

MARCH 1994

# Z180 Family

## QUESTIONS AND ANSWERS

*This application note contains the most commonly asked questions about the Z180. Obviously, not every possible question on the Z180 is answered. However, this application note should give you a good feel for how the Z180 works. Along with the technical manual and product specification, it should facilitate your Z180 design.*

### FEATURES

**Q: What are the differences between the Z180, Hitachi's HD64180R0/R1 and Z versions?**

A: Our Z180 is identical to Hitachi's HD64180Z version except some of the signal names are different in order to match Z80 signal names (Z180 was jointly developed with Hitachi).

The HD64180R0 version is the original version, and has some bugs in it. The R1 version is the version which corrected the bugs in the R0 version. The 68-pin PLCC and 80-pin QFP versions have 1M bytes of physical memory address space. Also, There is a "test" pin (output; not open to user) assigned to the pin which is not used on the R0 version.

The Z version corrected the R0 and R1 problems (the problems involve the Z80 peripheral interface.)

**Q: In Z mode of operation (M1E of OMCR cleared to 0), there is no interrupt from the Z80 PIO after enabling the Z80's PIO interrupt. Why?**

A: Write zero to the /M1TE bit of OMCR after enabling the Z80's PIO interrupt. Because the Z80's PIO interrupt control logic requires /M1 to activate its interrupt logic while /M1 only occurs during Interrupt Acknowledge and RETI cycles with /M1E cleared.

### CPU CLOCK

**Q: Can I stop the clock in order to minimize the power consumption?**

A: No. However, one possible way is save the registers into battery backed up RAM, and then remove power from the CPU.

**Q: What is the relationship between EXTAL and PHI clock output when an external clock is input through EXTAL?**

A: PHI output changes its status on the falling edge of EXTAL input. And the delay from the falling edge of EXTAL to PHI is about 30 ns (Reference only; not guaranteed value)

**Q: In our system, sometimes the PHI frequency is the same as XTAL frequency, not XTAL divided by two. Why?**

A: Please check the following points:

- Reset is held low at least 6 clock cycles.
- The status of ST line during reset. ST should not be tied low (ST line is the OUTPUT signal!)

### INSTRUCTIONS

**Q: Which instruction, RET or RETI, is used at the end of an interrupt service routine?**

A: If you don't have any Z80 peripherals (do not use "interrupt daisy chain"), then you can use either of these instructions at the end of interrupt service routine (for interrupt from on-chip peripherals). If you have Z80 peripherals on board, then you should use RETI for the interrupts of Z80 peripherals, and RET for the interrupts of on-chip peripherals.

The reason, from CPU stand point, is both instructions are the same (POP PC value from stack and return). But Z80 peripherals are looking for RETI sequence on the bus to correct its interrupt daisy chain status upon receiving that sequence. If you have Z80 peripherals and are using RETI for the interrupt for on-chip peripherals, Z80 peripherals are confused and thus set the wrong daisy chain status.

**Q: Is the instruction set of the Z180 fully identical to the Z80 CPU's except for new instructions?**

A: There are three instructions which are not the same. They are: DAA and RRD/RLD.

**INSTRUCTIONS** (Continued)

For DAA (Decimal adjust), if you execute this instruction after DEC instruction (especially DEC instruction on 00h, then execute DAA), Z180 results in F9H while Z80 results in 99H. It is because the Z80 CPU refers “Internal Carry flag” while the Z180 doesn’t.

For RLD/RRD (Rotate Left/Right Digit), Z180’s flag will reflect the contents of the memory location pointed by HL register, while Z80 reflects the contents of the Accumulator.

But, there are very few applications which use DAA instructions after DEC and use flag after RLD/RRD instructions.

**REFRESH**

**Q: Is the functionality of R register the same as Z80 CPU’s?**

A: No. Z180’s R register is counting M1 cycles, and there is no relationship with current refresh address.

**Q: Is the refresh mechanism of the Z180 different from the Z80 CPU’s?**

A: Yes. Z180’s refresh mechanism is “periodic refresh” and the refresh period can be programmed and disabled. Refresh address is 8-bit. On the Z80 CPU, refresh cycle is inserted after every M1 cycle and can not be disabled.

**Q: Can the refresh cycle occur during an on-chip DMA cycle?**

A: Yes. Because Z180’s refresh mechanism can not distinguish whether current activity is by CPU or on-chip DMA. Refresh cycle will be inserted after the end of DMA or CPU machine cycle.

**I/O ADDRESSING SPACE**

**Q: Z180’s technical manual states that I/O addressing space is 64K Bytes while Z80 CPU’s I/O address space is only 256 Bytes. How do you access “expanded I/O address space”?**

A: In fact, Z180’s I/O addressing space is the same as Z80 CPU’s. You can specify lower half (A7-A0) of I/O address directly, as on Z80 CPU, but the upper half depends on the instruction which you are using for access. There are four groups of I/O instructions on Z180 (On Z80 CPU, group D does not exist)

A) 8080 type instruction  
IN A,(n) OUT (n),A  
A15-A8 ←Acc

B) C register indirect  
IN A,(C) OUT (C),A  
A15-A8 ←B register

C) C register indirect with auto increment/decrement  
IND INDR INI INIR OUTD OTDR OUTI OTIR  
A15-A8 ←B register (B register is loop counter)

D) Z180 original instructions which force A15-A8 to 0  
IN0 OUT0 OTIM OTIMR OTDM OTDMR TSTIO  
A15-A8 ←0

To utilize “64K Bytes” of I/O address, you can use group A) or B) instructions with care, or use group D) instructions to access the “Page zero” I/O address space.

**Q: To access on-chip peripherals (and system control registers), Should A15-A8 be zero?**

A: Yes. It is a good idea to use Z180’s new I/O instructions for that purpose (These instructions force A15-A8 to 0).

**Q: What happens if off-chip peripheral’s address is assigned to the internal I/O devices (overlapped)?**

A: I/O read: data from addressed internal peripheral is read, and the data on the bus at that time is just ignored. I/O write: output the data to the data bus as well as to on-chip peripherals. Also, this transaction could write the data to off-chip peripherals.

**BUS TIMING**

**Q: For the Interrupt Acknowledge cycle timing chart, there are no timing specifications for PHI to M1 and PHI to IORQ. Do you have these numbers?**

A: Yes. These parameters are as follows:

PHI rising edge to /M1 falling edge (Interrupt Acknowledge cycle): Same as parameter #10 (t<sub>M1D1</sub>)

PHI falling edge (of first T<sub>WA</sub> state) to /IORQ falling edge (Interrupt Acknowledge cycle): Same as parameter #28 (t<sub>IOD1</sub>, Case IOC=1).

PHI rising edge to /M1 rising edge (Interrupt Acknowledge cycle): Same as parameter #14 (t<sub>M1D2</sub>).

PHI falling edge to /IORQ rising edge (Interrupt Acknowledge cycle): Same as parameter #29 (t<sub>IOD2</sub>).

**Q: What about the bus status during the access to the on-chip peripherals?**

A: During I/O access to the on-chip peripherals,

I/O read: Data bus - Hi-Z  
Address bus: holds the I/O address

I/O write: Data bus - Data to be written  
Address bus: Holds the I/O address

**Q: During sleep or bus release mode, is it possible to extend the E signal pulse width by inserting wait states?**

A: No. During sleep or bus release mode, the CPU won't sample the status of /WAIT input (cannot extend the cycle).

**Q: What is the timing of E output during -DMA, -Refresh, -I/O cycle?**

A: For refresh cycle, E output will be held low. For DMA and I/O cycle, E timing is identical to the CPU cycle. Thus:

E is active during...

Memory R/W: T2 rising edge to T3 falling edge.

I/O read: 1st Tw rising edge to T3 falling edge.

I/O write: 1st Tw rising edge to T3 falling edge.

**Q: Does the Z180 sample the data at the different points during memory read and opcode fetch cycles, as the Z80 CPU does?**

A: Yes. The data is sampled on the rising edge of T3 during opcode fetch cycles, falling edge of T3 for memory read cycles.

**WAIT STATES****Q: When using automatic wait state generator, does the Z180 sample external /WAIT state before automatic wait state insertion, or after?**

A: External /WAIT status will be sampled after the automatic wait state insertion.

**Q: Is it possible to insert wait state(s) into a refresh cycle by /WAIT input?**

A: No. WAIT is not sampled during refresh cycle. However, Z180 has a capability to insert "software" wait state into refresh cycle by setting REFW (Bit D6) of RCR register to 1.

**Q: Is the automatic wait state during I/O cycle removable?**

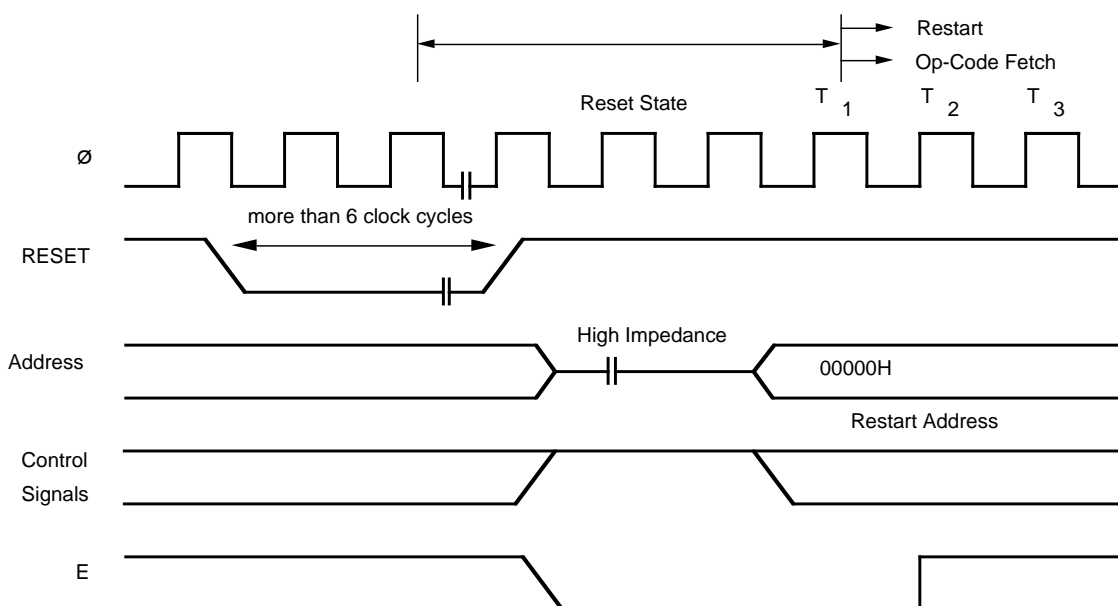
A: No.

**Q: When accessing on-chip peripherals, it seems that required access time varies case by case. Why?**

A: When accessing on-chip peripherals (ASCI, CSI/O, PRT data registers), zero to four wait states are automatically inserted depending on the status of CPU and peripherals. In those cases, the value set in DAM/WAIT control register is ignored.

**RESET****Q: How is the power-on reset sequence performed?**

A: The power-on reset sequence is as follows.



**Note:** /RESET pin should be asserted Low for at least 6 clock cycles to perform power-on reset correctly.

**POWER SAVE MODES**

**Q: How to exit from SYSTEM STOP mode, and what is the CPU's response after exiting this mode?**

A: /NMI, /INT (external) or RESET is necessary to exit from SYSTEM STOP mode. If receiving RESET, normal RESET sequence takes place. In case of /NMI, /NMI sequence takes place. In case of external /INT:

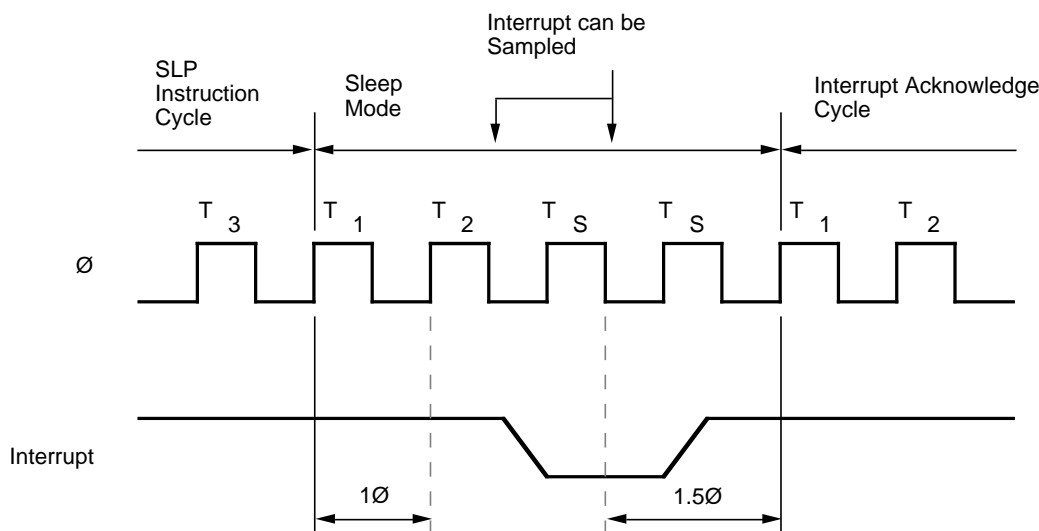
If interrupt is globally disabled (IEF1=0):  
CPU will execute the instruction following the SLEEP instruction.

If interrupt is globally enabled (IEF1=1):  
Appropriate normal interrupt response sequence will be performed.

Except for RESET, I/O STOP mode is maintained until the I/O stop bit is cleared to 0 after exiting from SYSTEM STOP MODE.

**Q: During SLEEP mode, when is the CPU sampling the status of INT line?**

A: The CPU starts to sample the status of INT line on the falling edge of the 1.5 clock cycles after entering the SLEEP mode. When CPU samples INT as active, 1.5 clock cycles later CPU will wake up from SLEEP mode.

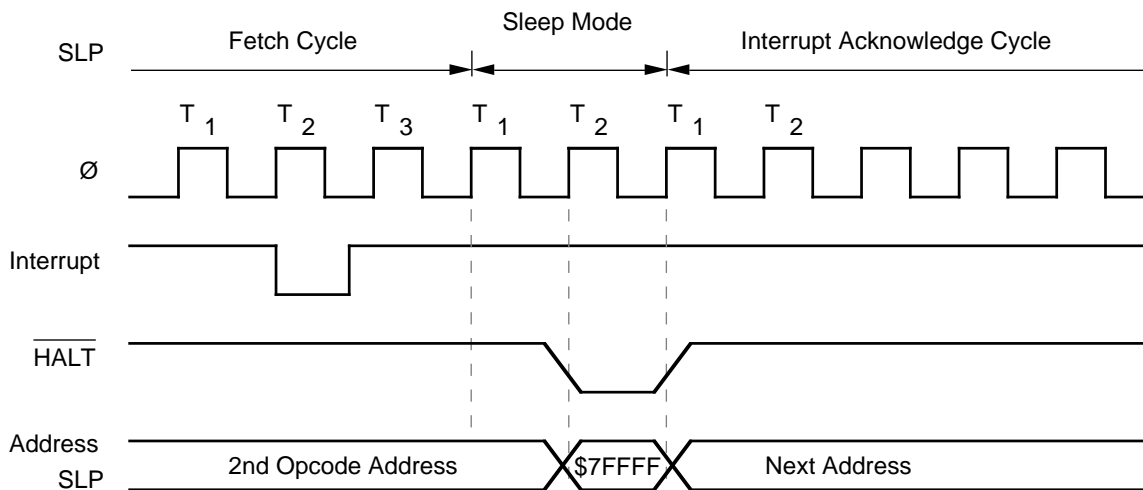


**Q: What is the status of the bus during SLEEP mode?**

A: A0-A19 are all one, /HALT stays low, and /MREQ, /M1, /RD stay high.

A: If an interrupt is received during a SLP instruction, /HALT is asserted low for one clock cycle and the address bus is set to all ones, and is then followed by an interrupt acknowledge cycle, as shown below.

**Q: What happens if an interrupt is received during the execution of the Sleep (SLP) instruction?**



**TRAP**

**Q: What happens if another trap condition (detecting undefined opcode) occurs before clearing the TRAP bit of the ITC register?**

A: You will have another TRAP. If the trap handling routine causes a TRAP it will probably get into an infinity loop, and crash. For this case, the TRAP bit remains 1 and the UFO bit shows the status for the previous TRAP, since the status of the UFO bit cannot be changed while TRAP=1.

**Q: What is the purpose of the UFO bit?**

A: UFO bit shows whether TRAP occurred during 2nd opcode fetch cycle or 3rd. If it is zero, TRAP has occurred in the 2nd opcode fetch cycle and the PC value pushed onto the stack is, 1st opcode address + 1. And if it is one, TRAP has occurred in the 3rd opcode fetch cycle and the PC value pushed onto the stack is, 1st opcode address + 2. All hex numbers for first opcode on Z180 are allocated for instructions, so TRAP won't happen on 1st opcode.

**Q: How do you determine the opcode address which caused TRAP using the UFO bit?**

A: Upon TRAP, the PC value pushed onto the stack for the address of the instruction which caused TRAP is:

Instruction start address =  
(value on the stack) - 1 when UFO=0  
Instruction start address =  
(value on the stack) - 2 when UFO=1

So that you can determine the starting address for the instruction which caused the TRAP.

**NMI AND INT**

**Q: Are the addresses of interrupt vectors treated as physical addresses or logical addresses?**

A: Z180 always treats those addresses as Logical addresses. So if you enabled the MMU, care must be taken.

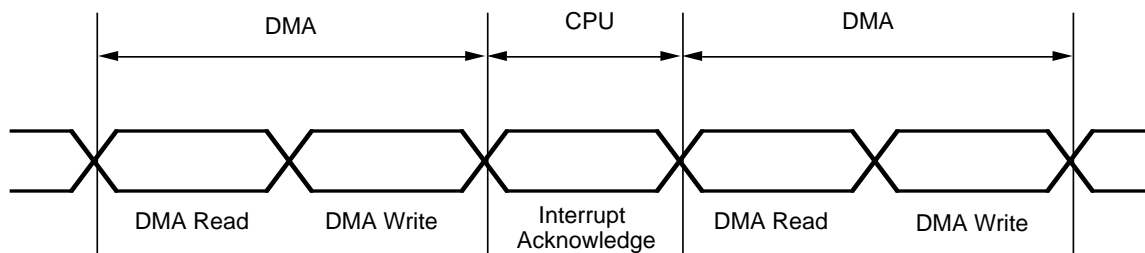
Here is one suggestion: If you can make the vector table in Common Area 0, the physical address is the same as the Logical address, which helps.

**Q: What happens if an interrupt is received during an on-chip DMA operation?**

A: If it is an /NMI, DMA operations will stop and an /NMI acknowledge cycle will be initiated. If it is /INT or interrupt from on-chip peripherals:

During Mem-Mem burst mode transfer, the interrupt is just ignored.

During Mem-Mem cycle steal mode of operation, interrupt acknowledge cycle is initiated between DMA cycles (See following figure).



**Q: For interrupt mode 2, shall I set bit D0 of interrupt vector to 0, as Z80 requires?**

A: Yes. However, the Z180 works correctly even if bit D0 of the interrupt vector is 1.

**Q: Is /NMI acknowledged during an interrupt acknowledge cycle?**

A: Yes. One instruction (except EI and DI instruction) is executed after /INT acknowledge cycle, then the /NMI acknowledge cycle starts. It is also the same for /NMI received during /NMI acknowledge cycle.

**Q: Are /NMI or /INT acknowledged immediately after RESET?**

A: No. /NMI and /INT are disabled for three clock cycles immediately after RESET (power on reset). After those three cycles, the instruction is executed (from 0000h) and then interrupt is enabled. Note that /NMI status is latched internally right after the power on reset cycle.

**Q: Is an interrupt (/NMI or /INT) acknowledged during a refresh cycle?**

A: /NMI status will be latched internally if /NMI occurs during the refresh cycle. When current instruction execution is completed, the following cycle will be an /NMI acknowledge cycle. /INT status is not sampled during a refresh cycle, and it will be sampled during the following instruction execution cycle.

**NMI AND INT** (Continued)

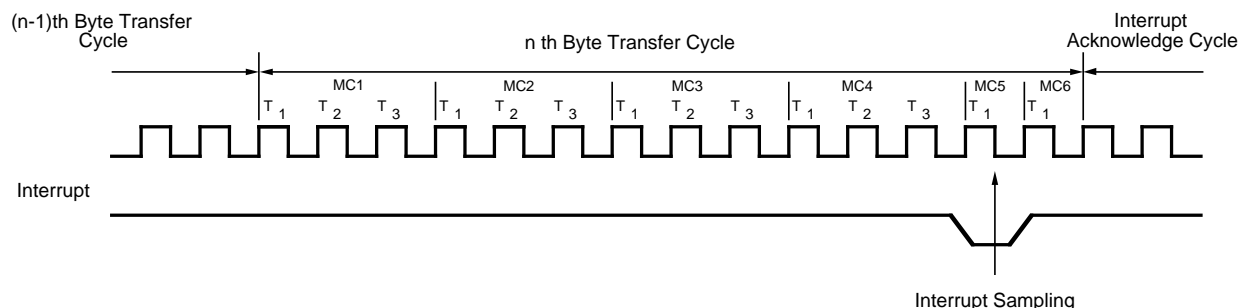
**Q: I have an /NMI input with a pulse width of a couple of hundred ns. Can it be accepted?**

A: The /NMI input will be accepted as long as you can satisfy the /NMI pulse width specification (120 ns min.), since /NMI is edge triggered input and its status is latched internally.

**Q: During the execution of EI instruction, can an interrupt be acknowledged?**

A: No. During execution of an EI instruction, interrupt is not sampled. So if interrupt (Maskable interrupt) goes active during the instruction just before an EI instruction, or during EI instruction, it will be acknowledged during the following instruction.

**Q: When does the Z180 sample an interrupt status?**



**Q: During interrupt Mode 0, if a call instruction is put onto the bus during the interrupt acknowledge cycle, software won't return the correct address. Why?**

A: The PC value pushed onto the stack during the interrupt acknowledge cycle is the PC value for the next instruction if the instruction put onto the bus is the RST (restart) instruction (one byte instruction).

However, call instruction is a three byte instruction and the return address pushed onto the stack by that Call instruction will be the PC value plus two (PC value stays the same during interrupt acknowledge cycle, and incremented by two during operand fetch for jump address). So at the end of the interrupt service routine, you need to decrement the return address pushed onto stack by two.

**Q: What happens if there is /NMI going active and at the same time (or just before) /DREQn is also going active when using DMA with /DREQn input?**

A: The input on /DREQn is ignored. To restart DMA, please refer to previous Q&A.

**Q: When interrupts from on-chip peripherals are sampled, is it the same as the external interrupts?**

A: Yes. Interrupt status is sampled on the falling edge of T2 or T1 cycle of the last machine cycle of the instruction.

A: The Z180 samples an interrupt status at the falling edge of PHI one cycle prior to the last machine cycle of each instruction (excluding EI instruction). Also, the status of the internal /NMI latch is sampled at the same time.

**Q: During execution of block transfer instructions, doesn't the Z180 sample an interrupt status until the end of block transfer?**

A: No. The Z180 checks the status of interrupt at the end of every sequence, as shown below. Note that if you accept interrupt during block transfer, your interrupt service routine will not destroy the contents of registers in order to resume from the interrupt correctly (save the registers using EX/EXX instructions at the beginning of interrupt service routine).

**Q: Is there any way to connect 8259 (interrupt controller) under mode 0 interrupt?**

A: When using Mode 0 interrupt, which is "8080 compatible interrupt mode", there is one important point, which is "concern about the INTA pulse".

Mode 0 is the mode which maintains the "software compatibility" with the 8080. That means in this mode, during the INTACK cycle, the Z180 fetches the data on the bus as an "instruction" and executes it, like the 8080.

However, from the hardware stand point, it is not true. Because the 8080 generates three INTA pulses during the interrupt acknowledge cycle, while the Z180 generates only one INTACK signal (which can be decoded from /M1 and /RD).

This system works fine if you are not using the 8259 and put "RST" (restart) instruction onto the bus during the Interrupt acknowledge cycle which is a one byte instruction. However, if you want to use the 8259 with the Z180, you'll have a problem, which is:

The 8259 requires three INTA pulses but the Z180 generates only one INTACK cycle.



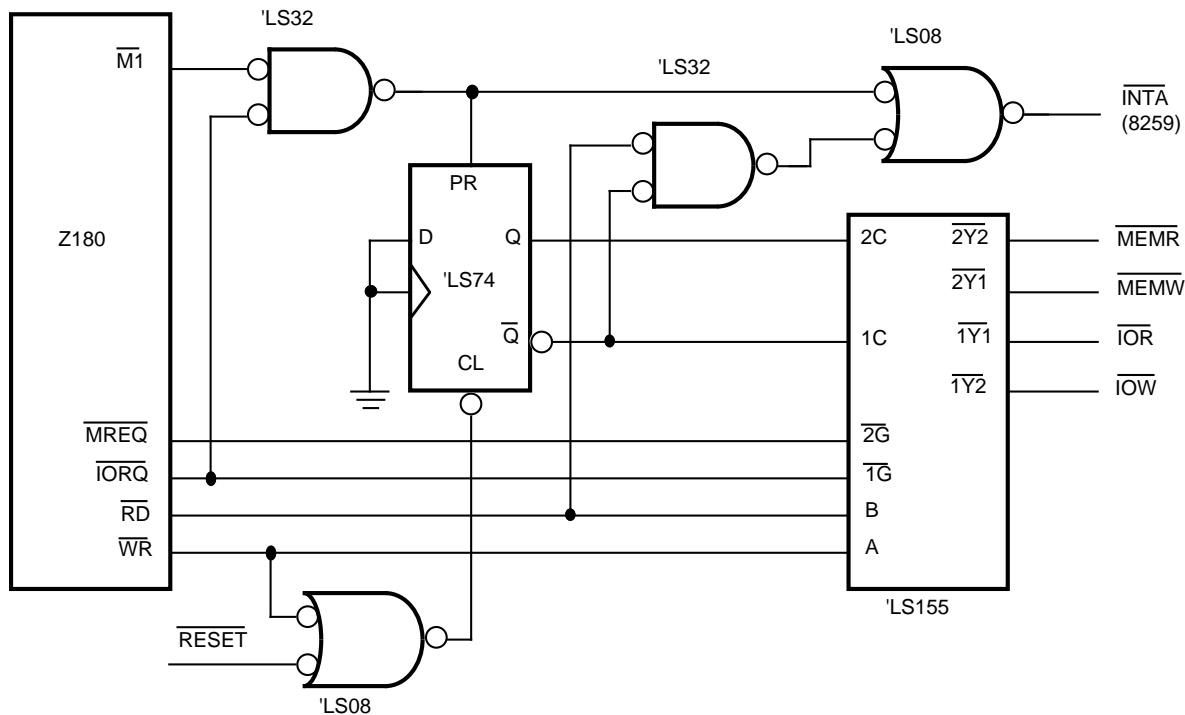
The best way to solve the problem is to “simulate an 8080 interrupt acknowledge cycle” - generating a total of three “INTA” pulses for 8259 upon the Z180’s interrupt acknowledge cycle by external logic.

The following figure is one example of the implementation. This circuit works as follows (Assume that the instruction sent by the 8259 was a “CALL” instruction). On interrupt acknowledge cycle, a decoded INTA signal is sent out as an INTA pulse for the 8259 and at the same time, sets the LS74 to indicate that the interrupt acknowledge cycle is started.

On the following memory read cycle of the jump address for the call instruction, this circuit generates two additional INTA pulses for the 8259 and also masks off the read signal for the memory to avoid bus contention problems.

On the following write cycle, /WR signal resets the LS74 to indicate that the interrupt acknowledge cycle is completed.

By using this circuit, you can use the 8259 with the Z180.



## MMU

**Q: When does the effect take place after changing the contents of MMU related registers?**

A: From the instruction following the I/O write instruction to the register.

**Q: What happens if the MMU base register is programmed to exceed 512K (64-pin DIP) or 1M Bytes (68-pin PLCC/80-pin QFP) of physical addressing space?**

A: 64-pin DIP version -

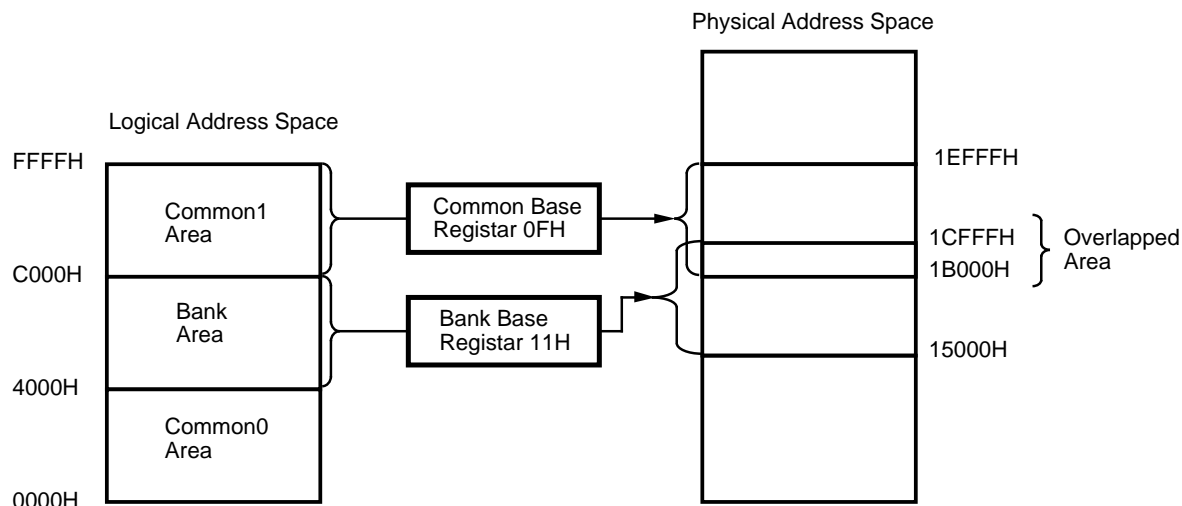
When “calculated address” is above 7FFFFH, carry bit and MSB are just ignored (A19 and carry from A19). That means the address wraps around to 00000H.

68-pin PLCC/80-pin QFP version -

When “calculated address” is above FFFFFH, carry bit is just ignored (carry from A19). That means the address wraps around to 00000H.

**Q: Can I have a Common Area 1 and Bank Area (or, Common Area 0 and Bank Area) overlapped by programming associated MMU registers?**

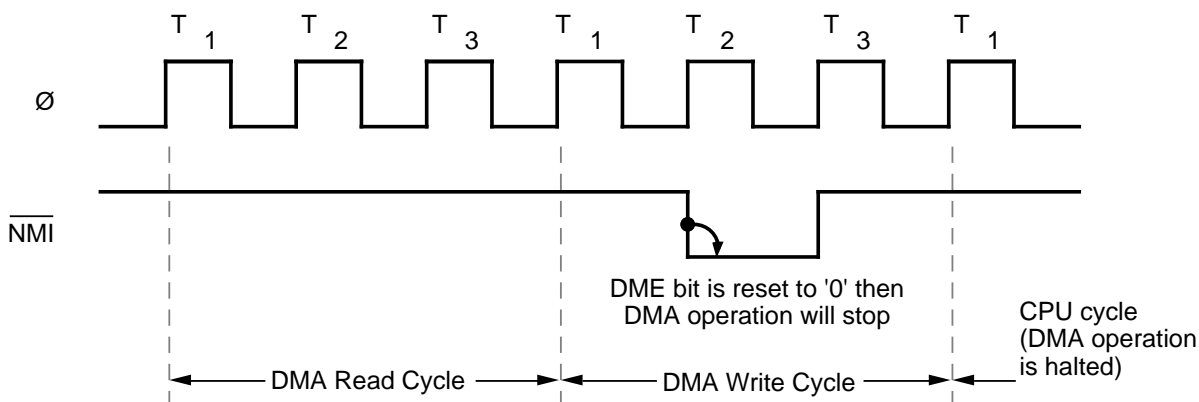
A: Yes. You can have overlapped areas by programming MMU Common Base/Bank Base registers (See figure on next page).

**MMU (Continued)****DMA CHANNELS**

**Q: What happens if an NMI occurs during a DMA cycle?**

A: When the DMA cycle is completed, the CPU gains

control. The DME (DMA enable) bit of that DMA channel is reset to Zero.



To restart DMA after the /NMI processing, DE (DE0 or DE1) bit should be set to 1 along with DME bit to 1.

Here is an example to restart DMA0.

```
LD    A, 01100011B ;DE0=1 with DME=1, /DWE=0
OUT0  (30H),A      ;Write out to DSTAT reg.
```

**Q: What is the purpose of the /DWE bit in the DSTAT register?**

A: To set/reset the DE0 or DE1 bit in DSTAT register, you also have to set the corresponding /DWE0 or /DWE1 bit

to 1. This scheme allows changing the status of the DE0 or DE1 bit without affecting the operation of the other DMA channel.

**Q: When DMA0 is used for data transfer from/to on-chip ASCI, does it accept the request from DREQ0 input?**

A: When DMA0 is used for data transfer for an internal peripheral (ASCI Tx or Rx data), external input on /DREQ0 is ignored. When using /DREQ0 input for DMA request from an off-chip peripheral, both bits of SAR17/16 or DAR17/16 should be cleared to zero.



**Q: When using DMA0 for ASCII data transfer, what value shall I program into SAR0 (DMA0 Source Address Register) or DAR0 (DMA0 Destination Address Register)?**

A: You have to program SAR0 or DAR0 as follows:

For Receive data transfer:

Set SAR0L (I/O address 20H) to:

08H for ASCII0 receive data  
09H for ASCII1 receive data

Set SAR0H (I/O address 21H) to 00H

Set SAR0B (I/O address 22H) to:

01H for ASCII0 receive data  
02H for ASCII1 receive data

For Transmit data transfer:

Set DAR0L (I/O address 23H) to:

06H for ASCII0 transmit data  
07H for ASCII1 transmit data

Set DAR0H (I/O address 24H) to 00H

Set DAR0B (I/O address 25H) to:

01H for ASCII0 transmit data  
02H for ASCII1 transmit data

**Q: For the DMA transfer from/to ASCII channels, does it require A15-A8 of SAR or DAR equal to 00H?**

A: Yes. Those bits have to be zero for ASCII data transfer. If the value is not zero, the access by DMA is not going to ASCII registers but will try to access off-chip I/O devices. The result is ASCII will continue to request DMA service (Request status would not be cleared).

**Q: I'm trying to transfer ASCII receive data using on-chip DMA. But after enabling the DMA channel, it won't transfer the data. Why?**

A: Please check whether ASCII has already received data when enabling the DMA channel. If ASCII already has

data (RDRF flag=1), DMA won't start the data transfer. For that case, before enabling DMA channel, read TSR register to reset the RDRF flag.

Remember that you cannot use "Level sense mode" for ASCII data transfer.

**Q: Can I transfer the memory data using DMA and using logical address?**

A: No, you can't. You have to know the physical address for the target memory as well as the Common0/Bank/Common1 address assignments. You may need to calculate the "physical address" using the value in MMU related registers - BBR, CBR and CBAR.

Another solution might be using Z180's "Block transfer instructions". For this case, you don't have to know the physical address for the target memory.

**Q: I want to transfer the data from a memory mapped I/O device (Fixed address) to one of the peripherals assigned to an I/O address. Can I do that?**

A: No. The following source/destination combinations are NOT allowed.

Memory (Fixed address) to Memory  
(Fixed address).

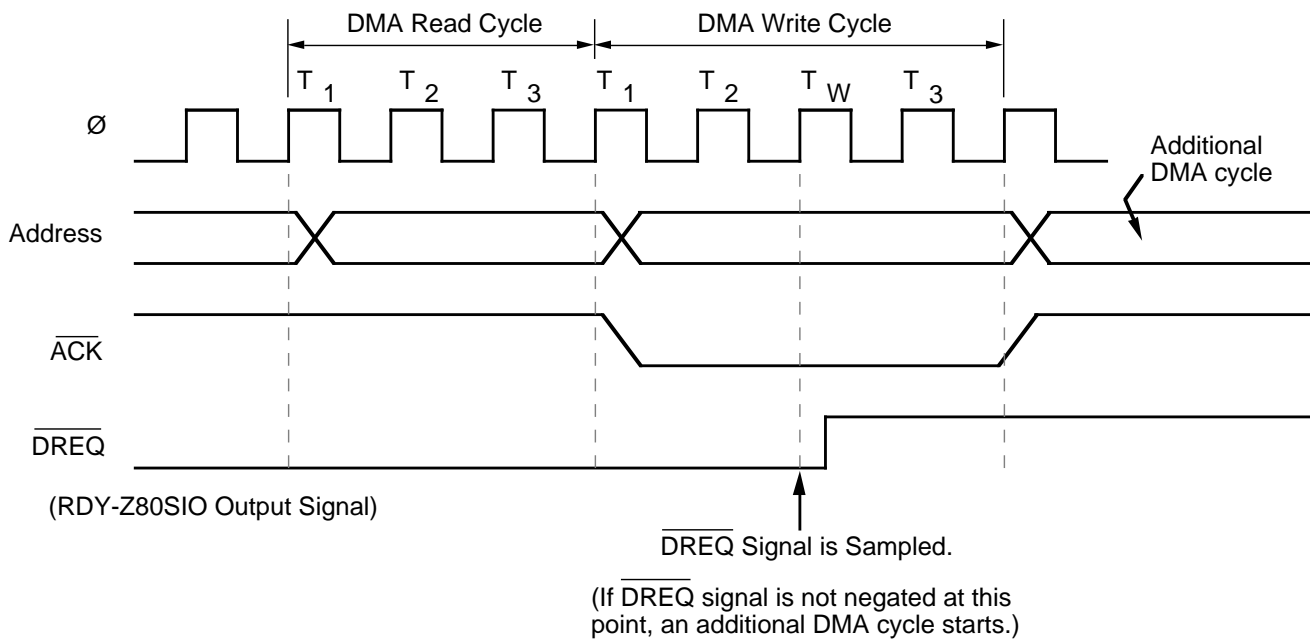
Memory (Fixed address) to I/O (Fixed address).

I/O(Fixed address) to I/O(Fixed address).

With the memory mapped I/O device (Fixed address), you can transfer data from/to Memory (variable address).

**Q: When using on-chip DMA for Z80 SIO Tx data transfers with level sense mode, sometimes Tx data is missed (It has "extra DMA cycle"). Why, and are there any workarounds for this situation?**

A: The reason for "extra DMA cycles" is the timing when DMA samples the DREQ status and the timing when SIO negates the DREQ signal. (See next page)



Here are possible workarounds.

- 1) Program DMA to edge sense mode.
- 2) Put in one wait state for the I/O write cycle.
- 3)  $\overline{\text{DREQ}}$  signal masked by the Chip enable signal of the Z80 SIO.

**Q: Which DMA channel has higher priority?**

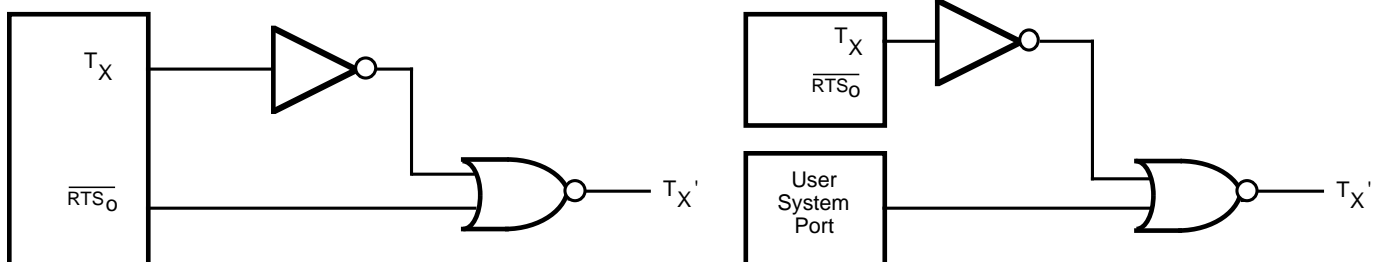
A: DMA Channel 0 has higher priority.

## ASCII CHANNELS

**Q: Are there any ways to send BREAK via software command?**

A: No. ASCII doesn't have any software commands to send

a break. One possible way to send a break is to utilize the unused modem control signal of the Z180, or using an output port bit and connect as follows:



**Q: How to calculate baud rate?**

A: Use the following formula to calculate.

$$\text{Baud rate} = \frac{\text{System Clock Speed}}{(\text{Clock factor}) \times (\text{PS bit}) \times (\text{Divide ratio})}$$

Where :

- Clock factor - 16 or 64
- PS bit - 10 or 30
- SS0-SS2 - 1, 2, 4, 8, 16, 32 or 64

**Q: I could not send data since TDRE always reads zero. Why?**

A: Please check the  $\overline{\text{CTS}}$  line status. TDRE is qualified by the status of the  $\overline{\text{CTS}}$  line. If  $\overline{\text{CTS}}$  stays high, the TDRE bit is zero. Transmit interrupt is generated by the TDRE bit, so with  $\overline{\text{CTS}}$  high, you won't receive transmit interrupt if you enabled the interrupt.

**Q: What happens if /CTS goes inactive at the middle of the data transmission ?**

A: The character being sent out will be sent completely.

**Q: When reading the status of /DCD0 found in STAT0, the status is still one but the real status of the /DCD pin has already changed to zero. How and why is this?**

A: The /DCD0 bit in the status register is reflecting the status of the /DCD0 pin. The low to high change on this pin updates the status of the /DCD0 bit right away. But if the change on the /DCD0 pin is high to low then the bit will remain the same until this bit is read (to change its status). The reason for this scheme is to serve the interrupt for /DCD correctly. If it changes its status at both transitions, and if there is an interrupt from ASCI0 before the service for /DCD, the status of /DCD would be lost.

**Q: What is the maximum baud rate for ASCI channels?**

A: When using the internal clock for baud rate generation, it is:

$$BRG_{max} = PHI / (1 \times 10 \times 16)$$

Where: 1 is divider value, 10 is prescaler value, and 16 is sampling rate.

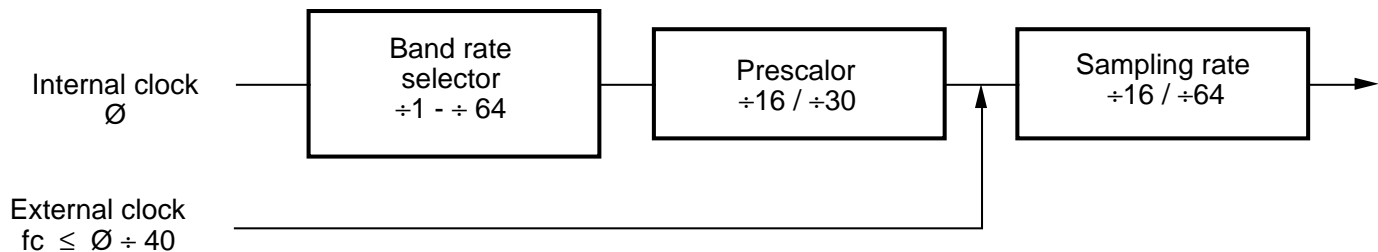
When using the external clock, clock frequency on the external clock input is limited up to  $PHI/40$ , so:

$$BRG_{max} = PHI / (40 \times 16)$$

Where: 16 is sampling rate.

**Q: When using the external clock for baud rate generation, can I use the on-chip clock divider/prescaler?**

A: No. When using external clock input, the on-chip clock divider/prescaler is skipped.



## TIMER

**Q: Does timer (PRT) Channel 0 have a timer output ?**

A: No. If you need to have timer output, you have to use PRT channel 1 and use  $T_{OUT}$ .

Note that  $T_{OUT}$  pin is multiplexed with A18, so if you want to have  $T_{OUT}$ , you need to consider physical memory assignments (For this case, physical memory space is half size).

**Q: Can I use PRT channel(s) to count external events (as counter)?**

A: No. PRT's input is always  $PHI$  divided by 20 (Works as timer only).

**Q: I'm using timer for generating PWM pulses. Some-time I cannot get the desired pulse width for a cycle. Why?**

A: When you modify RLDR (Timer Reload Register), make sure the write transactions (write 2 bytes; RLDRL then followed by RLDRH) start just after TDR (Timer Data Register) reaches 0000H, and complete before reaching 0000H again. If reload condition occurs in the middle of these two writes, the Timer loads "Old" RLDRH and "New" RLDRL values.

**Q: What happens to the Timer after resuming IOSTOP mode if IOSTOP mode is entered during operation?**

A: The timer will hold current settings and values, and resume operation from the point when IOSTOP mode was entered.

---

## NOTES:

© 1997 by Zilog, Inc. All rights reserved. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Zilog, Inc. The information in this document is subject to change without notice. Devices sold by Zilog, Inc. are covered by warranty and patent indemnification provisions appearing in Zilog, Inc. Terms and Conditions of Sale only. Zilog, Inc. makes no warranty, express, statutory, implied or by description, regarding the information set forth herein or regarding the freedom of the described devices from intellectual property infringement. Zilog, Inc. makes no warranty of merchantability or fitness for any purpose. Zilog, Inc. shall not be responsible for any errors that may appear in this document. Zilog, Inc. makes no commitment to update or keep current the information contained in this document.

Zilog's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the customer and Zilog prior to use. Life support devices or systems are those which are intended for surgical implantation into the body, or which sustains life whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

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>