Z182 PROGRAMMING THE MIMIC AUTOECHO ECHOZ182.S™ SAMPLE CODE

In a conventional Internal Modem design, a 16550 UART is required for the Modem to communicate with the PC. The Z182 took that 16550 UART register set and using Zilog's unique Superintegration™ technology, combined it with the widely used Z8S180 core to make an intelligent peripheral controller ideally suited for PC Modems.

INTRODUCTION

The Z80182 is a general-purpose datacommunications controller for internal, external, PCMCIA, and wireless modem/fax. The 16550 Mimic Core allows the use of the Z80182 in internal PC applications, with direct access to the PC XT/AT bus.

The 16450/550 UART is known as the PC Standard Serial Communication Device. PC communication software is written to communicate to a 16450/550 UART directly connected to the PC bus. In order to maintain compatibility with PC communication software, modem designers must implement a 16450/550 UART to interface the modem controller with the PC Bus. This is shown in Figure 1 and labeled as the conventional internal modem. A better idea would be to integrate the PC standard UART register set into a modem controller. Zilog addressed this need by superintegrating the 16550 Mimic into the Z182.

The Z182 design shown in Figure 1 demonstrates how the Z182 can be used to lower chip count, which also means lower costs and higher reliability. The 16550 Mimic is not a UART, but transfers byte data (as opposed to serial data) between the PC and Static Z180. Therefore, much higher performance can be achieved using the Z182's Mimic.

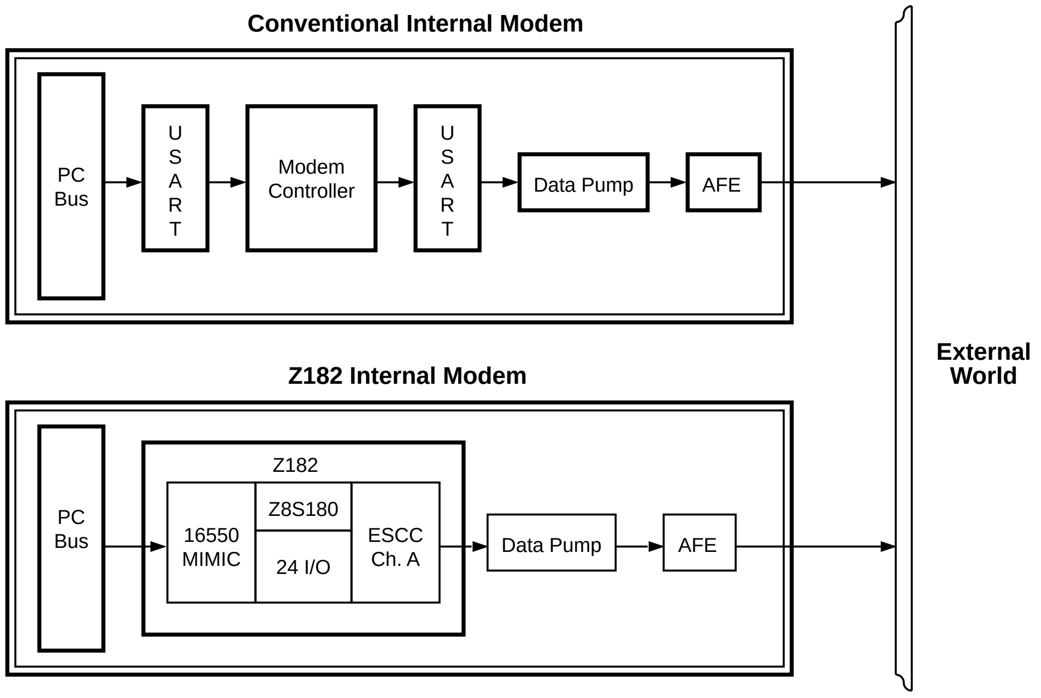

Figure 1. Modem Configurations

PURPOSE

In order for the Z182's Mimic to function properly in a modem application, it must be programmed properly. The Mimic is programmed by both the PC as well as the internal Static Z180 core of the Z182. There are many varieties of PC communication software for programming and managing the Mimic from the PC side. Therefore, a modem designer need not worry about programming the Mimic from the PC side.

Because the Mimic is an interface between the PC and the Static Z180 modem controller, the Mimic must also be programmed and managed by the Static Z180. This application note was written to address the Mimic programming requirements of the Static Z180. It is advised that the user become familiar with the Z182 device by reviewing the Z80182 Preliminary Product Specification in Section 3 of this databook. A complete listing and explanation of the Mimic registers can be found in that document. This Application Note is meant to be a supplement to the Product Specification.

The basic task of the Mimic is to transfer data between the Modem Controller and the PC Bus. In order to satisfy this basic task, a programming example is required that exercises the task of transferring data with the use of Z182's Mimic. The ECHO182 code was developed to serve as a no-frills example of how to program the Mimic.

The ECHO182 program is merely an autoecho driver for the Z182 Mimic Cell. It's purpose is to take data transmitted from the PC and "echo" it back to the PC.

Figure 2 is an illustration of the ECHO182 driver function. A user would send data by either pressing keys on the keyboard or initiating a file transfer. The transmission of this data is managed by the PC Communication software. The PC Communication software routes that data to the assigned Com Port connected to the PC Bus. The Z182's Mimic interface retrieves that data through its Transmit Holding FIFO and allows it to be handled by the Static Z180 controller. The ECHO182 driver would then take that data and store it in buffer RAM.

As data is being stored in RAM, the Static Z180 controller will also take that data and "ECHO" it back to the PC by means of the Mimic's Receive Buffer FIFO. The PC Communication software will then take the data echoed by the Z182 and display it on the computer monitor.

The overall effect of the driver is demonstrated when a key is pressed or file is transferred. What is transmitted will be echoed back and displayed on the monitor. Data that is received from the Z182 through the Mimic Receive Buffer FIFO is displayed on the computer monitor by the PC communication software. Additionally, a terminal can be connected to the Z182's ASCI0 UART to display data that the PC transmits to the Mimic Transmit Holding FIFO.

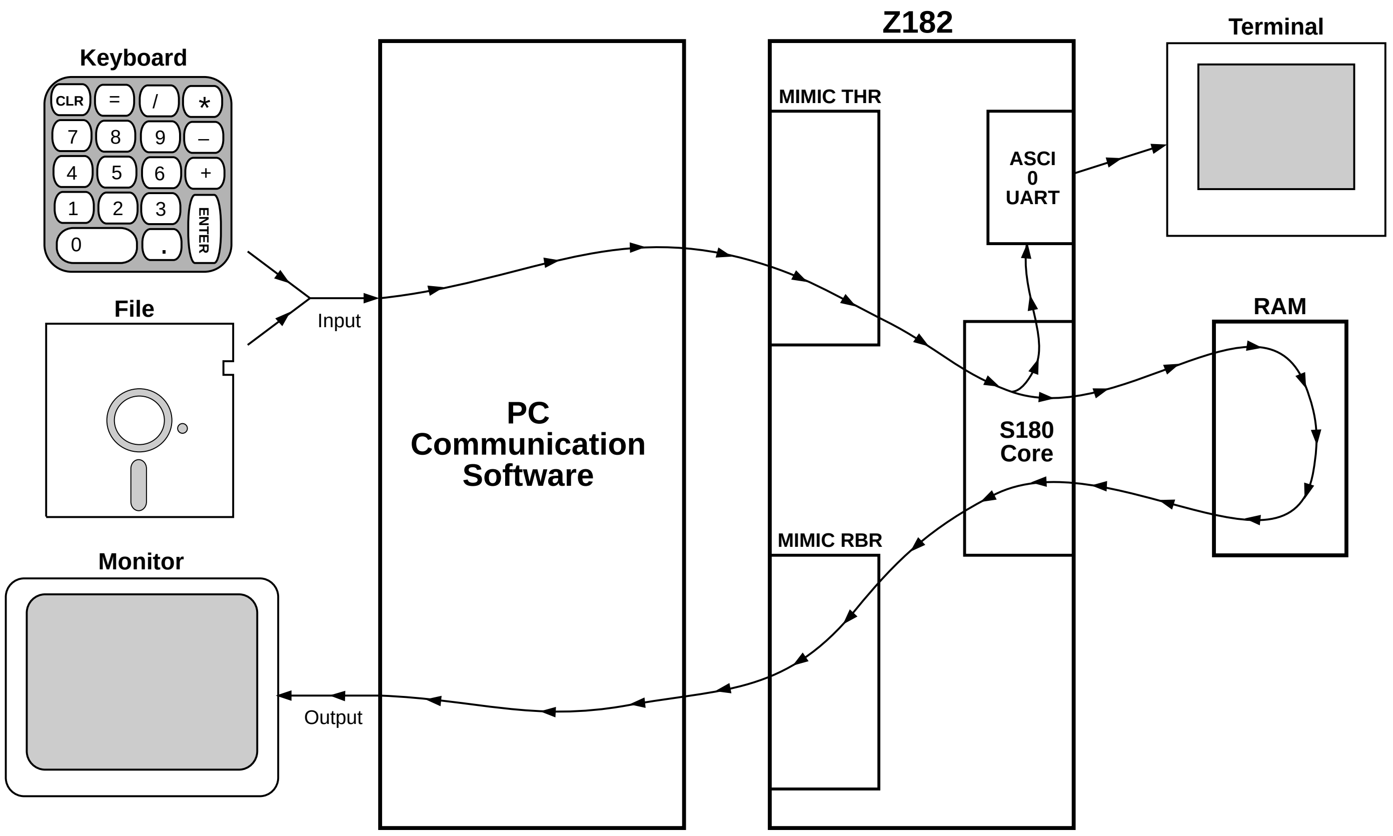

Figure 2. Echo Z182 Autoecho Code

ANALYSIS OF ECHO182 CODE

The complete listing of ECHO182 source code can be found in Appendix A of this Application Note. It is highly recommended that a copy of this code is made and kept handy throughout the study of this document. References to the ECHO182 code will be made frequently.

The first part of any code deals with the initialization of the device. Figures 3a and 3b display a flowchart of the Z182's initialization sequence. The first block involves initialization of the stack area as well as the location of the interrupt vector table.

The next section of code serves to initialize Static Z180 MPU registers, Z180 peripherals, ASCI0 UART (used to display transmitted data to terminal), as well as ESCC Channel B. Note that the ECHO182 code uses the OTIM command in a loop to do a block initialization of the Z182. An initialization table is then labelled as INITTAB which features a simple format of register then offset. The end of the table is denoted by 0ffH in the register field (the Z182 has no register addressed as 0ffh, so using 0ffh as end of table marker is fine).

Within the initialization sequence, the SYSCR (System Configuration Register, address = xxefH) is programmed such that Mimic pins are multiplexed over ESCC Channel B and Z180 peripheral pins. The Mimic pins are required to interface to the PC Bus. The Interrupt Daisy chain is set such that the Mimic has higher priority interrupts than the ESCC interrupts. Also, ASCI Channel 0 is multiplexed over Port B bit I/O pins (ASCI0 UART is used for monitoring data transmitted from PC to Mimic's Transmit Holding FIFO).

The next functional block of Figure 3a. flowchart shows initialization of ESCC Ch. B. The Mimic Timers need to have a clock source (more explanation on the timers later). The Mimic Timers get its clock pulse from the /TRxCB pin of the Z182. We can program the ESCC Ch. B Baud Rate Generator to put a periodic pulse on the /TRxCB pin. Since the Mimic timers count with an 8-bit constant and the ESCC provides baud rate generation programmability of 16 bits, the programmer has an access to a total of 24 bits to control the time constants of Mimic timers.

During the initialization of ESCC Ch. B, we set the Baud Rate Generator time constant to 01DEh. The output of the Baud Rate Generator is routed to the /TRxCB pin so that it can be used by the Mimic Timers. The formula for ESCC Baud rate generation is as follows:

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2X\ (\text{Desired Rate})} - 2$$

Using a 18.432 MHz XTAL in divide-by-two mode, the clock frequency is 9.216 MHz. The time constant of 01DEh is 478 in decimal. Using the above formula, you get roughly 9.6 kHz pulse rate. The time constant for this example was picked randomly for 9.6 kHz. In a modem application, the time constant can be picked such that the desired rate represents the actual baud rate.

In that case, the /TRxCB pin will pulse periodically for every bit time for 9600 baud data transmission. Once we have initialized ESCC Ch. B's Baud Rate Generator, we can enable it. Note that the Baud Rate Generator output cannot be observed since it is internally multiplexed out and replaced with HA0 input pin of the Mimic.

Once the Static Z180, Z180 peripherals, and ESCC Ch. B, are initialized, the code proceeds to send a start-up message to ASCI0 UART. The start up message will be displayed on a terminal connected to ASCI0 given that the terminal configuration matches the ASCI0 transmission rate. (ASCI is programmed for 9600 baud in this code.) The message is useful in indicating that the Z182 general initialization was successful. This concludes the general Z180/ESCC initialization.

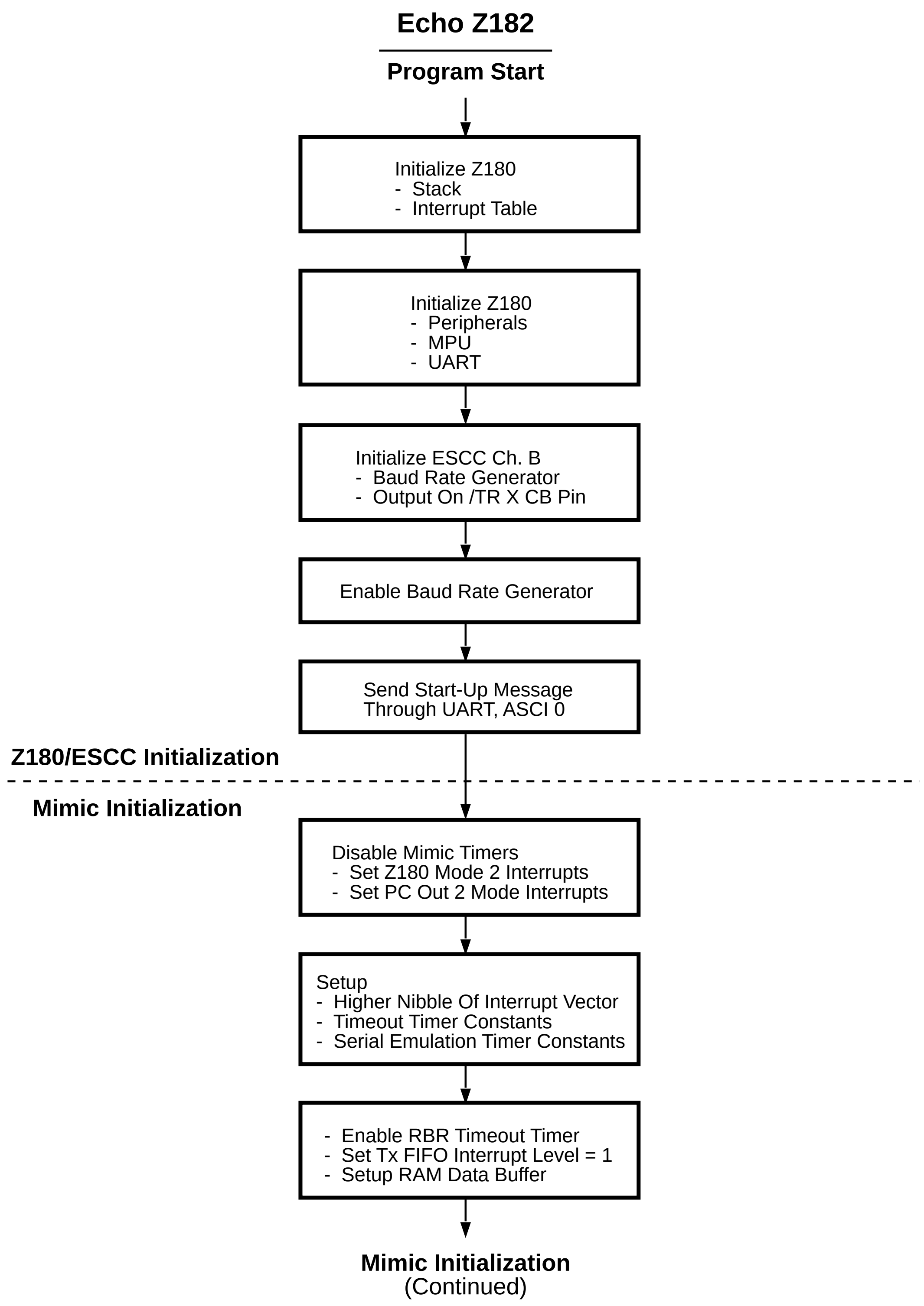

Figure 3a. Echo Z182 Initialization Flowchart

ANALYSIS OF ECHO182 CODE (Continued)

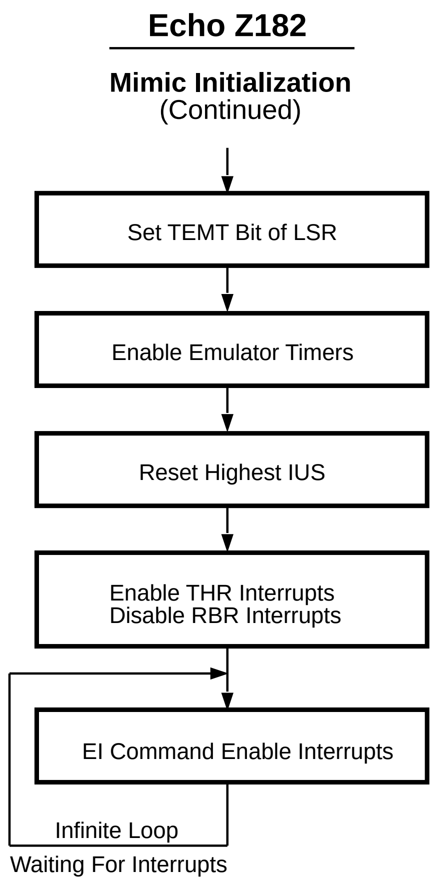

Figure 3b. Echo Z182 Initialization Flowchart

INITIALIZING THE MIMIC

The next functional blocks of Figures 3a. and 3b. describe the initialization of the Mimic core. The first step is to program the MMCR (Mimic Master Control Register, xxFFH). This register sets DMA transfers, Mimic Timers, PC Host Interrupts and Mimic/Static Z180 interrupt mode. Within modem applications, data transfers with the Mimic are generally interrupt driven, so DMAs are disabled. Errors can result when using the DMAs with the Mimic. Please consult the Z80182 Customer Procurement Specification errata prior to utilizing DMAs with Mimic.

The MMCR also programs the way interrupts are handled. The VIS (vector includes status) bit should be programed to the same interrupt mode as ESCC Ch. A interrupts. It is strongly suggested that Mode 2 be used since this mode translates to higher interrupt handling performance.

The PC Host interrupts can be fully driven in Normal Mode, wired AND, or OUT 2 Mode. PC Communication software usually programs the OUT 2 bit to enable PC Modem Interrupts so it is suggested that OUT 2 mode is used for modem designs.

Note that the Mimic timers are disabled here. Whenever one wishes to modify Mimic timer constants, the timers should be disabled. In other words, avoid changing Mimic timer constants on the fly (while the timers are operating).

The code then proceeds to program the IVEC Register (xxFCH) of the Mimic. In Mode 2 type interrupts, we can only modify the upper nibble of this register. The lower nibble is controlled by the type of interrupt that occurs. The contents of the IVEC and interrupt status determine what interrupt vector is exported during an interrupt acknowledge cycle.

By writing 80h to the IUSIP register (xxFEH), we reset the highest interrupt under service. ***This must be done once during initialization and at the end of each Mimic Interrupt Service Routine.*** Upon start-up, the RBR interrupt will be pending and we must reset the highest IUS to allow other interrupts to occur. If this is not done, further interrupts will be prevented from occurring.

INITIALIZING MIMIC TIMERS

The Mimic has four timers; Receive Timout Timer, Transmit Timout Timer, Transmitter Serial Emulation Timer, and Receiver Serial Emulation Timer. These timers are basically counters that count the number of pulses that occur on the /TRxCB. (Recall that we programmed the ESCC Ch. B Baud Rate Generator to output pulses on /TRxCB.) Each of the timers have a register that holds a timer constant.

The Receive Timeout Timer Constant contains an 8-bit constant for emulation of the 16550's four character timeout feature. The 16550 feature will cause a timeout if no FIFO transaction occurs within four character times of a byte entering the FIFO. Therefore, we should emulate four character times. If we assume a character is 10 bits in length (including start, stop, parity bits) we can say four character times is approximately 40 bit times. If ESCC Ch. B is programmed such that /TRxCB pulses at one bit time intervals, then we need only program the Timer constant to be 40 decimal (28h). The same should be done for the Transmitter Timeout Timer Constant.

In the ECHO182 example, we set the Timeout feature to be 28h. The only proviso here is that the Timout Timer Constants should be greater than the Transmitter Time Constants. Otherwise, you run the possibility of having a RBR or THR interrupt prior to the Data Ready/Available bit being set. The next set of timers are the Receive and Transmit Serial Emulation Timers. Since the Mimic transfers data in bytes (as opposed to serial bits with real 16550), data transfers between the Static Z180 and PC can occur at very high rates. The Serial Emulation Timers have been added to alleviate any software/hardware problems that higher data throughput can impose.

A true 16550 will add a delay due to shifting of serial data. The serial emulation timers can be used to slow down the data transfer just as if the Mimic had to shift data in and out. For example, if the modem is configured for 10 bits per character (with start bit and stop bit), we can set the serial emulation timers to emulate the shifting of 10 bits. If the ESCC Ch. B /TRxCB output pulses at one bit time, we can emulate the shifting of 10 bits by putting 10 decimal (0ah) in the counter register. If the timer constant is made longer than this, the data transfer efficiency will suffer. The Receive and Transmit timers are set to 0Ah in ECHO182. This causes the data transfer rate to drop down to 960 bytes per second (9.6 kHz TRxCB pulse/10 bit time emulation).

One may wish to make use of the advantages of parallel data transfer and make the serial emulation timer constant smaller than what is suggested. This is fine, but care should be taken such that the data throughput is manageable by PC communication software and hardware. Optimization of this type can only be accomplished through trial and error.

ENABLING FIFOS, TIMERS, AND INTERRUPTS

The next functional blocks of the flowchart of Figure 3a. works to enable the appropriate Mimic FIFOs, timers, and interrupts. The FIFO Status and Control Register (address xxECH) enable the Mimic transmit/receive timeouts as well as setting the THR FIFO Interrupt trigger level.

It is strongly suggested that THR XMIT Timeout is disabled and THR XMIT trigger level is set for 1 byte. Ideally, we would want to utilize the FIFO features by using a greater trigger level and enabling the timeouts. Doing this would allow less interrupts and more efficiency/performance by the Static Z180 core.

Unfortunately, PC communication software does not allow for use of the THR FIFO features.

Most, if not all, PC communication software transmits data as follows:

- Sends 1 byte to THR.
- Checks the THRE empty bit and polls until transmit holding register is empty.
- Once the THRE bit is logic 1, then another byte can be transferred. (This also applies to 16550 compatible communication software.)

So, if the PC writes 1 byte of data in the THR and the Mimic THR interrupt level is set at four, the Static Z180 will never read the data in the THR FIFO until there are four bytes. The PC will not write any more data to the THR the FIFO is empty. The Mimic will then appear to be in a locked state indefinitely, since the FIFO will never be empty nor reach four bytes. ***To avoid this problem, a THR level of 1 with no timeout is recommended for modem applications.***

Another precaution relating to PC communication software involves the polling of the TEMT bit. This bit of the LSR signifies that the transmitter has completed serial shifting of all data in the THR FIFO. PC communication software will poll this bit after sending a block of data during a file transfer. If this bit is not set, the PC software will not send any more data until TEMT is set. The 16550/450 will also reset this bit when data is loaded into the THR FIFO. Therefore, to maintain a file data transfer, one should set the TEMT bit when all data has been read from the THR FIFO. In the ECHO182 example, we force the TEMT bit in the Transmit Interrupt Service Routines.

Earlier in the code, we disabled the Mimic timers. ***The Mimic timers must be disabled, prior to any modification of timer constants.*** Since we have already programmed the timer constants, we can now enable them.

Also, Mode 2 vectored interrupts are chosen on the Static Z180 side, as well as OUT 2 Control Mode interrupts on the PC side. The PC will then program Mimic's OUT 2 bit to function as a Mimic-PC Interrupt enable.

Finally, we enable the Mimic Interrupts. This is accomplished by programming of the IE register (address xxFDH). MIE (Master Interrupt Enable) is set as well as setting the THR IRQ enable. Since the example deals with echoing data back, we must wait for the PC to transmit data to echo. Therefore, we only need to enable the THR interrupt which will cause a static Z180 interrupt when the PC writes one byte (remember we set the trigger level to one byte) to the THR FIFO. ***Do not enable RBR Interrupts, otherwise RBR Interrupts will occur indefinitely until data is put into RBR.***

In a modem application, FCR, DLM, DLL, LCR, MRC interrupts are enabled to flag the Static Z180 whenever the PC wants to change these values (baud rate, bits/char., parity, stop bits, etc.). The Z180 can then poll these registers and change the ESCC programming to match what serial link characteristics are requested by the PC communication software. Since the ECHO182 is not connected to any serial links, the interrupts are disabled and DLM, DLL, LCR, and MCR registers are ignored.

In the Z80182, a race condition exists that causes errors to occur. Although infrequent, the Interrupt vector is modified improperly for FCR, DLM, DLL, LCR, and MCR Interrupts. Instead of the corresponding modified vector, bits 1, 2, and 3 of the interrupt vector are forced to 000b. The workaround is to provide an interrupt service routine for the occasion when bits 1, 2 and 3 of the interrupt vector is 000b (NO IRQ vector of Mimic).

In the ECHO182 program, the interrupt service routine for NO IRQ would serve only to reset the highest interrupt under service, allowing other interrupts to occur. The interrupt service routine is labelled UKNIRQ. It's address is placed in the first table entry of Mimic's interrupt vector table. In a modem application, the NO IRQ interrupt service routine should poll DLM, DLL, LCR, and MCR registers and modify ESCC as if all of these registers were modified. ***The interrupt service routine should always reset the highest IUS of the Mimic at the end of every service routine corresponding to a Mimic interrupt as well as after any Mimic initialization.***

Now that Mimic initialization is completed, we can enable the Static Z180 interrupts by using the EI command. The ECHO182 will go into an infinite loop, jumping out to serve interrupts and jumping back in when done.

INTERRUPT SERVICE ROUTINES

The ECHO182 code contains three interrupt service routines; FIFOTHR, RBRIRQ, and UKNIRQ. In the previous section, the UKNIRQ was discussed and is used as a workaround if bits 1, 2, and 3 of the Mimic interrupt vector are returned as being 000b (NO IRQ vector). RBRIRQ and FIFOTHR relate to the transfer of data between the PC and Mimic.

FIFOTHR - Transmit Holding Register Interrupt
When the Transmit Holding Register of the Mimic is empty, it generates an interrupt to the PC given that the interrupt is enabled. The Mimic interrupts the PC to request for data to be transmitted to the external world. The PC will then satisfy this by writing data to the Mimic's THR register. The management of PC interaction with Mimic is handled by PC communication software and is not discussed in this Application Note.

When the Mimic THR register contains data it will generate an interrupt to the Static Z180. The FIFOTHR interrupt service routine flowchart is shown in Figure 4. FIFOTHR transfers all data in the THR FIFO to the RAM Buffer. Whenever the PC writes data to Mimic's THR FIFO, a THR interrupt is generated to the Static Z180.

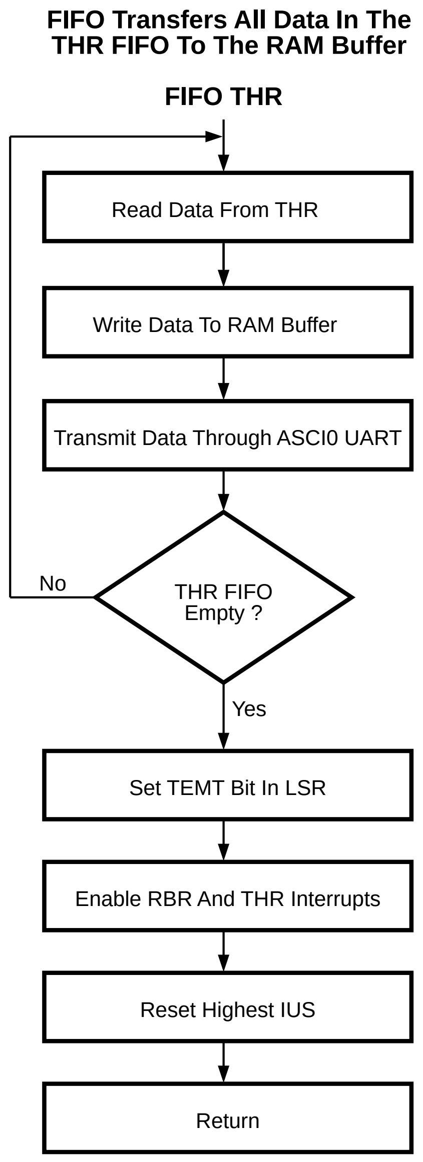
FIFO Transfers All Data In The
THR FIFO To The RAM Buffer
FIFO THR
Read Data From THR
Write Data To RAM Buffer
Transmit Data Through ASCI0 UART
THR FIFO Empty ?
No
Yes
Set TEMT Bit In LSR
Enable RBR And THR Interrupts
Reset Highest IUS
Return

Figure 4. FIFOTHR Interrupt Service Routine

INTERRUPT SERVICE ROUTINES (Continued)

The first functional blocks of the FIFOTHR Interrupt Service Routine causes the Static Z180 to read data from the Mimic's THR register. This data is then stored in the RAM buffer area. The data retrieved from the Mimic's THR register is also transmitted out of ASCI0 UART. If an optional Terminal is connected to the ASCI0, the terminal displays the data that the PC Communication software is transmitting to the THR FIFO.

The Static Z180 then polls the THRE bit in the LSR register of the Mimic to determine if there is any data left in the THR FIFO. If the Transmit Holding Register Empty bit is logic zero, then there must still be data in the THR FIFO.

If the THR FIFO is not empty, the code will read another byte from the Mimic's THR buffer and store that data in the next location of the RAM buffer. The process of reading the THR buffer and storing in RAM will continue until the THR FIFO is empty.

If THR FIFO is empty, the code will discontinue any further loading of the RAM buffer. The TEMT bit will be asserted to assure that PC communication software is allowed to transmit additional frames of data. If TEMT is not set, the PC communication software will wait for the Transmit Empty bit is set before transmitting additional data.

Next, RBR and THR interrupts are enabled. When there is no data in RAM buffer, we disable the RBR interrupt from retrieving any data from the RAM buffer. Since the FIFOTHR service routine just placed data in the RAM buffer, we can enable the RBR interrupt so that the data can be echoed back to the PC. Remember that all Mimic interrupt service routines require a Reset Highest IUS prior to returning from the interrupt.

The FIFOTHR interrupt service routine can read all the data contained in the THR FIFO. If the THR trigger level is set to one, there will only be one byte of data in the THR FIFO for each interrupt. Therefore, the THR interrupt need only transfer a SINGLE byte from the THR to RAM buffer. Figure 5 shows the flowchart for THRIRQ interrupt service routine. This service routine will only transfer one byte of data out of the THR into the RAM buffer. Note that it is very similar to the FIFOTHR interrupt service routine except that there is no facility to continue data transfer until THR FIFO is empty.

The code has both FIFOTHR and THRIRQ interrupt service routines. If you plan to utilize a THR interrupt trigger level other than 1, FIFOTHR is required for the THR interrupt. If you set the THR interrupt trigger level to 1 byte, THRIRQ of FIFOTHR can be used. In the ECHO182 example, FIFOTHR is used for THR interrupts and THRIRQ is used for THR timeout interrupts (although, THR timeout interrupt is disabled, THRIRQ was left in for example purposes).

**THRIRQ Transfers One Byte Of Data From
The THR Register To The RAM Buffer**

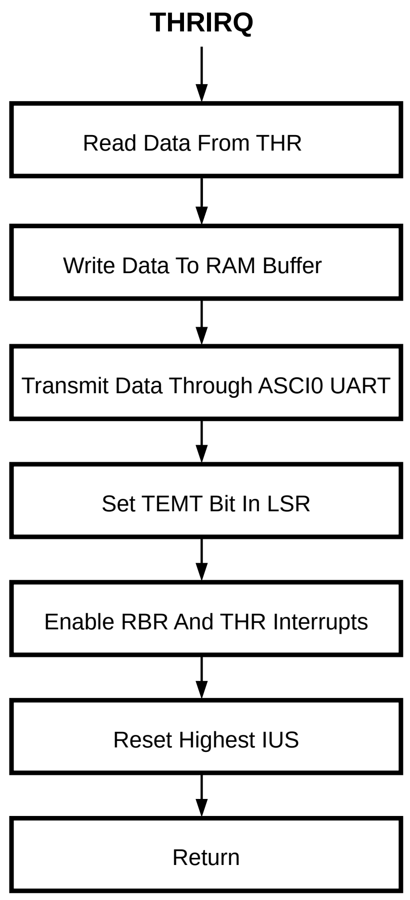

Figure 5. THRIRQ Interrupt Service Flowchart

RBRIRQ - Receive Buffer Register Interrupt
The RBRIRQ interrupt service routine functions to transfer data from the RAM buffer to the RBR register of the Mimic. The RBR interrupt asks the Static Z180 core to get data from the external world so that it can give that data to the PC. Whenever the Mimic RBR FIFO is empty, it will generate an interrupt when enabled. As a caution, one should be careful to choose the right times to enable this interrupt otherwise it will cause interrupts indefinitely when there is nothing in the RBR FIFO. In the ECHO182, the RBR interrupt is enabled when data is transmitted by the PC to the Mimic. The RBR interrupt is disabled when contents of RAM buffer have been completely transferred to the RBR FIFO.

A flowchart of the RBRIRQ interrupt service routine is shown in Figure 6. The first step of the Interrupt service routine is to evaluate if there is any data in the RAM buffer to echo back to the PC. This is done by simple checking of two RAM buffer pointers.

If there is data in the RAM buffer, the Static Z180 will transfer one byte from the RAM buffer to the RBR FIFO of the Mimic. Depending on the RBR interrupt level selected by the PC communication software, an interrupt to the PC will be generated such that the PC can read the data in the RBR. As a final step, ***Reset the Highest IUS*** of the Mimic prior to returning from the interrupt.

If there is no data in the RAM buffer, the RBR interrupts are disabled. Since there is no more data to transfer in the RAM buffer, the RBR interrupts should be stopped to prevent any more data being transferred from the RAM buffer to the RBR register of the Mimic. Note that if the RBR interrupts were not disabled, the RBR interrupts will occur indefinitely until data is put in the RBR. Remember to ***Reset the Highest IUS*** before returning from the interrupt.

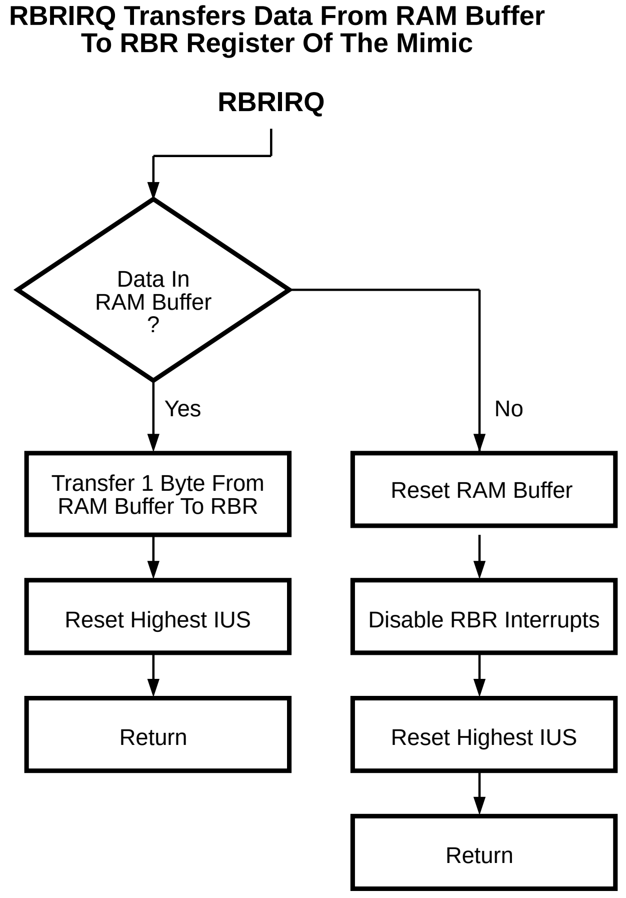

Figure 6. RBRIRQ Interrupt Service Flowchart

PRECAUTIONS / CONSIDERATIONS

- Make sure that the Mimic is multiplexed properly in the System Configuration Register (address xxEFH).
- Mimic Emulation timers should be utilized to prevent PC software “locking up." The emulation timers slow down data transfer as if the Mimic were an actual 16550 (time required to do serial shifting of data).
- Mimic Timers require that the ESCC Ch. B baud rate generator output is set for /TRxCB pin.
- Disable Mimic timers prior to modifying the timer constants.
- Reset the Mimic’s Highest Interrupt Under Service during initialization, at the end of every Mimic interrupt service routine, and in the main program loop.
- TEMT bit of the 16550 is set when transmit FIFO and transmit shift register is empty. There is no transmit shift register on the Mimic, so the Static Z180 must intelligently decide when to set this bit. This bit is automatically reset when data is written to the Transmit FIFO. PC communication software polls this bit after transmitting a block of data. The software will halt transmission if the TEMT bit is not set by the Static Z180.
- Intelligently select when RBR interrupts should be enabled, otherwise RBR interrupts will occur indefinitely (since the RBR FIFO is always empty until something is received).
- Mimic interrupt vectors may not be modified per specific register interrupts (FCR, LCR, MCR, DLC, DLM). Note that the RBR, TTO, and THR interrupt vectors are reliable. Unmodified vectors will interrupt with NOIRQ condition vector. Place a service routine to the handler of the NOIRQ vector that will check FCR, LCR, MCR, DLC, and DLM registers and modify the ESCC data link according to the changes in these registers. Don’t forget the Reset Highest IUS command at the very end of the interrupt service routine.
- Do not attempt to use the Z182’s internal DMAs for data transfer to/from the Mimic.
- Some trial and error may be required to find optimum values for Mimic serial emulation timers. If the timer constants are too low, the PC hardware/software may not be able to handle the interrupt cycle time. If the timer constant is too high, data transfer efficiency may suffer. If the RBR serial timer constant is set too high, the PC may be flagged with an error interrupt without any data in the RBR FIFO.

CONCLUSION

The Z182's Mimic core is not very difficult to program. This Application Note (ECHO182 source code is found in Appendix A) is meant to be a good example of how the Mimic can be programmed to work in conjunction with PC software programs for communication. The hardware design of the Z8018200ZCO evaluation board serves as a good hardware example of how the Z182 interfaces with the PC bus. The schematic of the Z8018200ZCO evaluation board is shown in Figures 7a, 7b, and 7c. With this Application Note, a developer can plug-and-play with the Z182 to gain familiarization with the Mimic. The ECHO182 can also be modified and optimized to fit specific applications.

EVALUATION BOARD SCHEMATICS

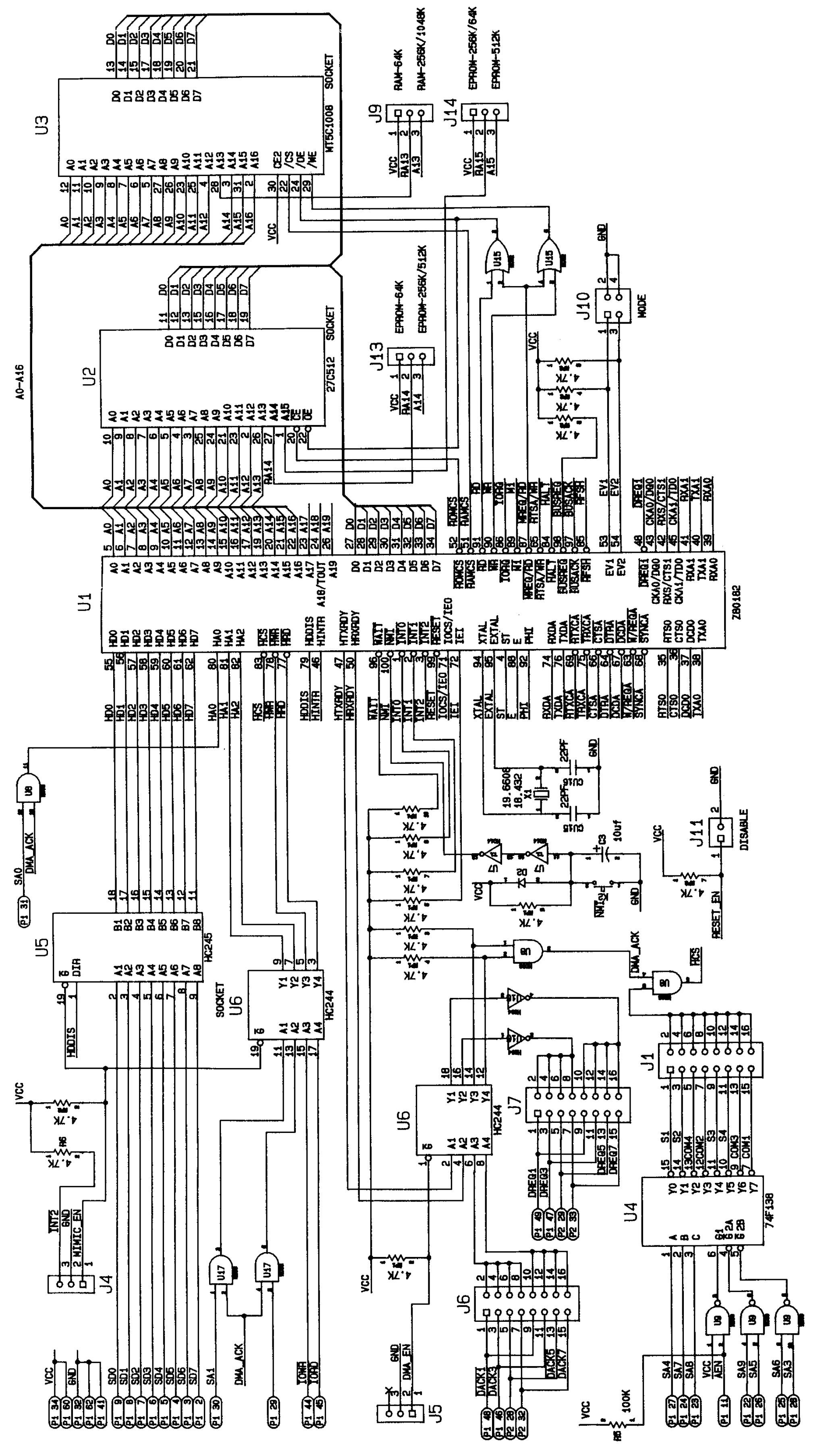

Figure 7a. Z80182 Evaluation Board

EVALUATION BOARD SCHEMATICS (Continued)

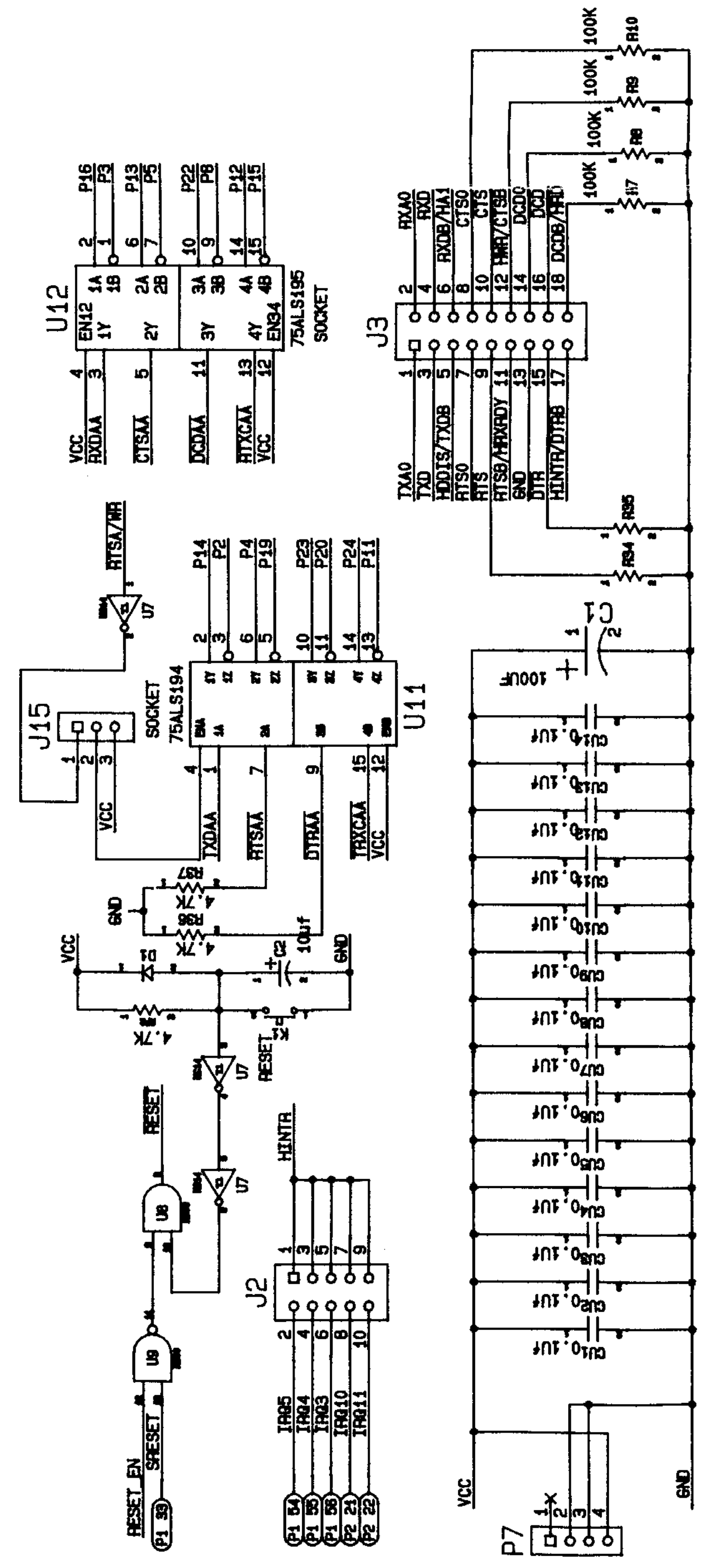

Figure 7b. Z80182 Evaluation Board

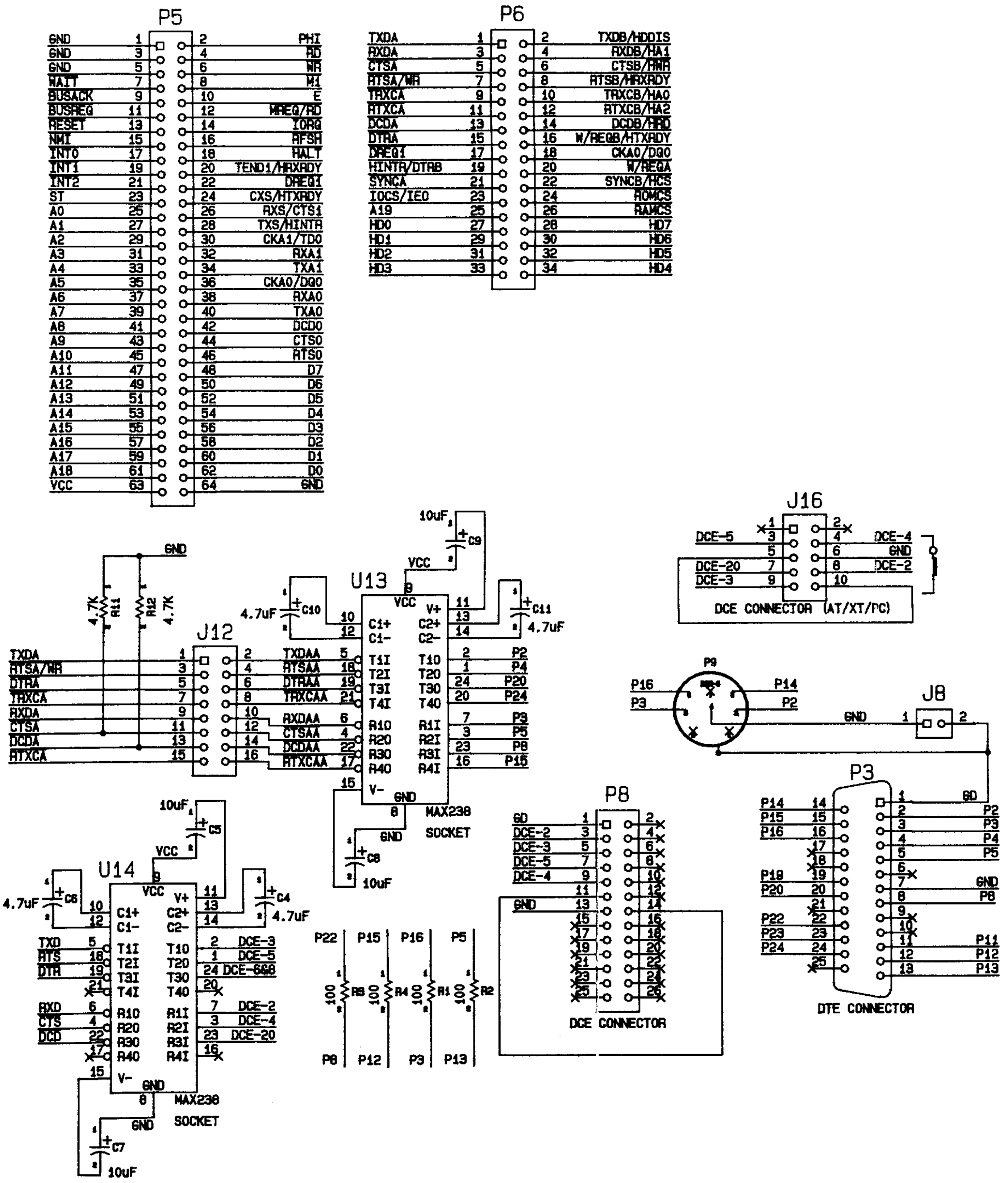

Figure 7c. Z80182 Evaluation Board

APPENDIX A
ECHO182.S Source Code

ECHO182.S™ written by Del Miranda, Zilog, Inc.
This program uses the Z182 Mimic Cell to interface to the PC. Whatever data is transmitted to com port is ECHOed back to PC.

(An assembler source and hex file of ECHO182 can be found in the Z80 Support Directory of Zilog's Bulletin Board Service (ZBBS (408) 370-8024 8-n-1.)

```
;******************************************************************************************************************************
*include 182macro.lib

ascii_lf:       equ     00ah
ascii_cr:       equ     00dh
null:           equ     00h

        org 00000h
        jp 1000h
;------------------------------------------------------------- Z182 Initialization -------------------------------------------------------------
        org 1000h
        di
        ld hl,0d400h            ;set up stack
        ld sp,hl
        im 2
        ld a,07h                ;setup int vector location
        ld i,a                  ;at 07xxh

        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                  ;standard /2 clock
        db 00h
        db itc                  ;disable interrupts first
        db 00h
        db icr                  ;standard IO mapping
        db 00h
        db dcntl                ;dma control - unnecessary
        db 0f0h
        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 61h                  ;given a 18.432 MHz XTAL in /2 mode
        db cntlb0
        db 21h

        db stat0                ;disable asci interrupts
        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 **

```
        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 0deh
        db sccbcnt              ;timer high
        db 0dh
        db sccbcnt
        db 01h
        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
```

APPENDIX A (Continued)
ECHO182.S Source Code

```
        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 -----------------------------------------------
initend:
        ld hl,prompt_msg        ;print my message to ASCI0
        call message
;------------------------------------------------------ Mimic Initialization ------------------------------------------------------
        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 and THR FIFO timeouts
        out0 (0ebh),a

        ld a,0ah                ;setup RBR and THR serial emulation timers
        out0 (0fah),a
        out0 (0fbh),a

        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 communication software. For use in modems, a THR interrupt trigger level of 1 is suggested.

```
;
;**********************************************************************************************************************************

        ld hl, 0d800h           ;setup buffer pointers
        ld de, 0d800h

        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:   nop                     ;constant looping, program root
        ei
        jp loop
;-------------------------------------------- Subroutines to Display My Message to ASCI --------------------------------------------

send_char:                      ;sends message out to ASCI0
                                ;only used to output my start message
        call out_char           ;has nothing to do with Mimic
        inc hl
message:
        ld a,(hl)
        cp a,null
        jr nz,send_char
        ret

out_char:

        push af
txlop:  in0 a,(stat0)
        bit 1,a
        jr z,txlop
        nop
        nop
        pop af
        out0 (tdr0),a
        ret

prompt_msg:

        .ascii  "   ECHO182 version 3.0", ascii_cr,ascii_lf
        .ascii  "   Auto Echo for Z182-18.432 MHz XTAL",ascii_cr,ascii_lf
        .ascii  "   9600 baud monitor on ASCI0",ascii_cr,ascii_lf
        .ascii  "   by Del Miranda - Zilog Euro Marketing",ascii_cr,ascii_lf
        .ascii  ascii_cr,ascii_lf,ascii_cr,ascii_lf,null

;---------------------------------------------------- End of Sendchar Subroutine ----------------------------------------------------

;***************************************************************************************************************************
;                                   INTERRUPT SERVICE ROUTINE - RBRIRQ
;***************************************************************************************************************************

rbrirq:
                                ;INT ROUTINE FOR RBR INTERRUPTS
                                ;OCCURS WHEN RBR IS EMPTY
        ld a,l                  ;compare buffer pointer
        cp e                    ; if hl=de, then get out
        jr z,out
```

APPENDIX A (Continued)
ECHO182.S Source Code

```
okay:   inc de
        ld a,(de)
        out0 (rbr),a            ; else, output data to RBR
        ld a,80h                ; reset highest IUS
        out0 (iusip),a

        ret
out:                            ;return to loop
        ld a,h
        cp d
        jr nz,okay

        ld hl,0d800h            ;reset buffer pointers
        ld de,0d800h

        ld a,0c0h               ;disable RBR interrupts
        out0 (mimie),a          ;otherwise RBR will always interrupt
        ld a,80h                ;reset highest IUS
        out0 (iusip),a
        ret                     ;return to loop

;**************************************************************************************************************
;                               INTERRUPT SERVICE ROUTINE - THRIRQ
;**************************************************************************************************************

                                ;INT ROUTINE FOR THR INTERRUPTS
                                ;OCCURS WHEN PC WRITES TO THR
                                ;AND TIMEOUT OCCURS - ONLY 1 BYTE IS READ
thrirq:
        inc hl
        in0 a,(thr)             ;increment pointer
        ld (hl),a               ;store THR date in buffer
        out0 (tdr0),a           ; output to asci


        ld a,40h
        out0 (lsr),a            ;set TEMT bit when transmit data is
notemt: ld a,0d0h               ;shifted out , we force it here
        out0 (mimie),a          ;enable RBR,THR ints

        ld a,80h
        out0 (iusip),a          ;reset Mimic highest int under service

        ret                     ;return to loop

;**************************************************************************************************************
;                               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
        out0 (tdr0),a           ; output to asci

        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,0d0h               ;enable RBR, THR ints
        out0 (mimie),a
        ld a,80h                ;reset highest Mimic int under service
        out0 (iusip),a
        ret                     ;return to loop

;***********************************************************************************************
;                        UNEXPLAINED INTERRUPT HANDLERS WORKAROUND
;***********************************************************************************************

uknirq:
        ld a,80h                ;workaround - dummy service routine
        out0 (iusip),a
        ret

;********************************* INTERRUPT VECTOR TABLE ************************************
        org 0700h
        dw uknirq               ;workaround, for NOINT vector
        dw uknirq
        dw uknirq
        dw uknirq
        dw uknirq
        dw rbrirq               ;table entry for rbr empty interrupt
        dw thrirq               ;table entry for timeout - disabled
        dw fifothr              ;table entry for thr has data interrupt

************************************************************************************************
.xlist
;***********************************************************************************************
;*      File name - 182macro.lib
;*      Macro library 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
```

APPENDIX A (Continued)
ECHO182.S Source Code

```
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
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
endm

in0             macro   ?r, ?p
db              11101101B
db              00000000B+(?&?r AND 7) SHL 3
db              ?p
endm
```

APPENDIX A (Continued)
ECHO182.S Source Code

```
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
```

ECHO182.S™ is a trademark of Zilog, Inc.

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