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	M68000
Number of Cores/Bus Width	1 Core, 8/16-Bit
Speed	20MHz
Co-Processors/DSP	Communications; RISC CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (46x46)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68302eh20cb1

Dedicated Mode

In this mode, the three interrupt request pins are configured as $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$ to provide dedicated request lines for three external sources at priority levels 1, 6, and 7. Each of these lines may be programmed to be edge-triggered or level-sensitive. In addition to level 4, which is reserved for INRQ interrupts, interrupt priority levels 2, 3, and 5 must not be assigned to external devices in this mode.

All INRQ and EXRQ sources are prioritized within the interrupt controller. The M68000 supports seven priority levels. Level 7, the highest priority level, is nonmaskable. EXRQ sources are given their own separate priority level. Priority level 4 is reserved exclusively for the INRQ sources with all the INRQ sources being further prioritized within this level. If more than one INRQ or EXRQ interrupt request is pending, the interrupt controller presents the highest priority interrupt to the M68000 core through an internal, hidden set of $\overline{IPL2}$–$\overline{IPL0}$ lines.

When the M68000 core executes the interrupt acknowledge cycle, a vector must be provided. If an INRQ source generated the interrupt, the interrupt controller always provides the vector. If an EXRQ source generated the interrupt, three options are available to generate the vector.

First, in most cases the interrupt controller can be configured to provide the vector to the M68000 core. This is usually the preferred solution.

Second, the external peripheral can generate the vector. To assist this process, the interrupt controller can provide up to three interrupt acknowledge outputs ($\overline{IACK7}$, $\overline{IACK6}$, and $\overline{IACK1}$).

Third, the external peripheral can assert the autovector ($\overline{AVEC}$) pin to cause the M68000 to use an autovector. The autovector method maps each interrupt level to a fixed vector location in the exception vector table, regardless of how many interrupt sources exist at that level.

To improve interrupt latency timing, a fast interrupt latency technique is supported in the IMP. On recognition of an interrupt, the IMP can assert the bus clear ($\overline{BCLR}$) signal externally, which can be used to force other bus masters off the bus. This involves the IPA and BCLM bits in the system control register (see 3.8 System Control).

3.2.2 Interrupt Priorities

INRQ and EXRQ interrupts are assigned to an interrupt priority level. INRQ interrupts are also assigned relative priorities within their given interrupt priority level. A fully nested interrupt environment is provided so that a higher priority interrupt is serviced before a lower priority interrupt.

3.2.2.1 INRQ and EXRQ Priority Levels

Seven levels of interrupt priority may be implemented in IMP system designs, with level 7 having the highest priority. INRQ interrupts are assigned to level 4 (fixed). EXRQ interrupts are assigned by the user to any of the remaining six priority levels in normal mode. In dedicated mode, EXRQ interrupts may be assigned to priority levels 7, 6, and 1.

Table 3-10. Bus Arbitration Priority Table

$\overline{\text{BCLR}}$ Ignored BCLM = 0	$\overline{\text{BCLR}}$ Used BCLM = 0	$\overline{\text{BCLR}}$ Ignored BCLM = 1	$\overline{\text{BCLR}}$ Used BCLM = 1
$\overline{\text{BR}}$ Pin SDMA IDMA M68000 Interrupts M68000	SDMA IDMA BR Pin M68000 Interrupts M68000	$\overline{\text{BR}}$ Pin SDMA M68000 Interrupts IDMA[4] M68000	SDMA M68000 Interrupts IDMA[4] BR Pin M68000

NOTES:
1. The SDMA on a given IMP always has a higher priority than the IDMA on that IMP.
2. This table assumes the M68000 core is not in disable CPU mode. In disable CPU mode, the SDMA and IDMA make requests to the M68000 bus when they wish to become bus masters.
3. "BCLR Used" means that the BCLR pin is used externally to force the external bus master off the bus, even though its priority is still the highest in the system from the standpoint of the IMP bus arbiter.
4. The bus arbitration priority for IDMA in the two rightmost columns of the table applies only to the case when the IDMA request is internally generated; for the cases of external request, the bus arbitration priority of IDMA is right below that of SDMA.

The IMP bus arbiter also supports an M68000 core low-interrupt latency option. When the M68000 core processor has an unmasked interrupt request, it asserts an internal interrupt pending signal (IPEND). The bus arbiter uses this signal according to BCLM in the SCR to assert external ($\overline{\text{BCLR}}$) and internal bus-clear (IBCLR) signals. These bus-clear signals allow the M68000 core to eliminate long latencies potentially associated with an external bus master or the IDMA, respectively.

The external $\overline{\text{BCLR}}$ is asserted whenever 1) one of the SDMA channels requests the bus when the IDMA is not the bus master or 2) the M68000 core has an unmasked pending interrupt request, provided BCLM in the SCR is set. In this case, $\overline{\text{BCLR}}$ will be asserted until the interrupt priority active (IPA) bit in the SCR is cleared. To implement this feature, $\overline{\text{BCLR}}$ would be used to force external devices to release bus ownership.

IBCLR to the IDMA is asserted whenever 1) an external bus master requests the bus ($\overline{\text{BR}}$ asserted); 2) the M68000 core has an unmasked pending interrupt request, provided BCLM in the SCR is set and the IDMA request is internally generated, and in this case, BCLR will be asserted until IPA is cleared (Note that $\overline{\text{BCLR}}$ could be used to negate $\overline{\text{DREQ}}$ when the IDMA is in external request mode); 3) the M68000 CPU is disabled, and BCLR is asserted.

The IBCLR signal causes the IDMA to release bus ownership at the end of the current operand transfer. IBCLR is not routed to the SDMA channels since they always release bus ownership after one operand transfer.

RMC is issued by the M68000 core and can be used by the internal bus arbiter to delay issuance of BG during read-modify-write cycles. This is controlled by the RMCST bit in the SCR. Otherwise, the MC68000/MC68008 core may be forced off the bus after any bus cycle.

3.8.5.2 External Bus Arbitration

An external bus master may gain ownership of the M68000 bus by asserting the bus request ($\overline{\text{BR}}$) pin. After gaining ownership, it may access the IMP registers or RAM or any system memory address. Chip selects and system control functions, such as the hardware watchdog, continue to operate.

to the DRAM bank. The PAL generates the RAS and CAS lines for the DRAM chips and controls the address multiplexing in the external address buffers. One of the MC68000 chip-select lines can be used as the DRAM bank enable signal, if desired.

The refresh operation is a byte read operation. Thus, $\overline{UDS}$ or $\overline{LDS}$ will be asserted from the MC68302, but not both. A refresh to an odd address will assert $\overline{LDS}$; whereas, a refresh to an even address will assert $\overline{UDS}$.

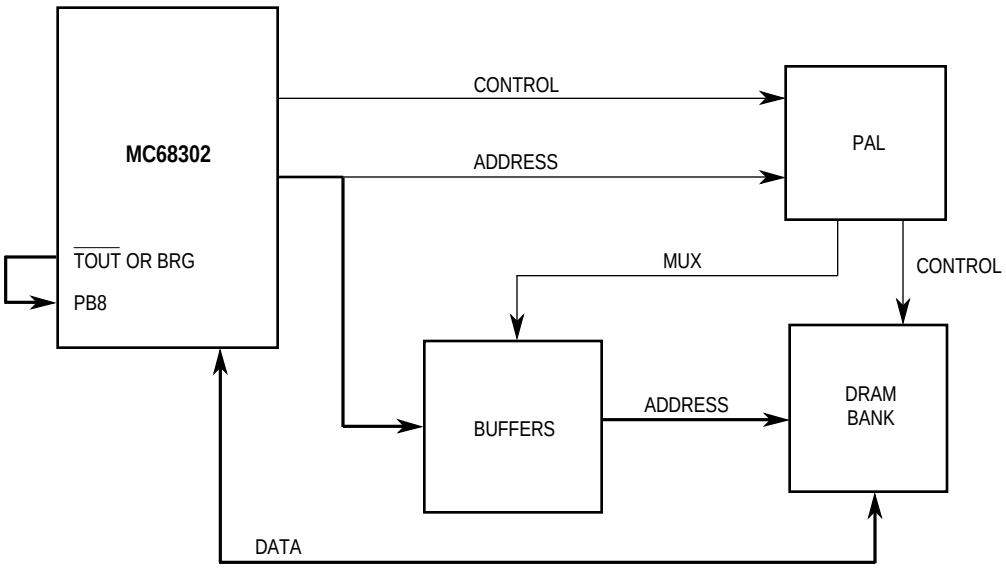


Figure 3-13. DRAM Control Block Diagram

3.10.2 DRAM Refresh Controller Bus Timing

The DRAM refresh controller bus cycles are actually SDMA byte read accesses (see 4.2 SDMA Channels for more details). All timings, signals, and arbitration characteristics of SDMA accesses apply to the DRAM refresh controller accesses. For example, DRAM refresh cycles activate the $\overline{BCLR}$ signal, just like the SDMA. Note that the function code bits may be used to distinguish DRAM refresh cycles from SDMA cycles, if desired.

A bus error on a DRAM refresh controller access causes the $\overline{BERR}$ channel number at offset BASE + $67C to be written with a $0001. This is also the value written if the SCC1 receive SDMA channel experiences a bus error; thus, these two sources cannot be distinguished upon a bus error. The DRAM refresh SDMA channel and SCC1 receive SDMA channel are separate and independent in all other respects.

3.10.3 Refresh Request Calculations

A typical 1-Mbyte DRAM needs one refresh cycle every 15.625 µs. The DRAM refresh controller is configured to execute one refresh cycle per request; thus, the PB8 pin should see a high-to-low transition every 15.625 µs. This is once every 260 cycles for a 16.67-MHz clock. Note that one refresh per request minimizes the speed loss on the SCC channels.

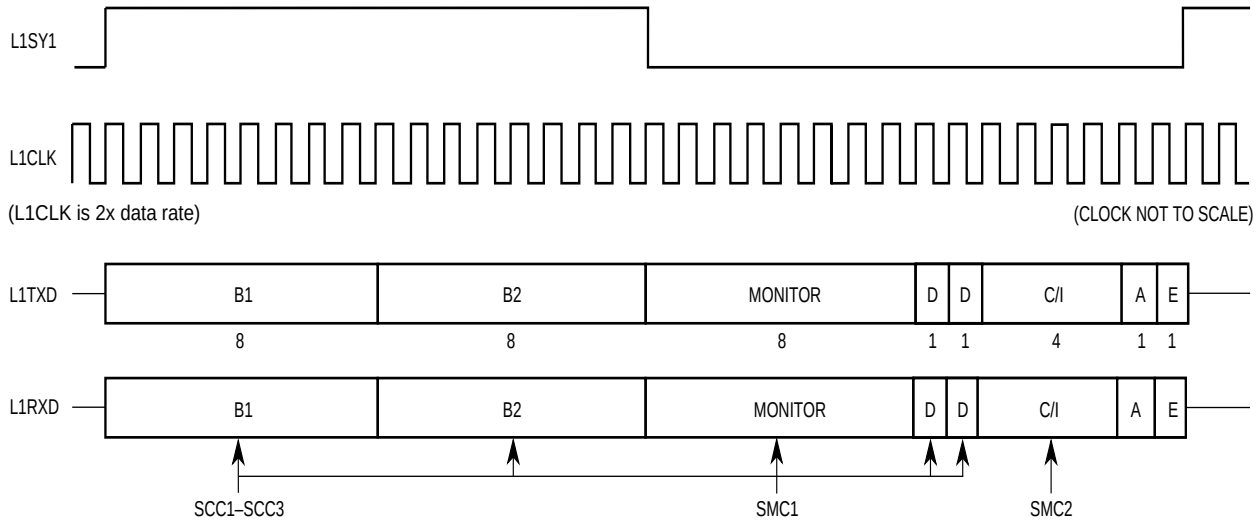


Figure 4-8. GCI Bus Signals

The GCI signals are as follows:

L1CLK	GCI clock; input to the IMP.
L1TXD	GCI transmit data; open drain output.
L1RXD	GCI receive data; input to the IMP.
L1SY1	GCI SYNC signal; input to the IMP.
L1GR	Grant permission to transmit on the D channel; input to the IMP.
SDS1	Serial data strobe 1; output from the IMP.
SDS2	Serial data strobe 2; output from the IMP.
GCIDCL	GCI interface data clock; output from the IMP.

NOTE

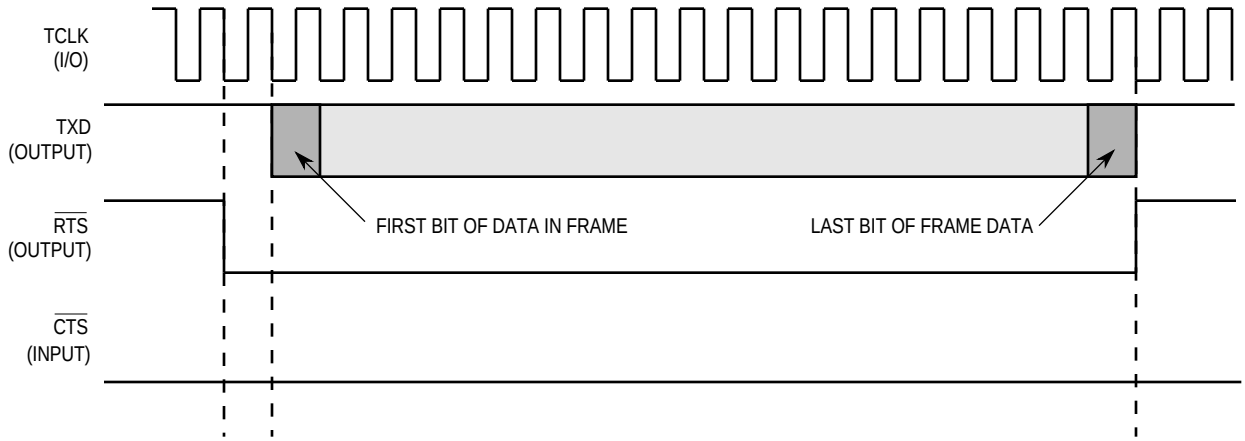
The GCI bus signals, L1TXD and L1RXD, require pull-up resistors in order to ensure proper operation with transceivers.

The GCI bus has five channels:

B1	64-kbps Bearer Channel (8 bits)
B2	64-kbps Bearer Channel (8 bits)
M	64-kbps Monitor Channel (8 bits)
D	16-kbps Signaling Channel (2 bits)
C/I, A, E	48-kbps Command/Indication Channel (6 bits)

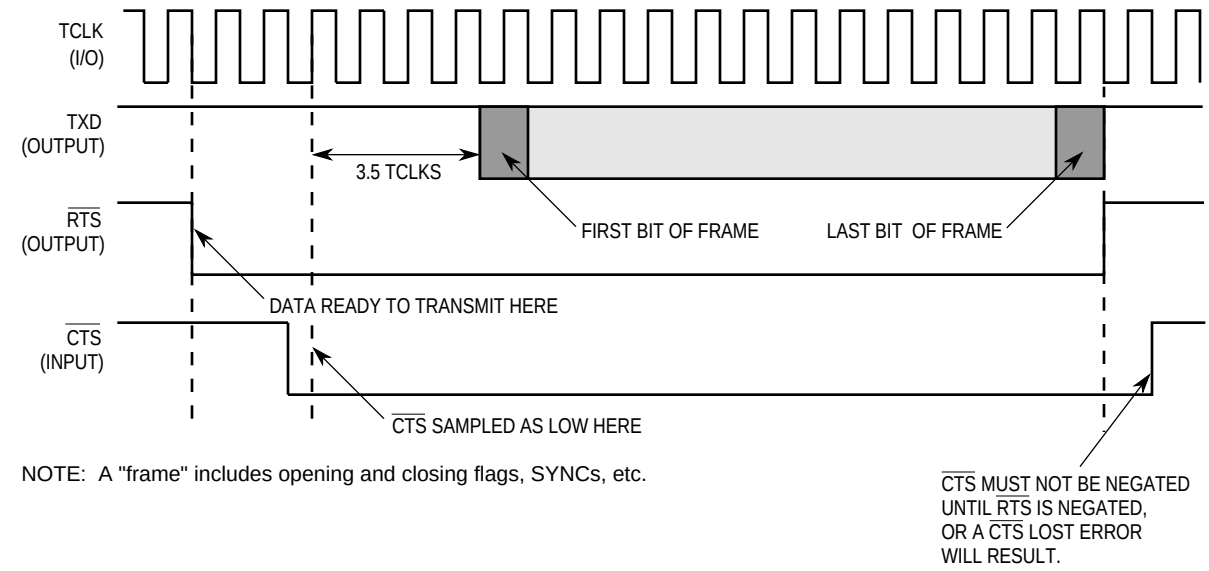
In addition to the 144-kbps ISDN 2B + D channels, GCI provides two channels for maintenance and control functions.

allel I/O lines in the PACNT register. To cause the TXD and $\overline{RTS}$ pins to simply remain high in NMSI1, NMSI2, and NMSI3 modes, use this loopback mode in conjunction with setting the SDIAG1–SDIAG0 bits in the SIMODE register to loopback control.



NOTE: A "frame" includes opening and closing flags in HDLC and SYNCs in BISYNC and DDCMP.

Figure 4-13. Output Delays from RTS Low, Synchronous Protocol



NOTE: A "frame" includes opening and closing flags, SYNCs, etc.

Figure 4-14. Output Delays from $\overline{CTS}$ Low, Synchronous Protocol

If an internal loopback is desired when this SCC is configured to a multiplexed physical interface, then only the SDIAG1–SDIAG0 bits need be set.
When using loopback mode, the clock source for the transmitter and the receiver (as set in the TCS and RCS bits in the SCON register), must be the same. Thus,

for an internal clock, TCS and RCS may both be zero, or, for an external clock, they may both be one. The other two combinations are not allowed in this mode.

NOTE

If external loopback is desired (i.e., external to the MC68302), then the DIAG1–DIAG0 bits should be set for either normal or software operation, and an external connection should be made between the TXD and RXD pins. Clocks may be generated internally, externally, or an internally generated TCLK may be externally connected to RCLK. If software operation is used, the $\overline{\text{RTS}}$, $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ pins need not be externally connected. If normal operation is used, the $\overline{\text{RTS}}$ pin may be externally connected to the $\overline{\text{CD}}$ pin, and the $\overline{\text{CTS}}$ pin may be grounded.

NOTE

Do not use this mode for loopback operation of IDL in the Serial Interface. Instead program the diag bits to Normal Operation, and (1) assert the L1GR pin externally from the S/T chip, or (2) configure the SDIAG1-0 bits in the SIMODE to Internal Loopback or Loopback Control.

10 = Automatic echo

In this mode, the channel automatically retransmits the received data on a bit-by-bit basis. The receiver operates normally, but the transmitter simply retransmits the received data. The $\overline{\text{CD}}$ pin must be asserted for the receiver to receive data, and the $\overline{\text{CTS}}$ line is ignored. The data is echoed out the TXD pin with a few nanosecond delay from RXD. No transmit clock is required, and the ENT bit in the SCC mode register does not have to be set.

NOTE

The echo function may also be accomplished in software by receiving buffers from an SCC, linking them to transmit buffer descriptors, and then transmitting them back out of that SCC.

11 = Software operation (CTS, CD lines under software control)

In this mode, the CTS and CD lines are just inputs to the SCC event (SCCE)and status (SCCS) registers. The SCC controller does not use these lines to enable/disable reception and transmission, but leaves low (i.e., active) in this mode. Transmission delays from RTS low are zero TCLKs (asynchronous protocols) or one TCLK (synchronous protocols).

NOTE

The MC68302 provides several tools for enabling and disabling transmission and/or reception. Choosing the right tool is application and situation dependent. For the receiver, the tools are 1) the empty bit in the receive buffer descriptor, 2) the ENR bit, and 3) the ENTER HUNT MODE command. For the transmitter, the

RCCR, CHARACTER

The UART controller can automatically recognize special characters and generate interrupts. It also allows a convenient method for inserting flow control characters into the transmit stream. See 4.5.11.7 UART Control Characters and Flow Control for more details.

If neither of these capabilities are desired, initialize CHARACTER1 to $8000 and CHARACTER8 to $0000 to disable both functions.

4.5.11.4 UART Programming Model

An SCC configured as a UART uses the same data structure as the other protocols. The UART data structure supports multibuffer operation. The UART may also be programmed to perform address comparison whereby messages not destined for a given programmable address are discarded. Also, the user can program the UART to accept or reject control characters. If a control character is rejected, an interrupt may be generated. The UART enables the user to transmit break and preamble sequences. Overrun, parity, noise, and framing errors are reported using the buffer descriptor (BD) table and/or error counters. An indication of the status of the line (idle) is reported through the status register, and a maskable interrupt is generated upon a status change.

In its simplest form, the UART can function in a character-oriented environment. Each character is transmitted with accompanying stop bits and parity (as configured by the user) and is received into separate one-byte buffers. Reception of each buffer may generate a maskable interrupt.

Many applications may want to take advantage of the message-oriented capabilities supported by the UART using linked buffers to receive or transmit data. In this case, data is handled in a message-oriented environment; users can work on entire messages rather than operating on a character-by-character basis. A message may span several linked buffers. For example, rather than being interrupted after the reception of each character, a terminal driver may want to wait until an end-of-line character has been typed by a user before handling the input data.

As another example, when transmitting ASCII files, the data may be transferred as messages ending on the end-of-line character. Each message could be both transmitted and received as a circular list of buffers without any intervention from the M68000 core. This technique achieves both ease in programming and significant savings in processor overhead.

On the receive side, the user may define up to eight control characters. Each control character may be configured to designate the end of a message (such as end of line) or to generate a maskable interrupt without being stored in the data buffer. This latter option is useful when flow-control characters such as XON or XOFF need to alert the M68000 core, yet do not belong to the message being received. Flow-control characters may also be transmitted at any time.

In the message-oriented environment, the data stream is divided into buffers. However, the physical format of each character (stop bits, parity, etc.) is not altered.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the UART event register will be set when this buffer has been completely filled by the CP, indicating the need for the M68000 core to process the buffer. The RX bit can cause an interrupt.

The following bits contain status information written by the CP after it has finished receiving data in the associated data buffer.

C—Control Character

0 = This buffer does not contain a control character.
1 = This buffer contains a user-defined control character in the last byte location.

A—Address

0 = The buffer contains data only.
1 = When working in nonautomatic multidrop mode (UM1–UM0 = 01), this bit indicates that the first byte of this buffer contains an address byte. The address comparison should be implemented in software. In automatic multidrop mode, this bit indicates that the BD contains a message received immediately following an address recognized in UADDR1 or UADDR2. This address is not written into the receive buffer.

M—Address Match

This bit is meaningful only if the A bit (bit 10) is set and UM1–UM0 = 11 in the UART mode register. Following an address match, this bit defines which address character matched the user-defined address character, enabling the UART to receive the data.

0 = The address-matched user-defined UADDR2
1 = The address-matched user-defined UADDR1

ID—Buffer Closed on Reception of Idles

The buffer was closed due to the reception of the programmable number of consecutive IDLE sequences (defined in MAX_IDL).

Bits 7–6, 2—Reserved for future use.

BR—Break Received

A break sequence was received while receiving data into this buffer.

FR—Framing Error

A character with a framing error was received and is located in the last byte of this buffer. A framing error is detected by the UART controller when no stop bit is detected in the receive data string.

PR—Parity Error

A character with a parity error was received and is located in the last byte of this buffer.

OV—Overrun

A receiver overrun occurred during message reception.

- Detection of Non-Octet Aligned Frames
- Detection of Frames That Are Too Long
- Programmable Flags (0–15) between Successive Frames
- Automatic Retransmission in Case of Collision

4.5.12.1 HDLC Channel Frame Transmission Processing

The HDLC transmitter is designed to work with almost no intervention from the M68000 core. When the M68000 core enables one of the transmitters, it will start transmitting flags or idles as programmed in the HDLC mode register. The HDLC controller will poll the first buffer descriptor (BD) in the transmit channel's BD table. When there is a frame to transmit, the HDLC controller will fetch the data from memory and start transmitting the frame (after first transmitting the user-specified minimum number of flags between frames). When the end of the current BD has been reached and the last buffer in the frame bit is set, the cyclic redundancy check (CRC), if selected, and the closing flag are appended.

Following the transmission of the closing flag, the HDLC controller writes the frame status bits into the BD and clears the ready bit. When the end of the current BD has been reached, and the last bit is not set (working in multibuffer mode), only the ready bit is cleared. In either mode, an interrupt is issued according to the interrupt bit in the BD. The HDLC controller will then proceed to the next BD in the table. In this way, the user may be interrupted after each buffer, after a specific buffer has been transmitted, or after each frame.

To rearrange the transmit queue before the IMP has completed transmission of all buffers, issue the STOP TRANSMIT command. This technique can be useful for transmitting expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller will abort the current frame being transmitted and start transmitting idles or flags. When the HDLC controller is given the RESTART TRANSMIT command, it resumes transmission.

4.5.12.2 HDLC Channel Frame Reception Processing

The HDLC receiver is also designed to work with almost no intervention from the M68000 core. The HDLC receiver can perform address recognition, CRC checking, and maximum frame length checking. The received frame (all fields between the opening and closing flags) is made available to the user for performing any HDLC-based protocol.

When the M68000 core enables one of the receivers, the receiver waits for an opening flag character. When the receiver detects the first byte of the frame, the HDLC controller will compare the frame address against the user-programmable addresses. The user has four 16-bit address registers and an address mask available for address matching. The HDLC controller will compare the received address field to the user-defined values after masking with the address mask. The HDLC controller can also detect broadcast (all ones) addressed frames, if one address register is written with all ones.

If a match is detected, the HDLC controller will fetch the next BD and, if empty, will start to transfer the incoming frame to the BD's associated data buffer starting with the first address byte. When the data buffer has been filled, the HDLC controller clears the empty bit in the BD and generates an interrupt if the interrupt bit in the BD is set. If the incoming frame ex-

ceeds the length of the data buffer, the HDLC controller will fetch the next BD in the table and, if it is empty, will continue to transfer the rest of the frame to this BD's associated data buffer.

During this process, the HDLC controller will check for a frame that is too long. When the frame ends, the CRC field is checked against the recalculated value and is written to the data buffer starting with the first address byte. The data length written to the last BD in the HDLC frame is the length of the entire frame. This enables HDLC protocols that “lose” frames to correctly recognize the frame-too-long condition. The HDLC controller then sets the last buffer in frame bit, writes the frame status bits into the BD, and clears the empty bit. The HDLC controller next generates a maskable interrupt, indicating that a frame has been received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames may be received with only a single shared flag between frames. Also, flags that share a zero will be recognized as two consecutive flags.

4.5.12.3 HDLC Memory Map

When configured to operate in HDLC mode, the IMP overlays the structure shown in Table 4-7 onto the protocol-specific area of that SCC parameter RAM. Refer to 2.8 MC68302 Memory Map for the placement of the three SCC parameter RAM areas and to Table 4-2 for the other parameter RAM values.

Table 4-8. HDLC-Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C SCC Base + 9E SCC Base + A0 # SCC Base + A2 # SCC Base + A4 SCC Base + A6	RCRC_L RCRC_H C_MASK_L C_MASK_H TCRC_L TCRC_H	Word Word Word Word Word Word	Temp Receive CRC Low Temp Receive CRC High Constant ($F0B8 16-Bit CRC, $DEBB 32-Bit CRC) Constant ($XXXX 16-Bit CRC, $20E3 32-Bit CRC) Temp Transmit CRC Low Temp Transmit CRC High
SCC Base + A8 # SCC Base + AA # SCC Base + AC # SCC Base + AE # SCC Base + B0 #	DISFC CRCEC ABTSC NMARC RETRC	Word Word Word Word Word	Discard Frame Counter CRC Error Counter Abort Sequence Counter Nonmatching Address Received Counter Frame Retransmission Counter
SCC Base + B2 # SCC Base + B4	MFLR MAX_cnt	Word Word	Max Frame Length Register Max_Length Counter
SCC Base + B6 # SCC Base + B8 # SCC Base + BA # SCC Base + BC # SCC Base + BE #	HMASK HADDR1 HADDR2 HADDR3 HADDR4	Word Word Word Word Word	User-Defined Frame Address Mask User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address

Initialized by the user (M68000 core).

NOTE

An incorrect initialization of C_MASK may be used to “force” receive CRC errors for software testing purposes. The transmit CRC will not be affected.

4.5.12.4 HDLC Programming Model

The M68000 core configures each SCC to operate in one of four protocols by the MODE1–MODE0 bits in the SCC mode register (SCM). MODE1–MODE0 = 00 selects HDLC mode.

character recognition and stripping is desired to be performed in software. The bit should be set (or reset) within the time taken to receive the following data byte. When this bit is reset, the BCS calculations exclude the latest fully received data byte. When RBCS is set, the BCS calculations continue normally.

0 = Disable receive BCS
1 = Enable receive BCS

SYNF—Transmit SYN1–SYN2 or IDLE between Messages and Control the RTS Pin

0 = Send ones between messages; $\overline{\text{RTS}}$ is negated between messages. The BISYNC controller can transmit ones in both NRZ and NRZI encoded formats.
1 = Send SYN1–SYN2 pairs between messages; $\overline{\text{RTS}}$ is always asserted.

ENC—Data Encoding Format

0 = Non-return to zero (NRZ). A one is a high level; a zero is a low level.
1 = Non-return to zero inverted (NRZI). A one is represented by no change in the level; a zero is represented by a change in the level. The receiver decodes NRZI, but a clock must be supplied. The transmitter encodes NRZI.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.13.10 BISYNC Receive Buffer Descriptor (Rx BD)

- The CP reports information about the received data for each buffer using BD. The Rx BD is shown in Figure 4-32. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:
- Receiving a user-defined control character
- Detecting an error
- Detecting a full receive buffer
- Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	C	B	—	—	—	—	—	DL	PR	CR	OV	CD
OFFSET +2	DATA LENGTH															
OFFSET +4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET + 6																

Figure 4-32. BISYNC Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of this BD.
1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the CP. After it sets this bit, the M68000 core

4.5.13.14 Programming the BISYNC Controllers

There are two general techniques that the software may employ to handle data received by the BISYNC controllers. The simplest way is to allocate single-byte receive buffers, request (in the status word in each BD) an interrupt on reception of each buffer (i.e., byte), and implement the BISYNC protocol entirely in software on a byte-by-byte basis. This simple approach is flexible and may be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is as follows. Multibyte buffers are prepared and linked to the receive buffer table. Software is used to analyze the first (two to three) bytes of the buffer to determine what type of block is being received. When this has been determined, reception can continue without further intervention to the user's software until a control character is encountered. The control character signifies the end of the block, causing the software to revert back to a byte-by-byte reception mode.

To accomplish this, the RCH bit in the BISYNC mask register should initially be set, enabling an interrupt on every byte of data received. This allows the software to analyze the type of block being received on a byte-by-byte basis. After analyzing the initial characters of a block, the user should either set the receiver transparent mode (RTR) bit in the BISYNC mode register or issue the RESET BCS CALCULATION command. For example, if DLE-STX is received, transparent mode should be entered. By setting the appropriate bit in the BISYNC mode register, the BISYNC controller automatically strips the leading DLE from <DLE-character> sequences. Thus, control characters are only recognized when they follow a DLE character. The RTR bit should be cleared after a DLE-ETX is received.

Alternatively, after receiving an SOH, the RESET BCS CALCULATION command should be issued. This command causes the SOH to be excluded from BCS accumulation and the BCS to be reset. Note that the RBCS bit in the BISYNC mode register (used to exclude a character from the BCS calculation) is not needed here since SYNCs and leading DLEs (in transparent mode) are automatically excluded by the BISYNC controller.

After recognizing the type of block above, the RCH interrupt should be masked. Data reception then continues without further interruption of the M68000 core until the end of the current block is reached. This is defined by the reception of a control character matching that programmed in the receive control characters table.

The control characters table should be set to recognize the end of the block as follows:

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next Entry	1	X	X

After the end of text (ETX), a BCS is expected; then the buffer should be closed. Hunt mode should be entered when line turnaround occurs. ENQ characters are used to abort transmis-

By setting its SCC mode register (SCM), any of the SCC channels may be configured to function as a DDCMP controller. The DDCMP link can be either synchronous (by programming the MODE1–MODE0 bits of the SCC mode register to DDCMP) or asynchronous (by programming the MODE1–MODE0 bits of the SCC mode register to asynchronous and setting the DDCMP bit in the UART mode register). The DDCMP controller handles the basic functions of the DDCMP protocol in both cases.

The SCC in DDCMP mode can work in either IDL, GCI, PCM highway, or NMSI interfaces. When the SCC is used with a modem interface (NMSI), the serial outputs are connected directly to the external pins. The modem interface uses seven dedicated pins: transmit data (TXD), receive data (RXD), receive clock (RCLK), transmit clock (TCLK), carrier detect (CD), clear to send (CTS), and request to send (RTS). Other modem lines can be supported through the parallel I/O pins.

The DDCMP controller consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to the other SCCs. Each clock can be supplied either from the baud rate generator or externally. More information on the baud rate generator is available in 4.5.2 SCC Configuration Register (SCON).

The DDCMP controller key features are as follows:

- Synchronous or Asynchronous DDCMP Links Supported
- Flexible Data Buffers
- Four Address Comparison Registers with Mask
- Automatic Frame Synchronization
- Automatic Message Synchronization by Searching for SOH, ENQ, or DLE
- CRC16 Generation/Checking
- NRZ/NRZI Data Encoding
- Maintenance of Four 16-Bit Error Counters

4.5.14.1 DDCMP Channel Frame Transmission Processing

The DDCMP transmitter is designed to work with almost no intervention from the M68000 core (see Figure 4-35).

When the M68000 core enables the DDCMP transmitter and the link is synchronous, it starts transmitting SYN1–SYN2 pairs (programmed in the data synchronization register) or IDLEs as determined in the DDCMP mode register. The DDCMP controller polls the first buffer descriptor (BD) in the channel's transmit BD table. When there is a message to transmit, the DDCMP controller fetches the data from memory and starts transmitting the message (after first transmitting the SYN1–SYN2 pair when the link is synchronous).

following message reception. Bit 15 determines whether the M68000 core or the CP may currently access the BD.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M6800 core is free to examine or write to any fields of the BD.

1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the DDCMP controller. The M68000 core should not write to any fields of this BD after it sets this bit. Note that the empty bit will remain set while the DDCMP controller is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.

1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.

1 = This is the last BD in the Rx BD table. After this buffer has been used, the DDCMP controller places incoming data into the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been closed.

1 = The RBD or RBK bits in the DDCMP event register will be set when this buffer has been closed by the DDCMP controller, which can cause interrupts.

The following status bits are written by the DDCMP controller after it has finished receiving data in the associated data buffer.

L—Last in Message

0 = The buffer is not the last in a message.

1 = The buffer is the last in a message.

H—Header in Buffer

0 = The buffer does not contain a message header.

1 = The buffer contains a message header.

NOTE

To correctly identify buffers containing headers, the buffer size should be eight or more bytes in length so that the header will fit in a single buffer.

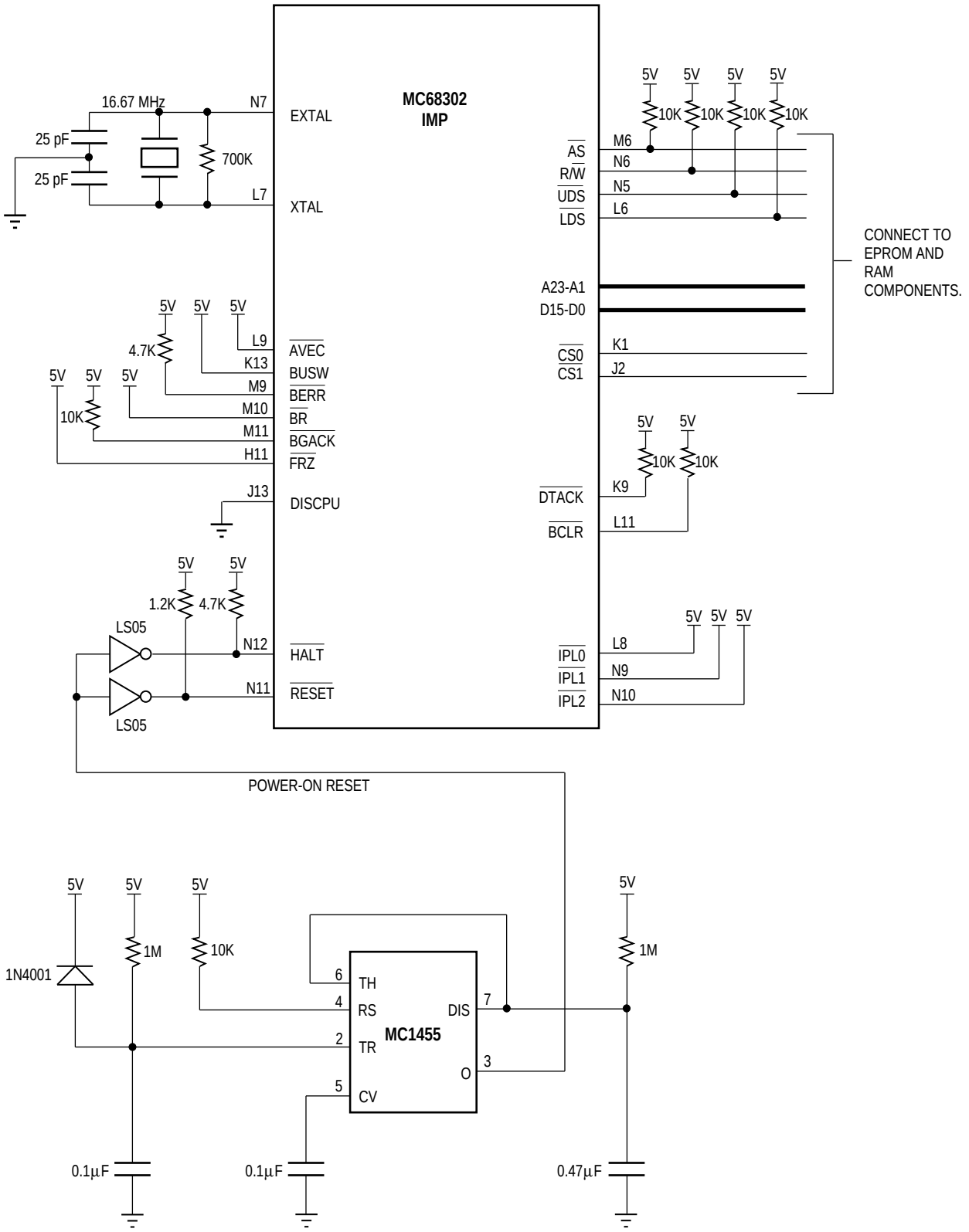


Figure D-1. MC68302 Minimum System Configuration (Sheet 1 of 2)

```
          CMPI.W    #0,D2             ;If they are set
          BNE.B     TX_ INT           ;Handle Transmitter's interrupt
* The handling of other events, e.g., CTS, CD IDL, BSY, is left  to
* the users as desired.
OthrlNT   MOVE.W #$2000,1SR           ;Clear SCC1 bit in ISR
          RTE

***************Receiver portion of SCC1 interrupt routine***************
* This routine handles received (nonempty) BD: set data length = 0,
* clear status bits, set empty =1, and update PRD
* Clear the identified events as soon as possible, so no lost
* events occur during the interrupt handling
RX_INT    MOVE.B    #9,SCCE1          ;Clear RXF and RXB in SCCE1
* While Not-Empty continue to process the next Rx BD, Else Exit.
NxtPRD    BTST.B    #EMPTY,ST_BD(A3)  ;Test PRD -, Empty Bit
*         BNE.B     EXIT_RX           ;Don't need to process if the
*                                     ;Rx BD is still empty.
*** Check status in RXBD for erratic events ***
* If status bits are all 0 then continue, else SHUTDOWN the receiving
* process. This in turn shuts down the whole program, since all of
* Rx BDs will soon be unavailable (all BDs Empty = 8). Thus, the
* status of this BD will be saved for examination later.
          CMPI.B    #0,SS_BD(A3)      ;Check status bits
          BNE.B     EXIT_RX
* Status bits are all 0
          CLR.W     LN_BD(A3)         ;data length = 0
          CLR.B     SS_BD(A3)         ;Clear out all status bits
          BSET.B    #EMPTY,ST_BD(A3)  ;Empty = 1
          BTST.B    #WRAP,ST_BD(A3)   ;Test Wrap bit
          BNE.B     Wrap_R
          ADDQ.W    #SZ_BD,A3         ;Increment PRD to next BD
          BRA.B     NxtPRD            ;Back to while loop
Wrap_R    LEA.L     RXBD_01,A3        ;Wrap back to the first Rx BD
          BRA.B     NxtPRD            ;Back to the while loop
EXIT_RX   JMP       CK_TX             ;Exit receiver potion of the handler

***************Conflrmer portion of SCC1 Interrupt routine ***************
* This routine handles transmitted (Not-ready) BD: set data length = 0,
* clear status bits, set ready = 1, and update CTD.
* Same as the Rx Interrupt handler, the first thing to do is to
* clear the identified events
TX_INT    MOVE.B    #$1 2,SCCE1       ;Clear TXF and TXB in SCCE1
* While Not-Ready continue to process the next Rx BD, Else Exit.
* The Ready bit should be cleared by the CP
NxtCTD    BTST.B    #READY,ST BD(A1)  ;Test PRD-> Ready Bit
          BNE.B     EXIT_TX           ;Don't need to process if the
*                                     ;Tx BD is Ready.
* Check data length, length must be > 0 to continue the confirming process
          CMPI.W    #0,LN_BD(A1)      ;Test CTD –> data length
          BEQ.B     EXIT_TX
          CLR.W     LN_BD(A1)         ;data length = 0

***Check status in TXBD for erratic events ***
* If status bits are all 0 then continue, else SHUTDOWN the confirming
* process. This in turn shuts down the whole program, since soon
```

The IDL bus connects one IDL master to one or more IDL slaves. The IDL bus timing is driven by the master device.

The bus signals are as follows:

- CLOCK — always provided from the master to the slave. It provides the bit timing for the data traveling across the IDL.
- SYNC —provides the framing for the IDL. The SYNC occurs once per 125-µsec frame and is active high one full clock cycle in the bit immediately preceding the data transaction.
- TXDATA — drives the data from one chip to another. The line is in high impedance when no data transaction occurs.
- RXDATA — the input line that receives data from the TXDATA of another part.

D.6.6 IMP/IDL Interconnection

The MC68302 directly connects to the IDL bus with no glue logic. The MC68302 is an IDL slave (accepts IDL timing from the bus). In the application described, the IDL master device is the MC145475 S/T interface chip (see Figure D-14).

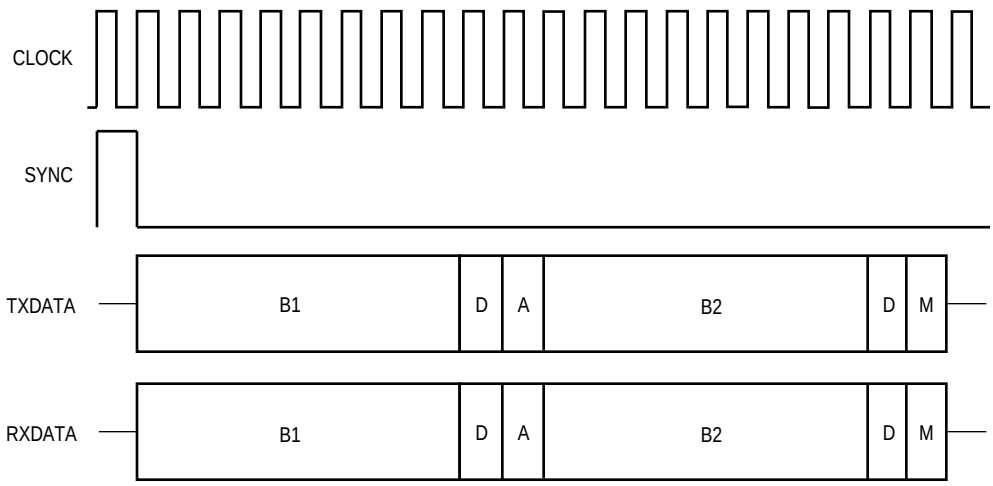


Figure D-13. IDL Frame Structure

stream with the L bit in all Tx BDs cleared, then the byte alignment timing will remain constant.

D.8.11 Initializing Transparent Mode

Full examples of the assembler code required to initialize the HDLC and UART protocols are given in D.3 MC68302 Buffer Processing and Interrupt Handling and D.4 Configuring A Uart on the MC68302. A transparent mode initialization follows the same flow as these subsections except that different values would be used. The HDLC and UART examples also show writing of the BAR and full configuration of the interrupt controller to allow SCC interrupts, etc., which are not duplicated here.

The following example shows a step-by-step list of the SCC-related registers as they would be initialized to create a transparent SCC2 channel in the NMSI mode. The registers in this example are configured for external loopback with TCLK2 externally connected to RCLK2, TXD2 externally connected to RXD2, $\overline{\text{CTS2}}$ a don't care, and $\overline{\text{RTS2}}$ externally connected to $\overline{\text{CD2}}$ (sync). The functionality of this configuration is the same as that shown in Figure D-28. This example may be easily checked on the ADS302 board, either with the menu interface software already on the ADS302 board or with user-written software downloaded to the ADS302 board. The external connections can be made by placing three jumper cables on row B of the serial bus connector P8: B5-to-B6, B7-to-B8, and B10-to-B11.

1. To use SCC2 in the NMSI mode, we need to chose NMSI2 pins instead of parallel I/O pins. To do this, we write a one to the PACNT register in every bit position that we want an SCC pin to be active. For this example, we will assume all seven NMSI2 pins are active.
   PACNT = $xx7F
2. The SIMODE register is set to its default setting. This configuration chooses NMSI mode on all three SCCs. Actually, all we need is that NMSI mode be selected for SCC2.
   SIMODE = $0000
3. The SCON register configures the clocking options. Here we chose to generate about a 65-kHz clock on the TCLK2 pin with the internal baud rate generator. RCLK2 will take its input externally; thus, we connect the TCLK2 pin externally to RCLK2. SCON2 = $1200
4. The setting shown for SCM2 sets the EXSYN and NTSYN bits, sets the DIAG1-DIAG0 bits for software operation, and sets the protocol to BISYNC (which is actually transparent since the NTSYN bit is set).
   Since we are implementing an external loopback with the MC68302, the DIAG1-DIAG0 bits are *not set* for loopback mode. Setting the DIAG1-DIAG0 bits for loopback mode causes *internal* loopback. (To implement an internal loopback, externally connect only $\overline{\text{RTS2}}$ to $\overline{\text{CD2}}$ (sync), set SCON2 to $0200, and SCM2 to $6013. Later, the very last step is to set SCM2 to $601 F. With internal loopback, RCLK and TCLK should be directly supplied with the same clock source—either both from the internal baud rate generators or both from the same externally generated clock source.)
   SCM2 = $6033
5. The DSR2 does not need to be written and can be left at its default value since we are

Table E-1 (c). SCCx Register Set

Initialized by User	Offset Hex	Name	
	00	Reserved	
Yes	02	SCC Configuration Register (SCON)	
Yes	04	SCC Mode Register (SCM)	
Yes	06	SCC Data Synchronization Register (DSR)	
Yes	08	Event Register (SCCE)	Reserved
Yes	0A	Mask Register (SCCM)	Reserved
	0C	Status Register (SCCS)	Reserved
	0E	Reserved	

NOTE: The offset is from the MC68302 base address + ($880 for SCC1, $890 for SCC2, or $8A0 for SCC3).

Table E-1 (d). General Registers (Only One Set)

Initialized by User	Offset Hex	Name	
	860	Command Register (CR)	Reserved
Yes	8B2	Serial Interface Mask Register (SIMASK)	
Yes	8B4	Serial Interface Mask Register (SIMODE)	

NOTE: The offset is from the MC68302 base address.

E.3.1.1 COMMUNICATIONS PROCESSOR (CP) REGISTERS. The CP has one set of three registers that configure the operation of the serial interface for all three SCCs. These registers are discussed in the following paragraphs.

E.3.1.1.1 Command Register (CR). The command register is an 8-bit register located at offset $860 (on D15-D8 of a 16-bit data bus). This register is used to issue commands to the CP. The user should set the FLG bit when a command is written to the command register. The CP clears the FLG bit during command processing to indicate that it is ready for the next command. Reserved bits in registers should be written as zeros.

7	6	5	4	3	2	1	0
RST	GCI	OPCODE		—	CH. NUM.		FLG

RST—Software Reset Command (set by the user and cleared by the CP)

0 = No software reset command issued or cleared by CP during software reset sequence.
1 = Software reset command (FLG bit should also be set if it is not already set).

GCI—GCI Commands

0 = Normal operation.
1 = The OPCODE bits are used for GCI commands (user should set CH. NUM. to 10 and FLG to 1).

E