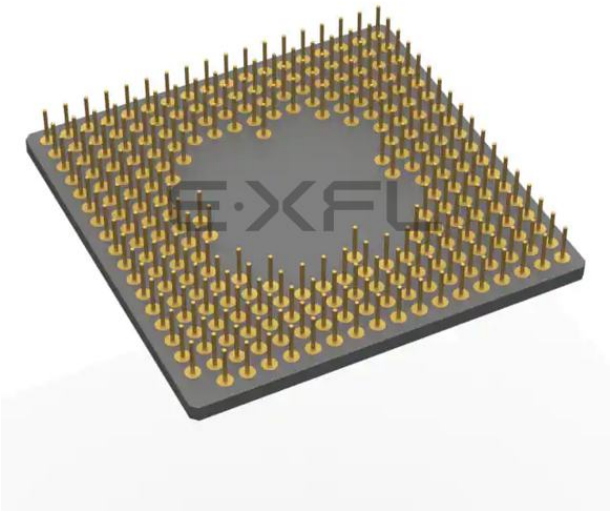


Welcome to E-XFL.COM

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	33MHz
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	0°C ~ 70°C (TA)
Security Features	-
Package / Case	241-BEPGA
Supplier Device Package	241-PGA (47.24x47.24)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68360rc33l

aligned on word or long-word boundaries, respectively. The QUICC IDMAs, when used, reduce the misalignment overhead to a minimum.

4.1 BUS TRANSFER SIGNALS

The bus transfers information between the QUICC and external memory or a peripheral device. External devices can accept or provide 8, 16, or 32 bits in parallel and must follow the handshake protocol described in this section. The maximum number of bits accepted or provided during a bus transfer is defined as the port width. The QUICC contains an address bus that specifies the address for the transfer and a data bus that transfers the data. Control signals indicate the beginning and type of the cycle as well as the address space and size of the transfer. The selected device then controls the length of the cycle with the signal(s) used to terminate the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity of the address and provide timing information for the data.

Both asynchronous and synchronous operation is possible for any port width. In asynchronous operation, the bus and control input signals are internally synchronized to the QUICC clock, introducing a delay. This delay is the time required for the QUICC to sample an input signal, synchronize the input to the internal clocks, and determine whether it is high or low. In synchronous mode, the bus and control input signals must be timed to setup and hold times. Since no synchronization is needed, bus cycles can be completed in three clock cycles in this mode. Additionally, using the fast-termination option of the chip-select signals, two-clock operation is possible.

Furthermore, for all inputs, the QUICC latches the level of the input during a sample window around the falling edge of the clock signal. This window is illustrated in Figure 4-1, where t_{su} and t_h are the input setup and hold times, respectively. To ensure that an input signal is recognized on a specific falling edge of the clock, that input must be stable during the sample window. If an input makes a transition during the window time period, the level recognized by the QUICC is not predictable; however, the QUICC always resolves the latched level to either a logic high or low before using it. In addition to meeting input setup and hold times for deterministic operation, all input signals must obey the protocols described in this section.

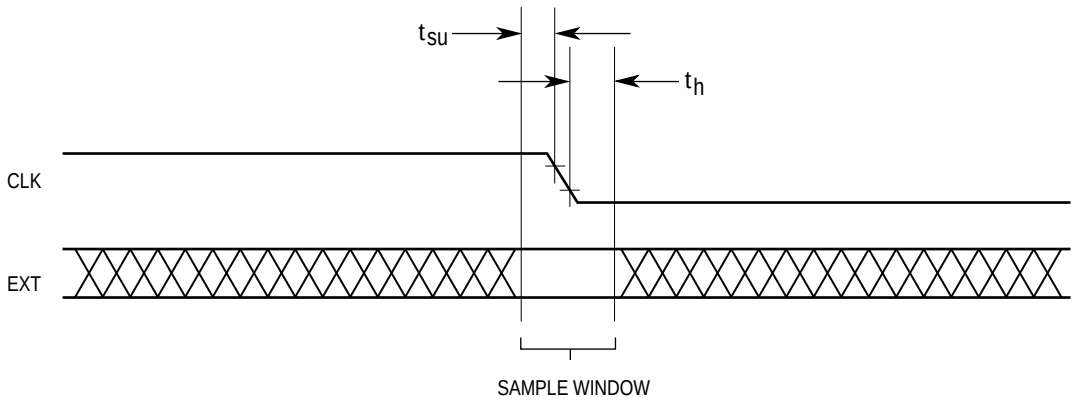


Figure 4-1. Input Sample Window

5.3.4.2 TABLE EXAMPLE 2: COMPRESSED TABLE. In Example 2 (see Figure 5-8), the data from Example 1 has been compressed by limiting the maximum value of the independent variable. Instead of the range $0 \leq X = 65535$, X is limited to $0 \leq X \leq 1023$. The table has been compressed to only five entries, but up to 256 levels of interpolation are allowed between entries.

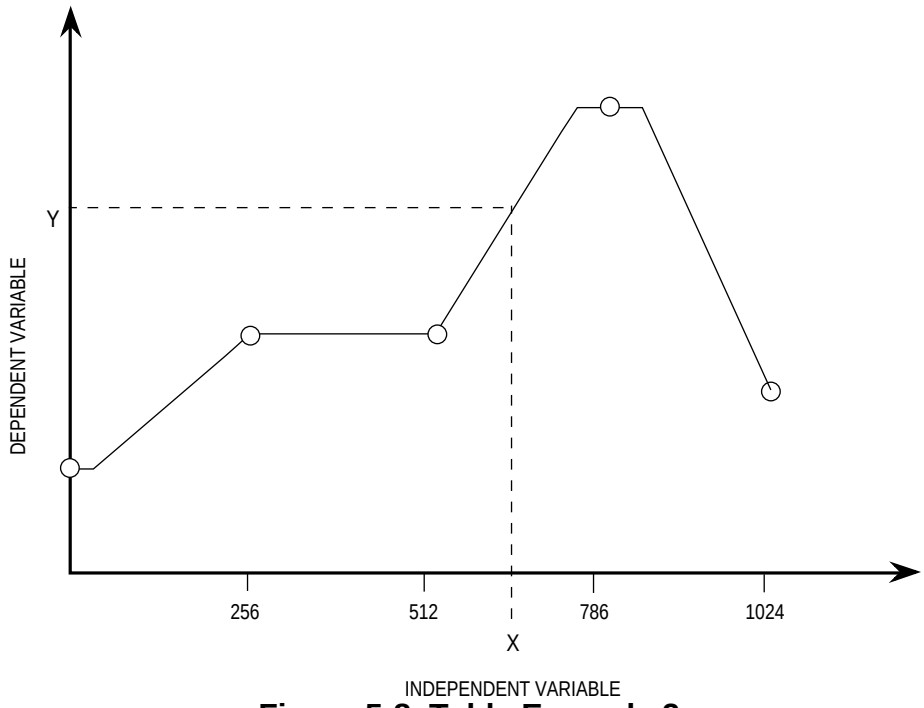


Figure 5-8. Table Example 2

NOTE

Extreme table compression with many levels of interpolation is possible only with highly linear functions. The table entries within the range of interest are listed in Table 5-14.

Table 5-14. Compressed Table Entries

Entry Number	X-Value	Y-Value
2	512	1311
3	786	1966

Since the table is reduced from 257 to 5 entries, independent variable X must be scaled appropriately. In this case the scaling factor is 64, and the scaling is done by a single instruction:

LSR.W #6,Dx

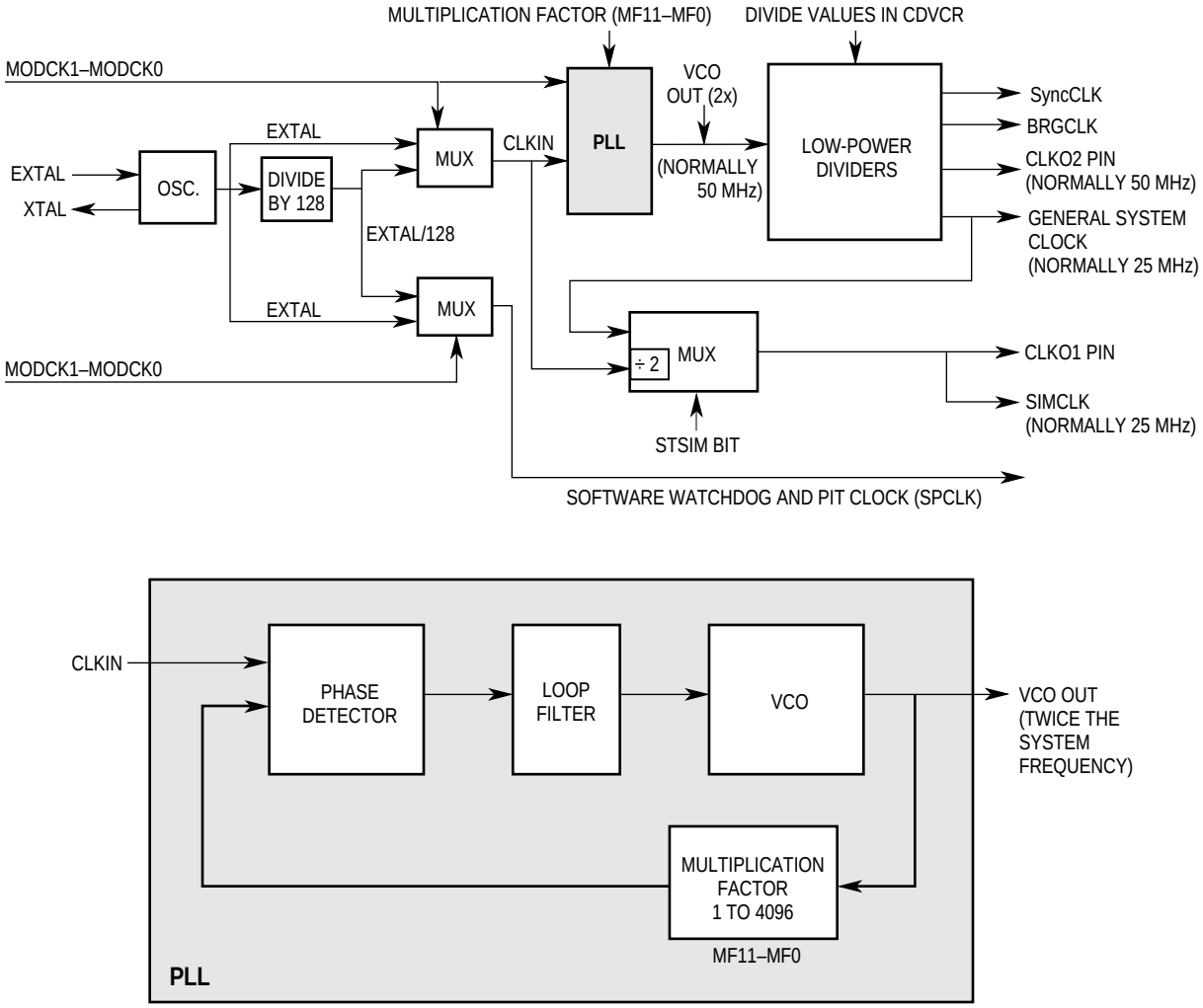


Figure 6-5. System Clocks Schematic

NOTE

User must select the proper MODCK configuration as well as correct XFC capacitor value when selecting the input clock frequency and desired system frequency.

For more information on MODCK see 6.5.8 Configuration Pins (MODCK1–MODCK0). For more information on XFC see 10.8 External Capacitor for PLL.

6.5.2 Oscillator Prescaler (Divide by 128)

In some applications, the use of a ~ 32-kHz crystal is attractive because of cost and low power, but is not attractive due to the extra startup time required for such a slow frequency crystal. Therefore, the SIM60 has an option for the user to provide a higher frequency crystal (for instance, in the ~ 4-MHz range) and divide it by 128 (back down to the 32-kHz range) before it is used by the QUICC. This results in much faster startup time than a 32-kHz crystal, plus low cost (if a common 4.192-MHz frequency is chosen), with only a small impact on power consumption during low-power modes (since the ~4-MHz frequency is immediately

conflicts, external peripherals must not attempt to initiate cycles during show cycles with arbitration disabled.

Table 6-3. Show Cycle Control Bits

SHEN1	SHEN0	Action
0	0	Normal operation. Split buses mode. Show cycles is disabled and external arbitration is enabled.
0	1	Show cycles enabled. External arbitration is disabled and $\overline{BG}$ is never asserted.
1	x	Show cycles enabled. External arbitration is enabled and internal activity is halted when $\overline{BG}$ is asserted by the QUICC.

NOTE

During normal operation, $\overline{BERR}$ and $\overline{DSACKx}$ for internal cycles will not appear as an external cycle in master mode.

For fast termination cycles, $\overline{DSACKx}$ is never asserted externally regardless of the show cycle bit settings.

In slave mode, these bits default to 00, and writes by the user have no effect on operation.

In case 00 (show cycles disabled), if the external bus is available when an internal-to-internal access occurs, the address and function code pins will reflect the internal access.

Case 01 may be used as a debugging aid to eliminate the external bus master as a possible cause of the problem or to prevent interference in a user debug session.

Although case 00 is recommended for normal operation, case 1x may be used during initial development for visibility on the internal bus, at the expense of performance. Moving from 1x to 00 increases performance for two reasons: 1) both the internal and external buses may be used simultaneously and 2) the external bus master will obtain the $\overline{BG}$ signal assertion more quickly after asserting $\overline{BR}$.

SUPV—Supervisor/User Data Space

The SUPV bit defines the SIM60 global registers as either supervisor data space or user (unrestricted) data space. It is a don't care on the SIM60 and is reserved for future expansion.

0 = The SIM60 registers defined as supervisor/user data are unrestricted (FC2 is a don't care).

1 = The SIM60 registers defined as supervisor/user are restricted to supervisor data access (FC3–FC0 = $5). Any attempted user space write is ignored and returns $\overline{BERR}$.

master may generate a bus error as a result of this register, and for parity errors, the $\overline{\text{PERR}}$ pin may be externally connected to an interrupt input.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPER	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0

Bits 15–9—Reserved

WPER—Write Protect Error

This bit is asserted when a write protect error occurs. A bus monitor ($\overline{\text{BERR}}$ assertion) will (if enabled) prompt the user to read this register if no $\overline{\text{DSACK}}$ is provided on a write cycle. The accessed address will be in the $\overline{\text{BERR}}$ exception descriptor. WPER is cleared by writing one to this bit or by performing a system reset. Writing a zero has no effect on WPER.

PERx—Parity Error

These bits indicate that a parity error was detected when reading from bank N. $\overline{\text{BERR}}$ is internally asserted if PBEE in the GMR is set and if an internal master performs this cycle. The $\overline{\text{PERR}}$ signal is continuously asserted until all PERx bits are cleared. PERx is cleared by writing one or by performing a system reset. Writing a zero has no effect on PERx.

NOTE

If external masters of the MC68030-type (including QUICCs) are chosen to be asynchronous (configured by clearing the SYNC bit in the GMR), then they have no parity support.

6.13.3 Base Register (BR)

This register is used for both DRAM and SRAM banks. Most bits are valid for both the DRAM and SRAM banks, but some bits are only valid for SRAM banks. This register is a 32-bit read-write register that may be accessed at any time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	BA11	FC3	FC2	FC1	FC0	TRLXQ	BACK40	CSNT40	CSNTQ	PAREN	WP	V
0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

V—Valid Bit

This bit indicates that the contents of the BR and OR pair are valid. The $\overline{\text{CS}}$/$\overline{\text{RAS}}$ signal will not assert until the V-bit is set.

tire timer table. The TIME bit would normally be turned on at this time; however, it can be turned on later if it is required that all RISC timers be synchronized.

2. Determine the maximum number of timers to be located in the timer table and configure TM_BASE in the RISC timer table parameter RAM to point to a location in the dual port RAM with $4 \times N$ bytes available, where N is the number of timers. If N is less than 16, use timer 0 through timer N–1 (for space efficiency).
3. Clear the TM_cnt in the RISC timer table parameter RAM to show how many ticks have elapsed since the RISC internal timer was enabled. This step is optional.
4. Clear the RISC timer event register if it is not already cleared. (Ones are written to clear this register.)
5. Configure the RTMR to enable those timers that should generate interrupts. (Ones enable interrupts.)
6. Set the RISC timer table bit in the CPM interrupt mask register to generate interrupts to the system. (The CPM interrupt controller may require other initialization not mentioned here.)
7. Configure the TM_cmd field of the RISC timer table parameter RAM. At this point, determine whether a timer is to be enabled or disabled, one-shot or restart, and what its timeout period should be. If the timer is being disabled, the parameters (other than the timer number) are ignored.
8. Issue the SET TIMER command by writing $0861 to the CR.
9. Repeat the preceding two steps for each timer to be enabled or disabled.

7.4.7 RISC Timer Initialization Example

The following sequence initializes RISC timer 0 to generate an interrupt approximately every second using a 25-MHz general system clock:

1. Write the TIMEP bits of the RCCR with 111111 to generate the slowest clock. This value will generate a tick every 65536 clocks, which is every 2.6 ms at 25 MHz.
2. Configure TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with 4 bytes available. Assuming the beginning of dual-port RAM is available, write $0000 to TM_BASE.
3. Write $0000 to TM_cnt in the RISC timer table parameter RAM to see how many ticks have elapsed since the RISC internal timer was enabled. This step is optional.
4. Write $FFFF to the RTER to clear any previous events.
5. Write $0001 to the RTMR to enable RISC timer 0 to generate an interrupt.
6. Write $00020000 to the CPM interrupt mask register to allow the RISC timers to generate a system interrupt. Initialize the CPM interrupt configuration register.
7. Write $C0000EE6 to the TM_cmd field of the RISC timer table parameter RAM. This enables RISC timer 0 to time out after 3814 (decimal) ticks of the timer. The timer will automatically restart after it times out.
8. Write $0851 to the CR to issue the SET TIMER command.
9. Set the TIME bit in the RCCR to enable the RISC timer to begin operation.

RCI — RISC Controls IDMA

0 = Single Buffer Mode. The user programs all IDMA registers for each buffer transfer.
1 = Auto buffer or buffer chaining mode. The RISC reconfigures the IDMA channel at the end of each buffer transfer according to the buffer descriptor ring. The choice between auto buffer and buffer chaining is made in the buffer descriptor itself.

REQG — Request Generation

The REQG bits define what generates the requests for IDMA activity over the bus.

00 = Internal request at limited rate (limited burst bandwidth) set by BT bits
01 = Internal request at maximum rate (one burst)
10 = External request burst transfer mode ($\overline{\text{DREQx}}$ is level sensitive)
11 = External request cycle steal ($\overline{\text{DREQx}}$ is edge sensitive)

SAPI — SAPR Increment

0 = SAPR is not incremented after each transfer.
1 = SAPR is incremented by one, two, or four after each transfer, according to the SSIZE bits. (SAPR may be incremented by an amount less than the SSIZE value at the beginning or end of a block transfer, depending on the source starting address or byte count.)

DAPI — DAPR Increment

0 = DAPR is not incremented after each transfer.
1 = DAPR is incremented by one, two, or four after each transfer, according to the DSIZE bits. (DAPR may be incremented by an amount less than the DSIZE value at the beginning or end of a block transfer, depending on the destination starting address or byte count.)

SSIZE — Source Size

The following decoding shows the definitions for the SSIZE bits. The user should set these bits to the port size of the source (e.g., choose byte for an 8-bit peripheral).

00 = Long word
01 = Byte
10 = Word
11 = Reserved

DSIZE — Destination Size

The following decoding shows the definitions for the DSIZE bits. The user should set these bits to the port size of the destination (e.g., choose byte for an 8-bit peripheral).

00 = Long word
01 = Byte
10 = Word
11 = Reserved

RFSDx—Receive Frame Sync Delay for TDM A or B

These two bits determine the number of clock delays between the receive sync and the first bit of the receive frame. Even if the CRTx bit is set, these bits do not control the delay for the transmit frame.

00 = No bit delay (The first bit of the frame is transmitted/received on the same clock as the sync; use for GCI.)
01 = 1-bit delay (Use for IDL.)
10 = 2-bit delay
11 = 3-bit delay

Refer to Figure 7-29 and Figure 7-30 for an example of the use of these bits.

DSCx—Double-Speed Clock for TDM A or B

Some TDMs such as GCI define the input clock to be 2× faster than the data rate. This bit controls this option.

0 = The channel clock (L1RCLKx and/or L1TCLKx) is equal to the data clock. (Use for IDL and most TDM formats.)
1 = The channel clock rate is twice the data rate. (Use for GCI.)

CRTx—Common Receive and Transmit Pins for TDM A or B

This bit is useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx pins can be used as general-purpose I/O pins.

0 = Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing, and the transmit section uses L1TCLKx and L1TSYNCx for framing.
1 = Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. (Use for IDL and GCI.)

STZx—Set L1TXDx to Zero for TDM A or B

0 = Normal operation.
1 = L1TXDx is set to zero until serial clocks are available, which is useful for GCI activation. Refer to 7.8.7.1 SI GCI Activation/Deactivation Procedure.

CEx—Clock Edge for TDM A or B

When DSCx =0
0 = The data is transmitted on the rising edge of the clock and received on the falling edge. (Use for IDL and GCI.)
1 = The data is transmitted on the falling edge of the clock and received on the rising edge.

When DSCx = 1
0 = The data is transmitted on the rising edge of the clock and received on the rising edge. (Use for IDL and GCI.)
1 = The data is transmitted on the falling edge of the clock and received on the falling edge.

HDLC, particularly the framing structure of HDLC: namely, SDLC, SS#7, AppleTalk, LAPB, and LAPD. The framing structure of HDLC is shown in Figure 7-50.

HDLC uses a zero insertion/deletion process (commonly known as bit-stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer to provide a method of clocking and synchronizing the transmitter/receiver.

Since the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or packet and circuit switched systems, an address field is needed to carry the frame's destination address. The length of this field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. For instance, SDLC and LAPB use an 8-bit address. SS#7 has no address field at all because it is always used in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one piece of equipment. It also defines a broadcast address. Some HDLC-type protocols also allow for extended addressing beyond 16 bits.

The 8 or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends upon the protocol using the frame.

Data is transmitted in the data field, which can vary in length depending upon the protocol using the frame. Layer 3 frames are carried in this data field.

Error control is implemented by appending a CRC (CRC) to the frame, which in most protocols is 16-bits long but may be as long as 32-bits.

In HDLC, the LSB of each octet is transmitted first, and the MSB of the CRC is transmitted first.

When the MODE bits of the GSMR select the HDLC mode, then that SCC functions as an HDLC controller. When an SCC in HDLC mode is used with a nonmultiplexed modem interface, then the SCC outputs are connected directly to the external pins. Modem signals may be supported through the port C pins. The receive and transmit clocks can be supplied either from the bank of baud rate generators, by the DPLL, or externally. The HDLC controller may also be connected to one of the two TDM channels of the serial interface and used with the TSA.

The HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with the CPU32+ core and may be either synchronous or asynchronous with respect to the other SCCs. The user can allocate up to 196 BDs for receive and transmit tasks so that many frames may be transmitted or received without host intervention.

7.10.17.1 HDLC CONTROLLER KEY FEATURES. The HDLC contains the following key features:

- Flexible Data Buffers with Multiple Buffers per Frame
- Separate Interrupts for Frames and Buffers (Receive and Transmit)
- Received Frames Threshold To Reduce Interrupt Overhead

DRT—Disable Receiver While Transmitting

0 = Normal operation
1 = While data is being transmitted by the SCC, the receiver is disabled, being gated by the internal $\overline{\text{RTS}}$ signal. This configuration is useful if the HDLC channel is configured onto a multidrop line and the user does not wish to receive his own transmission.

BUS—HDLC Bus Mode

0 = Normal HDLC operation
1 = HDLC Bus operation selected. See 7.10.18 HDLC Bus Controller for more details.

BRM—HDLC Bus $\overline{\text{RTS}}$ Mode

This bit is only valid if BUS = 1; otherwise, it is ignored.

0 = Normal $\overline{\text{RTS}}$ operation during HDLC Bus mode. $\overline{\text{RTS}}$ is asserted on the first bit of the transmit frame and negated after the first collision bit is received.
1 = Special $\overline{\text{RTS}}$ operation during HDLC Bus mode. $\overline{\text{RTS}}$ is delayed by one bit with respect to the normal case. This is useful when the HDLC Bus protocol is run locally, and at the same time, transmitted over a long-distance transmission line. Data may be delayed by one bit before it is sent over the transmission line; thus, $\overline{\text{RTS}}$ may be used to enable the transmission line buffers. The result is a clean signal level sent over the transmission line.

MFF—Multiple Frames in FIFO

0 = Normal operation. The transmit FIFO can never contain more than one HDLC frame. The CTS lost status will be reported accurately on a per-frame basis. The receiver is not affected by this bit.
1 = The transmit FIFO can contain multiple frames, but CTS lost is not guaranteed to be reported on the exact buffer/frame on which it truly occurred. This option, however, can improve the performance of HDLC transmissions in cases of small back-to-back frames or in cases where the user desires to strongly limit the number of flags transmitted between frames. The receiver is not affected by this bit.

7.10.17.9 HDLC RECEIVE BUFFER DESCRIPTOR (RX BD). The HDLC controller uses the Rx BD to report information about the received data for each buffer. An example of the Rx BD process is shown in Figure 7-52.

is still obtained from the SCC's individual $\overline{\text{CTSx}}$ pin; thus, the $\overline{\text{CTS}}$ pin must be configured in port C to connect to the desired SCC. Since the SCC only receives clocks during its time slot, the $\overline{\text{CTS}}$ pin is only sampled during the transmit clock edges of the SCC's particular time slot.

7.10.18.3 HDLC BUS MEMORY MAP AND PROGRAMMING. HDLC bus on the QUICC is implemented using the HDLC controller with certain bits set. Otherwise, the user should consult the HDLC controller section for detailed information on the programming of HDLC.

7.10.18.3.1 GSMR Programming. The GSMR programming sequence is as follows:

1. Set the MODE bits to HDLC.
2. Set the ENT and ENR bits as desired.
3. Set the DIAG bits for normal operation.
4. Set the RDCR and TDCR bits for 1x clock.
5. Set the TENC and RENC bits for NRZ.
6. Clear RTSM.
7. Set CTSS to one and all other bits to zero or to their default condition.

7.10.18.3.2 PSMR Programming. The PSMR programming sequence is as follows:

1. Set the NOF bits as desired.
2. Set the CRC to 16-bit CRC CCITT.
3. Set the RTE bit.
4. Set the BUS bit.
5. Set the BRM bit to one or zero as desired.
6. Set all other bits to zero or to their default condition.

7.10.18.3.3 HDLC Bus Controller Example. Except for the previously discussed register programming, the HDLC Example #1 may be followed.

7.10.19 AppleTalk Controller

AppleTalk is a set of protocols developed by Apple Computer Inc. to provide a LAN service between Macintosh computers and printers. Although AppleTalk can be implemented over a variety of physical and link layers, including Ethernet, the AppleTalk protocols have traditionally been most closely associated with one particular physical and link layer protocol called LocalTalk.

The term LocalTalk refers to an HDLC-based link layer and physical layer protocol that runs at the rate of 230.4 kbps. In this document, the term AppleTalk controller refers to a support that the QUICC provides for the LocalTalk protocol.

The AppleTalk controller provides the required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames. These capabilities, as well as the use of the HDLC controller in conjunction with the DPLL operating in FM0 mode, provide the proper connection formats to the LocalTalk bus.

20. Clear GADDR1–GADDR4. The group hash table is not used.
21. Write PADDR1_H with $0000, PADDR1_M with $0000, and PADDR1_L with $0040 to configure the physical address.
22. Write P_Per with $0000. It is not used.
23. Clear IADDR1–IADDR4. The individual hash table is not used.
24. Clear TADDR_H, TADDR_M, and TADDR_L for the sake of clarity.
25. 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.
26. Initialize the Tx BD. Assume the Tx data frame is at $00002000 in main memory and contains fourteen 8-bit characters (destination and source addresses plus the type field). Write $FC00 to Tx_BD_Status. Add PAD to the frame and generate a CRC. Write $000D to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
27. Write $FFFF to the SCCE to clear any previous events.
28. Write $001A to the SCCM to enable the TXE, RXF, and TXB interrupts.
29. Write $40000000 to the CIMR to allow SCC1 to generate a system interrupt. (The CICR should also be initialized.)
30. Write $00000000 to GSMR_H1 to enable normal operation of all modes.
31. Write $1088000C to GSMR_L1 to configure the $\overline{\text{CTS}}$ (CLSN) and $\overline{\text{CD}}$ (RENA) pins to automatically control transmission and reception (DIAG bits) and the Ethernet mode. TCI is set to allow more setup time for the EEST to receive the QUICC's transmit data. TPL and TPP are set as required for Ethernet. The DPLL is not used with Ethernet. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
32. Write $D555 to DSR
33. Set the PSMR1 to $0A0A to configure 32-bit CRC, promiscuous mode (receive all frames), and begin searching for the start frame delimiter 22 bits after RENA.
34. Enable the TENA pin ($\overline{\text{RTS}}$). Since the MODE bits in GSMR have been written to Ethernet, the TENA signal is low. Write PCPAR bit 0 with a one. Write PCDIR bit 0 with a zero.
35. Write $1088003C to GSMR_L1 to enable the SCC1 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 14 bytes and the 46 bytes of automatic pad (plus the 4 bytes of CRC) have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after a frame is received. Any additional receive data beyond 1520 bytes or a single frame will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

destructive configurations. The user must avoid situations in which the QUICC output drivers are enabled into actively driven networks.

The QUICC includes on-chip circuitry to detect the initial application of power to the device. Power-on reset (POR), the output of this circuitry, is used to reset both the system and scan chainlogic. The purpose for applying POR to the scan chaincircuitry is to avoid the possibility of bus contention during power-on. The time required to complete device power-on is power-supply dependent. However, the scan chainTAP controller remains in the test-logic-reset state while POR is asserted. The TAP controller does not respond to user commands until POR is negated.

The QUICC features a low-power stop mode, which is invoked using a CPU instruction called LPSTOP. The interaction of the scan chaininterface with low-power stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller in the test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to V_{CC} or ground.
3. The TMS and TDI pins include on-chip pullup resistors. In low-power stop mode, these two pins should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

8.6 NON-SCAN CHAIN OPERATION

In non-scan chain operation, there are two constraints. First, the TCK input does not include an internal pullup resistor and should not be left unconnected to preclude mid-level inputs. The second constraint is to ensure that the scan chaintest logic is kept transparent to the system logic by forcing TAP into the test-logic-reset controller state, using either of two methods. During power-up, POR forces the TAP controller into this state. After power-up is concluded, TMS must be sampled as a logic one for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{CC}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

2. Testing of the device itself.
3. Observation or modification of the activity on the board.

The test logic is comprised of features that include the boundary scan register and is accessed through the test access port (TAP).

9.7.1 Board Layout

The test equipment interfaces to the board through a test bus, with the tester acting as a test bus master. This bus is comprised of the boundary scan signals and consists of one or several parallel signal paths. Although different architectures are described in the JTAG standard document, this application shows only one serial connection. As shown in Figure 9-22, one data signal loop is created with relevant device. The test data in (TDI) signal enters the device and exits the device as test data out (TDO) at the other end of the shift register. In addition, a clock signal (TCK) and a mode signal (TMS) are distributed to all devices in parallel.

An example of a board designed with the boundary scan path architecture is shown in Figure 9-22. Most devices, like device 2 through device 5, are included in the loop. Other devices such as memories can be tested directly from a microprocessor. Usually, the board will also contain small logic functions or analog ICs that do not contain boundary scan logic.

Once the board is tested with the test equipment, it is typically stored until it is installed in the final system. This storage time is comprised of warehouse and shipping time. How then can the board be retested before it begins to interact with a larger system in the end application? This is of particular importance in telecommunication systems with distributed intelligence, since an error can propagate throughout the entire system if one node is malfunctioning.

Today, almost every board design includes some kind of processor or controller. This controller can be used as the test bus master to perform a board test using an existing boundary scan path. The following paragraphs describe how this concept can be achieved using the QUICC as the board controller.

former, see the DRAM multiplexing scheme in 9.4 Using the QUICC MC68040 Companion Mode.

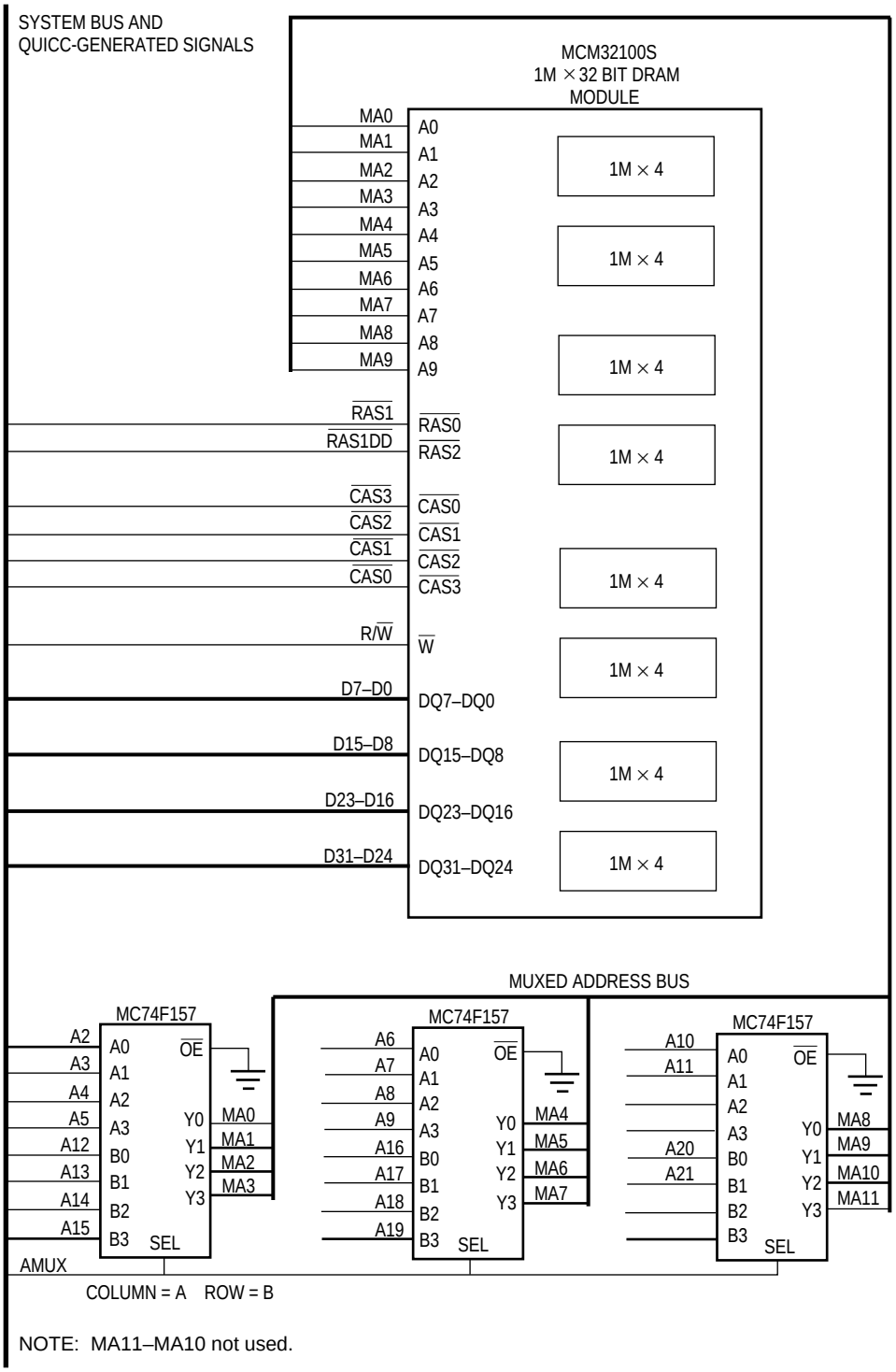
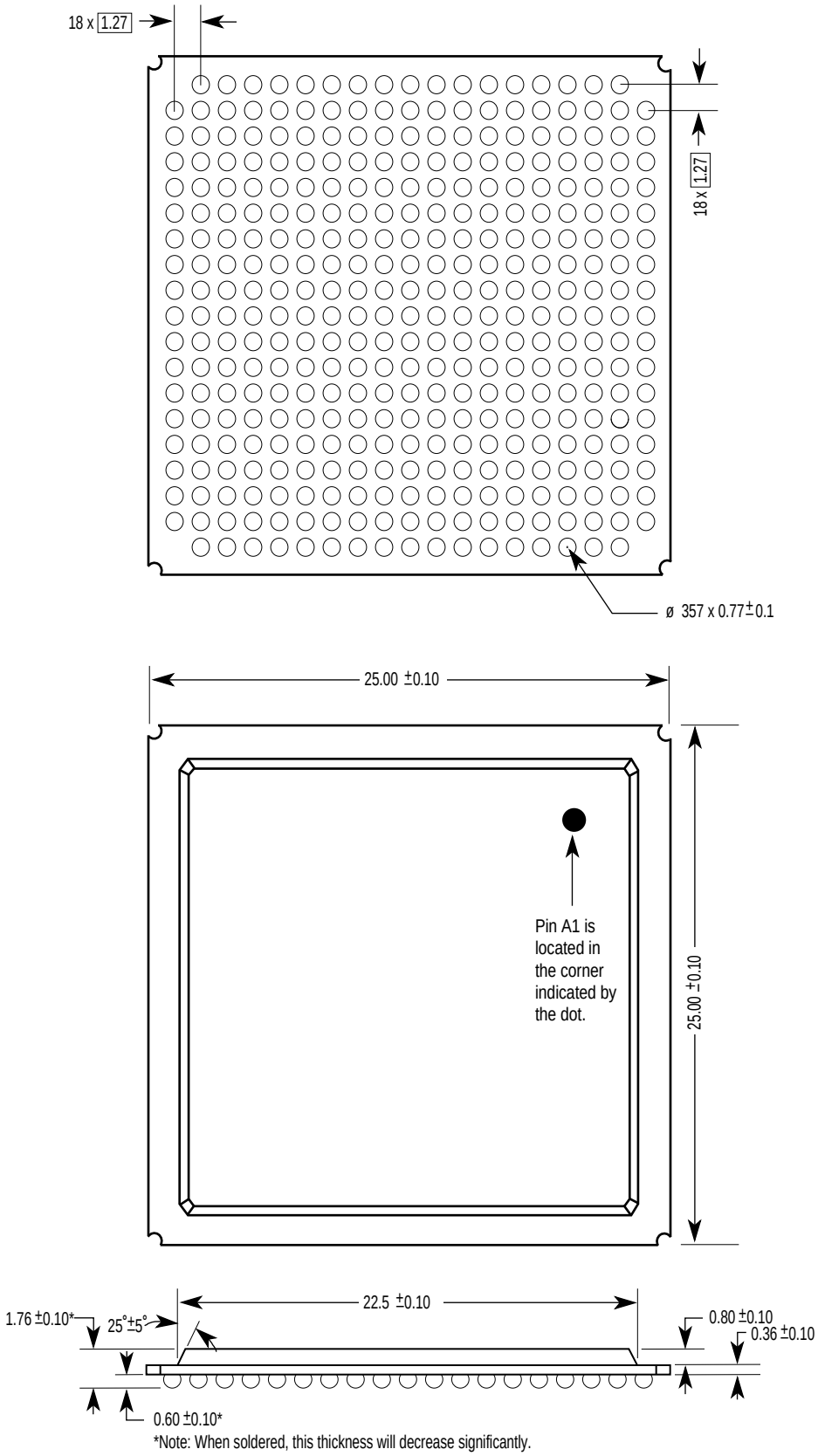
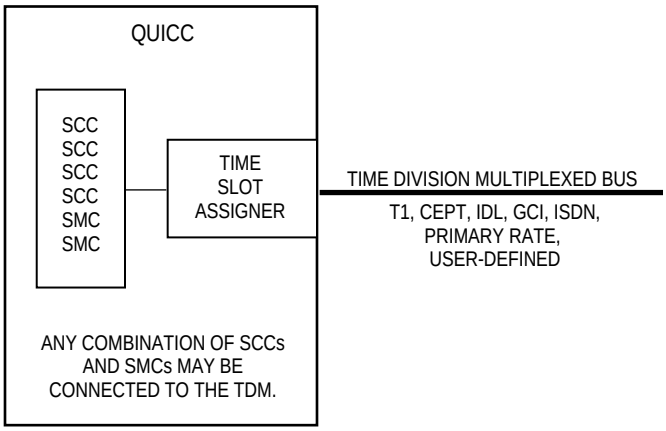


Figure 9-32. 4-Mbyte DRAM Bank—32 Bits Wide

11.7 PACKAGE DIMENSIONS—BGA (ZP SUFFIX)



*Note: When soldered, this thickness will decrease significantly.

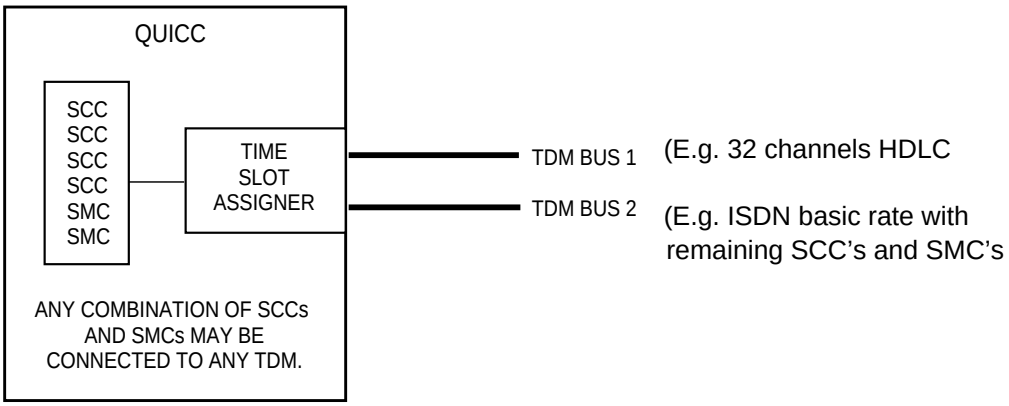


NOTE: Independent receive and transmit clocking, routing, and syncs are supported.

Figure D-3. Serial Channel to TDM Bus Implementation

Figure D-4 shows that the QUICC32 time-slot assigner can support two TDM buses. Each TDM bus can be of a different format—for example, one TDM can be a T1 line, and one can be a CEPT line. Also this technique could be used to bridge frames from basic rate ISDN to a T1/CEPT line, etc.

The QUICC32 can route channels to and from the QMC protocol to the two different TDM buses in any combination.



NOTE: Two TDM buses may be simultaneously supported with the time slot assigner.

Figure D-4. Dual TDM Bus Implementation

Figure D-5 shows a TDM application having one line termination device that extracts clocks and frame sychronization pulses. For T1/E1 line termination, devices exist that perform this function. Alternatively, this can be achieved by a DSP from the Motorola 56K family.

For line termination the QUICC32 is capable of handling up to 32 channels and it may be sufficient to omit the block labeled “Other PCM line devices”. If it is desired to incorporate other PCM line devices in the system as shown in Figure D-5, the QUICC32 can provide strobe signals to other devices that do not have a built-in time slot assigner.

CCITT 1.460 7-97
CLK 7-86
Frame Sync Delay 7-81
GCI 7-80, 7-96
GCIDCL 7-97
Grant Support 7-86
IDL 7-80, 7-90
IDL Interface Example 7-91
IDL Interface Programming 7-95
L1CLKOx 7-97
L1GRx 7-93
L1RCLKx 7-93, 7-96
L1RQx 7-79, 7-93
L1RSYNCx 7-80, 7-93, 7-96
L1RXDx 7-73, 7-93, 7-96
L1STA1 7-73
L1STB1 7-73
L1TXDx 7-73, 7-93, 7-96
Programming SI RAM Entries 7-72
S/T 7-96
SCIT 7-96, 7-98
SCIT Programming 7-98
SCP 7-93
Shadow RAM 7-87
SI RAM 7-68
SI RAM Dynamic Changes 7-75
SI RAM Programming Example 7-75
SMC 7-99
SPI 7-93
Strobes 7-73
TSA 7-170
SI GCI Programming 7-98
SI GCI Support 7-96
SI RAM 7-68
SI RAM Dynamic Changes 7-75
SI RAM Programming Example 7-75
SI Registers 7-77
SIA 7-236
SICMR 7-87
SICR 7-86
SIGMR 7-77
Signal 2-2
Bus Control Signals 2-7
Parity 2-6
Signal Descriptions 2-1
Signals 4-64
16BM 2-6
A 6-48
A26–A0 2-1
A31–A28 2-1
ACK 7-333
Address Bus 2-1
AMUX 2-7, 2-9, 6-48, 6-49
AS 2-8, 2-13, 4-3, 6-30, 6-63, 7-41, 7-44, 7-58
AVEC 2-8, 4-6, 4-41, 6-26, 6-48
AVECO 6-26, 6-48
BCLRI 4-58, 6-25, 6-33
BCLRIID 6-25
BCLRO 2-9, 2-10, 6-25, 6-31, 6-33, 7-44, 7-51, 7-58
BERR 2-10, 4-6, 4-20, 4-41, 7-51
BG 2-9, 2-10, 4-49, 4-53, 6-26, 6-31, 6-32, 7-44
BGACK 2-9, 4-49, 6-26, 6-31, 7-44
BKPT 2-11, 4-31
BKPTO 6-26
BR 2-9, 2-10, 4-49, 4-52, 6-26, 6-31, 6-32, 7-44
BRG 7-101
BRGO 7-108, 7-364
BUSY Signal 7-335
CAS 2-7, 6-48, 6-60, 6-63, 6-67
CASx 2-13
CD 7-63, 7-101, 7-114, 7-118, 7-128, 7-129, 7-130, 7-187, 7-199, 7-219, 7-223, 7-229, 7-233, 7-240, 7-266, 7-358
Chip-Select 2-6
CLK 2-11, 2-13, 7-101
CLK2 7-104
CLK6 7-104
CLKO 2-10
CLKO Power Pins 6-19
CONFIG 2-9, 2-12, 2-14
CONFIG0 2-9
CONFIG1 2-9
CONFIG1/BCLRO/RAS2DD 6-49
CONFIG2 2-12
CS 2-6, 2-10, 6-30, 6-48, 6-67
CS0 9-5
CSO 6-25
CTS 7-63, 7-101, 7-114, 7-118, 7-120, 7-128, 7-129, 7-130, 7-187, 7-

TER 7-22
TFCR 7-125, 7-272, 7-321
TGATE 7-19
TGCR 7-20
Thermal Characteristics 10-2
Timeout 7-308
Timer Block Diagram 7-17
Timer Examples 7-23
Timers 7-17, 9-25
  Cascaded Mode 7-19
  Deriving SWT Timeout 6-36
  General-Purpose Timer 7-17
  RISC Timer Tables 7-11
  Timer Block Diagram 7-17
  Timer Examples 7-23
  TM_BASE 7-13
  Wake-Up Timer 7-151
Time-Slot 1-15, 14
TIN 7-18
TM_BASE 7-13
TMR 7-21
TMS 2-12
TODR 7-121, 7-175, 7-200
Totally Transparent 7-220
TOUT 7-18
TP Bit 5-49
TR Bit 5-49, 5-51, 5-52, 5-53
Trace 5-37, 5-39
Trace Exception 5-39, 5-42, 5-45, 5-46, 5-49, 5-51, 5-56
Trace Mode 5-7
Trace on Instruction Execution 5-59
Transfer Format 7-319
Transition to BDM 5-62
Transmission Line Configuration 7-194
TRANSMIT ABORT REQUEST Command 7-307
Transmit on Demand 7-121
Transmit/Receive Process 7-315
Transmitter 7-275
Transmitter Shortcut 7-275
Transmitting and Receiving 7-208
Transparent 7-115, 7-116, A-1
  4-Bit SYNC 7-223
  Achieving Synchronization in Transparent Mode 7-223
  External Synchronization Signals 7-223
  In-Line Synchronization Pattern 7-223
  PIP 7-338
  Promiscuous 7-220
  SCC Transparent Example 7-233
  Sending Transparent Frames Between QUICCs 7-225
  SMC 7-291
  Totally Transparent 7-220
  Transparent Channel Frame Reception Processing 7-222
  Transparent Channel Frame Transmission Processing 7-221
  Transparent Command Set 7-226
  Transparent Controller 7-220
  Transparent Error-Handling Procedure 7-227
  Transparent Receive Buffer Descriptor 7-228
Transparent Channel Frame Reception Processing 7-222
Transparent Channel Frame Transmission Processing 7-221
Transparent Command Set 7-226, 7-297
Transparent Controller 7-220, 7-291
Transparent CRC 7-112
Transparent Error-Handling Procedure 7-227, 7-298
Transparent Memory Map 7-225
Transparent Mode Register 7-228
Transparent NMSI Example 7-303
Transparent Protocol 7-268
Transparent Receive Buffer Descriptor 7-228
Transparent Reception Processing 7-293
Transparent Transmit Buffer Descriptor 7-230
Transparent TSA Example 7-304
TRAP instruction 5-39
TRIS 2-12, 6-26
TRLXQ 6-74
TRR 7-22
TRST 2-12
TS 6-26, 6-60, 6-63, 6-68, 9-32
TSA 7-170
  Using the TSA 7-195
TSA Transmission Line Configuration 7-195
TXD 7-101, 7-358