Greenspring IP-CM302

Intelligent Multiprotocol Communication IndustryPack

In Stock

New From Surplus Stock

Open Web Page

https://www.artisantg.com/98411-1

Intelligent Multiprotocol Communication IndustryPack® Manual

Hardware Revision A
Part # 89002060 Rev. 1.0
20050124

IP-CM302

Intelligent Multiprotocol Communication IndustryPack®

SBS Technologies, Inc.
1284 Corporate Center Drive
St. Paul, MN 55121
Tel (651) 905-4700
FAX (651) 905-4701
Email: support.commercial@sbs.com
http://www.sbs.com

Table of Contents

List of Figures

Product Description

The IP-CM302 is part of the IndustryPack® (IP) family of modular I/O components. It is an intelligent multiprotocol communication IP. The IP incorporates the Motorola MC68302 Integrated Multiprotocol Processor (IMP) running at 16 Mhz. The 68302 contains a 68000 core processor, a microcoded RISC communications processor, 6 dedicated DMA channels, 3 timers, an interrupt controller, three independent full-duplex serial channels, 1152 bytes of dual-port RAM, and a conventional parallel 68000 bus interface as well as an expandable serial interface.

A variety of physical layer standards, such as RS-232-D, RS-422, RS-485, ISDN, are supported by the use of transition modules. The architecture makes it easy for customers to implement special physical interface requirements by substituting their own transition module. Transition module is installed between the IP I/O connector and other communication terminals.

Internal microcode in the 68302 supports a variety of protocols: BISYNC, HDLC, SDLC, UART, DDCMP, V.110, V.120, X.25, ISDN (2B+D channels) and Transparent Modes.

The high density implementation is made possible by the use of TSOP memory package and field programmable gate array Xilinx® in fine-pitch SMT package. The Xilinx includes IP logic interface, control registers, control signals and base clock to the 68302.

The IP implements 256 Kbytes of high speed volatile static RAM (SRAM) as dual-port memory for both 68302 code and communications data. The code must be downloaded into the SRAM by a host CPU prior to removing reset to the 68302. The reset signal is controlled by one of the control registers in Xilinx. As soon as reset is removed, the 68302 begins executing code from the SRAM. Upon power-up the IP is in state of reset and ready to be downloaded with code. Subsequent system resets or reset from Xilinx does not alter the SRAM content.

The SRAM, 68302 registers and on-chip RAM are dual ported. Dual-portable is defined as accessibility by both the 68000 core processor and the host CPU.

Both the IP and the host CPU are capable of generating interrupts to each other.

The 16 Mhz clock to the 68302 is derived by multiplying the IP clock by two by on-board logic. Alternatively a crystal may be installed on-board to provide a different base clock to the 68302. However, this is a custom feature that requires a customer to contact factory for a special quote.

An external baud rate generator clock may be provided from the IP I/O connector. This clock is used to generate baud rates the base clock is incapable of generating. Typically this is not needed since the 16Mhz base clock delivers all the standard baud rates.

Key Features

- MC68302 Integrated Multiprotocol Communication Processor @ 16Mhz
- 3 independent full-duplex serial channels
- 6 DMA channels
- 256 Kbytes of dual-port static RAM
- 16-bit interface
- HDLC, SDLC, BISYNC, DDCMP, UART, V.110, V.120, X.25, ISDN (2B+D channels).
- Physical layer implemented with transition modules

- Single-high IndustryPack
- All CMOS surface mount—low power, rugged
- Up to 12 serial channels/24 DMA channels per VME slot

Throughout this User Manual, references are made to the 68302 features and signals that are specific to the IP. The user should consult the MC68302 User Manual for a thorough understanding of the 68302 features and performance.

Shown below in figure 1 is a block diagram of the IP-CM302.

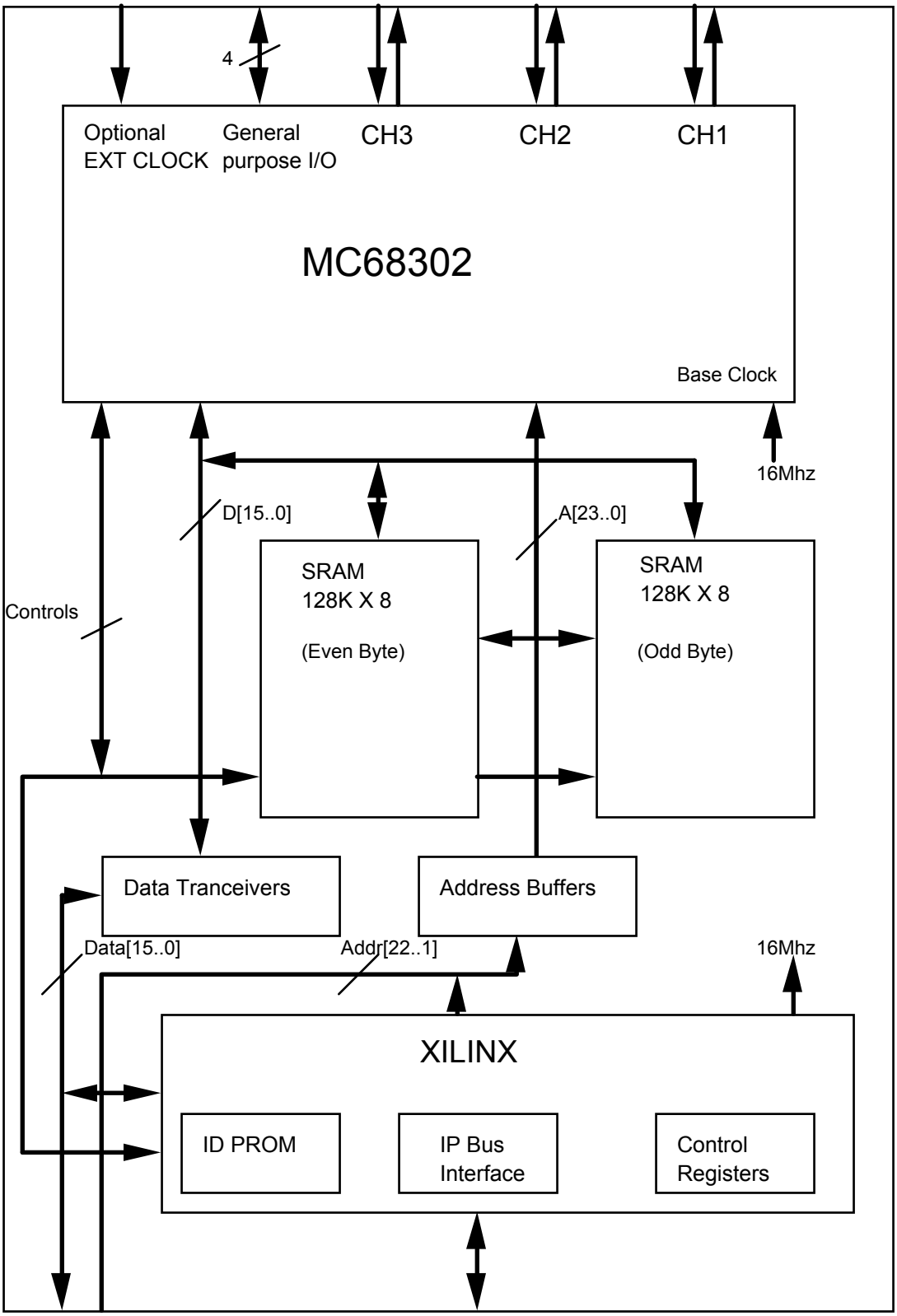

Figure 1 IP-CM302 Block Diagram

VMEbus Addressing

The address map of the MC68302 registers on the VMEbus is the same as tables 2-6 through 2-9 in the MC68302 User Manual Rev. 2. Even byte accesses are on data lines D7..0. Odd byte accesses are on data lines D15..8. Word accesses are on D15..0. This is also true for other resources on the IP. The SRAM and the 68302 on-chip RAM are word wide. Control registers in Xilinx are byte wide. The figure 2 below provides few examples of addressing 68302 registers plus all the Xilinx control registers. All resources on the IP are mapped in the IP memory space except Xilinx control registers which are mapped in the I/O space.

Register Name	68K Address	Access
MC68302 Registers		
Global Interrupt Mode Register (GMIR)	$812	Word, R/W
Timer Unit 1 Event Register (TER1)	$849	byte, R/W
SCC1 Mask Register (SCCM1)	$88A	byte, R/W
Xilinx Control Registers		
Reset_Halt (RHR)	$01	Byte, R/W
Interrupt Register (IR)	$03	Byte, R/W
Interrupt Vector Register (IVR)	$05	Byte, R/W
Enable Memory Register (EMR)	$07	Byte, W

Figure 2 VMEbus Address Map

All addresses in figure 2 are offsets from the I/O and memory base address of the carrier board.

Since the 68302 is Motorola architecture, the code downloaded into the SRAM must follow the same byte ordering as VMEbus.

See Programming section below and Motorola MC68302 documentation for register definition details.

NuBus Addressing

NuBus addressing requires computing the address from the byte addresses given above under VMEbus Addressing. The formula is:

NuBus byte address = (VMEbus byte address * 2) – 1

All byte data is still transferred on data lines D7..D0.

Words addresses on the NuBus are the same as for VME. Word data is transferred on data lines D15..D0.

Interrupt mapping is a function of the selected carrier board. See your IP carrier board User Manual for more information.

ISA (PC-AT) Bus Addressing

The figure 2 in VMEbus section is redefined in figure 3 for ISAbus addressing. ISAbus implements Intel byte ordering architecture as VMEbus implements Motorola. The byte ordering in Intel architecture is reverse of Motorola. The formulas for VMEbus to ISAbus byte ordering are:

ISAbus even byte address = VMEbus even address - 1
ISAbus odd byte address = VMEbus odd address +1

All resources on the IP are mapped in the IP memory space except Xilinx control registers which are mapped in the I/O space.

Register Name	68K Address	Access
MC68302 Registers		
Global Interrupt Mode Register (GMIR)	$812	Word, R/W
Timer Unit 1 Event Register (TER1)	$848	byte, R/W
SCC1 Mask Register (SCCM1)	$88B	byte, R/W
Xilinx Control Registers		
Reset_Halt Register (RHR)	$00	Byte, R/W
Interrupt Register (IR)	$02	Byte, R/W
Interrupt Vector Register (IVR)	$04	Byte, R/W
Enable Memory Register (EMR)	$06	Byte, W

Figure 3 ISAbus Address Map

The above formulas must be used when downloading code into the SRAM.

Interrupt mapping is a function of the selected carrier board. See your IP carrier board User Manual for more information.

See Programming section below and Motorola MC68302 documentation for register definition details.

Programming

General

The IP-CM302 upon power-up or system reset initializes the MC68302 to total-reset. Total-reset is a state at which the 68302 is held with its Reset and Hold pins at logic zero. During total-reset, the SRAM is ready to be downloaded with code. After download, one of the control registers in Xilinx must be configured to release the 68302 from its state of total-reset and begin to take reset exception vector .

The SRAM respond to 68302 chip select signals CS0 and CS1. As soon as the 68302 is released from the state of total-reset, CS0 by default activates to fetch code from the SRAM beginning at address zero. CS1 may be activated by the code to partition part of the SRAM as program data or data descriptor buffers for communication channels. The access to SRAM is at six wait states by default; however, the code should change the number of wait states to one for faster SRAM access. Due to SRAM speed, zero wait state is not allowed.

During the 68302 operation, the host CPU may access any IP resources by becoming the 68000 bus master through arbitration. Arbitration is handled by the hardware and it is transparent to the host. The host simply address the resources at addresses mapped by the the 68302 code. The resources are SRAM, the 68302 registers and on-chip RAM.

The host CPU may access the SRAM during 68302 operation by driving the address bus per configured Base Registers (BR0 and BR1) and Option Registers (OR0 and OR1) of 68302 to activate CS0 and CS1. Address bit A23 and function codes FC2..FC0 are driven to zero and 110 respectively to the 68302 during the host access to the SRAM. This must be taken into consideration when configuring BR0, BR1, OR0 and OR1. The host CPU accesses the SRAM with the exact number of wait states programmed in the OR0 and OR1 registers. The EMWS bit in the SCR register must be left to its default value of zero.

The SRAM is 256 Kbyte in size and therefore only decode for address bits A17..A1. Because of this minimum address decoding, the SRAM is repeated every 256 Kbyte block during total-reset access or CS0 and CS1 activation. The user may configure the carrier board and CS0 and CS1 to avoid repetition.

The E1 shunt not installed prohibits access to the SRAM by the host CPU. Any SRAM access attempt results in IP not acknowledging the cycle. The Xilinx control register EMR must be written the hex value of $AD by the host CPU to remove this prohibition. This scheme is useful to prevent operating systems to map the SRAM into their memory search table. E1 installed allows the host CPU to access the SRAM after power-up or system reset. E1 must be configured prior to IP power-up.

The 68302 BAR and SCR registers are configured by the code to map the on-chip RAM and peripheral registers into 4 Kbyte block of address space. The 4 kbyte block should not overlap the CS0 and CS1 address space since any overlapping effectively reduces the SRAM size. The host CPU may drive this address space to access the entire 4 Kbyte. During host CPU access, the address bit A23 is driven to zero. This should be taken into consideration when configuring BAR and SCR. The host CPU accesses the 4 Kbyte space including the BAR and the SCR asynchronously with three wait states. The user should clear the SAM bit of SCR (default after power-up) for this requirement. The EMWS bit of the SCR must be left to its default value of zero.

Read-modify-write cycle to the SRAM is supported by the 68302 core processor. The RMC signal of 68302 must be programmed to prevent the arbiter from issuing bus grants

until the completion of an 68302-core-initiated read-modify-write cycle. Read-modify-write cycles from the host CPU is not supported.

The IP does not acknowledge any host CPU access which results in 68302 bus error (BERR). The 68302 bus error may be reported to the host CPU via interrupt.

The 68302 watchdog timer expiration may be reported to the host CPU via interrupt for appropriate action. One possible action by the host CPU is asserting total-reset to the 68302.

Four general purpose I/O lines PB11..PB8 are available at the I/O connector. These lines may be programmed with interrupt capability.

An external clock may be provided to the 68302 TIN1 pin. The external clock if needed should be provided via the I/O connector. See the I/O Pin Wiring section.

The power-up initialization of the IP takes 70 milliseconds during which any access to the IP by the host CPU is not acknowledged. The host CPU may read the IP ID PROM as verification of successful IP initialization.

The above discussion is intended to inform the user of the features specific to the IP. This discussion complements the MC68302 User Manual. Therefore the user should consult the MC68302 User Manual for a broader and more specific information on 68302.

Register Definition

Refer to the MC68302 documentation for specific information about all of the registers and ports in the 68302 register map except for the Xilinx control registers. The functions of Xilinx control registers are explained below.

Reset_Halt Register (RHR)

Data Bit	7	6	5	4	3	2	1	0
Rd / Wrt	-	-	-	-	-	-	HALT	RESET

Figure 4 IP Xilinx Reset_Halt Register (RHR)

The Reset_Halt Register is written by the host CPU to assert and negate Reset and Halt signals to the 68302. The RESET and HALT bits must be written binary value 00 or 11 at once. The user must write binary value 11 to negate the 68302 from its state of total-reset and begin executing code from the SRAM. The 68302 is initialized to state of total-reset by writing the binary value 00. Total-reset also negates any pending interrupts or bus request from the host CPU to the 68302. Data written to the most six significant bits are ignored and data read from these bits are undefined. The initial value of these two bits after power-up or system reset is binary value 00.

Interrupt Register (IR)

Data Bit	7	6	5	4	3	2	1	0
Rd / Wrt	-	-	-	-	-	IRQ1	IRQ6	IRQ7

Figure 5 IP Xilinx Interrupt Register (IR)

The Interrupt Register is written by the host CPU to generate interrupts to the 68302. Please refer to the Interrupt section for description of interrupts and Interrupt Register

bits. Data written to the most five significant bits are ignored and data read from these bits are undefined. The initial value of these three bits after power-up or system reset is binary value 000.

Bit [0] IRQ7 [R/W]
Bit 0 set high generates interrupt on level 7. This bit is cleared by hardware during interrupt vector ID fetch cycle or upon total_reset.

Bit [1] IRQ6 [R/W]
Bit 1 set high generates interrupt on level 6. This bit is cleared by hardware during interrupt vector ID fetch cycle or upon total_reset.

Bit [2] IRQ1 [R/W]
Bit 2 set high generates interrupt on level 1. This bit is cleared by hardware during interrupt vector ID fetch cycle or upon total_reset.

Interrupt Vector Register (IVR)

Data Bit	7	6	5	4	3	2	1	0
Rd / Wrt	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0

Figure 6 IP Xilinx Interrupt Vector Register (IVR)

The Interrupt Vector Register may be written or read by the host CPU or the 68302. This register provides the interrupt vector ID to the host CPU during interrupt acknowledge cycle. The initial value of this register after power-up or system reset is hex value $FF. Total_reset does not alter the content of this register.

Bits [7..0] IV7..0 [R/W]
IV7 is the most significant and IV0 is the least significant bit.

Enable Memory Register (EMR)

Data Bit	7	6	5	4	3	2	1	0
Wrt	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0

Figure 7 IP Xilinx Enable Memory Register (EMR)

If E1 shunt is not installed, this register must be written the hex value of $AD prior to code download. This allows access to the SRAM by the host CPU. If E1 installed, this register is meaningless.

Bit [0] EM0 [W] : set to high
Bit [1] EM1 [W] : set to low
Bit [2] EM2 [W] : set to high
Bit [3] EM3 [W] : set to high
Bit [4] EM4 [W] : set to low
Bit [5] EM5 [W] : set to high
Bit [6] EM6 [W] : set to low
Bit [7] EM7 [W] : set to high

Code Example

The following Motorola MC68000 example code implements some of the IP requirements as discussed above.

```
* Register initialization values before execution:
*
* PC = 00000400 USP = 00020000
*
* Registers and Parameters
     BASE      EQU    $080000  ; Offset address
     BAR       EQU    $0F2     ; Base Address Register
     SCR       EQU    $0F4     ; System Control Register
     BR0       EQU    BASE+$0830; Base Register 0
     OR0       EQU    BASE+$0832; Option Register 0
     BR1       EQU    BASE+$0834; Base Register 1
     OR1       EQU    BASE+$0836; Option Register 1

* Typical IP-CM302 Initialization Code
     ORG       $00000400       ; begin code at this address
     MOVE.W    #$2700,SR       ; SR = 2700, mask off interrupts

* Set Base Address = $080000
* Now all 68302 on-chip RAM and Registers begin at address $08 0000
     MOVE.W    #$0080,BAR      ; BAR = 0080

* Set System Control Register
     MOVE.L    #0,SCR          ; SCR = 0, SAM = 0, EMWS = 0

* Partition 258 Kbyte SRAM between CS0 and CS1 into 128 Kbyte block size
each. * CS0 block is code and read only. CS1 block is program data and
read/write
     MOVE.W    #$3FC3,OR0      ; OR0 = 3FC3, CS0 one wait state, 128 Kbyte
     MOVE.W    #$C001,BR0      ; BR0 = C001, CS0 read only
     MOVE.W    #$3FC0,OR1      ; OR1 = 3FC0, CS1 one wait state, 128 Kbyte
     MOVE.W    #$C043,BR1      ; BR1 = C043, CS1 read and write

* More Demonstration Code
LOOP MOVE.W    #$ABCD,$02F000 ; Store $ABCD into CS1 block
     MOVE.W    $02F000,$080000    ; Store the same value into on-chip RAM
     BRA       LOOP            ; Keep repeating
```

While the 68302 is busy executing this code, the host CPU may access:

- Program code at address range of $00 0000 .. $01 FFFF as read only
- Program data at address range of $02 0000 .. $03 FFFF as read/write
- 68302 BAR and SCR registers at $0F2 and $0F4 as read/write
- 68302 registers and on-chip RAM at $08 0000 .. $08 0FFF

The above example requires one Mbyte of IP memory space beginning at address $000000.

Interrupts

The interrupt architecture of the IP is such that interrupts may be generated from the host CPU to the IP and vice versa. The IP may interrupt the host to report 68302 bus error, watchdog timer expiration, any communication channel receive or transmit buffer full or empty, and etc. The host CPU may interrupt the IP as mailbox, a common inter-processor communication.

The host CPU may generate interrupt to the 68302 on levels 1, 6, and 7. Xilinx register IR (figure 5 is repeated below) bits must be written logic 1 to generate interrupt to the 68302.

Data Bit	7	6	5	4	3	2	1	0
Rd / Wrt	-	-	-	-	-	IRQ1	IRQ6	IRQ7

The user may only generate interrupts one level at a time. The interrupt bit is cleared during the 68302 interrupt acknowledge cycle. The host CPU must make sure the three interrupt bits of IR are clear before generating another interrupt to the IP. This ensures the previous interrupt has been acknowledged by the 68302. All interrupts to the 68302 are autovectored on the level the interrupt was generated. The 68302 code should install interrupt handler routines for autovector levels 1, 6, and 7 before unmasking interrupts.

The 68302 generates interrupt to the host CPU by driving its CS3 signal and address bit A1 to logic 1. The 68302 may read the status of its generated interrupt by reading CS3 with address bit A1 high on the LSB of the data bus. This bit is cleared by the host CPU interrupt acknowledge cycle. This bit must be clear before the 68302 generates another interrupt to the host CPU. This ensures the previous interrupt has been acknowledged by the host CPU. The 68302 interrupts the host on level 0 of the IP logic interface.

The Xilinx Interrupt Vector Register (IVR) provides interrupt vector ID to the host CPU during its interrupt acknowledge cycle. The IVR is dual-ported by the host CPU and the 68302. Reads from the IVR may take place simultaneously. There is no arbitration for writes to the IVR. Therefore any potential simultaneous writes to the IVR corrupts it. It is recommended that the 68302 to have the sole write privilege to the IVR. The IVR may be written with different vectors to report different conditions to the host CPU. For example, the 68302 may report any buss error on a different vector than watchdog timer expiration. Of course the host CPU must have corresponding interrupt vector handlers for these different vectors from the 68302.

The IVR is written and read by the 68302 by driving its CS3 signal and address bit A1 to logic 0. The data transfer takes place on D7..D0. The CS3 signal must be configured for one wait state.

The 68302 Global Interrupt Mode Register (GMIR) must be configured as dedicated mode.

The IRQ1 is sourced from either the Xilinx or externally from the IP I/O connector to the 68302. The IRQ1 is an open collector signal from the Xilinx and pulled high by an on-board pull-up resistor. If IRQ1 is sourced from the I/O then the host CPU should avoid generating interrupt at this level. In the case where IRQ1 is not intended to be sourced from the I/O connector, the user must make sure IRQ1 is left unconnected. This is necessary to avoid electrical and spurious interrupts conflicts with Xilinx IRQ1.

If IRQ1 is asserted externally, the user may program one of the general purpose I/O lines in its interrupt routine handler to signal the interrupter to release the IRQ1.

The four general purpose I/O lines PB11..PB8 may be configured to have interrupt capability to the 68302. These lines are available at the I/O connector. See I/O Wiring section.

Code Example

The following Motorola MC68000 example code implements some of the IP requirements as discussed above. This example illustrates how interrupts are handled from the host CPU as well as interrupts generated by the 68302 to the host CPU.

```
* Register initialization values before execution:
*
* PC = 00000400 USP = 00020000
*
* Registers and parameters
     BASE      EQU    $080000   ; Offset address
     BAR       EQU    $0F2      ; Base Address Register
     SCR       EQU    $0F4      ; System Control Register
     BR0       EQU    BASE+$0830; Base Register 0
     OR0       EQU    BASE+$0832; Option Register 0
     BR1       EQU    BASE+$0834; Base Register 1
     OR1       EQU    BASE+$0836; Option Register 1
     BR3       EQU    BASE+$083C; Base Register 3
     OR3       EQU    BASE+$083E; Option Register 3
     GMIR      EQU    BASE+$0812; Global Interrupt Mode Register

* Typical IP-CM302 Initialization Code
     ORG       $00000400       ; begin code at this location
     MOVE.W    #$2700,SR       ; SR = 2700, mask off interrupts

* Set Base Address = $080000
* Now all 68302 on-chip RAM and Registers begin at address $080xxx
     MOVE.W    #$0080,BAR      ; BAR = 0080

* Set System Control Register
     MOVE.L    #0,SCR          ; SCR = 0, SAM = 0, EMWS = 0

* CS0 block is code and read only.
     MOVE.W    #$3FC3,OR0      ; OR0 = 3FC3, CS0 one wait state, 128 Kbyte
     MOVE.W    #$C001,BR0      ; BR0 = C001, CS0 read only

* Configure CS3 for one wait state
     MOVE.W    #$3FFC,OR3      ; OR3 = 3FFC, CS3 one wait state
     MOVE.W    #$C07F,BR3      ; BR3 = C07F, CS3 read and write

* Initialize the interrupt vector table for autovector level 1, 6, and 7
     MOVE.l    #$020100,$000064 ; Autovector 1 interrupt handler at
$020100
     MOVE.l    #$020200,$000078 ; Autovector 6 interrupt handler at
$020200
     MOVE.l    #$020300,$00007C ; Autovector 7 interrupt handler at
$020300

* Initilize the interrupt Vector Register IVR. The host should initilize
its
* interrupt vector table to this value
     Move.b    #$FC,$03E001    ; IVR = FC

* How to read IVR
     Move.b    $03E001,$080000     ; Copy IVR into on-chip RAM

* Configure Global Interrupt Mode Register to dedicated mode
     MOVE.W    #$F000,GMIR     ; GMIR = F000

* Get ready to receive interrupts from the host CPU
* Unmask interrupts
     MOVE.W    #$2000,SR       ; SR = 2000

* Generate interrupt to the host CPU
```

```
LP0  MOVE.W   #0,$03E002     ; CS3 and A1=1 assert as write

* Did my interrupt get acknowledged by the host CPU?
LP1  BTST     #0,$03E002     ; Is LSB set?
     BNE      LP0            ; Ready to generate another interrupt to
host
     BRA      LP1            ; Keep cheking till it is clear
     END
```

While the 68302 is busy executing this code, the host CPU may access:

- IVR at VME address of $03 E001 in byte as read/write
- Generate interrupt to itself by any write to VME address $03 E002
- Check the status of interrupt by examining D0 at VME address $03 E002

The above example requires one Mbyte of IP memory space beginning at address $000000.

Getting Started

This section is intended to assist the user to get to a quick start with the IP-CM302. The following steps install some minimum code into the IP SRAM and executes this code to perform some minimum functions. The user should consult the Programming Section for bit interpretation of registers given in this exercise. The required equipment are an IP-CM302 and a host CPU system with system debugger as software tool.

1) Install E1 shunt and power-up the IP-CM302 and download the following code into the IP-CM302 SRAM beginning at address $000400. Alternatively the user may directly assemble the code into the SRAM.

* Registers and Parameters

```
     BASE      EQU    $080000  ; Offset address
     BAR       EQU    $0F2     ; Base Address Register
     SCR       EQU    $0F4     ; System Control Register
     BR0       EQU    BASE+$0830; Base Register 0
     OR0       EQU    BASE+$0832; Option Register 0
     BR1       EQU    BASE+$0834; Base Register 1
     OR1       EQU    BASE+$0836; Option Register 1
```

* Typical IP-CM302 Initialization Code

```
     ORG       $00000400       ; begin code at this address
     MOVE.W    #$2700,SR       ; SR = 2700, mask off interrupts
```

* Set Base Address = $080000
* Now all 68302 on-chip RAM and Registers begin at address $08 0000

```
     MOVE.W    #$0080,BAR      ; BAR = 0080
```

* Set System Control Register

```
     MOVE.L    #0,SCR          ; SCR = 0, SAM = 0, EMWS = 0
```

* Partition 258 Kbyte SRAM between CS0 and CS1 into 128 Kbyte block size each. * CS0 block is code and read only. CS1 block is program data and read/write

```
     MOVE.W    #$3FC3,OR0      ; OR0 = 3FC3, CS0 one wait state, 128 Kbyte
     MOVE.W    #$C001,BR0      ; BR0 = C001, CS0 read only
     MOVE.W    #$3FC0,OR1      ; OR1 = 3FC0, CS1 one wait state, 128 Kbyte
     MOVE.W    #$C043,BR1      ; BR1 = C043, CS1 read and write
```

* More Demonstration Code

```
LOOP MOVE.W    #$ABCD,$02F000 ; Store $ABCD into CS1 block
     MOVE.W    $02F000,$080000    ; Store the same value into on-chip RAM
     BRA       LOOP           ; Keep repeating
```

2) Use debugger to write long words USP = $000f0000 to location $000000 and PC = 00000400 to location $000004.
3) Remove reset from the MC68302 by writing byte value $03 to IP I/O offset $01 (VME address). This begins code execution by 68302.
4) In the debugger verify the successful operation by reading the word value $ABCD at IP memory locations $02F000 and $080000. Write some word patterns at locations $02F002 and read the exact pattern back.
5) In the debugger access the 68302 registers at addresses according to the table 2-9 of the MC68302 User Manual.

Transition Module

The SBS offers an IP-CM302 transition module to implement physical layer interface for RS-232D, RS422, and RS-485 across three channels. Transition Module for ISDN (2B+D channels) is planned in future. The transition module is installed between the IP carrier board I/O connector and other communication equipment. The SBS transition modules may be conveniently installed in a 3U VME slot. A user may contact the factory for literature on IP-CM302 transition modules.

ID PROM

Every IP contains an IP PROM, whose size is at least 32 x 8 bits. The ID PROM aids in software auto-configuration and configuration management. The user's software, or a supplied driver, may verify that the device it expects is actually installed at the location it expects, and is nominally functional. The ID PROM contains the manufacturing revision level of the IP. If a driver requires a particular revision IP, it may check for it directly.

Standard data in the ID PROM on the IP-CM302 is shown in Figure 8 below.
For more information on IP ID PROMs refer to the IndustryPack Logic Interface Specification, available from SBS. The ID PROM is implemented in the Xilinx.

The location of the ID PROM in the host's address space is dependent on the carrier used. Normally for VMEbus carriers the ID PROM space is directly above the IP's I/O space, or at IP-base + $80. Macintosh drivers use the ID PROM automatically. RM1260 address may be derived from Figure below by multiplying the addresses given by two, then subtracting one. RM1270 addresses may be derived by multiplying the addresses given by two, then adding one.

3F 2D	(available for user)	
17	CRC for bytes used	(A4)
15	No of bytes used	(0C)
13	Driver ID, high byte	(00)
11	Driver ID, low byte	(01)
0F	reserved	(00)
0D	Revision	(A1)
0B	Model No IP-CM302	(46)
09	Manufacturer ID SBS	(F0)
07	ASCII "C"	(43)
05	ASCII "A"	(41)
03	ASCII "P"	(50)
01	ASCII "I"	(49)

Figure 8 ID PROM Data (hex)

I/O Pin Wiring

This section gives the pin assignment for the IP-CM302

The pin numbers given in Figures 9 correspond to numbers on the 50-pin IndustryPack I/O connector, to the wires on a 50-pin flat cable plugged into a standard IP carrier board, and to the screw terminal numbers on the IP-Terminal block. The third column gives the IP 68302-PQFP132 pin numbers as connected to the 50-pin IP I/O connector.

I/O Cable Pin Number	Signal	MC68302 Pin Number
1	RXD1/L1RXD	52
2	GND	
3	TXD1/L1TXD	80
4	GND	
5	RCLK1/L1CLK	82
6	GND	
7	TCLK1/L1SY0/SDS1	81
8	GND	
9	CD1/L1SY1	50
10	GND	
11	CTS1/L1GR	51
12	GND	
13	RTS1/L1RQ/GCIDCL	79
14	GND	
15	BRG1	76
16	GND	
17	RXD2/PA0	53
18	GND	
19	TXD2/PA1	54
20	GND	
21	RCLK2/PA2	55
22	GND	
23	TCLK2/PA3	56
24	GND	
25	CTS2/PA4	58
26	RTS2/PA5	59
27	CD2/PA6	60
28	BRG2/SDS2/PA7	61
29	GND	
30	RXD3/PA8	63
31	GND	
32	TXD3/PA9	64
33	GND	
34	RCLK3/PA10	65
35	GND	
36	TCLK3/PA11	66
37	GND	
38	CTS3/SPRXD	49
39	RTS3/SPTXD	78
40	CD3/SPCLK	77
41	BRG3/PA12	68
42	GND	
43	TIN1/PB3	111
44	GND	
45	IRQ1	97
46	GND	
47	PB8	118
48	PB9	119
49	PB10	120
50	PB11	121

Figure 9 IP-CM302 I/O Pin Assignment

IndustryPack Logic Interface Pin Assignment

Figure 10 below gives the pin assignments for the IndustryPack Logic Interface on the IP-CM302. Pins marked n/c below are defined by the specification, but not used on IP-CM302. Also see the User Manual for your IP Carrier board for more information.

GND	GND	1	26	
CLK	+5V		2	27
Reset*	R/W*	3	28	
D0	IDSel*		4	29
D1	n/c	5	30	
D2	MEMSel*		6	31
D3	n/c	7	32	
D4	INTSel*		8	33
D5	n/c	9	34	
D6	IOSel*		10	35
D7	n/c	11	36	
D8	A1		12	37
D9	n/c	13	38	
D10	A2		14	39
D11	n/c	15	40	
D12	A3		16	41
D13	IntReq0*	17	42	
D14	A4		18	43
D15	n/c	19	44	
BS0*	A5		20	45
BS1*	n/c	21	46	
n/c	A6		22	47
n/c	Ack*	23	48	
+5V	n/c		24	49
GND	GND	25	50	

Note 1: The no-connect (n/c) signals above are defined by the IndustryPack Logic Interface Specification, but not used by this IP. See the Specification for more information.

Note 2: The layout of the pin numbers in this table corresponds to the physical placement of pins on the IP connector. Thus this table may be used to easily locate the physical pin corresponding to a desired signal. Pin 1 is marked with a square pad on the IndustryPack.

Figure 10 Logic Interface Pin Assignment

Construction and Reliability

IndustryPacks were conceived and engineered for rugged industrial environments. The IP-CM302 is constructed out of 0.062 inch thick FR4 V0 material. The six copper layers consist of two signal layers on the top and bottom, and four internal layers. Two internal layers are dedicated to power and ground planes. Two additional layers are used for signal wiring.

The IndustryPack connectors are keyed, shrouded and gold plated on both contacts and receptacles. They are rated at 1 Amp per pin, 200 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

The IP is secured to the carrier with four metric M2 stainless steel screws. The heads of the screws are countersunk into the IP. The four screws provide significant protection against shock, vibration, and incomplete insertion. For most applications they are not required.

The IndustryPack provides a low temperature coefficient of 0.89 W/°C for uniform heat. This is based on the temperature coefficient of the base FR4 material of .31 W/m-°C, and taking into account the thickness and area of the IP. This coefficient means that if 0.89 Watts is applied uniformly on the component side, that the temperature difference between the component and the solder side is one degree Celsius.

Repair

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. SBS will not be responsible for damages due to improper packaging of returned items. For service on SBS products not purchased directly from SBS contact your reseller. Products returned to SBS for repair by other than the original customer will be treated as out-of-warranty.

Specifications

This section gives the technical specification for the IP-CM302

LSI Chip:	Motorola MC68302-FC running at 16 Mhz
Memory:	High-speed 256 Kbyte volatile Dual-Port SRAM
Number of Channels:	Three serial channels
I/O Interface:	Serial synchronous and asynchronous. RS232, RS-422, RS485
Protocols:	UART, HDLC, SDLC, BISYNC, DDCMP, V.110, V.120, X.25, ISDN (2B+D)
Performance:	See the MC68302 User Manual
Software Interface:	Control Register ID PROM MC68302
Initialization:	Hardware Reset initializes on-board logic and 68302. Software reset through control register also resets MC68302
Access Modes:	Word in memory space Word in I/O Space Word in ID Space Vectored interrupt
Wait States:	Depends on access type. 0 wait states are required for ID PROM and I/O access. See the body for memory access.
Interrupt:	Host CPU to IP on levels 1, 6, and 7. IP to host CPU on level 0
Onboard Options:	All Options are Software Programmable
Interface Options:	50 pin flat cable 50 screw terminal block interface User cable
Dimensions:	Standard Single IndustryPack width and length. 1.8 x 3.9 inches
Construction:	6 Layer Printed Circuit, Through Hole and Surface Mount Components. Programmable parts are socketed.
Temperature Coefficient:	.89 W/°C for uniform heat across IP
Test conditions	20°C, typical

Power Requirements	+5 VDC, 350 mA typ +12 VDC, 0 mA typ –12 VDC, 0 mA typ
Environmental	Operating temperature: 0 to 70°C Humidity: 5 - 95% non-condensing Storage temperature: –10 to +85°C

Appendix A

Appendix A discusses a working example of software that configures the MC68302 SCC3 for the UART mode. The code receives data from the UART receiver on a character-by-character basis and retransmits it out of the UART transmitter. The code, which runs on the IP-CM302, is an excellent starting point for understanding how to program the UART mode or to create a simple UART handler to support a debug monitor.

The code is very similar to to the **D.4.5 UART Code Listing** in **Appendix D** in the MC68302 User Manual. Few changes were made only to port this code to IP-CM302.

Purpose of the Code

This code is a character "echo" generator. The code receives and transmits data on a character-by-character basis, with interrupts generated on each character received. Although the MC68302 UART mode has much more flexibility and power than what is used by this code, many applications just require a simple low-speed UART channel for debugging during the design. SCC3 is often chosen to run at 9600 baud, 8-bit character, no parity, two stop bits, for this purpose as shown in the example.

The code is shown in standard M68000 assembler. Efficiency was compromised, when necessary, to enhance readability. The code is comprised of three basic parts. The first part initializes the MC68302 with everything required to set up SCC3 for UART mode. The initialization corresponds to the recommended order described in **4.5.7 SCC Initialization** in MC68302 User Manual. The second part is a set of loops waiting for data to arrive to be transmitted out of the SCC3. The third part is the SCC3 receive interrupt handler. Transmit interrupts are masked in this example.

Organization of Buffers

In the MC68302, There is no such thing as a receive register (Rx) or transmit (Tx). Rather, a flexible structure called a buffer descriptor (BD) is used. In this example, two Rx BDs and two Tx BDs are used. Each BD is set up to point to a one-byte location for data. Thus, the receiver and transmitter are double-buffered. The number of Rx or Tx BDs can be changed simply by changing the number of BDs initialized in the code (and two other lines documented in the code). The MC68302 on-chip RAM is used for storing the BDs.

Assumptions about the system

The code, which was run on the IP-CM302, assumes that the MC68302 peripherals are placed at the default position of $070000 (i.e. BAR is written with $0070). It also assumes that SCC3 is used. Either of the above assumptions can be modified if desired. The code was assembled with an OS-9 M68000 assembler. The code was then downloaded into the IP-CM302 SRAM by a M68040 processor.

UART Features Not Discussed

The following UART capabilities are not discussed in this example: fractional stop bit transmission, inserting flow control characters into the transmit data stream, recognizing

special control characters, using CTS and CD to control transmission and reception, use of an external clock, use of multiple bytes per buffer, sending idles before messages, recognizing addresses, freezing transmission, idle timeout, and others. Refer to **4.5.11 UART Controller** in the Motorola MC68302 User Manual for more detail.

UART Code Listing

```
* Simple UART "ECHO" Code
* Register Initialization values before execution:
*
* SSP = 00020000  PC = 00000400  SR = 2700
*
* Initialize SCC3 Receiver vector
* Address $000002A0 should be initialized to $0000F000
*
BAR             EQU     $0F2
SCR             EQU     $0F4
BR0             EQU     $70830
OR0             EQU     $70832
BR1             EQU     $70834
OR1             EQU     $70836
GIMR            EQU     $70812
IPR             EQU     $70814
IMR             EQU     $70816
ISR             EQU     $70818
PACNT           EQU     $7081E
SIMODE          EQU     $708B4
SCON3           EQU     $708A2
SCM3            EQU     $708A4
SCCE3           EQU     $708A8
SCCM3           EQU     $708AA
```

Download the following code into the IP SRAM beginning at address $400

*** IP configuration Code**

```
START:          MOVE.W          #$0070,BAR          ; BAR = 0070
* Base Address = $70000, so all MC68302 on-chip peripherals begin at
address
* $70xxx.

                MOVE.L          #$0,SCR             ; SCR = 0
                MOVE.W          #$3FC2,OR0          ; OR0 = 3FC2
                MOVE.W          #$C001,BR0          ; BR0 = C001
                MOVE.W          #$3FC0,OR1          ; OR1 = 3FC0
                MOVE.W          #$C043,BR1          ; BR1 = C043
* 256 Kbyte SRAM is partitioned into two block size of 128 Kbyte each:
* CS0 is active for address range of $000000..$01FFFF
* CS1 is active for address range of $020000..$3FFFFF
```

*** SCC3 Initialization Code**

```
                MOVE.W #$00A0,GIMR  ; GIMR = 00A0
                MOVE.W #$FFFF,IPR   ; clear IPR
                MOVE.W #$0300,PACNT ; PACNT = 0300
* Causes the SCC3 TXD3 and RXD3 pins to be enabled

                MOVE.W #$0,SIMODE           ; SIMODE = 0
* SCC3 is set up for NMSI (i.e. modem) operation

                MOVE.W #$00D8,SCON3 ; ~9600 baud at 16 Mhz
*               MOVE.W #$0102,SCON3 ; ~9600 baud at 20 Mhz
*               MOVE.W #$0142,SCON3 ; ~9600 baud at 25 Mhz
* Baud Rate generator is used for transmit and receive.

                MOVE.W #$171,SCM3           ; SCM3 = 171
* No parity. Normal UART operation. 8-bit characters. 2 stop bits
```

```
                MOVE.L #$10000000,$70640   ; Set up Tx BD0 Status and Count
                MOVE.L #$70000,$70644      ; Set up Tx BD0 Buffer and
Address
                MOVE.L #$30000000,$70648   ; Set up Tx BD1 Status and Count
                MOVE.L #$70001,$7064C      ; Set up Tx BD1 Buffer Address
```
* Set up Tx BDs

```
                MOVE.L #$90000000,$70600   ; Set up Rx BD0 Status and Count
                MOVE.L #$70002,$70604      ; Set up Rx BD0 Buffer Address
                MOVE.L #$B0000000,$70608   ; Set up Rx BD1 Status and Count
                MOVE.L #$70003,$7060C      ; Set up Rx BD1 Buffer Address
```
* Set up Rx BDs

```
                MOVE.W #$0,$70680   ; clear RFCR/TFCR (function Code setup)
                MOVE.W #$1,$70682   ; MRBLR = 0001 (one-byte receive
buffers)
                MOVE.W #$4,$7069C   ; MAX_IDL don't care since MRBLR = 1
                MOVE.W #$1,$706A0   ; BRKCR = 1
                MOVE.W #$0,$706A2   ; PAREC = 0000
                MOVE.W #$0,$706A4   ; FRMEC = 0000
                MOVE.W #$0,$706A6   ; NOSEC = 0000
                MOVE.W #$0,$706A8   ; BRKEC = 0000
                MOVE.W #$0,$706AA   ; UADDR1 = 0000
                MOVE.W #$0,$706AC   ; UADDR2 = 0000
                MOVE.W #$0,$706BE
                MOVE.W #$8000,$706B0 ; charater1 = 8000
                MOVE.B #$FF,SCCE3   ; clear SCCE3
                MOVE.B #$15,SCCM3   ; SCCM3 = 15
                MOVE.W #$0100,IMR   ; IMR = 0100. Allows SCC3 interrupts
only
                MOVE.W #$17D,SCM3   ; SCM3 = 017D

                CLR.L  D0           ; clear all used data registers
                CLR.L  D1
                CLR.L  D3
                CLR.L  D4
                CLR.L  D5
                CLR.L  D6

                MOVEA.L #$70600,A0  ; Load A0 as current Rx BD pointer
                MOVEA.L #$70002,A1  ; Load A1 as current "next Tx Byte"
pointer
                MOVEA.L #$70640,A2  ; Load A2 as current Tx BD pointer
                MOVE.W #$2000,SR    ; Enable interrupts. Stay in spvr mode
```

*** Transmit Code**

```
OUTLOOP:        CMPI.B #0,D6        ; something to send (is D6>=1?)
                BEQ.B  OUTLOOP      ; stay in outerloop if 0
                SUBQ.B #1,D6        ; decrement send status by 1 (0 = empty)
INLOOP:         BTST.B #7,(A2)      ; test Ready bit of Tx
                BNE.B  INLOOP       ; fall thru if 0, else wait in inner
loop
```

* The following sets up and sends out the transmit buffer

```
                ADDQ.W #1,D4        ; increment the number of chars
transmitted
                MOVEA.L A2,A3       ; A3 will be used to find Tx data buffer
                ADDQ.W #4,A3        ; inc A3 to point to Tx data pointer
                MOVEA.L (A3),A3           ; A3 now points to Tx Data Buffer
                MOVE.B (A1),(A3)    ; move char from Rx Buffer to Tx Buffer
                ADDQ.W #2,A2        ; inc A2 to point to byte count
                MOVE.W #1,(A2)      ; set Tx BD byte count to 1
                SUBQ.W #2,A2        ; A2 now points to beginning of Tx BD
                BSET.B #7(A2)       ; set Ready bit of Tx BD
```

* The following updates A2 to point to the next Tx BD

```
                BTST.B #5,(A2)      ; test Wrap bit
                BNE.B  REINIT2      ; if set, reinit A2 to 70640
                ADDA.W #8,A2        ; else inc A2 by 8 to next Tx BD
                BRA    CONT         ; jump to Continue on
REINIT2:        MOVEA.L #$70640,A2  ; reinitialize A2
```

```
CONT:          ADDQ.W #1,A1         ; inc A1 to next byte to send
               CMPA.L #$70004,A1    ; is A1 = 70004?
               BEQ.B  NEWA1         ; if so, go to NEWA1
               BRA    OUTLOOP       ; jump back to outerloop and wait
NEWA1:         SUBQ.W #2,A1         ; set A1 back to 70002
               BRA    OUTLOOP       ; jump back to outerloop and wait
```

The following code is the SCC3 interrupt routine handler. Download the handler code into the IP SRAM beginning at address $F000

*** SCC3 Interrupt Routine**

```
INTHDLR:       MOVEA.L #$70020,A4
               MOVE.B SCCE3,(A4)
               MOVE.B #$15,SCCE3    ; clear only BRK, BSY and RX in SCCE3
               BTST.B #2,(A4)       ; is BSY set?
               BNE.B  BUSY          ; jump to BUSY handler if set
BRKTEST:       BTST.B #2,(A4)       ; is BRK set?
               BNE.B  BREAK         ; jump to BREAK handler if set
RECTEST:       BTST.B #0,(A4)       ; is RX set?
               BNE.B  RECEIVE       ; jump to RECEIVE handler if set
               BRA    ALMDONE       ; jump to About Done (impossible)
BUSY:          ADDQ.B #1,D5         ; inc Busy counter (no receive buffers)
               BSET.B #7,(A0)        ; set Empty bit of current Rx BD
               BRA    BRKTEST       ; jump to test for BREAK
BREAK:         NOP                  ; this code ignores received breaks
               BRA    RECTEST       ; jump to test for RECEIVE
RECEIVE:       ADDQ.W #1,D3         ; increment number of chars received
               ADDQ.B #1,D6         ; charater ready to send
               ADDQ.W #1,A0         ; A0 points to Rx BD byte status
               CMPI.B #0,(A0)       ; does status = 0?
               BNE.B  BSTAT         ; jump to Bad Status if not 00
INCPTR:        SUBQ.W #1,A0         ; A0 points to beginning of Rx BD
               ANDI.W #$FF00,(A0)   ; clear out Rx BD status
               BSET.B #7,(A0)       ; set Empty bit of Rx BD
               BTST.B #5,(A0)       ; test Wrap bit
               BNE.B  REINIT0       ; if set, reinit A0 to 70600
               ADDA.W #8,A0         ; else A0 points to next Rx BD
               BRA    ALMDONE       ; jump to Almost Done
REINIT0:       MOVEA.L #$70600,A0   ; reinitialize A0
               BRA    ALMDONE       ; jump to Almost Done
BSTAT:         NOP                  ; bad status handler would go here
               BRA    INCPTR        ; jump back to Receive handler
ALMDONE:       MOVE.W #$0100,ISR    ; clear SCC3 bit in the ISR
               RTE                  ; end of interrupt handler
```