Application Notes

on the

Atari Computer System Interface (ACSI)

The Atari Corporation

Sunnyvale, California

27 September 1985

CONFIDENTIAL

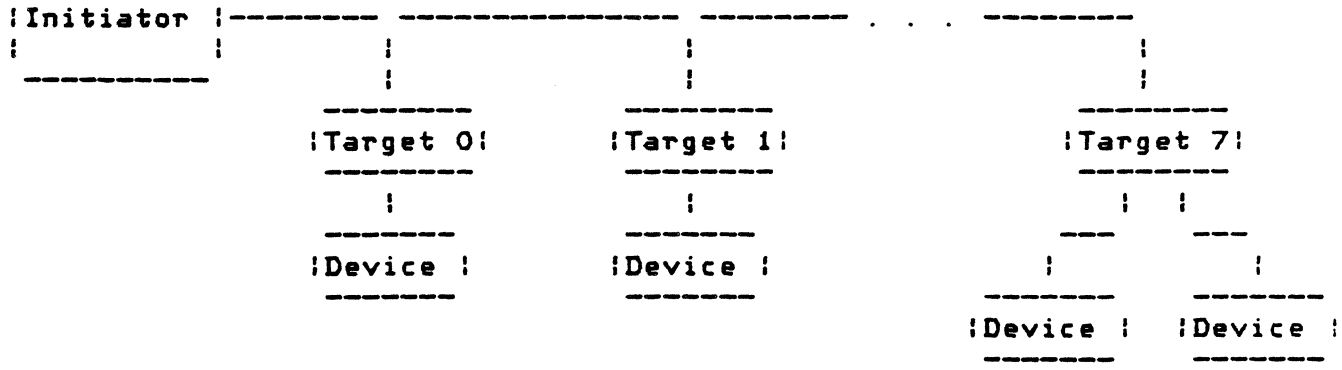
Table of Contents

THE SCOPE OF THIS DOCUMENT is limited to a set of rough application notes on the Atari Computer System Interface. This is a preliminary document and is subject to change without notice.

1. ACSI Bus

- o control signals and a bidirectional bus.
- o target does not receive a command and hold it pending controller ready -- an immediate DEVICE NOT READY error must be sent or the initiator will time out and assume controller nonexistent.
- o controller self test -- recalibrate, ram check, rom checksum, etc.
- o self test always performed following reset -- eliminates need for self test command.
- o initiator could time out (duration to be determined) on a command and reset the target.
- o once the status byte is returned the bus is free.
- o maximum eight bus ports.
- o data transfer rate is up to 8 Mbit/sec.

----- ACSI Bus Topology ----------



----- Control and Data Signals ----------

Mnemonic	Name	Characteristics
_RST	Reset	TTL levels, active low.
A1	Address 1	TTL levels.
_IRQ	Interrupt Request	TTL levels, active low, 1 Kohm pullup on initiator side.
_CS	Chip Select	TTL levels, active low.
R/_W	Read/Write	TTL levels.
_DRQ	Data Request	TTL levels, active low, 1 Kohm pullup on initiator side.
_ACK	Acknowledge	TTL levels, active low.
DATA	Data Bus (0-7)	TTL levels.

----- Initiator ACSI Port Pin Assignments ----------

INITIATOR DB 19S		TARGET
1	<--- Data 0 --->	
2	<--- Data 1 --->	
3	<--- Data 2 --->	
4	<--- Data 3 --->	
5	<--- Data 4 --->	
6	<--- Data 5 --->	
7	<--- Data 6 --->	
8	<--- Data 7 --->	
9	---- Chip Select ---->	
10	<--- Interrupt Request ----	
11	---- Ground ----	
12	---- Reset ---->	
13	---- Ground ----	
14	---- Acknowledge ---->	
15	---- Ground ----	
16	---- A1 ---->	
17	---- Ground ----	
18	---- Read/Write ---->	
19	<--- Data Request ----	

2. ACSI Compliance

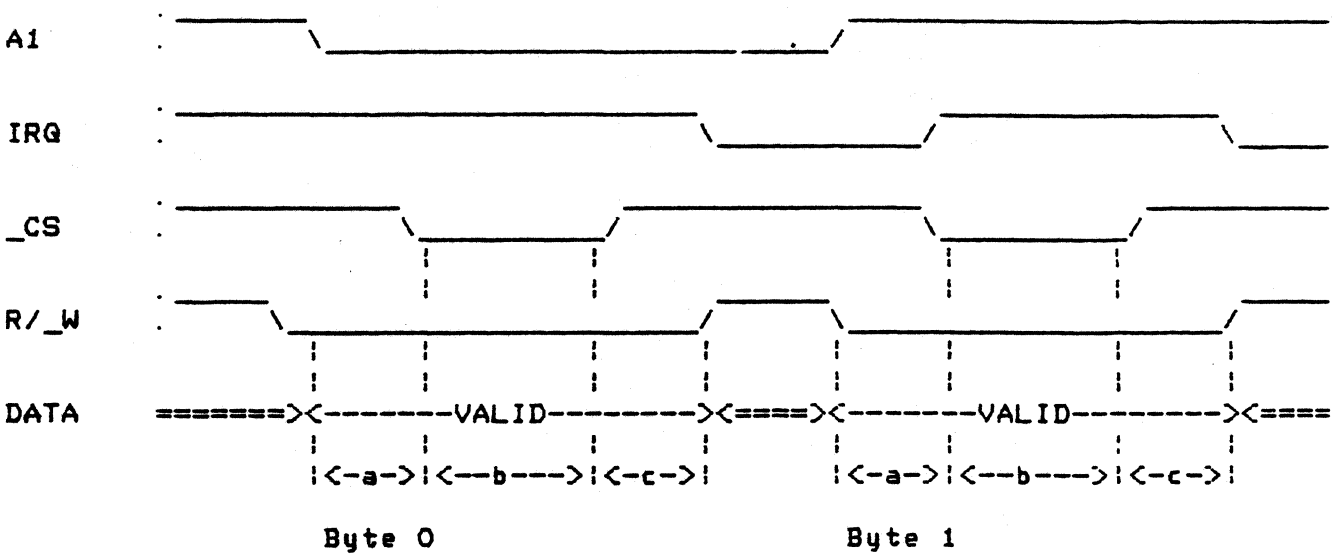
2.1. Level 1

- target will speak only when spoken to.
- listen to bus during idle -- no disconnect.
- abort initiator via interrupt.
- abort target via reset -- software reset must be provided in initiator.
- RESET HOLD TIME is 12 microseconds.
- reset has highest bus priority.
- reset cannot be asserted by a target whether active or inactive.
- 100 milliseconds before initiator times out on target acknowledgement.
- CAVEAT: if an initiator prematurely issues a command while the target is executing a command, then the results are unpredictable.
- device driver in initiator will wait until status byte is returned -- otherwise time out (TBD) and reset target.
- after receipt of command byte, transaction belongs to controller.
- target has complete control of bus until status byte is returned.
- each target should have a user select controller number.

HARDWARE.

----- Command Phase ----------

Data direction: FROM initiator TO target.

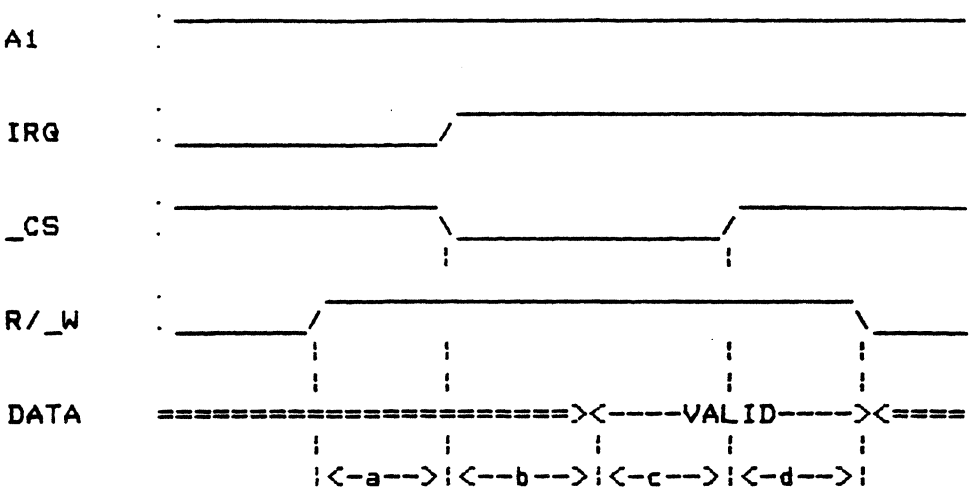


Timing

a) 60 ns (max)
b) 250 ns (max)
c) 20 ns (max)

----- Status Phase ----------

Data direction: FROM target TO initiator.



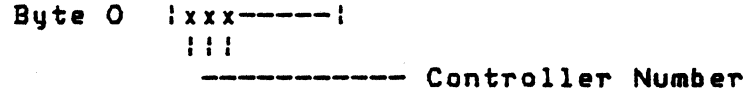
Byte 0

Timing

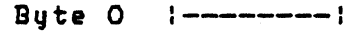
a) 50 ns (max)
b) 150 ns (max)
c) 100 ns (max)
d) 80 ns (max)

SOFTWARE.

----- Controller Select Byte ----------



----- Completion Status Byte ----------

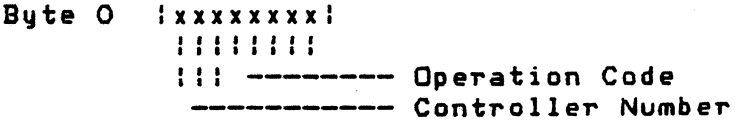


2.2. Level 2

- o include Level 1.
- o TEST UNIT READY command is used as a poll.
- o NO ERROR is to be interpreted as controller ready.
- o DEVICE NOT READY is to be interpreted as controller not ready.

SOFTWARE.

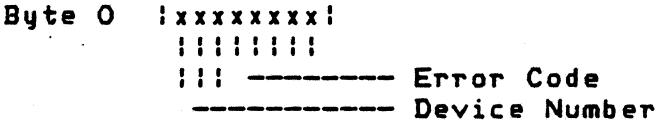
----- Command Descriptor Byte ----------



----- Command Summary Table ----------

OpCode	Command
0x00	Test Unit Ready

----- Completion Status Byte ----------



Device Errors

0x00 No Error
0x04 Device Not Ready

Miscellaneous Errors

0x30 Controller Self Test Failed

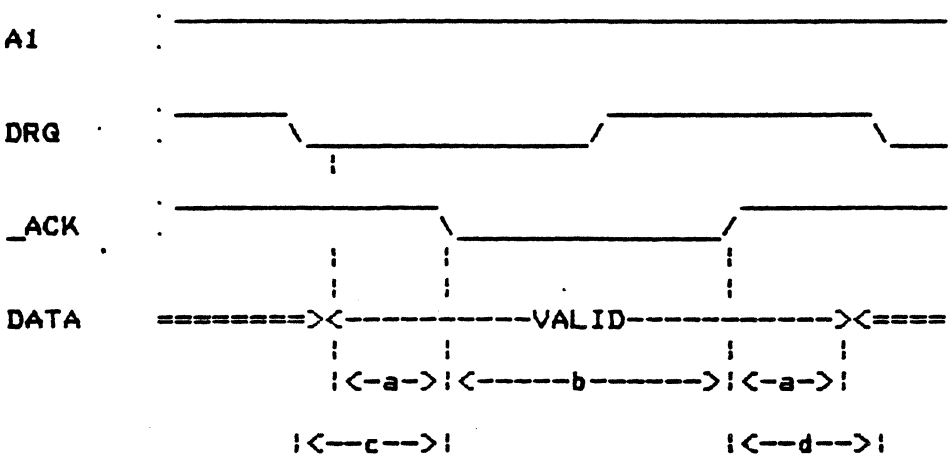
2.3. Level 3

o include Level 1 and Level 2.

HARDWARE.

----- Data Out Phase ----------

Data direction: FROM initiator TO target.

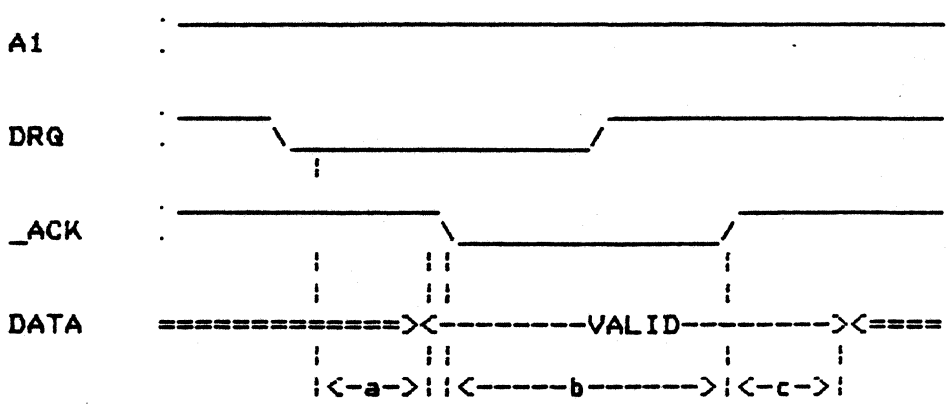


Timing

a) 60 ns (max)
b) 250 ns (max)
c) 240 ns (max)
d) 240 ns (min)

----- Data In Phase ----------

Data direction: FROM target TO initiator.

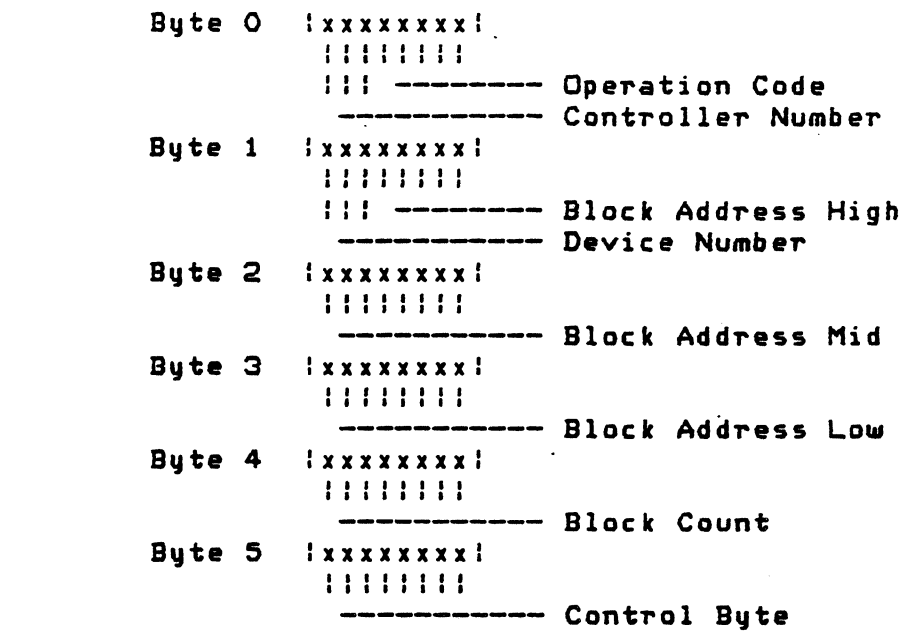


Timing

a) 60 ns (max)
b) 250 ns (max)
c) 50 ns (min)

SOFTWARE.

----- ACSI Command Descriptor Block ----------



----- Command Summary Table ----------

OpCode	Command	
0x00	Test Unit Ready	
0x08	Read	*
0x0a	Write	*
0x0b	Seek	
0x1a	Mode Sense	

* multisector transfer with implied seek

Command Errors

0x20	Invalid Command
0x21	Invalid Address
0x23	Volume Overflow
0x24	Invalid Argument
0x25	Invalid Device Number

3. ACSI Initiator

- must transfer data in 16 byte increment blocks.
- must use ST BIOS system variable flock (see A Hitchhiker's Guide to the BIOS).

----- Initiator Handshake Sequence ----------

- load DMA Base Address Register.
- toggle Write/_Read to clear status (DMA Mode Control Register).
- select DMA read or write (DMA Mode Control Register).
- select DMA Sector Count Register (DMA Mode Control Register).
- load DMA Sector Count Register (DMA operation trigger).
- select controller internal command register (DMA Mode Control Register).
- issue controller select byte by clearing A0 to 0.
- set A0 to 1 for remaining command bytes.
- after last command byte select controller (DMA Mode Control Register).
- DMA active until sector count is zero (DMA Status Register, do not poll during DMA active).
- check DMA error status (DMA Status Register).
- check controller status byte.

```
loadable        equ     1               ; nonzero for loadable driver

*-------------------------------------------------------------------------
*                                                                        :
*       ST SASI hard disk driver                                         :
*       (C)1985 Atari Corp.                                              :
*                                                                        :
*-----                                                                   :
*  9-Apr-1985 lmd       Hacked it up.  "Gee, it seems to work ..."       :
* 14-Apr-1985 lmd       linked with BIOS (***FOR NOW***)                 :
* 20-Apr-1985 lmd       hacked for WD controller (now, wired...)         :
* 24-Jun-1985 jwt       hacked for Adaptec, new kludge board             :
* 01-Jul-1985 jwt       seems to work, add more formatting and more      :
*                        detailed error reporting                        :
* 22-Jul-1985 jwt       change timing of wdc/wd1 at start of command,    :
*                        added extra move.w $8a,wd1 to change A1         :
* 23-Jul-1985 jwt       use a move.l instruction for all wdc/wd1 write   :
*                        pairs since it changes A1 quickly enough that   :
*                        the (old) DMA chip does not incorrectly         :
*                        generate two chip selects                       :
*                                                                        :
*-------------------------------------------------------------------------


flock           equ     $43e            ; FIFO lock variable
hdv_init        equ     $46a            ; hdv_init()
hdv_bpb         equ     $472            ; hdv_bpb(dev)
hdv_rw          equ     $476            ; hdv_rw(rw, buf, count, recno, dev)
hdv_boot        equ     $47a            ; hdv_boot()
hdv_mediach     equ     $47e            ; hdv_mediach(dev)
_drvbits        equ     $4c2            ; block device bitVector
_dskbufp        equ     $4c6            ; pointer to common disk buffer

nretries        equ     3               ; #retries-1


* ---------------- Installer ----------------
        .globl  i_sasi
i_sasi: bra     i_sasi2

        dc.b    '@(#)ahdx v0.04',$0d,$0a,0,$1A


* ---------------- Front End ----------------


*+
* LONG hbpb(dev) - return ptr to BPB (or NULL)
*
* Passed:       dev     4(sp).W
*
*-
hbpb:
        move.w  4(sp),d0                ; d0 = devno
        move.l  o_bpb,a0                ; a0 -> pass-through vector
```

```
        lea     _sasi_bpb(pc),a1        ; a1 -> our handler
        bra     check_dev               ; do it


*+
* LONG rw(rw, buf, count, recno, dev)
*
* Passed:       dev     $e(sp).W
*               recno   $c(sp).W
*               count   $a(sp).W
*               buf     6(sp).L
*               rw      4(sp).W
*
*-
hrw:
        move.w  $e(sp),d0               ; d0 = devno
        move.l  o_rw,a0                 ; a0 -> pass-through vector
        lea     _sasi_rw(pc),a1         ; a1 -> our handler
        bra     check_dev               ; do it


*+
* LONG mediach(dev)
*
* Passed:       dev     4(sp).W
*
*-
hmediach:
        move.w  4(sp),d0                ; d0 = devno
        move.l  o_mediach,a0            ; a0 -> pass-through vector
        lea     _sasi_mediach(pc),a1    ; a1 -> our handler


*+
* check_dev - use handler, or pass vector through
*
* Passed:       d0.w = device#
*               a0 ->  old handler
*               a1 ->  new handler
*               a5 ->  $0000 (zero-page ptr)
*
* Jumps-to:     (a1) if dev in range for this handler
*               (a0) otherwise
*
*-
check_dev:
        cmp.w   #2,d0                   ; devnos match?
        bne     chkd_f                  ; (no)
        move.l  a1,a0                   ; yes -- follow success vector
chkd_f: jmp     (a0)                    ; do it


* ---------------- Medium level driver ----------------
```

```
*+
* _sasi_init - initialize SASI dev
* Passed:       nothing
* Returns:      d0 < 0:  error
*               d0 ==0:  success
* function performed by _hinit.... and the assembler won't
*  let me have a forward reference here
*-
*       .globl  _sasi_init
*_sasi_init: equ         _hinit

*+
* _sasi_bpb - return BPB for hard drive
* Synopsis:     LONG _sasi_bpb(dev)
*               WORD dev;
*
* Returns:      NULL, or a pointer to the BPB buffer
*
*-
        .globl  _sasi_bpb
_sasi_bpb:
        move.l  #thebpb,d0
        rts


*+
* _sasi_rw - read/write hard sectors
* Synopsis:     _sasi_rw(rw, buf, count, recno, dev)
*
* Passed:       dev     $e(sp).W
*               recno   $c(sp).W
*               count   $a(sp).W
*               buf     6(sp).L
*               rw      4(sp).W         ; non-zero -> write
*
*-
        .globl  _sasi_rw
_sasi_rw:
        move.w  #nretries,retrycnt      ; setup retry counter

sasrw1: moveq   #0,d0                   ; coerce word to long, unsigned
        move.w  $c(sp),d0               ; sect.L
        move.w  $a(sp),d1               ; count.W
        move.l  6(sp),d2                ; buf.L
        move.w  4(sp),d3                ; rw

        clr.w   -(sp)                   ; dev = 0
        move.l  d2,-(sp)                ; buf
        move.w  d1,-(sp)                ; count
        move.l  d0,-(sp)                ; sect
        tst.w   d3                      ; read or write?
        bne     sasrw3                  ; (write)
        bsr     _hread                  ; read sectors
        bra     sasrw2
sasrw3: bsr     _hwrite                 ; write sectors
```

```
sasrw2: add.w   #12,sp                  ; (cleanup stack)
        tst.l   d0                      ; errors?
        beq     sasrwr                  ; no -- success
        subq.w  #1,retrycnt             ; drop retry count and retry
        bpl     sasrw1

sasrwr: rts


*+
* _sasi_mediach - see if hard disk media has changed (it never does)
* Synopsis:     _sasi_mediach(dev)
*               WORD dev;
*
* Returns:      0L
*
*-
        .globl  _sasi_mediach
_sasi_mediach:
        clr.l   d0
        rts


*+
* BPB for 10MB drive
* Approximate only.  Tweak me.
*
*-
thebpb: dc.w    512                     ; #bytes/sector
        dc.w    2                       ; #sectors/cluster
        dc.w    1024                    ; #bytes/cluster
        dc.w    16                      ; rdlen (256 root files) (in sectors)
        dc.w    41                      ; FATsiz (10300 FAT entries) (sectors)
        dc.w    42                      ; 2nd FAT start
        dc.w    99                      ; data start (in sectors)
        dc.w    10300                   ; #clusters (approximate here)
        dc.w    1                       ; flags (16-bit FATs)


* ---------------- Low-level driver ----------------

*----- Globals
flock           equ     $43e            ; FIFO lock variable
_hz_200         equ     $4ba            ; 200hz system ticker

*----- Hardware:
wdc             equ     $ff8604
wdl             equ     $ff8606
wdcwdl          equ     wdc             ; used for long writes
dmahi           equ     $ff8609
dmamid          equ     dmahi+2
```

```
dmalow          equ     dmamid+2
gpip            equ     $fffa01


*----- Tunable:
ltimeout                equ     $80000          ; long-timeout
stimeout                equ     $80000          ; short-timeout


*+
* LONG _qdone() - Wait for operation complete
* Passed:       nothing
*
* Returns:      EQ: no timeout
*               MI: timeout condition
*
* Uses:         D0
*-
_qdone:
        move.l  #ltimeout,tocount
qd1:    subq.l  #1,tocount              ; drop timeout count
        bmi     qdq                     ; (i give up, return NE)
        move.b  gpip,d0                 ; interrupt?
        and.b   #$20,d0
        bne     qd1                     ; (not yet)

        moveq   #0,d0                   ; return EQ (no timeout)
qdq:    rts


*+
* WORD _endcmd()
* Wait for end of SASI command
* Passed:       d0 value to be written to wd1
*
* Returns:      EQ: success (error code in D0.W)
*               MI: timeout
*               NE: failure (SASI error code in D0.W)
*
* Uses:         d0,d1
*-
_endcmd: move   d0,d1                   ; preserve wd1 value

        bsr     _qdone                  ; wait for operation complete
        bmi     endce                   ; (timed-out, so complain)

        move.w  d1,wd1
        nop
        move.w  wdc,d0                  ; get the result
        and.w   #$00ff,d0               ; (clean it up), if non-zero should

endce:  rts                             ;  do a ReadSense command to learn more


*+
* _hinit(dev)
```

```
* WORD dev;
* Initialize hard disk
*
* Returns:      -1 if hard disk not there
*
*-
        .globl  _sasi_init
_sasi_init:
_hinit:
        pea     actur                   ; push test unit read command block adr
        bsr     _dosahdxc
        addq.l  #4,sp
        rts

*-
* _hread(sectno, count, buf, dev)
* LONG sectno;          4(sp)
* WORD count;           8(sp)
* LONG buf;            $a(sp)  $b=high, $c=mid, $d=low
* WORD dev;            $e(sp)
*
* Returns:      -1 on timeout
*               0 on success
*               nonzero on error
*
*-
        .globl  _hread
_hread:
        st      flock                   ; lock FIFO

        move    #$88,wdl
        move.l  #$0008008a,wdcwdl       ; 08 wdc, 8a wdl

        move.l  $a(sp),-(sp)            ; set DMA address
        bsr     _setdma
        addq    #4,sp

        bsr     _setss                  ; set sector and size
        bmi     _hto

        move.w  #$190,wdl
        nop
        move.w  #$90,wdl
        nop
        move.w  8(sp),wdc               ; write sector count to DMA chip
        nop
        move.w  #$8a,wdl
        nop
        move.l  #$00000000,wdcwdl       ; control byte  0 wdc 0 wdl

        move.w  #$8a,d0
        bsr     _endcmd

hrx:    bra     _hdone                  ; cleanup after IRQ
```

```
*-
* _hwrite(sectno, count, buf, dev)
* LONG sectno;              4(sp)
* WORD count;               8(sp)
* LONG buf;                $a(sp)   $b=high, $c=mid, $d=low
* WORD dev;                $e(sp)
*
*-
        .globl  _hwrite
_hwrite:
        st      flock                   ; lock FIFO

        move.l  $a(sp),-(sp)            ; set DMA address
        bsr     _setdma
        addq    #4,sp

        move.w  #$88,wdl
        move.l  #$000a008a,wdcwdl       ; 0a wdc 8a wdl

        bsr     _setss
        bmi     _hto

        move.w  #$90,wdl
        nop
        move.w  #$190,wdl
        nop
        move.w  8(sp),wdc               ; sector count for DMA ct
        nop
        move.w  #$18a,wdl
        nop
        move.l  #$00000100,wdcwdl

        move.w  #$18a,d0
        bsr     _endcmd

hwx:    bra     _hdone                  ; cleanup after IRQ


*+
* _wd_format - format WD hard disk
* Passed:       nothing
* Returns:      0, or -N
* Uses:         <..?..>
*
*-
        .globl  _wd_format
_wd_format: lea acfmt,a0                ; pick up pointer to the
        clr.w   d0
        st      flock                   ; lock FIFO
        move.w  #$88,wdl
        move.b  (a0)+,d0                ; get the command byte
        swap    d0
        move.w  #$8a,d0
        move.l  d0,wdc                  ; byte wdc 8a wdl

        moveq   #(5-1),d1               ; write remaining 5 byte
```

```
nt1:    bsr     _qdone                  ;  (presumes only one unit)
        bmi     _hto
        move.b  (a0)+,d0                ; next byte of command
        swap    d0
        move.w  #$8a,d0
        move.l  d0,wdcwd1
        dbra    d1,fmt1

nt2:    btst    #5,gpip                 ; wait (forever) for completion
        bne     fmt2

        move.w  wdc,d0                  ; get the status
        andi.w  #$00FF,d0               ; only low byte is significant

        bra     _hdone                  ; cleanup after IRQ

*
 _wd_setup - setup parameters for WD hard disk

*
        .globl  _wd_setup
wd_setup:
        st      flock
        pea     adap_parms(pc)
        bsr     _setdma
        addq    #4,sp

        move.w  #$88,wd1
        move.l  #$0015008a,wdcwd1       ; mode select command  15 wdc 8a wd1

        bsr     _qdone
        bmi     wdx
        move.l  #$0000008a,wdcwd1
        bsr     _qdone
        bmi     wdx
        move.l  #$0000008a,wdcwd1
        bsr     _qdone
        bmi     wdx
        move.l  #$0000008a,wdcwd1
        bsr     _qdone
        bmi     wdx
        move.l  #$0016008a,wdcwd1       ; 22 bytes of parameters

        bsr     _qdone
        bmi     wdx
        move.w  #$90,wd1                ; reset the DMA chip
        nop
        move.w  #$190,wd1
        nop
        move.w  #$01,wdc                ; 1 sector of DMA (actually less)
        nop
        move.w  #$18a,wd1
        nop
        move.l  #$00000100,wdcwd1       ; control byte

        move.w  #$18a,d0                ; wd1 value
```

```
        bsr     _endcmd
wdx:    bra     _hdone

*--- parameters for 10MB WD
adap_parms: dc.b $00,$00,$00,$08,$00,$00,$00,$00,$00,$00
        dc.b    $02,$00,$01,$02,$62,$02,$01,$00,$01,$00,$00,$02


*+
* LONG _dosahdxc( addr ) BYTE *addr;
*       do a simple (no DMA) ahdx command
*-
        .globl  _dosahdxc
_dosahdxc: movea.l 4(sp),a0             ; pick up pointer to the command block
        clr.w   d0
        st      flock                   ; lock FIFO
        move.w  #$88,wdl
        move.b  (a0)+,d0                ; get the command byte
        swap    d0
        move.w  #$8a,d0
        move.l  d0,wdcwdl               ; send it to the controller

        moveq   #(5-1),d1               ; write remaining 5 bytes of command
dosac1: bsr     _qdone                  ;  (presumes only one unit)
        bmi     _hto
        move.b  (a0)+,d0                ; next byte of command
        swap    d0
        move.w  #$8a,d0
        move.l  d0,wdcwdl
        dbra    d1,dosac1

        bsr     _qdone                  ; wait for the command to complete
        bmi     _hto

        move.w  wdc,d0                  ; get the status
        andi.w  #$00FF,d0               ; only low byte is significant

        bra     _hdone                  ; cleanup after IRQ


*+
* void _setdma(addr)
* LONG addr;
*-
_setdma:
        move.b  7(sp),dmalow
        move.b  6(sp),dmamid
        move.b  5(sp),dmahi
        rts

*+
* WORD _setss  -- set sector number and number of sectors
*-

_setss: move.w  #$8a,wdl
```

```
        bsr     _qdone                  ; wait for controller to take command
        bmi     setsse

        move.b  9(sp),d0                ; construct sector#
        move.b  $e(sp),d1               ; ORed with devno
        lsl.b   #5,d1
        or.b    d1,d0
        swap    d0
        move.w  #$008a,d0
        move.l  d0,wdcwd1               ; write MSB sector# + devno
        bsr     _qdone
        bmi     setsse

        move.b  10(sp),d0               ; write MidSB sector#
        swap    d0
        move.w  #$008a,d0
        move.l  d0,wdcwd1
        bsr     _qdone
        bmi     setsse

        move.b  11(sp),d0               ; write LSB sector#
        swap    d0
        move.w  #$008a,d0
        move.l  d0,wdcwd1
        bsr     _qdone
        bmi     setsse

        move.w  12(sp),d0               ; write sector count
        swap    d0
        move.w  #$008a,d0
        move.l  d0,wdcwd1
        bsr     _qdone

setsse: rts

_hto:   moveq   #-1,d0                  ; indicate timeout
_hdone: move.w  #$80,wd1                ; Landon's code seems to presume we
        nop                             ;  put this back to $80
        tst.w   wdc
        clr     flock                   ; NOW, signal that we are done
        rts

savssp:         dc.l    1               ; (saved SSP)
tocount:        dc.l    1               ; timeout counter
retrycnt:       dc.w    1               ; retry counter
o_init:         dc.l    1
o_bpb:          dc.l    1
o_rw:           dc.l    1
o_mediach:      dc.l    1

i_sasi2: nop

 ifne loadable
        clr.l   -(sp)                   ; it's a bird...
        move.w  #$20,-(sp)              ;    ... it's a plane ...
        trap    #1                      ;       ... no, its:
```

```
        addq    #6,sp                   ; SOOUPERUSER!
        move.l  d0,savssp               ; "Faster than a prefetched opcode..."
 endc

        bsr     _sasi_init              ; kick controller
        tst.w   d0
        bne     isase                   ; punt -- disk didn't respond correctly

        clr.l   d0
        or.l    _drvbits,d0             ; include C: bit in devVector
        or.l    #$4,d0
        move.l  d0,_drvbits

        clr.l   a5                      ; zeropage ptr
        move.l  hdv_bpb(a5),o_bpb               ; save old vectors
        move.l  hdv_rw(a5),o_rw
        move.l  hdv_mediach(a5),o_mediach

        move.l  #hbpb,hdv_bpb(a5)               ; install our new ones
        move.l  #hrw,hdv_rw(a5)
        move.l  #hmediach,hdv_mediach(a5)

isasq:  nop                             ; stupid assembler

 ifne loadable
        move.l  savssp,-(sp)            ; become a mild mannered user process
        move.w  #$20,-(sp)
        trap    #1
        addq    #6,sp
 endc

 ifne loadable
        move.w  #0,-(sp)        ; exit code
        move.l  #((i_sasi2-i_sasi)+$0100),-(sp) ; save code, data, & basepage
        move.w  #$31,-(sp)      ; terminate and stay resident
        trap    #1              ; should never come back...
 endc

        rts

isase:  lea     nodmsg,a0
        bsr     msg

 ifne loadable
        move.l  savssp,-(sp)            ; become a mild mannered user process
        move.w  #$20,-(sp)
        trap    #1
        addq    #6,sp
 endc

        move.w  #1,-(sp)                        ; flag error status
        move.w  #$4c,-(sp)              ; terminate
        trap    #1

msg:    move.l  a0,-(sp)
        move.w  #9,-(sp)                ; print null terminated string
```

```
        trap    #1
        addq.l  #6,sp
        rts

actur:  dc.b    0,0,0,0,0,0             ; atari command:  test unit ready
acfmt:  dc.b    4,0,0,0,1,0             ;                 format disk

nodmsg: dc.b    'No AHDX disk response. ',$0d,$0a,0

        .even

        end
```

MOTOROLA

MC6850 (1.0 MHz)
MC68A50 (1.5 MHz)
MC68B50 (2.0 MHz)

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

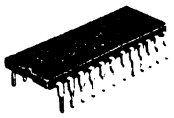
The MC6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the MC6800 Microprocessing Unit.

The bus interface of the MC6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bidirectional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking. The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation, three control lines are provided. These lines allow the ACIA to interface directly with the MC6860L 0-600 bps digital modem.

- 8- and 9-Bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Programmable Control Register
- Optional ÷1, ÷16, and ÷64 Clock Modes
- Up to 1.0 Mbps Transmission
- False Start Bit Deletion
- Peripheral/Modem Control Functions
- Double Buffered
- One- or Two-Stop Bit Operation

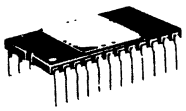
MOS
(N-CHANNEL, SILICON-GATE)

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER



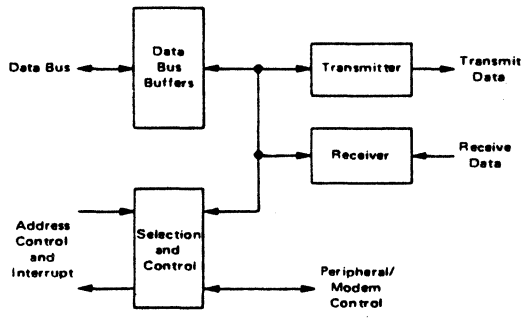
S SUFFIX
CERDIP PACKAGE
CASE 623

P SUFFIX
PLASTIC PACKAGE
CASE 709

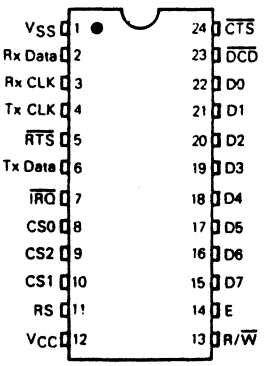


L SUFFIX
CERAMIC PACKAGE
CASE 716

MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER BLOCK DIAGRAM



PIN ASSIGNMENT



967

MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6850, MC68A50, MC68B50 MC6850C, MC68A50C, MC68B50C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic Ceramic Cerdip	θ_{JA}	 120 60 65	°C/W

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{PORT}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts − Chip Internal Power

P_{PORT} = Port Power Dissipation, Watts − User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, $T_A = T_L$ to T_H unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage		V_{IH}	V_{SS} + 2.0	–	V_{CC}	V
Input Low Voltage		V_{IL}	V_{SS} − 0.3	–	V_{SS} + 0.8	V
Input Leakage Current (V_{in} = 0 to 5.25 V)	R/W, CS0, CS1, $\overline{CS2}$, Enable RS, Rx D, Rx C, $\overline{CTS}$, $\overline{DCD}$	I_{in}	–	1.0	2.5	μA
Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V)	D0-D7	I_{TSI}	–	2.0	10	μA
Output High Voltage (I_{Load} = −205 μA, Enable Pulse Width < 25 μs) (I_{Load} = −100 μA, Enable Pulse Width < 25 μs)	D0-D7 Tx Data, $\overline{RTS}$	V_{OH}	V_{SS} + 2.4 V_{SS} + 2.4	– –	– –	V
Output Low Voltage (I_{Load} = 1.6 mA, Enable Pulse Width < 25 μs)		V_{OL}	–	–	V_{SS} + 0.4	V
Output Leakage Current (Off State) (V_{OH} = 2.4 V)	$\overline{IRQ}$	I_{LOH}	–	1.0	10	μA
Internal Power Dissipation (Measured at $T_A = T_L$)		P_{INT}	–	300	525	mW
Internal Input Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz) E, Tx CLK, Rx CLK, R/W, RS, Rx Data, CS0, CS1, $\overline{CS2}$, $\overline{CTS}$, $\overline{DCD}$	D0-D7	C_{in}	– –	10 7.0	12.5 7.5	pF
Output Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	RTS, Tx Data $\overline{IRQ}$	C_{out}	– –	– –	10 5.0	pF

SERIAL DATA TIMING CHARACTERISTICS

Characteristic		Symbol	MC6850		MC68A50		MC68B50		Unit
			Min	Max	Min	Max	Min	Max	
Data Clock Pulse Width, Low (See Figure 1)	÷ 16, ÷ 64 Modes	PW_{CL}	600	–	450	–	280	–	ns
	÷ 1 Mode		900	–	650	–	500	–	
Data Clock Pulse Width, High (See Figure 2)	÷ 16, ÷ 64 Modes	PW_{CH}	600	–	450	–	280	–	ns
	÷ 1 Mode		900	–	650	–	500	–	
Data Clock Frequency	÷ 16, ÷ 64 Modes	f_C	–	0.8	–	1.0	–	1.5	MHz
	÷ 1 Mode		–	500	–	750	–	1000	kHz
Data Clock-to-Data Delay for Transmitter (See Figure 3)		t_{TDD}	–	600	–	540	–	460	ns
Receive Data Setup Time (See Figure 4)	÷ 1 Mode	t_{RDS}	250	–	100	–	30	–	ns
Receive Data Hold Time (See Figure 5)	÷ 1 Mode	t_{RDH}	250	–	100	–	30	–	ns
Interrupt Request Release Time (See Figure 6)		t_{IR}	–	1.2	–	0.9	–	0.7	μs
Request-to-Send Delay Time (See Figure 6)		t_{RTS}	–	560	–	480	–	400	ns
Input Rise and Fall Times (or 10% of the pulse width if smaller)		t_r, t_f	–	1.0	–	0.5	–	0.25	μs

FIGURE 1 – CLOCK PULSE WIDTH, LOW-STATE

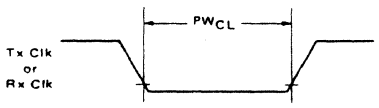


FIGURE 2 – CLOCK PULSE WIDTH, HIGH-STATE

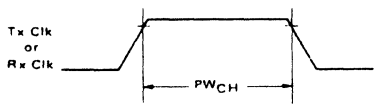


FIGURE 3 – TRANSMIT DATA OUTPUT DELAY

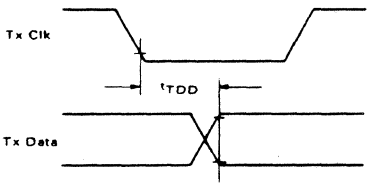


FIGURE 4 – RECEIVE DATA SETUP TIME
(÷ 1 Mode)

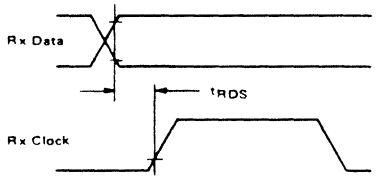


FIGURE 5 – RECEIVE DATA HOLD TIME
(÷ 1 Mode)

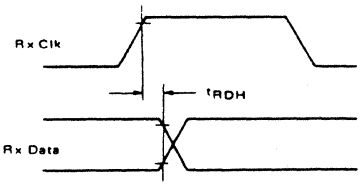
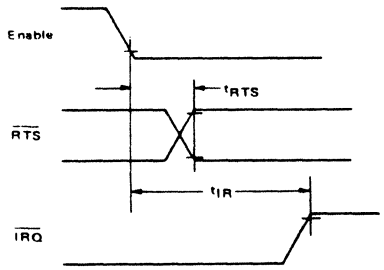


FIGURE 6 – REQUEST-TO-SEND DELAY AND INTERRUPT-REQUEST RELEASE TIMES

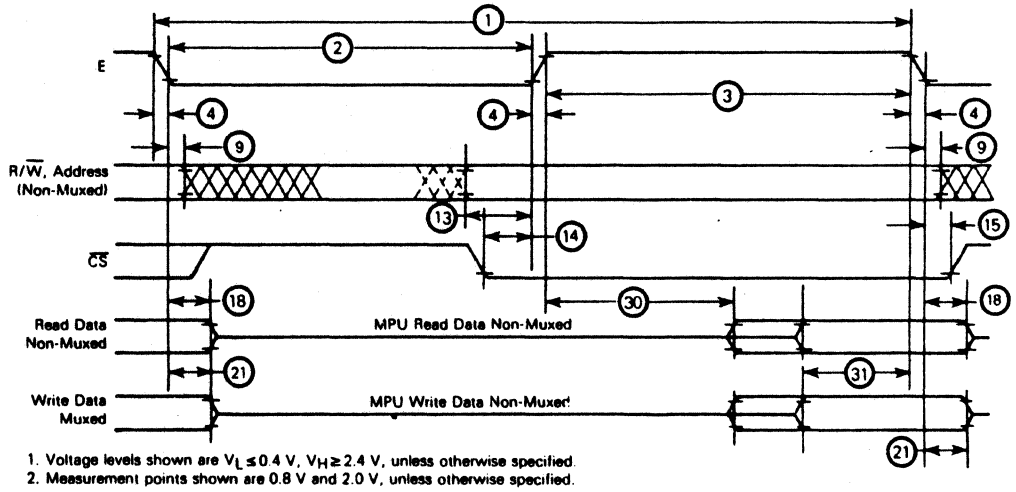


Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

BUS TIMING CHARACTERISTICS (See Notes 1 and 2 and Figure 7)

Ident. Number	Characteristic	Symbol	MC6850		MC68A50		MC68B50		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.67	10	0.5	10	µs
2	Pulse Width, E Low	PW_{EL}	430	9500	280	9500	210	9500	ns
3	Pulse Width, E High	PW_{EH}	450	9500	280	9500	220	9500	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time	t_{AH}	10	–	10	–	10	–	ns
13	Address Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	Chip Select Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	Chip Select Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	100	20	100	20	100	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ns
30	Output Data Delay Time	t_{DDR}	–	290	–	180	–	150	ns
31	Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

FIGURE 7 — BUS TIMING CHARACTERISTICS



1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.

FIGURE 8 — BUS TIMING TEST LOADS

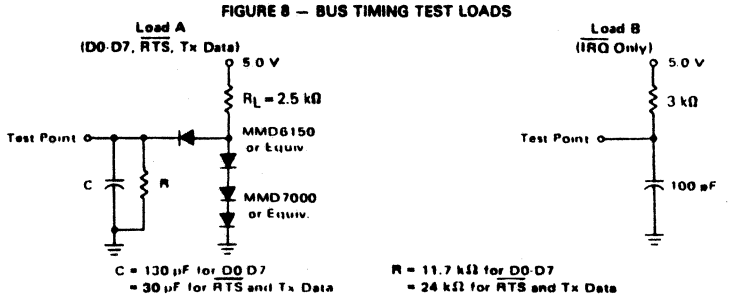
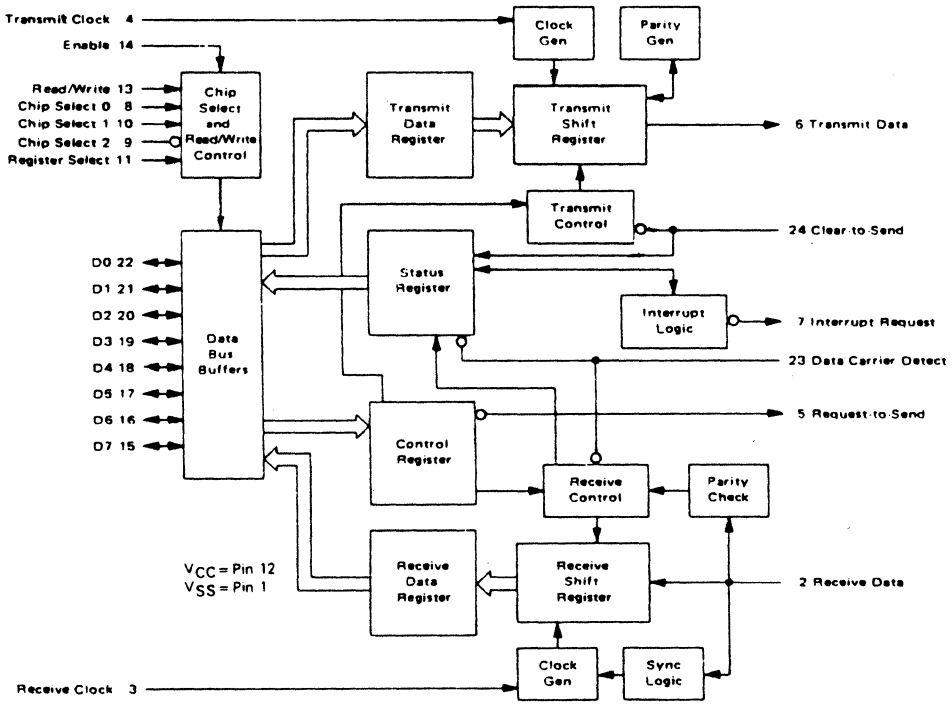


FIGURE 9 — EXPANDED BLOCK DIAGRAM



DEVICE OPERATION

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.

POWER ON/MASTER RESET

The master reset (CR0, CR1) should be set during system initialization to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. During the first master reset, the $\overline{IRQ}$ and $\overline{RTS}$ outputs are held at level 1. On all other master resets, the $\overline{RTS}$ output can be programmed high or low with the $\overline{IRQ}$ output held high. Control bits CR5 and CR6 should also be programmed to define the state of $\overline{RTS}$ whenever master reset is utilized. The ACIA also contains internal power-on reset logic to detect the power line turn-on transition and hold the chip in a reset state to prevent erroneous output transitions prior to initialization. This circuitry depends on clean power turn-on transitions. The power-on reset is released by means of the bus-programmed master reset which must be applied prior to operating the ACIA. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, parity (even, odd, or none), etc.

TRANSMIT

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of

double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

RECEIVE

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of 8 or 32 low samples on the receive line in the divide-by-16 and 64 modes respectively. False start bit deletion capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for a 7-bit word (7 bits plus parity), the receiver strips the parity bit (D7 = 0) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. The above sequence continues until all characters have been received.

INPUT/OUTPUT FUNCTIONS

ACIA INTERFACE SIGNALS FOR MPU

The ACIA interfaces to the M6800 MPU with an 8-bit bidirectional data bus, three chip select lines, a register select line, an interrupt request line, read/write line, and enable line. These signals permit the MPU to have complete control over the ACIA.

ACIA Bidirectional Data (D0-D7) – The bidirectional data lines (D0-D7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs an ACIA read operation.

ACIA Enable (E) – The Enable signal, E, is a high-impedance TTL-compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the MC6800 $\phi 2$ Clock or MC6809 E clock.

Read/Write (R/$\overline{W}$) – The Read/Write line is a high-impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When Read/Write is high (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is low, the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the Read/Write signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS0, CS1, $\overline{CS2}$) – These three high-impedance TTL-compatible input lines are used to address the ACIA. The ACIA is selected when CS0 and CS1 are high and $\overline{CS2}$ is low. Transfers of data to and from the ACIA are then performed under the control of the Enable Signal, Read/Write, and Register Select.

Register Select (RS) – The Register Select line is a high-impedance input that is TTL compatible. A high level is used to select the Transmit/Receive Data Registers and a low level the Control/Status Registers. The Read/Write signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$) – Interrupt Request is a TTL-compatible, open-drain (no internal pullup), active low output that is used to interrupt the MPU. The $\overline{IRQ}$ output remains low as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set. The $\overline{IRQ}$ status bit, when high, indicates the $\overline{IRQ}$ output is in the active state.

Interrupts result from conditions in both the transmitter and receiver sections of the ACIA. The transmitter section causes an interrupt when the Transmitter Interrupt Enabled condition is selected (CR5•$\overline{CR6}$), and the Transmit Data Register Empty (TDRE) status bit is high. The TDRE status bit indicates the current status of the Transmitter Data Register except when inhibited by Clear-to-Send ($\overline{CTS}$) being high or the ACIA being maintained in the Reset condition. The interrupt is cleared by writing data into the Transmit Data Register. The interrupt is masked by disabling the Transmitter Interrupt via CR5 or CR6 or by the loss of $\overline{CTS}$ which inhibits the TDRE status bit. The Receiver section causes an interrupt when the Receiver Interrupt Enable is set and the Receive Data Register Full (RDRF) status bit is high, an Overrun has occurred, or Data Carrier Detect ($\overline{DCD}$) has gone high. An interrupt resulting from the RDRF status bit can be cleared by reading data or resetting the ACIA. Interrupts caused by Overrun or loss of $\overline{DCD}$ are cleared by reading the status register after the error condition has occurred and then reading the Receive Data Register or resetting the ACIA. The receiver interrupt is masked by resetting the Receiver Interrupt Enable.

CLOCK INPUTS

Separate high-impedance TTL-compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16, or 64 times the data rate may be selected.

Transmit Clock (Tx CLK) – The Transmit Clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK) – The Receive Clock input is used for synchronization of received data. (In the ÷ 1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

SERIAL INPUT/OUTPUT LINES

Receive Data (Rx Data) – The Receive Data line is a high-impedance TTL-compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used.

Transmit Data (Tx Data) – The Transmit Data output line transfers serial data to a modem or other peripheral.

PERIPHERAL/MODEM CONTROL

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

Clear-to-Send ($\overline{CTS}$) – This high-impedance TTL-compatible input provides automatic control of the transmitting end of a communications link via the modem Clear-to-Send active low output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

Request-to-Send ($\overline{RTS}$) – The Request-to-Send output enables the MPU to control a peripheral or modem via the data bus. The RTS output corresponds to the state of the Control Register bits CR5 and CR6. When CR6 = 0 or both CR5 and CR6 = 1, the $\overline{RTS}$ output is low (the active state). This output can also be used for Data Terminal Ready (DTR).

Data Carrier Detect ($\overline{DCD}$) – This high-impedance TTL-compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem Data Carrier Detect output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low-to-high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set. The Rx CLK must be running for proper $\overline{DCD}$ operation.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers on the chip that are used for the status, control, receiving, and transmitting of data. The content of each of the registers is summarized in Table 1.

TRANSMIT DATA REGISTER (TDR)

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed with RS high and R/$\overline{W}$ low. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go low. Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 1-bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

RECEIVE DATA REGISTER (RDR)

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) in the status buffer to go high (full). Data may then be read through the bus by addressing the ACIA and selecting the Receive Data Register with RS and R/$\overline{W}$ high when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

TABLE 1 – DEFINITION OF ACIA REGISTER CONTENTS

Data Bus Line Number	Buffer Address			
	RS • R/$\overline{W}$ Transmit Data Register (Write Only)	RS • R/$\overline{W}$ Receive Data Register (Read Only)	$\overline{RS}$ • $\overline{R/W}$ Control Register (Write Only)	$\overline{RS}$ • R/$\overline{W}$ Status Register (Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select 1 (CR0)	Receive Data Register Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select 2 (CR1)	Transmit Data Register Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear to Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Transmit Control 1 (CR5)	Receiver Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Transmit Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Receive Interrupt Enable (CR7)	Interrupt Request ($\overline{IRQ}$)

* Leading bit – LSB – Bit 0
** Data bit will be zero in 7 bit plus parity modes
*** Data bit is "don't care" in 7 bit plus parity modes

CONTROL REGISTER

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are low. This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1) – The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set high to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4) – The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6) – Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

CR6	CR5	Function
0	0	$\overline{RTS}$ = low, Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = low, Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = high, Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = low, Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7) – The following interrupts will be enabled by a high level in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a low-to-high transition on the Data Carrier Detect ($\overline{DCD}$) signal line.

STATUS REGISTER

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is low and R/$\overline{W}$ is high. Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0 – Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect being high also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1 – The Transmit Data Register Empty bit being set high indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The low state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2 – The Data Carrier Detect bit will be high when the $\overline{DCD}$ input from a modem has gone high to indicate that a carrier is not present. This bit going high causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains high after the $\overline{DCD}$ input is returned low until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ inout remains high after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains high and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3 – The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send status bit.

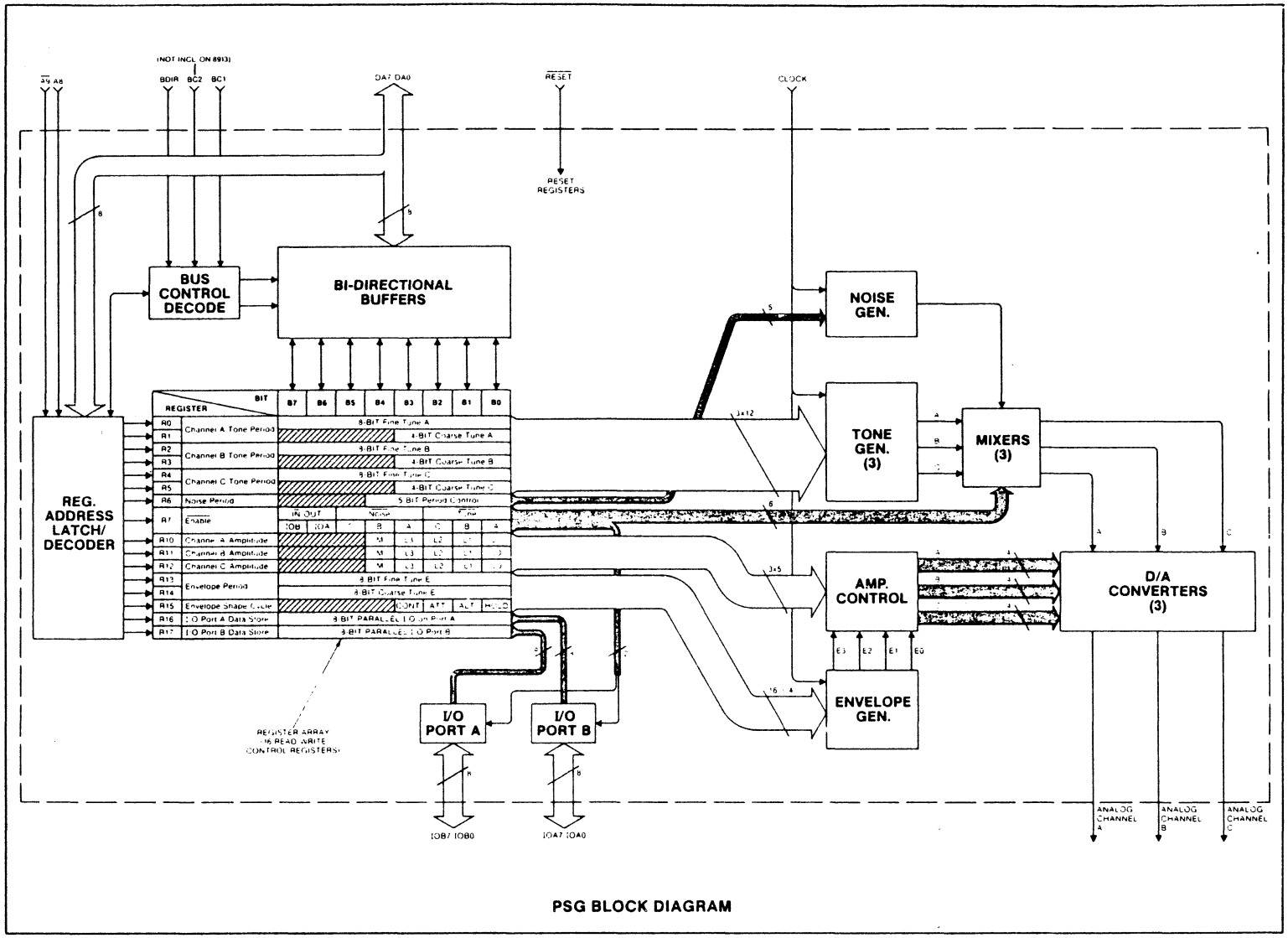
Framing Error (FE), Bit 4 – Framing error indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the first stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The framing error flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5 – Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has

been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6 – The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver partiy check results are inhibited.

Interrupt Request ($\overline{IRQ}$), Bit 7 – The $\overline{IRQ}$ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is low the $\overline{IRQ}$ bit will be high to indicate the interrupt or service request status. $\overline{IRQ}$ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register.



PSG BLOCK DIAGRAM

OPERATION

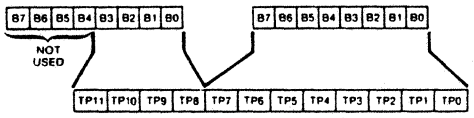
Since all functions of the PSG are controlled by the processor via a series of register loads, a detailed description of the PSG operation can best be accomplished by relating each PSG function to the control of its corresponding register. The function of creating or programming a specific sound or sound effect logically follows the control sequence listed:

Operation	Registers	Function
Tone Generator Control	R0--R5	Program tone periods.
Noise Generator Control	R6	Program noise period.
Mixer Control	R7	Enable tone and/or noise on selected channels.
Amplitude Control	R10--R12	Select "fixed" or "envelope-variable" amplitudes.
Envelope Generator Control	R13--R15	Program envelope period and select envelope pattern

Tone Generator Control
(Registers R0, R1, R2, R3, R4, R5)

The frequency of each square wave generated by the three Tone Generators (one each for Channels A, B, and C) is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 12-bit Tone Period value. Each 12-bit value is obtained in the PSG by combining the contents of the relative Coarse and Fine Tune registers, as illustrated in the following:

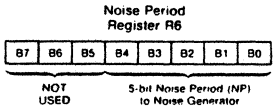
Coarse Tune Register	Channel	Fine Tune Register
R1	A	R0
R3	B	R2
R5	C	R4



12-bit Tone Period (TP) to Tone Generator

Noise Generator Control
(Register R6)

The frequency of the noise source is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 5-bit Noise Period value. This 5-bit value consists of the lower 5 bits (B4--B0) of register R6, as illustrated in the following:



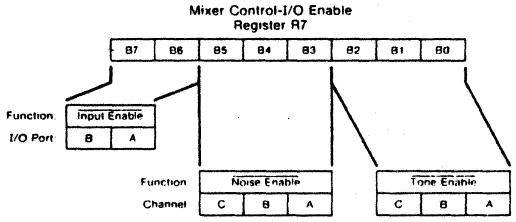
Mixer Control-I/O Enable
(Register R7)

Register R7 is a multi-function $\overline{\text{Enable}}$ register which controls the three Noise/Tone Mixers and the two general purpose I/O Ports.

The Mixers, as previously described, combine the noise and tone frequencies for each of the three channels. The determination of combining neither/either/both noise and tone frequencies on each channel is made by the state of bits B5--B0 of R7.

The direction (input or output) of the two general purpose I/O Ports (IOA and IOB) is determined by the state of bits B7 and B6 of R7.

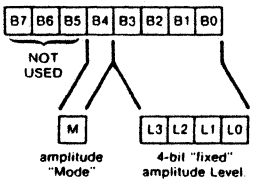
These functions are illustrated in the following:



Amplitude Control
(Registers R10, R11, R12)

The amplitudes of the signals generated by each of the three D/A Converters (one each for Channels A, B, and C) is determined by the contents of the lower 5 bits (B4--B0) of registers R10, R11, and R12 as illustrated in the following:

Amplitude Control Register	Channel
R10	A
R11	B
R12	C

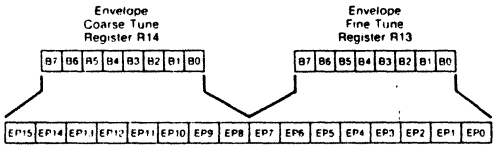


Envelope Generator Control
(Registers R13, R14, R15)

To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG: first, it is possible to vary the frequency of the envelope using registers R13 and R14; and second, the relative shape and cycle pattern of the envelope can be varied using register R15. The following paragraphs explain the details of the envelope control functions, describing first the envelope period control and then the envelope shape/cycle control.

ENVELOPE PERIOD CONTROL (Registers R13, R14)

The frequency of the envelope is obtained in the PSG by first counting down the input clock by 256, then by further counting down the result by the programmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated in the following:

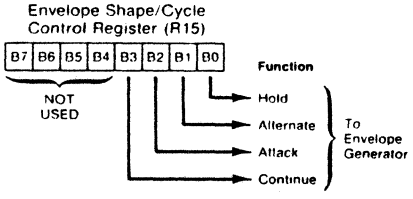


16-bit Envelope Period (EP) to Envelope Generator

ENVELOPE SHAPE/CYCLE CONTROL (Register R15)

The Envelope Generator further counts down the envelope frequency by 16, producing a 16-state per cycle envelope pattern as defined by its 4-bit counter output, E3 E2 E1 E0. The particular shape and cycle pattern of any desired envelope is accomplished by controlling the count pattern (count up/count down) of the 4-bit counter and by defining a single-cycle or repeat-cycle pattern.

This envelope shape/cycle control is contained in the lower 4 bits (B3--B0) of register R15. Each of these 4 bits controls a function in the envelope generator, as illustrated in the following:



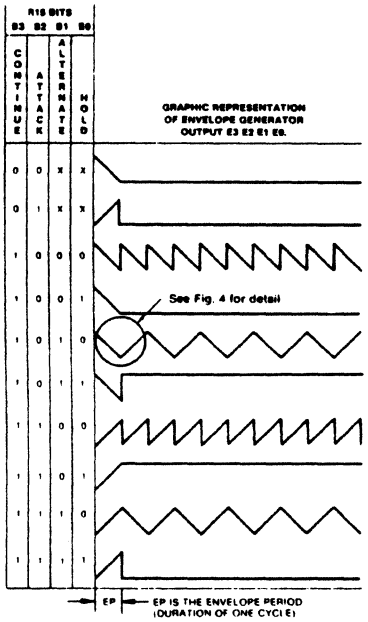


Fig. 1 ENVELOPE SHAPE/CYCLE OPERATION

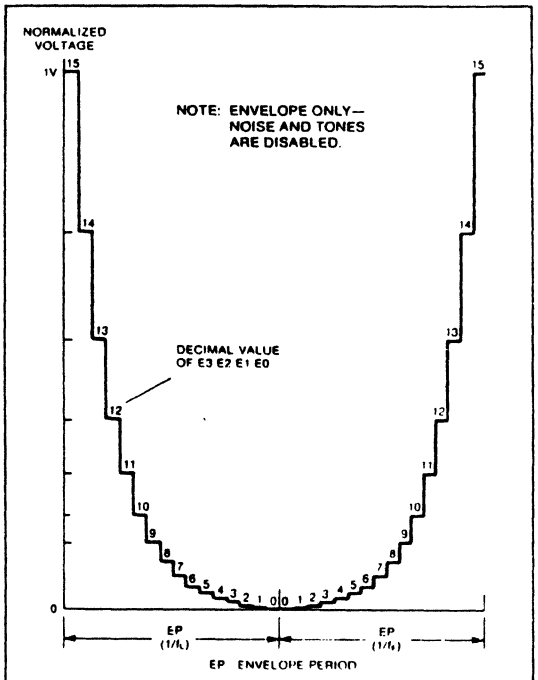


Fig. 3 D/A CONVERTER OUTPUT

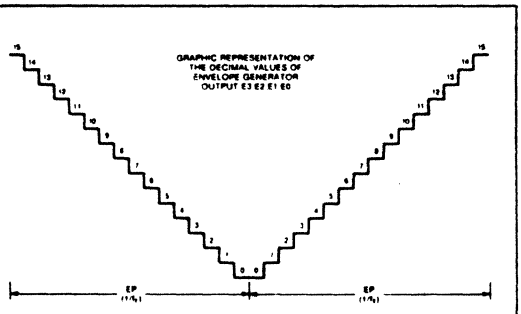


Fig. 2 DETAIL OF TWO CYCLES OF Fig. 1 (ref. waveform "1010" in Fig. 1)

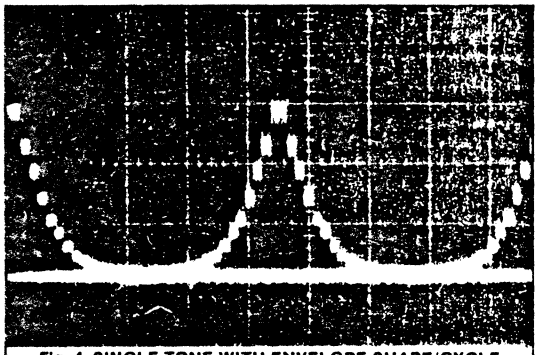


Fig. 4 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1010

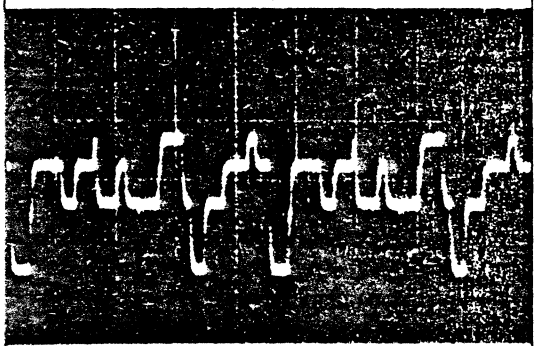


Fig. 5 MIXTURE OF THREE TONES WITH FIXED AMPLITUDES

I/O Port Data Store
(Registers R16, R17)

Registers R16 and R17 function as intermediate data storage registers between the PSG/CPU data bus (DA0--DA7) and the two I/O ports (IOA7--IOA0 and IOB7--IOB0). Both ports are available in the AY-3-8910; only I/O Port A is available in the AY-3-8912; none are available on the AY-3-8913. Using registers R16 and R17 for the transfer of I/O data has no effect on sound generation.

D/A Converter Operation

Since the primary use of the PSG is to produce sound for the highly imperfect amplitude detection mechanism of the human ear, the D/A conversion is performed in logarithmic steps with a normalized voltage range of from 0 to 1 Volt. The specific amplitude control of each of the three D/A Converters is accomplished by the three sets of 4-bit outputs of the Amplitude Control block, while the Mixer outputs provide the base signal frequency (Noise and/or Tone).

980

ELECTRICAL CHARACTERISTICS (AY-3-8910, AY-3-8912)

Maximum Ratings*

Storage Temperature ..... −55°C to +150°C
Operating Temperature ..... 0°C to +40°C
V_{CC} and all other Input/Output
Voltages with Respect to V_{SS} ..... −0.3V to +8.0V

*Exceeding these ratings could cause permanent damage to the device. This is a stress rating only and functional operation of this device at these conditions is not implied—operating ranges are specified in Standard Conditions. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Data labeled "typical" is presented for design guidance only and is not guaranteed.

Standard Conditions (unless otherwise noted):

V_{CC} = +5V ±5%
V_{SS} = GND
Operating Temperature = 0°C to +40°C

Characteristics	Sym	Min	Typ**	Max	Units	Conditions
DC CHARACTERISTICS						
All Inputs						
Low Level	V_{IL}	0	—	0.6	V	
High Level	V_{IH}	2.4	—	V_{CC}	V	
All Outputs (except Analog Channel Outputs)						
Low Level	V_{OL}	0	—	0.5	V	I_{OL} = 1.6mA, 20pf
High Level	V_{OH}	2.4	—	V_{CC}	V	I_{OH} = 100μA, 20pf
Analog Channel Outputs	V_o	0	—	60	dB	Test Circuit: Fig. 6
Power Supply Current	I_{CC}	—	45	85	mA	
AC CHARACTERISTICS						
Clock Input						
Frequency	f_C	1	—	2	MHz	Fig. 7
Rise Time	t_r	—	—	50	ns	Fig. 7
Fall Time	t_f	—	—	50	ns	Fig. 7
Duty Cycle	—	25	50	85	%	Fig. 7
Bus Signals (BDIR, BC2, BC1)						
Associative Delay Time	t_{BD}	—	—	50	ns	
Reset						
Reset Pulse Width	t_{RW}	500	—	—	ns	Fig. 8
Reset to Bus Control Delay Time	t_{RB}	100	—	—	ns	Fig. 8
A9, A8, DA7--DA0 (Address Mode)						
Address Setup Time	t_{AS}	400	—	—	ns	Fig. 9
Address Hold Time	t_{AH}	100	—	—	ns	Fig. 9
DA7--DA0 (Write Mode)						
Write Data Pulse Width	t_{DW}	500	—	10,000	ns	Fig. 10
Write Data Setup Time	t_{DS}	50	—	—	ns	Fig. 10
Write Data Hold Time	t_{DH}	100	—	—	ns	Fig. 10
DA7--DA0 (Read Mode)						
Read Data Access Time	t_{DA}	—	250	500	ns	Fig. 11
DA7--DA0 (Inactive Mode)						
Tristate Delay Time	t_{TS}	—	100	200	ns	Fig. 11

**Typical values are at +25°C and nominal voltages.

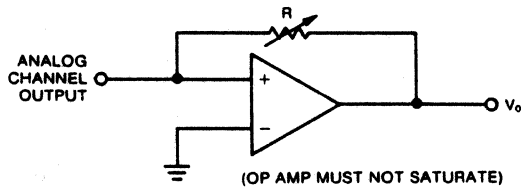


Fig. 6 ANALOG CHANNEL OUTPUT TEST CIRCUIT

981

ELECTRICAL CHARACTERISTICS (AY-3-8913)

Maximum Ratings*

Storage Temperature ..... −55°C to +150°C
Operating Temperature ..... 0°C to +70°C
V_{CC} and all other Input/Output Voltages with Respect to V_{SS} ..... −0.3V to +8.0V

Standard Conditions (unless otherwise noted):

V_{CC} = +5V ±5%
V_{SS} = GND
Operating Temperature = 0°C to +70°C

*Exceeding these ratings could cause permanent damage to the device. This is a stress rating only and functional operation of this device at these conditions is not implied—operating ranges are specified in Standard Conditions. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Data labeled "typical" is presented for design guidance only and is not guaranteed.

Characteristics	Sym	Min	Max	Units	Conditions
DC CHARACTERISTICS					
Input Voltage Levels					
Low Level	V_{IL}	0	0.7	V	
High Level	V_{IH}	2.2	V_{CC}	V	
Output Voltage Levels (except Analog Channel Outputs)					
Low Level	V_{OL}	0	0.4	V	1 TTL Load
High Level	V_{OH}	2.4	V_{CC}	V	+100pf
Analog Channel Outputs	V_O	0	2000	μA	Test Circuit: Fig. 6
Power Supply Current	I_{CC}	—	85	mA	
AC CHARACTERISTICS					
Clock Input					
Frequency	f_c	1	2.5	MHz	Fig. 7
Rise Time	t_r	—	50	ns	
Fall Time	t_f	—	50	ns	
Duty Cycle	—	40	60	%	
Bus Signals (BDIR, BC2, BC1)					
Associative Delay Time	t_{BD}	—	50	ns	
Reset					
Reset Pulse Width	t_{RW}	5	—	μs	Fig. 8
Reset to Bus Control Delay Time	t_{RB}	100	—	ns	
A9, A8, DA7--DA0 (Address Mode)					
Address Setup Time	t_{AS}	300	—	ns	Fig. 9
Address Hold Time	t_{AH}	50	—	ns	
DA7--DA0 (Write Mode)					
Write Data Pulse Width	t_{DW}	1800	—	ns	Fig. 10
Write Data Setup Time	t_{DS}	50	—	ns	
Write Data Hold Time	t_{DH}	100	—	ns	
DA7--DA0 (Read Mode)					
Read Data Access Time	t_{DA}	—	350	ns	Fig. 11
DA7--DA0 (Inactive Mode)					
Tristate Delay Time	t_{TS}	—	400	ns	

TIMING DIAGRAMS

Fig. 7 CLOCK AND BUS SIGNAL TIMING

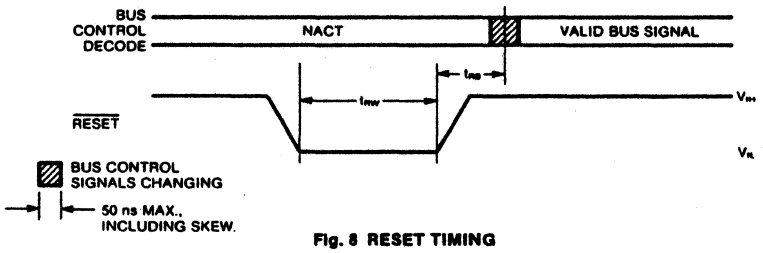


Fig. 8 RESET TIMING

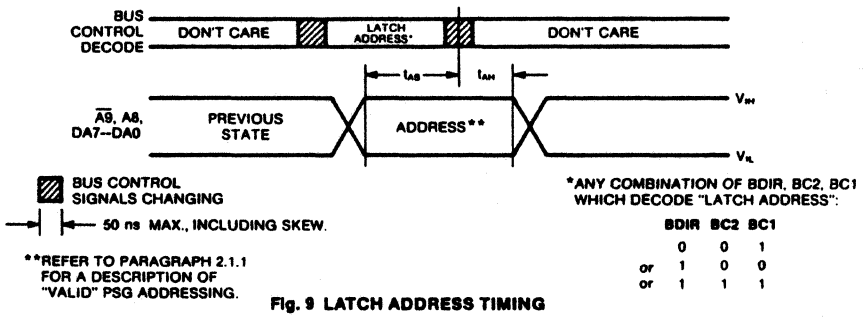


Fig. 9 LATCH ADDRESS TIMING

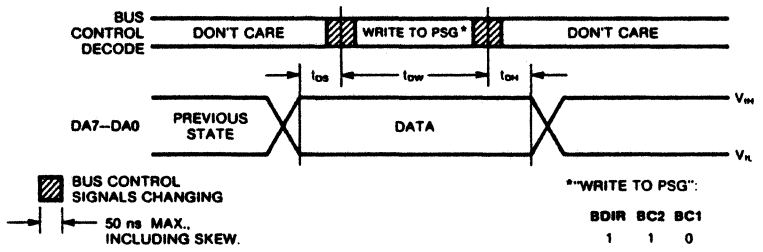


Fig. 10 WRITE DATA TIMING

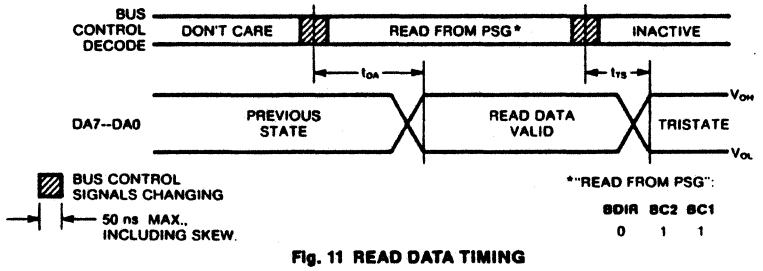


Fig. 11 READ DATA TIMING

983

UNITED TECHNOLOGIES MOSTEK

MICROCOMPUTER COMPONENTS

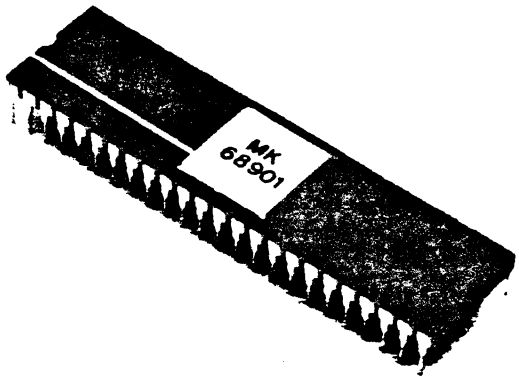
PRELIMINARY

MK68901 MULTI-FUNCTION PERIPHERAL

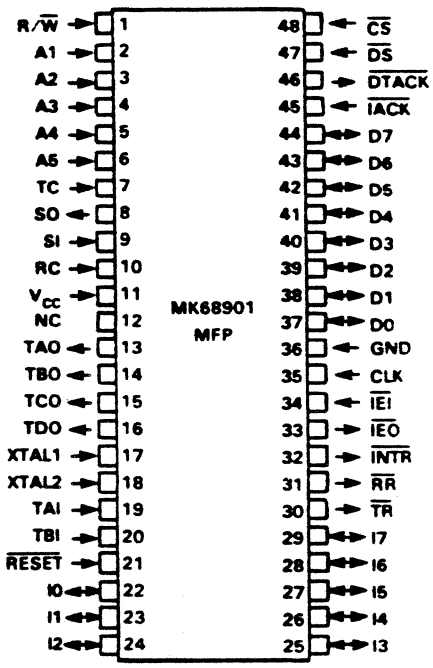
FEATURES

- 8 Input/Output Pins
  - Individually programmable direction
  - Individual interrupt source capability
    - Programmable edge selection
- 16 Source interrupt controller
  - 8 Internal sources
  - 8 External sources
  - Individual source enable
  - Individual source masking
  - Programmable interrupt service modes
    - Polling
    - Vector generation
      - Optional In-service status
  - Daisy chaining capability
- Four timers with individually programmable prescaling
  - Two multimode timers
    - Delay mode
    - Pulse width measurement mode
    - Event counter mode
  - Two delay mode timers
  - Independent clock input
  - Time out output option
- Single channel USART
  - Full Duplex
  - Asynchronous to 62.5 kbps
  - Byte synchronous to 1 Mbps
  - Internal/external baud rate generation
  - DMA handshake signals

MK68901
Figure 1



DEVICE PINOUT
Figure 2



984

- Modem control
- Loop back mode

□ 68000 Bus compatible

□ 48 Pin DIP

INTRODUCTION

The MK68901 MFP (Multi-Function Peripheral) is a combination of many of the necessary peripheral functions in a microprocessor system. Included are:

Eight parallel I/O lines

Interrupt controller for 16 sources

Four timers

Single channel full duplex USART

The use of the MFP in a system can significantly reduce chip count, thereby reducing system cost. The MFP is completely 68000 bus compatible, and 24 directly addressable internal registers provide the necessary control and status interface to the programmer.

The MFP is a derivative of the MK3801 STI, a Z80 family peripheral.

PIN DESCRIPTION

GND: Ground

Vcc: +5 volts (± 5%)

$\overline{CS}$: Chip Select (input, active low). $\overline{CS}$ is used to select the MK68901 MFP for accesses to the internal registers. $\overline{CS}$ and $\overline{IACK}$ must not be asserted at the same time.

$\overline{DS}$: Data Strobe (input, active low). $\overline{DS}$ is used as part of the chip select and interrupt acknowledge functions.

R/$\overline{W}$: Read/Write (input). R/$\overline{W}$ is the signal from the bus master indicating whether the current bus cycle is a Read (High) or Write (Low) cycle.

$\overline{DTACK}$: Data Transfer Acknowledge. (output, active low, tri-stateable). $\overline{DTACK}$ is used to signal the bus master that data is ready, or that data has been accepted by the MK68901 MFP.

A1-A5: Address Bus (inputs). The address bus is used to address one of the internal registers during a read or write cycle.

D_0-D_7: Data Bus (bi-directional, tri-stateable). The data bus is used to receive data from or transmit data to one of the internal registers during a read or write cycle. It is also used to pass a vector during an interrupt acknowledge cycle.

CLK: Clock (input). This input is used to provide the internal timing for the MK68901 MFP.

$\overline{RESET}$: Device reset. (input, active low). Reset disables the USART receiver and transmitter, stops all timers and forces the timer outputs low, disables all interrupt channels and clears any pending interrupts. The General Purpose Interrupt/ I/O lines will be placed in the tri-state input mode. All internal registers (except the timer, USART data registers, and transmit status register) will be cleared.

$\overline{INTR}$: Interrupt Request (output, active low, open drain). $\overline{INTR}$ is asserted when the MK68901 MFP is requesting an interrupt. $\overline{INTR}$ is negated during an interrupt acknowledge cycle or by clearing the pending interrupt(s) through software.

$\overline{IACK}$: Interrupt Acknowledge (input, active low). $\overline{IACK}$ is used to signal the MK68901 MFP that the CPU is acknowledging an interrupt. $\overline{CS}$ and $\overline{IACK}$ must not be asserted at the same time.

$\overline{IEI}$: Interrupt Enable In (input, active low). $\overline{IEI}$ is used to signal the MK68901 MFP that no higher priority device is requesting interrupt service.

$\overline{IEO}$: Interrupt Enable Out (output, active low). $\overline{IEO}$ is used to signal lower priority peripherals that neither the MK68901 MFP nor another higher priority peripheral is requesting interrupt service.

I_0-I_7: General Purpose Interrupt I/O lines. These lines may be used as interrupt inputs and/or I/O lines. When used as interrupt inputs, their active edge is programmable. A data direction register is used to define which lines are to be Hi-Z inputs and which lines are to be push-pull TTL compatible outputs.

SO: Serial Output. This is the output of the USART transmitter.

SI: Serial Input. This is the input to the USART receiver.

RC: Receiver Clock. This input controls the serial bit rate of the USART receiver.

TC: Transmitter Clock. This input controls the serial bit rate of the USART transmitter.

$\overline{RR}$: Receiver Ready. (output, active low) DMA output for receiver, which reflects the status of Buffer Full in port number 15.

$\overline{TR}$: Transmitter Ready. (output, active low) DMA output for transmitter, which reflects the status of Buffer Empty in port number 16.

TAO,TBO, TCO, TDO: Timer Outputs. Each of the four timers has an output which can produce a square wave. The output will change states each timer cycle; thus one full period of the timer out signal is equal to two timer cycles. TAO or TBO can be reset (logic "0") by a write to TACR, or TBCR respectively.

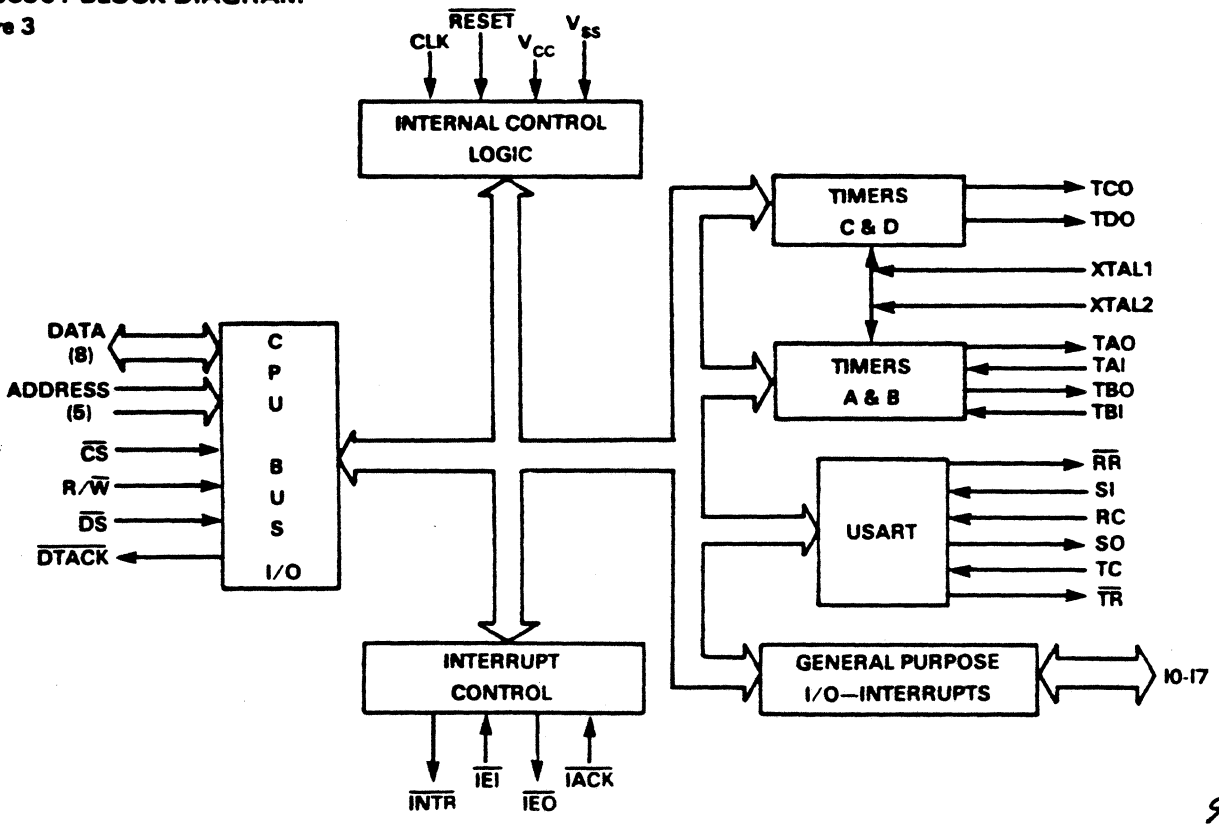
XTAL1, XTAL2: Timer Clock inputs. A crystal can be connected between XTAL1 and XTAL2, or XTAL1 can be driven with a TTL level clock. When driving XTAL1 with a TTL level clock, XTAL2 must be allowed to float. When using a crystal, external capacitors are required. See Figure 27. All chip accesses are independent of the timer clock.

TAI,TBI: Timer A,B inputs. Used when running the timers in the event count or the pulse width measurement mode. The interrupt channels associated with I4 and I3 are used for TAI and TBI, respectively. Thus, when running a timer in the pulse width measurement mode, I4 or I3 can be used for I/O only.

REGISTER MAP
Figure 4

Address Port No.	Abbreviation	Register Name
0	GPIP	GENERAL PURPOSE I/O
1	AER	ACTIVE EDGE REGISTER
2	DDR	DATA DIRECTION REGISTER
3	IERA	INTERRUPT ENABLE REGISTER A
4	IERB	INTERRUPT ENABLE REGISTER B
5	IPRA	INTERRUPT PENDING REGISTER A
6	IPRB	INTERRUPT PENDING REGISTER B
7	ISRA	INTERRUPT IN-SERVICE REGISTER A
8	ISRB	INTERRUPT IN-SERVICE REGISTER B
9	IMRA	INTERRUPT MASK REGISTER A
A	IMRB	INTERRUPT MASK REGISTER B
B	VR	VECTOR REGISTER
C	TACR	TIMER A CONTROL REGISTER
D	TBCR	TIMER B CONTROL REGISTER
E	TCDCR	TIMERS C AND D CONTROL REGISTER
F	TADR	TIMER A DATA REGISTER
10	TBDR	TIMER B DATA REGISTER
11	TCDR	TIMER C DATA REGISTER
12	TDDR	TIMER D DATA REGISTER
13	SCR	SYNC CHARACTER REGISTER
14	UCR	USART CONTROL REGISTER
15	RSR	RECEIVER STATUS REGISTER
16	TSR	TRANSMITTER STATUS REGISTER
17	UDR	USART DATA REGISTER

MK68901 BLOCK DIAGRAM
Figure 3



986

INTERRUPTS

The General Purpose I/O-Interrupt Port (GPIP) provides eight I/O lines that may be operated either as inputs or outputs under software control. In addition, each line may generate an interrupt on either a positive going edge or a negative going edge of the input signal.

The GPIP has three associated registers. One allows the programmer to specify the Active Edge for each bit that will trigger an interrupt. Another register specifies the Data Direction (input or output) associated with each bit. The third register is the actual data I/O register used to input or output data to the port. These three registers are illustrated in Figure 5.

The Active Edge Register (AER) allows each of the General Purpose Interrupts to produce an interrupt on either a 1-0 transition or a 0-1 transition. Writing a zero to the appropriate bit of the AER causes the associated input to produce an interrupt on the 1-0 transition, while a 1 causes the interrupt on the 0-1 transition. The edge bit is simply one input to an exclusive-or gate, with the other input coming from the input buffer and the output going to a 1-0 transition detector. Thus, depending upon the state of the input, writing the AER can cause an interrupt-producing transition, which will cause an interrupt on the associated channel, if that channel is enabled. One would then normally configure the AER before enabling interrupts via IERA and IERB. Note: changing the edge bit, with the interrupt enabled, may cause an interrupt on that channel.

The Data Direction Register (DDR) is used to define I0-I7 as inputs or as outputs on a bit by bit basis. Writing a zero into a bit of the DDR causes the corresponding Interrupt-I/O pin to be a Hi-Z input. Writing a one into a bit of the DDR causes the corresponding pin to be configured as a push-pull output. When data is written into the GPIP, those pins defined as inputs will remain in the Hi-Z state while those pins defined as outputs will assume the state (high or low) of their corresponding bit in the GPIP. When the GPIP is read, the data read will come directly from the corresponding bit of the GPIP register for all pins defined as output, while the data read on all pins defined as inputs will come from the input buffers.

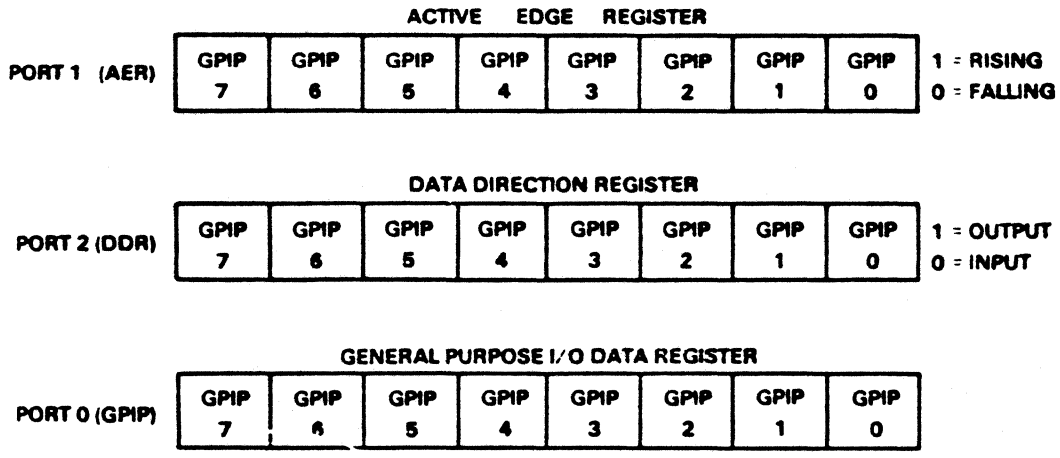
Each individual function in the MK68901 is provided with a unique interrupt vector that is presented to the system during the interrupt acknowledge cycle. The interrupt vector returned during the interrupt acknowledge cycle is shown in Figure 6, while the vector register is shown in Figure 7.

There are 16 vector addresses generated internally by the MK68901, one for each of the 16 interrupt channels.

The Interrupt Control Registers (Figure 8) provide control of interrupt processing for all I/O facilities of the MK68901. These registers allow the programmer to enable or disable

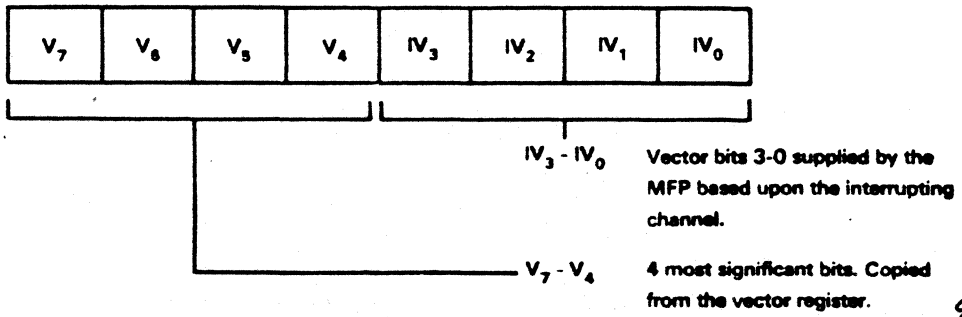
GENERAL PURPOSE I/O REGISTERS

Figure 5



INTERRUPT VECTOR

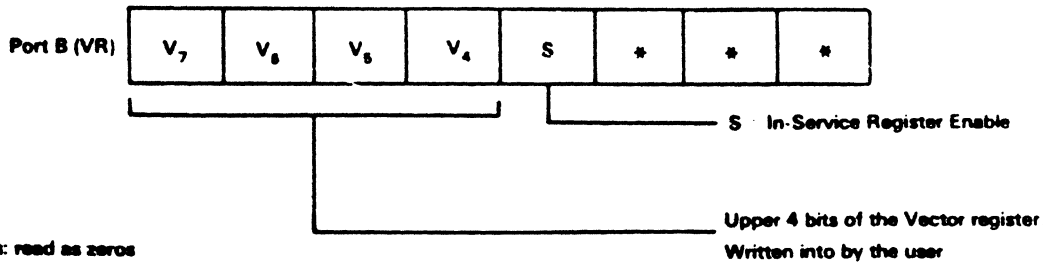
Figure 6



any or all of the 16 interrupts, providing masking for any interrupts, and provide access to the pending and in-service status of the interrupts. Optional end-of-interrupt modes are available under software control. All the interrupts are prioritized as shown in Figure 9

VECTOR REGISTER

Figure 7



INTERRUPT CONTROL REGISTERS

Figure 8

INTERRUPT ENABLE REGISTERS

ADDRESS		7	6	5	4	3	2	1	0
PORT 3	A (IERA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 4	B (IERB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

INTERRUPT PENDING REGISTERS

		7	6	5	4	3	2	1	0
PORT 5	A (IPRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 6	B (IPRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

WRITING 0 = CLEAR
WRITING 1 = UNCHANGED

INTERRUPT IN-SERVICE REGISTERS

		7	6	5	4	3	2	1	0
PORT 7	A (ISRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 8	B (ISRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

INTERRUPT MASK REGISTERS

		7	6	5	4	3	2	1	0
PORT 9	A (IMRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT A	B (IMRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

1 = UNMASKED 0 = MASKED

INTERRUPT CONTROL REGISTER DEFINITIONS

Figure 9

Priority	Channel	Description
HIGHEST	1111	General Purpose Interrupt 7(I7)
	1110	General Purpose Interrupt 6(I6)
	1101	Timer A
	1100	Receive Buffer Full
	1011	Receive Error
	1010	Transmit Buffer Empty
	1001	Transmit Error
	1000	Timer B
	0111	General Purpose Interrupt 5(I5)
	0110	General Purpose Interrupt 4(I4)
	0101	Timer C
	0100	Timer D
	0011	General Purpose Interrupt 3(I3)
	0010	General Purpose Interrupt 2(I2)
	0001	General Purpose Interrupt 1(I1)
LOWEST	0000	General Purpose Interrupt 0(I0)

Interrupts may be either polled or vectored. Each channel may be individually enabled or disabled by writing a one or a zero in the appropriate bit of Interrupt Enable Registers (IERA,IERB--see Figure 8 for all registers in this section). When disabled, an interrupt channel is completely inactive. Any internal or external action which would normally produce an interrupt on that channel is ignored and any pending interrupt on that channel will be cleared by disabling that channel. Disabling an interrupt channel has no effect on the corresponding bit in Interrupt In-Service Registers (ISRA,ISRB); thus, if the In-service Registers are used and an interrupt is in service on that channel when the channel is disabled, it will remain in service until cleared in the normal manner. IERA and IERB are also readable.

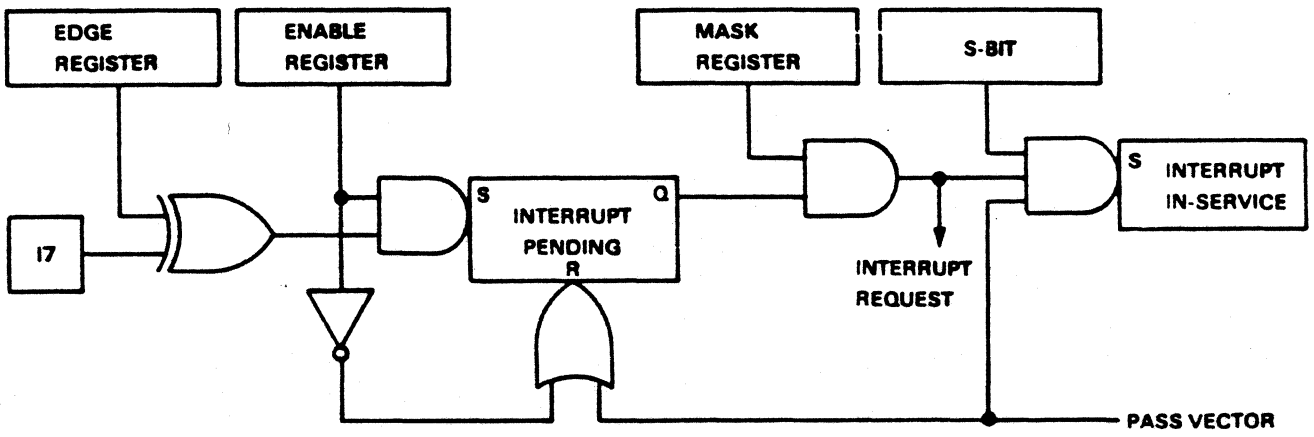
When an interrupt is received on an enabled channel, its corresponding bit in the pending register will be set. When that channel is acknowledged it will pass its vector, and the corresponding bit in the Interrupt Pending Register (IPRA or IPRB) will be cleared. IPRA and IPRB are readable; thus by polling IPRA and IPRB, it can be determined whether a channel has a pending interrupt. IPRA and IPRB are also writeable and a pending interrupt can be cleared without going through the acknowledge sequence by writing a zero to the appropriate bit. This allows any one bit to be cleared, without altering any other bits, simply by writing all ones except for the bit position to be cleared to IPRA or IPRB. Thus a fully polled interrupt scheme is possible. Note: writing a one to IPRA, IPRB has no effect on the interrupt pending register.

The interrupt mask registers (IMRA and IMRB) may be used to block a channel from making an interrupt request. Writing a zero into the corresponding bit of the mask register will still allow the channel to receive an interrupt and latch it into its pending bit (if that channel is enabled), but will prevent that channel from making an interrupt request. If that channel is causing an interrupt request at the time the corresponding bit in the mask register is cleared, the request will cease. If no other channel is making a request, $\overline{\text{INTR}}$ will go inactive. If the mask bit is re-enabled, any pending interrupt is now free to resume its request unless blocked by a higher priority request for service. IMRA and IMRB are also readable. A conceptual circuit of an interrupt channel is shown in Figure 10.

There are two end-of-interrupt modes: the automatic end-of-interrupt mode and the software end-of-interrupt mode. The mode is selected by writing a one or a zero to the S bit of the Vector Register(VR). If the S bit of the VR is a one, all channels operate in the software end-of-interrupt mode. If the S bit is a zero, all channels operate in the automatic end-of-interrupt mode, and a reset is held on all in-service bits. In the automatic end-of-interrupt mode, the pending bit is cleared when that channel passes its vector. At that point, no further history of that interrupt remains in the MK68901 MFP. In the software end-of-interrupt mode, the in-service

A CONCEPTUAL CIRCUIT OF AN INTERRUPT CHANNEL

Figure 10



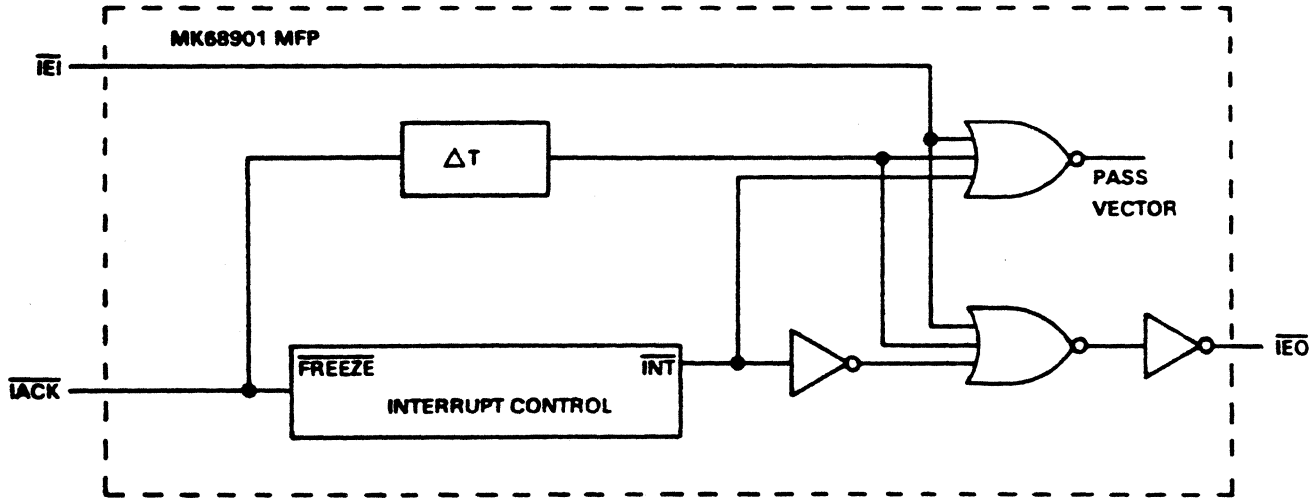
bit is set and the pending bit is cleared when the channel passes its vector. With the in-service bit set, no lower priority channel is allowed to request an interrupt or to pass its vector during an acknowledge sequence; however, a lower priority channel may still receive an interrupt and latch it into the pending bit. A higher priority channel may still request an interrupt and be acknowledged. The in-service bit of a particular channel may be cleared by writing a zero to the corresponding bit in ISRA or ISRB. Typically, this will be done at the conclusion of the interrupt routine just before the return. Thus no lower priority channel will be allowed to request service until the higher priority channel is complete, while channels of still higher priority will be allowed to request service. While the in-service bit is set, a second interrupt on that channel may be received and latched into the pending bit, though no service request will be made in response to the second interrupt until the in-service bit is cleared. ISRA and ISRB may be read at any time. Only a zero may be written into any bit of ISRA and ISRB; thus the in-service bits may be cleared in software but cannot be set in software. This allows any one bit to be cleared, without altering any other bits, simply by writing all ones except for the bit position to be cleared to ISRA or ISRB, as with IPRA and IPRB.

Each interrupt channel responds with a discrete 8-bit vector when acknowledged. The upper four bits of the vector are set by writing the upper four bits of the VR. The four low order bits (Bit 3-Bit 0) are generated by the interrupting channel.

To acknowledge an interrupt, $\overline{IACK}$ goes low, the $\overline{IEI}$ input must go low (or be tied low) and the MK68901 MFP must have an acknowledgeable interrupt pending. The Daisy Chaining capability (Figure 11) requires that all parts in a chain have a common $\overline{IACK}$. When the common $\overline{IACK}$ goes

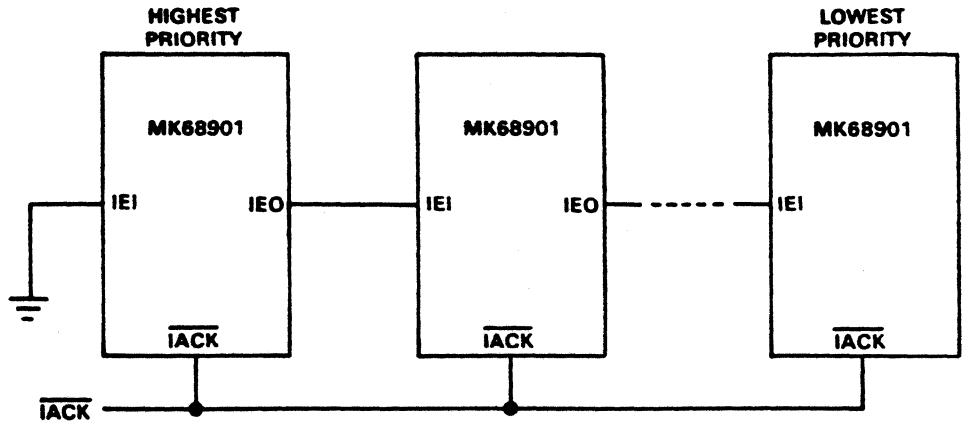
A CONCEPTUAL CIRCUIT OF THE MK68901 MFP DAISY CHAINING

Figure 11a



DAISY CHAINING

Figure 11b



990

low, all parts freeze and prioritize interrupts in parallel. Then priority is passed down the chain, via $\overline{IEI}$ and $\overline{IEO}$, until a part which has a pending interrupt is reached. The part with the pending interrupt, passes a vector, does not propagate $\overline{IEO}$, and generates $\overline{DTACK}$.

Figure 9 describes the 16 prioritized interrupt channels. As shown, General Purpose Interrupt 7 has the highest priority, while General Purpose Interrupt 0 is assigned the lowest priority. Each of these channels may be reprioritized, in effect, by selectively masking interrupts under software control. The binary numbers under "channel" correspond to the modified bits IV3, IV2, IV1, and IV0, respectively, of the Interrupt Vector for each channel (see Figure 6).

Each channel has an enable bit contained in IERA or IERB, a pending latch contained in IPRA or IPRB, a mask bit contained in IMRA or IMRB, and an in-service latch contained in ISRA or ISRB. Additionally, the eight General Purpose Interrupts each have an edge bit contained in the Active Edge Register (AER), a bit to define the line as input or output contained in the Data Direction Register (DDR) and an I/O bit in the General Purpose Interrupt-I/O Port (GPIP).

TIMERS

There are four timers on the MK68901 MFP. Two of the timers (Timer A and Timer B) are full function timers which can perform the basic delay function and can also perform event counting, pulse width measurement, and waveform generation. The other two timers (Timer C and Timer D) are delay timers only. One or both of these timers can be used to supply the baud rate clocks for the USART. All timers are prescaler/counter timers with a common independent clock input (XTAL1, XTAL2). In addition, all timers have a time-out output function that toggles each time the timer times out.

The four timers are programmed via three Timer Control Registers and four Timer Data Registers. Timers A and B are controlled by the control registers TACR and TBCR, respectively (see Figure 12), and by the data registers TADR and TBDR (Figure 13). Timers C and D are controlled by the control register TCDCR (see Figure 14) and two data registers TCDR and TDDR. Bits in the control registers allow the selection of operational mode, prescale, and control, while the data registers are used to read the timer or write into the time constant register. Timer A and B input pins, TAI and TBI, are used for the event and pulse width modes for timers A and B.

With the timer stopped, no counting can occur. The timer contents will remain unaltered while the timer is stopped (unless reloaded by writing the Timer Data Register), but any residual count in the prescaler will be lost.

In the delay mode, the prescaler is always active. A count pulse will be applied to the main timer unit each time the prescribed number of timer clock cycles has elapsed. Thus, if the prescaler is programmed to divide by ten, a count pulse will be applied to the main counter every ten cycles of the timer clock.

Each time a count pulse is applied to the main counter, it will decrement its contents. The main counter is initially loaded by writing to the Timer Data Register. Each count pulse will cause the current count to decrement. When the timer has decremented down to "01", the next count pulse will not cause it to decrement to "00". Instead, the next count pulse will cause the timer to be reloaded from the Timer Data Register. Additionally, a "Time out" pulse will be produced. This Time Out pulse is coupled to the timer interrupt channel, and, if that channel is enabled, an interrupt will be produced. The Time Out pulse is also coupled to the timer output pin and will cause the pin to change states. The

TIMER A AND B CONTROL REGISTERS

Figure 12

Port C (TACR)	*	*	*	TIMER A RESET	AC_3	AC_2	AC_1	AC_0
Port D (TBCR)	*	*	*	TIMER B RESET	BC_3	BC_2	BC_1	BC_0

C_3	C_2	C_1	C_0	
0	0	0	0	Timer Stopped
0	0	0	1	Delay Mode, ÷ 4 Prescale
0	0	1	0	Delay Mode, ÷ 10 Prescale
0	0	1	1	Delay Mode, ÷ 16 Prescale
0	1	0	0	Delay Mode, ÷ 50 Prescale
0	1	0	1	Delay Mode, ÷ 64 Prescale
0	1	1	0	Delay Mode, ÷ 100 Prescale
0	1	1	1	Delay Mode, ÷ 200 Prescale
1	0	0	0	Event Count Mode
1	0	0	1	Pulse Width Mode, ÷ 4 Prescale
1	0	1	0	Pulse Width Mode, ÷ 10 Prescale
1	0	1	1	Pulse Width Mode, ÷ 16 Prescale
1	1	0	0	Pulse Width Mode, ÷ 50 Prescale
1	1	0	1	Pulse Width Mode, ÷ 64 Prescale
1	1	1	0	Pulse Width Mode, ÷ 100 Prescale
1	1	1	1	Pulse Width Mode, ÷ 200 Prescale

* Unused bits: read as zeros

output will remain in this new state until the next Time Out pulse occurs. Thus the output will complete one full cycle for each two Time Out pulses.

If, for example, the prescaler were programmed to divide by ten, and the Timer Data Register were loaded with 100 (decimal), the main counter would decrement once for every ten cycles of the timer clock. A Time Out pulse will occur (hence an interrupt if that channel is enabled) every 1000 cycles of the timer clock, and the timer output will complete one full cycle every 2000 cycles of the timer clock.

The main counter is an 8-bit binary down counter. It may be read at any time by reading the Timer Data Register. The information read is the information last clocked into the timer read register when the $\overline{DS}$ pin had last gone high prior to the current read cycle. When written, data is loaded into the Timer Data Register, and the main counter, if the timer is stopped. If the Timer Data Register is written while the timer is running, the new word is not loaded into the timer until it counts through H"01". However, if the timer is written while it is counting through H"01", an indeterminate value will be written into the time constant register. This may be circumvented by ensuring that the data register is not written when the count is H"01".

If the main counter is loaded with "01", a Time Out Pulse will occur every time the prescaler presents a count pulse to the main counter. If loaded with "00", a Time Out pulse will occur after every 256 count pulses.

Changing the prescale value with the timer running can cause the first Time Out pulse to occur at an indeterminate time, (no less than one nor more than 200 timer clock cycles times the number in the time constant register), but subsequent Time Out pulses will then occur at the correct interval.

In addition to the delay mode described above, Timers A and B can also function in the Pulse Width Measurement mode or in the Event Count mode. In either of these two modes, an auxiliary control signal is required. The auxiliary control input for Timer A is TAI, and for Timer B, TBI is used. The interrupt channels associated with I4 and I3 are used for TAI and TBI, respectively, in Pulse Width mode. See Figure 15.

The pulse width measurement mode functions much like the delay mode. However, in this mode, the auxiliary control signal on TAI or TBI acts as an enable to the timer. When the control signal on TAI or TBI is inactive, the timer will be stopped. When it is active, the prescaler and main counter are allowed to run. Thus the width of the active pulse on TAI or TBI is determined by the number of timer counts which occur while the pulse allows the timer to run. The active state of the signal on TAI or TBI is dependent upon the associated Interrupt Channel's edge bit (GPIP 4 for TAI and GPIP 3 for TBI; see Active Edge Register in Figure 5.) If the edge bit associated with the TAI or TBI input is a one, it will be active high; thus the timer will be allowed to run when the input is at a high level. If the edge bit is a zero, the TAI or TBI input will be active low. As previously stated, the

TIMER DATA REGISTERS (A, B, C, AND D)

Figure 13

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Port F (TADR)								
Port 10 (TBDR)								
Port 11 (TCDR)								
Port 12 (TDDR)								

TIMER C AND D CONTROL REGISTER

Figure 14

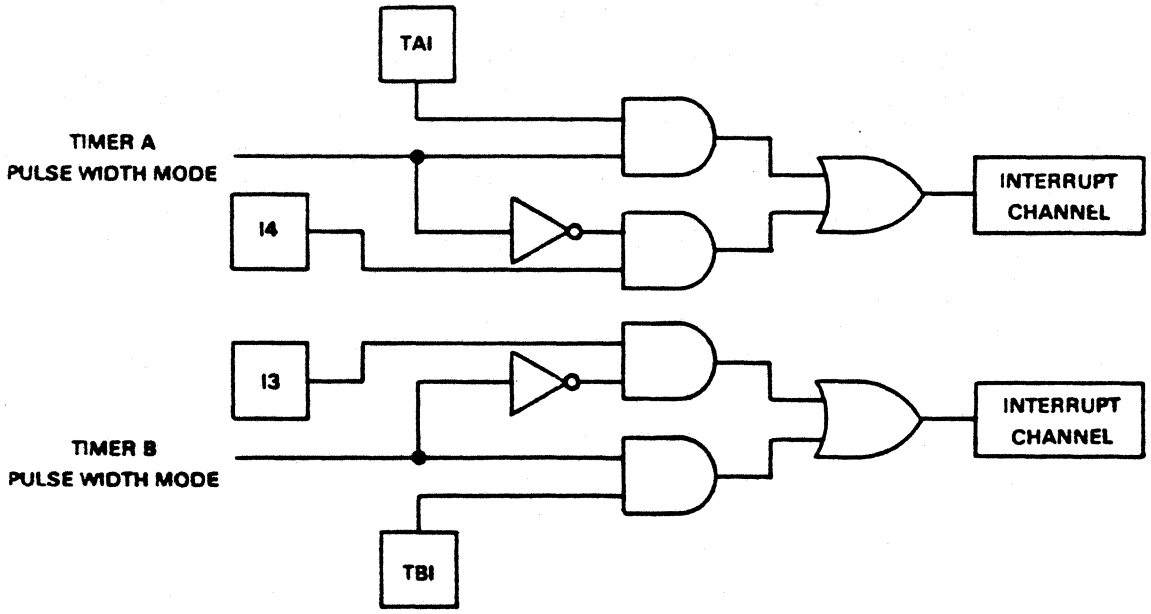
	*	CC_2	CC_1	CC_0	*	DC_2	DC_1	DC_0
Port E (TCDCR)								

C_2	C_1	C_0	
0	0	0	Timer Stopped
0	0	1	Delay Mode, ÷ 4 Prescale
0	1	0	Delay Mode, ÷ 10 Prescale
0	1	1	Delay Mode, ÷ 16 Prescale
1	0	0	Delay Mode, ÷ 50 Prescale
1	0	1	Delay Mode, ÷ 64 Prescale
1	1	0	Delay Mode, ÷ 100 Prescale
1	1	1	Delay Mode, ÷ 200 Prescale

* Unused bits: read as zeros

992

A CONCEPTUAL CIRCUIT OF THE MFP TIMERS IN THE PULSE WIDTH MEASUREMENT MODE
Figure 15



interrupt channel (I3 or I4) associated with the input still functions when the timer is used in the pulse width measurement mode. However, if the timer is programmed for the pulse width measurement mode, the interrupt caused by transitions on the associated TAI or TBI input will occur on the opposite transition.

For example, if the edge bit associated with the TAI input (AER-GPIP 4) is a one, an interrupt would normally be generated on the 0-1 transition of the I4 input signal. If the timer associated with this input (Timer A) is placed in the pulse width measurement mode, the interrupt will occur on the 1-0 transition of the TAI signal instead. Because the edge bit (AER-GPIP 4) is a one, Timer A will be allowed to count while the input is high. When the TAI input makes the high to low transition, Timer A will stop, and it is at this point that the interrupt will occur (assuming that the channel is enabled). This allows the interrupt to signal the CPU that the pulse being measured has terminated; thus Timer A may now be read to determine the pulse width. (Again note that I3 and I4 may still be used for I/O when the timer is in the pulse width measurement mode.) If Timer A is re-programmed for another mode, interrupts will again occur on the transition, as normally defined by the edge bit. Note that, like changing the edge bit, placing the timer into or taking it out of the pulse width mode can produce a transition on the signal to the interrupt channel and may cause an interrupt. If measuring consecutive pulses, it is obvious that one must read the contents of the timer and then reinitialize the main counter by writing to the timer data register. If the timer data register is written while the pulse is going to the active state, the write operation may result in an indeterminate value being written into the main counter. If the timer is written after the pulse goes active, the timer counts from the previous contents, and when it counts through H"01", the correct value is written into the timer. The pulse width then includes counts from before the timer was reloaded.

In the event count mode, the prescaler is disabled. Each time the control input on TAI or TBI makes an active transition as defined by the associated Interrupt Channel's edge bit, a count pulse will be generated, and the main counter will decrement. In all other respects, the timer functions as previously described. Altering the edge bit while the timer is in the event count mode can produce a count pulse. The interrupt channel associated with the input (I3 for TBI or I4 for TAI) is allowed to function normally. To count transitions reliably, the input must remain in each state (1/O) for a length of time equal to four periods of the timer clock; thus signals of a frequency up to one fourth of the timer clock can be counted.

The manner in which the timer output pins toggle states has previously been described. All timer outputs will be forced low by a device RESET. The output associated with Timers A and B will toggle on each Time Out pulse regardless of the mode the timers are programmed to. In addition, the outputs from Timers A and B can be forced low at any time by writing a "1" to the reset location in TACR and TBCR, respectively. The output will be forced to the low state during the WRITE operation, and at the conclusion of the operation, the output will again be free to toggle each time a Time Out pulse occurs. This feature will allow waveform generation.

During reset, the Timer Data Registers and the main counters are not reset. Also, if using the reset option on Timers A or B, one must make sure to keep the other bits in the correct state so as not to affect the operation of Timers A and B.

993

USART

Serial Communication is provided by a full-duplex double-buffered USART, which is capable of either asynchronous or synchronous operation. Variable word length and start/stop bit configurations are available under software control for asynchronous operation. For synchronous operation, a Sync Word is provided to establish synchronization during receive operations. The Sync Word will also be repeatedly transmitted when no other data is available for transmission. Moreover, the MK68901 allows stripping of all Sync Words received in synchronous operation. The handshake control lines RR (Receiver Ready) and TR (Transmitter Ready) allow DMA operation. Separate receive and transmit clocks are available, and separate receive and transmit status and data bytes allow independent operation of the transmit and receive sections.

The USART is provided with three Control/Status Registers and a Data Register. The USART Data Register form is illustrated in Figure 16. The programmer may specify operational parameters for the USART via the Control Register, as shown in Figure 17. Status of both the Receiver and Transmitter sections is accessed by means of the two Status Registers, as shown in Figures 18 and 19. Data written to the Data Register is passed to the transmitter, while reading the Data Register will access data received by the USART.

USART DATA REGISTER

Figure 16

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Port 17 (UDR)								

USART CONTROL REGISTER (UCR)

Figure 17

	UCR_7							UCR_0
Port 14 (UCR)	1 = ÷16 0 = ÷1	WL_1	WL_0	ST_1	ST_0	PARITY ENABLED ON 1	1 = EVEN 0 = ODD	*

* Unused bits; read as zero

÷16/÷1 : When this bit is zero, data will be clocked into and out of the receiver and transmitter at the frequency of their respective clocks. When this bit is loaded with a one, data will be clocked into and out of the receiver and transmitter at one sixteenth the frequency of their respective clocks. Additionally, when placed in the divide by sixteen mode, the receiver data transition resynchronization logic will be enabled.

WL0-WL1 : Word Length Control. These two bits set the length of the data word (exclusive of start bits, stop bits,and parity bits) as follows:

WL1	WL0	Word Length
0	0	8 bits
0	1	7 bits
1	0	6 bits
1	1	5 bits

ST0-ST1 : Start/Stop bit control (format control). These two bits set the format as follows

ST1	ST0	Start Bits	Stop Bits	Format
0	0	0	0	SYNC
0	1	1	1	ASYNC
†1	0	1	1½	ASYNC
1	1	1	2	ASYNC

† NOTE ÷16 only

PARITY : Parity Enabled. When set ("1"), parity will be checked by the receiver, parity will be calculated, and a parity bit will be inserted by the transmitter. When cleared ("0"), no parity check will be made and no parity bit will be inserted for transmission.

For a word length of 8 the MFP calculates the parity and appends it when transmitting a sync character. For shorter lengths, the parity must be stored in the Sync Character Register (SCR) along with the sync character.

E/O : Even-Odd. When set ("1"), even parity will be used if parity is enabled. When cleared ("0"), odd parity will be used if parity is enabled.

Note that the synchronous or asynchronous format may be selected independently of a ÷ 1 or ÷ 16 clock. Thus it is possible to clock data synchronously into the device but still use start and stop bits. In this mode, all normal asynchronous format features still apply. Data will be shifted in after a start bit is encountered, and a stop bit will be checked to determine proper framing. If a transmit underrun condition occurs, the output will be placed in a marking state, etc. It is conversely possible to clock data in asynchronously using a synchronous format. There is data transition detection logic built into the receive clock circuitry which will re-synchronize the internal shift clock on each data transition so that, with sufficiently frequent data transitions, start bits are not required. In this mode, all other common synchronous features function normally. This re-synchronization logic is only active in ÷ 16 clock mode.

RECEIVER

The receiver section of the USART is configured by the UCR as previously described. The status of the receiver can be determined by reading and writing to the Receiver Status Register (RSR). The RSR is configured as follows:

RECEIVER STATUS REGISTER (RSR)

Figure 18

	RSR_7							RSR_0
Port 15 (RSR)	BUFFER FULL	OVERRUN ERROR	PARITY ERROR	FRAME ERROR	FOUND/$\overline{\text{SEARCH}}$ OR BREAK DETECT	MATCH/CHARACTER IN PROGRESS	SYNC STRIP ENABLE	RECEIVER ENABLE

BF : Buffer Full. This bit is set when the incoming word is transferred to the receive buffer. The bit is cleared when the receive buffer is read by reading the UDR. This bit of the RSR is read only.

OE : Overrun Error.This flag is set if the incoming word is completely received and due to be transferred to the receive buffer, but the last word in the receive buffer has not yet been read. When this condition occurs, the word in the receive buffer is not overwritten by the new word. Note that the status flags always reflect the status of the data word currently in the receive buffer. As such, the OE flag is not actually set until the good word currently in the buffer has been read. The interrupt associated with this error will also not be generated until the old word in the receive buffer has been read.

OE flag is cleared by reading the receiver status register, and new data words cannot be shifted to the receive buffer until this is done.

PE : Parity Error. This flag is set if the word received has a parity error. The flag is set when the received word is transferred from the shift register to the receive buffer if the error condition exists. The flag is cleared when the next word which does not have a parity error is transferred to the receive buffer.

FE : Frame Error. This flag only applies to the asynchronous format. A frame error is defined as a non-zero data word which is not followed by a stop bit. Like the PE flag, the FE flag is set or cleared when a word is transferred to the receive buffer.

F/$\overline{\text{S}}$: Found/$\overline{\text{Search}}$. This combination control bit and flag bit is only used with the synchronous format. It can be set or cleared by writing to this bit of the RSR. When this bit is cleared, the receiver is placed in the search mode. In this mode, a bit by bit comparison of the incoming data to the character in the Sync Character Register (SCR) is made. The word length counter is disabled. When a match is found, this bit will be set automatically, and the word length counter will start as sync has now been achieved. An interrupt will be generated on the receive error channel when the match occurs. The word just shifted in will, of necessity, be equal to the sync character, and it will not be transferred to the receive buffer.

B : Break. This flag is used only when the asynchronous format is selected. This flag will be set when an all zero data word, followed by no stop bit, is received. The flag will stay set until both a non-zero bit is received and the RSR has been read at least once since the flag was set. Break indication will not occur if the receive buffer is full.

M/CIP : Match/Character in Progress. If the synchronous format is selected, this flag is the Match flag. It will be set each time the word transferred to the receive buffer matches the sync character. It will be reset each time the word transferred to the receive buffer does not match the sync

995

character. If the asynchronous format is selected, this flag represents Character in Progress. It will be set upon a start bit detect and cleared at the end of the word.

SS : Sync Strip Enable. If this bit is set to a one, data words that match the sync character will not be loaded into the receive buffer, and no buffer full signal will be generated.

RE: : Receiver Enable. This control bit is used to enable or disable the receiver. If a zero is written to this bit of the RSR, the receiver will turn off immediately. All flags including the F/$\overline{S}$ bit will be cleared. If a one is written to this bit, normal receiver operation is enabled. The receive clock has to be running before the receiver is enabled.

There are two interrupt channels associated with the receiver. One channel is used for the normal Buffer Full condition, while the other channel is used whenever an error condition occurs. Only one interrupt is generated per word received, but dedicating two channels allows separate vectors: one for the normal condition, and one for an error condition. If the error channel is disabled, an interrupt will be generated via the Buffer Full Channel, whether the word received is normal or in error. Those conditions which produce an interrupt via the error channel are: Overrun, Parity Error, Frame Error, Sync Found, and Break. If a received word has an error associated with it, and the error interrupt channel is enabled, an interrupt will occur on the error channel only.

Each time a word is transferred into the receive buffer, a corresponding set of flags is latched into the RSR. No flags (except CIP) are allowed to change until the data word has been read from the receive buffer. Reading the receive buffer allows a new data word to be transferred to the receive buffer when it is received. Thus one should first read the RSR then read the receive buffer (UDR) to ensure that the flags just read match the data word just read. If done in the reverse order, it is possible that subsequent to reading the data word from the receive buffer, but prior to reading the RSR, a new word may be received and transferred to the receive buffer and, with it, its associated flags latched into the RSR. Thus, when the RSR is read, those flags may actually correspond to a different data word. It is good practice, also, to read the RSR prior to a data read as, when an overrun error occurs, the receiver will not assemble new characters until the RSR has been read.

As previously stated, when overrun occurs, the OE flag will not be set and the associated interrupt will not be generated until the receive buffer has been read. If a break occurs, and the receive buffer has not yet been read, only the B flag will be set (OE will not be set). Again, this flag will not be set until the last valid word has been read from the receive buffer. If the break condition ends and another whole data word is received before the receive buffer is read, both the B and OE flags will be set once the receive buffer is read.

If a break occurs while the OE flag is set, the B flag will also be set.

A break generates an interrupt when the condition occurs and again when the condition ends. If the break condition ends before it is acknowledged by reading the RSR, the receiver error interrupt indicating end of break will be generated once the RSR is read.

Anytime the asynchronous format is selected, start bit detection is enabled. New data is not shifted into the shift register until a zero bit is detected. If a ÷16 clock is selected, along with the the asynchronous format, false start bit detection is also enabled. Any transition has to be stable for 3 positive going edges of the receive clock to be called a valid transition. For a start bit to be good, a valid 0-1 transition must not occur for 8 positive clock transitions after the initial valid 1-0 transition.

After a good start bit has been detected, valid transitions in the data are checked for continously. When a valid transition is detected, the counter is forced to state zero, and no more transition checking is started until state four. At state eight, the "previous state" of the transition checking logic is clocked into the receiver.

As a result of this resynchronization logic, it is possible to run with asynchronous clocks without start and stop bits if there are sufficient valid transitions in the data stream. This logic also makes the unit more tolerant of clock skew for normal asynchronous communications than a device which employs only start bit synchronization.

TRANSMITTER STATUS REGISTER (TSR)

Figure 19

	TSR_7							TSR_0
Port 16 (TSR)	BUFFER EMPTY	UNDERRUN ERROR	AUTO TURNAROUND	END OF TRANSMISSION	BREAK	HIGH	LOW	TRANSMITTER ENABLE

TRANSMITTER

The transmitter section of the USART is configured as to format, word length, etc. by the UCR, as previously described. The status of the transmitter can be determined by reading or writing the Transmitter Status Register (TSR). The TSR is configured as follows:

BE : Buffer Empty. This status bit is set when the word in the transmit buffer is transferred to the output shift register and thus the transmit buffer may be reloaded with the next data word. The flag is cleared when the transmit buffer is reloaded. The transmit buffer is loaded by writing to the UDR.

UE : This bit is set when the last word has been shifted out of the transmit shift register before a new word has been loaded into the transmit buffer. It is not necessary to clear this bit before loading the UDR.

This bit may be cleared by either reading the TSR or by disabling the transmitter. After the setting of the UE bit, one full transmitter clock cycle is required before this bit can be cleared by a read. The timing in some systems may allow a read of the TSR before the required clock cycle has been completed. This would result in the UE bit not being cleared until the following read. To avoid this problem, a dummy read of the TSR should be performed at the end of the UE service routine.

Only one underrun error may be generated between loads of the UDR regardless of the number of transmitter clock cycles between UDR loads.

AT : This bit causes the receiver to be enabled at the end of the transmission of the last word in the transmitter if the transmitter has been disabled. The AT bit is cleared at the end of the transmission.

END : End of transmission. When the transmitter is turned off with a character still in the output shift register, transmission will continue until that character is shifted out. Once it has cleared the output register, the END bit will be set. If no character is being transmitted when the transmitter is disabled, the transmitter will stop at the next rising edge of the internal shift clock, and END will immediately be set. The END bit is cleared by re-enabling the transmitter.

B : Break. This control bit will cause a break to be transmitted. When a "1" is written to the B bit of the TSR, a break will be transmitted upon completion of the character (if any) currently being transmitted. A break will continue to be transmitted until the B bit is cleared by writing a "0" to this bit of the TSR. At that time, normal transmission will resume. The B bit has no function in the synchronous format. Setting the "B" bit to a one keeps the "BE" bit from being set to a one. So, if there were a word in the buffer at the start of break, it would remain there until the end of break, at which time it would be transmitted (if the transmitter is still enabled). If the buffer were not full at the start of break, it could be written at any time during the break. If the buffer is empty at the end of break, the underrun flag will be set (unless the transmitter is disabled).

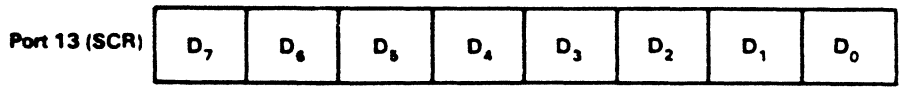
The BREAK bit cannot be set until the transmitter has been enabled and the transmitter has had sufficient time (one clock cycle) to perform the internal reset and initialization functions.

H,L : High and Low. These two control bits are used to configure the transmitter output, when the transmitter is disabled, as follows:

H	L	Output State
0	0	Hi-Z
0	1	Low ("0")
1	0	High
1	1	Loop-Connects transmitter output to receiver input, and TC to Receiver Clock (RC and SI are not used; they are bypassed internally). In loop back mode, transmitter output goes high when disabled.

997

SYNC CHARACTER REGISTER
Figure 20



Altering these two bits after Transmitter Enable (XE) is set will alter the output state until END is false. These bits should be set prior to enabling the transmitter. The state of these bits determine the state of the first transmitted character after the transmitter is enabled. If the high impedance mode was selected prior to the transmitter being enabled, the first bit transmitted is indeterminate.

XE : Transmitter Enable. This control bit is used to enable or disable the transmitter. When set, the transmitter is enabled. When cleared, the transmitter will be disabled. If disabled, any word currently in the output register will continue to be transmitted until finished. If a break is being transmitted when XE is cleared, the transmitter will turn off at the end of the break character boundary, and no end of break stop bit is transmitted. The transmit clock must be running before the transmitter is enabled. A "one" bit always precedes the first word out of the transmitter after the transmitter is enabled There is a delay between the time the transmitter enable bit is written and when the transmitter reset goes low; therefore, the H & L bits should be written with the desired state prior to enabling the transmitter.

Like the receiver section, there are two separate interrupt channels associated with the transmitter. The buffer Empty condition causes an interrupt via one channel, while the Underrun and END conditions will cause an interrupt via the second channel. When underrun occurs in the synchronous format, the character in the SCR will be transmitted until a new word is loaded into the transmit buffer. In the asynchronous format, a "Mark" will be continuously transmitted when underrun occurs.

The transmit buffer can be loaded prior to enabling the transmitter. When the transmitter is disabled, any character currently in the process of being transmitted will continue to conclusion, but any character in the transmit buffer will not be transmitted and will remain in the buffer. Thus no buffer empty interrupt will occur nor will the BE flag be set. If the buffer were already empty, the BE flag would be set and would remain set. When the transmitter is disabled with a character in the output register but with no character in the transmit buffer, an Underrun Error will not occur when the character in progress concludes.

Often it is necessary to send a break for some particular period. To aid in timing a break transmission, a transmit error interrupt will be generated at every normal character boundary time during a break transmission. The status register information is unaffected by this error condition interrupt. It should be noted that an underrun error, if present, must be cleared from the TSR, and the interrrupt pending register must be cleared of pending transmitter errors at the beginning of the break transmission or no interrupts will be generated at the character boundary time.

If the synchronous format is selected, the sync character should be loaded into the Sync Character Register (SCR) as shown in Figure 20. This character is compared to the received serial data during a Search, and will be continuously transmitted during an underrun condition

All flags in the RSR or TSR will continue to function as described whether their associated interrupt channel is disabled or enabled. All interrupt channels are edge triggered and, in many cases, it is the actual output of a flag bit or flag bits which is coupled to the interrupt channel. Thus, if a normal interrupt producing condition occurs while the interrupt channel is disabled, no interrupt would be produced even if the channel was subsequently enabled, because a transition did not occur while the interrupt channel was enabled. That particular flag bit would have to occur a second time before another "edge" was produced, causing an interrupt to be generated.

Error conditions in the USART are determined by monitoring the Receive Status Register and the Transmitter Status Register. These error conditions are only valid for each word boundary and are not latched. When executing block transfers of data, it is necessary to save any errors so that they can be checked at the end of a block. In order to save error conditions during data transfer, the MK68901 MFP interrupt controller may be used by enabling error interrupts for the desired channel (Receive error or Transmit error) and by masking these bits off. Once the transfer is complete, the Interrupt Pending Register can be polled,to determine the presence of a pending error interrupt, and therefore an error.

Unused bits in the sync character register are zeroed out; therefore, word length should be set up prior to writing the sync word in some cases. Sync word length is the word length plus one when parity is enabled. The user has to determine the parity of the sync word when the word length

is not 8 bits. The MK68901 MFP does not add a parity bit to the sync word if the word length is less than 8 bits. The extra bit in the sync word is transmitted as the parity bit. With a word length of eight, and parity selected, the parity bit for the sync word is computed and added on by the MK68901 MFP.

$\overline{RR}$ RECEIVER READY

$\overline{RR}$ is asserted when the Buffer Full bit is set in the RSR unless a parity error or frame error is detected by the receiver.

$\overline{TR}$ TRANSMITTER READY

$\overline{TR}$ is asserted when the Buffer Empty bit is set in the TSR unless a break is currently being transmitted.

REGISTER ACCESSES

All register accesses are dependent on CLK as shown in the timing diagrams. To read a register, $\overline{CS}$ and $\overline{DS}$ must be asserted, and R $\overline{W}$ must be high. The internal read control signal is essentially the combination of $\overline{CS}$, $\overline{DS}$, and RD/$\overline{WR}$. Thus, the read operation will begin when $\overline{CS}$ and $\overline{DS}$ go active and will end when either $\overline{CS}$ or $\overline{DS}$ goes inactive. The address bus must be stable prior to the start of the operation and must remain stable until the end of the operation. Unless a read operation or interrupt acknowledge cycle is in progress the data bus (D_0-D_7) will remain in the tri-state condition.

To write a register, $\overline{CS}$ and $\overline{DS}$ must be asserted and R/$\overline{W}$ must be low. The address must be stable prior to the start of the operation and must remain stable until the end of the operation. After the MK68901 asserts $\overline{DTACK}$, the CPU negates $\overline{DS}$. At this time, the MFP latches the data bus and writes the contents into the appropriate register. Also, when $\overline{DS}$ is negated, the MFP rescinds $\overline{DTACK}$.

For an interrupt acknowledge, the operation starts when $\overline{IACK}$ goes low, and ends when $\overline{IACK}$ goes high. The data bus is tri-stated when either $\overline{IACK}$ or $\overline{DS}$ goes high.

MK68901 ELECTRICAL SPECIFICATIONS - PRELIMINARY

ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias ..... -25°C to +100°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin with Respect to Ground ..... -0.3 V to +7 V
Power Dissipation ..... 1.5 W

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

D. C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = +5 V ± 5% unless otherwise specified.

SYM	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{IH}	Input High Voltage	2.0	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage (except $\overline{DTACK}$)	2.4		V	I_{OH} = -120 µA
V_{OL}	Output Low Voltage (except $\overline{DTACK}$)		0.5	V	I_{OL} = 2.0 mA
I_{LL}	Power Supply Current		180	mA	Outputs Open
I_{LI}	Input Leakage Current		±10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.5 V
I_{OH}	$\overline{DTACK}$ output source current		-400	µA	V_{OUT} = 2.4
I_{OL}	$\overline{DTACK}$ output sink current		5.3	mA	V_{OUT} = 0.5

All voltages are referenced to ground

999

CAPACITANCE

T_A = 25°C, f = 1 MHz unmeasured pins returned to ground.

SYM	PARAMETER	MAX	UNIT	TEST CONDITION
C_{IN}	Input Capacitance	10	pf	Unmeasured pins returned to ground
C_{OUT}	Tri-state Output Capacitance	10	pf	

AC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ± 5%, GND = 0 Vdc, T_A = 0°C to 70°C)

NUM	CHARACTERISTIC	MK68901 MIN	MK68901 MAX	UNIT	FIG	NOTE
1	$\overline{CS}$, $\overline{DS}$ Width High	50		ns	21,22	
2	R/$\overline{W}$, A1-A5 Valid to falling $\overline{CS}$ (Setup)	0		ns	21,22	
3	Data Valid Prior to Rising $\overline{DS}$ (Setup)	280		ns	21	
4	$\overline{CS}$, $\overline{IACK}$ Valid to Falling Clock (Setup)	50		ns	21-24	3
5	CLK Low to $\overline{DTACK}$ Low		220	ns	21,22	
6	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ high		60	ns	21-24	
7	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ Tri-state		100	ns	21-24	
8	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to Data Invalid (Hold Time)	0		ns	21-24	
9	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to Data Tri-state		50	ns	21-24	
10	$\overline{CS}$ or $\overline{DS}$ High to R/$\overline{W}$, A1-A5 Invalid (Hold Time)	0		ns	21,22	
11	Data Valid from $\overline{CS}$ Low		310	ns	21	
12	Read Data Valid to $\overline{DTACK}$ Low (Setup Time)	50		ns	21	
13	$\overline{DTACK}$ Low to $\overline{DS}$, $\overline{CS}$ or $\overline{IACK}$ High (Hold Time)	0		ns	21-24	
14	$\overline{IEI}$ low to falling $\overline{CLK}$ (Setup)	50		ns	23	2
15	$\overline{IEO}$ Valid from Clock Low (Delay)		180	ns	23	1
16	Data Valid From Clock Low (Delay)		300	ns	23	
17	$\overline{IEO}$ Invalid from $\overline{IACK}$ High (Delay)		150	ns	23,24	
18	$\overline{DTACK}$ Low from Clock High (Delay)		180	ns	23,24	2
19	$\overline{IEO}$ Valid from $\overline{IEI}$ Low (Delay)		100	ns	24	1
20	Data Valid from $\overline{IEI}$ Low (Delay)		220	ns	24	
21	Clock Cycle Time	250	1000	ns	21-24	
22	Clock Width Low	110		ns	21-24	
23	Clock Width High	110		ns	21-24	
24	$\overline{CS}$, $\overline{IACK}$ Inactive to Rising Clock (Setup)	100		ns	21-24	4
25	I/O Minimum Active Pulse Width	100		ns		
26	$\overline{IACK}$ width High	150		ns	23-24	

AC ELECTRICAL CHARACTERISTICS (Continued) (V_{CC} = 5.0 Vdc ± 5%, GND = 0 Vdc, T_A = 0°C to 70°C)

NUM	CHARACTERISTIC	MK68901 MIN	MK68901 MAX	UNIT	FIG	NOTE
27	I/O Data valid from Rising $\overline{CS}$ or $\overline{DS}$		450	ns		
28	Receiver Ready Delay from Rising RC		600	ns		
29	Transmitter Ready Delay from Rising TC		600	ns		
30	Timer Output Low from Rising Edge of $\overline{CS}$ or $\overline{DS}$ (A & B) (Reset T_{OUT})		450	ns		
31	T_{OUT} Valid from Internal Timeout		2 t_{CLK} +300	ns		
32	Timer Clock Low Time	110		ns		
33	Timer Clock High Time	110		ns		
34	Timer Clock Cycle Time	250	1000	ns		
35	$\overline{RESET}$ Low Time	2		μs		
36	Delay to Falling $\overline{INTR}$ from External Interrupt Active Transition		380	ns		
37	Transmitter Internal Interrupt Delay from Rising of Falling Edge of TC	550		ns		
38	Receiver Buffer Full Interrupt Transition Delay from Rising Edge of RC	800		ns		
39	Receiver Error Interrupt Transition Delay from Falling Edge of RC	800		ns		
40	Serial In Set Up Time to Rising Edge of RC (Divide by one only)	80		ns		
41	Data Hold Time from rising edge of RC (Divide by one only)	350		ns		
42	Serial Output Data Valid from Falling Edge of TC (÷1)		440	ns		
43	Transmitter Clock Low Time	500		ns		
44	Transmitter Clock High Time	500		ns		
45	Transmitter Clock Cycle Time	1.05	∞	μs		
46	Receiver Clock Low Time	500		ns		
47	Receiver Clock High Time	500		ns		
48	Receiver Clock Cycle Time	1.05	∞	μs		
49	$\overline{CS}$, $\overline{IACK}$, $\overline{DS}$ Width Low		80	T_{CLK}		
50	Serial Output Data Valid from Falling Edge of TC (÷16)		490	ns		

NOTES:

1 $\overline{IEO}$ only goes low if no acknowledgeable interrupt is pending. If $\overline{IEO}$ goes low, $\overline{DTACK}$ and the data bus remain tri-stated.

2 $\overline{DTACK}$ will go low at A if spec. 14 is met; otherwise, $\overline{DTACK}$ will go low at B.

3 If the setup time is not met, $\overline{CS}$ or $\overline{IACK}$ will not be recognized until the next falling CLK.

4 If this setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.

1001

TIMER A. C. CHARACTERISTICS

Definitions:

Error = Indicated Time Value - Actual Time Value

tpsc = t_{CLK} x Prescale Value

Internal Timer Mode

Single Interval Error (free running) (Note 2)	± 100 ns
Cumulative Internal Error	0
Error Between Two Timer Reads	± (tpsc + 4 t_{CLK})
Start Timer to Stop Timer Error	2 t_{CLK} + 100 ns to - (tpsc + 6t_{CLK} - 100 ns)
Start Timer to Read Timer Error	0 to - (tpsc + 6t_{CLK} + 400 ns)
Start Timer to Interrupt Request Error (Note 3)	-2 t_{CLK} to - (4t_{CLK} +800 ns)

Pulse Width Measurement Mode

Measurement Accuracy (Note 1)	2 t_{CLK} to - (tpsc + 4t_{CLK})
Minimum Pulse Width	4t_{CLK}

Event Counter Mode

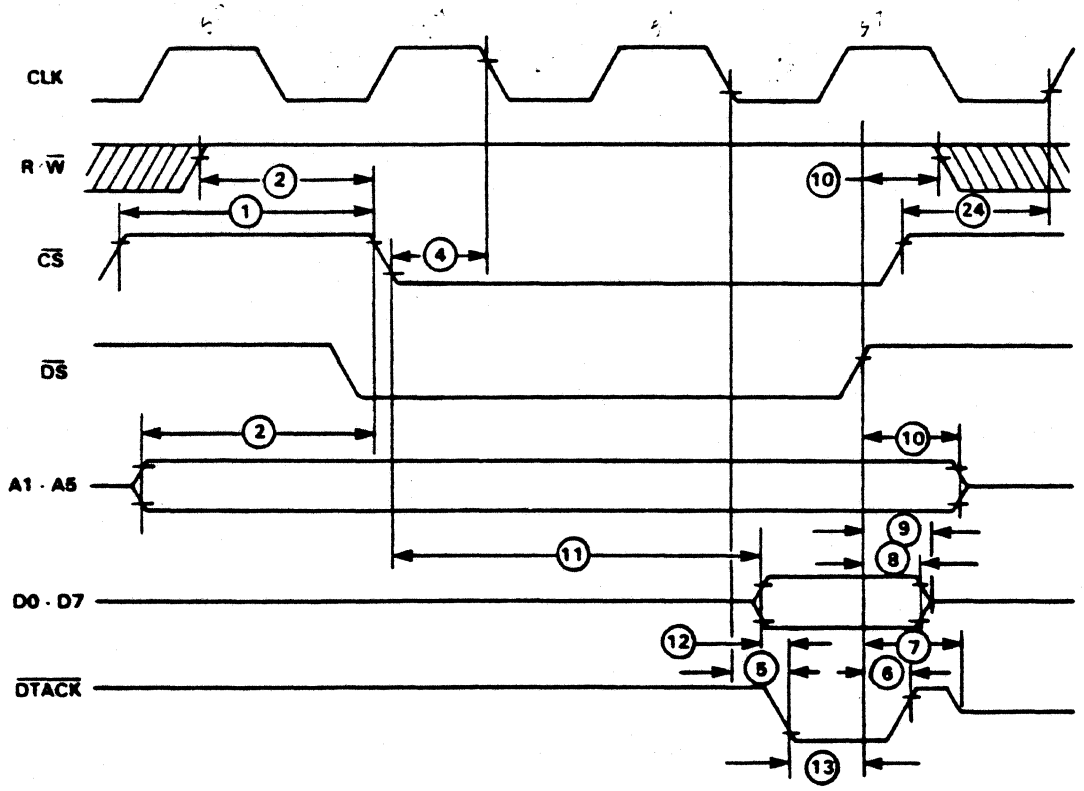
Minimum Active Time of TAI, TBI	4t_{CLK}
Minimum Inactive Time of TAI, TBI	4t_{CLK}

NOTES:

1. Error may be cumulative if repetitively performed
2. Error with respect to T_{OUT} or INT if note 3 is true
3. Assuming it is possible for the timer to make an interrupt request immediately

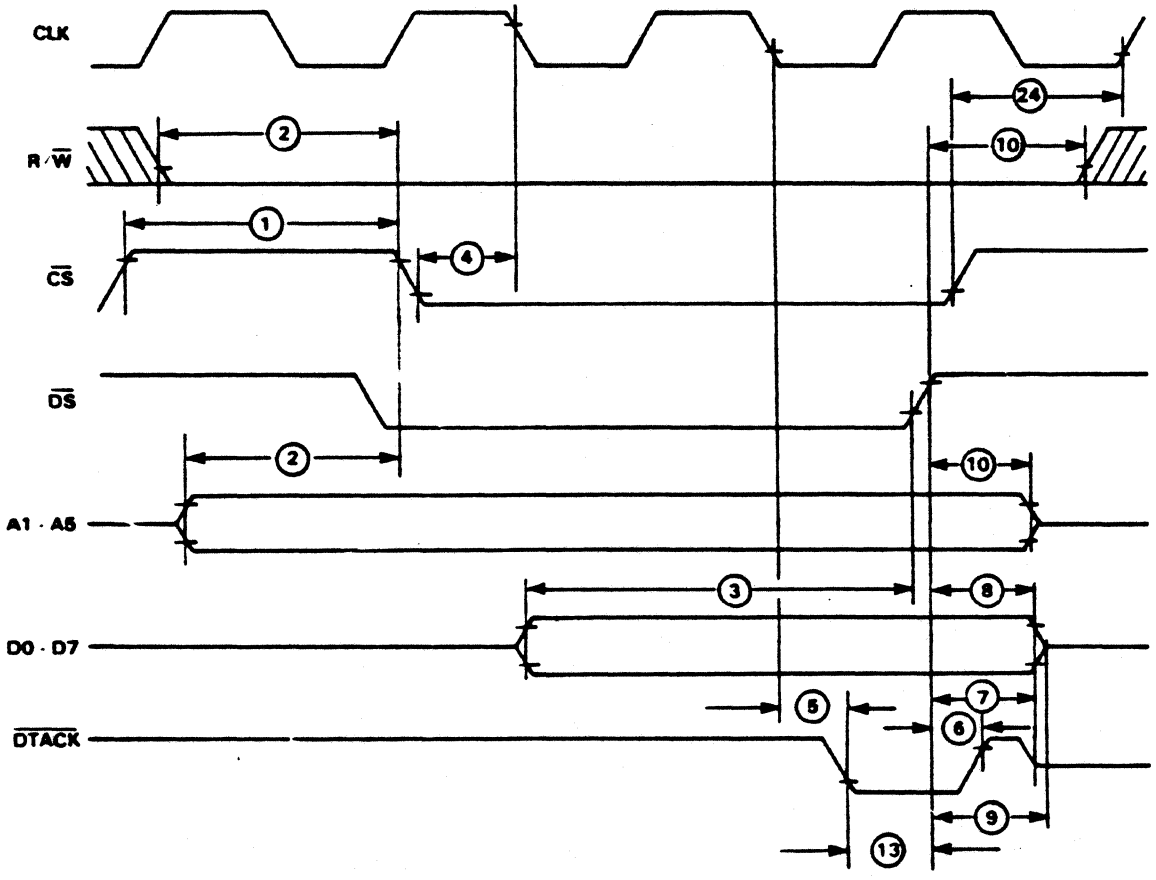
READ CYCLE

Figure 21



WRITE CYCLE

Figure 22

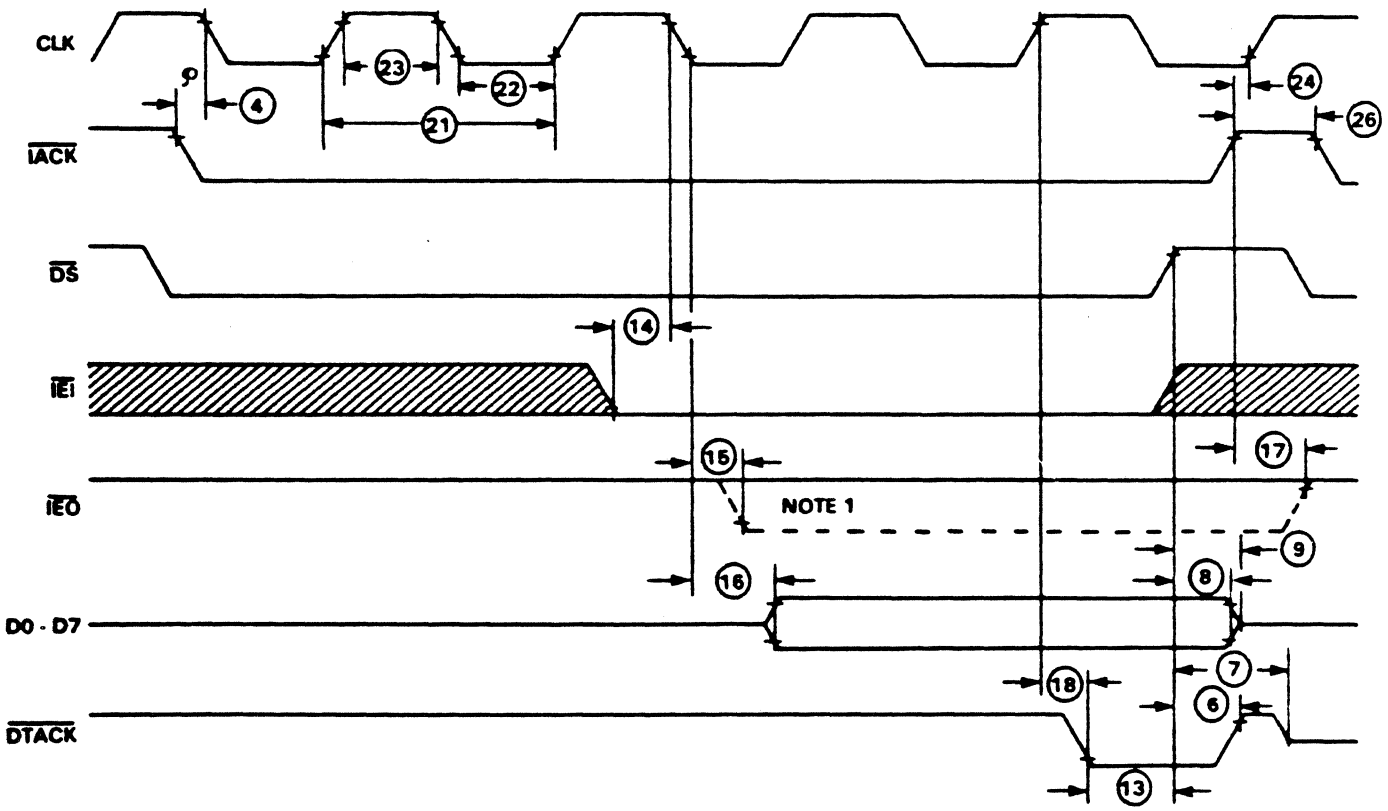


NOTE:
$\overline{CS}$ and $\overline{IACK}$ must be a function of $\overline{DS}$.

1003

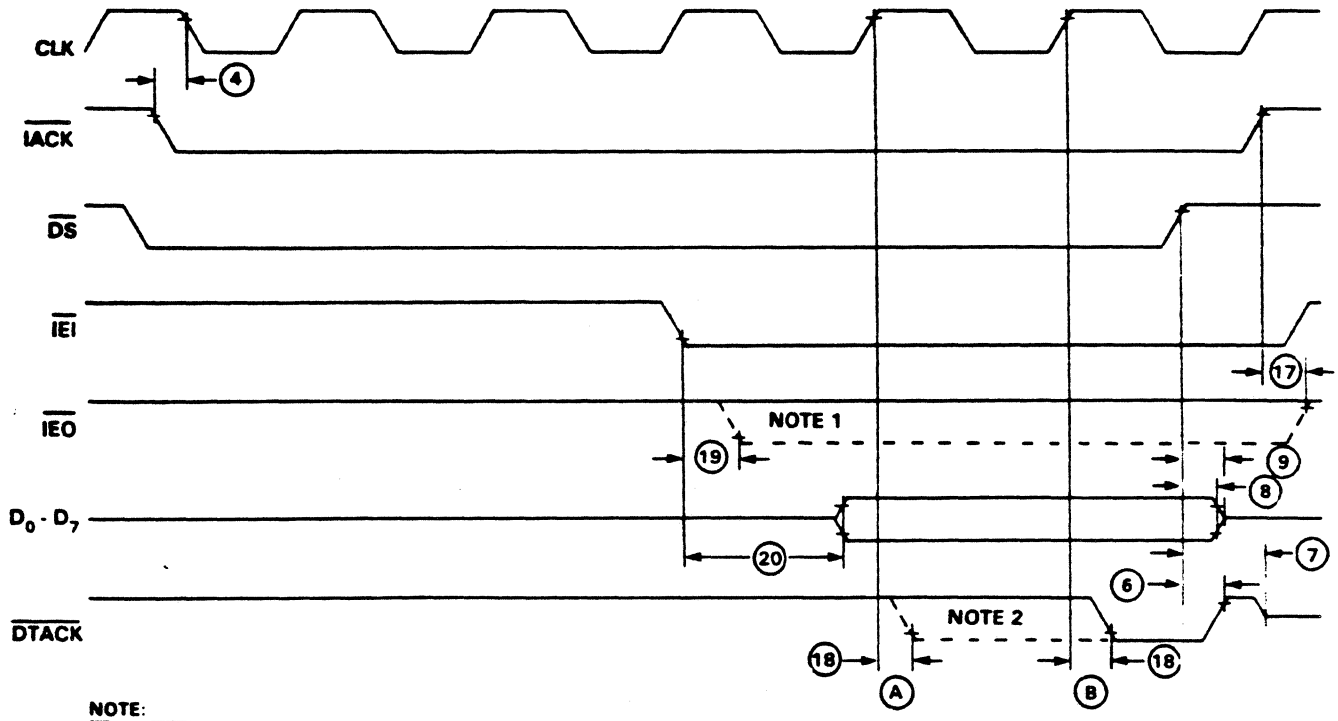
INTERRUPT ACKNOWLEDGE ($\overline{IEI}$ LOW)

Figure 23



INTERRUPT ACKNOWLEDGE CYCLE ($\overline{IEI}$ HIGH)

Figure 24

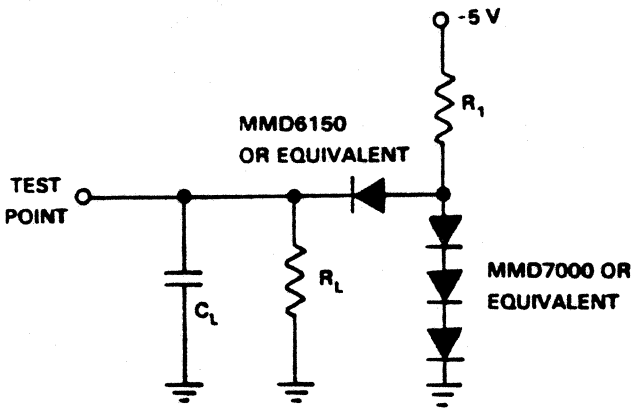


NOTE:
$\overline{CS}$ and $\overline{IACK}$ must be a function of $\overline{DS}$

1004

TYPICAL OUTPUT

Figure 25



for all outputs except $\overline{\text{DTACK}}$

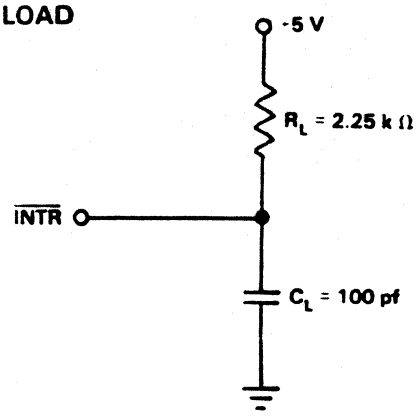
C_L = 100 pf
R_L = 20 k Ω
R_1 = 1.90 k Ω

for $\overline{\text{DTACK}}$

C_L = 130 pf
R_L = 6 k Ω
R_1 = 740 Ω

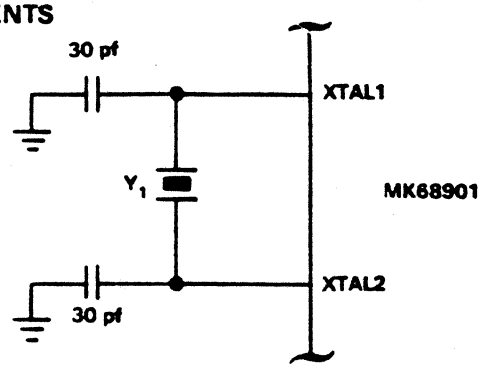
$\overline{\text{INTR}}$ TEST LOAD

Figure 26



MK68901 MFP EXTERNAL OSCILLATOR COMPONENTS

Figure 27



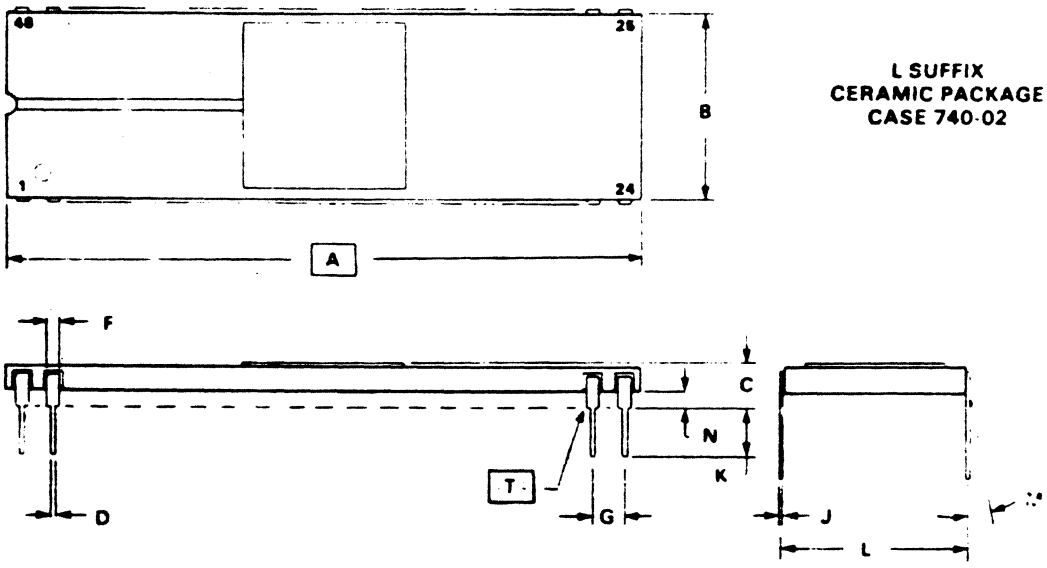
CRYSTAL PARAMETERS:

Parallel resonance, fundamental mode AT cut
$R_S \leq 150\ \Omega$ (F_R = 2.8 - 4.0 MHz);
$R_S \leq 300\ \Omega$ (F_R = 2.0 - 2.7 MHz)
C_L = 18 pf; C_M = 0.02 pf; C_h = 5 pf; L_M = 96 MHz
F_R (typ) = 2.4576 MHz

1005

MK68901 PIN PACKAGE DRAWING

Figure 28



L SUFFIX
CERAMIC PACKAGE
CASE 740-02

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	60 35	61 57	2 376	2 424
B	14 63	15 34	0 576	0 604
C	3 05	4 32	0 120	0 160
D	0 381	0 533	0 015	0 021
F	0 762	1 397	0 030	0 055
G	2 54 BSC		0 100 BSC	
J	0 203	0 330	0 008	0 013
K	2 54	4 19	0 100	0 165
L	14 99	15 65	0 590	0 616
M	0°	10°	0°	10°
N	1 016	1 524	0 040	0 060

NOTES
1 DIMENSION [A] IS DATUM
2 POSITIONAL TOLERANCE FOR LEADS
[Ø 0 25 (0 010) T AM]
3 [T] IS SEATING PLANE
4 DIMENSION "L" TO CENTER OF
LEADS WHEN FORMED PARALLEL
5 DIMENSIONING AND TOLERANCING
PER ANSI Y14 5, 1973

MK68901 ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX. CLOCK FREQUENCY	TEMPERATURE RANGE
MK68901	Ceramic	4.0 MHz	0° to 70°C

1006

UNITED TECHNOLOGIES MOSTEK

MICROCOMPUTER COMPONENTS

TEMPORARY ERRATA SHEET

MK68901 REV C

ERRATA SHEET FOR MK68901 REV C

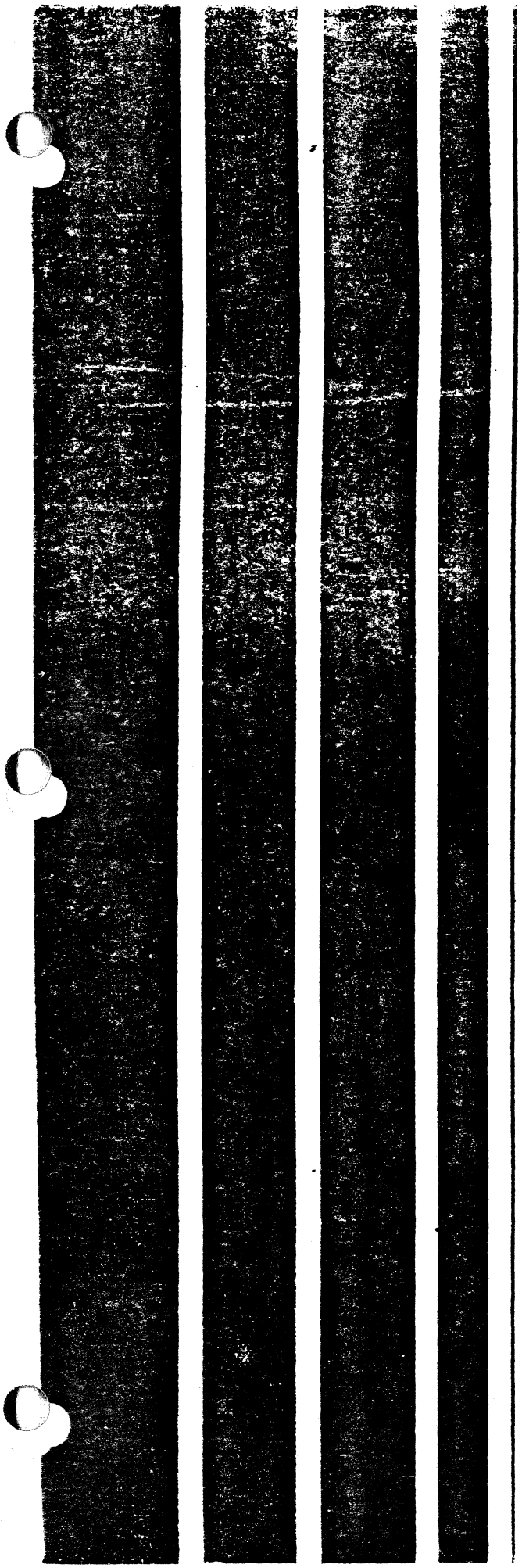
The counters are initially loaded by writing to the Timer Data Register. When the timers are stopped, the information that was written to the TDR may not be correctly transferred to the internal time constant register. This would cause the first count sequence of the timer to be incorrect, but subsequent count sequences would be correct. If the initial count sequence is critical to system operation, a read should be used to verify that the correct data was loaded. If the readback indicates an error, the write/read loop should be performed until the data is verified.

UNITED TECHNOLOGIES MOSTEK

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214 466-6000
In Europe, Contact: Mostek Brussels
270-272 Avenue de Tervuren (BTE21), B-1150 Brussels, Belgium;
Telephone: 762.18.80

PRINTED IN USA December 1982
Publication No. 4420330

1007



AY-3-8910/8912 PROGRAMMABLE SOUND GENERATOR DATA MANUAL

ARCHITECTURE

OPERATION

INTERFACING

MUSIC GENERATION

SOUND EFFECTS GENERATION

ELECTRICAL SPECIFICATIONS

TABLE OF CONTENTS—PAGE 2

FEBRUARY 1979

1008

Table of Contents

1009

List of Illustrations

1010

List of Illustrations (cont.)

1011

1 INTRODUCTION

It is apparent that any microprocessor is capable of producing acceptable sounds with only a transducer if the processor has no other tasks to perform while the sound is sustained. In real world microprocessor use, however, video games need refreshing, keyboards need scanning, etc. For example, in order to produce a single channel of ninth octave C (8372 Hz) the signal needs attention every sixty microseconds. Software required to produce this simple effect and still perform other activities would in the least be very complex if not impossible. In the extreme, random noise requires periodic attention even more frequently.

This need for software-produced sounds without the constant attention of the processor is now satisfied with the availability of the General Instrument AY-3-8910 and AY-3-8912 Programmable Sound Generators.

1.1 Description

The AY-3-8910/8912 Programmable Sound Generator (PSG) is a Large Scale Integrated Circuit which can produce a wide variety of complex sounds under software control. The AY-3-8910/8912 is manufactured in GI's N-Channel Ion Implant Process. Operation requires a single 5V power supply, a TTL compatible clock, and a microprocessor controller such as the GI 16-bit CP1600/1610 or one of GI's PIC 1650 series of 8-bit microcomputers.

The PSG is easily interfaced to any bus oriented system. Its flexibility makes it useful in applications such as music synthesis, sound effects generation, audible alarms, tone signalling and FSK modems. The analog sound outputs can each provide 4 bits of logarithmic digital to analog conversion, greatly enhancing the dynamic range of the sounds produced.

In order to perform sound effects while allowing the processor to continue its other tasks, the PSG can continue to produce sound after the initial commands have been given by the control processor. The fact that realistic sound production often involves more than one effect is satisfied by the three independently controllable channels available in the PSG.

All of the circuit control signals are digital in nature and intended to be provided directly by a microprocessor/microcomputer. This means that one PSG can produce the full range of required sounds with no change in external circuitry. Since the frequency response of the PSG ranges from sub-audible at its lowest frequency to post-audible at its highest frequency, there are few sounds which are beyond reproduction with only the simplest electrical connections.

Since most applications of a microprocessor/PSG system would also require interfacing between the outside world and the microprocessor, this facility has been designed into the PSG. The AY-3-8910 has two general purpose 8-bit I/O ports and is supplied in a 40 lead package; the AY-3-8912 has one port and 28 leads.

1012

1.2 Features

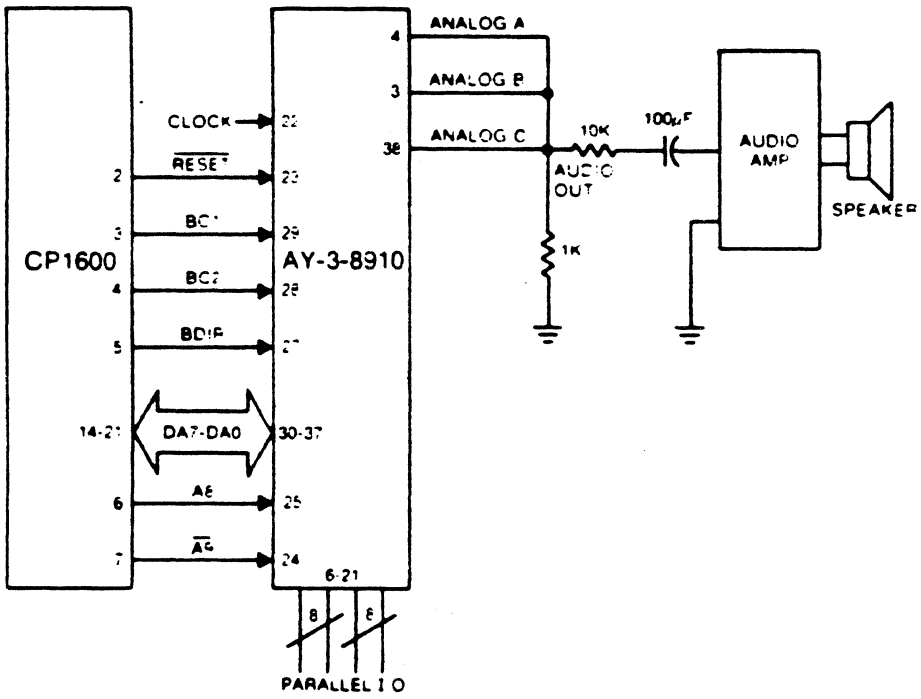
- ☐ Full software control of sound generation.
- ☐ Interfaces to most 8-bit and 16-bit microprocessors.
- ☐ Three independently programmed analog outputs.
- ☐ Two 8-bit general purpose I/O ports (AY-3-8910).
- ☐ One 8-bit general purpose I/O port (AY-3-8912).
- ☐ Single +5 Volt Supply.

1.3 Scope

This Data Manual is intended to introduce the techniques needed to cause the AY-3-8910/8912 Programmable Sound Generator to perform in its intended fashion. All of the programs, programming, and hardware designs have been tested to ensure that the methods are practical rather than purely theoretical.

Although the techniques described will produce powerful results, the range of sounds to be synthesized is so vast and the PSG capabilities so varied that this guide should be viewed merely as an introduction to the applications possibilities of the PSG.

Fig. 1 TYPICAL SYSTEM DIAGRAM



1013

2 ARCHITECTURE

The AY-3-8910/8912 is a register oriented Programmable Sound Generator (PSG). Communication between the processor and the PSG is based on the concept of memory-mapped I/O. Control commands are issued to the PSG by writing to 16 memory-mapped registers. Each of the 16 registers within the PSG is also readable so that the microprocessor can determine, as necessary, present states or stored data values.

All functions of the PSG are controlled through its 16 registers which once programmed, generate and sustain the sounds, thus freeing the system processor for other tasks

2.1 Basic Functional Blocks

An internal block diagram of the PSG showing the various functional blocks and data flow is shown in Fig. 2.

2.1.1 REGISTER ARRAY

The principal element of the PSG is the array of 16 read/write control registers. These 16 registers look to the CPU as a block of memory and as such occupy a 16 word block out of 1,024 possible addresses The 10 address bits (8 bits on the common data/address bus, and 2 separate address bits A8 and $\overline{A9}$) are decoded as follows:

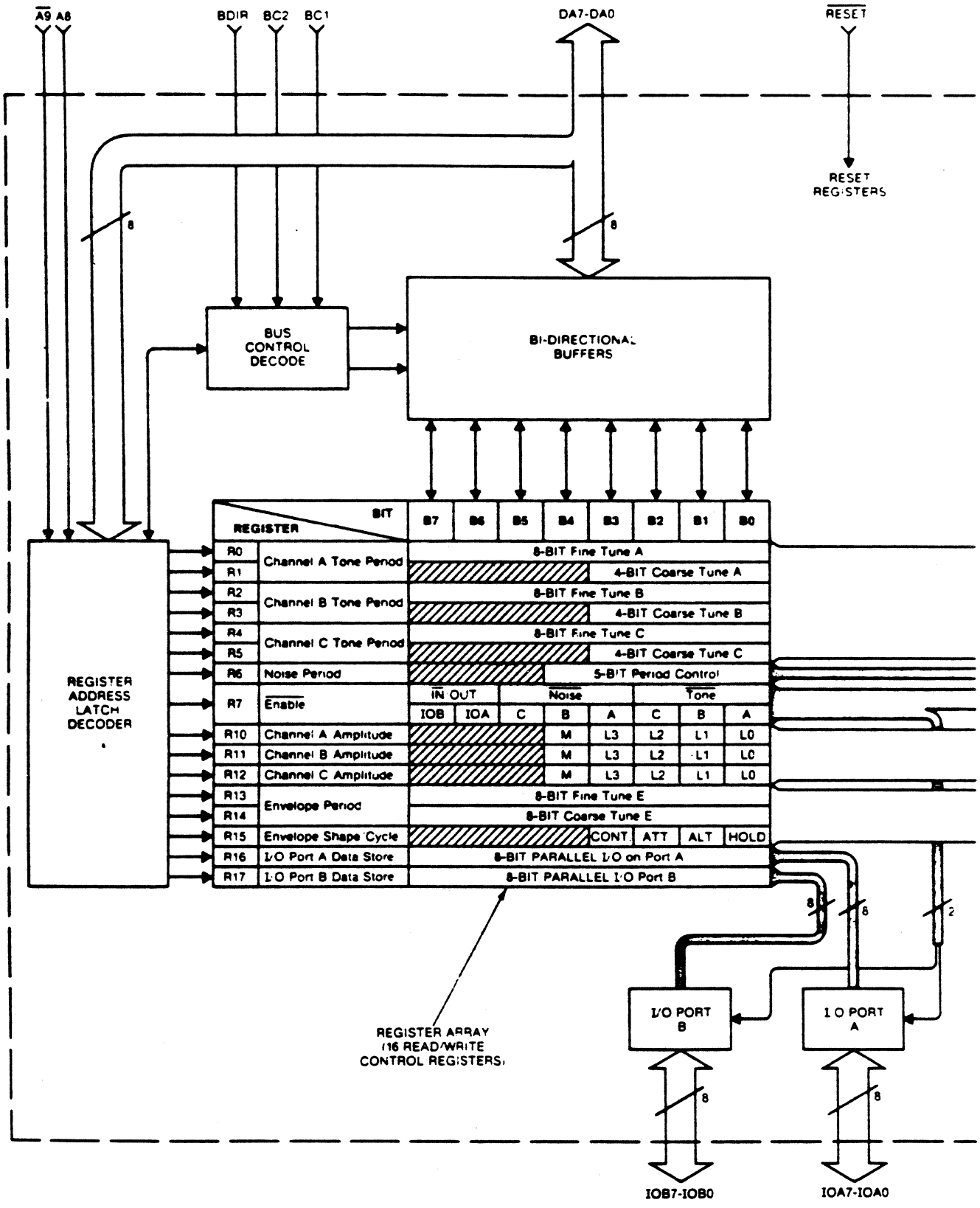
*$\overline{A9}$	A8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
0	1	0	0	0	0	0	0	0	0
THRU									
0	1	0	0	0	0	1	1	1	1
HIGH ORDER (Chip Select)						LOW ORDER (Register #)			

* $\overline{A9}$ is not provided on the AY-3-8912

The four low order address bits select one of the 16 registers (R0--$R17_8$). The six high order address bits function as "chip selects" to control the tri-state bidirectional buffers (when the high order address bits are "incorrect", the bidirectional buffers are forced to a high impedance state). High order address bits $\overline{A9}$ A8 are fixed in the PSG design to recognize a 01 code; high order address bits DA7--DA4 may be mask-programmed to any 4-bit code by a special order factory mask modification. Unless otherwise specified, address bits DA7--DA4 are programmed to recognize only a 0000 code. A valid high order address latches the register address (the low order 4 bits) in the Register Address Latch/Decoder block. A latched address will remain valid until the receipt of a new address, enabling multiple reads and writes of the same register contents without the need for redundant re-addressing.

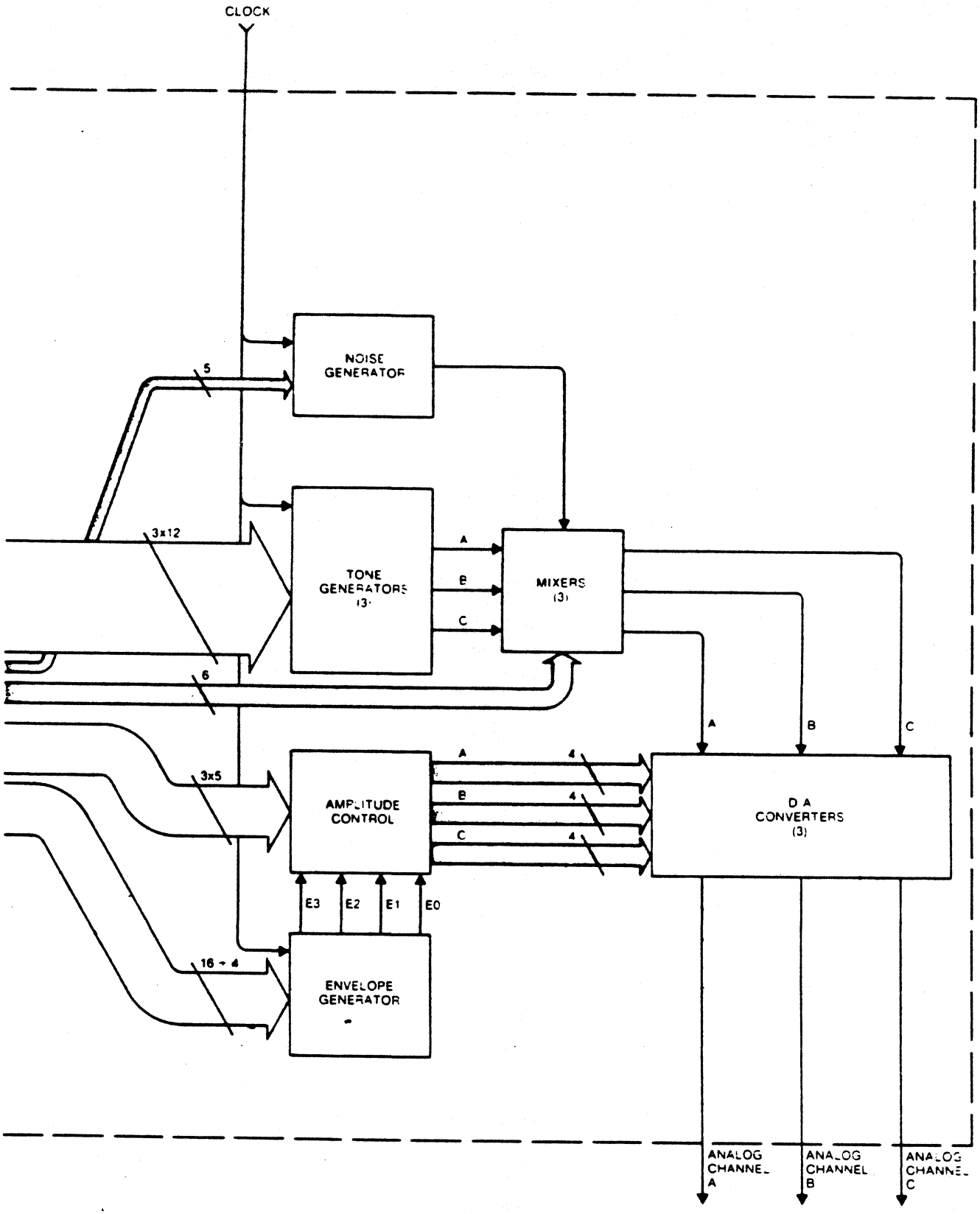
1014

Fig. 2 PSG BLOCK DIAGRAM



REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period				5-BIT Period Control				
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A Amplitude				M	L3	L2	L1	L0
R11	Channel B Amplitude				M	L3	L2	L1	L0
R12	Channel C Amplitude				M	L3	L2	L1	L0
R13	Envelope Period	8-BIT Fine Tune E							
R14		8-BIT Coarse Tune E							
R15	Envelope Shape Cycle					CONT	ATT	ALT	HOLD
R16	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R17	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

1015


CLOCK
NOISE GENERATOR
5
TONE GENERATORS (3)
3x12
A
B
C
MIXERS (3)
6
A
B
C
3x5
AMPLITUDE CONTROL
A
B
C
4
4
4
D A CONVERTERS (3)
E3
E2
E1
E0
16 ÷ 4
ENVELOPE GENERATOR
ANALOG CHANNEL A
ANALOG CHANNEL B
ANALOG CHANNEL C

1016

2.1 Basic Functional Blocks (cont.)

Conditioning of the Register Address Latch/Decoder and the Bidirectional Buffers to recognize the bus function required (inactive, latch address, write data, or read data) is accomplished by the Bus Control Decode block.

The function of each of the 16 PSG registers and the data flow of each register's contents are shown in context in Fig. 2 and explained in detail in Section 3, "Operation". For reference purposes, the Register Array details are reproduced in Fig. 3.

2.1.2 SOUND GENERATING BLOCKS

The basic blocks in the PSG which produce the programmed sounds include:

Tone Generators	produce the basic square wave tone frequencies for each channel (A,B,C)
Noise Generator	produces a frequency modulated pseudo random pulse width square wave output.
Mixers	combine the outputs of the Tone Generators and the Noise Generator. One for each channel (A,B,C).
Amplitude Control	provides the D/A Converters with either a fixed or variable amplitude pattern. The fixed amplitude is under direct CPU control; the variable amplitude is accomplished by using the output of the Envelope Generator.
Envelope Generator	produces an envelope pattern which can be used to amplitude modulate the output of each Mixer.
D/A Converters	the three D/A Converters each produce up to a 16 level output signal as determined by the Amplitude Control.

2.1.3 I/O PORTS

Two additional blocks are shown in the PSG Block Diagram which have nothing directly to do with the production of sound—these are the two I/O Ports (A and B). Since virtually all uses of microprocessor-based sound would require interfacing between the outside world and the processor, this facility has been included in the PSG. Data to/from the CPU bus may be read/written to either of two 8-bit I/O Ports without affecting any other function of the PSG. The I/O Ports are TTL-compatible and are provided with internal pull-ups on each pin. Both Ports are available on the AY-3-8910; only I/O Port A is available on the AY-3-8912.

1017

Fig. 3 PSG REGISTER ARRAY

REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period				5-BIT Period Control				
R7	Enable	IN OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A Amplitude				M	L3	L2	L1	L0
R11	Channel B Amplitude				M	L3	L2	L1	L0
R12	Channel C Amplitude				M	L3	L2	L1	L0
R13	Envelope Period	8-BIT Fine Tune E							
R14		8-BIT Coarse Tune E							
R15	Envelope Shape Cycle					CONT	ATT	ALT	HOLD
R16	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R17	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

1018

2.2 Pin Assignments

The AY-3-8910 is supplied in a 40 lead dual in-line package with the pin assignments as shown in Fig. 4. The AY-3-8912 is supplied in a 28 lead dual in-line package with the pin assignments as shown in Fig. 5

Fig. 4 AY-3-8910 PIN ASSIGNMENTS

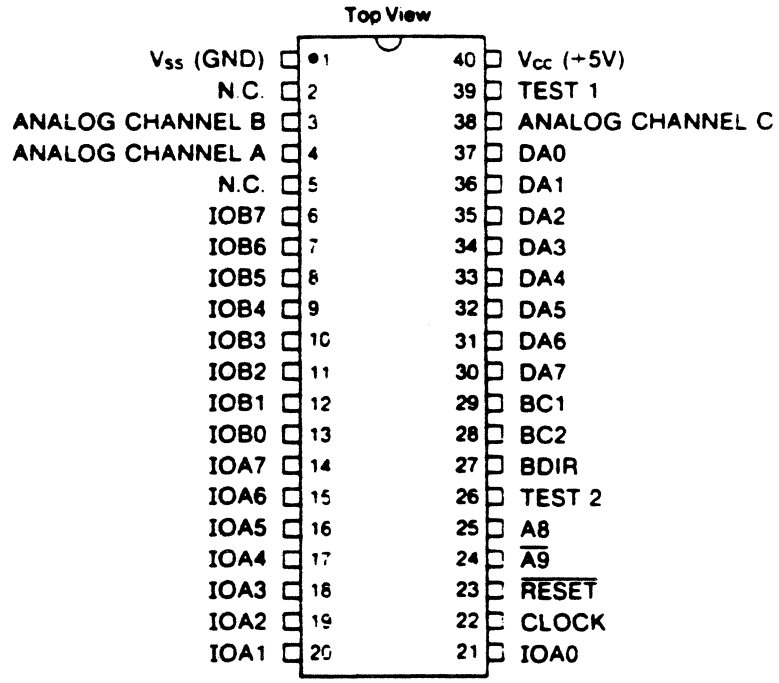
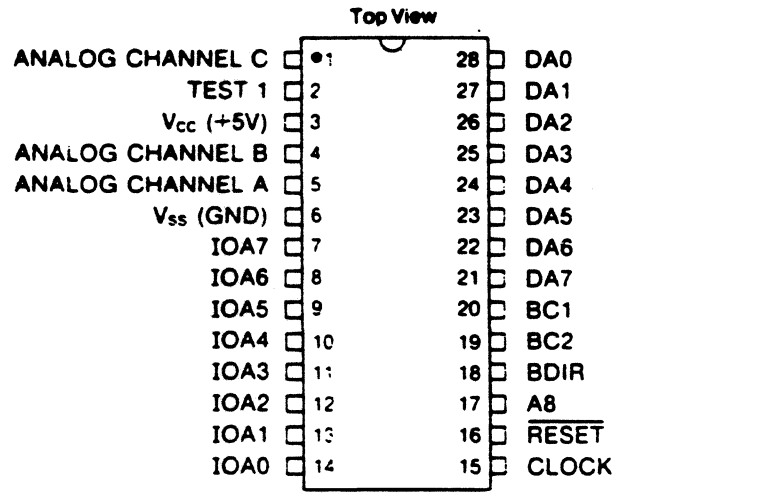


Fig. 5 AY-3-8912 PIN ASSIGNMENTS



1019

2.3 Pin Functions

DA7--DA0 (input/output/high impedance): pins 30--37 (AY-3-8910)
Data/Address 7--0: pins 21--28 (AY-3-8912)

These 8 lines comprise the 8-bit bidirectional bus used by the microprocessor to send both data and addresses to the PSG and to receive data from the PSG. In the data mode, DA7--DA0 correspond to Register Array bits B7--B0. In the address mode, DA3--DA0 select the register # ($0--17_8$) and DA7--DA4 in conjunction with address inputs $\overline{A9}$ and A8 form the high order address (chip select).

A8 (input): pin 25 (AY-3-8910)
pin 17 (AY-3-8912)

$\overline{\textbf{A9}}$ (input): pin 24 (AY-3-8910)
(not provided on AY-3-8912)

$\overline{\text{Address 9}}$, Address 8

These "extra" address bits are made available to enable the positioning of the PSG (assigning a 16 word memory space) in a total 1,024 word memory area rather than in a 256 word memory area as defined by address bits DA7--DA0 alone. If the memory size does not require the use of these extra address lines they may be left unconnected as each is provided with either an on-chip pull down ($\overline{A9}$) or pull-up (A8) resistor. In "noisy" environments, however, it is recommended that $\overline{A9}$ and A8 be tied to an external ground and +5V, respectively, if they are not to be used.

$\overline{\textbf{RESET}}$ (input): pin 23 (AY-3-8910)
pin 16 (AY-3-8912)

For initialization/power-on purposes, applying a logic "0" (ground) to the $\overline{\text{Reset}}$ pin will reset all registers to "0". The $\overline{\text{Reset}}$ pin is provided with an on-chip pull-up resistor.

CLOCK (input): pin 22 (AY-3-8910)
pin 15 (AY-3-8912)

This TTL-compatible input supplies the timing reference for the Tone, Noise and Envelope Generators.

BDIR, BC2, BC1 (inputs): pins 27,28,29 (AY-3-8910)
pins 18,19,20 (AY-3-8912)

Bus DIRection, Bus Control 2,1

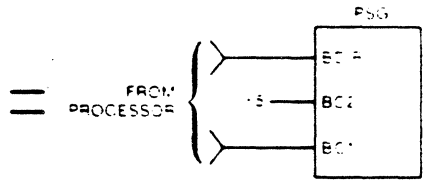
These bus control signals are generated directly by GI's CP1600 series of microprocessors to control all external and internal bus operations in the PSG. When using a processor other than the CP1600, these signals can be provided either by comparable bus signals or by simulating the signals on I/O lines of the processor. The PSG decodes these signals as illustrated in the following:

2.3 Pin Functions (cont.)

BDIR	BC2	BC1	CP1600 FUNCTION	PSG FUNCTION
0	0	0	NACT	INACTIVE See 010 (IAB) below.
0	0	1	ADAR	LATCH ADDRESS See 111 (INTAK) below
0	1	0	IAB	INACTIVE The PSG/CPU bus is inactive DA7--DA0 are in a high impedance state
0	1	1	DTB	READ FROM PSG. This signal causes the contents of the register which is currently addressed to appear on the PSG/CPU bus. DA7--DA0 are in the output mode.
1	0	0	BAR	LATCH ADDRESS. See 111 (INTAK) below.
1	0	1	DW	INACTIVE. See 010 (IAB) above.
1	1	0	DWS	WRITE TO PSG. This signal indicates that the bus contains register data which should be latched into the currently addressed register. DA7--DA0 are in the input mode
1	1	1	INTAK	LATCH ADDRESS. This signal indicates that the bus contains a register address which should be latched in the PSG. DA7--DA0 are in the input mode

While interfacing to a processor other than the CP1600 would simply require simulating the above decoding, the redundancies in the PSG functions vs. bus control signals can be used to advantage in that only four of the eight possible decoded bus functions are required by the PSG. This could simplify the programming of the bus control signals to the following, which would only require that the processor generate two bus control signals (BDIR and BC1, with BC2 tied to +5V):

BDIR	BC2	BC1	PSG FUNCTION
0	1	0	INACTIVE.
0	1	1	READ FROM PSG.
1	1	0	WRITE TO PSG
1	1	1	LATCH ADDRESS.



ANALOG CHANNEL A, B, C (outputs): pins 4, 3, 38 (AY-3-8910)
pins 5, 4, 1 (AY-3-8912)

Each of these signals is the output of its corresponding D/A Converter, and provides an up to 1V peak-peak signal representing the complex sound waveshape generated by the PSG.

IOA7--IOA0 (input/output): pins 14--21 (AY-3-8910)
pins 7--14 (AY-3-8912)

IOB7--IOB0 (input/output): pins 6--13 (AY-3-8910)
(not provided on AY-3-8912)

Input/Output A7--A0, B7--B0

Each of these two parallel input/output ports provides 8 bits of parallel data to/from the PSG/CPU bus from/to any external devices connected to the IOA or IOB pins. Each pin is provided with an on-chip pull-up resistor, so that when in the "input" mode, all pins will read normally high. Therefore, the recommended method for scanning external switches, for example, would be to ground the input bit.

1021

TEST 1: pin 39 (AY-3-8910)
pin 2 (AY-3-8912)

TEST 2: pin 26 (AY-3-8910)
(not connected on AY-3-8912)

These pins are for GI test purposes only and should be left open—do not use as tie-points.

V_{CC}: pin 40 (AY-3-8910)
pin 3 (AY-3-8912)

Nominal −5Volt power supply to the PSG.

V_{SS}: pin 1 (AY-3-8910)
pin 6 (AY-3-8912)

Ground reference for the PSG.

2.4 Bus Timing

Since the PSG functions are controlled by commands from the system processor, the common data/address bus (DA7--DA0) requires definition as to its function at any particluar time. This is accomplished by the processor issuing bus control signals, previously described, defining the state of the bus; the PSG then decodes these signals to perform the requested task.

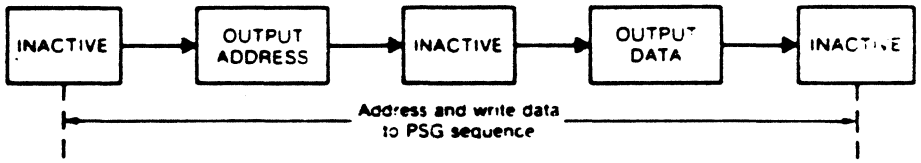
The conditioning of these bus control signals by the processor is the same as if the processor were interacting with RAM: (1) the processor outputs a memory address; and (2) the processor either outputs or inputs data to/from the memory. The "memory" in this case is the PSG's array of 16 read/write control registers.

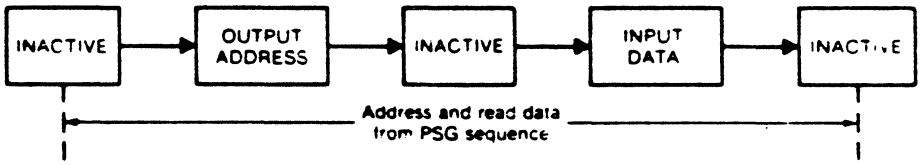
The timing relationships in issuing the bus control signals relative to the data or address signals on the bus are reviewed in general in the following section, and in detail in Section 7, Electrical Specifications.

1022

2.5 State Timing

While the state flow for many microprocessors can be somewhat involved for certain operations, the sequence of events necessary to control the PSG is simple and straightforward. Each of the three major state sequences (Latch Address, Write to PSG, and Read from PSG) consists of several operations (indicated below by rectangular blocks), defined by the pattern of bus control signals (BDIR, BC2, BC1).

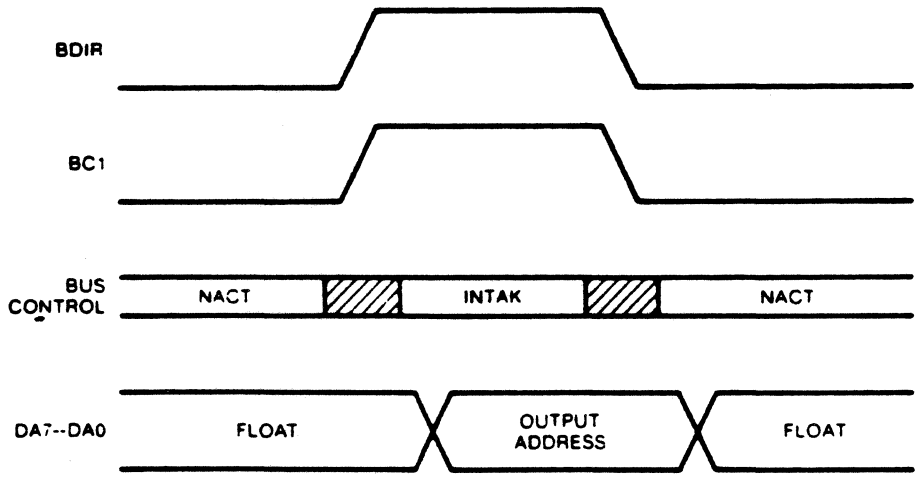




The functional operation and relative timing of the PSG control sequences are described in the following paragraphs (in all examples, BC2 has been assumed to be tied to logic "1", +5V).

2.5.1 ADDRESS PSG REGISTER SEQUENCE

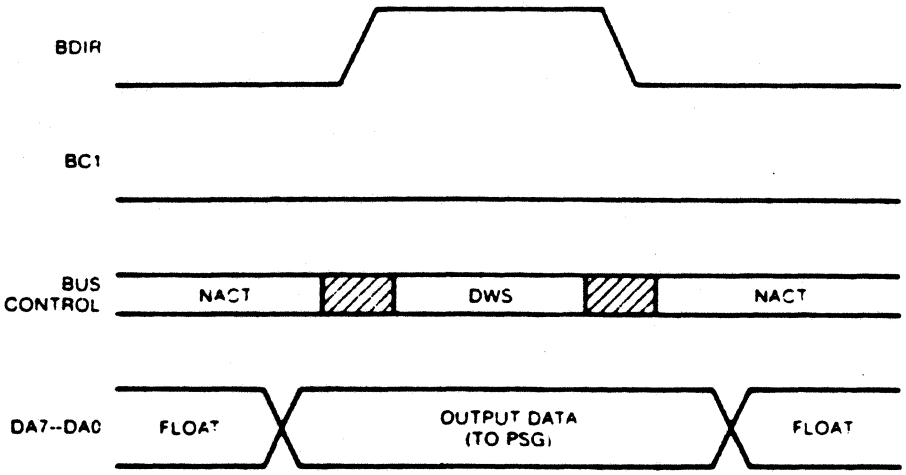
The "Latch Address" sequence is normally an integral part of the write or read sequences, but for simplicity is illustrated here as an individual sequence. Depending on the processor used the program sequence will normally require four principal microstates: (1) send NACT (inactive); (2) send INTAK (latch address); (3) put address on bus; (4) send NACT (inactive). [Note: within the timing constraints detailed in Section 7, steps (2) and (3) may be interchanged.]



1023

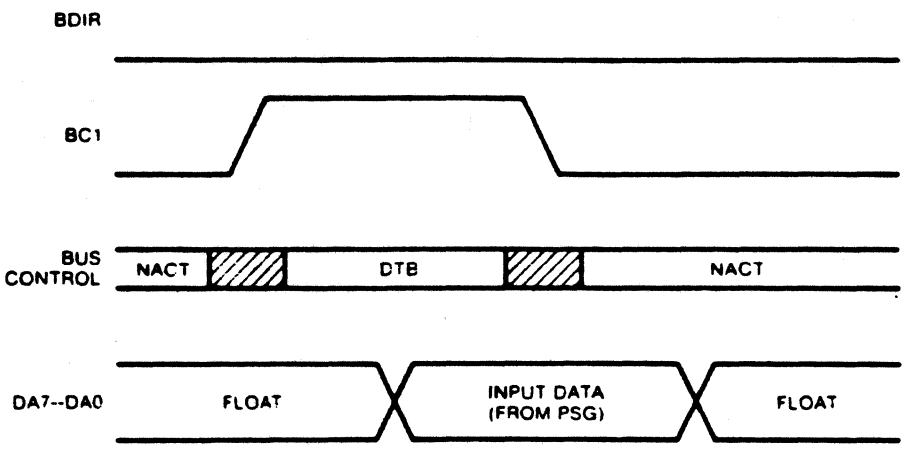
2.5.2 WRITE DATA TO PSG SEQUENCE

The "Write to PSG" sequence, which would normally follow immediately after an address sequence, requires four principal microstates. (1) send NACT (inactive); (2) put data on bus; (3) send DWS (write to PSG); (4) send NACT (inactive).



2.5.3 READ DATA FROM PSG SEQUENCE

As with the "Write to PSG" sequence, the "Read from PSG" sequence would also normally follow immediately after an address sequence. The four principal microstates of the read sequence are: (1) send NACT (inactive); (2) send DTB (read from PSG); (3) read data on bus; (4) send NACT (inactive).



2.5.4 WRITE TO/READ FROM I/O PORT SEQUENCE

Since the two I/O Ports (A and B) each have an 8-bit register assigned as a data store, writing to or reading from either port is identical to writing or reading to any other register. Hence, the state sequences are exactly the same as described in the preceding paragraphs.

1024

3 OPERATION

Since all functions of the PSG are controlled by the host processor via a series of register loads, a detailed description of the PSG operation can best be accomplished by relating each PSG function to the control of its corresponding register. The function of creating or programming a specific sound or sound effect logically follows the control sequence listed:

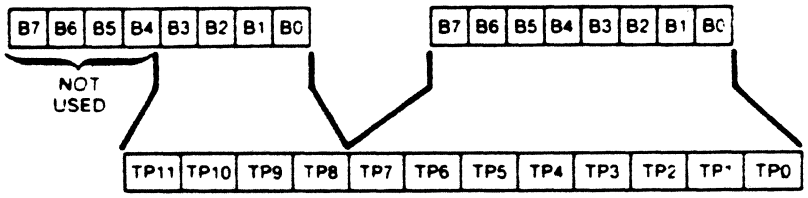
Section	Operation	Registers	Function
3.1	Tone Generator Control	R0--R5	Program tone periods
3.2	Noise Generator Control	R6	Program noise period
3.3	Mixer Control	R7	Enable tone and/or noise on selected channels
3.4	Amplitude Control	R10--R12	Select "fixed" or "envelope-variable" amplitudes
3.5	Envelope Generator Control	R13--R15	Program envelope period and select envelope pattern

3.1 Tone Generator Control

(Registers R0, R1, R2, R3, R4, R5)

The frequency of each square wave generated by the three Tone Generators (one each for Channels A, B, and C) is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 12-bit Tone Period value. Each 12-bit value is obtained in the PSG by combining the contents of the relative Coarse and Fine Tune registers, as illustrated in the following:

Coarse Tune Register	Channel	Fine Tune Register
R1	A	R0
R3	B	R2
R5	C	R4



12-bit Tone Period (TP) to Tone Generator

Note that the 12-bit value programmed in the combined Coarse and Fine Tune registers is a period value—the higher the value in the registers, the lower the resultant tone frequency.

Note also that due to the design technique used in the Tone Period count-down, the lowest period value is 000000000001 (divide by 1) and the highest period value is 111111111111 (divide by 4.095_{10}).

1025

The equations describing the relationship between the desired output tone frequency and the input clock frequency and Tone Period value are:

$$(a)\ f_T = \frac{f_{CLOCK}}{16TP_{10}} \qquad (b)\ TP_{10} = 256CT_{10} + FT_{10}$$

Where: f_T = desired tone frequency
f_{CLOCK} = input clock frequency
TP_{10} = decimal equivalent of the Tone Period bits TP11--TP0.
CT_{10} = decimal equivalent of the Coarse Tune register bits B3--B0 (TP11--TP8)
FT_{10} = decimal equivalent of the Fine Tune register bits B7--B0 (TP7--TP0)

From the above equations it can be seen that the tone frequency can range from a low of $\frac{f_{CLOCK}}{65,520}$ (wherein: $TP_{10}=4,095_{10}$) to a high of $\frac{f_{CLOCK}}{16}$ (wherein: $TP_{10}=1$). Using a 2 MHz input clock, for example, would produce a range of tone frequencies from 30.5 Hz to 125 kHz.

To calculate the values for the contents of the Tone Period Coarse and Fine Tune registers, given the input clock and the desired output tone frequencies, we simply rearrange the above equations, yielding.

$$(a)\ TP_{10} = \frac{f_{CLOCK}}{16f_T} \qquad (b)\ CT_{10} - \frac{FT_{10}}{256} = \frac{TP_{10}}{256}$$

Example 1: $f_T = 1kHz$
$f_{CLOCK} = 2MHz$

$$TP_{10} = \frac{2x10^6}{16(1x10^3)} = 125$$

Substituting this result into equation (b)

$$CT_{10} + \frac{FT_{10}}{256} = \frac{125}{256}$$

$\therefore CT_{10} = 0 \quad = 0000$ (B3--B0)
$FT_{10} = 125_{10} = 01111101$ (B7--B0)

Example 2: $f_T = 100Hz$
$f_{CLOCK} = 2MHz$

$$TP_{10} = \frac{2x10^6}{16(1x10^2)} = 1250$$

Substituting this result into equation (b).

$$CT_{10} + \frac{FT_{10}}{256} = \frac{1250}{256} = 4 + \frac{226}{256}$$

$\therefore CT_{10} = 4_{10} = 0100$ (B3--B0)
$FT_{10} = 226_{10} = 11100010$ (B7--B0)

1036

3.2 Noise Generator Control

(Register R6)

The frequency of the noise source is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 5-bit Noise Period value. This 5-bit value consists of the lower 5 bits (B4--B0) of register R6 as illustrated in the following:

Noise Period
Register R6

B7	B6	B5	B4	B3	B2	B1	B0
NOT USED			5-bit Noise Period (NP) to Noise Generator				

Note that the 5-bit value in R11 is a period value—the higher the value in the register, the lower the resultant noise frequency. Note also that, as with the Tone Period, the lowest period value is 00001 (divide by 1); the highest period value is 11111 (divide by 31_{10}).

The noise frequency equation is:

$$f_N = \frac{f_{CLOCK}}{16\ NP_{10}}$$

Where: f_N = desired noise frequency
f_{CLOCK} = input clock frequency
NP_{10} = decimal equivalent of the Noise Period register bits B4--B0.

From the above equation it can be seen that the noise frequency can range from a low of $\frac{f_{CLOCK}}{496}$ (wherein: $NP_{10} = 31_{10}$) to a high of $\frac{f_{CLOCK}}{16}$ (wherein: $NP_{10} = 1$). Using a 2 MHz input clock, for example, would produce a range of noise frequencies from 4 kHz to 125 kHz.

To calculate the value for the contents of the Noise Period register, given the input clock and the desired output noise frequencies, we simply rearrange the above equation, yielding:

$$NP_{10} = \frac{f_{CLOCK}}{16\ f_N}$$

1027

3.3 Mixer Control-I/O Enable

(Register R7)

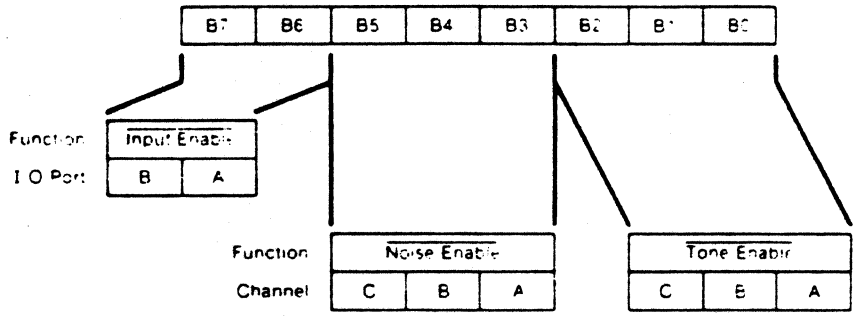
Register 7 is a multi-function Enable register which controls the three Noise/Tone Mixers and the two general purpose I/O Ports.

The Mixers, as previously described, combine the noise and tone frequencies for each of the three channels. The determination of combining neither/either/both noise and tone frequencies on each channel is made by the state of bits B5--B0 of R7.

The direction (input or output) of the two general purpose I/O Ports (IOA and IOB) is determined by the state of bits B7 and B6 of R7.

These functions are illustrated in the following:

Mixer Control-I/O Enable
Register R7



Noise Enable Truth Table:

R7 Bits B5	B4	B3	Noise Enabled on Channel C	B	A
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

Tone Enable Truth Table:

R7 Bits B2	B1	B0	Tone Enabled on Channel C	B	A
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

I/O Port Truth Table:

R7 Bits B7	B6	I/O Port Status IOB	IOA
0	0	Input	Input
0	1	Input	Output
1	0	Output	Input
1	1	Output	Output

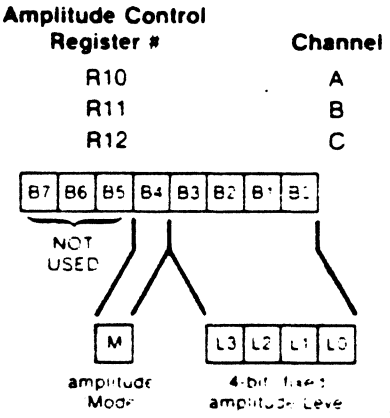
NOTE: Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register. R10, R11, or R12 (see Section 3.4).

1028

3.4 Amplitude Control

(Registers R10, R11, R12)

The amplitudes of the signals generated by each of the three D/A Converters (one each for Channels A, B, and C) is determined by the contents of the lower 5 bits (B4--B0) of registers R10, R11, and R12 as illustrated in the following:



The amplitude "mode" (bit M) selects either fixed level amplitude (M=0) or variable level amplitude (M=1). It follows then that bits L3--L0, defining the value of a "fixed" level amplitude, are only active when M=0. When fixed level amplitude is selected, it is "fixed" only in the sense that the amplitude level is under the direct control of the system processor (via bits D3--D0). Varying the amplitude when in this "fixed" amplitude mode requires in each instance the direct intervention of the system processor via an address latch/write data sequence to modify the D3--D0 data.

When M=1 (select "variable" level amplitudes), the amplitude of each channel is determined by the envelope pattern as defined by the Envelope Generator's 4-bit output E3 E2 E1 E0.

The amplitude "mode" (bit M) can also be thought of as an "envelope enable" bit; i.e., when M=0 the envelope is not used, and when M=1 the envelope is enabled. (A full description of the Envelope Generator function follows in Section 3.5).

1029

The full chart describing all combinations of the 5-bit Amplitude Control is as follows:

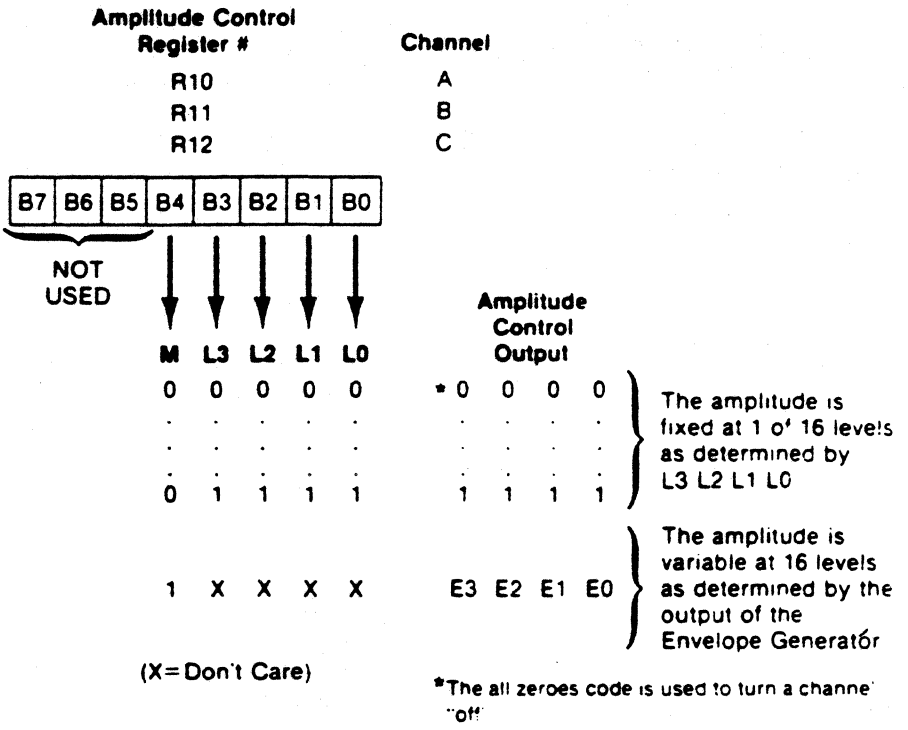
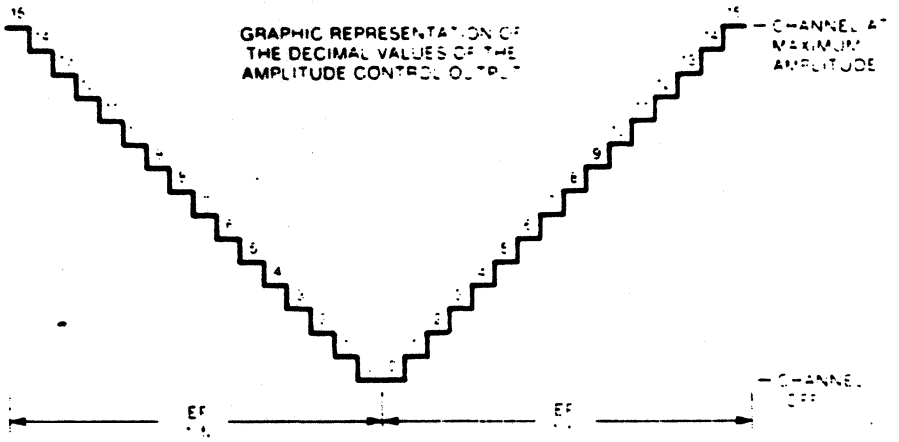


Fig. 6 graphically illustrates a selection of variable level (envelope-controlled) amplitude where the 16 levels directly reflect the output of the Envelope Generator. A fixed level amplitude would correspond to only one of the levels shown, with the level directly determined by the decimal equivalent of bits L3 L2 L1 L0.

Fig. 6 VARIABLE AMPLITUDE CONTROL (M=1)



1030

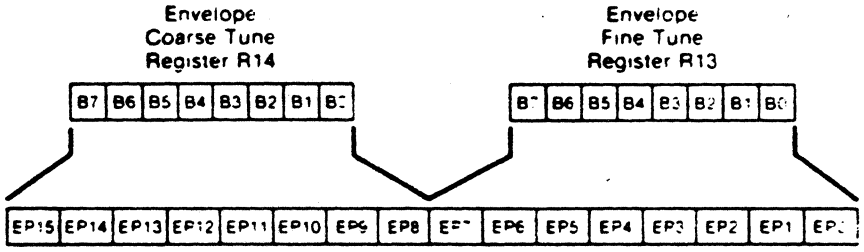
3.5 Envelope Generator Control

(Registers R13, R14, R15)

To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG: first, it is possible to vary the frequency of the envelope using registers R13 and R14; and second, the relative shape and cycle pattern of the envelope can be varied using register R15. The following paragraphs explain the details of the envelope control functions, describing first the envelope period control and then the envelope shape/cycle control.

3.5.1 ENVELOPE PERIOD CONTROL (Registers R13, R14)

The frequency of the envelope is obtained in the PSG by first counting down the input clock by 256, then by further counting down the result by the programmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated in the following:



16-bit Envelope Period (EP)
to Envelope Generator

Note that the 16-bit value programmed in the combined Coarse and Fine Tune registers is a period value—the higher the value in the registers, the lower the resultant envelope frequency.

Note also, that as with the Tone Period, the lowest period value is 0000000000000001 (divide by 1); the highest period value is 1111111111111111 (divide by $65{,}535_{10}$).

The envelope frequency equations are:

$$\text{(a) } f_E = \frac{f_{CLOCK}}{256EP_{10}} \qquad \text{(b) } EP_{10} = 256CT_{10} + FT_{10}$$

Where:
- f_E = desired envelope frequency
- f_{CLOCK} = input clock frequency
- EP_{10} = decimal equivalent of the Envelope Period bits EP15--EP0
- CT_{10} = decimal equivalent of the Coarse Tune register bits B7--B0 (EP15--EP8)
- FT_{10} = decimal equivalent of the Fine Tune register bits B7--B0 (EP7--EP0)

From the above equation it can bee seen that the envelope frequency can range from a low of $\frac{f_{CLOCK}}{16{,}776{,}960}$ (wherein: $EP_{10}=65{,}535_{10}$) to a high of $\frac{f_{CLOCK}}{256}$ (wherein: $EP_{10}=1$). Using a 2 MHz clock, for example, would produce a range of envelope frequencies from 0.12 Hz to 7812.5 Hz.

To calculate the values for the contents of the Envelope Period Coarse and Fine Tune registers, given the input clock and the desired envelope frequencies, we rearrange the above equations, yielding

$$\text{(a) } EP_{10} = \frac{f_{CLOCK}}{256 f_E} \qquad \text{(b) } CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$$

Example: $f_E = 0.5$ Hz
$f_{CLOCK} = 2$ MHz

$$EP_{10} = \frac{2 \times 10^6}{256(0.5)} = 15.625$$

Substituting this result into equation (b)

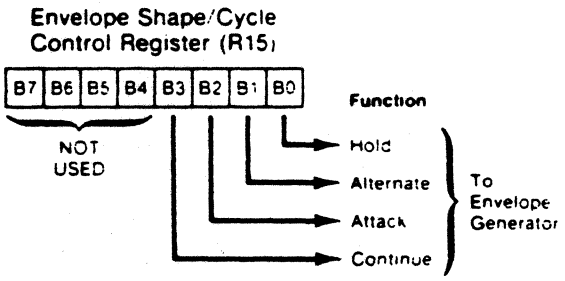
$$CT_{10} + \frac{FT_{10}}{256} = \frac{15.625}{256} = 61 + \frac{9}{256}$$

$CT_{10} = 61_{10} = 00111101$ (B7--B0)
$FT_{10} = 9_{10} = 00001001$ (B7--B0)

3.5.2 ENVELOPE SHAPE/CYCLE CONTROL (Register R15)

The Envelope Generator further counts down the envelope frequency by 16, producing a 16-state per cycle envelope pattern as defined by its 4-bit counter output, E3 E2 E1 E0. The particular shape and cycle pattern of any desired envelope is accomplished by controlling the count pattern (count up/count down) of the 4-bit counter and by defining a single-cycle or repeat-cycle pattern.

This envelope shape/cycle control is contained in the lower 4 bits (B3--B0) of register R15. Each of these 4 bits controls a function in the envelope generator, as illustrated in the following:



The definition of each function is as follows:

Hold — when set to logic "1", limits the envelope to one cycle, holding the last count of the envelope counter (E3--E0=0000 or 1111, depending on whether the envelope counter was in a count-down or count-up mode, respectively).

Alternate — when set to logic "1", the envelope counter reverses count direction (up-down) after each cycle.

NOTE: When both the Hold bit and the Alternate bit are ones, the envelope counter is reset to its initial count before holding

1032

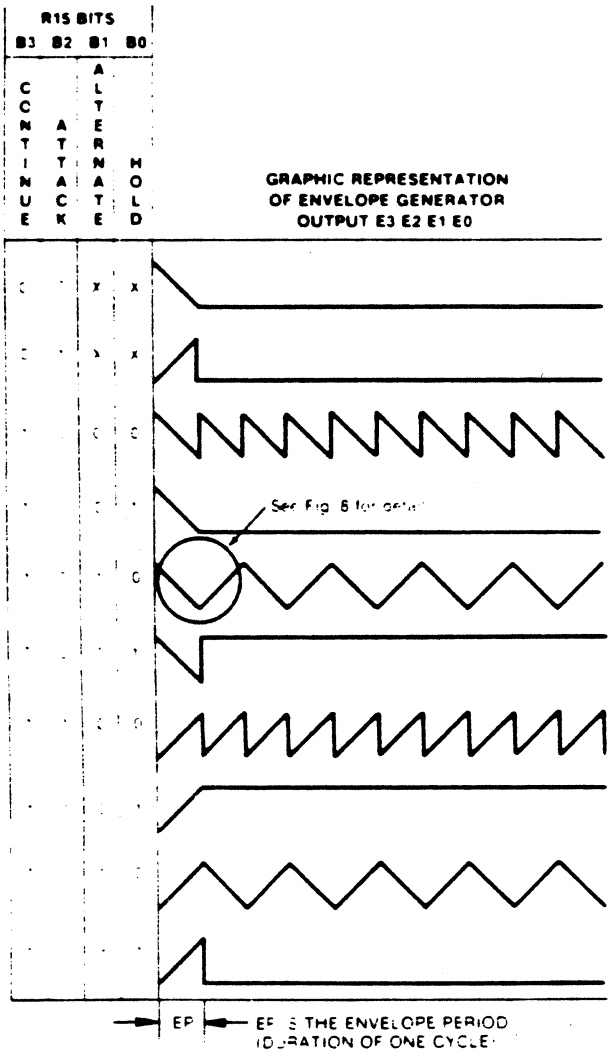
3.5 Envelope Generator Control (cont.)

Attack — when set to logic "1", the envelope counter will count up (attack) from E3 E2 E1 E0=0000 to E3 E2 E1 E0=1111; when set to logic "0", the envelope counter will count down (decay) from 1111 to 0000.

Continue — when set to logic "1", the cycle pattern will be as defined by the Hold bit; when set to logic "0", the envelope generator will reset to 0000 after one cycle and hold at that count.

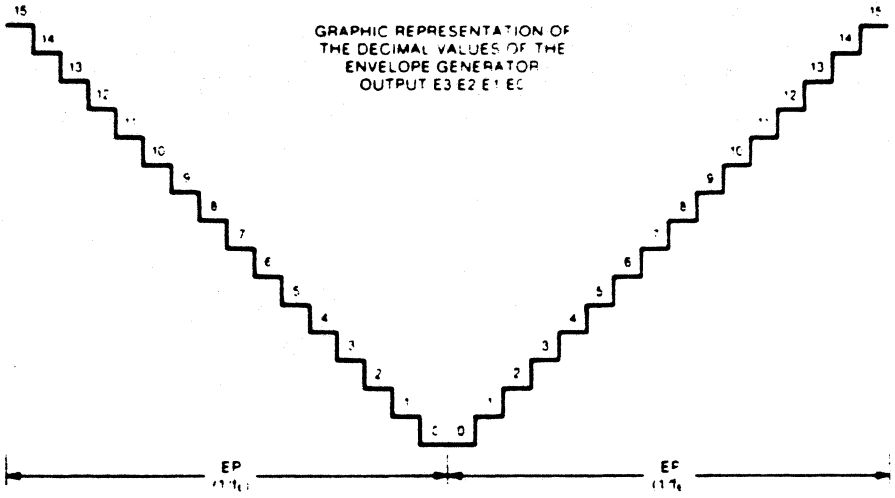
To further describe the above functions could be accomplished by numerous charts of the binary count sequence of E3 E2 E1 E0 for each combination of Hold. Alternate, Attack and Continue. However, since these outputs are used (when selected by the Amplitude Control registers) to amplitude modulate the output of the Mixers, a better understanding of their effect can be accomplished via a graphic representation of their value for each condition selected, as illustrated in Figs. 7 and 8.

Fig. 7 ENVELOPE SHAPE/CYCLE CONTROL



1033

Fig. 8 DETAIL OF TWO CYCLES OF Fig. 7
(ref. waveform "1010" in Fig. 7)



1034

3.6 I/O Port Data Store

(Registers R16, R17)

Registers R16 and R17 function as intermediate data storage registers between the PSG/CPU data bus (DA0--DA7) and the two I/O ports (IOA7--IOA0 and IOB7--IOB0). Both ports are available in the AY-3-8910; only I/O Port A is available in the AY-3-8912. Using registers R16 and R17 for the transfer of I/O data has no effect at all on sound generation.

To output data from the CPU bus to a peripheral device connected to I/O Port A would require only the following steps:

1. Latch address R7 (select $\overline{\text{Enable}}$ register)
2. Write data to PSG (setting B6 of R7 to "1")
3. Latch address R16 (select IOA register)
4. Write data to PSG (data to be output on I/O Port A)

To input data from I/O Port A to the CPU bus would require the following:

1. Latch address R7 (select $\overline{\text{Enable}}$ register)
2. Write data to PSG (setting B6 to R7 to "0")
3. Latch address R16 (select IOA register)
4. Read data from PSG (data from I/O Port A)

Note that once loaded with data in the output mode, the data will remain on the I/O port(s) until changed either by loading different data, by applying a reset (grounding the Reset pin), or by switching to the input mode.

Note also that when in the input mode, the contents of registers R16 and/or R17 will follow the signals applied to the I/O port(s). However, transfer of this data to the CPU bus requires a "read" operation as described above.

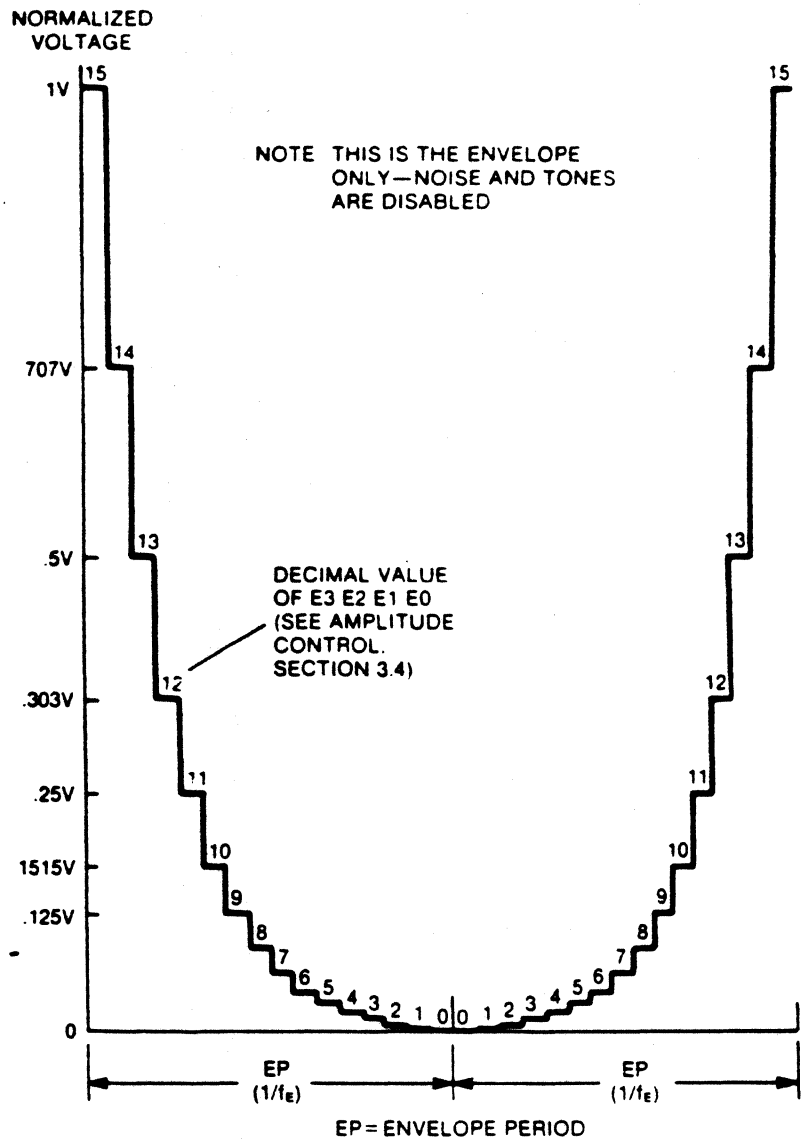
3.7 D/A Converter Operation

Since the primary use of the PSG is to produce sound for the highly imperfect amplitude detection mechanism of the human ear, the D/A conversion is performed in logarithmic steps with a normalized voltage range of from 0 to 1 Volt. The specific amplitude control of each of the three D/A Converters is accomplished by the three sets of 4-bit outputs of the Amplitude Control block, while the Mixer outputs provide the base signal frequency (Noise and/or Tone).

Fig. 9 illustrates the D/A Converter output which would result if noise and tones were disabled and an envelope-controlled variable amplitude were selected.

Figs. 10 through 13 illustrate other typical output waveforms.

Fig. 9 D/A CONVERTER OUTPUT (ref. Fig. 6)



1036

Fig. 10 SINGLE TONE WITH ENVELOPE SHAPE CYCLE PATTERN 1000
(R0=14_E, R1=37_H, R7=76_B, R12=20_B, R15=10_E, all other registers=0)

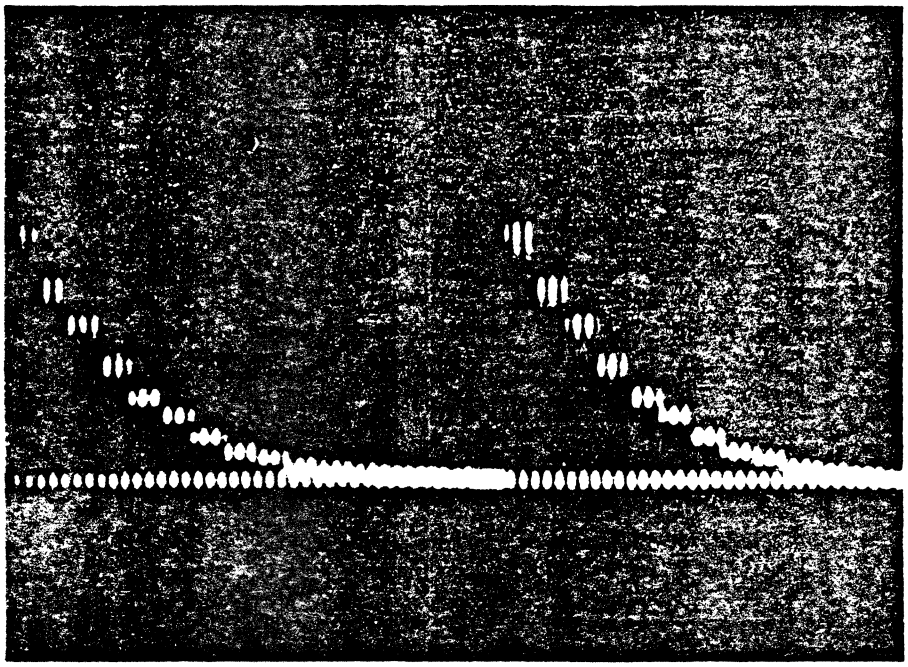
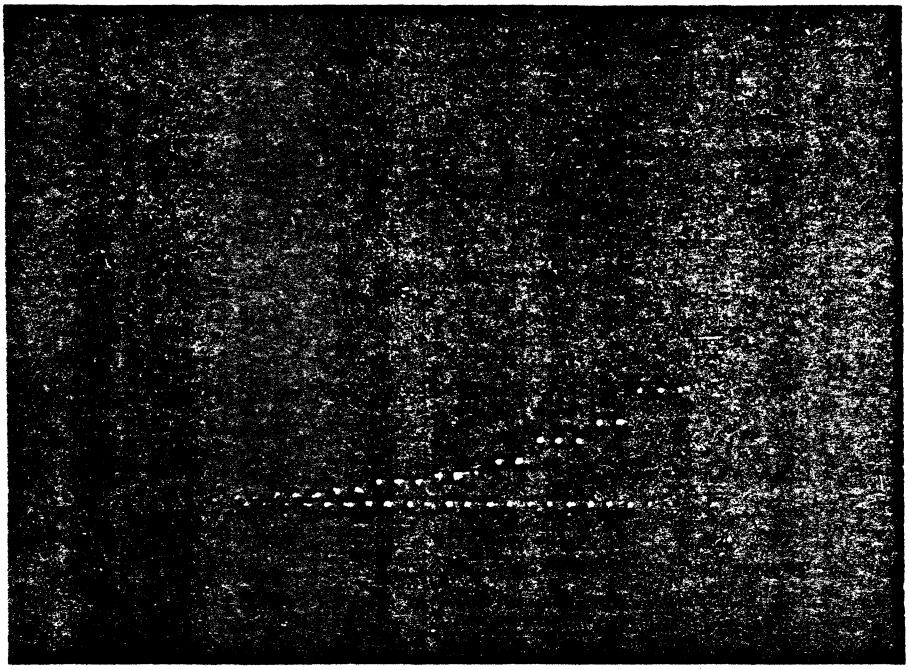


Fig. 11 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1100
(R15=14_5, all other registers same as Fig. 10)



1037

Fig. 12 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1010
(R15=12_8, all other registers same as Fig. 10)

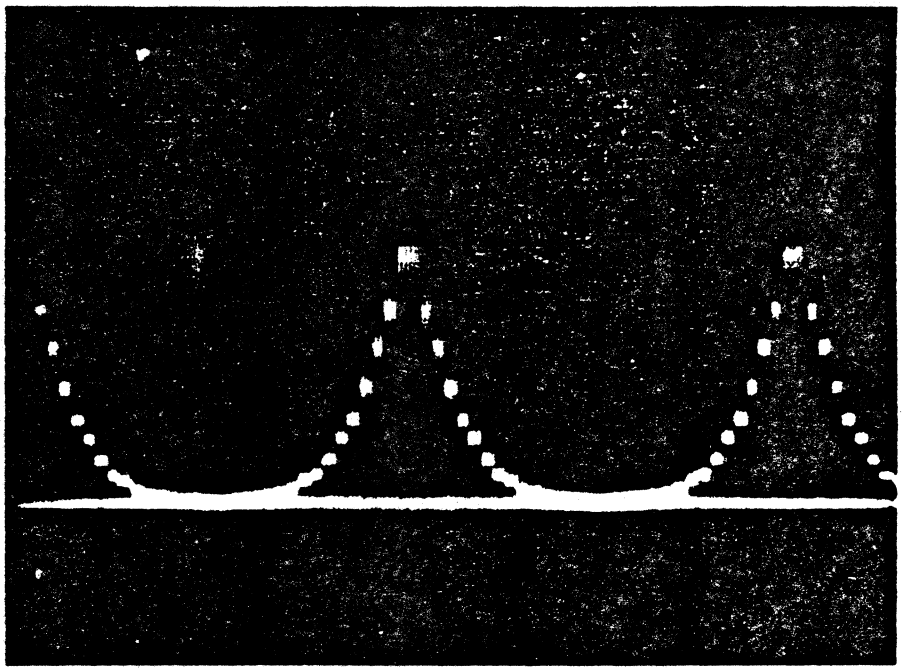
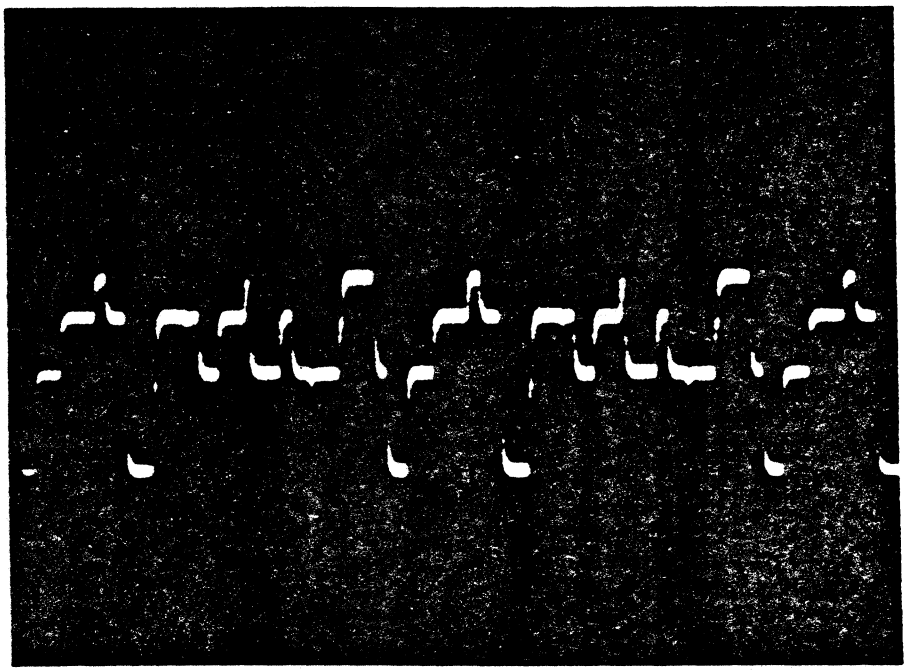


Fig. 13 MIXTURE OF THREE TONES WITH FIXED AMPLITUDES



1038

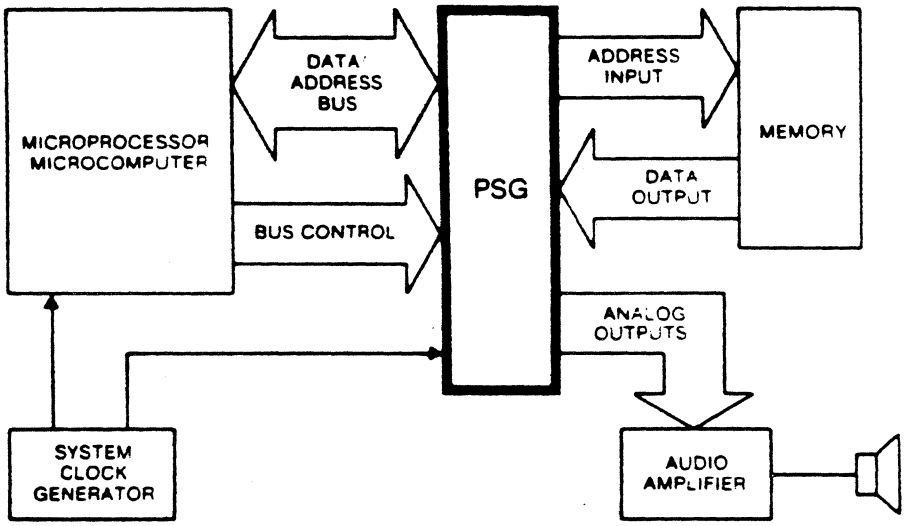
4 INTERFACING

4.1 Introduction

Since the AY-3-8910/8912 PSG must be used with support components, interfacing to the circuit is an obvious requirement. The PSG is designed to be controlled by a microprocessor or microcomputer, and drive directly into analog audio circuitry. It provides the link between the computer and a speaker to provide sounds or sound effects derived from digital inputs.

The following paragraphs provide examples and illustrations showing the ease with which an AY-3-8910/8912 Programmable Sound Generator may be utilized in a microprocessor/microcomputer system.

Fig. 14 SYSTEM BLOCK DIAGRAM

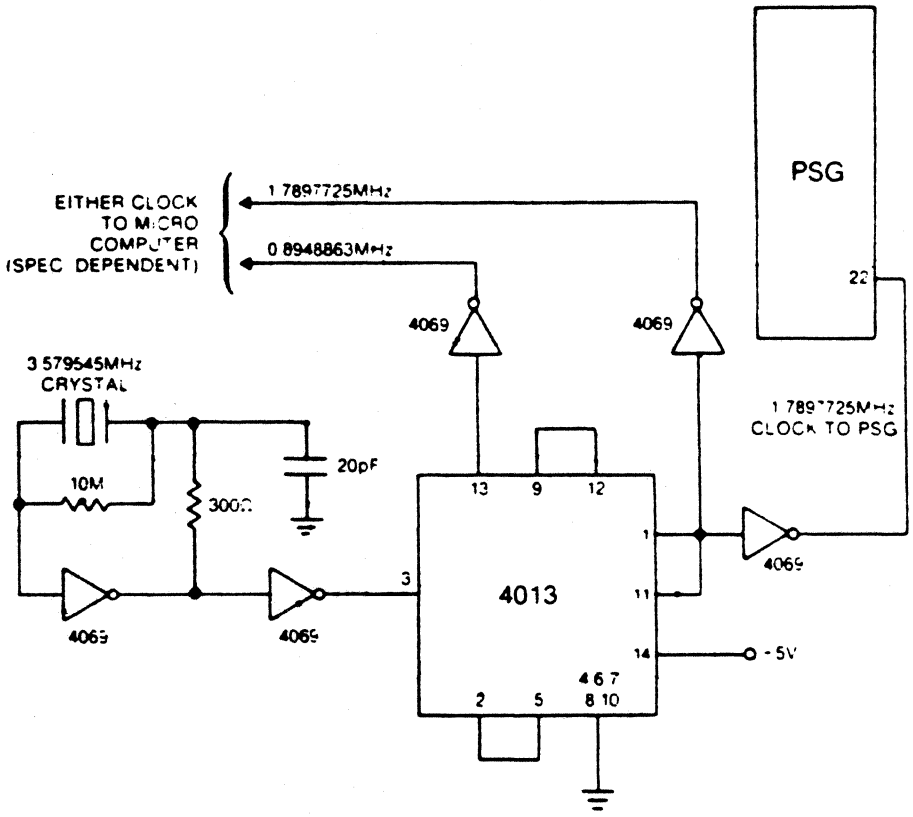


1039

4.2 Clock Generation

An economical solution to providing a system clock is shown in Fig. 15. It consists of a 3.579545MHz standard color burst crystal, a CD4069 CMOS inverter, and a CD4013 to divide the color burst frequency in half. The clock produced for the PSG runs at a 1.7897725MHz rate. Depending on the microcomputer used, its clock should be selected within its specified value.

Fig. 15 CLOCK GENERATION



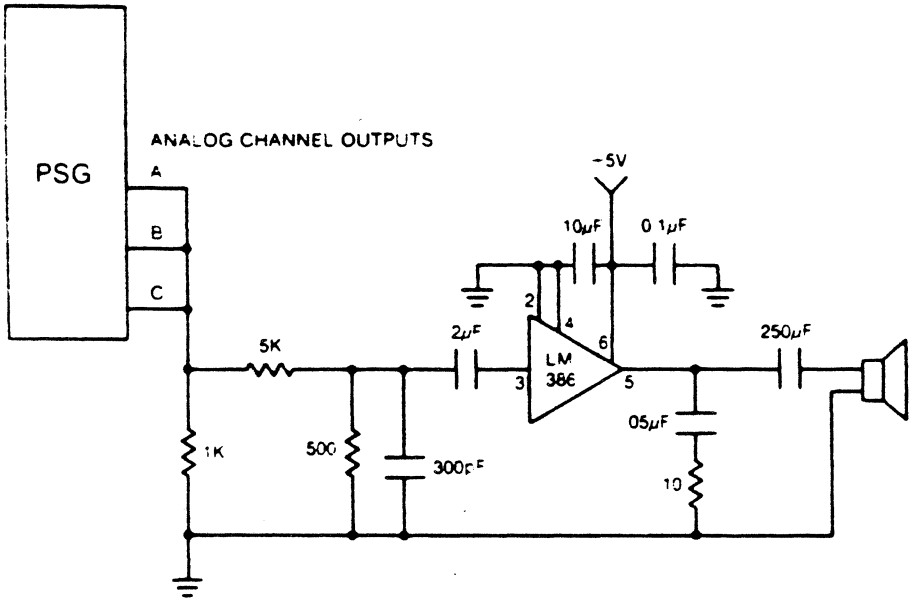
1040

4.3 Audio Output Interface

Fig. 16 illustrates the audio output connections to a commercially available LM386 audio amplifier. It shows channels A, B, and C summed together to enable complex waveforms to be composed and amplified through a single external amplifier. These channels may be individually amplified through separate channels for more exotic sound systems.

Each output channel is individually controlled by separate amplitude registers (R10, R11, R12) and an enable register (R7) in the PSG.

Fig 16 AUDIO OUTPUT INTERFACE



1041

4.4 External Memory Access

The ROM or PROM shown connected to the PSG in Fig. 17 illustrates an option for providing additional data information for processor support. The two I/O registers within the PSG are used in this case to address the memory via I/O Port A (8 Bits) and read data from the memory via I/O Port B (8 Bits).

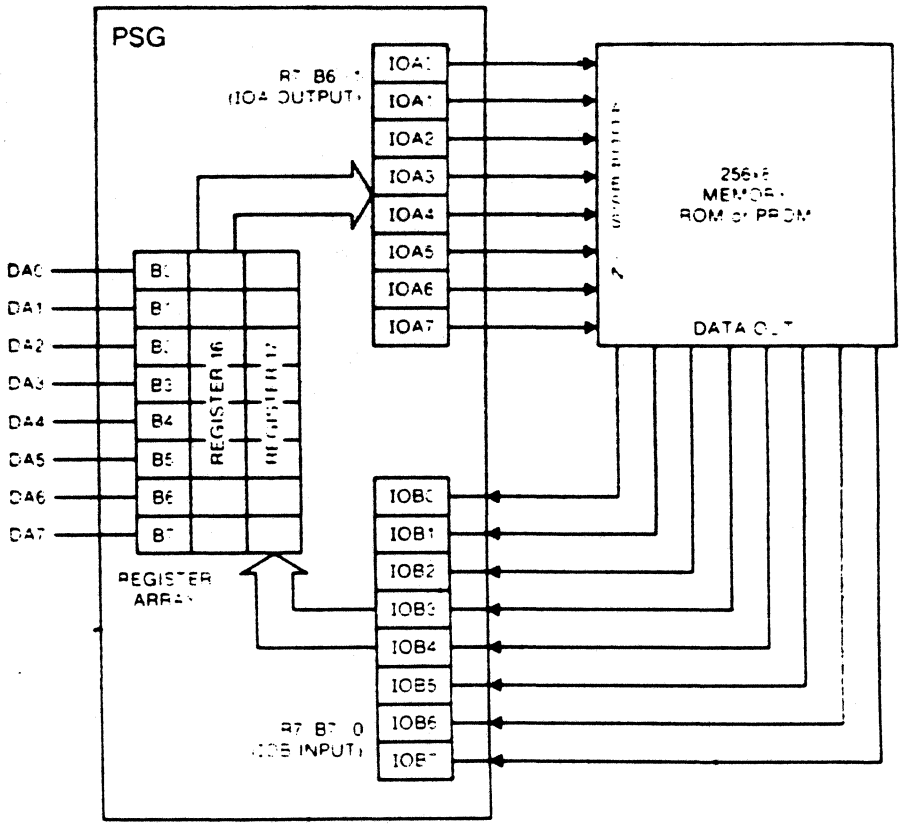
An example of the bus control sequence to address and read an external memory connected to I/O ports A and B would be as follows (Assume Port A addresses and Port B reads).

	Bus Codes			
Bus Control	**BDIR**	**BC2**	**BC1**	**Explanation of Bus Data (DA7--DA0)**
Latch address	1	1	1	00000111 Latch R7 to program I/O Ports
Write to PSG	1	1	0	01000000 Set B7 B6 to 0, 1 respectively
Latch address	1	1	1	00001110 Latch R16 to address memory
Write to PSG	1	1	0	00000001 Address data to memory
Latch address	1	1	1	00001111 Latch R17 to read memory
Read from PSG	0	1	1	XXXXXXXX Memory data contained in R17

NOTE BC2 in the above Bus Codes may be permanently tied to +5V thus requiring only two bus control lines for all control operations (refer to Section 2.3 for a complete explanation).

Also, RAM or EAROM may be used in place of the ROM or PROM shown by altering the program to use PORT B as an I/O. Port B then will be able to write data as an output and read data as an input.

Fig. 17 EXTERNAL MEMORY ACCESS



1042

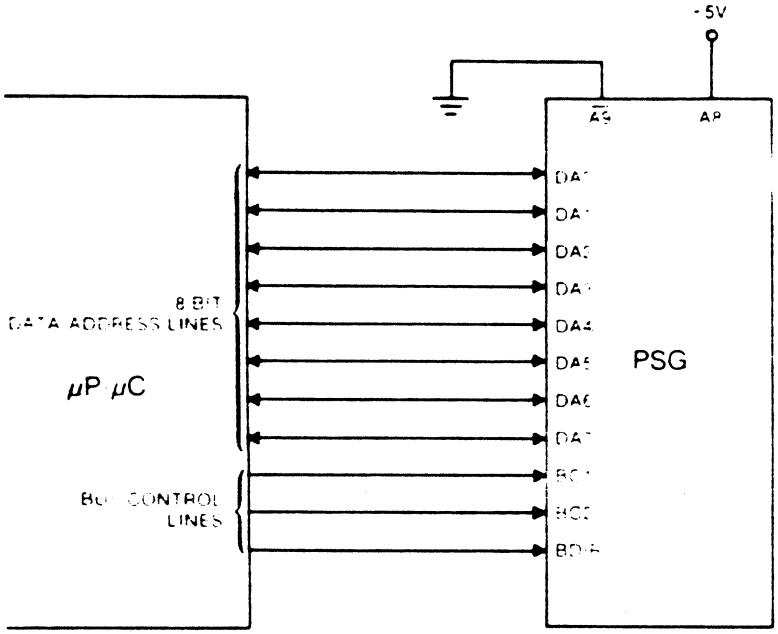
4.5 Microprocessor/ Microcomputer Interface

In Fig. 18, the lines identified DA7--DA0 are the input/output bus bits 7--0. This 8 bit bus is used to pass all data and address information between the AY-3-8910/8912 and the system processor.

BC1, BC2 and BDIR are bus control signals generated by the processor to direct all bus operations. These operations are identified as Latch Address, Write to PSG, Read from PSG, and Inactive.

The following Sections detail specific interfaces to several popular microprocessors/microcomputers.

Fig. 18 MICROPROCESSOR/MICROCOMPUTER INTERFACE



1043

4.6 Interfacing to the PIC 1650

Fig. 19 shows the schematic of an AY-3-8910 demonstrator circuit. This configuration uses a PIC 1650 as the main controller in the circuit. The PIC 1650 is used to scan the keyboard, fetch data from the PROMs, write data to the AY-3-8910 and provide the timing for the AY-3-8910.

The interfacing is direct since the PIC 1650 and the AY-3-8910 operate with compatible supplies and input/output voltages.

This particular schematic illustrates how a microcomputer with additional memory can produce a stand-alone music and sound effects circuit. The circuit as shown operates with manual keyboard selections.

As Fig. 19 shows, the design for the interface connects directly to the output pins of the 1650 and the BC1, BC2, BDIR pins. The software then has the responsibility of manipulating these signals to signal the PSG to perform the proper address latch, read or write operations.

The program routine in this section illustrates code which is used in a hand-held demonstrator unit. This demonstration unit illustrates the range of PSG capabilities, including music, sound effects and I/O control. Note that the generalized routines perform the address latching before every read for convenience.

The "READ ROM" routine illustrates use of the generalized read and write routines to access the outside world through the PSG to read and write.

4.6.1 WRITE DATA ROUTINE

```
 80                 ;WRITE FROM 1650 TO 8910
 81                 ;ADDRESS OF 8910 REG IN 'ADDRES'
 82                 ;DATA TO WRITE IN 'DATA'
 83  024  0066  WRIT1  MOVWF  ADDRES
 84  025  1026  WRITE  MOVF   ADDRES,W  ;GET REGISTER NO
 85  026  0045         MOVWF  IOA        ;SET ADDRESS
 86  027  1006         MOVF   IOB,W      ;GET PRESENT BC1, BC2, BDIR ETC
 87  030  7370         ANDLW  370
 88  031  6404         IORLW  4          ;SET BAR
 89  032  0046         MOVWF  IOB        ;SEND BAR
 90  033  7370         ANDLW  370
 91  034  0046         MOVWF  IOB        ;SEND NACT
 92  035  1027         MOVF   DATA,W
 93  036  0045         MOVWF  IOA        ;PUT DATA ON D/A PINS OF 8910
 94  037  1006         MOVF   IOB,W
 95  040  7370         ANDLW  370
 96  041  6406         IORLW  6
 97  042  0046         MOVWF  IOB        ;SEND DWS
 98  043  7370         ANDLW  370        ;SET UP NACT
 99  044  0046         MOVWF  IOB        ;SEND NACT
100  045  4000         RET               ;RETURN TO CALLING ROUTINE
```


1044

4.6 Interfacing to the PIC 1650 (cont.)

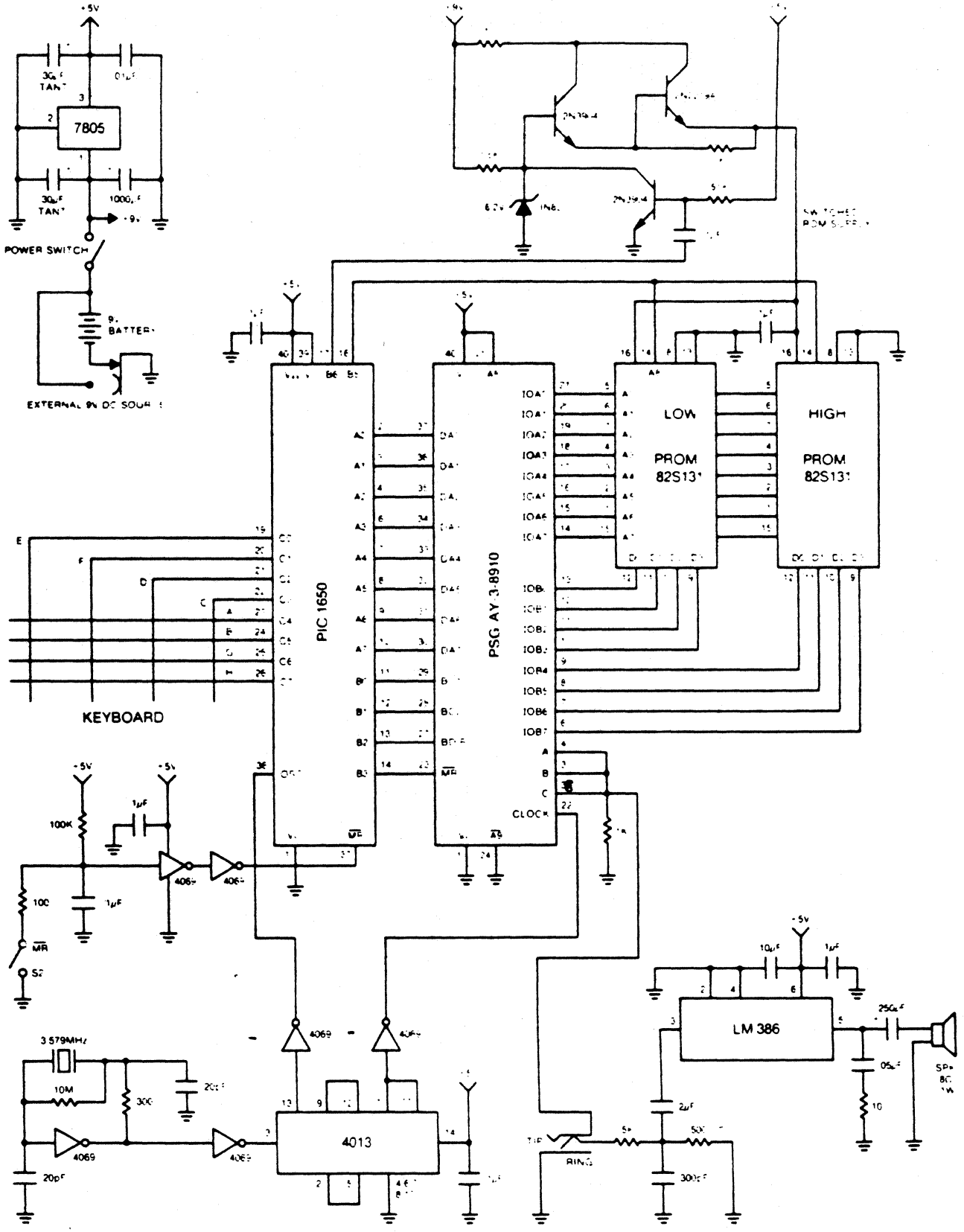
4.6.2 READ DATA ROUTINE

```
51                      ;ADDRESS OF READ IN REGISTER ADDRES
52                      ;AFTER READ, INPUT DATA IN REGISTER 'DATA'
53                      ;ENTRANCE READ1 ASSUMES THAT REGISTER NUM IN W.
54
55  000  0066  READ1   MOVWF   ADDRES     ;BYPASS ADDDRESS STORE
56  001  1026  READ    MOVF    ADDRES,W   ;GET REGISTER NO
57  002  0045          MOVWF   IOA        ;MOVE TO 8910 D A PINS
58  003  1006          MOVF    IOB,W      ;GET PRESENT BC1 BC2 BDIR ETC
59  004  6404          IORLW   4          ;SET BAR
60  005  0046          MOVWF   IOB        ;SEND BAR
61  006  7370          ANDLW   370
62  007  0046          MOVWF   IOB        ;SEND NACT
63  010  6377          MOVLW   377
64  011  0045          MOVWF   IOA        ;SET FOR INPUT
65  012  1006          MOVF    IOB,W
66  013  7370          ANDLW   370
67  014  6403          IORLW   3          ;SET DTB
68  015  0046          MOVWF   IOB        ;SEND DTB
69  016  1005          MOVF    IOA,W
70  017  0067          MOVWF   DATA       ;SAVE DATA
71  020  1006          MOVF    IOB,W
72  021  7370          ANDLW   370
73  022  0046          MOVWF   IOB        ;SEND NACT
74  023  4000          RET                ;RETURN TO CALLING ROUTINE
```

4.6.3 READ ROM ROUTINE

```
106                     ;ADDRESS OF ROM IN W AT ENTRANCE NEXROM
107                     ;ADDRESS OF ROM IN ROMAD AT ENTRANCE ROMRD
108
109                     ;INCREMENTS ROMAD AFTER READ. IF ROM ADDRESS
100                     ;CROSSES 256 BORDER, MAKE UPPER BANK SELECT
111
112                     ;USES 8910 REG 16 FOR ADDRESS
113                     ;8910 REG 17 FOR INPUT DATA
114  046  1030  NEXROM  MOVF    ROMAD,W
115  047  0067  ROMRD   MOVWF   DATA       ;PUT ADDRESS
116  050  6016          MOVLW   16         ;I O A ADDRESS
117  051  0066          MOVWF   ADDRES
118  052  2306          BCF     IOB,6      ;TURN ON ROM
119  053  4425          CALL    WRITE      ;SEND TO IOA
120  054  1266          INCF    ADDRES     ;TO IOB ADDRESS
121  055  4401          CALL    READ       ;GET DATA
122  056  2706          BSF     IOB,6      ;TURN OFF ROM
123  057  1770          INCFSZ  ROMAD      ;TO NEXT LOC
124  060  4000          RET
125  061  2646          BSF     IOB,5      ;SET HIGH SELECT
126  062  4000          RET                ;RETURN TO CALLING ROUTINE
```

Fig. 19 PIC 1650/AY-3-8910 SYSTEM EXAMPLE



1046

4.7 Interfacing to the CP1600/1610

As shown in Fig. 20, the wiring is direct between the AY-3-8910 and a CP1600/1610 microprocessor The levels are compatible thus eliminating any need for level converters. Even the terminology between the IC's remains constant to provide simple-to-follow connections

The CP1600/1610 acts as a controller in this configuration fetching data from ROM's contained elsewhere in the system. The CP1600/1610 also acts as the bus controller developing the necessary timing for the AY-3-8910.

4.7.1 WRITE DATA ROUTINE

The program necessary to write to a selected register is as follows:

```
MVI value, R0; move in value to be written
MVO R0, Reg; write to register
```

The routine to load all registers with the same value is as follows:

```
    MVII Reg 0, R4
    CLRR R0
Here MVO@ R0, R4
    CMPI Reg 0 + 17, R4
    BLT Here
```

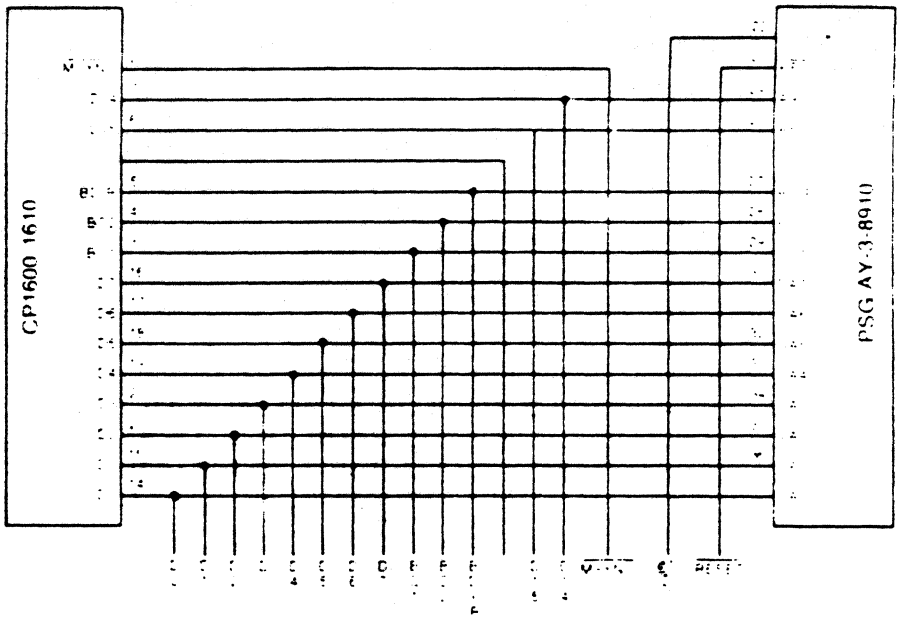
4.7.2 READ DATA ROUTINE

The routine to read from a selected register is as follows:

```
MVI Reg, R0; get data from reg in R0
MVO R0, value: store in memory
```


1047

Fig. 20 CP1600 1610 AY-3-8910 INTERFACE



1048

4.8 Interfacing to the M6800

An M6800 microprocessor can be interfaced with an AY-3-8910/8912 through the addition of an M6820 PIA chip. The I/O ports designated as PA0 to PA7 are used as the 8 bit bus lines and I/O ports PB0 to PB2 are used as the bus control lines. The software routines shown are used to control the latch address, write data, and read data functions for the AY-3-8910/8912.

4.8.1 LATCH ADDRESS ROUTINE

```
;AT ENTRY, B HAS ADDRESS VALUE
;
LATCH CLRA
   STAA 8005 ;GET D DIR A
   LDAA #FF
   STAA 8004 ;OUTPUTS
   LDAA #4
   STAA 8005 ;GET PERIPHERAL A
   STAB 8004 ;FORM ADDR
   STAA 8006
   CLRA
   STAA 8006 ;LATCH ADDRESS
   RTS ;RETURN
```

4.8.2 WRITE DATA ROUTINE

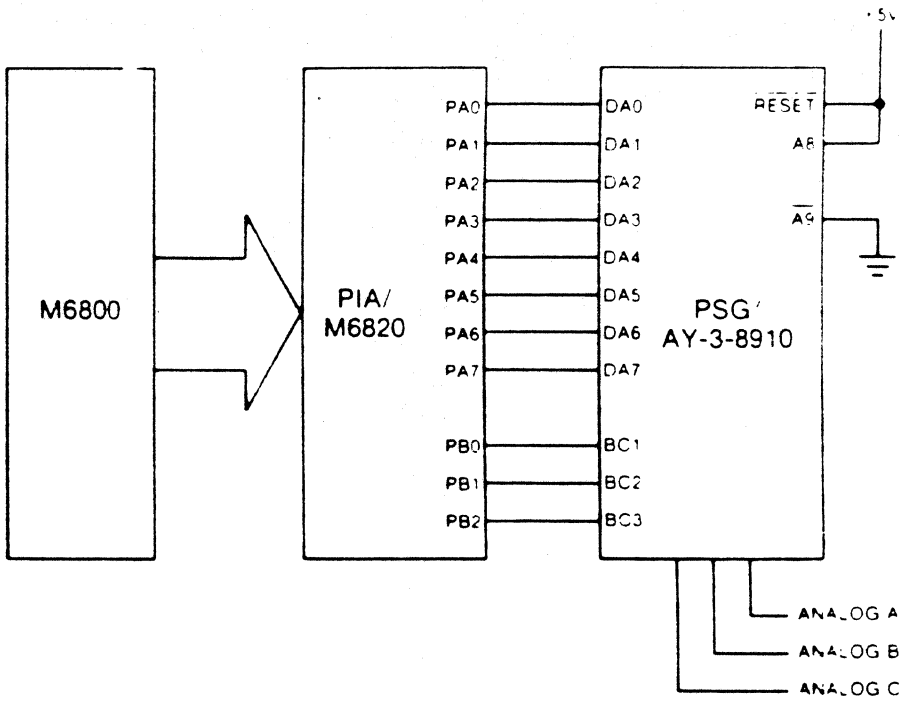
```
;AT ENTRY, B HAD DATA VALUE
;
WRITE STAB 8004 ;FORM DATA
   LDAA #6 ;DWS
   STAA 8006
   CLRA
   STAA 8006 ;WRITE DATA
   RTS ;RETURN
```

4.8.3 READ DATA ROUTINE

```
;AFTER READ, B HAS READ DATA
;
READ STA A 8005 ;GET D DIR
   STA A 8004 ;INPUTS
   LDAA #4
   STA A 8005 ;GET PERIPHERAL
   DECA
   STA A 8006 ;READ MODE
   LDA B 8004 ;READ DATA
   CLRA
   STA A 8006 ;REMOVE READ MODE
   RTS        ;RETURN
```


1041

Fig. 21 M6800 AY-3-8910 INTERFACE



1050

4.9 Interfacing to the 8080 S100 Bus

The sample S100 bus design provides for reading and writing the PSG using only an 8080 "IN" or "OUT" instruction to the proper address. Another feature of the design is the provision for multiple PSG devices to be connected to a single bus. The system described is presently running two PSG's, one to each of two stereo channels.

As can be seen from the read and write routines in the illustrative program, the program overhead necessary to communicate with the PSG is minimal.

4.9.1 LATCH ADDRESS ROUTINE

```
PORTADDR EQU 80H ;ADDRESS TRANSFER PORT ADDRESS
PORTDATA EQU 81H ;DATA TRANSFER PORT ADDRESS
;
;THIS ROUTINE WILL TRANSFER THE CONTENTS OF
;8080 REGISTER C TO THE PSG ADDRESS REGISTER
PSGBAR   MOV   A,C ;GET C IN A FOR OUT
         OUT   PORTBAR ;SEND TO ADDRESS PORT
         RET
```

.9.2 WRITE DATA ROUTINE

```
;
;ROUTINE TO WRITE THE CONTENTS OF 8080 REGISTER B
;TO THE PSG REGISTER SPECIFIED BY 8080 REGISTER C
;
PSGWRITE CALL  PSGBAR ;GET ADDRESS LATCHED
         MOV   A,B ;GET VALUE IN A FOR TRANSFER
         OUT   PORTDATA ;PUT TO PSG REGISTER
         RET
```

4.9.3 READ DATA ROUTINE

```
;
;ROUTINE TO READ THE PSG REGISTER SPECIFIED
;BY THE 8080 REGISTER C AND RETURN THE DATA
;IN 8080 REGISTER B
;
PSGREAD  CALL  PSGBAR
         IN    PORTDATA ;GET REGISTER DATA
         MOV   B,A GET IN TRANSFER REGISTER
         RET
```

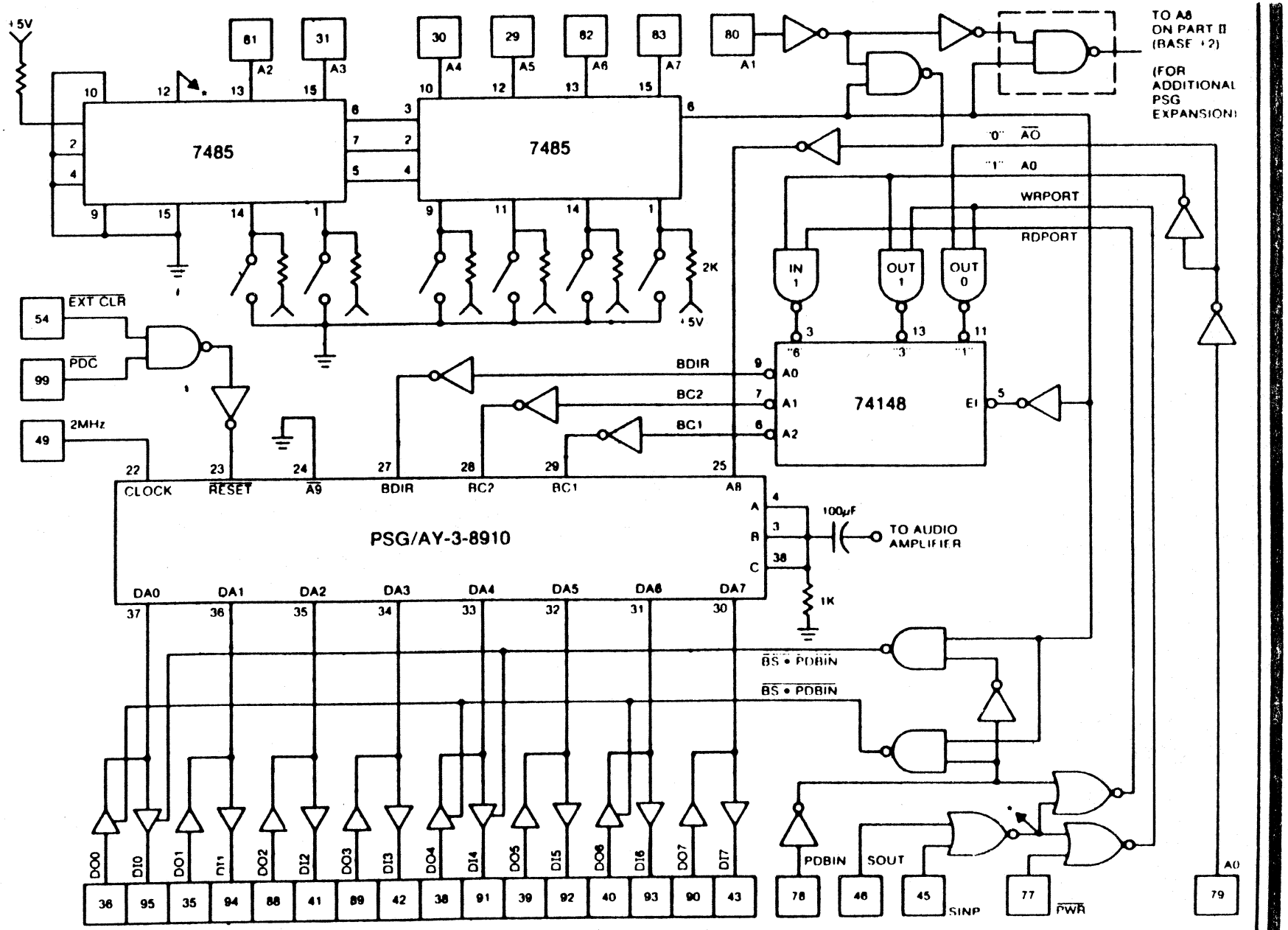


Fig. 22 8080 S100 BUS/AY-3-8910 INTERFACE

1052

5 MUSIC GENERATION

The production of music involves the creation of series of frequencies which are pleasing to the human ear (setting critical evaluation aside). This involves essentially mathematical relationships, making the application ideal for digital devices. For example, the shifting up or down in octaves is a multiplication or division by a power of 2, which is a simple shift operation for most microprocessors.

Another factor in music generation is "communication". The composer must be able to convey his tune ideas so that a musician or group of musicians can reproduce the composer's ideas—often on widely differing instruments. This concept involves "tuning" the instruments to a standard set of frequencies and following a set rhythm pattern. The tuning frequency most widely used is based on the third octave note "A" of 440Hz, the "Equal Tempered Chromatic Scale".

Although it is easy to construct recognizable tunes using only one note at a time, the simultaneous sounding of more than one note to produce chords and counterpoint vastly increases the quality of the sound. This feature is easily achieved in the PSG since three channels are provided, each independently programmable.

5.1 Note Generation

Since notes are formed by sustaining a particular frequency for a preset period of time at a varying amplitude, the PSG performs this function with a series of simple register loads. The method used in many cases is to obtain register load values for first octave notes and to shift to the correct octave at playtime.

The chart in Fig. 23 lists a full 8 octaves of notes from a low of C1 (32.703Hz) to a high of B8 (7902.080Hz). Assuming an input clock frequency of 1.78977MHz (one half the standard "color" crystal frequency of 3.579545MHz), and applying the formulas of Section 3.1 for calculating Tone Period register load values, results in the register values shown. The nature of the PSG divider scheme produces a high degree of accuracy for low frequencies, less for high frequencies. This can be seen in the chart in the comparison of "ideal frequencies" and "actual frequencies", with the ideal frequencies being those of the Equal Tempered Chromatic Scale, and the actual frequencies being the "best fit" values from the formula calculation.

NOTE	OCTAVE	IDEAL FREQUENCY	ACTUAL FREQUENCY	12-BIT REGISTER VALUE IN OCTAL			
C	1	32 703	32 698	6	5	3	5
C#	1	34 648	34 653	6	2	3	4
D	1	36 708	36 712	5	7	4	7
D#	1	38 891	38 895	5	4	7	4
E	1	41 203	41 201	5	2	3	3
F	1	43 654	43 662	5	0	0	2
F#	1	46 249	46 243	4	5	6	3
G	1	48 999	48 997	4	3	5	3
G#	1	51 913	51 908	4	1	5	3
A	1	55 000	54 995	3	7	6	2
A#	1	58 270	58 261	3	6	0	0
B	1	61 735	61 733	3	4	2	4
C	2	65 406	65 416	3	2	5	6
C#	2	69 296	69 307	3	1	1	6
D	2	73 416	73 399	2	7	6	4
D#	2	77 782	77 789	2	6	3	6
E	2	82 406	82 432	2	5	1	5
F	2	87 308	87 323	2	4	0	1
F#	2	92 498	92 523	2	2	7	1
G	2	97 998	98 037	2	1	6	5
G#	2	103 826	103 863	2	0	6	5
A	2	110 000	109 991	1	7	7	1
A#	2	116 540	116 522	1	7	0	0
B	2	123 470	123 467	1	6	1	2
C	3	130 812	130 831	1	5	2	7
C#	3	138 592	138 613	1	4	4	7
D	3	146 832	146 799	1	3	7	2
D#	3	155 564	155 578	1	3	1	7
E	3	164 812	164 743	1	2	4	7
F	3	174 616	174 510	1	2	0	1
F#	3	184 996	184 894	1	1	3	5
G	3	195 996	195 903	1	0	7	3
G#	3	207 652	207 534	1	0	3	3
A	3	220 000	220 198	0	7	7	4
A#	3	233 080	233 043	0	7	4	0
B	3	246 940	246 933	0	7	0	5
C	4	261 624	261 357	0	6	5	4
C#	4	277 184	276 883	0	6	2	4
D	4	293 664	293 598	0	5	7	5
D#	4	311 128	310 724	0	5	5	0
E	4	329 624	329 973	0	5	2	3
F	4	349 232	349 565	0	5	0	0
F#	4	369 992	370 400	0	4	5	6
G	4	391 992	392 494	0	4	3	5
G#	4	415 304	415 839	0	4	1	5
A	4	440 000	440 397	0	3	7	6
A#	4	466 160	466 087	0	3	6	0
B	4	493 880	494 959	0	3	4	2

NOTE	OCTAVE	IDEAL FREQUENCY	ACTUAL FREQUENCY	12-BIT REGISTER VALUE IN OCTAL			
C	5	523 248	522 714	0	3	2	6
C#	5	554 368	553 766	0	3	1	2
D	5	587 328	588 741	0	2	7	6
D#	5	622 256	621 449	0	2	6	4
E	5	659 248	658 005	0	2	5	2
F	5	698 464	699 130	0	2	4	0
F#	5	739 984	740 800	0	2	2	7
G	5	783 984	782 243	0	2	1	7
G#	5	830 608	828 598	0	2	0	7
A	5	880 000	880 794	0	1	7	7
A#	5	932 320	932 173	0	1	7	0
B	5	987 760	989 918	0	1	6	1
C	6	1046 496	1045 428	0	1	5	3
C#	6	1108 736	1107 532	0	1	4	5
D	6	1174 656	1177 482	0	1	3	7
D#	6	1244 512	1242 898	0	1	3	2
E	6	1318 496	1316 009	0	1	2	5
F	6	1396 928	1398 260	0	1	2	0
F#	6	1479 968	1471 852	0	1	1	4
G	6	1567 968	1575 504	0	1	0	7
G#	6	1661 216	1669 564	0	1	0	3
A	6	1760 000	1747 825	0	1	0	0
A#	6	1864 640	1864 346	0	0	7	4
B	6	1975 520	1962 470	0	0	7	1
C	7	2092 992	2110 581	0	0	6	5
C#	7	2217 472	2237 216	0	0	6	2
D	7	2349 312	2330 433	0	0	6	0
D#	7	2489 024	2485 795	0	0	5	5
E	7	2636 992	2663 352	0	0	5	2
F	7	2793 856	2796 520	0	0	5	0
F#	7	2959 936	2943 705	0	0	4	6
G	7	3135 936	3107 244	0	0	4	4
G#	7	3322 432	3290 023	0	0	4	2
A	7	3520 000	3495 649	0	0	4	0
A#	7	3729 280	3728 693	0	0	3	6
B	7	3951 040	3995 028	0	0	3	4
C	8	4185 984	4142 992	0	0	3	3
C#	8	4434 944	4474 431	0	0	3	1
D	8	4698 624	4660 866	0	0	3	0
D#	8	4978 048	5084 581	0	0	2	6
E	8	5273 984	5326 704	0	0	2	5
F	8	5587 712	5593 039	0	0	2	4
F#	8	5919 872	5887 410	0	0	2	3
G	8	6271 872	6214 488	0	0	2	2
G#	8	6644 864	6580 046	0	0	2	1
A	8	7040 000	6991 299	0	0	2	0
A#	8	7458 560	7457 385	0	0	1	7
B	8	7902 080	7990 056	0	0	1	6

Fig. 23 EQUAL TEMPERED CHROMATIC SCALE (f = 1.78977MHz)

5.2 Tune Entry/ Playback

One of the methods of entering a composition into a computer memory would be to utilize a keyboard to pass number and alphabetic information concerning the composer's wishes. An alternate method would be to scan a positional series of switches (like a piano keyboard) to determine note, volume and duration data.

Since flexibility in tune entry is desired, it is important to allow the composer to specify certain constants of entry such as octave, pitch or tempo, and have these entries normalized to a known value.

5.3 Tune Variations

One of the significant features of a microcomputer based music player is the ability to modify the tune once it has been recorded. Among the simpler variations are:

5.3.1 OCTAVE SHIFT

If an octave constant is added to the octave of the recorded note prior to storing the value in the PSG register, dynamic pitch changes can be obtained. The programming effect would be to shift one bit left for each lower octave and one bit right for each higher octave. For example, the effect will be that a tune written to play on a piano will sound like bells if a multiple octave up modification is performed.

5.3.2 KEY

One measure of the virtuosity of a musician is his ability to modify the "key" or suboctave shift of a composition. The logical description of key transposition is to shift each note up or down by a predetermined number of notes from the original. For example, a piece written in C and played in C# would have all C notes shifted to C#, C# shifted to D, etc. (Note that the case must be considered where B of one octave is shifted to C of the next higher octave.) All of these operations require that the one of twelve note identification must be retained in the recorded representation.

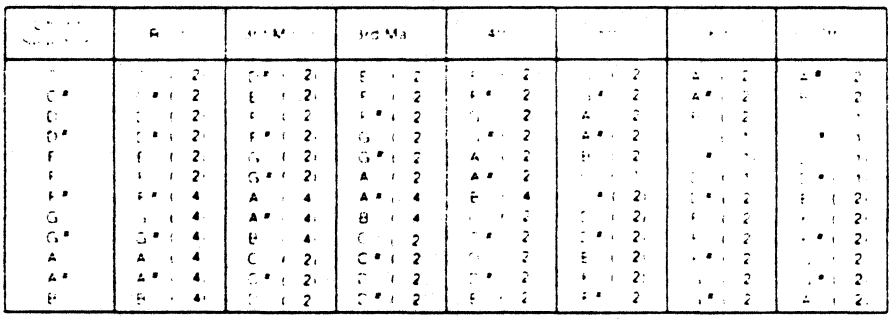
5.3.3 TEMPO

The duration of each recorded note is best expressed in terms of "ticks" of an overall "tempo clock". At playtime, the total duration can be obtained by programatically multiplying the individual note to "slow down" or "speed up" the tune without changing the crucial time relationship between the notes. This can be accomplished by imbedding the note timing loops within the tempo timing loops for simple operation.

5.3.4 CHORDS

There are certain combinations of notes which when played simultaneously produce pleasant combinations. These "chords" can be easily formed from a base note by performing octave and key changes on two notes, which are played with the main note. These relationships are illustrated in Fig. 24, which lists the various note constants which will produce musical chords. A chord with a particular quality may be formed by playing its root, a 3rd Minor or Major, and other notes from the chord chart. For example, a C Major chord is formed from C(÷2), E(÷2), and G(÷2).

Fig. 24 CHORD SELECTION CHART



1056

5.4 Sound Variation

5.4.1 RELATIVE CHANNEL VOLUME

The independently programmable amplitude control for each channel allows up to 16 levels if using the processor controlled amplitude mode (bit 4 of registers 10, 11 or 12=0). In the case of a decaying or steady note, when a note is played or "fired", a frequency may be set up in the coarse and fine tune registers and then an amplitude value placed in the respective register 10, 11 or 12. The value which is placed to play the tune can be an independent variable, allowing channels to play their respective melody lines with varying force.

5.4.2 DECAY

The main difference between a "piano" sound and an "organ" sound is the speed with which the note loses volume. If all of the notes can be decayed at a uniform rate, the automatic envelope generator can be set to produce a decaying waveform. Each of the three channels can have the same decay constant but differing playing times to simulate the same instrument with differing note-strike times.

5.4.3 OTHER EFFECTS

The addition of variable noise to any or all of the channels can produce modification effects such "breathing" with a wind instrument. Or noise can be used alone to produce a drum rhythm. The fact that the noise dominant frequencies are variable allows "synthesizer" type effects with simple processor interaction.

Other pleasing effects include vibrato and tremolo, the cyclical variation of the frequency and volume. Because an intelligent microprocessor is controlling the effect, they can be all keyed to the tune itself or to other external stimuli.

5.5 Applications

While many applications of the PSG in music generation are apparent, for instance in the area of toys and games, other applications are possible even in the area of high accuracy sophisticated musical instruments such as high-end electronic organs. With tone frequencies generated from another source to meet the exacting requirements of organ operation, the PSG can be used as a complex envelope generator. The PSG is also effective for generating bass notes and rhythms with percussion instruments, taking advantage of the PSG's high accuracy in producing low frequency notes. The following paragraphs detail examples of these applications.

5.5.1 ORGAN ENVELOPE GENERATION

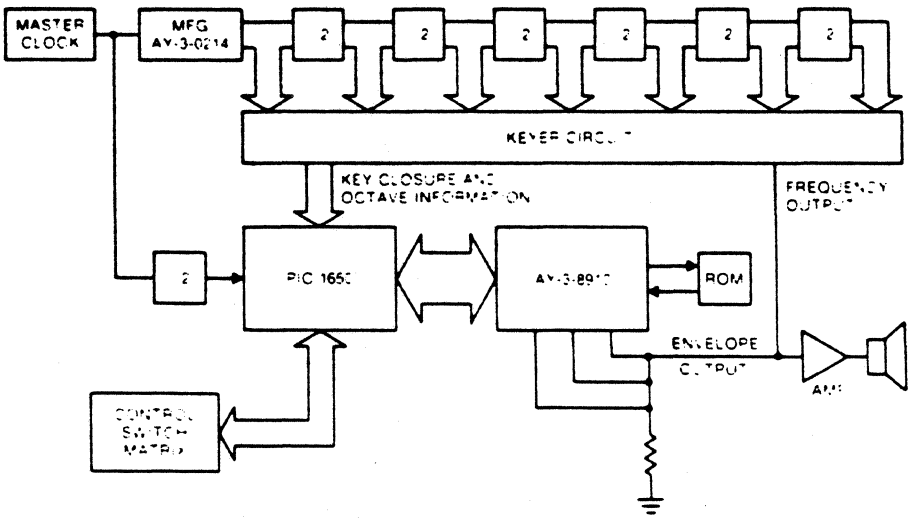
The envelope generation diagram shown in Fig. 25 illustrates how an AY-3-8910 can be configured to produce envelopes for organ voicing. All functions are controlled by a microcomputer.

The basis of this system consists of a master frequency generator with a string of dividers. This produces all frequencies for the keyboard. The microcomputer and the AY-3-8910 are actually used to replace the usual components of voicing filters that would ordinarily be used in an electronic organ.

The microcomputer shown is a GI PIC 1650 controlled by inputs from the keyboard keyer circuit and a control switch matrix. The keyer inputs octave and key closure information to develop the envelope amplitude and duration for the note to be played. The control switch matrix can be used to control sustain and add other special effects. The ROM shown connected to the AY-3-8910 is optional depending on the amount of data necessary for the microcomputer.

The system shown here may also consist of multiple AY-3-8910's, all controlled by a single microcomputer. It represents an economical solution to developing voicing control with a minimum of components.

Fig. 25 ORGAN ENVELOPE GENERATION



1057

5.5 Applications (cont.)

5.5.2 ORGAN RHYTHM GENERATION

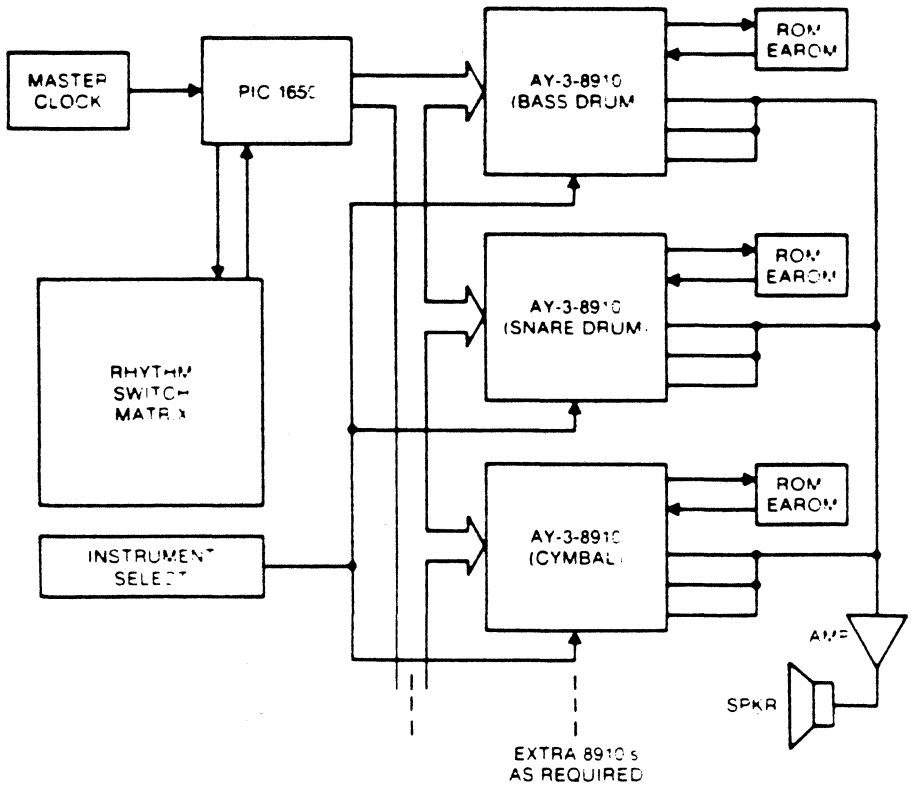
The rhythm generation diagram (Fig. 26) illustrates a simplified version of how a microcomputer can be implemented with the AY-3-8910 to provide a percussion instrument section for an electronic organ.

The microcomputer used in this case could be a GI PIC 1650 which can be internally programmed to drive a series of AY-3-8910's, all hardwired to an I/O port of the PIC. Each AY-3-8910 provides a separate output envelope and frequency of the instrument it is to synthesize.

The Rhythm Switch Matrix is used to select any preprogrammed rhythm pattern and tempo from the PIC. The Instrument Select switches allow manual in/out selection of the 8910's via the A8 and A9 address lines providing additional instrument sound variations. These switches are intended to be user-selected and mounted in a convenient position on the instrument.

In addition, optional ROMs could be added to the PSG I/O ports, saving microcomputer ports, to provide extra rhythm length or number of patterns. These ROMs could also be replaced by EAROMs to provide user rhythm programming from a modified Rhythm Switch Matrix. The programmable rhythm feature could be used to add new or original user rhythms to update the instrument.

Fig. 26 ORGAN RHYTHM GENERATION



1051

6 SOUND EFFECTS GENERATION

One of the main uses of the PSG is to produce non-musical sound effects to accompany visual action or as a feature in itself. The following sections outline techniques and provide actual examples of some popular effects. All examples are based on a 1.78977MHz PSG clock.

6.1 Tone Only Effects

Many effects are possible using only the tone generation capability of the PSG without adding noise and without using the PSG's envelope generation capability. Examples of this type of effect would include telephone tone frequencies (two distinct frequencies produced simultaneously) or the European Siren effect listed in Fig. 27 (two distinct frequencies sequentially produced).

Fig. 27 EUROPEAN SIREN SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R0	376	Set Channel A Tone period to 2.27ms
R1	000	(440Hz)
R7	076	Enable Tone only on Channel A only
R10	017	Select maximum amplitude on Channel A
	(Wait approximately 350ms before continuing)	
R0	126	Set Channel A Tone period to 5.346ms
R1	001	(187Hz).
	(Wait approximately 350ms before continuing)	
R10	000	Turn off Channel A to end sound effect

1060

6.2 Noise Only Effects

Some of the more commonly required sounds require only the use of noise and the envelope generator (or processor control of channel envelope if other channels are using the envelope generator).

Examples of this, which can be seen in Figs. 28 and 29, are gunshot and explosion. In both cases pure noise is used with a decaying envelope. In the examples shown the only changes are in the length of the envelope as modified by the coarse tune register and in the noise period. Note that a significantly lower explosion can be obtained by using all three channels operating with the same parameters.

Fig. 28 GUNSHOT SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R6	017	Set Noise period to mid-value
R7	007	Enable Noise only on Channels A,B,C
R10	020	Select full amplitude range under direct control of Envelope Generator
R11	020	
R12	020	
R14	020	Set Envelope period to 0.586 seconds
R15	000	Select Envelope "decay", one cycle only

Fig. 29 EXPLOSION SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R6	000	Set Noise period to max. value
R7	007	Enable Noise only, on Channels A,B,C
R10	020	Select full amplitude range under direct control of Envelope Generator
R11	020	
R12	020	
R14	070	Set Envelope period to 2.05 seconds
R15	000	Select Envelope "decay", one cycle only

1061

6.3 Frequency Sweep Effects

The Laser, Whistling Bomb, Wolf Whistle, and Race Car sounds in Figs. 30 thru 33 all utilize frequency sweeping effects. In all cases they involve the increasing or decreasing of the values in the tone period registers with variable start, end, and time between frequency changes. For example, the sweep speed of the Laser is much more rapid than the high gear accelerate in the race car, yet both use the same computer routine with differing parameters.

Other easily achievable results include "doppler" and noise sweep effects. The sweeping of the noise clocking register (R6) produces a "doppler" effect which seems well suited for "space war" type games.

Fig. 30 LASER SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R7	076	Enable Tone only on Channel A only
R10	017	Select maximum amplitude on Channel A
R0	060 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 3ms wait time between each step from 060 to 160 (0.429ms/2330Hz to 1.0ms/1000Hz)
R0	160 (end)	
R10	000	Turn off Channel A to end sound effect

Fig. 31 WHISTLING BOMB SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R7	076	Enable Tone only on Channel A only
R10	017	Select maximum amplitude on Channel A
R0	060 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 25ms wait time between each step from 060 to 300 (0.429ms/2330Hz to 1.72ms/582Hz).
R0	300 (end)	

After above loop is completed, follow with sequence in Fig. 28.

1062

6.4 Multi-Channel Effects

Because of the independent architecture of the PSG, many rather complex effects are possible without burdening the processor. For example, the Wolf Whistle effect in Fig. 32 shows two channels in use to add constant breath hissing noise to the three concentrated frequency sweeps of the whistle. Once the noise is put on the channel, the processor only need be concerned with the frequency sweep operation.

Fig 32 WOLF WHISTLE SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R6	001	Set Noise period to minimum value
R7	056	Enable Tone on Channel A, Noise on Channel B
R10	017	Select maximum amplitude on Channel A
R11	011	Select lower amplitude on Channel B
R0	100 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 12ms wait time between each step from 100 to 040 (0.572ms/1748Hz to 0.286ms/3496Hz)
R0	040 (end)	
(Wait approximately 150ms before continuing)		
R0	100 (start)	A processor loop with approximately 25ms wait time between each step from 100 to 060 (0.572ms/1748Hz to 0.429ms/2331Hz)
R0	060 (end)	
R0	060 (start)	A processor loop with approximately 6ms wait time between each step from 060 to 150 (0.429ms/2331Hz to 0.930ms/1075Hz)
R0	150 (end)	
R10	000	Turn off Channels A and B to end effect
R11	000	

Fig. 33 RACE CAR SOUND EFFECT CHART

Register #	Octal Load Value	Explanation
Any not specified	000	—
R3	017	Set Channel B Tone period to 34.33ms (29Hz)
R7	074	Enable Tones only on Channels A and B
R10	017	Select maximum amplitude on Channel A
R11	012	Select lower amplitude on Channel B
*R1/R0	013/000 (start)	Sweep effect for Channel A Tone period via a processor loop with approximately 3ms wait time between each step from 013/000 to 004/000 (25.17ms/39.7Hz to 9.15ms/109.3Hz)
*R1/R0	004/000 (end)	
R1/R0	011/000 (start)	A processor loop with approximately 3ms wait time between each step from 011/000 to 003/000 (20.6ms/48.5Hz to 6.87ms/145.6Hz)
R1/R0	003/000 (end)	
R1/R0	006/000 (start)	A processor loop with approximately 6ms wait time between each step from 006/000 to 001/000 (13.73ms/72.8Hz to 2.29ms/436.7Hz)
R1/R0	001/000 (end)	
R10	000	Turn off Channels A and B to end effect
R11	000	

*Decrement R1/R0 as a whole number e.g. start at 013/000, then 012/377 then 012/376 etc

1063

7 ELECTRICAL SPECIFICATIONS

7.1 Maximum Ratings

Storage Temperature ..... −55°C to +150°C
Operating Temperature ..... 0°C to +40°C
V_{CC} and all other input and output voltages with respect to V_{SS} ..... −0.3V to +8.0V

Exceeding these ratings could cause permanent damage to these devices Functional operation at these conditions is not implied—operating conditions are specified below

7.2 Standard Conditions

V_{CC} = +5V ±5%
V_{SS} = GND
Operating temperature: 0°C to +40°C

7.3 DC Characteristics

Characteristic	Sym	Min.	Typ.*	Max.	Units	Conditions
All Inputs						
Logic "0"	V_{IL}	0	—	0.6	V	
Logic "1"	V_{IH}	2.4	—	V_{CC}	V	
All Outputs (except Analog Channel Outputs)						
Logic "0"	V_{OL}	0	—	0.5	V	I_{OL} = 1.6 mA, 20pF
Logic "1"	V_{OH}	2.4	—	V_{CC}	V	I_{OH} = 100μA, 20pF
Analog Channel Outputs	V_O	0	—	60	dB	Test circuit Fig. 34
Power Supply Current	I_{CC}	—	45	75	mA	

*Typical values are at +25°C and nominal voltages

Fig. 34 ANALOG CHANNEL OUTPUT TEST CIRCUIT

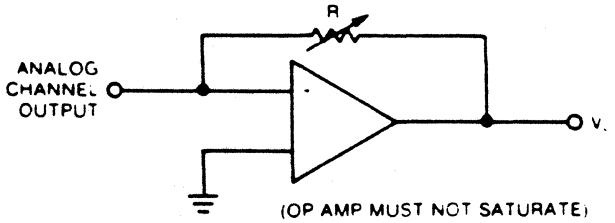
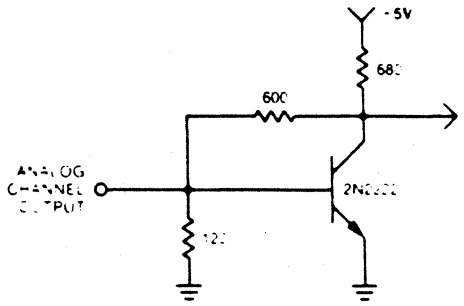


Fig. 35 CURRENT TO VOLTAGE CONVERTER



1064

7.4 AC Characteristics

Characteristic	Sym	Min.	Typ.*	Max.	Units	Conditions
Clock Input						
Frequency	f_c	1.0	—	2.0	MHz	Fig. 36
Rise time	t_r	—	—	50	ns	Fig. 36
Fall time	t_f	—	—	50	ns	Fig. 36
Duty Cycle	—	25	50	75	%	Fig. 36
Bus Signals (BDIR, BC2, BC1)						
Associative Delay Time	t_{BC}	—	—	50	ns	Fig. 36
$\overline{\text{Reset}}$						
$\overline{\text{Reset}}$ Pulse Width	t_{RW}	500	—	—	ns	Fig. 37
$\overline{\text{Reset}}$ to Bus Control Delay Time	t_{RB}	100	—	—	ns	Fig. 37
A9, A8, DA7--DA0 (Address Mode)						
Address Setup Time	t_{AS}	400	—	—	ns	Fig. 38
Address Hold Time	t_{AH}	100	—	—	ns	Fig. 38
DA7--DA0 (Write Mode)						
Write Data Pulse Width	t_{DW}	500	—	10 000	ns	Fig. 39
Write Data Setup Time	t_{DS}	50	—	—	ns	Fig. 39
Write Data Hold Time	t_{DH}	100	—	—	ns	Fig. 39
DA7--DA0 (Read Mode)						
Read Data Access Time	t_{DA}	—	250	500	ns	Fig. 40
DA7--DA0 (Inactive Mode)						
Tristate Delay Time	t_{TS}	—	100	200	ns	Fig. 40

*Typical values are at 25°C and nominal voltages.

Fig. 36 CLOCK AND BUS SIGNAL TIMING

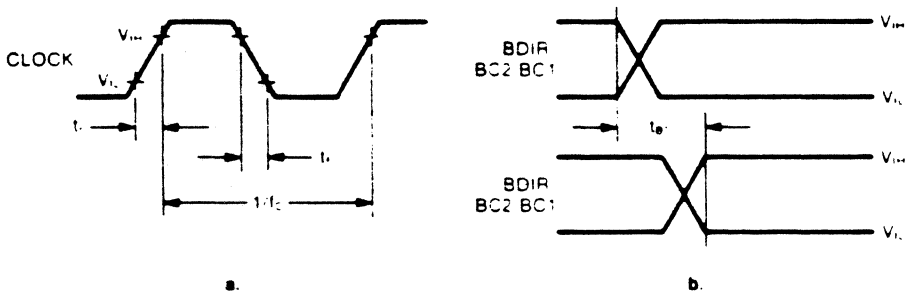
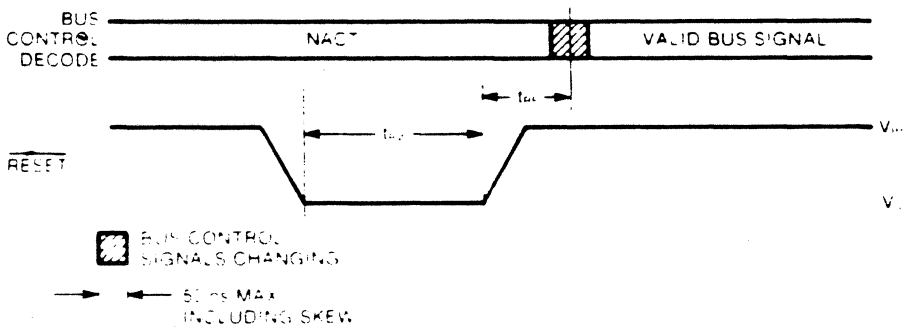


Fig. 37 RESET TIMING



1065

Fig. 38 **LATCH ADDRESS TIMING**

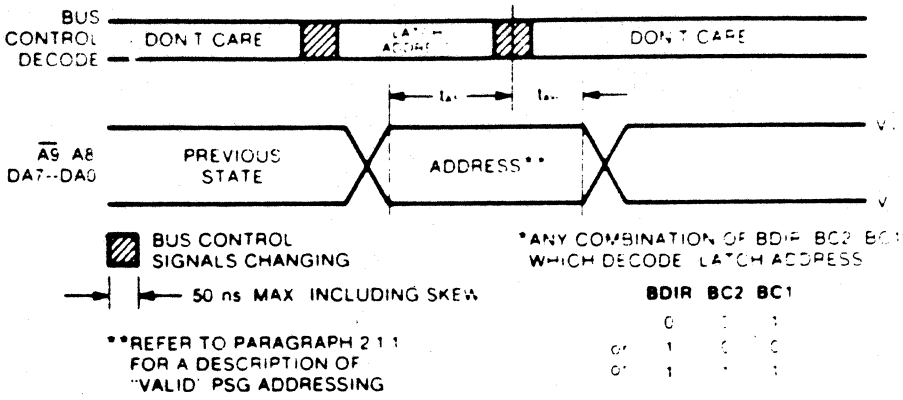


Fig. 39 **WRITE DATA TIMING**

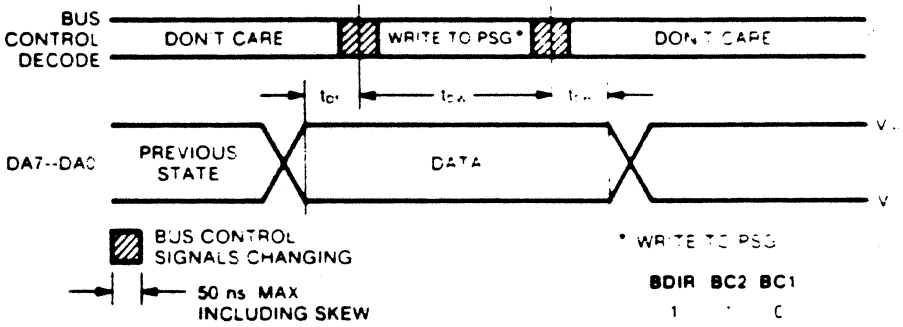
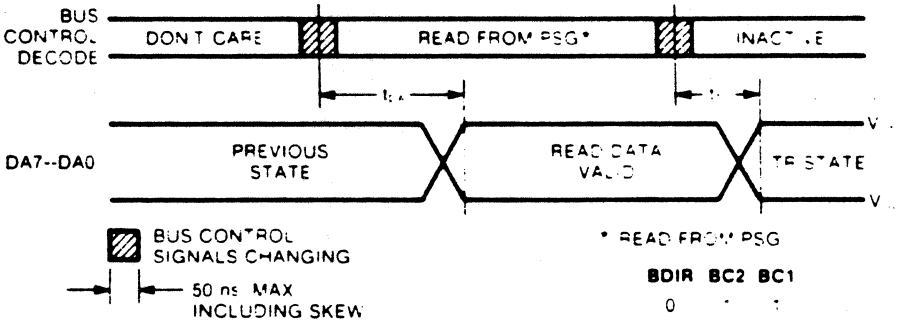


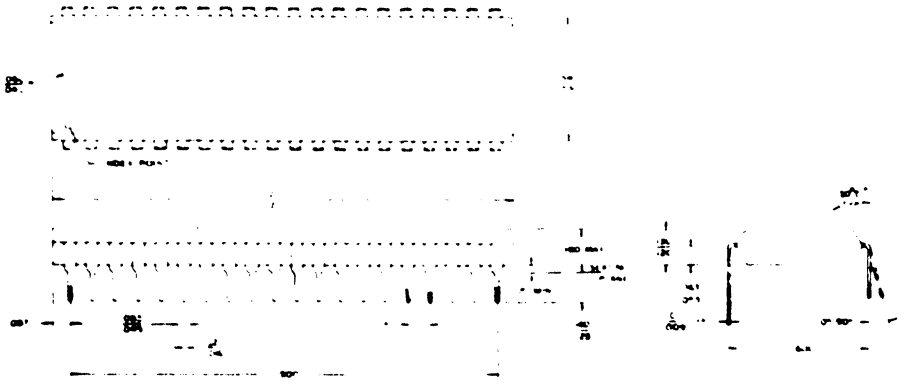
Fig. 40 **READ DATA TIMING**



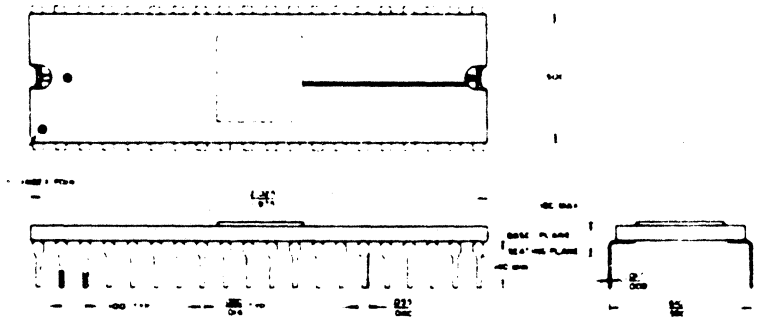
1066

7.5
Package Outlines

Fig. 41 40 LEAD DUAL IN LINE PACKAGES (for AY-3-8910)



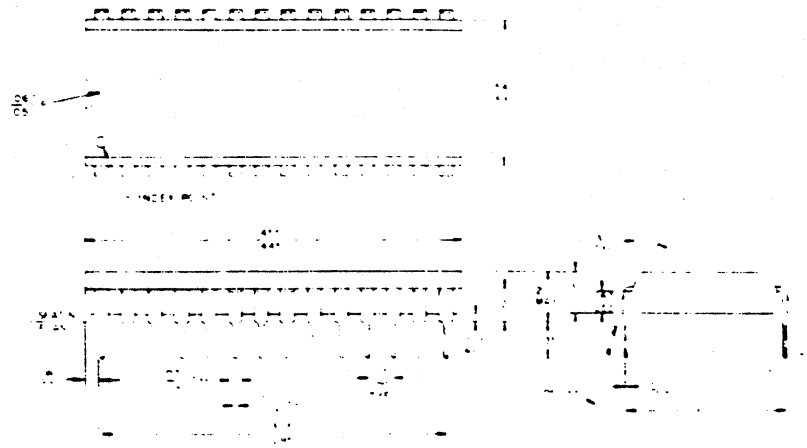
PLASTIC



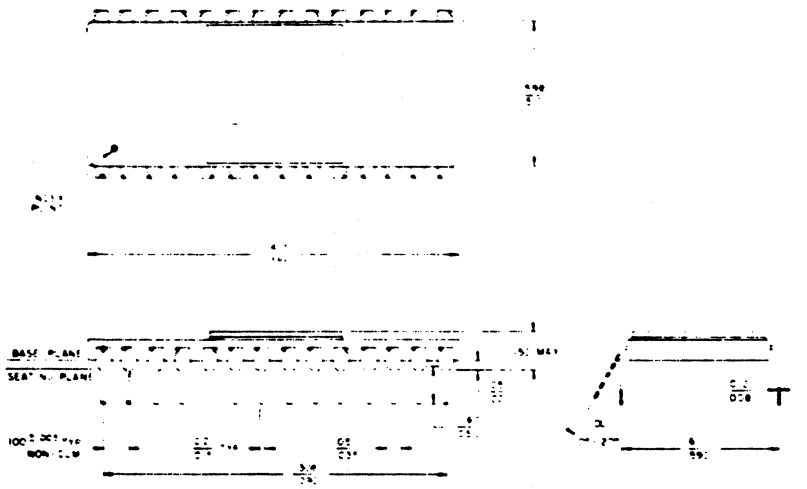
CERAMIC

1067

Fig 42 28 LEAD DUAL IN LINE PACKAGES (for AY-3-8912)



PLASTIC



CERAMIC

1068

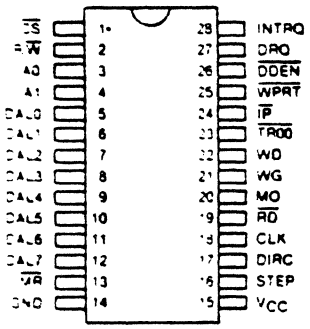
WESTERN DIGITAL
C O R P O R A T I O N

PRELIMINARY

WD1770/1772 5¼" Floppy Disk Controller/Formatter

FEATURES

- 28 PIN DIP
- SINGLE 5V SUPPLY
- BUILT-IN DATA SEPARATOR
- BUILT-IN WRITE PRECOMPENSATION
- 5¼" SINGLE AND DOUBLE DENSITY
- MOTOR CONTROL
- 128, 256, 512 OR 1024 SECTOR LENGTHS
- TTL COMPATIBLE
- 8 BIT BIDIRECTIONAL DATA BUS
- TWO VERSIONS AVAILABLE
  WD1770 = STANDARD 179X STEP RATES
  WD1772 = FASTER STEP RATES



PIN DESIGNATION

DESCRIPTION

The WD1770 is a MOS/LSI device which performs the functions of a 5¼" Floppy Disk Controller/Formatter. It is similar to its predecessor, the WD179X, but also contains a digital data separator and write precompensation circuitry. The drive side of the interface needs no additional logic except for buffers/receivers. Designed for 5¼" single or double density operation, the device contains a programmable Motor On signal.

The WD1770 is implemented in NMOS silicon gate technology and is available in a 28 pin dual-in-line.

The WD1770 is a low cost version of the FD179X Floppy Disk Controller/Formatter. It is compatible with the 179X, but has a built-in digital data separator and write precompensation circuits. A single read line (RD, Pin 19) is the only input required to recover serial FM or MFM data from the disk drive. The device has been specifically designed for control of 5¼" floppy disk drives with data rates of 125 KBits/Sec (single density) and 250 KBits/Sec (double density). In addition, write precompensation of 125 Nsec from nominal can be enabled at any point through simple software commands. Another programmable feature, Motor On, has been incorporated to enable the spindle motor automatically prior to operating a selected drive.

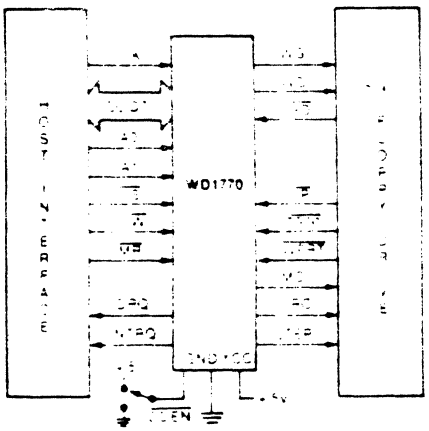
Two versions of the WD1770 are available. The standard version is compatible with the 179X stepping rates, while the WD1772 offers stepping rates of 2, 3, 5 and 6 msec.

The processor interface consists of an 8-bit bidirectional bus for transfer of status, data, and commands. All host communication with the drive occurs through these data lines. They are capable of driving one standard TTL load or three "LS" loads.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
1	CHIP SELECT	$\overline{CS}$	A logic low on this input selects the chip and enable Host communication with the device.
2	READ/WRITE	R/$\overline{W}$	A logic high on this input controls the placement of data on the $\overline{D0}$-$\overline{D7}$ lines from a selected register, while a logic low causes a write operation to a selected register.
3,4	ADDRESS 0,1	A0, A1	These two inputs select a register to Read/Write data: $\overline{CS}$ A1 A0 R/$\overline{W}$ = 1 R/$\overline{W}$ = 0 0 0 0 Status Reg Command Reg 0 0 1 Track Reg Track Reg 0 1 0 Sector Reg Sector Reg 0 1 1 Data Reg Data Reg
5-12	DATA ACCESS LINES 0 THROUGH 7	DAL0-DAL7	Eight bit bidirectional bus used for transfer of data, control, or status. This bus is enabled by $\overline{CS}$ and R/$\overline{W}$. Each line will drive one TTL load.
13	MASTER RESET	$\overline{MR}$	A logic low pulse on this line resets the device and initializes the status register (internal pull-up).
14	GROUND	GND	Ground.
15	POWER SUPPLY	V_{CC}	+5V ± 5% power supply input.
16	STEP	STEP	The Step output contains a pulse for each step of the drive's R/$\overline{W}$ head. The WD1770 and WD1772 offer different step rates.
17	DIRECTION	DIRC	The Direction output is high when stepping in towards the center of the diskette, and low when stepping out.
18	CLOCK	CLK	This input requires a free-running 50% duty cycle clock (for internal timing) at 8 MHZ ± 1%.
19	READ DATA	$\overline{RD}$	This active low input is the raw data line containing both clock and data pulses from the drive.
20	MOTOR ON	MO	Active high output used to enable the spindle motor prior to read, write or stepping operations.
21	WRITE GATE	WG	This output is made valid prior to writing on the diskette.
22	WRITE DATA	WD	FM or MFM clock and data pulses are placed on this line to be written on the diskette.
23	TRACK 00	$\overline{TR00}$	This active low input informs the WD1770 that the drive's R/$\overline{W}$ heads are positioned over Track zero (internal pull-up).
24	INDEX PULSE	$\overline{IP}$	This active low input informs the WD1770 when the physical index hole has been encountered on the diskette (internal pull-up).
25	WRITE PROTECT	$\overline{WPRT}$	This input is sampled whenever a Write Command is received. A logic low on this line will prevent any Write Command from executing (internal pull-up).
26	DOUBLE DENSITY ENABLE	$\overline{DDEN}$	This input pin selects either single (FM) or double (MFM) density. When $\overline{DDEN}$ = 0, double density is selected (internal pull-up).

1070

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
27	DATA REQUEST	DRQ	This active high output indicates that the Data Register is full (on a Read) or empty (on a Write operation).
28	INTERRUPT REQUEST	INTRQ	This active high output is set at the completion of any command or reset a read of the Status Register.



WD1770 SYSTEM BLOCK DIAGRAM

ARCHITECTURE

The Floppy Disk Formatter block diagram is illustrated on page 4. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register — This 8-bit register assembles serial data from the Read Data input ($\overline{RD}$) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register — This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations, the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations, information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command, the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register — This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when the device is busy.

Sector Register (SR) — This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

Command Register (CR) — This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read onto the DAL.

Status Register (STR) — This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register can be read onto the DAL, but not loaded from the DAL.

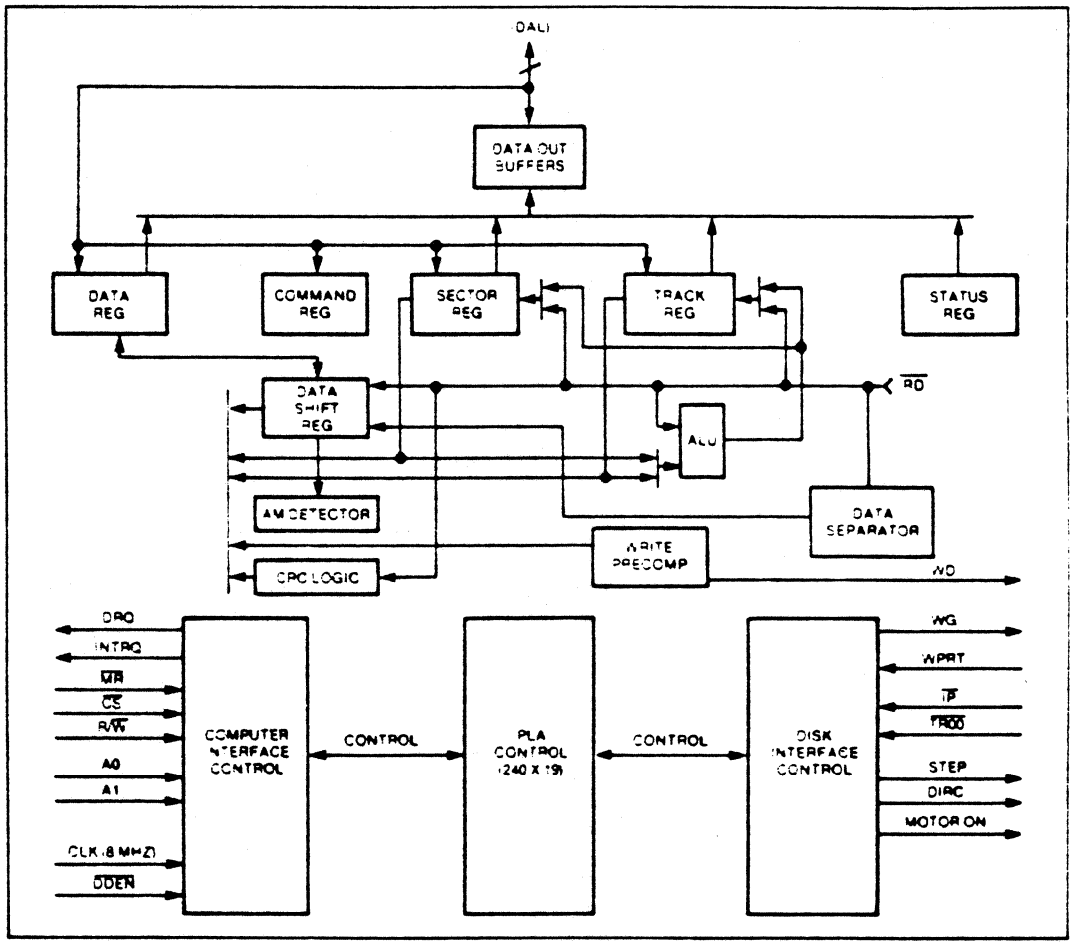
CRC Logic — This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is:

$G(x) = x^{16} + x^{12} + x^5 + 1$.

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) — The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

1071



WD1770 BLOCK DIAGRAM

Timing and Control — All computer and Floppy Disk interface controls are generated through this logic. The internal device timing is generated from an external crystal clock. The FD1770 has two different modes of operation according to the state of $\overline{DDEN}$. When $\overline{DDEN}$ = 0, double density (MFM) is enabled. When $\overline{DDEN}$ = 1, single density is enabled.

AM Detector — The address mark detector detects ID, data and index address marks during read and write operations.

Data Separator — A digital data separator consisting of a ring shift register and data window detection logic provides read data and a recovery clock to the AM detector.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the WD1770. The DAL are three state buffers that are enabled as output drivers when Chip Select ($\overline{CS}$) and R/$\overline{W}$ = 1 are active or act as input receivers when $\overline{CS}$ and R/$\overline{W}$ = 0 are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and $\overline{CS}$ is made low. The address bits A1 and A0, combined with the signal R/$\overline{W}$ during a Read operation or Write operation are interpreted as selecting the following registers:

1072

A1	A0	READ (R/$\overline{W}$ = 1)	WRITE (R/$\overline{W}$ = 0)
0	0	Status Register	Command Register
0	1	Track Register	Track Register
1	0	Sector Register	Sector Register
1	1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the WD1770 and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operations continues until the end of sector is reached.

On Disk Write operations the Data Request is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data is set in the Status Register.

At the completion of every command an INTRQ is generated. INTRQ is reset by either reading the status register or by loading the command register with a new command. In addition, INTRQ is generated if a Force Interrupt command condition is met.

The WD1770 has two modes of operation according to the state $\overline{DDEN}$ (Pin 26). When $\overline{DDEN}$ = 1, single density is selected. In either case, the CLK input (Pin 18) is at 8 MHZ.

GENERAL DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, $\overline{DDEN}$ should be placed to logical "1." For MFM formats, $\overline{DDEN}$ should be placed to a logical "0." Sector lengths are determined at format time by the fourth byte in the "ID" field.

SECTOR LENGTH TABLE	
SECTOR LENGTH FIELD (HEX)	NUMBER OF BYTES IN SECTOR (DECIMAL)
00	128
01	256
02	512
03	1024

The number of sectors per tract as far as the WD1770 is concerned can be from 1 to 255 sectors. The number of tracks as far as the WD1770 is concerned is from 0 to 255 tracks.

GENERAL DISK WRITE OPERATION

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the device before the Write Gate signal can be activated.

Writing is inhibited when the $\overline{Write\ Protect}$ input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set.

For Write operations, the WD1770 provides Write Gate (Pin 21) to enable a Write condition, and Write Data (Pin 22) which consists of a series of active high pulses. These pulses contain both Clock and Data information in FM and MFM. Write Data provides the unique missing clock patterns for recording Address Marks.

The Precomp Enable bit in Write commands allow automatic Write precompensation to take place. The outgoing Write Data stream is delayed or advanced from nominal by 125 nanoseconds according to the following table:

PATTERN				MFM	FM
X	1	1	0	Early	N/A
X	0	1	1	Late	N/A
0	0	0	1	Early	N/A
1	0	0	0	Late	N/A

Next Bit to be sent
Current Bit sending
Previous Bits sent

Precompensation is typically enabled on the innermost tracks where bit shifts usually occur and bit density is at its maximun.

COMMAND DESCRIPTION

The WD1770 will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in Table 1.

COMMAND SUMMARY

TYPE	COMMAND	BITS 7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step-in	0	1	0	u	h	V	r_1	r_0
I	Step-out	0	1	1	u	h	V	r_1	r_0
II	Read Sector	1	0	0	m	h	E	0	0
II	Write Sector	1	0	1	m	h	E	P	a_0
III	Read Address	1	1	0	0	h	E	0	0
III	Read Track	1	1	1	0	h	E	0	0
III	Write Track	1	1	1	1	h	E	P	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

FLAG SUMMARY

TYPE I COMMANDS

h = Motor On Flag (Bit 3)

h = 0, Enable Spin-Up Sequence
h = 1, Disable Spin-Up Sequence

V = Verify Flag (Bit 2)

V = 0, No Verify
V = 1, Verify on Destination Track

r_1, r_0 = Stepping Rate (Bits 1, 0)

r_1	r_0	WD1770	WD1772
0	0	6 ms	2 ms
0	1	12 ms	3 ms
1	0	20 ms	5 ms
1	1	30 ms	6 ms

u = Update Flag (Bit 4)

u = 0, No Update
u = 1, Update Track Register

TYPE II & III COMMANDS

m = Multiple Sector Flag (Bit 4)

m = 0, Single Sector
m = 1, Multiple Sector

a_0 = Data Address Mark (Bit 0)

a_0 = 0, Write Normal Data Mark
a_0 = 1, Write Deleted Data Mark

E = 30ms Settling Delay (Bit 2)

E = 0, No Delay
E = 1, Add 30ms Delay

P = Write Precompensation (Bit 1)

P = 0, Enable Write Precomp
P = 1, Disable Write Precomp

TYPE IV COMMANDS

I_3-I_0 Interrupt Condition (Bits 3-0)

I_0 = 1, Don't Care
I_1 = 1, Don't Care
I_2 = 1, Interrupt on Index Pulse
I_3 = 1, Immediate Interrupt
I_3-I_0 = 0, Terminate without Interrupt

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out commands. Each of the Type I Commands contains a rate field (r_0,r_1), which determines the stepping motor rate.

A 4μs (MFM) or 8 μs (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip will step the drive in the same direction it last stepped unless the command changes the direction.

The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 24μs before the first stepping pulse is generated.

After the last directional step an additional 30 milliseconds of head settling time takes place if the Verify flag is set in Type I commands. There is also a 30 ms head settling time if the E flag is set in any Type II or III command.

When a Seek, Step or Restore command is executed, an optional verification of Read/Write head position can be performed by setting bit 2 (V = 1) in the command word to a logic 1. The verification operation begins at the end of the 30 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status Bit 3), and the next encountered ID field is read from the disk for the verification operation.

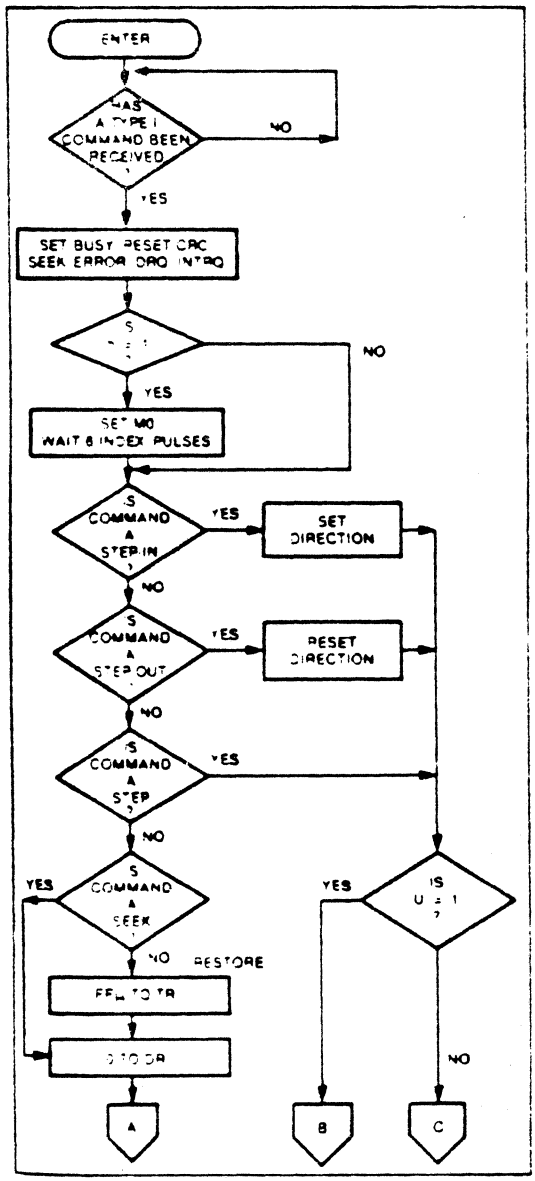
The WD1770 must find an ID field with correct track number and correct CRC within 5 revolutions of the media, otherwise the seek error is set and an INTRQ is generated. If V = 0, no verification is performed.

All commands, except the Force Interrupt command, may be programmed via the h Flag to delay for spindle motor start up time. If the h Flag is set and the Motor On line (Pin 20) is low when a command is received, the WD1770 will force Motor On to a logic 1 and wait 6 revolutions before executing the command. At 300 RPM, this guarantees a one second spindle start up time. If after finishing the command, the device remains idle for 10 revolutions, the Motor

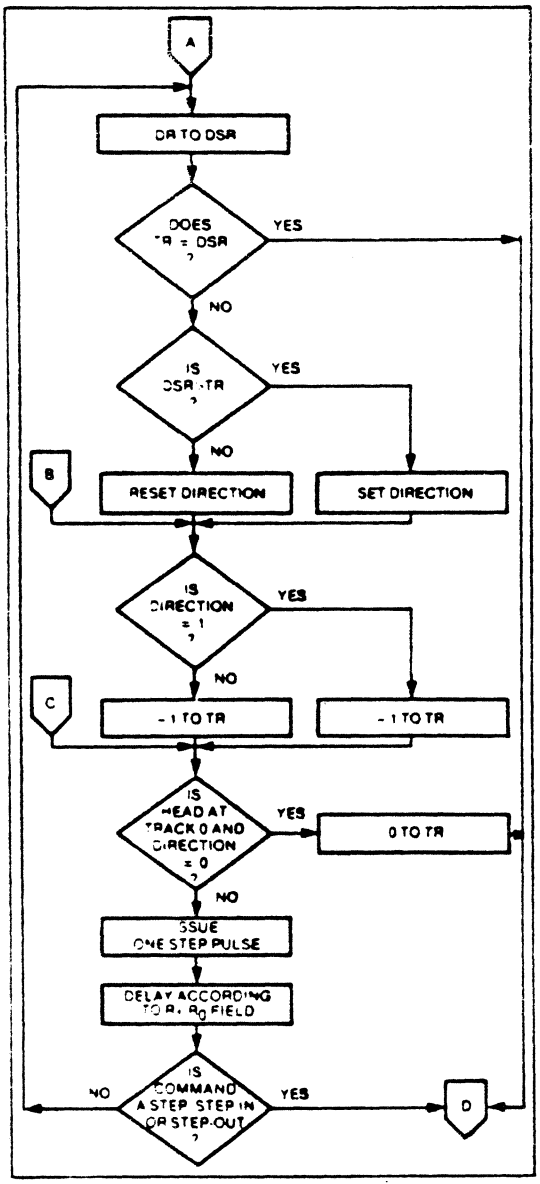
On line will go back to a logic 0. If a command is issued while Motor On is high, the command will execute immediately, defeating the 6 revolution start up. This feature allows consecutive Read or Write commands without waiting for motor start up each time; the WD1770 assumes the spindle motor is up to speed.

RESTORE (SEEK TRACK 0)

Upon receipt of this command, the Track 00 ($\overline{TR00}$) input is sampled. If $\overline{TR00}$ is active low indicating the Read/Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{TR00}$ is not active low, stepping pulses (Pin 16) at a rate specified by the r_1,r_0 field are issued until the $\overline{TR00}$ input is activated.



TYPE I COMMAND FLOW



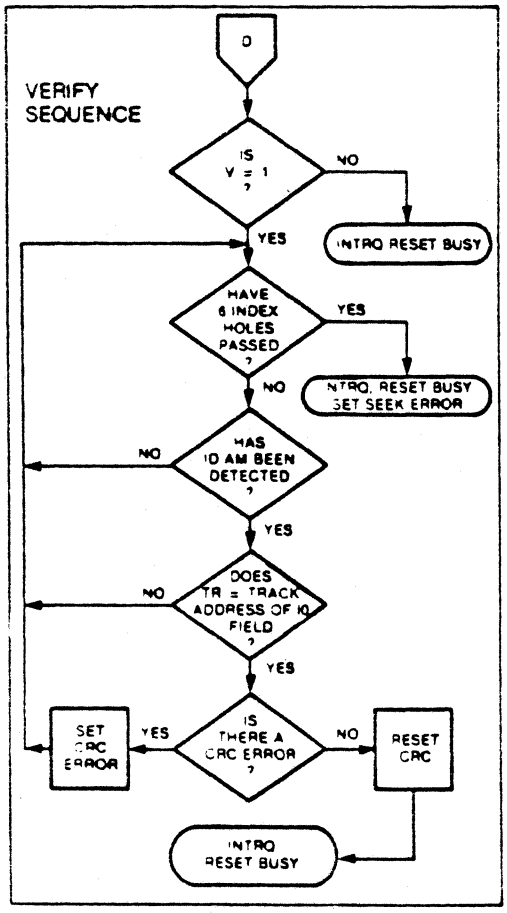
TYPE I COMMAND FLOW

1073

At this time, the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{TR00}$ input does not go active low after 255 stepping pulses, the WD1770 terminates operation, interrupts, and sets the Seek error status bit, providing the V flag is set. A verification operation also takes place if the V flag is set. The h bit allows the Motor On option at the start of command.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read/Write head and the Data Register contains the desired track number. The WD1770 will update the Track Register and issue stepping pulses in the appropriate direction until the contents of the Track Register are equal to the contents of the Data Register (the desired track location). A verification operation takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command. Note: When using multiple drives, the track register must be updated for the drive selected before seeks are issued.



TYPE I COMMAND FLOW

STEP

Upon receipt of this command, the WD1770 issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the $r_{1,r0}$ field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

STEP-IN

Upon receipt of this command, the WD1770 issues one stepping pulse in the direction towards track 76. If the U flag is on, the Track Register is incremented by one. After a delay determined by the $r_{1,r0}$ field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

STEP-OUT

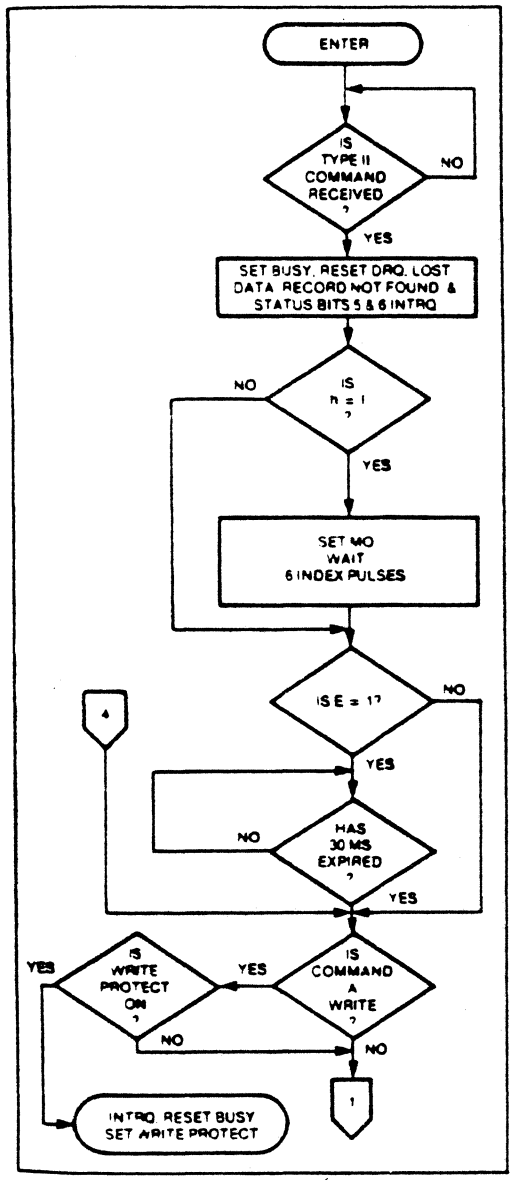
Upon receipt of this command, the WD1770 issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After delay determined by the $r_{1,r0}$ field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer must load the Sector Register with the desired sector number. Upon receipt of the Type II command, the busy status bit is set. If the E flag = 1 the command will execute after a 30 msec delay.

When an ID field is located on the disk, the WD1770 compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The WD1770 must find an ID field with a Track number, Sector number, and CRC within four revolutions of the disk, other-

1076



TYPE II COMMAND

wise, the Record not found status bit is set (Status Bit 4) and the command is terminated with an interrupt (INTRQ).

Each of the Type II Commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m = 0, a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The WD1770 will continue to read or write multiple records and update the sector register in numerical ascending sequence until the sector register exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the Command Register, which terminates the command and generates an interrupt.

For example: If the WD1770 is instructed to read sector 27 and there are only 26 on the track, the sector register exceeds the number available. The WD1770 will search for 5 disk revolutions, interrupt out, reset busy, and set the record not found status bit.

READ SECTOR

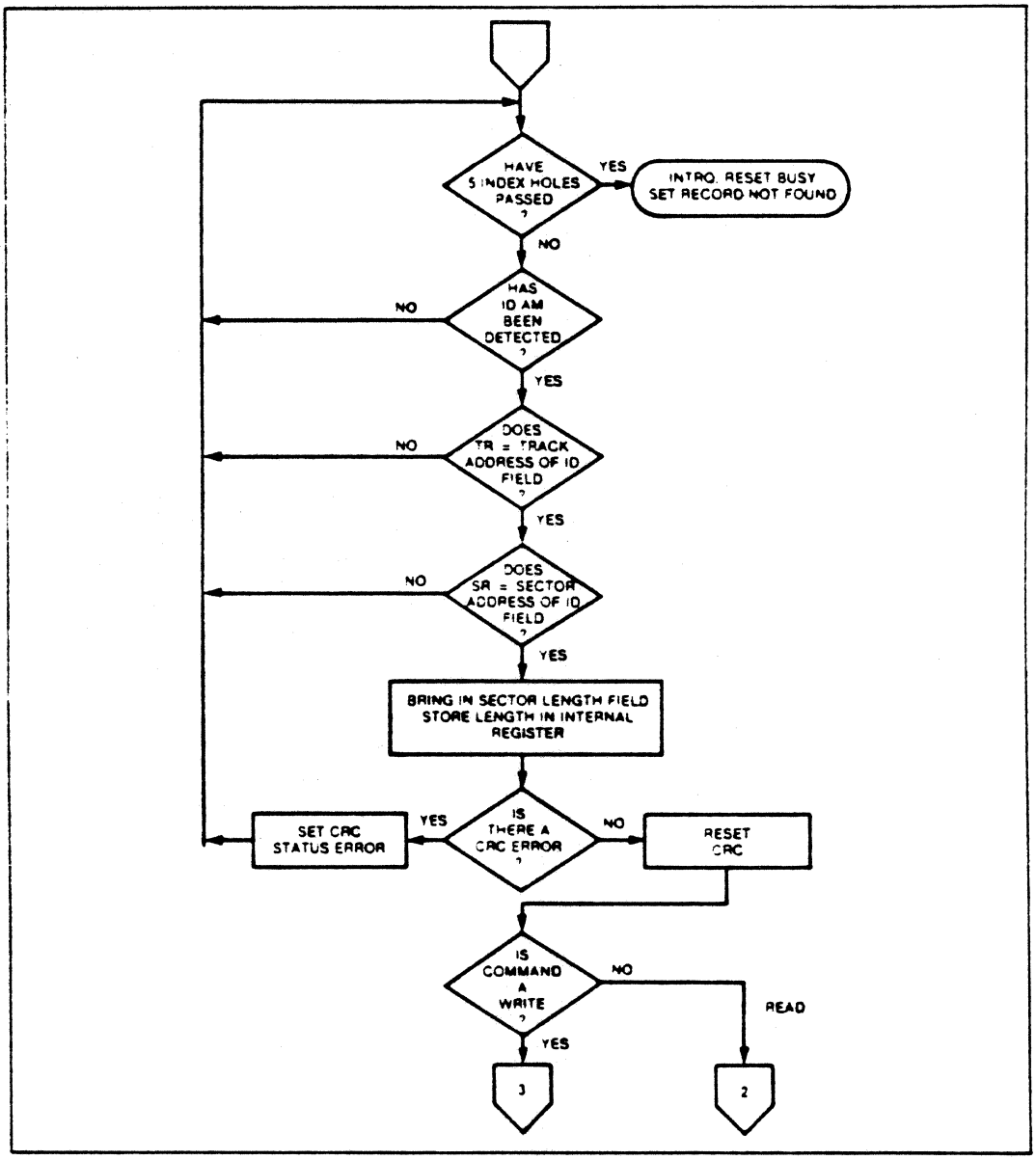
Upon receipt of the Read Sector command, the Busy status bit is set, and when a ID field is encountered that has the correct track number, correct sector number, and correct CRC, the data field is presented to the computer. The Data Address Mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the ID field is searched for and verified again followed by the Data Address Mark search. If after 5 revolutions the DAM cannot be found, the Record Not Found status bit is set and the operation is terminated. When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the computer has not read the previous contents of the DR before a new character is transferred that character is lost and the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown:

STATUS BIT 5	
1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector command, the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, and correct CRC, a DRQ is generated. The WD1770 counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated



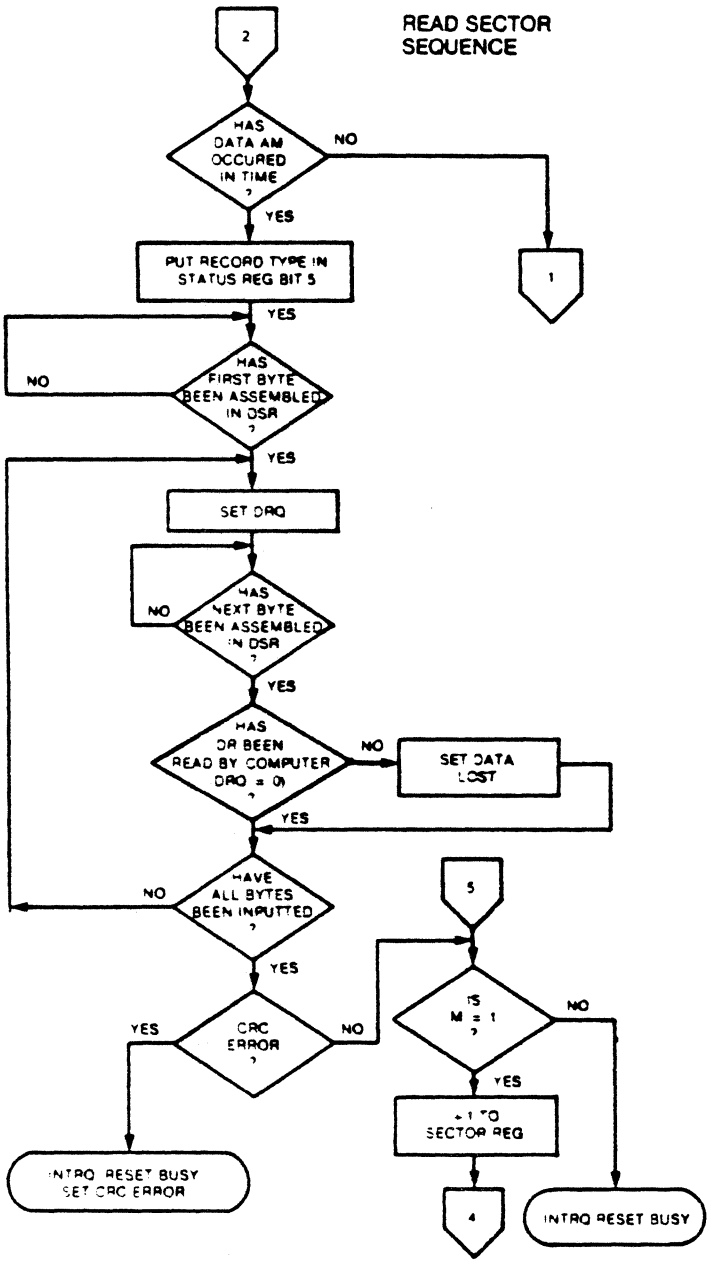
TYPE II COMMAND

and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeroes in single density and 12 bytes in double density are then written on the disk. At this time, the Data Address Mark is then written on the disk as determined by the a0 field of the command as shown below:

a0	DATA ADDRESS MARK (BIT 0)
1	Deleted Data Mark
0	Data Mark

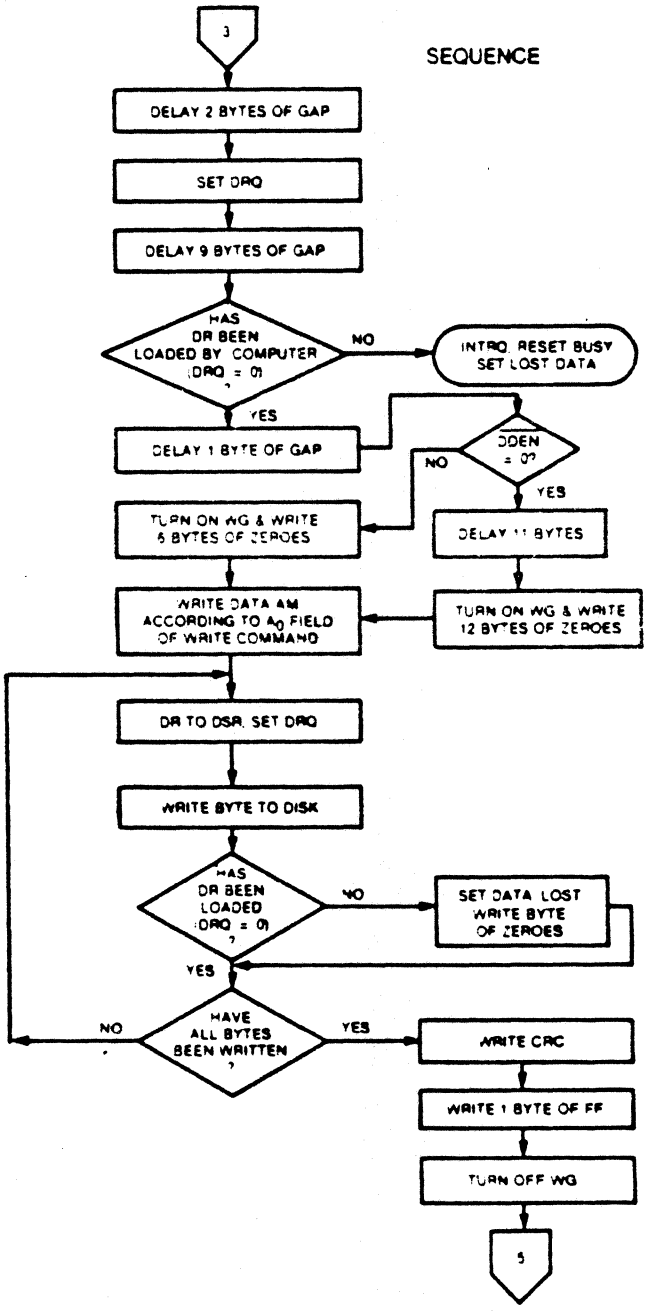
The WD1770 then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit

1078



TYPE II COMMAND

1078



TYPE II COMMAND

1080

is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated. INTRQ will set 24μsec (MFM) after the last CRC byte is written. For partial sector writing, the proper method is to write data and fill the balance with zeroes.

TYPE III COMMANDS

Read Address

Upon receipt of the Read Address command, the Busy Status Bit is set. The next encountered ID field is then read in from the disk, and six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

TRACK ADDR	SIDE NUMBER	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

Although the CRC characters are transferred to the computer, the WD1770 checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation an interrupt is generated and the Busy Status is reset.

Read Track

Upon receipt of the READ track command, the head is loaded and the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command.

This command has several characteristics which make it suitable for diagnostic purposes. They are: no CRC checking is performed; gap information is included in the data stream; and the address mark detector is on for the duration of the command. Because the AM detector is always on, write splices or noise may cause the chip to look for an AM.

The ID AM, ID field, ID CRC bytes, DAM, Data, and Data CRC Bytes for each sector will be correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.

WRITE TRACK FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA with a large amount of memory. Data and gap information must be provided at the computer interface. Formatting the disk is accomplished by positioning the R/W head over the desired track number and issuing the Write Track command.

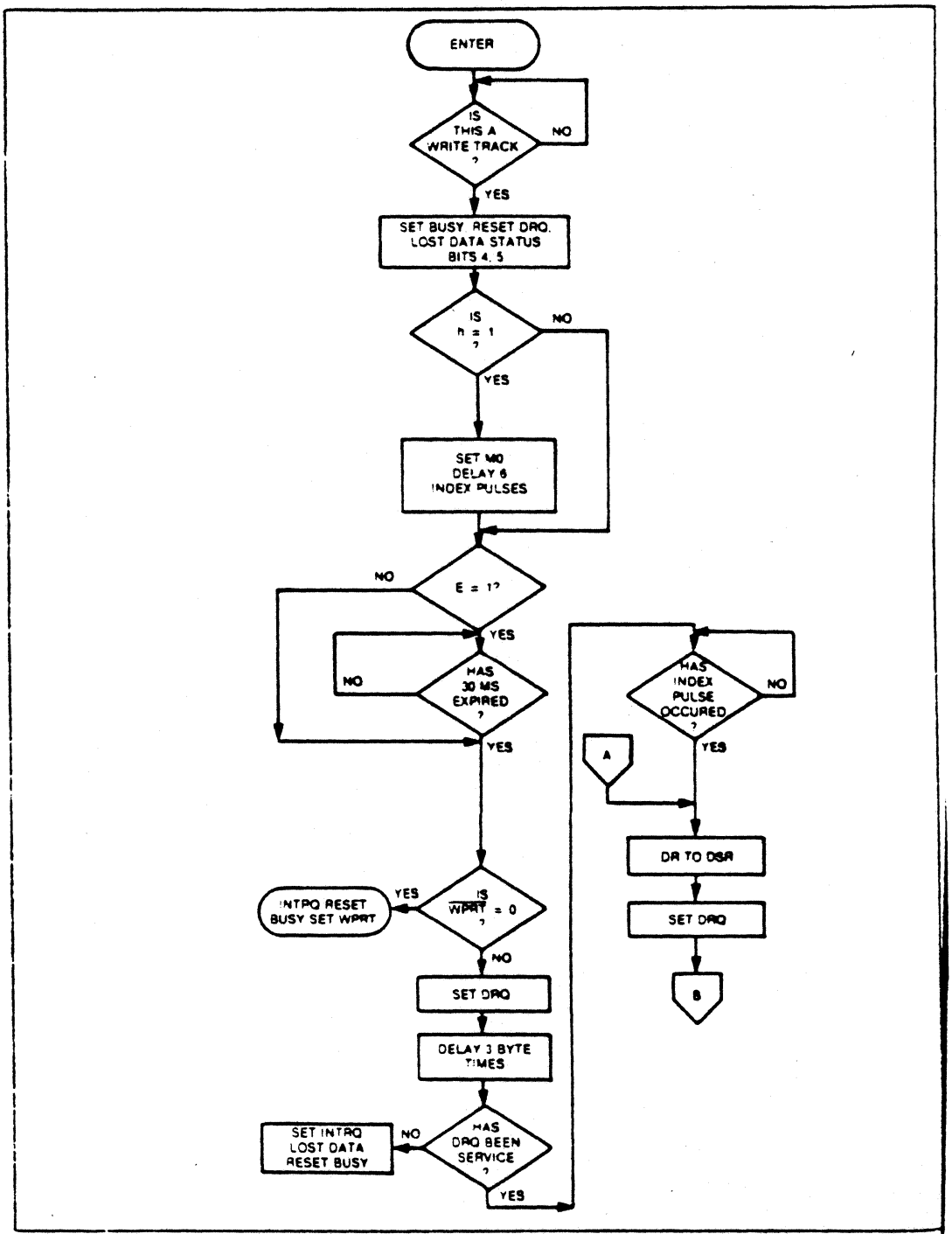
Upon receipt of the Write Track command, the Busy Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded within 3 byte times, the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the interrupt is activated. If a byte is not present in the DR when needed, a byte of zeroes is substituted.

This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the WD1770 detects a data pattern of F5 through FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation.

DATA PATTERN IN DR (HEX)	IN FM ($\overline{DDEN}$ = 1)	IN MFM ($\overline{DDEN}$ = 0)
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* In MFM, Present CRC
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, CLK = C7, Preset CRC	Write F8 thru FB, in MFM
FC	Write FC with CLK = D7	Write FC in MFM
FD	Write FD with CLK = FF	Write FD in MFM
FE	Write FE, CLK = C7, Preset CRC	Write FE in MFM
FF	Write FF with CLK = FF	Write FF in MFM

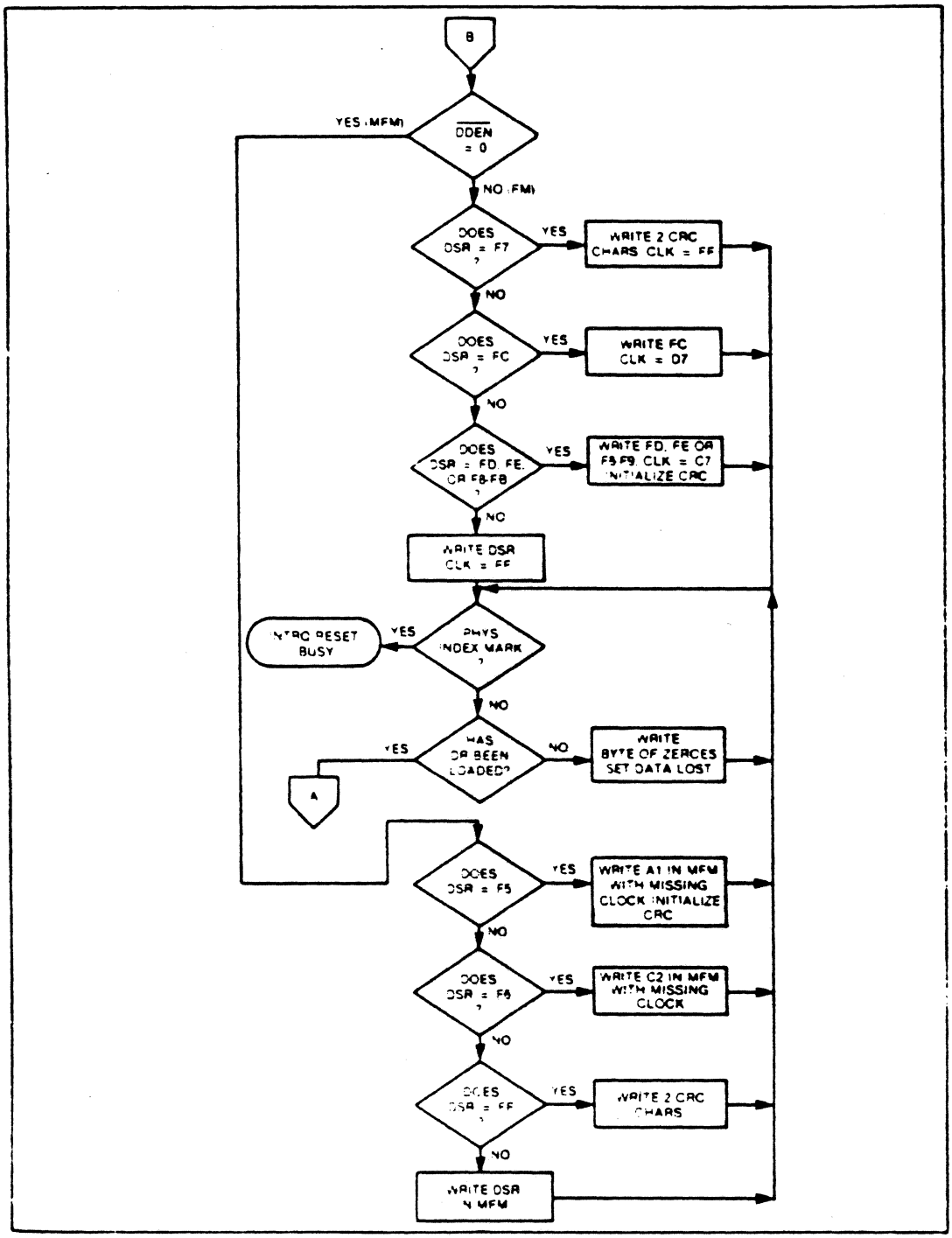
*Missing clock transition between bits 4 and 5.
**Missing clock transition between bits 3 and 4.

1081



TYPE III COMMAND WRITE TRACK

1082



TYPE III COMMAND WRITE TRACK

1083

The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 through FE must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

TYPE IV COMMANDS

The Forced Interrupt command is generally used to terminate a multiple sector read or write command or to insure Type I status in the status register. This command can be loaded into the command register at any time. If there is a current command under execution (busy status bit set) the command will be terminated and the busy status bit reset.

The lower four bits of the command determine the conditional interrupt as follows:

I_0 = Don't Care
I_1 = Don't Care
I_2 = Every Index Pulse
I_3 = Immediate Interrupt

The conditional interrupt is enabled when the corresponding bit positions of the command (I_3-I_0) are set to a 1. Then, when the condition for interrupt is met, the INTRQ line will go high signifying that the condition specified has occurred. If I_3-I_0 are all set to zero (HEX D0), no interrupt will occur but any command presently under execution will be immediately terminated. When using the immediate interrupt condition (I_3 = 1) an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the command register will not automatically clear the interrupt. The HEX D0 is the only command that will enable the immediate interrupt (HEX D8) to clear on a subsequent load command register or read status register operation. Follow a HEX D8 with D0 command.

Wait 16 micro sec (double density) or 32 micro sec (single density) before issuing a new command after issuing a forced interrupt. Loading a new command sooner than this will nullify the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates INTRQ when the specified condition is met. Forced interrupt will wait until ALU operations in progress are complete (CRC calculations, compares, etc.).

Status Register

Upon receipt of any command, except the Force Interrupt command, the Busy Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the Busy Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The user has the option of reading the status register through program control or using the DRQ line with DMA or interrupt methods. When the Data register is read the DRQ bit in the status register and the DRQ line are automatically reset. A write to the Data register also causes both DRQ's to reset.

The busy bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the INTRQ line. When using the INTRQ, a busy status check is not recommended because a read of the status register to determine the condition of busy will reset the INTRQ line.

The format of the Status Register is shown below:

(BITS)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

RECOMMENDED — 128 BYTES/SECTOR

Shown below is the recommended single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

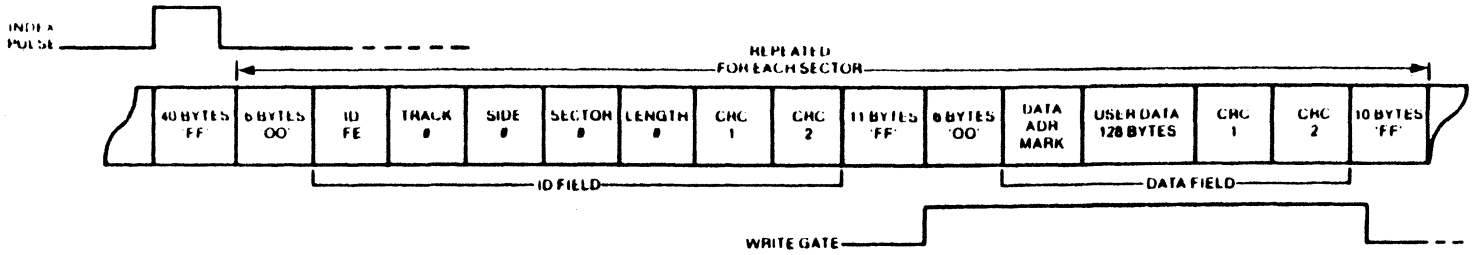
NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00 (Sector Length)
1	F7 (2 CRC's written)
11	FF (or 00)
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
10	FF (or 00)
369**	FF (or 00)

*Write bracketed field 16 times.
**Continue writing until WD1770 interrupts out. Approx. 369 bytes.

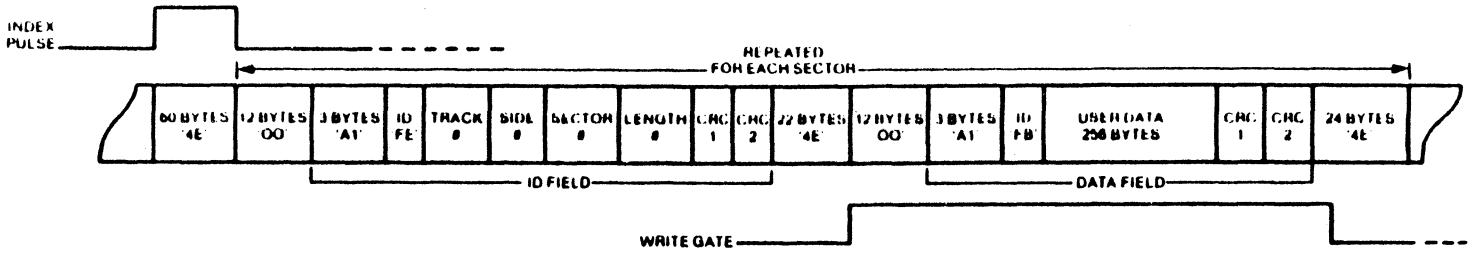
256 BYTES/SECTOR

Shown below is the recommended dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written, there is one data request.

1084



SINGLE DENSITY FORMAT



DOUBLE DENSITY FORMAT

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
60	4E
12	00
3	F5 (Writes A1)
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01 (Sector Length)
1	F7 (2 CRC's written)
22	4E
12	00
3	F5 (Writes A1)
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRC's written)
24	4E
668**	4E

*Write bracketed field 16 times.
**Continue writing until WD1770 interrupts out. Approx. 668 bytes.

1. Non-Standard Formats

Variations in the recommended formats are possible to a limited extent if the following requirements are met:

1) Sector size must be 128, 256, 512 of 1024 bytes.
2) Gap 2 cannot be varied from the recommended format.
3) 3 bytes of A1 must be used in MFM.

In addition, the Index Address Mark is not required for operation by the WD1770 Gap 1, 3, and 4 lengths can be as short as 2 bytes for WD1770 operation, however PLL lock up time, motor speed variation, write-splice area, etc. will add more bytes to each gap to achieve proper operation. It is recommended that the recommended format be used for highest system reliability.

	FM	MFM
Gap I	16 bytes FF	32 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
•	6 bytes 00	12 bytes 00
•		3 bytes A1
Gap III**	10 bytes FF	24 bytes 4E
	4 bytes 00	8 bytes 00
		3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.
**Byte counts are minimum, except exactly 3 bytes of A1 must be written.

STATUS REGISTER DESCRIPTION

BIT NAME	MEANING
S7 MOTOR ON	This bit reflects the status of the Motor On output.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Track: Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORD TYPE/SPIN-UP	When set, this bit indicates that the Motor Spin-Up sequence has completed (6 revolutions) on Type I commands. Type 2 & 3 commands, this bit indicates record Type. 0 = Data Mark. 1 = Deleted Data Mark.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA/ TRACK 00	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when update. On Type I commands, this bit reflects the status of the TRACK 00 Pin.
S1 DATA REQUEST/ INDEX	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit is reset to zero when updated. On Type I commands, this bit indicates the status of the Index Pin.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

DC ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

Storage Temperature ..... – 55°C to + 125°C
Operating Temperature ..... 0°C to 70°C Ambient

Maximum Voltage to Any Input with Respect to V_{SS} ..... (– 15 to – 0.3V)

1086

ATARI CORP.
1196 Borregas Avenue
Sunnyvale, CA 94086

DATE: May 14, 196

TO: Doug Renn

CC: Shiraz Shivji

FR: Tom Brightman

RE: Atari Monitor Summary Specifications

Please find attached list of our summary specifications for Atari monitors for the ST product line. Please review these specifications and make any necessary corrections.

Please also provide a drawing or specification for the SC1425 EST monitor connector pin out.

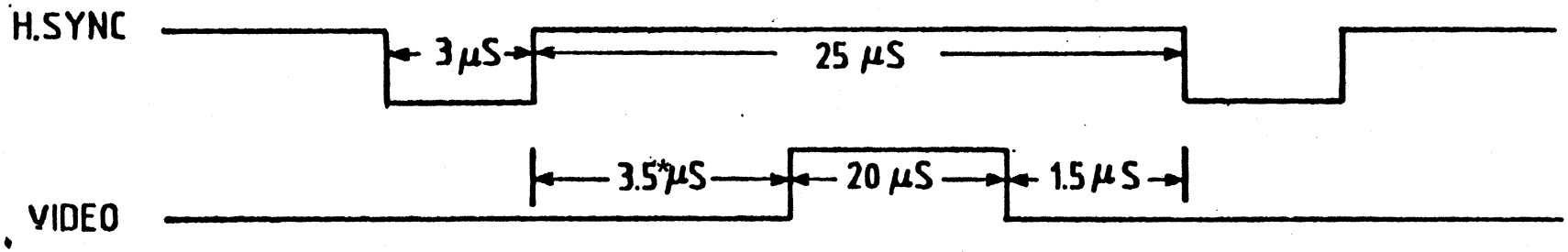
Thank you,

Tom Brightman

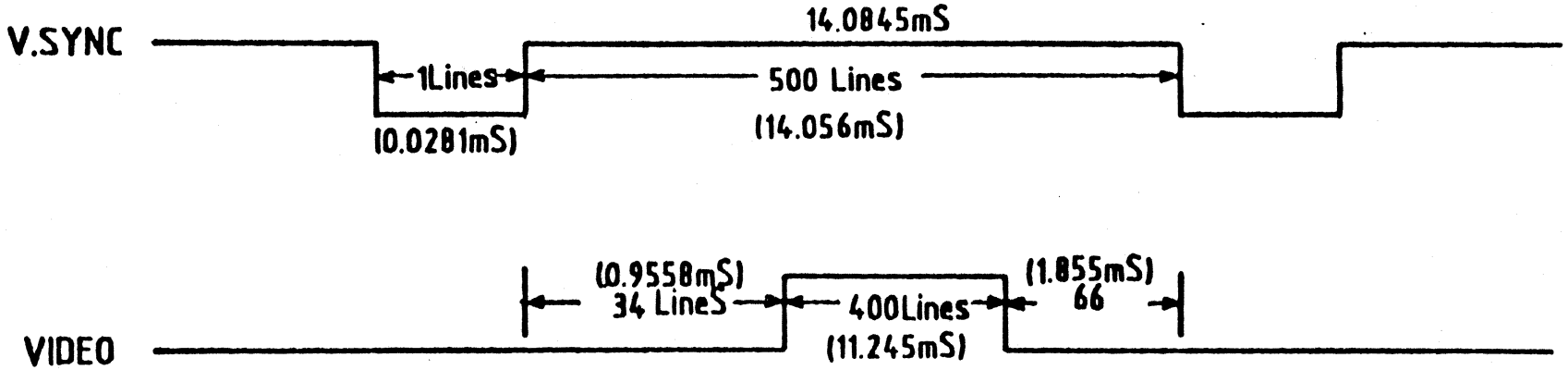
			SM 124	SC 1224	SC 1425
CRT	SIZE		12" (11" V)	12" (11" V	14" (13"V)
	Phosphor		-	Dot	Dot
	Dot Pitch		-	0.38mm	0.31mm
INTERFACE	Connector		13 pin male din	13 pin male din	TBD
	Audio		1 VPP/1.0K	1 VPP/1.0K	1 VPP/1.0K
	Video Input		Digital	RGB Analog	RGB Analog
	Input Impedance		75 /1.0VPP	75 /0-1.0VPP	75 /0-1.0VPP
	Dot Rate		32MHz	16MHz	32MHz
	HSYNC		TTL/3.3K /NEG	TTL/3.3K /NEG	TTL/3.3K /NEG
	VSYNC		TTL/3.3K /NEG	TTL/3.3K /NEG	TTL/3.3K /NEG
	Display Area		210mm x 131mm	210mm x 150mm	240mm x 180mm
	Resolution		640 x 400	640 x 200	640 x 480
DEFLECTION	Frequency	H	35.7KHz	15.75KHz	31.5KHz
		V	71.2KHz	60HZ	60Hz
	Retrace	H	4usec	10usec	6usec
		V	700usec	1msec	700usec
	Input Rise Time		10nsec	15nsec	10nsec
	Input Fall Time		10nsec	15nsec	10nsec
	Geo Distortion		1.5%	1.5%	1.5%
	Non-Linearity		9%	7%	7%
	Misconvergence (center)		N/A	0.6mm	0.4mm
	Misconvergence (corner)		N/A	1.0mm	0.6mm
USER CONTROLS	Power ON/OFF		Switch/Pot	Switch/Pot	Switch/Pot
	Audio Volume		Pot	Pot	Pot
	Brightness		Pot	Pot	Pot
	Contrast		Pot	Pot	Pot

NOTE: 1. On/Off & Volume may be combined.
2. On SM 124, Brightness/Contrast may be combined.
3. Audio frequency response of 100Hz - 15KHz 3db.

TIMING CHART

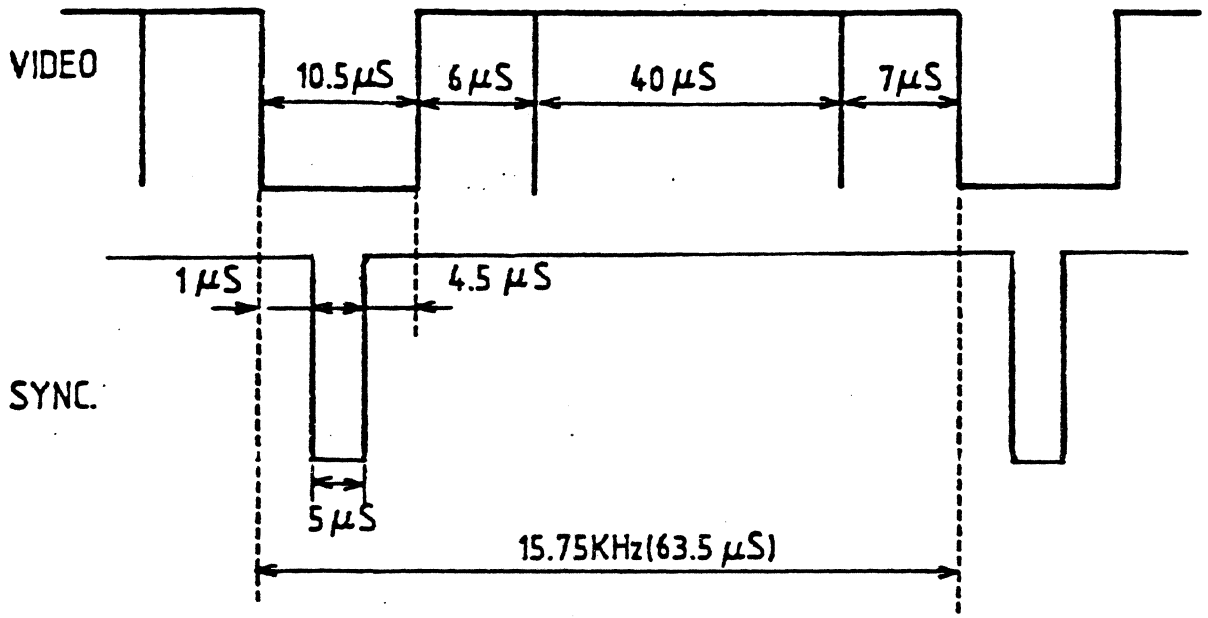


*NOMINAL
ACTUALLY 2.4µS - 4.5µS

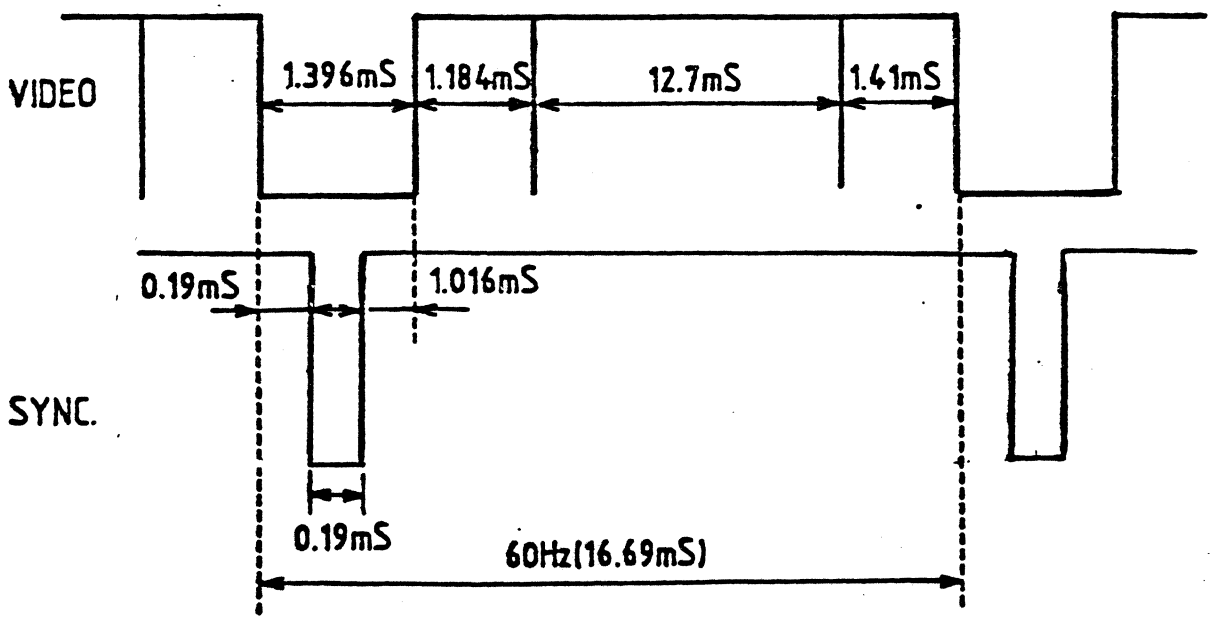


1087

10-1. HORIZONTAL TIMING



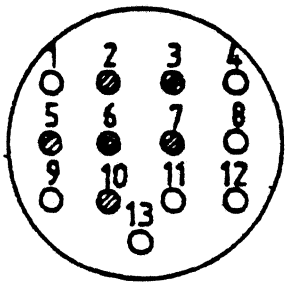
10-2. VERTICAL TIMING



SIGNAL DIN-CONNECTOR

PIN ASSIGNMENT

*13PIN DIN CONNECTOR
TO THE COMPUTER

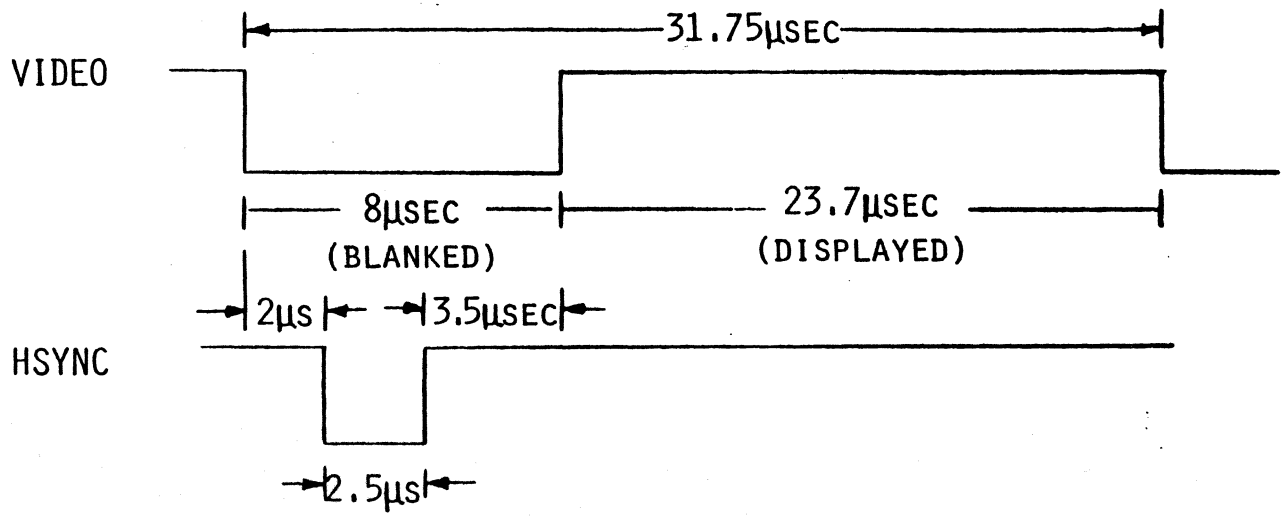


1 AUDIO
2 NOT CONNECTED
3 NOT CONNECTED
4 MONO/RGB*
5 NOT CONNECTED
6 GREEN
7 RED
8 12V PULLUP
9 H-SYNC
10 BLUE
11 VIDEO
12 V-SYNC
13 GND

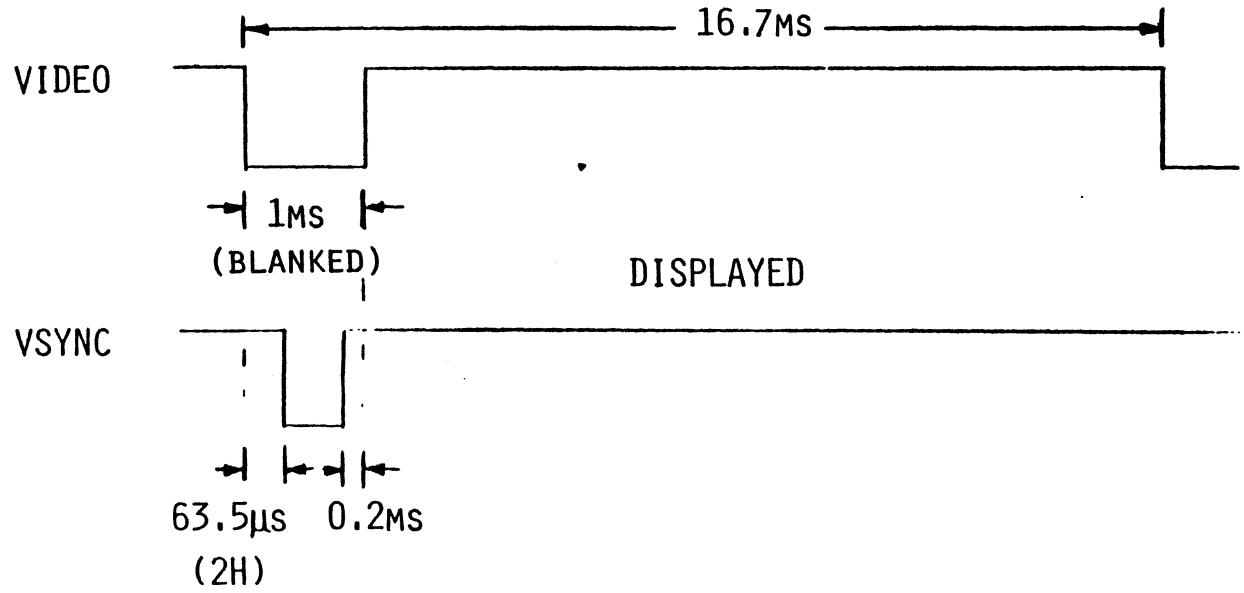
*NOTE: PIN 4 = "GROUND" FOR MONO/NOT CONNECTED FOR RGB.

	Dwg. No.	Rev.	Page

SC 1425



HORIZONTAL

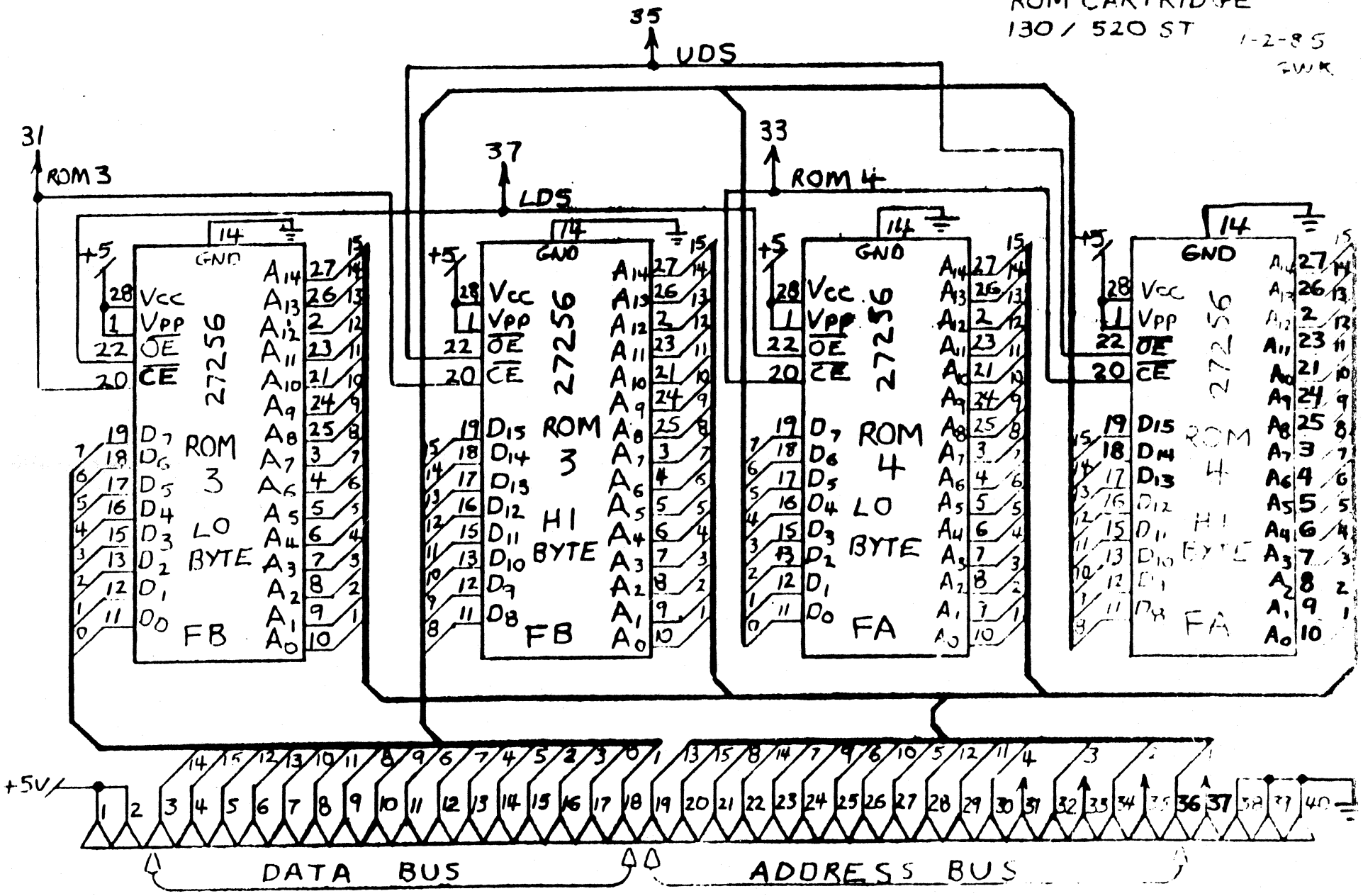


VERTICAL

1982

128 K BYTE
ROM CARTRIDGE
130 / 520 ST

1-2-85
EWK



1093