HIGH PERFORMANCE PC COMMUNICATION PORT USING THE Z182

To meet the demands of high speed communications on the Information Superhighway, Zilog's Z182-based PC Communication Port "COMPORT.S™" software code provides the user with increased data buffering capabilities for faster communication between PCs and modems.

INTRODUCTION

With the advent of the Information Superhighway, users will be able to access information quickly and easily by means of PC/Modem communications. However, user-friendly operating systems, capable of processing large amounts of data, become so performance intensive that data communication becomes a bottleneck. Although there have been many advances in modem and PC design, a satisfactory solution to the PC/modem communication bottleneck has not been found.

The PC/Modem communication bottleneck is caused by the PC standard serial cards used to connect to the modems. Conventional serial cards were designed to handle heavy traffic at only a sustained rate of 57.6 Kbps. This factor, combined with limited CPU bandwidth (due to complex operating systems), means users cannot take full advantage of the high throughput that today's modems can accommodate.

PC Week's labs (July 1992) analyzed the serial card bottleneck issue and their tests showed that a standard serial card could not reliably sustain data transfer over 19.2 Kbps. However, when using a 16550 UART loaded serial card, the user could expect 57.6 Kbps of reliable, sustained data transfer. This is great for V.32bis, but what about V.FAST and beyond?

Serial Port Limitations

If you look at a PC Standard Serial Card, you will find a 8250 or 16450 UART as well as RS-232 line drivers. The UART is the heart of the PC Serial Card, converting serial data from an external modem to parallel data which the PC can understand. The UART also converts the PC's parallel data to serial so the modem can understand it. Figure 1 shows a simplified Block Diagram of a single standard serial port.

The conventional PC Serial Card is not intelligent; in other words, it is completely dependent on the PC to manage data flow. The biggest drawback of a conventional PC serial card lies in its data handling.

The 16450 UART has a single receive shift register and receive buffer. When a byte of data fills the Receive Buffer, it requests an interrupt from the PC. Can a CPU respond quickly to interrupts in a multi-tasking system? The answer in most cases is no.

Complex operating systems need to provide a high level of internal bookkeeping to support interrupts in a complex task switching environment. As the operating system becomes more complex, the interrupt response time becomes extended.

If the PC does not read the data in the receive buffer, the recently assembled receive data cannot get shifted into the single receive buffer. When this happens, the next byte of incoming serial data will basically overwrite or "overrun" the byte that was not shifted into the receive buffer. Figure 2 describes the receive logic of a 16450 UART and an example of an overrun condition.

Serial Port Limitations (Continued)

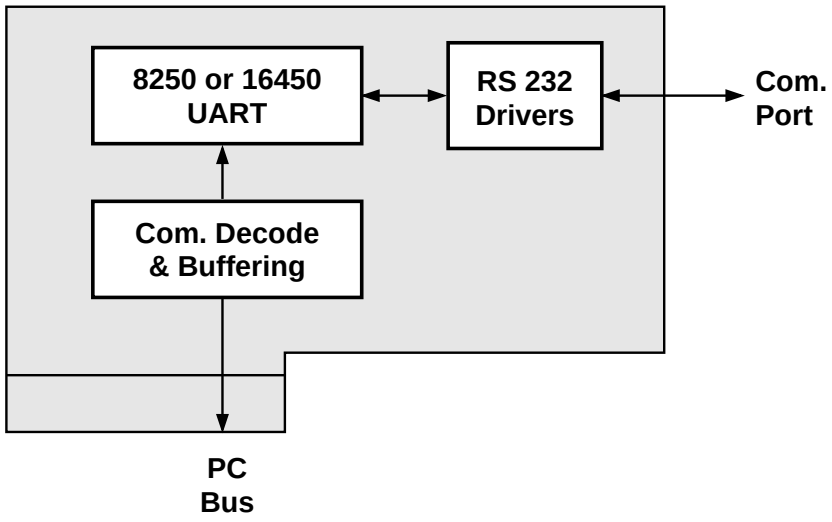

Figure 1. Conventional PC Com Port Block Diagram

Receiver Overrun occurs if the PC does not read DATA1 in time to prevent DATA 2, held in the Receive shift register, from being overwritten by DATA3 (Figure 2). When this condition occurs, the DATA2 information is completely lost. When receiving an executable file, the loss of one byte renders the whole file “non-executable” and worthless. Unless using high level error correcting protocols like ZMODEM, Overrun conditions become fatal to data communications. Even with the high level error correcting protocols handled by the PC, the number of errors and resends may cause the file transfer to slow to a snail's pace or completely abort.

The 16550 UART, offered on higher-end PC serial cards, has a 16 byte deep receive buffer FIFO, which provides some level of overrun immunity. In order to get full throughput from V.32bis external modems, modem manufacturers recommend the use of a 16550 loaded PC serial card. However, when moving to V.FAST, even the 16550 experiences overrun errors.

In order to prevent these overruns from occurring, it is evident that increased FIFO as well as an advanced data transfer technique is required for high speed modem communications. Many modem OEMs have defined innovative methods of allowing high speed modem communications such as use of the parallel Centronics Printer Port. An alternate solution is to use intelligent communication cards to combat the Com Port bottleneck.

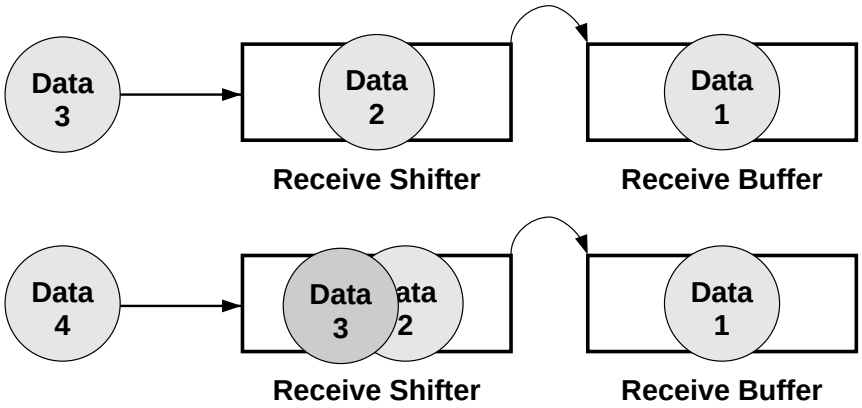

Figure 2. 16450 Overrun Condition

Z182 Intelligent Communication Port

Figure 3 illustrates the Z182-based Intelligent Com Port. Note that extra memory is used to provide a large data buffering area. The dual ASCI serial channels and bit I/O can be used for auxiliary features and RS-232 handshaking. The Ring Indicate and Data Set Ready inputs can be accommodated by the bit I/O since they are not supported in the ESCC cell. Appendix B shows a schematic of a single port Z182 intelligent communication card.

The majority of PC serial cards sold offer two serial ports. A secondary low-speed port (usually used for mouse) could be added by means of an inexpensive 8250 UART and a second set of RS-232 drivers.

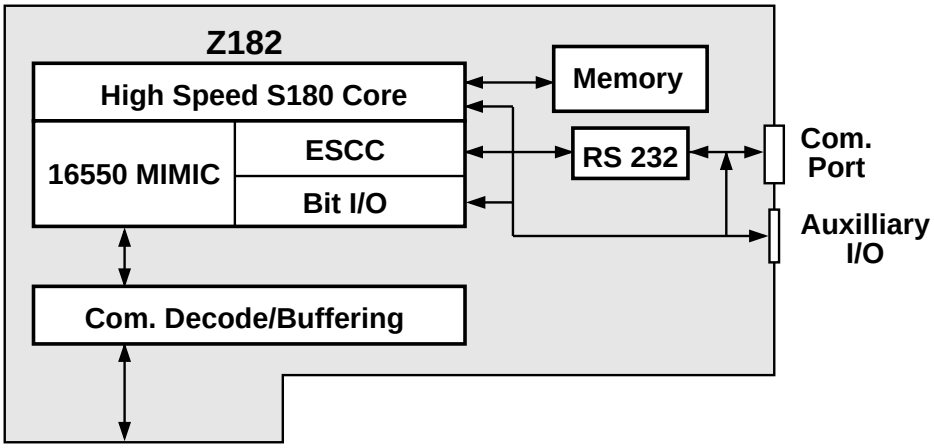

Figure 3. Z182-Based Intelligent Communication Card Block Diagram

Z182 Datacom Controller

The Z182 is ideally suited for PC communication cards as well as modems. The Z182 is a highly integrated, fully static, intelligent peripheral controller with the following features:

- The integrated 16550 MIMIC allows easy connection to the IBM PC BUS. To ensure PC Communication software compatibility, the MIMIC has the same register set as a 16550 UART.
- The ESCC cell is an enhanced industry-standard SCC. The ESCC features a 4-byte transmit FIFO and an 8-byte Receive FIFO. Multiple Protocol support allows the intelligent serial card to support synchronous protocols like X.25 as well as standard Asynchronous mode.
- There are 16 lines of general purpose I/O active during MIMIC usage. The I/O can be utilized to support extended handshaking, status LEDs, auxiliary PC I/O, etc.
- High Performance 33 MHz operation provides enough power for even the most demanding application. Although 16 MHz should provide more than enough bandwidth for a Com Port application, there is more bandwidth readily available for extra functionality/ features.
- Compact 100-pin QFP/VQFP packaging conserves space without sacrificing functionality.
- Low Power/Low Noise features are ideal for “GREEN PCs” and less demanding qualifications.

Figure 4 shows the block diagram of the Z80182. The high performance Static 180 core can intelligently buffer modem data as well as manage high-end communication port features (i.e. automatic flow control, protocol conversion, etc.). The 16550 MIMIC provides direct connection to the PC bus while the ESCC is used for serial data transfer to or from the modem.

Z182 Datacom Controller (Continued)

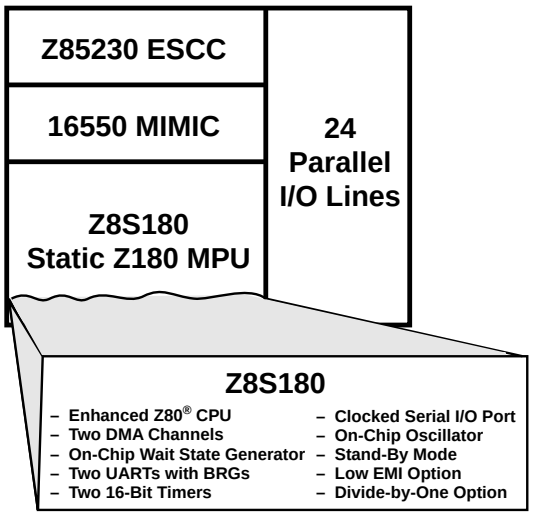

Figure 4. Z80182 Block Diagram

Z182 Intelligent COMPORT.S Software Code

The Z8018200ZCO board is designed to be a PC plug-in development card. It provides all hardware necessary to interface a PC to a modem without additional circuitry. The COMPORT.S code is written for the Z8018200ZCO's debug monitor. The ROM resident debug monitor can interface with the PC and upload assembled hex code into the evaluation board's RAM area and execute code from RAM.

The Z182 COMPORT.S code (Appendix A) is fully functional and has been tested to interface with external modems at a fixed 57.6 Kbps serial rate. As the code is in "core" form, many enhancements can be added to improve the board's functionality. However, the Z182 COMPORT.S code "as is" provides 1.5K bytes of transmit data buffering and 8K bytes of receive data buffering.

Receive buffering is important to avoid the Receiver Overrun problem and increase immunity to errors related to this condition. The Transmit buffer size is not as critical, but be aware that a large Transmit Buffer may cause problems. For example, if the Modem sends an XOFF to the PC, the transmitter needs to stop sending data to the modem. If the transmit buffer is large and full of data, the PC will not be able to prevent the contents of the large buffer from overflowing the modem's memory. This can be avoided by incorporating an automatic XON/XOFF support in the Z182 intelligent communication card. When an XOFF is detected by the Z182, it will automatically disable the MIMIC THR and ESCC Tx interrupts until an XON is detected.

COMPORT.S code was written around the ECHO182.S code and is functional with all current revisions of the Z182. The description of ECHO182.S can be found in Zilog Application Note "Z182 Programming the MIMIC Autoecho EchoZ182 Sample Code." In creating COMPORT.S code, the ECHO 182.S code core program was improved by adding an ESCC. Initialization and MIMIC Interrupt service routines are the same as that of ECHO182.S code. The only difference is the incorporation of two buffer areas instead of one. There is a separate buffer area for Receive and Transmit data. Registers D, E, H, and L are used for buffer pointers. The alternate register set is useful to allow each buffer to have its own set of complete registers.

ESCC Ch.A is programmed for 57.6 Kbps asynchronous data, 8-N-1 (8 bits, No parity, 1 stop). Most external modems should not have problems autobauding to this popular data rate/format. In addition to the ECHO182.S code there are two additional interrupt service routines labelled Txirq and Rxirq.

Txirq's task is to take data from the transmit data buffer and send it out serially via the ESCC Ch.A. Figure 5 describes the flow of this routine. When enabling the ESCC's transmit interrupt, the ESCC will only request an interrupt upon the transmitter BECOMING empty. Therefore, the main loop of COMPORT.S will serve to manually "KICK" the first character out of the ESCC transmitter. All remaining characters will be transferred by interrupts until the transmit memory buffer is empty.

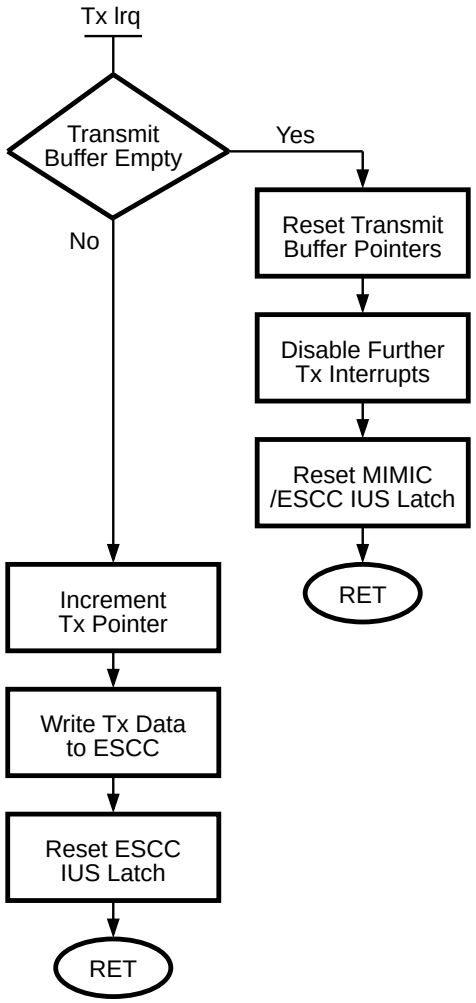

Figure 5. Tx Irq Interrupt Service Routine

The Rxirq service routine serves to load ESCC Rx data into the Rx memory buffer. A loop is provided to continue reading the ESCC until the Receiver is completely empty, reducing the possibility of receive overruns. Note that the Rxirq routine also enables RBR interrupts of the MIMIC. Figure 6 describes the flow of this routine.

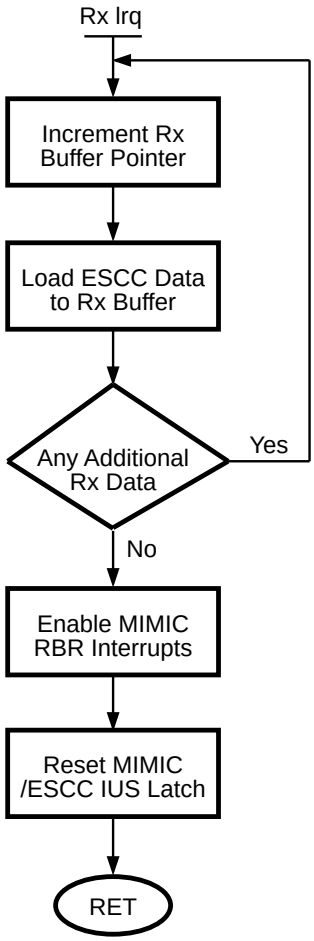

Figure 6. Rx Irq Interrupt Service Routine

Suggested Z182 COMPORT.S Code Enhancements

As discussed earlier, COMPORT.S code is the core module of a full featured communication port code. In order to have a complete Z182 com card, the following enhancements can be added to the code:

- **Complete RS-232 Handshaking Support.** Handshaking Signals RTS, CTS, DTR, DCD interface between ESCC and MIMIC can be added. In addition, Ring Indicate and Data Set Ready inputs can be accommodated by bit I/O.

- **Variable Data Rate/Framing Capability.** PC software programs the MIMIC's divisor latch for different baud rates. The Z180 could read these values and program the ESCC to reflect the PC selected baud rates. The PC can also choose different framing schemes (i.e., 7 bits, even parity, 2 stop) by programming the MIMIC. Z180 could read the framing and program the ESCC accordingly.

- **Burn Code into EPROM.** The COMPORT.S code is designed to become RAM loaded at logical address 0D400h. This can be transposed to ROM area by modifying ORG xxxxxh commands, defining a stack location, and modifying the i register to point at the transposed interrupt table location.

- **Maximum Utilization of ESCC/MIMIC FIFOs to Minimize Effects of Interrupt Latency Time.** The MIMICs RBR FIFO is 16 bytes deep. When the MIMIC's FIFO is enabled by the PC, the RBR can be filled with 16 bytes of data at once. A software register can be maintained to tally the fill level of the MIMIC's RBR FIFO to prevent overrun yet keep the MIMIC RBR filled. The same method can be used to keep the ESCC's 4-byte transmit FIFO filled with data. The Z182 DMAs could also be used to replace MIMIC THR and ESCC Rx interrupts.

Many other features can be added beyond what is required in a modem serial port. These extra features can be used to support "supercharged" modems (proprietary protocols and compression schemes) that are capable of transferring data above 115.2 Kbps. There are many other possible applications beyond modems that can utilize the following Z182 com card features:

- **Synchronous Protocol Support.** Async to Sync conversion to X.25, SDLC, Bisync

- **115.2 Kbps+ Asynchronous Support.** This feature can be used in proprietary analog modem designs that have greater than 4x compression using a 28.8 Kbps V.34 type connection. This can also be useful in digital modems that use compression schemes.

- **Intelligent XON/XOFF.** The PC no longer needs to manage XON/XOFF flow control. This can be useful to provide transparent data caching/serving with large memory-buffering areas.

- **Multiple Source Com Port Multiplexer.** PC Com Port drivers can be written to enable multiple input devices to be multiplexed onto a single Com Port and demultiplexed by the PC HOST software. This will allow many input devices to be used simultaneously (joysticks, midi, mice, pedals, etc.) while remaining plugged into a single card.

- **PC I/O Controller.** With 16 usable I/O lines on the Z80182 after using MIMIC, the lines can be used to control lights, mechanical devices, as well as analyze input from sensors. All special features can be selected by using the scratch register or upper bits of divisor latch as these are not commonly used in PC Communication Programs.

Overall, the COMPORT.S code is an excellent starting point for any application requiring communication between an outside device and a PC. Simple modifications to this code can be made to provide a full featured, intelligent Com Port with many additional enhancements.

APPENDIX A

```
;*********************************************************************
; COMPORT.S   written by Del Miranda, Zilog, Inc.
; This program uses the Z182 Mimic Cell to act as a buffered
; high performance communication port for the PC.
;************************************************************************************************** *include 182macro.lib
sccc:           equ     0e0h
sccd:           equ     0e1h
ascii_lf:       equ     00ah
ascii_cr:equ    00dh
null:           equ     00h
                org     0d400h
                di
                im 2
                ld a,   0d8h            ;setup int vector location
                ld i,a                  ;at d8xxh
                ld      hl,inittab
init0:  ld      a,(hl)
                cp 0ffh
                jr      z,initend
                ld      c,a             ;initialization
                inc     hl              ;goes to initialization table
                otim                    ;IO address first-data second
                jr init0                ;until ffh is given as address
inittab:
                db ccr                  ; /1 clock
                db 80h
                db itc                  ;disable interrupts first
                db 00h
                db icr                  ;standard IO mapping
                db 00h
                db dcntl                ;dma control - unnecessary
                db 50h
                db rcr                  ;refresh control - unneccessary
                db 3ch
                db omcr                 ;no Z80 ext peripherals, dont care
                db 3fh
                db itc                  ;enable interrupts now
                db 01h
                db pinmux               ;use /mreq for memory access
                db 00h
                db syscr                ;multiplex mimic, int vectors exported
                db 17h
                db romend               ;setup rom/ram boundries
                db 0ch                  ; ROM from 0000h to 0cfffh
                db ramstart             ; RAM from d000h to ffffh
                db 0dh
                db ramend
                db 0fh
                db cntla0               ;set up async port for 9600 baud
                db 64h                  ;given a 18.432MHz xtal in /1 mode
                db cntlb0
                db 22h
                db stat0                ;disable asci interrupts
```

APPENDIX A (Continued)

```
                db 00h
                db stat1
                db 00h
                db cntr                 ;disable csio ints
                db 0fh
                db tcr                  ;disable timer ints
                db 00h
                db dstat                ;disable dma ints
                db 32h
                db il                   ;set il=000
                db 00h
;********************ESCC Baud Rate Gen. Setup for MIMIC timers **********
; MIMIC TIMERS WILL UTILIZE BRG FROM ESCC CH.B,
; THE CLOCK PULSE WILL BE PROGRAMMED FOR /TRXCB OUTPUT.
                db sccbcnt              ;reset ESCC
                db 09h
                db sccbcnt
                db 0d0h
                db sccbcnt
                db 09h
                db sccbcnt
                db 00h
                db sccbcnt              ;timer low
                db 0ch
                db sccbcnt
                db 09eh
                db sccbcnt              ;timer high
                db 0dh
                db sccbcnt
                db 00h
                db sccbcnt              ;output baud rate to /TRxC
                db 0bh                  ;MIMIC reads this pin for timers
                db sccbcnt
                db 06h
                db sccbcnt              ;baud rate generator enable
                db 0eh
                db sccbcnt
                db 03h
                db sccbcnt              ;disable interrupts
                db 09h
                db sccbcnt
                db 00h
                db sccacnt              ;disable interrupts
                db 09h
                db sccacnt
                db 00h
                db 0ffh
;————————————————end of Z182 general initialization ————————————————
;    ESCC CH.A CONFIGURATION - FIXED 57,600BPS 8-N-1
;    CAN BE PROGRAMMED TO BE VARIABLE UPON PC SOFTWARE PROGRAM
;————————————————————————————————————————————————————————————————
```

```
initend:
                ld      a,09h           ;reset
                out0    (sccc),a
                ld      a,10000000b
                out0    (sccc),a
ld              a,01h                   ;no interrupt for recieve
out0            (sccc),a
ld              a,00h
out0            (sccc),a
ld              a,02h
out0            (sccc),a
ld              a,10h
out0            (sccc),a                ;00 int vector low
ld              a,03h
out0            (sccc),a
ld              a,11000001b             ;8 bits per char
out0            (sccc),a
ld              a,04h
out0            (sccc),a
ld              a,01000100b             ;1 stop , no parity
out0            (sccc),a
ld              a,05h
out0            (sccc),a
ld              a,01100000b             ;8 bits on transmit
out0            (sccc),a
ld              a,09h
out0            (sccc),a
ld              a,00h
out0            (sccc),a
ld              a,0ah
out0            (sccc),a
ld              a,00000000b
out0            (sccc),a
ld              a,0bh
out0            (sccc),a
ld              a,01010010b
out0            (sccc),a                ;baud rate generator
                                        ;used for rx and tx clock ld
                a,0ch
out0            (sccc),a
ld              a,08h                   ;0008h is time constant for
out0            (sccc),a                ;57.6K at 18.432MHz
ld              a,0dh
out0            (sccc),a
ld              a,00h
out0            (sccc),a
ld              a,0eh
out0            (sccc),a
ld              a,00000010b             ;baud gen input
out0            (sccc),a
ld              a,0eh
out0            (sccc),a
ld              a,00000011b             ;baud rate enable
out0            (sccc),a
ld              a,05h
```

APPENDIX A (Continued)

```
out0        (sccc),a
ld          a,01101000b
out0        (sccc),a                    ;enable transmit

            ld          a,10h
            out0        (sccc),a        ;enable status
            ld          a,01h
            out0        (sccc),a
            ld          a,11010000b
            out0        (sccc),a        ;int when rx buff full
            ld          a,09h
            out0        (sccc),a
            ld          a,00001001b
            out0        (sccc),a        ;main int enable
            ld          a,38h
            out0        (sccc),a
;————————————————MIMIC INITIALIZATION————————————————————
            ld hl,0da00h                ;transmit buffer locations
            ld de,0da00h                ;0da00-0dfffh  1.5Kbyte
                                        ;hl=mimicthrpointer
                                        ;de=escctxpointer
            exx
            ld hl 0e000h
            ld de,0e000h                ;receive buffer locations
            exx                         ;0e000-0ffffh 8.0Kbyte
                                        ;hl'=esccrxpointer
                                        ;de'=mimicrbrpointer
            ld a,05h                    ;disable mimic timers, INT mode 2
            out0 (mmcr),a               ;out 2 mode for HINTR line
            ld a,00h
            out0 (ivec),a               ;int vector of 070xh, x changes
            ld a,80h                    ;according to int condition
            out0 (iusip),a              ;reset highest MIMIC int under service
            ld a,28h
            out0 (0eah),a               ;setup RBR & THR FIFO timeouts
            out0 (0ebh),a
            ld a,0ah                    ;setup RBR&THR serial emulation timers
            out0 (0fah),a
            out0 (0fbh),a
            ld a,0ffh                   ;Modem Status Register set RI,DCD,CTS
            out0 (msr),a                ;these can be varied upon hardware
            ld a,020h                   ;enable RBR timeout, 1 byte THR trigger level
            out0 (fcr),a
;*************************************************************************
;Note: although setting a 1 byte THR interrupt trigger level means more
;interrupts for the Z182, some (if not all) 16550 PC code will not
;put more data in the THR buffer unless the THRE bit is set (transmit
;buffer is empty).  Setting the THR interrupt trigger level to 4,8, or
;14 bytes is suggested for proprietary designs where the application
;does not need to remain compatible to third party comms software.
;For use in modems a THR interrupt trigger level of 1 is suggested.
;*************************************************************************
;
```

```
                ld a,40h                ;set TEMT bit, PC software often reads this out0 (lsr),a
                ld a,0c5h               ;enable mimic timers, INT mode 2
                out0 (mmcr),a           ;out 2 mode for HINTR line.  Note
                                        ;that timers values are not changed
                                        ;while timer is running.
                ld a,0c0h
                out0 (mimie),a          ;enable MIMIC THR interrupts
;————————————————————end of MIMIC initializtion ————————————
loop:           ei                      ;constant looping, program root
                ld a,80h                ;enable interrupts
                out0 (iusip),a          ;reset MIMIC and ESCC IUS
                ld a,38h
                out0 (sccc),a
                ld a,e                  ;poll to check if there is data
                or a                    ;in transmit FIFO
                jr nz,loop
                ld a,l                  ;if there is data to send
                or a                    ;disable interrupts and force it out
                jr z,loop               ;of ESCC ch.A
                di
                call Txirq
                jp loop
;*************************************************************************
;               INTERRUPT SERVICE ROUTINE - RBRIRQ
;*************************************************************************
rbrirq:
;INT ROUTINE FOR RBR INTERRUPTS
;OCCURS WHEN RBR IS EMPTY
                exx
                ld a,l                  ;compare buffer pointer
                cp e                    ;if hl'=de', then get out
                jr z,out                ;this means buffer is empty
okay:           inc de
                ld a,(de)
                out0 (rbr),a            ;else, output data to RBR
                ld a,80h                ;reset highest ius
                out0 (iusip),a
                exx
                ei
                ret
out:                                    ;return to loop if not really
                ld a,h                  ;empty
                cp d
                jr nz,okay
                                        ;if buffer is empty then
                ld hl,0e000h            ;reset buffer pointers
                ld de,0e000h
                exx
                ld a,0c0h               ;disable RBR interrupts
                out0 (mimie),a          ;otherwise RBR will always interrupt
                ld a,80h                ;reset highest ius
                out0 (iusip),a
                ei
                ret                     ;return to loop
```

APPENDIX A (Continued)

```
;*********************************************************************
;                   INTERRUPT SERVICE ROUTINE - FIFOTHR
;*********************************************************************
;INT ROUTINE FOR THR FIFO INTERRUPTS
;READS ALL DATA IN FIFO UNTIL EMPTY
fifothr:
                    inc hl                          ;increment pointer
                    in0 a,(thr)                     ;write THR data to buffer
                    ld (hl),a
                    in0 a,(lsr)                     ;check to see if THR FIFO is empty
                    bit 5,a                         ;if not empty go back to fifothr
                    jr nz,notempt2
                    jr fifothr
notempt2:                                           ;else force TEMT bit
                    ld a,40h
                    out0 (lsr),a
                    ld a,0c0h                       ;enable  THR ints
                    out0 (mimie),a
                    ld a,01h                        ;enable ESCC Rx&Tx Interrupts
                    out0 (sccc),a
                    ld a,0d2h
                    out0 (sccc),a
                    ld a,80h                        ;reset highest MIMIC int under service
                    out0 (iusip),a
                    ei
                    ret                             ;return to loop
;*********************************************************************
;UNEXPLAINED INTERRUPT HANDLERS WORKAROUND
;REV C Z182 MIMIC MAY GIVE 00H FOR
LOWER VECTOR, RESET IUS IN THIS CASE
;*********************************************************************
uknirq:
                    ld a,80h                        ;workaround - dummy service routine
                    out0 (iusip),a
                    ei
ret ;*************************************************************
;                   ESCC CH.A DTE INTERRUPTS
;************************Tx ESCC Int*********************************
 Txirq:
txchk:  ld a,l                                      ;compare buffer pointer
                    cp e                            ;check if buffer empty
                    jr z,out1                       ;if so goto out1
okay1:  inc de                                      ;otherwise transmit data from
                    ld a,(de)                       ;buffer
                    out0 (sccd),a
                    ld a,38h                        ;reset escc ius
                    out0 (sccc),a
                    ei
                    ret
out1:               ld a,h                          ;return to loop
                    cp d                            ;if buffer is not really empty
                    jr nz, okay1
                    ld hl, 0da00h                   ;if tx buffer is empty, reset
```

```
                ld de, 0da00h           ;pointers and disable tx int
                ld a,80h                ;reset highest ius for mimic
                out0 (iusip),a
                ld a,01h                ;disable escc Tx interrupt
                out0 (sccc),a           ;or Tx int will keep interrupting
                ld a,0d0h
                out0 (sccc),a
                ld a,38h                ;reset highest ius for escc
                out0 (sccc),a
                ei
                ret                     ;return to loop
;************************Rx IRQ**********************************
;  ESCC CH.A  Receive Interrupt Service Routine
Rxirq:
                exx
readit:
                inc hl                  ;load received data into Rx buffer
                in0 a,(sccd)
                ld (hl),a
                in0 a,(sccc)            ;check if there is more data to read
                bit 0,a                 ;if so, read & load it
                jr nz,readit
                ld a,0d0h               ;enable mimic RBR interrupts
                out0 (mimie),a
                ld a,80h                ;reset highest mimic ius
                out0 (iusip),a
                ld a,38h                ;reset highest escc ius
                out0 (sccc),a
                exx
                ei
                ret
;************************INTERRUPT VECTOR TABLE************************
                        org 0d800h
                        dw uknirq       ;workaround, for NOINT vector
                        dw uknirq
                        dw uknirq
                        dw uknirq
                        dw uknirq
                        dw rbrirq       ;table entry for rbr empty interrupt
                        dw fifothr      ;table entry for timeout - disabled dw fifothr
                                        ;table entry for thr has data interrupt dw uknirq
                        dw uknirq
                        dw uknirq
                        dw uknirq
                        dw Txirq
                        dw uknirq
                        dw Rxirq
                        dw uknirq
```

APPENDIX A (Continued)

```
;*********************************************************
;*  File name - 182macro.lib ;*
;*  Macro libraly for Z180 new instructions for asm800  ;*
;*  1/26/89 Jim Nobugaki  ;*
;*  revised   7/14/92 Del Miranda  ;*
;*********************************************************
;Z180 System Control Registers
;ASCI Registers
cntla0:         equ     00h             ; ASCI Cont Reg A Ch0
cntla1:         equ     01h             ; ASCI Cont Reg A Ch1
cntlb0:         equ     02h             ; ASCI Cont Reg B Ch0
cntlb1:         equ     03h             ; ASCI Cont Reg B Ch1
stat0:          equ     04h             ; ASCI Stat Reg Ch0
stat1:          equ     05h             ; ASCI Stat Reg Ch1
tdr0:           equ     06h             ; ASCI Tx Data Reg Ch0
tdr1:           equ     07h             ; ASCI Tx Data Reg Ch1
rdr0:           equ     08h             ; ASCI Rx Data Reg Ch0
rdr1:           equ     09h             ; ASCI Rx Data Reg Ch1
;CSI/O Registers
cntr:           equ     0ah             ; CSI/O Cont Reg
trdr:           equ     0bh             ; CSI/O Tx/Rx Data Reg
;Timer Registers
tmdr0l:         equ     0ch             ; Timer Data Reg Ch0-low
tmdr0h:         equ     0dh             ; Timer Data Reg Ch0-high
rldr0l:         equ     0eh             ; Timer Reload Reg Ch0-low
rldr0h:         equ     0fh             ; Timer Reload Reg Ch0-high
tcr:            equ     10h             ; Timer Cont Reg
tmdr1l:         equ     14h             ; Timer Data reg Ch1-low
tmdr1h:         equ     15h             ; Timer Data Reg Ch1-high
rldr1l:         equ     16h             ; Timer Reload Reg Ch1-low
rldr1h:         equ     17h             ; Timer Reload Reg Ch1-high
frc:            equ     18h             ; Free Running Counter
;CPU Control Registers (Only for Z8S180)
ccr:            equ     1fh             ; CPU Control Reg.
;DMA Registers
sar0l:          equ     20h             ; DMA Source Addr Reg Ch0-low
sar0h:          equ     21h             ; DMA Source Addr Reg Ch0-high
sar0b:          equ     22h             ; DMA Source Addr Reg Ch0-b
dar0l:          equ     23h             ; DMA Dist Addr Reg Ch0-low
dar0h:          equ     24h             ; DMA Dist Addr Reg Ch0-high
dar0b:          equ     25h             ; DMA Dist Addr Reg Ch0-B
bcr0l:          equ     26h             ; DMA Byte Count Reg Ch0-low
bcr0h:          equ     27h             ; DMA Byte Count Reg Ch0-high
mar1l:          equ     28h             ; DMA Memory Addr Reg Ch1-low
mar1h:          equ     29h             ; DMA Memory Addr Reg Ch1-high
mar1b:          equ     2ah             ; DMA Memory Addr Reg Ch1-b
iar1l:          equ     2bh             ; DMA I/O Addr Reg Ch1-low
iar1h:          equ     2ch             ; DMA I/O Addr Reg Ch1-high
bcr1l:          equ     2eh             ; DMA Byte Count Reg Ch1-low
bcr1h:          equ     2fh             ; DMA Byte Count Reg Ch1-high
dstat:          equ     30h             ; DMA Stat Reg
dmode:          equ     31h             ; DMA Mode Reg
```

```
dcntl:          equ     32h             ; DMA/WAIT Control Reg
;System Control Registers
il:             equ     33h             ; INT Vector Low Reg
itc:            equ     34h             ; INT/TRAP Cont Reg
rcr:            equ     36h             ; Refresh Cont Reg
cbr:            equ     38h             ; MMU Common Base Reg
bbr:            equ     39h             ; MMU Bank Base Reg
cbar:           equ     3ah             ; MMU Common/Bank Area Reg
omcr:           equ     3eh             ; Operation Mode Control Reg
icr:            equ     3fh             ; I/O Control Reg
pinmux:         equ     0dfh            ;Interrupt edge/pin mux register
scr:            equ     0f7h            ;MIMIC scratch register
romend:         equ     0e8h            ;rom boundry
ramstart:       equ     0e7h            ;ram start boundry
ramend:         equ     0e6h            ;ram end boundry
syscr:          equ     0efh            ;system pin control
mmcr:           equ     0ffh            ;mimic master control register
iusip:          equ     0feh            ;int under service register
mimie:          equ     0fdh            ;mimic interrupt enable reg
ivec:           equ     0fch            ;mimic int vector
msr:            equ     0f6h
lsr:            equ     0f5h
fcr:            equ     0ech
rbr:            equ     0f0h
thr:            equ     0f0h
;PIO registers
ddra:           equ     0edh            ;data direction register a
ddrb:           equ     0e4h            ;data direction register b
ddrc:           equ     0ddh            ;data direction register c
dra:            equ     0eeh            ;port a data
drb:            equ     0e5h            ;port b data
drc:            equ     0deh            ;port c data
;ESCC registers
sccacnt:        equ     0e0h            ;ESCC control channel A
sccad:          equ     0e1h            ;ESCC data channel A
sccbcnt:        equ     0e2h            ;ESCC contol channel B
sccbd:          equ     0e3h            ;ESCC data channel B
?b              equ     0
?c              equ     1
?d              equ     2
?e              equ     3
?h              equ     4
?l              equ     5
?a              equ     7
??bc            equ     0
??de            equ     1
??hl            equ     2
??sp            equ     3
slp                     macro
                db      11101101B
                db      01110110B
                endm
mlt             macro   ?r
                db      11101101B
                db      01001100B+(??&?r AND 3) SHL 4
```

APPENDIX A (Continued)

```
                endm
in0             macro   ?r, ?p
                db      11101101B
                db      00000000B+(?&?r AND 7) SHL 3
                db      ?p
                endm
out0            macro   ?p, ?r
                db      11101101B
                db      00000001B+(?&?r AND 7) SHL 3
                db      ?p
                endm
otim            macro
                db      11101101B
                db      10000011B
                endm
otimr           macro
                db      11101101B
                db      10010011B
                endm
otdm            macro
                db      11101101B
                db      10001011B
                endm
otdmr           macro
                db      11101101B
                db      10011011B
                endm
tstio           macro   ?p
                db      11101101B
                db      01110100B
                db      ?p
                endm
tst             macro   ?r
                db      11101101B
                ifidn   <?r>,<(hl)>
                db      00110100B
                else
                ifdef   ?&?r
                db    00000100B+(?&?r AND 7) SHL 3 else
                db    01100100B
                db    ?r
                endif
                endif
                endm
                .list
end
```

APPENDIX B

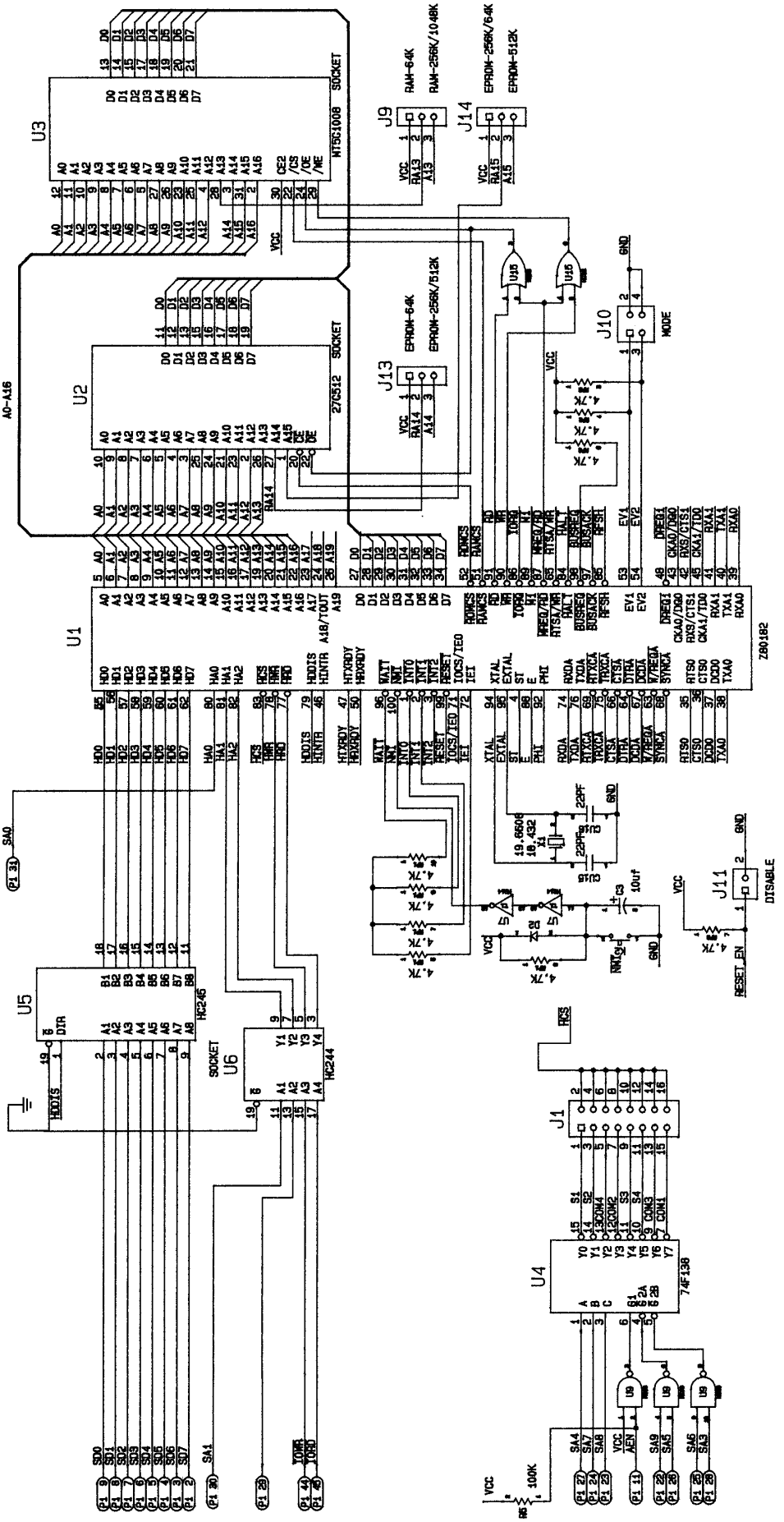

Figure 7a. Serial Communication Board Schematic

APPENDIX B (Continued)

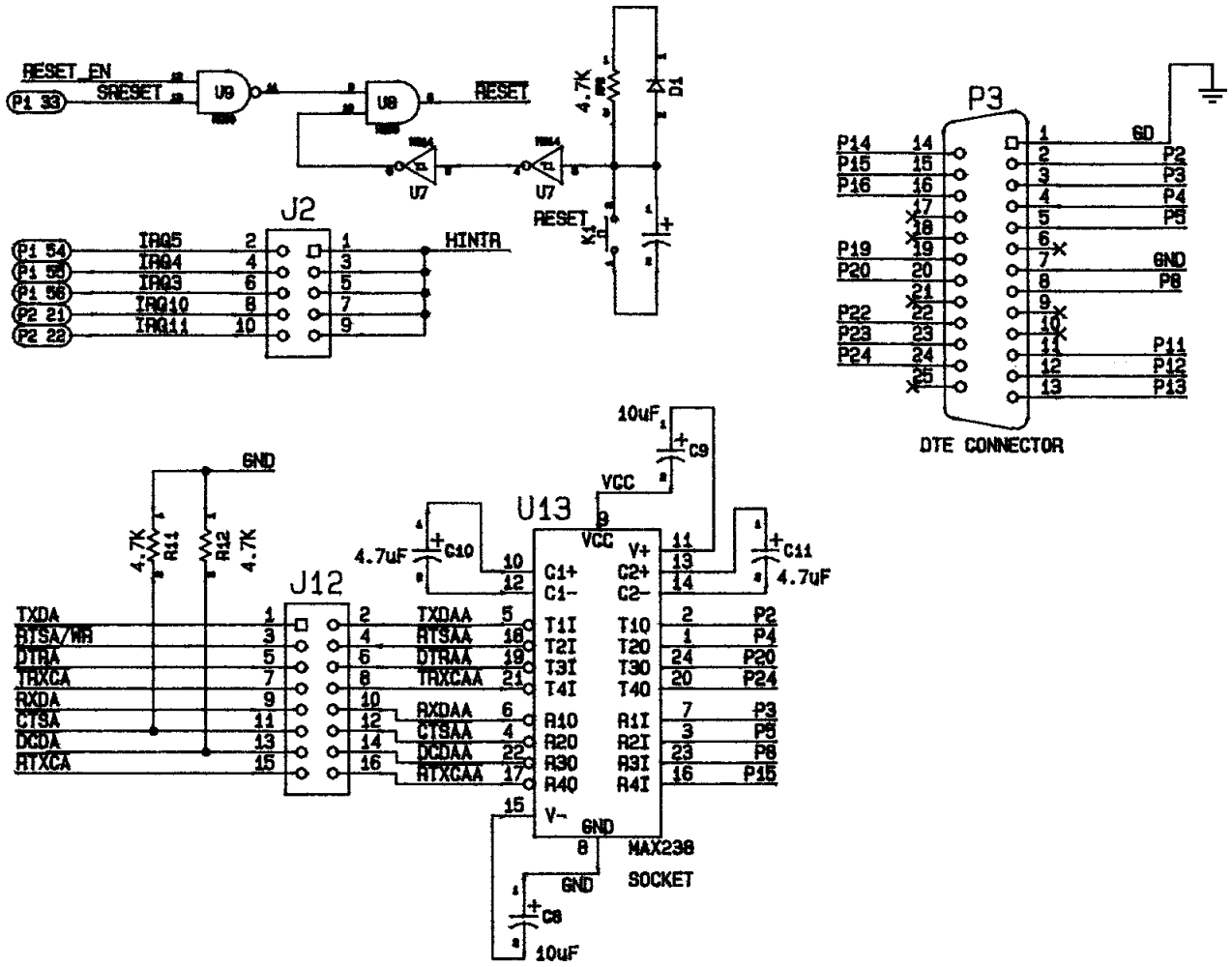

Figure 7b. Serial Communication Board Schematic

Zilog, Inc. 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
Telex 910-338-7621
FAX 408 370-8056
Internet: http://www.zilog.com