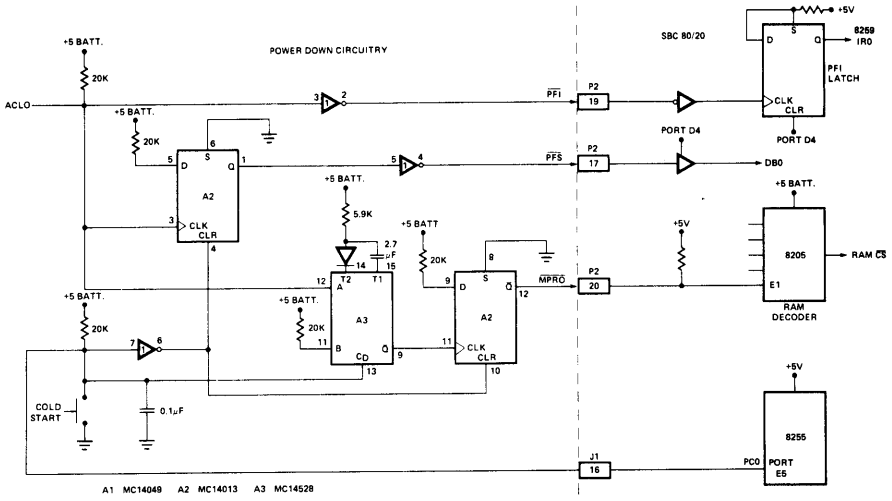


Figure 26. Power Down Circuit - SBC 80/20 Interface

The fully nested mode for the 8259A is used in its initial state to ensure the IR0 always has the highest priority. The remaining IR inputs can be used for any other purpose in the system. The only constraint is that the service routines must enable interrupts as early as possible. Obviously, this is to ensure that the power-down interrupt does not have to wait for service. If a rotating priority scheme is desired, another 8259A could be added as a slave and be programmed to operate in a rotating mode. The master would remain in the initial state of the fully nested mode so that the IR0 still remains the highest priority input.

The software to support the power-down circuitry is shown in Figure 27. The flow for each label will be discussed.

After any system reset, the processor starts execution at location 0000H (START). The $\overline{\text{PFS}}$ status is read and execution is transferred to CSTART if $\overline{\text{PFS}}$ indicates a cold start (i.e., someone is depressing the cold start switch) or WSTART if a warm start is indicated ($\overline{\text{PFS}}$ LOW). CSTART is the start of the user's program. The Stack Pointers (SP) and device initialization were included just to remind the reader that these must occur. The first EI instruction must appear after the 8259A has received its initialization sequence. The 8259A (and other devices) are initialized in the INIT subroutine.

When a power failure occurs, execution is vectored by the 8259A to REGSAV by way of the jump table at JSTART. The pre-power-down program counter is placed on the stack. REGSAV saves the processor registers and flags in the usual manner by pushing them onto the stack. Other items, such as output port status, programmable peripheral states, etc., are pushed onto the stack at this time. The Stack Pointer (SP) could be pushed onto the stack by way of the register pair HL but the top of the stack can exist anywhere in memory and there is no way then of knowing where that is when in the power-up routine. Thus, the SP is saved at a dedicated location in RAM. It isn't really necessary to send an EOI command to the 8259A in REGSAV since power will be removed from the 8259A, but one is included for completeness. The final instruction before actually losing power is a HALT. This minimizes somewhat spurious transitions on the various busses and lets the processor die gracefully.

On reset, when a warm start is detected, execution is transferred to WSTART. WSTART activates $\overline{\text{PFSR}}$ by way of the 8255 (all outputs go low then the 8255 is initialized). In the power-down circuitry, $\overline{\text{PFSR}}$ clears the PFS latch and removes the $\overline{\text{MPRO}}$ signal which then allows access to the RAM. WSTART also clears the PFI latch which arms the 8259A IR0 input. Then the 8259A is re-initialized along with any other devices. The SP is retrieved from RAM and the processor registers and flags are restored by popping them off the stack. Interrupts are then enabled. Now the power-down program counter is on top of the stack, so executing a RETurn instruction transfers the processor to exactly where it left off before the power failure.

Aside from illustrating the usefulness of the 8259A (and the SBC-80/20) in implementing a power failure protected microcomputer system, this application should also point out a way of preserving the processor status when using interrupts.

```
LOC  OBJ          SEQ      SOURCE STATEMENT

                         ;
                         ;
                         ;POWER DOWN AND RESTART FOR THE SBC 80/20
                         ;
                         ;
                         ;SYSTEM EQUATES
000A                     PT59A    EQU     0DAH             ;8259 PORT WITH A0=0
00DB                     PT59B    EQU     0DBH             ;8259 PORT WITH A0=1
00E7                     PPI1CT   EQU     0E7H             ;8255 #1 CONTROL PORT
00E6                     PPI1C    EQU     0E6H             ;8255 #1 PORT C
3800                     SPSAVE   EQU     3800H            ;SP STORAGE IN RAM
0001                     JPT      EQU     01H              ;MSB OF 8259 JUMP TABLE
                         ;
                         ;
                         ;STARTING POINT AFTER SYSTEM RESET
                         ;
                         ;
0000                              ORG     0H
0000 DBD4                START:   IN      0D4H             ;READ PFS/ STATUS
0002 1F                           RAR                      ;PFS/ ON DB0, PUT IN CARRY
0003 DA2001                       JC      CSTART           ;PFS/=1, THEN COLD START
                         ;
                         ;
                         ;WSTART LOCATION, PFS/=0, THEN WARM START
                         ;
                         ;
0006 3E80                WSTART:  MVI     A,80H            ;SET 8255 #1 TO OUTPUT MODE
0008 D3E7                         OUT     PPI1CT           ;8255 CONTROL PORT, PFSR/ GOES LOW
                                  ;
                                  ;OUTPUT COMMAND MAKES PFSR/ GO LOW WHICH REMOVES MPRO/ AND
                                  ;CLEARS PFS LATCH
                                  ;
000A 3E01                         MVI     A,01H            ;RETURN PFSR/ HIGH
000C D3E6                         OUT     PPI1C            ;8255 #1 PORT C
000E D3D4                         OUT     0D4H             ;RESET PFI LATCH
0010 CD1D00                       CALL    INIT             ;GO INITIALIZE EVERYTHING
0013 2A0038                       LHLD    SPSAVE           ;RETRIVE SP FROM RAM
0016 F9                           SPHL                     ;PUT BACK INTO SP
0017 C1                           POP     B                ;RESTORE BC
0018 D1                           POP     D                ;RESTORE DE
0019 E1                           POP     H                ;RESTORE HL
001A F1                           POP     PSW              ;RESTORE A PLUS FLAGS
001B FB                           EI                       ;ENABLE INTERRUPTS
001C C9                           RET                      ;PRE-POWER-DOWN PC ON TOP OF STACK
                                                           ;RETURN TO IT
                         ;
                         ;
                         ;INITIALIZATION ROUTINE, AT LEAST DO 8259 BUT OTHERS CAN BE ADDED
                         ;
                         ;
001D 3E16                INIT:    MVI     A,16H            ;F=1,S=1,A7-A5=0 ICW1
001F D3DA                         OUT     PT59A            ;8259 PORT WITH A0=0
0021 3E01                         MVI     A,JPT            ;MSB OF JUMP TABLE ICW2
0023 D3DB                         OUT     PT59B            ;8259 PORT WITH A0=1
                                  ;
                                  ;ADD ANY OTHER INITIALIZATIONS HERE
                                  ;
0025 C9                           RET                      ;RETURN
                         ;------                                    ----------
                         ;
                         ;POWER DOWN ROUTINE TO SAVE REGISTERS AND STACK
                         ;
                         ;
0026 F5                  REGSAV:  PUSH    PSW              ;SAVE A PLUS FLAGS
0027 E5                           PUSH    H                ;SAVE HL
0028 D5                           PUSH    D                ;SAVE DE
0029 C5                           PUSH    B                ;SAVE BC
002A 210000                       LXI     H,0000H          ;HL SET TO GET SP
002D 39                           DAD     SP               ;SP NOW IN HL
002E 220038                       SHLD    SPSAVE           ;SAVE SP IN RAM
                                  ;
                                  ;EOI NOT REALLY NEEDED BUT INCLUDED FOR COMPLETENESS
                                  ;
0031 3E20                         MVI     A,20H            ;NON-SPECIFIC EOI
0033 D3DA                         OUT     PT59A            ;8259 PORT WITH A0=0
0035 76                           HLT                      ;HALT - GO DOWN GRACEFULLY
                         ;
                         ;
                         ;8259 JUMP TABLE, ONLY IR0 IS USED, OTHERS DIRECTED TO RAM
                         ;

0100                              ORG     100H
0100 C32600              JSTART:  JMP     REGSAV           ;IR0
0103 00                           NOP
0104 C31038                       JMP     3810H            ;IR1
0107 00                           NOP
0108 C32038                       JMP     3820H            ;IR2
010B 00                           NOP
010C C33038                       JMP     3830H            ;IR3
010F 00                           NOP
0110 C34038                       JMP     3840H            ;IR4
0113 00                           NOP
0114 C35038                       JMP     3850H            ;IR5
0117 00                           NOP
0118 C36038                       JMP     3860H            ;IR6
011B 00                           NOP
011C C37038                       JMP     3870H            ;IR7
011F 00                           NOP
                         ;
                         ;
                         ;COLD START LOCATION, USER'S PROGRAM ENTERS HERE
                         ;
                         ;
0120 31803F              CSTART:  LXI     SP,3F80H         ;INITIALIZE SP
0123 CD1D00                       CALL    INIT             ;INITIALIZE EVERYTHING ELSE
0126 D3D4                         OUT     0D4H             ;RESET PFI LATCH
0128 FB                           EI                       ;ENABLE INTERRUPTS
                                  ;
                                  ;USER PROGRAM STARTS HERE
                                  ;
                                  END                      ;DONE
```

Figure 27. Power Down and Restart Software

5.2 78 LEVEL INTERRUPT SYSTEM

The second application illustrates an interrupt structure with greater than 64 levels for an 8080A or 8085A system. In the cascade mode, the 8259A supports up to 64 levels with direct vectoring to the service routine. Extending the structure to greater than 64 levels requires polling, using the poll command. A 78 level interrupt structure is used as an illustration; however, the principles apply to systems with up to 512 levels.

To implement the 78 level structure, 3 tiers of 8259A's are used. Nine 8259A's are cascaded in the master-slave scheme, giving 64 levels at tier 2. Two additional 8259A's are connected, by way of the INT outputs, to two of the 64 inputs. The 16 inputs at tier 3, combined with the 62 remaining tier 2 inputs, give 78 total levels. The fully nested structure is preserved over all levels, although direct vectoring is supplied for only the tier 2 inputs. Software is required to vector any tier 3 requests. Figure 28 shows the tiered structure used in this example. Notice that the tier 3 8259A's are connected to the bottom level slave (SA7). The master-slaves are interconnected as shown in "Interrupt Cascading", while the tier 3 8259A's are connected as "masters"; that is, the SP/EN pins are pulled high and the CAS pins are left unconnected. Since these 8259A's are only going to be used with the poll command, no $\overline{\text{INTA}}$ is required, therefore the $\overline{\text{INTA}}$ pins are pulled high.

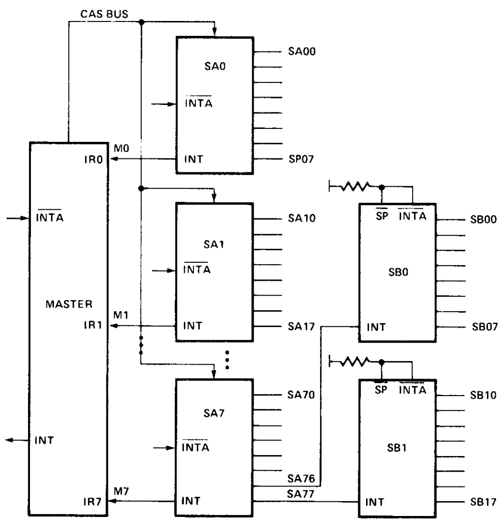


Figure 28. 78 Level Interrupt Structure

The concept used to implement the 78 levels is to directly vector to all tier 2 input service routines. If a tier 2 input contains a tier 3 8259A, the service routine for that input will poll the tier 3 8259A and branch to the tier 3 input service routine based on the poll word read after the poll command. Figure 29 shows how the jump table is organized assuming a starting location of 1000H and contiguous tables for all the tier 2 8259A's. Note that "SA35" denotes the IR5 input of the slave connected to the master IR3 input. Also note that for the normal tier 2 inputs, the jump table vectors the processor directly to the service routine for that input, while for the tier 2 inputs with 8259A's connected to their IR inputs, the processor is vectored to a service routine (i.e., SB0) which will poll to determine the actual tier 3 input requesting service. The polling routine utilizes the jump table starting at 1200H to vector the processor to the correct tier 3 service routine.

Each 8259A must receive an initialization sequence regardless of the mode. Since the tier 1 and 2 8259A's are in cascade and the special fully nested mode is used (covered shortly), all ICWs are required. The tier 3 8259A's don't require ICW3 or ICW4 since only polling will be used on them and they are connected as masters not in the cascade mode. The initialization sequence for each tier is shown in Figure 30. Notice that the master is initialized with a "dummy" jump table starting at 00H since all vectoring is done by the slaves. The tier 3 devices also receive "dummy" tables since only polling is used on tier 3.

As explained in "Interrupt Cascading", to preserve a truly fully nested mode within a slave, the master 8259A should be programmed in the special fully nested mode. This allows the master to acknowledge all interrupts at and above the level in service disregarding only those of lower priority. The special fully nested mode is programmed in the master only, so it only affects the immediate slaves (tier 2 not tier 3). To implement a fully nested structure among tier 3 slaves some special housekeeping software is required in all the tier-2-with-tier-3-slave routines. The software should simply save the state of the tier 2 IMR, mask all the lower tier 2 interrupts, then issue a specific EOI, resetting the ISR of the tier 2 interrupt level. On completion of the routine the IMR is restored.

Figure 31 shows an example flow and program for any tier 2 service routine without a tier 3 8259A. Figure 32 shows an example flow and program for any tier 2 service routine with a tier 3 8259A. Notice the reading of the ISR in both examples; this is done to determine whether or not to issue an EOI command to the master (refer to the section on "Special Fully Nested Mode" for further details).

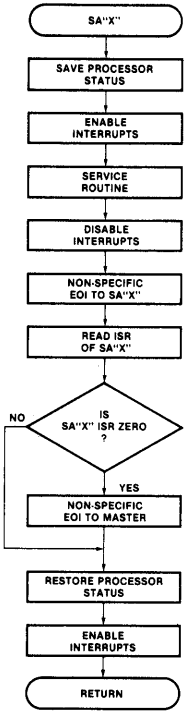
```
LOCATION    8259    CODE    
1000 H      SA0     JMP     SA00    ; SA00 SERVICE ROUTINE
  .
  .
101C H              JMP     SA07    ; SA07 SERVICE ROUTINE
1020 H      SA1     JMP     SA10    ; SA10 SERVICE ROUTINE
  .
  .
103C H              JMP     SA17    ; SA17 SERVICE ROUTINE
  .         .               .
  .         .               .       ; SA20-SA67 SERVICE ROUTINES
  .         .               .
10E0 H      SA7     JMP     SA70    ; SA70 SERVICE ROUTINE
  .
  .
10F8 H              JMP     SB0     ; SB0 POLL ROUTINE
10FC H              JMP     SB1     ; SB1 POLL ROUTINE
1200 H      SB0     JMP     SB00    ; SB00 SERVICE ROUTINE
  .
  .
121C H              JMP     SB07    ; SB07 SERVICE ROUTINE
1220 H      SB1     JMP     SB10    ; SB10 SERVICE ROUTINE
  .
  .
123C H              JMP     SB17    ; SB17 SERVICE ROUTINE
```

Figure 29. Jump Table Organization

```
; INITIALIZATION SEQUENCE FOR 78 LEVEL INTERRUPT STRUCTURE
;
; INITIALIZE MASTER
;
MINT:   MVI     A,15H       ; ICWI, LTM = 0, ADI = 1, S = 0, IC4 = 1
        OUT     MPTA        ; MASTER PORT A0 = 0
        MVI     A,00H       ; ICW2, DUMMY ADDRESS
        OUT     MPTB        ; MASTER PORT A0 = 1
        MVI     A,0FFH      ; ICW3, S7-S0 = 1
        OUT     MPTB        ; MASTER PORT A0 = 1
        MVI     A,I0H       ; ICW4, SFNM = 1
        OUT     MPTB        ; MASTER PORT A0 = 1
;
; INITIALIZE SA SLAVES - X DENOTES SLAVE ID (SEE KEY)
;
SAXINT: MVI     A,α         ; SEE KEY FOR ICW1, LTM = 0, ADI = 1, S = 0, IC4 = 1
        OUT     SAXPTA      ; SA"X" PORT A0 = 0
        MVI     A, 10H      ; ICW2, ADDRESS MSB
        OUT     SAXPTB      ; SA"X" PORT A0 = 1
        MVI     A0XH        ; ICW3, SA ID
        OUT     SAXPTB      ; SA"X" PORT A0 = 1
        MVI     A10H        ; ICW4, SFNM = 1
        OUT     SAXPTB      ; SA"X" PORT A0 = 1
;
; REPEAT ABOVE FOR EACH SA SLAVE
;
; INITIALIZE SB SLAVES - X DENOTES 0 or 1 (DO SB0, REPEAT FOR SB1)
;
SBXINT  MVI     A,I6H       ; ICW1, LTM = 0, ADI = 1, S = 1, IC4 = 0
        OUT     SBXPTA      ; SB"X" PORT A0 = 0
        MVI     A,00H       ; ICW2, DUMMY ADDRESS
        OUT     SBXPTB      ; SB"X" PORT A0 = 1
```

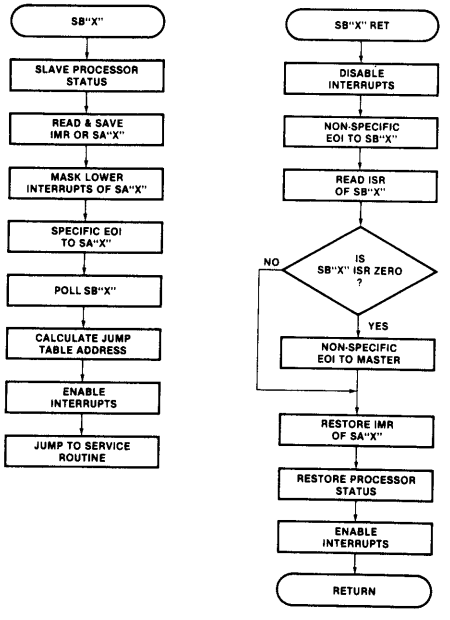
SA INITIALIZATION KEY		
SA"X"	α (ICW1)	JUMP TABLE START (H)
0	15	1000
1	35	1020
2	55	1040
3	75	1060
4	95	1080
5	B5	10A0
5	D5	10C0
7	F5	10E0

Figure 30. Initialization Sequence for 78 Level Interrupt Structure




```
; SA"X" ROUTINE - GENERAL INTERRUPT SERVICE ROUTINE
; FOR TIER 2 INTERRUPTS WITHOUT TIER 3 8259A
;
SAX:    PUSH D              ; SAVE DE
        PUSH B              ; SAVE BC
        PUSH H              ; SAVE HL
        PUSH PSW            ; SAVE A, FLAGS
        EI                  ; ENABLE INTERRUPTS
;
; SERVICE ROUTINE GOES HERE
;
        DI                  ; DISABLE INTERRUPTS
        MVI   20H           ; OCW2, NON-SPECIFIC EOI
        OUT   SAXPTA        ; SA"X" PORT A0 = 0
        MUI   A,08H         ; OCW3, READ REGISTER, ISR
        OUT   SAXPTA        ; SA"X" PORT A0 = 0
        IN    SAXPTA        ; SA"X" PORT A0 = 0, SA"X" ISR
        ANI   0FFH          ; TEST FOR ZERO
        JZN   SAXRSR        ; IF NOT ZERO, RESTORE STATUS
        MVI   A,0BH         ; OCW2, NON-SPECIFIC EOI
        OUT   MASPTA        ; MASTER PORT A0 = 0
SAXRSR: POP   PSW           ; RESTORE A, FLAGS
        POP   H             ; RESTORE HL
        POP   B             ; RESTORE BC
        POP   D             ; RESTORE DE
        EI                  ; ENABLE INTERRUPTS
        RET                 ; RETURN
```

Figure 31. Example Service Routine for Tier 2 Interrupt (SA"X") without Tier 3 8259A (SB"X")




```
; SB"X" ROUTINE - SERVICE ROUTINE FOR TIER 2
; INTERRUPTS WITH TIER 3 8259AS
 SBX:   PUSH D              ; SAVE DE
        PUSH B              ; SAVE BC
        PUSH H              ; SAVE HL
        PUSH PSW            ; SAVE A, FLAGS
        IN    SAXPTB        ; READ SA"X" IMR
        MOV   D,A           ; SAVE
        MVI   A,XXH         ; MASK SA"X" LOWER IR
        OUT   SAXPTB        ; SA"X" PORT A0 = 1
        MVI   A,8XH         ; OCW2 SPECIFIC EOI SA"X"
        OUT   SAXPTA        ; SA"X" PORT A0 = 1
        LXI   H,1200H       ; JUMP TABLE START
        MVI   B,00H         ; CLEAR B
        MVI   A,0CH         ; OCW3, POLL COMMAND
        OUT   SBXPTA        ; SB"X" PORT A0 = 0
        IN    SBXPTA        ; GET POLL WORD
        ANI   07H           ; LIMIT TO 3 BITS
        ADD   A             ; GET TABLE OFFSET
        ADD   A             ;
        MOV   C,A           ; OFFSET TO C
        DAD   B             ; HL HAS TABLE ADDRESS
        EI                  ; ENABLE INTERRUPTS
;
; SB"X"RET ROUTINE - FOR EOI AND MASK RESTORE
; AFTER SB"X" ROUTINE
;
 SBXRET DI                  ; DISABLE INTERRUPTS
        MVI   A,20H         ; OCW2, NON SPECIFIC EOI
        OUT   SBXPTA        ; SA"X" PORT A0 = 0
        MVI   A,0BH         ; OCW3, READ REGISTER ISR
        OUT   SAXPTA        ; SA"X" PORT A0 = 0
        IN    SBXPTA        ; SA"X" PORT A0 = 0, ISR
        ANI   0FFH          ; TEST FOR ZERO
        JNZ   SBXRSR        ; IF ≠ 0 RESTORE IMR
        MVI   A,20H         ; OCW2, NON SPECIFIC EOI
        OUT   MASPTA        ; MASTER PORT A0 = 0
SBXRSR: MOV   A,D           ; RESTORE SA"X" IMR
        OUT   SAXPTB        ; SA"X" PORT A0 = 1
        POP   PSW           ; RESTORE A, FLAGS
        POP   H             ; RESTORE HL
        POP   B             ; RESTORE BC
        POP   D             ; RESTORE DE
        EI                  ; RESTORE DE
        RET                 ; RETURN
```

Figure 32. Example Service Routine for Tier 2 Interrupt (SA"X") with Tier 3 8259A (SB"X")

5.3 TIMER CONTROLLED INTERRUPTS

In a large number of controller type microprocessor designs, certain timing requirements must be implemented throughout program execution. Such time dependent applications include control of keyboards, displays, CRTs, printers, and various facets of industrial control. These examples, however, are just a few of many designs which require device servicing at specific rates or generation of time delays. Trying to maintain these timing requirements by processor control alone can be costly in throughput and software complexity. So, what can be done to alleviate this problem? The answer, use the 8259A Programmable Interrupt Controller and external timing to interrupt the processor for time dependent device servicing.

This application example uses the 8259A for timer controlled interrupts in an 8086 system. External timing is done by two 8253 Programmable Interval Timers. Figure 33 shows a block diagram of the timer controlled interrupt circuitry which was built on the breadboard area of an SDK-86 (system design kit). Besides the 8259A and the 8253's, the necessary I/O decoding is also shown. The timer controlled interrupt circuitry interfaces with the SDK-86 which serves as the vehicle of operation for this design.

A short overview of how this application operates is as follows. The 8253's are programmed to generate interrupt requests at specific rates to a number of the 8259A IR inputs. The 8259A processes these requests by interrupting the 8086 and vectoring program execution to the appropriate service routine. In this example, the routines use the SDK-86 display panel to display the number of the interrupt level being serviced. These routines are merely for demonstration purposes to show the necessary procedures to establish the user's own routines in a timer controlled interrupt scheme.

Let's go over the operation starting with the actual interrupt timing generation which is done by two 8253 Programmable Interval Timers (8253 #1 and 8253 #2). Each 8253 provides three individual 16-bit counters (counters 0-2) which are software programmable by the processor. Each counter has a clock input (CLK), gate input (GATE), and an output (OUT). The output signal is based on divisions of the clock input signal. Just how or when the output occurs is determined by one of the 8253's six programmable modes, a programmable 16-bit count, and the state of the gate input.

Figure 34 shows the 8253 timing configuration used for generating interrupts to the 8259A. The SDK-86's PCLK (peripheral clock) signal provides a 400 ns period clock to CLK0 of 8253 #1. Counter 0 is used in mode 3 (square wave rate generator), and acts as a prescaler to provide the clock inputs of the other counters with a 10 ms period square wave. This 10 ms clock period made it easy to calculate exact timings for the other counters. Counter 2 of the 8253 #1 is used in mode 2 (rate generator), it is programmed to output a 10 ms pulse for every 200 pulses it receives (every 2 sec). The output of counter 2 causes an interrupt on IR1 of the 8259A. All the 8253 #2 counters are used in mode 5 (hardware triggered strobe) in which the gate input initiates counter operations. In this case the output of 8253 #1 counter 2 controls the gate of each 8253 #2 counter. When one of the 8253 #2 counters receive the 8253 #1 counter 2 output pulse on its gate, it will output a pulse (10 ms in duration) after a certain preprogrammed number of clock pulses have occurred. The programmed number of clock pulses for the 8253 #2 counters is as follows: 50 pulses (0.5 sec) for counter 0, 100 pulses (1 sec) for counter 1, and 150 pulses (1.5 sec) for counter 2. The outputs of these counters cause interrupt requests on IR2 through IR4 of the 8259A. Counter 1 of 8253 #1 is used in mode 0 (interrupt on terminal count). Unlike the other modes used which initialize operation automatically or by gate triggering, mode 0 allows software controlled counter initialization. When counter 1 of 8253 #1 is set during program execution, it will count 25 clocks (250 ms) and then pull its output high, causing an interrupt request on IR0 of the 8259A. Figure 35 shows the timing generated by the 8253's which cause interrupt request on the 8259A IR inputs.

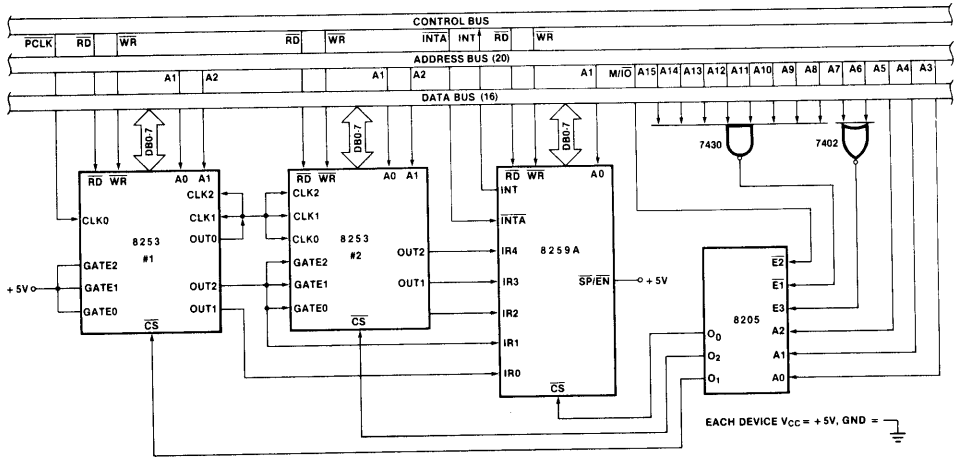


Figure 33. Timer Controlled Interrupt Circuit on SDK 86 Breadboard Area

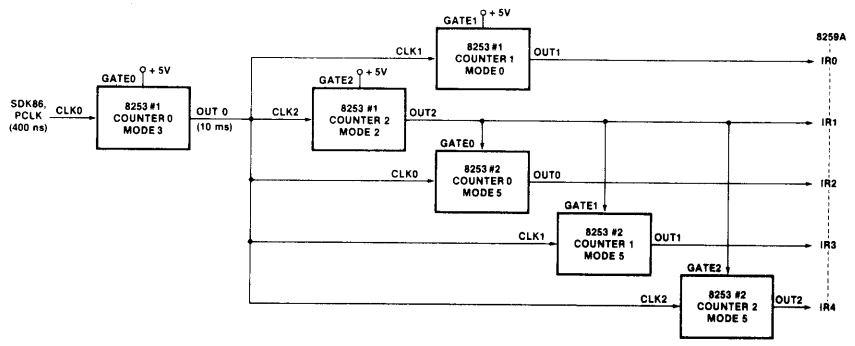


Figure 34. 8253 Timing Configuration for Timer Controlled Interrupts

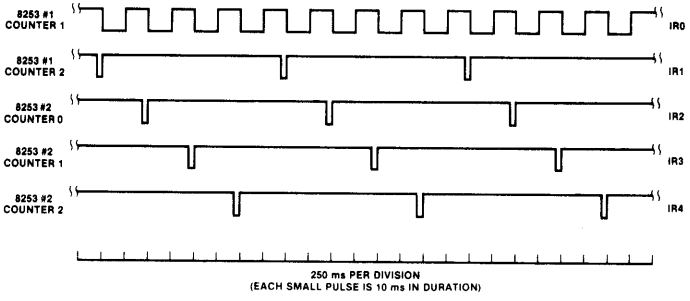


Figure 35. 8259A IR Input Signal From 8253S

There are basically two methods of timing generation that can be used in a timer controlled interrupt structure: dependent timing and independent timing. Dependent timing uses a single timing occurrence as a reference to base other timing occurrences on. On the other hand, independent timing has no mutual reference between occurrences. Industrial controller type applications are more apt to use dependent timing, whereas independent timing is prone to individual device control.

Although this application uses primarily dependent timing, independent timing is also incorporated as an example. The use of dependent timing can be seen back in Figure 34, where timing for IR2 through IR4 uses the IR1 pulse as reference. Each one of the 8253 #2 counters will generate an interrupt request a specific amount of times after the IR1 interrupt request occurs. When using the dependent method, as in this case, the IR2 through IR4 requests must occur before the next IR1 request. Independent timing is used to control the IR0 interrupt request. Note that its timing isn't controlled by any of the other IR requests. In this timer controlled interrupt configuration the dependent timing is initially set to be self running and the independent timing is software initialized. However, both methods can work either way by using the various 8253 modes to generate the same interrupt timing.

The 8259A processes the interrupts generated by the 8253's according to how it is programmed. In this application it is programmed to operate in the edge triggered mode, MCS-86/88 mode, and automatic EOI mode. In the edge triggered mode an interrupt request on an 8259A IR input becomes active on the rising edge. With this in mind, Figure 35 shows that IR0 will generate an interrupt every half second and IR1 through IR4 will each generate an interrupt every 2 seconds spaced apart at half second intervals. Interrupt vectoring in the MCS-86/88 mode is programmed so IR0, when activated, will select interrupt type 72. This means IR1 will select interrupt type 73, IR2 interrupt type 74, and so on through IR4. Since IR5 through IR7 aren't used, they are masked off. This prevents the possibility of any accidental interrupts and rids the necessity to tie the unused IR inputs to a steady level. Figure 36 shows the 8259A IR levels (IR0–IR4) with their corresponding interrupt type in the 8086 interrupt-vector table. Type 77 in the table is selected by a software "INT" instruction during program execution. Each type is programmed with the necessary code segment and instruction pointer values for vectoring to the appropriate service routine. Since the 8259A is programmed in the automatic EOI Mode, it doesn't require an EOI command to designate the completion of the service routine.

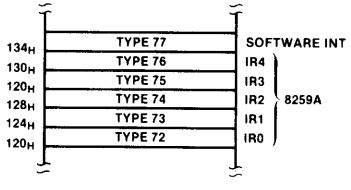


Figure 36. Interrupt "Type" Designation

As mentioned earlier, the interrupt service routines in this application are used merely to demonstrate the timer controlled interrupt scheme, not to implement a particular design. Thus a service routine simply displays the number of its interrupting level on the SDK-86 display panel. The display panel is controlled by the 8279 Keyboard and Display Controller. It is initialized to display "Ir" in its two left-most digits during the entire display sequence. When an interrupt from IR1 through IR4 occurs the corresponding routine will display its IR number via the 8279. During each IR1 through IR4 service routine a software "INT77" instruction is executed. This instruction vectors program execution to the service routine designated by type 77, which sets the 8253 counter controlling IR0 so it will cause an interrupt in 250 ms. When the IR0 interrupt occurs its routine will turn off the digit displayed by the IR1 through IR4 routines. Thus each IR level (IR1-IR4) will be displayed for 250 ms followed by a 250 ms off time caused by IR0. Figure 37 shows the entire display sequence of the timer controlled interrupt application.

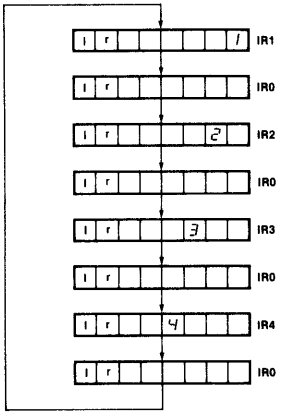


Figure 37. SDK Display Sequence for Timer Controlled Interrupts Program (Each Display Block Shown is 250 msec in Duration)

Now that we've covered the operation, let's move on to the program flow and structure of the timer controlled interrupt program. The program flow is made up of an initialization section and six interrupt service routines. The initialization program flow is shown in Figure 38. It starts by initializing some of the 8086's registers for program operation; this includes the extra segment, data segment, stack segment, and stack pointer. Next, by using the extra segement as reference, interrupt types 72 through 77 are set to vector interrupts to the appropriate routines. This is done by moving the code segment and instruction pointer values of each service routine into the corresponding type location. The 8253 counters are then programmed with the proper mode and count to provide the interrupt timing mentioned earlier. All counters with the exception of the 8253 #1, counter 1 are fully initialized at this point and will start counting. Counter 1 of 8253 #1 starts counting when its counter is loaded during the "INTR77" service routine, which will be covered shortly. Next, the 8259A is issued ICW1, ICW2, ICW4, and OCW1. The ICWs program the 8259A for the edge triggered mode, automatic EOI mode, and the proper interrupt vectoring (IR0, type 72). OCW1 is used to mask off the unused IR inputs (IR5-IR7). The 8279 is then set to display "IR" on its two left-most digits. After that the 8086 enables interrupts and a "dummy" main program is executed to wait for interrupt requests.

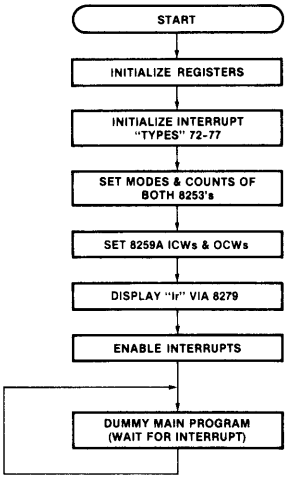


Figure 38. Initialization Program Flow for Timer Controlled Interrupts

There are six different interrupt service routines used in the program. Five of these routines, "INTR72" through "INTR76", are vectored to via the 8259A. Figure 39A-C shows the program flow for all six service routines. Note that "INTR73" through "INTR76" (IR1-IR4) basically use the same flow. These four similar routines display the number of its interrupting IR level on the SDK-86 display panel. The "INTR77" routine is vectored to by software during each of the previously mentioned routines and sets up interrupt timing to cause the "INTR72" (IR0) routine to be executed. The "INTR72" routine turns off the number on the SDK-86 display panel.

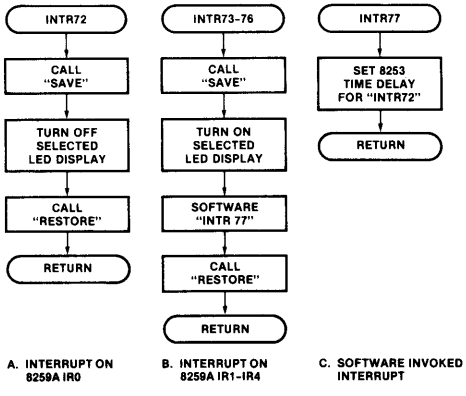


Figure 39. A-C. Interrupts Service Routine Flow for Timer Controlled Interrupts.

To best explain how these service routines work, let's assume an interrupt occurred on IR1 of the 8259A. The associated service routine for IR1 is "INTR73". Entering "INTR73", the first thing done is saving the pre-interrupt program status. This isn't really necessary in this program since a "dummy" main program is being executed; however, it is done as an example to show the operation. Rather than having code for saving the registers in each separate routine, a mutual call routine, "SAVE", is used. This routine will save the register status by pushing it on the stack. The next portion of "INTR73" will display the number of its IR level, "1", in the first digit of the SDK-86 display panel. After that, a software INT instruction is executed to vector program execution to the "INTR77" service routine. The "INTR77" service routine simply sets the 8253 #1 counter 1 to cause an interrupt on IR0 in 250 ms and then returns to "INTR73". Once back in "INTR73", the pre-interrupt status is restored by a call routine, "RESTORE". It does the opposite of "SAVE", returning the register status by popping it off the stack. The "INTR73" routine then returns to the "dummy " main program. The flow for the "INTR74" through "INTR76" routines are the same except for the digit location and the IR level displayed.

After 250 ms have elapsed, counter 1 of 8253 #1 makes an interrupt request on IR0 of the 8259A. This causes the "INTR72" service routine to be executed. Since this routine interrupts the main program, it also uses the "SAVE" routine to save pre-interrupt program status. It then turns off the digit displaying the IR level. In the case of the "INTR73" routine, the "1" is blanked out. The pre-interrupt status is then restored using the "RESTORE" routine and program execution returns to the "dummy" main program.

The complete program for the timer controlled interrupts application is shown in Appendix B. The program was executed in SDK-86 RAM starting at location 0500H (code segment = 0050, instruction pointer = 0).

CONCLUSION

This application note has explained the 8259A in detail and gives three applications illustrating the use of some of the numerous programmable features available. It should be evident from these discussions that the 8259A is an extremely flexible and easily programmable member of the Intel® MCS-80, MCS-85, MCS-86, and MCS-88 families.

This table is provided merely for reference information between the "Operation of the 8259A" and "Programming the 8259A" sections of this application note. It shouldn't be used as a programming reference guide (see "Programming the 8259A").

Operational Description	Command Words	Bits
MCS-80/85™ Mode	ICW1, ICW4*	IC4, μPM*
Address Interval for MCS-80/85 Mode	ICW1	ADI
Interrupt Vector Address for MCS-80/85 Mode	ICW1, ICW2	A5-A15
MCS-86/88 Mode	ICW1, ICW4	IC4, μPM
Interrupt Vector Byte for MCS-86/88 Mode	ICW2	T3-T7
Fully Nested Mode	OCW-Default	—
Non-Specific EOI Command	OCW2	EOI
Specific EOI Command	OCW2	SEOI, EOI, LO-L2
Automatic EOI Mode	ICW1, ICW4	IC4, AEOI
Rotate On Non-Specific EOI Command	OCW2	EOI
Rotate In Automatic EOI Mode	OCW2	R, SEOI, EOI
Set Priority Command	OCW2	L0-L2
Rotate on Specific EOI Command	OCW2	R, SEOI, EOI
Interrupt Mask Register	OCW1	M0-M7
Special Mask Mode	OCW3	ESMM-SMM
Level Triggered Mode	ICW1	LTIM
Edge Triggered Mode	ICW1	LTIM
Read Register Command, IRR	OCW3	ERIS, RIS
Read Register Command, ISR	OCW3	ERIS, RIS
Read IMR	OCW1	M0-M7
Poll Command	OCW3	P
Cascade Mode	ICW1, ICW3	SNGL, S0-7, ID0-2
Special Fully Nested Mode	ICW1, ICW4	IC4, SFNM
Buffered Mode	ICW1, ICW4	IC4, BUF, M/S

*Only needed if ICW4 is used for purposes other than μP mode set.

```
MCS-86 ASSEMBLER    TC159A


ISIS-II MCS-86 ASSEMBLER V1.0 ASSEMBLY OF MODULE TC159A
OBJECT MODULE PLACED IN :F1:TC159A.OBJ
ASSEMBLER INVOKED BY:  :F1:ASM86  F1 TC159A.SRC

LOC  OBJ                  LINE   SOURCE

                            1    ;******************** TIMER CONTROLLED INTERRUPTS ********************
                            2    ;
                            3    ;
                            4    ;
                            5    ;               EXTRA SEGMENT DECLARATIONS
                            6    ;
----                        7    EXTRA SEGMENT
                            8    ;
0120                        9            ORG     120H
0120 0401                  10    TP72IP  DW      INTR72          ;TYPE 72 INSTRUCTION POINTER
0122 ????                  11    TP72CS  DW      ?               ;TYPE 72 CODE SEGMENT
0124 1801                  12    TP73IP  DW      INTR73          ;TYPE 73 INSTRUCTION POINTER
0126 ????                  13    TP73CS  DW      ?               ;TYPE 73 CODE SEGMENT
0128 3001                  14    TP74IP  DW      INTR74          ;TYPE 74 INSTRUCTION POINTER
012A ????                  15    TP74CS  DW      ?               ;TYPE 74 CODE SEGMENT
012C 4801                  16    TP75IP  DW      INTR75          ;TYPE 75 INSTRUCTION POINTER
012E ????                  17    TP75CS  DW      ?               ;TYPE 75 CODE SEGMENT
0130 6001                  18    TP76IP  DW      INTR76          ;TYPE 76 INSTRUCTION POINTER
0132 ????                  19    TP76CS  DW      ?               ;TYPE 76 CODE SEGMENT
0134 7801                  20    TP77IP  DW      INTR77          ;TYPE 77 INSTRUCTION POINTER
0136 ????                  21    TP77CS  DW      ?               ;TYPE 77 CODE SEGMENT
                           22    ;
----                       23    EXTRA ENDS
                           24    ;
                           25    ;               DATA SEGMENT DECLARATIONS
                           26    ;
----                       27    DATA    SEGMENT
                           28    ;
0000 ????                  29    STACK1  DW      ?                       ;VARIABLE TO SAVE CALL ADDRESS
0002 ????                  30    AXTEMP  DW      ?                       ;VARIABLE TO SAVE AX REGISTER
0004 ??                    31    DIGIT   DB      ?                       ;VARIABLE TO SAVE SELECTED DIGIT
                           32    ;
----                       33    DATA    ENDS
                           34    ;
                           35    ;               CODE SEGMENT DECLARATION
                           36    ;
----                       37    CODE    SEGMENT
                           38    ;
                           39    ASSUME  ES:EXTRA,DS:DATA,CS:CODE
                           40    ;
                           41    ;               INITIALIZE REGISTERS
                           42    ;
0000 B80000                43    START:  MOV     AX,0H                   ;EXTRA SEGMENT AT 0H
0003 8EC0                  44            MOV     ES,AX
0005 B87000                45            MOV     AX,70H                  ;DATA SEGMENT AT 700H
0008 8ED8                  46            MOV     DS,AX
000A B87800                47            MOV     AX,78H                  ;STACK SEGMENT AT 780H
000D 8ED0                  48            MOV     SS,AX
000F BC8000                49            MOV     SP,80H                  ;STACK POINTER AT 80H (STACK=800H)
```

```
MCS-86 ASSEMBLER    TCI59A

LOC  OBJ                  LINE   SOURCE

                           50    ;
                           51    ;                LOAD INTERRUPT VECTOR TABLE
                           52    ;
0012 B80401                53    TYPE5:   MOV     AX,OFFSET (INTR72)      ;LOAD TYPE 72
0015 26A32001              54             MOV     TP72IP,AX
0019 268C0E2201            55             MOV     TP72CS,CS
001E B81801                56             MOV     AX,OFFSET (INTR73)      ;LOAD TYPE 73
0021 26A32401              57             MOV     TP73IP,AX
0025 268C0E2601            58             MOV     TP73CS,CS
002A B83001                59             MOV     AX,OFFSET (INTR74)      ;LOAD TYPE 74
002D 26A32801              60             MOV     TP74IP,AX
0031 268C0E2A01            61             MOV     TP74CS,CS
0036 B84801                62             MOV     AX,OFFSET (INTR75)      ;LOAD TYPE 75
0039 26A32C01              63             MOV     TP75IP,AX
003D 268C0E2E01            64             MOV     TP75CS,CS
0042 B86001                65             MOV     AX,OFFSET (INTR76)      ;LOAD TYPE 76
0045 26A33001              66             MOV     TP76IP,AX
0049 268C0E3201            67             MOV     TP76CS,CS
004E B87801                68             MOV     AX,OFFSET (INTR77)      ;LOAD TYPE 77
0051 26A33401              69             MOV     TP77IP,AX
0055 268C0E3601            70             MOV     TP77CS,CS
                           71    ;
                           72    ;                8253 INITIALIZATION
                           73    ;
005A BA0EFF                74    SET531:  MOV     DX,0FF0EH               ;8253 #1 CONTROL WORD
005D B036                  75             MOV     AL,36H                  ;COUNTER 0, MODE 3, BINARY
005F EE                    76             OUT     DX,AL
0060 B071                  77             MOV     AL,71H                  ;COUNTER 1, MODE 0, BCD
0062 EE                    78             OUT     DX,AL
0063 B0B5                  79             MOV     AL,0B5H                 ;COUNTER 2, MODE 2, BCD
0065 EE                    80             OUT     DX,AL
0066 BA08FF                81             MOV     DX,0FF08H               ;LOAD COUNTER 0 (10MS)
0069 B0A8                  82             MOV     AL,0A8H                 ;LSB
006B EE                    83             OUT     DX,AL
006C B061                  84             MOV     AL,61H                  ;MSB
006E EE                    85             OUT     DX,AL
006F BA0CFF                86             MOV     DX,0FF0CH               ;LOAD COUNTER 2 (2SEC)
0072 B000                  87             MOV     AL,00H                  ;LSB
0074 EE                    88             OUT     DX,AL
0075 B002                  89             MOV     AL,02H                  ;MSB
0077 EE                    90             OUT     DX,AL
0078 BA16FF                91    SET532:  MOV     DX,0FF16H               ;8253 #2 CONTROL WORD
007B B03B                  92             MOV     AL,3BH                  ;COUNTER 0, MODE 5,BCD
007D EE                    93             OUT     DX,AL
007E B07B                  94             MOV     AL,7BH                  ;COUNTER 1, MODE 5, BCD
0080 EE                    95             OUT     DX,AL
0081 B0BB                  96             MOV     AL,0BBH                 ;COUNTER 2, MODE 5, BCD
0083 EE                    97             OUT     DX,AL
0084 BA10FF                98             MOV     DX,0FF10H               ;LOAD COUNTER 0 (.5SEC)
0087 B050                  99             MOV     AL,50H                  ;LSB
0089 EE                   100             OUT     DX,AL
008A B000                 101             MOV     AL,00H                  ;MSB
008C EE                   102             OUT     DX,AL
008D BA12FF               103             MOV     DX,0FF12H               ;LOAD COUNTER 1 (1SEC)
0090 B000                 104             MOV     AL,00H                  ;LSB
```

```
MCS-86 ASSEMBLER    TCI59A

LOC  OBJ                LINE   SOURCE

0092 EE                  105           OUT     DX,AL
0093 B001                106           MOV     AL,01H                  ;MSB
0095 EE                  107           OUT     DX,AL
0096 BA14FF              108           MOV     DX,0FF14H               ;LOAD COUNTER 2 (1.5SEC)
0099 B050                109           MOV     AL,50H                  ;LSB
009B EE                  110           OUT     DX,AL
009C B001                111           MOV     AL,01H                  ;MSB
009E EE                  112           OUT     DX,AL
                         113   ;
                         114   ;               8259A INITIALIZATION
                         115   ;
009F BA00FF              116   SET59A: MOV     DX,0FF00H               ;8259A A0=0
00A2 B013                117           MOV     AL,13H                  ;ICW1-LTIM=0,S=1,IC4=1
00A4 EE                  118           OUT     DX,AL
00A5 BA02FF              119           MOV     DX,0FF02H               ;8259A A0=1
00A8 B048                120           MOV     AL, 48H                 ;ICW2-INTERRUPT TYPE 72 (120H)
00AA EE                  121           OUT     DX,AL
00AB B003                122           MOV     AL, 03H                 ;ICW4-SFNM=0,BUF=0,AEOI=1,MPM=1
00AD EE                  123           OUT     DX,AL
00AE B0E0                124           MOV     AL,0E0H                 ;OCW1-MASK IR5,6,7 (NOT USED)
00B0 EE                  125           OUT     DX,AL
                         126   ;
                         127   ;               8279 INITIALIZATION
                         128   ;
00B1 BAEAFF              129   SET79:  MOV     DX,0FFEAH               ;8279 COMMAND WORDS AND STATUS
00B4 B0D0                130           MOV     AL,0D0H                 ;CLEAR DISPLAY
00B6 EE                  131           OUT     DX,AL
00B7 EC                  132   WAIT79: IN      AL,DX                   ;READ STATUS
00B8 D0C0                133           ROL     AL,1                    ;"DU" BIT TO CARRY
00BA 72FB                134           JB      WAIT79                  ;JUMP IF DISPLAY IS UNAVAILABLE
00BC B087                135           MOV     AL, 87H                 ;DIGIT 8
00BE EE                  136           OUT     DX,AL
00BF BAE8FF              137           MOV     DX,0FFE8H               ;8279 DATA WORD
00C2 B006                138           MOV     AL,06H                  ;CHARACTER "1"
00C4 EE                  139           OUT     DX,AL
00C5 BAEAFF              140           MOV     DX,0FFEAH               ;8279 COMMAND WORD
00C8 B086                141           MOV     AL,86H                  ;DIGIT 7
00CA EE                  142           OUT     DX,AL
00CB BAE8FF              143           MOV     DX,0FFE8H               ;8279 DATA WORD
00CE B050                144           MOV     AL,50H                  ;CHARACTER "R"
00D0 EE                  145           OUT     DX,AL
00D1 FB                  146           STI                             ;ENABLE INTERRUPTS
                         147   ;
                         148   ;
                         149   ;               DUMMY PROGRAM
                         150   ;
00D2 EBFE                151   DUMMY:  JMP     DUMMY                   ;WAIT FOR INTERRUPT
                         152   ;
                         153   ;
00D4 A30200              154   SAVE:   MOV     AXTEMP,AX               ;SAVE AX
00D7 58                  155           POP     AX                      ;POP CALL RETURN ADDRESS
00D8 A30000              156           MOV     STACK1,AX               ;SAVE CALL RETURN ADDRESS
00DB A10200              157           MOV     AX,AXTEMP               ;RESTORE AX
00DE 50                  158           PUSH    AX                      ;SAVE PROCESSOR STATUS
00DF 53                  159           PUSH    BX
```

```
MCS-86 ASSEMBLER    TCI59A

LOC  OBJ                  LINE   SOURCE

00E0 51                   160          PUSH    CX
00E1 52                   161          PUSH    DX
00E2 55                   162          PUSH    BP
00E3 56                   163          PUSH    SI
00E4 57                   164          PUSH    DI
00E5 1E                   165          PUSH    DS
00E6 06                   166          PUSH    ES
00E7 A10000               167          MOV     AX,STACK1              ;RESTORE CALL RETURN ADDRESS
00EA 50                   168          PUSH    AX                     ;PUSH CALL RETURN ADDRESS
00EB C3                   169          RET
                          170  ;
00EC 58                   171  RESTOR: POP     AX                     ;POP CALL RETURN ADDRESS
00ED A30000               172          MOV     STACK1,AX              ;SAVE CALL RETURN ADDRESS
00F0 07                   173          POP     ES                     ;RESTORE PROCESSOR STATUS
00F1 1F                   174          POP     DS
00F2 5F                   175          POP     DI
00F3 5E                   176          POP     SI
00F4 5D                   177          POP     BP
00F5 5A                   178          POP     DX
00F6 59                   179          POP     CX
00F7 5B                   180          POP     BX
00F8 58                   181          POP     AX
00F9 A30200               182          MOV     AXTEMP,AX              ;SAVE AX
00FC A10000               183          MOV     AX,STACK1              ;RESTORE CALL RETURN ADDRESS
00FF 50                   184          PUSH    AX                     ;PUSH CALL RETURN ADDRESS
0100 A10200               185          MOV     AX,AXTEMP              ;RESTORE AX
0103 C3                   186          RET
                          187  ;
                          188  ;
                          189  ;                    INTERRUPT 72, CLEAR DISPLAY,IR0 8259A
                          190  ;
0104 E8CDFF               191  INTR72: CALL    SAVE                   ;ROUTINE TO SAVE PROCESSOR STATUS
0107 BAEAFF               192          MOV     DX,0FFEAH              ;8279 COMMAND WORD
010A A00400               193          MOV     AL,DIGIT               ;SELECTED LED DIGIT
010D EE                   194          OUT     DX,AL
010E BAE8FF               195          MOV     DX,0FFE8H              ;8279 DATA
0111 B000                 196          MOV     AL,00H                 ;BLANK OUT DIGIT
0113 EE                   197          OUT     DX,AL
0114 E8D5FF               198          CALL    RESTOR                 ;ROUTINE TO RESTORE PROCESSOR STATUS
0117 CF                   199          IRET                           ;RETURN FROM INTERRUPT
                          200  ;
                          201  ;
                          202  ;                    INTERRUPT 73, IR1 8259A
                          203  ;
0118 E8B9FF               204  INTR73: CALL    SAVE                   ;ROUTINE TO SAVE PROCESSOR STATUS
011B BAEAFF               205          MOV     DX,0FFEAH              ;8279 COMMAND WORD
011E B080                 206          MOV     AL,80H                 ;LED DISPLAY DIGIT 1
0120 A20400               207          MOV     DIGIT,AL
0123 EE                   208          OUT     DX,AL
0124 BAE8FF               209          MOV     DX,0FFE8H              ;8279 DATA
0127 B006                 210          MOV     AL,06H                 ;CHARACTER "1"
0129 EE                   211          OUT     DX,AL
012A CD4D                 212          INT     77                     ;TIMER DELAY FOR LED ON TIME
012C E8BDFF               213          CALL    RESTOR                 ;ROUTINE TO RESTORE PROCESSOR STATUS
012F CF                   214          IRET                           ;RETURN FROM INTERRUPT
```

```
MCS-86 ASSEMBLER    TCI59A

LOC  OBJ                LINE   SOURCE

                        215    ;
                        216    ;
                        217    ;               INTERRUPT 74, IR2 8259A
                        218    ;
0130 E8A1FF             219    INTR74: CALL    SAVE                    ;ROUTINE TO SAVE PROCESSOR STATUS
0133 BAEAFF             220            MOV     DX,0FFEAH               ;8279 COMMAND WORD
0136 B081               221            MOV     AL,81H                  ;LED DISPLAY DIGIT 2
0138 A20400             222            MOV     DIGIT,AL
013B EE                 223            OUT     DX,AL
013C BAE8FF             224            MOV     DX,0FFE8H               ;8279 DATA
013F B05B               225            MOV     AL,5BH                  ;CHARACTER "2"
0141 EE                 226            OUT     DX,AL
0142 CD4D               227            INT     77                      ;TIMER DELAY FOR LED ON TIME
0144 E8A5FF             228            CALL    RESTOR                  ;ROUTINE TO RESTORE PROCESSOR STATUS
0147 CF                 229            IRET                            ;RETURN FROM INTERRUPT
                        230    ;
                        231    ;
                        232    ;               INTERRUPT 75, IR3 8259A
                        233    ;
0148 E889FF             234    INTR75: CALL    SAVE                    ;ROUTINE TO SAVE PROCESSOR STATUS
014B BAEAFF             235            MOV     DX,0FFEAH               ;8279 COMMAND WORD
014E B082               236            MOV     AL,82H                  ;LED DISPLAY DIGIT 3
0150 A20400             237            MOV     DIGIT,AL
0153 EE                 238            OUT     DX,AL
0154 BAE8FF             239            MOV     DX,0FFE8H               ;8279 DATA
0157 B04F               240            MOV     AL,4FH                  ;CHARACTER "3"
0159 EE                 241            OUT     DX,AL
015A CD4D               242            INT     77                      ;TIMER DELAY FOR LED ON TIME
015C E8SDFF             243            CALL    RESTOR                  ;ROUTINE TO RESTORE PROCESSOR STATUS
015F CF                 244            IRET                            ;RETURN FROM INTERRUPT
                        245    ;
                        246    ;
                        247    ;               INTERRUPT 76, IR4 8259A
                        248    ;
0160 E871FF             249    INTR76: CALL    SAVE                    ;ROUTINE TO SAVE PROCESSOR STATUS
0163 BAEAFF             250            MOV     DX,0FFEAH               ;8279 COMMAND WORD
0166 B083               251            MOV     AL,83H                  ;LED DISPLAY DIGIT 4
0168 A20400             252            MOV     DIGIT,AL
016B EE                 253            OUT     DX,AL
016C BAE8FF             254            MOV     DX,0FFE8H               ;8279 DATA
016F B066               255            MOV     AL,66H                  ;CHARACTER "4"
0171 EE                 256            OUT     DX,AL
0172 CD4D               257            INT     77                      ;TIMER DELAY FOR LED ON TIME
0174 E875FF             258            CALL    RESTOR                  ;ROUTINE TO RESTORE PROCESSOR STATUS
0177 CF                 259            IRET                            ;RETURN FROM INTERRUPT
                        260    ;
                        261    ;
                        262    ;               INTERRUPT 77, TIMER DELAY, SOFTWARE CONTROLLED
                        263    ;
0178 BA0AFF             264    INTR77: MOV     DX,0FF0AH               ;LOAD COUNTER 1 8253 #1 (250 MSEC)
017B B025               265            MOV     AL,25H                  ;LSB
017D EE                 266            OUT     DX,AL
017E B000               267            MOV     AL,00H                  ;MSB
0180 EE                 268            OUT     DX,AL
0181 CF                 269            IRET                            ;RETURN FROM INTERRUPT
```

```
MCS-86 ASSEMBLER    TC159A


LOC  OBJ                  LINE   SOURCE

                          270   ;
                          271   ;
----                      272   CODE    ENDS;
                          273   ;
                          274   ;
  0000                    275           END     START

SYMBOL TABLE LISTING
------ ----- -------


NAME    TYPE     VALUE  ATTRIBUTES

??SEG   SEGMENT          SIZE=0000H PARA PUBLIC
AXTEMP  V WORD   0002H   DATA
CODE    SEGMENT          SIZE=0182H PARA
DATA    SEGMENT          SIZE=0005H PARA
DIGIT   V BYTE   0004H   DATA
DUMMY   L NEAR   0002H   CODE
EXTRA   SEGMENT          SIZE=0138H PARA
INTR72  L NEAR   0104H   CODE
INTR73  L NEAR   0118H   CODE
INTR74  L NEAR   0130H   CODE
INTR75  L NEAR   0148H   CODE
INTR76  L NEAR   0160H   CODE
INTR77  L NEAR   0178H   CODE
RESTOR  L NEAR   00ECH   CODE
SAVE    L NEAR   00D4H   CODE
SET531  L NEAR   005AH   CODE
SET532  L NEAR   0078H   CODE
SET59A  L NEAR   009FH   CODE
SET79   L NEAR   00B1H   CODE
STACK1  V WORD   0000H   DATA
START   L NEAR   0000H   CODE
TP72CS  V WORD   0122H   EXTRA
TP72IP  V WORD   0120H   EXTRA
TP73CS  V WORD   0126H   EXTRA
TP73IP  V WORD   0124H   EXTRA
TP74CS  V WORD   012AH   EXTRA
TP74IP  V WORD   0128H   EXTRA
TP75CS  V WORD   012EH   EXTRA
TP75IP  V WORD   012CH   EXTRA
TP76CS  V WORD   0132H   EXTRA
TP76IP  V WORD   0130H   EXTRA
TP77CS  V WORD   0136H   EXTRA
TP77IP  V WORD   0134H   EXTRA
TYPES   L NEAR   0012H   CODE
WAIT79  L NEAR   00B7H   CODE


ASSEMBLY COMPLETE, NO ERRORS FOUND
```

January 1979

Intel® MULTIBUS™ Interfacing

Joe Barthmaier
OEM Microcomputer Systems Applications

Related Intel Publications

MCS-80™ User's Manual, 98-153D

MCS-85™ User's Manual, 98-366C.

MCS-86™ User's Manual, 9800722A.

iSBC 80/20 and iSBC 80/20-4 Single Board Computer Hardware Reference Manual, 98-317C.

iSBC™ 86/12 Single Board Computer Hardware Reference Manual, 9800645A.

Intel® Multibus™ Specification, 9800683.

Intel® MULTIBUS™ Interfacing

Contents

I. INTRODUCTION

A significant measure of the power and flexibility of the Intel OEM Computer Product Line can be attributed to the design of the Intel MULTIBUS system bus. The bus structure provides a common element for communication between a wide variety of system modules which include: Single Board Computers, memory, digital, and analog I/O expansion boards, and peripheral controllers.

The purpose of this application note is to help you develop a working knowledge of the Intel MULTIBUS specification. This knowledge is essential for configuring a system containing multiple modules. Another purpose is to provide you with the information necessary to design a bus interface for a slave module. One of the tools that will be used to achieve this goal is the complete description of a MULTIBUS slave design example. Other portions of this application note provide an in depth examination of the bus signals, operating characteristics, and bus interface circuits.

This application note was originally written in 1977. Since 1977, the MULTIBUS specification has been significantly expanded to cover operation with both 8 and 16-bit system modules and with an auxiliary power bus. This application note now contains information on these new MULTIBUS specification features.

In addition, a detailed MULTIBUS specification has also been published which provides the user with further information concerning MULTIBUS interfacing. The MULTIBUS specification and other useful documents are listed in the overleaf of this note under Related Intel Publications.

II. MULTIBUS™ SYSTEM BUS DESCRIPTION

Overview

The Intel MULTIBUS signal lines can be grouped in the following categories: 20 address lines, 16 bidirectional data lines, 8 multilevel interrupt lines, and several bus control, timing and power supply lines. The address and data lines are driven by three-state devices, while the interrupt and some other control lines are open-collector driven.

Modules that use the MULTIBUS system bus have a master-slave relationship. A bus master module can drive the command and address lines: it can control the bus. A Single Board Computer is an example of a bus master. A bus slave cannot control the bus. Memory and I/O expansion boards are examples of bus slaves. The MULTIBUS architecture provides for both 8 and 16-bit bus masters and slaves.

Notice that a system may have a number of bus masters. Bus arbitration results when more than one master requests control of the bus at the same time. A bus clock is usually provided by one of the bus masters and may be derived independently from the processor clock. The bus clock provides a timing reference for resolving bus contention among multiple requests from bus masters. For example, a processor and a DMA (direct memory access) module may both request control of the bus. This feature allows different speed masters to share resources on the same bus. Actual transfers via the bus, however, proceed asynchronously with respect to the bus clock. Thus, the transfer speed is dependent on the transmitting and receiving devices only. The bus design prevents slow master modules from being handicapped in their attempts to gain control of the bus, but does not restrict the speed at which faster modules can transfer data via the same bus. Once a bus request is granted, single or multiple read/write transfers can proceed. The most obvious applications for the master-slave capabilities of the bus are multiprocessor configurations and high-speed direct-memory-access (DMA) operations. However, the master-slave capabilities of the bus are by no means limited to these two applications.

MULTIBUS™ Signal Descriptions

This section defines the signal lines that comprise the Intel MULTIBUS system bus. These signals are contained on either the P1 or P2 connector of boards compatible with the MULTIBUS specification. The P1 signal lines contain the address, data, bus control, bus exchange, interrupt and power supply lines. The P2 signal lines contain the optional auxiliary signal lines. Most signals on the bus are active-low. For example, a low level on a control signal on the bus indicates active, while a low level on an address or data signal on the bus represents logic "1" value.

NOTE

In this application note, a signal will be designated active-low by placing a slash (/) after the mnemonic for the signal.

Appendix A contains a pin assignment list of the following signals:

MULTIBUS P1 Signal Lines —

Initialization Signal Line

INIT/

Initialization signal; resets the entire system to a known internal state. INIT/ may be driven by one of the bus masters or by an external source such as a front panel reset switch.

Address and Inhibit Lines

ADR0/ - ADR13/

20 address lines; used to transmit the address of the memory location or I/O port to be accessed. The lines are labeled ADR0/ through ADR9/, ADRA/ through ADRF/ and ADR10/ through ADR13/. ADR13/ is the most significant bit. 8-bit masters use 16 address lines (ADR0/ - ADRF/) for memory addressing and 8 address lines (ADR0/ - ADR7/) for I/O port selection. 16-bit masters use all twenty address lines for memory addressing and 12 address lines (ADR0/ - ADRB/) for I/O port selection. Thus, 8-bit masters may address 64K bytes of memory and 256 I/O devices while 16-bit masters may address 1 megabyte of memory and 4096 I/O devices. (The 8086 CPU actually permits 16 address bits to be used to specify I/O devices, the MULTIBUS specification, however, states that only the low order 12 address bits can be used to specify I/O ports.) In a 16-bit system, the ADR0/ line is used to indicate whether a low (even) byte or a high (odd) byte of memory or I/O space is being accessed in a word oriented memory or I/O device.

BHEN/

Byte High Enable; the address control line which is used to specify that data will be transferred on the high byte (DAT8/ - DATF/) of the MULTIBUS data lines. With current iSBC boards, this signal effectively specifies that a word (two byte) transfer is to be performed. This signal is used only in systems which incorporate sixteen bit memory or I/O modules.

INH1/

Inhibit RAM signal; prevents RAM memory devices from responding to the memory address on the system address bus. INH1/ effectively allows ROM memory devices to override RAM devices when ROM and RAM memory are assigned the same memory addresses. INH1/ may also be used to allow memory mapped I/O devices to override RAM memory.

INH2/

Inhibit ROM signal; prevents ROM memory devices from responding to the memory address on the system address bus. INH2/ effectively allows auxiliary ROM (e.g., a bootstrap program) to override ROM devices when ROM and auxiliary ROM memory are assigned the same memory addresses. INH2/ may also be used to allow memory mapped I/O devices to override ROM memory.

Data Lines

DAT0/ - DATF/

16 bidirectional data lines; used to transmit or receive information to or from a memory location or I/O port. DATF/ being the most significant bit. In 8-bit systems, only lines DAT0/ - DAT7/ are used (DAT7/ being the most significant bit). In 16-bit systems, either 8 or 16 lines may be used for data transmission.

Bus Priority Resolution Lines

BCLK/

Bus clock; the negative edge (high to low) of BCLK/ is used to synchronize bus priority resolution circuits. BCLK/ is asynchronous to the CPU clock. It has a 100 ns minimum period and a 35% to 65% duty cycle. BCLK/ may be slowed, stopped, or single stepped for debugging.

CCLK/

Constant clock; a bus signal which provides a clock signal of constant frequency for unspecified general use by modules on the system bus. CCLK/ has a minimum period of 100 ns and a 35% to 65% duty cycle.

BPRN/

Bus priority in signal; indicates to a particular master module that no higher priority module is requesting use of the system bus. BPRN/ is synchronized with BCLK/. This signal is not bused on the backplane.

BPRO/

Bus priority out signal; used with serial (daisy chain) bus priority resolution schemes. BPRO/ is passed to the BPRN/ input of the master module with the next lower bus priority. BPRO/ is synchronized with BCLK/. This signal is not bused on the backplane.

BUSY/

Bus busy signal; an open collector line driven by the bus master currently in control to indicate that the bus is currently in use. BUSY/ prevents all other master modules from gaining control of the bus. BUSY/ is synchronized with BCLK/.

BREQ/

Bus request signal; used with a parallel bus priority network to indicate that a particular master module requires use of the bus for one or more data transfers. BREQ/ is synchronized with BCLK/. This signal is not bused on the backplane.

CBRQ/

Common bus request; an open-collector line which is driven by all potential bus masters and is used to inform the current bus master that another master wishes to use the bus. If CBRQ/ is high, it indicates to the bus master that no other master is requesting the bus, and therefore, the present bus master can retain the bus. This saves the bus exchange overhead for the current master.

Information Transfer Protocol Lines

A bus master provides separate read/write command signals for memory and I/O devices: MRDC/, MWTC/, IORC/ and IOWC/, as explained below. When a read/write command is active, the address signals must be stabilized at all slaves on the bus. For this reason, the protocol requires that a bus master must issue address signals (and data signals for a write operation) at least 50 ns ahead of issuing a read/write command to the bus, initiating the data transfer. The bus master must keep address signals unchanged until at least 50 ns after the read/write command is turned off, terminating the data transfer.

A bus slave must provide an acknowledge signal to the bus master in response to a read or write command signal.

MRDC/

Memory read command; indicates that the address of a memory location has been placed on the system address lines and specifies that the contents (8 or 16 bits) of the addressed location are to be read and placed on the system data bus. MRDC/ is asynchronous with respect to BCLK/.

MWTC/

Memory write command; indicates that the address of a memory location has been placed on the system address lines and that data (8 or 16 bits) has been placed on the system data bus. MWTC/ specifies that the data is to be written into the addressed memory location. MWTC/ is asynchronous with respect to BCLK/.

IORC/

I/O read command; indicates that the address of an input port has been placed on the system address bus and that the data (8 or 16 bits) at that input port is to be read and placed on the system data bus. IORC/ is asynchronous with respect to BCLK/.

IOWC/

I/O write command; indicates that the address of an output port has been placed on the system address bus and that the contents of the system data bus (8 or 16 bits) are to be output to the address port. IOWC/ is asynchronous with respect to BCLK/.

XACK/

Transfer acknowledge signal; the required response of a slave board which indicates that the specified read/write operation has been completed. That is, data has been placed on, or accepted from, the system data bus lines. XACK/ is asynchronous with respect to BCLK/.

Asynchronous Interrupt Lines

INT0/ - INT7/

8 Multi-level, parallel interrupt request lines;

used with a parallel interrupt resolution network. INT0 has the highest priority, while INT7 has lowest priority. Interrupt lines should be driven with open collector drivers.

INTA/

Interrupt acknowledge; an interrupt acknowledge line (INTA/), driven by the bus master, requests the transfer of interrupt information onto the bus from slave priority interrupt controllers (8259s or 8259As). The specific information timed onto the bus depends upon the implementation of the interrupt scheme. In general, the leading edge of INTA/ indicates that the address bus is active while the trailing edge indicates that data is present on the data lines.

MULTIBUS P2 Signal Lines — The signals contained on the MULTIBUS P2 auxiliary connector are used primarily by optional power back-up circuitry for memory protection. P2 signals are not bused on the backplane, and therefore, require a separate connector for each board using the P2 signals. Present iSBC boards have a slot in the card edge and should be used with a keyed P2 edge connector. Use of the P2 signal lines is optional.

ACLO

AC Low; this signal generated by the power supply goes high when the AC line voltage drops below a certain voltage (e.g., 103v AC in 115v AC line voltage systems) indicating D.C. power will fail in 3 msec. ACLO goes low when all D.C. voltages return to approximately 95% of the regulated value. This line must be pulled up by the optional standby power source, if one is used.

PFIN/

Power fail interrupt; this signal interrupts the processor when a power failure occurs, it is driven by external power fail circuitry.

PFSN/

Power fail sense; this line is the output of a latch which indicates that a power failure has occurred. It is reset by PFSR/. The power fail sense latch is part of external power fail circuitry and must be powered by the standby power source.

PFSR

Power fail sense reset; this line is used to reset the power fail sense latch (PFSN/).

MPRO/

Memory protect; prevents memory operation during period of uncertain DC power, by inhibiting memory requests. MPRO/ is driven by external power fail circuitry.

ALE

Address latch enable; generated by the CPU (8085 or 8086) to provide an auxiliary address latch.

HALT/

Halt; indicates that the master CPU is halted.

AUX RESET/

Auxiliary Reset; this externally generated signal initiates a power-up sequence.

WAIT/

Bus master wait state; this signal indicates that the processor is in a wait state.

Reserved — Several P1 and P2 connector bus pins are unused. However, they should be regarded as reserved for dedicated use in future Intel products.

Power Supplies — The power supply bus pins are detailed in Appendix A which contains the pin assignment of signals on the MULTIBUS backplane.

It is the designer's responsibility to provide adequate bulk decoupling on the board to avoid current surges on the power supply lines. It is also recommended that you provide high frequency

decoupling for the logic on your board. Values of 22uF for +5v and +12v pins and 10uF for -5v and -12v pins are typical on iSBC boards.

Operating Characteristics

Beyond the definition of the MULTIBUS signals themselves, it is important to examine the operating characteristics of the bus. The AC requirements outline the timing of the bus signals and in particular, define the relationships between the various bus signals. On the other hand, the DC requirements specify the bus driver characteristics, maximum bus loading per board, and the pull-up/down resistors.

The AC requirements are best presented by a discussion of the relevant timing diagrams. Appendix B contains a list of the MULTIBUS timing specifications. The following sections will discuss data transfers, inhibit operations, interrupt operations, MULTIBUS multi-master operation and power fail considerations.

Data Transfers — Data transfers on the MULTIBUS system bus occur with a maximum bandwidth of 5 MHz for single or multiple read/write transfers. Due to bus arbitration and memory access time, a typical maximum transfer rate is often on the order of 2 MHz.

Read Data

Figure 1 shows the read operation AC timing diagram. The address must be stable (t_{AS}) for a minimum of 50 ns before command (IORC/ or MRDC/). This time is typically used by the bus interface to decode the address and thus provide the required device selects. The device selects establish the data paths on the user system in anticipation of the strobe signal (command) which will follow. The minimum command pulse width is 100 ns. The address must remain stable for at least 50 ns following the command (t_{AH}). Valid data should not be driven onto the bus prior to command, and must not be removed until the command is cleared. The XACK/ signal, which is a response indicating the specified read/write operation has been completed, must coincide or follow both the read access and valid data (t_{DXL}). XACK/ must be held until the command is cleared (t_{XAH}).

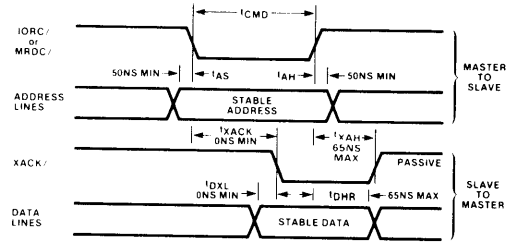


Figure 1. Read AC Timing

Write Data

The write operation AC timing diagram is shown in Figure 2. During a write data transfer, valid data must be presented simultaneously with a stable address. Thus, the write data setup time (t_{DS}) has the same requirement as the address setup time (t_{AS}). The requirement for stable data both before and after command (IOWC/ or MWTC/) enables the bus interface circuitry to latch data on either the leading or trailing edge of command.

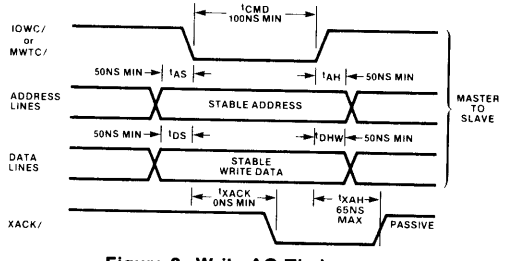


Figure 2. Write AC Timing

Data Byte Swapping in 16-bit Systems

A 16-bit master may transfer data on the MULTIBUS data lines using 8-bit or 16-bit paths depending on whether a byte or word (2 byte) operation has been specified. (A word transfer specified with an odd I/O or memory address will actually be executed as two single byte transfers.) An 8-bit master may only perform byte transfers on the MULTIBUS data lines DAT0/ - DAT7/.

In order to maintain compatibility with older 8-bit masters and slaves, a byte swapping buffer is included in all new 16-bit masters and 16-bit slaves. In the iSBC product line, all byte transfers will take place on the low 8 data lines DAT0/ - DAT7/. Figure 3 contains a example of 8/16-bit

data driver logic for 16-bit master and slave systems. In the 8/16-bit system, there are three sets of buffers; the lower byte buffer which accesses DAT0/ - DAT7/, the upper byte buffer which accesses DAT8/ - DATF/, and the swap byte buffer which accesses the MULTIBUS data lines DAT0/ - DAT7/ and transfers the data to/from the on-board data bus lines D8 - DF.

Figure 4 summarizes the 8 and 16-bit data paths used for three types of MULTIBUS transfers. Two signals control the data transfers.

Byte High Enable (BHEN/) active indicates that the bus is operating in sixteen bit mode, and Address Bit 0 (ADR0/) defines an even or odd byte transfer address.

On the first type of transfer, BHEN/ is inactive, and ADR0/ is inactive indicating the transfer of an even eight bit byte. The transfer takes place across data lines DAT0/ - DAT7/.

On the second type of transfer, BHEN/ is inactive, and ADR0/ is active indicating the transfer of a high (odd) byte. On this type of transfer, the odd (high) byte is transferred through the Swap Byte Buffer to DAT0/ - DAT7/. This makes eight bit and sixteen bit systems compatible.

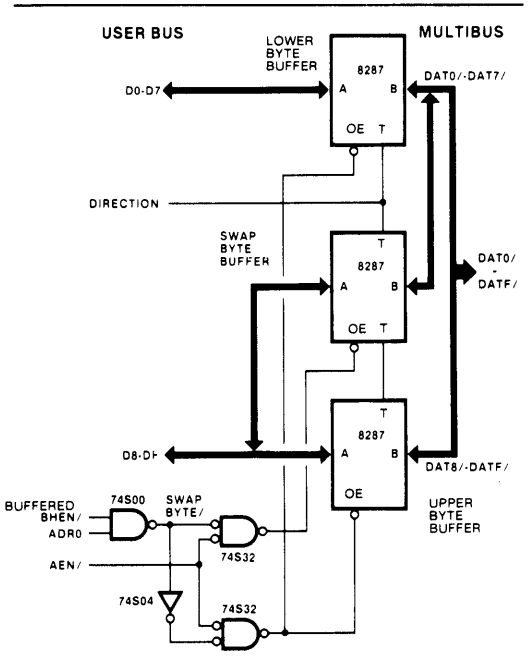


Figure 3. 8/16-Bit Data Drivers

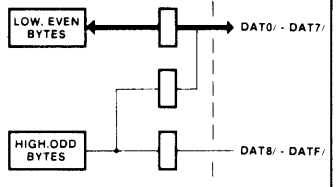
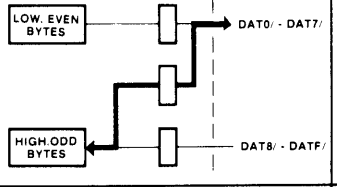
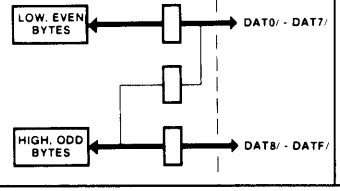
16-BIT DEVICE	MULTIBUS	BHEN/	ADR0/	MULTIBUS TRANSFER DATA PATH	DEVICE BYTE TRANSFERRED
		H	H	8-BIT, DAT0/ - DAT7/	EVEN
		H	L	8-BIT, DAT0/ - DAT7/	ODD
		L	H	16-BIT, DAT0/ - DATF/	EVEN AND ODD

Figure 4. 8/16-Bit Device Transfer Operation

The third type of transfer is a 16 bit (word) transfer. This is indicated by BHEN/ being active, and ADR0/ being inactive. On this type of transfer, the low (even) byte is transferred on DAT0/ - DAT7/ and the high (odd) byte is transferred on DAT8/ - DATF/.

Note that the condition when both BHEN/ and ADR0/ are active is not used with present iSBC boards. This condition could be used to transfer a high odd byte of data on DAT8/ - DATF/, thus eliminating the need for the swap byte buffer. However, this is not a recommended transfer type, because it eliminates the capability of communicating with 8-bit modules.

Inhibit Operations — Bus inhibit operations are required by certain bootstrap and memory mapped I/O configurations. The purpose of the inhibit operation is to allow a combination of RAM, ROM, or memory mapped I/O to occupy the same memory address space. In the case of a bootstrap, it may be desirable to have both ROM and RAM memory occupy the same address space, selecting ROM instead of RAM for low order memory only when the system is reset. A system designed to use memory mapped I/O, which has actual memory occupying the memory mapped I/O address space, may need to inhibit RAM or ROM memory to perform its functions.

There are two essential requirements for a successful inhibit operation. The first is that the inhibit signal must be asserted as soon as possible, within a maximum of 100 ns (t_{CI}), after stable address. The second requirement for a successful inhibit operation is that the acknowledge must be delayed (t_{XACKB}) to allow the inhibited slave to terminate any irreversible timing operations initiated by detection of a valid command prior to its inhibit.

This situation may arise because a command can be asserted within 50 ns after stable address (t_{AS}) and yet inhibit is not required until 100 ns (t_{ID}) after stable address. The acknowledge delay time (t_{XACKB}) is a function of the cycle time of the inhibited slave memory. Inhibiting the iSBC 016 RAM board, for example, requires a minimum of 1.5 usec. Less time is typically needed to inhibit other memory modules. For example, the iSBC 104 board requires 475 ns.

Figure 5 depicts a situation in which both RAM

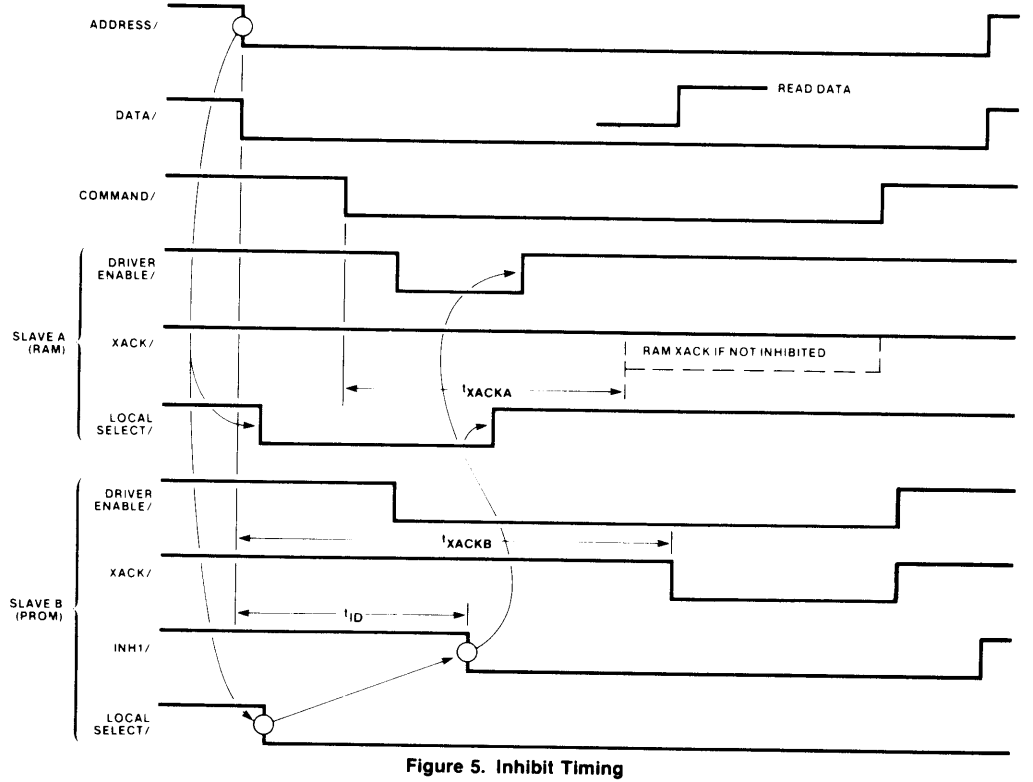


Figure 5. Inhibit Timing

and PROM memory have the same memory addresses. In this case, PROM inhibits RAM, producing the effect of PROM overriding RAM. After address is stable, local selects are generated for both the PROM and the RAM. The PROM local select produces the INH1/ signal which then removes the RAM local select and its driver enable. Because the slave RAM has been inhibited after it had already begun its cycle, the PROM XACK/ must be delayed (t_{XACKB}) until after the latest possible acknowledgement from the RAM (t_{XACKA}).

Interrupt Operations — The MULTIBUS interrupt lines INT0/ - INT7/ are used by a MULTIBUS master to receive interrupts from bus slaves, other bus masters or external logic such as power fail logic. A bus master may also contain internal interrupt sources which do not require the bus interrupt lines to interrupt the master. There are two interrupt implementation schemes used by bus interrupts, Non Bus Vectored Interrupts and Bus Vectored Interrupts. Non Bus Vectored Interrupts do not convey interrupt vector address information on the bus. Bus Vectored Interrupts are interrupts from slave Priority Interrupt Controllers (PICs) which do convey interrupt vector address information on the bus.

Non Bus Vectored Interrupts

Non Bus Vectored Interrupts are those interrupts whose interrupt vector address is generated by the bus master and do not require the MULTIBUS address lines for transfer of the interrupt vector address. The interrupt vector address is generated by the interrupt controller on the master and transferred to the processor over the local bus. The source of the interrupt can be on the master module or on other bus modules, in which case the bus modules use the MULTIBUS interrupt request lines (INT0/ - INT7/) to generate their interrupt requests to the bus master. When an interrupt request line is activated, the bus master performs it own interrupt operation and processes the interrupt. Figure 6 shows an example of Non Bus Vectored Interrupt implementation.

Bus Vectored Interrupts

Bus Vectored Interrupts (Figure 7) are those interrupts which transfer the interrupt vector address along the MULTIBUS address lines from the slave to the bus master using the INTA/ command signal for synchronization.

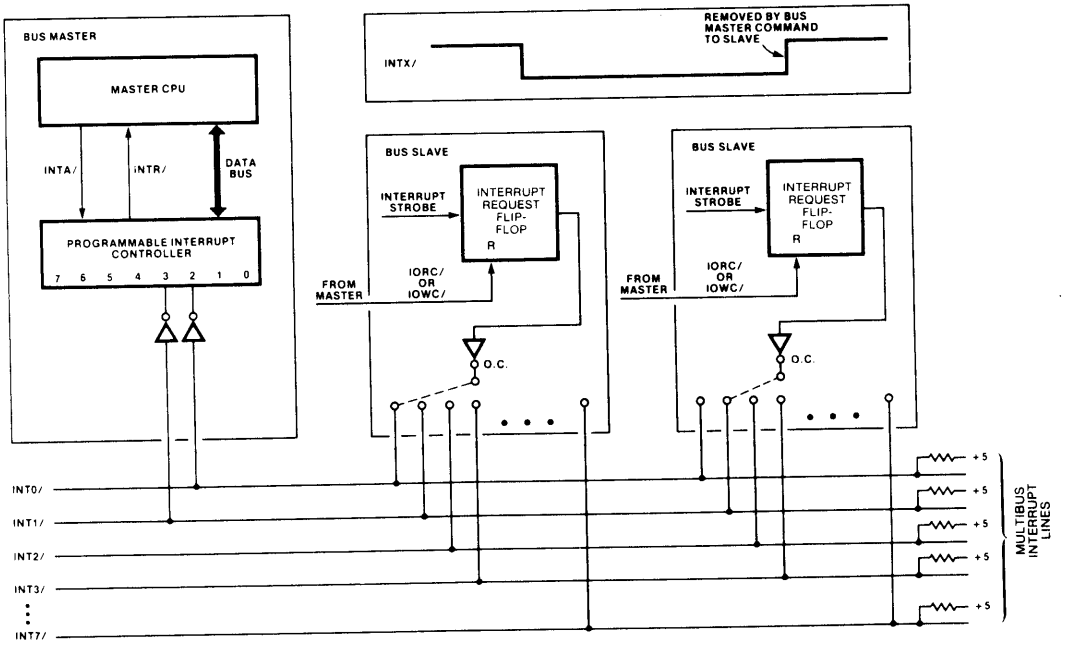


Figure 6. Non Bus Vectored Interrupt Implementation

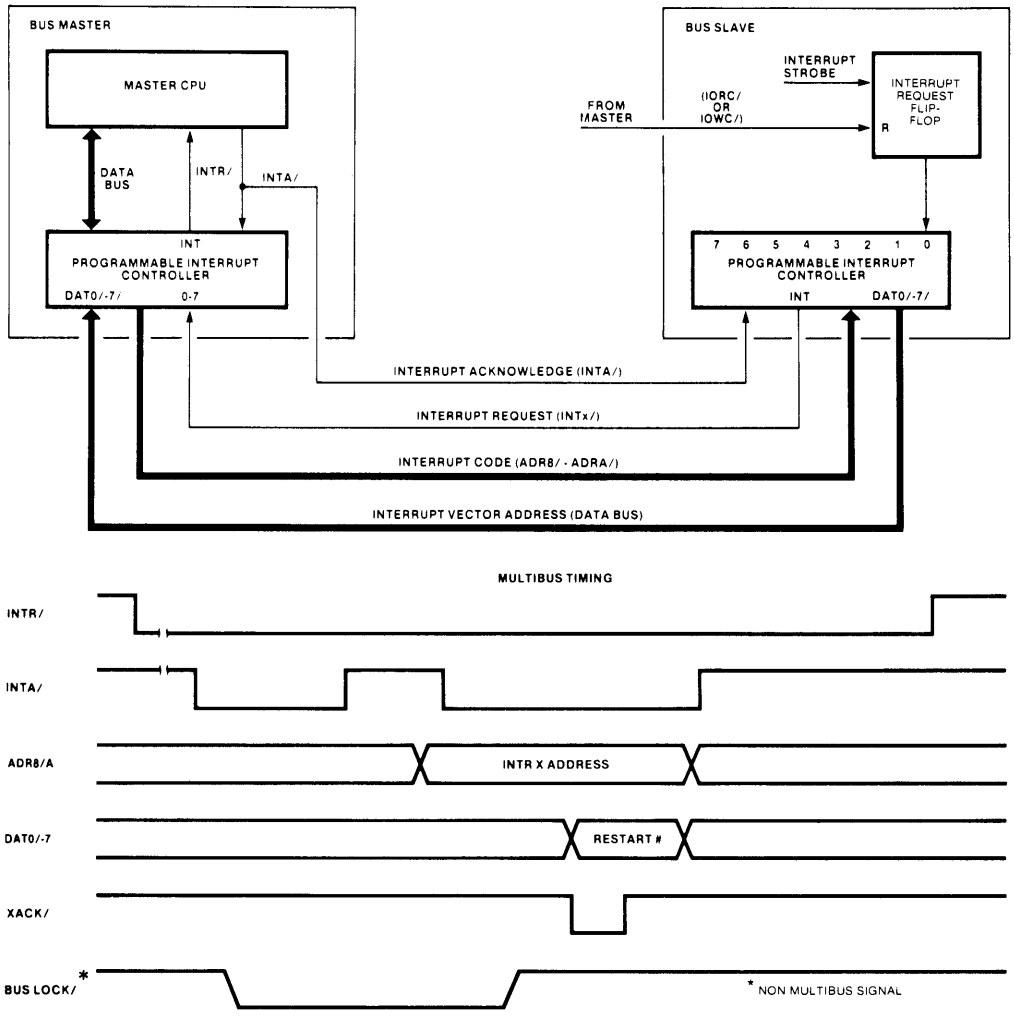


Figure 7. Bus Vectored Interrupt Logic (With 2 INTA/ Timing Diagram)

When an interrupt request from the MULTIBUS interrupt lines INT0/ - INT7/ occurs, the interrupt control logic on the bus master interrupts its processor. The processor on the bus master generates an INTA/ command which freezes the state of the interrupt logic on the MULTIBUS slaves for priority resolution. The bus master also locks (retains the bus between bus cycles) the MULTIBUS control lines to guarantee itself consecutive bus cycles. After the first INTA/ command, the bus master's interrupt control logic puts an interrupt code on to the MULTIBUS address lines ADR8/ - ADRA/. The interrupt code is the address of the highest priority active interrupt request line. At this point in the Bus Vectored Interrupt procedure, two different sequences could take place. The difference occurs, because the MULTIBUS specification can support masters which generate one additional INTA/ (8086 masters) or two additional INTA/s (8080A and 8085 masters).

If the bus master generates one additional INTA/, this second INTA/ causes the bus slave interrupt control logic to transmit an interrupt vector 8-bit pointer on the MULTIBUS data lines. The vector pointer is used by the bus master to determine the memory address of the interrupt service routine.

If the bus master generates two additional INTA/s, these two INTA/ commands allow the

bus slave to put a two byte interrupt vector address on to the MULTIBUS data lines (one byte for each INTA/). The interrupt vector address is used by the bus master to service the interrupt.

The MULTIBUS specification provides for only one type of Bus Vectored Interrupt operation in a given system. Slave boards which have an 8259 interrupt controller are only capable of 3 INTA/ operation (2 additional INTA/s after the first INTA/). Slave boards with the 8259A interrupt controller are capable of either 2 INTA/ or 3 INTA/ operation. All slave boards in a given system must operate in the same way (2 INTA/s or 3 INTA/s) if Bus Vectored Interrupts are to be used. However, the MULTIBUS specification does provide for Bus Vectored Interrupts and Non Bus Vectored Interrupts in the same system.

MULTIBUS Multi-Master Operation — The MULTIBUS system bus can accommodate several bus masters on the same system, each one taking control of the bus as it needs to affect data transfers. The bus masters request bus control through a bus exchange sequence.

Two bus exchange priority resolution techniques are discussed, a serial technique and a parallel technique. Figures 8 and 9 illustrate these two techniques. The bus exchange operation discussed later is the same for both techniques.

Serial Priority Technique

Serial priority resolution is accomplished with a daisy chain technique (see Figure 8). The priority input (BPRN/) of the highest priority master is tied to ground. The priority output (BPRO/) of the highest priority master is then connected to the priority input (BPRN/) of the next lower priority master, and so on. Any master generating a bus request will set its BPRO/ signal high to the next lower priority master. Any master seeing a high signal on its BPRN/ line will sets its BPRO/ line high, thus passing down priority information to lower priority masters. In this implementation, the bus request line (BREQ/) is not used outside of the individual masters. A limited number of masters can be accommodated by this technique, due to gate delays through the daisy chain. Using the current Intel MULTIBUS controller chip on the master boards up to 3 masters may be accommodated if a BCLK/ period of 100 ns is used. If more bus masters are required, either BCLK/ must be slowed or a parallel priority technique used.

Parallel Priority Technique

In the parallel priority technique, the priority is resolved in a priority resolution circuit in which the highest priority BREQ/ input is encoded with a priority encoder chip (74148). This coded value is then decoded with a priority decoder chip (74S138) to activate the appropriate BPRN/ line. The BPRO/ lines are not used in the parallel priority scheme. However, since the MULTIBUS backplane contains a trace from the BPRN/ signal of one card slot to the BPRO/ signal of the adjacent lower card slot, the BPRO/ must be disconnected from the bus on the board or the backplane trace must be cut. A practical limit of sixteen masters can be accommodated using the parallel priority technique due to physical bus length limitations. Figure 9 contains the schematic for a typical parallel resolution network. Note that the parallel priority resolution network must be externally supplied.

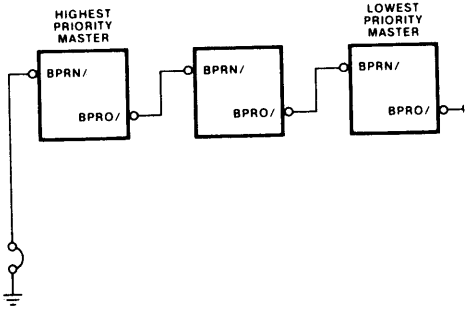


Figure 8. Serial Priority Technique

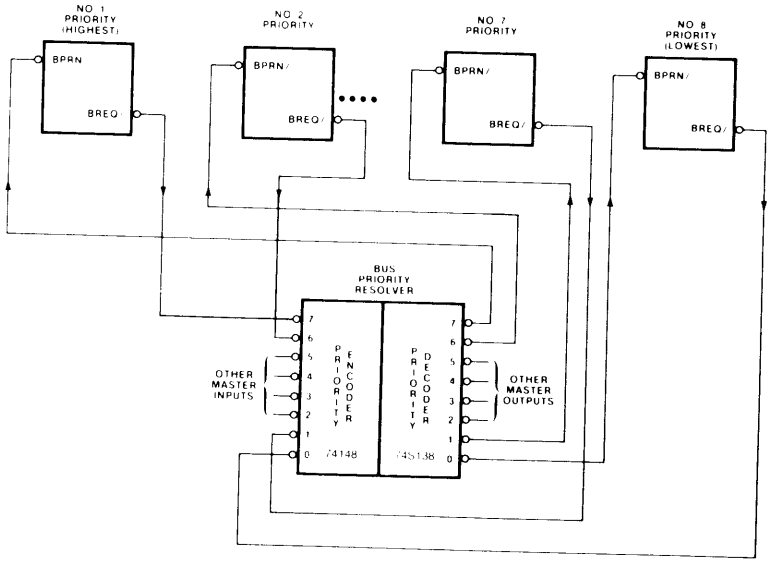


Figure 9. Parallel Priority Technique

MULTIBUS Exchange Operation — A timing diagram for the MULTIBUS exchange operation is shown in Figure 10. This implementation example uses a parallel resolution scheme, however, the timing would be basically the same for the serial resolution scheme.

In this example, master A has been assigned a lower priority than master B. The bus exchange occurs because master B generates a bus request during a time when master A has control of the bus.

The exchange process begins when master B requires the bus to access some resource such as an I/O or memory module while master A controls the bus. This internal request is synchronized with the trailing edge (high to low) of BCLK/ to generate a bus request (BREQ/). The bus priority resolution circuit changes the BPRN/ signal from active (low) to inactive (high) for master A and from inactive to active for master B. Master A must first complete the current bus command if one is in operation. After master A completes the command, it sets BUSY/ inactive on the next trailing edge of BCLK/. This allows the actual bus exchange to occur, because master A has relinquished control of the bus, and master B has been granted its BPRN/. During this time, the drivers for master A are disabled. Master B must take control of the bus with the next trailing edge of BCLK/ to complete the bus exchange. Master B takes control by activating BUSY/ and enabling its drivers.

It is possible for master A to retain control of the bus and prevent master B from getting control. Master A activates the Bus Override (or Bus Lock) signal which keeps BUSY/ active allowing control of the bus to stay with master A. This guarantees a master consecutive bus cycles for software or hardware functions which require exclusive, continuous access to the bus.

Note that in systems with only a single master it is necessary to ground the BPRN/ pin of the master, if slave boards are to be accessed. In single board systems which use a CPU board capable of Bus Vectored Interrupt operation, the BPRN/ pin must also be grounded.

In a single master system bus transfer efficiency may be gained if the BUS OVERRIDE signal is kept active continuously. This permits the master to maintain control of the bus at all times, therefore saving the overhead of the master reacquiring the bus each time it is needed.

The CBRQ/ line may be used by a master in control of the bus to determine if another master

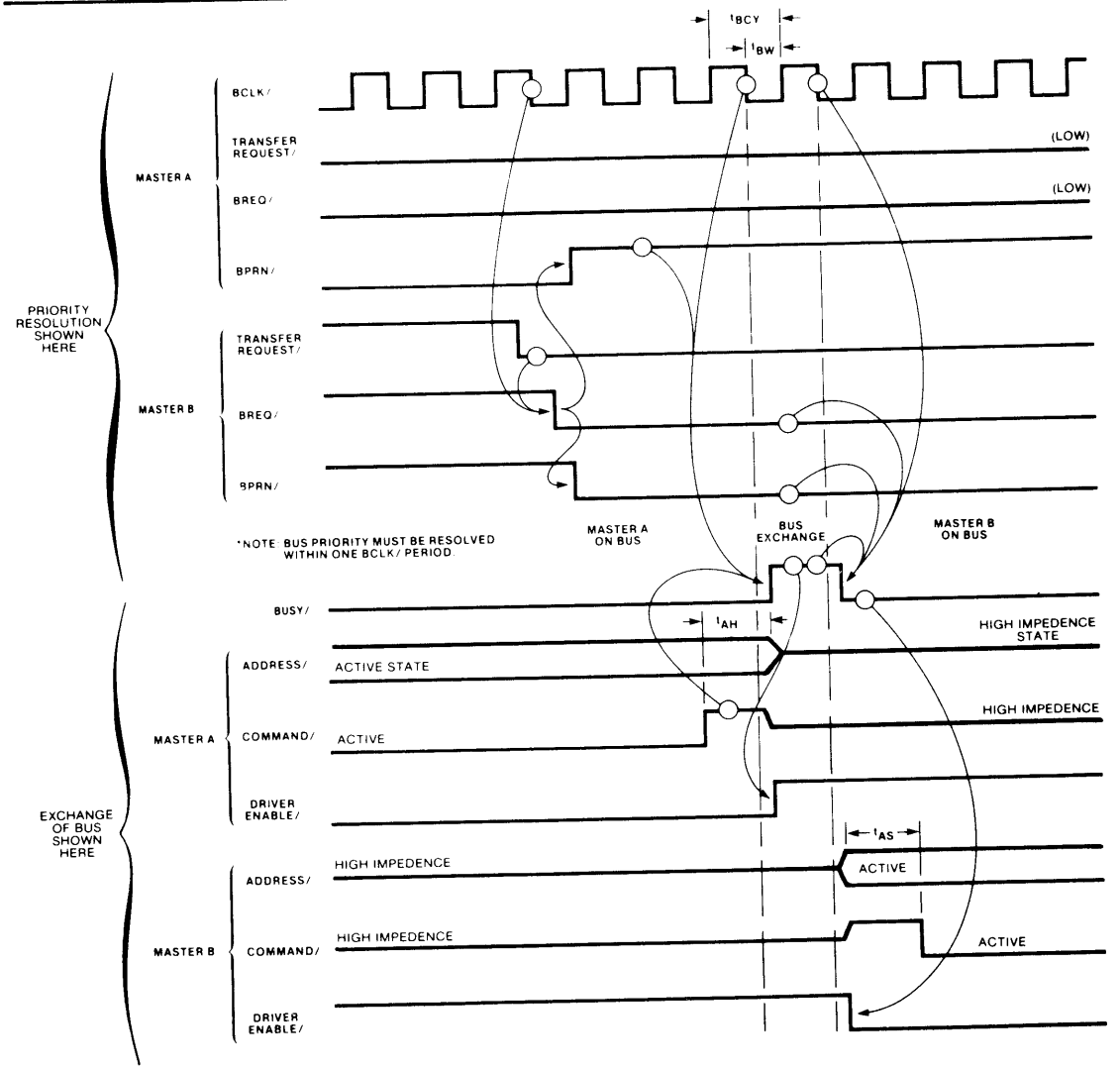


Figure 10. Bus Control Exchange Operation

requires the bus. If a master currently in control of the bus sees the CBRQ/ line inactive, it will maintain control of the bus between adjacent bus accesses. Therefore, when a bus access is required, the master saves the overhead of reacquiring the bus. If a current bus master sees the CBRQ/ line active, it will then relinquish control of the bus after the current bus access and will contend for the bus with the other master(s) requiring the bus. The relative priorities of the masters will determine which master receives the bus.

Note that except for the BUS OVERRIDE state, no single master may keep exclusive control of the bus. This is true because it is impossible for the CPU on a master to require continuous access to the bus. Other lower priority masters will always be able to gain access to the bus between accesses of a higher priority master.

Power Fail Considerations — The MULTIBUS P2 connector signals provide a means of handling power failures. The circuits required for power

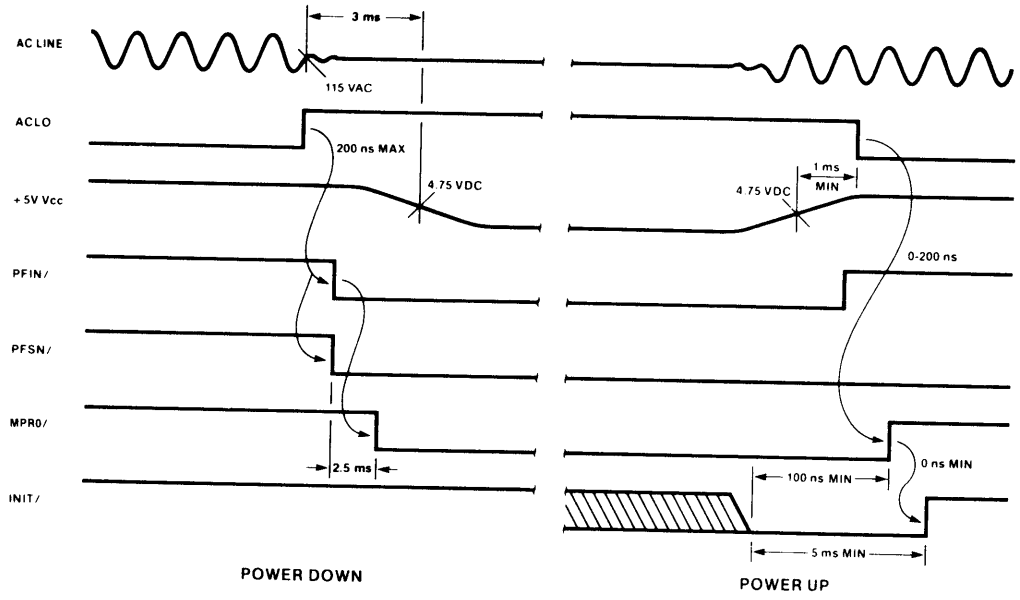


Figure 11. Power Fail Timing Sequence

failure detection and handling are optional and must be supplied by the user. Figure 11 shows the timing of a power fail sequence.

The power supply monitors the AC power level. When power drops below an acceptable value, the power supply raises ACLO which tells the power fail logic that a minimum of three milliseconds will elapse before DC power will fall below regulated voltage levels. The power fail logic sets a sense latch (PFSN/) and generates an interrupt (PFIN/) to the processor so the processor can store its environment. After a 2.5 millisecond timeout, the memory protect signal (MPRO/) is asserted by the power fail logic preventing any memory activity. As power falls, the memory goes on standby power. Note that the power fail logic must be powered from the standby source.

As the AC line revives, the logic voltage level is monitored by the power supply. After power has been at its operating level for one millisecond minimum, the power supply sets the signal ACLO low, beginning the restart sequence. First, the memory protect line (MPRO/) then the initialize line (INIT/) become inactive. The bus master now starts running. The bus master checks the power fail latch (PFSN/) and, if it finds it set, branches to a power up routine which resets the latch (PFSR/), restores the environment, and resumes execution.

Note that INIT/ is activated only after DC power has risen to the regulated voltage levels and must stay low for five milliseconds minimum before the system is allowed to restart. Alternatively, INIT/ may be held low through an open collector device by MPRO/.

How the power failure equipment is configured is left to the system designer. The backup power source may be batteries located on the memory boards or more elaborate facilities located off-board. The location of the power fail logic determines which MULTIBUS power fail lines are used. Pins on the P2 connector have been specified for the power failure functions for use as needed.

To further clarify the location and use of the power fail circuitry, an example of a typical power fail system block diagram is shown in Figure 12. A single board computer and a slave memory board are contained in the system. It is desired to power the memory circuit elements of the memory board from auxiliary power. The single board computer will remain on the main power supply. To accomplish this, user supplied power fail logic and

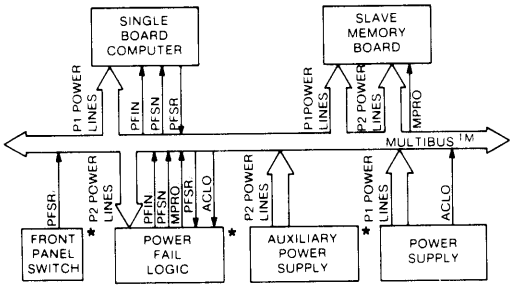


Figure 12. Typical Power Fail System Block Diagram

an auxiliary power supply have been included in the system.

The single board computer is powered from the P1 power lines and accesses the P2 signal lines PFIN/, PFSN/ and PFSR/ (only the P2 signal lines used by a particular functional block are shown on the block diagram). The PFSR/ line is driven from two sources: a front panel switch and the single board computer. The front panel switch is used during normal power-up to reset the power fail sense latch. The single board computer uses the PFSR/ line to reset the latch during a power-up sequence after a power failure. Current single board computers must access the PFSN/ and PFSR/ signals either directly with dedicated circuitry and a P2 pin connection or through the parallel I/O lines with a cable connection from the parallel I/O connector to the P2 connector.

The slave memory board uses both the P1 and P2 power lines, the P2 power lines are used (at all times) to power the memory circuit elements and other support circuits, the P1 power lines power all other circuitry. In addition, the MPRO/ line is input and used to sense when memory contents should be protected.

The power fail logic contains the power fail sense latch, and uses the PFSR/ and ACLO lines for inputs and the PFIN/ PFSN/, and MPRO/ lines for outputs. The power fail logic must be powered by the P2 power lines.

DC Requirements — The drive and load characteristics of the bus signals are listed in Appendix C. The physical locations of the drivers and loads, as well as the terminating resistor value for each bus line, are also specified. Appendix D contains the MULTIBUS power specifications.

MULTIBUS™ Slave Interface Circuit Elements

There are three basic elements of a slave bus interface: address decoders, bus drivers, and control signal logic. This section discusses each of these elements in general terms. A description of a detailed implementation of a slave interface is presented in a later section of this application note.

Address Decoding — This logic decodes the appropriate MULTIBUS address bits into RAM requests, ROM requests, or I/O selects. Care must be taken in the design of the address decode logic to ensure flexibility in the selection of base address assignments. Without this flexibility, restrictions may be placed upon various system configurations. Ideally, switches and jumper connections should be associated with the decode logic to permit field modification of base address assignments.

The initial step in designing the address decode portion of a MULTIBUS interface is to determine the required number of unique address locations. This decision is influenced by the fact that address decoding is usually done in two stages. The first stage decodes the base address, producing an enable for the second stage which generates the actual device selects for the user logic. A convenient implementation of this two stage decoding scheme utilizes a pair of decoders driven by the high order bits of the address for the first stage and a second decoder for the low order bits of the address bus. This technique forces the number of unique address locations to be a power of two, based at the address decoded by the first stage. Consider the scheme illustrated in Figure 13.

As shown in Figure 13, the address bits A_4 - A_B are used to produce switch selected outputs of the first stage of decoding. The 1 out of 8 binary decoders

have been used. The top decoder decodes address lines A_4 - A_7, and the bottom decoder decodes address lines A_8 - A_B. If only address lines A_0 - A_7 are being used for device selection, as in the case of I/O port selection in 8-bit systems, the bottom decoder may be disabled by setting switch S2 to the ground position. Address lines A_7 and A_B drive enable inputs E2 or E3 of the decoders. The address lines A_0 - A_3 enter the second stage address decoder to produce 8 user device selects. The second stage decoder must first be enabled by an address that corresponds to the switch-selected base address.

Address decoding must be completed before the arrival of a command. Since the command may become active within 50 ns after stable address, the decode logic should be kept simple with a minimal number of layers of logic. Furthermore, the timing is extremely critical in systems which make use of the inhibit lines.

A linear or unary select scheme in which no binary encoding of device address (e.g., address bit A_0 selects device 0, address bit A_1 selects device 1, etc.) is performed is not recommended because the scheme offers no protection in case multiple devices are simultaneously selected, and because the addressing within such a system is restricted by the extent of the address space occupied by such a scheme.

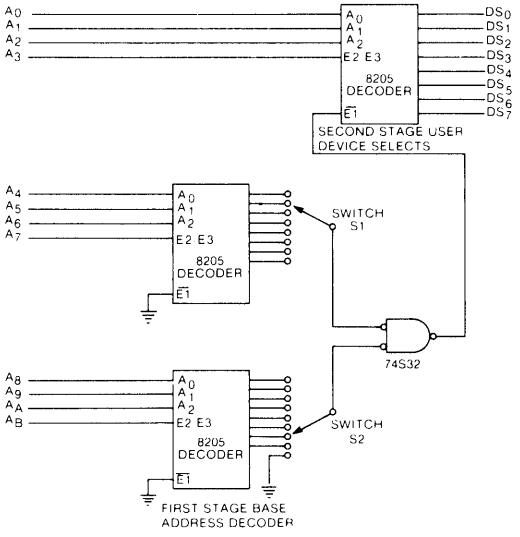


Figure 13. Two Stage Decoding Scheme

Data Bus Drivers — For user designed logic which simply receives data from the MULTIBUS data lines, this portion of the bus interface logic may only consist of buffers. Buffers are required to ensure that maximum allowable bus loading is not exceeded by the user logic.

In systems where the user designed logic must place data onto the MULTIBUS data lines, three-state drivers are required. These drivers should be enabled only when a memory read command (MRDC/) or an I/O read command (IORC/) is present and the module has been addressed.

When both the read and write functions are required, parallel bidirectional bus drivers (e.g., Intel 8226, 8287, etc.) are used. A note of caution must be included for the designer who uses this type of device. A problem may arise if data hold time requirements must be satisfied for user logic following write operations. When bus commands are used to directly produce both the chip select for the bidirectional bus driver and a strobe to a latch in the user logic, removal of that signal may not provide the user's latch with adequate data hold time. Depending on the specifics of the user logic, this problem may be solved by permanently enabling the data buffer's receiver circuits and controlling only the direction of the buffers.

Control Signal Logic — The control signal logic consists of the circuits that forward the I/O and memory read/write commands to their respective destinations, provide the bus with a transfer acknowledge response, and drive the system interrupt lines.

Bus Command Lines

The MULTIBUS information transfer protocol lines (MRDC/, MWTC/, IORD/. and IOWC/) should be buffered by devices with very high speed switching. Because the bus DC requirements specify that each board may load these lines with 2.0 mA, Schottky devices are recommended. LS devices are not recommended due to their poor noise immunity. The commands should be gated

with a signal indicating the base address has been decoded to generate read and write strobes for the user logic.

Transfer Acknowledge Generation

The user interface transfer acknowledge generation logic provides a transfer acknowledge response, XACK/, to notify the bus master that write data provided by the bus master has been accepted or that read data it has requested is available on the MULTIBUS data lines. XACK/ allows the bus master to conclude its current instruction.

Since XACK/ timing requirements depend on both the CPU of the bus master and characteristics of the user logic, a circuit is needed which will provide a range of easily modified acknowledge responses.

The transfer acknowledge signals must be driven by three-state drivers which are enabled when the bus interface is addressed and a command is present.

Interrupt Signal Lines

The asynchronous interrupt lines must be driven by open collector devices with a minimum drive of 16 mA.

In a typical Non Bus Vectored Interrupt system, logic must be provided to assert and latch-up an interrupt signal. In addition to driving the MULTIBUS interrupt lines, the latched interrupt signal would be read by an I/O operation such as reading the module's status. The interrupt signal would be cleared by writing to the status register.

III. MULTIBUS™ SLAVE DESIGN EXAMPLE

A MULTIBUS slave design example has been included in this application note to reinforce the theory previously discussed. The design example is of general purpose I/O slave interface. This design example could easily be modified to be used as a slave memory interface by buffering the address signals and using the appropriate MULTIBUS memory commands. In addition, to help the reader better understand an application for an I/O slave interface, two Intel 8255A Parallel Peripheral Interface (PPI) devices are shown connected to the slave interface.

The design example is shown in both 8/16-bit version and an 8-bit version. The 8/16-bit version is an I/O interface which will permit a 16-bit master to perform 8 or 16 bit data transfers. 8-bit masters may also use the 8/16-bit version of the design example to perform 8-bit data transfers.

The 8-bit version of the design example may be used by both 8 or 16-bit masters, but will only perform 8-bit data transfers. It does not contain the circuitry required to perform 16-bit data transfers.

Both the 8/16-bit version and the 8-bit version of the design example were implemented on an iSBC 905 prototype board. The schematics for each of the examples are given in Appendices F and G.

Functional/Programming Characteristics

This section describes the organization of the slave interface from two points of view, the functional point of view and the programming characteristics. First, the principal functions performed by the hardware are identified and the general data flow is illustrated. This point of view is intended as an introduction to the detailed description provided in the next section; Theory of Operation. In the second point of view, the information needed by a programmer to access the slave is summarized.

Functional Description — The function of this I/O slave is to provide the bus interface logic for general purpose I/O functions and for two Intel 8255A Parallel Peripheral Interface (PPI) devices. Eight device selects (port addresses) are available for general purpose I/O functions. One of these device select lines is used to read and reset the state of an interrupt status flip-flop, the other seven device selects are unused in this design. An additional eight I/O device port addresses are used by the two 8255A devices; four I/O port addresses per 8255A (three I/O port address for the three parallel ports A, B, and C and the fourth I/O port address for the device control register).

Figure 14 contains a functional block diagram of the slave design example. This block diagram shows the fundamental circuit elements of a bus slave: bidirectional data bus drivers/receivers, address decoding logic and bus control logic. Also shown is the address decoding logic for the low order four bits, the interrupt logic which is selected by this decoding logic, and the two 8255A devices.

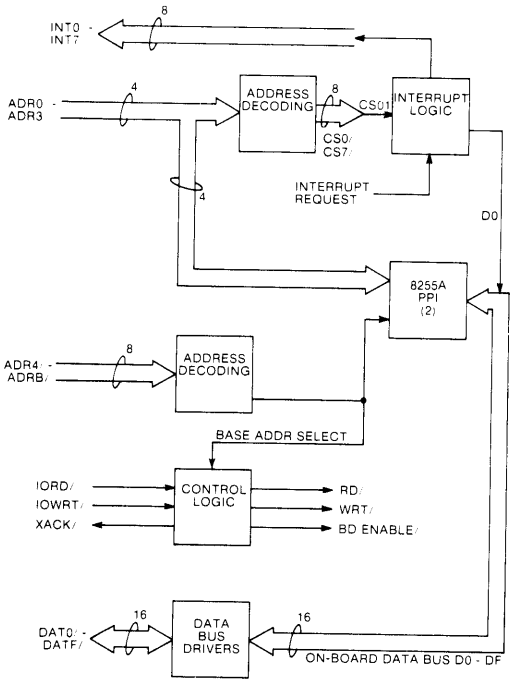


Figure 14. MULTIBUS™ Slave Design Example Functional Block Diagram

Programming Characteristics — The slave design example provides 16 I/O port addresses which may be accessed by user software. The base address of the 16 contiguous port addresses is selected by wire wrap connections on the prototype board. The wire wrap connections specify address bits ADR4/ - ADRB/. They allow the selection of a base address on any 16 byte boundary. Twelve address bits (ADR0/ - ADRB/) are used since 16-bit (8086 based) masters use 12 bits to specity I/O port addresses. If an 8 bit (8080 or 8085 based) master is used with this slave board, the high order address bits (ADR8/ - ADRB/) must not be used by the decoding circuits; a wire wrap jumper position (ground position) is provided for this.

The 16 I/O port addresses are divided into two groups of 8 port addresses by decoding address line ADR3/. Port addresses XX0 - XX7 are used for general I/O functions (XX indicates any hexidecimal digit combination). Port address XX0 is used for accessing the interrupt status flip-flop and port addresses XX1 - XX7 are not used in this example. Port addresses XX8 - XXF are used for accessing the PPIs. If port addresses XX8 - XXF are selected, then ADR0/ is used to specify which of two PPIs are selected. If the address is even (XX8, XXA, XXC, or XXE) then one PPI is selected. If the address is odd (XX9, XXB, XXD, or XXF), then the other PPI is selected. ADR1/ and ADR2/ are connected directly to the PPIs. Table 1 summarizes the I/O port addresses of the slave design example. Note that if a 16-bit master is used, it is possible to access the slave in a byte or word mode. If word access is used with port address XX8, XXA, XXC, or XXE, then 16 bit transfers will occur between the PPIs and the master. These 16 bit transfers occur because an even address has been specified and the MULTIBUS BHEN/ signal indicates that a 16-bit transfer is requested.

Theory of Operation

In the preceding section, each of the slave design example functional blocks was identified and briefly explained. This section explains how these functions are implemented. For detailed circuit information, refer to the schematics in Appendices F and G. The schematic in Appendix F is on a foldout page so that the following text may easily be related to the schematic.

The discussion of the theory of operation is divided into five segments, each of which discusses a different function performed by the MULTIBUS slave design example. The five segments are:

1. Bus address decoding
2. Data buffers
3. Control signals
4. Interrupt logic
5. PPI operation

Each of these topics are discussed with regard to the 8/16-bit version of the design example; followed by a discussion of the circuit elements which are required by the 8-bit version of the interface.

Bus Address Decoding — Bus address decoding is performed by two 8205 1 out of 8 binary decoders. One decoder (A3) decodes address bits ADR8/ - ADRB/ and the second decoder (A2) decodes address bits ADR4/ - ADR7/. The base address

Table 1

SLAVE DESIGN EXAMPLE PORT ADDRESSES

I/O PORT ADDRESS	READ	WRITE
BYTE ACCESS		
XX0	Bit 0 = Interrupt Status	Reset Interrupt Status
XX1 - XX7	Unused	Unused
XX8	Parallel Port A, Even PPI	Parallel Port A, Even PPI
XX9	Parallel Port A, Odd PPI	Parallel Port A, Odd PPI
XXA	Parallel Port B, Even PPI	Parallel Port B, Even PPI
XXB	Parallel Port B, Odd PPI	Parallel Port B, Odd PPI
XXC	Parallel Port C, Even PPI	Parallel Port C, Even PPI
XXD	Parallel Port C, Odd PPI	Parallel Port C, Odd PPI
XXE	Illegal Condition	Control, Even PPI
XXF	Illegal Condition	Control, Odd PPI
WORD ACCESS		
XX0	Bit 0 = Interrupt Status	Reset Interrupt Status
XX2 - XX6	Unused	Unused
XX8	Parallel Port A, Even and Odd PPIs	Parallel Port A, Even and Odd PPIs
XXA	Parallel Port B, Even and Odd PPIs	Parallel Port B, Even and Odd PPIs
XXC	Parallel Port C, Even and Odd PPIs	Parallel Port C, Even and Odd PPIs
XXE	Illegal Condition	Control, Even and Odd PPIs

XX = Any hex digits, assigned by jumpers; XX defines the base address.

selected is determined by the position of wire wrap jumpers. The outputs of the two decoders are ANDed together to form the BASE ADR SELECT/ signal. This signal specifies the base address for a group of 16 I/O ports. Using the wire wrap jumper positions shown in the schematic, a base address of E3 has been selected. Therefore, this MULTIBUS slave board will respond to I/O port addresses in the E30 - E3F range.

If this slave board is to be used with 8-bit MULTIBUS masters, the high order address bits must not be decoded. Therefore, the wire wrap jumper which selects the output of decoder A3 must be placed in the top (ground) position (pin 10 of gate A9 to ground).

The low order 4 address lines (ADR0/ - ADR3/) are buffered and inverted using 74LS04 inverters. These address lines are input to an 8205 for decoding a chip select for the interrupt logic; the address lines are also used directly by the PPIs. LS-Series logic is required for buffering to meet the MULTIBUS specification for I_{IL} (low level input current). S-Series or standard series logic will not meet this specification.

Address decoder A4 is used to decode addresses E30 - E37. The CS0/ output of this decoder is used to select the interrupt logic, thus I/O port address E30 is used to read and reset the interrupt latch. The remaining outputs from decoder A4 (CS1/ - CS7/) are not used in this example. They would normally be used to select other functions in a slave board with more capability. Note that in the schematic shown in Appendix G for the 8-bit version of this slave design example, the high order (ADR8/ - ADRB/) address decoder is not included and the BHEN/ signal is not used.

Data Buffers — Intel 8287 8-bit parallel bi-directional bus drivers are used for the MULTIBUS data lines DAT0/ - DATF/. In the 8/16-bit version of the slave board, three 8287 drivers are used.

When an 8-bit data transfer is requested, either driver A5, which is connected to on-board data

lines D0 - D7, or driver A6, which is connected to on-board data lines D8 - DF, is used. If a byte transfer is requested from an even address, driver A5 will be selected. If a byte transfer from an odd address is requested, driver A6 will be selected. All byte transfers take place on MULTIBUS data lines DAT0/ - DAT7/. When a word (16-bit) transfer is requested from an even address, drivers A5 and A7 will be used. Note that if a user program requests a word transfer from an odd address, 16-bit masters in the iSBC product line will actually perform two byte transfer requests.

The logic which determines the chip selection (8287 input signal OE, output enable) signals for the bus drivers uses the low order address bit (ADR0/) and the buffered Byte High Enable signal (BHENBL/). Note that the MULTIBUS signal BHEN/ has been buffered with an 74LS04 inverter. This is done to meet the bus address line loading specification. The SWAP BYTE/ signal which is generated is qualified by the BD ENBL/ signal and used to select the bus drivers.

The steering pin for the 8287 drivers is labelled T (transmit) and is driven by the signal RD. When an input (read) request is active or when neither a read or write command is being serviced, the direction of data transfer of the 8287 will be set for B to A.

The 8287 drivers are set to point IN (direction B to A) when no MULTIBUS I/O transfer command is being serviced for two reasons. First, if the driver were pointed OUT (direction A to B) and a write command occured, it would be necessary to turn the buffers IN and set the OE (output enable) signal active before the data could be transferred to the on-board bus. A possibility of a "buffer-fight" could occur in some designs if the OE signal permitted an 8287 to drive the MULTIBUS data lines momentarily before the steering signal could switch the direction of the 8287. In this case, both the MULTIBUS master and the slave would be driving the data lines; this is not recommended. (In this particular design, the steering signal will always stabilize before the OE signal becomes active.)

The second reason the driver is pointing IN when no command is present is due to the "data valid after WRITE" requirements of the 8255As. The 8255A requires that data remain on its data lines for 30 ns after the WRITE command ($\overline{WR}$ at the 8255A) is removed. This requirement will be met if the direction of the 8287 drivers is not switched when the MULTIBUS IOWC/ signal is removed (WRT/ could have been used to steer the 8287 instead of RD); and if the capacitance of the on-board data bus lines is sufficient to hold the data values on the bus after the 8287 OE signal and the 8255A PPI WRT/ signal go inactive. The on-board data bus may easily be designed such that the capacitance of the lines is sufficient to meet the 30 ns data hold time requirement. In addition, the current leakage of all devices connected to the on-board bus must be kept small to meet the 30 ns data hold time requirement.

The 8-bit version of this design example uses only one 8287 instead of the three required by the 8/16-bit version. The logic required to control the swap byte buffer is also not necessary. The chip select signal used for the 8287 is the BD ENBL/ signal.

Control Signals — The MULTIBUS control signals used by this slave design example are IORC/, IOWC/, and XACK/. IORC/ and IOWC/ are qualified by the BASE ADR SELECT/ signal to form the signals RD and WRT. RD and WRT are used to drive the interrupt logic, the PPI logic and the XACK/ (transfer acknowledge) logic.

For the XACK/ logic RD and WRT are ORed to form the BD ENBL/ signal which is inverted and used to drive the CLEAR pin of a shift register. When the slave board is not being accessed, the CLEAR pin of the shift register will be low (BD ENBL/ is high). This causes the shift register to remain cleared and all outputs of the shift register will be low. When the slave board is accessed, the CLEAR pin will be high, and the A and B inputs (which are high) will be clocked to the output pins by CCLK/. To select a delay for the XACK/ signal, a jumper must be installed from one of the shift register output pins to the 8089 tri-state driver. Each of the shift register output pins select an integer multiple of CCLK/ periods for the signal delay. Since the CCLK/ signal is asynchronous, the actual delay selected may only be specified with a tolerance of one CCLK/ period. In this example a delay of 3 - 4 CCLK/ periods was selected; with a CCLK/ period of 100 ns, the XACK/ delay would occur somewhere within the range of 300 - 400 ns from the time when the CLEAR signal goes high.

The control signal logic used in the 8-bit version of the slave design example is identical to the logic used in the 8/16-bit version.

Interrupt Logic — The interrupt logic uses a 74S74 flip-flop to latch an asynchronous interrupt request from some external logic. The Q output of the INTERRUPT REQUEST LATCH is output through an open collector gate to one of the MULTIBUS interrupt lines. The state of the INTERRUPT REQUEST LATCH is transferred to the INTERRUPT STATUS LATCH when a read command is performed on I/O port BASE ADDRESS+0 (E30 for the jumper configuration shown). The Q output of INTERRUPT STATUS LATCH is used to drive data line D0 of the on-board data bus by using an 8089 tri-state driver. If a user program performs an INPUT from I/O port E30, data bit 0 will be set to 1 if the INTERRUPT REQUEST LATCH is set.

The purpose of INTERRUPT STATUS LATCH is to minimize the possibility of the asynchronous interrupt occuring while the interrupt status is being read by a bus master. If the latch was not included in the design and an asynchronous interrupt did occur while a bus master is reading MULTIBUS data line DAT0/, a data buffer on the master could go into a meta-stable state. By adding the extra latch, which is clocked by the IORD/ command for I/O port E30, the possibility of data line DAT0/ changing during a bus master read operation is eliminated.

The INTERRUPT REQUEST LATCH is cleared when a user program performs an OUTPUT to I/O port E30.

This interrupt structure assumes that several interrupt sources may exist on the same MULTIBUS interrupt line (for example, INT3/). When the MULTIBUS master gets interrupted, it must poll the possible sources of the interrupt received and after determining the source of the interrupt, it must clear the INTERRUPT REQUEST LATCH for that particular interrupt source.

The interrupt logic for the 8-bit version of the design example is identical to the interrupt logic of the 8/16-bit version of the design example.

PPI Operation — Two 8255A Parallel Peripheral Interface (PPI) devices are shown interfaced to the slave design example logic. One PPI is connected to the on-board data bus lines D0 - D7 and is addressed with the even I/O port addresses E38, E3A, E3C, and E3E. The second PPI is connected to data bus lines D8 - DF and is addressed with the odd I/O port addresses E39, E3B, E3D, and E3F. The even or odd I/O port selection is controlled by using the ADR0 address line in the chip select term of the PPIs. In addition, the odd PPI (A11) is selected when the BHENBL term is high. This occurs when the MULTIBUS signal BHEN/ is low indicating that a word (16-bit) I/O instruction is being executed. When a word I/O instruction is executed, both PPIs will perform the I/O operation specified.

The specifications of the 8255A device state that the address lines A0 and A1 and the chip select lines must be stable before the $\overline{RD}$ or $\overline{WR}$ lines are activated. The MULTIBUS specification address set-up time of 50 ns and the short gate propagation delays in this design assure that the address lines are stable before $\overline{RD}$ or $\overline{WR}$ are active.

The data hold requirements of the 8255A were discussed in a previous section. The 8255A specification states that data will be stable on the data bus lines a maximum of 250 ns after a READ command. This specification was used to select the delay for the XACK/ signal.

The PPI operation for the 8-bit version of the design example is slightly different than that used for the 8/16-bit version. The chip select signal for the bottom PPI does not use the BHENBL term since 16-bit data transfers are not possible with an 8-bit I/O slave board. Also, the chip select and address signals have been swapped so the top PPI occupies I/O address range X8 - XB, and the bottom PPI occupies I/O address range XC - XF (X is the base address of the 8-bit version). This swapping of the address lines was not necessary; however, it was thought to be more convenient to access the PPIs in two groups of 4 contiguous I/O port addresses.

IV. SUMMARY

This application note has shown the structure of the Intel MULTIBUS system bus. The structure supports a wide range of system modules from the Intel OEM Microcomputer Systems product line that can be extended with the addition of user designed modules. Because the user designed modules are no doubt unique to particular applications, a goal of this application note has been to describe in detail the singular common element - the bus interface. Material has also been presented to assist the systems designer to understanding the bus functions so that successful systems integration can be achieved.

APPENDIX A

PIN ASSIGNMENT OF BUS SIGNALS ON MULTIBUS BOARD P1 CONNECTOR

		(COMPONENT SIDE)			(CIRCUIT SIDE)	
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Signal GND
	3	+5V	+5Vdc	4	+5V	+5Vdc
	5	+5V	+5Vdc	6	+5V	+5Vdc
	7	+12V	+12Vdc	8	+12V	+12Vdc
	9	-5V	-5Vdc	10	-5V	-5Vdc
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15	BPRN/	Bus Pri. In	16	BPRO/	Bus Pri. Out
	17	BUSY/	Bus Busy	18	BREQ/	Bus Request
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknowledge	24	INH1/	Inhibit 1 disable RAM
BUS CONTROLS AND ADDRESS	25		Reserved	26	INH2/	Inhibit 2 disable PROM or ROM
	27	BHEN/	Byte High Enable	28	AD10/	
	29	CBRQ/	Common Bus Request	30	AD11/	Address
	31	CCLK/	Constant Clk	32	AD12/	Bus
	33	INTA/	Intr Acknowledge	34	AD13/	
INTERRUPTS	35	INT6/	Parallel	36	INT7/	Parallel
	37	INT4/	Interrupt	38	INT5/	Interrupt
	39	INT2/	Requests	40	INT3/	Requests
	41	INT0/		42	INT1/	
ADDRESS	43	ADRE/		44	ADRF/	
	45	ADRC/		46	ADRD/	
	47	ADRA/	Address	48	ADRB/	Address
	49	ADR8/	Bus	50	ADR9/	Bus
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
DATA	59	DATE/		60	DATF/	
	61	DATC/		62	DATD/	
	63	DATA/	Data	64	DATB/	Data
	65	DAT8/	Bus	66	DAT9/	Bus
	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77		Reserved	78		Reserved
	79	-12V	-12Vdc	80	-12V	-12Vdc
	81	+5V	+5Vdc	82	+5V	+5Vdc
	83	+5V	+5Vdc	84	+5V	+5Vdc
	85	GND	Signal GND	86	GND	Signal GND