HA-8-3
Color Graphics
Board
Documentation

Appendices

AD7501/AD7503 data sheet	Appendix A
AD7574 data sheet	Appendix B
Am9511A data sheet	Appendix C
Am9512 data sheet	Appendix D
AY-3-8910 data sheet	Appendix E
TMS-9918A data sheet	Appendix F

FCC Interference Warning

This equipment is marketed pursuant to a waiver of FCC Rules Part 15 Subpart J. Operation of this computer in a residental area is likely to cause objectionable interference to radio and TV reception, because it emits more radio frequency energy than the FCC Rules allow. If interference occurs, the user will be required to take all steps necessary to correct the interference.

Table of Contents

1 Introduction 1

2 Specifications 2

3 Initial Setup 4

4 Interfacing Information 5

4.1 Joystick Interfacing Information 5
4.2 Composite Video Interface 6
4.3 Audio Interface 7

5 Distribution Diskette 8

5.1 Diagnostic Software 8
5.2 Colors 10
5.3 Color Bars 10
5.4 VDP Kaleidoscope 11

6 Adjustments 12

6.1 Adjusting Crystal Frequency 12
6.2 Adjusting Composite Video 12
6.3 Memory Options 13
6.4 Analog Multiplexer Options 13
6.5 9511A/9512 Options 13
6.6 Altering I/O Addresses 14
6.7 Board Enable 15
6.8 Interrupts 15

7 Programming the HA-8-3 16

7.1 TMS-9918A VDP 16
7.2 AY-3-8910 PSG 27
7.3 AD7574 A/D 41
7.4 9511A APU 41

8 Theory of Operation 48

8.1 H8 Bus Interface 48
8.2 Device Select 49
8.3 VDP Section 50
8.4 PSG Section 50
8.5 A/D Section 51
8.6 APU Section 51

9 Warranty and Service 52

1 Introduction

The HA-8-3 Color Graphics Board uses the TMS-9918A Color Video Display Generator to add color graphics to the H-8 computer. Also included is a Programmable Sound Generator for generating sound effects. An 8-input analog-to-digital converter allows for 4 X-Y joystick consoles (not included). Two 8-bit parallel I/O ports are also provided. As an option, the 9511A/9512 Arithmetic Processor Unit can be added.

The board comes pre-set for the recommended addresses and is supplied with sample user routines, diagnostic software, and demonstration programs.

2 Specifications

Power Requirements (+5,+12,-5 type 4116 RAMs)

+8 Volt Supply	650 ma (typ)	800 ma (max)
+18 Volt Supply	210 ma (typ)	250 ma (max)
-18 Volt Supply	20 ma (typ)	45 ma (max)

Additional Power Requirements for 9511A/9512

+8 Volt Supply	50 ma (typ)	95 ma (max)
+16 Volt Supply	50 ma (typ)	95 ma (max)

Temperature

Operating Temperature	0C (min)	40C (max)

Interfaces

H8 Bus Data Bus Drivers	74LS242
H8 Bus Other Inputs	1 LS TTL load or less
H8 Bus Open Collector Drivers	7407
Video Amplifier Output Impedance	75 Ohms
Video Output	Adjustable to NTSC composite video
Audio Output Impedance	> 1K Ohms
Audio Output	1 Volt peak-to-peak (max)
A/D Input Impedance	10 K (rec)
Parallel Inputs	Internal Pullups
Parallel Output	1 TTL Load

VDP Section

Video Display Processor	TMS-9918A
Crystal Frequency	10.738635Mhz
Crystal Type	Parallel (20pf)
RAM	16K X 8
Resolution	256 X 192 pixels
Colors	15
Text Mode	24 X 40 characters

PSG Section

Programmable Sound Generator	AY-3-8910
Tone Channels	3
Noise Channels	3
Envelope Generator	1
Parallel I/O Ports	2

A/D Section

Analog Multiplexer	AD7501 or AD7503
Analog Input Channels	8
A/D Converter	AD7574
A/D Resolution	8 bits, binary, unipolar
A/D Accuracy	+/-1 LSB
A/D Convert Time	20 us (max)
Analog Mux and Op-Amp Settling Time	35 us (max)

APU Section

APU (optional)	Am9511A or Am9512

3 Initial Setup

The HA-8-3 comes adjusted to be used with the GDZ-1320 color monitor. There are two cables to be connected. The cable with the male phono plug is the video cable. The end of this cable with the three pin connector plugs into the board at the COMP VIDEO connector. The other end of this cable connects to the video input of the GDZ-1320 color monitor. Many RF modulators and video tape recorders also use this type of composite video interface connector.

The cable with the miniature male phone plug is the audio cable. The end of this cable with the three pin connector plugs into the board at the AUDIO OUT connector. The other end of this cable connects to the audio input of the GDZ-1320 color monitor. Many video tape recorders also use this type of audio interface connector.

The 3 pin connectors are not polarized. It does not matter in which direction they are connected.

If the cables supplied with the HA-8-3 will not interface, refer to section 4.2 for information on how to interface to the COMP VIDEO connector, or refer to section 4.3 for information on how to interface to the AUDIO OUT connector.

The BIAS ADJ and GAIN ADJ trim pots should initially be at the center of rotation.

The HA-8-3 can be inserted into any backplane slot behind the CPU card (do not use the very last backplane slot).

Refer to section 5.1 and run the diagnostic program to check out the HA-8-3. If any adjustments are required, refer to section 6 for more information.

4 Interfacing Information

This section describes the interfacing requirement of the joystick connectors, composite video output, and audio output.

4.1 Joystick Interfacing Information

This section describes how to interface joysticks to the HA-8-3. Each joystick connects to a 10 pin male connector. Note that the fourth pin on each connector is cut. The shell can be polarized by installing a plug in the fourth opening of the 10 pin female shell.

The following is a detail table of the pins on the joystick connector:

	Joystick 1	Joystick 2	Joystick 3	Joystick 4
Pin 1	Vcc	Vcc	Vcc	Vcc
Pin 2	Gnd	Gnd	Gnd	Gnd
Pin 3	Vref	Vref	Vref	Vref
Pin 4	Cut	Cut	Cut	Cut
Pin 5	1	3	5	7 (Port B)
Pin 6	0	2	4	6 (Port B)
Pin 7	1	3	5	7 (Port A)
Pin 8	0	2	4	6 (Port A)
Pin 9	1	3	5	7 (Analog)
Pin 10	0	2	4	6 (Analog)

The interfacing information is broken into two parts: interfacing to the parallel ports, and interfacing to the analog ports.

4.1.1 Interfacing to the Parallel Ports

Note that two bits from each of the two parallel ports is brought out to each connector. If one port is enabled for input and one port enabled for output (see 7.2.1.6), each joystick connector will have 2 bits of parallel input for push button switches, etc., and two bits of parallel output to light LED's, etc. When one port is enabled for output and the other for input, it is recommended that port B be used for output and port A be used for input. It is possible to program both ports for input or both ports for output, if desired.

4.1.1.1 Parallel Input Interface

The PSG contains internal pullup resistors on each bit of the parallel ports. Therefore the recommended method of interfacing to an input port is to use a normally open push

button switch to ground a bit of the input port. Refer to the schematic (page 2) for a typical schematic of a push button switch interface connection. Ground is available at pin 2 of the connector.

4.1.1.2 Parallel Output Interface

The output drive of a parallel port is one TTL load. This is not enough to directly drive an LED. The parallel outputs must be buffered before they can be used to drive an LED. Vcc and Gnd are available on the connector at pins 1 and 2 respectively to power such a buffer. Refer to the schematic (page 2) for a typical schematic to drive an LED.

4.1.2 Interfacing to the Analog Inputs

The analog inputs must have voltages between Gnd and Vref. To do this a trim pot (linear trim) is used as a voltage divider. One end of the trim pot is connected to Vref and the other to Gnd. The wiper of the trim pot is connected to one of the analog channels. A 10K ohm trim pot is recommended for this application. Refer to the schematic (page 2) for a typical schematic of an analog input.

A joystick requires two trim pots - One for the X-axis, and one for the Y-axis. When a joystick is used, the following configuration is recommended:

1) The X-axis trim pot is connected to an even numbered analog channel.

2) The Y-axis trim pot is connected to an odd numbered analog channel.

3) The X-axis trim pot is wired so that as the joystick is moved to the right the wiper of the trim pot approaches Vref; and as the joystick is moved to the left the wiper of the trim pot approaches Gnd.

4) The Y-axis trim pot is wired so that as the joystick is moved up the wiper of the trim pot approaches Gnd; and as the joystick is moved down, the wiper of the trim pot approaches Vref.

4.2 Composite Video Interface

The 3 pin connector labeled COMP VIDEO is the composite video output of the HA-8-3. The output impedance of the video amplifier is 75 Ohms. The two outer pins on the COMP VIDEO

connector are both ground. The middle connector pin is the composite video output signal. Because the outer pins are both ground, the connector is not polarized.

If the cable supplied with the HA-8-3 does not match the video input connector of the display device, an adapter cable with a female phono socket on one end and the appropriate connector for the display device on the other end should be obtained.

4.3 Audio Interface

The 3 pin connector labeled AUDIO OUT is the audio output of the HA-8-3. The output is a 1 Volt peak-to-peak AC coupled signal. The two outer pins on the AUDIO OUT connector are both ground. The middle connector pin is the audio output signal. Because the outer pins are ground, the connector is not polarized.

If the cable supplied with the HA-8-3 does not match the audio input connector of the audio amplifier, an adapter cable with a female miniature phone socket on one end and the appropriate connector for the audio amplifier on the other end should be obtained.

5 Distribution Diskette

The distribution software for the HA-8-3 is on a 5.25" floppy diskette. In addition to the software support routines documented in section 4, this diskette has a diagnostic and some demonstration programs for the HA-8-3.

5.1 Diagnostic Software (HA83DIAG)

The diagnostic program is HA83DIAG. Both the source and the ABS file are included. HA83DIAG performs the following tests:

1) The first test performed by HA83DIAG is a memory test on the VRAM. This test requires about one minute. If no errors are detected, HA83DIAG prints the message "No VRAM failures detected.". If the test fails, then HA83DIAG lists the "U" numbers of the failing VRAMs.

2) After the memory diagnostic, HA83DIAG paints color bars on the color display. The order of the color bars is:

    1) White
    2) Gray
    3) Dark Blue
    4) Light Blue
    5) Cyan
    6) Dark Green
    7) Medium Green
    8) Light Green
    9) Dark Yellow
    10) Light Yellow
    11) Dark Red
    12) Medium Red
    13) Light Red
    14) Magenta

    The display should be examined for the proper color sequence.

3) After the color bars are displayed, HA83DIAG produces four sound effects. These sound effects are:

    A) The first sound effect is a laser. The sound is a frequency sweep effect on channel C.

    B) The second sound effect is a whistling bomb. The whistling effect is a frequency sweep effect on channel B. The explosion is noise effect under the control of the envelope generator. All three channels participate in the explosion effect.

C) The third sound effect is a wolf whistle. The whistling is produced on channel A. Channel B is used in this effect to produce white noise associated air movement through the lips during a whistle.

D) The fourth sound effect is a racing car changing gears. The effect of increasing engine RPM is produced on channel A, while a constant, high frequency engine whine effect is produced on channel B.

The sound effects should be listened to carefully for defects.

After the sound effects are produced, HA83DIAG will display 16 dots on the top of the display. The top row of dots are the bits from parallel port A, and the bottom row are the bits from parallel port B. The leftmost dot in each row is the high order bit. A white dot indicates the corresponding bit is ON, a black dot indicates the corresponding bit is OFF. Since the PSG has pullup resistors on the parallel ports, the dots should be displayed as white. The numbers 1 through 4 are also displayed. The position of these numbers is determined by the analog inputs on each joystick. The "X" position is determined by the voltage on the even numbered analog channel, and the "Y" position is determined by the voltage on the odd numbered analog channel. If no joysticks are available, the numbers can be moved randomly by touching the bottom two rows of the joystick connectors. The electrical noise produced by touching the connectors should be enough to randomly move the numbers.

To end the diagnostic, type two ^Z's.

5.2 Colors

There are 15 programs on the distribution diskette to paint the face of the color display a solid color. The following is a table of colors and programs:

Color	Program	Source
White	WHITE.ABS	WHITE.ASM
Black	BLACK.ABS	BLACK.ASM
Dark Blue	DBLUE.ABS	DBLUE.ASM
Light Blue	LBLUE.ABS	LBLUE.ASM
Cyan	CYAN.ABS	CYAN.ASM
Dark Green	DGREEN.ABS	DGREEN.ASM
Medium Green	MGREEN.ABS	MGREEN.ASM
Light Green	LGREEN.ABS	LGREEN.ASM
Dark Yellow	DYELLOW.ABS	DYELLOW.ASM
Light Yellow	LYELLOW.ABS	LYELLOW.ASM
Dark Red	DRED.ABS	DRED.ASM
Medium Red	MRED.ABS	MRED.ASM
Light Red	LRED.ABS	LRED.ASM
Magenta	MAGENTA.ABS	MAGENTA.ASM

To paint the face of the color display one of the above colors, execute the program whose name appears to the right of the color.

5.3 Color Bars

There are two programs on the distribution diskette to paint color bars on the color display. The program NTSCBARS paints color bars in the NTSC order. The program BARS paints color bars so that the color phase angle difference between colors is minimized.

5.3.1 NTSCBARS (NTSC Order Color Bars)

This program paints color bars in the NTSC order. The order of the color bars is given in the following table:

1) White
2) Gray
3) Dark Yellow
4) Light Yellow
5) Cyan
6) Dark Green
7) Medium Green
8) Light Green
9) Magenta
10) Dark Red
11) Medium Red
12) Light Red
13) Dark Blue
14) Light Blue

5.3.2 BARS (Phase Order Color Bars)

This program paints colors bars on the color display so that the phase angle difference between colors is minimized. The order of the color bars is given in the following table:

1) White
2) Gray
3) Dark Blue
4) Light Blue
5) Cyan
6) Dark Green
7) Medium Green
8) Light Green
9) Dark Yellow
10) Light Yellow
11) Dark Red
12) Medium Red
13) Light Red
14) Magenta

5.4 VDP Kaleidoscope

The program KALEIDO uses the color display to produce a kaleidoscope. To exit this program type a ^C.

6 Adjustments

Adjustments to the HA-8-3 are described in this section.

6.1 Adjusting the VDP crystal frequency

If the VDP crystal drifts off frequency, the HA-8-3 may no longer produce a color image, or the tint of the colors may not be correct. The frequency can be measured by attaching a frequency counter to U7 pin 3. The frequency on this pin should be the color burst frequency, i.e., 3.579545 Mhz +/- 15 Hz. If this frequency is off, adjust capacitor C6 until the frequency is correct. Since the capacitance of C6 will be increased by touching it with the adjusting instrument, it will be necessary to correct for this extra capacitance by adjusting the frequency about 50 Hz lower. When the adjusting instrument is removed, the frequency will increase about 50Hz because the extra capacitance of the adjusting instrument has been removed.

If a frequency counter is not available, a coarse adjustment can be made by adjusting C6 so that it is mid-range between the settings which produce acceptable colors. This method is not recommended.

6.2 Adjusting the Composite Video Output

The video output of the HA-8-3 has been designed to interface with the GDZ-1320 monitor. There are two adjustments for the composite video output. Each of the adjustments is described below:

6.2.1 Bias Adjustment

Trim pot R1 is labeled BIAS ADJ. This trim pot controls the DC component of the composite video signal. Most composite video inputs (including the GDZ-1320 color monitor) are 75 ohm AC coupled inputs. For these applications this trim pot should be adjusted to the center of rotation.

6.2.2 Gain Adjust

Trim pot R3 is labeled GAIN ADJ. For most applications (including the GDZ-1320 monitor) a setting at the center of rotation will produce the best results. Fine adjustments should be made for the best overall picture.

6.3 Memory Options

The HA-8-3 comes with 150 ns 4116 type RAM's. These RAM's come in a 5 volt only part or a +5, +12, -5 volt part. There are jumper options on the board to support either type of part. All parts on a board must be of the same type, however.

6.3.1 5 Volt Only RAM's

For this type of RAM jumpers J1, J2, and J3 must be removed, and jumper J4 installed. No other changes are required. This type of RAM greatly reduces the 12 volt requirements of the HA-8-3.

6.3.2 +5, +12, -5 Volt RAM's

For this type of RAM jumpers J1, J2, and J3 must be installed, and jumper J4 removed. The HA-8-3 is supplied with this type of part.

6.4 Analog Multiplexer Options

The HA-8-3 supports two types of analog multiplexers: the AD7501 or the AD7503. If the AD7501 is installed at U17 then the jumper labeled 7501 must be installed, and the jumper labeled 7503 must be removed. If the AD7503 is installed at U17 then the jumper labeled 7503 must be installed and the jumper labeled 7501 must be removed.

There is no functional difference between the AD7501 and the AD7503.

6.5 9511A / 9512 option

The HA-8-3 supports either the 9511A or the 9512 APU. Either device may be installed at U21. If interrupts are to be used with the APU then a jumper will have to be installed at 9511 or 9512.

If the 9511A is installed at U21 and interrupts are used, then the jumper labeled 9511 must be installed and the jumper labeled 9512 must be removed. If the 9512 is installed at U21 and interrupts are used, then the jumper labeled 9512 must be installed and the jumper labeled 9511 must be removed.

If interrupts are not being used (recommended), then neither jumper should be installed.

6.6 Altering I/O Addresses

A facility for changing the I/O addresses of the devices on the HA-8-3 is provided for. However, it is not recommended because it involves cutting traces and installing jumpers. The HA-8-3 will also not be usable with the software supplied with it.

Observe that there are three device select grids labeled DXX, XDX, and XXD. There is one select grid for each digit of the octal I/O address. Inside of each device select grid there are four device designators labeled A/D, APU, PSG, and VDP. To the left of each device designator there is a trace marked with an "X". The traces marked with an "X" set the default I/O addresses.

To change the I/O address of a device follow these steps:

1) In each of the three device select grids, cut one trace just to the left of the device designator being changed. This trace is clearly marked with an "X". IF YOU ARE NOT SURE DO NOT CUT THE TRACE! There are three traces to be cut for each device.

2) In each of the three device select grids, install a jumper from the device designator to the octal digit for the new I/O address. Note that the device designator pads have two holes. This will permit daisy chaining if more than one device requires the same octal digit. All I/O addresses must be even.

Note that in the device select grid XXD there is a hole labeled OFF above the digits. A device can be turned off by jumpering a device designator to OFF instead of to a digit.

To change the APU address to 324Q, the following would be done:

1) In the device select grid labeled DXX the trace marked with an "X" to left of the device designator APU must be cut.

2) In device select grid DXX a jumper between 3 and the device designator APU must be installed. The high order digit of the I/O address of the APU has now been changed to 3.

3) In the device select grid labeled XDX the trace marked with an "X" to the left of the device designator APU must be cut.

4) In device select grid XDX a jumper between 2 and the device designator APU must be installed. The middle digit of the I/O address of the APU has now been changed to a 2.

5) In this case, no changes have to be made to device select grid XXD because the default trace is already installed to 4.

6.7 Board Enable

A facility for disabling the HA-8-3 entirely is available by cutting a trace and installing a jumper. To disable the HA-8-3 cut the trace between BDEN and ON, then install a jumper between BDEN and OFF.

6.8 Interrupts

The HA-8-3 provides for interrupts by the VDP and APU. Interrupts are enabled by installing jumpers in the grid labeled INT SEL. To enable an interrupt for the APU or VDP, install a jumper between the device designator and the desired interrupt number. For example, to enable interrupt number 5 for the VDP, install a jumper between VDP and INT5. The use of interrupts is not recommended.

7 Programming the HA-8-3

This section describes how to program the four major components of the HA-8-3 using the support software on the distribution diskette. The data sheet for each of the major components of the HA-8-3 are located in the rear of this manual. Before reading how the support software works for a component, it is advisable to read the data sheet for that component.

7.1 The TMS-9918A Video Display Processor (VDP)

The basic support software for the VDP is contained in two files on the distribution diskette: VDPDEF.ACM and VDPIO.ACM. The file VDPDEF.ACM contains the definitions of the bits in the registers of the VDP. The file VDPIO.ACM contains the I/O routines to maintain the VDP's registers.

There are three other files on the distribution diskette which contain software aids for programming the VDP. The file MCXYSUBS.ACM contains software support for using the multicolor mode to produce "X-Y" plots. The file SPRITE.ACM contains software routines for sprite handling. The file VDPMOVES.ACM contains routines to move blocks of data to and from VRAM.

Unless otherwise noted, values requiring 8 bits or less are passed in register A. Values requiring more than 8 bits are passed in register HL. All registers are preserved except for those used to return data. Since the VDP has 16K of VRAM, only the lower 14 bits are used when passing VRAM addresses.

7.1.1 Primitive VDP I/O Routines

The file VDPIO.ACM contains the primitive functions for doing I/O on the VDP. Each routine is described below:

7.1.1.1 VP.SRA (Set VDP VRAM Read Address)

This is the primitive routine used to set the VDP's VRAM Read Address Register. The 14-bit VRAM read address is placed in register HL, and then VP.SRA is called. Subsequent calls to VP.RDV (Read VRAM) will start at the address passed in register HL.

To set the VRAM Read Address Register to location 8190, the following call to VP.SRA could be used:

```
LXI     HL,8190
CALL    VP.SRA
```

7.1.1.2 VP.RDV (Read VRAM)

This routine reads data from the VRAM at the location in the Read Address Register (see 7.1.1.1 VP.SRA to set the Read Address Register). The Read Address Register is autoincremented by the read. The VRAM data is returned in register A.

The following code could be used to place the contents of VRAM location 2000 into register B and the contents of VRAM location 2001 into register C:

```
LXI     HL,2000
CALL    VP.SRA
CALL    VP.RDV
MOV     B,A
CALL    VP.RDV
MOV     C,A
```

7.1.1.3 VP.RVD (Read VRAM Direct)

This routine is a combination of VP.SRA and VP.RDV. The 14-bit VRAM read address is placed in register HL and the data from VRAM location HL is returned in register A. The Read Address Register is autoincremented.

The following code can also be used to place the contents of VRAM location 2000 in register B, and the contents of location 2001 in register C:

```
LXI     HL,2000
CALL    VP.RVD
MOV     B,A
CALL    VP.RDV
MOV     C,A
```

7.1.1.4 VP.SWA (Set VDP VRAM Write Address)

This routine sets the VDP VRAM Write Address Register. The 14-bit read address is passed in register HL. Subsequent calls to VP.WRV (Write VRAM) will start at the address in register HL.

To set the VRAM Write Address Register to location 8190, the following call to VP.SWA could be used:

```
LXI     HL,8190
CALL    VP.SWA
```

7.1.1.5 VP.WRV (Write VRAM)

This routine writes the data in register A to the VRAM location in the Write Address Register (see 7.1.1.4 VP.SWA to set the Write Address Register). The Write Address Register is autoincremented by the write.

The following code could be used to write the contents of register B to VRAM location 2000 and the contents of register C to VRAM location 2001:

```
LXI     HL,2000
CALL    VP.SWA
MOV     A,B
CALL    VP.WRV
MOV     A,C
CALL    VP.WRV
```

7.1.1.6 VP.WVD (Write VRAM Direct)

This routine is a combination of VP.SWA and VP.WRV. The VRAM address to be written is placed in register HL and the data to be written is placed in register A. The Write Address Register is autoincremented by this operation.

The following code can also be used to write the contents of register B to VRAM location 2000 and the contents of register C to VRAM location 2001:

```
LXI     HL,2000
MOV     A,B
CALL    VP.WVD
MOV     A,C
CALL    VP.WRV
```

7.1.1.7 VP.WRR (Write VDP Register)

This routine is used to write the write only VDP registers. The VDP register number to be written is passed in register L, and the value to be written is passed in register A. This is an internal routine and should not be used. The routines described in 7.1.2 should be used to maintain the VDP registers.

7.1.2 VDP Register Maintenance Routines

The file VDPIO.ACM contains the software routines to maintain the nine VDP registers, and routines to read and write the 16K of VDP RAM (VRAM). These routines use the register pair

HL to pass values requiring more than 8 bits and register A to pass 8 bit values.

The file VDPDEF.ACM contains the bit names for the bits in the registers. Where applicable, the bit name from VDPDEF.ACM is given. In order to be consistent with the data sheet for the VDP, the description of the registers uses bit 7 for a low order bit, and bit 0 for a high order bit. Each of the VDP registers, along with the software support for the register, is described below:

7.1.2.1 Registers 0 and 1 (Option Control)

Registers 0 and 1 contain the option bits used to enable and disable various VDP features and modes. A description of each of the option bits follows:

Register 0:

BIT 7	(VP.NEV) (VP.EV)	OFF: Disable external video ON: Enable external video (not supported on HA-8-3)

Register 1:

BIT 7	(VP.M0) (VP.M1)	OFF: Magnification 0 sprites ON: Magnification 1 sprites
BIT 6	(VP.S0) (VP.S1)	OFF: Size 0 sprites ON: Size 1 sprites
BIT 4	(VP.MCM)	OFF: Multicolor mode not selected ON: Enable multicolor mode
BIT 3	(VP.TM)	OFF: Text mode not selected ON: Enable text mode
BIT 2	(VP.DI) (VP.EI)	OFF: Disable interrupts ON: Enable interrupts
BIT 1	(VP.DDP) (VP.EDP)	OFF: Disable (blank out) display ON: Enable display
BIT 0	(VP.4K) (VP.16K)	OFF: 4K dynamic VRAMs (do not use this) ON: 16K dynamic VRAMs (use this)

Note that it is not legal to select both the multicolor mode and the text mode together (VP.MCM+VP.TM). When neither the multicolor mode nor text mode is selected, the pattern mode (VP.PM) is selected.

To load the option control registers, place the 16-bit

option mask in register HL and call the routine VP.SOP. For example, to select the multicolor mode with size 0, magnification 0 sprites with the display active, the following code could be used:

```
LXI     HL,VP.NEV+VP.16K+VP.EDP+VP.DI+VP.MCM+VP.S0+VP.M0
CALL    VP.SOP
```

The VDP register maintenance routine VP.SOP saves the most recently set options in the variable VP.OPT.

7.1.2.2 Register 2 (Pattern Name Table Base Address)

The register maintenance routine for the Pattern Name Table Register is VP.SPN. The 14-bit VRAM address of the Pattern Name Table is placed in register HL. The Pattern Name Table address is stored in the variable VP.PNT. (VP.PNT should contain the most recent Pattern Name Table address.) The upper 4 bits of the 14-bit Pattern Name Table Address are then stored in the Pattern Name Table Register of the VDP. Since the VDP does not store the low order 10 bits of the Pattern Name Table address, this address must be evenly divisible by 1024 (the bottom ten bits must all be 0's). VP.SPN does not verify this.

To set the VRAM address of the Pattern Name Table to 2048, the following calling sequence could be used:

```
LXI     HL,2048
CALL    VP.SPN
```

After this call, the variable VP.PNT will contain the value 2048, and the value 2 (2048/1024) has been stored in the VDP's Pattern Name Table Register.

7.1.2.3 Register 3 (Color Generator Table Base Address)

The register maintenance routine for the Color Generator Table Register is VP.SCG. The 14-bit VRAM address of the Color Generator Table is placed in register HL. The Color Generator Table Address is stored in the variable VP.CGT. (VP.PNT should contain the most recent Color Generator Table address.) The upper 8 bits of the 14-bit Color Generator Table address are then stored in the Color Generator Table Address Register of the VDP. Since the VDP does not store the low order 6 bits of the Color Generator Table, this address must be evenly divisible by 64 (the bottom 6 bits must all be 0's). VP.SCG does not verify this.

To set the VRAM address of the Color Generator Table to be 256, the following calling sequence could be used:

```
LXI     HL,256
CALL    VP.SCG
```

After this call, the variable VP.CGT will contain the value 256, and the value 4 (256/64) has been stored in the VDP's Color Generator Table Register.

7.1.2.4 Register 4 (Pattern Generator Table Base Address)

The register maintenance routine for the Pattern Generator Table Register is VP.SPG. The 14-bit VRAM address of the Pattern Generator Table is placed in register HL. The Pattern Generator Table address is stored in the variable VP.PGT. (VP.PGT should contain the most recent Pattern Generator Table address.) The upper 3 bits of the 14-bit Pattern Generator Table Address are then stored in the Pattern Generator Table Register of the VDP. Since the VDP does not store the low order 11 bits of the Pattern Generator Table address, this address must be evenly divisible by 2048 (the bottom 11 bits must all be 0's). VP.SPG does not verify this.

To set the VRAM address of the Pattern Generator Table to be 6144, the following calling sequence could be used:

```
LXI     HL,6144
CALL    VP.SPG
```

After this call, the variable VP.PGT will contain the value 6144, and the value 3 (6144/2048) has been stored in the VDP's Pattern Generator Table Register.

7.1.2.5 Register 5 (Sprite Name Table Base Address)

The register maintenance routine for the Sprite Name Table Register is VP.SSN. The 14-bit VRAM address of the Sprite Name Table is placed in register HL. The Sprite Name Table address is stored in the variable VP.SNT. (VP.SNT should contain the most recent Sprite Name Table address.) The upper 7 bits of the 14-bit Sprite Name Table Address are then stored in the Sprite Name Table Register of the VDP. Since the VDP does not store the low order 7 bits of the Sprite Name Table address, this address must be evenly divisible by 128 (the bottom 7 bits must all be 0's). VP.SSN does not verify this.

To set the VRAM address of the Sprite Name Table to be 128, the following calling sequence could be used:

```
LXI     HL,128
CALL    VP.SSN
```

After this call, the variable VP.SNT will contain the value 128, and the value 1 (128/128) has been stored in the VDP's Sprite Name Table Register.

7.1.2.6 Register 6 (Sprite Pattern Generator Table Base Address)

The register maintenance routine for the Sprite Pattern Generator Table Register is VP.SSG. The 14-bit VRAM address of the Sprite Pattern Generator Table is placed in register HL. The Sprite Pattern Generator Table address is stored in the variable VP.SGT. (VP.SGT should contain the most recent Sprite Pattern Generator Table address.) The upper 3 bits of the 14-bit Sprite Pattern Generator Table Address are then stored in the Sprite Pattern Generator Table Register of the VDP. Since the VDP does not store the low order 11 bits of the Sprite Pattern Generator Table address, this address must be evenly divisible by 2048 (the bottom 11 bits must all be 0's). VP.SSG does not verify this.

To set the VRAM address of the Sprite Pattern Generator Table to be 4096, the following calling sequence could be used:

```
LXI     HL,4096
CALL    VP.SSG
```

After this call, the variable VP.SGT will contain the value 4096, and the value 2 (4096/2048) has been stored in the VDP's Sprite Pattern Generator Table Register.

7.1.2.7 Register 7 (Text/Backdrop Color)

The register maintenance routine for the Text/Backdrop Register is VP.STB. The contents of this register depend on the operating mode of the VDP. If the VDP is in the text mode, then the left nibble of register 7 contains the color of ON pattern bits, and the right nibble contains the color of OFF pattern bits. In all other modes, the left nibble is not used, and the right nibble contains the color of the backdrop.

The following is a table of colors and their names from VDPDEF.ACM:

Color	Name	Value
Transparent	VC.CLR	0
Black	VC.BLK	1
Medium Green	VC.MGR	2
Light Green	VC.LGR	3
Dark Blue	VC.DBL	4
Light Blue	VC.LBL	5
Dark Red	VC.DRD	6
Cyan	VC.CYN	7
Medium Red	VC.MRD	8
Light Red	VC.LRD	9
Dark Yellow	VC.DYL	10
Light Yellow	VC.LYL	11
Dark Green	VC.DGR	12
Magenta	VC.MAG	13
Gray	VC.GRY	14
White	VC.WHT	15

To place a color from the above table in the left nibble, multiply the name from the above table by VC.LFT. In the text mode, the ON bits could be set to black, and the OFF bits could be set to light blue by making the following call to VP.STB:

```
MVI     A,VC.BLK*VC.LFT+VC.LBL
CALL    VP.STB
```

The backdrop color could be set to dark green by making the following call to VP.STB:

```
MVI     A,VC.DGR
CALL    VP.STB
```

VP.STB maintains the most recent text/backdrop color in the variable VP.TBC.

7.1.2.8 Status Register (Read Only)

The VDP contains one read only status register. To read the status register, use the routine VP.RDR. The contents of the status register are returned in register A. The contents of this register are:

7.1.2.8.1 Interrupt Flag (Bit 0)

This bit is set if the VDP is requesting an interrupt. This bit is automatically reset by reading the status register. The mask value from VDPDEF.ACM is VP.IF.

7.1.2.8.2 Fifth Sprite Flag (Bit 1)

The VDP cannot display more than four sprites on any one raster scan line. If this rule is violated, this bit is set and the fifth sprite number is also placed in bits 3-7. The mask value from VDPDEF.ACM is VP.5S.

7.1.2.8.3 Coincidence Flag (Bit 2)

When two or more sprites have one or more overlapping pixels, this bit is turned on. The mask value from VDPDEF.ACM is VP.C.

7.1.2.8.4 Fifth Sprite Number (Bits 3-7)

When the Fifth Sprite Flag is set, these bits will contain the number of the first sprite not displayed (i.e., the fifth sprite on the same horizontal raster scan). The mask value from VDPDEF.ACM is VP.FSN.

7.1.3 "X-Y" Routines for the Multicolor Mode

The file MCXYSUBS.ACM contains two routines to aid in producing "X-Y" plots. The "X-Y" routines divide the screen into 48 rows by 64 columns of display. Each of these 3072 points can be any of the 16 colors. Transparent points will appear to be the same color as the backdrop color. Before these routines can be used the following registers must have been set:

1) The multicolor mode must be enabled using the routine VP.SOP and the option VP.MCM (see 7.1.2.1).

2) The pattern name table base address register must have been loaded using the routine VP.SPN (see 7.1.2.2).

3) The pattern generator table base address register must have been loaded using the routine VP.SPG (see 7.1.2.4).

MC.XYI and MC.XY make use of the Pattern Name Table pointed to by VP.PNT and the Pattern Generator Table address pointed to by VP.PGT. VP.SPN (see 7.1.2.2) will set VP.PNT to the Pattern

Name Table address most recently loaded to the VDP Pattern Name Table Register, and VP.SPG (see 7.1.2.4) will set VP.PGT to the Pattern Generator Table address most recently loaded to the VDP Pattern Generator Table Register.

MC.XYI and MC.XY are described below:

7.1.3.1 MC.XYI (Initialize "X-Y" Plotting)

This routine initializes the Pattern Name Table and the Pattern Generator Table for "X-Y" plotting. The screen is initialized to black. The routine uses a standard 768 Pattern Name Table and a 1536 byte Pattern Generator Table. This routine requires no data to be passed in the registers.

7.1.3.2 MC.XY (Plot an "X-Y" Point)

This routine changes the dot at the (X,Y) point specified in register (H,L) to the color specified by register A. The "X" value is passed in register H. The value contained in register H must be between 0 and 47 inclusive. The "Y" value is passed in register L. The value contained in register L must be between 0 and 63 inclusive. The color of the point is passed in the low order nibble of register A. The value passed must be one of the colors described in 7.1.2.7.

7.1.4 Sprite Processing Routines

The routines described in this section aid the user in programming sprites. The routines are found in the file SPRITE.ACM. These routines provide an easy to use mechanism to read and write the sprite attributes. The programmer need not be concerned with the location of the sprite in VRAM. All references to the sprites are via the sprite number. The routines will automatically convert the sprite number to a VRAM address and read or write the sprite attributes.

The following routines make use of the Sprite Name Table pointed to by VP.SNT, and the Sprite Pattern Generator Table pointed to by VP.SGT. VP.SSN (see 7.1.2.5) will set VP.SNT to point to the Sprite Name Table most recently loaded to the VDP Sprite Name Table Register, and VP.SSG (see 7.1.2.6) will set VP.SGT to point to the Sprite Pattern Generator Table most recently loaded to the VDP Sprite Pattern Generator Table Register.

A sprite attribute consists of four bytes. The first byte is the vertical ("Y") position of the sprite. The second byte is the horizontal ("X") position of the sprite. The third byte is the sprite name. For size 1 sprites, the sprite name must be evenly

divisible by four. The last byte contains the sprite color in the low order four bits (see color codes in 7.1.2.7), and an early clock bit in the high order bit. When the early clock bit is set, the horizontal ("X") value is offset 32 pixels to the left to allow the sprite to bleed in from the left edge of the display.

7.1.4.1 VP.SAA (Get Sprite Attribute Address)

This routine will return the base VRAM address of the sprite attribute block for the sprite number passed in register A. The base address of the sprite attribute block is returned in register HL.

7.1.4.2 VP.SPA (Get Sprite Pattern Address)

This routine will return the base VRAM address of the sprite pattern generator block for the sprite number passed in register A. The base address of the sprite pattern generator block is returned in register HL.

7.1.4.3 VP.GSC (Get Sprite Coordinates)

This routine will return the coordinates of the sprite number passed in register A. The horizontal ("X") value is returned in register H, and the vertical ("Y") value is returned in register L.

7.1.4.4 VP.SSC (Set Sprite Coordinates)

This routine will set the coordinates of the sprite number passed in register A. The horizontal ("X") value is passed in register H, and the vertical ("Y") value is passed in register L.

7.1.4.5 VP.GSA (Get Sprite Attributes)

This routine will return all the attributes of the sprite number passed in register A. The horizontal ("X") value is returned in register H, the vertical ("Y") value is returned in register L, the sprite name is returned in register D, and the sprite color/early clock bit is returned in register E.

7.1.4.6 VP.SSA (Set Sprite Attributes)

This routine will set the attributes of the sprite number passed in register A. This horizontal ("X") value is passed in register H, the vertical ("Y") value is passed in register L, the sprite name is passed in register D, and the sprite color/early

clock bit is passed in register E.

7.1.5 VRAM Maintenance Routines

The file VDPMOVES.ACM contains two routines to facilitate moving blocks of data between VRAM and memory. The routine VP.MTV is used to move data to VRAM, and the routine VP.VTM is used to transfer data from VRAM to memory. The most common use of these routines is to block load VRAM table from memory.

7.1.5.1 VP.MTV (Move from Memory to VRAM)

This routine is used to move data from memory to VRAM. The starting VRAM address is passed in register HL. The starting memory address is passed in register DE. The number of bytes to be transferred is passed in register BC.

7.1.5.2 VP.VTM (Move from VRAM to memory)

This routine is used to move data from VRAM to memory. The starting VRAM address is passed in register HL. The starting memory address is passed in register DE. The number of bytes to be transferred is passed in register BC.

7.2 The AY-3-8910 Programmable Sound Generator (PSG)

This section describes the support software for the PSG. The files PSGDEF.ACM, PSGIO.ACM, LASER.ACM, WBOMB.ACM, WOLF.ACM, and RACECAR.ACM contain the software support for the PSG.

Unless otherwise noted, values requiring 8 bits or less are passed in register A. Values requiring more than 8 bits are passed in register HL. All registers are preserved unless needed to return data.

7.2.1 PSG Register Maintenance Routines

The PSG contains registers to control the tone and amplitude for three channels, a white noise generator, an envelope generator, and two parallel input/output ports. Each of these registers is described below.

7.2.1.1 Tone Control Registers

The PSG has three independent tone channels. The frequency for each channel is control by writing a 12-bit tone period value to registers in the PSG. The tone period is inversely related to

the tone frequency (pitch) by the following forumlae:

Frequency = 111861 / Tone Period equ. 7.2.1.1-1

Tone Period = 111861 / Frequency equ. 7.2.1.1-2

The second equation above will be the more useful. For example, suppose the PSG is to sound the frequency 261.624 hertz (middle C). Since the frequency is known and a tone period is needed for the PSG register, equation 7.2.1.1-2 is applied:

Tone Period = 111861 / 261.624
Tone Period = 427.56
Tone Period = 428 (rounded)

Note the final result must be rounded because an integral tone period value is written to the PSG. Since the result was rounded, a slight frequency error will occur. The exact frequency produced can now be computed using equation 7.2.1.1-1. In this instance the tone period is known, and the exact frequency is not known.

Frequency = 111861 / 428
Frequency = 261.357

A tone period value of 0 is used to turn off a tone channel.

The value 111861 is dependent on the PSG clock. The 3.579545Mhz color burst clock from VDP pin 38 is divided by two yielding a 1.7897725Mhz PSG clock. The PSG internally divides this clock by 16 yielding a final value of 111860.78 which is rounded to 111861.

The PSG tone period registers are 12 bit registers. The tone period values are passed in register HL. No checking is done to insure the values passed do not exceed 4095 (12 bits). Each of the PSG tone period register maintenance routines are described below:

7.2.1.1.1 Channel A Tone Period Registers

The PSG register maintenance routines for the Channel A Tone Period Registers are PS.WTA (Write Tone period A) and PS.RTA (Read Tone period A).

7.2.1.1.1.1 PS.WTA

PS.WTA will write the 12-bit tone period value passed in register HL to PSG register 0 (low order or fine value) and register 1 (high order or coarse value).

To cause PSG tone channel A to sound Middle C, the following calling sequence could be used:

```
LXI     HL,428          ;11861/261.624
CALL    PS.WTA
```

After this call, PSG tone channel A will be set to the frequency 261.357. No sound will be produced until the channel is enabled for output (see 7.2.1.6) and the proper amplitude is set (see 7.2.1.2.1).

7.2.1.1.1.2 PS.RTA

PS.RTA returns the current tone period value for channel A in register HL. The fine value (contents of PSG register 0) is placed in register L, and the 4-bit coarse value (contents of PSG register 1) is placed in register H.

To read current tone period for channel A, the following calling sequence could be used:

```
CALL    PS.RTA
```

After this call register HL will contain the current tone period value for channel A.

7.2.1.1.2 Channel B Tone Period Registers

The PSG register maintenance routines for the Channel B Tone Period Registers are PS.WTB (Write Tone period B) and PS.RTB (Read Tone period B).

7.2.1.1.2.1 PS.WTB

PS.WTB will write the 12-bit tone period value passed in register HL to PSG register 2 (low order or fine value) and register 3 (high order or coarse value).

To cause PSG tone channel B to sound A-440 (440 hertz), the following calling sequence could be used:

```
LXI     HL,254          ;11861/440
CALL    PS.WTB
```

After this call, PSG tone channel B will be set to the frequency 440.398. No sound will be produced until the channel is enabled for output (see 7.2.1.6) and the proper amplitude is set (see 7.2.1.2.2).

7.2.1.1.2.2 PS.RTB

PS.RTB returns the current tone period value for channel B in register HL. The fine value (contents of PSG register 2) is placed in register L, and the 4-bit coarse value (contents of PSG register 3) is placed in register H.

To read current tone period for channel B, the following calling sequence could be used:

```
CALL    PS.RTB
```

After this call register HL will contain the current tone period value for channel B.

7.2.1.1.3 Channel C Tone Period Registers

The PSG register maintenance routines for the Channel C Tone Period Registers are PS.WTC (Write Tone period C) and PS.RTC (Read Tone period C).

7.2.1.1.3.1 PS.WTC

PS.WTC will write the 12-bit tone period value passed in register HL to PSG register 4 (low order or fine value) and register 5 (high order or coarse value).

To cause PSG tone channel C to sound Middle C, the following calling sequence could be used:

```
LXI     HL,428          ;11861/261.624
CALL    PS.WTC
```

After this call, PSG tone channel C will be set to the frequency 261.357. No sound will be produced until the channel is enabled for output (see 7.2.1.6) and the proper amplitude is set (see 7.2.1.2.3).

7.2.1.1.3.2 PS.RTC

PS.RTC returns the current tone period value for channel C in register HL. The fine value (contents of PSG register 4) is placed in register L, and the 4-bit coarse value (contents of PSG register 5) is placed in register H.

To read current tone period for channel C, the following calling sequence could be used:

```
CALL    PS.RTC
```

After this call register HL will contain the current tone period value for channel C.

7.2.1.2 Amplitude Control Registers

The amplitude for each of the three tone channels is controlled by writing an amplitude control mask to the Amplitude Control Register for that channel. The amplitude control mask allows for two modes of operation. A channel can either have a fixed amplitude value, or be placed under control of the envelope generator. Bit 4 of the amplitude control mask selects the mode of operation. If bit 4 is ON (PS.VLA in PSGDEF.ACM), the channel amplitude is placed under the control of the envelope generator (see 7.2.1.4). All other bits in the amplitude control mask are ignored in this mode. If bit 4 is OFF (PS.FLA in PSGDEF.ACM), then the amplitude level is set by the low order 4 bits of the amplitude control mask. This mode allows for 16 logarithmically related amplitude steps. Since the human ear responds logarithmically to amplitude changes, these steps will be heard as 16 equally stepped changes in amplitude.

The routines to maintain the Amplitude Control Register for each of the channels are described below:

7.2.1.2.1 Channel A Amplitude Control Register

The PSG register maintenance routines for the Channel A Amplitude Control Register are PS.WAA (Write channel A Amplitude control register) and PS.RAA (Read channel A Amplitude control register).

7.2.1.2.1.1 PS.WAA

PS.WAA will write the amplitude control mask passed in register A to the PSG Channel A Amplitude Register (register 8).

To set channel A's amplitude at maximum value, the following call could be made:

```
MVI     A,PS.FLA+15
CALL    PS.WAA
```

After this call, the channel A amplitude will be at its maximum level and not under the control of the envelope generator.

7.2.1.2.1.2 PS.RAA

PS.RAA returns the current amplitude control mask for channel A in register A.

To read the current amplitude control mask for channel A, the following call could be made:

```
CALL    PS.RAA
```

After this call, register A will contain the current amplitude control mask for channel A.

7.2.1.2.2 Channel B Amplitude Control Register

The PSG register maintenance routines for the Channel B Amplitude Control Register are PS.WBA (Write channel B Amplitude control register) and PS.RBA (Read channel B Amplitude control register).

7.2.1.2.2.1 PS.WBA

PS.WBA will write the amplitude control mask passed in register A to the PSG Channel B Amplitude Register (register 9).

To set channel B's amplitude under the control of the envelope generator, the following call could be made:

```
MVI     A,PS.VLA
CALL    PS.WBA
```

After this call, channel B's amplitude will be under the control of the envelope generator.

7.2.1.2.2.2 PS.RBA

PS.RBA returns the current amplitude control mask for channel B in register A.

To read the current amplitude control mask for channel B, the following call could be made:

```
CALL    PS.RBA
```

After this call, register A will contain the current amplitude control mask for channel B.

7.2.1.2.3 Channel C Amplitude Control Register

The PSG register maintenance routines for the Channel C Amplitude Control Register are PS.WCA (Write channel C Amplitude control register) and PS.RCA (Read channel C Amplitude control register).

7.2.1.2.3.1 PS.WCA

PS.WCA will write the amplitude control mask passed in register A to the PSG Channel C Amplitude Register (register 10).

To set channel C's amplitude at maximum value, the following call could be made:

```
MVI     A,PS.FLA+15
CALL    PS.WCA
```

After this call, the channel A amplitude will be at its maximum level and not under the control of the envelope generator.

7.2.1.2.3.2 PS.RCA

PS.RCA returns the current amplitude control mask for channel C in register A.

To read the current amplitude control mask for channel C, the following call could be made:

```
CALL    PS.RCA
```

After this call, register A will contain the current amplitude control mask for channel C.

7.2.1.3 Noise Generator Register

The PSG has a white noise generator. The noise frequency is controlled by writing a 5-bit noise period to the Noise Period Register. Equations 7.2.1.1-1 and 7.2.1.1-2 apply to frequency and period calculations for the noise generator. The 5-bit noise period is passed in register A. The Noise Period Register may be read or written with the routines described below:

7.2.1.3.1 PS.WNP

PS.WNP will write the 5-bit noise period passed in register A to PSG register 6 (Noise Period Register).

To set the noise frequency to 10 Khz, the following call could be made:

```
MVI     A,11    ;111861/10000
CALL    PS.WNP
```

After this call, the noise frequency will be set to 10.169 Khz. Before noise will be heard, the noise generator must be enabled for each channel where noise is to be heard (see 7.2.1.6.).

7.2.1.3.2 PS.RNP

PS.RNP returns the current noise period value in register A.

To read the current noise period, the following call could be made:

```
CALL    PS.RNP
```

After this call, register A will contain the noise period.

7.2.1.4 Envelope Generator

The envelope generator consists of two parts: an envelope period value and an envelope shape/cycle control mask.

The envelope period value controls the duration of the envelope, and the envelope shape/cycle control mask is used to define the envelope's shape and whether or not the envelope repeats. A complete description of the Envelope Period Register and the Envelope Shape/Cycle Register follows:

7.2.1.4.1 Envelope Period Registers

The PSG register maintenance routines for the Envelope Period Registers are PS.WEP (Write Envelope Period registers) and PS.REP (Read Envelope Period registers).

The envelope period value is a 16-bit value. The envelope period is inversely related to the envelope frequency by the following formulae:

Frequency = 6991 / Envelope Period equ. 7.2.1.4-1

Envelope Period = 6991 / Frequency equ. 7.2.1.4-2

Envelope Period = 6991 * Cycle Time equ. 7.2.1.4-3

Equation 7.2.1.4-3 will be the most useful equation when the envelope is used to control a decaying sound.

7.2.1.4.1.1 PS.WEP

PS.WEP writes the 16-bit tone period value in register HL to PSG register 11 (low order or fine value) and register 12 (high order or coarse value).

To write the envelope period value for a decaying sound which will last 2 seconds the following call could be made:

```
LXI     HL,13982        ;6991*2
CALL    PS.WEP
```

After this call, the Envelope Period Registers will be set for a 2 second period.

7.2.1.4.1.2 PS.REP

PS.REP returns the current envelope period value in register HL. The fine value (contents of PSG register 11) is placed in register L, and the coarse value (contents of PSG register 12) is placed into register H.

To read the current value of the Envelope Period Registers, the following call could be made:

```
CALL    PS.REP
```

After this call register HL will contain the current envelope period value.

7.2.1.4.2 Envelope Shape/Cycle Control Register

The Envelope Shape/Cycle Control Register contains a 4-bit mask which controls the shape and cycling of the envelope. The function of each of the bits in the mask is described below (the bit name from PSGDEF.ACM is enclosed in parenthesis):

BIT 0 (PS.HLD)	ON: Limit envelope to one cycle. OFF: Do not limit envelope to one cycle.
BIT 1 (PS.ALT)	ON: Reverse counter direction at cycle end. OFF: Do not reverse counter direction.
BIT 2 (PS.ATT)	ON: Start counter direction up. OFF: Start counter direction down.
BIT 3 (PS.CNT)	ON: Start cycle over if bit 0 not ON. OFF: Reset counter to zero after one cycle and hold.

The PSG register maintenance routines for the Envelope Shape/Cycle Control Register are PS.WEC (Write Envelope shape/cycle Control Register) and PS.REC (Read Envelope shape/cycle Control register).

7.2.1.4.2.1 PS.WEP

PS.WEP will write the Envelope Shape/Cycle Control Register (register 13) from the value passed in register A.

To set the envelope shape/cycle to a one cycle decay the following call could be made:

```
MVI     A,PS.HLD
CALL    PS.WEP
```

After this call the Envelope Shape/Cycle Control Register will be set to decay for one cycle. This shape/cycle will control the amplitude of all channels which have enabled the envelope generator in their respective Amplitude Control Register (see 7.2.1.2).

7.2.1.5 Parallel Ports

The PSG has two 8-bit parallel ports, each of which are capable of either input or output. Two bits from each port are brought out to each joystick connector as follows:

Bit	Joystick
0	1
1	1
2	2
3	2
4	3
5	3
6	4
7	4

For detailed information on interfacing to the parallel ports refer to section 4.1.1.

Software support for reading and writing the parallel ports is described below:

7.2.1.5.1 Parallel Port A

The PSG register maintenance routines for parallel port A are PS.WPA (Write Parallel port A) and PS.RPA (Read Parallel port A).

7.2.1.5.1.1 PS.WPA

PS.WPA writes the 8-bit value passed in register A to parallel port A. In order for this operation to work properly, parallel port A must have been enabled for output (see 7.2.1.6).

The following call could be made to set all bits on parallel port A:

```
MVI     A,11111111B
CALL    VP.WPA
```

After this call, all bits on parallel port A will be ON.

7.2.1.5.1.2 PS.RPA

PS.RPA returns the data on parallel port A in register A. In order for this operation to work properly, parallel port A must have been enabled for input (see 7.2.1.6).

7.2.1.5.2 Parallel Port B

The PSG register maintenance routines for parallel port B are PS.WPB (Write Parallel port B) and PS.RPB (Read Parallel port B).

7.2.1.5.2.1 PS.WPB

PS.WPB writes the 8-bit value passed in register A to parallel port B. In order for this operation to work properly, parallel port B must have been enabled for output (see 7.2.1.6).

The following call could be made to clear all bits on parallel port B:

```
XRA     A       ;CLEAR REGISTER A
CALL    PS.WPB
```

After this call, all bits on parallel port B will be OFF.

7.2.1.5.2.2 PS.RPB

PS.RPB returns the data on parallel port B in register A. In order for this operation to work properly, parallel port B must have been enabled for input (see 7.2.1.6).

7.2.1.6 PSG Enable Register

The PSG Enable Register controls the direction of the parallel ports and the three channels by enabling noise and tone on the channels. The PSG Enable Register is loaded with an 8-bit mask. The bit meanings of the mask are described below:

BIT 0	(PS.ETA)	ON: Enable tone on channel A.
	(PS.DTA)	OFF: Disable tone on channel A.
BIT 1	(PS.ETB)	ON: Enable tone on channel B.
	(PS.DTB)	OFF: Disable tone on channel B.
BIT 2	(PS.ETC)	ON: Enable tone on channel C.
	(PS.DTC)	OFF: Disable tone on channel C.
BIT 3	(PS.ENA)	ON: Enable noise on channel A.
	(PS.DNA)	OFF: Disable noise on channel A.
BIT 4	(PS.ENB)	ON: Enable noise on channel B.
	(PS.DNB)	OFF: Disable noise on channel B.
BIT 5	(PS.ENC)	ON: Enable noise on channel C.
	(PS.DNC)	OFF: Disable noise on channel C.
BIT 6	(PS.PAI)	ON: Enable parallel port A for input.
	(PS.PAO)	OFF: Enable parallel port A for output.
BIT 7	(PA.PBI)	ON: Enable parallel port B for input.
	(PA.PBO)	OFF: Enable parallel port B for output.

All other registers are usually initialized before the Enable Register is written. If the parallel ports are not being used, it is advisable to enable them for input. The PSG register maintenance routines for the Enable Register are PS.WER (Write Enable Register) and PS.RER (Read Enable Register).

7.2.1.6.1 PS.WER

PS.WER writes the 8-bit enable mask passed in register A to the PSG Enable Register.

Assuming the parallel ports are not needed, the following call could be made to set both parallel ports for input (allowing for accidental shorts), noise on channel B, and tone on channel A:

```
MVI     A,PS.PAI+PS.PBI+PS.ENB+PS.ETA
CALL    PS.WER
```

After this call both parallel ports will be enabled for input, channel B will be enabled for noise only, and channel A will be enabled for tone. If the other registers have been set properly and the audio cable is connected to an amplifier, the PSG will generate sound.

7.2.2 Sound Effect Examples

There are 4 ACM files which contain sound effect examples which closely follow the examples in chapter 6 of the PSG data sheet. These sound effects can be included in a program by using an XTEXT statement. The sound effects are written as subroutines. For example, the XTEXT statement for the sound effect should not appear in the main line program. A CALL statement for the sound effect should be used to invoke a sound effect subroutine. Each of the sound effects is described below:

7.2.2.1 Laser Sound Effect

This sound effect is in the file LASER.ACM. To use this sound effect, XTEXT the file LASER.ACM outside of the main line code. Whenever this sound effect is desired, CALL the subroutine LASER.

7.2.2.2 Whistling Bomb Effect

This sound effect is in the file WBOMB.ACM. To use this sound effect, XTEXT the file WBOMB.ACM outside of the main line code. Whenever this sound effect is desired, CALL the subroutine WBOMB.

7.2.2.3 Wolf Whistle Sound Effect

This sound effect is in the file WOLF.ACM. To use this sound effect, XTEXT the file WOLF.ACM outside of the main line code. Whenever this sound effect is desired, CALL the subroutine WOLF.

7.2.2.4 Race Car Sound Effect

This sound effect is in the file CAR.ACM. To use this sound effect, XTEXT the file RACECAR.ACM outside of the main line code. Whenever this sound effect is desired, CALL the subroutine CAR.

7.3 The AD7574 Analog to Digital Converter (A/D)

This section describes the programming of the 8 channel A/D. The software support consists of two files: ADDEF.ACM and ADIO.ACM. There is actually only one A/D on the HA-8-3. The 8 channels are obtained by using an 8 channel multiplexer.

The A/D generates an eight bit unsigned number based on the relationship of its input voltage to a reference voltage. The closer the input voltage to the reference voltage the higher the number generated, until eventually the input voltage equals the reference voltage and the A/D returns a value of 255. The formula which governs the number generated is:

Number = (Vin / Vref) * 255 equ. 7.3-1

or simply

Number = (Input Voltage / 10) * 255

Voltages which slightly exceed the reference voltage will convert to 255, voltages which greatly exceed the reference voltage may damage the A/D.

The only routine required to support the A/D is AD.RD.

7.3.1 AD.RD

AD.RD converts the voltage on the analog channel passed in register L to a number and returns the number in register A. The following call could be made to read the voltage on analog channel 3:

```
MVI     L,3
CALL    AD.RD
```

After this call, register A contains the number based on equation 7.3-1 where Vin is the voltage on analog channel 3.

7.4 The 9511A Arithmetic Processing Unit (APU)

This section describes the support software for the APU. The files APUDEF.ACM and APUIO.ACM contain the software support for the APU.

Currently there are no support routines for the 9512. However its operation is very similiar to the 9511A.

The APU contains a status register, command register, and an operand stack. Two terms are used to describe the elements on the stack, they are: TOS (Top Of Stack) which stands for the top element on the stack, and NOS (Next On Stack) which stands for the element just beneath the TOS.

The software support routines for the status register, command register, and operand stack are described below:

7.4.1 The Status Register

The APU contains a status register which contains status information for the APU. The status register is an 8-bit register with the following bit definitions:

7.4.1.1 Carry/Borrow (BIT 0)

When this bit is ON, the previous operation resulted in a carry or borrow from the most significant bit. This bit is valid only after the completion of a command. The bit name from APUDEF.ACM is AP.CRY.

7.4.1.2 Error Code (BITS 4-1)

These bits indicate the error status after the completion of a command. These bits are valid only after the completion of a command. To select the error code from the status byte, use the mask value AP.ERC in APUDEF.ACM. Once the error code is selected, the error code bits have the following meaning:

```
0000           - No Error has occurred.
1000 (AP.DZE)  - Divide by zero error.
0100 (AP.SRN)  - Square root or log of negative number.
1100 (AP.ARG)  - Arg to inverse sin, cos, or exp too large.
XX10 (AP.UFL)  - Underflow has occurred.
XX01 (AP.OFL)  - Overflow has occurred.
```

7.4.1.3 Zero (BIT 5)

When this bit is ON, the result of the previous operation was zero. This bit is valid only after the completion of a command. The bit name from APUDEF.ACM is AP.ZER.

7.4.1.4 Sign (BIT 6)

When this bit is ON, the result of the previous operation was negative. This bit is valid only after the completion of a command. The bit name from APUDEF.ACM is AP.NEG.

7.4.1.5 Busy (BIT 7)

When this bit is ON, the APU is currently executing a command and the other bits in the Status Register are not defined. The bit name from APUDEF.ACM is AP.BSY.

7.4.2 Status Register Software Support Routine

The software support routine for the status register is AP.STS (get STatuS register).

7.4.2.1 AP.STS

This routine will return the APU status register in register A.

The following code could be used to read the APU status register and check to see if an argument was too large.

```
APUBSY  CALL    AP.STS
        MOV     B,A
        ANI     AP.BSY
        JNZ     APUBSY
        MOV     A,B
        ANI     AP.ERC
        CPI     AP.ARG
        JZ      TOOLRG
```

After this code, register A contains the error number, and register B contains the Status Register. Note the loop to wait until the operation is complete. Without this loop the other bits in the Status Register may not have been valid.

7.4.3 The Command Register

To use the APU, the operand(s) for an operation are placed on the operand stack and a command byte is written to the Command Register.

A list of the APU commands, the command name from APUDEF.ACM, and a brief description of the commands follows:

Mneumonic	Value	Summary Description
16-bit Fixed-point Operations		
AP.SADD	6C	NOS = TOS + NOS ; POP
AP.SSUB	6D	NOS = NOS - TOS ; POP
AP.SMUL	6E	NOS = LOW (NOS * TOS) ; POP
AP.SMUU	76	NOS = HIGH (NOS * TOS) ; POP
AP.SDIV	6F	NOS = INT (NOS / TOS) ; POP
32-bit Fixed-point Operations		
AP.DADD	2C	NOS = TOS + NOS ; POP
AP.DSUB	2D	NOS = NOS - TOS ; POP
AP.DMUL	2E	NOS = LOW (NOS * TOS) ; POP
AP.DMUU	36	NOS = HIGH (NOS * TOS) ; POP
AP.DDIV	2F	NOS = INT (NOS / TOS) ; POP
32-bit Floating-point Primary Operations		
AP.FADD	10	NOS = TOS + NOS ; POP
AP.FSUB	11	NOS = NOS - TOS ; POP
AP.FMUL	12	NOS = NOS * TOS ; POP
AP.FDIV	13	NOS = NOS / TOS ; POP
32-bit Floating-point Derived Operations		
AP.SQRT	01	TOS = SQRT (TOS)
AP.SIN	02	TOS = SIN (TOS) [TOS in radians]
AP.COS	03	TOS = COS (TOS) [TOS in radians]
AP.TAN	04	TOS = TAN (TOS) [TOS in radians]
AP.ASIN	05	TOS = ARCSIN (TOS)
AP.ACOS	06	TOS = ARCCOS (TOS)
AP.ATAN	07	TOS = ARCTAN (TOS)
AP.LOG	08	TOS = LOG10 (TOS)
AP.LN	09	TOS = LN (TOS)
AP.EXP	0A	TOS = EXP (TOS)
AP.PWR	0B	NOS = NOS ** TOS ; POP

Data and Stack Manipulation Operations

AP.NOP	00	
AP.FIXS	1F	TOS = FIXS (TOS [floating])
AP.FIXD	1E	TOS = FIXD (TOS [floating])
AP.FLTS	1D	TOS = FLTS (TOS [fixed single])
AP.FLTD	1C	TOS = FLTD (TOS [fixed double])
AP.CHSS	74	TOS = -TOS [fixed single]
AP.CHSD	34	TOS = -TOS [fixed double]
AP.CHSF	15	TOS = -TOS [floating]
AP.PTOS	F7	PUSH ; TOS = NOS [fixed single]
AP.PTOD	B7	PUSH ; TOS = NOS [fixed double]
AP.PTOF	97	PUSH ; TOS = NOS [floating]
AP.POPS	78	POP [fixed single]
AP.POPD	38	POP [fixed double]
AP.POPF	18	POP [floating]
AP.XCHS	F9	XCHG (TOS,NOS) [fixed single]
AP.XCHD	39	XCHG (TOS,NOS) [fixed double]
AP.XCHF	19	XCHG (TOS,NOS) [floating]
AP.PUPI	1A	PUSH ; TOS = 3.1415926

7.4.4 Command Register Software Support Routine

The software support routine for the command register is AP.CMD (write CoMmanD).

7.4.4.1 AP.CMD

This routine will write the command passed in register A to the APU Command Register.

The following call could be made to push PI on the stack:

```
MVI     A,AP.PUPI
CALL    AP.CMD
```

After this code is executed, the APU TOS will contain the floating-point value PI (3.1415926).

7.4.5 Operand Stack Software Support Routines

The software support routines for the operand stack can be broken into a 16-bit group and a 32-bit group. When an operand transfer involves memory, register HL points to the low order byte in memory. Operand bytes are stored in ascending order of significance. For example, as the memory address increases, the operand bytes go from low order to high order.

7.4.5.1 16-bit Operand Group

The 16-bit operand group allows for the operand to be stored directly in register HL or allows register HL to contain the address of the operand in memory.

7.4.5.1.1 16-bit Register Operand Group

In this group register HL contains the data involved in the stack transfer. There are two operations in this group: AP.LSR (Load Single precision operand to Register HL), and AP.SSR (Store on stack Single precision operand from Register HL).

7.4.5.1.1.1 AP.LSR

AP.LSR stores the fixed-point single precision TOS to register HL. The operand stack is POP'ed by this operation.

7.4.5.1.1.2 AP.SSR

AP.SSR PUSH'es onto the operand stack the fixed-point single precision value passed in register HL.

7.4.5.1.2 16-bit Memory Operand Group

In this group register HL contains the memory address of the data involved in the stack transfer. There are two operations in this group: AP.LSO (Load Single precision Operand to memory), and AP.SSO (Store on stack Single precision Operand from memory).

7.4.5.1.2.1 AP.LSO

AP.LSO stores the single precision fixed-point TOS to the memory location pointed to by register HL. The operand stack is POP'ed by this operaion.

7.4.5.1.2.2 AP.SSO

AP.SSO PUSH'es the single precision fixed-point pointed to by register HL onto the operand stack.

7.4.5.2 32-bit Operand Group

The 32-bit operand group is used to PUSH and POP 32-bit operands. No distinction is made between 32-bit fixed-point operands and 32-bit floating-point operands. The exponent of floating-point operands is treated as the most significant byte.

There are two operations for the 32-bit operand group: AP.LDO (Load Double Operand from stack), and AP.SDO (Store Double Operand on stack).

7.4.5.2.1 AP.LDO

AP.LDO loads the 32-bit TOS to the memory pointed to by register HL. The operand stack is POP'ed by this operation.

7.4.5.2.2 AP.SDO

AP.SDO PUSH'es the 32-bit operand pointed to by register HL to the operand stack.

8 Theory of operation

The HA-8-3 Color Graphics Board consists of six major functions:

1) H8 bus interface

2) Device select

3) Video Display Processor (VDP)

4) Programmable Sound Generator (PSG)

5) Analog to Digital Converter (A/D)

6) Optional Arithmetic Processor Unit (APU)

Refer to the schematic of the HA-8-3 when reading the following theory of operation.

8.1 H8 Bus Interface

The high order eight bits (same as low order eight bits for I/O operations) of the address bus (A7 - A0) are buffered by U6. The high order seven bits (A7 - A1) are used only by the device select logic (see section 8.2). The eighth bit (A0) is used by the VDP, PSG and APU to select internal control or data registers.

The data bus (D7 - D0) is buffered by the bidirectional bus drivers U19 and U20. The direction of these drivers is controlled by U8 pin 11. When an I/O read operation is in progress (U12 pin 4) and the board is selected (U18 pin 6) then U19 and U20 buffer the internal data bus onto the H8 data bus. Otherwise, U19 and U20 buffer the H8 data bus onto the internal data bus.

There are four bus control signals which are buffered by U12. These control signals are: I/O read (U12 pin 3), I/O write (U12 pin 9), Reset (U12 pin 13), and the system clock (U12 pin 11).

The HA-8-3 uses wait states to synchronize slow operations with the H8 bus. The longest wait state (20 microseconds) is generated by the A/D. A request for a wait state is made at U18 pin 8. This in turn causes the open-collector buffer at U25 pin 6 to bring the "READY IN" bus line low. This causes a wait state to be generated by the CPU.

The remaining five open-collector buffers of U25 are used

for interrupts. The VDP and APU can each produce interrupts. An interrupt request by the VDP causes U25 pin 8 to be brought low. An interrupt request by the APU causes U25 pin 10 to be brought low. There are also three user interrupt requests provided for. The outputs of these open-collector buffers are brought to the interrupt select grid. Jumpers must be installed from the interrupt source to the desired interrupt request number. Since interrupts are not used by the supplied software, no interrupts are factory set, and no interrupt numbers have been assigned to the HA-8-3. This facility is made available in case user written software requires it.

8.2 Device Select

The high order seven address bits (A7 - A1) are decoded to form a three digit octal I/O address. Since the eighth (low order) address bit (A0) is used to select a control or data function, it is not decoded. Therefore only even/odd pairs of addresses are decoded.

The seven high order address bits are decoded by U4 and U5 only when the "Board Enable" jumper is "ON" and there is an I/O operation in progress. The board comes with the jumper for "Board Enable" set to "ON". An I/O operation is detected at U2 pin 11 whenever an I/O read (U12 pin 4) or an I/O write (U12 pin 8) occurs.

The two high order address bits (A7 and A6) are decoded by U5. These output are brought out to the "DXX" select grid at "0XX", "1XX", "2XX", and "3XX". The high order digit of the octal I/O address for each device is set by jumpering the "DXX" pads labeled "VDP", "PSG", "APU", and "A/D" to "0XX", "1XX", "2XX", or "3XX". Since the assigned high order digit for all devices is 2, there is a factory installed jumper to "2XX" for all four devices.

In a similar manner, the next three address bits (A5 - A3) are decoded by U4. The decoded output are brought out to the "XDX" select grid at "X0X", "X1X", ..., "X7X". The middle digit of the octal I/O address for each device is set by jumpering the "XDX" pads labeled "VDP", "PSG", "APU" and "A/D" to the decoded address output which corresponds to the middle digit of the I/O address. Since the assigned middle digit for all devices is 7, there is a factory installed jumper to ""7XX" for all four devices.

Finally, the low order digit of the octal I/O address is decoded by U5. Since the low order address bit (A0) is not decoded, even/odd address pairs are decoded. The decoded outputs are brought out to the "XXD" grid at "XX0", "XX2", "XX4", and "XX6". The low order digit of the octal I/O address for each device is set by jumpering the "XXD" pads labeled "VDP", "PSG",

"APU", and "A/D" to the decoded address output which corresponds to the low order digit of the octal I/O address. The assigned I/O address of the VDP is 270Q. There is a factory installed jumper from "XX0" to "VDP" on the "XXD" grid. The assigned I/O address of the PSG is 272Q. There is a factory installed jumper from "XX2" to "PSG" on the "XXD" grid. The assigned I/O address of the APU is 274Q. There is a factory installed jumper from "XX4" to "PSG" on the "XXD" grid. The assigned I/O address of the A/D is 276Q. There is a factory installed jumper from "XX6" to "PSG" on the "XXD" grid.

8.3 Video Display Processor Section

The VDP section of the HA-8-3 consists of a TMS-9918A (U14), eight 4116 150ns Dynamic RAMS (U26 - U33), and a video amplifier (Q1 - Q4). Timing for the VDP section is derived from a 10.738635 Mhz crystal (three times the color burst frequency). The crystal can be trimmed by adjusting C6.

Note that in the documentation for the TMS-9918A, the designation for a low order bit is opposite that of all other components on the HA-8-3. A "7" is used to designate the low order bit, while a "0" designates the high order bit.

The "CSW" pin of the TMS-9918A (U14 pin 18) is used to select the TMS-9918A for a write operation. U8 pin 6 detects when the TMS-9918A is selected and an I/O write operation is in progress.

The "CSR" pin of the TMS-9918A (U14 pin 15) is used to select the TMS-9918A for a read operation. U8 pin 8 detects when the TMS-9918A is selected and an I/O read operation is in progress.

The "MODE" pin of the TMS-9918A (U14 pin 13) is used to select a control or data operation. This pin is connected to the low order address bit (A0).

The video amplifier in the VDP section buffers the composite video output of the TMS-9918A (U14 pin 36). In order to interface to as many different devices as possible (color monitors, RF modulators, video tape recorders, etc.) the video amplifier has adjustments for peak to peak output voltage (R3), and the DC bias voltage (R1). The output impedance of the video amplifier is 75 ohms.

8.4 Programmable Sound Generator Section

The PSG section of the HA-8-3 consists of an AY-3-8910 (U1) and associated timing circuitry. The AY-3-8910 contains three tone generators, two 8 bit parallel I/O ports, one noise

generator, and one envelope generator. The two parallel I/O ports are brought out to the four joystick connectors.

Two bits from each of the two ports are brought out to each joystick. If one port is set up for input and the other for output, then each joystick can have two bits of input for switches, and two bits of output for LEDs.

The clock for the AY-3-8910 (U1 pin 22) is derived from the color burst clock on the TMS-9918A (U14 pin 38). The buffered 3.579545 Mhz clock is divided by U7 to produce the 1.7897725 Mhz clock for the AY-3-8910. This is the clock frequency used in the examples in the AY-3-8910 data sheet.

Due to the data bus setup and hold requirements of the AY-3-8910, the PSG section requires that wait states be added to I/O operations. This timing is generated by U3 and associated circuitry.

8.5 Analog to Digital Converter Section

The A/D section of the HA-8-3 consists of an AD7501 or AD7503 8 channel analog multiplexer (U17 and U24), a buffer amplifier (U22), an AD7424 Analog to Digital Converter (U23), and associated timing circuitry (around U35). The eight analog channels are brought in from the 4 joystick connectors.

The A/D uses a successive approximation technique to convert the analog voltage to a digital number. A conversion requires approximately 20 microseconds. During the conversion, the associated timing circuity of the A/D places the CPU in a wait state.

To use the A/D, a 3-bit channel number is first written to the A/D I/O port to select a channel. Before an A/D conversion can begin, time must be given for the analog multiplexer to switch and for the buffer amplifier to settle. This requires a total of 35 microseconds. This 35 microsecond delay is not required if the analog channel is not changed.

8.6 Arithmetic Processor Section

The HA-8-3 allows for the use of either the 9511A or 9512 Arithmetic Processor Unit (U21). Depending on the operation, the APU may inject up to 6 wait states (U21 pin 17 and U36).

9 Warranty and Service

The HA-8-3 Color Graphics Board is fully warranted against electrical failure for a period of 90 days after date of purchase. This warranty is limited to electrical failure of the HA-8-3, and does not cover physical damage to the HA-8-3 circuit board, external amplifiers, monitors, RF modulators, video recorders, or other devices attached to the HA-8-3. This warranty does not cover the backplane or any other component of the computer system.

Should the HA-8-3 require service, mail it insured and postage paid to:

New Orleans General Data Services, Inc.
7230 Chadbourne Drive
New Orleans, Louisiana 70126

It is recommended that the board be packaged carefully and insured. If the HA-8-3 is under warranty, include a copy of the sales receipt showing date and place of purchase. Warranty work will not be performed unless a copy of the sales receipt is enclosed.

The HA-8-3 will be repaired and returned via COD insured first-class mail or UPS for parts, labor, and postage. Warranty work will not be charged for labor or parts, but will be charged for shipping and insurance. The current charge for labor on the HA-8-3 is sixty dollars. This price is subject to change without prior notice. A request for a price quotation may be made before sending a board in for repair.

The above repair policy applies only to electrical failures. Boards with physical damage will be repaired or returned without repair at the discretion of New Orleans General Data Services, Inc., depending on the nature of and severity of the physical damage.

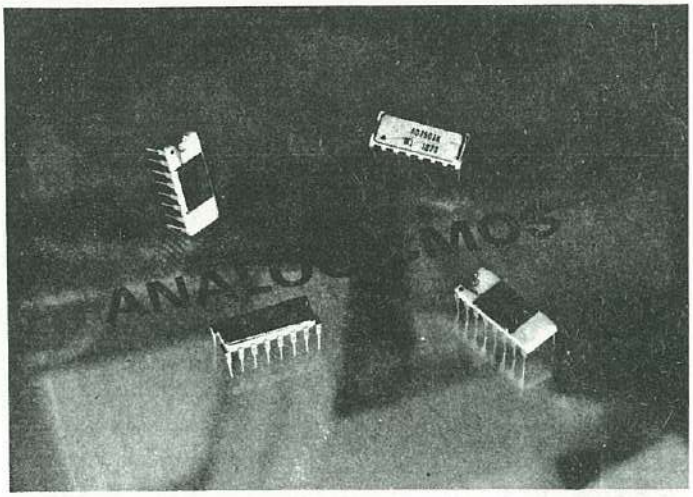
Appendix A

AD7501/AD7503 Data Sheet

CMOS 4/8 Channel Analog Multiplexers

AD7501, AD7502, AD7503

FEATURES
DTL/TTL/CMOS Direct Interface
Power Dissipation: 30μW
R_{ON}: 170Ω
Output "Enable" Control
AD7503 Replaces HI-1818



GENERAL DESCRIPTION

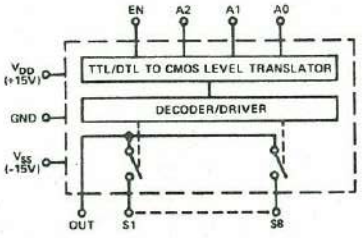
The AD7501 and AD7503 are monolithic CMOS, 8 channel analog multiplexers which switches one of 8 inputs to a common output depending on the state of three binary address lines and an "enable" input. The AD7503 is identical to the AD7501 except its "enable" logic is inverted. All digital inputs are TTL/DTL and CMOS logic compatible.

The AD7502 is a monolithic CMOS dual 4-channel analog multiplexer. Depending on the state of 2 binary address inputs and an "enable", it switches two output busses to two of 8 inputs.

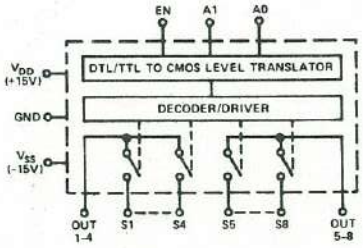
The AD7502 is an excellent example of a high breakdown CMOS process combined with a double layer interconnect for high density. Silicon nitride passivation ensures long term stability and reliability.

FUNCTIONAL DIAGRAMS

AD7501, AD7503

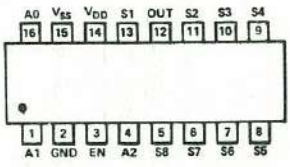


AD7502

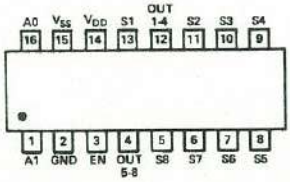


PIN CONFIGURATIONS (Top View)

AD7501, AD7503



AD7502



ABSOLUTE MAXIMUM RATINGS

(T_A = +25°C unless otherwise noted)

V_{DD} – GND	+17V
V_{SS} – GND	-17V
V Between Any Switch Terminals	25V
Switch Current (I_S, Continuous)	35mA
Switch Current (I_S, Surge)	
1ms duration, 10% duty cycle	50mA
Digital Input Voltage Range	V_{DD} to GND
Power Dissipation (package)	
16 pin Ceramic DIP	
Up to +75°C	450mW
Derates above +75°C by	6mW/°C
16 pin Plastic DIP	
Up to +70°C	670mW
Derates above +70°C by	8.3mW/°C
Operating Temperature	
Plastic	0 to +75°C
Ceramic (J, K versions)	-25°C to +85°C
Ceramic (S version)	-55°C to +125°C
Storage Temperature	-65°C to +150°C

CAUTION:

1. Do not apply voltages higher than V_{DD} and V_{SS} to any other terminal, especially when V_{SS} = V_{DD} = 0V all other pins should be at 0V.
2. The digital control inputs are zener protected; however, permanent damage may occur on unconnected units under high energy electrostatic fields. Keep unused units in conductive foam at all times.

Route 1 Industrial Park; P.O. Box 280; Norwood, Mass. 02062
Tel: 617/329-4700 **TWX: 710/394-6577**
West Coast 213/595-1783 **Mid-West 312/894-3300** **Texas 214/231-5094**

SPECIFICATIONS (V_{DD} = +15V, V_{SS} = -15V unless otherwise noted)

PARAMETER	VERSION[1]	SWITCH CONDITION	@25°C AD7501, AD7503	@25°C AD7502	OVER SPECIFIED TEMP. RANGE AD7501, AD7503	OVER SPECIFIED TEMP. RANGE AD7502	TEST CONDITIONS
ANALOG SWITCH							
R_{ON}	All	ON	170Ω typ, 300Ω max	*			-10V ≤ V_S ≤ +10V
R_{ON} vs. V_S	All	ON	20% typ	*			I_S = 1.0mA
R_{ON} vs. Temperature	All	ON	0.5%/°C typ	*			V_S = 0V, I_S = 1.0mA
ΔR_{ON} Between Switches	All	ON	4% typ	*			
R_{ON} vs. Temperature Between Switches	All	ON	±0.01%/°C	*			
I_S	J, K	OFF	0.2nA typ, 2nA max	*	50nA max	*	V_S = -10V, V_{OUT} = +10V and V_S = +10V, V_{OUT} = -10V
	S	OFF	0.5nA max	*	50nA max	*	
I_{OUT}	J, K	OFF	1nA typ, 10nA max	0.6nA typ, 5nA max	250nA max	125nA max	V_S = -10V, V_{OUT} = +10V and V_S = +10V, V_{OUT} = -10V
	S	OFF	5nA max	3nA max	250nA max	125nA max	AD7501: Enable HIGH AD7502, 03: Enable LOW
$\lvert I_{OUT} - I_S \rvert$	J, K	ON	12nA max	7nA max	300nA max	175nA max	V_S = 0
	S	ON	5.5nA max	3.5nA max	300nA max	175nA max	
DIGITAL CONTROL							
V_{INL}	All				0.8V max	*	
V_{INH}	J				3.0V min	*	Note 2
	K, S				2.4V min	*	
I_{INL} or I_{INH}	All		10nA typ	*			
C_{IN}	All		3pF typ	*			
DYNAMIC CHARACTERISTICS[3]							
t_{ON}	All		0.8μs typ	*			V_{IN} = 0 to +5.0V (See Test Circuit 2)
t_{OFF}	All		0.8μs typ	*			
C_S	All	OFF	5pF typ	*			
C_{OUT}	All	OFF	30pF typ	15pF typ			
C_{S-OUT}	All	OFF	0.5pF typ	*			
C_{SS} Between Any Two Switches	All	OFF	0.5pF typ	*			
POWER SUPPLY							
I_{DD}	J, K		1μA typ, 100μA max	*			All Digital Inputs Low
I_{SS}	J, K		1μA typ, 100μA max	*			
I_{DD}	S		500μA max	*	500μA max	*	
I_{SS}	S		500μA max	*	500μA max	*	
I_{DD}	J, K		200μA typ, 500μA max	*			All Digital Inputs High
I_{SS}	J, K		1μA typ, 100μA max	*			
I_{DD}	S		800μA max	*	800μA max	*	
I_{SS}	S		800μA max	*	800μA max	*	

NOTES:
*Same specifications as AD7501 and AD7503.
[1] JN, KN versions specified for 0 to +75°C; JD, KD versions for -25°C to +85°C; and SD versions for -55°C to +125°C.
[2] A pullup resistor, typically 1-2kΩ is required to make the AD7501J, AD7502J and AD7503J compatible with TTL/DTL levels. The maximum value is determined by the output leakage current of the driver gate when in the high state.
[3] AC parameters are sample tested to ensure conformance to specifications.

Specifications subject to change without notice.

TRUTH TABLES

AD7501

A_2	A_1	A_0	E_N	"ON"
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8
X	X	X	0	None

AD7503

A_2	A_1	A_0	E_N	"ON"
0	0	0	0	1
0	0	1	0	2
0	1	0	0	3
0	1	1	0	4
1	0	0	0	5
1	0	1	0	6
1	1	0	0	7
1	1	1	0	8
X	X	X	1	None

AD7502

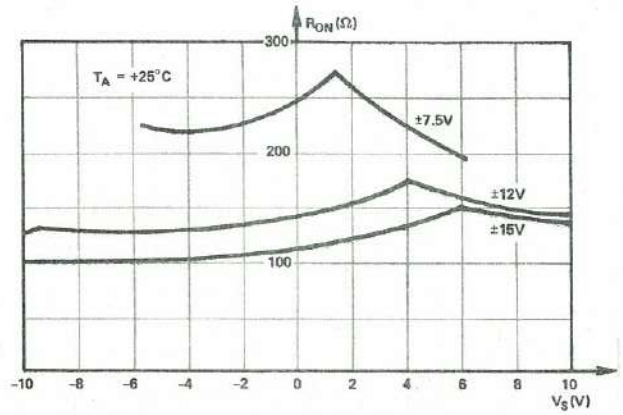
A_1	A_0	E_N	"ON"
0	0	1	1 & 5
0	1	1	2 & 6
1	0	1	3 & 7
1	1	1	4 & 8
X	X	0	None

ORDERING INFORMATION

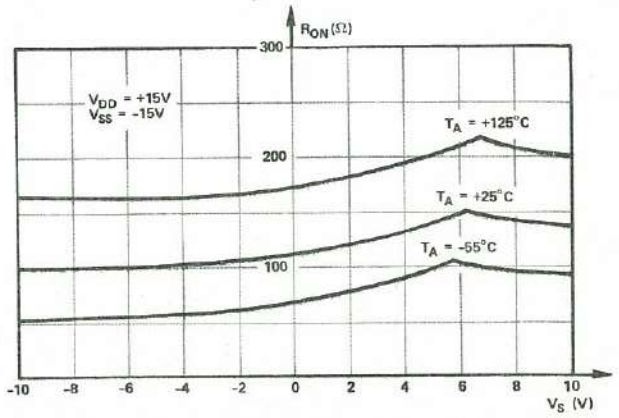
Plastic (Suffix N)	Ceramic (Suffix D)	Operating Temperature Range
AD7501JN AD7501KN AD7503JN AD7503KN		0 to +75°C
	AD7501JD AD7501KD AD7503JD AD7503KD	-25°C to +85°C
	AD7501SD AD7503SD	-55°C to +125°C
AD7502JN AD7502KN		0 to +75°C
	AD7502JD AD7502KD	-25°C to +85°C
	AD7502SD	-55°C to +125°C

Typical Performance Characteristics

1. R_{ON} As A Function Of Switch Voltage (V_S)

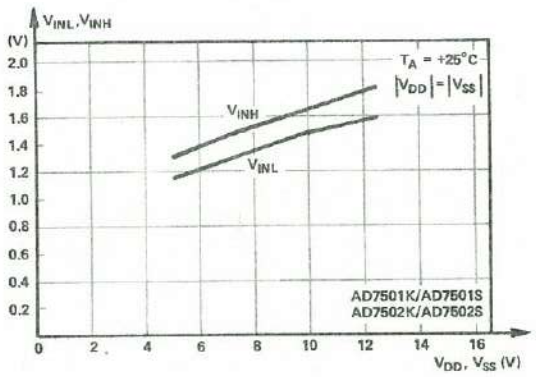


At Different Power Supplies

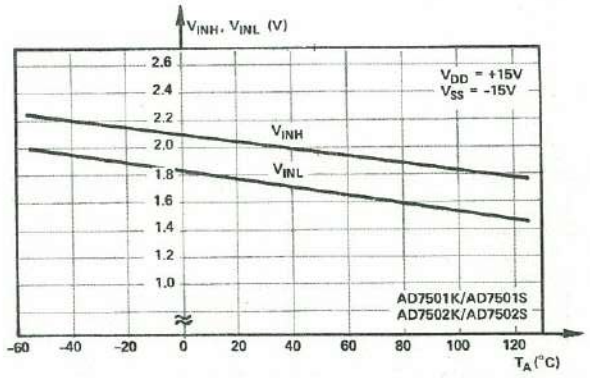


At Different Temperatures

2. Digital Threshold Voltage (V_{INH}, V_{INL})

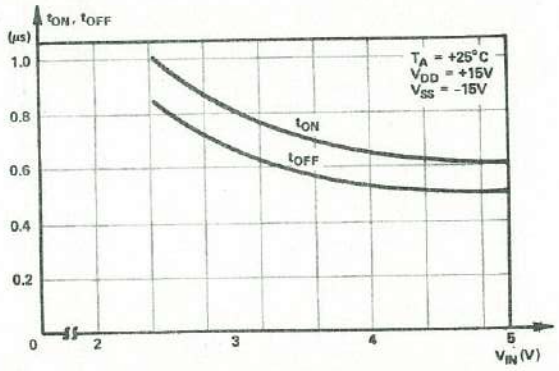


vs. Power Supply



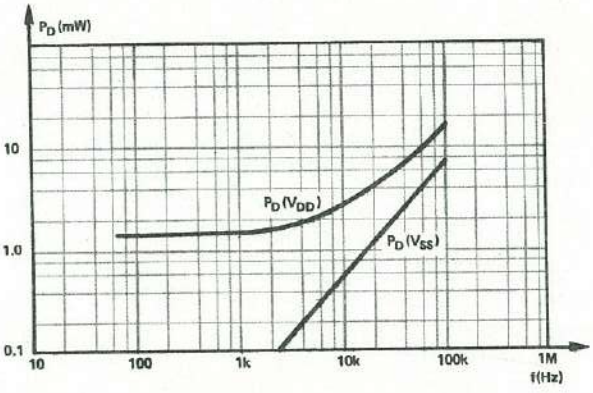
vs. Temperature

3. T_{ON}, T_{OFF}



vs. Digital Input Voltage

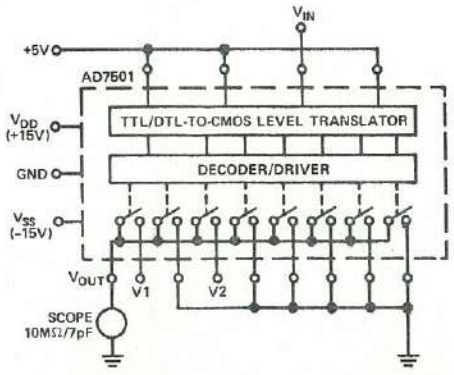
4. Power Dissipation



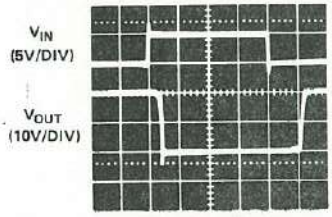
vs. Logic Frequency (50% Duty Cycle)

TYPICAL SWITCHING CHARACTERISTICS

TEST CIRCUIT 1

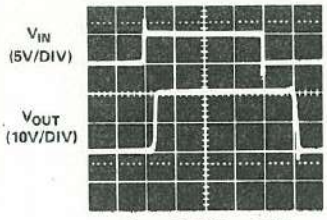


1μs/DIV



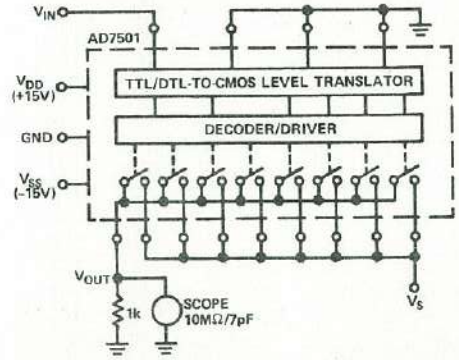
V_1 = -10V, V_2 = +10V

1μs/DIV

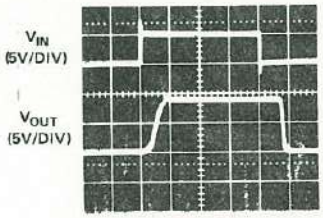


V_1 = +10V, V_2 = -10V

TEST CIRCUIT 2

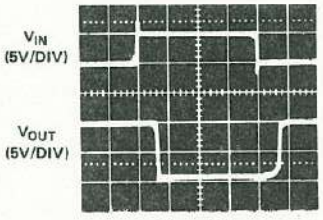


1μs/DIV



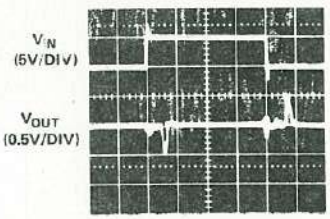
V_S = +10V

1μs/DIV



V_S = -10V

1μs/DIV

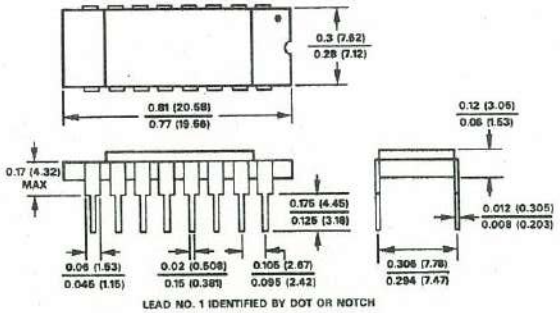


V_S = OPEN

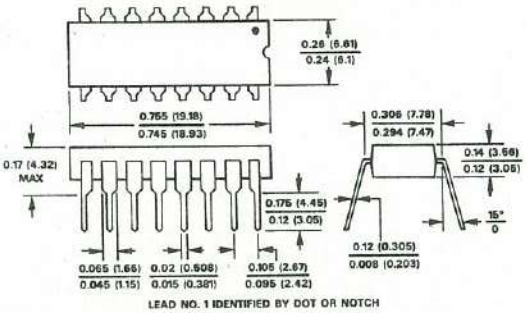
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

16-PIN CERAMIC DIP (SUFFIX D)

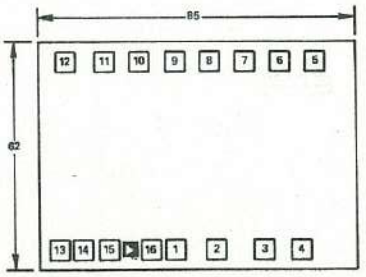


16-PIN PLASTIC DIP (SUFFIX N)

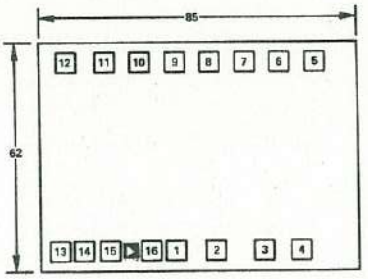


BONDING DIAGRAMS

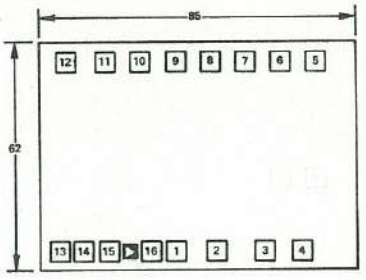
AD7501



AD7502



AD7503



All bonding pads are 4 x 4 MIL.
All pad numbers correspond with DIP package pin configuration.

Appendix B

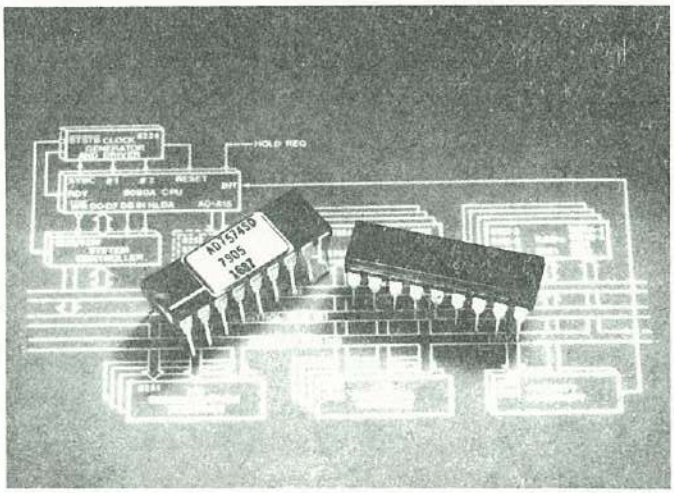
AD7574 Data Sheet

ANALOG DEVICES

CMOS µP Compatible 8-Bit ADC

AD7574

PRELIMINARY TECHNICAL DATA



FEATURES

- 8-Bit Resolution
- No Missed Codes over Full Temperature Range
- Fast Conversion Time: 15µs
- Interfaces to µP like RAM, ROM or Slow-Memory
- Low Power Dissipation: 30mW
- Ratiometric Capability
- Single +5V Supply
- Low Cost
- Internal Comparator and Clock Oscillator

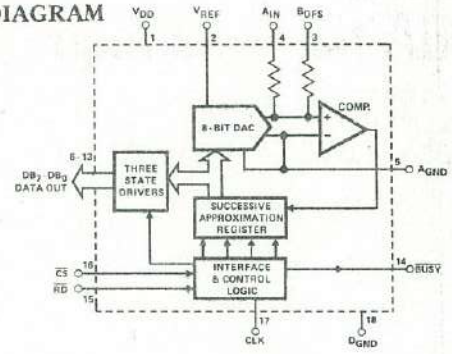
GENERAL DESCRIPTION

AD 7574 is a low-cost, 8-bit µP compatible ADC which uses the successive-approximations technique to provide a conversion time of 15µs.

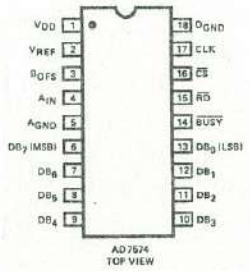
Designed to be operated as a memory mapped input device, the AD7574 can be interfaced like static RAM, ROM, or slow memory. It's $\overline{CS}$ (decoded device address) and $\overline{RD}$ ($\overline{READ}/\overline{WRITE}$ control) inputs are available in all µP memory systems. These two inputs control all ADC operations such as starting conversion or reading data. The ADC output data bits use three-state logic, allowing direct connection to the µP data bus or system input port.

Internal clock, +5V operation, on-board comparator and interface logic, as well as low power dissipation (30mW) and fast conversion time make the AD7574 ideal for most ADC/µP interface applications. Small size (18-pin DIP) and monolithic reliability will find wide use in avionics, instrumentation, and process automation applications.

FUNCTIONAL DIAGRAM



PIN CONFIGURATION



ORDERING INFORMATION

Differential Nonlinearity	Temperature Range and Package		
	Plastic 0°C to +70°C	Ceramic -25°C to +85°C	Ceramic -55°C to +125°C
±7/8 LSB	AD7574JN	[1] AD7574AD	[1] AD7574SD
±3/4 LSB	AD7574KN	[1] AD7574BD	[1] AD7574TD

Note 1: Available 100% screened to MIL-STD-883, Class B. To order, add "/883B" to part number shown. See note 6, page 2 for details.

Route 1 Industrial Park; P.O.Box 280; Norwood, Mass. 02062
Tel: 617/329-4700 TWX: 710/394-6577
West Coast 213/595-1783 | Mid-West 312/894-3300 | Texas 214/231-5094

CAUTION:
ESD (Electro-Static-Discharge) sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subject to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts. The foam should be discharged to the destination socket before devices are removed.

DC SPECIFICATIONS (V_{DD} = +5V, V_{REF} = -10V, Unipolar Configuration, R_{CLK} = 180kΩ, C_{CLK} = 100pF, unless otherwise noted)

PARAMETER	LIMITS T_A = +25°C	LIMITS T_{min}, T_{max}[1]	UNITS	CONDITIONS/COMMENTS
ACCURACY				
Resolution	8	8	Bits	
Relative Accuracy Error				
AD7574JN, AD, SD	±¾	±¾	LSB max	
AD7574KN, BD, TD	±½	±½	LSB max	Relative Accuracy and Differential Nonlinearity are measured dynamically using the external clock circuit of Fig. 7b, page 6. Clock frequency is 500kHz (conversion time 15μs)
Differential Nonlinearity				
AD7574JN, AD, SD	±⅞	±⅞	LSB max	
AD7574KN, BD, TD	±¾	±¾	LSB max	
Full Scale Error (Gain Error)				Full Scale Error is measured after calibrating out offset error. See Fig. 8a and associated calibration procedure for offset. Max Full Scale change from +25°C to T_{min} or T_{max} is ±2 LSB.
AD7574JN, AD, SD	±5	±6.5	LSB max	
AD7574KN, BD, TD	±3	±4.5	LSB max	
Offset Error[2]				
AD7574JN, AD, SD	±60	±80	mV max	Maximum Offset change from +25°C to T_{min} or T_{max} is ±20mV.
AD7574KN, BD, TD	±30	±50	mV max	
Mismatch Between B_{OFS} (pin 3) and A_{IN} (pin 4) Resistances[3]	±1.5	±1.5	%	
ANALOG INPUTS				
Input Resistance				
At V_{REF} (pin 2)	5/10/15	5/10/15	kΩ min/typ/max	
At B_{OFS} (pin 3)	10/20/30	10/20/30	kΩ min/typ/max	
At A_{IN} (pin 4)	10/20/30	10/20/30	kΩ min/typ/max	
V_{REF} (for specified performance)	-10	-10	V	±5% for specified transfer accuracy.
V_{REF} Range[4]	-5 to -15	-5 to -15	V	Degraded transfer accuracy.
Nominal Analog Input Range				
Unipolar Mode	0 to +|V_{REF}|		V	
Bipolar Mode	-|V_{REF}| to +|V_{REF}|		V	
LOGIC INPUTS				
RD (pin 15), CS (pin 16)				
V_{INH} Logic HIGH Input Voltage	+3.0	+3.0	V min	
V_{INL} Logic LOW Input Voltage	+0.8	+0.8	V max	
I_{IN} Input Current	1	10	μA max	V_{IN} = 0V, V_{DD}
C_{IN} Input Capacitance[5]	5	5	pF max	
CLK (pin 17)				
V_{INH} Logic HIGH Input Voltage	+3.0	+3.0	V min	
V_{INL} Logic LOW Input Voltage	+0.4	+0.4	V max	
I_{INH} Logic HIGH Input Current	+2	+3	mA max	During Conversion: $V_{IN(CLK)} \geq V_{INH(CLK)}$
I_{INL} Logic LOW Input Current	1	10	μA max	During Conversion: $V_{IN(CLK)} \leq V_{INL(CLK)}$ (see circuit of Fig. 7b if external clock operation is required).
LOGIC OUTPUTS				
BUSY (pin 14), DB_7 to DB_0 (pins 6-13)				
V_{OH} Output HIGH Voltage	+4.0	+4.0	V min	I_{SOURCE} = 40μA
V_{OL} Output LOW Voltage	+0.4	+0.8	V max	I_{SINK} = 1.6mA
I_{LKG} DB_7 to DB_0 Floating Stage Leakage	1	10	μA max	V_{OUT} = 0V or V_{DD}
Floating State Output Capacitance (DB_7 to DB_0)[5]	7	7	pF max	
Output Code	Unipolar Binary, Offset Binary			See Figs. 8a, 9a, 10a and 8b, 9b, 10b.
POWER REQUIREMENTS				
V_{DD}	+5	+5	V	±5% for specified performance.
I_{DD} (STANDBY)	5	5	mA max	A_{IN} = 0V, ADC in RESET condition.
I_{REF}	V_{REF} divided by 5kΩ		max	Conversion complete, prior to RESET.

PRICE ($) MODEL	1-24	25-99	100 up
AD7574JN	12.50	10.00	7.50
AD7574KN	15.00	12.00	9.00
AD7574AD	14.50	12.00	9.50
AD7574BD	17.00	14.00	11.00
AD7574AD/883B[6]	21.50	18.00	14.50
AD7574BD/883B[6]	24.00	20.00	16.00
AD7574SD	29.00	24.00	19.00
AD7574TD	34.00	28.00	22.00
AD7574SD/883B[6]	36.00	30.00	24.00
AD7574TD/883B[6]	41.00	34.00	27.00

Notes:
1. Temperature ranges as follows: JN, KN (0°C to +70°C)
   AD, BD (-25°C to +85°C)
   SD, TD (-55°C to +125°C)
2. Typical offset temperature coefficient is ±150μV/°C.
3. R_{BOFS}/R_{AIN} mismatch causes transfer function rotation about positive Full Scale. The effect is an offset and a gain term when using the circuit of Figure 9a, page 7.
4. Typical value, not guaranteed or subject to test.
5. Guaranteed but not tested.
6. Screening to MIL-STD-883 is available. /883B versions are 100% screened to method 5004 for a class B device. Final electrical tests are performed at +25°C and +85°C (AD, BD versions) or +25°C and +125°C (SD, TD versions).

Specifications subject to change without notice.

AC Specifications

(V_{DD} = +5V, C_{CLK} = 100pF, R_{CLK} = 180kΩ unless otherwise noted)

SYMBOL	SPECIFICATION	LIMIT at T_A = +25°C	LIMIT at T_A = T_{min}	LIMIT at T_A = T_{max}	CONDITIONS
STATIC RAM INTERFACE MODE (See Figure 1 and Table 1)					
t_{CS}	$\overline{CS}$ Pulse Width Requirement	100ns min	150ns min	150ns min	
t_{WSCS}	$\overline{RD}$ to $\overline{CS}$ Setup Time	0 min	0 min	0 min	
t_{CBPD}	$\overline{CS}$ to $\overline{BUSY}$ Propagation Delay	90ns typ 120ns max	70ns typ 120ns max	150ns typ 180ns max	$\overline{BUSY}$ Load = 20pF
		120ns typ 150ns max	100ns typ 150ns max	180ns typ 200ns max	$\overline{BUSY}$ Load = 100pF
t_{BSR}	$\overline{BUSY}$ to $\overline{RD}$ Setup Time	0 min	0 min	0 min	
t_{BSCS}	$\overline{BUSY}$ to $\overline{CS}$ Setup Time	0 min	0 min	0 min	
t_{RAD}	Data Access Time	120ns typ 150ns max	100ns typ 150ns max	180ns typ 220ns max	DB_0 - DB_7 Load = 20pF
		240ns typ 300ns max	220ns typ 300ns max	300ns typ 400ns max	DB_0 - DB_7 Load = 100pF
t_{RHD}	Data Hold Time	80ns typ 50ns min 120ns max	40ns typ 30ns min 80ns max	120ns typ 80ns min 180ns max	
t_{RHCS}	$\overline{CS}$ to $\overline{RD}$ Hold Time	250ns max	200ns max	500ns max	
t_{RESET}	Reset Time Requirement	3μs min	3μs min	3μs min	
$t_{CONVERT}$	Conversion Time using internal clock oscillator	See typical data of Figure 7a			
$t_{CONVERT}$	Conversion Time using external clock	15μs	15μs	15μs	f_{CLK} = 500kHz circuit of Figure 7b
ROM INTERFACE MODE (See Figure 2 and Table 2)					
t_{RAD}	Data Access Time	Same as RAM Mode			
t_{RHD}	Data Hold Time	Same as RAM Mode			
t_{WBPD}	$\overline{RD}$ HIGH to $\overline{BUSY}$ Propagation Delay	400ns typ 1.5μs max	350ns typ 1.0μs max	1μs typ 2.0μs max	$\overline{BUSY}$ Load = 20pF
t_{BSR}	$\overline{BUSY}$ to $\overline{RD}$ LOW Setup Time	$\overline{RD}$ can go LOW prior to $\overline{BUSY}$ = HIGH, but must not return HIGH until $\overline{BUSY}$ = HIGH. See Table 2			
$t_{CONVERT}$	Conversion Time using internal clock oscillator	See typical data of Figure 7a. Add 2μs to data shown in Figure 7a for ROM Mode			
SLOW – MEMORY INTERFACE MODE (See Figure 3 and Table 3)					
t_{CBPD}	$\overline{CS}$ to $\overline{BUSY}$ Propagation Delay	Same as RAM Mode			
t_{RESET}	Reset Time Requirement	Same as RAM Mode			
t_{RAD}	Data Access Time	Same as RAM Mode			
t_{RHD}	Data Hold Time	Same as RAM Mode			
$t_{CONVERT}$	Conversion Time	Same as RAM Mode			

ABSOLUTE MAXIMUM RATINGS

V_{DD} to A_{GND} . . . . . 0V, +7.0V
V_{DD} to D_{GND} . . . . . 0V, +7.0V
A_{GND} to D_{GND} . . . . . -0.3V, V_{DD}
Digital Input Voltage to D_{GND}
(pins 15 and 16) . . . . . -0.3V, +15.0V
Digital Output Voltage to D_{GND}
(pins 6 - 14) . . . . . -0.3V, V_{DD}
CLK Input Voltage (pin 17) to D_{GND} . . . . . -0.3V, V_{DD}
V_{REF} (pin 2) . . . . . ±20V
V_{BOFS} (pin 3) . . . . . ±20V
V_{AIN} (pin 4) . . . . . ±20V

Operating Temperature Range
JN, KN . . . . . 0°C to +70°C
AD, BD . . . . . -25°C to +85°C
SD, TD . . . . . -55°C to +125°C
Storage Temperature Range . . . . . -65°C to +150°C
Lead Temperature (soldering, 10 secs.) . . . . . +300°C
Power Dissipation (Package)
Plastic (suffix N)
to +70°C 670mW
Derate above +70°C by . . . . . 8.3mw/°C
Ceramic (suffix D)
to +75°C 450mW
Derate above +75°C by . . . . . 6mW/°C

TERMINOLOGY

RESOLUTION: Resolution is a measure of the *nominal* analog change required for a 1 - bit change in the A/D converter's digital output. While normally expressed in a number of bits, the analog resolution of an n - bit unipolar A/D converter is $(2^{-n})(V_{REF})$. Thus the AD7574, an 8 - bit A/D converter, can resolve analog voltages as small as $(1/256)(V_{REF})$ when operated in a unipolar mode. When operated in a bipolar mode, the resolution is $(1/128)(V_{REF})$. Resolution does not imply accuracy. *Usable* resolution is limited by the differential nonlinearity of the A/D converter.

RELATIVE ACCURACY: Relative accuracy is the deviation of the ADC's actual code transition points from a straight line drawn between the device's measured zero and measured full scale transition points. Relative accuracy, therefore, is a measure of code *position*.

DIFFERENTIAL NONLINEARITY: Differential nonlinearity in an ADC is a measure of the size of an analog voltage range associated with any digital output code. As such differential nonlinearity specifies code width (usable resolution). An ADC with a specified differential nonlinearity of ±n bits will exhibit codes ranging in width from 1LSB - nLSB to 1LSB + nLSB. A specified differential nonlinearity of less than ±1LSB guarantees no - missing - codes operation.

STATIC RAM INTERFACE MODE

Table 1 and Figure 1 show the truth table and timing requirements for AD7574 operation as a static RAM.

A convert start is initiated by executing a memory WRITE instruction to the address location occupied by the AD7574 (once conversion has started, subsequent memory WRITES have no effect). A data READ is performed by executing a memory READ instruction to the AD7574 address location.

$\overline{BUSY}$ must be HIGH before a data READ is attempted, i.e. the total delay between a convert start and a data READ must be at least as great as the AD7574 conversion time. The delay can be generated by inserting NOP instructions (or other program instructions) between the WRITE (start convert) and READ (read data) operations. Once $\overline{BUSY}$ is HIGH (conversion complete), a data READ is performed by executing a memory READ instruction to the address location occupied by the AD7574. The data readout is destructive, i.e. when RD returns HIGH, the converter is internally reset.

The RAM interface mode uses distinctly different commands to start conversion (memory WRITE) or read the data (memory READ). This is in contrast to the ROM mode where a memory READ causes a data READ *and* a conversion restart.

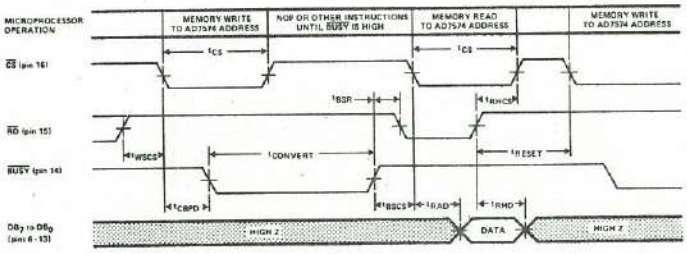


Figure 1. Static RAM Mode Timing Diagram

AD7574 INPUTS		AD7574 OUTPUTS		AD7574 OPERATION
$\overline{CS}$	$\overline{RD}$	$\overline{BUSY}$	DB_7 - DB_0	
L	H	H	HIGH Z	WRITE CYCLE (START CONVERT)
L	⊓	H	HIGH Z → DATA	READ CYCLE (DATA READ)
L	⊓	H	DATA → HIGH Z	RESET CONVERTER
H	X[1]	X	HIGH Z	NOT SELECTED
L	H	L	HIGH Z	NO EFFECT, CONVERTER BUSY
L	⊓	L	HIGH Z	NO EFFECT, CONVERTER BUSY
L	⊓	L	HIGH Z	NOT ALLOWED, CAUSES INCORRECT CONVERSION

Note 1: If $\overline{RD}$ goes LOW-to-HIGH, the ADC is internally reset, regardless of the state of $\overline{CS}$ or $\overline{BUSY}$.

Table 1. Truth Table, Static RAM Mode

ROM INTERFACE MODE

Table 2 and Figure 2 show the truth table and timing requirements for interfacing the AD7574 like Read Only Memory.

$\overline{CS}$ is held LOW and converter operation is controlled by the $\overline{RD}$ input. The AD7574 $\overline{RD}$ input is derived from the decoded device address. MEMRD should be used to enable the address decoder in 8080 systems. VMA should be used to enable the address decoder in 6800 systems. A data READ is initiated by executing a memory READ instruction to the AD7574 address location. The converter is automatically restarted when $\overline{RD}$ returns HIGH. As in the RAM mode, attempting a data READ before $\overline{BUSY}$ is HIGH will result in incorrect data being read.

The advantage of the ROM mode is its simplicity. The major disadvantage is that the data obtained is relatively poorly defined in time inasmuch as executing a data READ automatically starts a new conversion. This problem can be overcome by executing two READs separated by NO-OPS (or other program instructions) and using only the data obtained from the second READ.

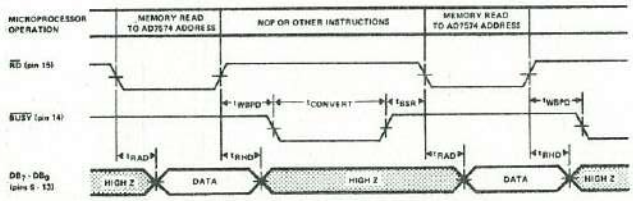


Figure 2. ROM Mode Timing Diagram ($\overline{CS}$ Held LOW)

AD7574 INPUTS		AD7574 OUTPUTS		AD7574 OPERATION
$\overline{CS}$	$\overline{RD}$	$\overline{BUSY}$	DB_7 - DB_0	
L	⊓	H	HIGH Z → DATA	DATA READ
L	⊓	⊓	DATA → HIGH Z	RESET AND START NEW CONVERSION
L	⊓	L	HIGH Z	NO EFFECT, CONVERTER BUSY
L	⊓	L	HIGH Z	NOT ALLOWED, CAUSES INCORRECT CONVERSION

Table 2. Truth Table, ROM Mode

SLOW-MEMORY INTERFACE MODE

Table 3 and Figure 3 show the truth table and timing requirements for interfacing the AD7574 as a slow-memory. This mode is intended for use with processors which can be forced into a WAIT state for at least 12μs (such as the 8080, 8085 and SC/MP). The major advantage of this mode is that it allows the μP to start conversion, WAIT, and then READ data with a single READ instruction.

In the slow-memory mode, $\overline{CS}$ and $\overline{RD}$ are tied together. It is suggested that the system ALE signal (8085 system) or SYNC signal (8080 system) be used to latch the address. The decoded device address is subsequently used to drive the AD7574 $\overline{CS}$ and $\overline{RD}$ inputs. $\overline{BUSY}$ is connected to the microprocessor READY input.

When the AD7574 is NOT addressed, the $\overline{CS}$ and $\overline{RD}$ inputs are HIGH. Conversion is initiated by executing a memory READ to the AD7574 address. $\overline{BUSY}$ subsequently goes LOW (forcing the μP READY input LOW) placing the μP in a WAIT state. When conversion is complete ($\overline{BUSY}$ is HIGH) the μP completes the memory READ.

Do not attempt to perform a memory WRITE in this mode, since three-state bus conflicts will arise.

(continued on page 5)

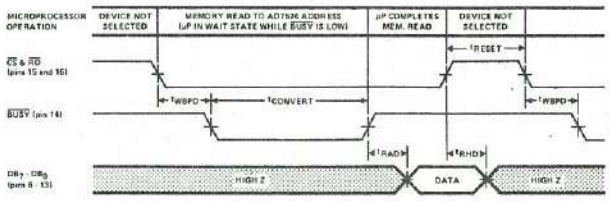


Figure 3. Slow Memory Mode Timing Diagram (CS and RD Tied Together)

AD7574 INPUTS	AD7574 OUTPUTS		AD7574 OPERATION
$\overline{CS}$ & $\overline{RD}$	$\overline{BUSY}$	DB_7 - DB_0	
H	H	HIGH Z	NOT SELECTED
⎍	H→L	HIGH Z	START CONVERSION
L	L	HIGH Z	CONVERSION IN PROGRESS, μP IN WAIT STATE
L	⎍	HIGH Z→DATA	CONVERSION COMPLETE, μP READS DATA
⎍	H	DATA→HIGH Z	CONVERTER RESET AND DESELECTED
H	H	HIGH Z	NOT SELECTED

Table 3. Truth Table, Slow Memory Mode

GENERAL CIRCUIT INFORMATION

BASIC CIRCUIT DESCRIPTION

The AD7574 uses the successive approximations technique to provide an 8-bit parallel digital output. The control logic was designed to provide easy interface to most microprocessors. Most applications require only passive clock components (R & C), a -10V reference, and +5V power.

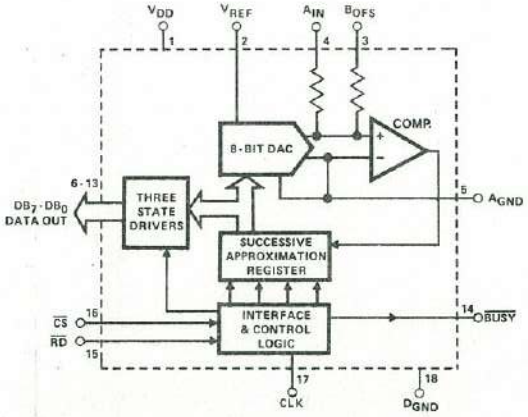


Figure 4. AD7574 Functional Diagram

Figure 4 shows the AD7574 functional diagram. Upon receipt of a start command either via the $\overline{CS}$ or $\overline{RD}$ pins (see pages 4 and 5 for Control Logic and Timing Details), $\overline{BUSY}$ goes low indicating conversion is in progress. Successive bits, starting with the most significant bit (MSB), are applied to the input of a DAC. The comparator determines whether the addition of each successive bit causes the DAC output to be greater than or less than the analog input, A_{IN}. If the sum of the DAC bits is less than A_{IN}, the trial bit is left ON, and the next smaller bit is tried. If the sum is greater than A_{IN}, the trial bit is turned OFF and the next smaller bit is tried.

Each successively smaller bit is tried and compared to A_{IN} in this manner until the least significant bit (LSB) decision has been made. At this time $\overline{BUSY}$ goes HIGH (conversion is complete) indicating the successive approximation register contains a valid representation of the analog input. The $\overline{RD}$ control (see page 4 for details) can then be exercised to activate the three-state buffers, placing data on the DB_0 - DB_7 data output pins. $\overline{RD}$ returning HIGH causes the clock oscillator to run for 1 cycle, providing an internal ADC reset (i.e. the SAR is loaded with code 10000000).

DAC CIRCUIT DETAILS

The current weighting D/A converter is a precision multiplying DAC. Figure 5 shows the functional diagram of the DAC as used in the AD7574. It consists of a precision Silicon Chromium thin film R/2R ladder network and 8 N-channel MOSFET switches operated in single-pole-double-throw.

The currents in each 2R shunt arm are binarily weighted, i.e. the current in the MSB arm is V_{REF} divided by 2R, in the second arm is V_{REF} divided by 4R, etc. Depending on the DAC logic input (A/D output) from the successive approximation register, the current in the individual shunt arms is steered either to A_{GND} or to the comparator summing point.

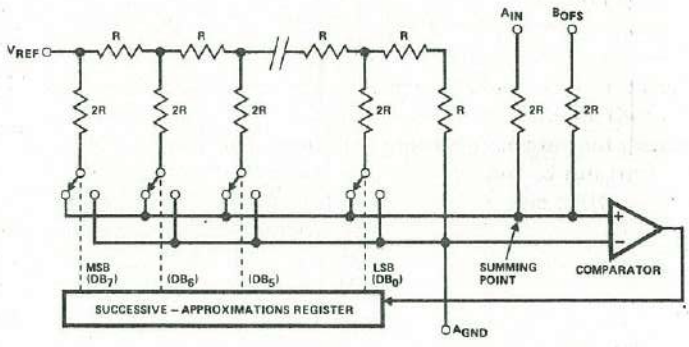


Figure 5. D/A Converter As Used In AD7574

OPERATING THE AD7574

APPLICATION HINTS

1. TIMING & CONTROL
   Failiure to observe the timing restrictions of figures 1, 2 or 3 may cause the AD7574 to change interface modes. For example, in the RAM mode, holding $\overline{CS}$ LOW too long after $\overline{RD}$ goes HIGH will cause a new convert start (i.e. the converter moved into the ROM mode).
2. LOGIC DEGLITCHING IN uP APPLICATIONS
   Unspecified states on the address bus (due to different rise and fall times on the address bus) can cause glitches at the AD7574 $\overline{CS}$ or $\overline{RD}$ terminals. These glitches can cause unwanted convert starts, reads, or resets. The best way to avoid glitches is to gate the address decoding logic with RD or WR (8080) or VMA (6800) when in the ROM or RAM mode. When in the slow-memory mode, the ALE (8085) or SYNC (8080) signal should be used to latch the address.
3. INPUT LOADING AT V_{REF}, A_{IN} AND B_{OFS}
   To prevent loading errors due to the finite input resistance at the V_{REF}, A_{IN} or B_{OFS} pins, low impedance driving sources must be used (i.e. op amp buffers or low output-Z reference).
4. RATIOMETRIC OPERATION
   Ratiometric performance is inherent to A/D converters such as the AD7574 which use a multiplying DAC weighting network. However, the user should recognize that comparator limitations such as offset voltage, input noise and gain will cause degradation of the transfer characteristics when operating with reference voltages less than -10V in magnitude.
5. OFFSET CORRECTION
   Offset error in the transfer characteristic can be trimmed by offsetting the buffer amplifier which drives the AD7574 A_{IN} pin (pin 4). This can be done either by summing a cancellation current into the amplifier's summing junction, or by tapping a voltage divider which sits between V_{DD} and V_{REF} and applying the tap voltage to the amplifier's positive input (an example of a resistive tap offset adjust is shown in Figure 10a where R_8, R_9 and R_{10} can be used to offset the ADC).
6. ANALOG AND DIGITAL GROUND
   It is recommended that A_{GND} and D_{GND} be connected locally to prevent the possibility of injecting noise into the AD7574. In systems where the A_{GND}-D_{GND} intertie is not local, connect back-to-back diodes (IN914 or equivalent) between the AD7574 A_{GND} and D_{GND} pins.
7. INITIALIZATION AFTER POWER – UP
   Execute a memory READ to the AD7574 address location, and subsequently ignore the data. The AD7574 is internally reset when reading out data, i.e. the data readout is destructive.

CLOCK OSCILLATOR

The AD7574 has an internal asynchronous clock oscillator which starts upon receipt of a convert start command, and ceases oscillating when conversion is complete.

The clock oscillator requires an external R and C as shown in figure 6. Nominal conversion time versus R_{CLK} and C_{CLK} is shown in Figure 7a. The curves shown in Figure 7a are applicable when operating in the RAM or slow-memory interface modes. When operating in the ROM interface mode, add 2μs to the typical conversion time values shown.

The AD7574 is guaranteed to provide transfer accuracy to published specifications for conversion times down to 15μs, as indicated by the unshaded region of Figure 7a. Conversion times faster than 15μs can cause transfer accuracy degradation.

OPERATION WITH EXTERNAL CLOCK

For applications requiring a conversion time close to or equal to 15μs, an external clock is recommended. Using an external clock precludes the possibility of converting faster than 15μs (which can cause transfer accuracy degradation) due to temperature drift – as may be the case when using the internal clock oscillator.

Figure 7b shows how the external clock must be connected. The $\overline{BUSY}$ output of the AD7574 is connected to the three-state enable input of a 74125 three-state buffer. R_1 is used as a pullup, and can be between 6kΩ and 100kΩ. A 500kHz clock will provide a conversion time of 15μs.

The external clock should be used only in the static-RAM or slow-memory interface mode, and *not* in the ROM mode.

Timing constraints for external clock operation are as follows:

STATIC RAM MODE

1. When initiating a conversion, $\overline{CS}$ should go LOW on a positive clock edge to provide optimum settling time for the MSB.
2. A data READ can be initiated any time after BUSY = 1.

SLOW-MEMORY MODE

1. When initiating a conversion, $\overline{CS}$ and $\overline{RD}$ should go LOW on a positive clock edge to provide optimum settling time for the MSB.

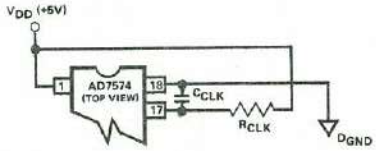


Figure 6. Connecting R_{CLK} and C_{CLK} To CLK Oscillator

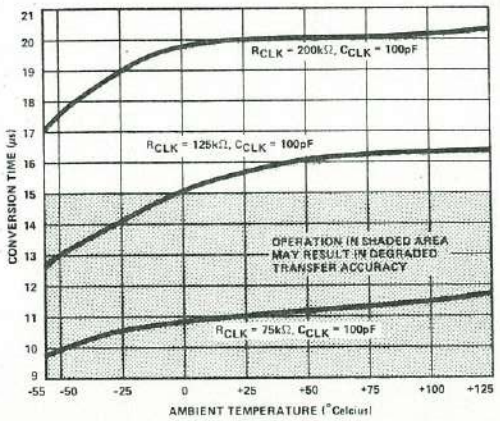


Figure 7a. Typical Conversion Time vs. Temperature For Different R_{CLK} and C_{CLK} (Applicable to RAM and Slow-Memory Modes. For ROM Mode add 2μs to values shown)

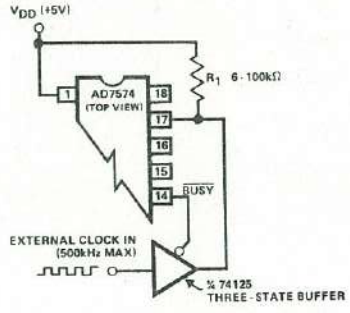


Figure 7b. External Clock Operation (Static RAM and Slow-Memory Mode)

(continued on page 7)

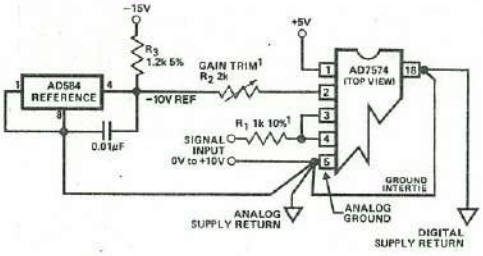
UNIPOLAR BINARY OPERATION

Figures 8a and 8b show the analog circuit connections and typical transfer characteristic for unipolar operation. An AD584 is used as the -10V reference.

Calibration is as follows:

OFFSET

Offset must be trimmed out in the signal conditioning circuitry used to drive the signal input terminals shown in Figure 8a. An example of an offset trim is shown in Figure 10a, where R_8, R_9 and R_{10} comprise a simple voltage tap which is applied to the amplifier's positive input.



Note 1: R_1 and R_2 can be omitted if gain trim is not required

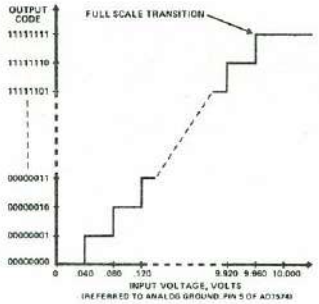
Figure 8a. AD7574 Unipolar (0V to +10V) Operation (Output Code is Straight Binary)

1. Apply -39.1mV (1 LSB) to the input of the buffer amplifier used to drive R_1 (i.e. +39.1mV at R_1).
2. While performing continuous conversions, adjust the offset potentiometer (described above) until DB_7 - DB_1 are LOW and the LSB (DB_0) flickers.

GAIN (FULL SCALE)

Offset adjustment must be performed before gain adjustment.

1. Apply -9.961V to the input of the buffer amplifier used to drive R_1 (i.e. +9.961V at R_1).
2. While performing continuous conversions, adjust trim pot R_2 until DB_7 - DB_1 are HIGH and the LSB (DB_0) flickers.



Note: Approximate bit weights are shown for illustration. Nominal bit weight for a -10V reference is ≈ 39.1mV

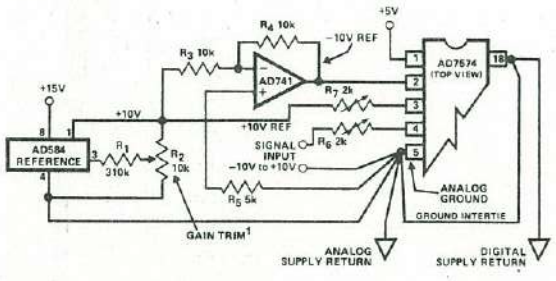
Figure 8b. Nominal Transfer Characteristic For Unipolar Circuit of Figure 8a

BIPOLAR (OFFSET BINARY) OPERATION

Figures 9a and 9b illustrate the analog circuitry and transfer characteristic for bipolar operation. Output coding is offset binary. As in unipolar operation, offset correction can be performed at the buffer amplifier used to drive the signal input terminals of Figure 9a (Resistors R_8, R_9 and R_{10} in Figure 10a show how offset trim can be done at the buffer amplifier).

Calibration is as follows:

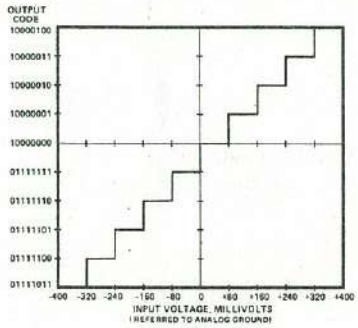
1. Adjust R_6 and R_7 for minimum resistance across the potentiometers.
2. Apply +10.000V to the buffer amplifier used to drive the signal input (i.e. -10.000V at R_6).
3. While performing continuous conversions, trim R_6 or R_7 (whichever required) until DB_7 - DB_1 are LOW and the LSB (DB_0) flickers.



Note 1: R_1 and R_2 can be omitted if gain trim is not required

Figure 9a. AD7574 Bipolar (-10V to +10V) Operation (Output Code is Offset Binary)

4. Apply 0V to the buffer amplifier used to drive the signal input terminals.
5. Doing continuous conversions, trim the offset circuit of the buffer amplifier until the ADC output code flickers between 01111111 and 10000000.
6. Apply +10.000V to the input of the buffer amplifier (i.e. -10.000V as applied to R_6).
7. Doing continuous conversions, trim R_2 until DB_7 - DB_1 are LOW and the LSB (DB_0) flickers.
8. Apply -9.922V to the input of the buffer amplifier (i.e. +9.922V at the input side of R_6).
9. If the ADC output code is not 11111110 ±1 bit, repeat the calibration procedure.



Note: Approximate bit weights are shown for illustration. Nominal bit weight for ± 10V full scale is ≈ 78.1mV

Figure 9b. Nominal Transfer Characteristic Around Major Carry for Bipolar Circuit of Figure 9a

(continued on page 8)

OPERATING THE AD7574

(continued from page 7)

BIPOLAR (COMPLEMENTARY OFFSET BINARY) OPERATION

Figure 10a shows the analog connections for complementary offset binary operation. The typical transfer characteristic is shown in Figure 10b. In this bipolar mode, the ADC is fooled into believing it is operated in a unipolar mode - i.e. the +10V to -10V analog input is conditioned into a 0 to +10V signal range. R_2 is the gain adjust, while R_9 is the offset adjust.

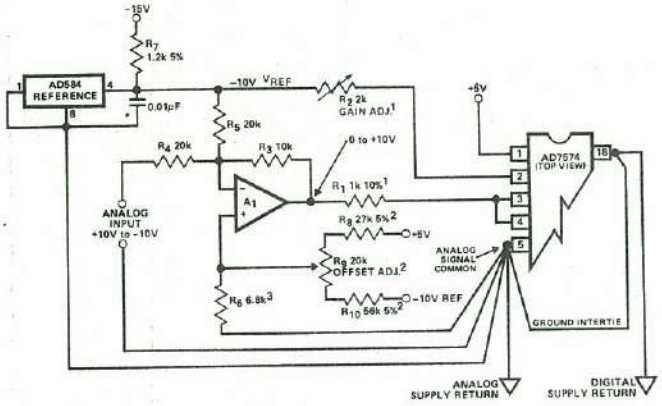
Calibration is as follows (adjust offset before gain):

OFFSET

1. Apply OV to the analog input shown in Figure 10a.
2. While performing continuous conversions, adjust R_9 until the converter output flickers between codes 01111111 and 10000000.

GAIN (FULL SCALE)

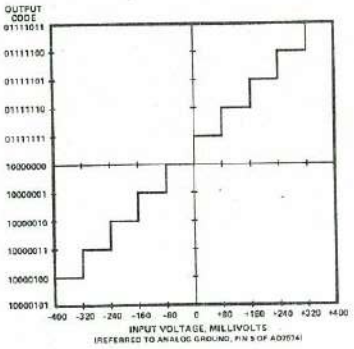
1. Apply -9.922V across the analog input terminals shown in Figure 10a.
2. While performing continuous conversions, adjust R_2 until DB_7 - DB_1 are HIGH and the LSB (DB_0) flickers between HIGH and LOW.



Notes:

1. R_1 and R_2 can be omitted if gain trim is not required
2. R_8, R_9 and R_{10} can be omitted if offset trim is not required
3. $R_6 \| R_8 \| R_{10} = 5k\Omega$. If R_8, R_9 and R_{10} not used, make $R_6 = 5k\Omega$

Figure 10a. AD7574 Bipolar Operation (-10V to +10V) (Output Code is Complementary Offset Binary)

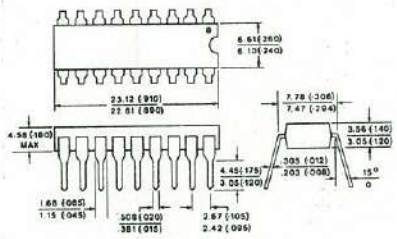


Note: Approximate bit weights are shown for illustration. Nominal bit weight for ±10V full scale is ≈78.1mV

Figure 10b. Nominal Transfer Characteristic Around Major Carry for Bipolar Circuit of Figure 10a

MECHANICAL INFORMATION

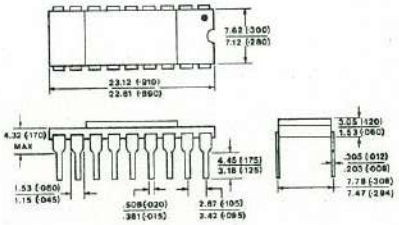
18 PIN PLASTIC DIP



Notes:

1. Lead no.1 identified by dot or notch.
2. Dimensions in mm (in.).
3. Leads are solder plated KOVAR or ALLOY 42.

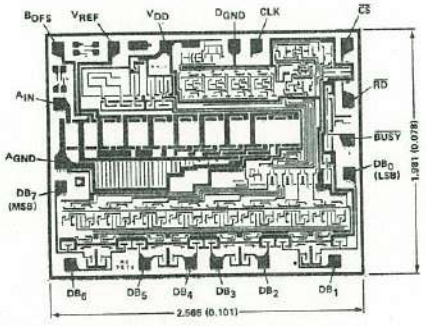
18 PIN CERAMIC DIP



Notes:

1. Lead no. 1 identified by dot or notch.
2. Leads are gold plate (50µin. min.) over Nickel (100µin. nominal). Base material is KOVAR or ALLOY 42.
3. Cavity lid is electrically isolated.

BONDING DIAGRAM



Notes:

1. Bond D_{GND} first to minimize ESD hazard.
2. Die dimensions are in mm. (in.), and may vary from nominal shown on layout by ±0.076mm (±0.003in.).
3. Die thickness is 0.508mm ±0.025mm (0.020in. ±0.001in.).
4. Gold backing is not available.
5. Passivation covers all topside surface area except bonding pads, test pads and scribe lines.
6. Surface matallization is Al, 10kÅ min.
7. Bonding pads are 0.102mm x 0.102mm (0.004in. x 0.004in.). Passivation window is 0.089mm x 0.089mm min. (0.0035in. x 0.0035in. min.).

Appendix C

Am9511A Data Sheet

Am9511A

Arithmetic Processor
Advanced Micro Devices
Advanced MOS/LSI

DISTINCTIVE CHARACTERISTICS

- Replaces Am9511
- Fixed point 16 and 32 bit operations
- Floating point 32 bit operations
- Binary data formats
- Add, Subtract, Multiply and Divide
- Trigonometric and inverse trigonometric functions
- Square roots, logarithms, exponentiation
- Float to fixed and fixed to float conversions
- Stack-oriented operand storage
- DMA or programmed I/O data transfers
- End signal simplifies concurrent processing
- Synchronous/Asynchronous operations
- General purpose 8-bit data bus interface
- Standard 24 pin package
- +12 volt and +5 volt power supplies
- Advanced N-channel silicon gate MOS technology
- 100% MIL-STD-883 reliability assurance testing

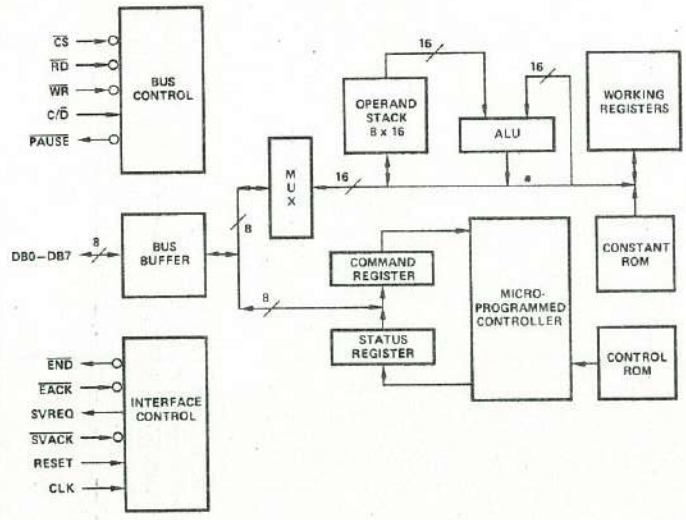
GENERAL DESCRIPTION

The Am9511A Arithmetic Processing Unit (APU) is a monolithic MOS/LSI device that provides high performance fixed and floating point arithmetic and a variety of floating point trigonometric and mathematical operations. It may be used to enhance the computational capability of a wide variety of processor-oriented systems.

All transfers, including operand, result, status and command information, take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack and a command is issued to perform operations on the data in the stack. Results are then available to be retrieved from the stack, or additional commands may be entered.

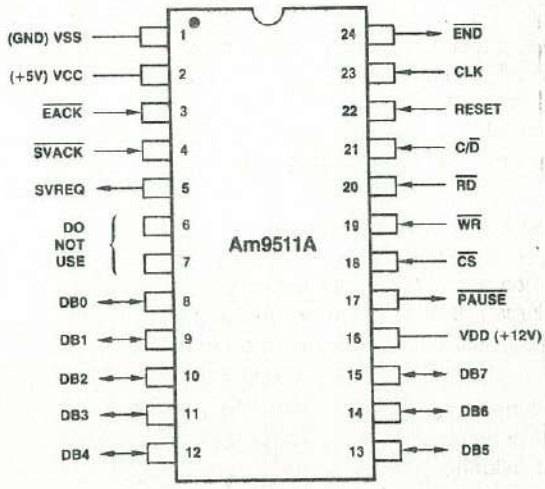
Transfers to and from the APU may be handled by the associated processor using conventional programmed I/O, or may be handled by a direct memory access controller for improved performance. Upon completion of each command, the APU issues an end of execution signal that may be used as an interrupt by the CPU to help coordinate program execution.

BLOCK DIAGRAM



MOS-046

CONNECTION DIAGRAM
Top View



Pin 1 is marked for orientation.

MOS-04

ORDERING INFORMATION

Package Type	Ambient Temperature	Maximum Clock Frequency	
		2MHz	3MHz
Hermetic DIP	$0°C \leq T_A \leq +70°C$	Am9511ADC	Am9511A-1DC
	$-55°C \leq T_A \leq +125°C$	Am9511ADM	Am9511A-1DM

INTERFACE SIGNAL DESCRIPTION

VCC: +5V Power Supply
VDD: +12V Power Supply
VSS: Ground

CLK (Clock, Input)

An external timing source connected to the CLK input provides the necessary clocking. The CLK input can be asynchronous to the $\overline{RD}$ and $\overline{WR}$ control signals.

RESET (Reset, Input)

A HIGH on this input causes initialization. Reset terminates any operation in progress, and clears the status register to zero. The internal stack pointer is initialized and the contents of the stack may be affected but the command register is not affected by the reset operation. After a reset the $\overline{END}$ output will be HIGH, and the SVREQ output will be LOW. For proper initialization, the RESET input must be HIGH for at least five CLK periods following stable power supply voltages and stable clock.

C/$\overline{D}$ (Command/Data Select, Input)

The C/$\overline{D}$ input together with the $\overline{RD}$ and $\overline{WR}$ inputs determines the type of transfer to be performed on the data bus as follows:

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Function
L	H	L	Push data byte into the stack
L	L	H	Pop data byte from the stack
H	H	L	Enter command byte from the data bus
H	L	H	Read Status
X	L	L	Undefined

L = LOW
H = HIGH
X = DON'T CARE

$\overline{END}$ (End of Execution, Output)

A LOW on this output indicates that execution of the current command is complete. This output will be cleared HIGH by activating the $\overline{EACK}$ input LOW or performing any read or write operation or device initialization using the RESET. If $\overline{EACK}$ is tied LOW, the $\overline{END}$ output will be a pulse (see $\overline{EACK}$ description). This is an open drain output and requires a pull up to +5V.

Reading the status register while a command execution is in progress is allowed. However any read or write operation clears the flip-flop that generates the $\overline{END}$ output. Thus such continuous reading could conflict with internal logic setting the $\overline{END}$ flip-flop at the completion of command execution.

$\overline{EACK}$ (End Acknowledge, Output)

This input when LOW makes the $\overline{END}$ output go LOW. As mentioned earlier HIGH on the $\overline{END}$ output signals completion of a command execution. The $\overline{END}$ output signal is derived from an internal flip-flop which is clocked at the completion of a command. This flip-flop is clocked to the reset state when $\overline{EACK}$ is LOW. Consequently, if the $\overline{EACK}$ is tied LOW, the $\overline{END}$ output will be a pulse that is approximately one CLK period wide.

SVREQ (Service Request, Output)

A HIGH on this output indicates completion of a command. In this sense this output is same as the $\overline{END}$ output. However, whether the SVREQ output will go HIGH at the completion of a command or not is determined by a service request bit in the command register. This bit must be 1 for SVREQ to go HIGH. The SVREQ can be cleared (i.e., go LOW) by activating the $\overline{SVACK}$ input LOW or initializing the device using the RESET. Also, the SVREQ will be automatically cleared after completion of any command that has the service request bit as 0.

$\overline{SVACK}$ (Service Acknowledge, Input)

A LOW on this input activates the reset input of the flip-flop generating the SVREQ output. If the $\overline{SVACK}$ input is permanently tied LOW, it will conflict with the internal setting of the flip-flop to generate the SVREQ output. Thus the SVREQ indication cannot be relied upon if the $\overline{SVACK}$ is tied LOW.

DB0-DB7 (Bidirectional Data Bus, Input/Output)

These eight bidirectional lines are used to transfer command, status and operand information between the device and the host processor. DB0 is the least significant and DB7 is the most significant bit position. HIGH on the data bus line corresponds to 1 and LOW corresponds to 0.

When pushing operands on the stack using the data bus, the least significant byte must be pushed first and most significant byte last. When popping the stack to read the result of an operation, the most significant byte will be available on the data bus first and the least significant byte will be the last. Moreover, for pushing operands and popping results, the number of transactions must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer will not be aligned properly. The Am9511A single precision format requires 2 bytes, double precision and floating-point formats require 4 bytes.

$\overline{CS}$ (Chip Select, Input)

This input must be LOW to accomplish any read or write operation to the Am9511A.

To perform a write operation data is presented on DB0 through DB7 lines, C/$\overline{D}$ is driven to an appropriate level and the $\overline{CS}$ input is made LOW. However, actual writing into the Am9511A cannot start until $\overline{WR}$ is made LOW. After initiating the write operation by a $\overline{WR}$ HIGH to LOW transition, the $\overline{PAUSE}$ output will go LOW momentarily (TPPWW).

The $\overline{WR}$ input can go HIGH after $\overline{PAUSE}$ goes HIGH. The data lines, C/$\overline{D}$ input and the $\overline{CS}$ input can change when appropriate hold time requirements are satisfied. See write timing diagram for details.

To perform a read operation an appropriate logic level is established on the C/$\overline{D}$ input and $\overline{CS}$ is made LOW. The Read operation does not start until the $\overline{RD}$ input goes LOW. $\overline{PAUSE}$ will go LOW for a period of TPPWR. When $\overline{PAUSE}$ goes back HIGH again, it indicates that read operation is complete and the required information is available on the DB0 through DB7 lines. This information will remain on the data lines as long as $\overline{RD}$ input is LOW. The $\overline{RD}$ input can return HIGH anytime after $\overline{PAUSE}$ goes HIGH. The $\overline{CS}$ input and C/$\overline{D}$ inputs can change anytime after $\overline{RD}$ returns HIGH. See read timing diagram for details.

$\overline{RD}$ (Read, Input)

A LOW on this input is used to read information from an internal location and gate that information on to the data bus. The $\overline{CS}$ input must be LOW to accomplish the read operation. The C/$\overline{D}$ input determines what internal location is of interest. See C/$\overline{D}$, $\overline{CS}$ input descriptions and read timing diagram for details. If the END output was LOW, performing any read operation will make the $\overline{END}$ output go HIGH after the HIGH to LOW transition of the $\overline{RD}$ input (assuming $\overline{CS}$ is LOW).

$\overline{WR}$ (Write, Input)

A LOW on this input is used to transfer information from the data bus into an internal location. The $\overline{CS}$ must be LOW to accomplish the write operation. The C/$\overline{D}$ determines which internal location is to be written. See C/$\overline{D}$, $\overline{CS}$ input descriptions and write timing diagram for details.

If the $\overline{END}$ output was LOW, performing any write operation will make the $\overline{END}$ output go HIGH after the LOW to HIGH transition of the $\overline{WR}$ input (assuming $\overline{CS}$ is LOW).

$\overline{PAUSE}$ (Pause, Output)

This output is a handshake signal used while performing read or write transactions with the Am9511A. A LOW at this output indicates that the Am9511A has not yet completed its information transfer with the host over the data bus. During a read operation, after $\overline{CS}$ went LOW, the $\overline{PAUSE}$ will become LOW shortly (TRP) after $\overline{RD}$ goes LOW. $\overline{PAUSE}$ will return high only after the data bus contains valid output data. The $\overline{CS}$ and $\overline{RD}$ should remain LOW when $\overline{PAUSE}$ is LOW. The RD may go high anytime after $\overline{PAUSE}$ goes HIGH. During a write operation, after $\overline{CS}$ went LOW, the $\overline{PAUSE}$ will be LOW for a very short duration (TPPWN) after $\overline{WR}$ goes LOW. Since the minimum of TPPWW is 0, the $\overline{PAUSE}$ may not go LOW at all for fast devices. $\overline{WR}$ may go HIGH anytime after $\overline{PAUSE}$ goes HIGH.

FUNCTIONAL DESCRIPTION

Major functional units of the Am9511A are shown in the block diagram. The Am9511A employs a microprogram controlled stack oriented architecture with 16-bit wide data paths.

The Arithmetic Logic Unit (ALU) receives one of its operands from the Operand Stack. This stack is an 8-word by 16-bit 2-port memory with last in-first out (LIFO) attributes. The second operand to the ALU is supplied by the internal 16-bit bus. In addition to supplying the second operand, this bidirectional bus also carries the results from the output of the ALU when required. Writing into the Operand Stack takes place from this internal 16-bit bus when required. Also connected to this bus are the Constant ROM and Working Registers. The ROM provides the required constants to perform the mathematical operations (Chebyshev Algorithms) while the Working Registers provide storage for the intermediate values during command execution.

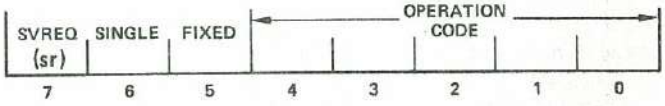
Communication between the external world and the Am9511A takes place on eight bidirectional input/output lines DB0 through DB7 (Data Bus). These signals are gated to the internal eight-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication between the internal eight and sixteen-bit buses. The Status Register and Command Register are also accessible via the eight-bit bus.

The Am9511A operations are controlled by the microprogram contained in the Control ROM. The Program Counter supplies the microprogram addresses and can be partially loaded from the Command Register. Associated with the Program Counter is the Subroutine Stack where return addresses are held during subroutine calls in the microprogram. The Microinstruction Register holds the current microinstruction being executed. This register facilitates pipelined microprogram execution. The Instruction Decode logic generates various internal control signals needed for the Am9511A operation.

The Interface Control logic receives several external inputs and provides handshake related outputs to facilitate interfacing the Am9511A to microprocessors.

COMMAND FORMAT

Each command entered into the Am9511A consists of a single 8-bit byte having the format illustrated below:



Bits 0-4 select the operation to be performed as shown in the table. Bits 5-6 select the data format for the operation. If bit 5 is a 1, a fixed point data format is specified. If bit 5 is a 0, floating point format is specified. Bit 6 selects the precision of the data to be operated on by fixed point commands (if bit 5 = 0, bit 6 must be 0). If bit 6 is a 1, single-precision (16-bit) operands are indicated; if bit 6 is a 0, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is a 1, the service request output (SVREQ) will go high at the conclusion of the command and will remain high until reset by a low level on the service acknowledge pin ($\overline{SVACK}$) or until completion of execution of a succeeding command where bit 7 is 0. Each command issued to the Am9511A requests post execution service based upon the state of bit 7 in the command byte. When bit 7 is a 0, SVREQ remains low.

COMMAND SUMMARY

Command Code								Command Mnemonic	Command Description
7	6	5	4	3	2	1	0		
FIXED-POINT 16-BIT									
sr	1	1	0	1	1	0	0	SADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	0	1	SSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	1	0	SMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	1	1	1	0	1	1	0	SMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	1	1	0	1	1	1	1	SDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FIXED-POINT 32-BIT									
sr	0	1	0	1	1	0	0	DADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	0	1	DSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	1	0	DMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	0	1	1	0	1	1	0	DMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	0	1	0	1	1	1	1	DDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FLOATING-POINT 32-BIT									
sr	0	0	1	0	0	0	0	FADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	0	1	FSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	0	FMUL	Multiply NOS by TOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	1	FDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
DERIVED FLOATING-POINT FUNCTIONS									
sr	0	0	0	0	0	0	1	SQRT	Square Root of TOS. Result in TOS.
sr	0	0	0	0	0	1	0	SIN	Sine of TOS. Result in TOS.
sr	0	0	0	0	0	1	1	COS	Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	0	0	TAN	Tangent of TOS. Result in TOS.
sr	0	0	0	0	1	0	1	ASIN	Inverse Sine of TOS. Result in TOS.
sr	0	0	0	0	1	1	0	ACOS	Inverse Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	1	1	ATAN	Inverse Tangent of TOS. Result in TOS.
sr	0	0	0	1	0	0	0	LOG	Common Logarithm (base 10) of TOS. Result in TOS.
sr	0	0	0	1	0	0	1	LN	Natural Logarithm (base e) of TOS. Result in TOS.
sr	0	0	0	1	0	1	0	EXP	Exponential (e^X) of TOS. Result in TOS.
sr	0	0	0	1	0	1	1	PWR	NOS raised to the power in TOS. Result in NOS. Pop Stack.
DATA MANIPULATION COMMANDS									
sr	0	0	0	0	0	0	0	NOP	No Operation
sr	0	0	1	1	1	1	1	FIXS	Convert TOS from floating point to 16-bit fixed point format.
sr	0	0	1	1	1	1	0	FIXD	Convert TOS from floating point to 32-bit fixed point format.
sr	0	0	1	1	1	0	1	FLTS	Convert TOS from 16-bit fixed point to floating point format.
sr	0	0	1	1	1	0	0	FLTD	Convert TOS from 32-bit fixed point to floating point format.
sr	1	1	1	0	1	0	0	CHSS	Change sign of 16-bit fixed point operand on TOS.
sr	0	1	1	0	1	0	0	CHSD	Change sign of 32-bit fixed point operand on TOS.
sr	0	0	1	0	1	0	1	CHSF	Change sign of floating point operand on TOS.
sr	1	1	1	0	1	1	1	PTOS	Push 16-bit fixed point operand on TOS to NOS (Copy)
sr	0	1	1	0	1	1	1	PTOD	Push 32-bit fixed point operand on TOS to NOS. (Copy)
sr	0	0	1	0	1	1	1	PTOF	Push floating point operand on TOS to NOS. (Copy)
sr	1	1	1	1	0	0	0	POPS	Pop 16-bit fixed point operand from TOS. NOS becomes TOS.
sr	0	1	1	1	0	0	0	POPD	Pop 32-bit fixed point operand from TOS. NOS becomes TOS.
sr	0	0	1	1	0	0	0	POPF	Pop floating point operand from TOS. NOS becomes TOS.
sr	1	1	1	1	0	0	1	XCHS	Exchange 16-bit fixed point operands TOS and NOS.
sr	0	1	1	1	0	0	1	XCHD	Exchange 32-bit fixed point operands TOS and NOS.
sr	0	0	1	1	0	0	1	XCHF	Exchange floating point operands TOS and NOS.
sr	0	0	1	1	0	1	0	PUPI	Push floating point constant "π" onto TOS. Previous TOS becomes NOS.

NOTES:

1. TOS means Top of Stack. NOS means Next on Stack.
2. AMD Application Brief "Algorithm Details for the Am9511A APU" provides detailed descriptions of each command function, including data ranges, accuracies, stack configurations, etc.
3. Many commands destroy one stack location (bottom of stack) during development of the result. The derived functions may destroy several stack locations. See Application Brief for details.
4. The trigonometric functions handle angles in radians, not degrees.
5. No remainder is available for the fixed-point divide functions.
6. Results will be undefined for any combination of command coding bits not specified in this table.

COMMAND INITIATION

After properly positioning the required operands on the stack, a command may be issued. The procedure for initiating a command execution is as follows:

1. Enter the appropriate command on the DB0-DB7 lines.
2. Establish HIGH on the C/$\overline{D}$ input.
3. Establish LOW on the $\overline{CS}$ input.
4. Establish LOW on the $\overline{WR}$ input after an appropriate set up time (see timing diagrams).
5. Sometime after the HIGH to LOW level transition of $\overline{WR}$ input, the $\overline{PAUSE}$ output will become LOW. After a delay of TPPWW, it will go HIGH to acknowledge the write operation. The WR input can return to HIGH anytime after $\overline{PAUSE}$ going HIGH. The DB0-DB7, C/$\overline{D}$ and $\overline{CS}$ inputs are allowed to change after the hold time requirements are satisfied (see timing diagram).

An attempt to issue a new command while the current command execution is in progress is allowed. Under these circumstances, the $\overline{PAUSE}$ output will not go HIGH until the current command execution is completed.

OPERAND ENTRY

The Am9511A commands operate on the operands located at the TOS and NOS and results are returned to the stack at NOS and then popped to TOS. The operands required for the Am9511A are one of three formats – single precision fixed-point (2 bytes), double precision fixed-point (4 bytes) or floating-point (4 bytes). The result of an operation has the same format as the operands except for float to fix or fix to float commands.

Operands are always entered into the stack least significant byte first and most significant byte last. The following procedure must be followed to enter operands onto the stack:

1. The lower significant operand byte is established on the DB0-DB7 lines.
2. A LOW is established on the C/$\overline{D}$ input to specify that data is to be entered into the stack.
3. The $\overline{CS}$ input is made LOW.
4. After appropriate set up time (see timing diagrams), the $\overline{WR}$ input is made LOW. The $\overline{PAUSE}$ output will become LOW.
5. Sometime after this event, the $\overline{PAUSE}$ will return HIGH to indicate that the write operation has been acknowledged.
6. Anytime after the $\overline{PAUSE}$ output goes HIGH the $\overline{WR}$ input can be made HIGH. The DB0-DB7, C/$\overline{D}$ and $\overline{CS}$ inputs can change after appropriate hold time requirements are satisfied (see timing diagrams).

The above procedure must be repeated until all bytes of the operand are pushed into the stack. It should be noted that for single precision fixed-point operands 2 bytes should be pushed and 4 bytes must be pushed for double precision fixed-point or floating-point. Not pushing all the bytes of a quantity will result in byte pointer misalignment.

The Am9511A stack can accommodate 8 single precision fixed-point quantities or 4 double precision fixed-point or floating-point quantities. Pushing more quantities than the capacity of the stack will result in loss of data which is usual with any LIFO stack.

DATA REMOVAL

Result from an operation will be available at the TOS. Results can be transferred from the stack to the data bus by reading the stack. When the stack is popped for results, the most significant byte is available first and the least significant byte last. A result is always of the same precision as the operands that produced it except for format conversion commands. Thus when the result is taken from the stack, the total number of bytes popped out should be appropriate with the precision – single precision results are 2 bytes and double precision and floating-point results are 4 bytes. The following procedure must be used for reading the result from the stack:

1. A LOW is established on the C/$\overline{D}$ input.
2. The $\overline{CS}$ input is made LOW.
3. After appropriate set up time (see timing diagrams), the $\overline{RD}$ input is made LOW. The $\overline{PAUSE}$ will become LOW.
4. Sometime after this, $\overline{PAUSE}$ will return HIGH indicating that the data is available on the DB0-DB7 lines. This data will remain on the DB0-DB7 lines as long as the $\overline{RD}$ input remains LOW.
5. Anytime after $\overline{PAUSE}$ goes HIGH, the $\overline{RD}$ input can return HIGH to complete transaction.
6. The $\overline{CS}$ and C/$\overline{D}$ inputs can change after appropriate hold time requirements are satisfied (see timing diagram).
7. Repeat this procedure until all bytes appropriate for the precision of the result are popped out.

Reading of the stack does not alter its data; it only adjusts the byte pointer. If more data is popped than the capacity of the stack, the internal byte pointer will wrap around and older data will be read again, consistent with the LIFO stack.

STATUS READ

The Am9511A status register can be read without any regard to whether a command is in progress or not. The only implication that has to be considered is the effect this might have on the END output discussed in the signal descriptions.

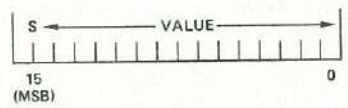
The following procedure must be followed to accomplish status register reading.

1. Establish HIGH on the C/$\overline{D}$ input.
2. Establish LOW on the $\overline{CS}$ input.
3. After appropriate set up time (see timing diagram) $\overline{RD}$ input is made LOW. The $\overline{PAUSE}$ will become LOW.
4. Sometime after the HIGH to LOW transition of $\overline{RD}$ input, the $\overline{PAUSE}$ will become HIGH indicating that status register contents are available on the DB0-DB7 lines. The status data will remain on DB0-DB7 as long as $\overline{RD}$ input is LOW.
5. The $\overline{RD}$ input can be returned HIGH anytime after $\overline{PAUSE}$ goes HIGH.
6. The C/$\overline{D}$ input and $\overline{CS}$ input can change after satisfying appropriate hold time requirements (see timing diagram).

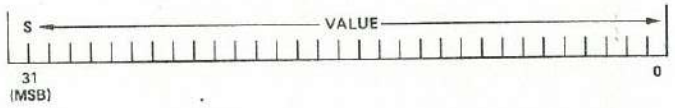
DATA FORMATS

The Am9511A Arithmetic Processing Unit handles operands in both fixed-point and floating-point formats. Fixed-point operands may be represented in either single (16-bit operands) or double precision (32-bit operands), and are always represented as binary, two's complement values.

16-BIT FIXED-POINT FORMAT



32-BIT FIXED-POINT FORMAT



The sign (positive or negative) of the operand is located in the most significant bit (MSB). Positive values are represented by a sign bit of zero (S = 0). Negative values are represented by the two's complement of the corresponding positive value with a sign bit equal to 1 (S = 1). The range of values that may be accommodated by each of these formats is −32,768 to +32,767 for single precision and −2,147,483,648 to +2,147,483,647 for double precision.

Floating point binary values are represented in a format that permits arithmetic to be performed in a fashion analogous to operations with decimal values expressed in scientific notation.

$$(5.83 \times 10^{2})(8.16 \times 10^{1}) = (4.75728 \times 10^{4})$$

In the decimal system, data may be expressed as values between 0 and 10 times 10 raised to a power that effectively shifts the implied decimal point right or left the number of places necessary to express the result in conventional form (e.g., 47,572.8). The value-portion of the data is called the mantissa. The exponent may be either negative or positive.

The concept of floating point notation has both a gain and a loss associated with it. The gain is the ability to represent the significant digits of data with values spanning a large dynamic range limited only by the capacity of the exponent field. For example, in decimal notation if the exponent field is two digits wide, and the mantissa is five digits, a range of values (positive or negative) from 1.0000×10^{-99} to $9.9999 \times 10^{+99}$ can be accommodated. The loss is that only the significant digits of the value can be represented. Thus there is no distinction in this representation between the values 123451 and 123452, for example, since each would be expressed as: 1.2345×10^{5}. The sixth digit has been discarded. In most applications where the dynamic range of values to be represented is large, the loss of significance, and hence accuracy of results, is a minor consideration. For greater precision a fixed point format could be chosen, although with a loss of potential dynamic range.

The Am9511 is a binary arithmetic processor and requires that floating point data be represented by a fractional mantissa value between .5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

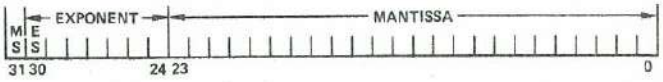
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

For example, the value 100.5 expressed in this form is 0.11001001×2^{7}. The decimal equivalent of this value may be computed by summing the components (powers of two) of the mantissa and then multiplying by the exponent as shown below:

$$\begin{aligned}\text{value} &= (2^{-1} + 2^{-2} + 2^{-5} + 2^{-8}) \times 2^{7} \\ &= (0.5 + 0.25 + 0.03125 + 0.00290625) \times 128 \\ &= 0.78515625 \times 128 \\ &= 100.5\end{aligned}$$

FLOATING POINT FORMAT

The format for floating-point values in the Am9511A is given below. The mantissa is expressed as a 24-bit (fractional) value; the exponent is expressed as an unbiased two's complement 7-bit value having a range of −64 to +63. The most significant bit is the sign of the mantissa (0 = positive, 1 = negative), for a total of 32 bits. The binary point is assumed to be to the left of the most significant mantissa bit (bit 23). All floating-point data values must be normalized. Bit 23 must be equal to 1, except for the value zero, which is represented by all zeros.



The range of values that can be represented in this format is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{18})$ and zero.

STATUS REGISTER

The Am9511A contains an eight bit status register with the following bit assignments:

BUSY	SIGN	ZERO	ERROR CODE				CARRY
7	6	5	4	3	2	1	0

BUSY: Indicates that Am9511A is currently executing a command (1 = Busy).

SIGN: Indicates that the value on the top of stack is negative (1 = Negative).

ZERO: Indicates that the value on the top of stack is zero (1 = Value is zero).

ERROR CODE: This field contains an indication of the validity of the result of the last operation. The error codes are:

- 0000 – No error
- 1000 – Divide by zero
- 0100 – Square root or log of negative number
- 1100 – Argument of inverse sine, cosine, or e^{x} too large
- XX10 – Underflow
- XX01 – Overflow

CARRY: Previous operation resulted in carry or borrow from most significant bit. (1 = Carry/Borrow, 0 = No Carry/No Borrow)

If the BUSY bit in the status register is a one, the other status bits are not defined; if zero, indicating not busy, the operation is complete and the other status bits are defined as given above.

Table 1.

Command Mnemonic	Hex Code (sr = 1)	Hex Code (sr = 0)	Execution Cycles	Summary Description
16-BIT FIXED-POINT OPERATIONS				
SADD	EC	6C	16-18	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	ED	6D	30-32	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	EE	6E	84-94	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
SMUU	F6	76	80-98	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
SDIV	EF	6F	84-94	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FIXED-POINT OPERATIONS				
DADD	AC	2C	20-22	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	AD	2D	38-40	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	AE	2E	194-210	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
DMUU	B6	36	182-218	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
DDIV	AF	2F	196-210	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FLOATING-POINT PRIMARY OPERATIONS				
FADD	90	10	54-368	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	91	11	70-370	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	92	12	146-168	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	93	13	154-184	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FLOATING-POINT DERIVED OPERATIONS				
SQRT	81	01	782-870	Square Root of TOS. Result to TOS.
SIN	82	02	3796-4808	Sine of TOS. Result to TOS.
COS	83	03	3840-4878	Cosine of TOS. Result to TOS.
TAN	84	04	4894-5886	Tangent of TOS. Result to TOS.
ASIN	85	05	6230-7938	Inverse Sine of TOS. Result to TOS.
ACOS	86	06	6304-8284	Inverse Cosine of TOS. Result to TOS.
ATAN	87	07	4992-6536	Inverse Tangent of TOS. Result to TOS.
LOG	88	08	4474-7132	Common Logarithm of TOS. Result to TOS.
LN	89	09	4298-6956	Natural Logarithm of TOS. Result to TOS.
EXP	8A	0A	3794-4878	e raised to power in TOS. Result to TOS.
PWR	8B	0B	8290-12032	NOS raised to power in TOS. Result to NOS. Pop Stack.
DATA AND STACK MANIPULATION OPERATIONS				
NOP	80	00	4	No Operation. Clear or set SVREQ.
FIXS	9F	1F	90-214	Convert TOS from floating point format to fixed point format.
FIXD	9E	1E	90-336	
FLTS	9D	1D	62-156	Convert TOS from fixed point format to floating point format.
FLTD	9C	1C	56-342	
CHSS	F4	74	22-24	Change sign of fixed point operand on TOS.
CHSD	B4	34	26-28	
CHSF	95	15	16-20	Change sign of floating point operand on TOS.
PTOS	F7	77	16	Push stack. Duplicate NOS in TOS.
PTOD	B7	37	20	
PTOF	97	17	20	
POPS	F8	78	10	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom.
POPD	B8	38	12	
POPF	98	18	12	
XCHS	F9	79	18	Exchange TOS and NOS.
XCHD	B9	39	26	
XCHF	99	19	26	
PUPI	9A	1A	16	Push floating point constant π onto TOS. Previous TOS becomes NOS.

COMMAND DESCRIPTIONS

This section contains detailed descriptions of the APU commands. They are arranged in alphabetical order by command mnemonic. In the descriptions, TOS means Top Of Stack and NOS means Next On Stack.

All derived functions except Square Root use Chebyshev polynomial approximating algorithms. This approach is used to help minimize the internal microprogram, to minimize the maximum error values and to provide a relatively even distribution of errors over the data range. The basic arithmetic operations are used by the derived functions to compute the various Chebyshev terms. The basic operations may produce error codes in the status register as a result.

Execution times are listed in terms of clock cycles and may be converted into time values by multiplying by the clock period used. For example, an execution time of 44 clock cycles when running at a 3MHz rate translates to 14 microseconds (44 x 32μs = 14μs). Variations in execution cycles reflect the data dependency of the algorithms.

In some operations exponent overflow or underflow may be possible. When this occurs, the exponent returned in the result will be 128 greater or smaller than its true value.

Many of the functions use portions of the data stack as scratch storage during development of the results. Thus previous values in those stack locations will be lost. Scratch locations destroyed are listed in the command descriptions and shown with the crossed-out locations in the Stack Contents After diagram.

Table 1 is a summary of all the Am9511A commands. It shows the hex codes for each command, the mnemonic abbreviation, a brief description and the execution time in clock cycles. The commands are grouped by functional classes.

The command mnemonics in alphabetical order are shown below in Table 2.

Table 2.

Command Mnemonics in Alphabetical Order.

ACOS	ARCCOSINE
ASIN	ARCSINE
ATAN	ARCTANGENT
CHSD	CHANGE SIGN DOUBLE
CHSF	CHANGE SIGN FLOATING
CHSS	CHANGE SIGN SINGLE
COS	COSINE
DADD	DOUBLE ADD
DDIV	DOUBLE DIVIDE
DMUL	DOUBLE MULTIPLY LOWER
DMUU	DOUBLE MULTIPLY UPPER
DSUB	DOUBLE SUBTRACT
EXP	EXPONENTIATION (e^X)
FADD	FLOATING ADD
FDIV	FLOATING DIVIDE
FIXD	FIX DOUBLE
FIXS	FIX SINGLE
FLTD	FLOAT DOUBLE
FLTS	FLOAT SINGLE
FMUL	FLOATING MULTIPLY
FSUB	FLOATING SUBTRACT
LOG	COMMON LOGARITHM
LN	NATURAL LOGARITHM
NOP	NO OPERATION
POPD	POP STACK DOUBLE
POPF	POP STACK FLOATING
POPS	POP STACK SINGLE
PTOD	PUSH STACK DOUBLE
PTOF	PUSH STACK FLOATING
PTOS	PUSH STACK SINGLE
PUPI	PUSH π
PWR	POWER (X^Y)
SADD	SINGLE ADD
SDIV	SINGLE DIVIDE
SIN	SINE
SMUL	SINGLE MULTIPLY LOWER
SMUU	SINGLE MULTIPLY UPPER
SQRT	SQUARE ROOT
SSUB	SINGLE SUBTRACT
TAN	TANGENT
XCHD	EXCHANGE OPERANDS DOUBLE
XCHF	EXCHANGE OPERANDS FLOATING
XCHS	EXCHANGE OPERANDS SINGLE

ACOS

32-BIT FLOATING-POINT INVERSE COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	0

Hex Coding: 86 with sr = 1
06 with sr = 0

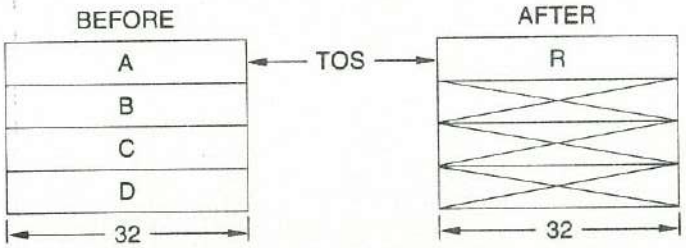
Execution Time: 6304 to 8284 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse cosine of A. The result R is a value in radians between 0 and π. Initial operands A, B, C and D are lost.
ACOS will accept all input data values within the range of −1.0 to +1.0. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ACOS exhibits a maximum relative error of 2.0 x 10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



ASIN

32-BIT FLOATING-POINT INVERSE SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	1

Hex Coding: 85 with sr = 1
05 with sr = 0

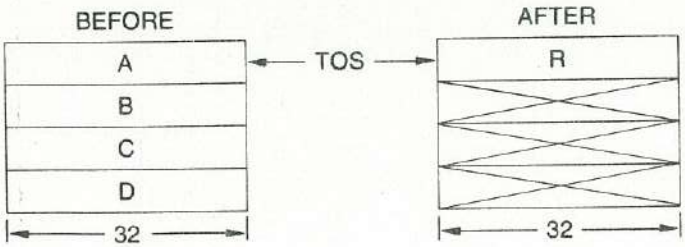
Execution Time: 6230 to 7938 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse sine of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, B, C and D are lost.
ASIN will accept all input data values within the range of −1.0 to +1.0. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ASIN exhibits a maximum relative error of 4.0 x 10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



ATAN

32-BIT FLOATING-POINT INVERSE TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	1

Hex Coding: 87 with sr = 1
07 with sr = 0

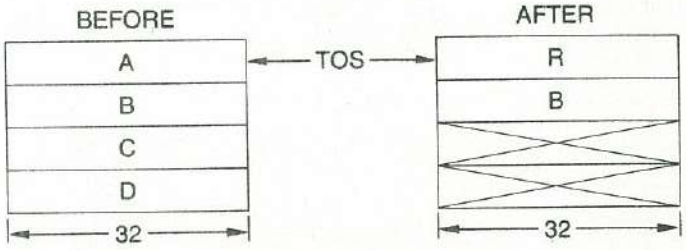
Execution Time: 4992 to 6536 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse tangent of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, C and D are lost. Operand B is unchanged.
ATAN will accept all input data values that can be represented in the floating point format.

Accuracy: ATAN exhibits a maximum relative error of 3.0 x 10^{-7} over the input data range.

Status Affected: Sign, Zero

STACK CONTENTS



CHSD

32-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	0	0

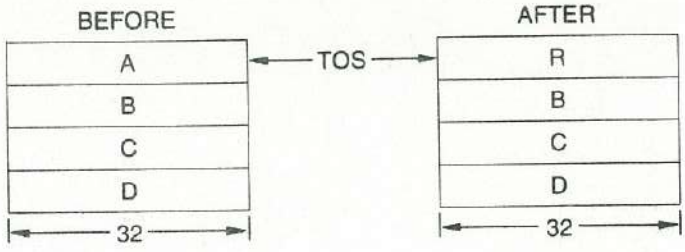
Hex Coding: B4 with sr = 1
34 with sr = 0

Execution Time: 26 to 28 clock cycles

Description:
The 32-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. Other entries in the stack are not disturbed.
Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Error Field (overflow)

STACK CONTENTS



CHSF

32-BIT FLOATING-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	0	1

Hex Coding: 95 with sr = 1
15 with sr = 0

Execution Time: 16 to 20 clock cycles

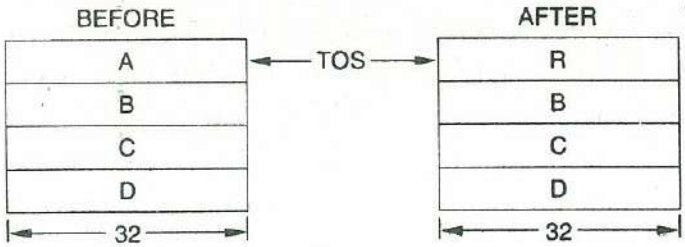
Description:

The sign of the mantissa of the 32-bit floating-point operand A at the TOS is inverted. The result R replaces A at the TOS. Other stack entries are unchanged.

If A is input as zero (mantissa MSB = 0), no change is made.

Status Affected: Sign, Zero

STACK CONTENTS



CHSS

16-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	0	0

Hex Coding: F4 with sr = 1
74 with sr = 0

Execution Time: 22 to 24 clock cycles

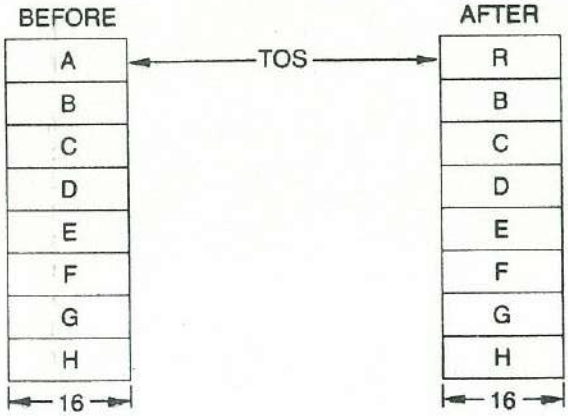
Description:

16-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. All other operands are unchanged.

Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



COS

32-BIT FLOATING-POINT COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	1

Hex Coding: 83 with sr = 1
03 with sr = 0

Execution Time: 3840 to 4878 clock cycles

Description:

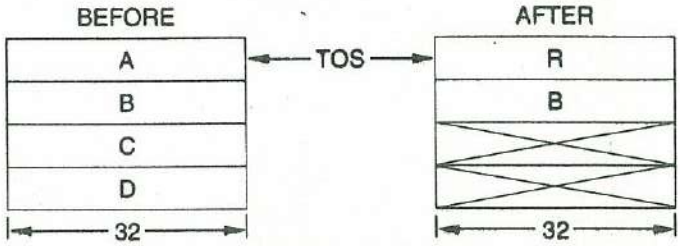
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point cosine of A. A is assumed to be in radians. Operands A, C and D are lost. B is unchanged.

The COS function can accept any input data value that can be represented in the data format. All input values are range reduced to fall within an interval of $-\pi/2$ to $+\pi/2$ radians.

Accuracy: COS exhibits a maximum relative error of 5.0×10^{-7} for all input data values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



DADD

32-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	0

Hex Coding: AC with sr = 1
2C with sr = 0

Execution Time: 20 to 22 clock cycles

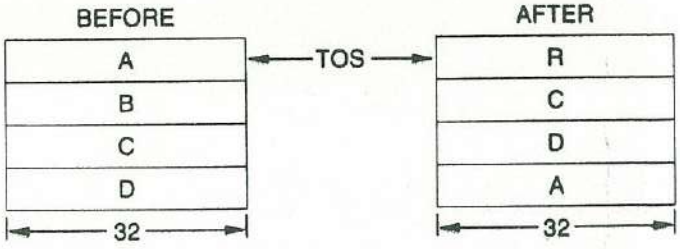
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is added to the 32-bit fixed-point two's complement integer operand B at the NOS. The result R replaces operand B and the Stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged. If the addition generates a carry it is reported in the status register.

If the result is too large to be represented by the data format, the least significant 32 bits of the result are returned and overflow status is reported.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



DDIV

32-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	1

Hex Coding: AF with sr = 1
2F with sr = 0

Execution Time: 196 to 210 clock cycles when A ≠ 0
18 clock cycles when A = 0.

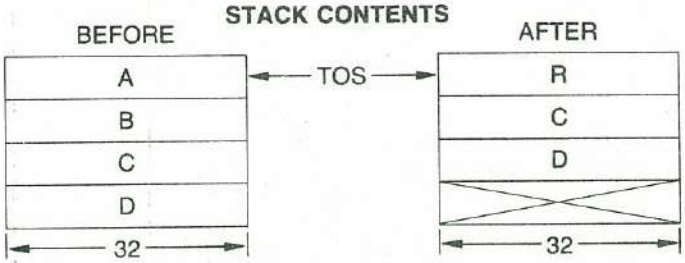
Description:

The 32-bit fixed-point two's complement integer operand B at NOS is divided by the 32-bit fixed-point two's complement integer operand A at the TOS. The 32-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. Operands C and D are unchanged.

If A is zero, R is set equal to B and the divide-by-zero error status will be reported. If either A or B is the most negative value possible in the format, R will be meaningless and the overflow error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



DMUL

32-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	0

Hex Coding: AE with sr = 1
2E with sr = 0

Execution Time: 194 to 210 clock cycles

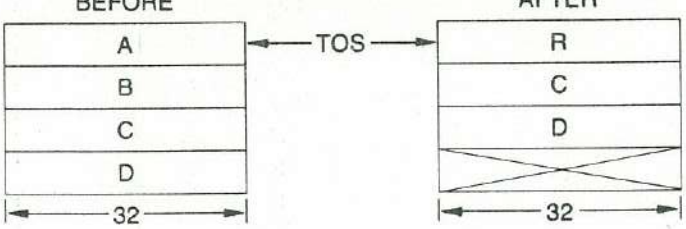
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.

The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DMUU

32-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	0

Hex Coding: B6 with sr = 1
36 with sr = 0

Execution Time: 182 to 218 clock cycles

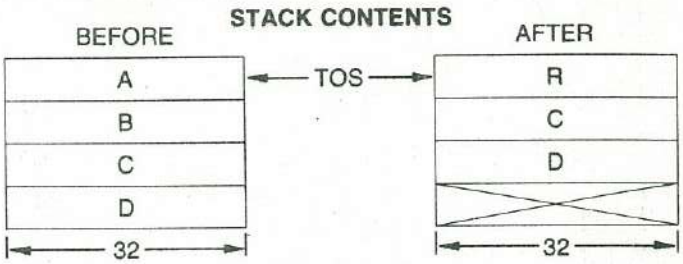
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.

If A or B was the most negative value possible in the format, overflow status is set and R is meaningless.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DSUB

32-BIT FIXED-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	1

Hex Coding: AD with sr = 1
2D with sr = 0

Execution Time: 38 to 40 clock cycles

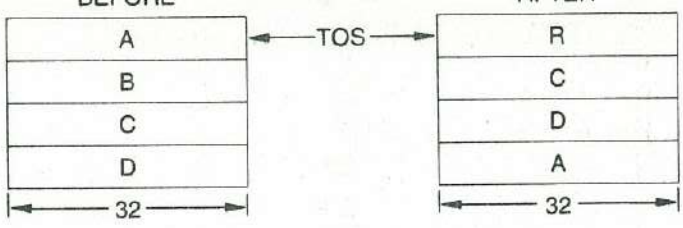
Description:

The 32-bit fixed-point two's complement operand A at the TOS is subtracted from the 32-bit fixed-point two's complement operand B at the NOS. The difference R replaces operand B and the stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged.

If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the data format range, the overflow bit is set and the 32 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Overflow

STACK CONTENTS



EXP

32-BIT FLOATING-POINT e^X

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	0

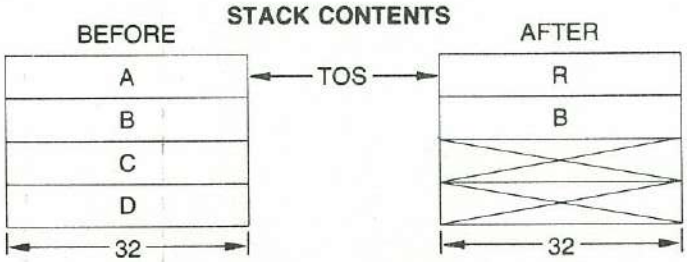
Hex Coding: 8A with sr = 1
0A with sr = 0

Execution Time: 3794 to 4878 clock cycles for $|A| \leq 1.0 \times 2^5$
34 clock cycles for $|A| > 1.0 \times 2^5$

Description:
The base of natural logarithms, e, is raised to an exponent value specified by the 32-bit floating-point operand A at the TOS. The result R of e^A replaces A. Operands A, C and D are lost. Operand B is unchanged.
EXP accepts all input data values within the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$. Input values outside this range will return a code of 1100 in the error field of the status register.

Accuracy: EXP exhibits a maximum relative error of 5.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field



FADD

32-BIT FLOATING-POINT ADD

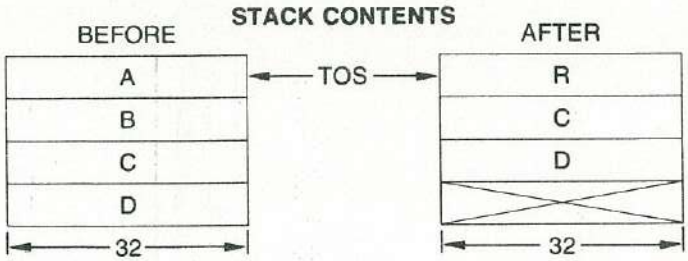
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	0

Hex Coding: 90 with sr = 1
10 with sr = 0

Execution Time: 54 to 368 clock cycles for A ≠ 0
24 clock cycles for A = 0

Description:
32-bit floating-point operand A at the TOS is added to 32-bit floating-point operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
Exponent alignment before the addition and normalization of the result accounts for the variation in execution time. Exponent overflow and underflow are reported in the status register, in which case the mantissa is correct and the exponent is offset by 128.

Status Affected: Sign, Zero, Error Field



FDIV

32-BIT FLOATING-POINT DIVIDE

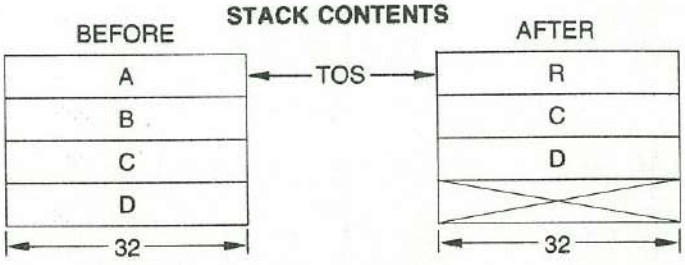
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	1

Hex Coding: 93 with sr = 1
13 with sr = 0

Execution Time: 154 to 184 clock cycles for A ≠ 0
22 clock cycles for A = 0

Description:
32-bit floating-point operand B at NOS is divided by 32-bit floating-point operand A at the TOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
If operand A is zero, R is set equal to B and the divide-by-zero error is reported in the status register. Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field



FIXD

32-BIT FLOATING-POINT TO 32-BIT FIXED-POINT CONVERSION

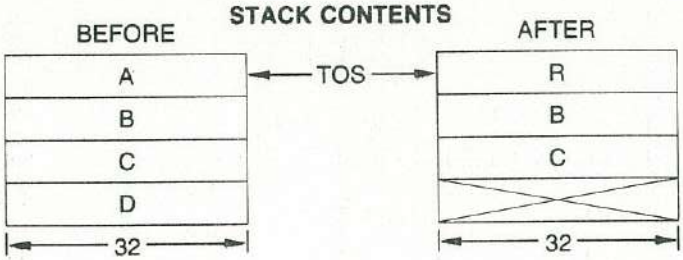
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	0

Hex Coding: 9E with sr = 1
1E with sr = 0

Execution Time: 90 to 336 clock cycles

Description:
32-bit floating-point operand A at the TOS is converted to a 32-bit fixed-point two's complement integer. The result R replaces A. Operands A and D are lost. Operands B and C are unchanged.
If the integer portion of A is larger than 31 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero Overflow



FIXS

32-BIT FLOATING-POINT TO 16-BIT FIXED-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	1

Hex Coding: 9F with sr = 1
1F with sr = 0

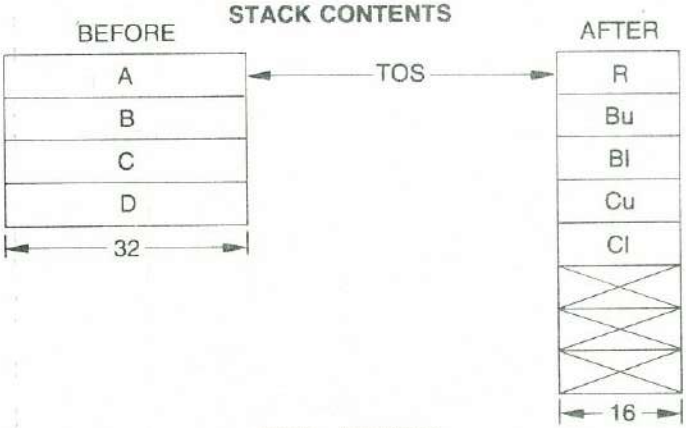
Execution Time: 90 to 214 clock cycles

Description:

32-bit floating-point operand A at the TOS is converted to a 16-bit fixed-point two's complement integer. The result R replaces the lower half of A and the stack is moved up by two bytes so that R occupies the TOS. Operands A and D are lost. Operands B and C are unchanged, but appear as upper (u) and lower (l) halves on the 16-bit wide stack if they are 32-bit operands.

If the integer portion of A is larger than 15 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero, Overflow



FLTD

32-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	0

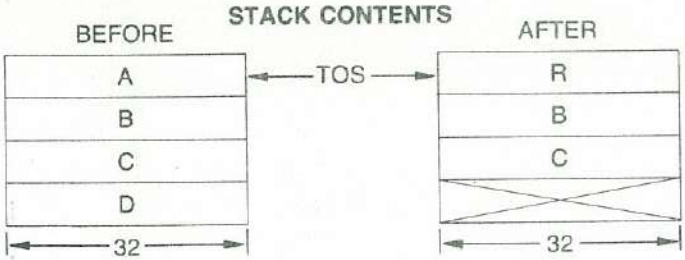
Hex Coding: 9C with sr = 1
1C with sr = 0

Execution Time: 56 to 342 clock cycles

Description:

32-bit fixed-point two's complement integer operand A at the TOS is converted to a 32-bit floating-point number. The result R replaces A at the TOS. Operands A and D are lost. Operands B and C are unchanged.

Status Affected: Sign, Zero



FLTS

16-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	1

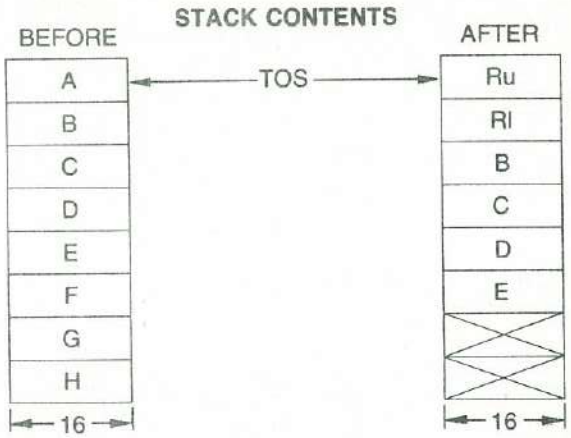
Hex Coding: 9D with sr = 1
1D with sr = 0

Execution Time: 62 to 156 clock cycles

Description:

16-bit fixed-point two's complement integer A at the TOS is converted to a 32-bit floating-point number. The lower half of the result R (Rl) replaces A, the upper half (Ru) replaces H and the stack is moved down so that Ru occupies the TOS. Operands A, F, G and H are lost. Operands B, C, D and E are unchanged.

Status Affected: Sign, Zero



FMUL

32-BIT FLOATING-POINT MULTIPLY

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	0

Hex Coding: 92 with sr = 1
12 with sr = 0

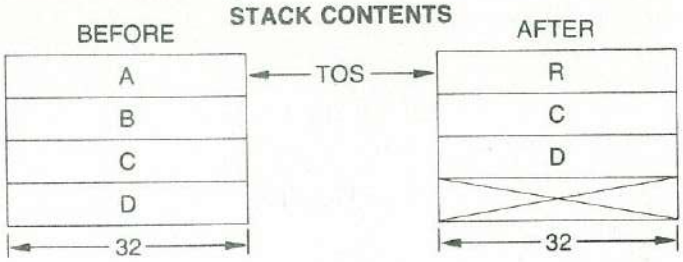
Execution Time: 146 to 168 clock cycles

Description:

32-bit floating-point operand A at the TOS is multiplied by the 32-bit floating-point operand B at the NOS. The normalized result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.

Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field



FSUB

32-BIT FLOATING-POINT SUBTRACTION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	1

Hex Coding: 91 with sr = 1
11 with sr = 0

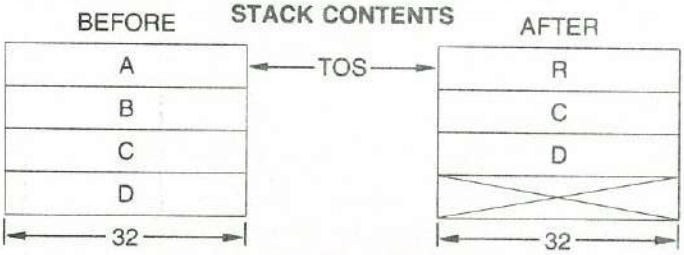
Execution Time: 70 to 370 clock cycles for A ≠ 0
26 clock cycles for A = 0

Description:
32-bit floating-point operand A at the TOS is subtracted from 32-bit floating-point operand B at the NOS. The normalized difference R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.

Exponent alignment before the subtraction and normalization of the result account for the variation in execution time.

Exponent overflow or underflow is reported in the status register in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field (overflow)



LOG

32-BIT FLOATING-POINT COMMON LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	0

Hex Coding: 88 with sr = 1
08 with sr = 0

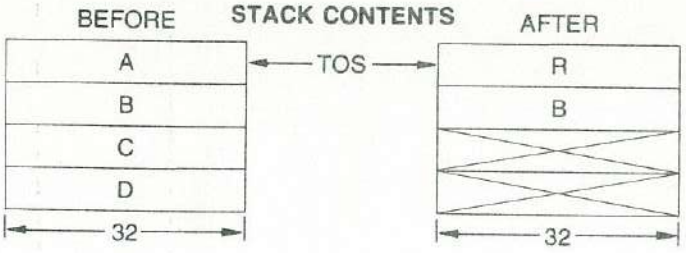
Execution Time: 4474 to 7132 clock cycles for A > 0
20 clock cycles for A ≤ 0

Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point common logarithm (base 10) of A. Operands A, C and D are lost. Operand B is unchanged.

The LOG function accepts any positive input data value that can be represented by the data format. If LOG of a non-positive value is attempted an error status of 0100 is returned.

Accuracy: LOG exhibits a maximum absolute error of 2.0×10^{-7} for the input range from 0.1 to 10, and a maximum relative error of 2.0×10^{-7} for positive values less than 0.1 or greater than 10.

Status Affected: Sign, Zero, Error Field



LN

32-BIT FLOATING-POINT NATURAL LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	1

Hex Coding: 89 with sr = 1
09 with sr = 0

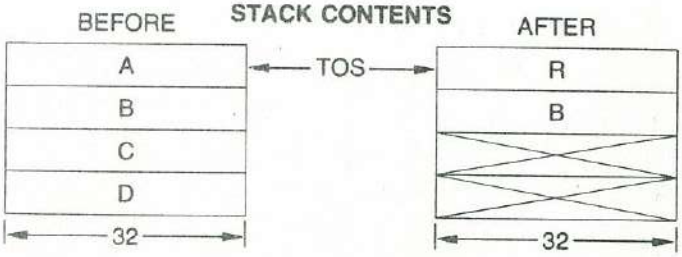
Execution Time: 4298 to 6956 clock cycles for A > 0
20 clock cycles for A ≤ 0

Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point natural logarithm (base e) of A. Operands A, C and D are lost. Operand B is unchanged.

The LN function accepts all positive input data values that can be represented by the data format. If LN of a non-positive number is attempted an error status of 0100 is returned.

Accuracy: LN exhibits a maximum absolute error of 2×10^{-7} for the input range from e^{-1} to e, and a maximum relative error of 2.0×10^{-7} for positive values less than e^{-1} or greater than e.

Status Affected: Sign, Zero, Error Field



NOP

NO OPERATION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	0

Hex Coding: 80 with sr = 1
00 with sr = 0

Execution Time: 4 clock cycles

Description:
The NOP command performs no internal data manipulations. It may be used to set or clear the service request interface line without changing the contents of the stack.

Status Affected: The status byte is cleared to all zeroes.

POPD

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	0

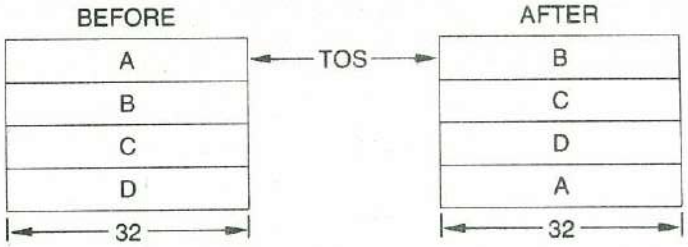
Hex Coding: B8 with sr = 1
38 with sr = 0

Execution Time: 12 clock cycles

Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged. POPD and POPF execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



POPS

16-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	0

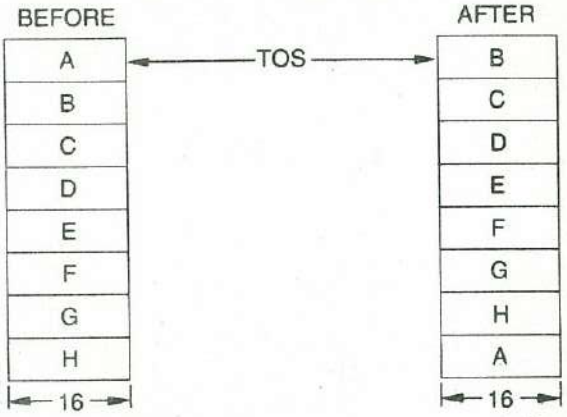
Hex Coding: F8 with sr = 1
78 with sr = 0

Execution Time: 10 clock cycles

Description:
The 16-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



POPF

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	0

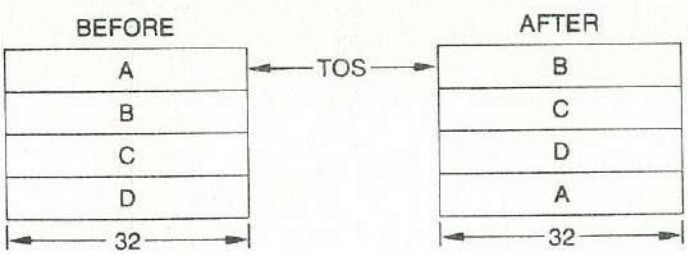
Hex Coding: 98 with sr = 1
18 with sr = 0

Execution Time: 12 clock cycles

Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The old TOS rotates to the bottom of the stack. All operand values are unchanged. POPF and POPD execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



PTOD

PUSH 32-BIT
TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	1

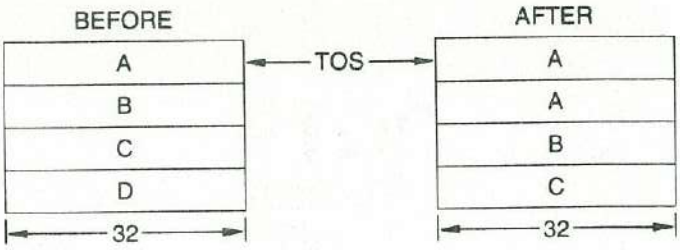
Hex Coding: B7 with sr = 1
37 with sr = 0

Execution Time: 20 clock cycles

Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOD and PTOF execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



PTOF

PUSH 32-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	1	1

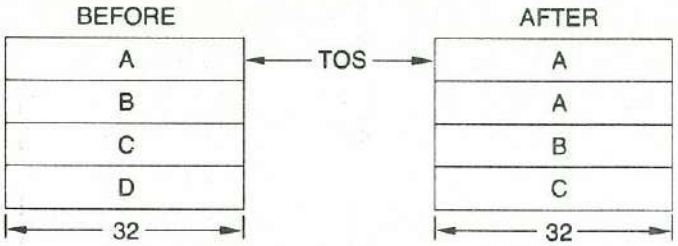
Hex Coding: 97 with sr = 1
17 with sr = 0

Execution Time: 20 clock cycles

Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOF and PTOD execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



PTOS

PUSH 16-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	1

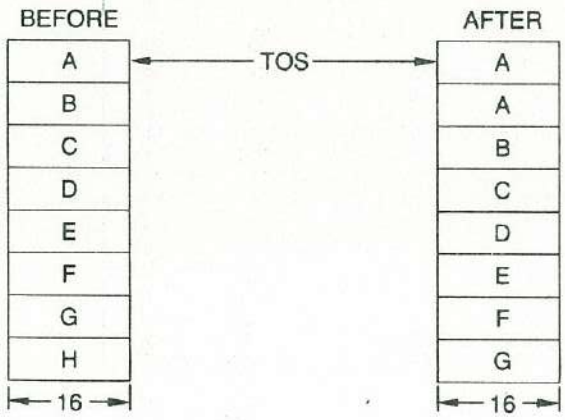
Hex Coding: F7 with sr = 1
77 with sr = 0

Execution Time: 16 clock cycles

Description:
The 16-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand H is lost and all other operand values are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



PUPI

PUSH 32-BIT FLOATING-POINT π

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	1	0

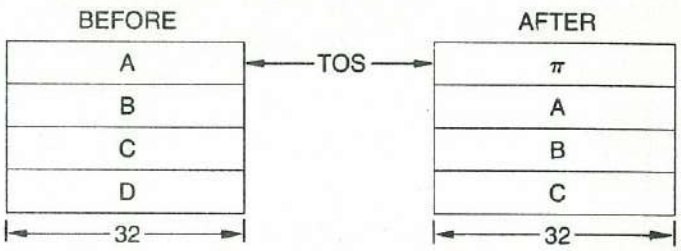
Hex Coding: 9A with sr = 1
1A with sr = 0

Execution Time: 16 clock cycles

Description:
The 32-bit stack is moved down so that the previous TOS occupies the new NOS location. 32-bit floating-point constant π is entered into the new TOS location. Operand D is lost. Operands A, B and C are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



PWR

32-BIT FLOATING-POINT X^Y

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	1

Hex Coding: 8B with sr = 1
0B with sr = 0

Execution Time: 8290 to 12032 clock cycles

Description:

32-bit floating-point operand B at the NOS is raised to the power specified by the 32-bit floating-point operand A at the TOS. The result R of B^A replaces B and the stack is moved up so that R occupies the TOS. Operands A, B, and D are lost. Operand C is unchanged.

The PWR function accepts all input data values that can be represented in the data format for operand A and all positive values for operand B. If operand B is non-positive an error status of 0100 will be returned. The EXP and LN functions are used to implement PWR using the relationship $B^A = EXP\,[A(LN\ B)]$. Thus if the term $[A(LN\ B)]$ is outside the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$ an error status of 1100 will be returned. Underflow and overflow conditions can occur.

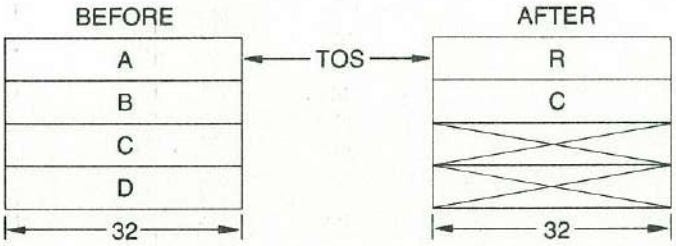
Accuracy: The error performance for PWR is a function of the LN and EXP performance as expressed by:

$|(\text{Relative Error})_{PWR}| = |(\text{Relative Error})_{EXP} + |A(\text{Absolute Error})_{LN}|$

The maximum relative error for PWR occurs when A is at its maximum value while $[A(LN\ B)]$ is near 1.0×2^5 and the EXP error is also at its maximum. For most practical applications the relative error for PWR will be less than 7.0×10^{-7}.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SADD

16-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	0

Hex Coding: EC with sr = 1
6C with sr = 0

Execution Time: 16 to 18 clock cycles

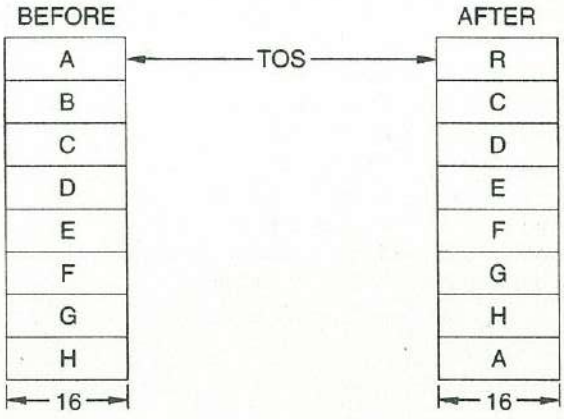
Description:

16-bit fixed-point two's complement integer operand A at the TOS is added to 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.

If the addition generates a carry bit it is reported in the status register. If an overflow occurs it is reported in the status register and the 16 least significant bits of the result are returned.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



SDIV

16-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	1

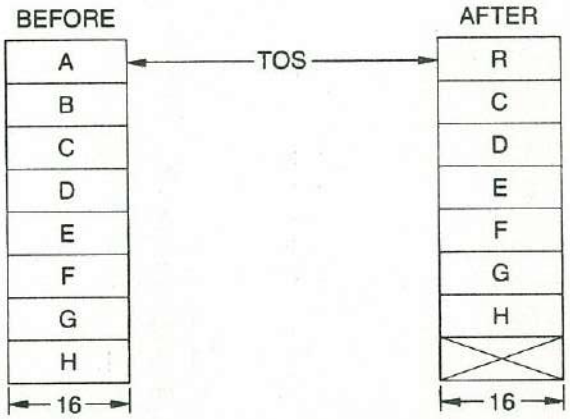
Hex Coding: EF with sr = 1
6F with sr = 0

Execution Time: 84 to 94 clock cycles for A ≠ 0
14 clock cycles for A = 0

Description:
16-bit fixed-point two's complement integer operand B at the NOS is divided by 16-bit fixed-point two's complement integer operand A at the TOS. The 16-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. All other operands are unchanged.
If A is zero, R will be set equal to B and the divide-by-zero error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SIN

32-BIT FLOATING-POINT SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	0

Hex Coding: 82 with sr = 1
02 with sr = 0

Execution Time: 3796 to 4808 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

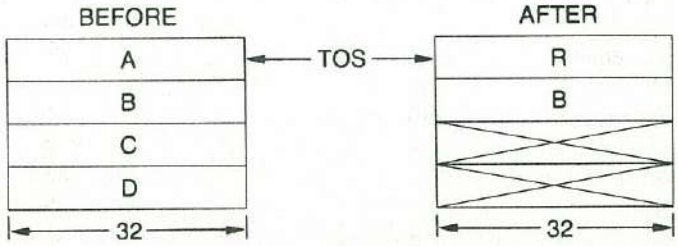
Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point sine of A. A is assumed to be in radians. Operands A, C and D are lost. Operand B is unchanged.

The SIN function will accept any input data value that can be represented by the data format. All input values are range reduced to fall within the interval $-\pi/2$ to $+\pi/2$ radians.

Accuracy: SIN exhibits a maximum relative error of 5.0 x 10^{-7} for input values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



SMUL

16-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	0

Hex Coding: EE with sr = 1
6E with sr = 0

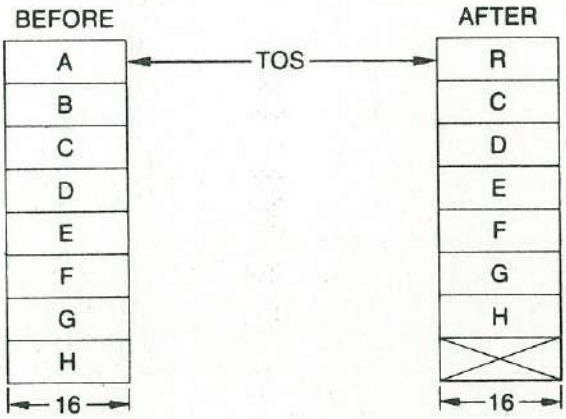
Execution Time: 84 to 94 clock cycles

Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. All other operands are unchanged. The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SMUU

16-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	0

Hex Coding: F6 with sr = 1
76 with sr = 0

Execution Time: 80 to 98 clock cycles

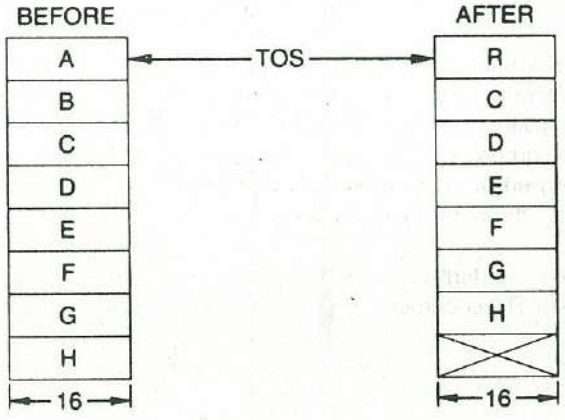
Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. All other operands are unchanged.

If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SQRT

32-BIT FLOATING-POINT SQUARE ROOT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	1

Hex Coding: 81 with sr = 1
01 with sr = 0

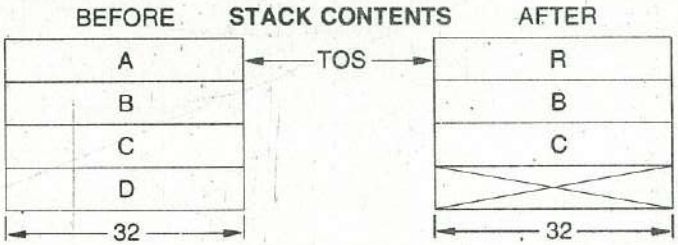
Execution Time: 782 to 870 clock cycles

Description:

32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point square root of A. Operands A and D are lost. Operands B and C are not changed.

SQRT will accept any non-negative input data value that can be represented by the data format. If A is negative an error code of 0100 will be returned in the status register.

Status Affected: Sign, Zero, Error Field



SSUB

16-BIT FIXED-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	1

Hex Coding: ED with sr = 1
6D with sr = 0

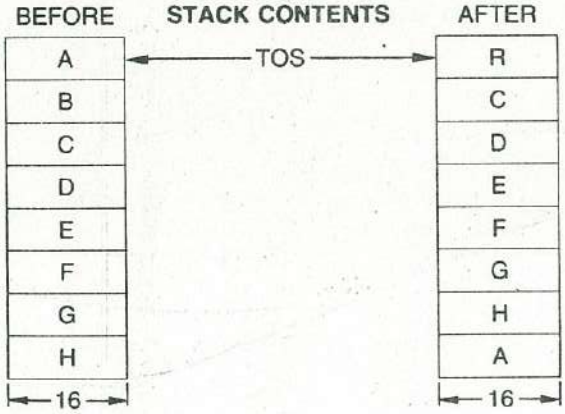
Execution Time: 30 to 32 clock cycles

Description:

16-bit fixed-point two's complement integer operand A at the TOS is subtracted from 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.

If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the format range, the overflow status is set and the 16 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Error Field



TAN

32-BIT FLOATING-POINT TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	0

Hex Coding: 84 with sr = 1
04 with sr = 0

Execution Time: 4894 to 5886 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

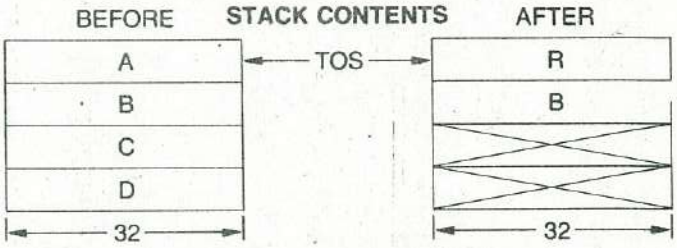
Description:

The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point tangent of A. Operand A is assumed to be in radians. A, C and D are lost. B is unchanged.

The TAN function will accept any input data value that can be represented in the data format. All input data values are range-reduced to fall within $-\pi/4$ to $+\pi/4$ radians. TAN is unbounded for input values near odd multiples of $\pi/2$ and in such cases the overflow bit is set in the status register. For angles smaller than 2^{-12} radians, TAN returns A as the tangent of A.

Accuracy: TAN exhibits a maximum relative error of 5.0×10^{-7} for input data values in the range of -2π to $+2\pi$ radians except for data values near odd multiples of $\pi/2$.

Status Affected: Sign, Zero, Error Field (overflow)



XCHD

EXCHANGE 32-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	1

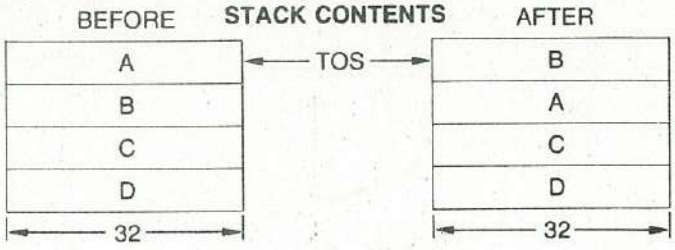
Hex Coding: B9 with sr = 1
39 with sr = 0

Execution Time: 26 clock cycles

Description:

32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero,



XCHF

EXCHANGE 32-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	1

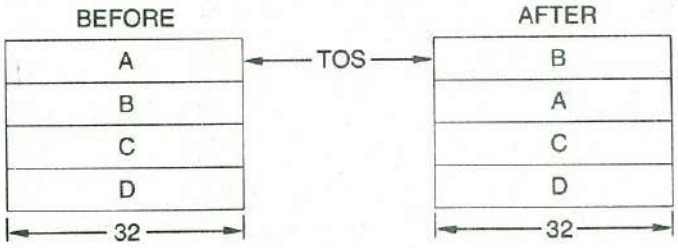
Hex Coding: 99 with sr = 1
19 with sr = 0

Execution Time: 26 clock cycles

Description:
32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



XCHS

EXCHANGE 16-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	1

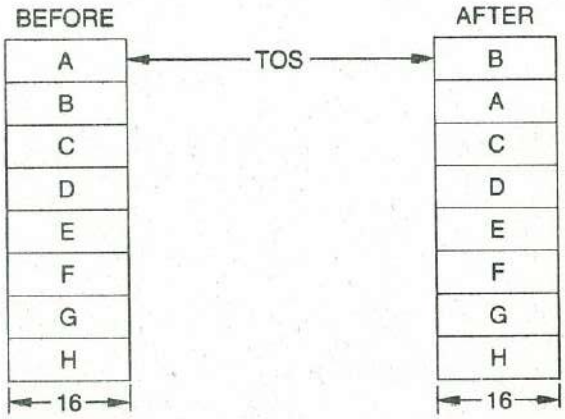
Hex Coding: F9 with sr = 1
79 with sr = 0

Execution Time: 18 clock cycles

Description:
16-bit operand A at the TOS and 16-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operand values are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



MAXIMUM RATINGS beyond which useful life may be impaired

Storage Temperature	−65°C to +150°C
Ambient Temperature Under Bias	−55°C to +125°C
VDD with Respect to VSS	−0.5V to +15.0V
/CC with Respect to VSS	−0.5V to +7.0V
All Signal Voltages with Respect to VSS	−0.5V to +7.0V
Power Dissipation (Package Limitation)	2.0W

The products described by this specification include internal circuitry designed to protect input devices from damaging accumulations of static charge. It is suggested, nevertheless, that conventional precautions be observed during storage, handling and use in order to avoid exposure to excessive voltages.

OPERATING RANGE

Part Number	Ambient Temperature	VSS	VCC	VDD
Am9511ADC	0°C ≤ T_A ≤ +70°C	0V	+5.0V ±5%	+12V ±5%
Am9511ADM	−55°C ≤ T_A ≤ +125°C	0V	+5.0V ±10%	+12V ±10%

ELECTRICAL CHARACTERISTICS Over Operating Range (Note 1)

Parameters	Description	Test Conditions	Min.	Typ.	Max.	Units
VOH	Output HIGH Voltage	IOH = −200μA	3.7			Volts
VOL	Output LOW Voltage	IOL = 3.2mA			0.4	Volts
VIH	Input HIGH Voltage		2.0		VCC	Volts
VIL	Input LOW Voltage		−0.5		0.8	Volts
IIX	Input Load Current	VSS ≤ VI ≤ VCC			±10	μA
IOZ	Data Bus Leakage	VO = 0.4V			10	μA
		VO = VCC			10	
ICC	VCC Supply Current	T_A = +25°C		50	90	mA
		T_A = 0°C			95	
		T_A = −55°C			100	
IDD	VDD Supply Current	T_A = +25°C		50	90	mA
		T_A = 0°C			95	
		T_A = −55°C			100	
CO	Output Capacitance	fc = 1.0MHz, Inputs = 0V		8	10	pF
CI	Input Capacitance			5	8	pF
CIO	I/O Capacitance			10	12	pF

SWITCHING CHARACTERISTICS over operating range (Notes 2, 3)

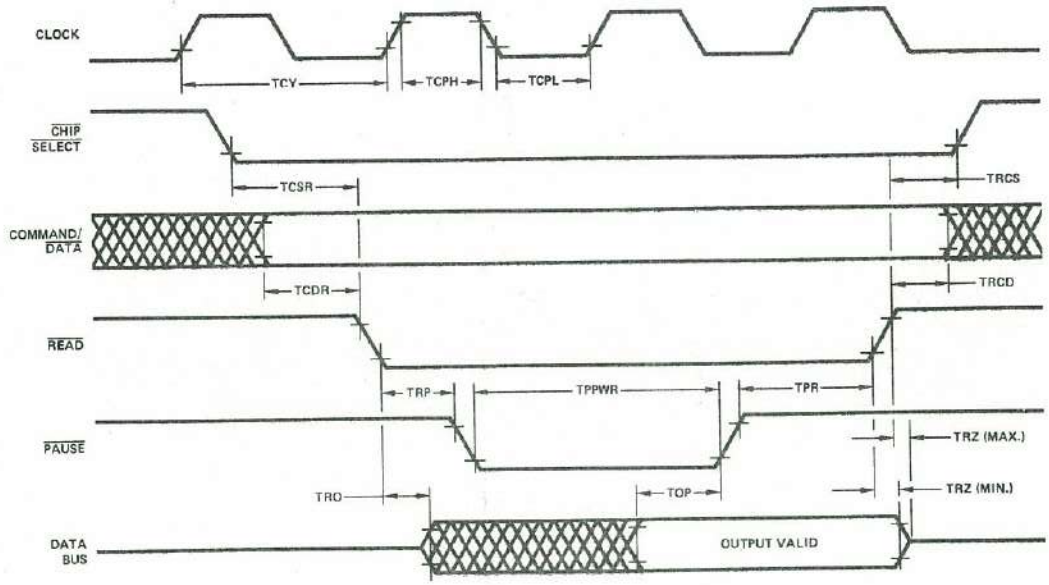
Parameters	Description		Am9511A Min.	Am9511A Max.	Am9511A-1 Min.	Am9511A-1 Max.	Units
TAPW	$\overline{EACK}$ LOW Pulse Width		100		75		ns
TCDR	C/$\overline{D}$ to $\overline{RD}$ LOW Set up Time		0		0		ns
TCDW	C/$\overline{D}$ to $\overline{WR}$ LOW Set up Time		0		0		ns
TCPH	Clock Pulse HIGH Width		200		140		ns
TCPL	Clock Pulse LOW Width		240		160		ns
TCSR	$\overline{CS}$ LOW to $\overline{RD}$ LOW Set up Time		0		0		ns
TCSW	$\overline{CS}$ LOW to $\overline{WR}$ LOW Set up Time		0		0		ns
TCY	Clock Period		480	5000	320	3300	ns
TDW	Data Bus Stable to $\overline{WR}$ HIGH Set up Time		150		100 (Note 9)		ns
TEAE	$\overline{EACK}$ LOW to $\overline{END}$ HIGH Delay			200		175	ns
TEPW	$\overline{END}$ LOW Pulse Width (Note 4)		400		300		ns
TOP	Data Bus Output Valid to $\overline{PAUSE}$ HIGH Delay		0		0		ns
TPPWR	$\overline{PAUSE}$ LOW Pulse Width Read (Note 5)	Data	3.5TCY+50	5.5TCY+300	3.5TCY+50	5.5TCY+200	ns
		Status	1.5TCY+50	3.5TCY+300	1.5TCY+50	3.5TCY+200	
TPPWW	$\overline{PAUSE}$ LOW Pulse Width Write (Note 8)			50		50	ns
TPR	$\overline{PAUSE}$ HIGH to $\overline{RD}$ HIGH Hold Time		0		0		ns
TPW	$\overline{PAUSE}$ HIGH to $\overline{WR}$ HIGH Hold Time		0		0		ns
TRCD	$\overline{RD}$ HIGH to C/$\overline{D}$ Hold Time		0		0		ns
TRCS	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH Hold Time		0		0		ns
TRO	$\overline{RD}$ LOW to Data Bus ON Delay		50		50		ns
TRP	$\overline{RD}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)	ns
TRZ	$\overline{RD}$ HIGH to Data Bus OFF Delay		50	200	50	150	ns
TSAPW	$\overline{SVACK}$ LOW Pulse Width		100		75		ns
TSAR	$\overline{SVACK}$ LOW to SVREQ LOW Delay			300		200	ns
TWCD	$\overline{WR}$ HIGH to C/$\overline{D}$ Hold Time		60		30		ns
TWCS	$\overline{WR}$ HIGH to $\overline{CS}$ HIGH Hold Time		60		30		ns
TWD	$\overline{WR}$ HIGH to Data Bus Hold Time		20		20		ns
TWI	Write Inactive Time (Note 8)	Command	3TCY		3TCY		ns
		Data	4TCY		4TCY		
TWP	$\overline{WR}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)	ns

NOTES

1. Typical values are for $T_A = 25°C$, nominal supply voltages and nominal processing parameters.
2. Switching parameters are listed in alphabetical order.
3. Test conditions assume transition times of 20ns or less, output loading of one TTL gate plus 100pF and timing reference levels of 0.8V and 2.0V.
4. $\overline{END}$ low pulse width is specified for $\overline{EACK}$ tied to VSS. Otherwise TEAE applies.
5. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, $\overline{PAUSE}$ LOW Pulse Width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
6. $\overline{PAUSE}$ is pulled low for both command and data operations
7. TEX is the execution time of the current command (see the Command Execution Times table).
8. $\overline{PAUSE}$ low pulse width is less than 50ns when writing into the data port or the control port as long as the duty cycle requirement (TWI) is observed and no previous command is being executed. TWI may be safely violated as long as the extended TPPWW that results is observed. If a previously entered command is being executed, $\overline{PAUSE}$ LOW Pulse Width is the time to complete execution plus the time shown.
9. 150ns for Am9511A-1DM.

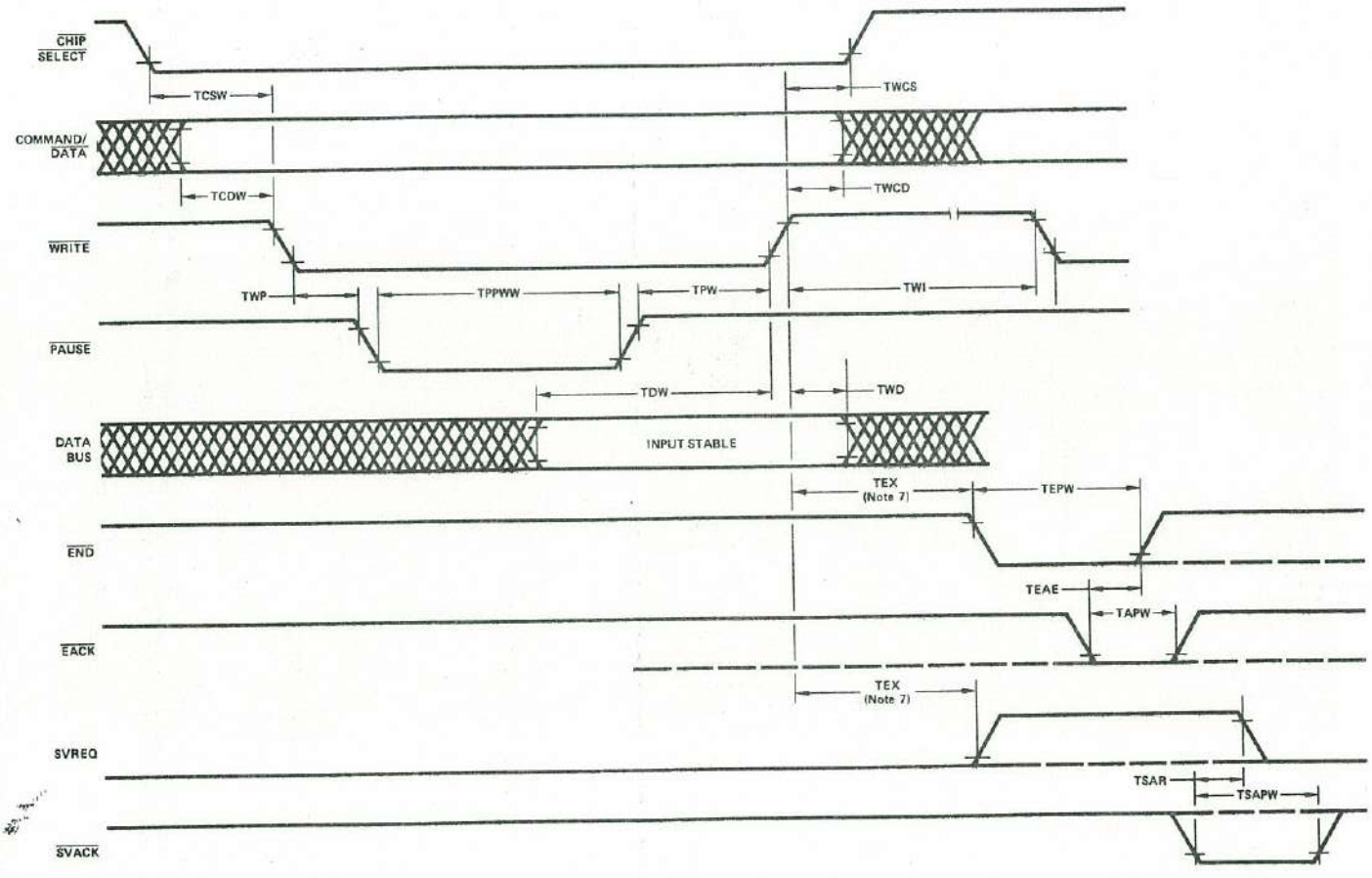
SWITCHING WAVEFORMS

READ OPERATIONS



MOS-048

WRITE OPERATIONS



MOS-049

APPLICATION INFORMATION

The diagram in Figure 2 shows the interface connections for the Am9511A APU with operand transfers handled by an Am9517 DMA controller, and CPU coordination handled by an Am9519 Interrupt Controller. The APU interrupts the CPU to indicate that a command has been completed. When the performance enhancements provided by the DMA and Interrupt operations are not required, the APU interface can be simplified as shown in Figure 1. The Am9511A APU is designed with a general purpose 8-bit data bus and interface control so that it can be conveniently used with any general 8-bit processor.

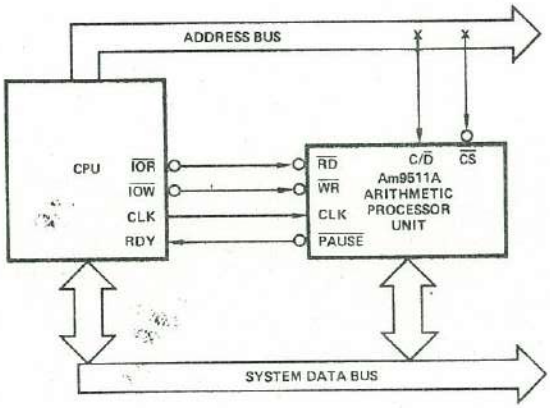


Figure 1. Am9511A Minimum Configuration Example.

MOS-050

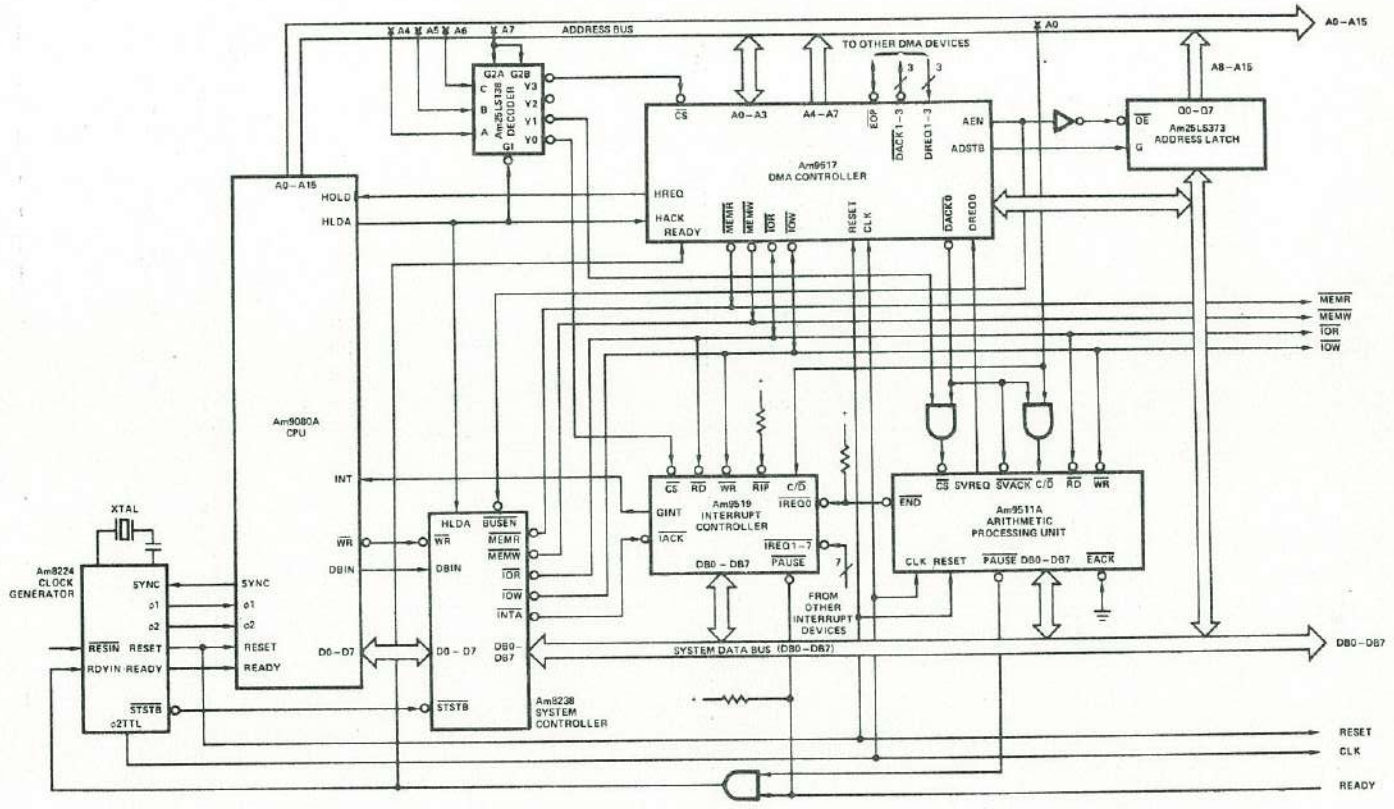
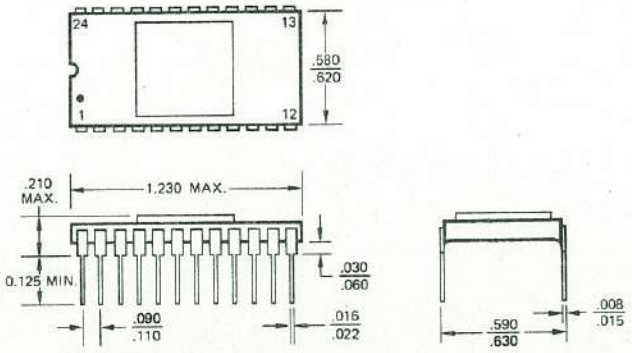


Figure 2. Am9511A High Performance Configuration Example.

MOS-051

PHYSICAL DIMENSIONS
Dual-In-Line

24-Pin Side-Brazed



ADVANCED MICRO DEVICES, INC.
901 Thompson Place
P.O. Box 453
Sunnyvale,
California 94086
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450

Appendix D

Am9512 Data Sheet

Am9512

Floating Point Processor
Advanced Micro Devices
Advanced MOS/LSI

DISTINCTIVE CHARACTERISTICS

- Single (32-bit) and double (64-bit) precision capability
- Add, subtract, multiply and divide functions
- Compatible with proposed IEEE format
- Easy interfacing to microprocessors
- 8-bit data bus
- Standard 24-pin package
- 12V and 5V power supplies
- Stack oriented operand storage
- Direct memory access or programmed I/O Data Transfers
- End of execution signal
- Error interrupt
- All inputs and outputs TTL level compatible
- Advanced N-channel silicon gate MOS technology
- 100% MIL-STD-883 reliability assurance testing

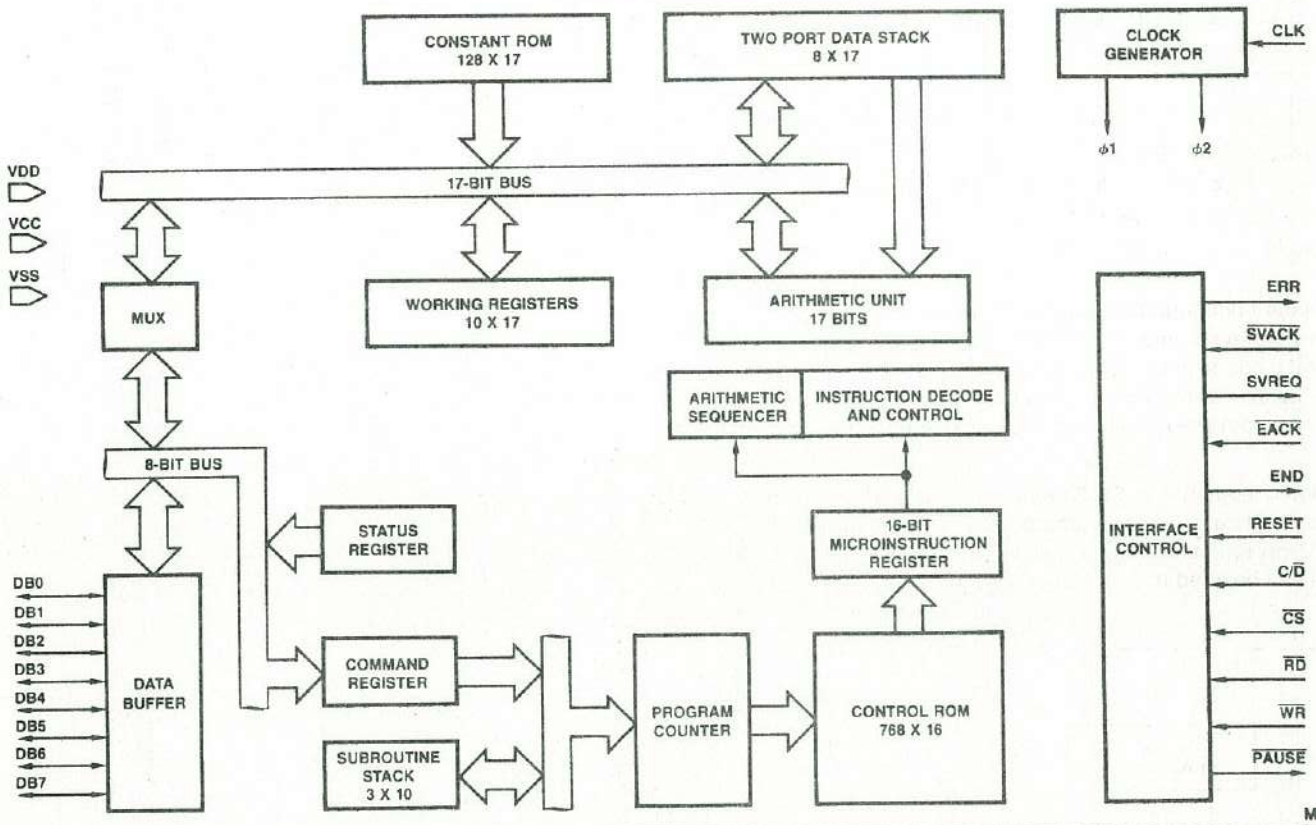
GENERAL DESCRIPTION

The Am9512 is a high performance floating-point processor unit (FPU). It provides single precision (32-bit) and double precision (64-bit) add, subtract, multiply and divide operations. It can be easily interfaced to enhance the computational capabilities of the host microprocessor.

The operand, result, status and command information transfers take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack by the host processor and a command is issued to perform an operation on the data stack. The results of this operation are available to the host processor by popping the stack.

Information transfers between the Am9512 and the host processor can be handled by using programmed I/O or direct memory access techniques. After completing an operation, the Am9512 activates an "end of execution" signal that can be used to interrupt the host processor.

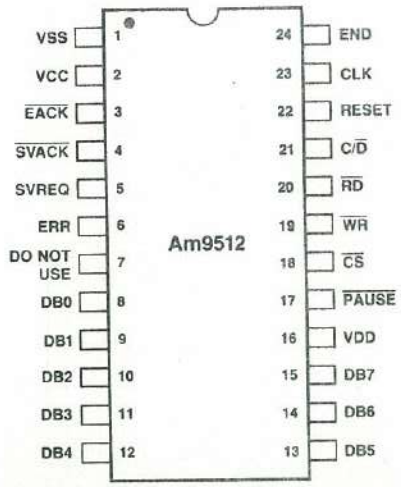
BLOCK DIAGRAM



ORDERING INFORMATION

Package Type	Ambient Temperature	Maximum Clock Frequency	
		2MHz	3MHz
Hermetic DIP	$0°C \leq T_A \leq 70°C$	AM9512DC	AM9512-1DC
	$-55°C \leq T_A \leq +125°C$	AM9512DM	AM9512-1DM

CONNECTION DIAGRAM
Top View



Note: Pin 1 is marked for orientation. MOS-204

INTERFACE SIGNAL DESCRIPTION

VCC: +5V Power Supply

VDD: +12V Power Supply

VSS: Ground

CLK (Clock, Input)

An external timing source connected to the CLK input provides the necessary clocking.

RESET (Reset, Input)

A HIGH on this input causes initialization. Reset terminates any operation in progress, and clears the status register to zero. The internal stack pointer is initialized and the contents of the stack may be affected. After a reset the END output, the ERR output and the SVREQ output will be LOW. For proper initialization, RESET must be HIGH for at least five CLK periods following stable power supply voltages and stable clock.

C/$\overline{D}$ (Command/Data Select, Input)

The C/$\overline{D}$ input together with the $\overline{RD}$ and $\overline{WR}$ inputs determines the type of transfer to be performed on the data bus as follows:

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Function
L	H	L	Push data byte into the stack
L	L	H	Pop data byte from the stack
H	H	L	Enter command
H	L	H	Read Status
X	L	L	Undefined

L = LOW
H = HIGH
X = DON'T CARE

END (End of Execution, Output)

A HIGH on this output indicates that execution of the current command is complete. This output will be cleared LOW by activating the $\overline{EACK}$ input LOW or performing any read or write operation or device initialization using the RESET. If $\overline{EACK}$ is tied LOW, the END output will be a pulse (see $\overline{EACK}$ description).

Reading the status register while a command execution is in progress is allowed. However any read or write operation clears the flip-flop that generates the END output. Thus such continuous reading could conflict with internal logic setting of the END flip-flop at the end of command execution.

$\overline{EACK}$ (End Acknowledge, Input)

This input when LOW makes the END output go LOW. As mentioned earlier HIGH on the END output signals completion of a command execution. The END signal is derived from an internal flip-flop which is clocked at the completion of a command. This flip-flop is clocked to the reset state when $\overline{EACK}$ is LOW. Consequently, if $\overline{EACK}$ is tied LOW, the END output will be a pulse that is approximately one CLK period wide.

SVREQ (Service Request, Output)

A HIGH on this output indicates completion of a command. In this sense this output is the same as the END output. However, the SVREQ output will go HIGH at the completion of a command. This bit must be 1 for SVREQ to go HIGH. The SVREQ can be cleared (i.e., go LOW) by activating the $\overline{SVACK}$ input LOW or initializing the device using the RESET. Also, the SVREQ will be automatically cleared after completion of any command that has the service request bit as 0.

$\overline{SVACK}$ (Service Acknowledge, Input)

A LOW on this input clears SVREQ. If the $\overline{SVACK}$ input is permanently tied LOW, it will conflict with the internal setting of the SVREQ output. Thus the SVREQ indication cannot be relied upon if the $\overline{SVACK}$ is tied LOW.

DB0-DB7 (Data Bus, Input/Output)

These eight bidirectional lines are used to transfer command, status and operand information between the device and the host processor. DB0 is the least significant and DB7 is the most significant bit position. HIGH on a data bus line corresponds to 1 and LOW corresponds to 0.

When pushing operands on the stack using the data bus, the least significant byte must be pushed first and most significant byte last. When popping the stack to read the result of an operation, the most significant byte will be available on the data bus first and the least significant byte will be the last. Moreover, for pushing operands and popping results, the number of transactions must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer will not be aligned properly. The Am9512 single precision format requires 4 bytes and double precision format requires 8 bytes.

ERR (Error, Output)

This output goes HIGH to indicate that the current command execution resulted in an error condition. The error conditions are: attempt to divide by zero, exponent overflow and exponent underflow. The ERR output is cleared LOW on read status register operation or upon RESET.

The ERR output is derived from the error bits in the status register. These error bits will be updated internally at an appropriate time during a command execution. Thus ERR output going HIGH may not correspond with the completion of a command. Reading of the status register can be performed while a command execution is in progress. However it should be noted that reading the status register clears the ERR output. Thus reading the status register while a command execution in progress may result in an internal conflict with the ERR output.

$\overline{CS}$ (Chip Select, Input)

This input must be LOW to accomplish any read or write operation to the Am9512.

To perform a write operation, appropriate data is presented on DB0 through DB7 lines, appropriate logic level on the C/$\overline{D}$ input and the $\overline{CS}$ input is made LOW. Whenever $\overline{WR}$ and $\overline{RD}$ inputs are both HIGH and $\overline{CS}$ is LOW, $\overline{PAUSE}$ goes LOW. However actual writing into the Am9512 cannot start until $\overline{WR}$ is made LOW. After initiating the write operation by the HIGH to LOW transition on the $\overline{WR}$ input, the $\overline{PAUSE}$ output will go HIGH indicating the write operation has been acknowledged. The $\overline{WR}$ input can go HIGH after $\overline{PAUSE}$ goes HIGH. The data lines, C/$\overline{D}$ input and the $\overline{CS}$ input can change when appropriate hold time requirements are satisfied. See write timing diagram for details.

To perform a read operation an appropriate logic level is established on the C/$\overline{D}$ input and $\overline{CS}$ is made LOW. The $\overline{PAUSE}$ output goes LOW because $\overline{WR}$ and $\overline{RD}$ inputs are HIGH. The read operation does not start until the $\overline{RD}$ input goes LOW. $\overline{PAUSE}$ will go HIGH indicating that read operation is complete and the required information is available on the DB0 through DB7 lines. This information will remain on the data lines as long as $\overline{RD}$ is LOW. The $\overline{RD}$ input can return HIGH anytime after $\overline{PAUSE}$ goes HIGH. The $\overline{CS}$ input and C/$\overline{D}$ input can change anytime after $\overline{RD}$ returns HIGH. See read timing diagram for details. If the $\overline{CS}$ is tied LOW permanently, $\overline{PAUSE}$ will remain LOW until the next Am9512 read or write access.

$\overline{RD}$ (Read, Input)

A LOW on this input is used to read information from an internal location and gate that information onto the data bus. The $\overline{CS}$ input must be LOW to accomplish the read operation. The C/$\overline{D}$ input determines what internal location is of interest. See C/$\overline{D}$, $\overline{CS}$ input descriptions and read timing diagram for details. If the END output was HIGH, performing any read operation will make the END output go LOW after the HIGH to LOW transition of the $\overline{RD}$ input (assuming $\overline{CS}$ is LOW). If the ERR output was HIGH performing a status register read operation will make the ERR output LOW. This will happen after the HIGH to LOW transition of the RD input (assuming $\overline{CS}$ is LOW).

$\overline{WR}$ (Write, Input)

A LOW on this input is used to transfer information from the data bus into an internal location. The $\overline{CS}$ must be LOW to accomplish the write operation. The C/$\overline{D}$ determines which internal location is to be written. See C/$\overline{D}$, $\overline{CS}$ input descriptions and write timing diagram for details.

If the END output was HIGH, performing any write operation will make the END output go LOW after the LOW to HIGH transition of the $\overline{WR}$ input (assuming $\overline{CS}$ is LOW).

$\overline{PAUSE}$ (Pause, Output)

This output is a handshake signal used while performing read or write transactions with the Am9512. If the $\overline{WR}$ and $\overline{RD}$ inputs are both HIGH, the $\overline{PAUSE}$ output goes LOW with the $\overline{CS}$ input in anticipation of a transaction. If $\overline{WR}$ goes LOW to initiate a write transaction with proper signals established on the DB0-DB7, C/$\overline{D}$ inputs, the $\overline{PAUSE}$ will return HIGH indicating that the write operation has been accomplished. The $\overline{WR}$ can be made HIGH after this event. On the other hand, if a read operation is desired, the $\overline{RD}$ input is made LOW after activating $\overline{CS}$ LOW and establishing proper C/$\overline{D}$ input. (The $\overline{PAUSE}$ will go LOW in response to $\overline{CS}$ going LOW.) The $\overline{PAUSE}$ will return HIGH indicating completion of read. The $\overline{RD}$ can return HIGH after this event. It should be noted that a read or write operation can be initiated without any regard to whether a command execution is in progress or not. Proper device operation is assured by obeying the $\overline{PAUSE}$ output indication as described.

FUNCTIONAL DESCRIPTION

Major functional units of the Am9512 are shown in the block diagram. The Am9512 employs a microprogram controlled stack oriented architecture with 17-bit wide data paths.

The Arithmetic Unit receives one of its operands from the Operand Stack. This stack is an eight word by 17-bit two port memory with last in – first out (LIFO) attributes. The second operand to the Arithmetic Unit is supplied by the internal 17-bit bus. In addition to supplying the second operand, this bidirectional bus also carries the results from the output of the Arithmetic Unit when required. Writing into the Operand Stack takes place from this internal 17-bit bus when required. Also connected to this bus are the Constant ROM and Working Registers. The ROM provides the required constants to perform the mathematical operations while the Working Registers provide storage for the intermediate values during command execution.

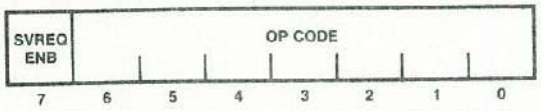
Communication between the external world and the Am9512 takes place on eight bidirectional input/output lines, DB0 through DB7 (Data Bus). These signals are gated to the internal 8-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication between the internal eight and 17-bit buses. The Status Register and Command Register are also located on the 8-bit bus.

The Am9512 operations are controlled by the microprogram contained in the Control ROM. The Program Counter supplies the microprogram addresses and can be partially loaded from the Command Register. Associated with the Program Counter is the Subroutine Stack where return addresses are held during subroutine calls in the microprogram. The Microinstruction Register holds the current microinstruction being executed. The register facilitates pipelined microprogram execution. The Instruction Decode logic generates various internal control signals needed for the Am9512 operation.

The Interface Control logic receives several external inputs and provides handshake related outputs to facilitate interfacing the Am9512 to microprocessors.

COMMAND FORMAT

The Operation of the Am9512 is controlled from the host processor by issuing instructions called commands. The command format is shown below:



The command consists of 8 bits; the least significant 7 bits specify the operation to be performed as detailed in the accompanying table. The most significant bit is the Service Request Enable bit. This bit must be a 1 if SVREQ is to go high at end of executing a command.

The Am9512 commands fall into three categories: Single precision arithmetic, double precision arithmetic and data manipulation. There are four arithmetic operations that can be performed with single precision (32-bit), or double precision (64-bit) floating-point numbers: add, subtract, multiply and divide. These operations require two operands. The Am9512 assumes that these operands are located in the internal stack as Top of Stack

(TOS) and Next on Stack (NOS). The result will always be returned to the previous NOS which becomes the new TOS. Results from an operation are of the same precision and format as the operands. The results will be rounded to preserve the accuracy. The actual data formats and rounding procedures are described in a later section. In addition to the arithmetic operations, the Am9512 implements eight data manipulating operations. These include changing the sign of a double or single precision operand located in TOS, exchanging single precision operands located at TOS and NOS, as well as copying and popping single or double precision operands. See also the sections on status register and operand formats.

The Execution times of the Am9512 commands are all data dependent. Table 2 shows one example of each command execution time:

Table 1. Command Decoding Table.

Command Bits								Mnemonic	Description
7	6	5	4	3	2	1	0		
X	0	0	0	0	0	0	1	SADD	Add TOS to NOS Single Precision and result to NOS. Pop stack.
X	0	0	0	0	0	1	0	SSUB	Subtract TOS from NOS Single Precision and result to NOS. Pop stack.
X	0	0	0	0	0	1	1	SMUL	Multiply NOS by TOS Single Precision and result to NOS. Pop stack.
X	0	0	0	0	1	0	0	SDIV	Divide NOS by TOS Single Precision and result to NOS. Pop stack.
X	0	0	0	0	1	0	1	CHSS	Change sign of TOS Single Precision operand.
X	0	0	0	0	1	1	0	PTOS	Push Single Precision operand on TOS to NOS.
X	0	0	0	0	1	1	1	POPS	Pop Single Precision operand from TOS. NOS becomes TOS.
X	0	0	0	1	0	0	0	XCHS	Exchange TOS with NOS Single Precision.
X	0	1	0	1	1	0	1	CHSD	Change sign of TOS Double Precision operand.
X	0	1	0	1	1	1	0	PTOD	Push Double Precision operand on TOS to NOS.
X	0	1	0	1	1	1	1	POPD	Pop Double Precision operand from TOS. NOS becomes TOS.
X	0	0	0	0	0	0	0	CLR	CLR status.
X	0	1	0	1	0	0	1	DADD	Add TOS to NOS Double Precision and result to NOS. Pop stack.
X	0	1	0	1	0	1	0	DSUB	Subtract TOS from NOS Double Precision and result to NOS. Pop stack.
X	0	1	0	1	0	1	1	DMUL	Multiply NOS by TOS Double Precision and result to NOS. Pop stack.
X	0	1	0	1	1	0	0	DDIV	Divide NOS by TOS Double Precision and result to NOS. Pop Stack.

Notes: X = Don't Care Operation for bit combinations not listed above is undefined.

Table 2. Am9512 Execution Time in Cycles.

Single Precision

	Min	Typ	Max
Add	58	220	512
Subtract	56	220	512
Multiply	192	220	254
Divide	228	240	264

Double Precision

	Min	Typ	Max
Add	578	1200	3100
Subtract	578	1200	3100
Multiply	1720	1770	1860
Divide	4560	4920	5120

Note: Typical for add and subtract, assumes the operands are within six decimal orders of magnitude. Max is derived from the maximum execution time of 1000 executions with random 32-bit or 64-bit patterns.

Table 3. Some Execution Examples.

Command	TOS	NOS	Result	Clock periods
SADD	3F800000	3F800000	40000000	58
SSUB	3F800000	3F800000	00000000	56
SMUL	40400000	3FC00000	40900000	198
SDIV	40000000	3F800000	3F000000	228
CHSS	3F800000	–	BF800000	10
PTOS	3F800000	–	–	16
POPS	3F800000	–	–	14
XCHS	3F800000	4000000	–	26
CHSD	3FF0000000000000	–	BFF0000000000000	24
PTOD	3FF0000000000000	–	–	40
POPD	3FF0000000000000	–	–	26
CLR	3FF0000000000000	–	–	4
DADD	3FF00000A0000000	8000000000000000	3FF00000A0000000	578
DSUB	3FF00000A0000000	8000000000000000	3FF00000A0000000	578
DMUL	BFF8000000000000	3FF8000000000000	C002000000000000	1748
DDIV	BFF8000000000000	3FF8000000000000	BFF0000000000000	4560

Note: TOS, NOS and Result are in hexadecimal; Clock period is in decimal.

COMMAND INITIATION

After properly positioning the required operands in the stack, a command may be issued. The procedure for initiating a command execution is as follows:

1. Establish appropriate command on the DB0-DB7 lines.
2. Establish HIGH on the C/$\overline{D}$ input.
3. Establish LOW on the $\overline{CS}$ input. Whenever $\overline{WR}$ and $\overline{RD}$ inputs are HIGH the $\overline{PAUSE}$ output follows the $\overline{CS}$ input. Hence $\overline{PAUSE}$ will become LOW.
4. Establish LOW on the $\overline{WR}$ input after an appropriate set up time (see timing diagrams).
5. Sometime after the HIGH to LOW level transition of $\overline{WR}$ input, the $\overline{PAUSE}$ output will become HIGH to acknowledge the write operation. The $\overline{WR}$ input can return to HIGH anytime after $\overline{PAUSE}$ goes HIGH. The DB0-DB7, C/$\overline{D}$ and $\overline{CS}$ inputs are allowed to change after the hold time requirements are satisfied (see timing diagram).

An attempt to issue a new command while the current command execution is in progress is allowed. Under these circumstances, the $\overline{PAUSE}$ output will not go HIGH until the current command execution is completed.

OPERAND ENTRY

The Am9512 commands operate on the operands located at the TOS and NOS and results are returned to the stack at NOS and then popped to TOS. The operands required for the Am9512 are one of two formats – single precision floating-point (4 bytes) or double precision floating-point (8 bytes). The result of an operation has the same format as the operands. In other words, operations using single precision quantities always result in a single precision result while operations involving double precision quantities will result in double precision result.

Operands are always entered into the stack least significant byte first and most significant byte last. The following procedure must be followed to enter operands into the stack:

1. The lower significant operand byte is established on the DB0-DB7 lines.
2. A LOW is established on the C/$\overline{D}$ input to specify that data is to be entered into the stack.
3. The $\overline{CS}$ input is made LOW. Whenever the $\overline{WR}$ and $\overline{RD}$ inputs are HIGH, the $\overline{PAUSE}$ output will follow the $\overline{CS}$ input. Thus $\overline{PAUSE}$ output will become LOW.
4. After appropriate set up time (see timing diagrams), the $\overline{WR}$ input is made LOW.
5. Sometime after this event, $\overline{PAUSE}$ will return HIGH to indicate that the write operation has been acknowledged.
6. Anytime after the $\overline{PAUSE}$ output goes HIGH the $\overline{WR}$ input can be made HIGH. The DB0-DB7, C/$\overline{D}$ and $\overline{CS}$ inputs can change after appropriate hold time requirements are satisfied (see timing diagrams).

The above procedure must be repeated until all bytes of the operand are pushed into the stack. It should be noted that for single precision operands 4 bytes should be pushed and 8 bytes must be pushed for double precision. Not pushing all the bytes of a quantity will result in byte pointer misalignment.

The Am9512 stack can accommodate 4 single precision quantities or 2 double precision quantities. Pushing more quantities than the capacity of the stack will result in loss of data which is usual with any LIFO stack.

REMOVING THE RESULTS

Result from an operation will be available at the TOS. Results can be transferred from the stack to the data bus by reading the stack. When the stack is popped for results, the most significant byte is available first and the least significant byte last. A result is always of the same precision as the operands that produced it. Thus when the result is taken from the stack, the total number of bytes popped out should be appropriate with the precision – single precision results are 4 bytes and double precision results are 8 bytes. The following prodedure must be used for reading the result from the stack:

1. A LOW is established on the C/$\overline{D}$ input.
2. The $\overline{CS}$ input is made LOW. When $\overline{WR}$ and $\overline{RD}$ inputs are both HIGH, the $\overline{PAUSE}$ output follows the $\overline{CS}$ input, thus $\overline{PAUSE}$ will be LOW.
3. After appropriate set up time (see timing diagrams), the $\overline{RD}$ input is made LOW.
4. Sometime after this, $\overline{PAUSE}$ will return HIGH indicating that the data is available on the DB0-DB7 lines. This data will remain on the DB0-DB7 lines as long as the $\overline{RD}$ input remains LOW.
5. Anytime after $\overline{PAUSE}$ goes HIGH, the $\overline{RD}$ input can return HIGH to complete transaction.
6. The $\overline{CS}$ and C/$\overline{D}$ inputs can change after appropriate hold time requirements are satisfied (see timing diagram).
7. Repeat this procedure until all bytes appropriate for the precision of the result are popped out.

Reading of the stack does not alter its data; it only adjusts the byte pointer. If more data is popped than the capacity of the stack, the internal byte pointer will wrap around and older data will be read again, consistent with the LIFO stack.

READING STATUS REGISTER

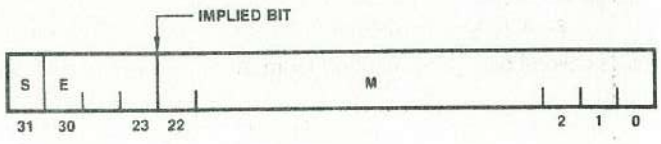
The Am9512 status register can be read without any regard to whether a command is in progress or not. The only implication that has to be considered is the effect this might have on the END and ERR outputs discussed in the signal descriptions.

The following procedure must be followed to accomplish status register reading.

1. Establish HIGH on the C/$\overline{D}$ input.
2. Establish LOW on the $\overline{CS}$ input. Whenever $\overline{WR}$ and $\overline{RD}$ inputs are HIGH, $\overline{PAUSE}$ will follow the $\overline{CS}$ input. Thus, $\overline{PAUSE}$ will go LOW.
3. After appropriate set up time (see timing diagram) $\overline{RD}$ is made LOW.
4. Sometime after the HIGH to LOW transition of $\overline{RD}$, $\overline{PAUSE}$ will become HIGH indicating that status register contents are available on the DB0-DB7 lines. These lines will contain this information as long as $\overline{RD}$ is LOW.
5. The $\overline{RD}$ input can be returned HIGH anytime after $\overline{PAUSE}$ goes HIGH.
6. The C/$\overline{D}$ input and $\overline{CS}$ input can change after satisfying appropriate hold time requirements (see timing diagram).

DATA FORMATS

The Am9512 handles floating-point quantities in two different formats – single precision and double precision. The single precision quantities are 32-bits long as shown below.



Bit 31:

S = Sign of the mantissa. 1 represents negative and 0 represents positive.

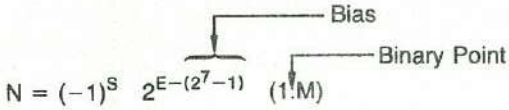
Bits 23-30

E = These 8-bits represent a biased exponent. The bias is $2^7 - 1 = 127$

Bits 0-22

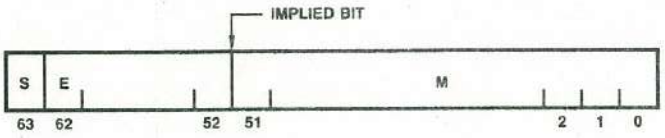
M = 23-bit mantissa. Together with the sign bit, the mantissa represents a signed fraction in sign-magitude notation. There is an implied 1 beyond the most significant bit (bit 22) of the mantissa. In other words, the mantissa is assumed to be a 24-bit normalized quantity and the most significant bit which will always be 1 due to normalization is implied. The Am9512 restores this implied bit internally before performing arithmetic; normalizes the result and strips the implied bit before returning the results to the external data bus. The binary point is between the implied bit and bit 22 of the mantissa.

The quantity N represented by the above notation is



Provided E ≠ 0 or all 1's.

A double precision quantity consists of the mantissa sign bit(s), an 11 bit biased exponent (E), and a 52-bit mantissa (M). The bias for double precision quantities is $2^{10} - 1$. The double precision format is illustrated below.



Bit 63:

S = Sign of the mantissa. 1 represents negative and 0 represents positive.

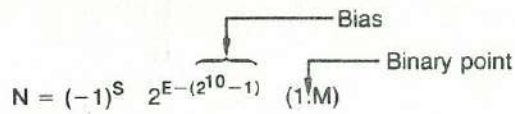
Bits 52-62

E = These 11 bits represent a biased exponent. The bias is $2^{10} - 1 = 1023$.

Bit 0-51

M = 52-bit mantissa. Together with the sign bit, the mantissa represents a signed fraction in sign-magnitude notation. There is an implied 1 beyond the most significant bit (bit 51) of the mantissa. In other words, the mantissa is assumed to a 53-bit normalized quantity and the most significant bit, which will always be a 1 due to normalization, is implied. The Am9512 restores this implied bit internally before performing arithmetic; normalizes the result and strips the implied bit before returning the result to the external data bus. The binary point is between the implied bit and bit 51 of the mantissa.

The quantity N represented by the above notation is



Provided E ≠ 0 or all 1's.

STATUS REGISTER

The Am9512 contains an 8-bit status register with the following format.

BUSY	SIGN S	ZERO Z	RESERVED	DIVIDE EXCEPTION D	EXPONENT UNDERFLOW U	EXPONENT OVERFLOW V	RESERVED
7	6	5	4	3	2	1	0

Bit 0 and bit 4 are reserved. Occurrence of exponent oerflow (V), exponent underflow (U) and divide exception (D) are indicated by bits 1, 2 and 3 respectively. An attempt to divide by zero is the only divide exception. Bits 5 and 6 represent a zero result and the sign of a result respectively. Bit 7 (Busy) of the status register indicates if the Am9512 is currently busy executing a command. All the bits are initialized to zero upon reset. Also, executing a CLR (Clear Status) command will result in all zero status register bits. A zero in Bit 7 indicates that the Am9512 is not busy and a new command may be initiated. As soon as a new command is issued, Bit 7 becomes 1 to indicate the device is busy and remains 1 until the command execution is complete, at which time it will become 0. As soon as a new command is issued, status register bits 0, 1, 2, 3, 4, 5 and 6 are cleared to zero. The status bits will be set as required during the command execution. Hence, as long as bit 7 is 1, the remainder of the status register bit indications should not be relied upon unless the ERR occurs. The following is a detailed status bit description.

Bit 0 Reserved

Bit 1 Exponent overflow (V): When 1, this bit indicates that exponent overflow has occurred. Cleared to zero otherwise.

Bit 2 Exponent Underflow (U): When 1, this bit indicates that exponent underflow has occurred. Cleared to zero otherwise.

Bit 3 Divide Exception (D): When 1, this bit indicates that an attempt to divide by zero is made. Cleared to zero otherwise.

Bit 4 Reserved

Bit 5 Zero (Z): When 1, this bit indicates that the result returned to TOS after a command is all zeros. Cleared to zero otherwise.

Bit 6 Sign (S): When 1, this bit indicates that the result returned to TOS is negative. Cleared to zero otherwise.

Bit 7 Busy: When 1, this bit indicates the Am9512 is in the process of executing a command. It will become zero after the command execution is complete.

All other status register bits are valid when the Busy bit is zero.

ALGORITHMS OF FLOATING-POINT ARITHMETIC

1. Floating Point to Decimal Conversion

   As an introduction to floating-point arithmetic, a brief description of the Decimal equivalent of the Am9512 floating-point format should help the reader to understand and verify the validity of the arithmetic operations. The Am9512 single precision format is used for the following discussions. With a minor modification of the field lengths, the discussion would also apply to the double precision format.

   There are three parts in a floating point number:

   a. The sign – the sign applies to the sign of the number. Zero means the number is positive or zero. One means the number is negative.

b. The exponent – the exponent represents the magnitude of the number. The Am9512 single precision format has an excess 127_{10} notation which means the code representation is 127_{10} higher than the actual value. The following are a few examples of actual versus coded exponent.

Actual	Coded
$+127_{10}$	$+254_{10}$
0	127_{10}
-126_{10}	$+1_{10}$

c. The mantissa – the mantissa is a 23-bit value with the binary point to the left of the most significant bit. There is a hidden 1 to the left of the binary point so the mantissa is always less than 2 and greater than or equal to 1.

To find the Decimal equivalent of the floating point number, the mantissa is multiplied by 2 to the power of the actual exponent. The number is negated if the sign bit = 1. The following are two examples of conversion:

Example 1

Floating Point No. = 0 10000011 11000000000000000000000B
Sign — Exponent — Mantissa

Coded Exponent = 10000011B
Actual Exponent = 10000011B – 01111111B = 00000100B = 4_{10}
Mantissa = 1.11000000000000000000000B
= 1 + 1/2 + 1/4 = 1.75_{10}
Decimal No. = 2^4 x 1.75 = 16 x 1.75 = 28_{10}

Example 2

Floating Point No. = 1 01111010 01100000000000000000000B
Sign — Exponent — Mantissa

Code Exponent = 01111010B
Actual Exponent = 01111010B – 01111111B = 11111011B = -5_{10}
Mantissa = 1.01100000000000000000000B
= 1 + 1/4 + 1/8 = 1.375_{10}
Decimal No. = -2^{-5} x 1.375 = $-.04296875_{10}$

2. Unpacking of the Floating-Point Numbers
The Am9512 unpacks the floating point number into three parts before any of the arithmetic operation. The number is divided into three parts as described in Section 1. The sign and exponent are copied from the original number as 1 and 8-bit numbers respectively. The mantissa is stored as a 24-bit number. The least significant 23 bits are copied from the original number and the MSB is set to 1. The binary point is assumed to the right of the MSB.

The abbreviations listed below are used in the following sections of algorithm description:

SIGN – Sign of Result
EXP – Exponent of Result
MAN – Mantissa of Result
SIGN (TOS) – Sign of Top of Stack
EXP (TOS) – Exponent of Top of Stack
MAN (TOS) – Mantissa of Top of Stack
SIGN (NOS) – Sign of Next on Stack
EXP (NOS) – Exponent of Next on Stack
MAN (NOS) – Mantissa of Next on Stack

3. Floating-Point Add/Subtract
The floating-point add and subtract essentially use the same algorithm. The only difference is that floating-point subtract changes the sign of the floating-point number at top of stack and then performs the floating-point add.

The following is a step by step description of a floating-point add algorithm (Figure 1):

a. Unpack TOS and NOS.
b. The exponent of TOS is compared to the exponent of NOS.
c. If the exponents are equal, go to step f.
d. Right shift the mantissa of the number with the smaller exponent.
e. Increment the smaller exponent and go to step b.
f. Set sign of result to sign of larger number.
g. Set exponent of result to exponent of larger number.
h. If sign of the two numbers are not equal, go to m.
i. Add Mantissas.
j. Right shift resultant mantissa by 1 and increment exponent of result by 1.
k. If MSB of exponent changes from 1 to 0 as a result of the increment, set overflow status.
l. Round if necessary and exit.
m. Subtract smaller mantissa from larger mantissa.
n. Left shift mantissa and decrement exponent of result.
o. If MSB of exponent changes from 0 to 1 as a result of the decrement, set underflow status and exit.
p. If the MSB of the resultant mantissa = 0, go to n.
q. Round if necessary and exit.

4. Floating-Point Multiply
Floating-point multiply basically involves the addition of the exponents and multiplication of the mantissas. The following is a step by step description of a floating multiplication algorithm (Figure 2):

a. Check if TOS or NOS = 0.
b. If either TOS or NOS = 0, Set result to 0 and exit.
c. Unpack TOS and NOS.
d. Convert EXP (TOS) and EXP (NOS) to unbiased form.
EXP (TOS) = EXP (TOS) -127_{10}
EXP (NOS) = EXP (NOS) -127_{10}
e. Add exponents.
EXP = EXP (TOS) + EXP (NOS)
f. If MSB of EXP (TOS) = MSB of EXP (NOS) = 0 and MSB of EXP = 1, then set overflow status and exit.
g. If MSB of EXP (TOS) = MSB of EXP (NOS) = 1 and MSB of EXP = 0, then set underflow status and exit.
h. Convert Exponent back to biased form.
EXP = EXP + 127_{10}
i. If sign of TOS = sign of NOS, set sign of result to 0, else set sign of result to 1.
j. Multiply mantissa.
k. If MSB of resultant = 1, right shift mantissa by 1 and increment exponent of resultant.
l. If MSB of exponent changes from 1 to 0 as a result of the increment, set overflow status.
m. Round if necessary and exit.

5. Floating-Point Divide
The floating-point divide basically involves the subtraction of exponents and the division of mantissas. The following is a step by step description of a division algorithm (Figure 3).

a. If TOS = 0, set divide exception error and exit.
b. If NOS = 0, set result to 0 and exit.
c. Unpack TOS and NOS.
d. Convert EXP (TOS) and EXP (NOS) to unbiased form.
EXP (TOS) = EXP (TOS) – 127_{10}
EXP (NOS) = EXP (NOS) – 127_{10}
e. Subtract exponent of TOS from exponent of NOS.
EXP = EXP (NOS) – EXP (TOS)
f. If MSB of EXP (NOS) = 0, MSB of EXP (TOS) = 1 and MSB of EXP = 1, then set overflow status and exit.
g. If MSB of EXP (NOS) = 1, MSB of EXP (TOS) = 0, and MSB of EXP = 0, then set underflow status and exit.

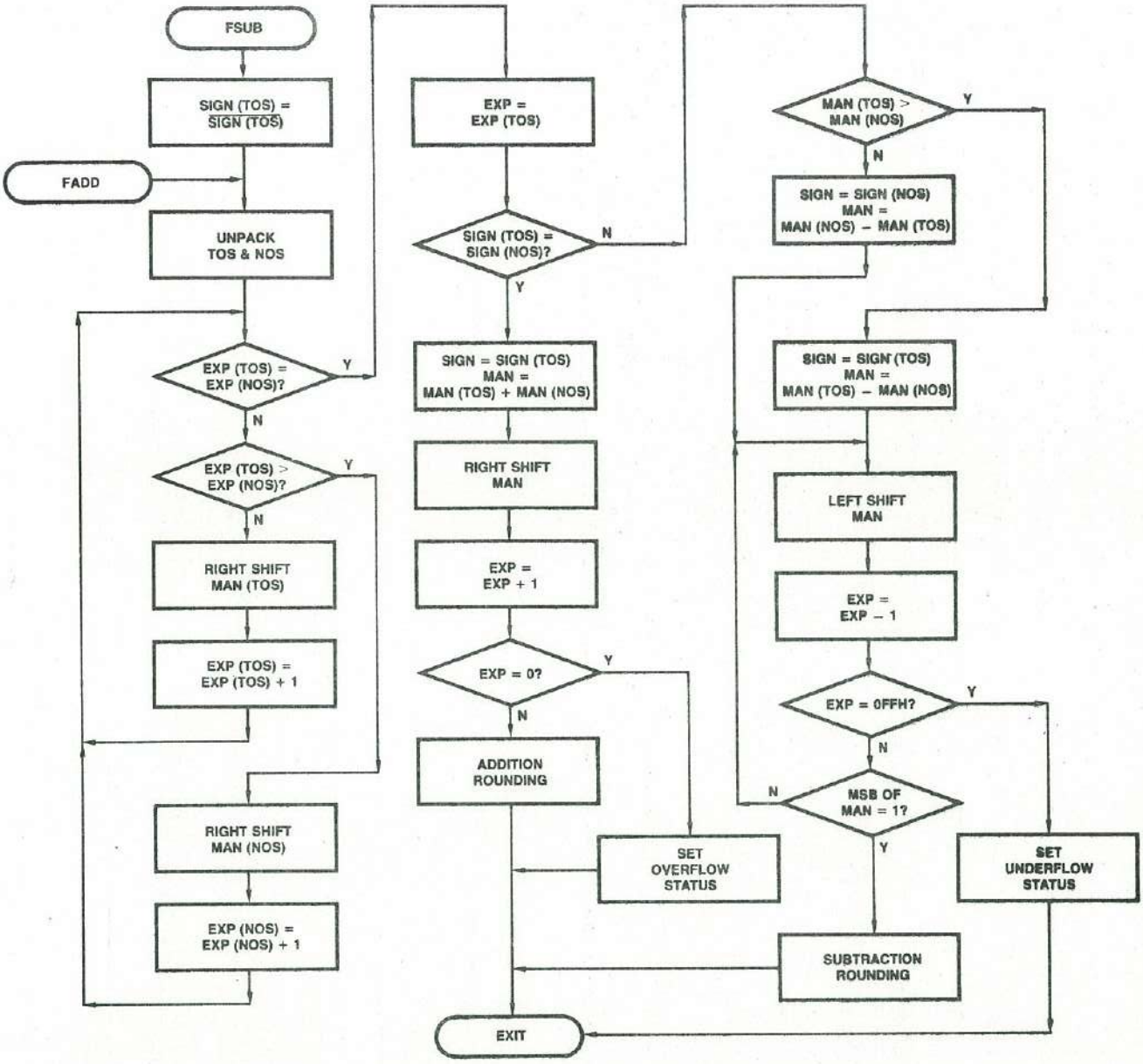


Figure 1. Conceptual Floating-Point Addition/Subtraction.

MOS-205

h. Add bias to exponent of result.
   $EXP = EXP + 127_{10}$
i. If sign of TOS = sign of NOS, set sign of result to 0, else set sign of result to 1.
j. Divide mantissa of NOS by mantissa of TOS.
k. If MSB = 0, left shift mantissa and decrement exponent of resultant, else go to n.
l. If MSB of exponent changes from 0 to 1 as a result of the decrement, set underflow status.
m. Go to k.
n. Round if necessary and exit.

The algorithms described above provide the user a means of verifying the validity of the result. They do not necessarily reflect the exact internal sequence of the Am9512.

6. Rounding

The Am9512 adopts a rounding algorithm that is consistent with the Intel® standard for floating-point arithmetic. The following description is an excerpt from the paper published in proceedings of Compsac 77, November 1977, pp. 107-112 by Dr. John F. Palmer of Intel Corporation.

The method used for doing the rounding during floating-point arithmetic is known as "Round to Even", i.e., if the resultant number is exactly halfway between two floating point numbers, the number is rounded to the nearest floating-point number whose LSB of the mantissa is 0. In order to simplify the explanation, the algorithms will be illustrated with 4-bit arithmetic. The existence of an accumulator will be assumed as shown:

OF	B1	B2	B3	B4	G	R	ST

The bit labels denote:

OF – The overflow bit
B1-B4 – The 4 mantissa bits
G – The Guard bit
R – The Rounding bit
ST – The "Sticky" bit

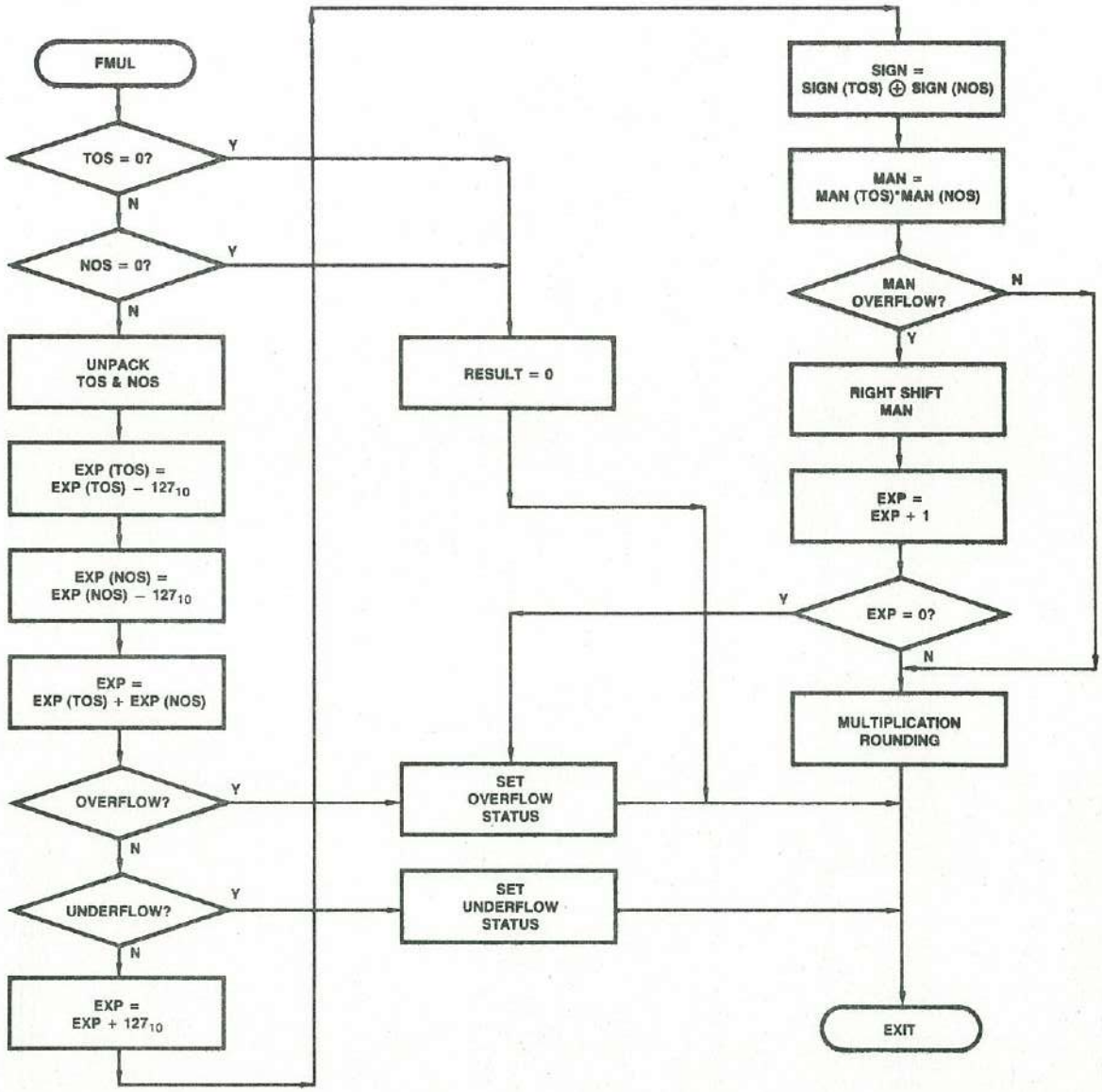


Figure 2. Conceptual Floating-Point Multiplication.

MOS-206

The Sticky bit is set to one if any ones are shifted right of the rounding bit in the process of denormalization. If the Sticky bit becomes set, it remains set throughout the operation. All shifting in the Accumulator involves the OF, G, R and ST bits. The ST bit is not affected by left shifts but, zeros are introduced into OF by right shifts.

Rounding during addition of magnitudes – add 1 to the G position, then if G=R=ST=0, set B4 to 0 ("Rounding to Even").

Rounding during subtraction of magnitudes – if more than one left shift was performed, no rounding is needed, otherwise round the same way as addition of magnitudes.

Rounding during multiplication – let the normalized double length product be:

B1	B2	B3	B4	B5	B6	B7	B8

Then G=B5, R=B6, ST=B7 V B8. The rounding is then performed as in addition of magnitudes.

Rounding during division – let the first six bits of the normalized quotient be

B1	B2	B3	B4	B5	B6

Then G=B5, R=B6, ST=0 if and only if remainder = 0. The rounding is then performed as in addition of magnitudes.

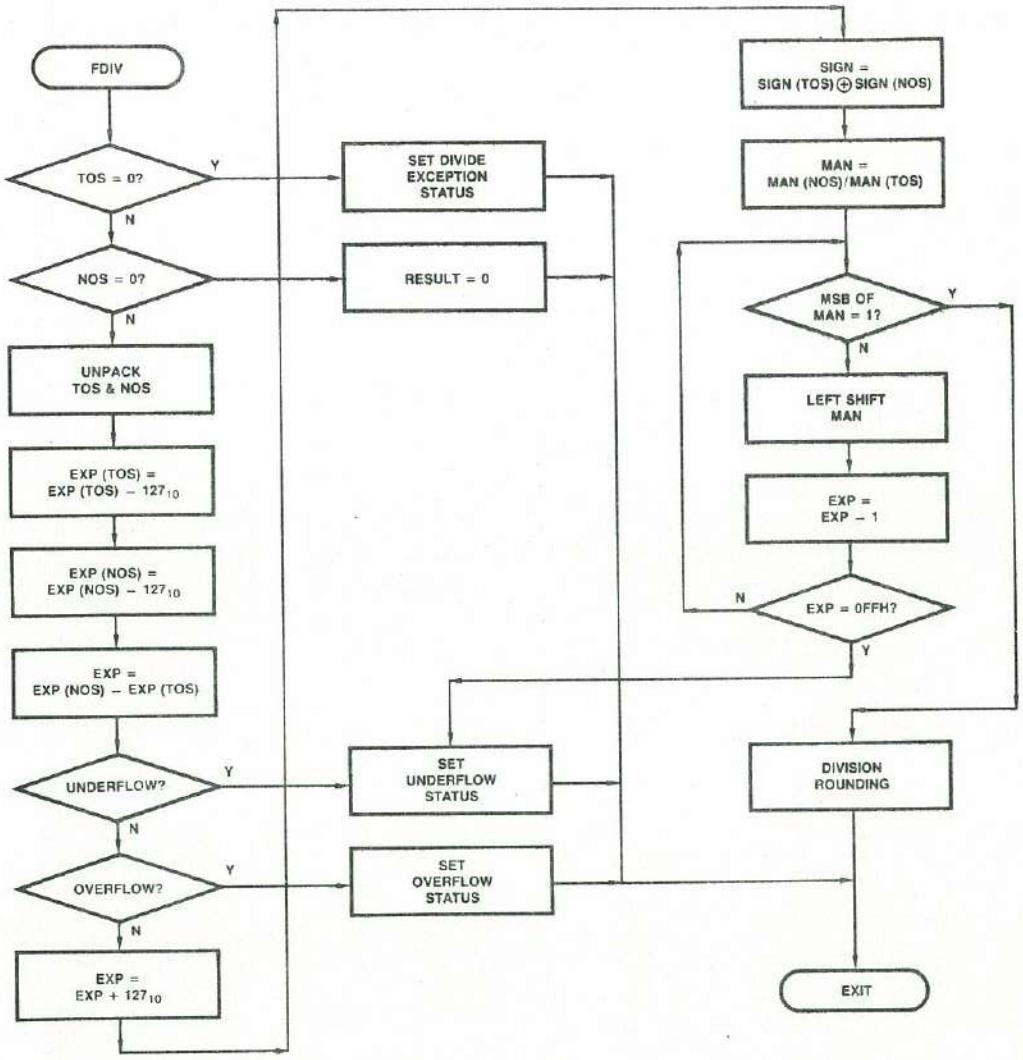


Figure 3. Conceptual Floating-Point Division.

MOS-207

CHSD

CHANGE SIGN DOUBLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	1	0	1

Hex Coding: AD IF SRE = 1
2D IF SRE = 0

Execution Time: See Table 2

Description:

The sign of the double precision TOS operand A is complemented. The double precision result R is returned to TOS. If the double precision operand A is zero, then the sign is not affected. The status bit S and Z indicate the sign of the result and if the result is zero. The status bits U, V and D are always cleared to zero.

Status Affected: S, Z. (U, V, D always zero.)

STACK CONTENTS

BEFORE		AFTER
A	TOS	R
B	NOS	B

CHSS

CHANGE SIGN SINGLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	1	0	1

Hex Coding: 85 IF SRE = 1
05 IF SRE = 0

Execution Time: See Table 2

Description:

The sign of the single precision operand A at TOS is complemented. The single precision result R is returned to TOS. If the exponent field of A is zero, all bits of R will be zeros. The status bits S and Z indicate the sign of the result and if the result is zero. The status bits U, V and D are cleared to zero.

Status Affected: S, Z. (U, V, D always zero.)

STACK CONTENTS

BEFORE		AFTER
A	← TOS →	R
B	← NOS →	B
C		C
D		D

CLR

CLEAR STATUS

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	0	0	0

Hex Coding: 80 IF SRE = 1
00 IF SRE = 0

Execution Time: 4 clock cycles

Description:
The status bits S, Z, D, U, V are cleared to zero. The stack is not affected. This essentially is a no operation command as far as operands are concerned.

Status Affected: S, Z, D, U, V always zero.

DADD

DOUBLE PRECISION FLOATING-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	0	0	1

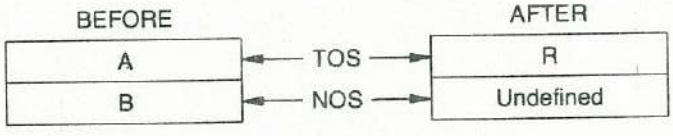
Hex Coding: A9 IF SRE = 1
29 IF SRE = 0

Execution Time: See Table 2

Description:
The double precision operand A from TOS is added to the double precision operand B from NOS. The result is rounded to obtain the final double precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENTS



DSUB

DOUBLE PRECISION FLOATING-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	0	1	0

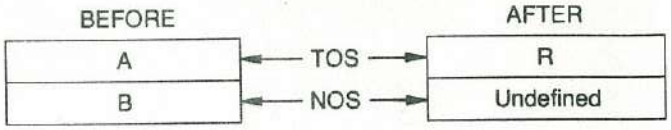
Hex Coding: AA IF SRE = 1
2A IF SRE = 0

Execution Time: See Table 2

Description:
The double precision operand A at TOS is subtracted from the double precision operand B at NOS. The result is rounded to obtain the final double precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENTS



DMUL

DOUBLE PRECISION FLOATING-POINT MULTIPLY

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	0	1	1

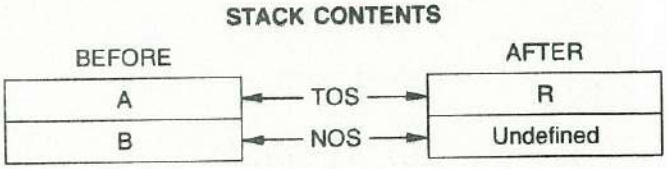
Hex Coding: AB IF SRE = 1
2B IF SRE = 0

Execution Time: See Table 2

Description:
The double precision operand A from TOS is multiplied by the double precision operand B from NOS. The result is rounded to obtain the final double precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENTS



DDIV

DOUBLE PRECISION FLOATING-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Code:	SRE	0	1	0	1	1	0	0

Hex Coding: AC IF SRE = 1
2C IF SRE = 0

Execution Time: See Table 2

Description:

The double precision operand B from NOS is divided by the double precision operand A from TOS. The result (quotient) is rounded to obtain the final double precision result R which is returned to TOS. The status bits, S, Z, D, U and V are affected to report sign of the result, if the result is zero, attempt to divide by zero, exponent underflow and exponent overflow respectively.

Status Affected: S, Z, D, U, V

STACK CONTENT

BEFORE		AFTER
A	← TOS →	R (see note)
B	← NOS →	Undefined

Note: If A is zero, then R = B (Divide exception).

SADD

SINGLE PRECISION FLOATING-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	0	0	1

Hex Coding: 81 IF SRE = 1
01 IF SRE = 0

Execution Time: See Table 2

Description:

The single precision operand A from TOS is added to the single precision operand B from NOS. The result is rounded to obtain the final single precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report the sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENT

BEFORE		AFTER
A	← TOS →	R
B	← NOS →	C
C		D
D		Undefined

SSUB

SINGLE PRECISION FLOATING-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	0	1	0

Hex Coding: 82 IF SRE = 1
02 IF SRE = 0

Execution Time: See Table 2

Description:

The single precision operand A at TOS is subtracted from the single precision operand B at NOS. The result is rounded to obtain the final single precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report the sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENTS

BEFORE		AFTER
A	← TOS →	R
B	← NOS →	C
C		D
D		Undefined

SMUL

SINGLE PRECISION FLOATING-POINT MULTIPLY

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	0	1	1

Hex Coding: 83 IF SRE = 1
03 IF SRE = 0

Execution Time: See Table 2

Description:

The single precision operand A from TOS is multiplied by the single precision operand B from NOS. The result is rounded to obtain the final single precision result R which is returned to TOS. The status bits S, Z, U and V are affected to report the sign of the result, if the result is zero, exponent underflow and exponent overflow respectively. The status bit D will be cleared to zero.

Status Affected: S, Z, U, V. (D always zero.)

STACK CONTENTS

BEFORE		AFTER
A	← TOS →	R
B	← NOS →	C
C		D
D		Undefined

SDIV

SINGLE PRECISION FLOATING-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	1	0	0

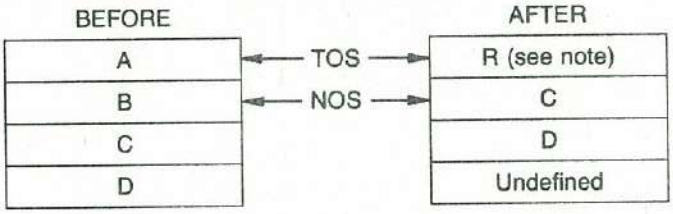
Hex Coding: 84 IF SRE = 1
04 IF SRE = 0

Execution Time: See Table 2

Description:
The single precision operand B from NOS is divided by the single precision operand A from TOS. The result (quotient) is rounded to obtain the final result R which is returned to TOS. The status bits S, Z, D, U and V are affected to report the sign of the result, if the result is zero, attempt to divide by zero, exponent underflow and exponent overflow respectively.

Status Affected: S, Z, D, U, V

STACK CONTENTS



Note: If exponent field of A is zero then R = B (Divide exception).

POPS

POP STACK SINGLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	1	1	1

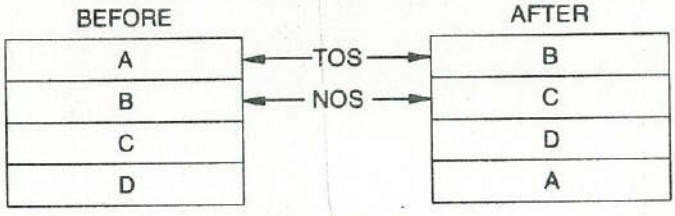
Hex Coding: 87 IF SRE = 1
07 IF SRE = 0

Execution Time: See Table 2

Description:
The single precision operand A is popped from the stack. The internal stack control mechanism is such that A will be written at the bottom of the stack. The status bits S and Z are affected to report the sign of the new operand at TOS and if it is zero, respectively. The status bits U, V and D will be cleared to zero. Note that only the exponent field of the new TOS is checked for zero, if it is zero status bit Z will set to 1.

Status Affected: S, Z. (U, V, D always zero.)

STACK CONTENTS



PTOD

PUSH STACK DOUBLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	1	1	0

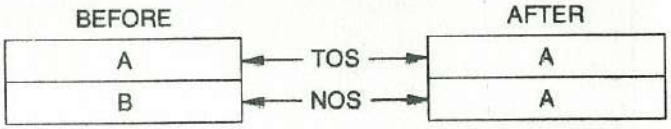
Hex Coding: AE IF SRE = 1
2E IF SRE = 0

Execution Time: See Table 2

Description:
The double precision operand A from the TOS is pushed back on to the stack. This is effectively a duplication of A into two consecutive stack locations. The status S and Z are affected to report sign of the new TOS and if the new TOS is zero respectively. The status bits U, V and D will be cleared to zero.

Status Affected: S, Z. (U, V, D always zero.)

STACK CONTENTS



PTOS

PUSH STACK SINGLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	0	1	1	0

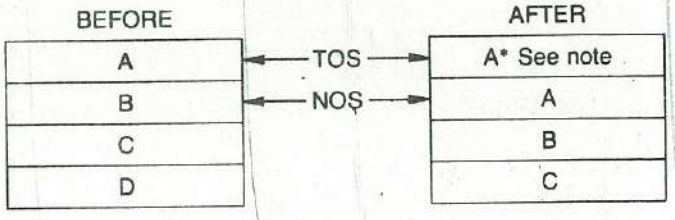
Hex Coding: 86 IF SRE = 1
06 IF SRE = 0

Execution Time: See Table 2

Description:
This instruction effectively pushes the single precision operand from TOS on to the stack. This amounts to duplicating the operand at two locations in the stack. However, if the operand at TOS prior to the PTOS command has only its exponent field as zero, the new content of the TOS will all be zeroes. The contents of NOS will be an exact copy of the old TOS. The status bits S and Z are affected to report the sign of the new TOS and if the content of TOS is zero, respectively. The status bits U, V and D will be cleared to zero.

Status Affected: S, Z. (U, V, D always zero.)

STACK CONTENTS



Note: A* = A if Exponent field of A is not zero.
A* = 0 if Exponent field of A is zero.

POPD

POP STACK DOUBLE PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	1	0	1	1	1	1

Hex Coding: AF IF SRE = 1
2F IF SRE = 0

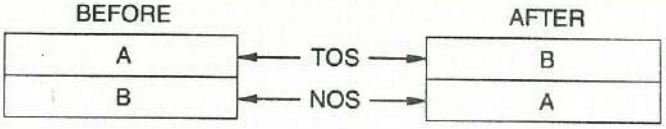
Execution Time: See Table 2

Description:

The double precision operand A is popped from the stack. The internal stack control mechanism is such that A will be written at the bottom of the stack. This operation has the same effect as exchanging TOS and NOS. The status bits S and Z are affected to report the sign of the new operand at TOS and if it is zero, respectively. The status bits U, V and D will be cleared to zero.

Status Affected: S, Z (U, V and D always zero.)

STACK CONTENTS



XCHS

EXCHANGE TOS AND NOS SINGLE-PRECISION

	7	6	5	4	3	2	1	0
Binary Coding:	SRE	0	0	0	1	0	0	0

Hex Coding: 88 IF SRE = 1
08 IF SRE = 0

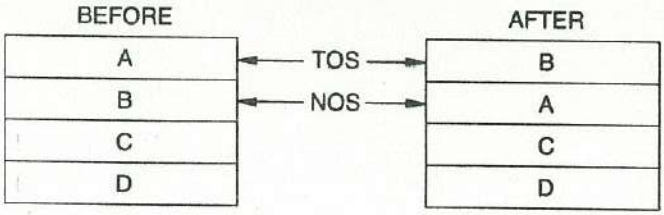
Execution Time: See Table 2

Description:

The single precision operand A at the TOS and the single precision operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All other operands are unchanged.

Status Affected: S, Z (U, V and D always zero.)

STACK CONTENTS



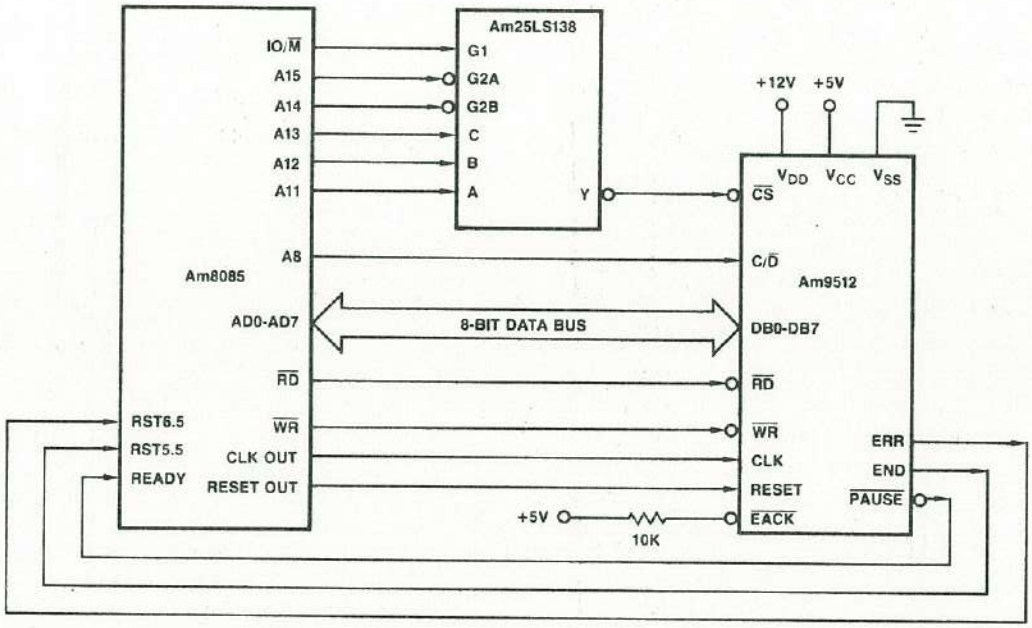


Figure 1. Am9512 to Am8085 Interface.

MOS-213

MAXIMUM RATINGS beyond which useful life may be impaired

Storage Temperature	−65°C to +150°C
Ambient Temperature Under Bias	−55°C to +125°C
VDD with Respect to VSS	−0.5V to +15.0V
VCC with Respect to VSS	−0.5V to +7.0V
All Signal Voltages with Respect to VSS	−0.5V to +7.0V
Power Dissipation (Package Limitation)	2.0W

The products described by this specification include internal circuitry designed to protect input devices from damaging accumulations of static charge. It is suggested, nevertheless, that conventional precautions be observed during storage, handling and use in order to avoid exposure to excessive voltages.

OPERATING RANGE

Part Number	Ambient Temperature	VSS	VCC	VDD
Am9512DC	$0°C \leq T_A \leq +70°C$	0V	+5.0V ±5%	+12V ±5%
Am9512DM	$-55°C \leq T_A \leq +125°C$	0V	+5.0V ±10%	+12V ±10%

ELECTRICAL CHARACTERISTICS Over Operating Range (Note 1)

Parameters	Description	Test Conditions	Min.	Typ.	Max.	Units
VOH	Output HIGH Voltage	IOH = −200μA	3.7			Volts
VOL	Output LOW Voltage	IOL = 3.2mA			0.4	Volts
VIH	Input HIGH Voltage		2.0		VCC	Volts
VIL	Input LOW Voltage		−0.5		0.8	Volts
IIX	Input Load Current	VSS ≤ VI ≤ VCC			±10	μA
IOZ	Data Bus Leakage	VO = 0.4V			10	μA
		VO = VCC			10	
ICC	VCC Supply Current	T_A = +25°C		50	90	mA
		T_A = 0°C			95	
		T_A = −55°C			100	
IDD	VDD Supply Current	T_A = +25°C		50	90	mA
		T_A = 0°C			95	
		T_A = −55°C			100	
CO	Output Capacitance	fc = 1.0MHz, Inputs = 0V		8	10	pF
CI	Input Capacitance			5	8	pF
CIO	I/O Capacitance			10	12	pF

SWITCHING CHARACTERISTICS

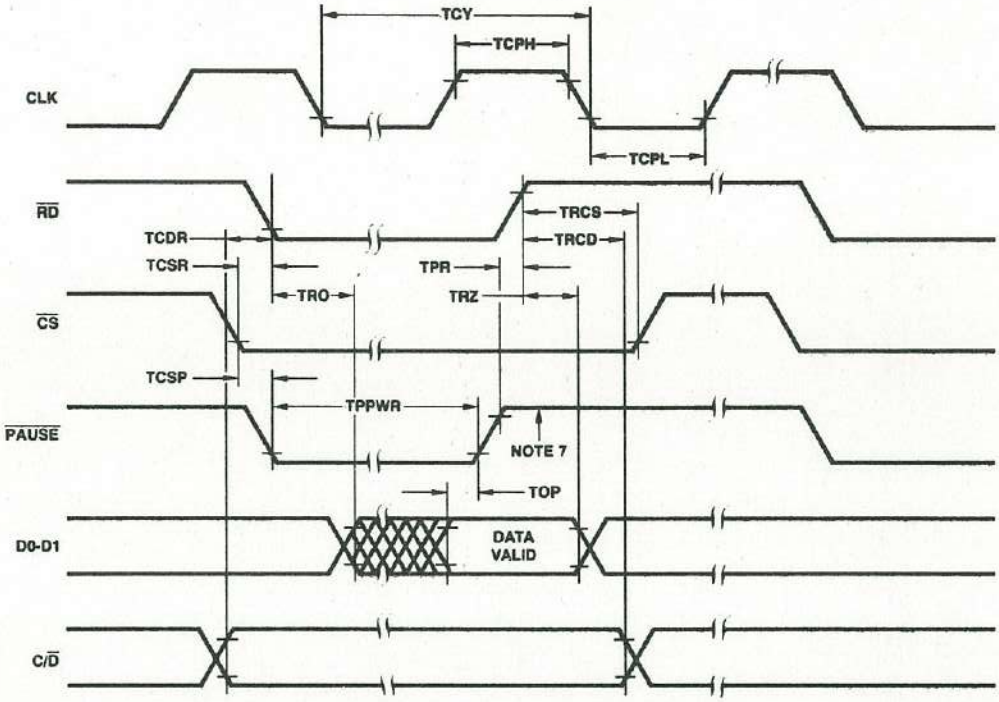
Parameters	Description		Am9512DC Min	Am9512DC Max	Am9512-1DC Min	Am9512-1DC Max	Units
TAPW	$\overline{EACK}$ LOW Pulse Width		100		75		ns
TCDR	C/$\overline{D}$ to $\overline{RD}$ LOW Set-up Time		0		0		ns
TCDW	C/$\overline{D}$ to $\overline{WR}$ LOW Set-up Time		0		0		ns
TCPH	Clock Pulse HIGH Width		200	500	140	500	ns
TCPL	Clock Pulse LOW Width		240		160		ns
TCSP	$\overline{CS}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 5)		150		100		ns
TCSR	$\overline{CS}$ to $\overline{RD}$ LOW Set-up Time		0		0		ns
TCSW	$\overline{CS}$ LOW to $\overline{WR}$ LOW Set-up Time		0		0		ns
TCY	Clock Period		480	5000	320	2000	ns
TDW	Data Valid to $\overline{WR}$ HIGH Delay		150		100		ns
TEAE	$\overline{EACK}$ LOW to END LOW Delay			200		175	ns
TEHPHR	END HIGH to $\overline{PAUSE}$ HIGH Data Read when Busy			5.5TCY+300		5.5TCY+200	ns
TEHPHW	END HIGH to $\overline{PAUSE}$ HIGH Write when Busy			200		175	ns
TEPW	END HIGH Pulse Width		400		300		ns
TEX	Execution Time		See Table 2				ns
TOP	Data Bus Output Valid to $\overline{PAUSE}$ HIGH Delay		0		0		ns
TPPWR	$\overline{PAUSE}$ LOW Pulse Width Read	Data	3.5TCY+50	5.5TCY+300	3.5TCY+50	5.5TCY+200	ns
		Status	1.5TCY+50	3.5TCY+300	1.5TCY+50	3.5TCY+200	
TPPWRB	END HIGH to $\overline{PAUSE}$ HIGH Read when Busy	Data	See Table 2				ns
		Status	1.5TCY+50	3.5TCY+300	1.5TCY+50	3.5TCY+200	
TPPWW	$\overline{PAUSE}$ LOW Pulse Width Write when Not Busy			TCSW+50		TCSW+50	ns
TPPWWB	$\overline{PAUSE}$ LOW Pulse Width Write when Busy		See Table 2				ns
TPR	$\overline{PAUSE}$ HIGH to Read HIGH Hold Time		0		0		ns
TPW	$\overline{PAUSE}$ HIGH to Write HIGH Hold Time		0		0		ns
TRCD	$\overline{RD}$ HIGH to C/$\overline{D}$ Hold Time		0		0		ns
TRCS	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH Hold Time		0		0		ns
TRO	$\overline{RD}$ LOW to Data Bus On Delay		50		50		ns
TRZ	$\overline{RD}$ HIGH to Data Bus Off Delay		50	200	50	150	ns
TSAPW	SVACK LOW Pulse Width		100		75		ns
TSAR	SVACK LOW to SVREQ LOW Delay			300		200	ns
TWCD	$\overline{WR}$ HIGH to C/D Hold Time		60		30		ns
TWCS	$\overline{WR}$ HIGH to CS HIGH Hold Time		60		30		ns
TWD	$\overline{WR}$ HIGH to Data Bus Hold Time		20		20		ns

NOTES:

1. Typical values are for T_A = 25°C, nominal supply voltages and nominal processing parameters.
2. Switching parameters are listed in alphabetical order.
3. Test conditions assume transition times of 20ns or less, output loading of one TTL gate plus 100pF and timing reference levels of 0.8V and 2.0V.
4. END HIGH pulse width is specified for $\overline{EACK}$ tied to VSS. Otherwise TEAE applies.
5. $\overline{PAUSE}$ is pulled low for both command and data operations.
6. TEX is the execution time of the current command (see the Command Execution Times table).
7. $\overline{PAUSE}$ will go low at this point if $\overline{CS}$ is low and $\overline{RD}$ and $\overline{WR}$ are high.

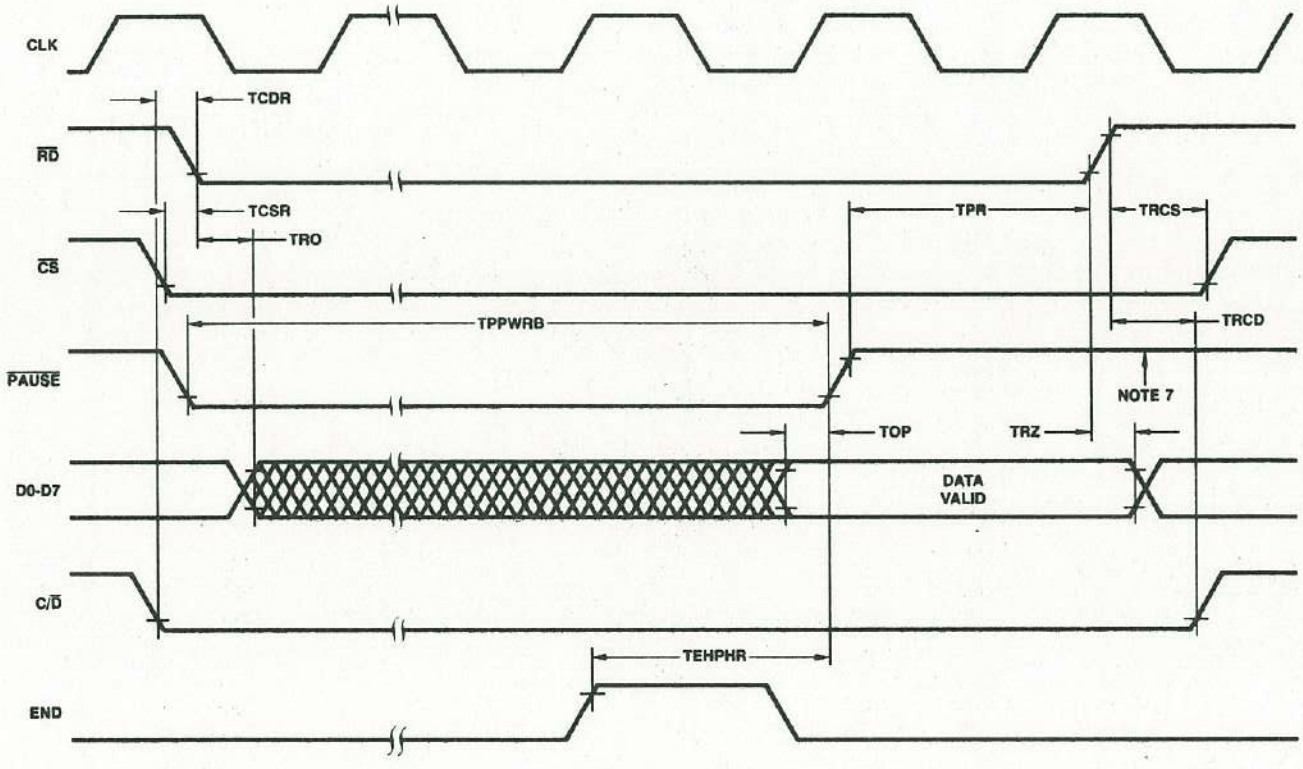
TIMING DIAGRAMS

READ OPERATION



MOS-208

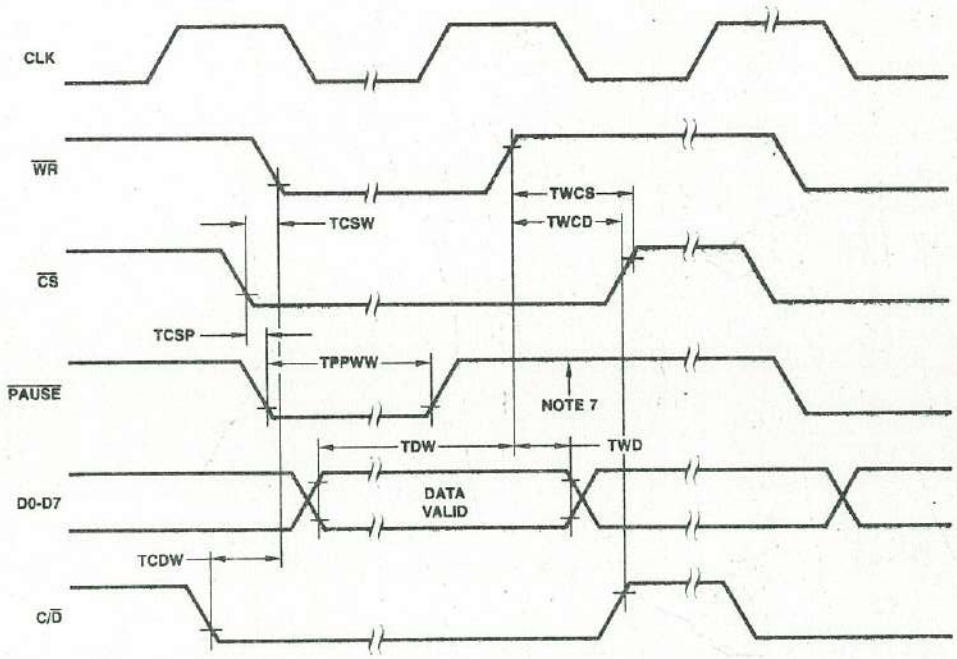
OPERAND READ WHEN Am9512 IS BUSY



MOS-209

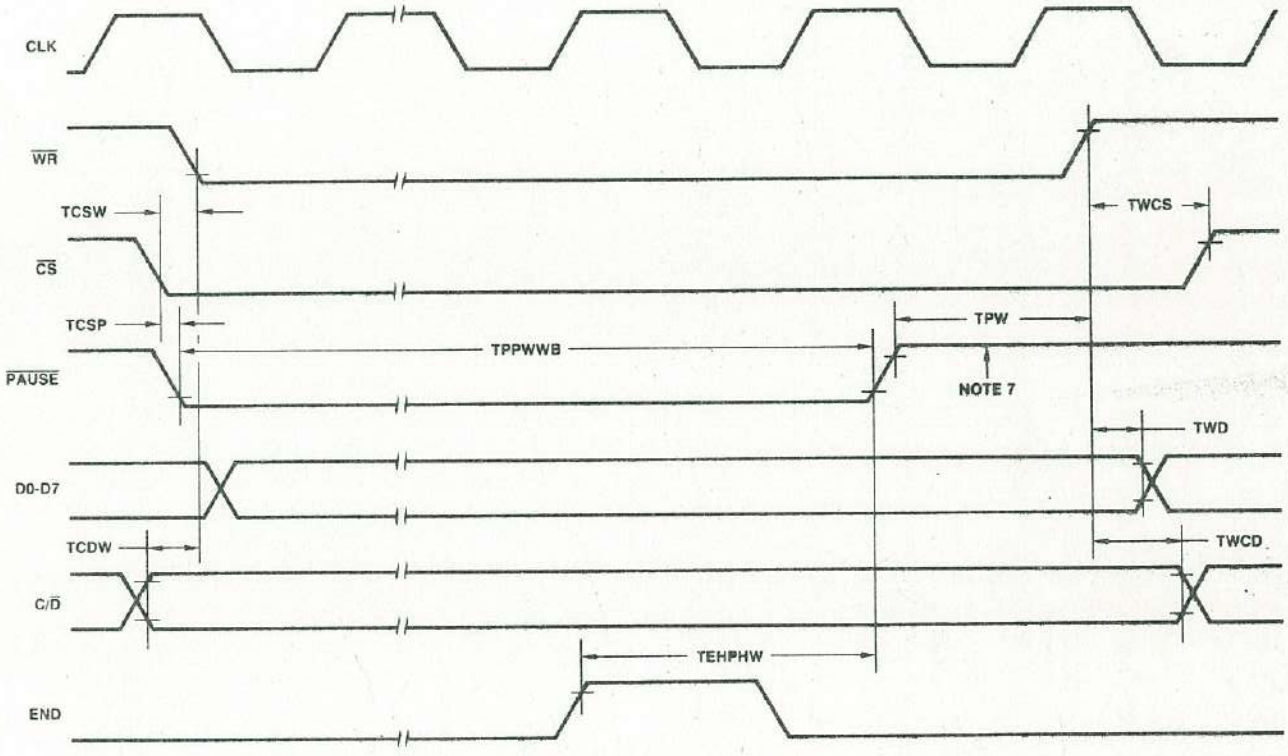
TIMING DIAGRAMS (Cont.)

OPERAND ENTRY



MOS-210

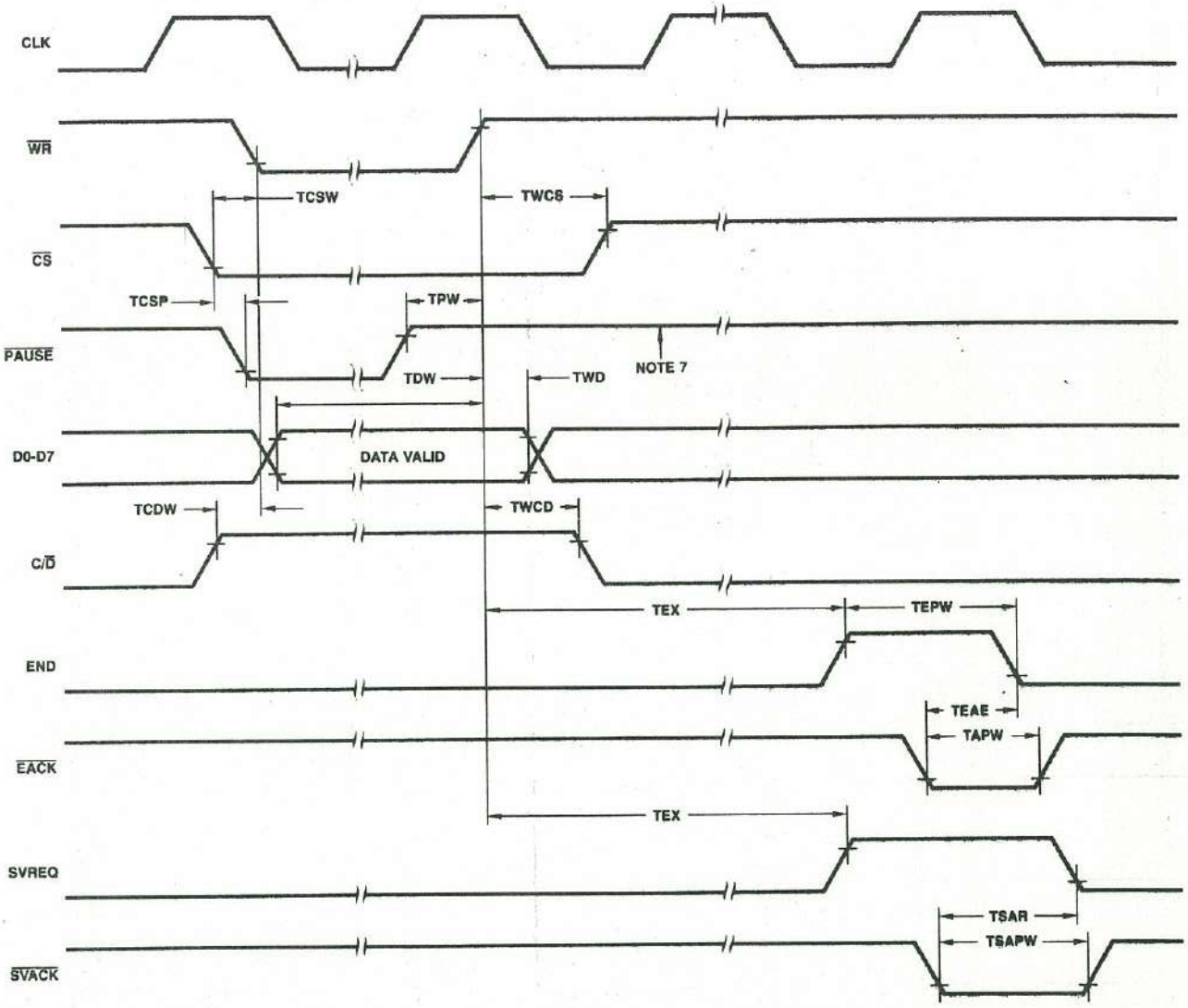
COMMAND OR DATA WRITE WHEN Am9512 IS BUSY



MOS-211

TIMING DIAGRAMS (Cont.)

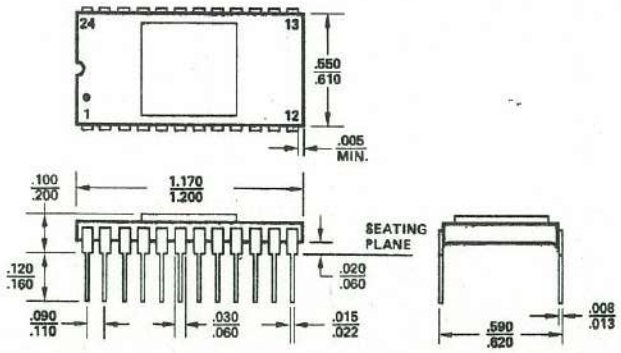
COMMAND INITIATION



MOS-212

PHYSICAL DIMENSIONS
Dual-In-Line

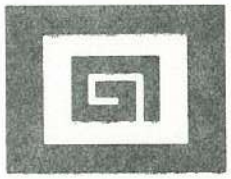
24-Pin Side-Brazed



ADVANCED MICRO DEVICES, INC.
901 Thompson Place
P.O. Box 453
Sunnyvale,
California 94086
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450

Appendix E

AY-3-8910 Data Sheet



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

GENERAL INSTRUMENT CORPORATION
MICROELECTRONICS

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

Table of Contents

1. INTRODUCTION

1.1 Description 5
1.2 Features 6
1.3 Scope 6

2. ARCHITECTURE

2.1 Basic Functional Blocks 7
2.1.1 Register Array 7
2.1.2 Sound Generating Blocks 10
2.1.3 I/O Ports 10
2.2 Pin Assignments 12
2.3 Pin Functions 13
2.4 Bus Timing 15
2.5 State Timing 16
2.5.1 Address PSG Register Sequence 16
2.5.2 Write Data to PSG Sequence 17
2.5.3 Read Data from PSG Sequence 17
2.5.4 Write to/Read from I/O Port Sequence 17

3. OPERATION

3.1 Tone Generator Control 18
3.2 Noise Generator Control 20
3.3 Mixer Control—I/O Enable 21
3.4 Amplitude Control 22
3.5 Envelope Generator Control 24
3.5.1 Envelope Period Control 24
3.5.2 Envelope Shape/Cycle Control 25
3.6 I/O Port Data Store 28
3.7 D/A Converter Operation 29

4. INTERFACING

4.1 Introduction 32
4.2 Clock Generation 33
4.3 Audio Output Interface 34
4.4 External Memory Access 35
4.5 Microprocessor/Microcomputer Interface 36
4.6 Interfacing to the PIC 1650 37
4.6.1 Write Data Routine 37
4.6.2 Read Data Routine 38
4.6.3 Read ROM Routine 38
4.7 Interfacing to the CP1600/1610 40
4.7.1 Write Data Routine 40
4.7.2 Read Data Routine 40
4.8 Interfacing to the M6800 42
4.8.1 Latch Address Routine 42
4.8.2 Write Data Routine 42
4.8.3 Read Data Routine 42
4.9 Interfacing to the 8080 S100 Bus 44
4.9.1 Latch Address Routine 44
4.9.2 Write Data Routine 44
4.9.3 Read Data Routine 44

5. MUSIC GENERATION
5.1 Note Generation 46
5.2 Tune Entry/Playback 48
5.3 Tune Variation 48
5.3.1 Octave Shift 48
5.3.2 Key 48
5.3.3 Tempo 48
5.3.4 Chords 49
5.4 Sound Variation 50
5.4.1 Relative Channel Volume 50
5.4.2 Decay 50
5.4.3 Other Effects 50
5.5 Applications 51
5.5.1 Organ Envelope Generation 51
5.5.2 Organ Rhythm Generation 52

6. SOUND EFFECTS GENERATION
6.1 Tone Only Effects 53
6.2 Noise Only Effects 54
6.3 Frequency Sweep Effects 55
6.4 Multi-Channel Effects 56

7. ELECTRICAL SPECIFICATIONS
7.1 Maximum Ratings 57
7.2 Standard Conditions 57
7.3 DC Characteristics 57
7.4 AC Characteristics 58
7.5 Package Outlines 60

List of Illustrations

Fig. 1 Typical System Diagram 6
Fig. 2 PSG Block Diagram 8
Fig. 3 PSG Register Array 11
Fig. 4 AY-3-8910 Pin Assignments 12
Fig. 5 AY-3-8912 Pin Assignments 12
Fig. 6 Variable Amplitude Control 23
Fig. 7 Envelope Shape/Cycle Control 26
Fig. 8 Detail of Two Cycles of Fig. 7 27
Fig. 9 D/A Converter Output 29
Fig. 10 Single Tone with Envelope Shape/Cycle Pattern 1000 30
Fig. 11 Single Tone with Envelope Shape/Cycle Pattern 1100 30
Fig. 12 Single Tone with Envelope Shape/Cycle Pattern 1010 31
Fig. 13 Mixture of Three Tones with Fixed Amplitudes 31
Fig. 14 System Block Diagram 32
Fig. 15 Clock Generation 33
Fig. 16 Audio Output Interface 34
Fig. 17 External Memory Access 35
Fig. 18 Microprocessor/Microcomputer Interface 36
Fig. 19 PIC 1650/AY-3-8910 System Example 39
Fig. 20 CP1600/1610/AY-3-8910 Interface 41
Fig. 21 M6800/AY-3-8910 Interface 43
Fig. 22 8080 S100 Bus/AY-3-8910 Interface 45
Fig. 23 Equal Tempered Chromatic Scale 47
Fig. 24 Chord Selection Chart 49
Fig. 25 Organ Envelope Generation 51
Fig. 26 Organ Rhythm Generation 52
Fig. 27 European Siren Sound Effect Chart 53
Fig. 28 Gunshot Sound Effect Chart 54
Fig. 29 Explosion Sound Effect Chart 54

List of Illustrations (cont.)

Fig. 30 Laser Sound Effect Chart 55
Fig. 31 Whistling Bomb Sound Effect Chart 55
Fig. 32 Wolf Whistle Sound Effect Chart 56
Fig. 33 Race Car Sound Effect Chart 56
Fig. 34 Analog Channel Output Test Circuit 57
Fig. 35 Current to Voltage Converter 57
Fig. 36 Clock and Bus Signal Timing 58
Fig. 37 Reset Timing 58
Fig. 38 Latch Address Timing 59
Fig. 39 Write Data Timing 59
Fig. 40 Read Data Timing 59
Fig. 41 40 Lead Dual In Line Packages 60
Fig. 42 28 Lead Dual In Line Packages 61

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.

1.2 Features

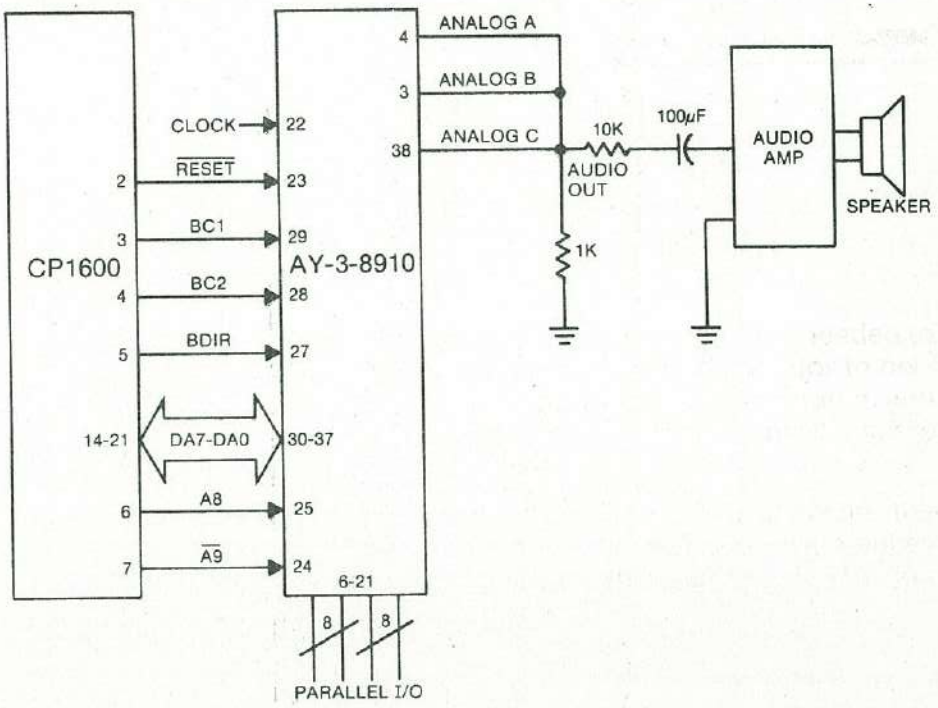
- ☐ Full software control of sound generation.
- ☐ Interfaces to most 8-bit and 16-bit microprocessors.
- ☐ Three independently programmed analog outputs.
- ☐ Two 8-bit general purpose I/O ports (AY-3-8910).
- ☐ One 8-bit general purpose I/O port (AY-3-8912).
- ☐ Single +5 Volt Supply.

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



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

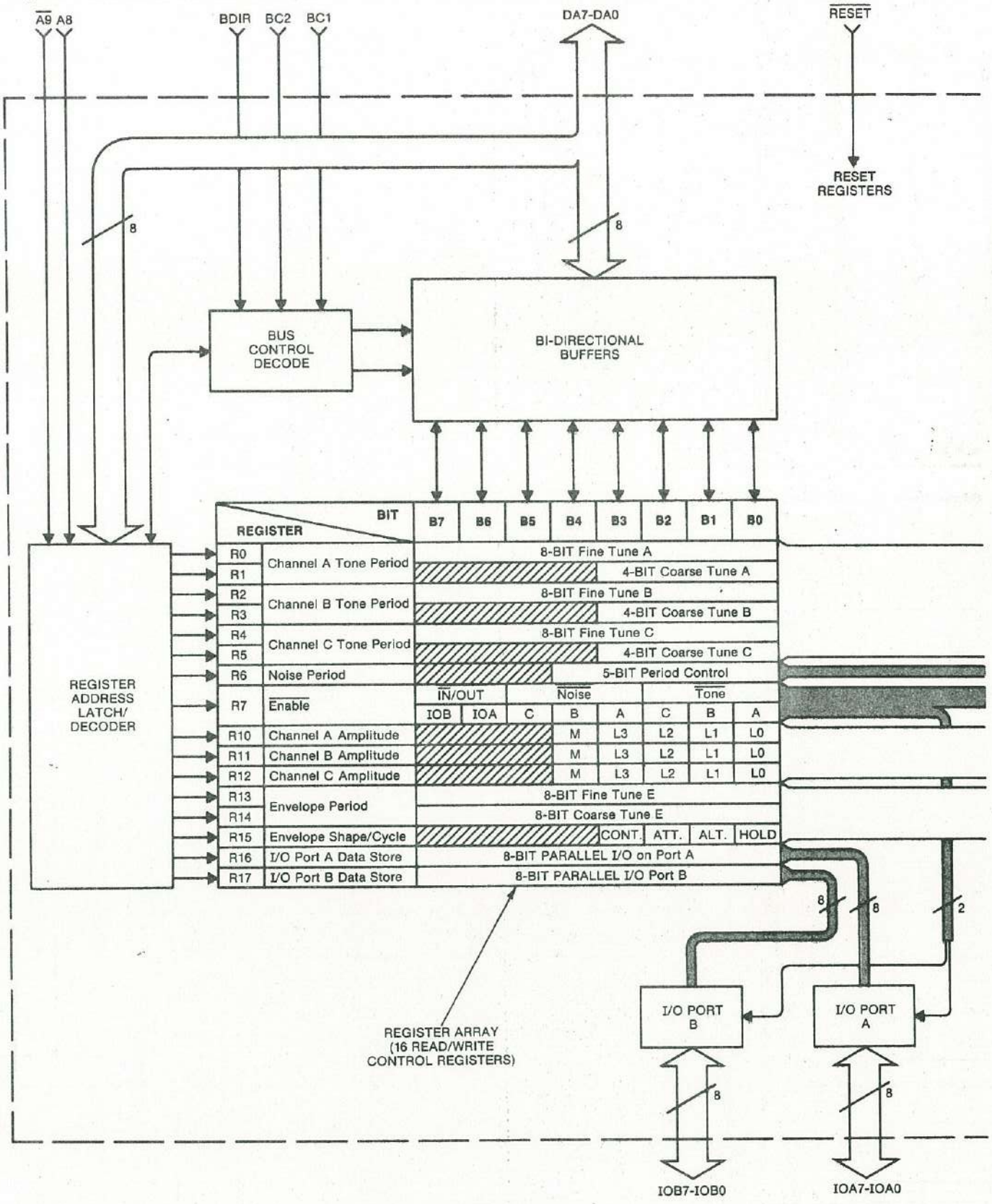
The principal element of the PSG is the array of 16 read/write control registers. These 16 registers look to the CPU as a block of memory and as such occupy a 16 word block out of 1,024 possible addresses. The 10 address bits (8 bits on the common data/address bus, and 2 separate address bits A8 and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

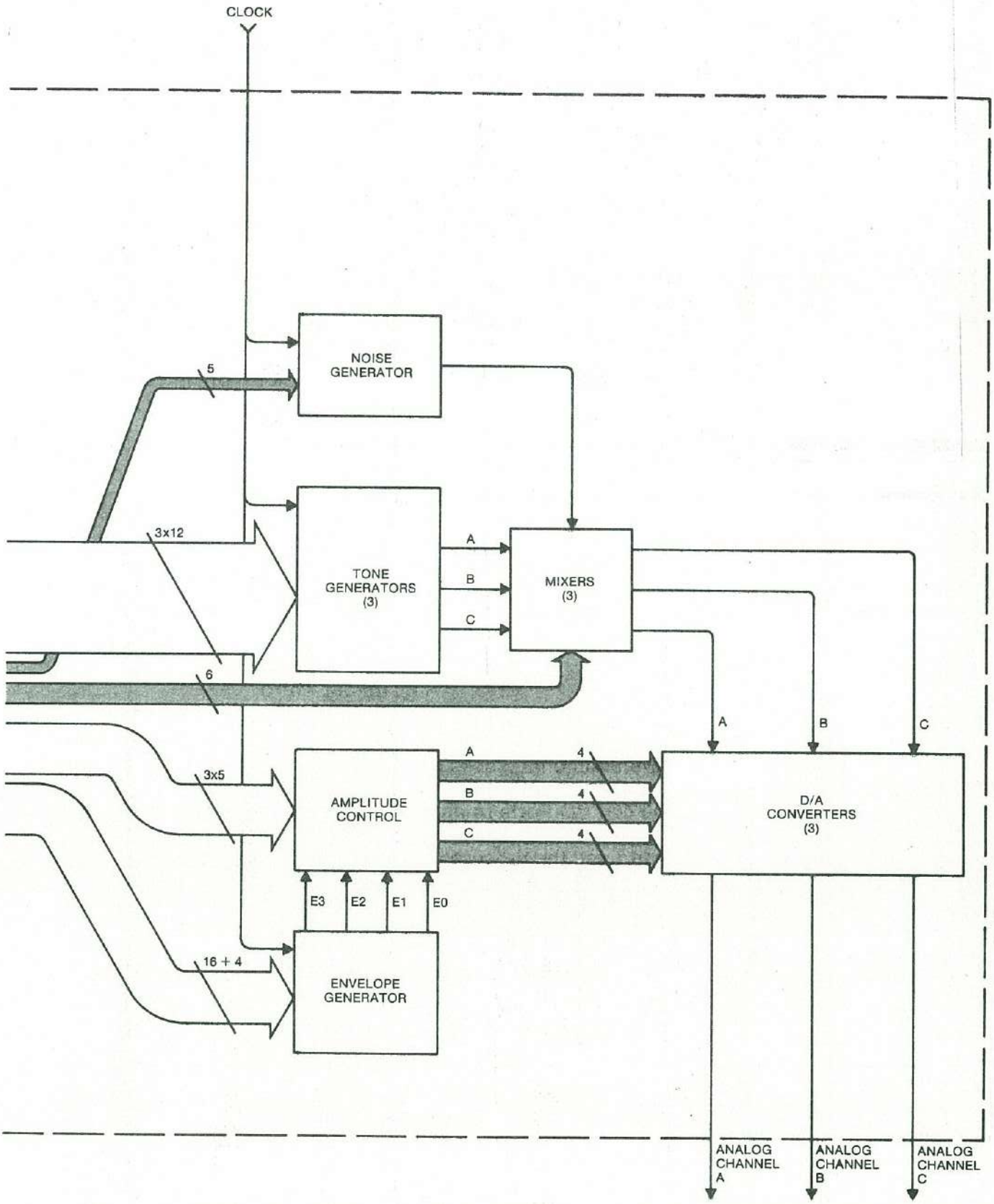
*$\overline{A9}$	A8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
0	1	0	0	0	0	0	0	0	0
THRU									
0	1	0	0	0	0	1	1	1	1
HIGH ORDER (Chip Select)						LOW ORDER (Register #)			

* $\overline{A9}$ 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.

Fig. 2 PSG BLOCK DIAGRAM




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

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.

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	$\overline{\text{IN}}$/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							

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

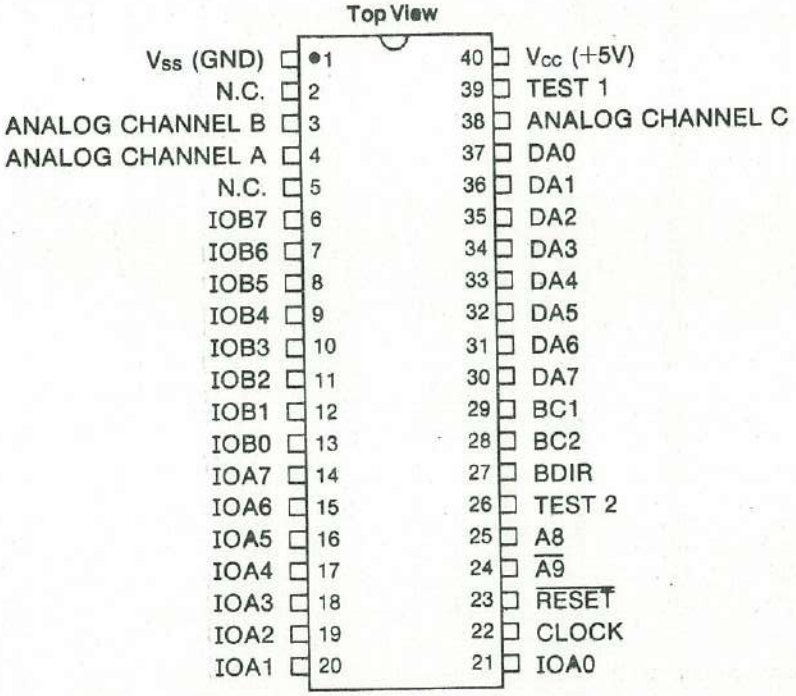
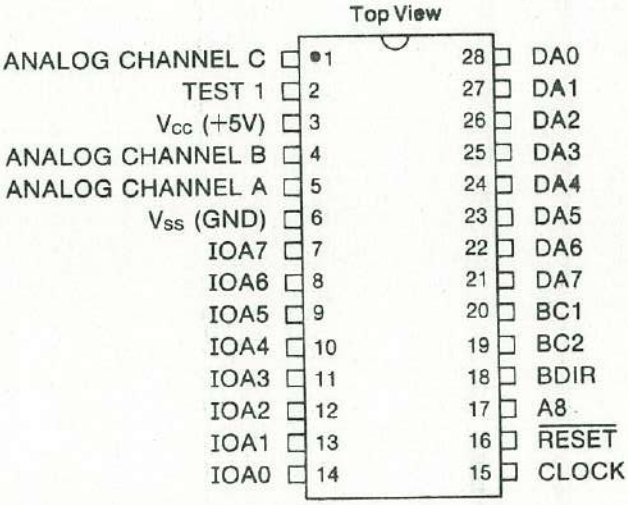


Fig. 5 AY-3-8912 PIN ASSIGNMENTS



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)

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

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.

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

For initialization/power-on purposes, applying a logic "0" (ground) to the $\overline{Reset}$ pin will reset all registers to "0". The $\overline{Reset}$ 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

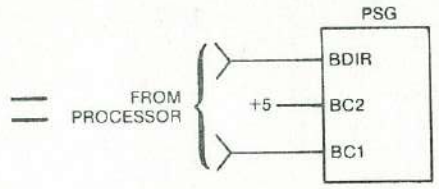
These bus control signals are generated directly by GI's CP1600 series of microprocessors to control all external and internal bus operations in the PSG. When using a processor other than the CP1600, these signals can be provided either by comparable bus signals or by simulating the signals on I/O lines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

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

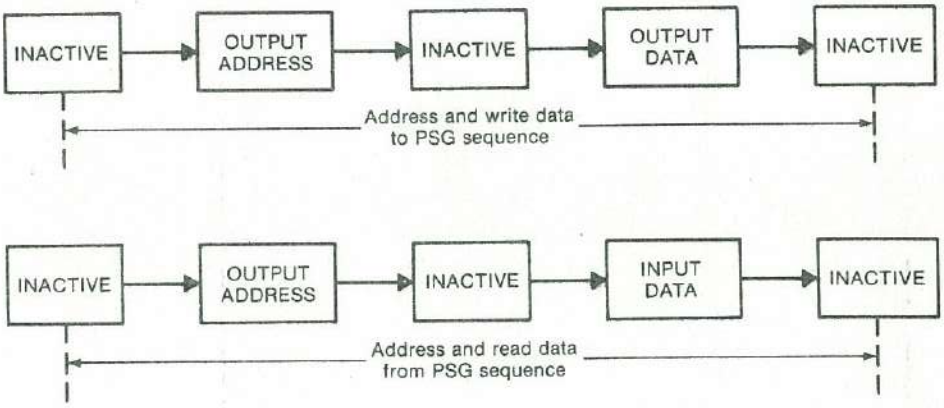
Since the PSG functions are controlled by commands from the system processor, the common data/address bus (DA7--DA0) requires definition as to its function at any particluar time. This is accomplished by the processor issuing bus control signals, previously described, defining the state of the bus; the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

2.5 State Timing

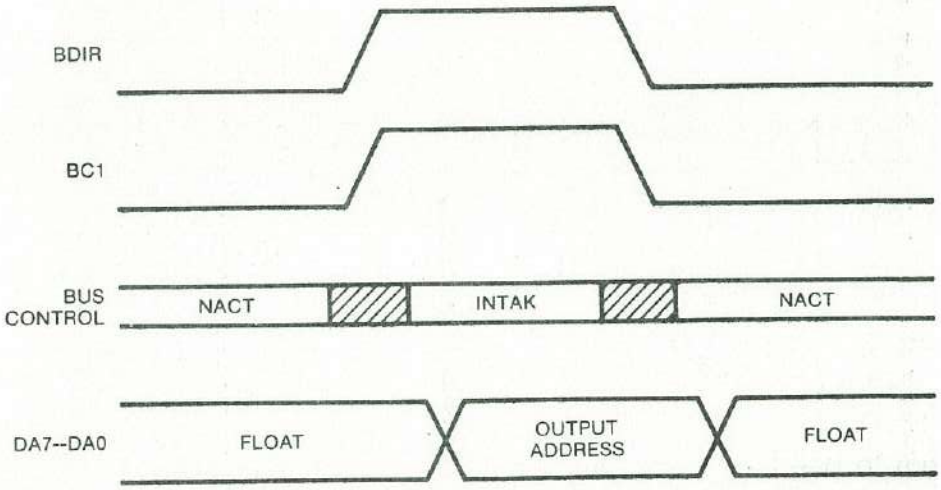
While the state flow for many microprocessors can be somewhat involved for certain operations, the sequence of events necessary to control the PSG is simple and straightforward. Each of the three major state sequences (Latch Address, Write to PSG, and Read from PSG) consists of several operations (indicated 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).

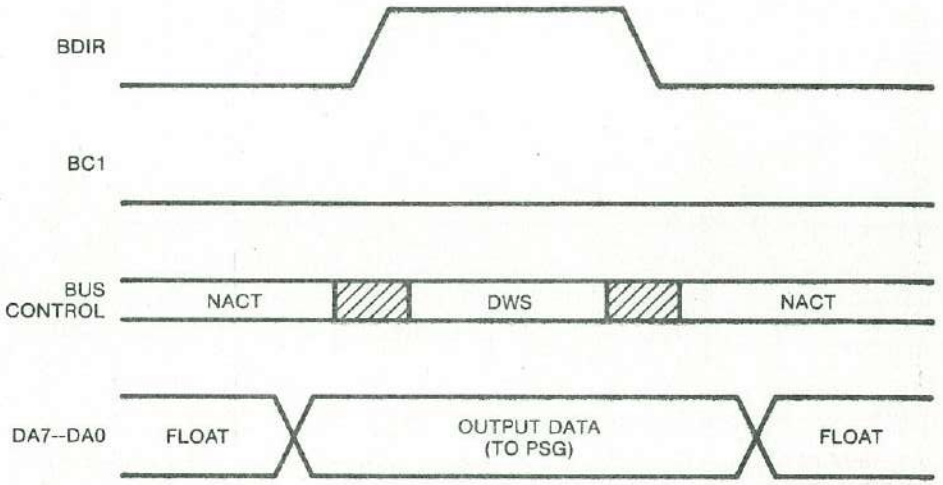
2.5.1 ADDRESS PSG REGISTER SEQUENCE

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.]



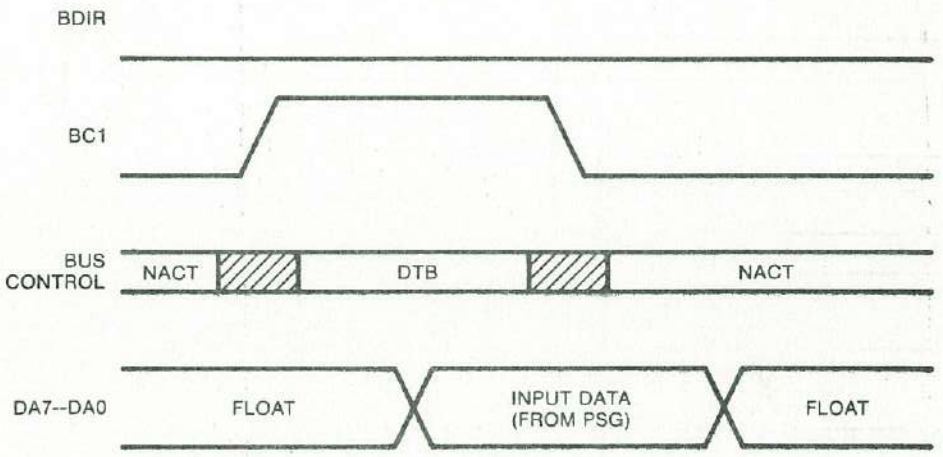
2.5.2 WRITE DATA TO PSG SEQUENCE

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.

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:

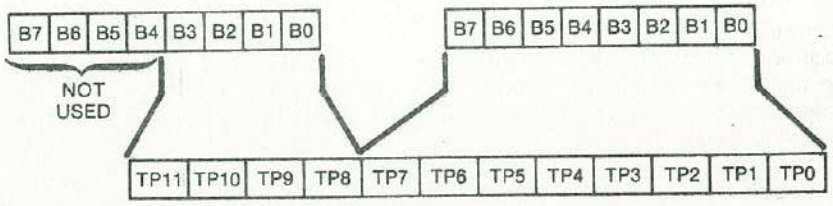
Section	Operation	Registers	Function
3.1	Tone Generator Control	R0--R5	Program tone periods.
3.2	Noise Generator Control	R6	Program noise period.
3.3	Mixer Control	R7	Enable tone and/or noise on selected channels.
3.4	Amplitude Control	R10--R12	Select "fixed" or "envelope-variable" amplitudes.
3.5	Envelope 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 $4{,}095_{10}$).

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

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

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:

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

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)

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:

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

3.3 Mixer Control-I/O Enable

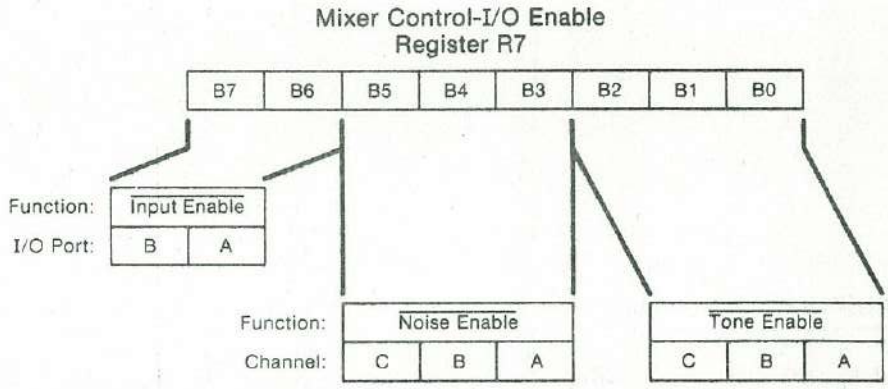
(Register R7)

Register 7 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:



Noise Enable Truth Table:

R7 Bits B5	B4	B3	Noise Enabled on Channel		
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		
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:

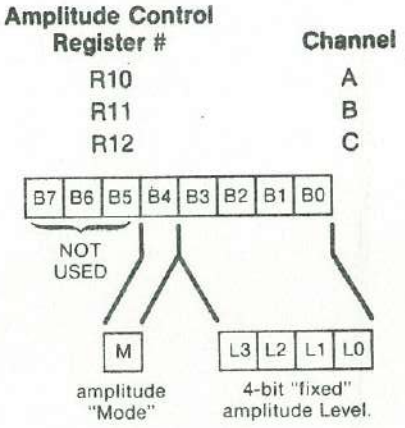
R7 Bits B7	B6	I/O Port Status IOB	IOA
0	0	Input	Input
0	1	Input	Output
1	0	Output	Input
1	1	Output	Output

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).

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).

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

Amplitude Control Register #	Channel
R10	A
R11	B
R12	C

B7	B6	B5	B4	B3	B2	B1	B0
NOT USED							

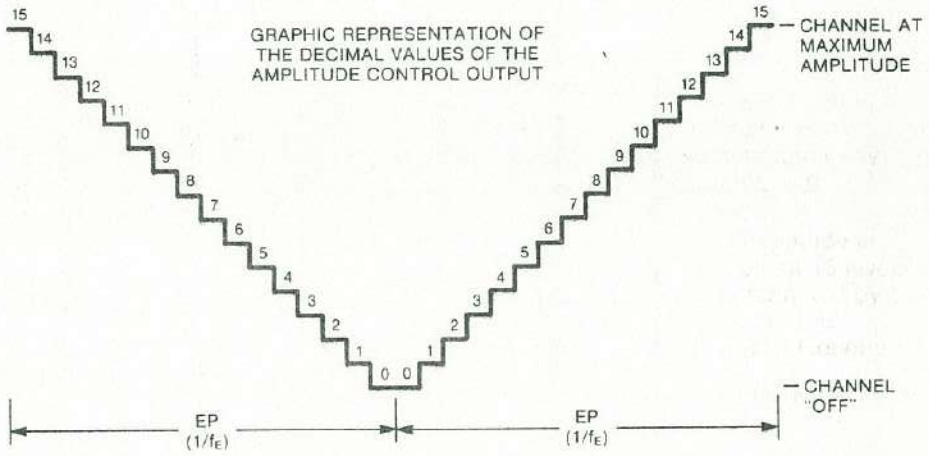
M	L3	L2	L1	L0	Amplitude Control Output	
0	0	0	0	0	* 0 0 0 0	The amplitude is fixed at 1 of 16 levels as determined by L3 L2 L1 L0.
.	.	.	.	.	. . . .	
.	.	.	.	.	. . . .	
.	.	.	.	.	. . . .	
0	1	1	1	1	1 1 1 1	
1	X	X	X	X	E3 E2 E1 E0	The amplitude is variable at 16 levels as determined by the output of the Envelope Generator.

(X=Don't Care)

*The all zeroes code is used to turn a channel "off".

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)



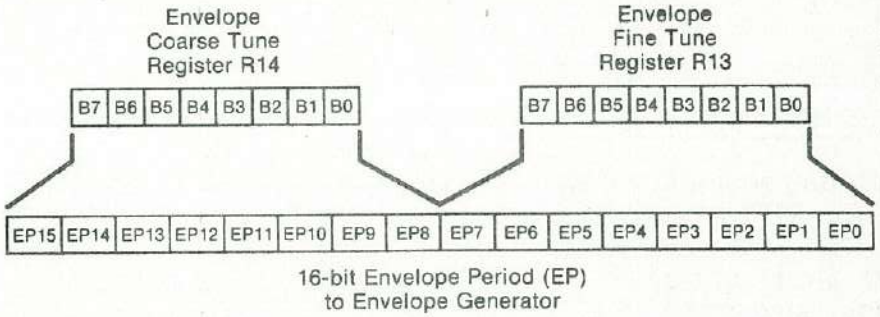
3.5 Envelope Generator Control

(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:



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:

(a) $f_E = \frac{f_{CLOCK}}{256EP_{10}}$ (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 $\frac{f_{CLOCK}}{16{,}776{,}960_{10}}$ (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):

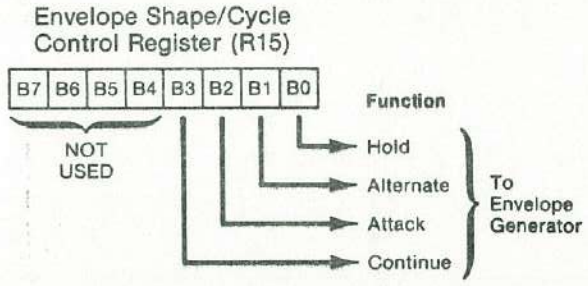
$$CT_{10} + \frac{FT_{10}}{256} = \frac{15{,}625}{256} = 61 + \frac{9}{256}$$

$CT_{10} = 61_{10} = 00111101$ (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.

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

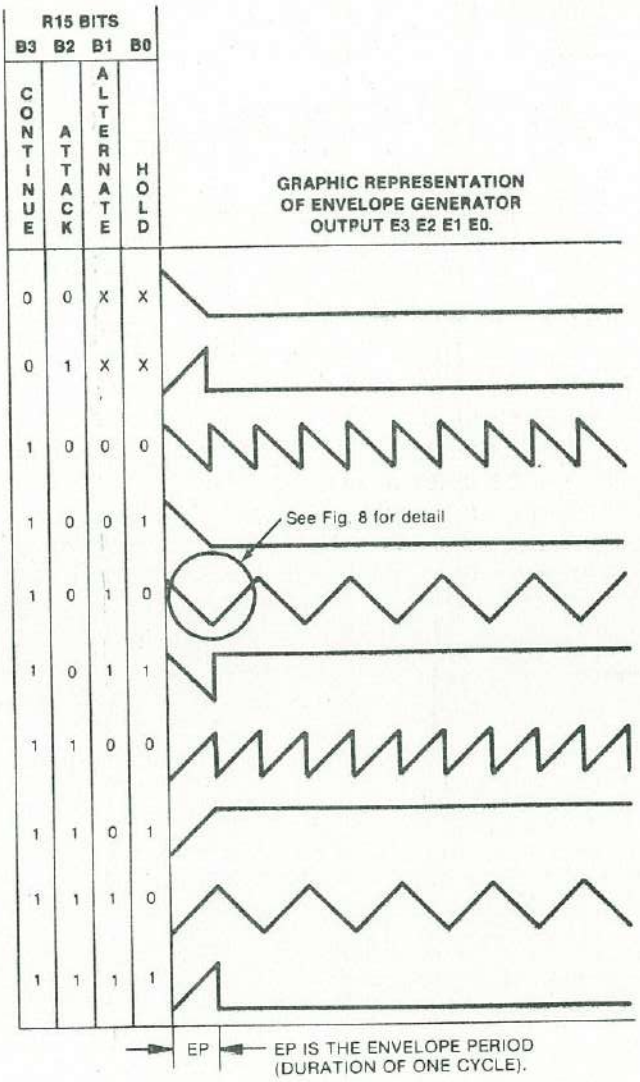
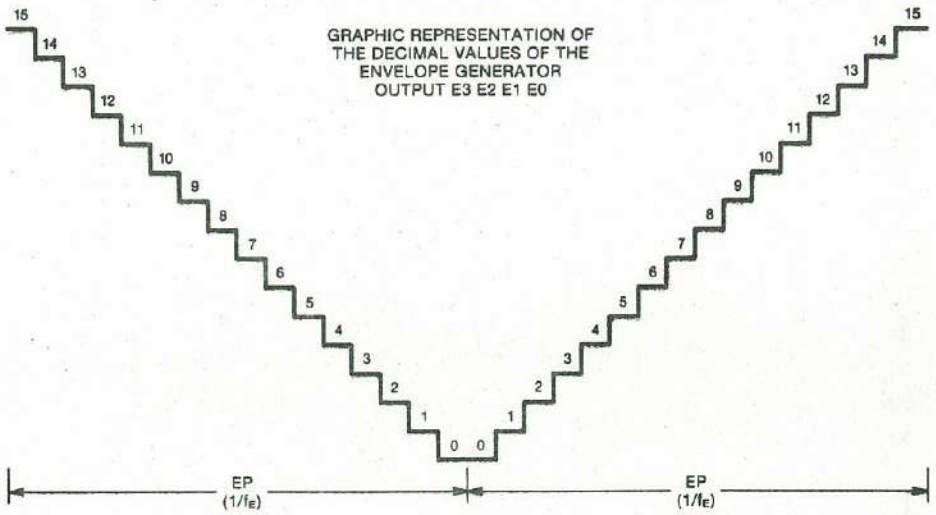


Fig. 8 DETAIL OF TWO CYCLES OF Fig. 7 (ref. waveform "1010" in Fig. 7)



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)

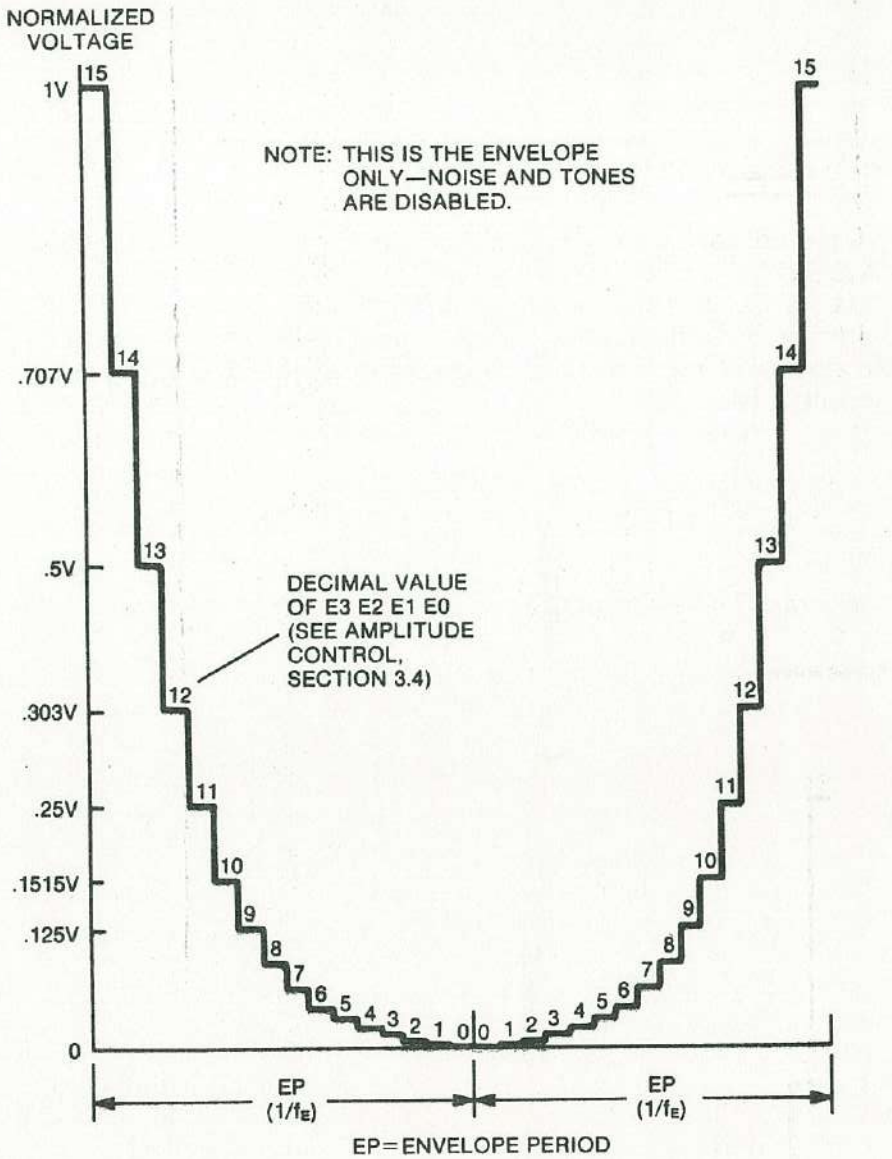


Fig. 10 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1000 (R0=14_8, R1=37_8, R7=76_8, R12=20_8, R15=10_8, all other registers=0)

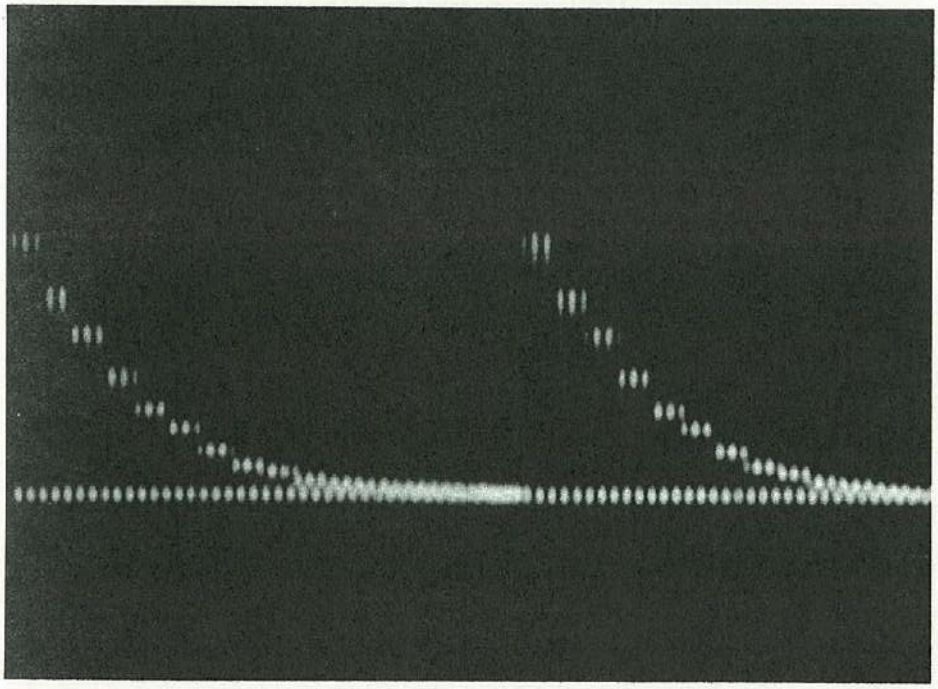


Fig. 11 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1100 (R15=14_8, all other registers same as Fig. 10)

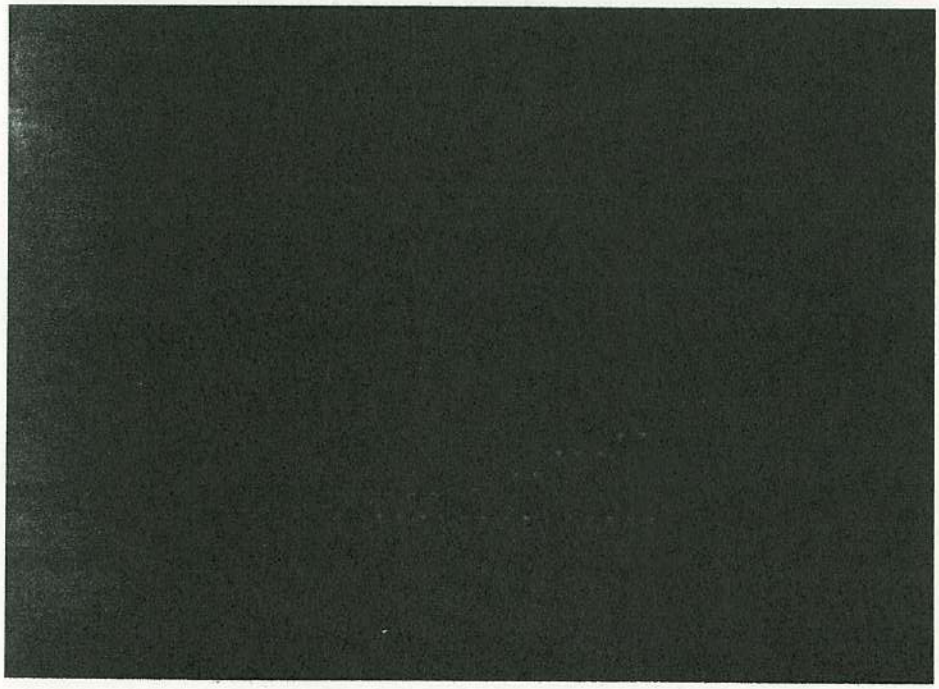


Fig. 12 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1010 (R15=12_8, all other registers same as Fig. 10)

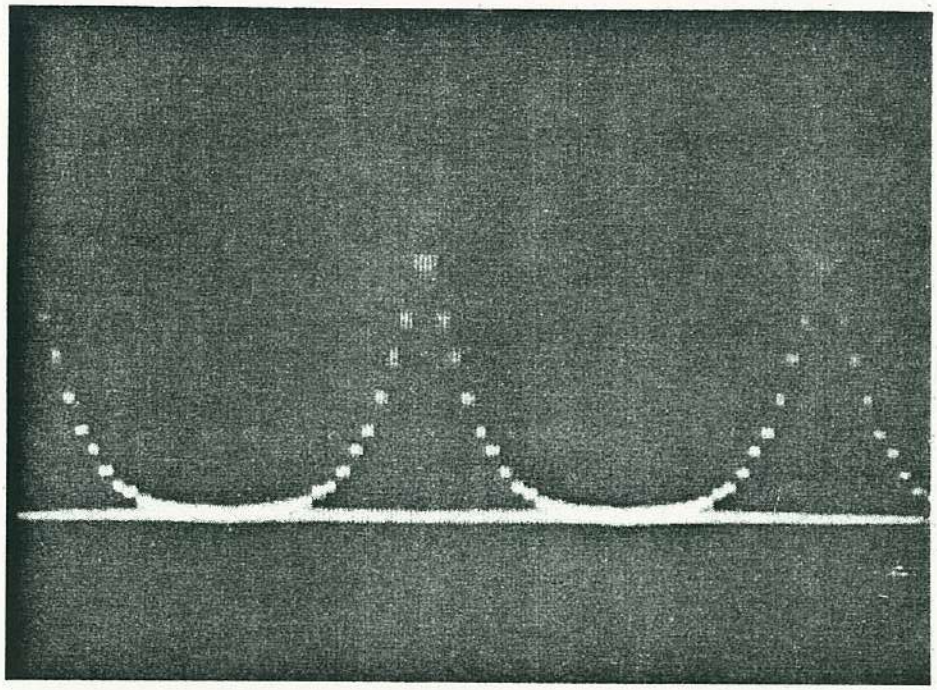
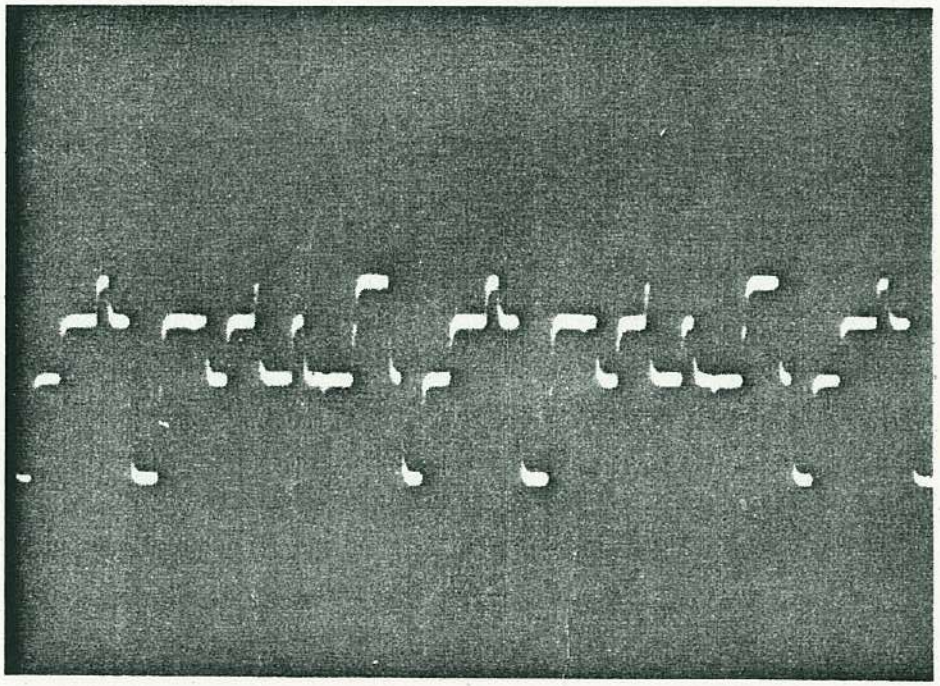


Fig. 13 MIXTURE OF THREE TONES WITH FIXED AMPLITUDES



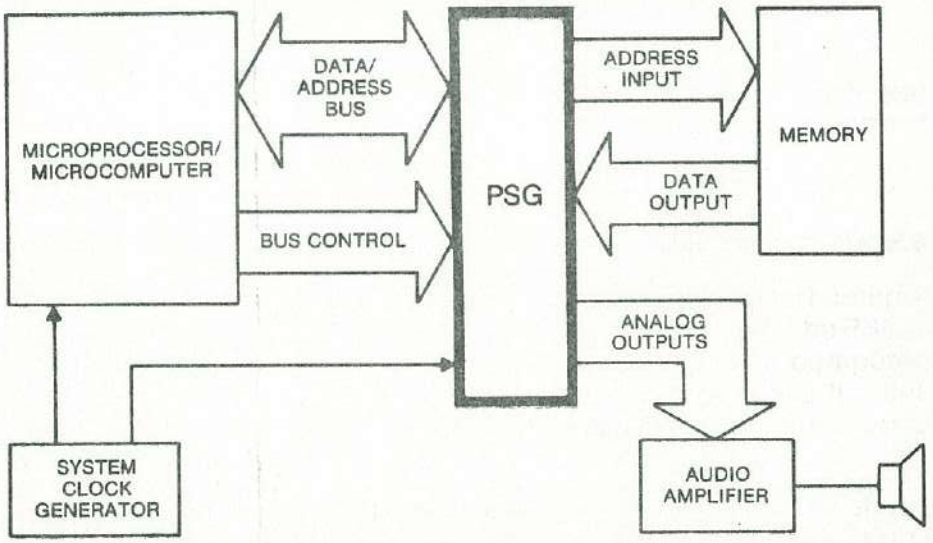
4 INTERFACING

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.

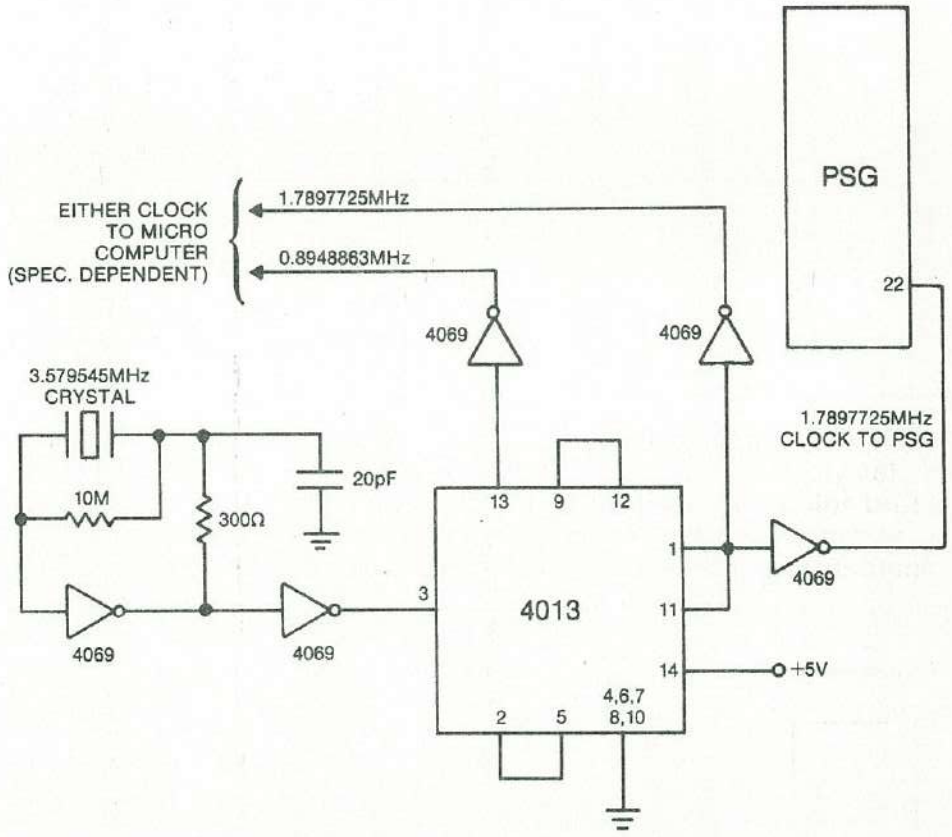
Fig. 14 SYSTEM BLOCK DIAGRAM



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

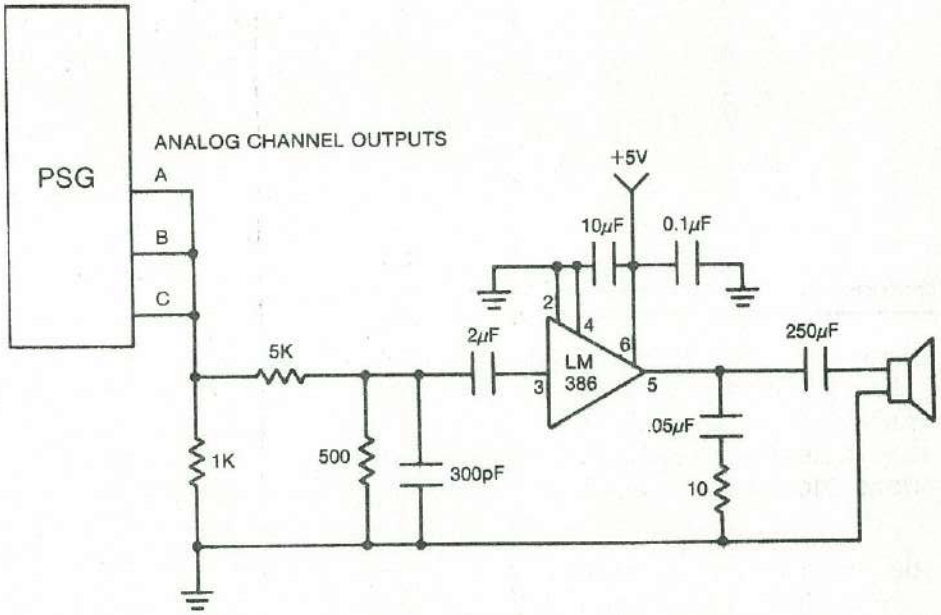


4.3 Audio Output Interface

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



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).

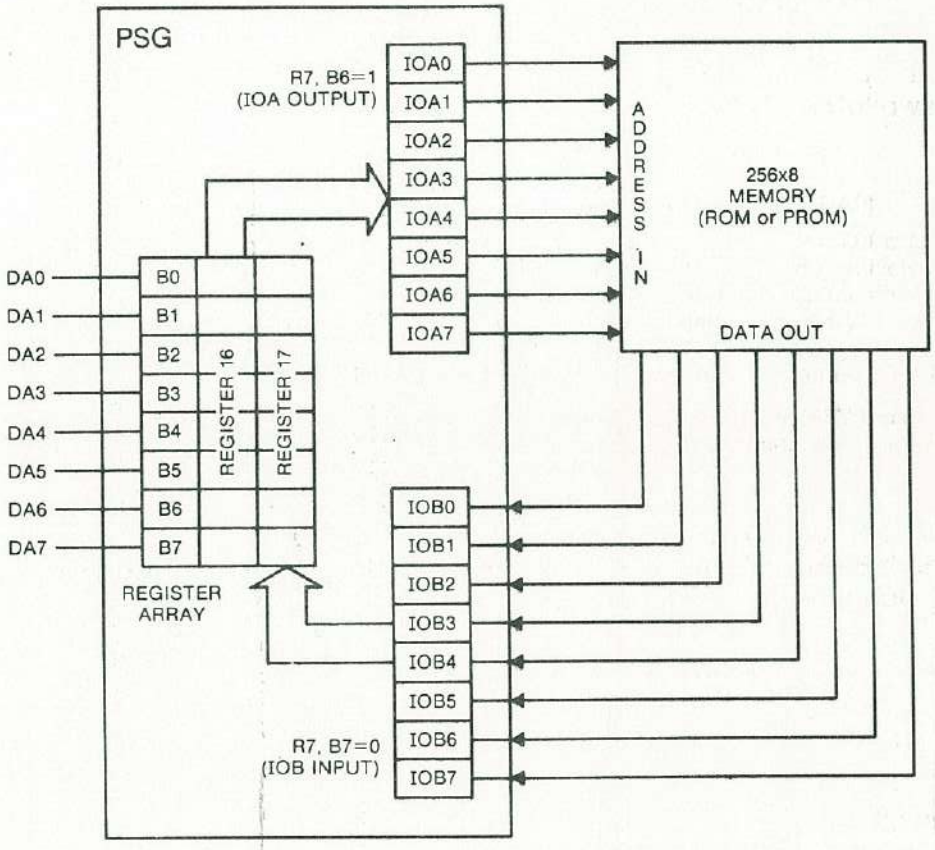
An example of the bus control sequence to address and read an external memory connected to I/O ports A and B would be as follows (Assume Port A addresses and Port B reads):

Bus Control	Bus Codes 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



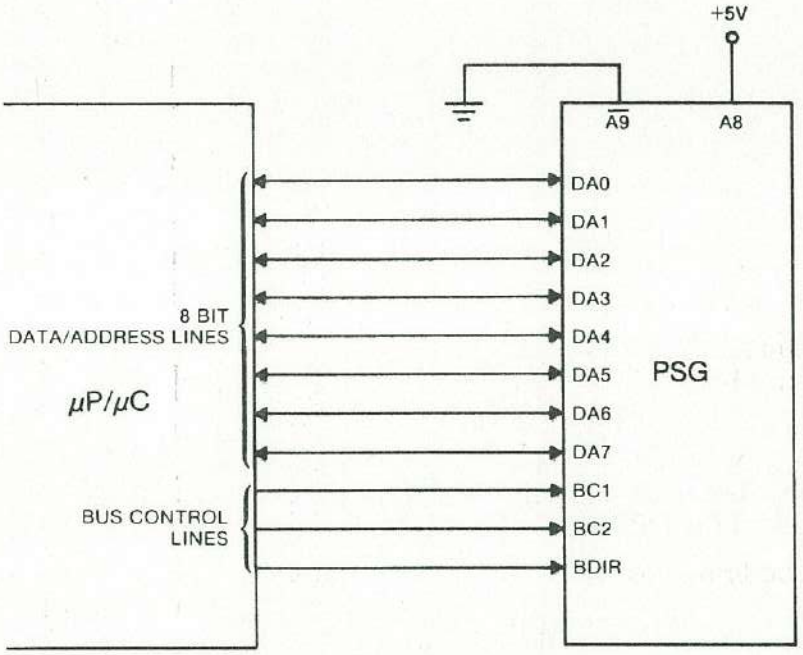
4.5 Microprocessor/ Microcomputer Interface

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



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
```

4.6 Interfacing to the PIC 1650 (cont.)

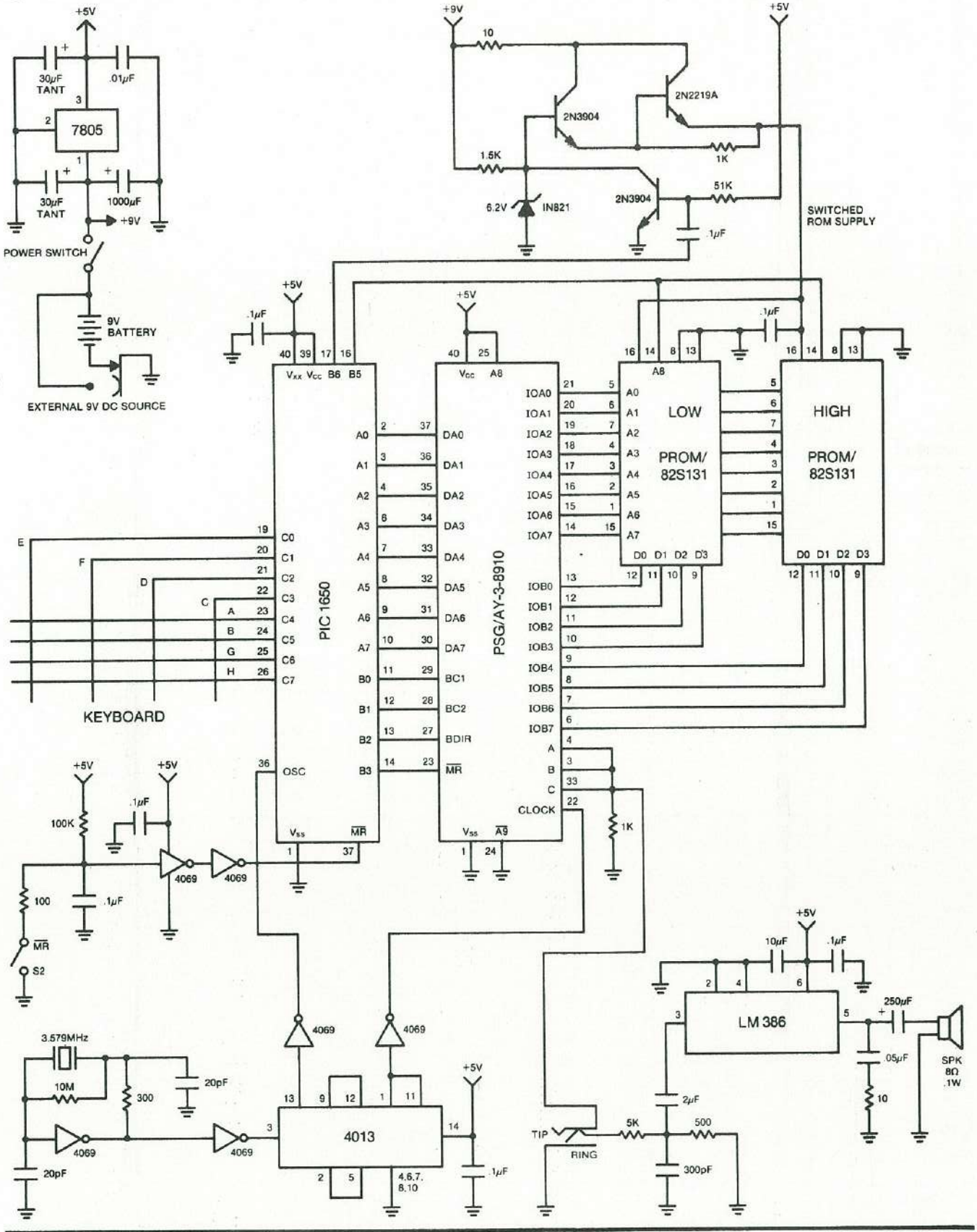
4.6.2 READ DATA ROUTINE

```
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=1
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



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:

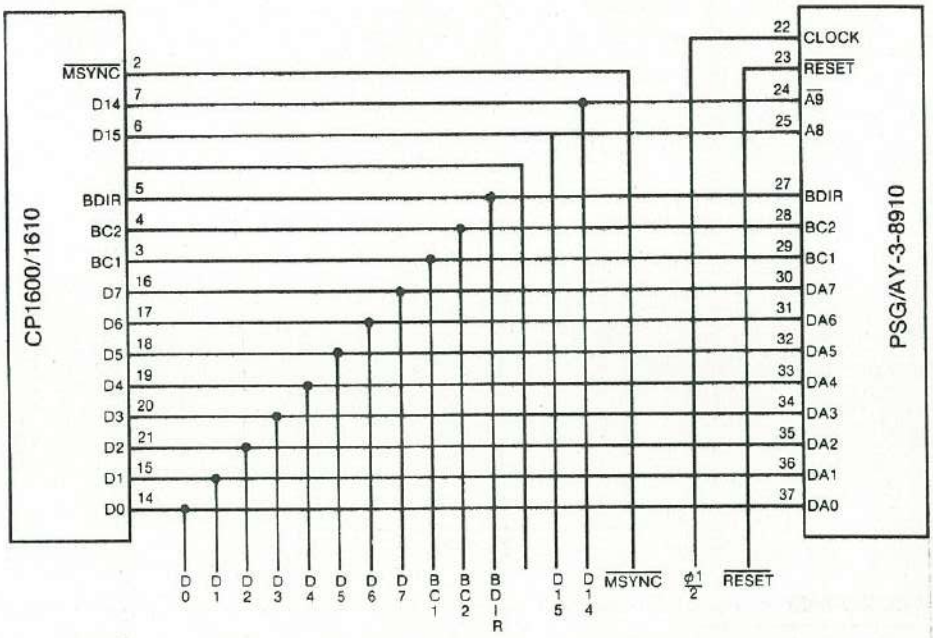
```
MVII Reg 0, R4
CLRR R0
Here MVO@ R0, R4
CMPI Reg 0 + 17, R4
BLT Here
```

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
```

Fig. 20 CP1600/1610/AY-3-8910 INTERFACE



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

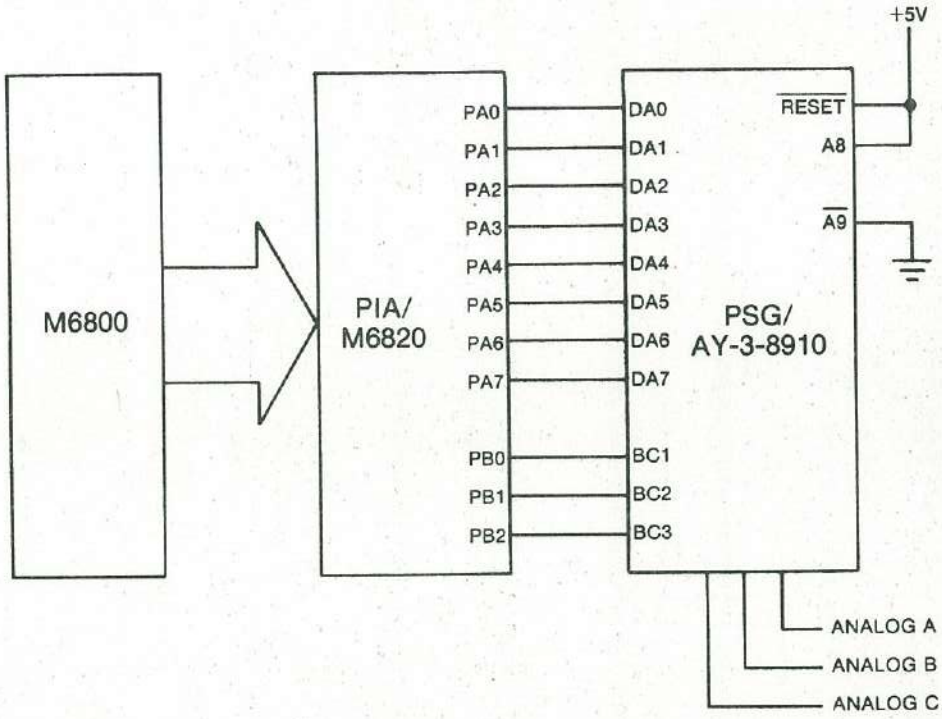
```
;AT ENTRY, B HAD DATA VALUE
;
WRITE STAB 8004 ;FORM DATA
   LDAA #6 ;DWS
   STAA 8006
   CLRA
   STAA 8006 ;WRITE DATA
   RTS ;RETURN

   !
```

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
```

Fig. 21 M6800/AY-3-8910 INTERFACE



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
```

4.9.2 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
```

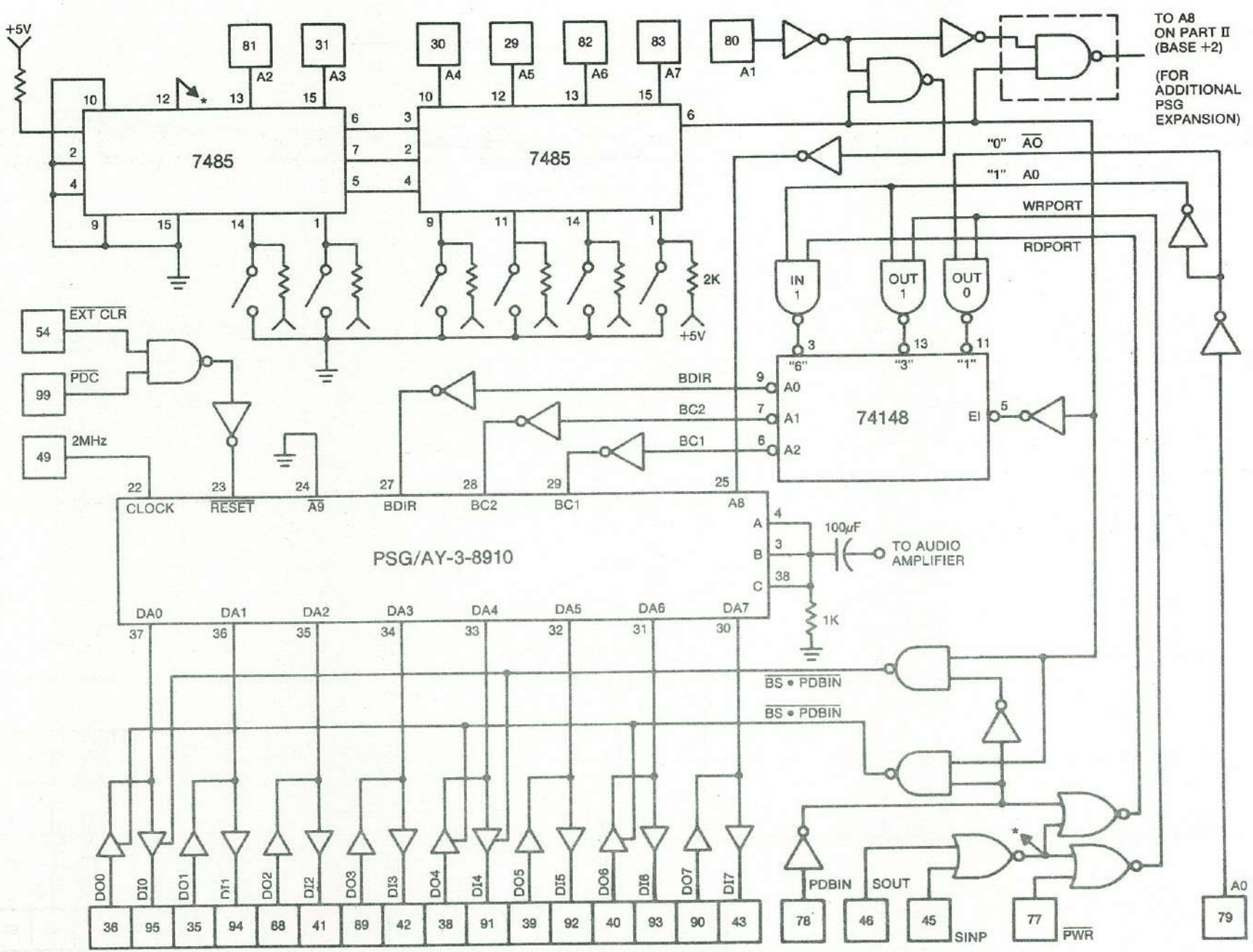


Fig. 22 8080 S100 BUS/AY-3-8910 INTERFACE

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
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_{CLOCK}=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

Chord Selection	Root	3rd Minor	3rd Major	4th	5th	6th	7th
C	C (÷2)	D# (÷2)	E (÷2)	F (÷2)	G (÷2)	A (÷2)	A# (÷2)
C#	C# (÷2)	E (÷2)	F (÷2)	F# (÷2)	G# (÷2)	A# (÷2)	B (÷2)
D	D (÷2)	F (÷2)	F# (÷2)	G (÷2)	A (÷2)	B (÷2)	C (÷1)
D#	D# (÷2)	F# (÷2)	G (÷2)	G# (÷2)	A# (÷2)	C (÷1)	C# (÷1)
E	E (÷2)	G (÷2)	G# (÷2)	A (÷2)	B (÷2)	C# (÷1)	D (÷1)
F	F (÷2)	G# (÷2)	A (÷2)	A# (÷2)	C (÷1)	D (÷1)	D# (÷1)
F#	F# (÷4)	A (÷4)	A# (÷4)	B (÷4)	C# (÷2)	D# (÷2)	E (÷2)
G	G (÷4)	A# (÷4)	B (÷4)	C (÷2)	D (÷2)	E (÷2)	F (÷2)
G#	G# (÷4)	B (÷4)	C (÷2)	C# (÷2)	D# (÷2)	F (÷2)	F# (÷2)
A	A (÷4)	C (÷2)	C# (÷2)	D (÷2)	E (÷2)	F# (÷2)	G (÷2)
A#	A# (÷4)	C# (÷2)	D (÷2)	D# (÷2)	F (÷2)	G (÷2)	G# (÷2)
B	B (÷4)	D (÷2)	D# (÷2)	E (÷2)	F# (÷2)	G# (÷2)	A (÷2)

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

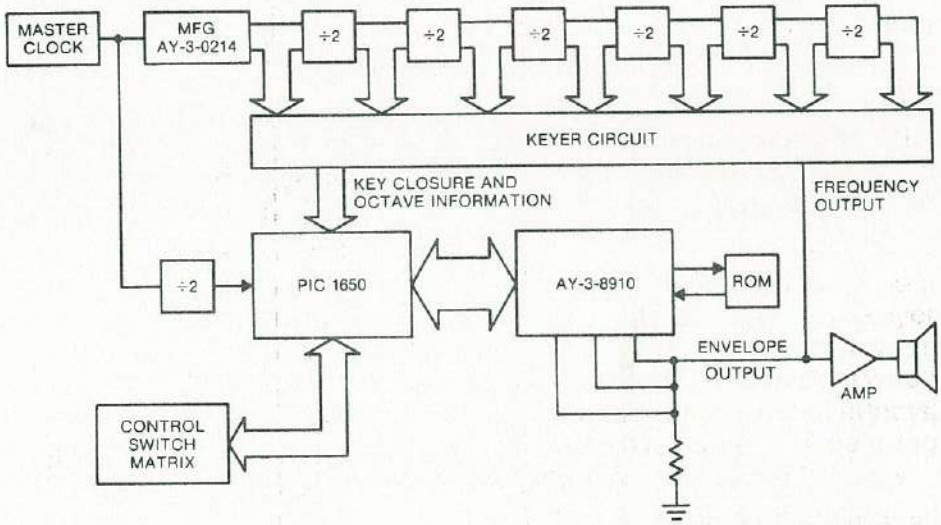
The envelope generation diagram shown in Fig. 25 illustrates how an AY-3-8910 can be configured to produce envelopes for organ voicing. All functions are controlled by a microcomputer.

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



5.5 Applications (cont.)

5.5.2 ORGAN RHYTHM GENERATION

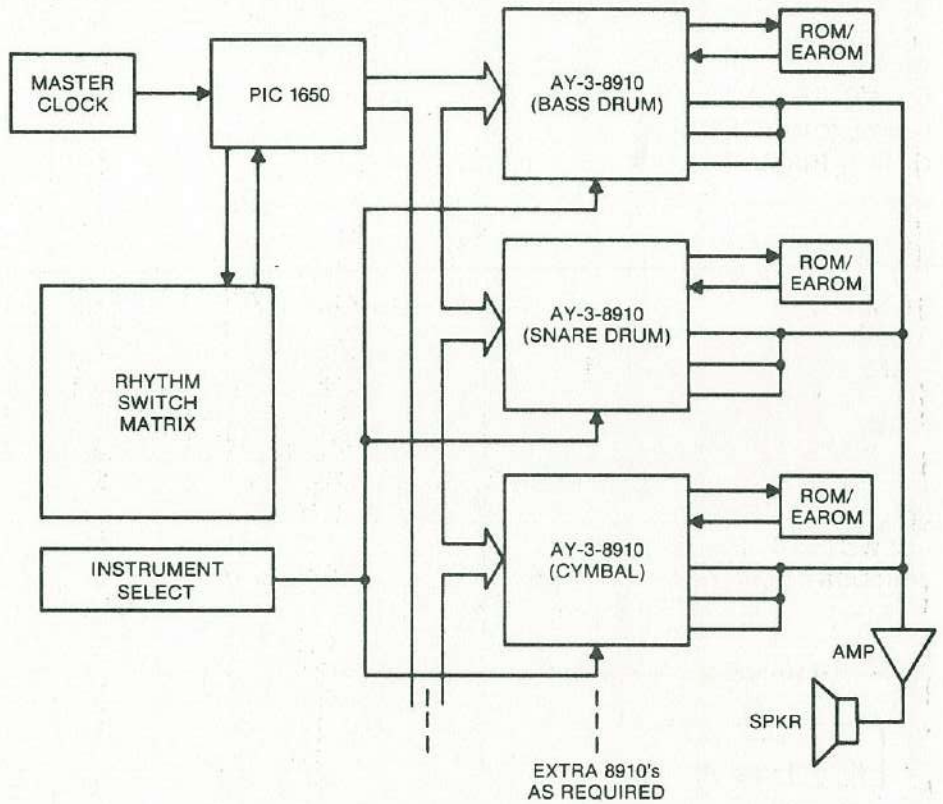
The rhythm generation diagram (Fig. 26) illustrates a simplified version of how a microcomputer can be implemented with the AY-3-8910 to provide a percussion instrument section for an electronic organ.

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



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 (440Hz).
R1	000	
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 (187Hz).
R1	001	
	(Wait approximately 350ms before continuing)	
R10	000	Turn off Channel A to end sound effect.

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.

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.

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.

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_0	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

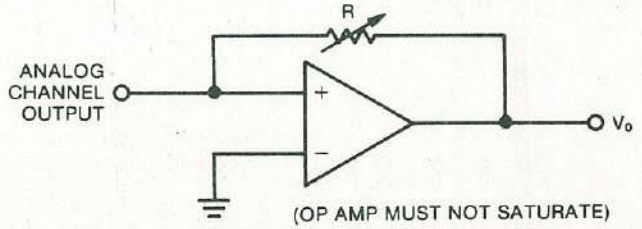
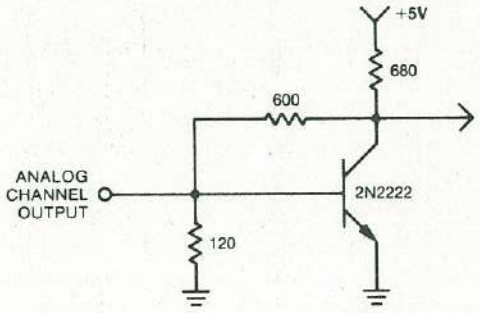


Fig. 35 CURRENT TO VOLTAGE CONVERTER



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	
Fall time	t_f	—	—	50	ns	
Duty Cycle	—	25	50	75	%	
Bus Signals (BDIR, BC2, BC1)						
Associative Delay Time	t_{BD}	—	—	50	ns	
$\overline{\text{Reset}}$						
$\overline{\text{Reset}}$ Pulse Width	t_{RW}	500	—	—	ns	Fig. 37
$\overline{\text{Reset}}$ to Bus Control Delay Time	t_{RB}	100	—	—	ns	
$\overline{\text{A9}}$, A8, DA7--DA0 (Address Mode)						
Address Setup Time	t_{AS}	400	—	—	ns	Fig. 38
Address Hold Time	t_{AH}	100	—	—	ns	
DA7--DA0 (Write Mode)						
Write Data Pulse Width	t_{DW}	500	—	10,000	ns	Fig. 39
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}	—	250	500	ns	Fig. 40
DA7--DA0 (Inactive Mode)						
Tristate Delay Time	t_{TS}	—	100	200	ns	

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

Fig. 36 CLOCK AND BUS SIGNAL TIMING

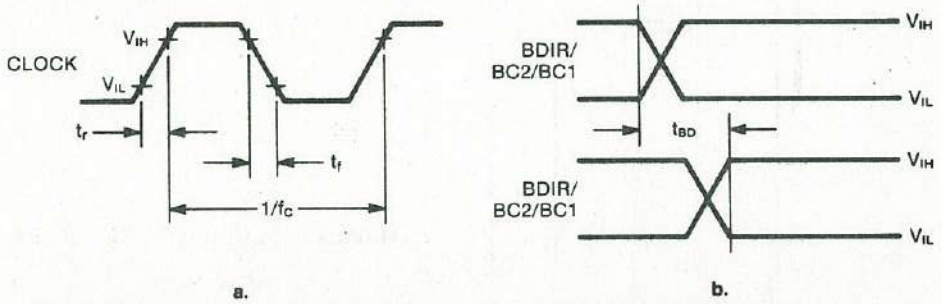


Fig. 37 RESET TIMING

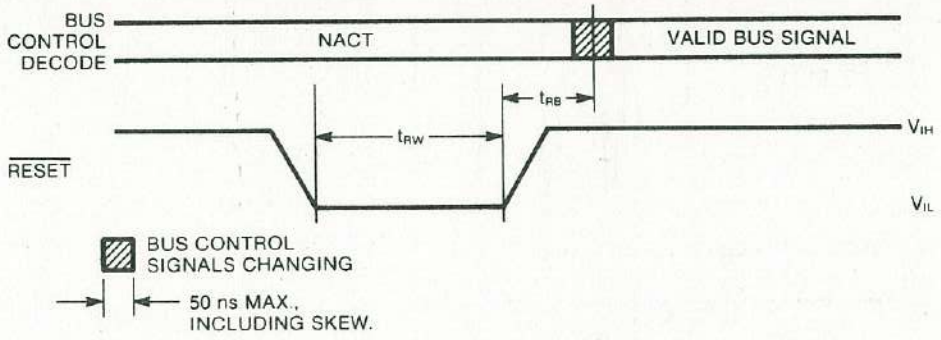


Fig. 38 LATCH ADDRESS TIMING

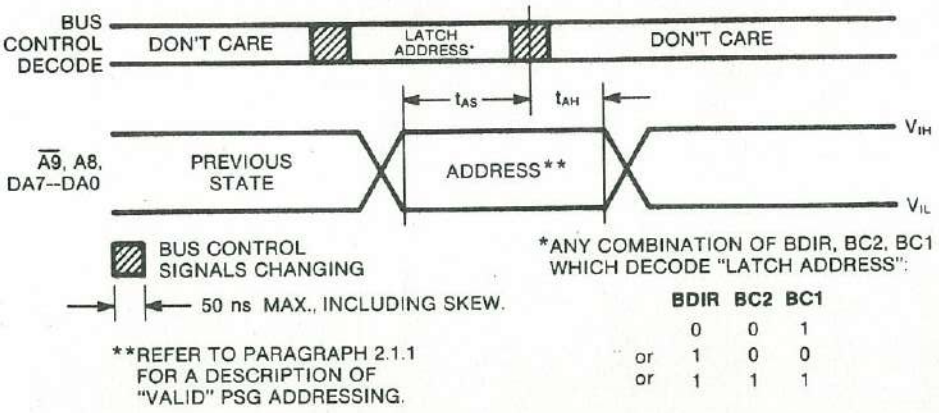


Fig. 39 WRITE DATA TIMING

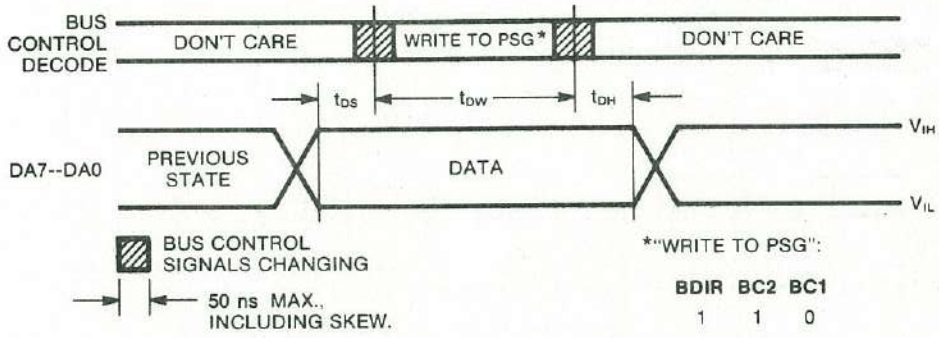
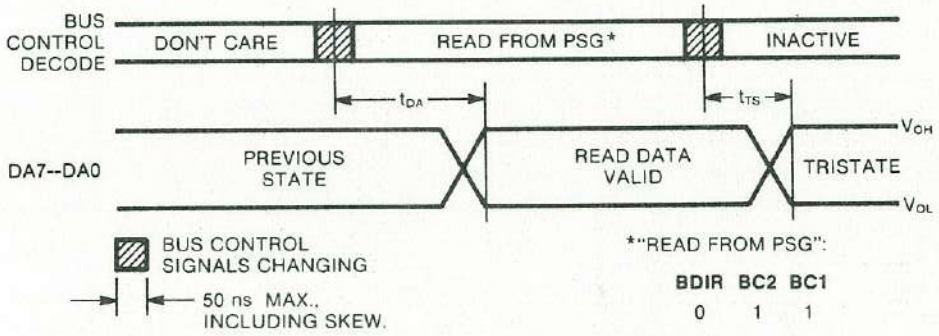
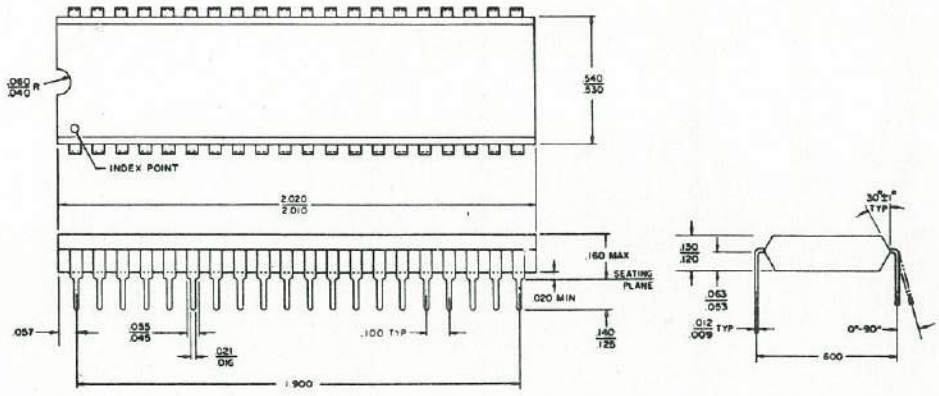


Fig. 40 READ DATA TIMING

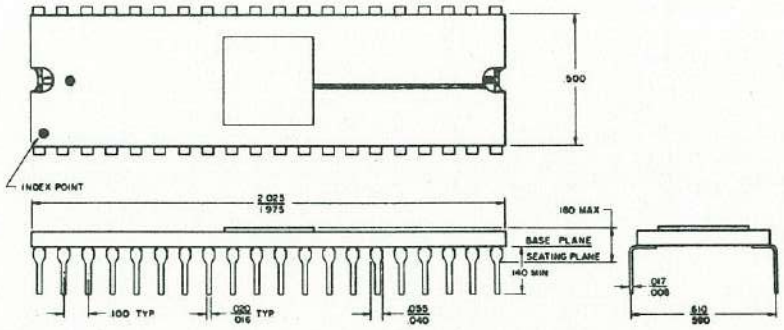


7.5
Package Outlines

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

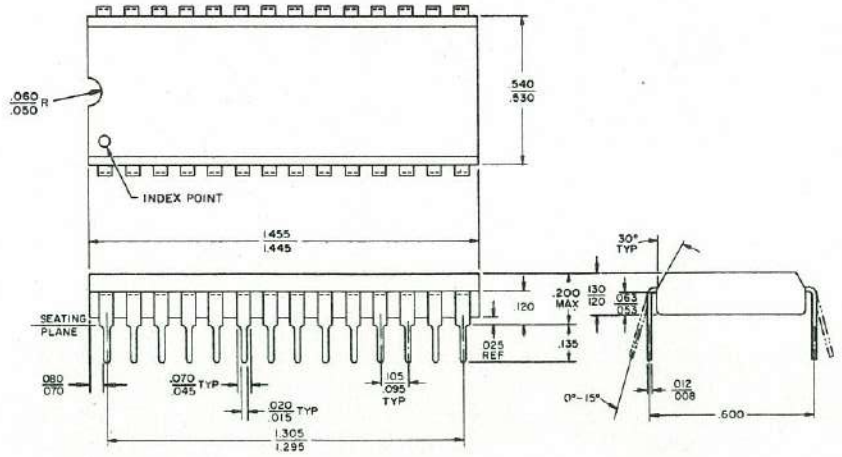


PLASTIC

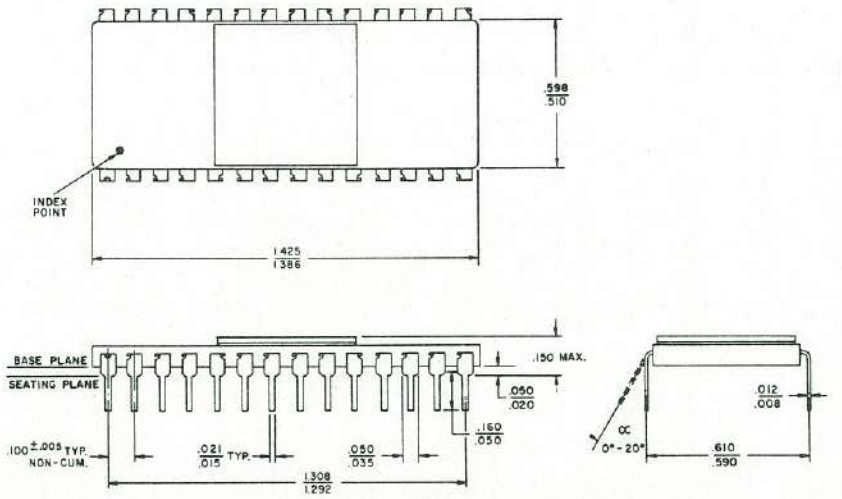


CERAMIC

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



PLASTIC

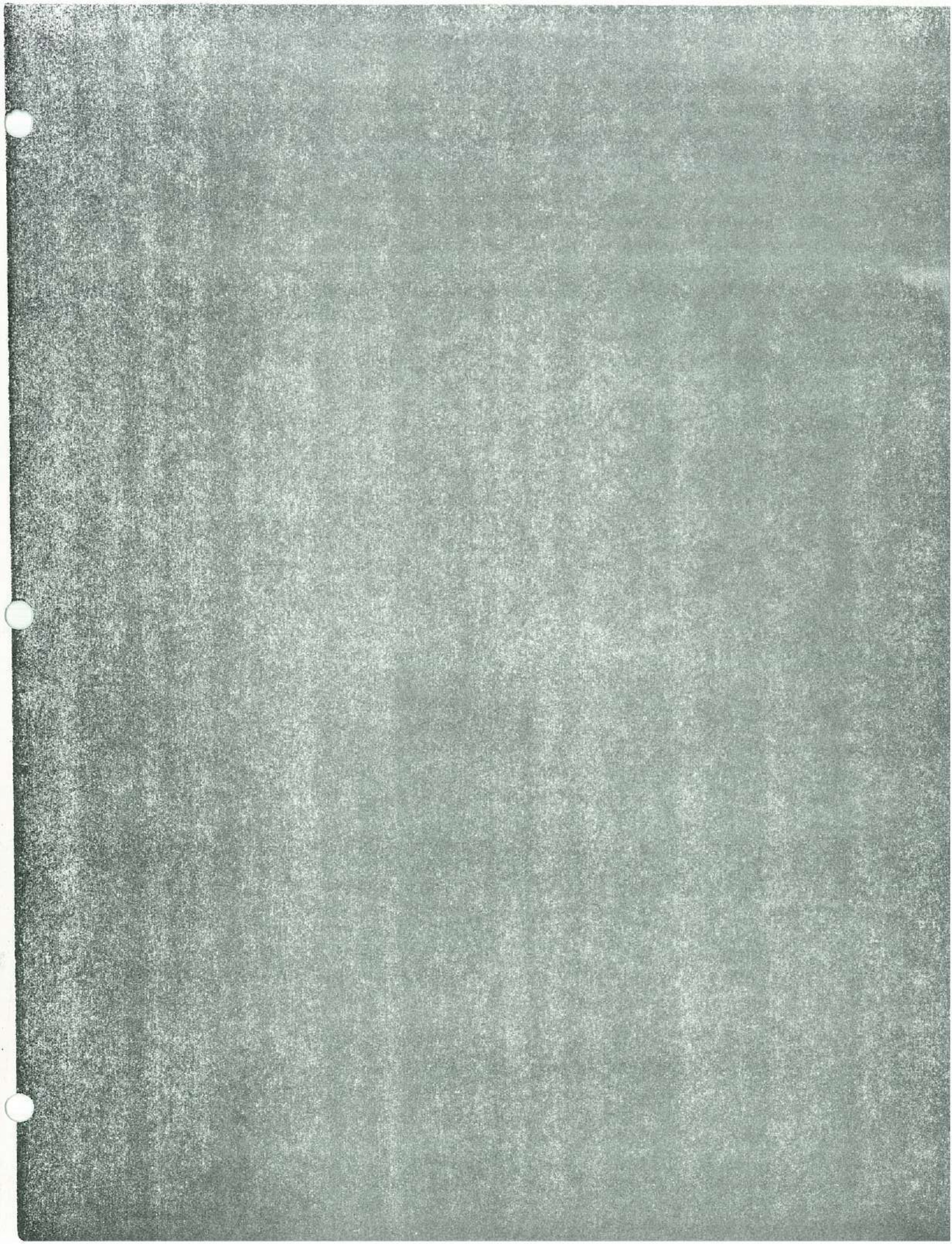


CERAMIC

NOTES

NOTES

NOTES



Appendix F

TMS 9918A Data Sheet

The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TMS 9918A VIDEO DISPLAY PROCESSOR DATA MANUAL

NOVEMBER 1980

TEXAS INSTRUMENTS
INCORPORATED

1. INTRODUCTION

1.1 DESCRIPTION

The TMS 9918A Video Display Processor (VDP) is an N-channel MOS LSI device used in video systems where data display on a raster-scanned home color television set or color monitor is desired. The TMS 9918A generates all necessary video, control, and synchronization signals and also controls the storage, retrieval, and refresh of display data in the dynamic screen refresh memory. The interfaces to the microprocessor, refresh memory, and the TV require a minimum of additional electronics.

The VDP has four video display modes: Graphics I, Graphics II, Multicolor and Text mode. The Text mode provides twenty-four 40-character rows in two colors and is intended to maximize the capacity of the TV screen to display alphanumeric characters. The Multicolor mode provides an unrestricted 64 X 48 color-dot display utilizing 15 colors plus transparent. The Graphics I mode provides a 256 X 192 pixel display for generating pattern graphics in 15 colors plus transparent. The Graphics II mode is an enhancement of Graphics I mode, providing the capability to generate more complex color and pattern displays.

The video display consists of 35 planes, external video, backdrop, pattern plane, and 32 Sprite Planes. The planes are vertically stacked with the external video being the bottom or innermost plane. The backdrop plane is the next plane followed by the pattern plane that contains Graphics I and Graphics II patterns with the 32 Sprite Planes as the top planes. (A sprite is an object-oriented animation pattern that can be moved smoothly across the screen.)

The TMS 9918A VDP utilizes either a 4K, 8K, or 16K-type low-cost dynamic memory (TMS 4027, TMS 4108, TMS 4116) for storage of the display parameters.

1.2 FEATURES

- Single-chip interface to color TV's (excluding RAM and RF modulator).
- 256 X 192 resolution on TV screen
- 15 unique colors plus transparent
- General 8-bit bidirectional interface to CPU
- Direct wiring to 4K, 8K, or 16K dynamic RAM memories
- Automatic and transparent refresh of dynamic RAMs
- External video input capability
- NTSC - standard composite video output
- Unique planar representation for 3D simulation
- Standard 40-pin package

1.3 TYPICAL APPLICATIONS

- Color computer terminals
- Home computers
- Drafting/design aids
- Teaching aids
- Industrial process monitoring
- Home educational systems
- Animation aids

The following example of a typical application may help introduce the user to the TMS 9918A VDP. Figure 1-1 is a block diagram of a typical application. Each of the concepts presented below is described more fully in later sections of this manual.

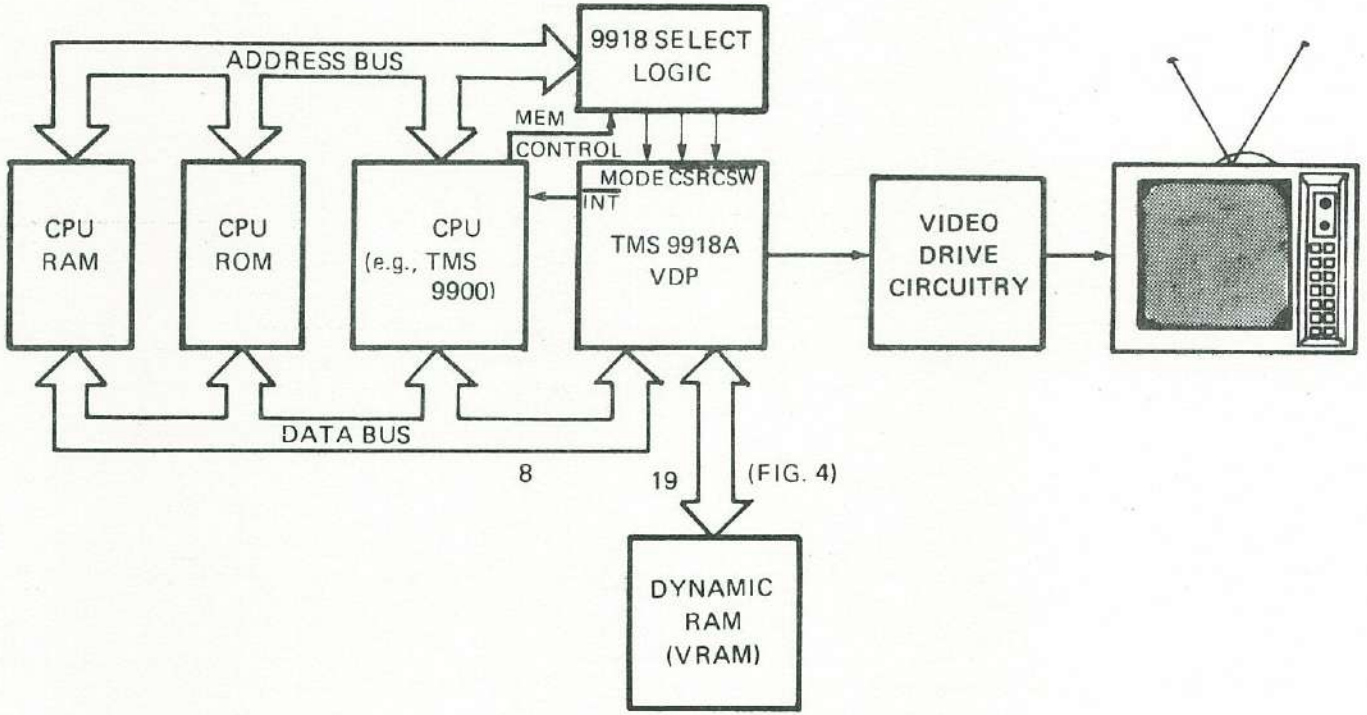


FIGURE 1-1 – SYSTEM BLOCK DIAGRAM

The VDP basically has three interfaces: CPU, color television, and dynamic refresh RAM (VRAM) the contents of which define the TV image. The TMS 9918A VDP also has eight write-only registers and a read-only status register.

The VDP communicates with the CPU via an 8-bit bidirectional data bus. Three control lines, decoded from the CPU address and enable lines, determine interpretation of the bus. Through the bus, the CPU can write to VRAM, read from VRAM, write to VDP registers, and read the VDP status. The VDP also generates an interrupt signal after every refresh of the TV display, which may be useful to the CPU.

The dynamic RAM interface consists of direct wiring of eight 4K X 1, 8K X 1, or 16K X 1 dynamic RAS/CAS-type RAMs to the TMS 9918 VDP. The amount of RAM required is dependent upon the features selected for use in the application.

The interface to the TV can consist of wiring the VDP's composite video output pin (suitably buffered) to the input of a color or black-and-white monitor, or an appropriate RF modulator may be used to feed the signal into a TV antenna terminal.

The VDP can operate in any one of four modes, each of which can affect the way the VRAM is mapped onto the television screen. In Graphics I and II modes, characters are mapped onto the screen in 8 X 8 pixel blocks, yielding 24 lines of 32 blocks (or "pattern positions") each. In Text mode, there are 24 lines of 40 blocks, each of which is 6 X 8 pixels. In Multicolor mode, there are 48 lines of 64 blocks, each of which is composed of 4 X 4 pixels, all of one solid color. In addition to these, objects termed "sprites" can be superimposed onto the television image. Furthermore, signals entering the VDP through the external video input can be used as a background to VDP-generated images.

1.4 DEFINITIONS

The following definitions will be useful in understanding the use of the TMS 9918 VDP:

pixel — the smallest point on the TV screen that can be independently controlled

NTSC — National Television Standards Committee which specifies the television signal standard for the USA

VRAM — Video RAM; refers to the dynamic RAMs that connect to the VDP and whose contents define the TV image

sprite — An object whose pattern is relative to a specified X, Y coordinate and whose position can therefore be controlled by that coordinate with a positional resolution of one pixel

2. ARCHITECTURE

The TMS 9918A Video Display Processor (VDP) is designed to provide a simple interface between a microprocessor and a raster-scanned color television. Shown in Figure 2-1 is a block diagram of the major portions of the VDP architecture. Described below are details of the various interfaces to the VDP, CPU, VRAM, and color television.

2.1 CPU INTERFACE

The VDP interfaces to the CPU using an 8-bit bidirectional data bus, three control lines, and an interrupt as shown in Figure 2-2. Through this interface the CPU can conduct four operations: write data bytes to VRAM, read data bytes from VRAM, write to one of the eight VDP write-only registers, and read the VDP Status Register. Each of these operations requires one or more data transfers to take place over the CPU/VDP data bus interface. The interpretation of the data transfer is determined by the three control lines of the VDP. It should be noted that the CPU can communicate with the VDP simultaneously and asynchronously with the VDP's TV screen refresh operations. The VDP performs memory management and allows periodic intervals of CPU access to VRAM even in the middle of a raster scan.

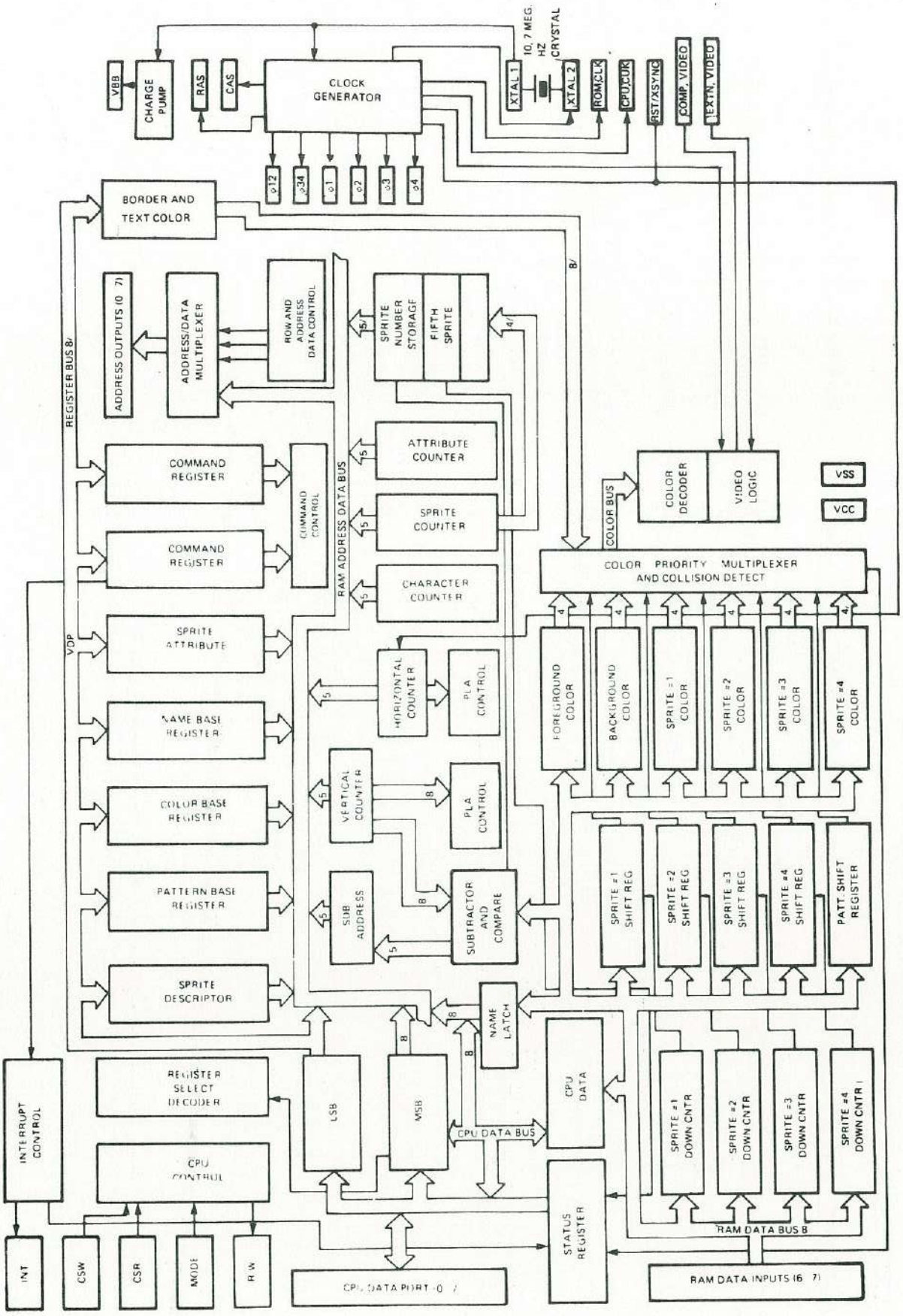


FIGURE 2-1 — TMS 9918 VDP BLOCK DIAGRAM

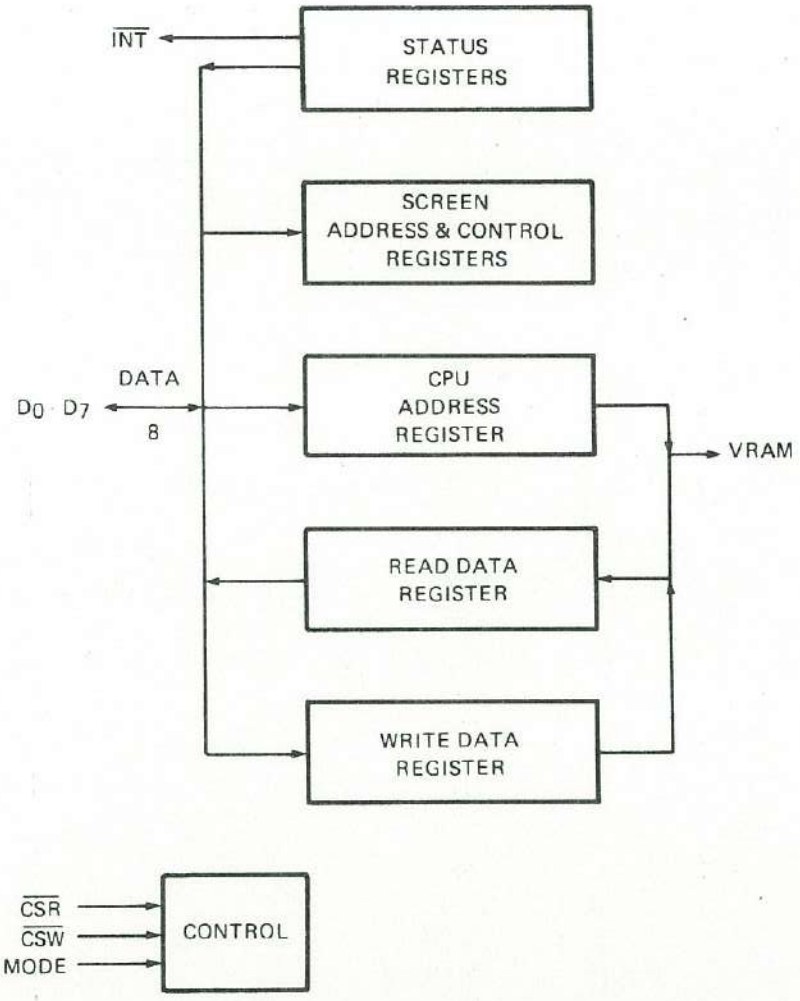


FIGURE 2-2 – VDP TO CPU INTERFACE

2.1.1 CPU Interface Control Signals

The type and direction of data transfers are controlled by the $\overline{CSW}$, $\overline{CSR}$, and MODE inputs. $\overline{CSW}$ is the CPU-to-VDP write select. When it is active (low), the 8 bits on D0-D7 are strobed into the VDP. $\overline{CSR}$ is the CPU-from-VDP read select. When it is active (low), the VDP outputs 8 bits on D0-D7 to the CPU. $\overline{CSW}$ and $\overline{CSR}$ should never be simultaneously low. If both are low, the VDP outputs data on D0-D7 and latches in invalid data.

MODE determines the source or destination of a read or write data transfer. MODE is normally tied to a CPU low order address line (A14 for TMS 9900).

2.1.2 CPU Write to VDP Register

The VDP has eight write-only registers and one read-only status register. The write-only registers control the VDP operation and determine the way in which VRAM is allocated. The status register contains interrupt, sprite coincidence and fifth sprite status flags.

Each of the eight VDP write-only registers can be loaded using two 8-bit data transfers from the CPU. Table 1 describes the required format for the two bytes. The first byte transferred is the data byte, and the second byte transferred controls the destination. The most-significant bit of the second byte must be a '1'. The next four bits are '0's, and the lowest three bits make up the destination register number. The MODE input is high for both byte transfers.

To rewrite the data for an internal register after a byte of data has been loaded, the status register must be read so that internal logic will accept the next byte as data and not as a register destination. This situation may be encountered in interrupt-driven program environments. Whenever the status of VDP write parameters is in question, this procedure should be used. Note that the CPU address is destroyed by writing to the VDP register.

2.1.3 CPU Write to VRAM

The CPU transfers data to the VRAM through the VDP using a 14-bit autoincrementing address register. Two-byte transfers are required to set up the address register. A one-byte-transfer is then required to write the data to the addressed VRAM byte. The address register is then autoincremented. Sequential VRAM write require only one-byte-transfer since the address register is already set up. During setup of the address register, the two most-significant bits of the second address byte must be '0' and '1' respectively. MODE is high for both address transfers and low for the data transfer. $\overline{CSW}$ is used in all transfers to strobe the 8 bits into the VDP. See Table 1.

TABLE 1 – CPU/VDP DATA TRANSFERS

OPERATION	BIT 0	1	2	3	4	5	6	7	$\overline{CSW}$	$\overline{CSR}$	MODE
WRITE TO VDP REGISTER											
BYTE 1 DATA WRITE	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	0	1	1
BYTE 2 REGISTER SELECT	1	0	0	0	0	RS_0	RS_1	RS_2	0	1	1
WRITE TO VRAM											
BYTE 1 ADDRESS SET UP	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	0	1	1
BYTE 2 ADDRESS SET UP	0	1	A_0	A_1	A_2	A_3	A_4	A_5	0	1	1
BYTE 3 DATA WRITE	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	0	1	0
READ FROM VDP REGISTER											
BYTE 1 DATA READ	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	1	0	1
READ FROM VRAM											
BYTE 1 ADDRESS SET UP	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	0	1	1
BYTE 2 ADDRESS SET UP	0	0	A_0	A_1	A_2	A_3	A_4	A_5	0	1	1
BYTE 3 DATA READ	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	1	0	0

2.1.4 CPU Read from VDP Status Register

The CPU can read the contents of the status register with a single-byte transfer. MODE is high for the transfer. $\overline{CSR}$ is used to signal the VDP that a read operation is required.

2.1.5 CPU Read from VRAM

The CPU reads data from the VRAM through the VDP using the autoincrementing address register. A one-byte transfer is then required to read the data from the addressed VRAM byte. The address register is then autoincremented. Sequential VRAM data reads require only a one-byte transfer since the address register is already set up. During setup of the address register, the two most-significant bits of the second address byte must be '0's. By setting up the address this way, a read cycle to VRAM is initiated and read data will be available for the first data transfer to the CPU. (see Table 1). MODE is high for the address byte transfers and low for the data transfers. The VDP requires approximately 8 microseconds to fetch the VRAM byte following a data transfer and 3 microseconds following address setup.

2.1.6 VDP Interrupt

The VDP $\overline{INT}$ output pin is used to generate an interrupt at the end of each active-display scan, which is about every 1/60 second (color burst frequency/60, 192). The $\overline{INT}$ output is active when the interrupt Enable bit (IE) in VPD register 1 is a '1' and the F bit of the status register is a '1'. Interrupts are cleared when the status register is read.

2.1.7 VDP Initialization

The VDP is externally initialized whenever the $\overline{RESET}$ input is active (low) and must be held low for a minimum of 3 microseconds. The external reset synchronizes all clocks with its falling edge, sets the horizontal and vertical counters to known states, and clears VDP registers 0 and 1. The video display is automatically blanked since the BLANK bit in VDP register 1 becomes a '0'. The VDP, however, continues to refresh the VRAM even though the display is blanked. While the $\overline{RESET}$ line is active, the VDP does not refresh VRAM.

2.2 VDP/VRAM INTERFACE

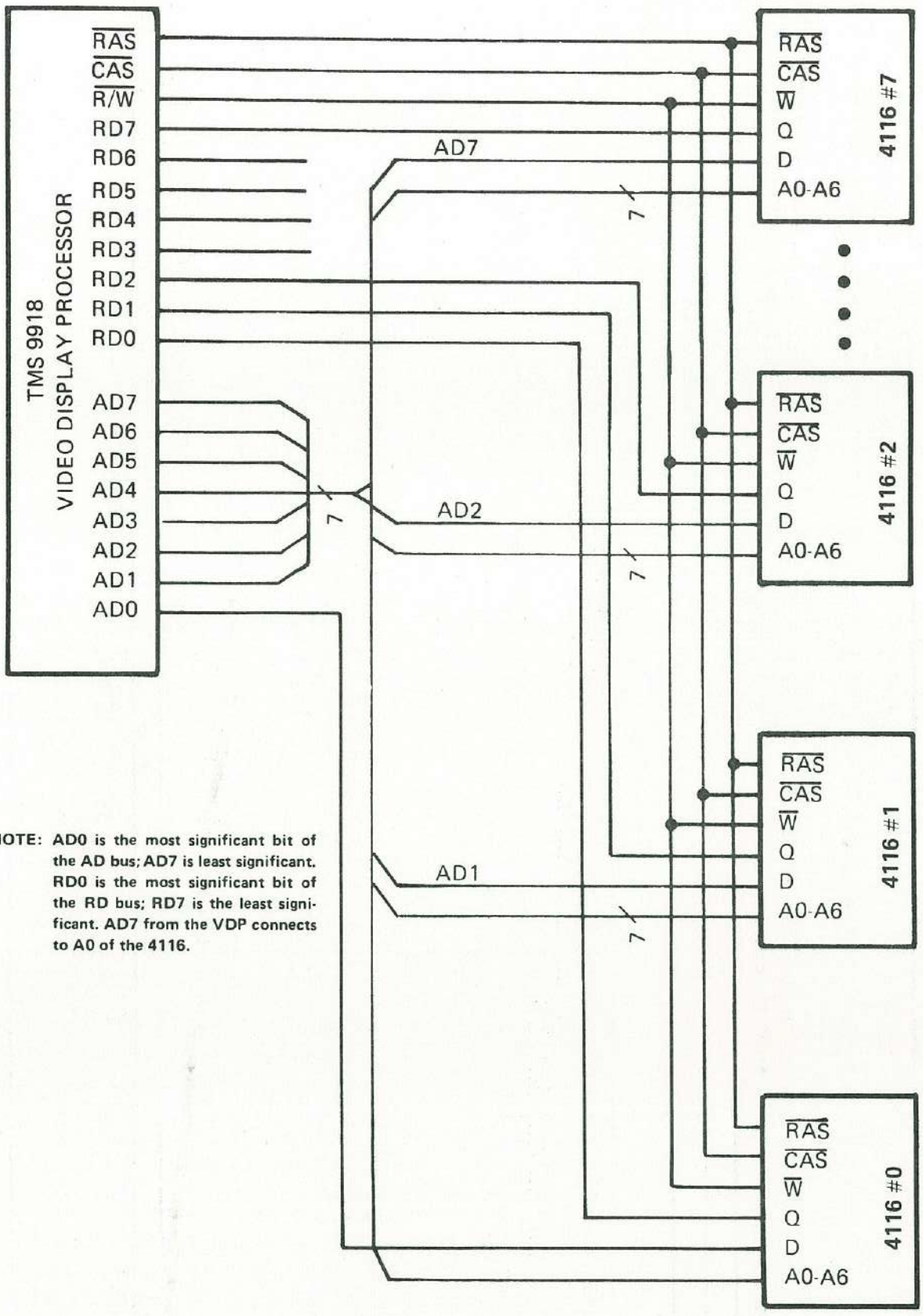
The VDP can access up to 16,384 bytes of VRAM using a 14-bit VRAM address. The VDP fetches data from the VRAM in order to process the video image as described later. The VDP also stores data in or reads in data from the VRAM during a CPU-VRAM data transfer. The VDP automatically refreshes the VRAM.

2.2.1 VRAM Interface Control Signals

The VDP-VRAM interface consists of two unidirectional 8-bit data buses and three control lines as shown in Figure 2-3. The VRAM outputs data to the VDP on the VRAM read data bus (RD0-RD7). The VDP outputs both the address and data to the VRAM over the VRAM address/data bus (AD0-AD7). The VRAM row address is output when $\overline{RAS}$ is active (low). The column address is output when $\overline{CAS}$ is active (low). Data is output to the VRAM when $\overline{R/W}$ is active (low).

2.2.2 VRAM Memory Types

The VDP can utilize 4027-type 4K, 4108-type 8K, or 4116-type 16K dynamic RAMs. The 4/16K bit in VDP register 1 is a '0' for 4027 type RAMs and a '1' for 4108- and 4116-type RAMs. Note that there is a minor difference between the way 4027's and 4108's/4116's are wired to the VDP. In the 4027, all $\overline{CE}$ pins are tied to ground. In the 4108/4116 the A6 lines on the 4116 and 4108 (the same pin as $\overline{CE}$ on 4027's) are all tied to AD1 on the TMS 9918A. A jumper can be used to select the VRAM-type.



NOTE: AD0 is the most significant bit of the AD bus; AD7 is least significant. RD0 is the most significant bit of the RD bus; RD7 is the least significant. AD7 from the VDP connects to A0 of the 4116.

FIGURE 2-3 – VRAM INTERFACE

2.3 VDP/TV INTERFACE

The composite video output signal coming from the VDP drives an NTSC color monitor. This signal incorporates all necessary horizontal and vertical synchronization signals as well as luminance and chrominance information. In monitor applications, the requirements of the monitor should be studied to determine if the VDP can be connected directly to it. The internal output buffer device on the composite video pin is a source-follower MOS transistor that requires an external pull-down reisstor to V_{SS} as shown in Figure 2-4. Typically a 1 kilohm resistor is recommended to provide a 1.9-volt synchronization level.

In some cases, it may be necessary to provide a simple interface circuit to match the VDP output voltages with the monitor specifications. To drive a standard television that is not outfitted with a composite video input, the signal can be run into the television antenna terminals by using an appropriate RF modulator on the VDP output. Care should be taken to ensure proper matchup between VDP, RF modulator, and TV.

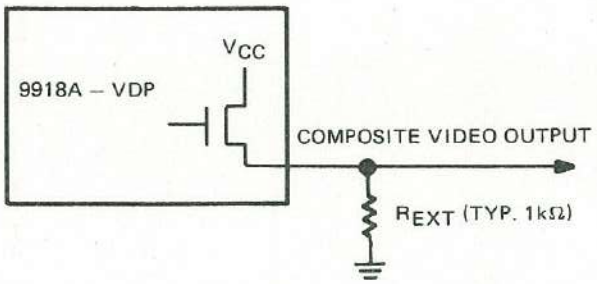


FIGURE 2-4 – COMPOSITE VIDEO PULL-DOWN CIRCUIT

2.4 WRITE-ONLY REGISTERS

The eight VDP write-only registers are shown in Figure 2-5. They are loaded by the CPU as described in Section 2.1.2. Registers 0 and 1 contain flags to enable or disable various VDP features and modes Registers 2 through 6 contain values that specify starting locations of various sub-blocks of VRAM. The definitions of these sub-blocks are described in Section 2.7. Register 7 is used to define backdrop and text colors.

The following is a description of each register:

2.4.1 Register 0

Register 0 contains two VDP option control bits. All other bits are reserved for future use and must be '0's.

BIT 6 M3 (mode bit 3) See Section 2.4.2 for table and description.

BIT 7 External Video enable/disable

'1' enables external video input
'0' disables external video input

2.4.2 Register 1

Register 1 contains 8 VDP option control bits.

BIT 0 4/16K selection

'0' selects 4027 RAM operation
'1' selects 4108/4116 RAM operation

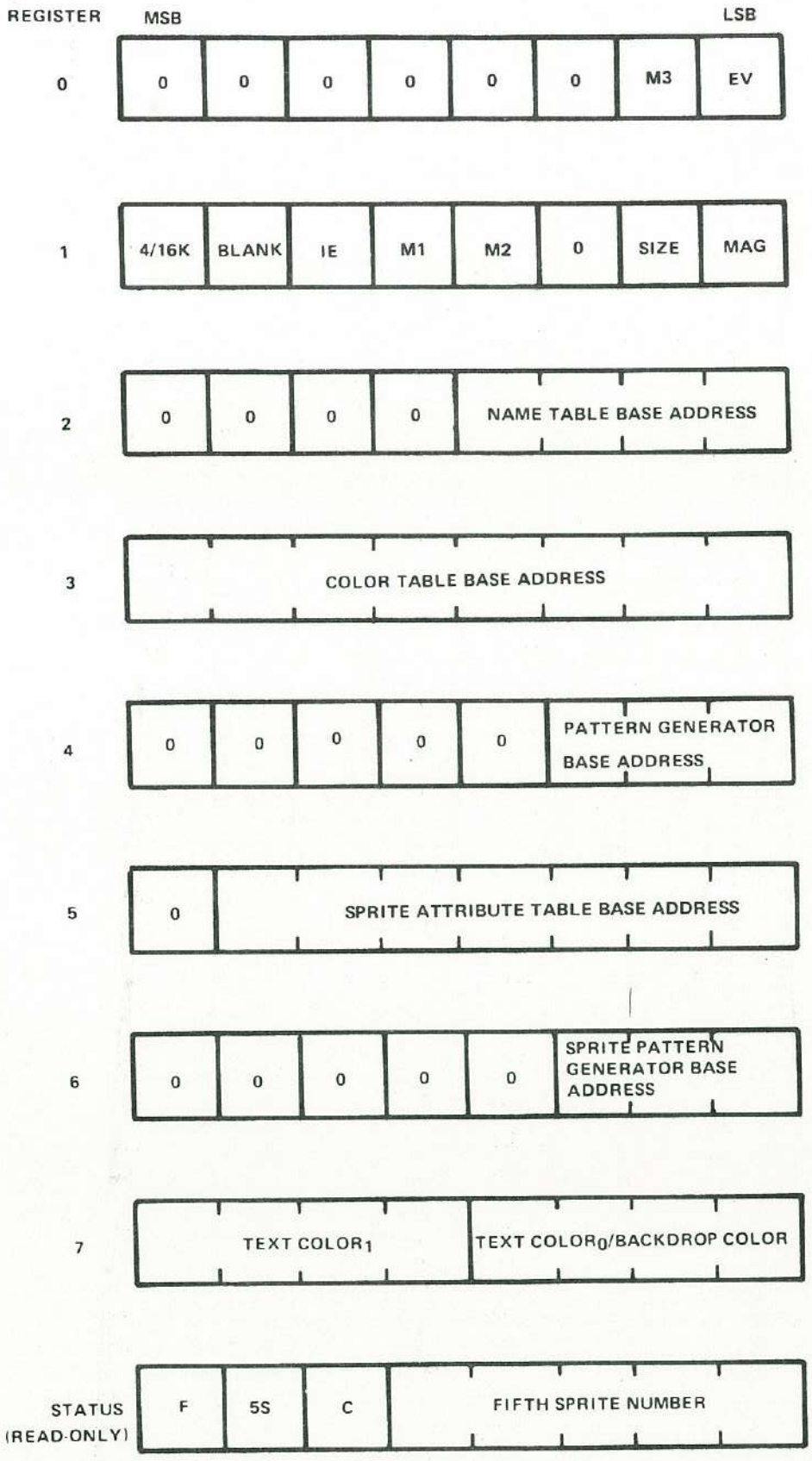


FIGURE 2-5 – VDP REGISTERS

BIT 1 BLANK enable/disable

'0' causes the active display area to blank
'1' enables the active display
Blanking causes the display to show border color only

BIT 2 IE (Interrupt Enable)

'0' disable VDP interrupt
'1' enable VDP interrupt

BIT 3,4 M1, M2 (mode bits 1 and 2)
M1, M2 and M3 determine the operating mode of the VDP:

M1	M2	M3	
0	0	0	Graphics I mode
0	0	1	Graphics II mode
0	1	0	Multicolor mode
1	0	0	Text mode

BIT 5 Reserved

BIT 6 Size (sprite size select)

'0' selects Size 0 sprites (8 X 8 bit)
'1' selects Size 1 sprites (16 X 16 bits)

BIT 7 MAG (Magnification option for sprites)

'0' selects MAG0 sprites (1X)
'1' selects MAG1 sprites (2X)

2.4.3 Register 2

Register 2 defines the base address of the Name Table sub-block. The range on its contents is from 0 to 15. The contents of the register form the upper 4 bits of the 14-bit Name Table addresses; thus the Name Table base address is equal to (register 2) * 400 (hex).

2.4.4 Register 3

Register 3 defines the base address of the Color Table sub-block. The range on its contents is from 0 to 255. The contents of the register form the upper 8 bits of the 14-bit Color Table addresses; thus the Color Table base address is equal to (register 3) * 40 (hex).

2.4.5 Register 4

Register 4 defines the base address of the Pattern, Text or Multicolor Generator sub-block. The range of its contents is 0 through 7. The contents of the register form the upper 3 bits of the 14-bit Generator addresses; thus the Generator base address is equal to (register 4) * 800 (hex).

2.4.6 Register 5

Register 5 defines the base address of the Sprite Attribute Table sub-block. The range of its contents is from 0 through 127. The contents of the register form the upper 7 bits of the 14-bit Sprite Attribute Table addresses; thus the base address is equal to (register 5) * 80 (hex).

2.4.7 Register 6

Register 6 defines the base address of the Sprite Pattern Generator sub-block. The range of its contents is 0 through 7. The contents of the register form the upper 3 bits of the 14-bit Sprite Pattern Generator addresses; thus the Sprite Pattern Generator base address is equal to (register 6) * 800 (hex).

2.4.8 Register 7

The upper 4 bits of register 7 contain the color code of color 1 in the Text mode. The lower 4 bits contain the color code for color 0 in the Text mode and the backdrop color in all modes. See Table 3 for color codes.

2.5 STATUS REGISTER

The VDP has a single 8-bit status register that can be accessed by the CPU. The status register contains the interrupt pending flag, the sprite coincidence flag, the fifth sprite flag, and the fifth sprite number, if one exists. The format of the status register is shown in Figure 2-5. A discussion of the contents follows.

The status register may be read at any time to test the F, C, and 5S status bits. Reading the status register will clear the interrupt flag, F. Asynchronous reads will, however, cause the frame flag (F) bit to be reset and therefore missed. Consequently, the status register should be read only when the VDP interrupt is pending.

2.5.1 Interrupt Flag (F)

The F status flag in the status register is set to '1' at the end of the raster scan of the last line of the active display. It is reset to a '0' after the status register is read or when the VDP is externally reset. If the Interrupt Enable bit in VDP register 1 is active ('1'), the VDP interrupt output ($\overline{INT}$) will be active (low) whenever the F status flag is a '1'.

2.5.2 Coincidence Flag (C)

The C status flag in the status register is set to a '1' if two or more sprites "coincide". Coincidence occurs if any two sprites on the screen have one or more overlapping pixels. Transparent colored sprites, as well as those that are partially or completely off the screen, are also considered. Sprites beyond the Sprite Attribute Table terminator ($D0_{16}$) are not considered. The 'C' flag is cleared to a '0' after the status register is read or the VDP is externally reset.

2.5.3 Fifth Sprite Flag (5S) and Number

The 5S status flag in the status register is set to a '1' whenever there are five or more sprites on a horizontal line (lines 0 to 192) and the frame flag is equal to a '0'. The 5S status flag is cleared to a '0' after the status register is read or the VDP is externally reset. The number of the fifth sprite is placed into the lower 5 bits of the status register when the 5S flag is set and is valid whenever the 5S flag is '1'. The setting of the fifth sprite flag will not cause an interrupt.

2.6 OSCILLATOR AND CLOCK GENERATION

The VDP is designed to operate with a 10.738635 megahertz ±50 ppm crystal input to generate the required internal clock signals. A fundamental frequency, parallel-mode crystal is used as the frequency reference for the internal clock oscillator, which is the master time base for all system operations. This master clock is divided by two to generate the pixel clock (5.3 megahertz) and by three to provide the CPUCLK (3.58 megahertz). The GROMCLK is developed from the master clock frequency divided by 24.

2.6.1 Color Phase Generation

The 10.7+ megahertz master clock and its complement are used to generate an internal six-phase 3.579545 megahertz (±10 hertz) clock to provide the video color signals and the color burst reference for use in developing the composite video output signal. While the VDP signals are not exact equivalents to the standard NTSC colors, the differences can easily be adjusted with the color and tint controls of the target color television.

2.6.2 Video Sync and Control Generation

The vertical and horizontal control signals are generated by decoding the outputs of the horizontal and vertical counters. The horizontal counter is driven by the pixel clock, and the horizontal counter in turn increments the vertical counter. Table 2 gives the relative count values of the screen display parameters. Within the active display area during Graphics I mode, the 3 least-significant bits of the horizontal counter address the individual picture element of each pattern displayed. Also, during the vertical active display period, the 3 least-significant bits of the vertical counter address each individual line in the 8 X 8 patterns. The Graphics II, Multicolor and Text modes use the counters similarly.

The VDP operates at 262 lines per frame and approximately 60 frames per second in a non-interlaced mode of operation.

TABLE 2 – SCREEN DISPLAY PARAMETERS

PARAMETER	PIXEL CLOCK CYCLES	
HORIZONTAL	**PATTERN OR MULTICOLOR**	**TEXT**
HORIZONTAL ACTIVE DISPLAY	256	240
RIGHT BORDER	15	25
RIGHT BLANKING	8	8
HORIZONTAL SYNC	26	26
LEFT BLANKING	2	2
COLOR BURST	14	14
LEFT BLANKING	8	8
LEFT BORDER	13	19
	342	342
VERTICAL	**LINE**	
VERTICAL ACTIVE DISPLAY	192	
BOTTOM BORDER	24	
BOTTOM BLANKING	3	
VERTICAL SYNC	3	
TOP BLANKING	13	
TOP BORDER	27	
	262	

2.7 VIDEO DISPLAY MODES

The VDP displays an image on the screen that can best be envisioned as a set of display planes sandwiched together. Figure 2-6a shows the definition of each of the planes. Objects on planes closest to the viewer have higher priority. In cases where two entities on two different planes are occupying the same spot on the screen, the entity on the higher priority plane will show at that point. For an entity on a specific plane to show through, all planes in front of that plane must be transparent at that point. The first 32 planes (Figure 2-6b) each may contain a single sprite. (Sprites are pattern objects whose positions on the screen are defined by horizontal and vertical coordinates in VRAM.) The areas of the Sprite Planes, outside of the sprite itself, are transparent. Since the coordinates of the sprite are in terms of pixels, the sprite can be positioned and moved about very accurately. Sprites are available in three sizes: 8 X 8 pixels, 16 X 16 pixels, and 32 X 32 pixels. Behind the Sprite Plane is the Pattern Plane. The Pattern Plane is used for textual and graphics images generated by the Text, Graphics I, Graphics II, or Multicolor modes. Behind the Pattern Plane is the backdrop, which is larger in area than the other planes so that it forms a border around the other planes. The last and lowest priority plane is the External Video Plane. Its image is defined by the external video input pin. The backdrop consists of a single color used for the display borders and as the default color for the active display area. The default color is stored in the VDP register 7. When the backdrop color register contains the transparent code, the backdrop automatically defaults to black if the external video mode is not selected.

The 32 Sprite Planes are used for the 32 sprites in the Multicolor and Graphics modes. They are not used in the Text mode and are automatically transparent. Each of the sprites can cover an 8 X 8, 16 X 16, or 32 X 32 pixel area on its plane. Any part of the plane not covered by the sprite is transparent. All or part of each sprite may also be transparent. Sprite 0 is on the outside or highest plane, and sprite 31 is on the plane immediately adjacent to Pattern Plane. Whenever a pixel in a Sprite Plane is transparent, the color of the next plane can be seen through that plane. If, however, the sprite pixel is non-transparent, the colors of the lower planes are automatically replaced by the sprite color. There is also a restriction on the number of sprites on a line. Only four sprites can be active on any horizontal line. Additional sprites on a line will be automatically made transparent for that line. Only those sprites that are active on the display will cause the coincidence flag to set. The VDP status register provides a flag bit and the number of the fifth sprite whenever this occurs. The Pattern Plane is used in the Text, Multicolor, and Graphics modes for display of the graphic patterns of characters. Whenever a pixel on the Pattern Plane is non-transparent, the backdrop color is automatically replaced by the Pattern Plane color. When a pixel in the Pattern Plane is transparent, the backdrop color can be seen through the Pattern Plane.

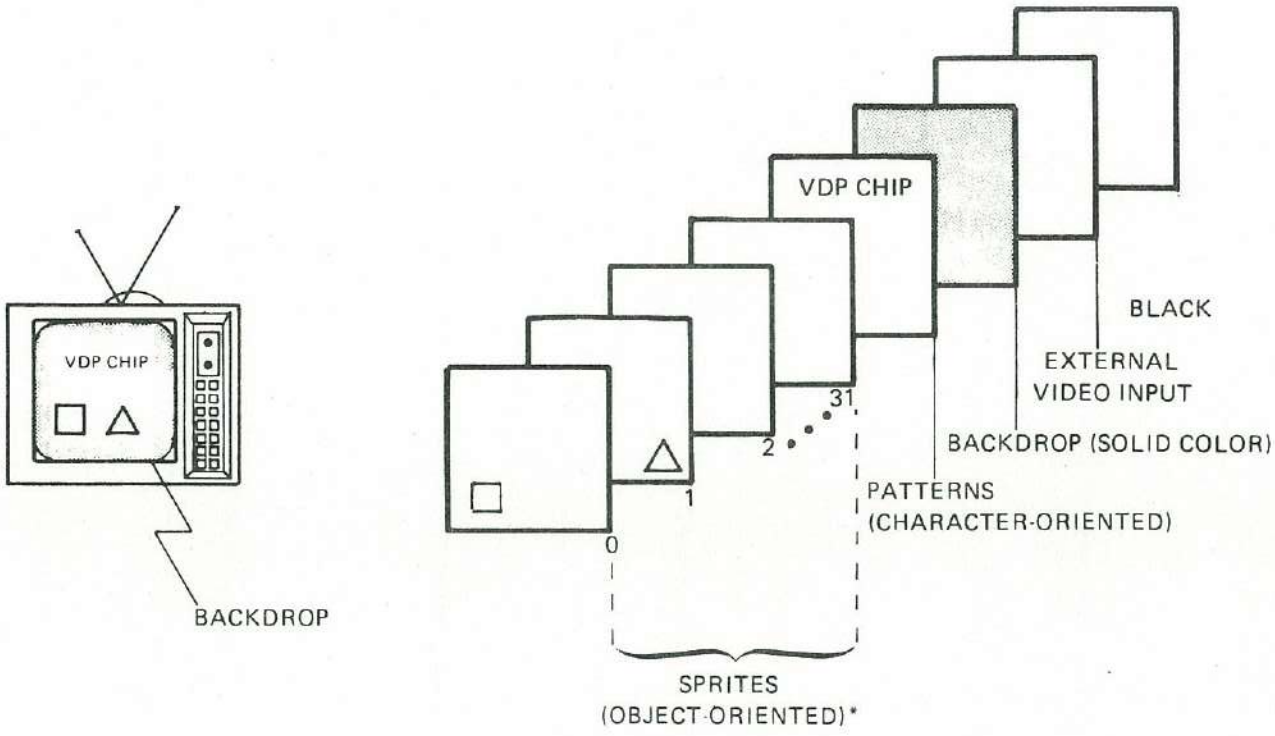


FIGURE 2-6a – VDP DISPLAY PLANES

The VDP has four video color display modes that appear on the Pattern Plane: Graphics I mode, Graphics II mode, Text mode, and Multicolor mode. Graphics I and Graphics II modes cause the Pattern Plane to be broken up into groups of 8 X 8 pixels, called pattern positions. Since the full image is 256 X 192 pixels, there are 32 X 24 pattern positions on the screen in the graphics modes. In Graphics I mode, 256 possible patterns may be defined for the 768 pattern positions with two unique colors allowed for each pattern definition. Graphics II mode provides, through a unique mapping scheme, 768 pattern definitions for the 768 pattern positions. Graphics II mode also allows the selection of two unique colors for each line of a pattern definition. Thus, all 15 colors plus transparent may be used in a single pattern position. In Text mode, the Pattern Plane is broken into groups of 6 X 8 pixels, called text positions. There are 40 X 24 text positions on the screen in this mode. In Text mode, sprites do not appear on the screen and two colors are defined for the entire screen. In Multicolor mode, the screen is broken into a grid of 64 X 48 positions, each of which is a 4 X 4 pixel. Within each position, one unique color is allowed.

The VDP registers define the base addresses for several sub-blocks within VRAM. These sub-blocks form tables which are used to produce the desired image on the TV screen. The Pattern Name Table, the Pattern Generator Table and the Sprite Generator Table are used to form the sprites. The contents of these tables must all be provided by the microprocessor. Animation is achieved by altering the contents of VRAM in real time.

The VDP can display the 15 colors shown in Table 3. The VDP colors also provide eight different gray levels for displays on monochrome televisions; the luminance values in the table indicate these levels, 0.00 being black and 1.00 being white. Whenever all planes are of the transparent color at a given point, the color shown at that point will be black.

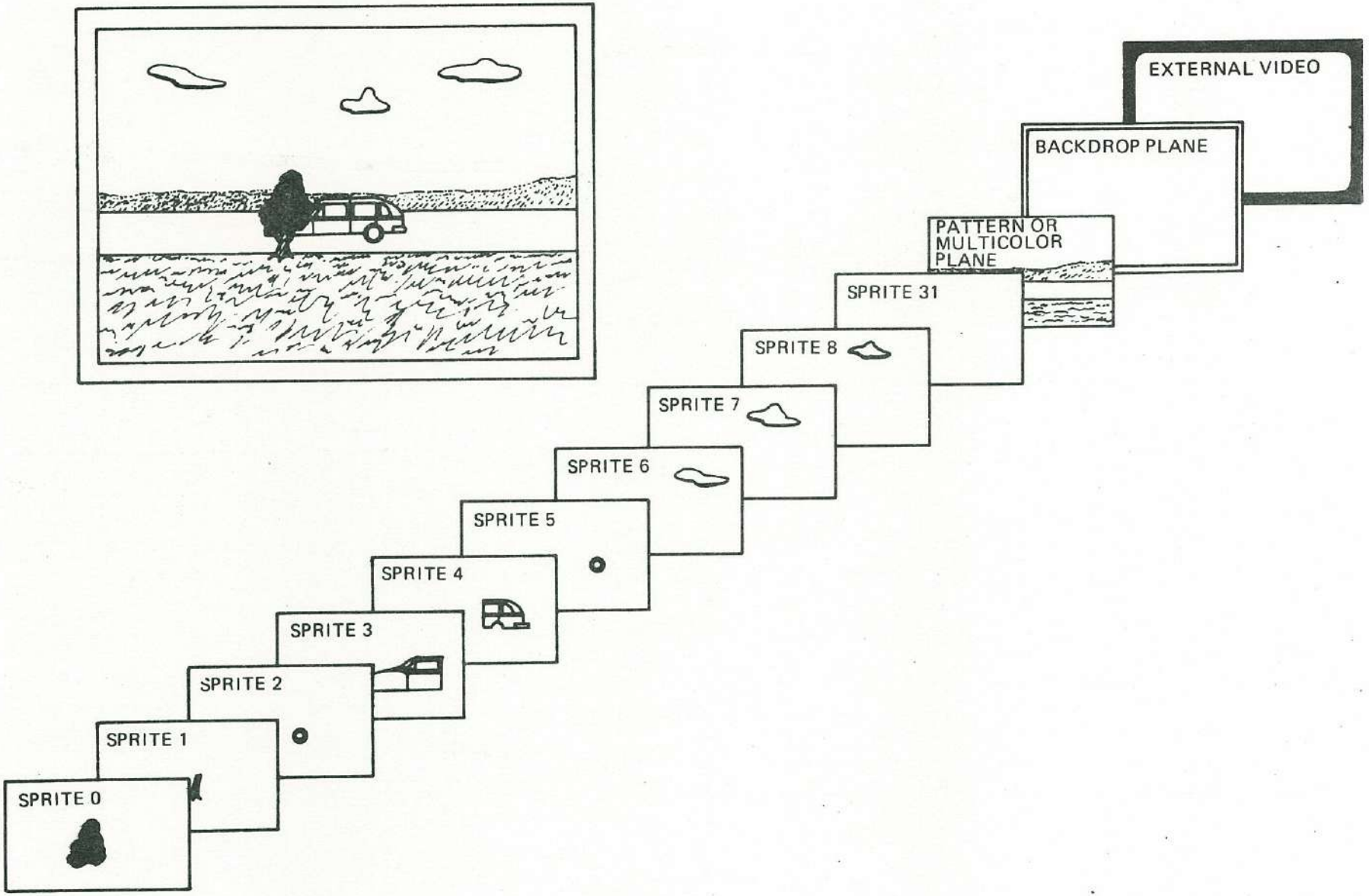


FIGURE 2-6b – VDP DISPLAY PLANES

TABLE 3. COLOR ASSIGNMENTS

COLOR (HEX)	COLOR	LUMINANCE (DC VALUE)	CHROMINANCE (AC VALUE)
0	TRANSPARENT	0.00	—
1	BLACK	0.00	—
2	MEDIUM GREEN	.60	.60
3	LIGHT GREEN	.80	.53
4	DARK BLUE	.47	.73
5	LIGHT BLUE	.67	.60
6	DARK RED	.53	.53
7	CYAN	.80	.73
8	MEDIUM RED	.67	.73
9	LIGHT RED	.80	.73
A	DARK YELLOW	.87	.53
B	LIGHT YELLOW	1.00	.40
C	DARK GREEN	.47	.60
D	MAGENTA	.60	.47
E	GRAY		—
F	WHITE	1.00	—
—	BLACK LEVEL	0.00	—
—	COLOR BURST	0.00	.40
—	SYNC LEVEL	−0.40	—

2.7.1 Graphics I Mode

The VDP is in Graphics I mode when M1, M2, and M3 bits in VDP registers 1 and 0 are zero. In Graphics I mode the Pattern Plane is divided into a grid of 32 columns by 24 rows of pattern positions (see Figure 2-7). Each of the pattern positions contains 8 X 8 pixels. The tables in VRAM used to generate the Pattern Plane are the Pattern Color Table. Figure 2-8 illustrates the mapping of these tables into the Pattern Plane. A total of 2848 VRAM bytes are required for the Pattern Name, Color and Generator tables. Less memory is required if all 256 possible pattern definitions are not required. The tables can be overlapped to reduce the amount of VRAM needed for pattern generation. Examples of VRAM memory allocation are provided in Section 3.

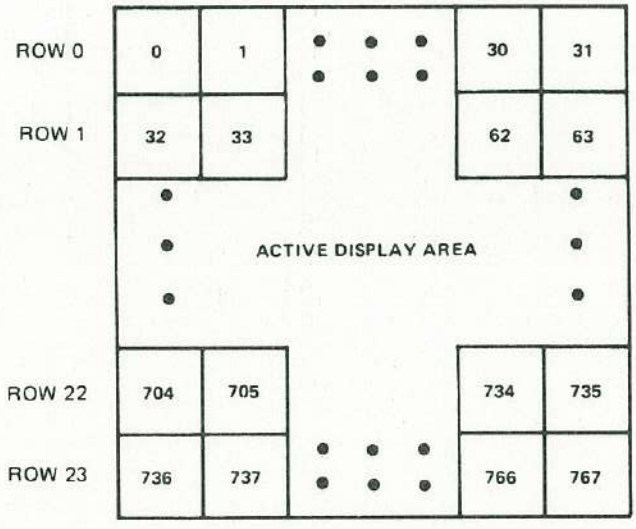


FIGURE 2-7 – PATTERN GRAPHICS NAME TABLE MAPPING

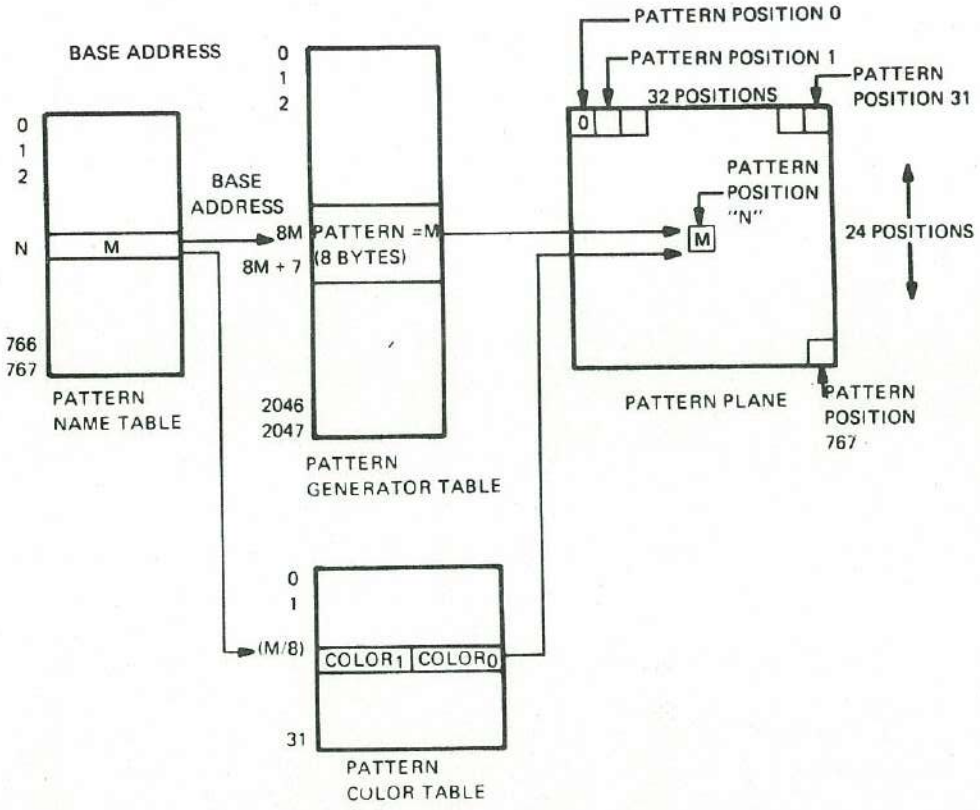


FIGURE 2-8 – PATTERN MODE MAPPING

The Pattern Generator Table contains a library of patterns that can be displayed in the pattern positions. It is 2048 bytes long, and is arranged into 256 patterns, each of which is eight bytes long, yielding 8 X 8 bits. All of the '1's in the eight-byte pattern can designate one color (color 1), while all the '0's can designate another color (color 0).

The full 8-bit pattern name is used to select one of the 256 pattern definitions in the Pattern Generator Table. The table is a 2048-byte block in VRAM beginning on a 2 kilobyte boundary. The starting address of the table is determined by the generator base address in VDP register 4. The base address forms the three most-significant bits of the 14-bit VRAM address for each Pattern Generator Tableentry. The next 8 bits indicate the 8-bit name of the selected pattern definition. The lowest 3 bits of the VRAM address indicate the row number within the pattern definition.

Eight bytes are required for each of the 256 possible unique 8 X 8 pattern definitions. The first byte defines the first row of the pattern, and the second byte defines the second row. The first bit of each of the eight bytes define the first column of the pattern. The remaining rows and columns are similarily defined. Each bit entry in the pattern definition selects one of the two colors for that pattern. A '1' bit selects the color code (color 1) contained in the most-significant four bits of the corresponding color table byte. A '0' bit selects the other color code (color 0). An example of pattern definition mapping is provided in Figure 2-9.

	PATTERN					PATTERN DEFINITION							
0	C	C	C	C	C	0	1	1	1	1	1	0	0
1					C	0	0	0	0	0	1	0	0
2					C	0	0	0	0	0	1	0	0
3		C	C	C	C	0	0	1	1	1	1	0	0
4					C	0	0	0	0	0	1	0	0
5					C	0	0	0	0	0	1	0	0
6	C	C	C	C	C	0	1	1	1	1	1	0	0
7						0	0	0	0	0	0	0	0
ROW/BYTE	**COLUMN**					**BIT**							
	0	**1**	**2**	**3**	**4**	**5**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

NOTES: VDP register 7 entry: 71_{16}.
Color code 7 is cyan (signified above by 'C').
Color code 1 is black (signified above by a space).
Bit 0 is the most significant bit of each data byte.

FIGURE 2-9 – PATTERN DISPLAY MAPPING

The color of the '1's and '0's is defined by the Pattern Color Table that contains 32 entries each of which is one byte long. Each entry defines two colors: the most-significant 4 bits of each entry define the color of the '1's, and the least-signigicant 4 bits define the color of the '0's. The first entry in the color table defines the colors for patterns 0 to 7; the next entry for patterns 8 to 15, and so on. (See Table 4 for assignments.) Thus, 32 different pairs of colors may be displayed simultaneously.

The Pattern Name Table is located in a contiguous 768-byte block in VRAM beginning on a 1 kilobyte boundary. The starting address of the Name Table is determined by the 4-bit Name Table base address field in VDP register 2. The base address forms the upper four bits of the 14-bit VRAM address. The lower 10 bits of the VRAM address are formed from the row and column counters. An example of pattern name table addressing is given in Section 3.

Each byte entry in the Name Table is the name of or the pointer to a pattern definition in the Pattern Generator Table. The upper five bits of the eight-bit name identify the color group of the pattern. There are 32 groups of eight patterns. The same two colors are used for all eight patterns in a group; the color codes are stored in the VDP Color Table. The Color Table is located in a 32-byte block in VRAM beginning on a 64-byte boundary. The table starting address is determined by the 8-bit Color Table base address in VDP register 3. The base address forms the upper eight bits of the 14-bit Color Table entry VRAM address. The next bit is a '0' and the lowest 5 bits are equal to the upper 5 bits of the corresponding Name Table entries. An example of Color Table addressing is provided in Section 3.

Since the tables in VRAM have their base addresses defined by the VDP registers, a complete switch of the values in the tables can be made by simply changing the values in the VDP registers. This is especially useful when one wishes to time-slice between two or more screens of graphics.

When the Pattern Generator Table is loaded with a pattern set. manipulation of the Pattern Name Table contents can change the appearance of the screen. Alternatively, a dynamically changing set of patterns throughout the course of a graphics session is easily accomplished since all tables are in VRAM.

For textual applications, the desired character set is typically loaded into the Pattern Generator first. The official USASCII character set might be loaded into the Pattern Generator in such a way that the pattern numbers correspond to the 8-bit ASCII codes for that pattern; e.g., the pattern for the letter "A" would be loaded into pattern number 41_{16} in the Pattern Generator. Next the Pattern Color Table would be loaded up with the proper color set. To print a textual message on the screen, write the proper ASCII codes out to the Pattern Name Table.

Images can be formed using the Pattern Plane. To display an object of size 8 X 8 pixels or smaller, only one pattern would need to be defined. To display a larger figure, the figure should be broken up into smaller 8 X 8 squares. Then multiple patterns can be defined, and the Pattern Generator and Pattern Name Table set up appropriately. Note that rough motion of objects requires merely updating entries in the Pattern Name Table.

TABLE 4. PATTERN COLOR TABLE

Byte No.	Pattern No.
0	0..7
1	8..15
2	16..23
3	24..31
4	32..39
5	40..47
6	48..55
7	56..63
8	64..71
9	72..79
10	80..87
11	88..95
12	96..103
13	104..111
14	112..119
15	120..127
16	128..135
17	136..143
18	144..151
19	152..159
20	160..167
21	168..175
22	176..183
23	184..191
24	192..199
25	280..207
26	208..215
27	216..223
28	224..231
29	232..239
30	240..247
31	248-255

A total of 2848 VRAM bytes are required for the Pattern, Name, Color and Generator tables. Less memory is needed if all 256 possible pattern definitions are not required; the tables can be overlapped to reduce the amount of VRAM needed for pattern generation. Examples of VRAM memory allocation are provided in Section 3.

2.7.2 Graphics II Mode

The VDP is in the Graphics II mode when mode bits M1 = 0, M2 = 0, and M3 = 1. The Graphics II mode is similar to Graphics I mode except it allows a larger library of patterns so that a unique pattern generator entry may be made for each of the 768 (32 X 24) pattern positions on the video screen. Additionally, more color information is included in each 8 X 8 graphics pattern. Thus two unique colors may be specified for each byte of the 8 X 8 pattern. A larger amount of VRAM (12 kilobytes) is required to implement the full usage of the Graphics II mode.

Like Graphics I mode, the Graphics II mode Pattern Name Table contains 768 entries which correspond to the 768 pattern positions on the display screen. Because the Graphics I mode pattern names are only 8 bits in length, a maximum of 256 pattern definitions may be addressed using the addressing scheme discussed in the previous section. Graphics II mode, however, segments the display screen into three equal parts of 256 pattern positions each and also segments the Pattern Generator Table into three equal blocks of 2048 bytes each. Pattern definitions in the first third correspond to pattern positions in the upper third of the display screen. Likewise pattern definitions in the second and third blocks of the Pattern Generator Table correspond to the second and third areas of the Pattern Plane. The pattern Name Table is also segmented into three blocks of 256 names each so that names found in the upper third, reference pattern definitions found in the upper 2048 bytes in Pattern Generator Table. Likewise the second and third blocks reference pattern definitions in the second 2048 byte block and third 2048 byte block respectively. Thus, if 768 patterns are uniquely specified, an 8-bit pattern name will be used three times, once in each segment of the Pattern Name Table. The Pattern Generator Table falls on eight kilobyte boundaries and may be located in the upper or lower half of 16K memory based on the MSB of the pattern generator base in VDP register 4. The LSB's must be set to all '1's.

The Color Table is also 6144 bytes long and is segmented into three equal blocks of 2048 bytes. Each entry in the Pattern Color Table is eight bytes which provides the capability to uniquely specify color 1 and color 0 for each of the eight bytes of the corresponding pattern definition. The addressing scheme is exactly like that of the Pattern Generator Table except for the location of the table in VRAM. This is controlled by the loading of the MSB of the color base in VDP register 3. The LSB's must be set to all '1's.

Figure 2-10 is an example of the Graphics II mode mapping scheme. Note that pattern names P1, P2, P3 correspond to pattern generator entries in the three blocks of the Pattern Generator Table. Note also how these three names map to the display screen. Figure 2-11 an example of a Pattern Generator and Pattern Color Table entry.

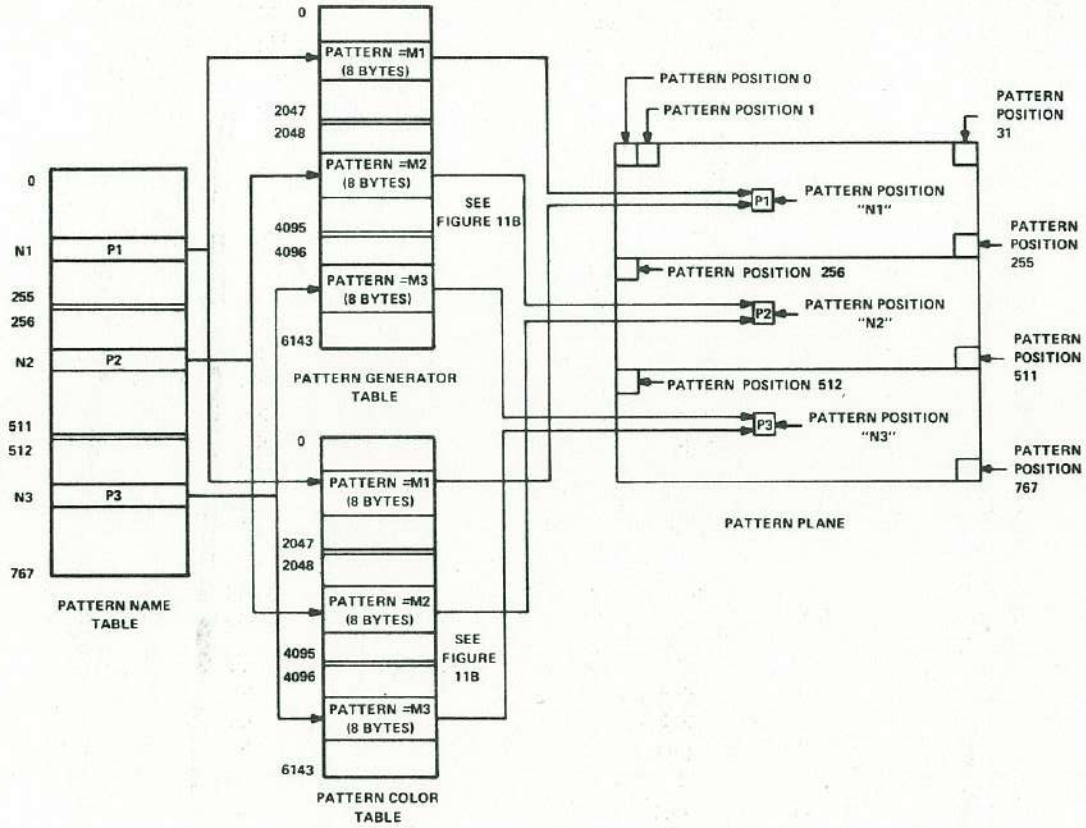


FIGURE 2-10 – GRAPHICS II MODE MAPPING

ROW	PATTERN GENERATOR TABLE ENTRY	PATTERN	PATTERN COLOR TABLE ENTRY	ROW
0	0 1 0 0 0 0 0 1	B 1 B B B B B 1	1 (BLACK) | B (LT. YELLOW)	0
1	0 0 1 0 0 0 1 0	B B 7 B B B 7 B	7 (CYAN) | B (LT. YELLOW)	1
2	0 0 0 1 0 1 0 0	B B B C B C B B	C (GREEN) | B (LT. YELLOW)	2
3	0 0 0 0 1 0 0 0	B B B B E B B B	E (GRAY) | B (LT. YELLOW)	3
4	0 0 0 0 1 0 0 0	B B B B 8 B B B	8 (MED. RED) | B (LT. YELLOW)	4
5	0 0 0 0 1 0 0 0	B B B B 5 B B B	5 (LT. BLUE) | B (LT. YELLOW)	5
6	0 0 0 0 1 0 0 0	B B B B 6 B B B	6 (DK. RED) | B (LT. YELLOW)	6
7	0 0 0 0 1 0 0 0	B B B B D B B B	D (MAGENTA) | B (LT. YELLOW)	7

FIGURE 2-11 – PATTERN DISPLAY MAPPING

2.7.3 Multicolor Mode

The VDP is in Multicolor mode when mode bits M1 = 0, M2 = 1, and M3 = 0. Multicolor mode provides an unrestricted 64 X 48 color square display. Each color square contains a 4 X 4 block of pixels. The color of each of the color squares can be any one of the 15 video display colors plus transparent. Consequently, all 15 colors can be used simultaneously in the Multicolor mode. The Backdrop and Sprite Planes are still active in the Multicolor mode.

The Multicolor Name Table is the same as that for the graphics modes, consisting of 768 name entries. The name no longer points to a color list; rather color is now derived from the Pattern Generator Table. The name points to an eight-byte segment of VRAM in the Pattern Generator Table.

Only two bytes of the eight-byte segment are used to specify the screen image. These two bytes specify four colors, each color occupying a 4 X 4 pixel area. The four MSB's of the first byte define the color of the upper left quarter of the multicolor pattern; the LSB's define the color of the upper right quarter. The second byte similarly defines the lower left and right quarters of the multicolor pattern. The two bytes thus map into a 8 X 8 pixel multicolor pattern. (See Figure 2-12).

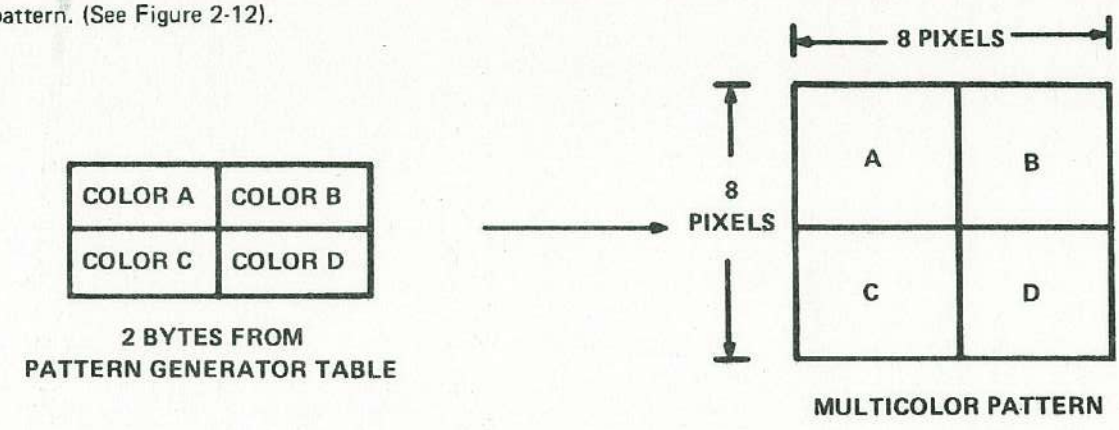


FIGURE 2-12 – MULTICOLOR LIST MAPPING

The location of the two bytes within the eight-byte segment pointed to by the name is dependent upon the screen position where the name is mapped. For names in the top row (names 0-31), the two bytes are the first two within the groups of eight-byte segments pointed to by the names. The next row of names (32-63) uses the third and fourth bytes within the eight-byte segments. The next row of names uses the fifth and sixth bytes while the last row of names uses the seventh and eighth. This series repeats for the remainder of the screen.

For example, referring to Figure 2-13, if Name Table entry 0 (pattern position 0) multicolor block #N (name = N), the multicolor pattern displayed will be an 8 X 8 pixel block consisting of colors A, B, C, D comprising the first two bytes of the Multicolor Table. If, however, name #N is located in Name Table entry 33, (Pattern position 33), the colors displayed will be colors E, F, G, H as specified by the third and fourth bytes of the multicolor block pointed to by the name. Likewise pattern positions which lie in rows 2 and 3 would cause colors I, J, K, L and colors M, N, O, P respectively to be displayed. Thus it can be seen that the color displayed from the multicolor generator block is dependent upon pattern position on the screen. Figure 2-14 illustrates the multicolor mode mapping scheme.

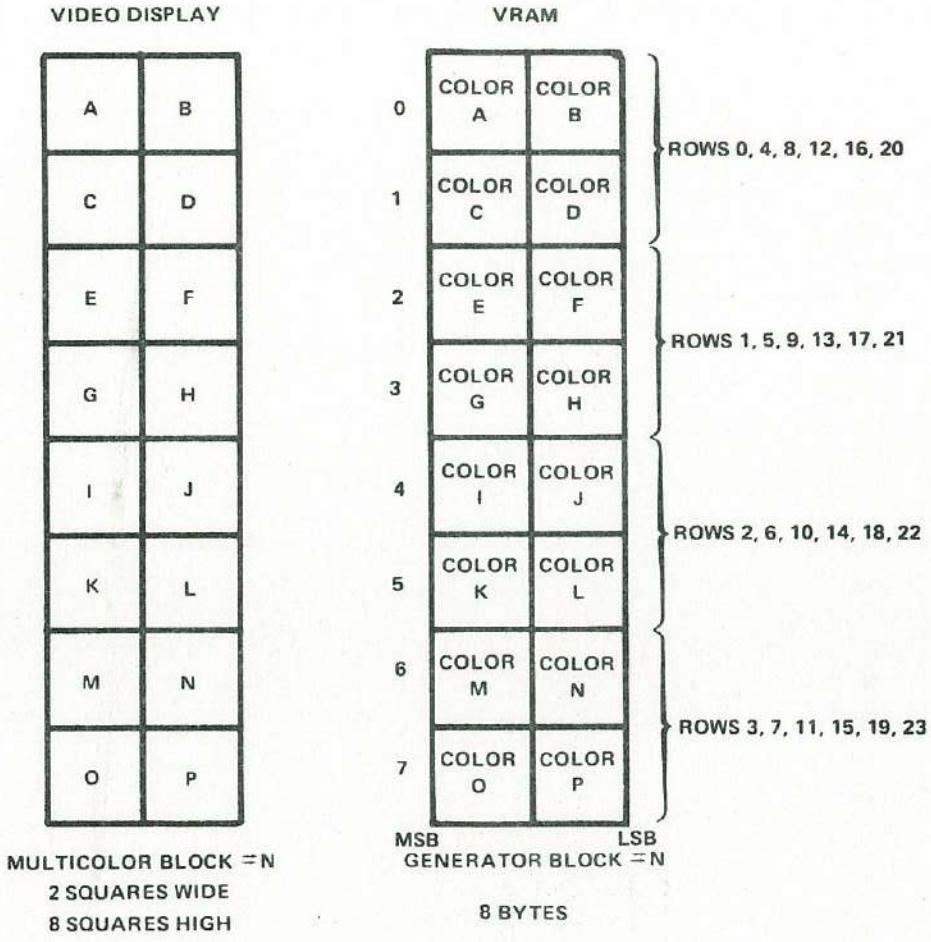


FIGURE 2-13 – MULTICOLOR BLOCK DISPLAY

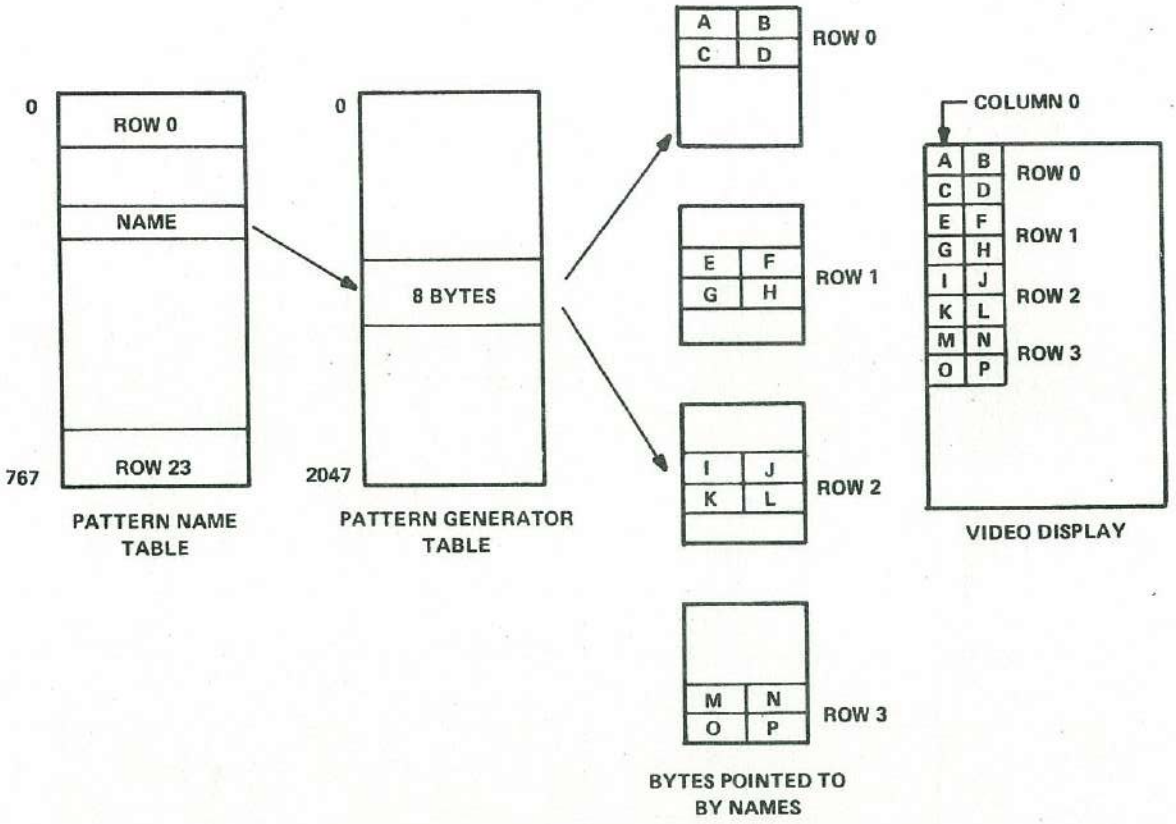


FIGURE 2-14 – MULTICOLOR MODE MAPPING

The mapping of VRAM contents to screen image is simplified by using duplicate names in the Name Table.Since the series of bytes used within the eight-byte segment repeats every four rows, the four rows in the same column can use the same name. Then the eight-byte segment specifies a 2 X 8 color square pattern on the screen as a straightforward translation from the eight-byte segment in VRAM pointed to by the common name.

When used in this manner, 768 bytes are still used for the NameTable and 1536 bytes are used for the color information in the Pattern Generator Table (24 rows X 32 columns X 8 bytes/pattern position). Thus a total of 1728 bytes in VRAM are required. It should be noted that the tables begin on even 1K and 2K boundaries and are therefore not contiguous. An example of multicolor VRAM memory allocation is provided in Section 3.

2.7.4 Text Mode

The VDP is in Text mode when mode bits M1 = 1, M2 = 0, and M3 = 0. In the Text mode, the screen is divided into a grid of 40 text positions across and 24 down. (See Figure 2-15). Each of the text positions contains six pixels across and eight pixels down. The tables used to generate the Pattern Plane are the Pattern Name Table and the Pattern Generator Table. There can be up to 256 unique patterns defined at any time. The pattern definitions are stored in the Pattern Generator Table in VRAM and can be dynamically changed. The VRAM contains a Pattern Name Table which maps the pattern definitions into each of the 960 pattern cells on the Pattern Plane (Figure 2-16). Sprites are not available in Text mode.

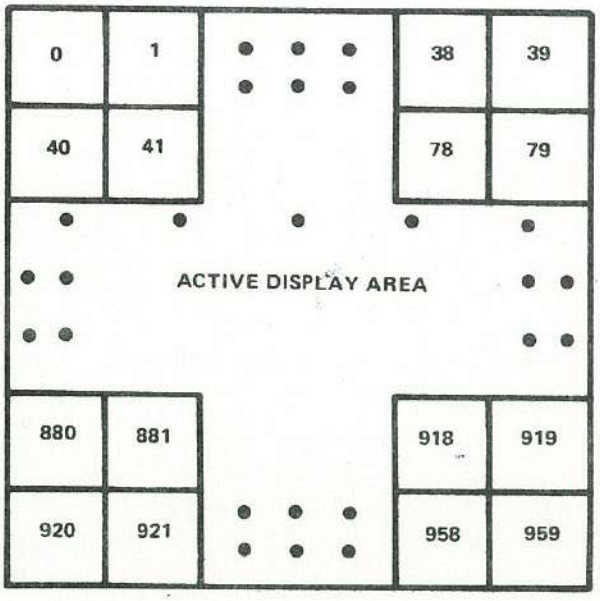


FIGURE 2-15 – TEXT MODE NAME TABLE PATTERN POSITIONS

As in the case of the Graphics modes, the Pattern Generator Table contains a library of text patterns that can be displayed in the text positions. It is 2048 bytes long, and is arranged in 256 text patterns, each of which is eight bytes long. Since each text position on the screen is only six pixels across, the least-significant 2 bits of each text pattern are ignored, yielding 6 X 8 bits in each text pattern. Each block of eight bytes defines a text pattern in which all the '1's in the text pattern take on one color when displayed on the screen, while all the '0's take on another color. These colors are chosen by loading VDP register 7 with the color 1 and color 0 in the left and right nibbles respectively (see Section 2.4).

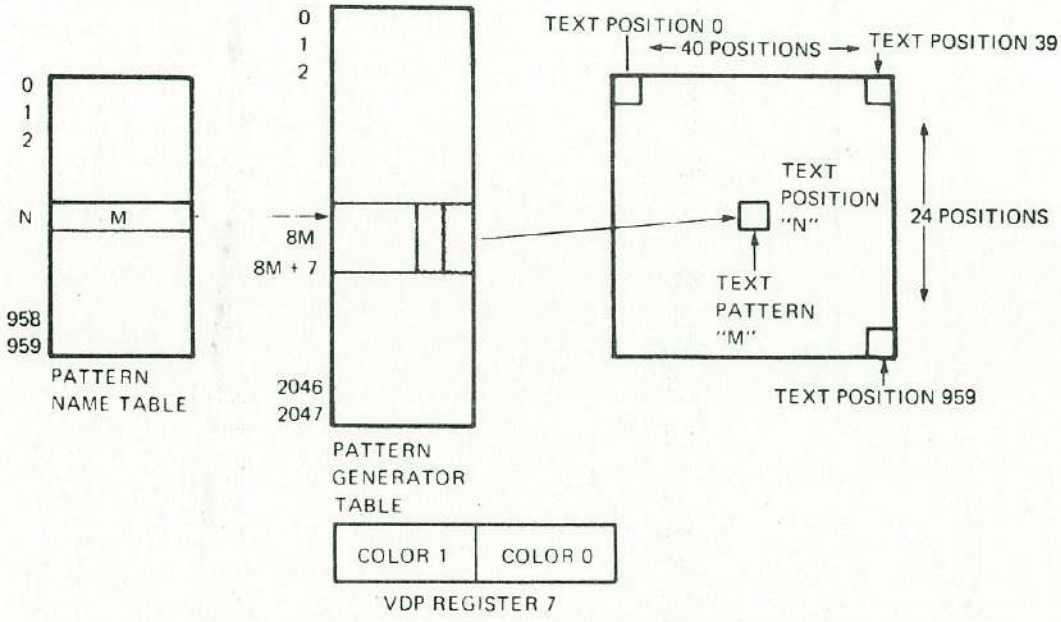


FIGURE 2-16 – MAPPING OF VRAM INTO THE PATTERN PLANE IN TEXT MODE

In the Text mode, the Pattern Name Table determines the position of the text pattern on the screen. There are 960 entries in the Pattern Name Table, each one byte long. There is a one-to-one correspondence between text pattern positions on the screen and entries in the Pattern Name Table (40*24 = 960). The first 40 entries corresponds to the top row of text pattern positions on the screen, the next forty to the second row, and so on. The value of an entry in the Pattern Name Table indicates which of the 256 text patterns is to be placed at that spot on the Pattern plane. The Pattern Name Table is located in a contiguous 960-byte block in VRAM beginning on a 1 kilobyte boundary. The starting address of the name table is determined by the 4-bit Name Table base address field in VDP register 2. The base address forms the upper 4 bits of the 14-bit VRAM address. The lower 10 bits of the VRAM address point to one of 960 pattern cells. The name table is organized by rows. An example of Pattern Name Table addressing is given in Section 3. Each byte entry in the name table is the pointer to a pattern definition in the Pattern Generator Table. The same two colors are used for all 256 patterns; the color codes are stored in VDP register 7.

As its name implies, the Text mode is intended mainly for textual applications, especially those in which the 32 patterns-per-line in Graphics modes is insufficient. The advantage is that eight more patterns can be fitted onto one line; the disadvantages are that sprites cannot be used, and only two colors are available for the entire screen. With care, the same text pattern set that is used in Text mode can be also used in Graphics I mode. This is done by ensuring that the least-significant 2 bits of all the character patterns are '0'. A switch from Text mode to Pattern mode, then, results in a stretching of the space between characters, and a reduction of the number of characters per line from 40 to 32. As with the Graphics Modes, once a character set has been defined and placed into the Pattern Generator, updating the Pattern Name Table will produce and manipulate textual material on the screen.

The full 8-bit pattern name is used to select one of the 256 pattern definitions in the pattern generator table. The table is a 2048-byte block in VRAM beginning on a 2 kilobyte boundary. The starting address of the table is determined by the generator base address in VDP register 4. The base address forms the 3 most-significant bits of the 14-bit VRAM address for each Pattern Generator Table entry. The next 8 bits are equal to the 8-bit name of the selected pattern definition. The lowest 3 bits of the VRAM address are equal to the row number within the pattern definition.

Eight bytes are required for each of the 256 possible unique 6 X 8 pattern definitions. The first byte defines the first row of the pattern, and the second byte defines the second row. The two least-significant bits in each byte are not used. It is, however, strongly recommended that these bits be '0's. Each bit entry in the pattern definition selects one of the two colors for that pattern. A '1' bit selects the color code (color 1) contained in the most-significant 4 bits of VDP register 7. A '0' bit selects the other color code (color 0) which is in the least-significant 4 bits of the same VDP Register. An example of pattern definition mapping is provided in Figure 2-16.

A total of 3005 VRAM bytes are required for the Pattern Name and Generator Tables. Less memory is required if all 256 possible pattern definitions are not required; the tables can be overlapped to reduce the amount of VRAM needed for pattern generation. Examples of VRAM memory allocation are provided in Appendix B.

2.7.5 Sprites

The video display can have up to 32 sprites on the highest priority video planes. The sprites are special animation patterns which provide smooth motion and multilevel pattern overlaying. The location of a sprite is defined by the top left-hand corner of the sprite pattern. The sprite can be easily moved pixel-by-pixel by redefining the sprite origin. This provides a simple but powerful method of quickly and smoothly moving special patterns. The sprites are not active in the Text mode. The 32 Sprite Planes are fully transparent outside of the sprite itself.

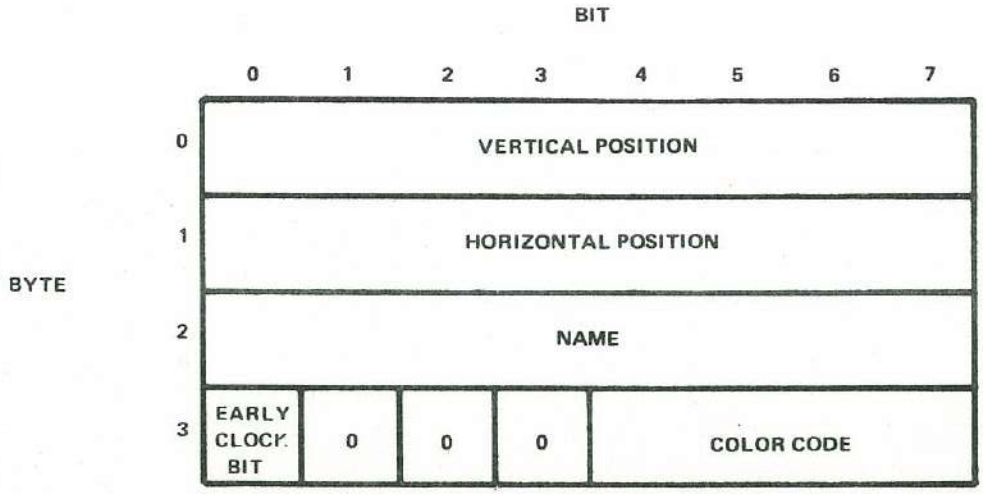


FIGURE 2-17 – SPRITE ATTRIBUTE TABLE ENTRY

The sub-blocks in VRAM that define sprites are the Sprite Attribute Table (see example of entry in Figure 2-17) and the Sprite Generator Table. These tables are similar to their equivalents in the pattern realm in that the Sprite Attribute Table specifies where the sprite goes on the screen, while the Sprite Generator Table describes what the sprite looks like. Sprite Pattern formats are given in Table 5.

Figure 2-18 illustrates the manner in which the VRAM tables map into the existence of sprites on the display. Since there are 32 sprites available for display, there are 32 entries in the Sprite Attribute Table. Each entry consists of four bytes. The entries are ordered so that the first entry corresponds to the sprite on the sprite 0 plane, the next to the sprite on the sprite 1 plane, and so on. The Sprite Attribute Table is 4*32 = 128 bytes long. The Sprite Attribute Table is located in a contiguous 128-byte block in VRAM beginning on a 128-byte boundary. The starting address of the Attribute Table is determined by the 7-bit Sprite Attribute Table base address in VDP register 5. The base address forms the upper seven bits of the 14-bit VRAM address. The next 5 bits of the VRAM address are equal to the sprite number. The lowest 2 bits select one of the four bytes in the Attribute Table entry for each sprite. Each Sprite Attribute Table entry contains four bytes which specify the sprite position, sprite pattern name, and color as shown in Figure 2-17.

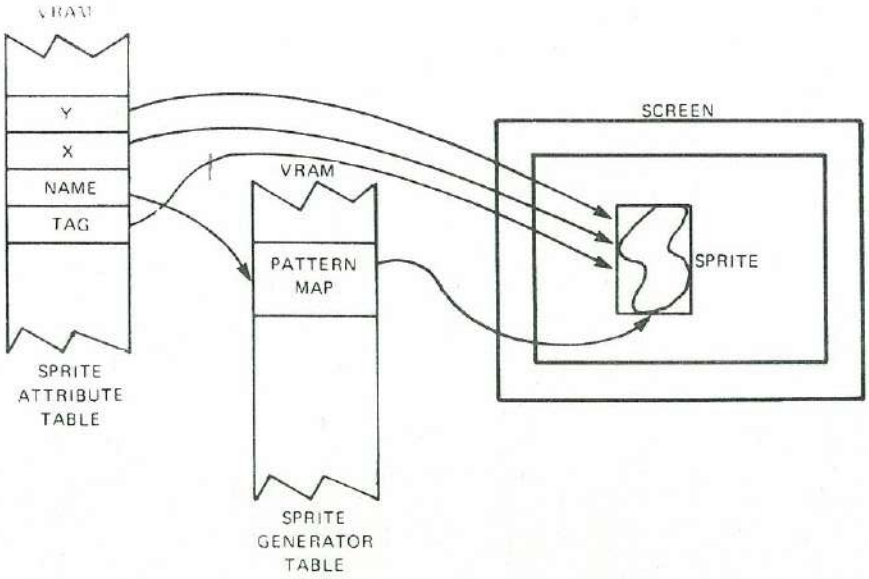


FIGURE 2-18 – SPRITE MAPPING

TABLE 5. SPRITE PATTERN FORMATS

SIZE	MAG	AREA	RESOLUTION	BYTES/PATTERN
0	0	8 × 8	single pixel	8
1	0	16 × 16	single pixel	32
0	1	16 × 16	2 × 2 pixels	8
1	1	32 × 32	2 × 2 pixels	32

The first two bytes of each entry of the Sprite Attribute Table determine the position of the sprite on the display. The first byte indicates the vertical distance of the sprite from the top of the screen, in pixels. It is defined such that a value of −1 puts the sprite butted up at the top of the screen, touching the backdrop area. The second bytes describes the horizontal displacement of the sprite from the left edge of the display. A value of 0 butts the sprite up against the left edge of the backdrop. Note that it is from the upper left pixel of the sprite that all measurement are taken.

When the first two bytes of an entry position a sprite overlapping the backdrop, the part of the sprite that is within the backdrop is displayed normally. The part of the sprite that overlaps the backdrop is hidden from view by the backdrop. This allows the animator to move a sprite into the display from behind the backdrop. The displacement in the first byte is partially signed, in that values for vertical displacement between –31 and 0 ($E1_{16}$ to 0) allow a sprite to "bleed in" from the top edge of the backdrop. Likewise, values in the range of 207 to 191 allow the sprite to bleed in from the bottom edge of the backdrop. Similarly, horizontal displacement values in the vicinity of 255 allow a sprite to bleed-in from the right side of the screen. To allow sprites to bleed-in from the left edge of the backdrop, a special bit in the third byte of the Sprite Attribute Table entry is used, as described in a later paragraph.

Byte 3 of the Sprite Attribute Table entry contains the pointer to the Sprite Generator Table that specifies what the sprite should look like. This is an 8-bit pointer to the sprite patterns definition, the Sprite Generator Table. The sprite name is similar to that in the Patterns Graphics mode.

Byte 4 of the Sprite Attribute Table entry contains the color of the sprite in its lower 4 bits (see Table 2 for color codes). The most-significant bit is the Early Clock bit (EC). This bit, when set to a '0', does nothing. When set to '1', the horizontal poisition of the sprite is shifted to the left by 32 pixels. This allows a sprite to bleed-in from the left edge of the backdrop. Values for horizontal displacement (byte 2 in the entry) in the range 0 to 32 cause the sprite to overlap with the left-hand border of the backdrop.

The Sprite Generator Table is a maximum of 2048 bytes long beginning on the 2 kilobyte boundaries. It is arranged into 256 blocks of 8 bytes each. The third byte of the Sprite Attribute Table entry, then, specifies which eight byte block to use to specify a sprite's shape. The '1's in the Sprite Generator cause the sprite to be defined at that point; '0's cause the transparent color to be used. The starting address of the table is determined by the sprite generator base address in VDP register 6. The base address forms the 3 most-significant bits of the 14-bit VRAM address. The next 8 bits of the address are equal to sprite name, and the last 3 bits are equal to the row number within the sprite pattern. The address formation is slightly modified for $SIZE_1$ sprites.

There is a maximum limit of four sprites that can be displayed on one horizontal line. If this rule is violated, the four highest-priority sprites on the line are displayed normally. The fifth and subsequent sprites are not displayed on that line. Furthermore, the fifth-sprite bit in the VDP status register is set to a '1', and the number of the violating fifth sprite is loaded into the status register (see Section 2.5).

Larger sprites than 8 X 8 pixels can be used if desired. The MAG and SIZE bits in VDP register 1 are used to select the various options. The options are described here:

MAG = 0, SIZE = 0: No options chosen;

MAG = 1, SIZE = 0: Eight bytes are still used in the Sprite Generator Table to describe the sprite; however, each bit in the Sprite Generator maps into 2 X 2 pixels on the TV screen, effectively doubling the size of the sprite to 16 X 16.

MAG = 0, SIZE = 1: 31 bytes are used in the Sprite Generator Table to define the sprite shape; the result is a 16 X 16 pixel sprite. The mapping of the 32 bytes into the sprite image is as shown in Figure 2-19. Mapping is still one bit-to-one pixel.

MAG = 1, SIZE = 1: Same as MAG = 0, SIZE = 1 except each bit now maps into a 2 X 2 pixel area, yielding a 32 X 32 sprite.

The VDP provides sprite coincidence checking. The coincidence status flag in the VDP status register is set to a '1' whenever two active sprites have '1' bits at the same screen location.

Sprite processing is terminated if the VDP finds a value of 208 ($D0_{16}$) in the vertical position field of any entry in the Sprite Attribute Table. This permits the Sprite Attribute Table to be shortened to the minimum size required; it also permits the user to blank out part or all of the sprites by simply changing one byte in VRAM.

A total of 2176 VRAM bytes are required for the Sprite Name and Pattern Generator Tables. Significantly less memory is required if all 256 possible sprite pattern definitions are not required. The Sprite Attribute Table can also be shortened as described above. The tables can be overlapped to reduce the amount of VRAM required for sprite generation. Examples of VRAM memory allocation are provided in Section 3.

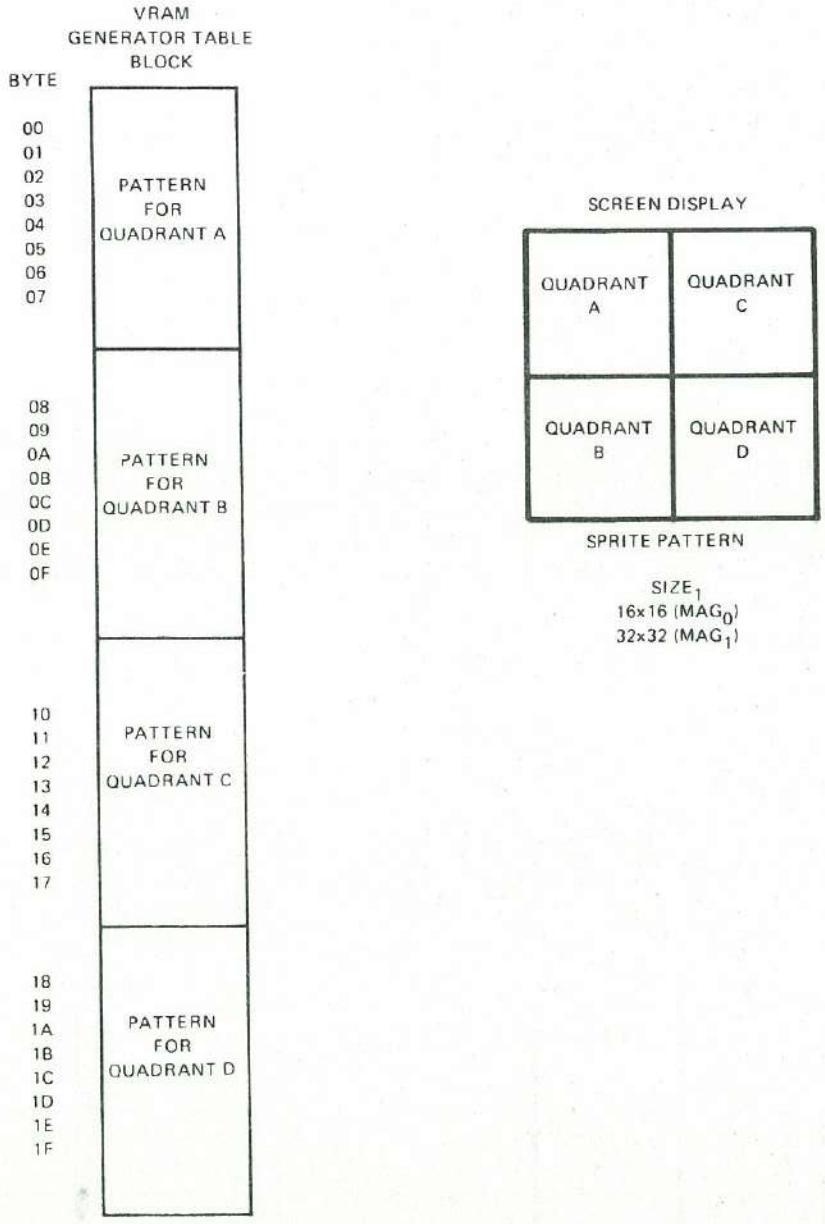


FIGURE 2-19 – SIZE 1 SPRITE MAPPING

2.8 EXTERNAL VIDEO

The external video interface allows mixing an external video source and VDP generated video under software control. As shown in Figure 2-20, composite video signals that are input to the 9918A through the external video input pin appear on the television screen in the plane behind the backdrop when the transparent color is programmed. Thus VDP-generated images on the Pattern Plane and the Sprite Planes can be superimposed upon an incoming signal. The source of the signal may be a standard NTSC broadcast signal, the output from an NTSC-compatible video-tape recorder, another VDP chip output, or any other NTSC-compatible composite video signal.

2.8.1 Hardware Design for External Video

The interface for external video operation consists of an external composite video signal on the EXTVID pin, a composite sync signal derived from the external composite video signal on the $\overline{\text{RESET}}$/SYNC pin, and a method of synchronizing the external color subcarrier with the VDP's colors. The interface is designed so that the external video source provides the sync, blanking, and color burst to the television or monitor. The VDP controls the visible portion of the frame by gating in the external video signal when the transparent color is programmed on all planes. This preserves the colors of the external video picture by causing any inaccuracy in the external color lock circuit to modify the VDP colors but not, for example, the flesh tones of an external image.

The external video signal must be biased correctly and be of the proper amplitude so that the luminance levels of the external and VDP colors are matched and the external video does not bleed through into the composite video output of the VDP. The internal circuit assures that a perfect match results if the external video is of the same amplitude as the VDP's composite video and its dc level is increased by a MOS threshold voltage (typically 0.7 volts). This adjustment may be varied to change the relative luminance levels of the two video signals and thus modify the picture appearance.

The composite sync signal must also be of the proper levels. The $\overline{\text{RESET}}$/SYNC pin is a tri-level input, utilizing 0, 5, and 12 volts. The 0-volt level activates reset of the VDP as described in section 2.1.7. The 5-volt level is the inactive state. The 12-volt level is the sync level. The input composite sync should thus swing from 5 volts to 12 volts. The timing of the composite sync controls the VDP in the following manner. A positive going edge to the $\overline{\text{RESET}}$/SYNC pin is recognized as a horizontal sync pulse and resets the internal horizontal counter into the horizontal sync state. A sync pulse, which lasts for greater than 7.2 microseconds, is recognized as a vertical sync pulse and resets the internal vertical counter to the vertical sync state. Thus the VDP will be interlaced or not interlaced depending upon the composite sync signal.

To lock the colors of the external video with the VDP colors, a phase-lock loop maintains the VDP's input clock frequency at three times the external video subcarrier frequency. CPUCLK of the VDP is used as the feedback to maintain the color lock. See Figure 2-20 for a block diagram of the external video interface.

The input sync and phase-locked loop are not required when the external video source is another VDP. By running the VDP's from the same crystal or clock input and resetting the VDP's together with a fast edge (rise/fall time less than 30 ns), the VDP's will be locked on an open loop basis. See Figure 2-21 for the VDP to VDP interface.

There is a limitation in the resoltuion of the VDP-generated images when the external video signal is not from another VDP. The VDP can resolve only ±1 pixel (186 nanoseconds) and, for a true NTSC-timed composite video signal, will cause a small sawtooth edge to surround vertical edges of the VDP-generated image. This results because the relationship of NTSC color subcarrier (2/3 of the VDP pixel rate) to the NTSC horizontal scan rate results in 341¼ pixels/line. Since the VDP can't resolve a ¼ pixel, a sawtooth edge is created by the timing. This sawtooth edge may be eliminated with loss of color by reducing the VDP input frequency to an integer multiple of the external horizontal sync. The phased-locked loop should lock the sync to the VDP input frequency. Thus the color lock is replaced with a sync lock. Color is no longer available since it is not within the subcarrier frequency.

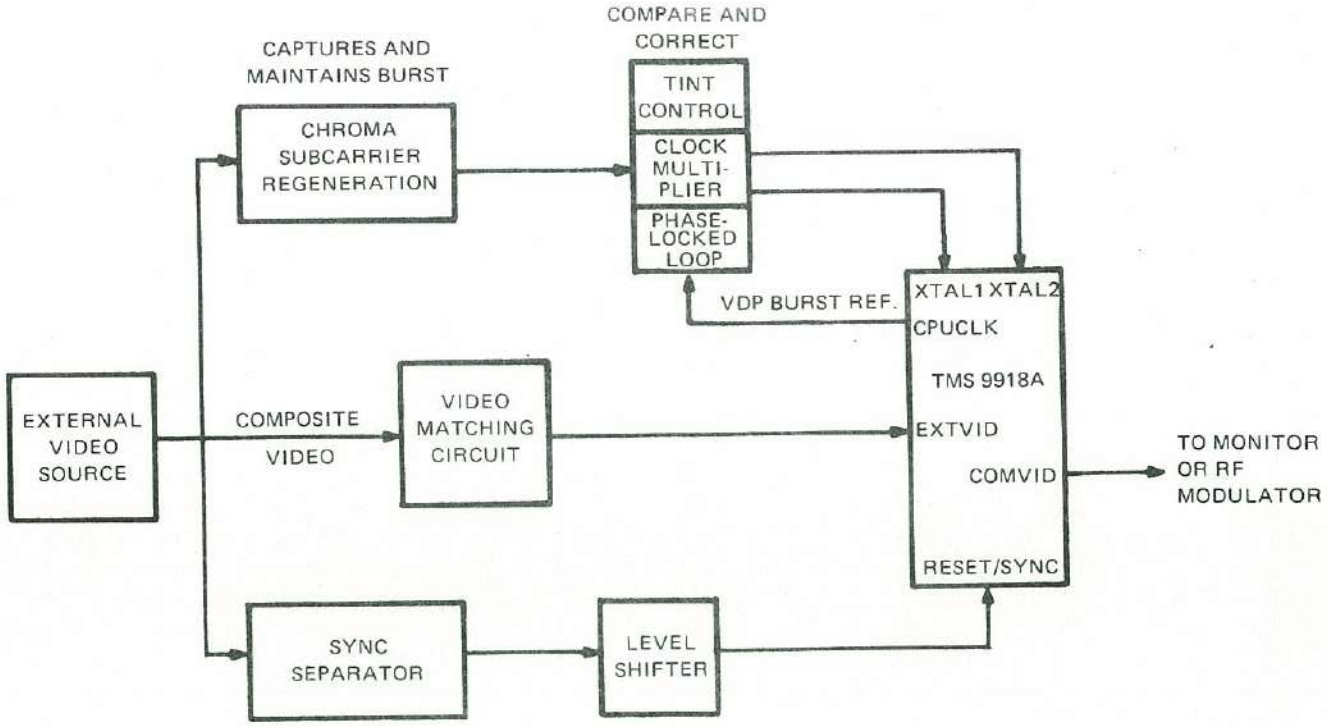


FIGURE 2-20 – EXTERNAL VIDEO INTERFACE BLOCK DIAGRAM

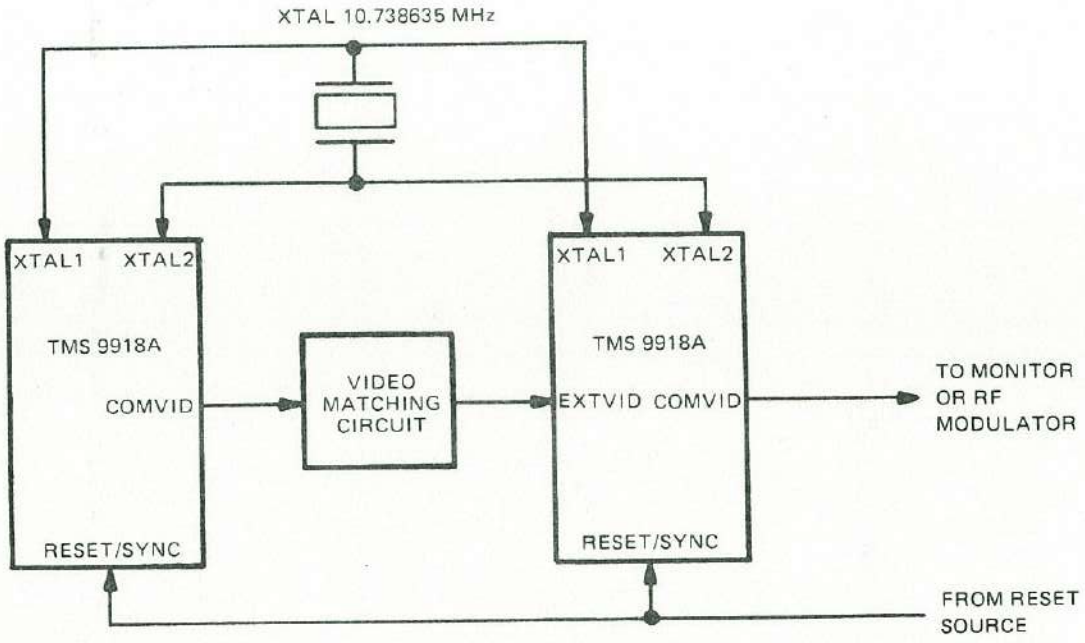


FIGURE 2-21 – CASCADING TWO TMS 9918A VDP'S

2.8.2 Software Requirements for External Video

For the External Video input plane to be visible, the External Video Enable bit in VDP Register 0 (EXVID) should be set to a '1'. The backdrop color (VDP Register 7, lower 4 bits) should be set to transparent (0). For the External Video plane to show through at a given spot on the screen, the Pattern color at that spot should be transparent, and all sprites should not be in the way (alternatively, a sprite that was in the way could be made transparent in color). Note that the External Video feature can be used in Graphics I, Graphics II, Text or Multicolor mode.

2.9 VRAM ADDRESS DERIVATION

Table summarizes the VRAM address derivation for all modes of operation for the TMS 9918A. Section 3 contains examples of how typical VRAM addresses are computed by the VDP.

TABLE 6 – PATTERN GRAPHICS ADDRESS LOCATION TABLE

GRAPHICS I MODE ADDRESS LOCATION TABLE

ADDRESS TYPE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	COMMENTS
1) PATTERN NAME ADDRESS	NTB				ROW					COLUMN					PATTERN NAME TABLE BASE (VDP REG2) PATTERN POSITION
2) PATTERN COLOR ADDRESS	COLB								0	NAME (0-4)					PATTERN COLOR TABLE BASE (VDP REG3) ALWAYS "0" IN BIT 8 FIVE MOST SIGNIFICANT BITS OF NAME
3) PATTERN GENERATOR ADDRESS	PGB					NAME						XXX			PATTERN GENERATOR BASE (VDP REG4) ALL 8 BITS OF NAME THREE LSB'S FORM PATTERN ROW POSITION

GRAPHICS II MODE ADDRESS LOCATION TABLE

ADDRESS TYPE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	COMMENTS
1) PATTERN NAME ADDRESS	NTB					ROW					COLUMN				PATTERN NAME TABLE BASE (VDP REG2) PATTERN POSITION ROW PATTERN POSITION COLUMN
2) PATTERN COLOR ADDRESS		XX					NAME					XXX			PATTERN COLOR TABLE BASE MSB (VDP REG3) TWO MSB FROM VERTICAL COUNTER ALL 8 BITS OF NAME COLOR TABLE BYTE/LINE
3) PATTERN GENERATOR ADDRESS		XX					NAME					XXX			PATTERN NAME TABLE BASE MSB (VDP REG4) TWO MSB FROM VERTICAL COUNTER ALL 8 BITS OF NAME PATTERN GENERATOR BYTE/LINE NUMBER

TEXT MODE ADDRESS LOCATION TABLE

ADDRESS TYPE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	COMMENTS
TEXT MODE NAME ADDRESS	NTB					TEXT POSITION									PATTERN NAME TABLE BASE (VDP REG2) EQUAL (TEXT POSITION ROW * TIMES 40) PLUS (TEXT POSITION COLUMN NUMBER)
TEXT MODE PATTERN ADDRESS	PGB						NAME					XXX			PATTERN GENERATOR BASE (VDP REG4) NAME BYTE/LINE NUMBER

SPRITE ADDRESS LOCATION TABLE

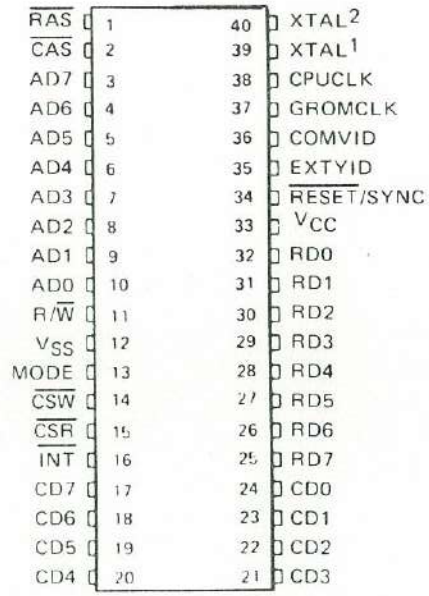
ADDRESS TYPE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	COMMENTS
SPRITE ATTRIBUTE ADDRESS	SAB														SPRITE ATTRIBUTE TABLE BASE (VDP REG5) SPRITE NUMBER ATTRIBUTE NUMBER: 00 FOR VERTICAL POSITION 01 FOR HORIZONTAL POSITION 10 FOR NAME 11 FOR TAG (EARLY CLOCK AND COLOR)
								SPRITE							
													XX		
SIZE = 0 SPRITE PATTERN GENERATOR	SPGB														SPRITE PATTERN GENERATOR BASE (VDP REG4) NAME ATTRIBUTE OF SPRITE THREE LSB'S GIVE BYTE/LINE NUMBER
								NAME							
													XXX		
SIZE = 1 SPRITE PATTERN GENERATOR	SPGB														SPRITE PATTERN GENERATOR BASE (VDP REG4) SIX MSB OF NAME SIZE = 1 SPRITE BYTE NUMBER (SEE FIGURE 19)
							NAME (0-5)								
												XXXXX			

MULTICOLOR ADDRESS LOCATION TABLE

ADDRESS TYPE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	COMMENTS
4) MULTICOLOR NAME ADDRESS	NTB														NAME TABLE BASE (VDP REG2) PATTERN POSITION ROW PATTERN POSITION COLUMN
								ROW							
												COLUMN			
5) MULTICOLOR COLOR GENERATOR ADDRESS	PGB														PATTERN GENERATOR BASE (VDP REG4) NAME FROM NAME FETCH THREE LSB'S FORM BYTE/SQUARE ROW
								NAME							
												XXX			

2.10 VDP TERMINAL ASSIGNMENTS

SIGNATURE	TERMINAL	I/O	DESCRIPTION
CD0 MSB	24	I/O	CPU data bus (CD0) is the most significant
CD1	23	I/O	bit
CD2	22	I/O	
CD3	21	I/O	
CD4	20	I/O	
CD5	19	I/O	
CD6	18	I/O	
CD7	17	I/O	
MODE	13	I	CPU interface mode select; usually a processor address line
$\overline{CSR}$	15	I	CPU-VDP read strobe
$\overline{CSW}$	14	I	CPU-VDP write strobe
V_{CC}	33	I	+5 volt supply
V_{SS}	12	I	Ground Reference
RD0 MSB	32	I	VRAM read data bus (RD0 is the most significant bit)
RD1	31	I	
RD2	30	I	
RD3	29	I	
RD4	28	I	
RD5	27	I	
RD6	26	I	
RD7	25	I	
AD0 MSB	10	O	VRAM address/data bus (multiplexed high and low order VRAM address and output data bytes)
AD1	9	O	AD0 is the most significant bit and is used only for data and not for addressing.*
AD2	8	O	
AD3	7	O	
AD4	6	O	
AD5	5	O	
AD6	4	O	
AD7	3	O	
$\overline{RAS}$	1	O	VRAM row address strobe
$\overline{CAS}$	2	O	VRAM column address strobe
R/$\overline{W}$	11	O	VRAM write strobe
XTAL1, XTAL2	40,39	I	10.7 + MHz crystal inputs.**
GROMCLK	37	O	VDP output clock = XTAL/24. Typically not used
$\overline{RESET}$/SYNC	34	I	$\overline{RESET}$—This pin is a trilevel input pin. When it is below 0.8 volts, $\overline{RESET}$ initializes the VDP. When it is above 9 volts, $\overline{RESET}$ is the synchronizing input for external video.



SIGNATURE	TERMINAL	I/O	DESCRIPTION
EXTVID	35	I	External video input
CPUCLK	38	O	NTSC color burst frequency clock. Typically not used.
INT	16	O	CPU interrupt output.
COMVID	36	O	NTSC composite video output.

* The least-significant address bit, AD7, is wired to A0 of the dynamic RAMs. Likewise, AD6 is wired to A1 of the RAMs. Care must be exercised in assuring proper orientation of the 9918A address outputs to the dynamic RAM address inputs.

** When driven externally, both inputs must be driven.

4. TMS 9918A PRELIMINARY ELECTRICAL SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (unless otherwise noted)*

Supply voltage, V_{CC}	−0.3 to 20 V
All input voltages	−0.3 to 20 V
Output voltage	−2 to 7 V
Continuous power dissipation	1.8 W
Operating free-air temperature range	0°C to 55°C
Storage temperature range	−55°C to +150°C

* Stresses beyond 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 conditions beyond those indicated in the "Recommended Operating Condtions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

4.2 RECOMMENDED OPERATING CONDTIONS*

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.75		5.25	V
Supply voltage, V_{SS}			0		V
Input voltage, V_I, $\overline{\text{RESET}}$/SYNC pin	SYNC active	10		12	V
	RESET active			0.6	V
	SYNC and RESET inactive	3		6	V
High-level input voltage, V_{IH}	XTAL1, XTAL2	2.75			V
	All other inputs	2.2			V
Input voltage, V_I, EXTVID pin	SYNC level				V
	White level				V
	Black level				
Low-level input voltage, V_{IL}				0.8	V
Operating free-air temperature, T_A		0		55	°C

* All voltage values are with respect to V_{SS}.

4.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGES OF RECOMMENDED OPERATING CONDITIONS (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	$\overline{RAS}$, $\overline{CAS}$, R/$\overline{W}$	I_{OH} = −400 μA		2.7			V
		All other outputs			2.4			
V_{OL}	Low-level output voltage	CPU data	I_{OL} = 1.2 mA				0.6	V
		DRAM interface	I_{OL} = 800 μA				0.6	
I_{OZH}	Off-state output current high-level voltage applied, D0-D7 outputs		V_O = 5.5 V				100	μA
I_{OZL}	Off-state output current low-level voltage applied, D0-D7 outputs		V_O = 0 V				−100	μA
I_{IH}	High-level input current		V_I = 5.5 V,	All other pins at 0 V			10	μA
I_{IL}	Low-level input current		V_I = 0 V,	All other pins at 0 V			−10	μA
	Video voltage level of white, COMVID output					3.2		V
	Video voltage level of black (blank), COMVID output					2.3		V
	Video voltage level of sync, COMVID output					1.9		V
	Video voltage (peak-to-peak) of burst, COMVID output					0.5		V
	Video voltage difference, white − black, COMVID output				0.5			V
I_{CC}	Average supply current from V_{CC}		T_A = 25°C			200	250	mA
C_i	Input capacitance	D0-D7	f = 1 MHz,	Unmeasured pins at 0 V			20	pF
		All other inputs					10	
C_o	Output capacitance		f = 1 MHz,	Unmeasured pins at 0 V			20	pF

†All typical values are at V_{CC} = 5.25 V, T_A = 25°C.

4.4 TIMING REQUIREMENTS OVER FULL RANGES OF RECOMMENDED OPERATING CONDITIONS

CPU-VDP interface

PARAMETER		MIN	NOM	MAX	UNIT
$t_{su(ARL)}$	Address setup time before CSR low		0		ns
$t_{su(AWL)}$	Address setup time before CSW low		30		ns
$t_{h(WLA)}$	Address hold time after CSW low		30		ns
$t_{su(DWH)}$	Data setup time before CSW high		100		ns
$t_{h(WHD)}$	Data hold time after CSW high		30		ns
$t_{w(WL)}$	Pulse width, CSW low		200		ns
$t_{w(CSH1)}$	Pulse width, chip select high (requesting memory access)		8		μs
$t_{w(CSH2)}$	Pulse width, chip select high (not requesting memory access)		3		μs

VDP-VRAM interface

PARAMETER		MIN	NOM	MAX	UNIT
t_c	Memory read or write cycle time	372			ns
$t_{su(DCH)}$	Input data setup time before CAS high	160			ns
$t_{h(CHD)}$	Input data hold time after CAS high	0			ns

external clock source

PARAMETER		MIN	TYP	MAX	UNIT
f_{ext}	External source frequency		10.738		MHz
t_r/T_f	External source rise/fall time		10		ns
t_{WH}	External source high level pulse width	42	47	52	ns
t_{WL}	External source low level pulse width	42	47	52	ns
t_{PD}	External source phase delay from XTAL1 falling edge to XTAL2 falling edge	42	47	52	ns

4.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS

CPU-VDP interface

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{a(CSR)}$	Data access time from $\overline{CSR}$ low	C_L = 300 pF		300		ns
t_{PVX}	Data disable time after $\overline{CSR}$ high			200		ns
$t_{PVX,A}$	Data invalid time from address changes			0		ns
f_{CPUCLK}	CPU clock output frequency ($f_{ext} \div 3$)			3.58		MHz
$f_{GROMCLK}$	GROM clock output frequency (ext ÷ 24)			447.5		kHz

VDP-VRAM interface

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{w(CH)}$	Pulse width, $\overline{CAS}$ high	C_L = 50 pF	60			ns
$t_{w(CL)}$	Pulse width, $\overline{CAS}$ low		100		10,000	ns
$t_{w(RH)}$	Pulse width, $\overline{RAS}$ high		100			ns
$t_{w(RL)}$	Pulse width, $\overline{RAS}$ low		150		10,000	ns
$t_{w(W)}$	Pulse width, write pulse		45			ns
t_{CA-CL}	Delay time, column address to $\overline{CAS}$ low		−10			ns
t_{RA-RL}	Delay time, row address to $\overline{RAS}$ low		0			ns
t_{d-WL}	Delay time, data to $\overline{W}$ low		0			ns
t_{WH-CL}	Delay time, R/$\overline{W}$ high to $\overline{CAS}$ low		0			ns
t_{W-CH}	Delay time, R/$\overline{W}$ low to $\overline{CAS}$ high		60			ns
t_{W-RH}	Delay time, R/$\overline{W}$ low to $\overline{RAS}$ high		60			ns
t_{CL-CA}	Column address valid after $\overline{CAS}$ low		45			ns
t_{RL-RA}	Row address valid after $\overline{RAS}$ low		20			ns
t_{RL-CA}	Column address valid after $\overline{RAS}$ low		95			ns
t_{CL-D}	Data valid after $\overline{CAS}$ low		45			ns
t_{RL-D}	Data valid after $\overline{RAS}$ low		95			ns
t_{WL-D}	Data valid after R/$\overline{W}$ low		45			ns
t_{CH-WL}	Read command valid after $\overline{CAS}$ high		0			ns
t_{CL-W}	Write command valid after $\overline{CAS}$ low		45			ns
t_{CH-RL}	Delay time, $\overline{CAS}$ high to $\overline{RAS}$ low		−20			ns
t_{CL-RH}	Delay time, $\overline{CAS}$ low to $\overline{RAS}$ high		100			ns
t_{RL-CL}	Delay time, $\overline{RAS}$ low to $\overline{CAS}$ low		20		50	ns

composite video output

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{xsad}	SYNC to active display			11.2		μs
t_{xhsw}	Horizontal SYNC width		1		5	μs
t_{xvsw}	Vertical SYNC width		7.8		50	μs

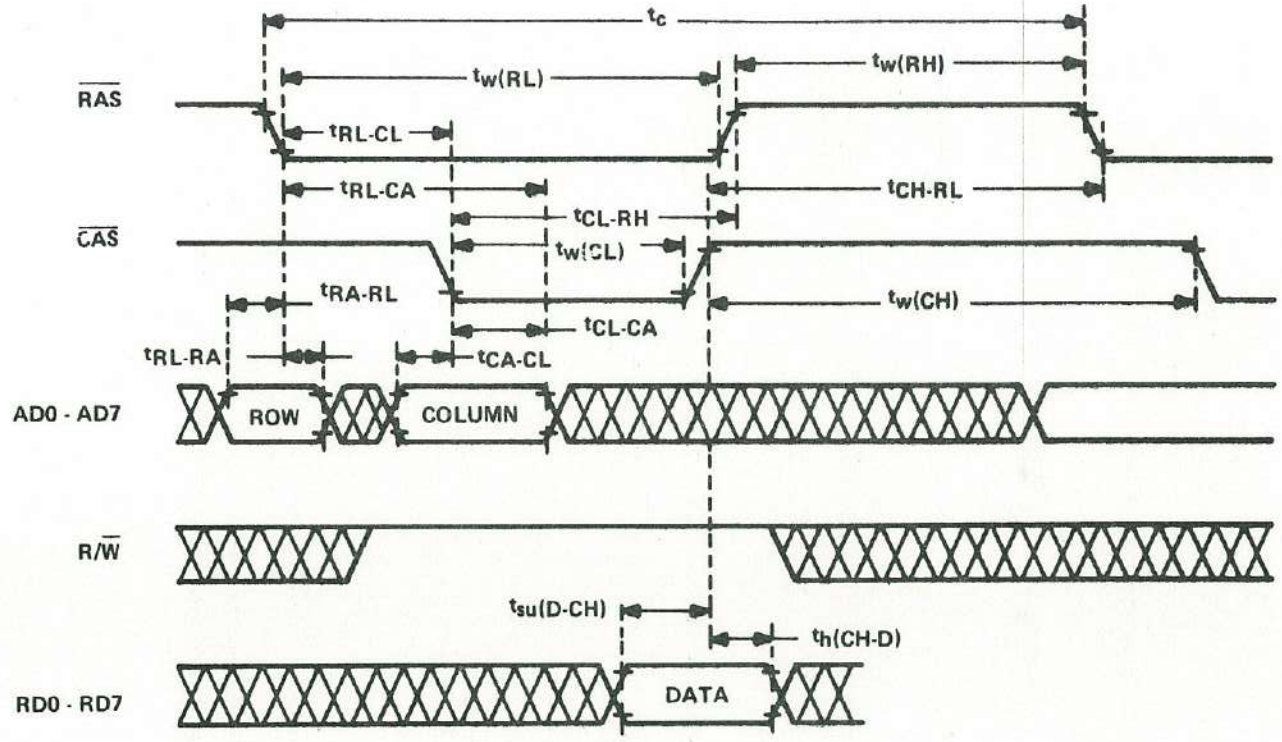


FIGURE 4-1 – VRAM READ CYCLE

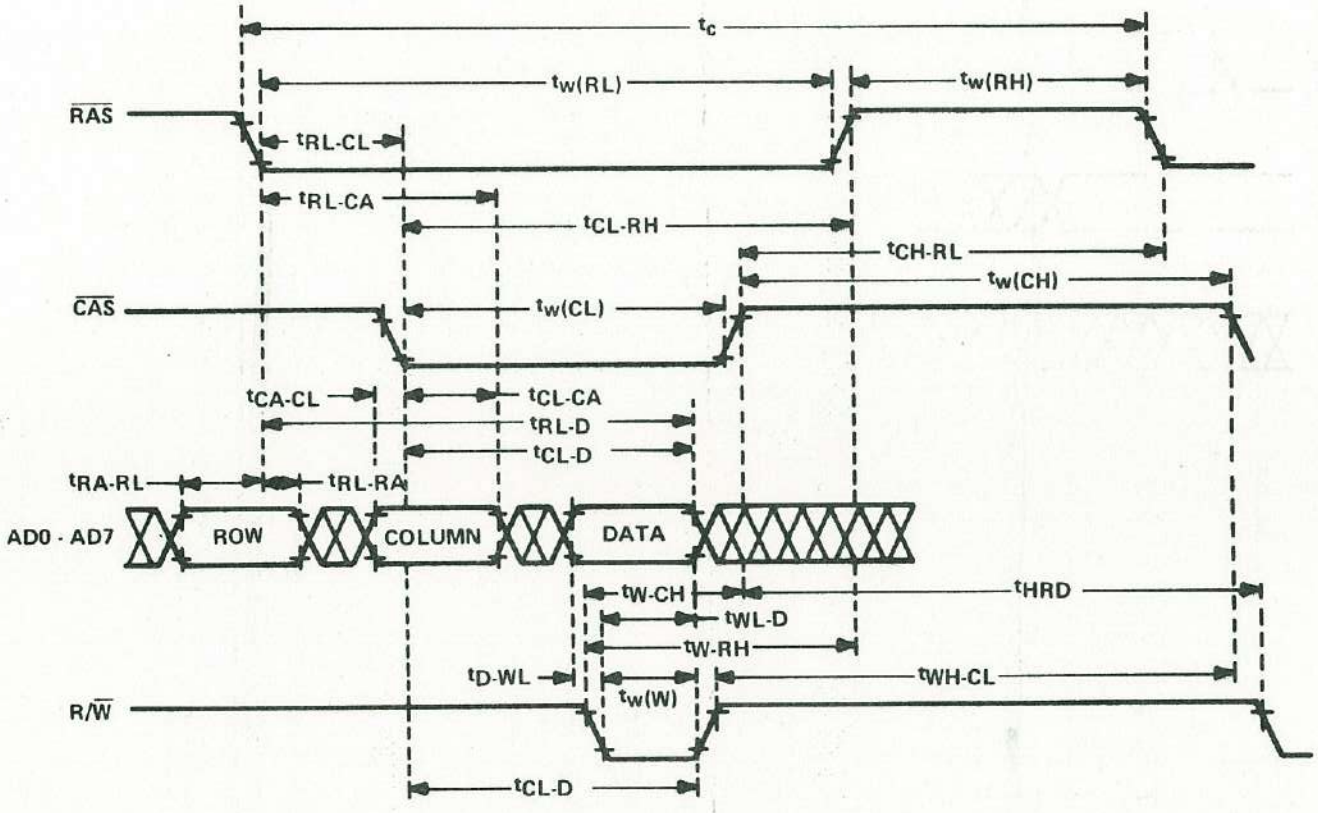
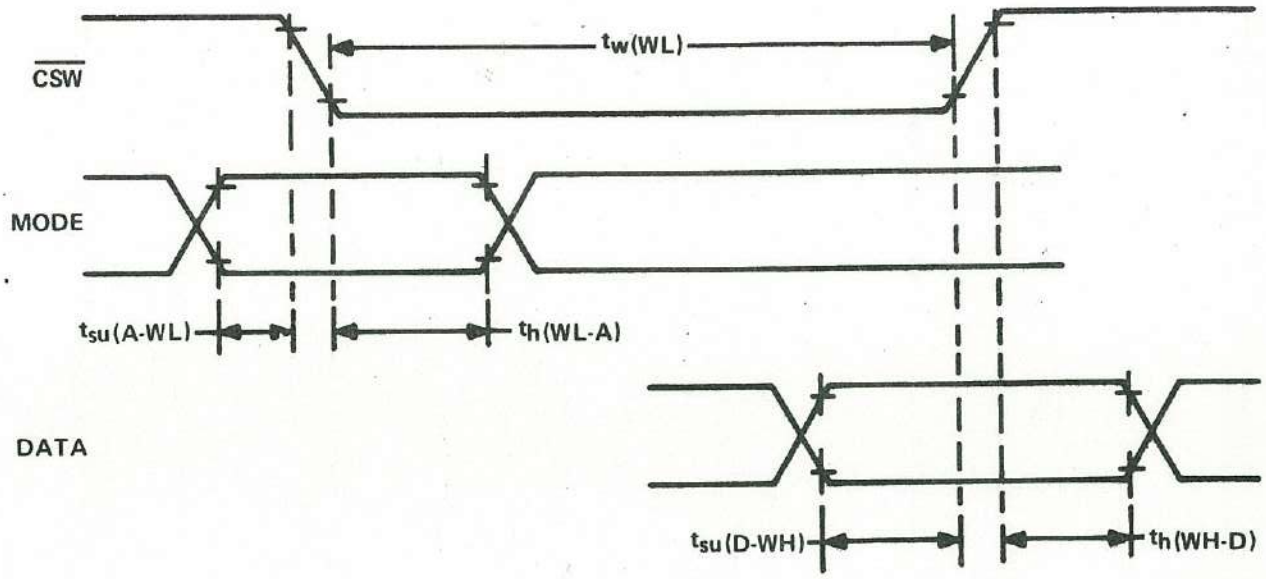
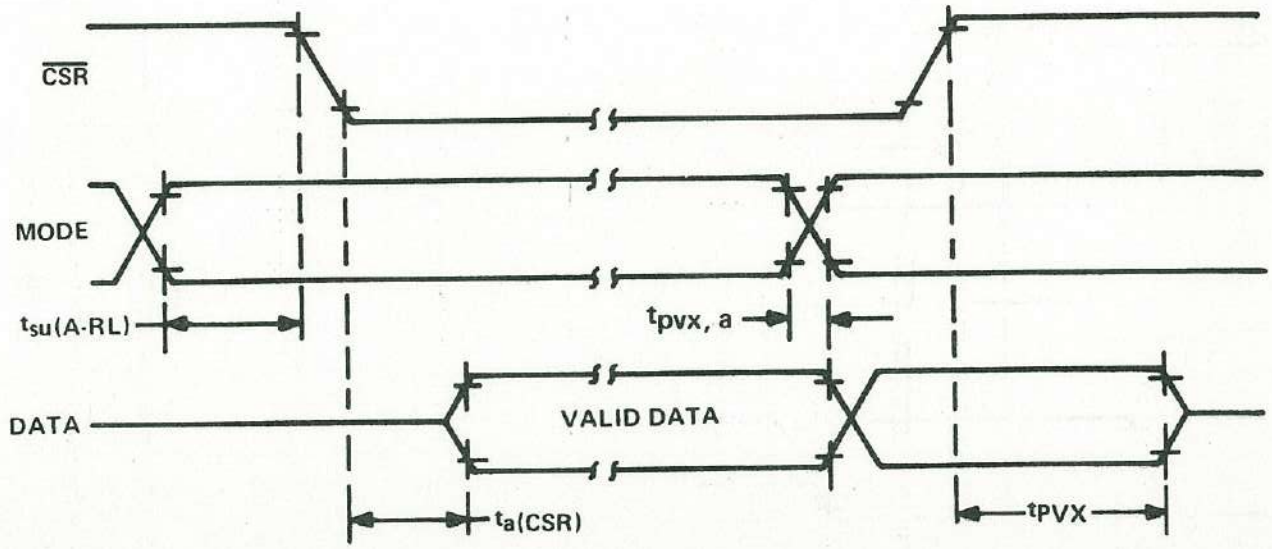


FIGURE 4-2 – VRAM WRITE CYCLE

WRITE CYCLE



READ CYCLE



NOTE: All measurements are made at 10% and 90% points.

FIGURE 4-3 – CPU-VDP WRITE CYCLE

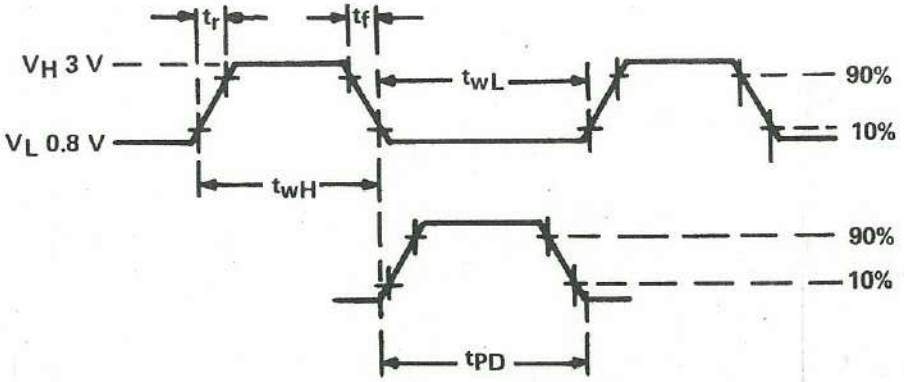
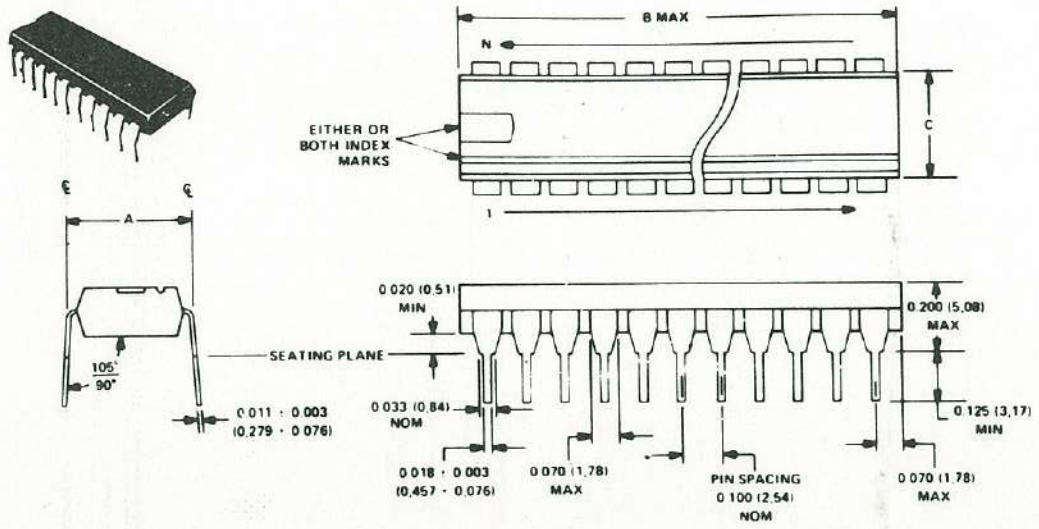


FIGURE 4-4 – EXTERNAL CLOCK TIMING WAVEFORM

5. MECHANICAL DATA

5.1 TMS 9918 — 40-PIN PLASTIC PACKAGE

plastic packages

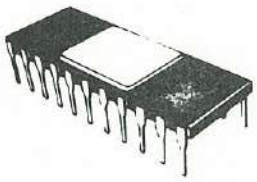


ALL LINEAR DIMENSIONS ARE IN INCHES AND PARENTHETICALLY IN MILLIMETERS. INCH DIMENSIONS GOVERN.

DIM \ PINS	8	16	18	20	22	24	28	40
A ± 0.010 (0,26)	0.300 (7,62)	0.300 (7,62)	0.300 (7,62)	0.300 (7,62)	0.400 (10,16)	0.600 (15,24)	0.600 (15,24)	0.600 (10,24)
B MAX	0.390 (9,9)	0.870 (22,1)	0.920 (23,4)	1.070 (27,2)	1.100 (28,0)	1.290 (32,8)	1.440 (36,6)	2.090 (53,1)
C NOM	0.250 (6,4)	0.250 (6,4)	0.250 (6,4)	0.265 (6,7)	0.350 (8,9)	0.550 (14,0)	0.550 (14,0)	0.550 (14,0)

5.2 TMS 9918 — 40-PIN CERAMIC PACKAGE

ceramic packages with side-brazed leads and metal or epoxy or glass lid seal

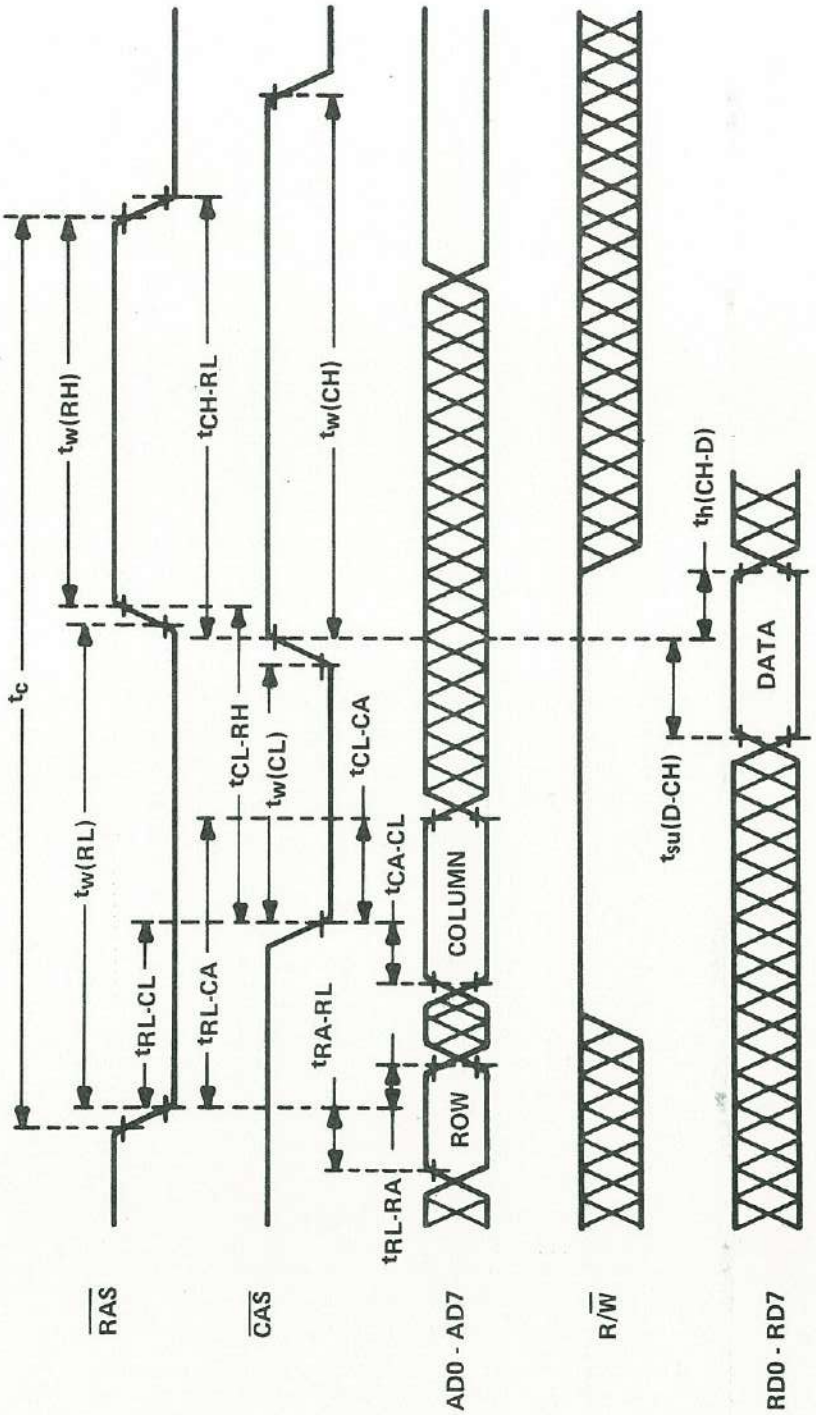


NOTES: a. This minimum spacing is valid for printed circuit board mounting with 0.033 (0,84) diameter holes for the leads.
b. All linear dimensions are in inches and parenthetically in millimeters. Inch dimensions govern.
c. The index is placed in this area to identify pin 1 and to provide other information as follows:

1	Pin 1 connected to chip-mounting pad.
△XX	Pin XX connected to chip-mounting pad.
•	No connection to chip-mounting pad.

Other symbols may indicate any combination of up to 4 pins connected to the chip-mounting pad.

DIM \ PINS	16	18	20	22	24	28	40
A ± 0.010 (0,26)	0.300 (7,62)	0.300 (7,62)	0.300 (7,62)	0.400 (10,16)	0.600 (15,24)	0.600 (15,24)	0.600 (15,24)
B MAX	0.840 (21,4)	0.910 (23,1)	1.020 (25,9)	1.100 (28,0)	1.290 (32,8)	1.415 (36,0)	2.020 (51,3)
C NOM	0.290 (7,4)	0.290 (7,4)	0.290 (7,4)	0.390 (9,9)	0.590 (15,0)	0.590 (15,0)	0.590 (15,0)


t_c
$t_{w(RH)}$
$t_{w(RL)}$
t_{RL-CL}
t_{RL-CA}
t_{CH-RL}
t_{CL-RH}
$t_{w(CL)}$
$t_{w(CH)}$
t_{CL-CA}
t_{CA-CL}
t_{RA-RL}
t_{RL-RA}
$t_{su(D-CH)}$
$t_{h(CH-D)}$
ROW
COLUMN
DATA
$\overline{RAS}$
$\overline{CAS}$
AD0 - AD7
R/$\overline{W}$
RD0 - RD7