 10X1 (for QC12) 0001 00X1 (for QC34)	0000 XXXX	XXX0 XX00	-
	Pass through mode			XXX1 XX10	QnREQ set to Rx & Tx
	Rceive Que mode			XXX1 XX00	QnREQ set to Rx
	Transmit Que mode			XXX1 XX00	QnREQ set to Tx

C. Setting of buffer size and base address, QCnSZB

When QnSML(QCnCR) = 0:
Buffer base address = {QCnSZB[3:0], 0000, 0000}
Buffer size = 256 bytes x (QCnSZB[7:4] + 1)

When QnSML(QCnCR) = 1:
Buffer base address = {0000, QCnSZB[3:0], 0000}
Buffer size = 16 bytes x (QCnSZB[7:4] + 1)

D. Assign Tx/Rcv source modules

(for Passthrough, TxQue and RcvQue mode only)

Programming Examples:

Example 1: Setup Que Channel 1 & 2 in Passthrough mode
(i.e. Passthrough mode, 16 bit transfer, 256 byte block size, all interrupts disabled;
Buffer size of 256 bytes, base address = $0200;
Source 1 = Tx, Source 4 = Rx)

1. Enable IQUE module
Write 0000 0001 to IQUECR

2. Initialize QC1CR
   Write 0001 1110 to QC1CR

3. Set Buffer size and base address of QC1SZB
   Write 0001 0010 to QC1SZB

4. Assign IPF modules to request mapping register QC1REQ
   Write 0011 to QC1REQ;
   Write 1000 to QC2REQ

Example 2: Setup Que Channel 2 in Software FIFO mode:
(i.e. Passthru mode, 8 bit transfer, 256 byte block size, all interrupts enabled;
Buffer size of 256 bytes, base address = $0200)

1. Enable IQUE module
   Write 0000 0001 to IQUECR

2. Initialize QC2CR
   Write 0000 1000 to QC1CR

3. Set Buffer size and base address of QC2SZB
   Write 0001 0010 to QC2SZB

Example 3: Setup Que Channel 3 in Transmit Que mode:
(ie. Tx Que, 8 bit enable, 16 bytes block size, all interrupts enable;
Buffer size of 64 bytes, base address = $0010, Source 3 = Tx)

1. Enable IQUE module
   Write 0000 0001 to IQUECR

2. Initialize QC3CR
   Write 1111 1000 to QC3CR

3. Set Buffer size and base address of QC3SZB
   Write 0011 0001 to QC3SZB

4. Assign IPF module to request mapping register QC1REQ
   Write 0111 to QC3REQ

Index

–I–

Block Guide End Sheet

FINAL PAGE OF
38
PAGES