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	16MHz
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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68302pv16c

PREFACE

The complete documentation package for the MC68302 consists of the M68000PM/AD, *MC68000 Family Programmer's Reference Manual,* MC68302UM/AD, *MC68302 Integrated Multiprotocol Processor User's Manual,* and the MC68302/D, *MC68302 Integrated Multiprotocol Processor Product Brief*.

The *MC68302 Integrated Multiprotocol Processor User's Manual* describes the programming, capabilities, registers, and operation of the MC68302; the *MC68000 Family Programmer's Reference Manual* provides instruction details for the MC68302; and the *MC68302 Low Power Integrated Multiprotocol Processor Product Brief* provides a brief description of the MC68302 capabilities.

This user's manual is organized as follows:

Section 1	General Description
Section 2	MC68000/MC68008 Core
Section 3	System Integration Block (SIB)
Section 4	Communications Processor (CP)
Section 5	Signal Description
Section 6	Electrical Characteristics
Section 7	Mechanical Data And Ordering Information
Appendix B	Development Tools and Support
Appendix C	RISC Microcode from RAM
Appendix D	MC68302 Applications
Appendix E	SCC Programming Reference
Appendix F	Design Checklist

ELECTRONIC SUPPORT:

The Technical Support BBS, known as AESOP (Application Engineering Support Through On-Line Productivity), can be reach by modem or the internet. AESOP provides commonly asked application questons, latest device errata, device specs, software code, and many other useful support functions.

Modem: Call 1-800-843-3451 (outside US or Canada 512-891-3650) on a modem that runs at 14,400 bps or slower. Set your software to N/8/1/F emulating a vt100.

Internet: This access is provided by telneting to pirs.aus.sps.mot.com [129.38.233.1] or through the World Wide Web at http://pirs.aus.sps.mot.com.

— Sales Offices —

For questions or comments pertaining to technical information, questions, and applications, please contact one of the following sales offices nearest you.

LIST OF FIGURES

The MC68302 can also be used in applications such as board-level industrial controllers performing real-time control applications with a local control bus and an X.25 packet network connection. Such a system provides the real-time response to a demanding peripheral while permitting remote monitoring and communication through an X.25 packet network.

1.2 FEATURES

The features of the IMP are as follows:

- On-Chip HCMOS MC68000/MC68008 Core Supporting a 16- or 8-Bit M68000 Family-System
- IB Including:
  - —Independent Direct Memory Access (IDMA) Controller with Three Handshake Signals: $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$.
  - —Interrupt Controller with Two Modes of Operation
  - —Parallel Input/Output (I/O) Ports, Some with Interrupt Capability
  - —On-Chip 1152-Byte Dual-Port RAM
  - —Three Timers Including a Watchdog Timer
  - —Four Programmable Chip-Select Lines with Wait-State Generator Logic
  - —Programmable Address Mapping of the Dual-Port RAM and IMP Registers
  - —On-Chip Clock Generator with Output Signal
  - —System Control:
    - Bus Arbitration Logic with Low-Interrupt Latency Support
    - System Status and Control Logic
    - Disable CPU Logic (M68000)
    - Hardware Watchdog
    - Low-Power (Standby) Modes
    - Freeze Control for Debugging
    - DRAM Refresh Controller
- CP Including:
  - —Main Controller (RISC Processor)
  - —Three Independent Full-Duplex Serial Communications Controllers (SCCs)
  - —Supporting Various Protocols:
    - High-Level/Synchronous Data Link Control (HDLC/SDLC)
    - Universal Asynchronous Receiver Transmitter (UART)
    - Binary Synchronous Communication (BISYNC)
    - Synchronous/Asynchronous Digital Data Communications Message Protocol (DDCMP)
    - Transparent Modes
    - V.110 Rate Adaption
  - —Six Serial DMA Channels for the Three SCCs
  - —Flexible Physical Interface Accessible by SCCs Including:
    - Motorola Interchip Digital Link (IDL)
    - General Circuit Interface (GCI, also known as IOM[3]-2)
    - Pulse Code Modulation (PCM) Highway Interface

[3] IOM is a trademark of Siemens AG

Nonmultiplexed Serial Interface (NMSI) Implementing Standard Modem Signals
—SCP for Synchronous Communication
—Two Serial Management Controllers (SMCs) To Support IDL and GCI Auxiliary Channels

1.3 MC68302 SYSTEM ARCHITECTURE

Most general-purpose microprocessor-based systems use an architecture that interfaces all peripheral devices directly onto a single microprocessor bus (see Figure 1-2).

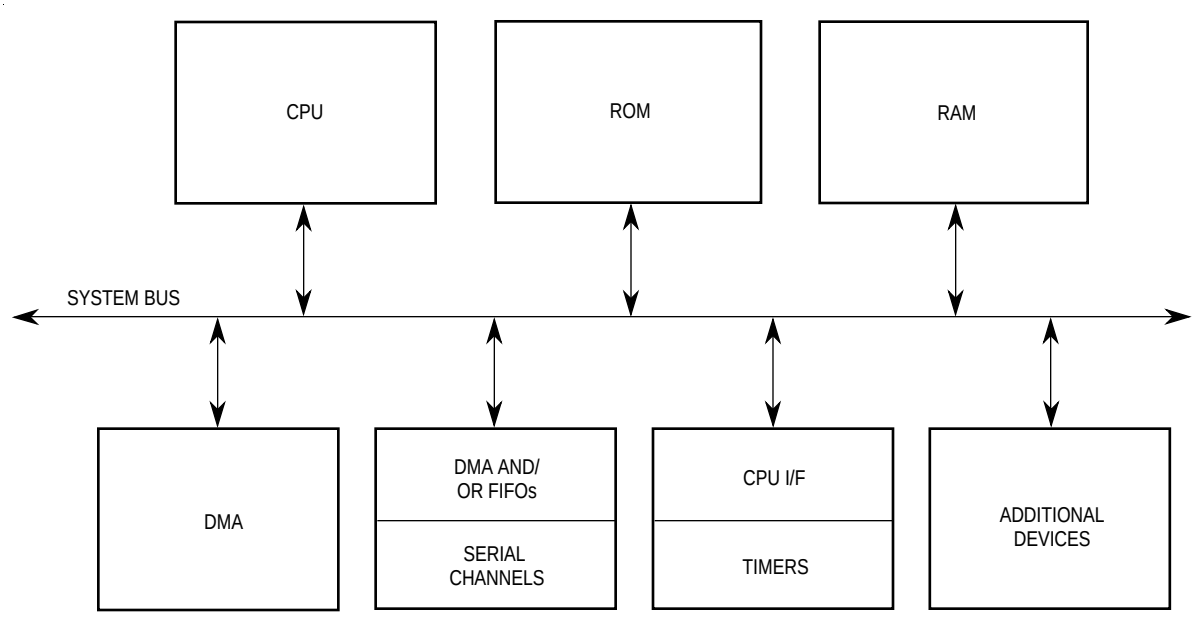


Figure 1-2. General-Purpose Microprocessor System Design

The MC68302 microprocessor architecture is shown in Figure 1-3. In this architecture, the peripheral devices are isolated from the system bus through a dual-port memory. Various parameters and counters and all memory buffer descriptor tables reside in the dual-port RAM. The receive and transmit data buffers may be located in the on-chip RAM or in the off-chip system RAM. Six DMA channels are dedicated to the six serial ports (receive and transmit for each of the three SCC channels). If data for an SCC channel is programmed to be located in the external RAM, the CP will program the corresponding DMA channel for the required accesses, bypassing the dual-port RAM. If data resides in the dual-port RAM, then the CP accesses the RAM with one clock cycle and no arbitration delays.

Table 2-9. Internal Registers

Base + 840	TMR1	16	Timer	Timer Unit 1 Mode Register	0000
Base + 842	TRR1	16	Timer	Timer Unit 1 Reference Register	FFFF
Base + 844	TCR1	16	Timer	Timer Unit 1 Capture Register	0000
Base + 846	TCN1	16	Timer	Timer Unit 1 Counter	0000
Base + 848	RES	8	Timer	Reserved	
! Base + 849	TER1	8	Timer	Timer Unit 1 Event Register	00
Base + 84A	WRR	16	WD	Watchdog Reference Register	FFFF
Base + 84C	WCN	16	WD	Watchdog Counter	0000
Base + 84E	RES	16	Timer	Reserved	
Base + 850	TMR2	16	Timer	Timer Unit 2 Mode Register	0000
Base + 852	TRR2	16	Timer	Timer Unit 2 Reference Register	FFFF
Base + 854	TCR2	16	Timer	Timer Unit 2 Capture Register	0000
Base + 856	TCN2	16	Timer	Timer Unit 2 Counter	0000
Base + 858	RES	8	Timer	Reserved	
! Base + 859	TER2	8	Timer	Timer Unit 2 Event Register	00
Base + 85A	RES	16	Timer	Reserved	
Base + 85C	RES	16	Timer	Reserved	
Base + 85E	RES	16	Timer	Reserved	
Base + 860	CR	8	CP	Command Register	00
Base + 861 • • • Base + 87F				Reserved (Not Implemented)	
Base + 880	RES	16	SCC1	Reserved	
Base + 882	SCON1	16	SCC1	SCC1 Configuration Register	0004
Base + 884	SCM1	16	SCC1	SCC1 Mode Register	0000
Base + 886	DSR1	16	SCC1	SCC1 Data Sync. Register	7E7E
! Base + 888	SCCE1	8	SCC1	SCC1 Event Register	00
Base + 889	RES	8	SCC1	Reserved	
Base + 88A	SCCM1	8	SCC1	SCC1 Mask Register	00
Base + 88B	RES	8	SCC1	Reserved	
Base + 88C	SCCS1	8	SCC1	SCC1 Status Register	00
Base + 88D	RES	8	SCC1	Reserved	
Base + 88E	RES	16	SCC1	Reserved	
Base + 890	RES	16	SCC2	Reserved	
Base + 892	SCON2	16	SCC2	SCC2 Configuration Register	0004
Base + 894	SCM2	16	SCC2	SCC2 Mode Register	0000
Base + 896	DSR2	16	SCC2	SCC2 Data Sync. Register	7E7E
! Base + 898	SCCE2	8	SCC2	SCC2 Event Register	00
Base + 899	RES	8	SCC2	Reserved	
Base + 89A	SCCM2	8	SCC2	SCC2 Mask Register	00
Base + 89B	RES	8	SCC2	Reserved	
Base + 89C	SCCS2	8	SCC2	SCC2 Status Register	00
Base + 89D	RES	8	SCC2	Reserved	
Base + 89E	RES	16	SCC2		

The DAPR contains 24 (A23–A0) address bits of the destination operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA write cycle, the address on the master address bus is driven from this register. The DAPR may be programmed by the DAPI bit to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if overflow occurs. For example, if a register contains $00FFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one or two depending on the DSIZE bit and the starting address.

3.1.2.4 Function Code Register (FCR)

The FCR is an 8-bit register.

7	6–4	3	2–0
1	DFC	1	SFC

The SFC and the DFC bits define the source and destination function code values that are output by the IDMA and the appropriate address registers during an IDMA bus cycle. The address space on the function code lines may be used by an external memory management unit (MMU) or other memory-protection device to translate the IDMA logical addresses to proper physical addresses. The function code value programmed into the FCR is placed on pins FC2–FC0 during a bus cycle to further qualify the address bus value.

NOTE

This register is undefined following power-on reset. The user should always initialize it and should not use the function code value “111” in this register.

3.1.2.5 Byte Count Register (BCR)

This 16-bit register specifies the amount of data to be transferred by the IDMA; up to 64K bytes (BCR = 0) is permitted. This register is decremented once for each byte transferred successfully. BCR may be even or odd as desired. DMA activity will terminate as soon as this register reaches zero. Thus, an odd number of bytes may be transferred in a 16-bit operand scenario.

3.1.2.6 Channel Status Register (CSR)

The CSR is an 8-bit register used to report events recognized by the IDMA controller. On recognition of an event, the IDMA sets its corresponding bit in the CSR (regardless of the INTE and INTN bits in the CMR). The CSR is a memory-mapped register which may be read at any time. A bit is cleared by writing a one and is left unchanged by writing a zero. More than one bit may be cleared at a time, and the register is cleared at reset.

7–4	3	2	1	0
RESERVED	DNS	BES	BED	DONE

Bits 7–4—These bits are reserved for future use.

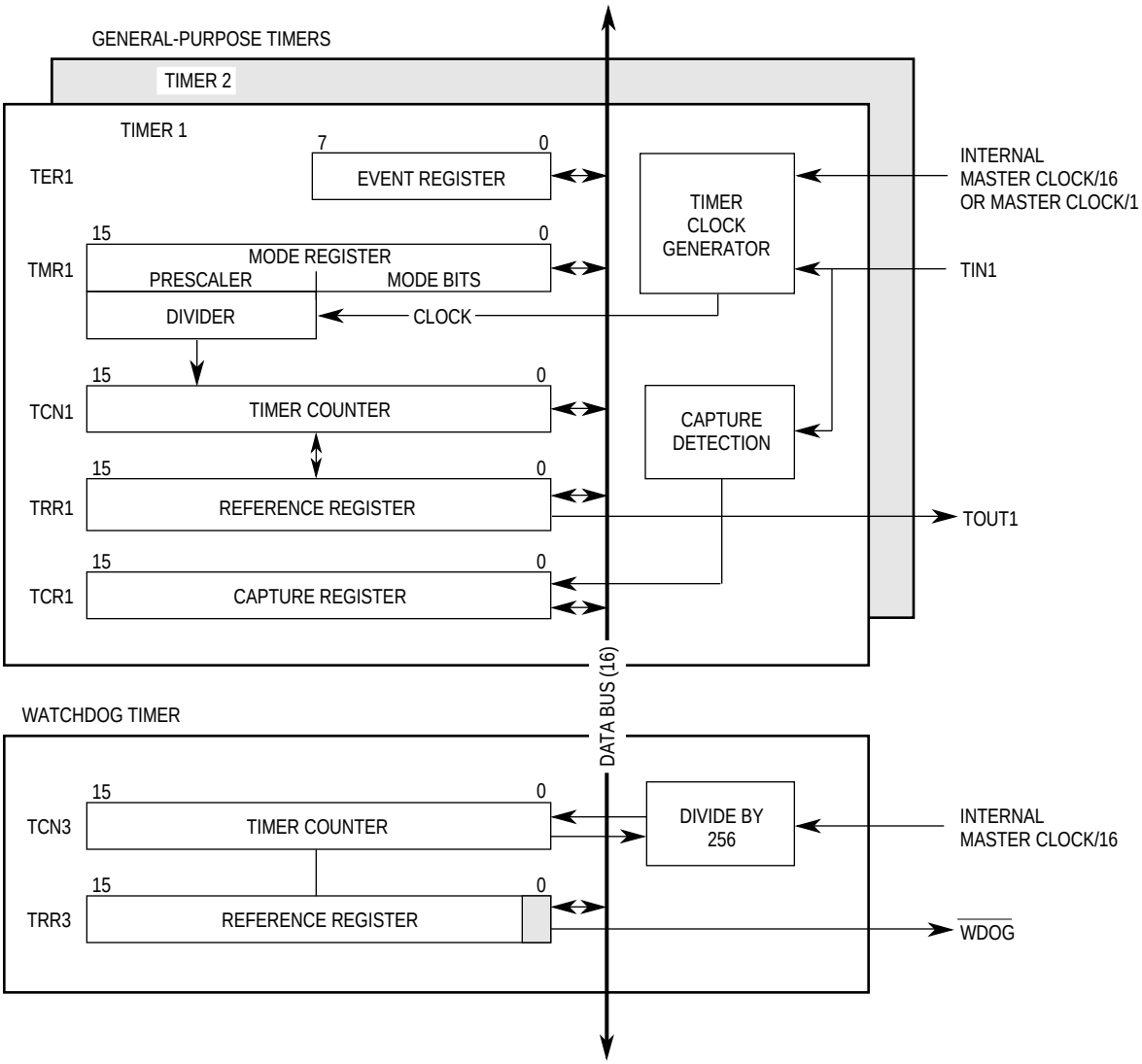


Figure 3-8. Timer Block Diagram

3.5.1 Timer Key Features

The two identical general-purpose timer units have the following features:

- Maximum Period of 16 Seconds (at 16.67 MHz)
- 60-ns Resolution (at 16.67 MHz)
- Programmable Sources for the Clock Input
- Input Capture Capability
- Output Compare with Programmable Mode for the Output Pin
- Two Timers Cascadable to Form a 32-Bit Timer
- Free Run and Restart Modes

questing the M68000 bus with $\overline{BR}$. See 3.8.6 Hardware Watchdog for further details.

3.8.5 Bus Arbitration Logic

Both internal and external bus arbitration are discussed in the following paragraphs.

3.8.5.1 Internal Bus Arbitration

The IMP bus arbiter supports three bus request sources in the following standard priority:

1. External bus master ($\overline{BR}$ pin)
2. SDMA for the SCCs (six channels)
3. IDMA (one channel)

When one of these sources desires the bus, the M68000 core will be forced off through an internal bus request signal ($\overline{CBR}$) from the bus arbiter to the M68000 core (see Figure 3-12). When the arbiter detects the assertion of the M68000 core bus grant ($\overline{CBG}$) signal, it asserts the requester's bus grant signal according to its priority. Thus, as seen externally, the SDMA and IDMA channels do not affect $\overline{BR}$ and $\overline{BG}$, but only $\overline{BGACK}$ (unless disable CPU mode is used).

The MC68302 provides several options for changing the preceding bus master priority list. The options are configured by setting the BCLM bit in the SCR and deciding whether or not the $\overline{BCLR}$ pin is used externally to cause external bus masters to relinquish the bus (see Table 3-10).

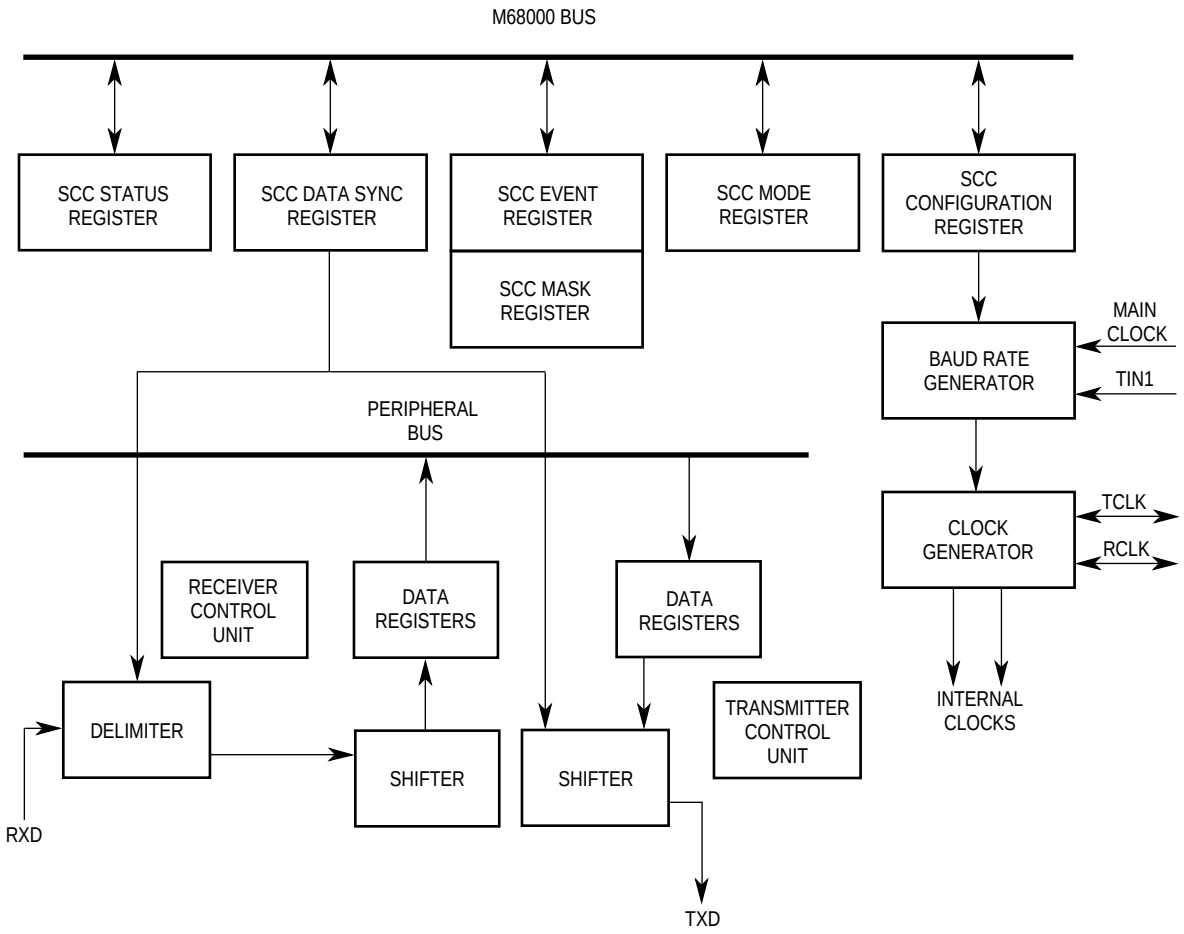


Figure 4-11. SCC Block Diagram

4.5.1 SCC Features

Each SCC channel has the following features:

- HDLC/SDLC, BISYNC, DDCMP, UART, Transparent, or V.110 Protocols
- Programmable Baud Rate Generator Driven by Main Clock or an External Clock
- Data Clocked by the Baud Rate Generator or Directly by an External Pin
- Supports Modem Signals (RXD, TXD, RCLK, TCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$)
- Full-Duplex Operation
- Echo Mode
- Local Loopback Mode
- Baud Rate Generator Outputs Available Externally

4.5.2 SCC Configuration Register (SCON)

Each SCC controller has a configuration register that controls its operation and selects its clock source and baud rate. Figure 4-12 shows one of the three SCC baud rate generators.

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

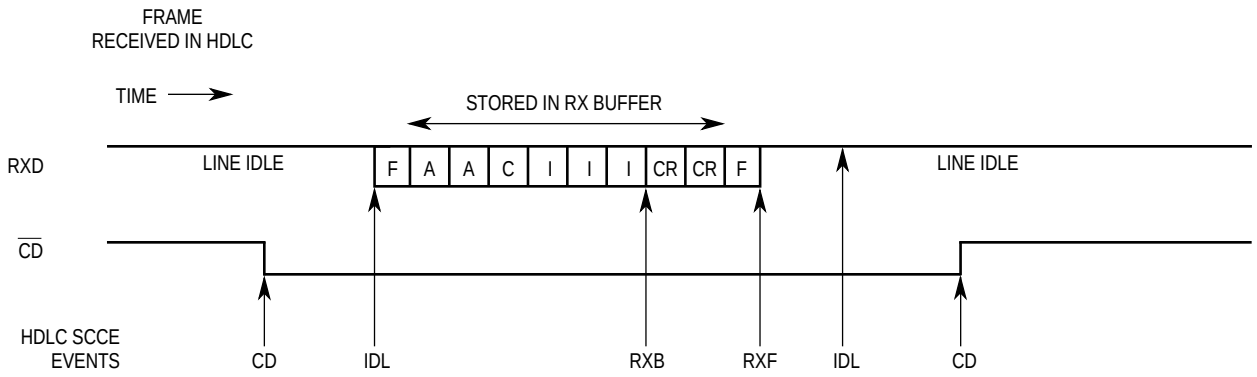
A character with a framing error was received and is located in the last byte of this buffer. A framing error is detected by the UART controller when no stop bit is detected in the receive data string.

PR—Parity Error

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

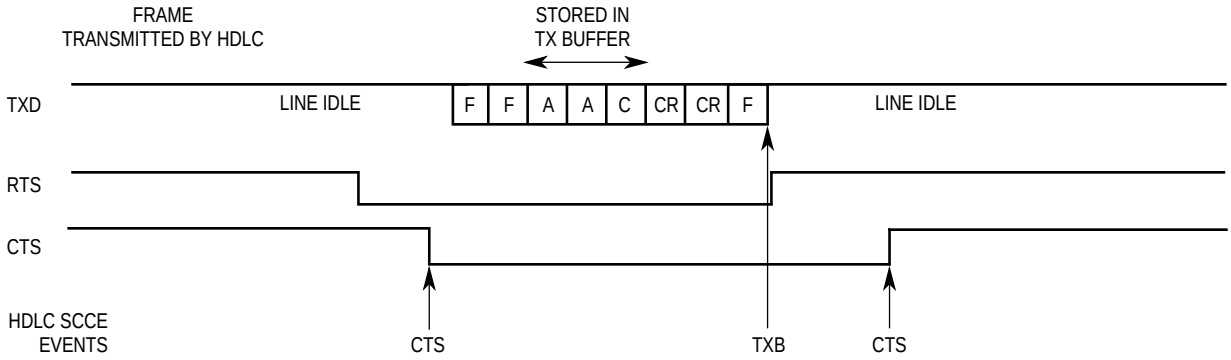
OV—Overrun

A receiver overrun occurred during message reception.



NOTES:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.

LEGEND:
F = Flag A = Address byte C = Control byte I = Information byte CR = CRC byte



NOTE: TXB event shown assumes all three bytes were put into a single buffer.
Example shows one additional opening flag. This is programmable.

Figure 4-29. HDLC Interrupt Events Example

7	6	5	4	3	2	1	0
CTS	CD	IDL	TXE	RXF	BSY	TXB	RXB

CTS—Clear-To-Send Status Changed

A change in the status of the $\overline{\text{CTS}}$ line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the $\overline{\text{CD}}$ line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

- Supports external loopback between two channels or on the same channel
- Trace option for reporting to system management each primitive issued by the LAPD module to the layer 3 module or to layer 2 management

The LAPB module features are as follows:

- Implements *1988 CCITT Recommendation X25,* chapters 2.1 through 2.4
- Supports up to twelve distinct physical channels with each operating as an independent station
- Modulo 8 or modulo 128 operation
- Applicable for DTE and DCE applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each LAPB link
- Special mode for internal loopback (frames are not sent to the driver)
- Supports external loopback between two MC68302 serial channels or on the same MC68302 serial channel
- Trace option for reporting to system management each primitive issued by the LAPB module to layer 3 or to layer 2 management

The X.25 module features are as follows:

- Fully implements 1988 CCITT Recommendation X.25, chapters 3.1 through 7.3
- May be used with both layer 2 modules: LAPD or LAPB
- Unlimited number of layer 2 entities (interfaces)
- Supports up to 4095 logical channels for each interface
- Many DTE/DCE interface parameters (configurable for each interface):

  —- DTE/DCE
  —- Modulo 8/128
  —- Window size
  —- Maximum receive and transmit packet lengths

- Layer 4 message fragmentation/assembly using M-BIT
- Q-BIT support
- All standard CCITT X.25 facilities
- Compatible with X.213 interface primitives
- Link parameters (configurable separately for each interface):

  —- Interface ID

e. Clear all status bits to zero (the default condition).
f. Set (PRD -> Empty) = 1.
g. Move PRD to point to the next BD (If the wrap bit is set, point to the first BD).

If the interrupt was due to a buffer being transmitted, perform the following confirm process:

2. While (CTD -> Ready) = 0 and (CTD -> data length) > 0, do the following:
   /* Look at frames that have been transmitted, but not confirmed */

   a. Check for errors in the BD.
   b. Clear all status bits to zero (the default condition).
   c. Clear out the data length field so that it is zero to start with. This procedure allows new data be placed in this transmit buffer by the transmit algorithm.
   d. Move CTD to point to the next BD. (If the wrap bit is set, point to the first BD.)

Finally:

3. Clear the SCC bit in the in-service register (ISR) of the interrupt controller. This is standard procedure.
4. Return from interrupt.

D.3.7 Final Comments

Note that nowhere in the algorithm does RBD# or TBD# need to be read. The empty and ready bits provide all the necessary information.

Whether receiving buffers or confirming buffers, the interrupt routines deal with all the buffers they can before the interrupt routine is exited. This approach is not mandatory, but if not all buffers are dealt with, then some provision needs to be made for getting the work done without waiting for another interrupt.

For frame-oriented protocols such as HDLC, the user may wish to only process frames, not buffers. In this case, the algorithm would be the same, but the interrupt mask would be set for frame interrupts only.

D.3.8 HDLC Code Listing

The following code shows the initialization of the HDLC protocol and the implementation of the buffer processing algorithms just described. HDLC frames are continuously transmitted and received in the SCC loopback mode.

```
************************** BUFFER PROCESSING CODE**************************
************************************EQU TABLE*******************************
* The following three values are application dependent
BASE       EQU       $0700000          ;This is set according to the val in BAR
INIT       EQU       $0030300          ;Initialization Routine
INT_VEC    EQU       $0031000          ;Interrupt Vector for SCC1
* Commonly used Registers and Parameters
BAR        EQU       $0F2              ;Base Address Register
SCR        EQU       $0F4              ;System Control Register
CKCR       EQU       $0F6              ;Clock Control Register
GIMR       EQU       BASE +  $812      ;Global Interrupt Mode Register
IPR        EQU       BASE +  $814      ;Interrupt Pending Register
```

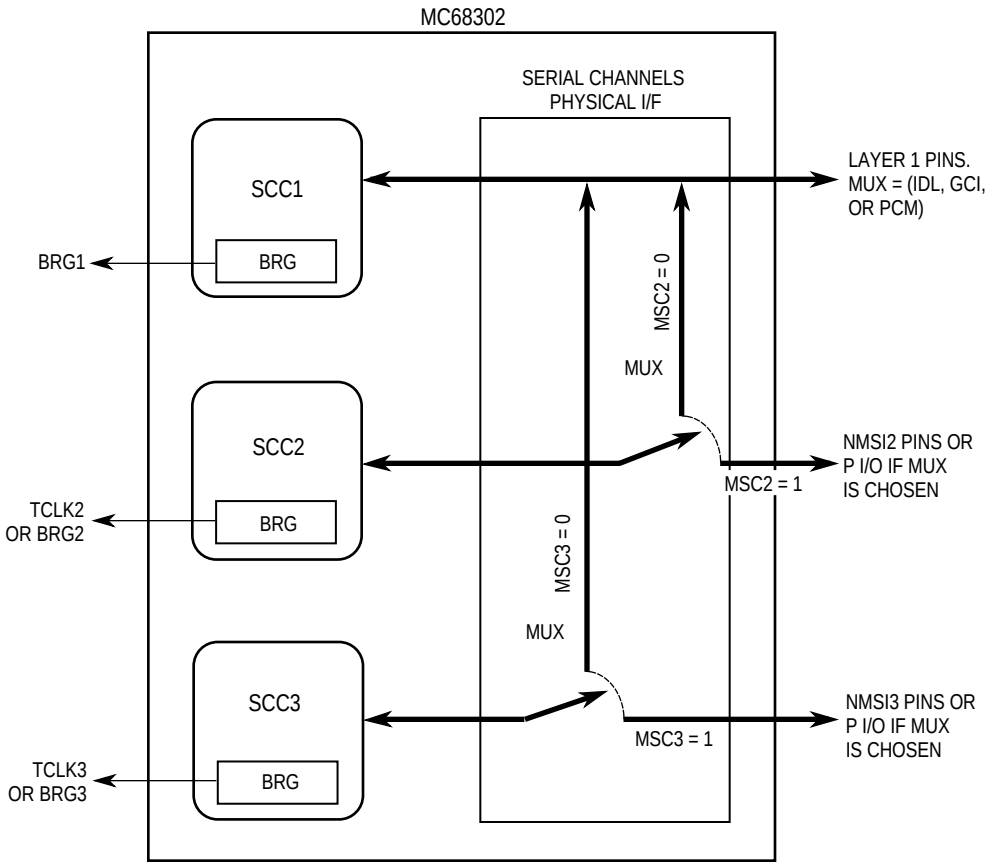


Figure D-22. Multiplexed Modes Example

PCM highway is any time-division multiplexed serial interface (i.e., data is transferred over time slots). The most common examples of a PCM highway (serial bus) are a T1 line or a CEPT line; however, there are many other possible PCM highway configurations. T1 has 24 8-bit channels and is clocked at 1.544 MHz. CEPT has 32 8-bit channels and is clocked at 2.048 MHz. In both cases, the actual rate of an 8-bit channel is 64 kbps. If more data throughput is needed, multiple 8-bit time slots can be grouped together for faster data transfer. You may wish to choose PCM highway, even if you only have one SCC using the interface (and therefore one long time slot) as will be shown.

The following examples illustrate how the different types of interfaces can be combined.

Example 1. If you need multiple transparent channels on separate physical interfaces, then all of them can be NMSI or all but one as NMSI. Thus, you can choose from the following four combinations:

1. NMSI1, NMSI2, and NMSI3
2. PCM, NMSI2, and NMSI3
3. IDL, NMSI2, and NMSI3
4. GCI, NMSI2, and NMSI3

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

E.2.2.1 INITIALIZATION.

1. Write the port A and port B control registers (PACNT and PBCNT) to configure SCC2 or SCC3 serial interface pins as peripheral pins, if SCC2 or CC3 is used.
2. Write SIMODE to configure the SCCs physical interface.
3. Write SIMASK if IDL or GCI multiplexed mode was selected in SIMODE.

E.2.2.2 GENERAL AND UART PROTOCOL-SPECIFIC RAM INITIALIZATION.

4. Write RFCR/TFCR.
5. Write MRBLR.
6. Write MAX_IDLE.
7. Write BRKCR.
8. Write PAREC.
9. Write FRMEC.
10. Write NOSEC.
11. Write BRKEC.
12. Write UADDR1 and UADDR2.
13. Write CHARACTER1-8 in the control character table.

E.2.2.3 SCC INITIALIZATION.

14. Write SCON.
15. Write SCM without setting the ENR and ENT bits.
16. Write DSR.
17. Write SCCE with $FF to clear any previous events.
18. Write SCCM.
19. Write IMR.

E.2.2.4 SCC OPERATION.

20. Write the Rx buffer descriptor control/status, buffer pointer high, and buffer pointer low words for all of the buffer descriptors that are going to be used. Set the W bit in the last buffer descriptor to be used in the queue.
21. Prepare transmit buffers as required to transmit data on the SCC. Set the R bit in each Tx buffer descriptor's control/status word when the data buffer is ready for transmission. Set the W bit in the last Tx buffer descriptor in the table so that the IMP will use the first Tx buffer descriptor (after the user sets the R bit) for the next transmission.
22. Write SCM, setting the ENR and ENT bits to enable reception and transmission on the SCC.
23. Prepare more transmit buffers as required to transmit data on the SCC.

E—Empty

0 = This data buffer is full or has been closed due to an error condition.
1 = This data buffer is empty; must be set by the user to enable reception into this buffer.

X—External Buffer

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

W—Wrap (final BD in table)

0 = This is not the last BD in the receive BD table.
1 = This is the last BD in the receive BD table.

I—Interrupt

0 = No interrupt is generated when this buffer is closed.
1 = The RX bit in the event register is set when this buffer is closed.

Bits 11-2—Reserved for future use; should be written with zero by the user.

OV—Overrun

0 = No receiver overrun occurred.
1 = A receiver overrun condition occurred during frame reception.

CD—Carrier Detect Lost (valid only in NMSI mode)

0 = No CD lost was detected.
1 = CD was negated during frame reception.

E.3.1.4.2 Receive Buffer Data Length. This 16-bit value is written by the IMP to indicate the number of data bytes received into the data buffer.

E.3.1.4.3 Receive Buffer Pointer. This 32 bit value is written by the user to indicate the address where the data is to be stored.

E.3.1.5 TRANSMIT BUFFER DESCRIPTORS. Each SCC has eight transmit buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	—	—	—	—	—	—	—	—	—	UN	CT
OFFSET +2	DATA LENGTH															
OFFSET +4	TX BUFFER POINTER															
OFFSET +6																

E.3.1.5.1 Transmit BD Control/Status Word. To initialize the buffer, the user should write bits 15-11 and clear bits 1-0. The IMP clears bit 15 when the buffer is transmitted or closed due to an error and sets bits 1-0 depending on which error occurred.