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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/kmc68302ag20c

TABLE OF CONTENTS

bus cycle after the completion of the current instruction.

3. The interrupt controller recognizes the interrupt acknowledge cycle and places the interrupt vector for that interrupt request onto the M68000 bus.
4. The M68000 reads the vector, reads the address of the interrupt handler in the exception vector table, and then begins execution at that address.

Steps 2 and 4 are the responsibility of the M68000 core on the IMP; whereas, steps 1 and 3 are the responsibility of the interrupt controller on the IMP.

The M68000 core is not modified on the IMP; thus, steps 2 and 4 operate exactly as they would on the MC68000. In step 2, the M68000 status register (SR) is available to mask interrupts globally or to determine which priority levels can currently generate interrupts (see 2.5 Interrupt Processing for more details). Also in step 2, the interrupt acknowledge cycle is executed.

The interrupt acknowledge cycle carries out a standard M68000 bus read cycle, except that FC2–FC0 are encoded as 111, A3–A1 are encoded with the interrupt priority level (1–7, with 7 (i.e., 111) being the highest), and A19–A16 are driven high. $\overline{UDS}$ and $\overline{LDS}$ are both driven low.

In step 4, the M68000 reads the vector number, multiplies it by 4 to get the vector address, fetches a 4-byte program address from that vector address (seeTable 2-5), and then jumps to that 4-byte address. That 4-byte address is the location of the first instruction in the interrupt handler.

Steps 1 and 3 are the responsibility of the interrupt controller on the IMP. In steps 1 and 3, a number of configuration options are available. For instance, in step 1, there are two modes for handling external interrupts: normal and dedicated. In step 3, there are several different ways of generating vectors. These and other interrupt controller options are introduced in the following paragraphs.

3.2.1.2 Interrupt Controller Overview

The interrupt controller receives interrupts from internal sources such as the timers, the IDMA controller, the serial communication controllers, and the parallel I/O pins (port B pins 11–8). These interrupts are called internal requests (INRQ). The interrupt controller allows for masking each INRQ interrupt source. When multiple events within a peripheral can cause the INRQ interrupt, each event is also maskable in a register in that peripheral.

In addition to the INRQ interrupts, the interrupt controller can also receive external requests (EXRQ). EXRQ interrupts are input to the IMP according to normal or dedicated mode. In the normal mode, EXRQ interrupts are encoded on the $\overline{IPL2}$–$\overline{IPL0}$ lines. In the dedicated mode, EXRQ interrupts are presented directly as $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$.

Normal Mode

In this mode, the three external interrupt request pins are configured as $\overline{IPL2}$–$\overline{IPL0}$ as in the original MC68000. Up to seven levels of interrupt priority may be encoded. Level 4 is reserved for IMP INRQ interrupts and may not be generated by an external device.

pendix A SCC Performance). Also, the minimum 1:2.5 serial to CLKO clock ratio must be maintained at all times.

The following list gives a step-by-step example of how to achieve the lowest possible power using an external clock. For this example, an external wakeup signal is issued to the PB11 pin to exit the lowest power mode.

1. Set the lower byte of the SCR (location $F7) to $A0. This sets the LPREC bit and the LPEN bits only.
2. Disable all interrupts except PB11 in the IMR.
3. Turn off any unneeded peripherals, such as the SCCs, by clearing the ENR and ENT bits. Also, turn off any unneeded baud rate generators by setting the EXTC bits in the SCON registers. This procedure can save as much as 4 mA per SCC at 16.67 MHz. (EXTC is cleared by default on after reset.)
4. Start off a timer now to toggle a $\overline{\text{TOUT}}$ pin in approximately 20 clocks. Do not wait for this to occur, but continue on to the next step.
5. Execute the STOP instruction. The IMP is now safely in the lowest power mode.
6. Use the toggled $\overline{\text{TOUT}}$ pin to switch the EXTAL clock rate to approximately 50 kHz. Ensure no glitches occur on the EXTAL signal which exceed the maximum clock frequency.
7. Power consumption is now the lowest.
8. A wakeup signal comes from the system.
9. The wakeup signal switches the clock frequency back to the 8–16.67-MHz range and pulls the PB11 pin low. These two events can happen simultaneously.
10. The IMP generates the PB11 interrupt, and a M68000 core reset is generated.
11. After the IMP is reset, software processing continues from the exception vector table reset vector address. The M68000 is reset, but the rest of the IMP retains its state.

The low-power logic uses eight bits in the SCR.

LPCD4–LPCD0—Low-power Clock Divider Selects

The low-power clock divider select bits (LPCD4—LPCD0) specify the divide ratio of the low-power clock divider equal to LPCD4—LPCD0 + 1. The system clock is divided by 2, then divided by the clock divider value (1 to 32). Thus, a divide ratio of 2 to 64 (LPCD4—LPCD0 0 to 31) can be selected. After a system reset, these bits default to zero.

LPEN—Low-power Enable

0 = The low-power modes are disabled.
1 = The low-power modes are enabled.

After a system reset, this bit defaults to zero to disable the low-power modes.

The HDLC controller uses the same data structure as the UART, BISYNC, and DDCMP controllers. This data structure supports multibuffer operation and address comparisons.

The receive errors (overrun, nonoctet aligned frame, $\overline{CD}$ lost, aborted frame, and CRC error) are reported through the receive BD. The transmit errors (underrun and $\overline{CTS}$ lost) are reported through the transmit BD. An indication about the status of the lines (idle, $\overline{CD}$, and $\overline{CTS}$) is reported through the SCC status register (SCCS), and a maskable interrupt is generated upon a status change in any one of those lines.

4.5.12.5 HDLC Command Set

The following commands are issued to the command register.

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every eight or sixteen transmit clocks as determined by the FLG bit in the HDLC mode register.

The channel STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the HDLC controller during frame transmission, transmission of that frame is aborted after the contents of the FIFO are transmitted (up to four words). The TBD# is not advanced. No new BD is accessed, and no new frames are transmitted for this channel. The transmitter will transmit an abort sequence (if the command was given during frame transmission) and then begin to transmit flags or idles as indicated by the HDLC mode register. The abort sequence on transmit is a zero followed by seven ones (01111111).

This command is useful for performing frame retransmission. The M68000 core may issue the STOP TRANSMIT command, reorganize the transmit BD table, and issue the RESTART TRANSMIT command. The STOP TRANSMIT command may also be used in the X.25 protocol to send a reject frame or a link reset command.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter is to be re-enabled at a later time.

RESTART TRANSMIT Command

The RESTART TRANSMIT command re-enables the transmission of characters on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and disabling the channel in its SCC mode register, or after transmitter error (underrun or $\overline{CTS}$ lost when no automatic frame retransmission is performed). The HDLC controller will resume transmission from the current transmitter BD (TBD#) in the channel's transmit BD table.

If the transmitter is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

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. During an idle condition, with the FLG bit cleared, the line will be forced to a high state.

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.

4.5.12.10 HDLC Receive Buffer Descriptor (Rx BD)

The HDLC controller uses the Rx BD to report information about the received data for each buffer. The Rx BD is shown in Figure 4-26.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	F	—	—	—	—	LG	NO	AB	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-26. HDLC Receive Buffer Descriptor

An example of the HDLC receive process is shown in Figure 4-27. This shows the resulting state of the Rx BDs after receipt of a complete frame spanning two receive buffers and a second frame with an unexpected abort sequence. The example assumes that MRBLR = 8 in the SCC parameter RAM.

The first word of the Rx BD contains control and status bits. Bits 15–10 are written by the user before the buffer is linked to the Rx BD table, and bits 5–0 are set by the CP following frame reception. Bit 15 is set by the M68000 core when the buffer is available to the HDLC controller; it is cleared by the HDLC controller when the buffer is full.

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

The purpose of the control characters table is to enable automatic recognition (by the BISYNC controller) of the end of the current block. See 4.5.13.14 Programming the BISYNC Controllers for more information. Since the BISYNC controller imposes no restrictions on the format of the BISYNC blocks, user software must respond to the received characters and inform the BISYNC controller of mode changes and certain protocol events (e.g., resetting the BCS). However, correct use of the control characters table allows the remainder of the block to be received without interrupting the user software.

Up to eight control characters may be defined. These characters inform the BISYNC controller that the end of the current block has been reached and whether a BCS is expected following this character. For example, the end of text (ETX) character implies both an end of block (ETB) and a BCS should be received. An enquiry (ENQ) character designates end of block without a subsequent BCS. All the control characters are written into the data buffer.

The BISYNC controller uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, an end-of-table bit, a BCS expected bit, and a hunt mode bit. The control characters table is shown in Figure 4-31. To disable the entire control characters table, write $8000 to the first table entry.

	15	14	13	12	11	10	9	8	7 … 0
OFFSET + 0	E	B	H						CHARACTER1
OFFSET +2	E	B	H						CHARACTER2
OFFSET + 4	E	B	H						CHARACTER3
•									
•									
•									
OFFSET + E	E	B	H						CHARACTER8

Figure 4-31. BISYNC Control Characters Table

CHARACTER8–CHARACTER1—Control Character Value

These fields define control characters.

NOTE

When using 7-bit characters with parity, the parity bit should be included in the control character value.

E—End of Table

0 = This entry is valid. The lower eight bits will be checked against the incoming character.

1 = The entry is not valid. No valid entries exist beyond this entry.

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

Figure 4-40. V.110 Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits. Bits 15–13 are written by the user before the buffer is linked to the Rx BD table, and bits 1 and 3 are set by the IMP following message reception. Bit 15 is set by the M68000 core when the buffer is available to the V.110 controller and is cleared by the V.110 controller after filling the buffer.

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 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 V.110 controller. The M68000 core should not write to any fields of this BD after it sets this bit. The empty bit will remain set while the V.110 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 V.110 controller receives incoming data by placing it in 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.

Bits 12–4, 2, 0—Reserved for future use.

SE—Synchronization Error

A frame with a synchronization error was received. A synchronization error is detected by the V.110 controller when the MSB of a byte (except the all-zeros byte) is not one.

OV—Overrun

A receiver overrun occurred during message reception.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.
1 = When this buffer has been closed by the transparent controller, the RX bit in the transparent event register will be set, which can cause an interrupt.

The following status bits are written by the CP after the received data has been placed into the associated data buffer.

Bits 11–2—Reserved for future use. Should be written with zero by the user.

OV—Overrun

A receiver overrun occurred during reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during buffer reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer. It is written only once by the CP as the buffer is closed.

NOTE

The actual buffer size should be greater than or equal to the MRBLR.

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be even. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.16.9 Transparent Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) using the BDs to inform the processor that the buffers have been serviced. The Tx BD is shown in Figure 4-43.

	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 (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-43. Transparent Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. These bits are prepared by the user before transmission and are set by the CP after the buffer has been transmitted.

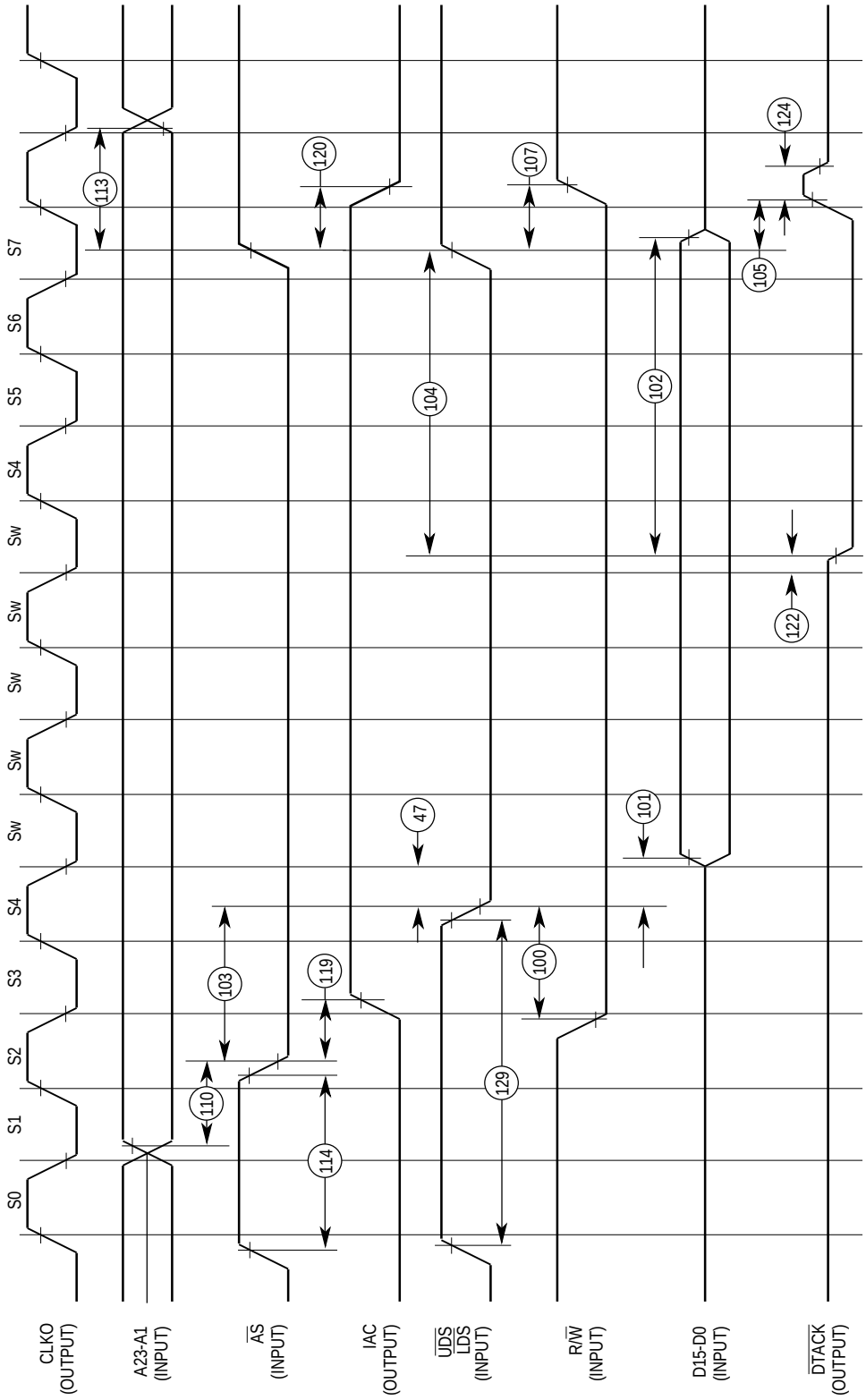

Figure 6-9. External Master Internal Asynchronous Write Cycle Timing Diagram

The LAPB, LAPD, and X.25 modules are available in C source code form. The source code for these modules requires a license from Motorola.

The EDX kernel is available in assembler source code form. The source of the EDX kernel requires a license from Motorola.

B.3 THIRD-PARTY SOFTWARE SUPPORT

Since the IMP is a memory-mapped device based on a full M68000 core, existing compilers, source-level debuggers, assemblers, and linkers designed for the M68000 Family may be successfully used. Many third parties supply such tools.

B.4 IN-CIRCUIT EMULATION SUPPORT

Full in-circuit emulation support is available from multiple third parties, but is not discussed in this manual.

B.5 302 FAMILY ADS SYSTEM

The M68302FADS is an integrated Family Applications Development System (FADS) designed to aid hardware and software developers of the MC68302, MC68LC302, MC68PM302, and MC68EN302 in quickly evaluating and developing applications for these devices. All of the hardware resources needed to download and debug application software are provided, such as large blocks of flash and static RAM for the processors, serial ports, clock generation circuitry, logic analyzer connectors, expansion connectors as well as monitor/debugger hardware and software. The logic analyzer connectors provide the user with access to all of the processors' pins in order to monitor bus activity. The expansion connectors let the user attach hardware applications and to use board resources to verify a design.

To serve as a convenient platform for software development, the M68302FADS is provided with a monitor/debugger for the Integrated Multiprotocol Processor (IMP) section. The monitor/debugger provides the following operations: memory dump and set (with optional disassembly of 68K instructions), single instruction execution, breakpoints, and downloads. The debugger interface can work together in separate Windows 3.0 DOS shells on the same x86 based PC to communicate with the on-board IMP hardware. Future support for SUN platforms is planned.

The M68302FADS board has sockets for and is shipped with the MC68302RC25, MC68LC302RC20, MC68PM302RC20. An adapter card for the MC68EN302 will be available separately.

- General Features
  - —Supports the 68302 family of processors: MC68302, MC68LC302, MC68PM302 and MC68EN302 (with an adaptor).
  - —On board IMP(68302) debugger software with host debugger interface.
  - —Separate external clock generators for the IMP(68302)

APPENDIX D
MC68302 APPLICATIONS

This appendix describes different applications for the MC68302.

D.1 MINIMUM SYSTEM CONFIGURATION

The following paragraphs describe a minimum 16-bit MC68302 system. As Figure D-1 shows, this system can be easily built with very few components.

D.1.1 System Configuration

The crystal circuit shown in Figure D-1 is a typical configuration. The values used are not required by Motorola. Some deviation of the capacitance or resistance values is allowed. Crystal parameters need not be anything special—Co < 10 pF and Rx = 50 Ω is used. Of course, an oscillator could be used in place of the crystal circuit.

$\overline{\text{AVEC}}$ is pulled high since autovectoring for external interrupts is not needed. If external devices were added (not shown) the MC68302 interrupt controller could handle the interrupt vector generation for up to seven external sources using $\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$, $\overline{\text{1RQ1}}$, PB11, PB10, PB9, and PB8. The $\overline{\text{IPL2}}$-$\overline{\text{IPL0}}$ lines are also pulled high (inactive) since no external interrupts are required.

BUSW is pulled high for 16-bit operation and may not be modified dynamically. Choice of 8-bit operation could be used to eliminate two of the memory chips.

$\overline{\text{BERR}}$ is pulled high since it is an open-drain signal. It will be asserted low by the MC68302 if the hardware watchdog terminates a stalled bus cycle.

$\overline{\text{BR}}$ is tied high since no external bus masters exist in this design.

$\overline{\text{BGACK}}$ is pulled high (inactive). It is asserted low during an IDMA or SDMA bus cycle.

$\overline{\text{FRZ}}$ is tied high since the MC68302 freeze debugging logic is not used in this design.

DISCPU is tied low to allow the M68000 core to function normally. Tying this pin high causes the part to enter the disable CPU mode when the core is disabled and the part is an intelligent peripheral.

All unused lines should be terminated.

circuit will also place the MC68302 into reset if V_{CC} drops below a threshold, an advantage over a simple RC circuit.

D.1.3 Memory Interface

$\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{LDS}$, A23–A1, D15–D0, $\overline{CS0}$, and $\overline{CS1}$ are used in the memory interface. The bidirectional control signals have pullups so that they are inactive when the MC68302 is not actively driving them. $\overline{CS0}$ and $\overline{CS1}$ are only outputs and do not need pullups.

$\overline{DTACK}$ is generated internally by the wait-state generation logic, and therefore it does not need to be externally connected. It is driven by the wait-state generation logic during chip-select accesses and is pulled up externally through a resistor.

$\overline{BCLR}$ is an open-drain signal that is output by the MC68302 when the SDMA requests the bus or (if enabled) when an interrupt request is pending. It is therefore pulled up externally.

D.1.4 Memory Circuit

The EPROM design is a straight connection. Larger EPROM sizes may be easily substituted; it is also possible to use a single 16-bit EPROM, if desired, and reduce the component count by one.

$\overline{CS0}$ is used to select the EPROM since $\overline{CS0}$ is designed to be used for a boot ROM. $\overline{CS0}$ comes up enabled for the first 8 kbytes of the system address space. Before the program jumps outside of this space, it should configure the $\overline{CS0}$ range for 32 kbytes and enable the RAM chip select to cover the RAM range and desired starting address. To switch the RAM down to low-order memory to allow exception vectors to be altered, see D.2 Switching the External ROM and RAM Using the MC68302**.**

$\overline{LDS}$ and $\overline{UDS}$ define the lower and upper bytes of the program word, respectively, and are connected to the EPROM output enable.

The RAM design is similar to the EPROM design, except that the R/$\overline{W}$ signal is used to qualify the output enable and the $\overline{LDS}$/$\overline{UDS}$ signals are used to qualify whether to write the particular byte of the data word (the M68000 supports byte operations).

$\overline{CS1}$ is used to select the RAM.

No external buffers are required in this small system design.

D.1.5 Memory Timing Analysis

The design as shown will work with zero wait states with a 100-ns EPROM and a static RAM, such as the MCM6202. The RAM writes are controlled by the enable ($\overline{E}$) rather than by the write ($\overline{W}$). Two of the MC68302 required specifications deserve special note.

The time between $\overline{CS}$ low and data valid must be calculated for proper timing of read cycles. This time is approximately 2 1/2 clocks. $\overline{CS}$ is asserted by the MC68302 after the rising edge of S2. Data must be valid a setup time before the S6 falling edge. Thus, for a 16.67-MHz device, the equation for the time between $\overline{CS}$ low and data valid is as follows:

```
* Set up PACNT, PBCNT, etc., ignore for this example, only SCC1 is used
* Select Serial Interface Mode: normal operation, NMSI mode
          MOVE.W    #0,SIMODE          ;Same as default after reset

*** SCC1 Initialization ***
*  Interrupt Vector: SCC1 interrupt handler is at INT_VEC = $31000
*  v7-v5 =5, v4-v0 = $ad -> vector = $ad -> Exception vector = ($ad<<2) = $2b4
          MOVE.L    #INT_VEC, $02B4

* Determine Configuration
* Use Baud Rate Generator for transmit and receive, Rate is 130 kbps.
          MOVE.W    #$07E,SCON1

* Select SCC Mode
* HDLC, Loopback mode, CRC16, RTS negate between frames, NRZ mode.
          MOVE.W    #$10,SCM1

* Set up Parameter RAM
          MOVE.W    #0,FCR_1           ; Clear RFCR and TFCR
          MOVE.W    #$08,MRBLR_1       ; Max Buffer Length = 8
          MOVE.W    #$F0B8,CMSKL_1     ; 16 bit CRC
          MOVE.W    #$070,MFLR_1       ; Max Frame Length = $70 bytes
          MOVE.W    #0,HMASK_1         ; Do not check address
          MOVE.W    #0,DISFC_1         ; Clear the counter
          MOVE.W    #0,CRCEC_1         ; Clear the counter
          MOVE.W    #0,ABTSC 1         ; Clear the counter
          MOVE.W    #0,NMARC_1         ; Clear the counter
          MOVE.W    #0,RETRC_1         ; Clear the counter

* Clear Event Register
          MOVE.B    #$FF,SCCE1

* Determine Maskable Interrupt Events by setting SCCM
* Allow the following interrupt: TXE, RFX, TXB, and RFB
          MOVE.B    #$1B,SCCM1

* Clear M68000 data registers
          CLR.L     D0
          CLR.L     D1
          CLR.L     D2
          CLR.L     D3
          CLR.L     D4
          CLR.L     D5

***Prepare Buffer Descriptors ***
*SCC1 Rx Buffer Descriptors Initialization values before execution:
*00700400 D000 0000 0003 0000 D000 0000 0003 0010
*00700410 D000 0000 0003 0020 D000 0000 0003 0030
*00700420 D000 0000 0003 0040 D000 0000 0003 0050
*00700430 D000 0000 0003 0060 F000 0000 0003 0070
          LEA.L     RXBD_01,A0         ;A0 points to the first RXBD of SCC1
          LEA.L     RXBF_01,A1         ;A1 points to the first buffer
          MOVE.W    #$D000,D1          ;D1 is used for setting the status of BD
*                                      ;Empty = 1, External = 1, Int=1
          MOVE.W    #$F000,D2          ;D2 is for the last BD, Wrap = 1
```

The L1SY1-L1SY0 signals determine 1) when clocks are sent to the particular SCC and 2) when the synchronization signal is sent to the SCC. If the L1SY1-L1SY0 signals are not active, then the SCC is not being clocked. The rising edge of L1SY1-L1SY0 starts the clocking and sends an internal synchronization pulse to the SCC. These facts are very important in determining when the first byte of real data from a buffer will be transmitted onto the PCM.

In transparent mode, if data is not ready to transmit, $FFs will be sent during the time slot. Once data transfer begins, data will be clocked out during every clock of the time slot. When the time slot ends, the SCC will wait without being clocked until the next time slot arrives. Similarly, on the receive side, data and the clock will only be presented to the SCC when that SCC's time slot is active.

When using transparent mode with PCM, it is often of interest to know exactly when the very first buffer of transmit data will go out. The $\overline{\text{RTS}}$ signal gives an important clue here. Once the $\overline{\text{RTS}}$ signal for an SCC is asserted, the next rising edge of the L1SY1-L1SY0 pins for this channel (i.e., the SCC's next time slot) will begin clocking out the following pattern:

$FF, data1, data2, data3, . . .

where data1 is the first byte of data stored in the transmit data buffer. For example, if the PCM was configured with individual 8-bit time slots for this SCC, $FF would be clocked on the first time slot, data1 on the second, etc. **It is assumed in that this buffer in this example does not immediately follow the previous buffe**r. A string of buffers with their L bits cleared will follow each other immediately without any delay—only the first will have this delay.

From the time the ENT bit is set and a buffer is ready to transmit, it can take a number of serial clocks (usually less than 36) for $\overline{\text{RTS}}$ to be asserted (it could be more for higher data rates). Thus, this clock delay must be taken into account if data transmission delays need to be consistent. The delay can be accounted for by synchronizing the setting of the ENT bit with the time slot itself. If the time slot is long, it should be sufficient to set the ENT bit before one time slot to guarantee data transmission during the next time slot (see Figure D-30). The algorithm can work as follows:

1. After the last buffer is transmitted, give STOP TRANSMIT command.
2. Clear ENT.
3. Give RESTART TRANSMIT command.
4. Set ready bit of next Tx BD to transmit.
5. Generate interrupt to MC68302 on falling L1SY1 /L1SY0 pin.
6. Now that time slot is inactive, set ENT bit.

EXTC—External Clock Source

0 = The internal main clock is the source for the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on TCLK pin.

CD10-CD0—Clock Divide

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.2.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3). The SCM register configures the operation of the SCC and defines UART specific parameters.

15	14	13	12	11	10	9	8
TPM1	TPM0	RPM	PEN	UM1	UM0	FRZ	CL

7	6	5	4	3	2	1	0
RSTM	SL	DIAG1	DIAG0	ENR	ENT	MODE1	MODE0

TPM1,TMP0—Transmitter Parity Mode

00 = Odd parity; always send an odd number of ones.
01 = Force low parity; always send a zero in the parity bit position.
10 = Even parity; always send an even number of ones.
11 = Force high parity; always send a one in the parity bit position.

RPM—Receiver Parity Mode

0 = Odd parity.
1 = Even parity.

PEN—Parity Enable

0 = No parity.
1 = Parity is enabled for the transmitter and receiver.

E.2.1.2.3 SCC Data Synchronization Register (DSR). This 16-bit register is located at offset $886 (SCC1) $896 (SCC2) and $8A6 (SCC3). Bits 14-12 of the DSR are used to program the length of the last stop bit transmitted. These bits are decoded as follows:

DSR(14-12)—Fractional Stop Bits

111 16/16 (default value after reset).
110 15/16.
101 14/16.
100 13/16.
011 12/16.
010 11/16.
001 10/16.
000 9/16.

E.2.1.2.4 UART Event Register (SCCE). This 8-bit register is located at offset $888 (SCC1) $898 (SCC2) and $8A8 (SCC3) on D15-D8 of a 16-bit data bus. The SCCE is used to report events recognized by the UART channel. Bits must be cleared by the user to avoid missing interrupt events. Bits are cleared by writing ones to the corresponding bit positions.

7	6	5	4	3	2	1	0
CTS	CD	IDL	BRK	CCR	BSY	TX	RX

CTS—Clear-To-Send Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CTS}}$ was detected.

CD—Carrier Detect Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CD}}$ was detected.

IDL—IDLE Sequence Status Changed

0 = No interrupt.
1 = A change in the status of the receive data serial line was detected.

BRK—Break Character Received

0 = No interrupt.
1 = Break character received.

CCR—Control Character Received

0 = No interrupt
1 = Control character received (with reject (R) character = 1) and stored in the receive control character register (RCCR).

BSY—Busy Condition

0 = No interrupt.
1 = A character was received and discarded due to lack of buffers.

E.3.1.2 PER SCC REGISTERS. Each of the three SCCs has a set of the following six registers. These registers configure the SCC and the protocol operation. Some parameters and register bits are protocol independent. The transparent functions have been given for those parameters and bits that are protocol specific.

E.3.1.2.1 Serial Configuration Register (SCON). This 16-bit register is located at offset $882 (SCC1), $892 (SCC2), and $8A2 (SCC3). The SCON register is used to select the clock source and baud rate for the SCC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOMS	EXTC	TCS	RCS	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV4

WOMS—Wired-OR Mode Select

0 = TXD driver operates normally.
1 = TXD driver functions as an open-drain output and may be wired together with other TXD pins.

EXTC—External Clock Source

0 = The internal main clock is the source of the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on TCLK pin.

CD1 0-CD0—Clock Divider

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.3.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3) The SCM register configures the operation of the SCC and defines transparent-specific parameters. Note that reserved bits in registers should be written as zeros.

15	14	13	12	11	10	9	8
—	EXSYN	NTSYN	REVD	—	—	—	—

7	6	5	4	3	2	1	0
—	—	DIAG1	DIAG0	ENR	ENT	MODE1	M0DE0

APPENDIX F
DESIGN CHECKLIST

When integrating the MC68302 into an application, it may be helpful to go through the following design checklist. In this checklist are a number of common problems and their resolutions that have been found while debugging real MC68302 applications.

1. **Version, Mask**
   Older versions of the MC68302—called Rev A and Rev B with masks “B14M” written on the device—do not contain all the features listed in this manual. These devices have ceased production and are longer available. A newer version called Rev C, which is identified by mask number “C65T” or later, contains all the features listed in this manual. The missing features in the old versions include clock output control and a few other minor differences.
2. **External Pin Configurations**
   A good checklist of external pin configurations may be found in D.1 Minimum System Configuration. Common problems are also listed in some of the following paragraphs.
3. **Clock Present**
   If you are using an external clock source to the 68302, make sure that it is driving the clock input within 10 msec of powerup. Otherwise, part damage can occur.
4. **$\overline{\text{AVEC}}$, $\overline{\text{DTACK}}$**
   If $\overline{\text{AVEC}}$ is not used, it needs to be pulled high (to + 5 V); otherwise, erratic behavior and bus cycles may occur. A pullup resistor may be used, if desired. If $\overline{\text{AVEC}}$ is used, it should be asserted instead of $\overline{\text{DTACK}}$ (not in addition to $\overline{\text{DTACK}}$) during interrupt acknowledge cycles.
5. **Pullup, $\overline{\text{DTACK}}$**
   Sometimes a 10K-ohm resistor may not be strong enough to pull up $\overline{\text{DTACK}}$ to provide adequate rise times to the $\overline{\text{DTACK}}$ signal. This is a loading-dependent issue.
6. **Pullup, Floating $\overline{\text{BR}}$, $\overline{\text{FRZ}}$, BUSW**
   Unexpected behavior can result if the signals $\overline{\text{BR}}$, $\overline{\text{FRZ}}$, BUSW, or other inputs are left floating. Of these, the most common mistake is to leave $\overline{\text{FRZ}}$ floating.
   If no external requests are made, $\overline{\text{BR}}$ may be pulled directly high. If external requests are made, $\overline{\text{BR}}$ may need to be pulled high through a resistor, such as 1 K ohm, to guarantee adequate $\overline{\text{BR}}$ rise time to meet bus arbitration specifications.
7. **Pullup, $\overline{\text{IPL}}$**
   $\overline{\text{IPL}}$ lines should be pulled high if not used. These signals may be pulled directly high, if desired.
8. **$\overline{\text{RESET}}$, Rise Time**
   The rise time of the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ pins after a total system reset must be within

F

G

H

I