Chapter 2
Am9513A/Am9513 Interfacing

Am9513 — CPU INTERFACING

The Am9513 is designed to interface easily to both the Am8080A/8085A 8-bit family of CPUs and to the AmZ8000 16-bit family of CPUs. Master Mode register bit MM13 allows the user to program the Am9513 data bus for either an 8- or 16-bit width, allowing the Am9513's data bus to be tailored to match that of the host CPU.

Figure 2-1 shows an interface between the Am9513 and an Am8085A CPU. The Am9513 is configured to appear in the CPU's I/O space; connecting the IO/$\overline{M}$ output of the CPU to the $\overline{G2A}$ input of the decoder and tying G1 high will memory-map the Am9513. In the configuration shown, the Am9513 operates with an 8-bit data bus. Master Mode register bit MM13 should be 0 and data bus pins DB13-DB15 should be tied high as shown in the diagram.

Figure 2-2 shows a suggested connection diagram between the Am9513 and an AmZ8001* or AmZ8002* CPU. In this diagram the Am9513 appears in both Regular and Special I/O space, by virtue of the decoding of status lines ST1-ST3. Status line ST0 should be decoded also if it is necessary to separate the Regular and Special I/O spaces. The AmZ8136 is a latched decoder which stores the address information on the rising edge of $\overline{AS}$, providing the Am9513 with a stable $\overline{CS}$ for the duration of the transfer. The Am25LS158 multiplexer generates $\overline{RD}$ and $\overline{WR}$ from the CPU's $\overline{DS}$ and R/$\overline{W}$ lines. For maximum data bandwidth between the CPU and the Am9513, Master Mode register bit MM13 should be set to 1 to configure the Am9513 for a 16-bit data bus width. This can be accomplished by writing command opcode FFEF (hex) to the Am9513 following each reset and power-up.

CLOCK GENERATION

An internal oscillator is provided on the Am9513 for generation of timing frequencies to drive the source inputs for the five counters and the source for the FOUT pin. Note that a clock signal is not required for reads and writes to the Am9513. In applications which do not use the internal oscillator, the X2 input should be tied either High or Low to prevent accumulation of static charge. The X1 output is driven by an inverter contained in the Am9513 and accordingly, X1 should be left floating to avoid damaging the inverter's output stage.

Applications using the internal oscillator can drive the X1 and X2 inputs with an RC network, an external non-TTL level squarewave or a crystal. Figure 2-3 shows the recommended methods of connecting different frequency sources to the internal oscillator's input.

A crystal provides a highly accurate frequency source at moderate cost, and will usually be the preferred method of operation. The Am9513 is designed to use a crystal in parallel-resonant fundamental mode operation using the connection diagram shown in Figure 2-3a. Most series-resonant crystals can also be used, but the oscillator frequency will be different by up to a few percent from the series resonant crystal's rated frequency. Two ceramic capacitors should be connected between X1 and X2 to ground to ensure proper crystal loading. (The crystal loading is the capacitance the crystal should be driving to ensure on-frequency operation and reliable oscillator startup). Although the crystal sees the capacitors on X1 and X2 in series, and neglects the ground connection in the center, the use of two capacitors stabilizes the bias on the crystal by referencing it to ground and provides superior performance over the one capacitor equivalent circuit. Ceramic capacitors are the best type for this application because of their stability over time and temperature and their superior high frequency characteristics.

An RC network provides a very low cost frequency source but may exhibit large frequency variations over recommended power supply and temperature ranges, negating much of the precision available in the Am9513's counters. The RC connection is shown in Figure 2-3b. Note that although there is an internal resistor between X1 and X2, because this internal resistance is quite high, an external resistor should always be used in the RC operating configurations.

*Z8001 and Z8002 are trademarks of Zilog, Inc.

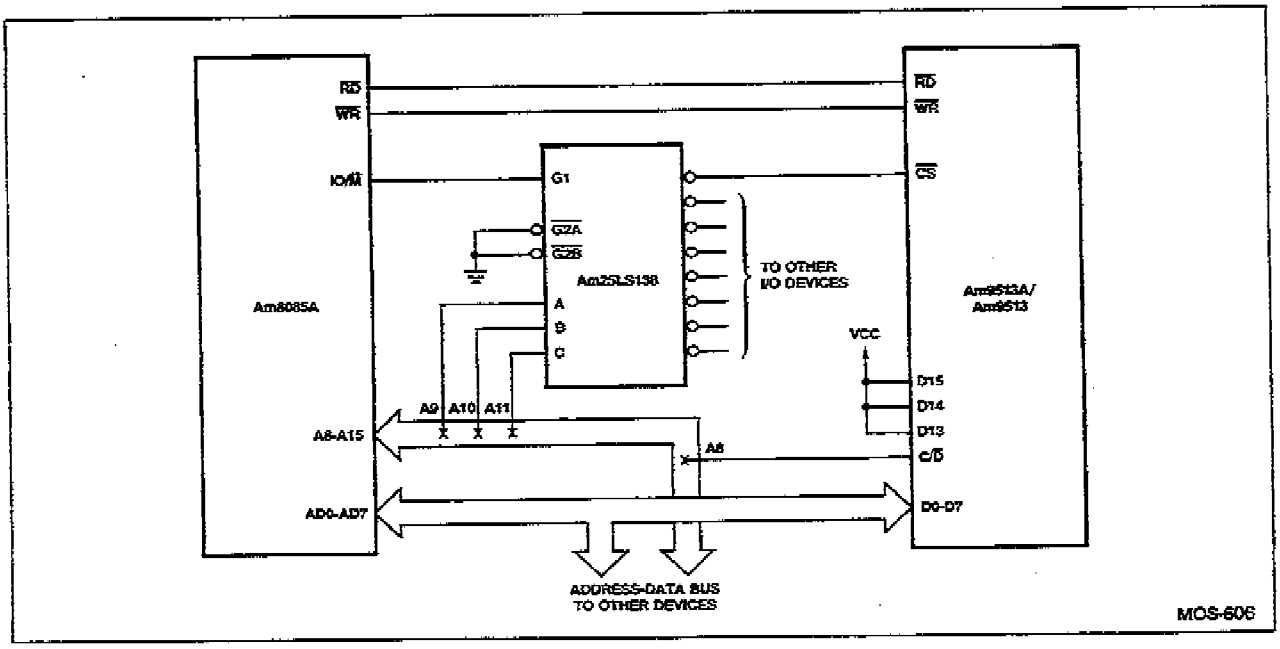

Figure 2-1. Am9513 — Am8085 Interfacing

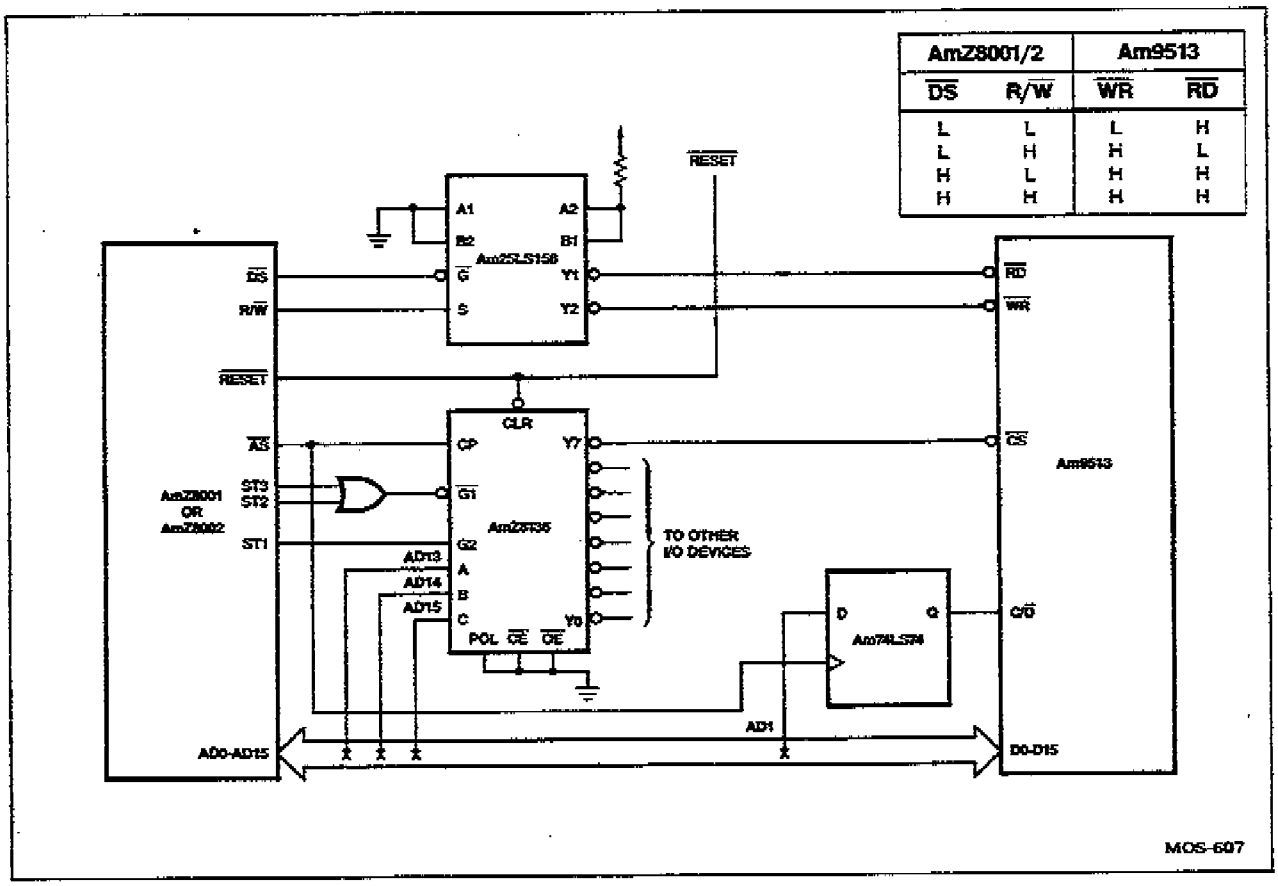

Figure 2-2. AmZ8001/8002 — Am9513 Interface

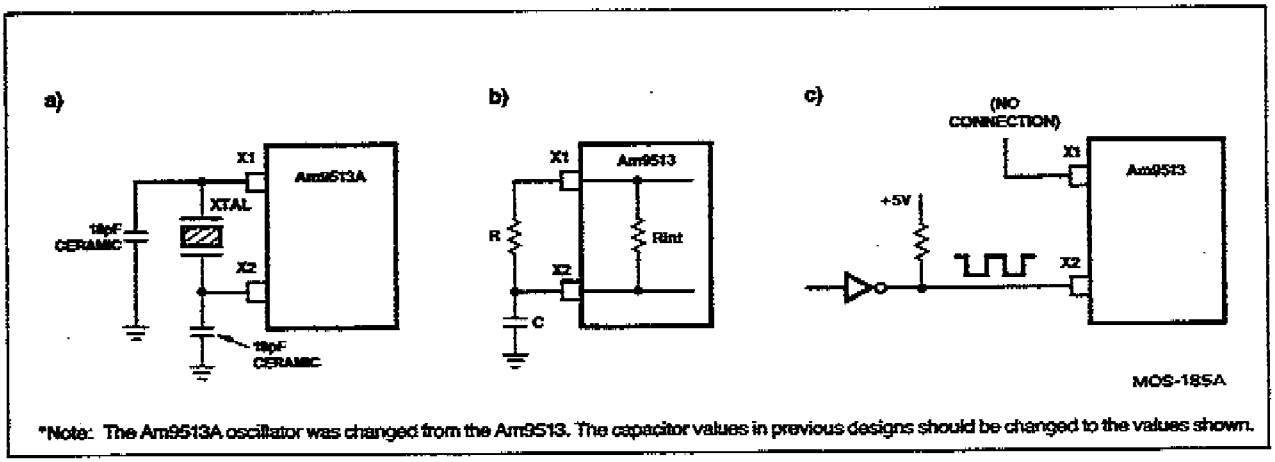

Figure 2-3. Driving the X1 and X2 Inputs

The Am9513 internal oscillator can also be driven by an external signal as shown in Figure 2-3c. The Am9513 Electrical Specification should be consulted for the voltage levels required on the X2 input to guarantee proper oscillator operation in this configuration. Most circuits can generate this non-TTL level using a pull-up resistor and a 74LS04 inverter, or equivalent. In some cases a pull-up resistor can be used to increase the high level output voltage of an MOS device, such as on the Am8085A CLK output, without the need for a bipolar buffer. Care must be taken in this bufferless circuit to choose a pull-up resistor low enough to meet the Am9513's high level voltage needs without choosing a resistor value so low that the Am8085A has to sink excessively large currents when pulling the CLK signal low.

REGISTER ACCESS

Information Transfer Protocols

The control signal configurations for all information transfers on the Am9513 data bus are summarized in Figure 2-4. The interface control logic assumes these conventions:

1. $\overline{RD}$ and $\overline{WR}$ are never active at the same time.
2. $\overline{RD}$, $\overline{WR}$ and C/$\overline{D}$ are ignored unless $\overline{CS}$ is Low.

The following discussion provides software oriented examples of Am9513 register accesses. Software examples are given for an Am8085 CPU with an 8-bit Am9513 data bus interface and for an AmZ8002 CPU with a 16-bit Am9513 data bus interface. The descriptions assume that the Am9513 Control port (CMDPRT) is located at address 12 (hex) and the Am9513 Data port (DATAPRT) is located at address 10 (hex). Later sections of this document present complete software listings for representative Am9513 applications.

Software Initialization

Figure 2-5 shows a Z8000 Software Initialization Sequence for the 9513. It is important to note the "DUMMY" LOAD COUNTER COMMAND; this insures proper operation of the part. The 16-bit mode command is not used for 8-bit CPUs. The sequence then is to Reset the device; Load all counters; Command 16-bit mode; set Data Pointer to the Master Mode Register; set Master Mode Register to desired value; set Data Pointer to counter #1 Mode Register and initialize counters to desired mode of operation. Note $\overline{CS}$ must be high during power-up or the internal reset circuitry will not function correctly. This will result in part ignoring all commands issued to it except software reset.

Command Initiation

Commands are issued to the Am9513 by writing the appropriate command code to the Am9513 Control Port. Figure 2-6 shows an example of command initiation, in this case opcode

Signal Configuration				Data Bus Operation
$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
0	0	0	1	Transfer contents of register addressed by Data Pointer to the data bus.
0	0	1	0	Transfer contents of data bus to data register addressed by Data Pointer.
0	1	0	1	Transfer contents of Status register to data bus.
0	1	1	0	Transfer contents of data bus into Command register.
X	X	1	1	No transfer.
1	X	X	X	No transfer.
X	X	0	0	Illegal Condition.

Figure 2-4. Data Bus Transfers

AD (hex), which saves the contents of Counters 1, 3 and 4 in their associated Hold registers. In both the Am8080A/8085A and AmZ8002 coding examples, the command is loaded into an internal CPU register and output to the appropriate port. Note that in the AmZ8002 case since a 16-bit data bus interface is assumed the upper byte of data output to the Command port must be FF (hex).

The procedure for executing a command is as follows:

1. Establish the appropriate command on the DB0-DB7 lines. Figure 1-21 lists the command codes. When using the Am9513 in 16-bit mode, data bus lines DB8-DB15 should be set high during the write operation. In 8-bit data bus mode, DB13-DB15 should be set high during the write operation.

```
MACRO8000:          Version 2.0    9/19/80
MACZ   9513INIT S,P,L,O,W,D
INIT


0000                              %THIS IS A SAMPLE INITIALIZATION SEQUENCE
0000                              %FOR THE AM9513 COUNTER TIMER
0000
0000                        MODULE 'INIT';
0000
0000                        CONST   CMDPRT=12H,
0000                                DATAPRT=10H;
0000
0000   2101 FFFF            INIT:   LD      R1,#FFFF;
0004   3B16 0012                    OUT     CMDPRT,R1;      %SEND RESET
0008   2101 FF5F                    LD      R1,#FF5F;
000C   3B16 0012                    OUT     CMDPRT,R1;      %LOAD ALL COUNTERS
0010   2101 FFEF                    LD      R1,#FFEF;
0014   3B16 0012                    OUT     CMDPRT,R1;      %COMMAND 16 BIT MODE
0018   2101 FF17                    LD      R1,#FF17;
001C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO MASTER MODE REG
0020   2101 2CEF                    LD      R1,#2CEF;
0024   3B16 0010                    OUT     DATAPRT,R1;     %MASTER MODE SETTING
0028   2101 FF01                    LD      R1,#FF01;
002C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO CNTR 1 MODE REG
0030
0030                              END.
```

Figure 2-5. Am9513 Initialization

```
0100                        ORG     100H
                            ;
                            ;       AM9513 PORT ADDRESSES
                            ;
0012 =          CMDPRT      EQU     012H
0010 =          DATAPRT     EQU     010H
                            ;
                            ;       AM9513 COMMAND INITIATION
                            ;
0100 3EAB                   MVI     A,0ADH  ;SAVE CTRS. 1 3 & 4
0102 D312                   OUT     CMDPRT
                            ;
                            ;
                            PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1    Page 1


0000                            PROGRAM AM9513_EXAMPLES;
0000                            ORIGIN  0H;
0000                            %
0000                            %
0000                            %       AM9513 PORT ADDRESSES
0000                            %
0000                    CONST   CMDPRT=12H,
0000                            DATAPRT=10H;
0000                            %
0000                    AM9513_EXAMPLES:
0000                            %
0000                            %       AM9513 COMMAND INITIATION
0000                            %
0000    2102 FFAD               LD      R2,0FFADH;      %SAVE CTRS. 1 3 & 4
0004    3B26 0012               OUT     CMDPRT,R2;
0008                            %
0008                            %
0008                            EJECT;
```

b) AmZ8000 Code

Figure 2-6. Command Initiation Software

2. Establish a High on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on the $\overline{WR}$ input.
5. Sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, drive $\overline{WR}$ high, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
6. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).

A new read or write operation to the Am9513 should not be performed until the write recovery time is met (see Timing Diagram in Electrical Specification.)

Setting the Data Pointer Register

The Data Pointer register selects which internal Am9513 register is to be accessed through the Data port. Setting the Data Pointer register automatically sets the Byte Pointer to 1, indicating a least significant byte is expected for 8-bit data bus interfacing. If Master Mode register bit MM14 = 0, the Data Pointer will automatically sequence through one of the cycles shown in Figure 1-11 after reading or writing each register, allowing sequential access to internal registers. If MM14 = 1, auto-sequencing is disabled and a single internal register can be repetitively accessed without reloading the Data Pointer. For convenience, bit MM14 can be set or cleared by software command.

The Pointer is set as follows:

1. Using Figures 1-9 and 1-10, select the appropriate Data Pointer Group and Element codes for the register to be accessed. Note that two codes are provided for the Hold registers, to accommodate both the Hold Cycle and Element Cycle autosequencing modes shown in Figure 1-11. If autosequencing is disabled, either Hold code may be used.
2. Using the "Writing to the Command Register" procedure given above, write the appropriate "Load Data Pointer" command to the Command register.

```
                        ;
                INTSR:  ;           ;INTERRUPT SERVICE ROUTINE
                        ;
                        ;           DISABLE INTERRUPTS
                        ;
0104 F3                 DI
                        ;
                        ;           SET DATA POINTER TO COUNTER 1 HOLD REG
                        ;
0105 3E19               MVI         A,019H
0107 D312               OUT         CMDPRT
                        ;
                        ;           ENABLE AUTO-SEQUENCING
                        ;
0109 3EE0               MVI         A,0E0H
010B D312               OUT         CMDPRT
                        ;
                        ;           CODE TO ACCESS REGISTERS
                        ;
                        ;           DISABLE AUTO-SEQUENCING
                        ;
010D 3EE8               MVI         A,0E8H
010F D312               OUT         CMDPRT
                        ;
                        ;           ENABLE INTERRUPTS AND RETURN
                        ;
0111 FB                 EI
0112 C9                 RET
                        ;
                        ;
                        PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 2

0008                            Z
0008                    INTSR:  Z           INTERRUPT SERVICE ROUTINE
0008                            Z
0008    7C00                    DI          NVI,VI;
000A                            Z
000A                            Z           SET DATA POINTER TO COUNTER 1 HOLD REG.
000A                            Z
000A    2102 FF19               LD          R2,0FF19H;
000E    3B26 0012               OUT         CMDPRT,R2;
0012                            Z
0012                            Z           ENABLE AUTO-SEQUENCING
0012                            Z
0012    2102 FFE0               LD          R2,0FFE0H;
0016    3B26 0012               OUT         CMDPRT,R2;
001A                            Z
001A                            Z           CODE TO ACCESS REGISTERS
001A                            Z
001A                            Z           DISABLE AUTO-SEQUENCING
001A                            Z
001A    2102 FFE8               LD          R2,0FFE8H;
001E    3B26 0012               OUT         CMDPRT,R2;
0022                            Z
0022                            Z           ENABLE INTERRUPTS AND RETURN
0022                            Z
0022    7C04                    EI          NVI,VI;
0024    7B00                    IRET;
0026                            Z
0026                            Z
0026                            EJECT;
```

b) AmZ8000 Code

Figure 2-7. Am9513 Interrupt Service Routine

In many systems the Am9513 counters will be serviced by interrupt routines. In such systems, it is important that the Am9513 service routines not be interrupted by another Am9513 service routine while register accesses are occurring. Consider, for example, an interrupt service routine which reads the Hold register value in the Counter 1 logic group. This routine will set the Data Pointer register and read the Hold register value. Consider the sequence of events which would occur if, after this routine set the Data Pointer registers, but before it read the Hold register, it was interrupted by a second Am9513 interrupt routine. This second routine might, for example, read the Counter 3 logic group Hold register value. When this second interrupt routine finishes, it returns control to the last half of the first interrupt routine. Because the second routine has changed the Data Pointer register, the first routine will not read the Hold register 1 contents. As can be seen from the above scenario, the sequence of operations of setting the Data Pointer register and accessing internal register locations must not be interrupted by another Am9513 service routine.

One way of ensuring that this restriction is met is to disable interrupts before setting the Data Pointer and not enabling interrupts until the register accesses are performed. Note that when auto-sequencing is used, interrupts should not be enabled until all registers have been accessed. An alternative method of meeting this restriction is to use software semaphores to prevent nesting of Am9513 service routines.

Figure 2-7 shows sample interrupt service routines which set the Data Pointer register to point to Counter 1's Hold register and enable Hold cycle auto-sequencing by clearing MM14. In the AmZ8002 case, a 16-bit data bus interface is assumed, requiring that the upper command byte be FF (hex). In the coding examples given interrupts are disabled and enabled by software command. Since the AmZ8002 architecture loads a new Flag and Control Word (FCW) when responding to an interrupt request, the FCW loaded can disable further interrupts. This provides an alternative interrupt inhibiting mechanism for AmZ8002 systems and may be used in lieu of the software commands.

Reading the Status Register

The Am9513 Status register can be read either through the Control port or through the Data port. Figure 2-8 shows sample programs reading the Status register contents through the Control port into the accumulator (A register) of an Am8080A/8085A system or the R0 register of an AmZ8002 system. It is assumed that the AmZ8002 system has a 16-bit data bus; since the status register is only eight bits wide, the high byte of register R0 is undefined.

The procedure for reading the Status register through the Control port is given in the following.

1. Establish a High on the C/$\overline{D}$ input.
2. Establish a Low on the $\overline{CS}$ input.
3. After the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram) make $\overline{RD}$ Low.
4. Sometime after $\overline{RD}$ goes Low, the Status register contents will appear on the data bus. These lines will contain the information as long as $\overline{RD}$ is Low. If the state of an OUT pin changes while $\overline{RD}$ is Low, this will be reflected by a change in the information on the data bus.
5. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
6. $\overline{CS}$ and C/$\overline{D}$ can change after meeting the appropriate hold time requirements (see Timing Diagram).

A new read or write operation to the Am9513 should not be attempted until the read recovery time is met (see Timing Diagram in Electrical Specification).

```
                                ;
                                ;       CODE TO READ STATUS REGISTER
                                ;
          0113 DB12             IN      CMDPRT
                                ;
                                ;
                                PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 3

0026                            %
0026                            %       CODE TO READ FROM STATUS REGISTER
0026                            %
0026    3B04 0012               IN      R0,CMDPRT;
002A                            %
002A                            %
002A                            EJECT;
```

b) AmZ8000 Code

Figure 2-8. Reading the Status Register

Reading From the Data Port

The registers which can be read from the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2, the Master Mode register and the Status register. The Status register can also be read from the Control port. Reading the Status register with a 16-bit data bus interface will return undefined information on DB8-DB15.

The procedure for reading these registers is as follows:

1. Prior to performing the actual read operation, the Data Pointer should be set to point to the register to be read, as outlined in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data Pointer each time. Special care must be taken to reset the Data Pointer after issuing a command other than "Load Data Pointer" to the Am9513 or when operating a counter in modes N, O, Q or R. See the "Prefetch Circuit" section of this document for elaboration.
2. Establish a Low on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on $\overline{RD}$ after waiting for the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram).
5. Sometime after $\overline{RD}$ goes Low, the register contents will appear on the data bus. In both 8- and 16-bit bus modes the low register byte will appear on DB0-DB7. In addition, in 16-bit bus mode, the upper register byte will appear on the DB8-DB15. For 8-bit bus mode, pins DB8-DB15 are not driven by the Am9513. This information will remain stable as long as $\overline{RD}$ is Low. If the register value is changed during the read, the change will not be reflected by a change in the data being read, for the reasons outlined in the "Prefetch Circuit" section of this document.
6. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
7. $\overline{CS}$ and C/$\overline{D}$ can change after meeting appropriate hold time requirements (see Timing Diagram).
8. After waiting the minimum read recovery time (see Timing Diagram), a new read or write operation can be started. For 8-bit bus mode, steps 2 through 7 should be repeated to read out the high register byte on DB0-DB7. (If the Status register is being read in 8-bit mode, the two reads will return the Status register each time. In 16-bit mode, reads from the Status register return undefined data on DB8-DB15.) The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" register section, the Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to Data port registers.

Figure 2-9 shows sample programs for reading a Data port register. The Am8080A/8085A code reads the data in two byte reads (low byte first) and assembles it into the HL register pair. The AmZ8002 program assumes that a 16-bit data interface is being used and reads the data into register R0 in a single word read. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                        ;
                        ;       CODE TO READ FROM DATA PORT REG.
                        ;
0115 DB10               IN      DATAPRT
0117 6F                 MOV     L,A
0118 DB10               IN      DATAPRT
011A 67                 MOV     H,A
                        ;
                        ;
                        PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1    Page 4

002A                            Z
002A                            Z       CODE TO READ FROM DATA PORT REG.
002A                            Z
002A    3B24 0010               IN      R2,DATAPRT;
002E                            Z
002E                            Z
002E                            EJECT;
```

b) AmZ8000 Code

Figure 2-9. Reading Through the Data Port

Writing to the Data Port

The registers which can be written to through the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2 and the Master Mode register. The procedure for writing to these registers is as follows:

1. Prior to performing the actual write operation, the Data Pointer should be set to point to the register to be written to, as outlined above in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data pointer each time.
2. Establish the appropriate data on the DB0-DB7 lines (8-bit bus mode) or DB0-DB15 (16-bit bus mode). When using the 8-bit bus mode, data bus lines DB13-DB15 should be set High during the write operation and DB0-DB7 should be set to the lower data byte for the first write and to the upper data byte for the second write.
3. Establish a Low on the C/$\overline{D}$ input.
4. Establish a Low on the $\overline{CS}$ input.
5. Establish a Low on the $\overline{WR}$ input.
6. Drive $\overline{WR}$ High sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
7. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).
8. After meeting the write recovery time (see Timing Diagram) a new read or write operation can be performed. For the 8-bit bus mode, steps 2 through 7 should be repeated, this time placing the high data byte on pins DB0-DB7. The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" section, Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to the Data port registers.

Figure 2-10 shows sample programs for writing a 16-bit value to a Data port register. The Am8080A/8085A code loads the register by making two byte transfers (low byte first) to the Am9513 Data port. A 16-bit data bus interface is assumed for the AmZ8002 coding example; accordingly, a single word transfer can be used to load a register. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                              ;
                              ;       CODE TO WRITE TO DATA PORT REG.
                              ;
 011B 7D                      MOV     A,L
 011C D310                    OUT     DATAPRT
 011E 7C                      MOV     A,H
 011F D310                    OUT     DATAPRT
                              ;
                              ;
                              ;
 0121                         END

A>
```

a) 8080 Code

```
AM9513_EXAMPLES                MACRO8000 AmZ8000 Assembler   1.0.1     Page 5

002E                           !
002E                           !       CODE TO WRITE TO DATA PRT REG.
002E                           !
002E   3B26 0010               OUT     DATAPRT,R2;
0032                           !
0032                           !
0032                           !
0032                           END.
```

b) AmZ8000 Code

Figure 2-10. Writing Through the Data Port