



AN009602-0808

Abstract

This Application Note describes how to build a simple system to prove and test Zilog's Z180 Microprocessor Unit (MPU) interfacing Zilog's Serial Communications Controller (SCC) at 10 MHz. Replacing Zilog's Z80[®] with the Z180 provides higher integration, reduced parts, more board space, increased processing speed, and greater reliability.

It also describes the design of a system using Z80180 MPU and Z85C30 SCC, both running at 10 MHz. Hereinafter, all references are to the Z180 and SCC.

Features

The system board is a vehicle for demonstration and evaluation of the 10 MHz interface and includes the following parts:

- Z8018010VSC Z180 MPU 10 MHz, PLCC package
- Z85C3010VSC C-MOS Z8530 SCC, 10 MHz, PLCC package
- 27C256 EPROM
- 55257 Static RAM

Discussion

Zilog's Z180 is a Z80-compatible High Integration device with various peripherals on-board. Using this device as an alternative to the Z80 CPU, reduces the number of parts and board space while increasing processing speed and reliability.

The serial communication devices on the Z180 include:

1. Two asynchronous channels.
2. One clocked serial channel.

This means handling synchronous serial communications protocols requires an off-chip 'multi-protocol SCC.' The SCC is the ideal device for this purpose.

SCC is the multi-protocol (@ 10 MHz) universal serial communication controller which supports most serial communication applications including Monosync, Bisync and SDLC at 2.5 Mbps speeds. Further, the wide acceptance of this device by the market ensures it is an 'industry standard' SCC. Also, the Z180 has special numbers for system clock frequencies of 6.144 and 9.216 MHz which generate exact baud rates for on-chip asynchronous serial communication channels. This is due to the SCC's on-chip, 16-bit wide baud rate generator for asynchronous ASCII communications.

The following 10 MHz interface explanation defines how the interrupt structure works. Also included is a discussion of the hardware and software considerations involved in running the system's communication board. This application note is written for a reader with strong working knowledge of the Z180 and SCC. It is not a tutorial for each device.



Interfaces

The following subsections explain the interfaces between the:

- Z180 and Memory
- Z180 and I/O
- Z180 and SCC

Basic goals of this system design are:

- System clock up to 10 MHz
- Using the Z8018010VSC (Z180 10 MHz PLCC package) to take advantage of 1 MB addressing space and compactness (DIP versions' addressing range is half; 512 KB)
- Using Z85C3010VSC (CMOS SCC 10 MHz PLCC package)
- Minimum parts count
- Worst case design
- Using EPLD for glue wherever possible
- Expendability

The design method for EPLD is using TTLs (74HCT) and then translating them into EPLD logic. This design uses TTLs and EPLDs.

Z180 to Memory Interface

The memory access cycle timing of the Z180 is similar to the Z80[®] CPU memory access cycle timing. The three classifications are:

- Opcode fetch cycle (Figure 1)
- Memory read cycle (Figure 2 on page 4)
- Memory write cycle (Figure 3 on page 7)

Table 1 on page 2 lists the Z180's basic timing elements for the opcode's fetch/memory read/write cycle.

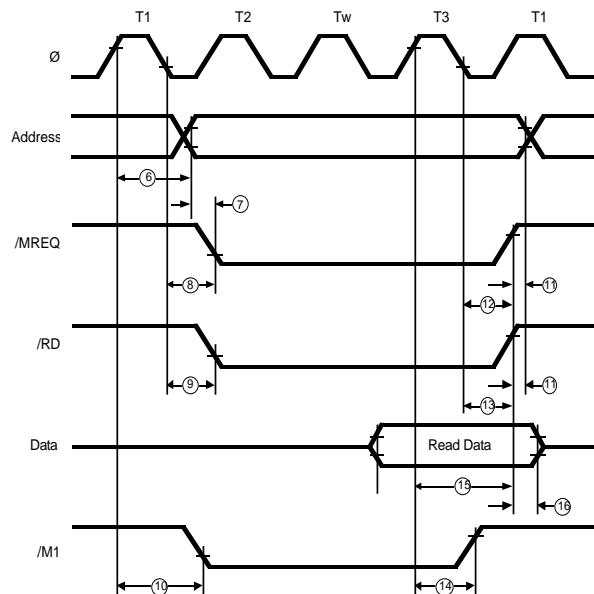


Figure 1. Z180 Opcode Fetch Cycle Timing (One Wait State)

Table 1. Z8018010 Timing Parameters for Opcode Fetch Cycle (Worst Case: Z180 10 MHz)

No	Symbol	Parameter	Min	Max	Units
1	tcyc	Clock Cycle Period	100		ns
2	tCHW	Clock Cycle High Width	40		ns
3	tCLW	Clock Cycle Low Width	40		ns
4	tcf	Clock Fall Time		10	ns

**Table 1. Z8018010 Timing Parameters for Opcode Fetch Cycle
(Worst Case: Z180 10 MHz) (Continued)**

No	Symbol	Parameter	Min	Max	Units
6	tAD	Clock High to Address Valid		70	ns
8	tMED1	Clock Low to /MREQ Low		50	ns
9	tRDD1	Clock Low to /RD Low		50	ns
11	tAH	Address Hold Time	10		ns
12	tMED2	Clock Low to /MREQ High		50	ns
15	tDRS	Data to Clock Setup	25		ns
16	tDRH	Data Read Hold Time	0		ns
22	tWRD1	Clock High to /WR Low		50	ns
23	tWDD	Clock Low to Write Data Delay		60	ns
24	tWDS	Write Data Setup to /WR Low	15		ns
25	tWRD2	Clock Low to /WR High		50	ns
26	tWRP	/WR Pulse Width		110	ns
27	tWDH	/WR High to Data Hold Time	10		ns

Note: Parameter numbers in this table are in the Z180 technical manual.

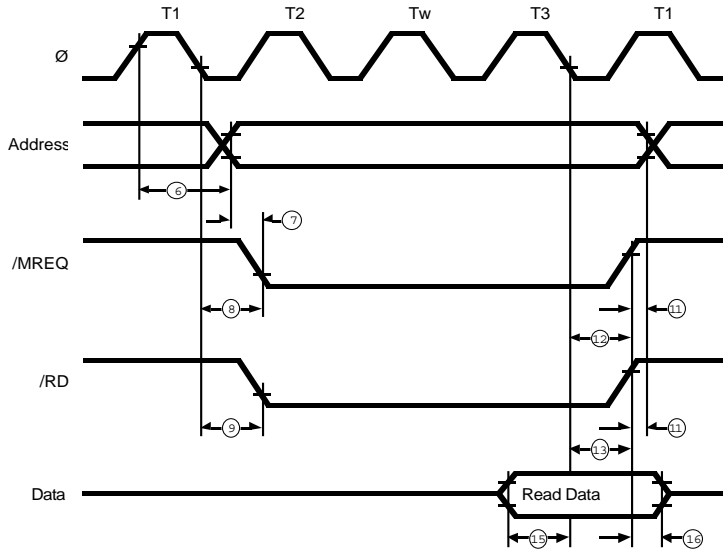


Figure 2. Z180 Memory Read Cycle Timing (One Wait State)

EPROM Interface

During an Opcode fetch cycle, data sampling of the bus is on the rising PHI clock edge of T3 and on the falling edge of T3 during a memory read cycle. Opcode fetch cycle data sample timing is half a clock cycle earlier. Table 2 lists how a memory read cycles’ timing requirements are easier than an opcode fetch cycle by half a PHI cycle time. If the

timing requirements for an Opcode fetch cycle meet specifications, the design satisfies the timing requirements for a memory read cycle.

Table 2 has some equations for an opcode fetch, memory read/write cycle.

Table 2. Parameter Equations (10 MHz) Opcode Fetch/Memory Read/Write Cycle

Parameters	Z180 Equation	Value (min)	Units
Address Valid to Data Valid (Opcode Fetch)	$2(1+w)tcyc-tAD-tDRS$	$105+100w$	ns
Address Valid to Data Valid (Memory Read)	$2(1+w)tcyc+tCHW+tcf-tAD-tDRS$	$155+100w$	ns
/MREQ Active to Data Valid (Opcode Fetch)	$(1+w)tcyc+tCLW-tMED1-tDRS$	$55+100w$	ns
/MREQ Active to Data Valid (Memory Read)	$(2+w)tcyc-tMED1-tDRS$	$105+100w$	ns



Table 2. Parameter Equations (10 MHz) Opcode Fetch/Memory Read/Write Cycle (Continued)

Parameters	Z180 Equation	Value (min)	Units
/RD Active to Data Valid (Opcode Fetch)	$(1+w)tcyc+tCLW-tRRD1-tDRS$	55+100w	ns
/RD Active to Data Valid (Memory Read)	$(2+w)tcyc-tRRD1-tDRS$	105+100w	ns
Memory Write Cycle /WR Pulse Width	$tWRP+w*tcyc$	110+100w	ns

Note: * w is the number of wait states.

The propagation delay for the decoded address and gates in the previous calculation is zero. Hence, on the real design, subtracting another 20-30 ns to pay for propagation delays, is possible. The 27C256

provides the EPROM for this board. Typical timing parameters for the 27C256 are in [Table 3](#).

Table 3. EPROM (27C256) Key Timing Parameters (Values May Vary Depending On Mfg.)

Parameter	Access Time		
	170 ns Max	200 ns Max	250 ns Max
Addr Access Time	170	200	250
/E to Data Valid	170	200	250
/OE to Data Valid	75	75	100

Note: 'Access Time' applies /E to data valid. '/OE active to data valid' is shorter than 'address access time'. Hence, the interface logic for the EPROM is: Realize a 170 ns or faster EPROM access time by adding one wait state (using the on-chip wait state generator of the Z180). A 200 ns requirement uses two wait states for memory access.

SRAM Interface

Table 4 has timing parameters for 256 kb SRAM for this design.

Table 4. 256 kb SRAM Key Timing parameters (Values May Vary Depending On Mfg.)

Parameter	Access Time		
	85 ns	100 ns	150 ns
	Min	Min	Min
Read Cycle			
/E to Data Valid	85	100	150
/G to Data Valid	45	40	60
Write Cycle			
Write Cycle Time	85	100	150
Addr Valid to End of Write	75	80	100
Chip Select to End of Write	75	80	100
Data Select to End of Write	40	40	60
Write Pulse Width	60	60	90
Addr Setup Time	0	0	0

SRAM Read Cycle—An SRAM read cycle shares the same considerations as an EPROM interface.

Like EPROM, SRAMs’ “access time” applies /G to data valid, and “/E active to data valid” is shorter than “access time.” This design allows the use of a 150 ns access time SRAM by adding one wait state (using the on-chip wait state generator of the Z180). The circuit is common to the EPROM memory read cycle.

No wait states are necessary if there is a 85 ns, or faster, access time by using SRAMs. Since the Z180 has on-chip MMU with 85 ns or faster SRAM just copy the contents of EPROM (application program starts at logical address 0000h) into SRAM after power on. Set up the MMU to SRAM area to override the EPROM area and stop inserting wait states. With this scheme, you can get the highest performance with moderate cost.

SRAM Write Cycle—During a Z180 memory write cycle, the Z180 write data is stable before the falling edge of /WR (Z180 parameter #24; 15 ns min at 10 MHz). It is stable throughout the write cycle (Z180 parameter #27; 10 ns min at 10 MHz). Further, the address is fixed before the falling edge of /WR. The /WR pulse width meets the SRAM’s spec (see Table 2 on page 4).

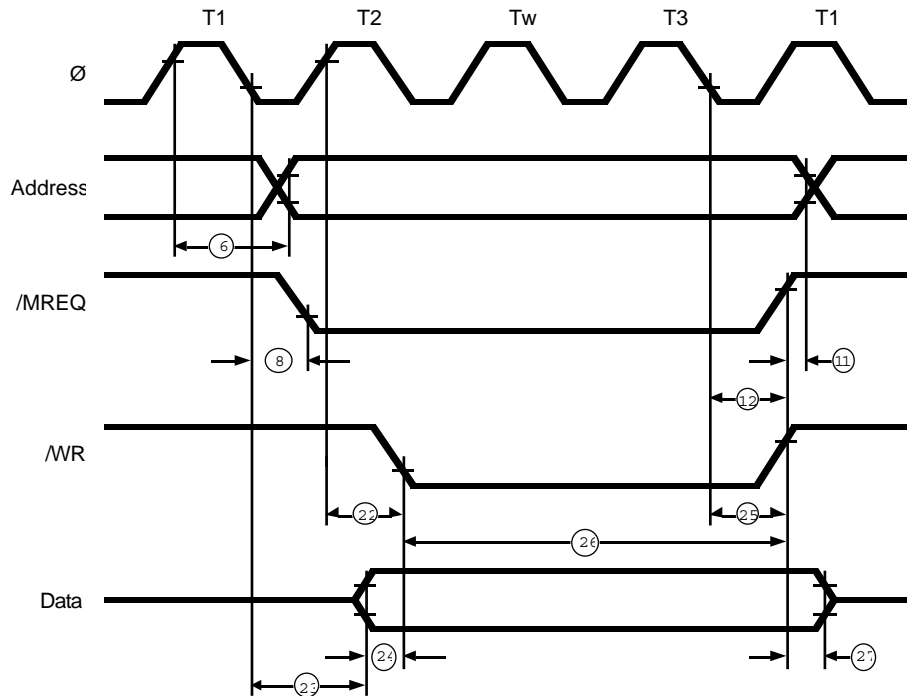


Figure 3. Z180 Memory Write Cycle Timing (One Wait State)

Memory Interface Logic

The memory devices (EPROM and SRAM) for this design are 256 kb (32 KB). There are two possible memory interface designs:

Connect Address Decode output to /E input. Put the signal generated by /RD and /MREQ ANDed together to /OE of EPROM and SRAM. Put the signal generated by /WR and /MREQ ANDed together to the /WE pin of SRAM (Figure 4 on page 8).

Connect the signal Address ANDed together with inactive /IORQ to the /E input. Connect /RD to /OE of EPROM and SRAM, and /WR to /WE pin of SRAM (Figure 5 on page 9).

Using the second method, there could be a narrow glitch on the signal to the /E-pin during I/O cycles and the Interrupt acknowledge cycle. During I/O cycles, /IORQ and /RD or /WR go active at almost the same time. Since the delay times of these signals are similar there is no “overlapping time” between /CE generated by the address (/IORQ inactive), and /WR or /RD active. During the Interrupt Acknowledge cycle, /WR and /RD signals are inactive.

To keep the design simple and flexible, use the second method (Figure 5 on page 9). To expand memory, decode the address A15 NANDed with /USRRAM//USRRAM and /IORQ to produce /CSRAM or /CSROM. These are chip select inputs to chips 55257 or 27C256, respectively. This either



disables or enables on-board ROM or RAM depending upon selection control.

The circuit on [Figure 5](#) on page 9 gives the physical memory address as displayed in [Figure 6](#) on page 9.

If there are no Z80[®] peripherals and /M1 is enabled (M1E bit in Z180 OMCR register set to 1), active wait states occur only during opcode fetch cycles ([Figure 7](#) on page 10). If the M1E bit is cleared to

0, /M1E is active only during the Interrupt Acknowledge cycle and Return from Interrupt cycle. This case depends on the propagation delay of the address decoder which uses 135 ns or faster EPROM access time (assume there is 20 ns propagation delay). [Figure 7](#) on page 10 displays the example of this implementation. (Extends Opcode Fetch Cycle Only; Not Working in Z Mode of Operation)

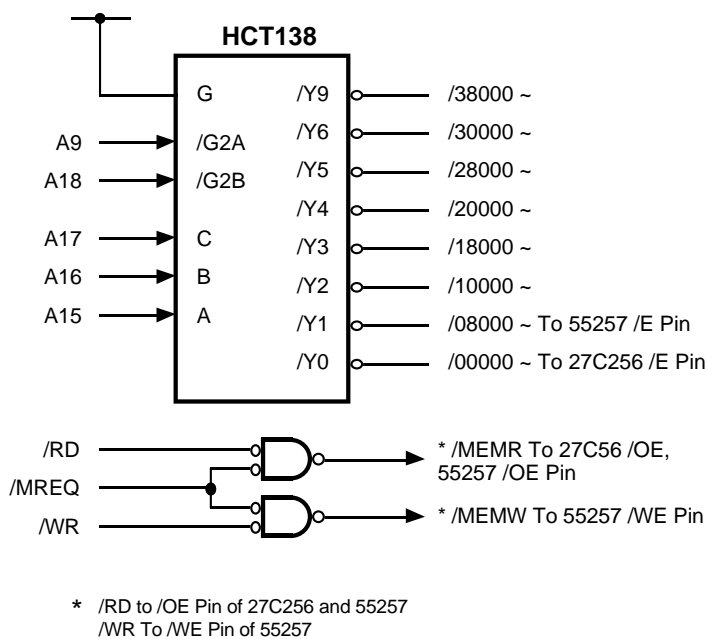


Figure 4. Memory Interface Logic

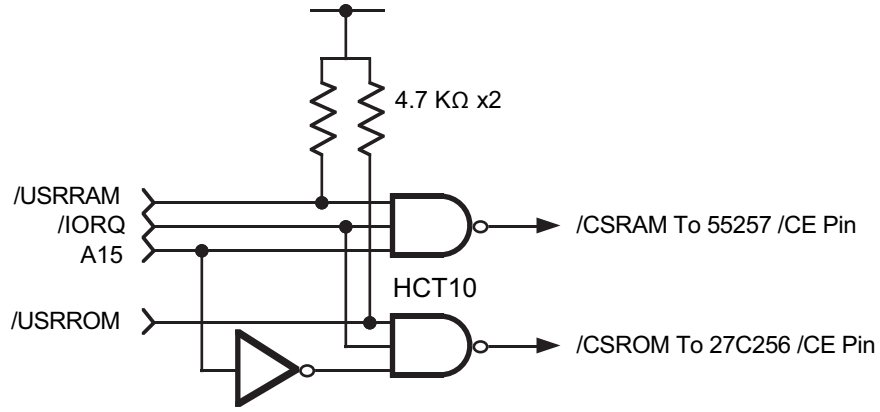


Figure 5. Memory Interface Logic

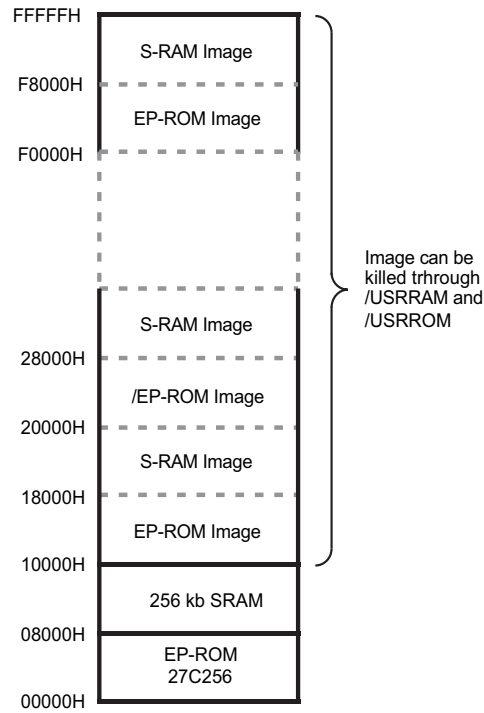


Figure 6. Physical Memory Address Map

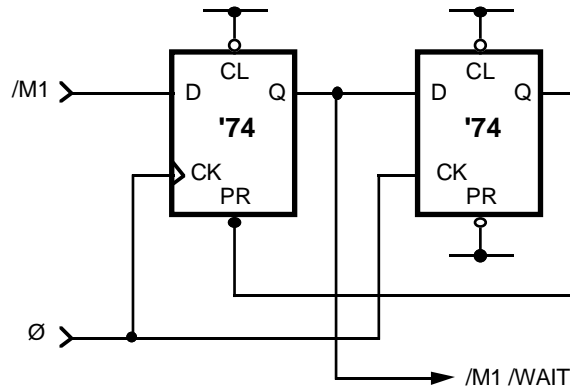


Figure 7. Wait State Generator Logic

Z180 to I/O Interface

The Z180 I/O read/write cycle is similar to the Z80[®] CPU if you clear the /IOC bit in the OMCR register to 0 (Figure 8 and Figure 9 on page 11).

Table 5 on page 11 lists the Z180 key parameters for an I/O cycle.

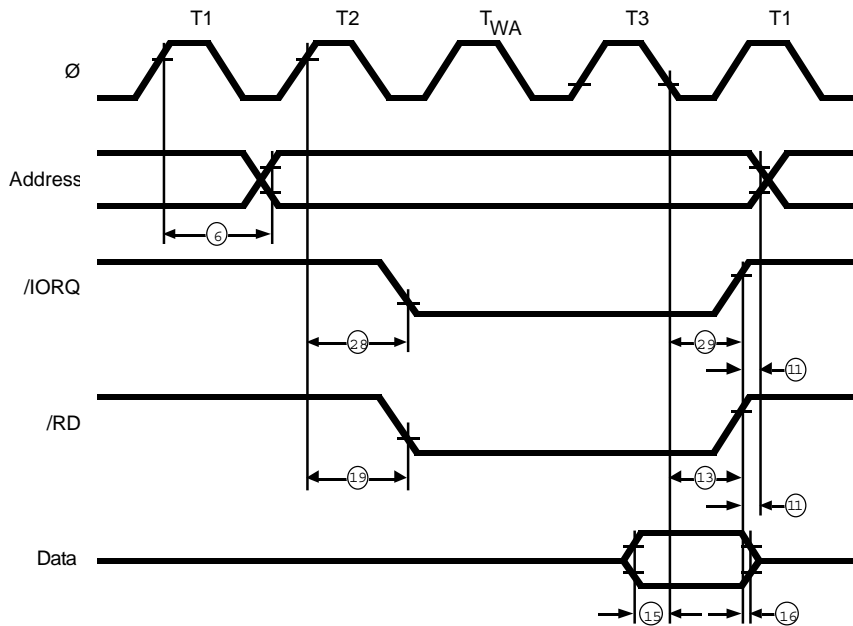


Figure 8. Z180 I/O Read Cycle Timing (/IOC = 0)

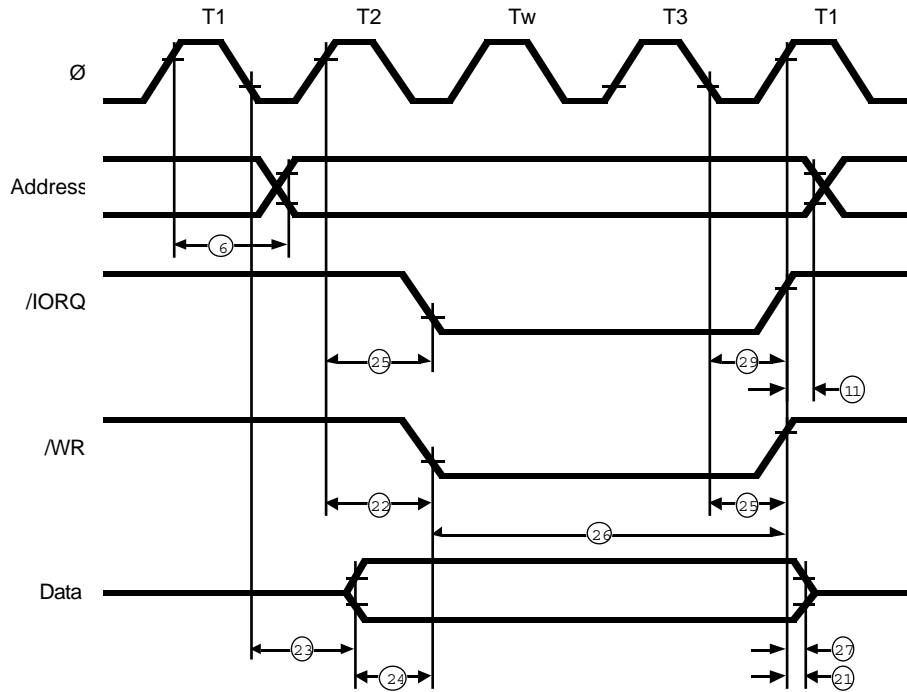


Figure 9. Z180 I/O Write Cycle Timing

Table 5. Z8018010 Timing Parameters for I/O Cycle (Worst Case)

Sr. No	Symbol	Parameter	Min	Max	Units
1	t _{cy}	Clock Cycle Period	100		ns
2	t _{CHW}	Clock Cycle High Width	40		ns
3	t _{CLW}	Clock Cycle Low Width	40		ns
4	t _{cf}	Clock Fall Time		10	ns
6	t _{AD}	Clock High to Address Valid		70	ns
9	t _{RDD1}	Clock High to /RD Low IOC=0		55	ns
11	t _{AH}	Address Hold Time	10		ns
13	t _{RDD2}	Clock Low to /RD High		50	ns
15	t _{DRS}	Data to Clock Setup	25		ns



Table 5. Z8018010 Timing Parameters for I/O Cycle (Worst Case) (Continued)

Sr. No	Symbol	Parameter	Min	Max	Units
16	tDRH	Data Read Hold Time	0		ns
21	tWDS	Clock High to Data Float Delay		60	ns
22	tWRD1	Clock High to /WR Low		50	ns
23	tWDD	Clock Low to Write Data Delay		60	ns
24	tWDS	Write Data Setup to /WR Low	15		ns
25	tWRD2	Clock Low to /WR High		50	ns
26a	tWRP	/WR Pulse Width (I/O Write)	210		ns
27	tWDH	/WR High to Data Hold Time	10		ns
28	tIOD1	Clock High to /IORQ Low IOC=0		55	ns
29	tIOD2	Clock Low to /IORQ High		50	ns

Note: Parameter numbers in this table are the numbers in the Z180 technical manual.

If you are familiar with the Z80[®] CPU design, the same interfacing logic applies to the Z180 and I/O interface (see Figure 10). This circuit generates /IORQ (Read) or IORD (Write) for peripherals from inputs /IORQ, /RD, and /WR. The address decodes the Chip Select signal.

► **Note:** *If you have Z80 peripherals, the decoder logic decodes only from addresses (does not have /IORQ).*

The Z180 signals /IORQ, /RD, and /WR are active at about the same time (Parameters #9, 22, 28). However, most of the Z80 peripherals require /CE to /RD or /WR setup time.

Since the Z180 occupies 64 bytes of I/O addressing space for system control and on-chip peripherals, there are no overlapping I/O addresses for off-chip peripherals. In this design, leave the area as default or assign on-chip registers at I/O address 0000h to 003Fh.

Figure 10 displays a simple address decoder (the required interface signals, other than address

decode outputs, are discussed in the following sections).

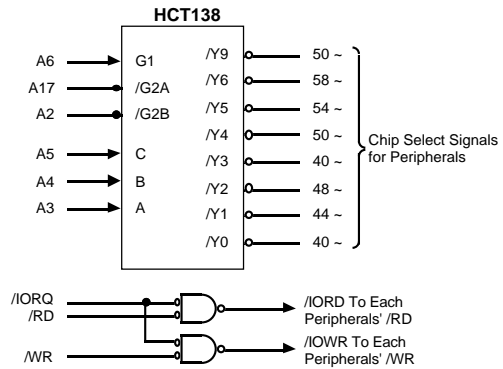


Figure 10. I/O Interface Logic (Example)

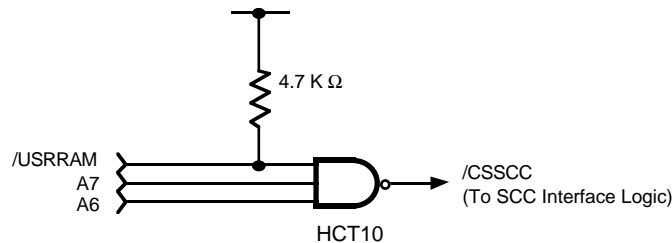


Figure 11. I/O Address Decoder for this Board

When expanding this board to enable other peripherals, the decoded address A6/A7 is Nanded with USRIO to produce the Chip Enable (CSSCC) output signal (HC10). The SCC registers are assigned from address xxC0h to xxC3h ; with image, they occupy xxC0h to xFFh . To add wait states during I/O transactions, use the Z180 on-chip wait state generator instead of external hardware logic.

If there is a Z80[®] PIO on board in a Z-mode of operation (that is, clear /M1E in OMCR register to zero) and after enabling a Z80 PIO interrupt, zero is written to M1TE in the OMCR register. Without a zero, there is no interrupt from the Z80 PIO. The Z80 PIO requires /M1 to activate an interrupt circuit after enabling interrupt by software.

Z180 to SCC Interface

The following subsections discuss the various parameters between the Z180/SCC interface:

- CPU hardware
- I/O operation (read/write)
- SCC interrupts
- Z80 interrupt daisy-chain operation
- SCC interrupt daisy-chain operation
- I/O cycles

CPU Hardware Interfacing

The hardware interface has three basic groups of signals:

1. Data bus
2. System control
3. Interrupt control.

For more detailed signal information, refer to Zilog's Technical Manuals, and Product Specifications for each device.

Data Bus Signals

D7-D0—Data bus (Bidirectional, tri-state). This bus transfers data between the Z180 and SCC.

System Control Signals

A//B, C//D—Register select signals (Input). These lines select the registers.

/CE—Chip enable (Input, active Low). /CE selects the proper peripheral for programming. /CE is gated with /IORQ or /MREQ to prevent false chip selects during other machine cycles.

/RD+—Read (input, active Low). /RD activates the chip-read circuitry and gates data from the chip onto the data bus.



/WR+—Write (Input, active Low). /WR strobes data from the data bus into the peripheral.

Chip reset occurs when /RD and /WR are active simultaneously.

Interrupt Control

/INTACK—Interrupt Acknowledge (input, active Low). This signal shows an Interrupt Acknowledge cycle which combines with /RD to gate the interrupt vector onto the data bus.

/INT—Interrupt request (output, open-drain, active Low).

IEI—Interrupt Enable In (input, active High).

IEO—Interrupt Enable Out (Output, active High).

These lines control the interrupt daisy chain for the peripheral interrupt response.

SCC I/O Operation

The SCC generates internal control signals from /RD or /WR. Since PCLK has no required phase relationship to /RD or /WR, the circuitry generating these signals provides time for meta stable conditions to disappear.

The SCC starts the different operating modes by programming the internal registers. Accessing these internal registers occurs during I/O Read and Write cycles, described below.

Read Cycle Timing

Figure 12 displays the SCC Read cycle timing. All register addresses and /INTACK are stable throughout the cycle. The timing specification of SCC requires that the /CE signal (and address) be stable when /RD is active

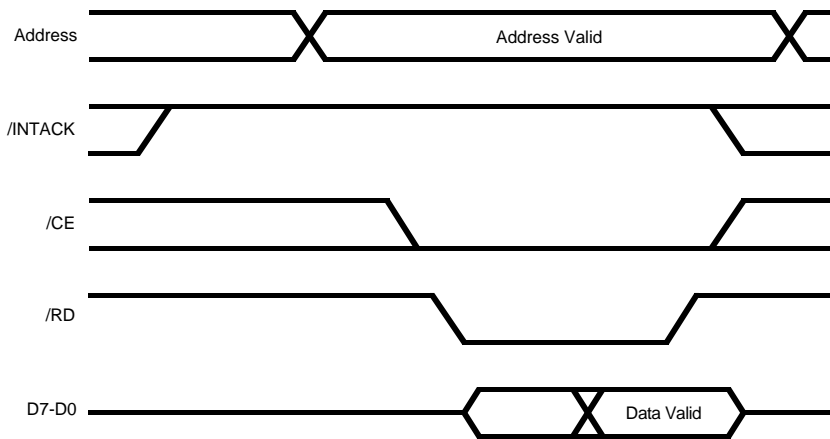


Figure 12. SCC Read Cycle Timing

Write Cycle Timing

Figure 13 on page 15 displays the SCC Write cycle timing. All register addresses and /INTACK are stable throughout the cycle. The timing specification of the SCC requires that the /CE signal (and

address) be stable when /RD is active. Data is available to the SCC before the falling edge of /WR and remains active until /WR goes inactive

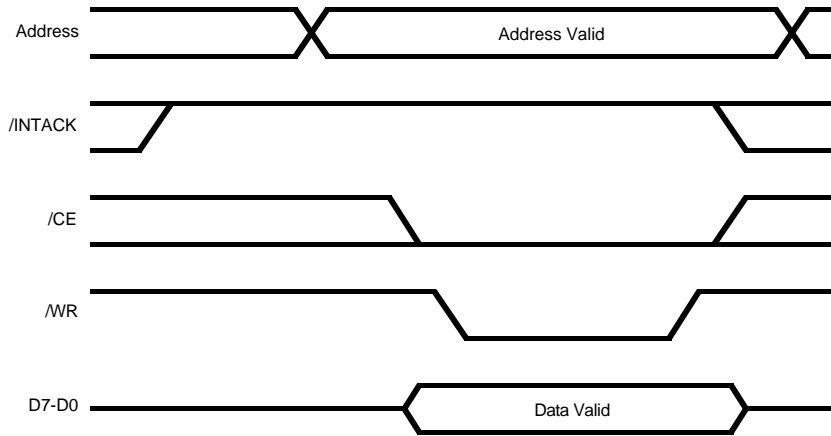


Figure 13. SCC Write Cycle Timing

SCC Interrupt Operation

Understanding SCC interrupt operations requires a basic knowledge of the Interrupt Pending (IP) and Interrupt Under Service (IUS) bits in relation to the daisy chain. The Z180 and SCC design allow no additional interrupt requests during an Interrupt Acknowledge cycle. This permits the interrupt daisy chain to settle, ensuring proper response of the interrupt device.

The IP bit sets in the SCC for CPU intervention requirements (that is, buffer empty, character available, error detection, or status changes). The interrupt acknowledge cycle does not reset the IP bit. The IP bit clears by a software command to the SCC, or when the action that generated the interrupt ends, for example, reading a receive character for receive interrupt. Others are, writing data to the transmitter data register, issuing Reset Tx interrupt pending command for Tx buffer empty interrupt, etc.). After servicing the interrupt, other interrupts can occur.

The IUS bit means the CPU is servicing an interrupt. The IUS bit sets during an Interrupt Acknowledge cycle if the IP bit sets and the IEI line is High. If the IEI line is Low, the IUS bit is not set. This

keeps the device from placing its vector onto the data bus.

The IUS bit clears in the Z80[®] peripherals by decoding the RETI instruction. A software command also clears the IUS bit in the Z80 peripherals. Only software commands clear the IUS bit in the SCC.

Z80 Interrupt Daisy-Chain Operation

In the Z80 peripherals, both IP and IUS bits control the IEO line and the lower portion of the daisy chain. When a peripheral's IP bit sets, the IEO line goes Low. This is true regardless of the state of the IEI line. Additionally, if the peripheral's IUS bit clears and its IEI line is High, the /INT line goes Low.

The Z80 peripherals sample for both /M1 and /IORQ active (and /RD inactive) to identify an Interrupt Acknowledge cycle. When /M1 goes active and /RD is inactive, the peripheral detects an Interrupt Acknowledge cycle and allows its interrupt daisy chain to settle. When the /IORQ line goes active with /M1 active, the highest priority interrupting peripheral places its interrupt vector onto the data bus. The IUS bit also sets to show that the peripheral is now under service. As long as

the IUS bit sets, the IEO line remains Low. This inhibits any lower priority devices from requesting an interrupt.

When the Z180 CPU executes the RETI instruction, the peripherals check the data bus and the highest priority device under service resets its IUS bit.

SCC Interrupt Daisy-Chain Operation

In the SCC, the IUS bit normally controls the state of the IEO line. The IP bit affects the daisy chain

only during an Interrupt Acknowledge cycle. Since the IP bit is normally not part of the SCC interrupt daisy chain, there is no need to decode the RETI instruction. To allow for control over the daisy chain, the SCC has a Disable Lower Chain (DLC) software command that pulls IEO Low. This selectively deactivates parts of the daisy chain regardless of the interrupt status. Table 6 lists the truth table for the SCC interrupt daisy chain control signals during certain cycles. Table 12 on page 27 lists SCC Test Program for the Interrupt.

Table 6. SCC Daisy Chain Signal Truth Table

During Idle State				During INTACK Cycle			
IEI	IP	IUS	IEO	IEI	IP	IUS	IEO
0	X	X	0	0	X	X	0
1	X	0	1	1	1	X	0
1	X	1	0	1	X	1	0
1	0	0	1				

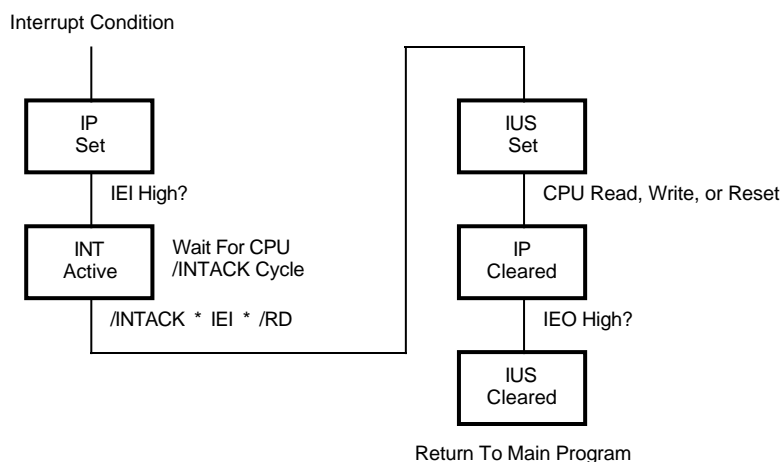


Figure 14. SCC Interrupt Status Diagram

The SCC uses /INTACK (Interrupt Acknowledge) for recognition of an interrupt acknowledge cycle. This pin, used with /RD, allows the SCC to gate its interrupt vector onto the data bus. An active /RD signal during an interrupt acknowledge cycle performs two functions. First, it allows the highest priority device requesting an interrupt to place its vector on the data bus. Secondly, it sets the IUS bit in the highest priority device to show the device is now under service.

Input/Output Cycles

Although the SCC is a universal design, certain timing parameters differ from the Z180 timing. The following subsections discuss the I/O interface for the Z180 MPU and SCC.

Z180 MPU to SCC Interface

Table 7 lists key parameters of the 10 MHz SCC for I/O read/write cycles.

Table 7. 10 MHz SCC Timing Parameters for I/O Read/Write Cycle (Worst Case)

No	Symbol	Parameter	Min	Max	Units
6	TsA(WR)	Address to /WR Low Setup	50		ns
7	ThA(WR)	Address to /WR High Hold	0		ns
8	TsA(RD)	Address to /RD Low Setup	50		ns
9	ThA(RD)	Address to /RD High Hold	0		ns
16	TsCEI(WR)	/CE Low to /WR Low Setup	0		ns
17	ThCE(WR)	/CE to /WR High Hold	0		ns
19	TsCEI(RD)	/CE Low to /RD Low Setup	0		ns
20	ThCE(RD)	/CE to /RD High Hold	0		ns
22	TwRDI	/RD Low Width	125		ns
25	TdRDf(DR)	/RD Low to Read Data Valid		120	ns
27	TdA(DR)	Address to Read Data Valid		180	ns
28	TwWRI	/WR Low Width	125		ns
29	TsDW(WR)	Write Data to /WR Low Setup	10		ns
30	TdWR(W)	Write Data to /WR High Hold	0		ns

SCC I/O Read/Write Cycle

Assume that the Z180 MPU's /IOC bit in the OMCR (Operation Mode Control Register) clears to 0 (this condition is a Z80[®] compatible timing mode for /IORQ and /RD). The following are several design points to consider (also see Table 3 on page 5).

I/O Read Cycle

Parameters 8 and 9 explains that Address is stable 20 ns before the falling edge of /RD and until /RD goes inactive.

Parameters 19 and 20 explains that /CE is stable at the falling edge of /RD and until /RD goes inactive.

Parameter 22 explains the /RD pulse width is wider than 125 ns.

Parameters 25 and 27 explains that Read data is available on the data bus 120 ns later than the falling edge of /RD and 180 ns from a stable Address.

I/O Write Cycle

Parameters 6 and 7 explains that Address is stable 50 ns before the falling edge of /WR and is stable until /WR goes inactive.

Parameters 16 and 17 explains that /CE is stable at the falling edge of /WR and is stable until /W goes inactive.

Parameter 28 explains /WR pulse width is wider than 125 ns.

Parameters 28 and 29 explains that Write data is on the data bus 10 ns before the falling edge of /WR. It is stable until the rising edge of /WR.

[Table 8](#) and [Table 9](#) list the worst case SCC parameters calculating Z180 parameters at 10 MHz.

Table 8. Parameter Equations Worst Case (Without Delay Signals - No Wait State)

SCC			
Parameters	Z180 Equation	Value (min)	Units
TsA(RD)	$t_{cyc} - t_{AD} + t_{RDD1}$	30	ns
TdA(DR)	$3t_{cyc} + t_{CHW} + t_{cf} - t_{AD} - t_{DRS}$	245	ns
TdRDf(DR)	$2t_{cyc} + t_{CHW} + t_{cf} - t_{RDD1} - t_{DRS}$	160	ns
TwRDI	$2t_{cyc} + t_{CHW} + t_{cf} - t_{DRS} + t_{RDD2}$	185	ns
TsA(WR)	$t_{cyc} - t_{AD} + t_{WRD1}$	30	ns
TsDW(WR)	tWDS	15	ns
TwWRI	tWRP	210	ns

Table 9. Parameter Equations

Z180			
Parameters	SCC Equation	Value (min)	Units
tDRS	Address		
	$3t_{cyc} + t_{CHW} - t_{AD} - t_{dA(DR)RD}$	241	ns
	$2t_{cyc} + t_{CHW} - t_{RDD1} - t_{dRD(DR)}$	184	ns

I/O Read Cycle

These tables show that a delay of the falling edge of /RD satisfies the SCC TsA(RD) timing requirement of 50 ns min. The Z180 calculated value is 30 ns min for the worst case. Also, Z180 timing specification tAH (Address Hold time) is 10 ns min. The SCC timing parameters ThA(RD) {Address to /RD

High Hold} and ThCE(RD) {/CE to /RD High Hold} are minimum at 0 ns. The rising edge of /RD is early to guarantee these parameters when considering address decoders and gate propagation delays.

I/O Write Cycle

Delay the falling edge of /WR to satisfy the SCC TsA (/WR) timing requirement of 50 ns min. The Z180 calculates 30 ns min worst case. Further, the Z180 timing specifications tAH (Address Hold time) and tWDH (/WR High to data hold time) are both 10 ns min. The SCC timing parameters ThA(WR) {Address to /WR High Hold}, ThCE(WR) {/CE to /WR High Hold} and TdWR(W) {Write data to /WR High hold} are a

minimum of 0 ns. The rising edge of /WR is early to guarantee these parameter requirements.

This circuit depicts logic for the I/O interface and the Interrupt Acknowledge Interface for 10 MHz clock of operation. Figure 15 on page 19 is the I/O read/write timing chart (discussions of timing considerations on the Interrupt Acknowledge cycle and the circuit using EPLD occur later).

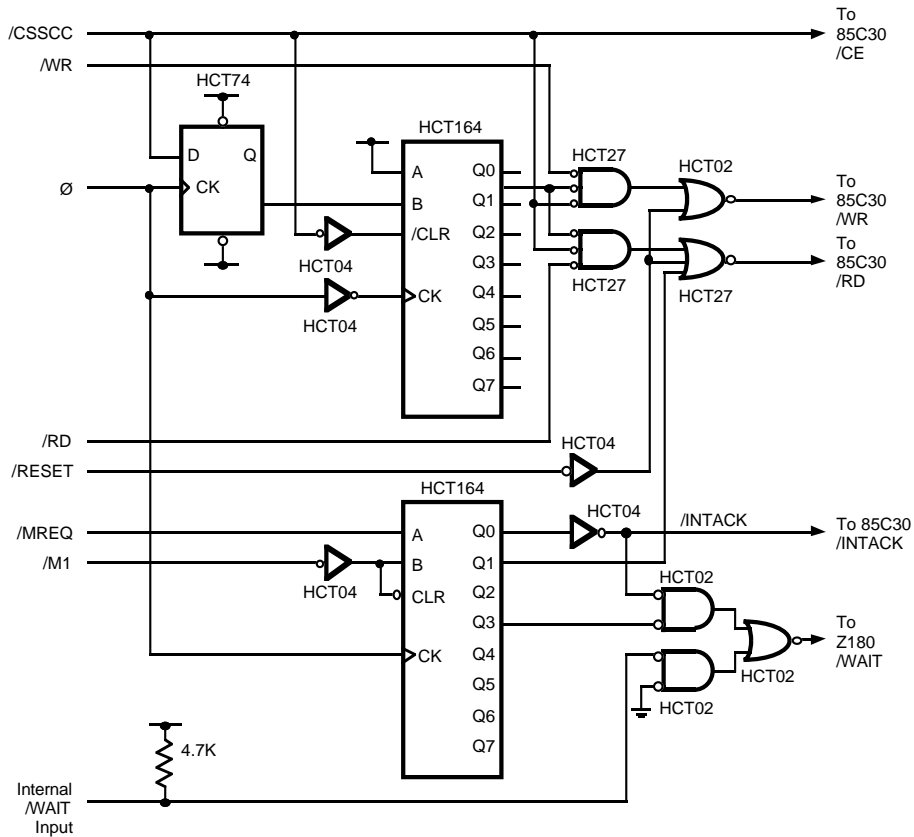


Figure 15. SCC I/O Read/Write Cycle Timing

This circuit works when [(Lower HCT164's CLK. to Z180 /WAIT.) + t_{ws} < t_{CHW}]. If you are running your system slower than 8 MHz, remove the HCT74, D-Flip/Flop in front of HCT164. Connect the inverted CSSCC to the HCT164 B input. This is a required Flip/Flop because the Z180 timing specification on tIOD1 (Clock High to /IORQ Low,

IOC=0) is maximum at 55 ns This is longer than half the PHI clock cycle. Sample it using the rising edge of clock, otherwise, HCT164 does not generate the same signals.

The RESET signal feeds the SCC /RD and /WR through HCT27 and HCT02 to supply the hard-

ware reset signal. To reduce the gate count, drop these gates and make the SCC reset by its software command. The SCC software reset–0C0h to Write Register 9, “Hardware Reset command” has the same effect as hardware reset by “Hardware.”

cycle. The SCC timing parameters that are significant during Interrupt Acknowledge cycles are in [Table 10](#) on page 20. The Z180 timing parameters are in [Table 10](#). The reference numbers in [Table 10](#) and [Table 11](#) refer to [Figure 15](#) on page 19.

Interrupt Acknowledge Cycle Timing

The primary timing differences between the Z180 and SCC occur in the Interrupt Acknowledge

Table 10. 10 MHz SCC Timing Parameters for Interrupt Acknowledge Cycle

No	Symbol	Parameter	Min	Max	Units
13	TsIAi(RD)	/INTACK Low to /RD Low Setup	130		ns
14	ThIA(RD)	/INTACK High to /RD High Hold	0		ns
15	ThIA(PC)	/INTACK to PCLK High Hold	30		ns
38	TwRDA	/INTACK Low to /RD Low Delay (Acknowledge)	125		ns
39	TwRDA	/RD (Acknowledge) Width	125		ns
40	TdRDA(DR)	/RD Low (Acknowledge) to Read Data Valid Delay		120	ns
41	TsIEI(RDA)	IEI to /RD Low (Acknowledge) Setup Time	95		ns
42	ThIEI(RDA)	IEI to /RD High (Acknowledge) Hold Time	0		ns
43	TdIEI(IEO)	IEI to IEO Delay		175	ns

Table 11. Z180 Timing Parameters Interrupt Acknowledge Cycles (Worst Case Z180)

No	Symbol	Parameter	Min	Max	Units
10	tM1D1	Clock High to /M1 Low		60	ns
14	tM1D2	Clock High to /M1 High		60	ns
15	tDRS	Data to Clock Setup	25		ns
16	tDRH	Data Read Hold Time	0		ns
28	tIOD1	Clock LOW to /IORQ Low		50	ns
29	tIOD2	Clock LOW to /IORQ High		50	ns
30	tIOD3	/M1 Low to /IORQ Low Delay	200		ns

Note: Parameter numbers in this table are the numbers in the Z180 technical manual.

During an Interrupt Acknowledge cycle, the SCC requires both /INTACK and /RD to be active at certain times. Since the Z180 does not issue either /INTACK or /RD, external logic generates these signals.

The Z180 is in a Wait condition until the vector is valid. If there are other peripherals added to the interrupt priority daisy chain, more Wait states may be necessary to give it time to settle. Allow enough time between /INTACK active and /RD active for the entire daisy chain to settle.

There is no need of decoding the RETI instruction used by the Z80[®] peripherals since the SCC daisy chain does not use IP, except during Interrupt Acknowledge. The SCC and other Z8500 peripherals have commands that reset the individual IUS flag.

External Interface for Interrupt Acknowledge Cycle: The bottom half of [Figure 16](#) on page 22 is the interface logic for the Interrupt Acknowledge cycle.

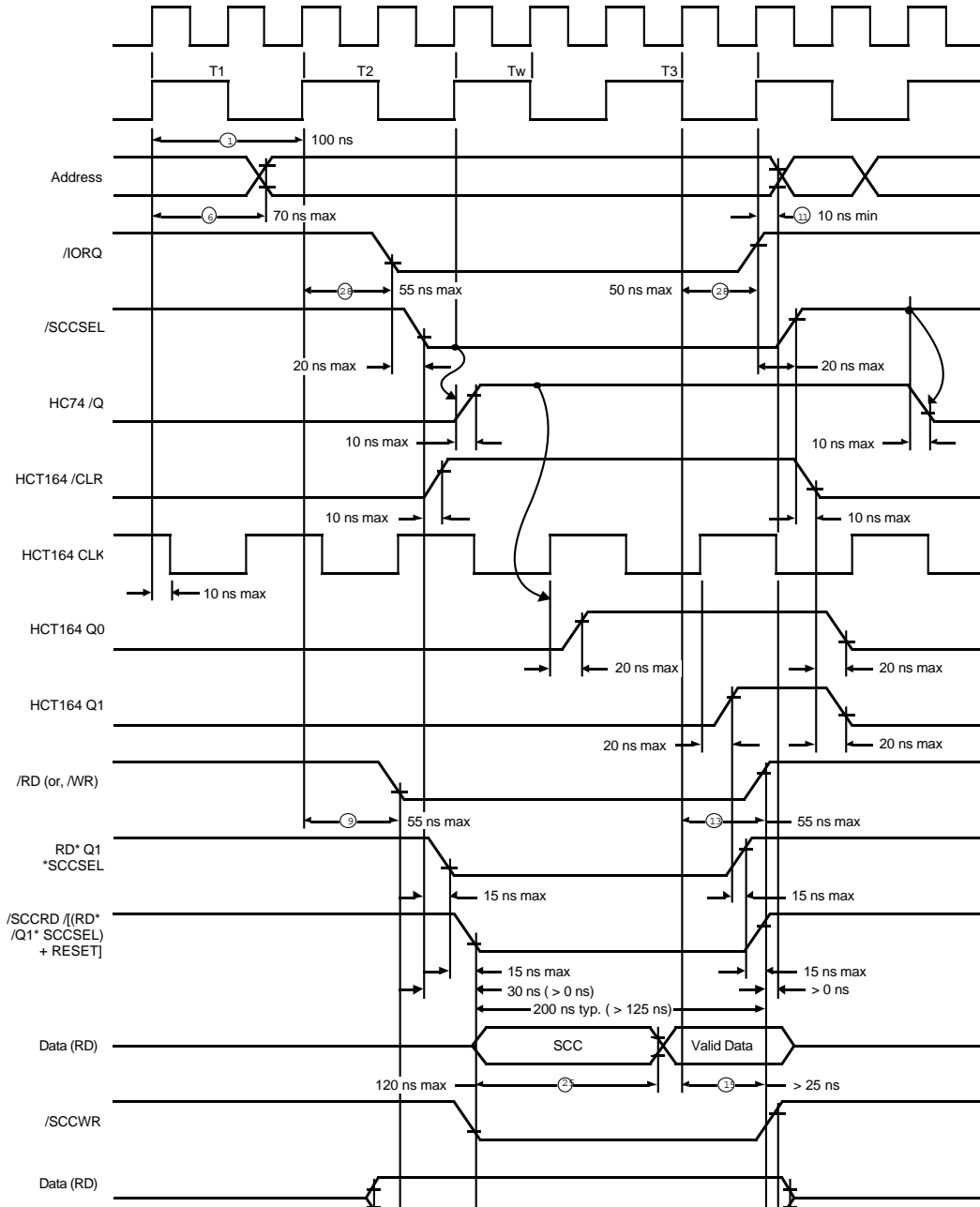


Figure 16. Z180 to SCC Interface Logic (Example)

The primary chip in this logic is the Shift register (HCT164), which generates /INTACK, /SCCRD and /WAIT. During I/O and normal memory access cycles, the Shift Register (HCT164) remains cleared because the /M1 signal is inactive during the opcode fetch cycle. Since the Shift Register

output is Low, control of /SCCRD and /WAIT is by other system logic and gated through the NOR gate (HCT27). During I/O and normal memory access cycles, /SCCRD and /SCCW are generated from the system /RD and /WR signals, respectively. The generation is by the logic at the top of [Figure 17](#)

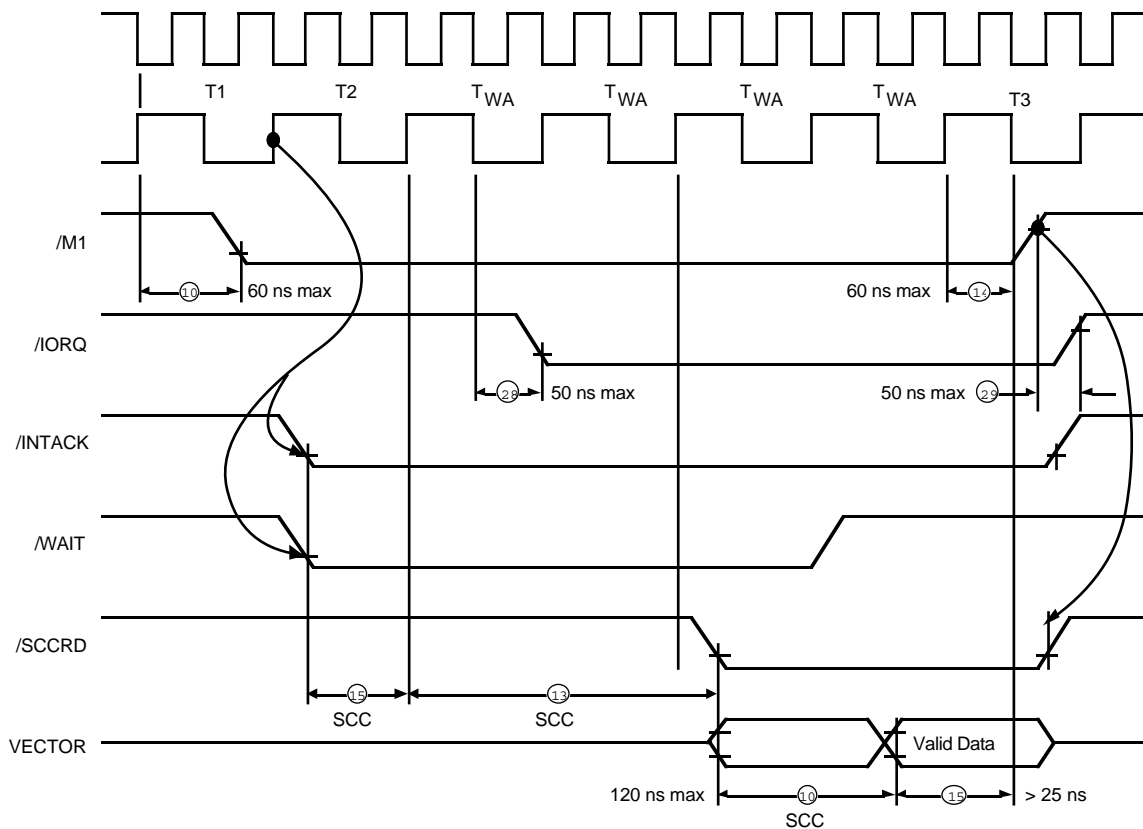


Figure 17. SCC Interrupt Acknowledge Cycle Timing

Normally, an Interrupt Acknowledge cycle appears from the Z180 during /M1 and /IORQ active (which is detected on the third rising edge of PHI after T1). To get an early sign of an Interrupt Acknowledge cycle, the Shift register decodes an active /M1. This is during the presence of an inactive /MREQ on the rising edge of T2.

During an Interrupt Acknowledge cycle, the /INTACK signal is generated on the rising edge of

T2. Since it is the presence of /INTACK and an active /SCCRD that gates the interrupt vector onto the data bus, the logic also generates /SCCRD at the proper time. The timing parameter of concern here is TdIAi(RD) [/INTACK to /RD (Acknowledge) Low delay]. This time delay allows the interrupt daisy chain to settle so the device requesting the interrupt places its interrupt vector onto the data bus.

The Shift Register allows enough time delay from the generation of /INTACK before it generates /SCCRD. During this delay, it places the Z180 into a Wait state until the valid interrupt vector is placed onto the data bus. If the time between these two signals is not enough for daisy chain settling, more time is added by taking /SCCRD and /WAIT from a later position on the Shift Register. If there is a requirement for more wait states, the time is calculated by PHI cycles.

Using EPLD

Figure 18 and Figure 19 on page 25 display the logic using either random gate logic to EPLD is to disassemble MSIs' logic EPLD or the circuit of this system. The EPLD is ALTERA into SSI level, and then simplify the logic. 610 which is a 24-Pin EPLD. The method to convert random gate logic to EPLD is to disassemble MSIs' logic into SSI level, and then simplify the logic.

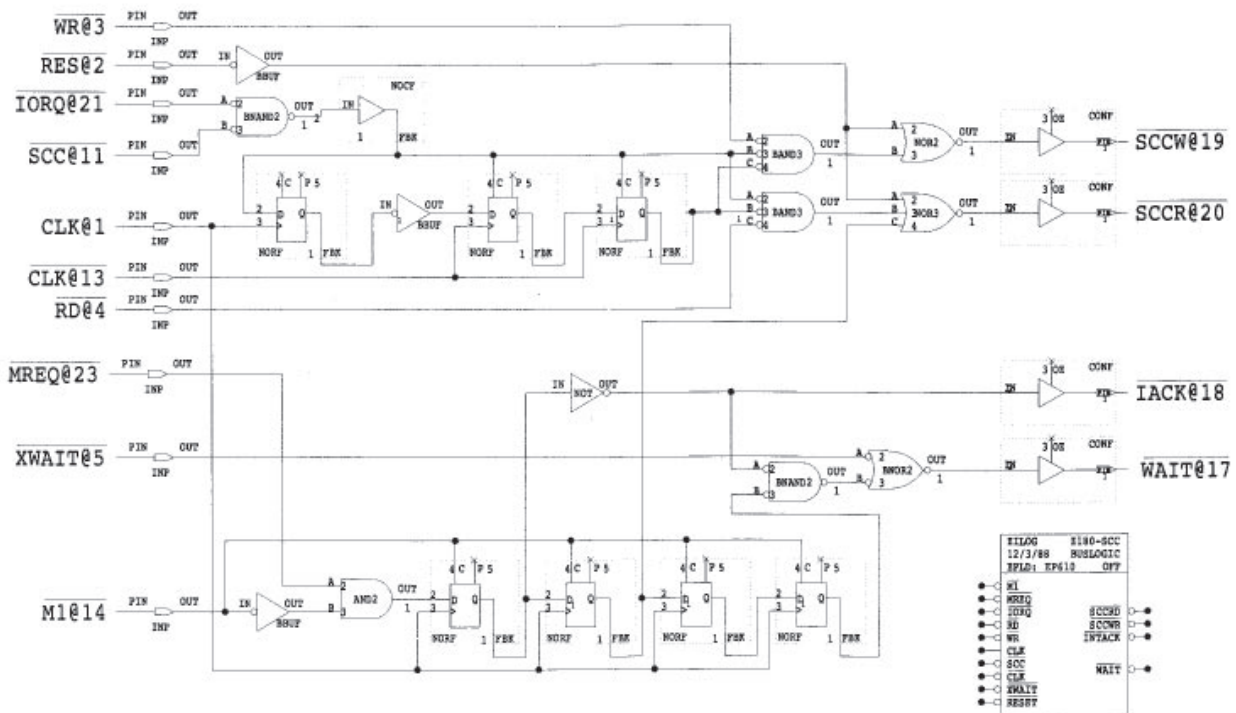


Figure 18. EPLD Circuit Implementation

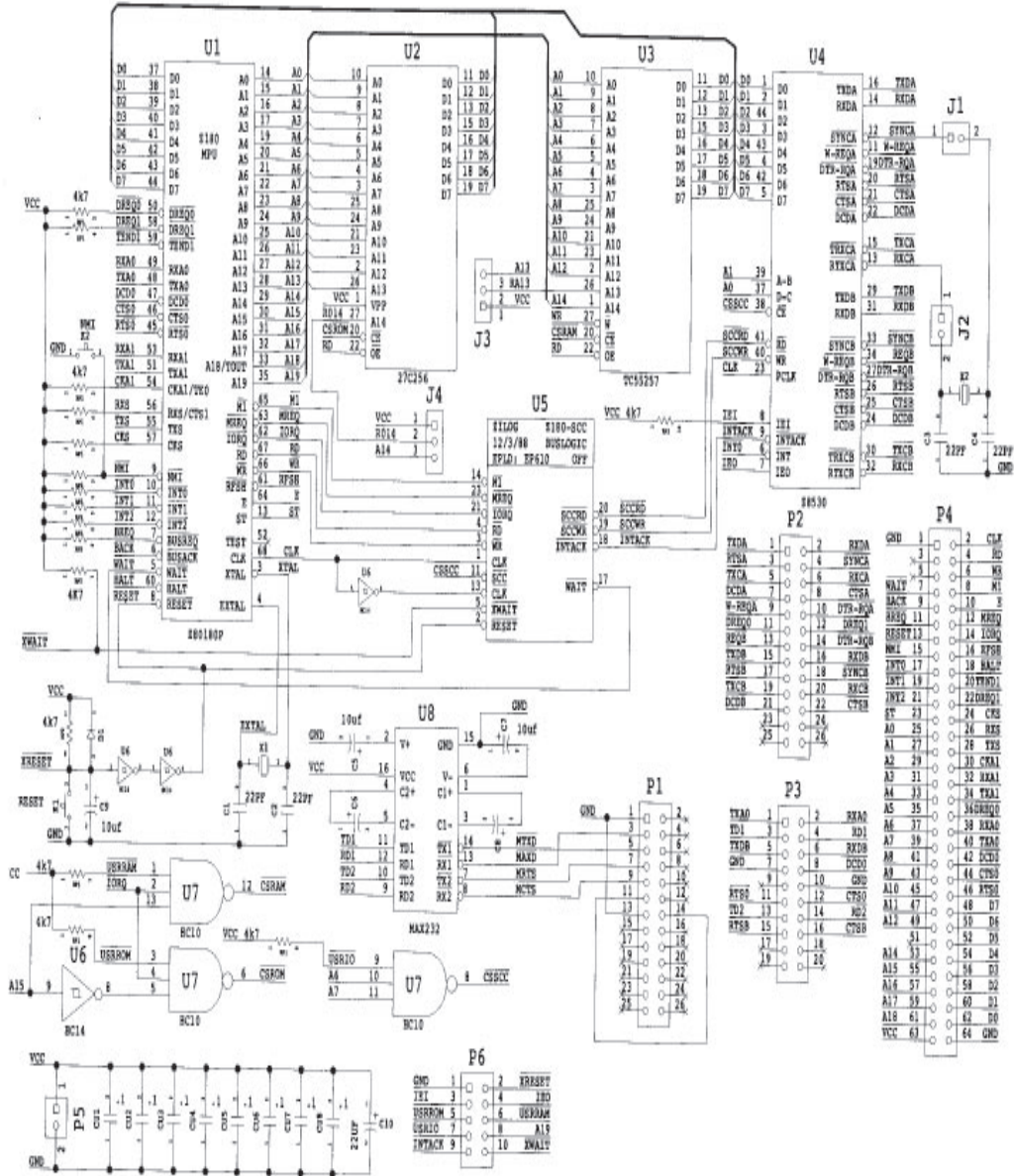


Figure 19. ELPD Circuit Implementation

System Checkout

After completion of the board (PC board or wire wrapped board, etc.), the following methods verify that the board is working.

Software Considerations

Based on the previous discussion, it is necessary to program the Z180 internal registers, as follows, before system checkout:

- Z80[®] mode of operation - Clear /M1E bit in OMCR register to zero (to provide expansion for Z80 peripherals).
- Z80 compatible mode - Clear IOC bit in OMCR register to zero.
- Put one wait state in memory cycle, and no wait state for I/O cycle DMCR register bits 7 and 6 to '1' and bits 5 and 4 to '0'.

SCC Read Cycle Proof

Read cycle checking is first because it is the simplest operation. The SCC Read cycle is checked by reading the bits in RR0. First, the SCC is hardware reset by simultaneously pulling /RD and /WR LOW (The circuit above includes the circuit for this). Then, reading out the Read Register 0 returns:

```
D7-D0 = 01xxx100b
Bit D2, D6:1
Bit D7, D1, D0:0
Bit D5: Reflects /CTS pin
Bit D4: Reflects /SYNC
Bit D3: Reflects /DCD pin
```

SCC Write Cycle Proof

Write cycle checking involves writing to a register and reading back the results to the registers which return the written value. The Time Constant registers (WR12 and WR13) and External/Status Interrupt Enable register (WR15) are on the SCC.

Interrupt Acknowledge Cycle

Checking an Interrupt Acknowledge (/INTACK) cycle consists of several steps. First, the SCC makes an Interrupt Request (/INT) to the Z180. When the processor is ready to service the interrupt, it shows an Interrupt Acknowledge (/INTACK) cycle. The SCC then puts an 8-bit vector on the bus and the Z180 uses that vector to get the correct service routine. The following test checks the simplest case.

First, load the Interrupt Vector Register (WR2) with a vector, disable the Vector Interrupt Status (VIS) and enable interrupts (IE=1, MIE=1 IEI=1). Disabling VIS guarantees only one vector on the bus. The address of the service routine corresponding to the 8-bit vector number loads the Z180 vector table, and the Z180 is under Interrupt Mode 2.

Because the SCC Interrupt Pending bit (IP) cannot be set, setting an interrupt sequence is difficult. An interrupt is generated indirectly via the CTS pin by enabling the following explanation.

Enable interrupt by /CTS (WR15, 20h), External/Status Interrupt Enable (WR1, 01h), and Master Interrupt Enable (WR9, 08h). Any change on the /CTS pin begins the interrupt sequence. The interrupt is re-enabled by Reset External/Status Interrupt (WR0, 10h) and Reset Highest IUS (WR0, 38h). A sample program of an SCC Interrupt Test is listed in [Table 12](#) on page 27. It uses the Assembler for the Z80[®] CPU. The programs in [Table 12](#) on page 27, [Table 13](#) on page 29, and [Table 14](#) on page 32 assume that the 180 is correctly initialized

Table 12. SCC Test Program – Interrupt for 180/SCC Application Board (Under Mode2 Interrupt)

```

;*      B register returns status info:
;*      Bit D0: current /cts stat
;*      D1set: /cts int received
;*
;*
.z800
;Read in Z180 register names and
*include 180macro.lib ;macro for Z180 new instructions

;SCC
Registers
scc_ad:      equ      0C3h      ;addr of scc ch a - data
scc_ac:      equ      0C2h      ;addr of scc ch a - control
scc_bd:      equ      0C1h      ;addr of scc ch b - data
scc_bc:      equ      0C0h      ;addr of scc ch b - control
scc_a:       equ      000h      ;set 0ffh to test ch a
;clear 00h to test ch b.

        if      scc_a
scc_cont:    equ      scc_ac
        else
scc_cont:    equ      scc_bc
        endif

org      09000h      ;top of user ram area

inttest:    ld      sp,top_of_sp      ;init sp
            ld      a,high sccvect and 0ffh ;init i reg
            ld      i,a
            im      2      ;set interrupt mode 2
            call   initscc      ;initialize scc
            ld      b,0      ;clear status
            ei      ;enable interrupt

wait_loop:  bit      1,b      ;check int status
            jr      z,wait_loop ;if not, loop again

wait_here:  jr      $      ;interrupt has been received
;you can set breakpoint here!

;subroutine to initialize scc registers
;initialization table format is
;register number, then followed by the data to be written
;and the register number is 0ffh, then return

initscc:   ld      hl,scctab      ;initialize scc
init0:     ld      a,(hl)      ;get register number
            cp      0ffh      ;reached at the end of table?
            ret      z      ;yes, return.
            out     (scc_cont),a ;write it
            inc     hl      ;point to next data
            ld      a,(hl)      ;get the data to be written
            out     (scc_cont),a ;write it
            inc     hl      ;point to next data
            jr      init0      ;then loop

```

Table 12. SCC Test Program – Interrupt for 180/SCC Application Board (Under Mode2 Interrupt) (Continued)

```

;external/status interrupt
service routine

ext_stat:   ld           a,10h
           out          (scc_cont),a           ;reset ext/stat int
           in           a,(scc_cont)          ;read stat
           and          00100000b           ;mask off bits other than /cts
           rra          ;shift into D0 loc
           rra
           rra
           rra
           rra
           rra
           set          1,a                   ;set interrupt flag
           ld           b,a                   ;save it
           ld           a,38h
           out          (scc_cont),a         ;reset highest ius
           ei          ;enable int
           ret          ;return from int

;initialization data table for scc
;table format - register number, then value for the register
;and ends with 0ffh - since scc doesn't have
;register 0ffh...

scctab:    db           09h                   ;select WR9
           if          scc_a
           db           10000000b           ;ch a reset
           else
           db           01000000b           ;ch b reset
           endif
           db           0eh                   ;select WR15
           db           20h                   ;only enable /cts int
           db           01h                   ;select WR1
           db           00000001b           ;enable ext/stat int
           db           10h                   ;reset ext/stat int
           db           10h                   ;twice
           db           09h                   ;select WR9
           db           08h                   ;mie, vect not incl. stat
           db           0ffh                  ;end of table

;interrupt vector table
           org          inttest + 100h
sccvect:   dw          ext_stat

           .block      100h                   ;reserve area for stack

top_of_sp:
           end

```



Table 13 lists a “macro” to enable the Z180 to use the Z80[®] Assembler, as well as register definitions.

using the Z180 onchip DMA. The SCC self loop-back test transfers data using the Z180 DMA at the highest transmission rate.

There is one good test to ensure proper function. Generate a data transfer between the Z180/SCC

Table 13. Program Example – Z180 CPU Macro Instructions

```

;*          File name - 180macro.lib
;*          Macro library for Z180 new instructions for asm800
;*
;*
;*
;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

;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

```

Table 13. Program Example – Z180 CPU Macro Instructions (Continued)

```

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

?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

out0     macro    ?p, ?r
db        11101101B
db        00000001B+(?&?r AND 7) SHL 3
db        ?p
endm

otim     macro
db        11101101B

```

Table 13. Program Example – Z180 CPU Macro Instructions (Continued)

```

                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

```

Table 14 lists a program example for the Z180/SCC DMA transfer test.

Table 14. Test Program—Z180/SCC DMA Transfer

```

;
;* Test program for 180 DMA/SCC
;
;*
;* Test 180's DMA function with SCC
;*
;* 180 dma - dma0 for scc rx data
;* dma1 for scc tx data
;* async, X1 mode, 1 stop, speed = pclk/4
;* self loop-back
;* Connect W/REQ to DREQ0 of 180
;* DTR/REQ to DREQ1 of 180
;*
;* B register returns status info:
;* Bit D0 set : Tx DMA end
;* D1 set : Rx DMA end
;* D2 set : Data doesn't match
;*
;
.z800
;
; Read in Z180 register names and
;*include 180macro.lib ;macro for Z180 new instructions
;SCC Registers

scc_ad: equ 0C3h ;addr of scc ch a - data
scc_ac: equ 0C2h ;addr of scc ch a - control
scc_bd: equ 0C1h ;addr of scc ch b - data
scc_bc: equ 0C0h ;addr of scc ch b - control
scc_a: equ 00h ;if test ch. a, set this to 0ffh
;for ch.b, set this to 00h

if scc_a
scc_cont: equ scc_ac
scc_data: equ scc_ad
else
scc_cont: equ scc_bc
scc_data: equ scc_bd
endif

length: equ 1000h ;transfer length
org 09000h ;top of user ram area

sccdma: ld sp,tx_buff ;init sp
ld a,(high z180vect) and 0ffh ;init i reg
ld i,a
ld a,00h ;init il
out0 (il),a
im 2 ;Set interrupt mode 2
call fill_mem ;initialize tx/rx buffer area
call initscc ;initialize scc

```


Table 14. Test Program–Z180/SCC DMA Transfer (Continued)

		call	initdma	
		ld	b,0	;init status
		ld	a,00h	;load 1st data to be sent
		out	(scc_data),a	
		ld	a,11001100b	;enable dmac and int from DMA0
		out0	(dstat),a	
		ld	a,05h	;select WR5
		out	(scc_cont),a	
		ld	a,01101000b	;start tx
		out	(scc_cont),a	
		ei		;wait here for completion
loop:		bit	1,b	;rx dma end?
		jr	z,loop	;not, then loop again
		push	bc	;save bc reg
		ld	bc,length	;compare tx data with rx data
		ld	de,tx_buff	
		ld	hl,rx_buff	
chkloop:	ld	a,(de)		
		cpi		
		jr	nz,bad_data	
		jp	v,good	
		inc	de	
		jr	chkloop	
bad_data:		pop	bc	;restore bc
		set	2,b	;set error flag
		jr	enddma	
good:		pop	bc	;restore bc
enddma:		jr	\$;tx/rx completed
				you can put breakpoint here
fill_mem:	l	d	hl,temp	; prepare data to be sent
		ld	bc,length	; set length
		ld	de,tx_buff	
		ld	(hl),00h	
fill_loop:		ldi		
		jp	nv,fill_00	
		dec	hl	
		inc	(hl)	
		jr	fill_loop	
fill_00:		ld	bc,length	; clear rx buffer area to zero
		ld	de,rx_buff	
		ld	(hl),00h	
fill_00l:		ldi		
		ret	nv	
		dec	hl	
		jr	fill_00l	

Table 14. Test Program–Z180/SCC DMA Transfer (Continued)

```

initscc:      ld      hl,scctab      ; initialize scc
init0:       ld      a,(hl)
             cp      0ffh
             ret      z
             out     (scc_cont),a
             inc     hl
             ld      a,(hl)
             out     (scc_cont),a
             inc     hl
             jr      init0

;initialize z180's scc
;

initdma:     ld      hl,addrtab    ;initialize DMA

             ld      c,sar0l
             ld      b,dstat - sar0l
             otimr
             ld      a,00001100b   ;dmac0 - i/o to mem++
             out0   (dmode),a
             ld      a,01001000b   ;1 mem wait, no i/o wait,
                                     ;should be EDGE for Tx DMA
                                     ;NOT level
                                     ;- because of DTR/REQ timing
             ret

txend:      ld      a,00010100b   ;isr for dma1 int-complete tx
             out0   (dstat),a
             set     0,b           ;disable dma1
                                     ;set status
             ei
             ret

rxend:      ld      a,00100000b   ;isr for dma0 int
             out0   (dstat),a
             set     1,b           ;disable dma0
                                     ;set status
             ei
             ret

;initialization data table for scc
;table format - register number, then value for the register
;and ends with 0ffh - since scc doesn't have
;register 0ffh...
scctab:     db      09h           ;select WR9
             if scc_a
             db      10000000b    ;reset ch a
             else
             db      01000000b    ;Reset Ch B
             endif
             db      04h           ;select WR4
             db      00000100b    ;async,x1,1stop,parity off

```

Table 14. Test Program–Z180/SCC DMA Transfer (Continued)

```

db      01h      ;select WR1
db      01100000b ;REQ on Rx

db      02h      ;select WR2
db      00h      ;00h as vector base

db      03h      ;select WR3
db      11000000b ;Rx 8bit/char

db      05h      ;select WR5
db      01100000b ;tx 8bit/char

db      06h      ;select WR6
db      00h      ;

db      07h      ;select WR7
db      00h      ;

db      09h      ;select WR9
db      00000001b ;stat low, vis

db      0ah      ;select WR10
db      00000000b ;set as default
db      0bh      ;select WR11
db      01010110b ;
;      0      No xtal
;      1010    TxC,RxC from BRG
;      110     TRxC = BRG output

db      0ch      ;select WR12
db      00h      ;BR TC Low

db      0dh      ;select WR12
db      00h      ;BR TC high

db      0eh      ;select WR14
db      00010110b ;
;      000     nothing about DPLL
;      1      Local loopback
;      0      No local echo
;      1      DTR/REQ is req
;      1      BRG source = PCLK
;      0      Not enabling BRG yet

db      0eh      ;select WR14
db      00010111b ;
;      000     nothing about DPLL
;      1      Local loopback
;      0      No local echo
;      1      DTR/REQ is REQ
;      1      BRG source = PCLK
;      1      Enable BRG

db      03h      ;select WR3
db      11000001b ;rx enable

```

Table 14. Test Program–Z180/SCC DMA Transfer (Continued)

```

        db      01h          ;select WR1
        db      11100000b   ;enable DMA

        db      0fh          ;select WR15
        db      00000000b   ;don't use any of ext/stat int
        db      10h          ;reset ext/stat twice
        db      10h

        db      01h          ;select WR1
        db      11100000b   ;no int

        db      09h          ;select WR9
        db      00001001b   ;enable int

        db      0ffh        ;end of table

;source/dist addr table for Z180's dma
addrtab:
        db      scc_data    ;dmac0 source
        db      00h
        db      00h

        dw      rx_buff     ;dmac0 dist
        db      00h

        dw      length      ;byte count

        dw      tx_buff+1   ;mar
        db      00h

        db      scc_data    ;iar
        db      00h

        db      00h        ;dummy!
        dw      length-1    ;byte count

;interrupt vector table

z180vect:
        org      sccdma + 200h
        .block  2          ;180 int1 vect 0000
        .block  2          ;180 int2 vect 0010
        .block  2          ;180 prt0 vect 00100
        .block  2          ;180 prt1 vect 00110
        dw      rxend      ;180 dmac0 vect 01000
        dw      txend      ;180 dmac1 vect 01010
        .block  2          ;180 csi/o vect 01100
        .block  2          ;180 asci0 vect 01110
        .block  2          ;180 asci1 vect 10000

        org      sccdma + 1000h
tx_buff:  .block  length
rx_buff:  .block  length
temp:     .block  1

end

```

First, this program (Table 14 on page 32) initializes the SCC by Async, X1 mode, 8-bit 1 stop, Non-parity. Tx and Rx clock from BRG, and BRG set to PCLK/4 Self Loopback. Then, it initializes 4 KB of memory with a repeating pattern beginning with 00h and increases by one to FFh (uses this as Tx buffer area). Also, it begins another 4 KB of memory as a Rx buffer with all zeros. After starting, DMA initialization follows.

DMAC0—For Rx data transfer I/O to Mem, Source address- fixed, Destination address-increasing. Edge sense mode Interrupt on end of transfer.

DMAC1—For Tx data transfer Mem to I/O, Source address-increasing, Destination address - fixed. Edge sense mode Interrupt on end of transfer.

Now, start sending with DMA.

On completion of the transfer, the Z180 DMAC1 generates an interrupt. Then, wait for the interrupt from DMAC0 which shows an end of receive. Now, compare received data with sent data. If the transfer was successful (source data matched with destination), 00h is left in the accumulator. If not successful, 0FFh is left in the accumulator.

This program example specifies a way to initialize the SCC and the Z180 DMA.

Summary

This Application Note describes only one example of implementation, but gives you an idea of how to design the system using the Z180 and SCC. For further design assistance, a completed board together with the Debug/Monitor program and the listed sample program are available. For more information, contact your local Zilog sales office.



Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

©2008 by Zilog, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z80 is a registered trademark of Zilog, Inc. All other product or service names are the property of their respective owners.