Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	CPU32+
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	Communications; CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10Mbps (1)
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	357-BBGA
Supplier Device Package	357-PBGA (25x25)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68360cvr25lr2

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

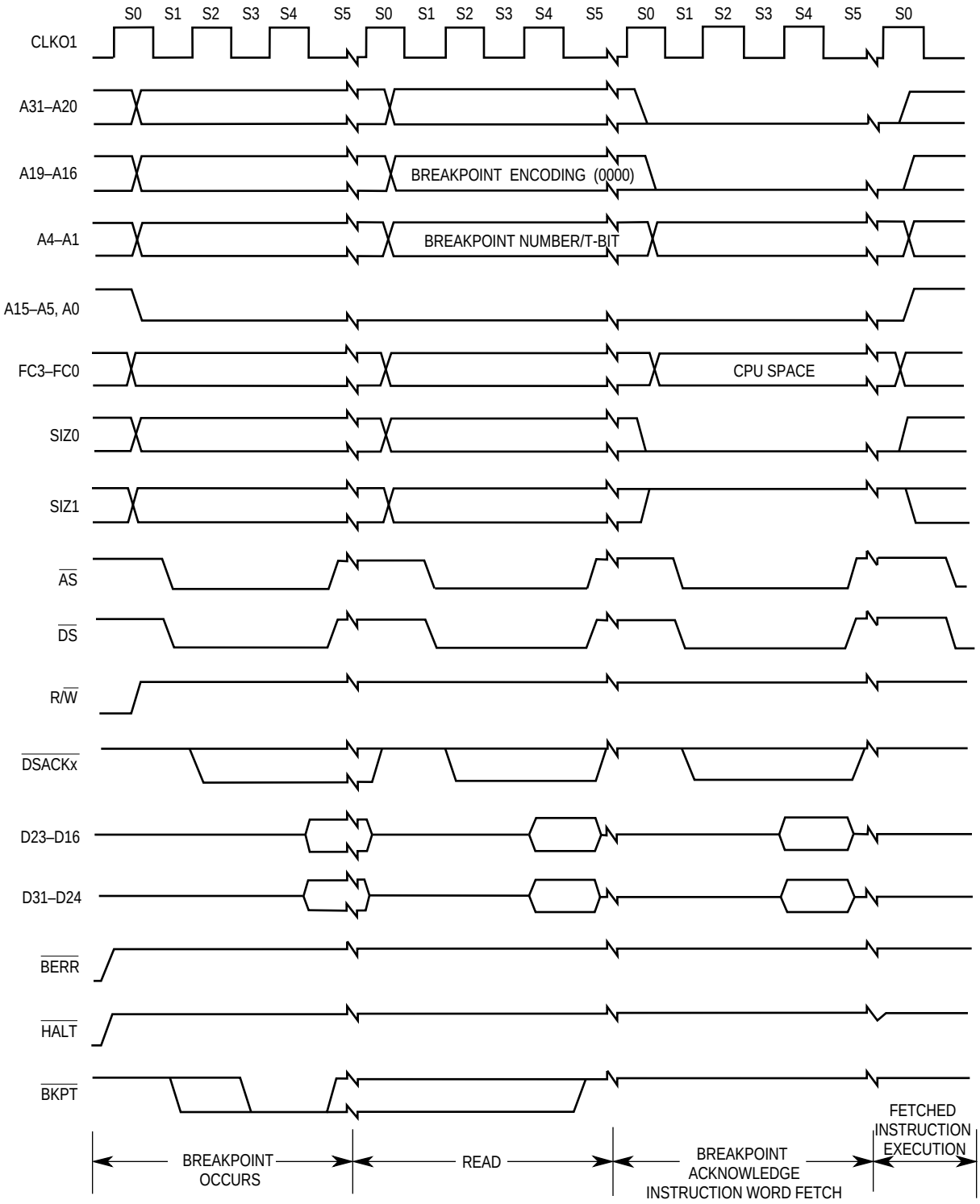


Figure 4-24. Breakpoint Acknowledge Cycle Timing (Opcode Returned)

Freescale Semiconductor, Inc.

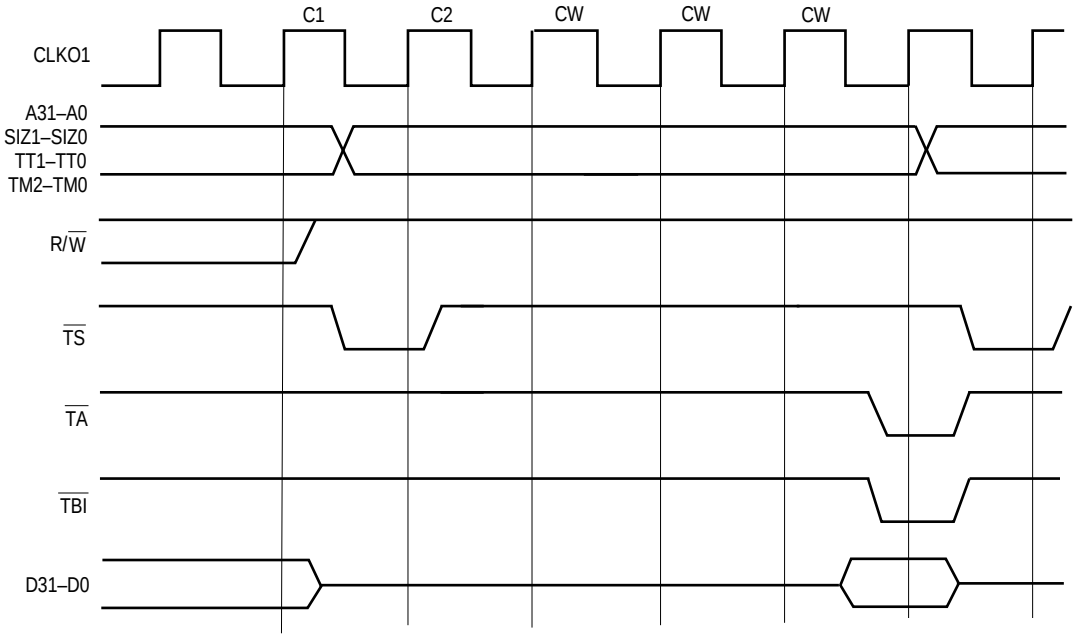


Figure 4-41. MC68EC040 Internal Registers Read Cycle

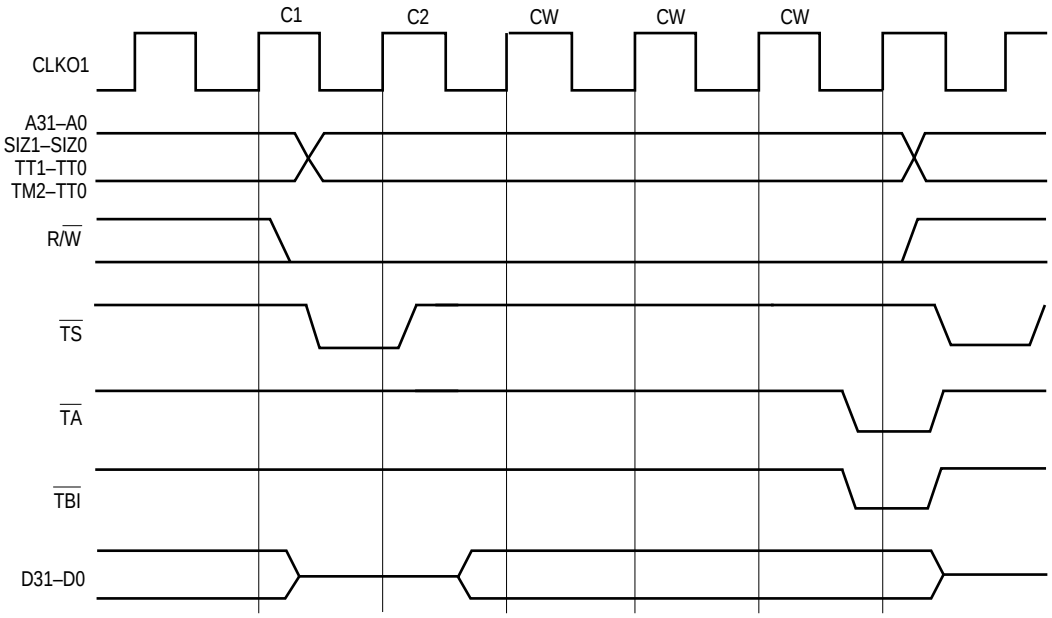


Figure 4-42. MC68EC040 Internal Registers Write Cycle

Freescale Semiconductor, Inc.

4.6.8 Show Cycles

The QUICC can perform data transfers with its internal modules without using the external bus, but when debugging, it is desirable to have address and data information appear on the external bus. These external bus cycles, called show cycles, are distinguished by the fact that $\overline{AS}$ is not asserted externally. $\overline{DS}$ is used to signal address strobe timing in show cycles.

After reset, show cycles are disabled and must be enabled by writing to the SHEN bits in the module configuration register. When show cycles are disabled, the address bus, function codes, size, and read/write signals continue to reflect internal bus activity. However, $\overline{AS}$ and $\overline{DS}$ are not asserted externally, and the external data bus remains in a high impedance state. When show cycles are enabled, $\overline{DS}$ indicates address strobe timing and the external data bus contains data. The following paragraphs are a state-by-state description of show cycles, and Figure 4-45 illustrates a show cycle timing diagram. Refer to Section 10 Electrical Characteristics for specific timing information.

State 0 – During state 0, the address and function codes become valid, R/$\overline{W}$ is driven to indicate a show read or write cycle, and the size pins indicate the number of bytes to transfer. During a read, the addressed peripheral is driving the data bus, and the user must take care to avoid bus conflicts.

State 41 – One-half clock cycle later, $\overline{DS}$ (rather than $\overline{AS}$) is asserted to indicate that address information is valid.

State 42– No action occurs in state 42. The bus controller remains in state 42 (wait states will be inserted) until the internal read cycle is complete.

State 43– When $\overline{DS}$ is negated, show data is valid on the next falling edge of the system clock. The external data bus drivers are enabled so that data becomes valid on the external bus as soon as it is available on the internal bus.

State 0 – The address, function codes, read/write, and size pins change to begin the next cycle. Data from the preceding cycle is valid through state 0.

Freescale Semiconductor, Inc.

Table 5-1. Instruction Set

Mnemonic	Description		Mnemonic	Description
ABCD	Add Decimal with Extend		MOVEA	Move Address
ADD	Add		MOVE CCR	Move Condition Code Register
ADDA	Add Address		MOVE SR	Move to/from Status Register
ADDI	Add Immediate		MOVE USP	Move User Stack Pointer
ADDQ	Add Quick		MOVEC	Move Control Register
AND	Logical AND		MOVEM	Move Multiple Registers
ANDI	Logical AND Immediate		MOVEP	Move Peripheral Data
ASL	Arithmetic Shift Left		MOVEQ	Move Quick
ASR	Arithmetic Shift Right		MOVES	Move Alternate Address Space
Bcc	Branch Conditionally (16 Tests)		MULS	Signed Multiply
BCHG	Bit Test and Change		MULU	Unsigned Multiply
BCLR	Bit Test and Clear		NBCD	Negate Decimal with Extend
BGND	Enter Background Mode		NEG	Negate
BKPT	Breakpoint		NEGX	Negate with Extend
BRA	Branch Always		NOP	No Operation
BSET	Bit Test and Set		NOT	Ones Complement
BSR	Branch to Subroutine		OR	Logical Inclusive OR
BTST	Bit Test		ORI	Logical Inclusive OR Immediate
CHK	Check Register against Bounds		PEA	Push Effective Address
CHK2	Check Register against Upper and		RESET	Reset External Devices
	Lower Bounds		ROL, ROR	Rotate Left and Right
CLR	Clear Operand		ROXL, ROXR	Rotate with Extend Left and Right
CMP	Compare		RTD	Return and Deallocate
CMPA	Compare Address		RTE	Return from Exception
CMPI	Compare Immediate		RTR	Return and Restore
CMPM	Compare Memory		RTS	Return from Subroutine
CMP2	Compare Register against Upper		SBCD	Subtract Decimal with Extend
	and Lower Bounds		Scc	Set Conditionally
DBcc	Test Condition, Decrement and		STOP	Stop
	Branch (16 Tests)		SUB	Subtract
DIVS, DIVSL	Signed Divide		SUBA	Subtract Address
DIVU, DIVUL	Unsigned Divide		SUBI	Subtract Immediate
EOR	Logical Exclusive OR		SUBQ	Subtract Quick
EORI	Logical Exclusive OR Immediate		SUBX	Subtract with Extend
EXG	Exchange Registers		SWAP	Swap Data Register Halves
EXT, EXTB	Sign Extend		TAS	Test and Set Operand
ILLEGAL	Take Illegal Instruction Trap		TBLS, TBLSN	Table Lookup and Interpolate,
JMP	Jump			Signed
JSR	Jump to Subroutine		TBLU, TBLUN	Table Lookup and Interpolate,
LEA	Load Effective Address			Unsigned
LINK	Link and Allocate		TRAPcc	Trap Conditionally (16 Tests)
LPSTOP	Low-Power Stop		TRAPV	Trap on Overflow
LSL, LSR	Logical Shift Left and Right		TST	Test
MOVE	Move		UNLK	Unlink

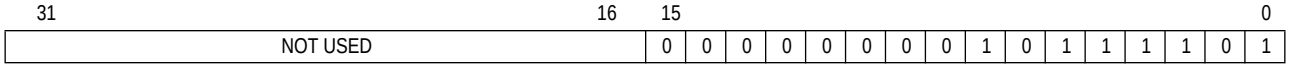
Freescale Semiconductor, Inc.

Table 5-15. T8-Bit Independent Variable Entries

X (Subroutine)	X (Instruction)	Y
0	0	0
1	256	16
2	512	32
3	768	48
4	1024	64
5	1280	80
6	1536	96
7	1792	112
8	2048	128
9	2304	112
10	2560	96
11	2816	80
12	3072	64
13	3328	48
14	3584	32
15	3840	16
16	4096	0

The first column is the value passed to the subroutine, the second column is the value expected by the table instruction, and the third column is the result returned by the subroutine.

The following value has been calculated for independent variable X:



Since X is an 8-bit value, the upper four bits are used as a table offset, and the lower four bits are used as an interpolation fraction. The following results are obtained from the subroutine:

Table Entry Offset ⇒ Dx [4:7] = $B = 11

Interpolation Fraction ⇒ Dx [0:3] = $D = 13

Thus, Y is calculated as follows:

Y = 80 + (13 (64 – 80)) / 16 = 67

If the 8-bit value for X were used directly by the table instruction, interpolation would be incorrectly performed between entries 0 and 1. Data must be shifted to the left four places before use:

LSL.W #4, Dx

Freescale Semiconductor, Inc.

5.5.2 Processing of Specific Exceptions

The following paragraphs provide details concerning sources of specific exceptions, how each arises, and how each is processed.

5.5.2.1 RESET. Assertion of $\overline{\text{RESET}}$ by external hardware or assertion of the internal $\overline{\text{RESET}}$ signal by an internal module causes a reset exception. The reset exception has the highest priority of any exception. Reset is used for system initialization and for recovery from catastrophic failure. When the reset exception is recognized, it aborts any processing in progress, and that processing cannot be recovered. Reset performs the following operations:

1. Clears T0 and T1 in the SR to disable tracing
2. Sets the S-bit in the SR to establish supervisor privilege
3. Sets the interrupt priority mask to the highest priority level (%111)
4. Initializes the VBR to zero ($00000000)
5. Generates a vector number to reference the reset exception vector
6. Loads the first long word of the vector into the interrupt SP
7. Loads the second long word of the vector into the PC
8. Fetches and initiates decode of the first instruction to be executed

Figure 5-11 is a flowchart of the reset exception

After initial instruction prefetches, normal program execution begins at the address in the PC. The reset exception does not save the value of either the PC or the SR.

If a bus error or address error occurs during reset exception processing, a double bus fault occurs, the processor halts, and the $\overline{\text{HALT}}$ signal is asserted to indicate the halted condition.

Execution of the RESET instruction does not cause a reset exception nor does it affect any internal CPU register. The SIM60 registers and the module control register in each internal peripheral module (DMA, timers, and serial modules) are not affected. All other internal peripheral module registers are reset the same as for a hardware reset. The external devices connected to the $\overline{\text{RESET}}$ signal are reset at the completion of the reset instruction

5.5.2.2 BUS ERROR. A bus error exception occurs when an assertion of the $\overline{\text{BERR}}$ signal is acknowledged. The $\overline{\text{BERR}}$ signal can be asserted by one of three sources:

1. External logic by assertion of the $\overline{\text{BERR}}$ input pin
2. Direct assertion of the internal $\overline{\text{BERR}}$ signal by an internal module
3. Direct assertion of the internal $\overline{\text{BERR}}$ signal by the on-chip hardware watchdog after detecting a no-response condition

Bus error exception processing begins when the processor attempts to use information from an aborted bus cycle.

Freescale Semiconductor, Inc.

The pointers provided by this register indicate the SI RAM entry word offset that is currently in progress. The register is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	V	RbPTR					—	—	V	RaPTR				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	V	TbPTR					—	—	V	TaPTR				

In all cases, the value in the TxPTR or RxPTR increments by one for each entry (i.e., 16-bit SI RAM word) that is processed by the SI. Since each TxPTR and RxPTR is 5 bits each, the values in each TxPTR and RxPTR can range from 0 to 31, corresponding to 32 different SI RAM entries.

The full pointer range may not necessarily be used. For instance, if the last bit is set in the fifth SI RAM entry, then the pointer will only reflect values from 0 to 4. Once the fifth entry is processed by the SI, the pointer is reset to 0.

The V-bit in each entry shows that the entry is valid. This information is particularly useful if the PTR value happens to be zero. Additionally, the V-bits save the user from having to read both the SIRP and the SISTR to obtain the needed information.

The pointer values are described based on the four possible ways the SI RAM can be configured.

7.8.5.6.1 SIRP When RDM = 00 (One Static TDM). •In this case, since 64 entries cannot be signified with a single 5-bit pointer, two 5-bit pointers are used—one for the first 32 entries and one for the second 32 entries.

RaPTR and RbPTR contain the address of the RAM entry currently active. When the SI services entries 1–32, RaPTR will be incremented, and RbPTR will be continuously cleared. When the SI services entries 33–64, RaPTR will be continuously cleared, and RbPTR will be incremented.

TaPTR and TbPTR contain the address of the Tx entry currently active. When the SI services entries 1–32, TaPTR will be incremented, and TbPTR will be continuously cleared. When the SI services entries 33–64, TaPTR will be continuously cleared, and TbPTR will be incremented.

7.8.5.6.2 SIRP When RDM = 01 (One Dynamic TDM). •For the receiver, either RaPTR or RbPTR is used, depending on which portion of the SI Rx RAM is currently active. For the transmitter, either TaPTR or TbPTR is used, depending on which portion of the SI Tx RAM is currently active.

If its V-bit is set, RaPTR contains the address of the Rx entry currently active. The SI RAM receive address block in use is 0–63, and CRORa = 0 in SISTR.

If its V-bit is set, RbPTR contains the address of the Rx entry currently active. The SI RAM receive address block in use is 64–127, and CRORa = 1 in SISTR.

If its V-bit is set, TaPTR contains the address of the Tx entry currently active. The SI RAM transmit address block in use is 128–191, and CROTa = 0 in SISTR.

Freescale Semiconductor, Inc.

NOTE

The 1:2 ratio of the SyncCLK to the serial clock does not apply when the DPLL is used to recover the clock in the 8×, 16×, or 32× modes. The synchronization actually occurs internally after the receive clock is generated by the DPLL; therefore, even the fastest DPLL clock generation (the 8× option) easily meets the required 1:2 ratio clocking limit.

7.10.13 Clock Glitch Detection

A clock glitch occurs when an input clock signal transitions between a one and zero state twice, within a small enough time period to violate the minimum high/low time specification of the input clock. Spikes are one type of glitch. Additionally, glitches can occur when excessive noise is present on a slowly rising/falling signal.

Glitched clocks are a worry to many communications systems. Not only can they cause systems to experience errors, they can potentially cause errors to occur without even being detected by the system. Systems that supply an external clock to a serial channel are often susceptible to glitches from situations such as noise, connecting/disconnecting the physical cable from the application board, or excessive ringing on the clock lines.

The SCCs on the QUICC have a special circuit designed to detect glitches that may occur in the system. The glitch circuit is designed to detect glitches that could cause the SCC to transition to the wrong state. This status information can be used to alert the system of a problem at the physical layer.

The glitch detect circuit is not a specification test. Thus, if the user develops a circuit that does not meet the input clocking specifications for the SCCs, erroneous data may be received/transmitted that is not indicated by the glitch detection logic. Conversely, if a glitch indication is signaled, it does not guarantee that erroneous data was received/transmitted.

Regardless of whether the DPLL is used, the received clock is passed through a noise filter that eliminates any noise spikes that affect a single sample. This sampling is enabled with the GDE bit of the GSMR.

If a spike is detected, a maskable receive or transmit glitched clock interrupt is generated in the event register of the SCC channel. Although the user may choose to reset the SCC receiver or transmitter or to continue operation, he should keep statistics on clock glitches for later evaluation. In addition, the glitched status indication may be used as a debugging aid during the early phases of prototype testing.

7.10.14 Disabling the SCCs on the Fly

If an SCC is not needed for a period of time, it may be disabled and reenabled later. In this case, a sequence of operations is followed.

These sequences ensure that any buffers in use will be properly closed and that new data will be transferred to/from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description

Freescale Semiconductor, Inc.

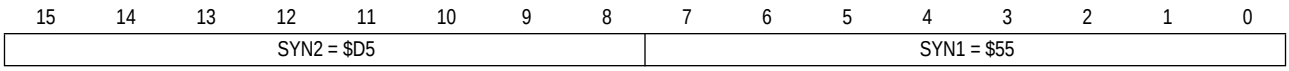
IADDR1–4. These four registers are used in the hash table function of the individual addressing mode. The user may write zeros to these values after reset and before the Ethernet channel is enabled to disable all individual hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table.

TADDR. This parameter allows the user to add and delete addresses from the individual and group hash tables. After placing an address in TADDR, the user would then issue the SET GROUP ADDRESS command. TADDR_L is the lowest order word, and TADDR_H is the highest order word.

7.10.23.9 ETHERNET PROGRAMMING MODEL. The host configures SCC to operate as an Ethernet controller by the MODE bits in the GSMR.

The receive errors (collision, overrun, nonoctet aligned frame, short frame, frame too long, and CRC error) are reported through the Rx BD. The transmit errors (underrun, heartbeat, late collision, retransmission limit, and carrier sense lost) are reported through the Tx BD.

Several bit fields in the GSMR must be programmed to special values for Ethernet. See the GSMR for more details. The user should program the DSR as shown below. The 6 bytes of preamble programmed in the GSMR, in combination with the programming of the DSR shown below, causes 8 bytes of preamble on transmit (including the 1-byte start delimiter with the value $D5).



7.10.23.10 ETHERNET COMMAND SET. Ethernet:Ethernet Command SetThe following transmit and receive commands are issued to the CR.

NOTE

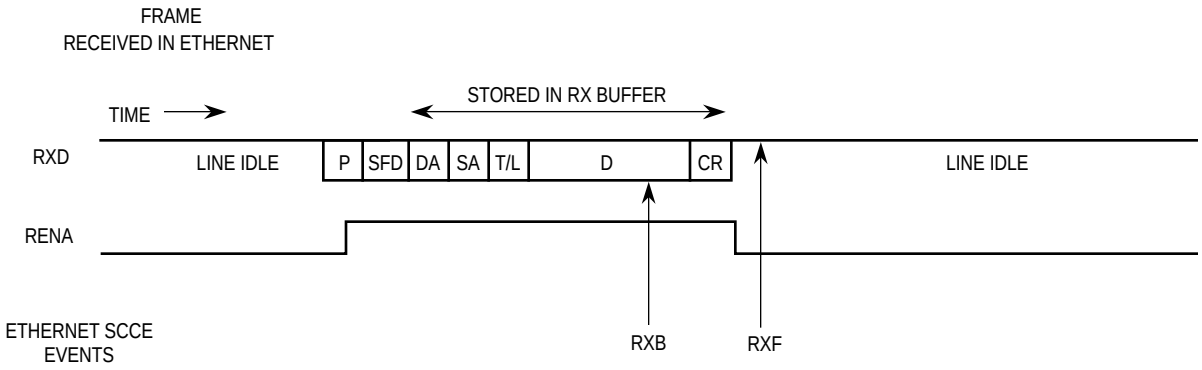
Before issuing the CP RESET command, configure the TENA ($\overline{RTS}$) pin to be an input. See step 3 of 7.10.23.23 SCC Ethernet Example. for more information.

7.10.23.10.1 Transmit Commands. The following paragraphs describe the Ethernet transmit commands.

STOP TRANSMIT Command**.** When used with the Ethernet controller, this command violates specified behavior of an Ethernet/IEEE 802.3 station. It should not be used.

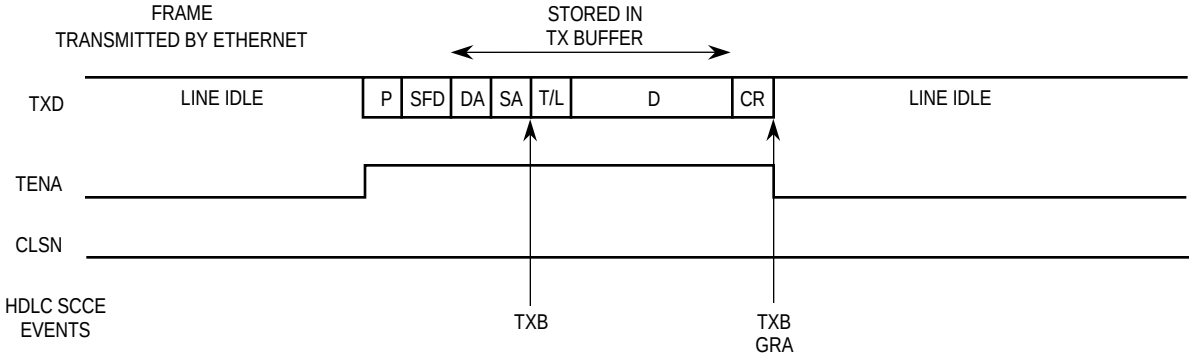
GRACEFUL STOP TRANSMIT Command**.** The channel GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way. It stops transmission after the current frame has completed transmission or undergoes a collision (immediately if there is no frame being transmitted). The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the Ethernet transmit parameters, including BDs, may be modified by the user. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

Freescale Semiconductor, Inc.



NOTES:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RENA events, if required, must be programmed in the port C parallel I/O, not in the SCC itself.
3. The RXF interrupt may occur later than RENA due to receive FIFO latency.

LEGEND:
P = Preamble, SFD = Start Frame Delimiter, DA and SA = Source/Destination Address, T/L = Type/Length, D = Data, and CR = CRC bytes.



NOTES:
1. TXB events assume the frame required two transmit buffers.
2. The GRA event assumes a GRACEFUL STOP TRANSMIT command was issued during frame transmission.
3. The TENA or CLSN events, if required, must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 7-72. Ethernet Interrupt Events Example;

7.10.23.21 ETHERNET MASK REGISTER (SCCM). The SCCM is referred to as the Ethernet mask register when the SCC is operating as an Ethernet controller. It is a 16-bit read-write register that has the same bit formats as the Ethernet event register. If a bit in the Ethernet mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.23.22 ETHERNET STATUS REGISTER (SCCS). This register is not valid for the Ethernet protocol. The current state of the RENA and CLSN signals may be read in port C.

Freescale Semiconductor, Inc.

R—Ready

0 = The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered.
1 = The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.

Bits 14, 10, 8–2—Reserved

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = The TXB or TXE bit in the event register is set when this buffer is serviced. TXB and TXE can cause interrupts if they are enabled.

L—Last

0 = This buffer does not contain the last character of the message.
1 = This buffer contains the last character of the message.

CM—Continuous Mode

This bit is valid only when the SPI is configured as a master; it should be written as a zero in slave mode.

0 = Normal operation.
1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD.

The following status bits are written by the SPI after it has finished transmitting the associated data buffer.

UN—Underrun

The SPI encountered a transmitter underrun condition while transmitting the associated data buffer. This error condition is valid only when the SPI is configured as a slave.

ME—Multi-Master Error

This buffer was closed because the $\overline{\text{SPISEL}}$ pin was asserted when the SPI was operating as a master. This indicates a synchronization problem between multiple masters on the SPI bus.

Freescale Semiconductor, Inc.

12. Write $00000020 to the CIMR to allow the SPI to generate a system interrupt. (The CICR should also be initialized.)
13. Write $0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
14. Write PBDAT bit 0 with zero to assert the SPI select pin.
15. Set the STR bit in the SPCOM to start the transfer.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 5 bytes have been received because the L-bit of the Tx BD was set.

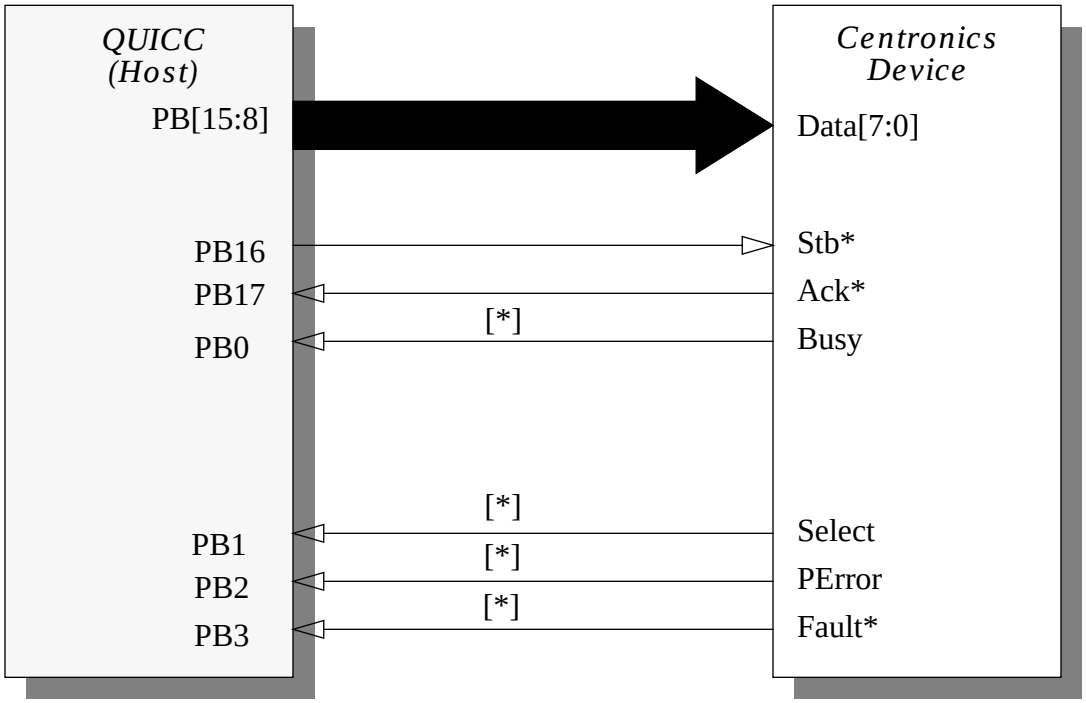
7.12.7 SPI Slave Example

The following list is an initialization sequence for use of the SPI as a slave. It is very similar to the SPI master example except that the $\overline{\text{SPISEL}}$ pin is used, rather than a general-purpose I/O pin.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port B pins to enable the SPIMOSI, SPIMISO, $\overline{\text{SPISEL}}$, and SPICLK pins. Write PBPAR bits 0, 1, 2, and 3 with ones. Write PBDIR bits 0, 1, 2, and 3 with ones. Write PBODR bits 0, 1, 2, and 3 with zeros.
3. Write RBASE and TBASE in the SPI parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
4. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
5. Write RFCR with $18 and TFCR with $18 for normal operation.
6. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
7. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
8. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $B800 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
9. Write $FF to the SPIE to clear any previous events.
10. Write $37 to the SPIM to enable all possible SPI interrupts.
11. Write $00000020 to the CIMR to allow the SPI to generate a system interrupt. (The

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com



[*] - optional **Figure 7-95. Centronics Transmitter Configuration**

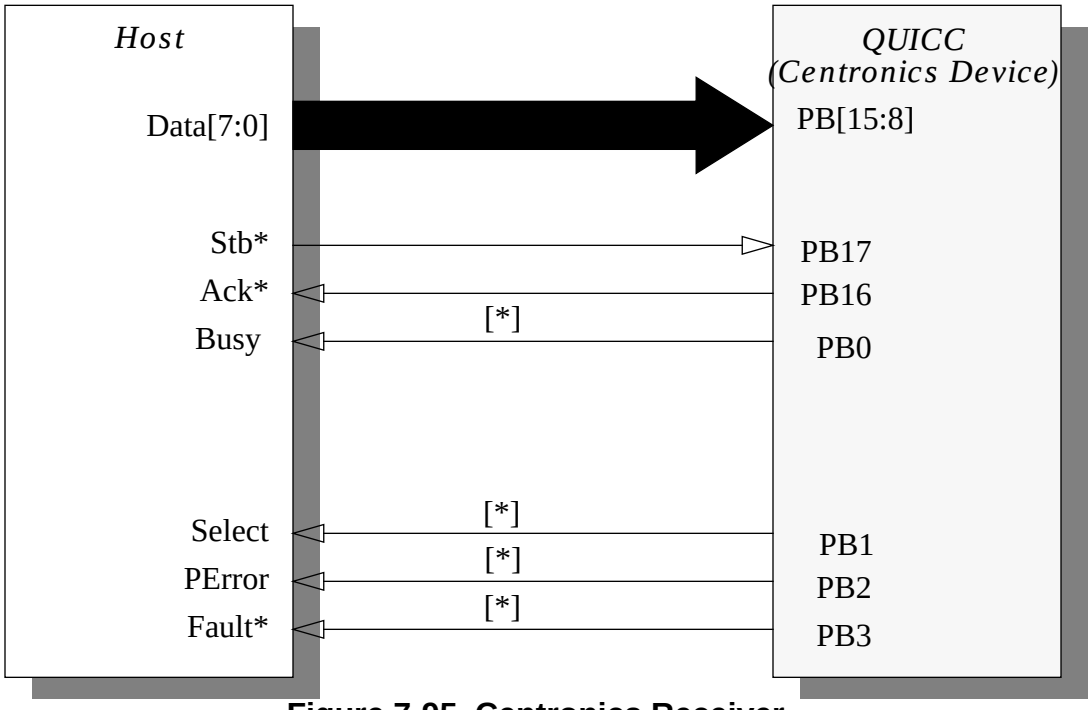


Figure 7-95. Centronics Receiver

7.13.8.1 CENTRONICS CONTROLLER KEY FEATURES. Super-set of the Centronics standard

- 8-bit or 16-Bit Data Transfer

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

The CPIC prioritizes all interrupt sources based upon their assigned priority level. The highest priority interrupt request is presented to the CPU32+ core for servicing. After the vector number corresponding to this interrupt is passed to the CPU32+ core during an interrupt acknowledge cycle, that interrupt request is cleared. If there are remaining interrupt requests, they are then prioritized, and another interrupt request may be presented to the CPU32+ core.

The 3-bit mask in the CPU32+ status register ensures that a subsequent interrupt request at a higher interrupt priority level will suspend handling of a lower priority interrupt. The mask indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

The CISR and the mask register in the CPU32+ core can be used together to allow a higher priority interrupt within the same interrupt level to be presented to the CPU32+ core before the servicing of a lower priority interrupt is completed. Each bit in the CISR corresponds to a CPM interrupt source. During an interrupt acknowledge cycle for a CPM interrupt, the in-service bit in the CISR is set by the CPIC for that interrupt source. The setting of the bit prevents any subsequent CPM interrupt requests at this priority level or lower (within the CPIC interrupt table), until the servicing of the current interrupt has completed and the in-service bit is cleared by the user. (Pending interrupts for these sources are still set in the CPIC during this time).

Thus, in the interrupt service routine for the CPM interrupts, the user can lower the core's mask to the next lower level (the level being serviced minus 1) to allow higher priority interrupts within this level to generate an interrupt request. This capability provides nesting of interrupt requests for CPM interrupt level sources in a similar manner as the CPU32+ core's interrupt mask provides nesting of interrupt requests for the seven interrupt priority levels.

7.15.3 Masking Interrupt Sources in the CPM

By programming the CPM interrupt mask register (CIMR), the user may mask the CPM interrupts to prevent an interrupt request to the CPU32+ core. Each bit in the CIMR corresponds to one of the CPM interrupt sources. To enable an interrupt, write a one to the corresponding CIMR bit.

When a masked CPM interrupt source has a pending interrupt request, the corresponding bit in the CIPR is still set, even though the interrupt is not generated to the CPU32+ core. By masking all interrupt sources in the CIMR, the user may implement a polling interrupt servicing scheme for the CPM interrupts.

When a CPM interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that block. Table 7-22 indicates the interrupt sources that have multiple interrupting events. Figure 7-100 shows an example of how the masking occurs, using an SCC as an example.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

3. Decide which events in the SCCE1 will be handled in this handler and clear those bits as soon as possible. (SCCE bits are cleared by writing ones.)
4. Handle events in the SCC1 Rx or Tx BD tables.
5. Clear the SCC1 bit in the CISR.
6. Execute the RTE instruction. If any unmasked bits in SCCE1 remain at this time (either not cleared by the software or set by the QUICC during the execution of this handler), this interrupt source will be made pending again immediately following the RTE instruction.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The AM27–AM11 bits should be programmed to determine the block size of the chip select or $\overline{\text{RASx}}$ line. This should be the total number of bytes in each memory array except for the EEPROM, which should be 32 Kbytes, rather than 8 Kbytes.

FCM3–FCM0 may be set to all ones to allow the chip select or $\overline{\text{RASx}}$ line to assert on all function codes except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to ones, at least during the initial stages of debugging.

BCYC1–BCYC0 is not applicable.

PGME should be set to enable page mode and cleared otherwise.

SPS1–SPS0 should be cleared (32-bit SRAM port).

DSSEL should be set only if this is a DRAM bank.

9.8.4 Interfacing Multiple QUICCs to an MC68EC030

It is possible to interface multiple QUICCs to an MC68EC030. The first QUICC can be configured as previously shown in this subsection. Additional QUICCs should be configured as noted in the following list:

- The additional QUICCs should have their CONFIG2–CONFIG0 pins configured for slave mode, global chip select *disabled*, and MBAR at $003FF04.
- The MBAR of the additional QUICCs should be programmed using the $\overline{\text{MBARE}}$ pin and MBARE register as described in the Section 6 System Integration Module (SIM60).
- An external bus arbiter is required to take the bus request of the additional QUICC (which is an output because of the CONFIG2–CONFIG0 pins) and prioritize it with the other QUICCs, present it to the MC68EC030, and issue a bus grant to the appropriate QUICC.
- An external interrupt prioritizer is required to determine which QUICC $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins are currently routed to the MC68EC030. Alternatively, the additional QUICC should have its interrupts brought out on a single $\overline{\text{RQOUT}}$ pin, which is routed to one of the original QUICC interrupt inputs. This would eliminate the external logic.

9.8.5 Using a Higher Speed MC68EC030 Master with the QUICC

It is possible to interface an MC68EC030 and QUICC through an asynchronous bus. This should allow an external master to operate at higher frequencies than those of the QUICC with minimal effort. As of this writing, the QUICC top frequency is 25 MHz; whereas, MC68EC030s are available up to 40 MHz. One potentially attractive option for a designer would be to consider disabling the CPU32+ core and increasing system performance by adding a 40-MHz MC68EC030 asynchronously. While this option is available, it is important for the designer to consider what effects a higher speed MC68EC030 would ultimately have on system cost and performance over using the QUICC CPU32+ at a lower frequency.

For the designer to take full advantage of a high-speed MC68EC030, it will be necessary to add additional glue to that shown in Figure 9-27. The additional circuitry takes the form of a DRAM controller, which is used instead of using the QUICC memory controller. The need for the additional logic is twofold. First, if the QUICC memory controller capabilities are used, all memory accesses would be at the clock rate of 25 MHz. In addition, since the

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

INDEX

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.