Data-Flow Processing Element

design manual (1982)

M.W. Rawling and E.A. Zuk

Contents

1. Introduction

This document describes in detail the design of a data-flow processing element based on the Fast Processing Element Structure proposed in refernence 1.

It is not the intention of this design manual to explain the data-flow system in any great detail, but rather the design of a processing element ideally suited to implement that system. Further information on the data-flow concept is available from referrences 1,2,3.

1.1 Structure of the Processing Element

The ideal fast structure proposed in (1) is not really suited to discrete design techniques such as are used in this design. As a result, the structure was altered slightly to that shown in figure 1.1.

As can be seen from figure 1.1, the element has been broken up into three sections; CPU1, CPU2 and four hardware fifo queues. An S100 interface board was also designed and built.

In the original proposed fast element, the units or small blocks shown in figure 1.1 were all specialised hardware devices. In the design, the hardware blocks have been replaced by software operations, except for the fifo queues and the floating point unit.

1.2 Choice of CPU

The CPU must be very fast to replace all of the hardware units. In addition, speed greatly depends upon the rate at which data can be passed within the unit, a wide data path is therefore very desirable.

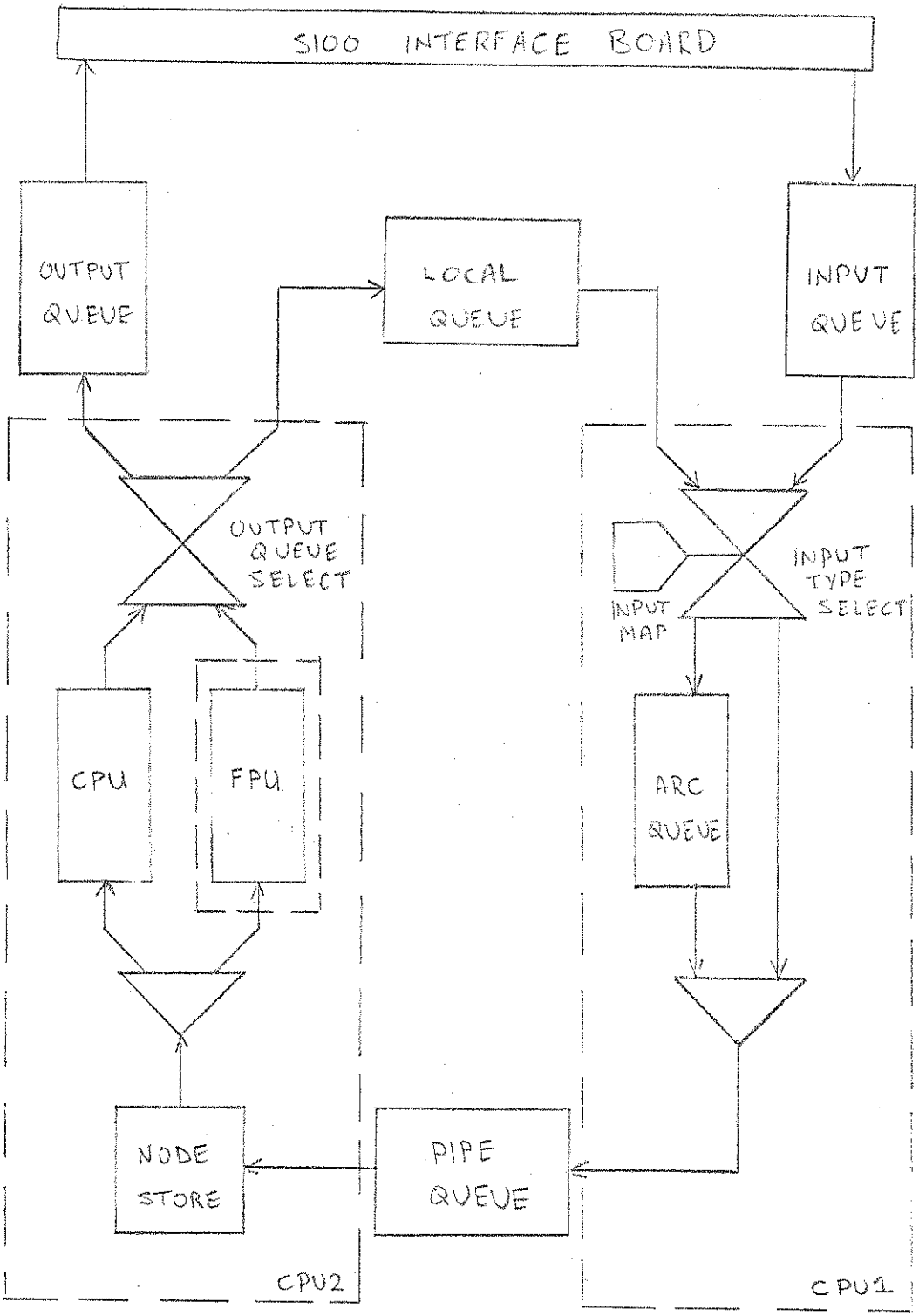


FIG 1.1 Data-Flow Processing Element Structure

The choice is thus quickly narrowed down to one of the modern 16 bit microprocessors. The Motorola MC68000 is without doubt the fastest and most flexible of these. Apart from speed, its most desirable features are probably ease of interfacing (asynchronous bus) and large, linear address range.

1.3 Specifications

The only real specification imposed on the design is that it must implement the data-flow model. However in designing the processing element, every effort has been made to make it as advanced and flexible as possible, with the intentions of using it for the basis of future projects and study into the data-flow concept. As an example, the element is designed to be entirely interrupt driven, however hardware facilities are provided that allow fully polled operation. The unit can handle floating point data also using an Am9511 arithmetic processing unit. On board ram is directly expandable to 512 Mbytes.

1.4 Referrences

(1) FLO: A Decentralised Data-flow System, Part 1 of 2

G.K. Egan and C.P. Richardson, Manchester Uni. 1979

(2) ibid. Part 2 of 2

(3) A User's Manual to DL1

C.P. Richardson and G.K. Egan, Manchester Uni. 1980

2. Circuit Description and Operation

Section 2 describes the processing element in detail, giving all circuit diagrams, operating instructions, etc..

Much of the information necessary for operating the element is contained in the data sheets of many of the chips used in the design. Correspondingly, many of the relevant data sheets are included in this manual for the reader's referrence.

2.1 Refresh Timing

Fig. 2.1.1 shows the circuit used to generate refresh requests at 2 ms intervals. An asynchronous refresh request can be software initiated by writing to port 6 ($FFFF06) for CPU1 and by reading at port 1 ($FFFF01) for CPU2; in each case the data involved is irrelevant.

2.2 Clock Generation

Fig. 2.2.1 shows the circuit used to generate symmetrical 4 and 8 Mhz clocks. The former is only used for the Am9511 floating point unit controlled by CPU2, whilst the 8 MHz clock is used for all other clocks in the circuit, in particular for CPU1 and CPU2.

2.3 Reset and Halt

Fig. 2.3.1 shows the reset/halt circuit. Two push button switches (PB1 and PB2) are used to manually reset or halt the processing element respectively. Resets and halts can also be software initiated by either CPU, but the entire processing element will react since there is only one reset path and only one halt path.

Provision has been made for a system reset signal, i.e., an external reset input. The prototype PCB has system reset tied high through a fine link which is easily cut if the system reset facility is required in the future. The main foreseeable use of this input is to reset a data-flow machine comprising of a number of processing elements.

Note that any reset also forces a halt as is required by the 68000 CPUs.

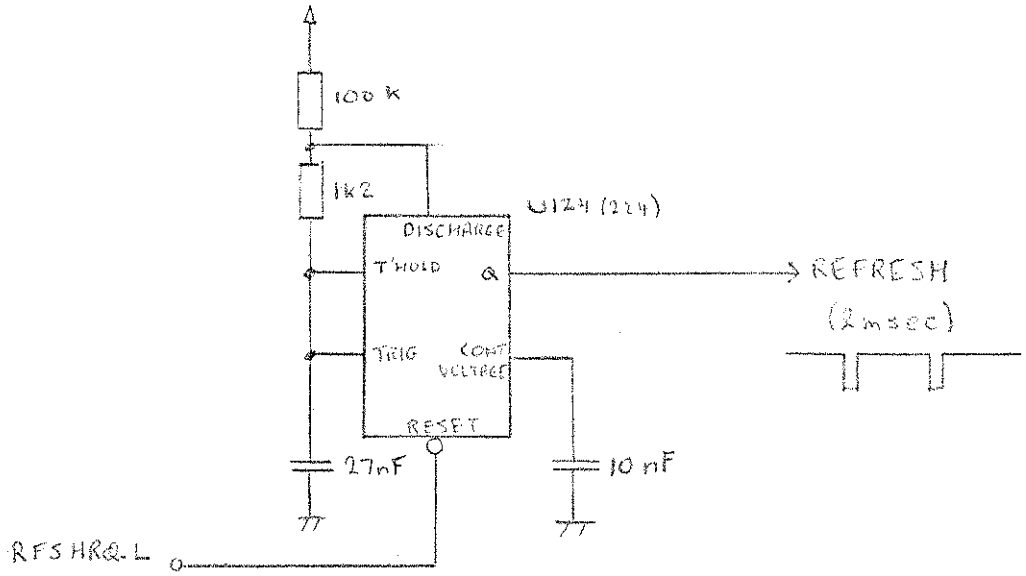


Fig. 2.1.1 Refresh Timer

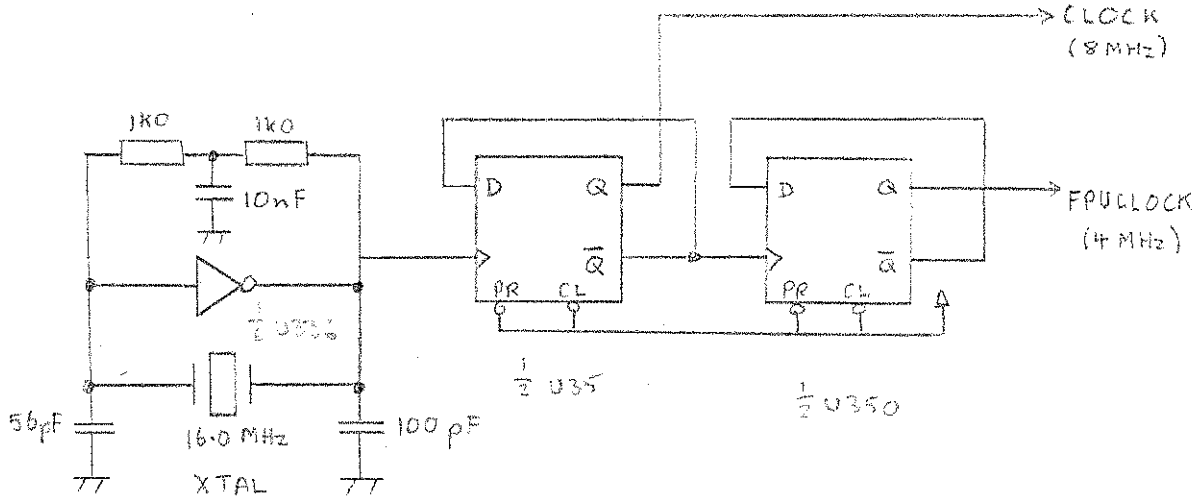


Fig. 2.2.1 Clock Generation

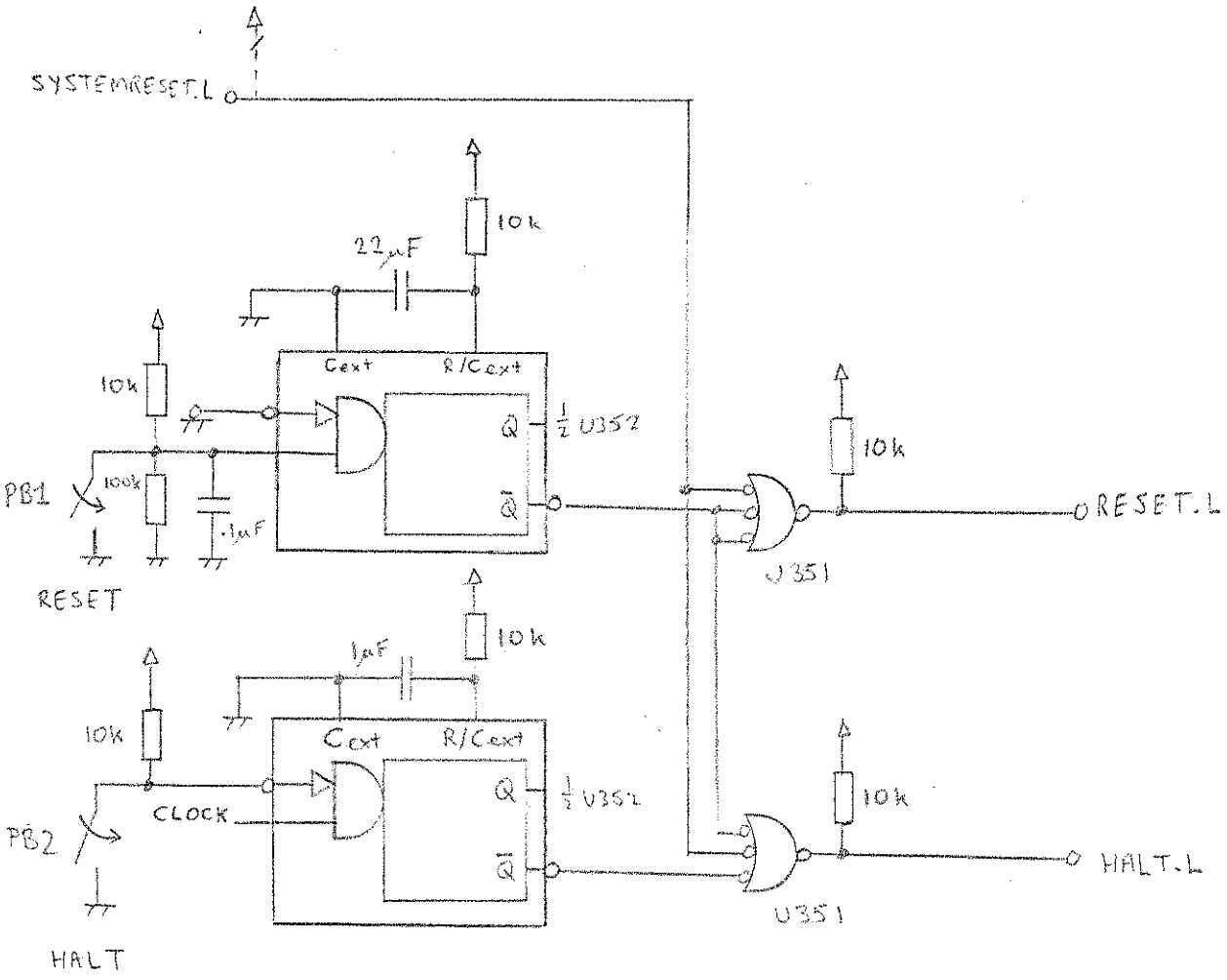


Fig.2.3.1 Reset and Halt Circuit

The monostable is used to provide a debounce circuit for the halt push button.

2.4 Memory Decoding

Fig. 2.4.1 shows the memory decoding circuit. The circuit is identical for each CPU. A 'memory address' is indicated by A20-23 being low which generates the MAD signal. This effectively allows for 2^{20} bytes of memory (or $\frac{1}{2}$ Mword), although at the moment the circuit can only accept $\frac{1}{4}$ Mword, using 256 K dynamic rams.

Port Address.L is true whenever A 8-23 are high and A4-7 are low. This gives a total of eight 16 bit ports in the address range $FFFT00 - $FFFT0F, where A0 indicates either the upper or lower byte. The port access signals are verified by the address strobe. 16 port access signals are generated being eight read/write ports.

Table 2.4.1 shows the current port allocations. Not all ports are presently used thus allowing the addition of extra peripherals, although the data acknowledge signal also needs to be generated for any additional ports. (ram $0-$07FFFF (512k) rom $0-$000FFF (4k))

2.5 RAM/ROM Arbitration

The memory address signal MAD refers to both Ram and Rom addresses. However, when a reset is generated, the Eprom.L signal goes true and enables Rom (fig. 2.5.1). The Eprom read.L signal provides a data acknowledge for eprom reads and is also used in the ram circuit to inhibit ram cycles when a read is being done. Ram can be written to

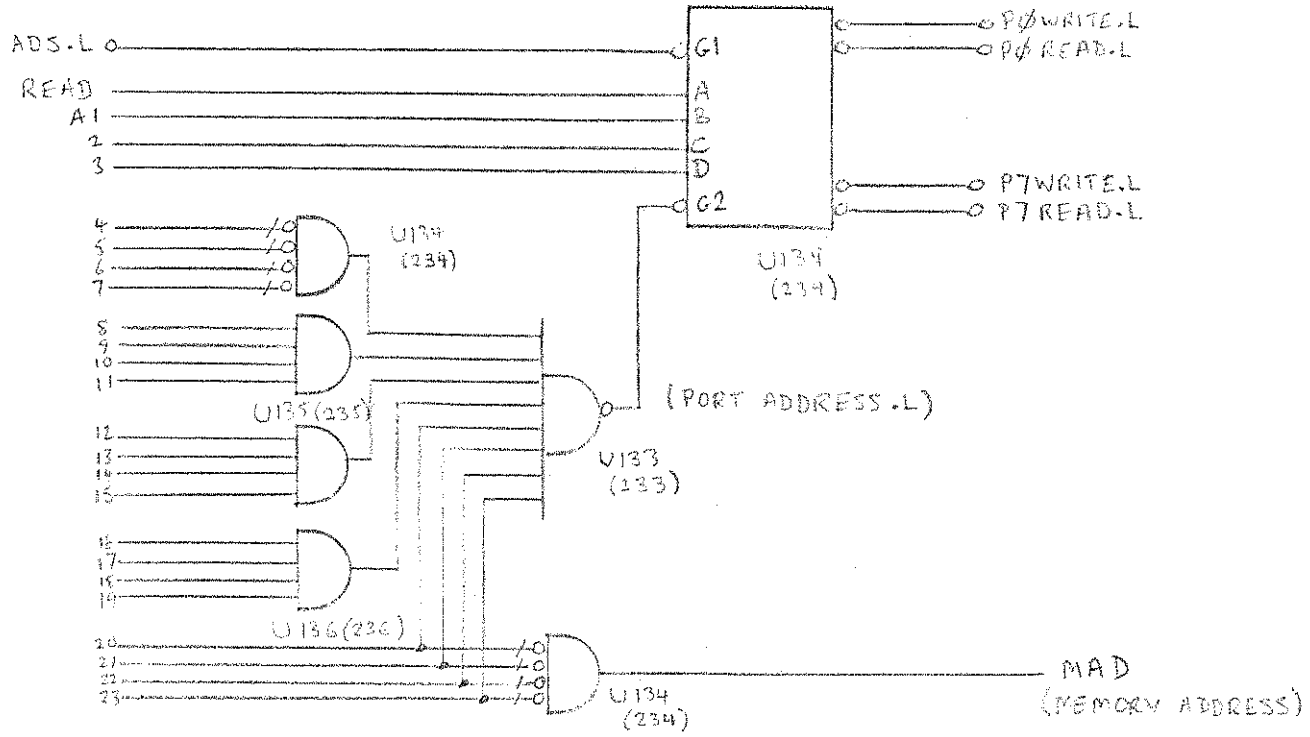


Fig. 2.4.1 Memory Decoding

PORT	R/W	ADDRESS	CPU	IMPLEMENTATION
0	R	$FFFF00	1	Not used
	W			" "
	R		2	Read token from pipe queue
	W			Write token to output queue
1	R	$FFFF02	1	Not used
	W			" "
	R		2	Initiate burst refresh
	W			Write token to local queue
2	R	$FFFF04	1	Port read
	W			Port write
	R		2	Not used
	W			" "
3	R	$FFFF06	1	Not used
	W			" "
	R		2	" "
	W			" "
4	R	$FFFF08	1	Not used
	W			" "
	R		2	Port read
	W			Port write
5	R	$FFFF0A	1	Not used
	W			" "
	R		2	" "
	W			" "
6	R	$FFFF0C	1	Read token from input queue
	W			Initiate burst refresh
	R	$FFFF0D	2	Pop data byte (9511)
	W			Push data byte(9511)
7	R	$FFFF0E	1	Read token from local queue
	W			Write token to pipe queue
	R	$FFFF0F	2	Read FPU status
	W			Enter FPU command

Table 2.4.1 Port Allocations (FPU uses byte addresses)

at any time. System initialisation should be performed by copying rom to ram and then inhibiting rom (control signals are discussed in the Port Read/Port Write sections). Note that Eprom read.L is verified by address strobe but $\overline{\text{Eprom read.H}}$ (the same signal) is not! However there will be no spurious data acknowledge signals from the ram circuit due to this apparent anomaly since the DTACK circuit is designed to suppress data acknowledge when the address strobe is false. Also note that the ram will not be spuriously accessed since address strobe is needed before a CAS signal is generated.

2.6 Floating Point Operations

The computation processor, CPU2, accesses the Am9511 floating point unit through two of the memory mapped I/O ports described in the memory decoding section. The eight bit FPU data register is interfaced to the lower data bus of CPU2 (D0-D7), hence this unit should be accessed with byte size data (although word size transfers can be used).

Table 2.6.1 shows how to access the FPU.

PORT	R/W	ADDRESS	COMMAND
6	W	$FFFFOD	Push Data Byte
6	R	" "	Pop Data Byte
7	W	$FFFFOF	Enter Command Byte
7	R	" "	Read FPU Status

Table 2.6.1 Accessing the FPU

For details of data formats, commands, etc., see the 9511 data sheets enclosed in this document. Fig. 2.6.1 shows the FPU circuit.

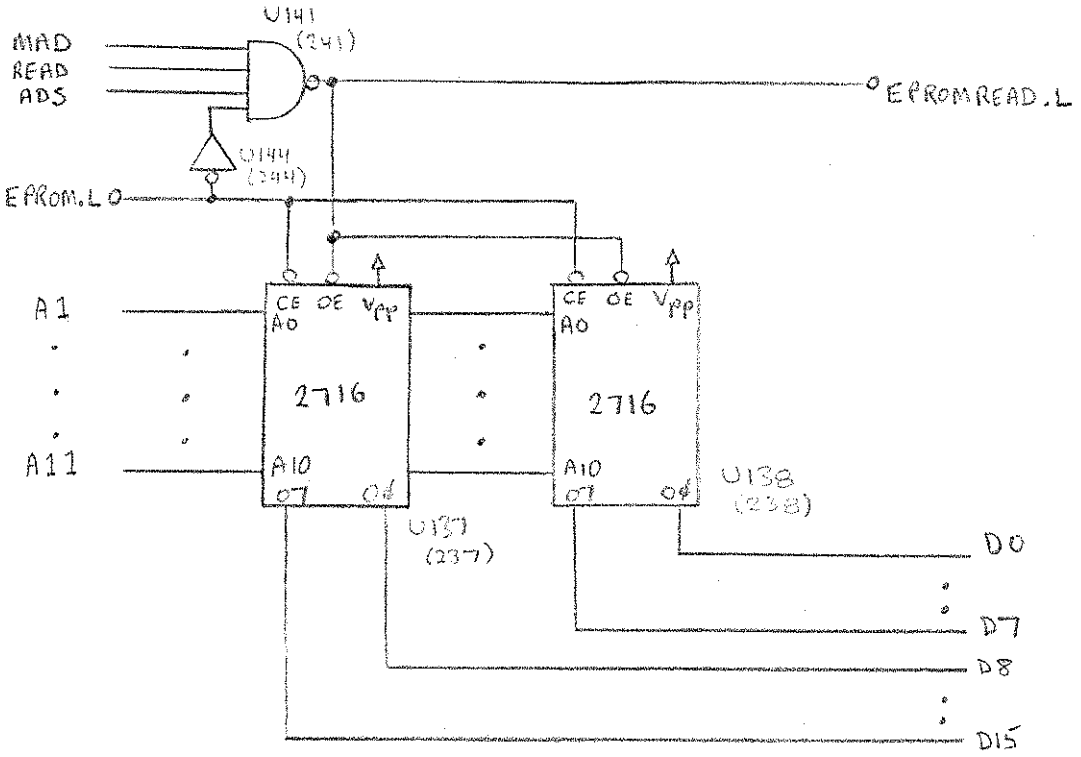


Fig.2.5.1 Eprom Circuit

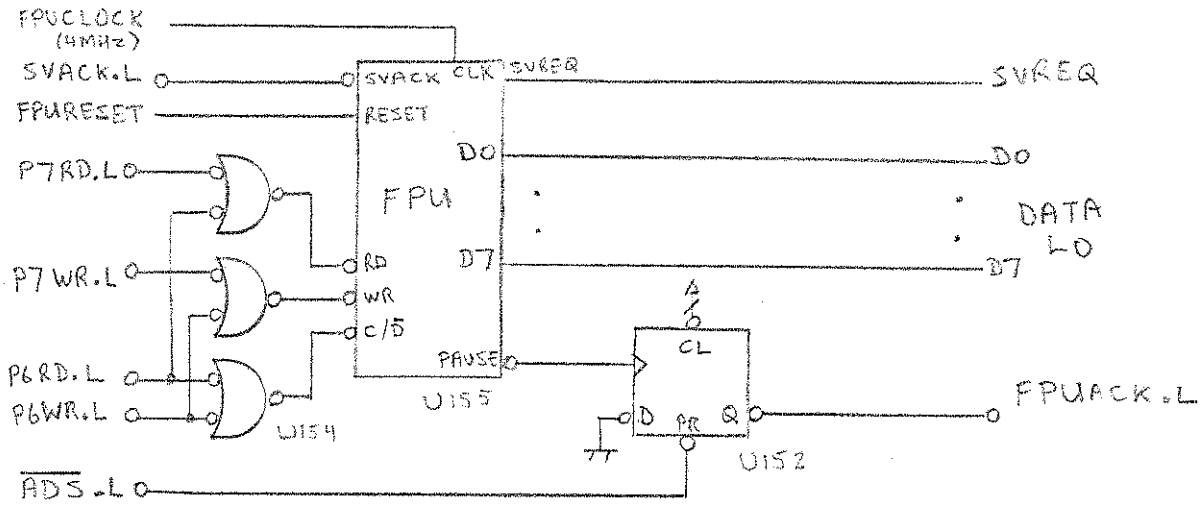


Fig.2.6.1 Floating Point Unit Circuit

2.7 Dynamic Memory Description

Fig. 2.7.1 shows the DRAM arrangement. 150 nsec devices must be used for correct operation. Special ram driver chips, Am2966, are used to eliminate overshoot and undershoot on the ram data lines. The dynamic memory controller used, National DP8409, provides similar buffering on all of its outputs. The 22 ohm resistors are used to interface LS and S TTL devices to the rams, i.e., CAS and parity bits. Notice that the ram address signal RAD is not verified by the address strobe, but this is no problem since no CAS signal will arrive without the address strobe having been active and no data acknowledge will result since it is suppressed when address strobe is false.

The ram circuit is directly compatible with the newer 256k chips since the DP8409 has nine address outputs; the PCB will directly accept 256k chips.

The prototype ram array is 64k x 18 bits, where two bits are used for byte parity. Odd parity is used (9 bits).

2.8 Parity Circuitry

Fig. 2.8.1 gives details of the parity circuit. In the event of a parity error, a bus error is generated on the following cycle allowing the appropriate error recovery to take place. Due to certain features of the 68000 processor, certain 'tricks' can be used to find the error address; see software.

Parity errors are inhibited during reset to prevent spurious errors. The circuit uses 74S280 devices to generate and check parity for all

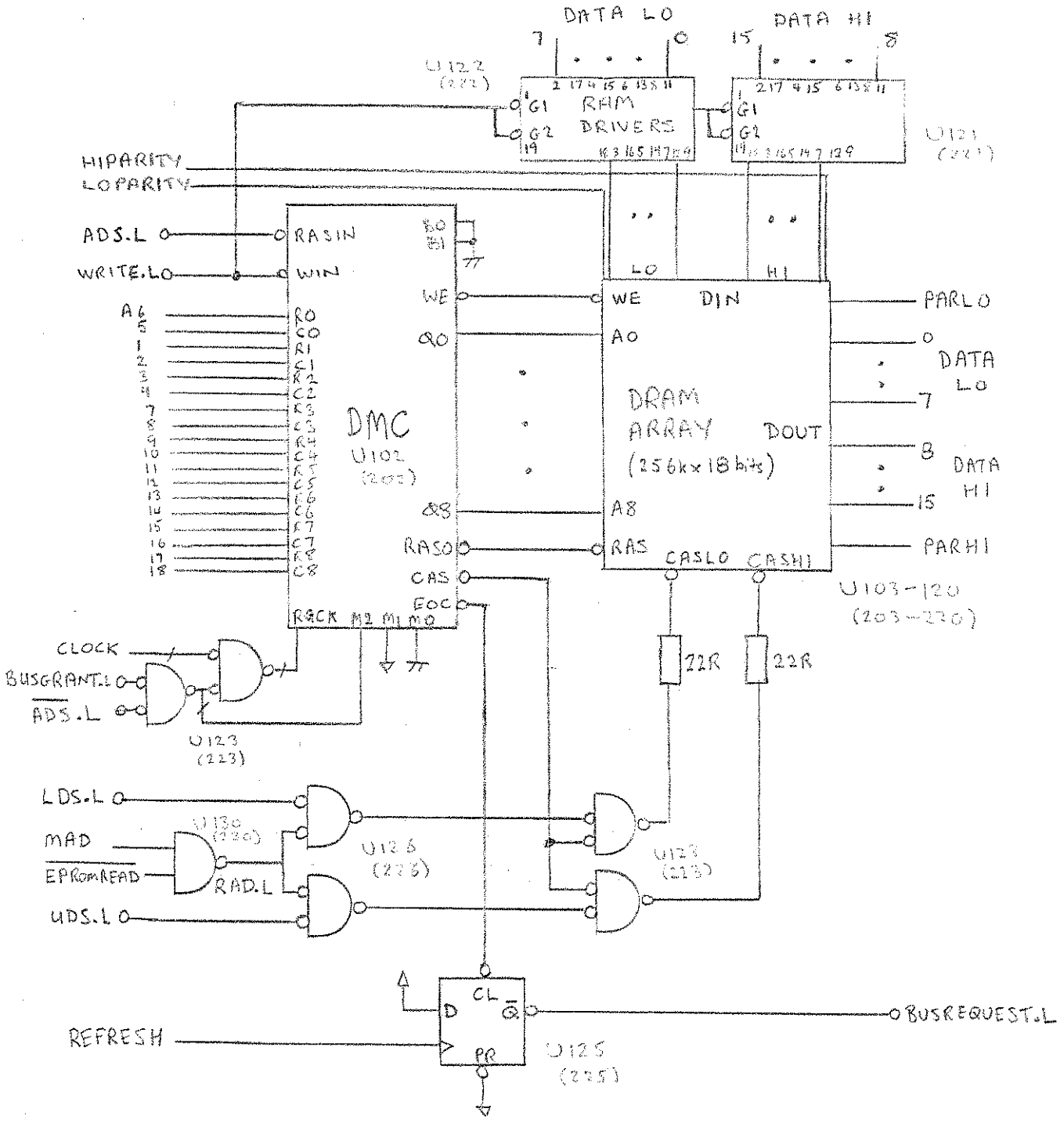


Fig.2.7.1 Dynamic Memory Interface

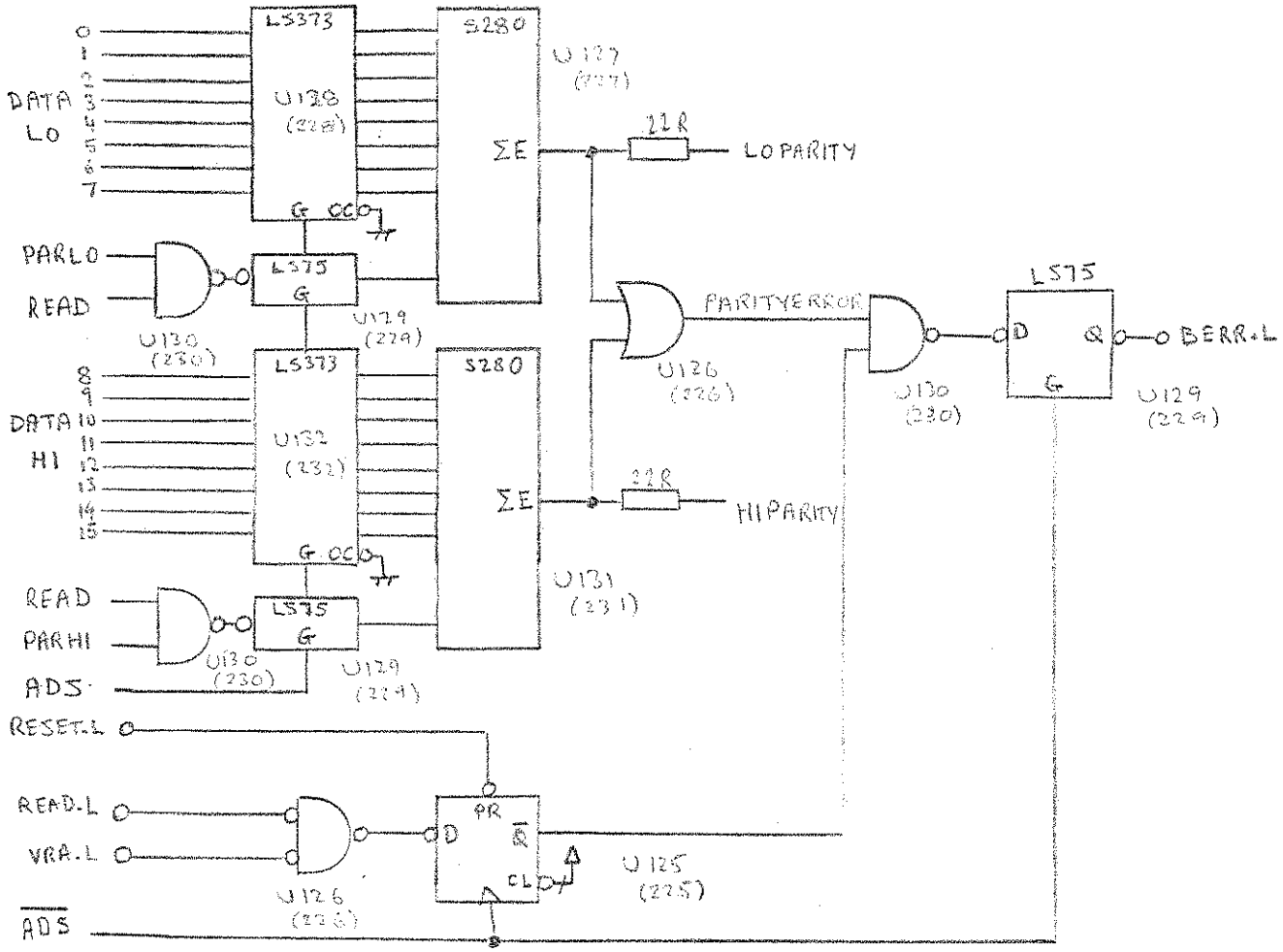


Fig.2.8.1 Dynamic Memory Parity Generator/Checker Circuit

DRAM accesses.

For a 256 kbyte system like the prototype machine, the mean time to fail is 832 days for soft errors (alpha induced), and considerably longer for hard errors; refer to Advanced Micro Devices memory support handbook for further details.

2.9 Ports

Fig. 2.9.1 shows the port details for CPU1 and fig. 2 .9.2 shows the details for CPU2. There are two eight bit ports for each CPU. Table 2.4.1 shows the word addresses of the ports but the eight bit ports are to be used separately and independently, so that only byte transfers should be used. The address of the high data ports is as shown in the table whilst the address of the low data port is one higher (A0=1), as explained, byte transfers should be used in each case. Thus there are four ports effectively for each CPU described as port write/read higher/lower (PWH, PWL, PRH, PRL), as shown in figures 2.9.1 and 2.9.2 . These figures also show the allocation of each data bit for use in the ports, for example, to establish any of the interrupt masks first observe that they are active low, the appropriate bit can be set low and the port rewritten (PWH). Further discussion of the ports follows.

2.9.1 Status Output (PWL)

Status information can be made available externally through the external status ports at address $FFFF05 or $FFFF09.

The eight bit status information is transferred from the low data bus into a latch at the address $FFFF05 for CPU1 and $FFFF09 for CPU2. The latch signal is made external through sockets SK2 and SK3 so that external devices can sample the signal and detect when status information has been made available.

The proposed use of the status ports is for sending information external to the processing element but at the same time bypassing the output queue, thus minimising the interference to data flow. Such information could typically be run time statistics, etc..

2.9.2 Command Input

Together with the status output, PWL, there is provision for an external command input. Commands are read by issuing a PRL sequence, the addresses are $FFFF05 for CPU1 and $FFFF09 for CPU2.

Commands are assembled on bits C0-C7 and a Command WR.L signal is generated. This signal is treated as a non-maskable interrupt by the processing element which is programmed to immediately read the command. To ensure that commands do not 'overrun', the command read signal can be monitored by the commanding device since this signal is made accessable through sockets SK2 and SK3.

2.9.3 Internal Status Reading

CPU1 has access to the FIFO status lines, full, ½full and empty, of the input, pipe and local queues. CPU2 has access to pipe, local, output statuses. The internal status is read with a port read high

PRH access, the addresses are $FFFF04 for CPU1 and $FFFF08 for CPU2.

The information returned on bits D8-D15 is detailed in figures 2.9.1 and 2.9.2. Together with the interrupt masks available in the internal control ports, PWH, the status ports can be used to allow the processing element to operate in a polled mode as opposed to its normal completely interrupt driven operation mode.

A half full signal is not available for the pipe, local and output fifos since they are 2k words long and the half full signal, specified for a 4k fifo is thus used as the full signal for these fifos. The input fifo is 4k long (allowing greater buffering on the input) and has empty, full and half full signals all operational.

2.9.4 Internal Control Ports

The internal control ports serve a dual purpose in that they provide control over device resets, interrupt masks, etc. and are also cleared by an element reset. The data bits associated with these ports are generally active low since a reset clears the outputs to this level.

2.9.4.1 Interrupt Masks

The four interrupt masks provided for each CPU can be used to inhibit certain fifo interrupt requests, thus allowing polled operation for these fifos, this significantly increases software flexibility. As pointed out the masks are active low and are thus active immediately following an element reset. To enable the interrupts, the masks must be negated following a reset. Further discussion is given in the interrupt section.

2.9.4.2 Fifo Resets

The fifo resets are active low, so that an element reset will result in the fifos also being reset.

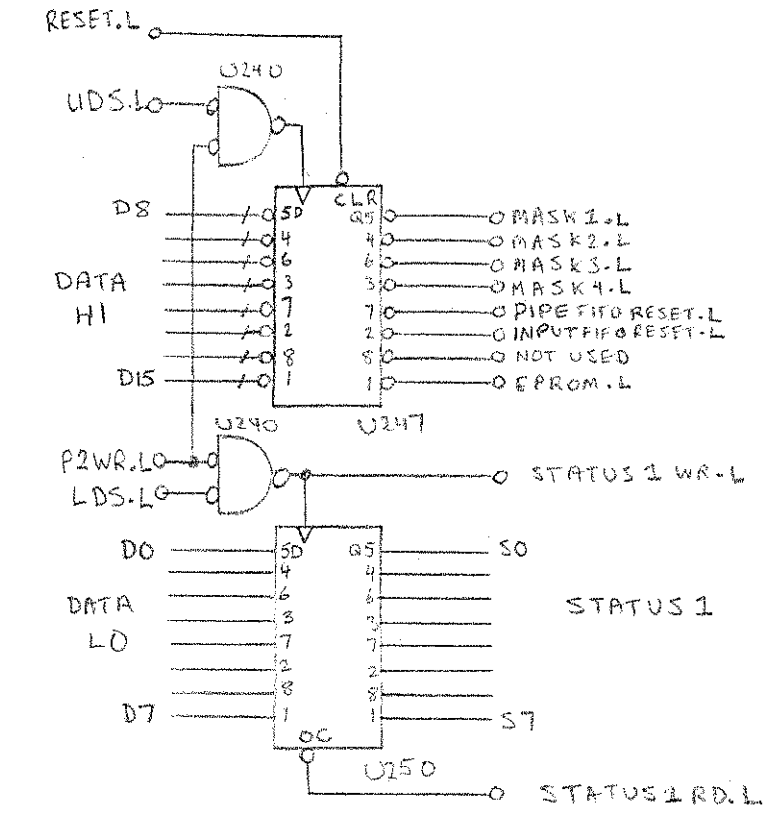
CPU1 has reset control over the input and pipe fifos whilst CPU2 can reset the local and output fifos. Resetting the fifos effectively clears the corresponding queues. Ofcourse the fifos may be reset at any time by writing the appropriate data to the control ports, so that a system reset need not be generated to clear any of the fifo queues.

In general, the fifos are reset by the CPU that writes to them, the only exception to this is the input fifo which is reset by CPU1 even though it only reads the input queue. For an external device to clear the input queue, a command must be sent to CPU1 to have it clear the queue; such a command could actually be passed through the input queue if so desired, but is much more likely to be sent through the command port.

Note that one signal is used to reset both local and output fifos connected to CPU2 and thus these fifos must be reset together.

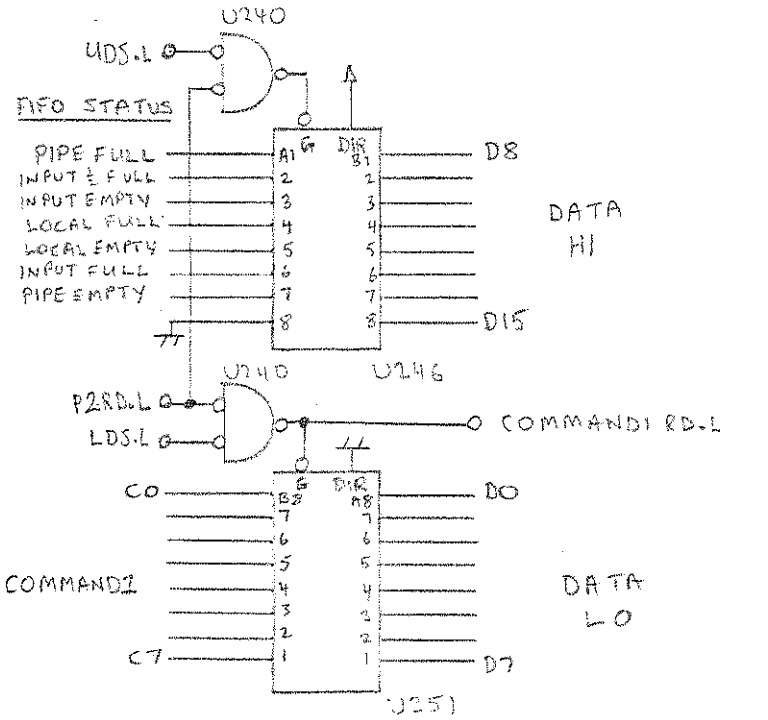
2.9.4.3 Enabling ROM

Eprom.L is the active low signal used to enable rom. This signal is asserted when an element is reset for both CPUs and can also be controlled individually by each CPU. Rom is only ever active for read accesses so that ram can always be written to but read from only when Eprom.L is negated.



Port Write High (PWH1)
($FFFF04)

Port Write Low (PWL1)
($FFFF05)

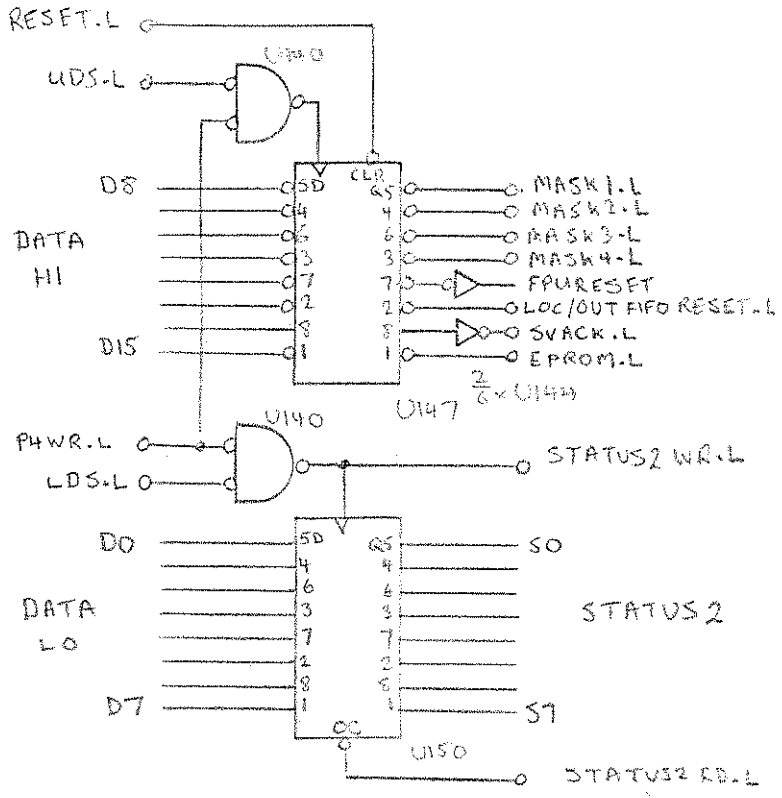


Port Read High (PRH1)
($FFFF04)

Port Read Low (PRL1)
($FFFF05)

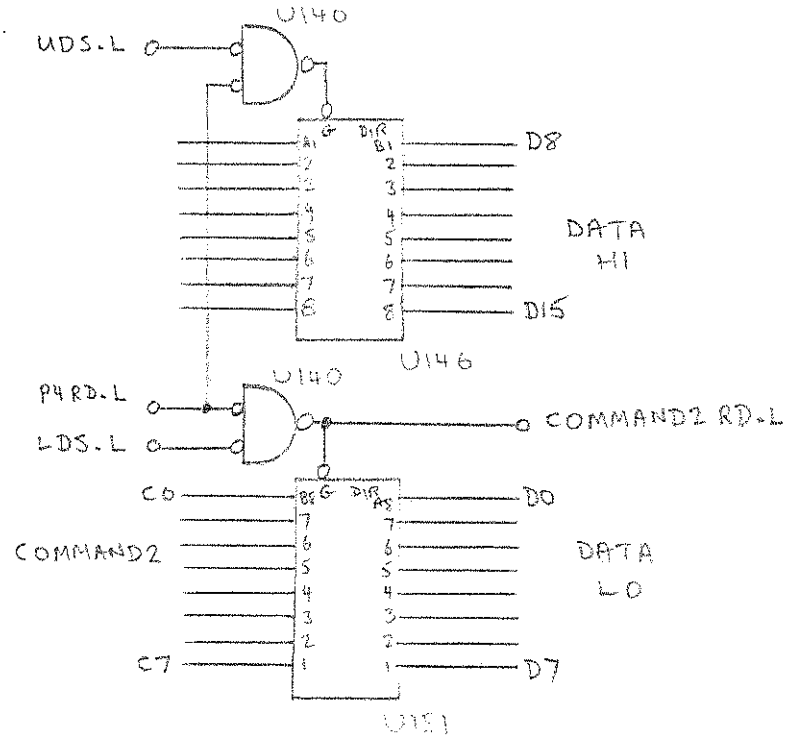
Fig. 2.9.1 CPU1 Port Details a) Port Write

b) Port Read



Port Write High (PWH2)
($FFFF08)

Port Write Low (PWL2)
($FFFF09)



Port Read High (PRH2)
($FFFF08)

Port Read Low (PRL2)
($FFFF09)

Fig. 2.9.2 CPU2 Port Details a) Port Write

b) Port Read

It is anticipated that rom will be copied to ram and then disabled upon system initialisation.

2.9.4.4 FPU Reset

The floating point unit is also reset by an element reset, i.e., it is also an active low signal at the control port (an inverter is used to make it active high as required by the 9511 device). This signal is only relevant to CPU2.

2.9.4.5 FPU Service Acknowledge

Svack is made active high at the control port so that it is negated upon an element reset. In this case an inverter is again used but this time to convert the active high signal to active low. Refer to fig.2.9.2 for the data pin assignments for this signal, D14.

2.10 FIFO Circuits

2.10.1 Input FIFO description

Fig. 2.10.1 shows the input fifo circuit. Note that the write is delayed through two inverters to meet ram set-up requirements. The input queue is 4k x 16 bits and is used to pass tokens into the processing element. A Signetics 8X60 is used as the fifo ram controller, which generates all the ram interface signals needed to operate the ram as first-in first-out memory.

The data input and output buses are isolated by the 74LS245 buffers to prevent contentions on the data buses allowing fully asynchronous operation of input and output devices.

The FRC chip provides empty, ½ full and full status signals which are used extensively as interrupts within the processing element. For example, if the input queue becomes full whilst the local queue is being read from, then an interrupt will be issued and the appropriate action taken; in this case the input queue could be read into ram temporarily. The empty signal is most important as it is the only way of telling if there is data waiting to be collected from a queue.

The input fifo must also provide a handshake signal to the external device writing tokens to the queue. This is achieved using the Fifo write.L signal generated by the FRC. The handshake sequence proceeds as follows:-

A token to be written to the input queue is assembled on the 16 input lines. A shift in request is issued to the processing element board, in particular to the input fifo ram controller. This signal will eventually result in the FRC generating a write signal to the ram, but this signal can be significantly delayed if a shift-out operation is already in progress. The FRC has the ability to stack up commands in this manner (see the 8X60 specs for further details). Upon receiving the write handshake, the external device must remove the shift-in request and the sequence is complete.

CPU1 can read from the input queue by reading a word from address $FFFF0C as shown in table 2.4.1. In this case the transaction is completed upon receipt of the ram read signal which is used in the data acknowledge circuit.

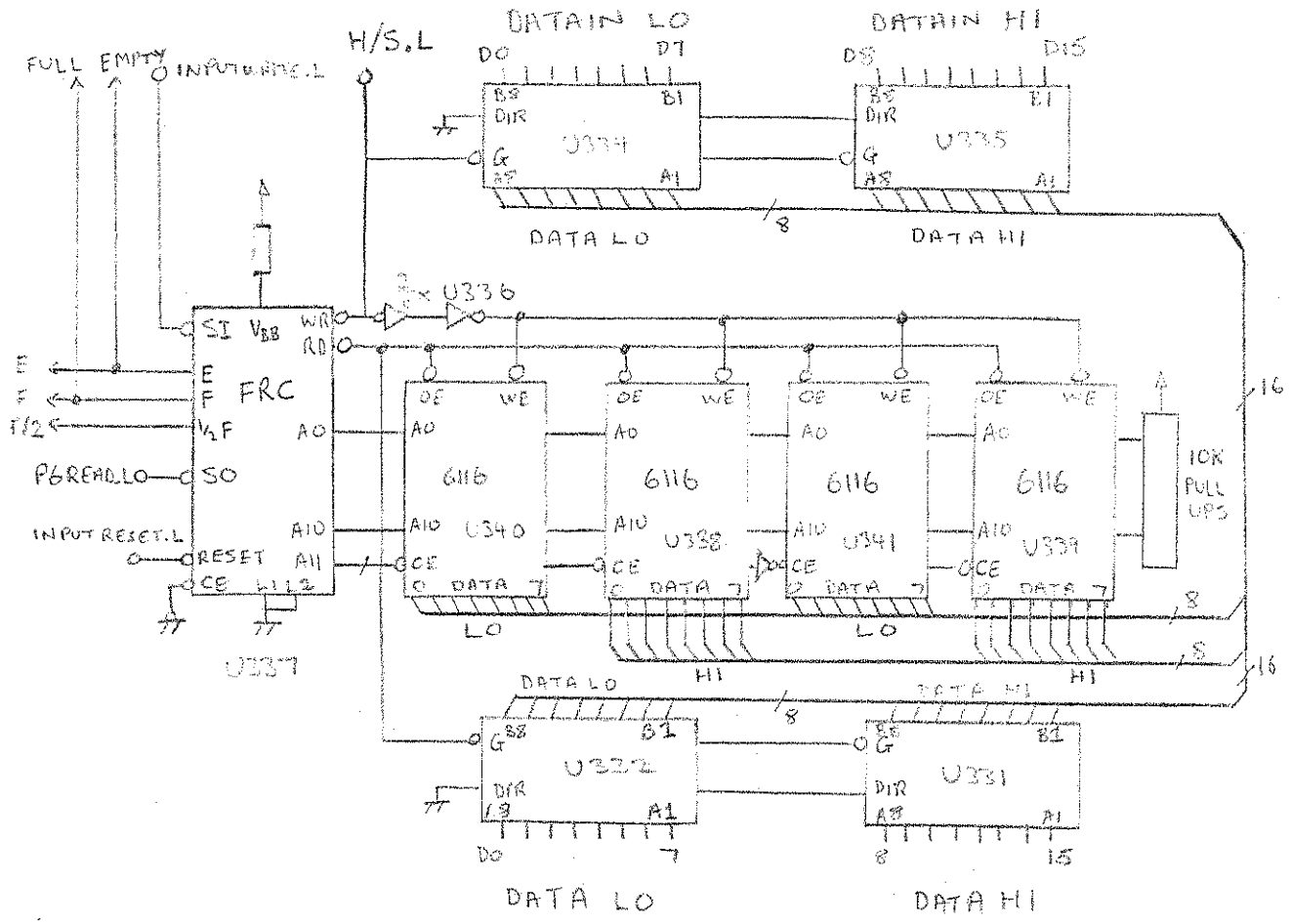


Fig.2.10.1 Input FIFO Circuit

CPU1 can reset the input fifo by asserting (clearing) bit 6 in the internal control port at address $FFFF04.

2.10.2. Pipe Fifo Queue

The pipe fifo carries tokens from CPU1 to CPU2. The circuit is given in fig. 2.10.2 The main peculiarity of this circuit is that the ½ full signal is actually treated as a full signal since the fifo is set up for 4k but only uses 2k (words) of ram.

CPU1 writes to this queue at address $FFFF0E and CPU2 reads the tokens from address $FFFF00. Again the write and read signals from the FRC are used in the data acknowledge circuits of CPU1 and CPU2 respectively.

CPU1 can reset the pipe queue by asserting bit5 in the control port at address $FFFF04.

2.10.3 Local Fifo Queue

The local queue is identical to the pipe queue except that tokens are passed from CPU2 to CPU1. CPU2 can reset the local queue by asserting bit6 in the control port at address $FFFF08. CPU1 reads the local queue at address $FFFF0E and CPU2 writes to the queue at address $FFFF02.

2.10.4 Output Fifo Queue

CPU2 writes tokens to the output queue at address $FFFF00 and can reset the FRC by asserting bit6 in the control port at address $FFFF08.

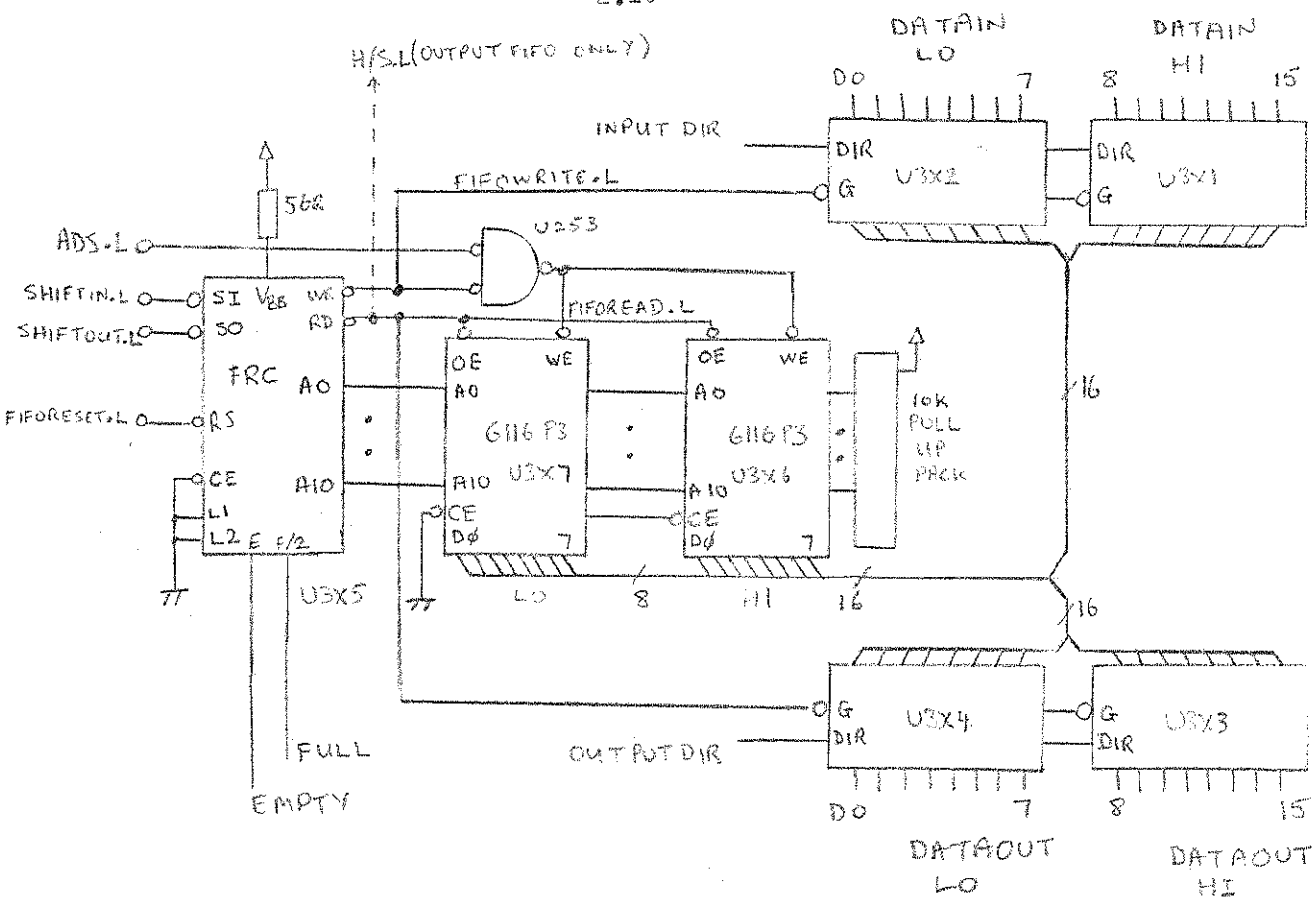


Fig.2.10.2 Pipe, Local and Output FIFO Circuit
X= 0 1 2

A handshake sequence is used at the output very similar to that used at the input. In this case, the external device generates a shift-out request to the output FRC which will eventually respond with a ram read signal which is used as the handshake by the requesting device. Fig. 2.10.2 shows the circuit for the output fifo.

Note that the fifo status signals full anf empty are made available externally for both the input and output fifos.

2.11 Interrupt Structure

Figure 2.11.1 shows the interrupt generation circuit for both CPU1 and CPU2.

Note that interrupt level 0gives no interrupt and level 7 gives a non-maskable interrupt. The masks are active low and can inhibit interrupts as explained in section 2.9.4.1. Ofcourse the overall interrupt level is determined by software and previous interrupts in the usual manner. The 74LS148 interfaces directly to the MC68000 cpu, and is used to priority encode encode interrupt requests. The command signals generate non-maskable interrupts as explained in section 2.9.2.

CPU1 gives greatest priority to the local queue and so this is assigned to level 6 (local empty). Similar reasoning has led to the remaining interrupt levels being assigned as shown.

Figure 2.11.2 shows the interrupt acknowledge and vector number acquisition circuit. The vector numbers used are the MC68000 auto-vectors although auto-vectoring is not used. This is achieved by reading in A1, A2, A3 as shown, which gives vector numbers from $19 to $1E.

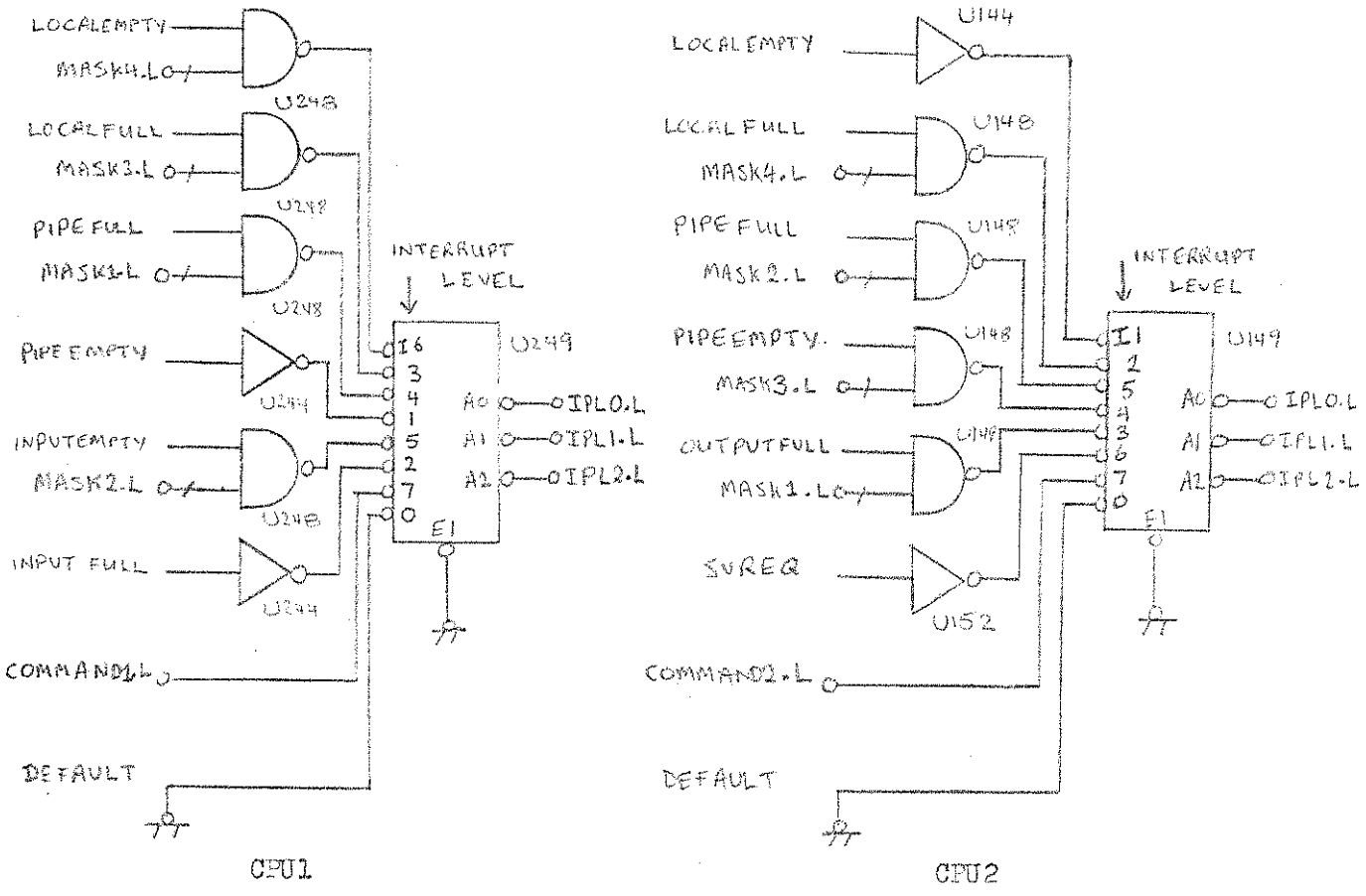


Fig.2.11.1 Interrupt Circuit with Priority Details

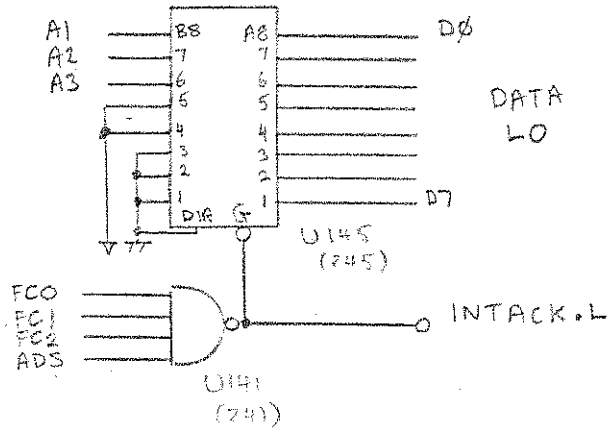


Fig.2.11.2 Interrupt Acknowledge and Vectoring

2.12 Data Acknowledgement

Fig. 2.12.1,2 show the DTACK circuit for CPU1,2. The flip flops are used to generate wait states where appropriate. Note that DTACK is suppressed when the address strobe is false by presetting the flip flops. This prevents spurious DTACK signals from arising and also ensures that DTACK is negated after the address strobe goes false in a normal cycle.

2.13 Socket Wiring

The wiring of the on board interface sockets is shown in fig. 2.13.1.

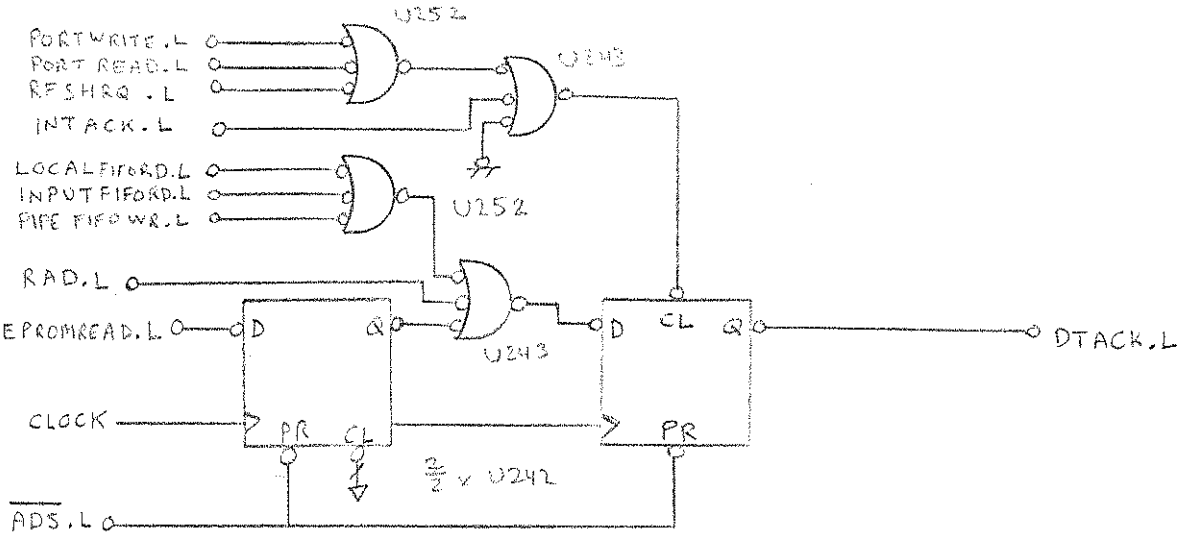


Fig. CPU1 Data Acknowledge (DTACK) Circuit

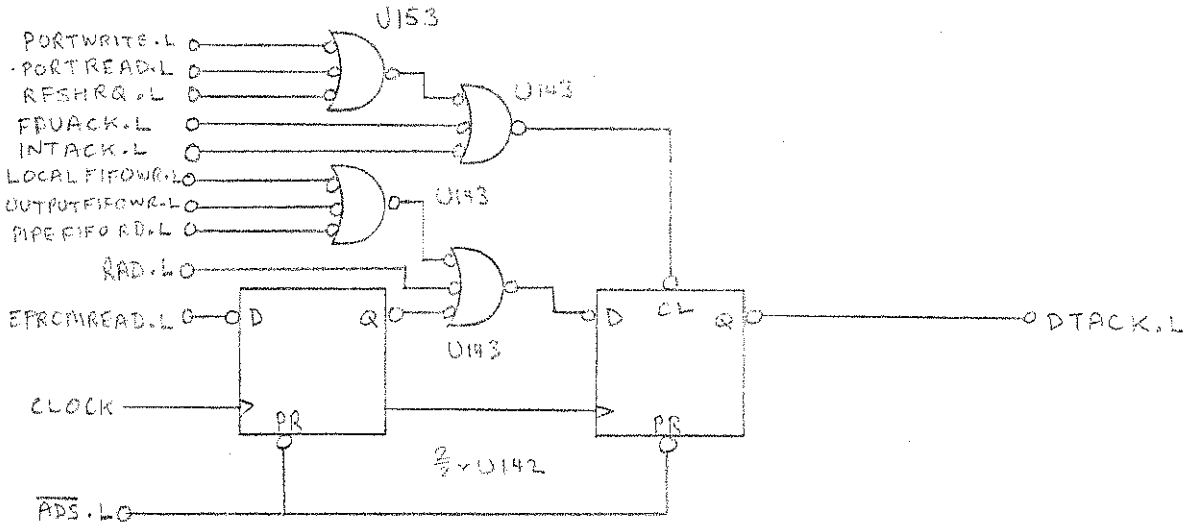
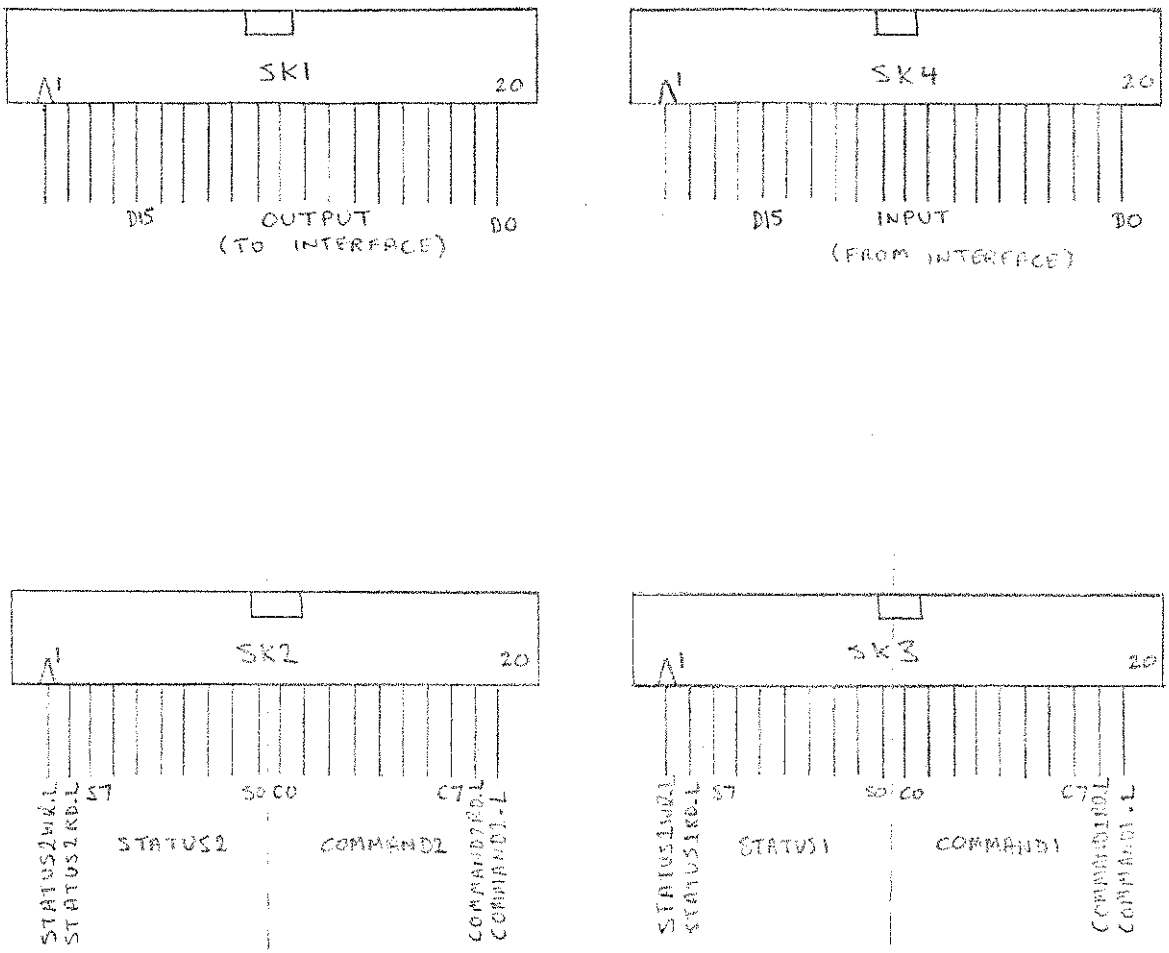


Fig. CPU2 Data Acknowledge (DTACK) Circuit



(All pins 21-40 grounded)

Fig. 2.13.1 Signal Connections to Interface Sockets

3. S100 Interface

A wire-wrap board was built to interface the prototype processing element to the S100 bus. This allows the unit to be controlled and monitored for study purposes with the very convenient and well known CPM/Z80 system, although any S100 system can be used.

The circuit provides a good illustration of how the various hand-shakes are operated and of general device interfacing.

3.1 Circuit Description

The circuit diagram of the S100 interface is shown in figure 3.1.1.

3.1.1 Memory Decoding

Address lines A3-A15 are decoded by U1, U2 and U10. These addresses must correspond to that set on the switch banks. Note that the device is memory mapped. A0-A2 are decoded to give the operations defined in table 3.1.1.

3.1.2 Giving Commands

CPU1 is commanded by writing to address Base + 4 where Base is given by A3-A15. CPU2 is sent a command by writing to address Base + 5. Such actions will set U9 generating a non-maskable interrupt in the processing element. U9 remains set until the command is read by the data-flow element.Alternatively, U9 can be cleared by any access to addresses Base + 0 for CPU1 and Base + 1 for CPU2. An internal status bit is set whenever a command is pending.

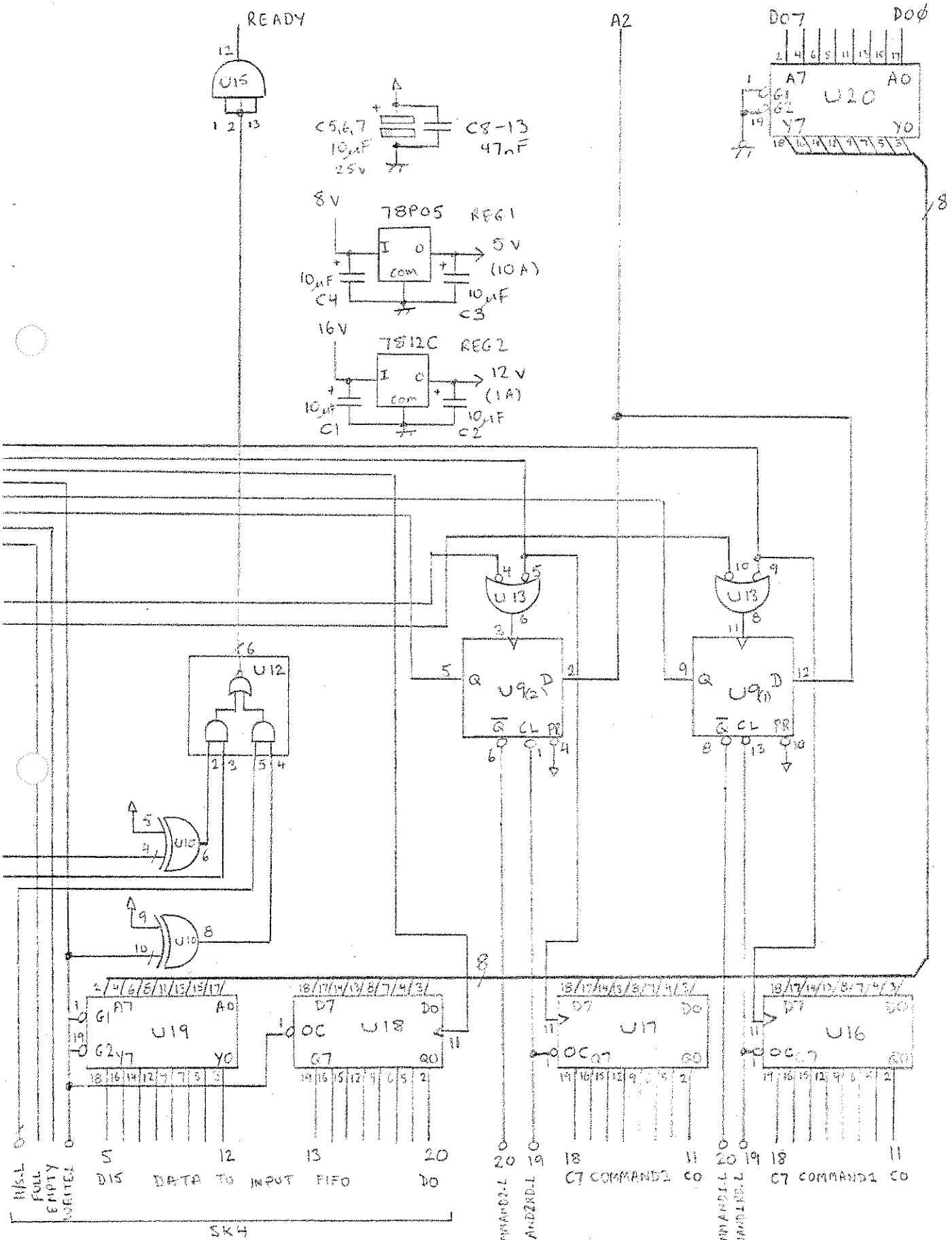
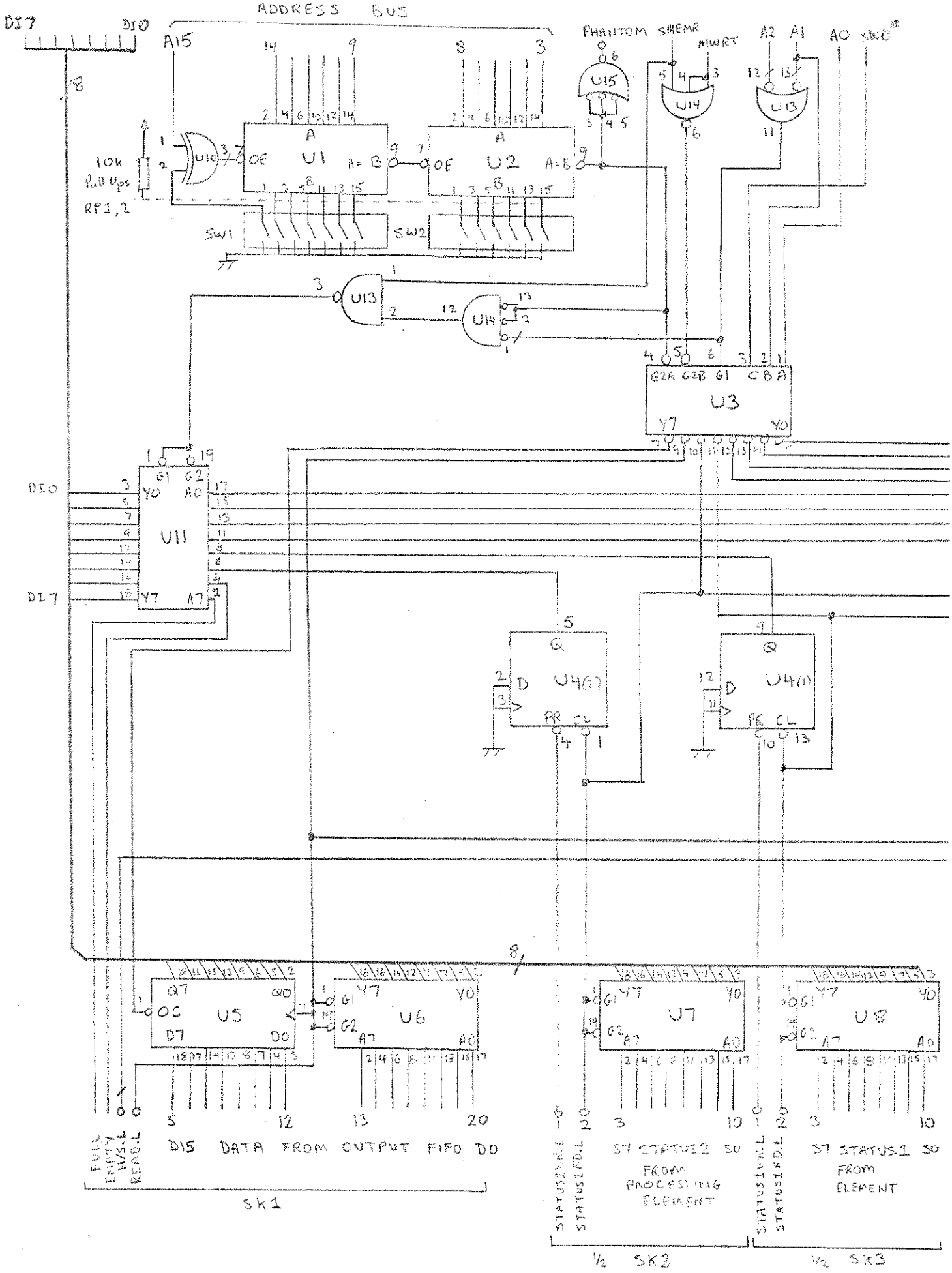


Fig. 3.1.1 S100 Interface Circuit


DI7
DI0
ADDRESS BUS
A15
PHANTOM
SMEMR
MIWRT
A2
A1
A0
SWC*
U15
U14
U13
U10
U1
U2
10k
Pull Ups
RP1,2
SW1
SW2
G2A G2B G1 C B A
U3
Y7
Y0
DI0
DI7
U11
G1
G2
A0
A7
U4(2)
U4(1)
D
Q
PR
CL
U5
U6
U7
U8
OC
D7
D0
DIS DATA FROM OUTPUT FIFO D0
FULL
EMPTY
H/S*
REQ0-L
SK1
S7 STATUS2 S0
FROM PROCESSING ELEMENT
S7 STATUS1 S0
FROM ELEMENT
STATUS2RD-L
STATUS1RD-L
½ SK2
½ SK3

ADDRESS A2 A1 A0	R/W	OPERATION
0 0 0	R	status1 read, clear command1 interrupt
	W	command1 write with no interrupt
0 0 1	R	status2 read, clear command2 interrupt
	W	command2 write with no interrupt
0 1 0	R	read output fifo, word transfer
	W	latch lower byte for input fifo
0 1 1	R	read higher byte from latch
	W	write to input fifo, word transfer
1 0 0	R	status1 read, set command1 interrupt
	W	command1 write with interrupt
1 0 1	R	status2 read, set command2 interrupt
	W	command2 write with interrupt
1 1 X	R	read status information
	W	no action

Table 3.1.1 S100 Interface Operations

3.1.3 Reading Data-flow Status

Status1 is read by reading from address Base + 0 (clearing command1) or Base + 4 (setting command1). Status2 is the same with the addresses incremented by one.

3.1.4 Writing to Input Queue

A sixteen bit word must be assembled in two stages from the eight bit S100 bus. This is achieved by writing the LOWER byte to address Base + 2 which latches the byte. The sixteen bit transfer is completed by writing the HIGHER byte to address Base + 3 which then transfers the entire sixteen bit word or token to the input queue. U10 and U12 are used to implement the handshake by inserting wait states until the token has been written into the input queue.

3.1.5 Reading from Output Queue

A sixteen bit word must be read in two stages by the S100 system, but can be transferred from the output queue in one stage in a manner similar to the writing of an input word discussed above. The sixteen bits are transferred to the interface board by reading from address Base + 2. At the same time, the LOWER byte goes to the S100 bus whilst the HIGHER byte is latched into U5. To read the HIGHER byte from U5 a read access is made to address Base + 5.

3.1.6 Internal Status Register

An internal status register is maintained that contains information

shown in table 3.1.6.

The status information is accessed by reading from address Base + 6 or Base + 7. A write to either of these addresses has no effect.

BIT	INFORMATION (When Set)
D0	Command1 Pending
D1	Command2 Pending
D2	Input Fifo Empty
D3	Input Fifo Full
D4	Status1 Available
D5	Status2 Available
D6	Output Fifo Empty
D7	Output Fifo Full

Table 3.1.6 Status Information

NB. Reading status1 or status2 clears bits 4, 5.

3.2 Power Supply

The interface board provides regulated 5v and 12 v supplies to itself and to the processing element.

4. Hardware

4.1 S100 Interface Hardware

Figure 4.1.1 shows the board layout for the S100 interface. This board has been constructed using wire-wrap. Socket numbers and pin assignments correspond to those on the main processing element board.

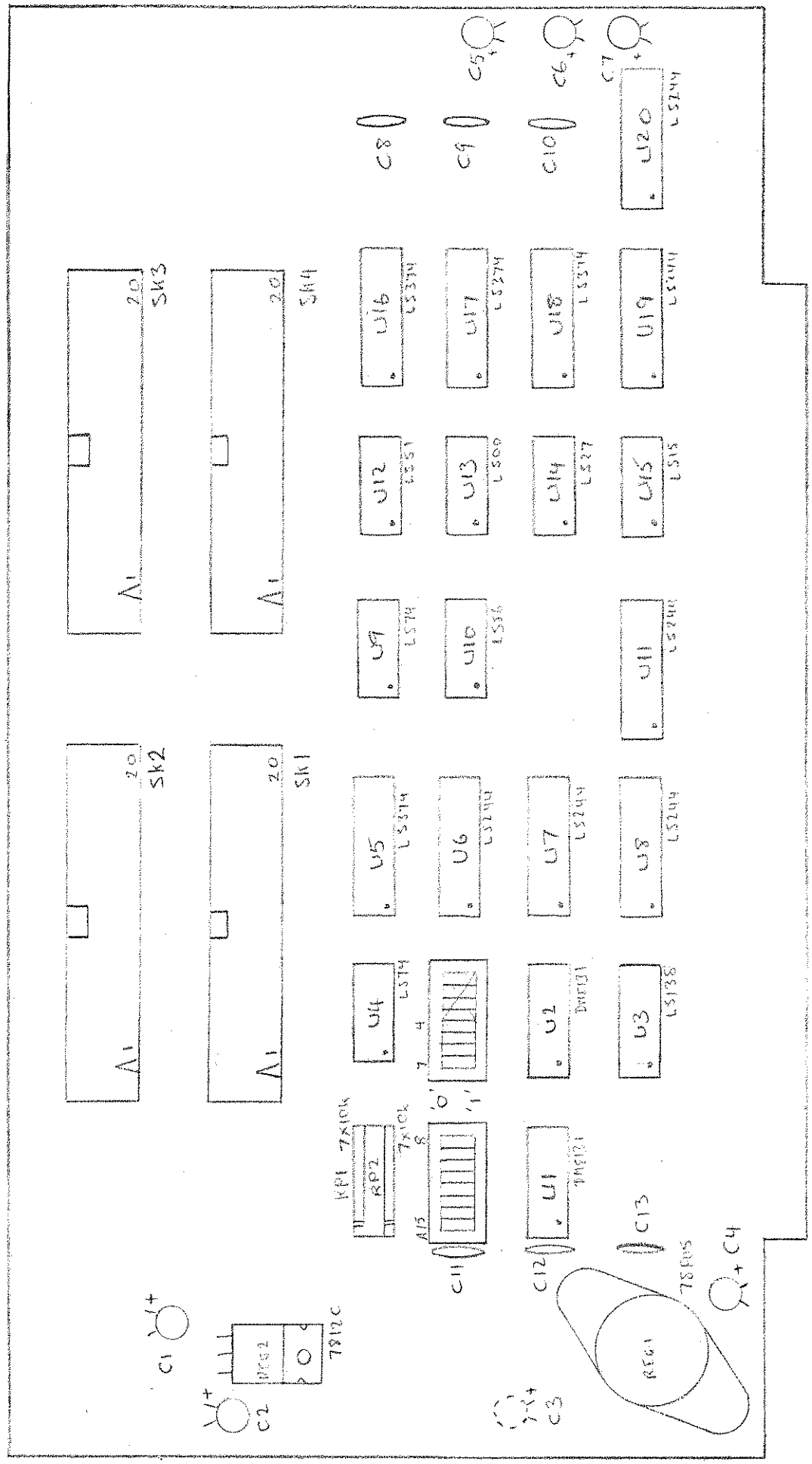
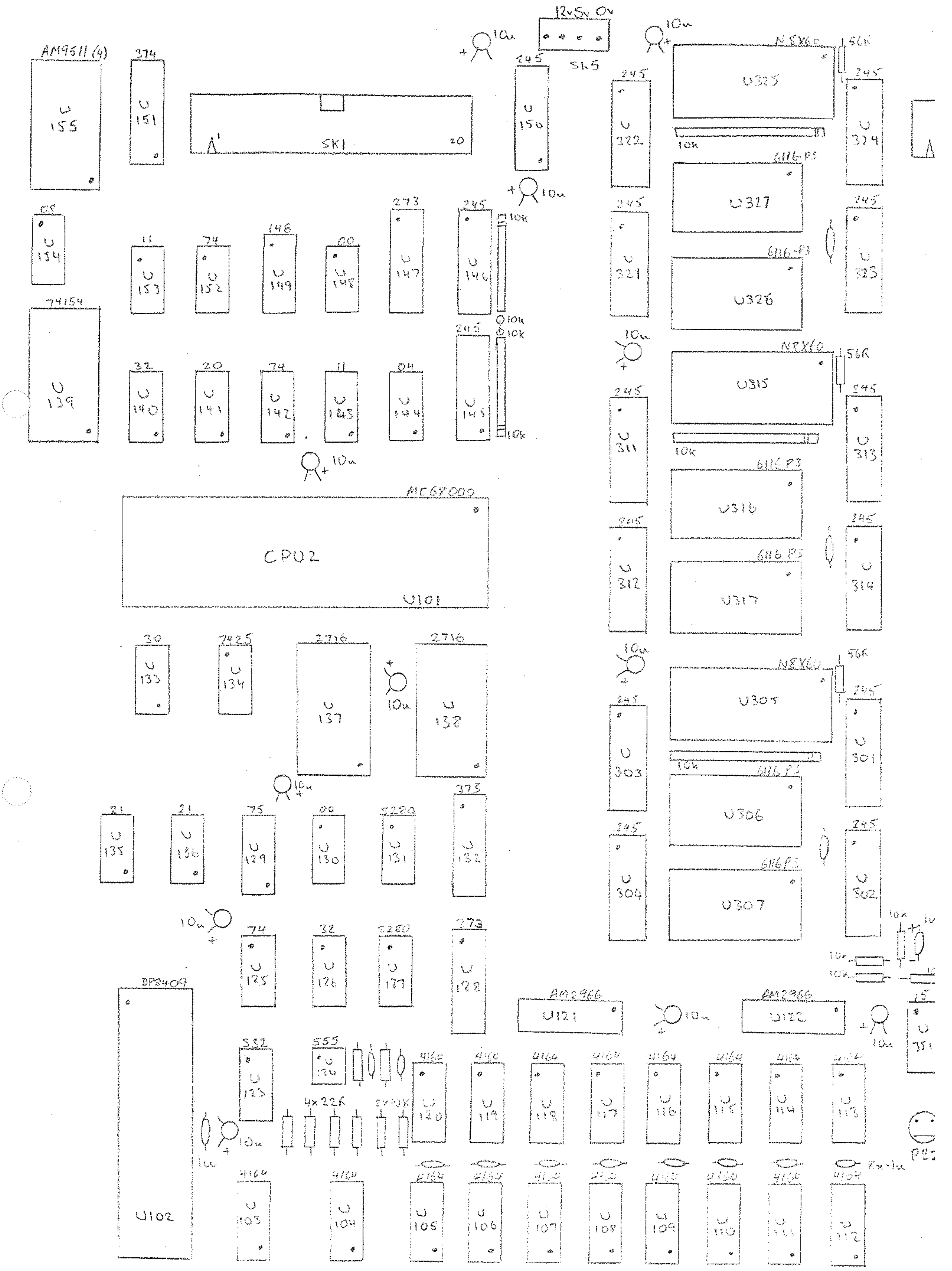


Fig. 4.1.1 S100 Interface Parts Placement



12v 5v 0v
SK5
SK1
AM9511 (4)
U 155
374
U 151
245
U 150
U 322
NEC
U325
U 324
U 154
U 153
U 152
U 149
U 148
273
U 147
U 146
6116-P3
U327
U 321
U 323
U326
74154
U 139
U 140
U 141
U 142
U 143
U 144
U 145
NEC
U315
U 311
U 313
U316
U 312
U317
U 314
MC68000
CPU2
U101
U 133
U 134
2716
U 137
U 138
U305
U 303
U 301
U306
U 135
U 136
U 129
U 130
U 131
373
U 132
U 304
U307
U 302
U 125
U 126
U 127
U 128
DP8409
U102
AM2966
U121
U122
U 123
555
4x22R
U 120
U 119
U 118
U 117
U 116
U 115
U 114
U 113
4164
U 103
U 104
U 105
U 106
U 107
U 108
U 109
U 110
U 111
U 112
56R
10k
10u

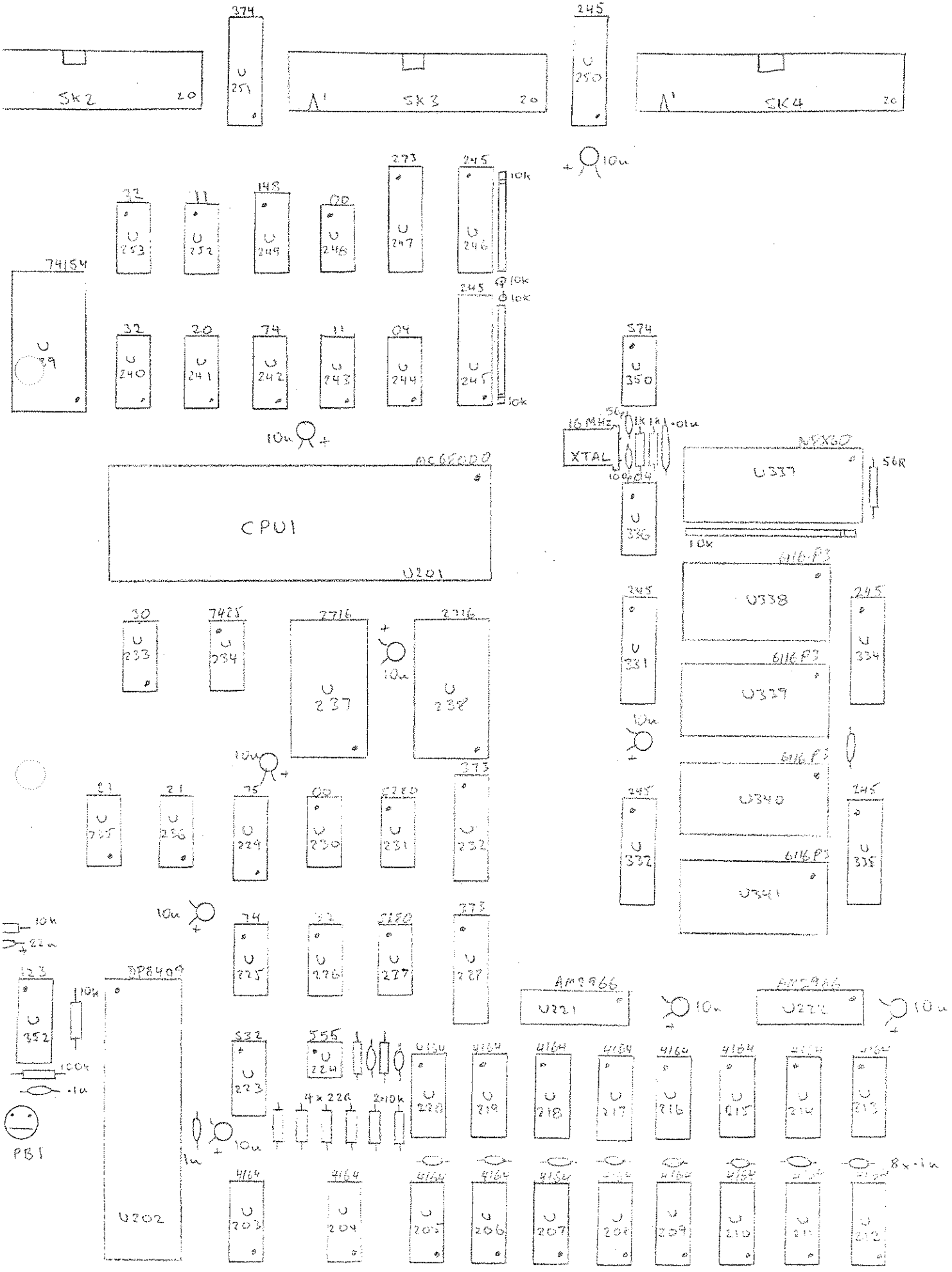


Figure 4.2 Main Board Layout

5.1 SOFTWARE MODEL.

The software is based upon a model of the element. The model is slightly different for both CPUs. The elements in the model are the CPU, memory eprom and the I/O ports.

5.1.1 CPU N° 1

INQ ; The input FIFO from the communications structure.

LOCALQ : the input FIFO from the 2nd CPU.

PIPE : the output fifo to the 2nd CPU.

REFRESH : WRITE address that causes a burst refresh of RAM.

STATOUT : output port.

- STATUS : output byte containing the current status of CPU.
- CONTROL : byte controlling the hardware -
  - reset the INQ FIFO
  - reset the PIPE FIFO
  - EPROM mapped in or out.
  - 4 bit interupt mask.

STATIN : input port.

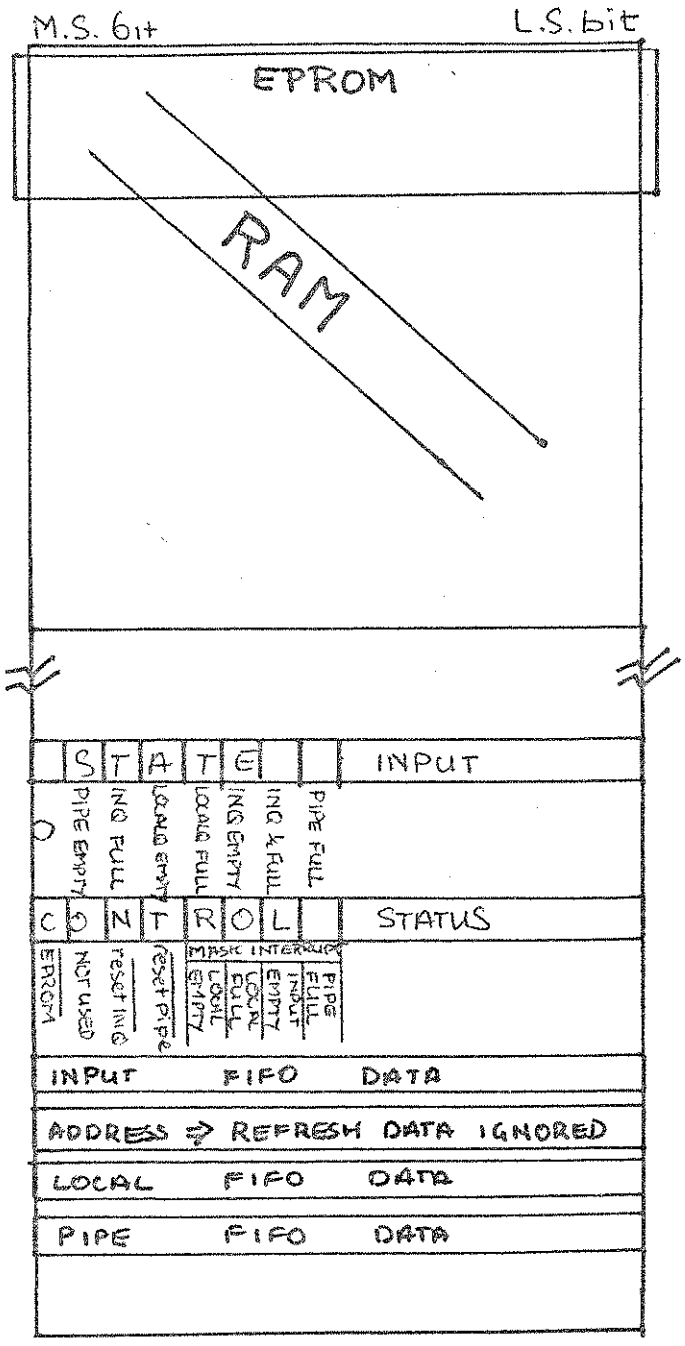
- INPUT : input byte to CPU.
- STATE : the state of the hardware . -
  - which FIFOs are empty
  - which FIFOs are full

The total system is interupt driven. Action taken by the CPU depends upon which interupts are active. Interupts from highest to lowest priority are: NMI

LOCALQ	empty	(maskable)
INQ	empty	(maskable)
PIPE	full	(maskable)
LOCALQ	full	(maskable)
INQ	full	
PIPE	empty	

CPU #1 MODEL

ADDRESS HEX	NAME
000000	RAM / EPROM
000FFF	END EPROM
01FFFF	END RAM
FFFF04 rd	STATIN
FFFF04 wr	STATOUT
FFFF0C rd	INQ
FFFF0C wr	REFRESH
FFFF0E rd	LOCALQ
FFFF0E wr	PIPE



5.1.2 CPU N°2

LOCALQ : the output FIFO to the first CPU.

PIPE : the input FIFO from the first CPU.

OUTQ : the output FIFO to the communications structure.

REFRESH : READ address that causes a burst refresh of RAM.

STATOUT : output port.

STATUS : output byte containing current status of CPU .

CONTROL : Byte controlling the hardware -

- SVACK to 9511
- reset 9511
- reset the OUTQ & LOCAL FIFOs
- EPROM mapped in or out
- 4 bit interrupt mask.

STATIN : input port.

INPUT : input byte to CPU

STATE : state of the hardware -

- which FIFOs are empty
- which FIFOs are full
- 9511 requests service
- 9511 completed operation

9511DATA : PUSH /POP data on 9511 stack

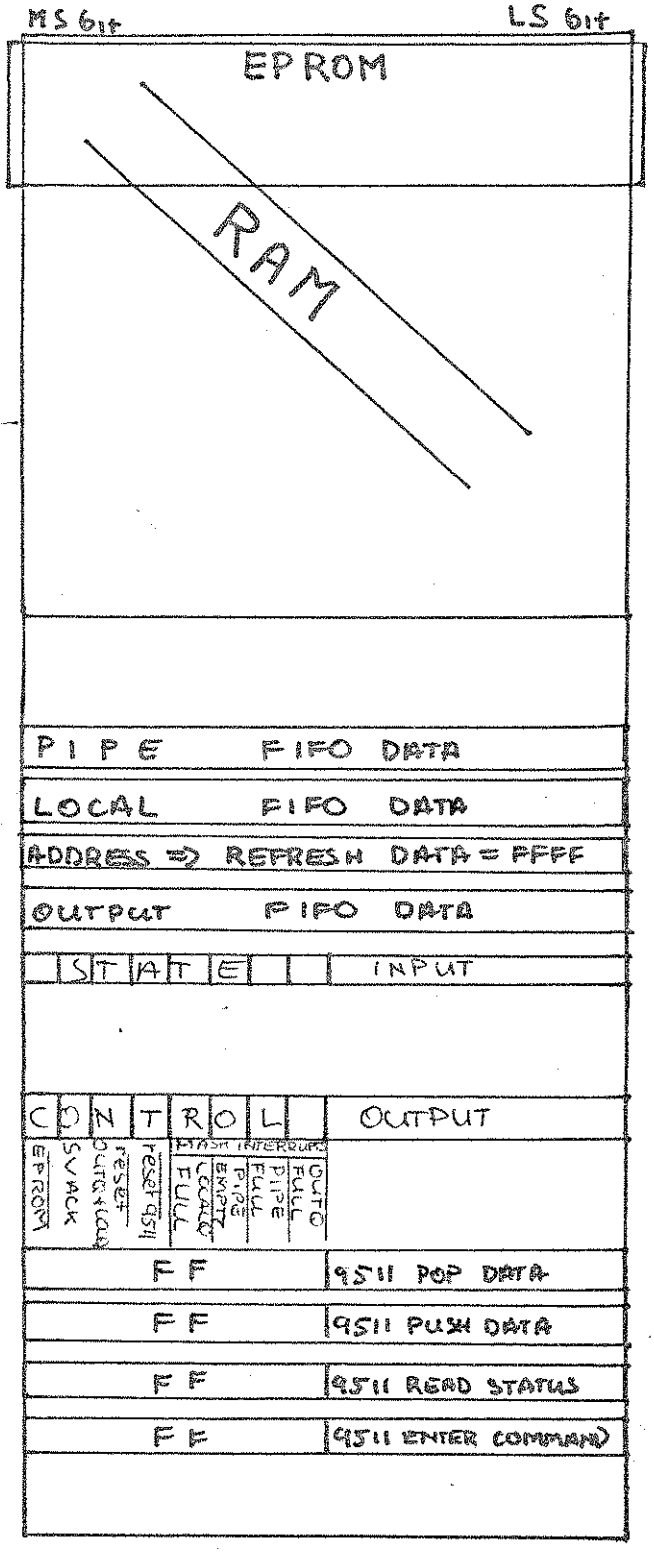
9511port : read status /enter command to 9511

Interrupts from highest to lowest priority are:

	NMI	
9511	request	
PIPE	full	(maskable)
PIPE	empty	(maskable)
OUTQ	full	(maskable)
LOCALQ	full	(maskable)
LOCALQ	empty	

CPU #2 MODEL

ADDRESS hex		NAME
000000		RAM/EPROM
000FFF		END EPROM
01FFFF		END RAM
FFFF00	rd	PIPE
FFFF00	wr	LOCAL@
FFFF02	rd	REFRESH
FFFF02	wr	OUT@
FFFF08	rd	STATIN
FFFF08	wr	STATOUT
FFFF0E	rd	9511 DATA
FFFF0E	wr	"
FFFF0F	rd	9511 PORT
FFFF0F	wr	"



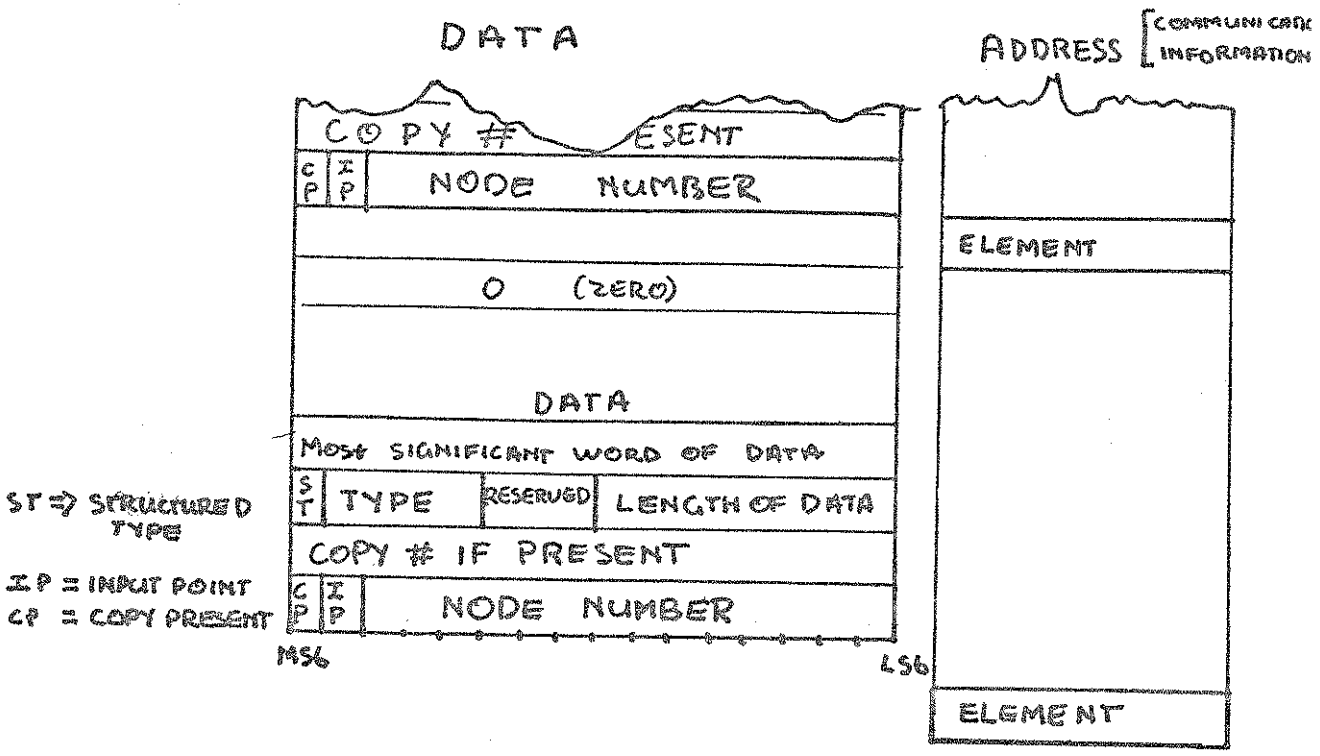
5.2 INTERRUPT STRUCTURE.

The 68000 provides for seven levels of interrupts. Of the seven one is NON-MASKABLE and causes an interrupt every time the line is asserted. The hardware is organised so that only seven interrupts can be produced. Each interrupt has its own level as indicated in the software model.The NONMASKABLE interrupt can be produced off the board , it is design to help with the handshaking during data exchange / interrogation. This interrupt is normally associated with the INPUT port.

Most causes of interrupts are external events. (The input queue empty for example) While the interrupt is being processed, it and any lower interrupt is masked by the 68000. To allow lower priority interupts to be effective, some interrupts are hardware maskable. This allows the 68000 to lower its priority again after recognising which interrupt conditions are present.

The software is designed so that the CPU tries to perform the main loop continuously. Interrupts break into this loop when this is not possible or a section needs attention. Using this procedure we minimize the amount of time spent on polling FIFO status, the program automatically proceeds with the most important task & the main loop need not consider what the hardware status is.

5.3 Token Format.



The format of the token begins with a 16 bit element number. This is used by the communications structure to address the correct element. Element numbers are sent to the output FIFO, but as they are "consumed" BY the communications structure, do not pass to input FIFOs.

The first word read in from FIFOs contains information about which node the token is destined for, which input of a 2 input node it is to enter and if a copy number is present.

If the token is going to a node in a sub-graph the copy number follows. The type and length of data proceeds the data.
To check the integrity of the token a ZERO word ends the token. This word is not counted when computing the length of the data. The ZERO word ensure that we have proper token boundaries in the fifos . A better technique is to send a 16 bit checksum, this is not incorporated in the program. (The reason is to keep the program simple at first to allow debugging. A zero word gives quick and easy test for token boundaries, which is useful when checking FIFOs manually.)

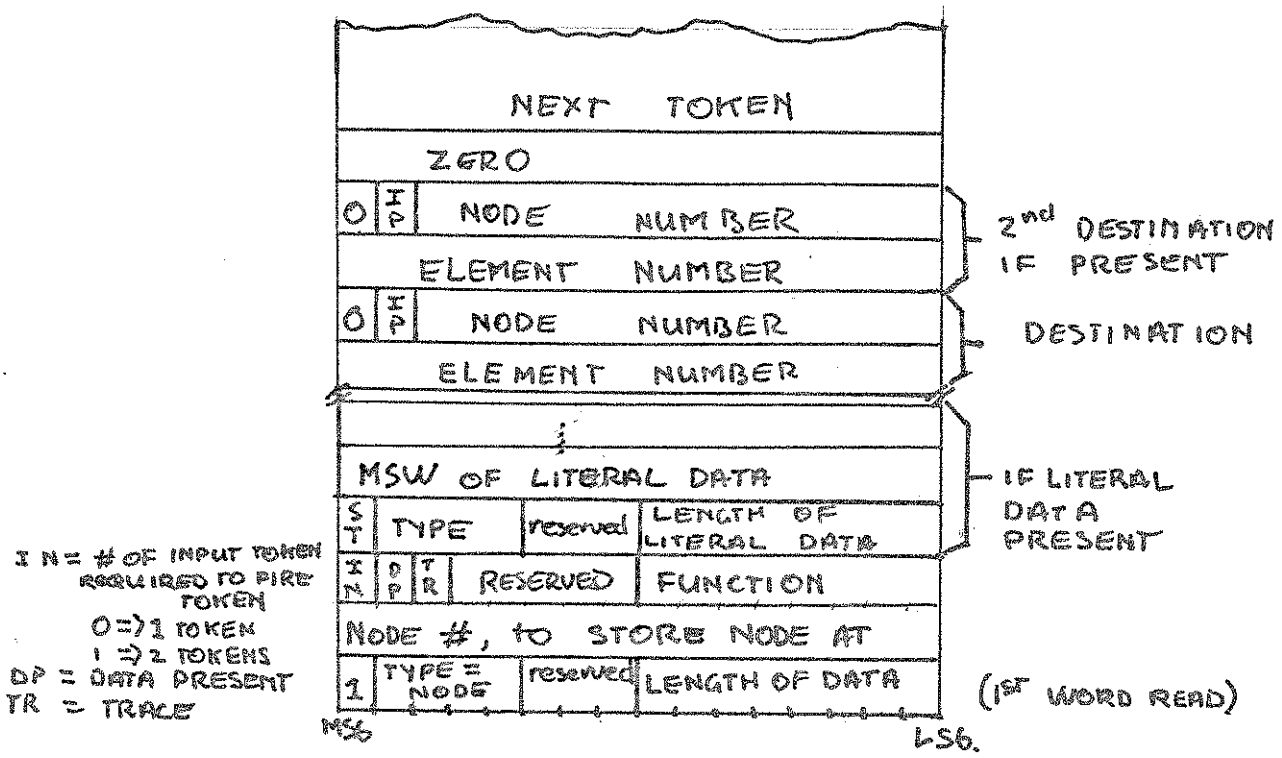
The following table outlines the types of data that may be present in a token.

TYPE	DESCRIPTION
00	real number (32 bits 9511 format)
01	integer, 2's complement 16 bits
02	integer, 2's complement 32 bits
03	bit string
04	character string
05	copy number
06	occurrence
07	link destination
08	destination
09	error (don't know)
10	stream
-	
14	node (a node description)
15	program (used for loading programs directly to RAM)

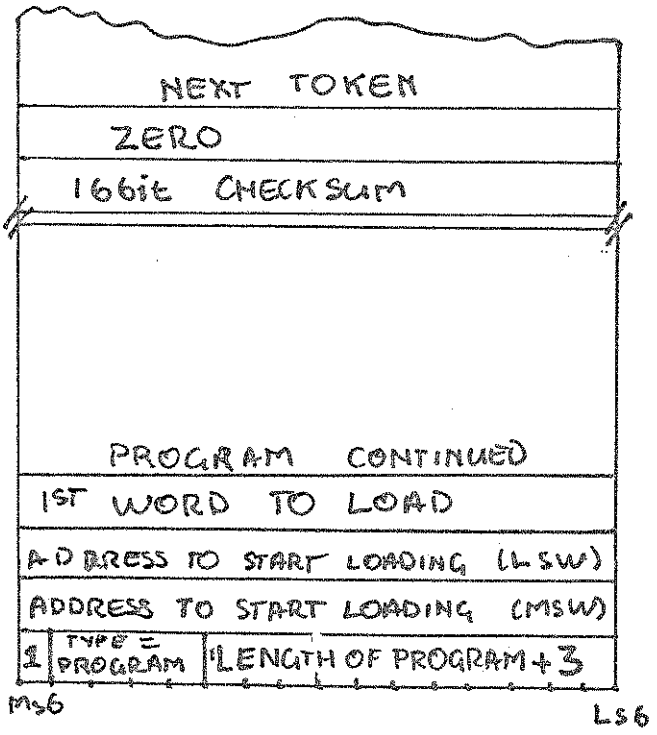
5.3.1 Special tokens.

Node descriptions and program type of data is structured as shown in the next diagram . The diagram shows the token from the type/ length of data word. Note that for the program node , that when the load address is equal to zero this means that the code is relocatable. Also the program node allows pogram lengths of up to 4kbytes, this in practice should be less than 4 kBytes to ensure that the WHOLE token will fit in the input FIFO.

NODE TOKEN



PROGRAM TOKEN



5.3.2 Types of Nodes.

The list of node functions is given below. They are presented in sequential order. i.e. type 0 = duplicate ; 1 = add

Duplicate	Add	Subtract	Multiply
DVD	DIV	NEG	ARG
RET	ENT	SWI	PIT
PIF	PIP	PRS	MOD
EXP	LNE	ABS	SIN
COS	TAN	ATN	
SQT	AND	OR	IMP
EQV	NQV	NOT	TSB
STB	CLB	EQ	LT
LE	GT	GE	NE
SUC	PRE	BRA	UNB
ORD	CHR	RND	TRN
FLT	STORAGE	CPT	DEC
D	STD	YLC	STC

For a more detailed description of the nodes refer to "FLO A Decentralised Data Flow System " ,G.K. Egan and C.P. richardson.

System Nodes.

Element nodes 16319 to 16383 are reserved for system functions. Nodes defined are:

16383 : Program store

16382 : Node store

16381 : Error node

5.4 CPU N^{o} 1 Task.

The flow chart for the first CPU is given on the next page. The main task of the CPU is to preprocess the tokens. Tokens are read in from either the LOCALQ or INQ (LOCALQ has priority) and passed through the input map section of the data flow machine. If the token is directed to a one input node, the token is passed directly to the second CPU. In the case of two input nodes, tokens are stored in a token store until tokens for each arc are present.

The input map organised as a list of 4 bytes in sequence with the node numbers. Because of this the node numbers should be generated sequentially by the compiler. Three of the four bytes contain a link address to the token store if the node has two inputs. The fourth byte contain information describing the node:

Node not present (has not been loaded)

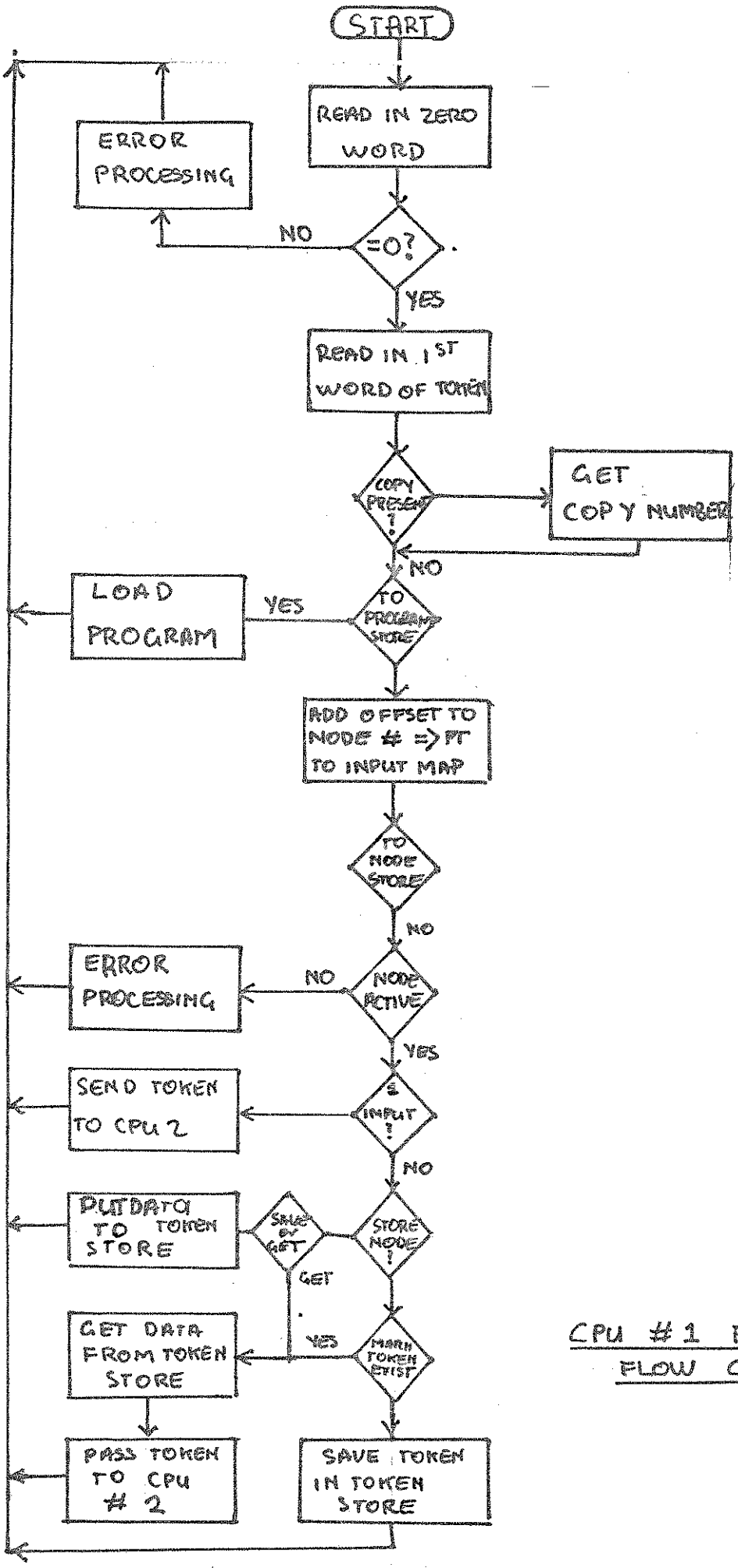
Node has one input

The node is the storage node.

2 input node and if any tokens are present.

There were three possible techniques to use to implement the token store, so that it would handle copy numbers correctly.

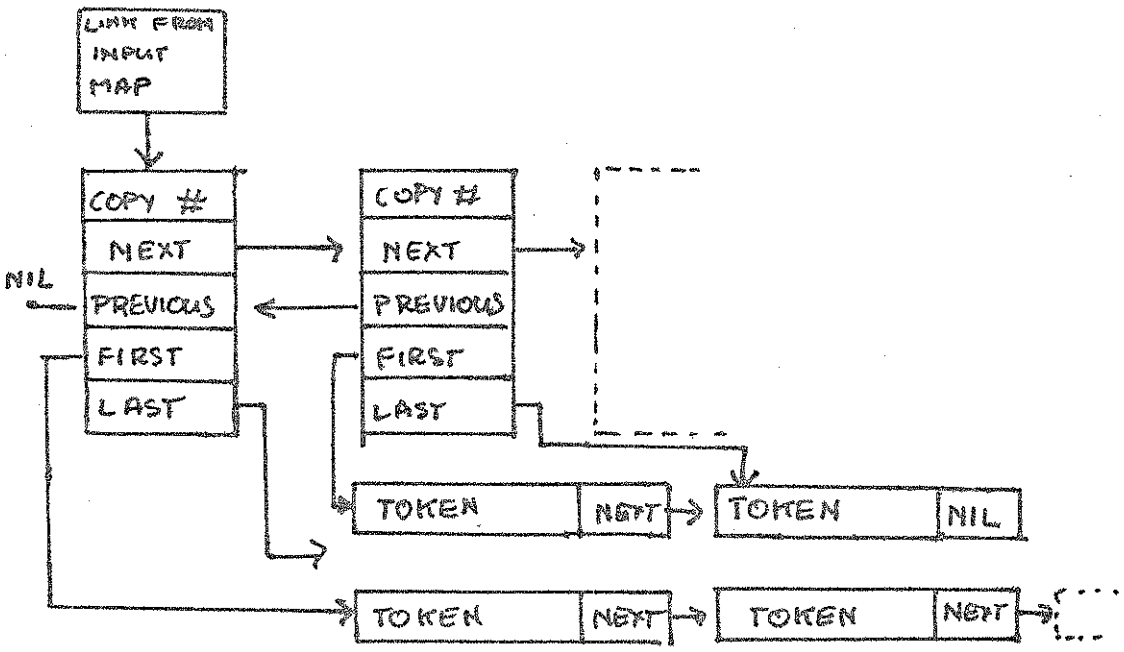
1) Use a hashing table to hash into a linked list of tokens with the same copy number
2) Use a binary tree or a partial binary tree to get to the desire copy number linked list.
3) Store all the tokens for the node as a linked list. When searching for a copy number simply run through the list until the first token with the desired copy number is found.



CPU # 1 FOREGROUND
FLOW CHART

Simulations on the data flow machine have shown that when copy numbers are involved that usually only one and maybe two tokens are queued. This makes the first two schemes uneconomical. The amount of storage required to maintain a hashing table or the binary tree will be much greater than the amount of storage required for the tokens. The execution time of the simple linear search will be comparable to the other methods when there are not many tokens to search through.

Finally a enhenced version of the linear search method was adopted. Instead of one linked list of all the tokens for a node, a doublely linked list of linked list of tokens with the same copy number and node destimations is used. The linked list of linked lists allows for quicker searching. It should be a good improvement, especially when many copy numbers are active (and so there would be a large number of tokens to search). The cost in terms of memory usage is minimal.



The doublely linked structure allows searching for the correct copy number to proceed in reverse. This will increase performance as the tokens most likely to be matched first are on the lowest level. (e.g. in the case of recursion the lowest graph is completed before the calling graphs are completed.

5.4.1 Background Tasks.

Besides performing the foreground task the CPU must handle a lot of background routines :

1) Manage the memory
2) Communicate with the other elements and outside inputs
3) Output its status for analysis
4) Provide error handling (e.g. for parity errors etc.).
5) Manage interrupts and other exceptions.
6) Manage the FIFOs
7) Storing tokens when FIFOs overflow.

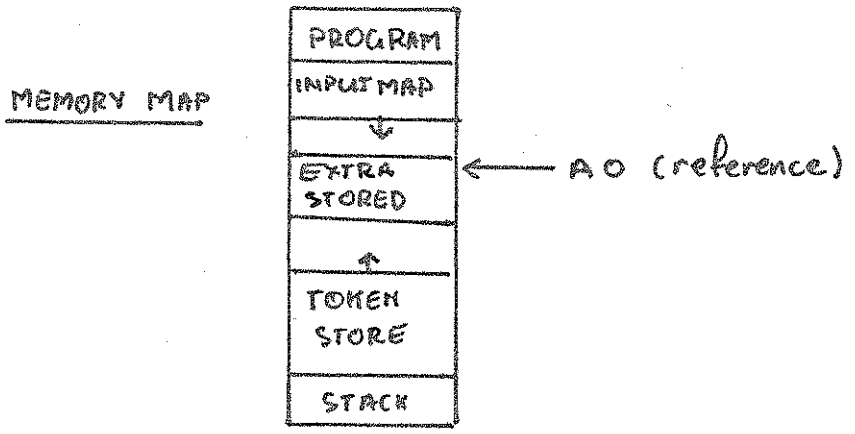
5.4.2 Memory Management.

Tokens stored in the token store are continously changing, this means that the memory requirements are changing. Storing and fetching of tokens occurs randomly and we get an increase in entropy in the memory structure. The difficulty is separating the useful information from the spent garbage. A large number of techniques were overviewed. Techniques involved heaps, garbage collection and compaction routines, as well as some more simple routines. A good compromise between efficient memory usage and fast execution was difficult to find.

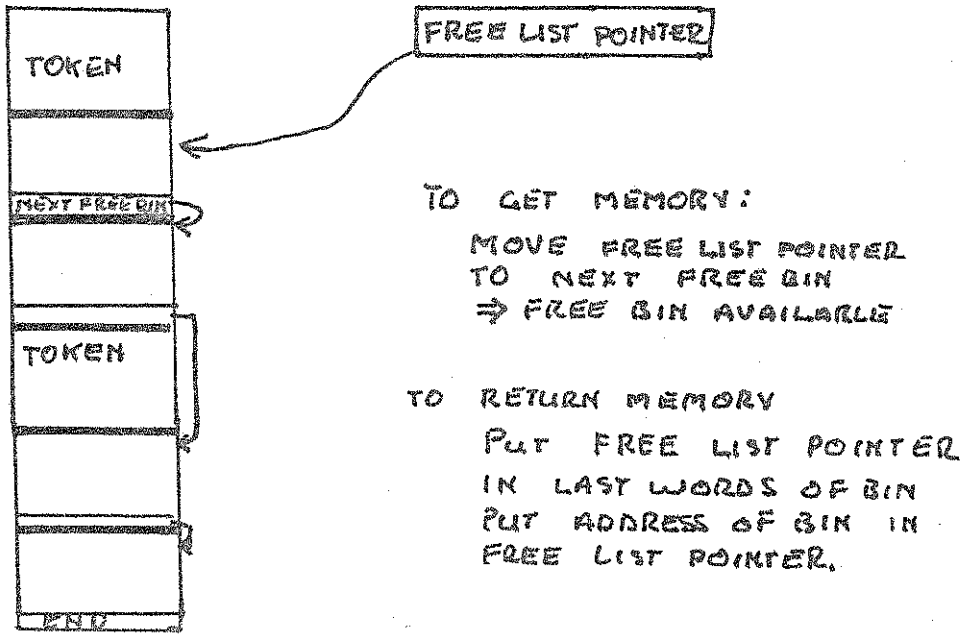
The technique decided upon is one were the memory is broken up into bins. When memory is required a bin, which are of fixed length, is taken. Useful information is always linked into the data structure. When storage is no longer required it is put on a free list. Bins on the free list are available to be used later. The free bins are managed by linking them all together. A free list pointer points to the first free bin which in turn should be linked to every other free bin.

The advantages of this system is that garbage collection is trivial (linking the spent bin to the list) Memory is always available. It is fast.
The major disadvantages is that if an error does occur anywhere is the data structure or in the linked list the result will be catastrophic.

Another memory problem is where to store tokens when FIFOs overflow. What happens is that space between the input map and token store is used for this purpose. When memory is require by either the input map or the token store, the memory area is relocated. Correct referencing to the memory is achieved by using indexed addressing and having the relocation stored in one of the registers.



TOKEN STORE.



5.5 CPU N^{o} 2 Tasks.

The second Cpu 's primary task is to execute the node function. The foreground routine is straight forward as indicated by the flow chart. The only complication arises in the node store. The aim is to achieve fast access of the node information without consuming masses of memory. A similar approach is used as that for the memory mangement problem. Node numbers are generated sequentially by the compliler. Initially nodes are given a fixed amount of memory in which to fit. The node description is accessed by a linear transformation on the node number.

Address = Node number X init. node length + offset.

If the node cannot fit into the initial amount of memory, the node is linked into an extension area where the remainder of the node is stored. This method gives extremely fast access of most nodes.

To make full use of this method the initial node length had to be carefully chosen. This was set to 10 bytes and covers a vast majority of the nodes.

2 Byte are for the node function

4 bytes are for the destination.

4 bytes are for a second destination (duplicate node is the most common)

or for 16 bits of literal data

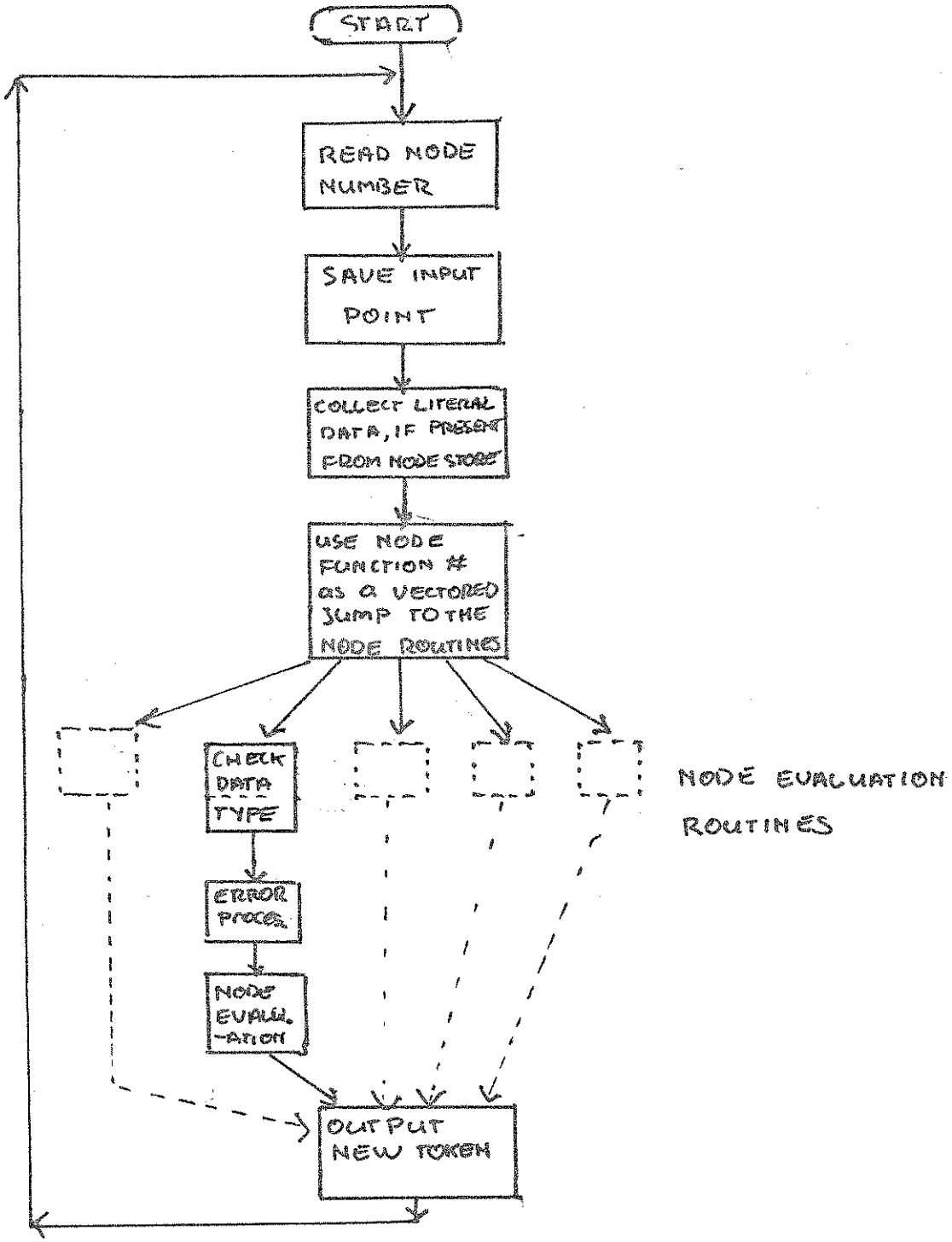
or for the link to the rest of the description.

Alternatives to this scheme where:

1) use node number as address to node store, limits the number of nodes available.

2) Use a translation table. This always requires one level of indirection to get to the node and uses a lot of memory for the translation table.

The background tasks for the cpu are similar to the first CPU.



CPU #2 FOREGROUND TASK

FLOW CHART

Signetics

8X60
March 1981

8X60 FIFO RAM Controller (FRC)

FIFO RAM CONTROLLER (FRC)

8X60

PRODUCT DESCRIPTION

The Signetics 8X60 FIFO RAM Controller (FRC) is an address and status generator designed to implement a high-speed/high-capacity First-In/First-Out (FIFO) stack utilizing standard off-the-shelf RAMs—see APPLICATIONS on the last page of this data sheet. The FRC can control up to 4096 words of buffer memory; intermediate buffer sizes can be selected—refer to the memory length table on the next page. Built-in arbitration logic handles read/write operations on a first-come/first-served basis.

As shown in Figure 1, the FRC consists of:

- A 12-Bit Write Address Generation Counter (Counter #1) and a 12-Bit Read Address Generation Counter (Counter #2).
- A 12-Bit Up/Down Status Counter (Counter #3).
- Twelve Three-State Address Drivers.
- Control Logic.

The two address counters, #1 and #2, respectively, are used to generate write and read addresses; the outputs of these counters are multiplexed to the three-state address drivers. Counter #3 generates *full*, *empty*, and *half full* status.

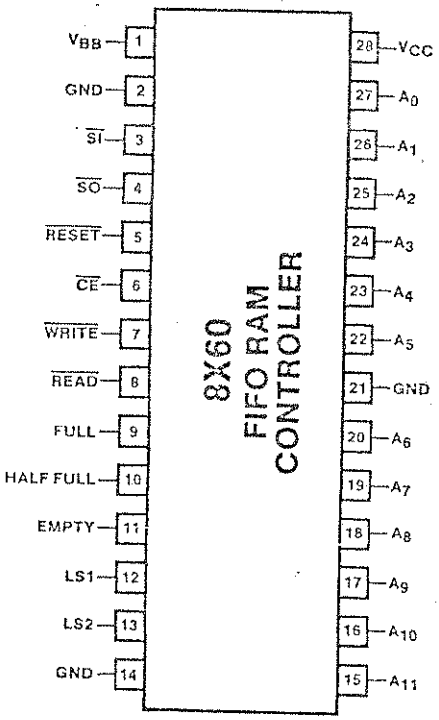
DESIGN FEATURES

- 12-Bit FIFO Address Generator
- Data Rate Exceeding 8MHz
- Asynchronous Read/Write Operations
- Three-State Address Outputs
- User-Defined Word Width
- Specifically Designed for Use with High-Speed Bipolar RAMs (Adaptable for Use with MOS RAMs)
- TTL Input and Output
- 16mA Address-Drive Capability

USE AND APPLICATION

- Interface Between Independently-Clocked Systems
- Buffer Memories for Disk and/or Tape
- Data Communication Concentrators
- CPU/Terminal Buffering
- DMA Applications
- CRT Terminals

PACKAGE AND PIN DESIGNATIONS



PIN NO.	IDENTIFIER	FUNCTION
1	V_{BB}	Supply voltage for internal circuits.
2, 14, 21	GND	Circuit ground.
3	$\overline{SI}$	Shift-In request for write cycle; active-low input.
4	$\overline{SO}$	Shift-Out request for read cycle; active-low input.
5	$\overline{RESET}$	Active-low master reset input.
6	$\overline{CE}$	Active-low chip enable input.
7	$\overline{WRITE}$	Write cycle address valid; active-low output.
8	$\overline{READ}$	Read cycle address valid; active-low output.
9	FULL	Memory full status output; also, override input capability. Active when *high*.
10	HALF FULL	Memory half-full status output; active-high.
11	EMPTY	Memory empty status output; also, override input capability. Active when *high*
12	LS1	Least significant bit (LSB) of the memory length select input.
13	LS2	Most significant bit (MSB) of the memory length select input.
15-20 22-27	A_{11}-A_0	Three-state address outputs; A_0 = LSB.
28	V_{CC}	Supply voltage.

FIFO RAM CONTROLLER (FRC)

8X60

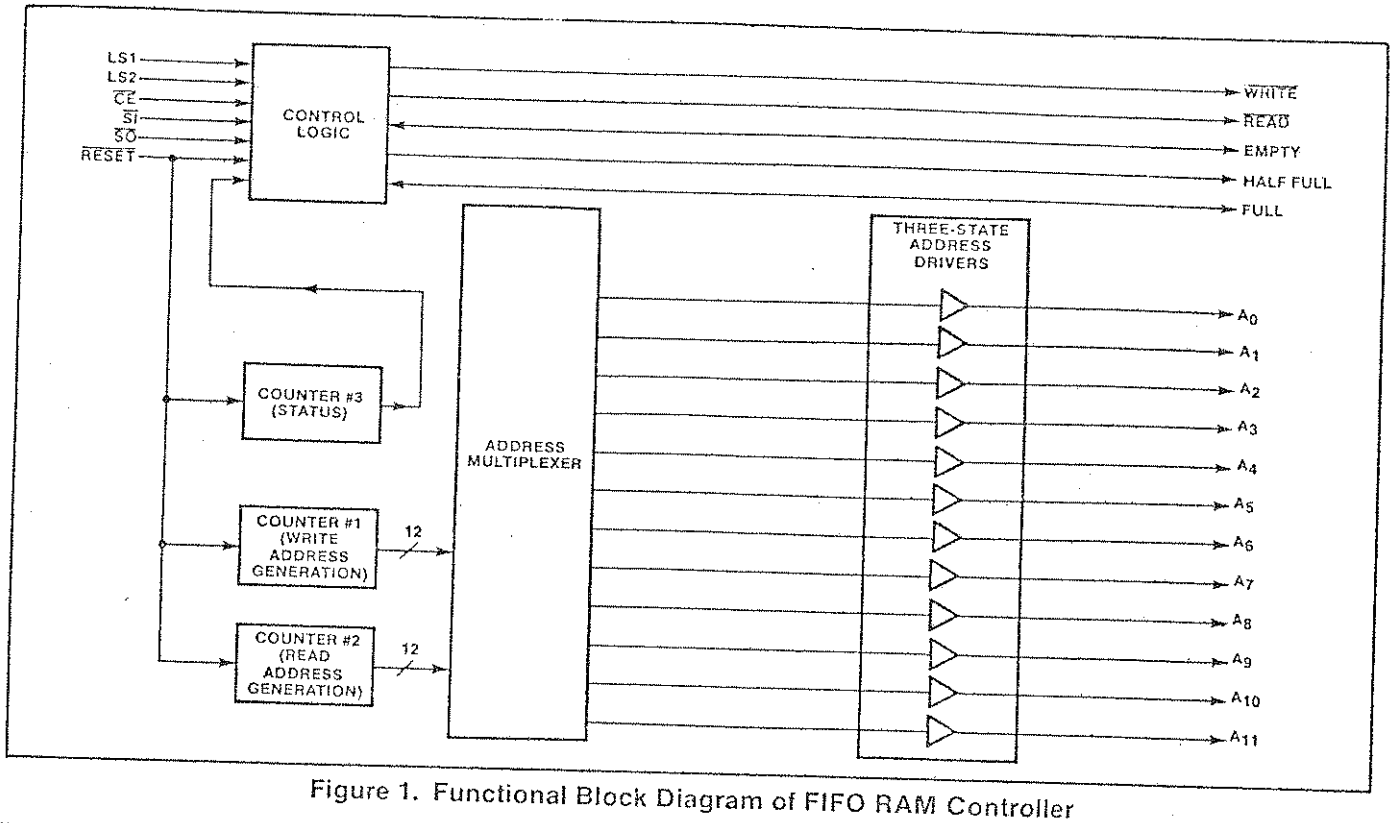


Figure 1. Functional Block Diagram of FIFO RAM Controller

FUNCTIONAL OPERATION

The FRC operates in either of two basic modes—*write* into the FIFO buffer memory or *read* from the FIFO buffer memory. These two operations are described in subsequent paragraphs and the complete sequence is summarized in Table 1. Typical Write/Read timing relationships, arbitration logic, and chip-enable control are shown in the TIMING DIAGRAMS.

FIFO BUFFER MEMORY—WRITE CYCLE

To perform a write operation, $\overline{SO}$ must be *high* and $\overline{SI}$ must be *low*. When these conditions exist and other control parameters (Table 1) are satisfied, the write address in Counter #1 (Figure 1) is output to the address bus via the multiplexer and the $\overline{WRITE}$ output goes *low*. (*Note*. Normally, the $\overline{WRITE}$ output goes *low* after the address output becomes stable—refer to *WRITE CYCLE TIMING DIAGRAM*. The $\overline{WRITE}$ output may then act as a *write* or *chip* enable for the RAMs that are used to implement the memory.)

When the *write* cycle is ended ($\overline{SI}$ is forced high), the $\overline{WRITE}$ output goes *high*, the address output buffers return to a high-impedance state. Counter #1 (Write Address Generation) and Counter #3 (Status) are both incremented, and Counter #2 (Read Address Generation) remains unchanged.

FIFO BUFFER MEMORY—READ CYCLE

To perform a read operation, $\overline{SI}$ must be *high* and $\overline{SO}$ must be *low*. When these conditions exist and other control parameters (Table 1) are satisfied, the read address contained in Counter #2 (Figure 1) is output to the address bus and the $\overline{READ}$ output goes *low*.

When the *read* cycle is ended ($\overline{SO}$ is forced *high*), the $\overline{READ}$ output goes *high*, the output buffers return to a high-impedance state, Counter #2 (Read Address Generation) is incremented, Counter #3 (Status) is decremented, and Counter #1 (Write Address Generation) remains unchanged.

CONTROL LOGIC

To prevent the possibility of operational conflicts, $\overline{SI}$ and $\overline{SO}$ are treated on a first-come/first-served basis; these two input signals are controlled by internal arbitration logic—refer to the applicable TIMING DIAGRAMS and AC CHARACTERISTICS for functional and timing relationships. If one cycle is requested while the other cycle is in progress, the requested cycle will commence as soon as the current cycle is complete (provided other control parameters are satisfied).

As shown in the accompanying diagram, the buffer length of the FIFO memory can be hardware-selected via the Length Select (LS1, LS2) inputs. When less than the maximum length is selected, the unused high-order bits of the address outputs are held in the high-impedance state.

MEMORY LENGTH

LS1	LS2	HALF LENGTH	FULL LENGTH
L	L	2048	4096
H	L	32	64
L	H	512	1024
H	H	128	256

Generation of the status output signals (HALF FULL, FULL, and EMPTY) is a function of the Length Select (LS1, LS2) inputs and the current state of Status Counter #3. In general, the status outputs reflect the conditions that follow:

- **HALF FULL**—this status output signal goes *high* on the positive-going edge of $\overline{SI}$ if the MSB of the selected length of Counter #3 becomes a "1". The HALF FULL signal will go from *high*-to-*low* on the positive-going edge of $\overline{SO}$ when, after the *read* cycle, the selected length of Counter #3 changes from "100...00" to "011...11". For example, if the selected memory length is 256 words (FULL = 256), then HALF FULL = 128 words; hence, on the

positive-going edge of $\overline{SO}$ when Counter #3 reaches a count of 127, the HALF FULL output will go from *high*-to-*low*.

- **FULL**—this signal serves both as a status output and as an override input. The FULL signal goes *high* on the negative-going edge of $\overline{SI}$ if all bits of Counter #3 for selected length are equal to "1". The FULL output goes from *high*-to-*low* on the negative-going edge of $\overline{SO}$.
- **EMPTY**—this signal also serves as a status output and as an override input. On the negative-going edge of $\overline{SO}$, the EMPTY output is driven *high* if Status Counter #3 contains a value of "1"; on the positive-going edge of $\overline{SO}$, the counter is decremented to "0". The EMPTY output goes from *high*-to-*low* on the negative-going edge of $\overline{SI}$.

Once the FULL signal is *high*, further Write Cycle Requests ($\overline{SI}$ = low) are ignored; similarly, once the EMPTY signal is *high*, further Read Cycle requests ($\overline{SO}$ = low) are ignored. However, to accommodate diversified applications, the FULL and EMPTY outputs are open-collector with on-chip 4.7K passive pull-up resistors. If either the FULL or EMPTY pins are forced *low* via external control, the corresponding *write* or *read* cycle may resume (provided the external FULL or EMPTY input is held *low* until the corresponding $\overline{WRITE}$ or $\overline{READ}$ output goes *low*) and the address/status counters will continue normal operation*—refer to Table 1.

The user must force the $\overline{RESET}$ input *low* to initialize the chip. (*Note.* If the $\overline{RESET}$ signal is driven *low* during a *write* or *read* cycle, the address output may have a short period of uncertainty before assuming a high-impedance state.) The following actions occur when $\overline{RESET}$ is active:

- All internal counters are set to "0".
- All address output lines are forced to the high-impedance state.
- HALF FULL and FULL outputs are forced *low*.
- $\overline{WRITE}$, $\overline{READ}$, and EMPTY outputs are forced high.

When $\overline{CE}$ is *high*, the address output lines are forced to the high-impedance state, further *write* or *read* cycle requests are ignored, and all counters remain unchanged. If $\overline{CE}$ switches from *low*-to-*high* during a *write* or *read* cycle, the cycle in progress is always completed before the disabled state is entered. For details of these operations, refer to the timing information shown later in this data sheet.

*Refer to Note on inside back cover

Table 1. Summary of Operation

INPUTS				INITIAL CONDITIONS	RESULTING OUTPUTS			COMMENTS
$\overline{RESET}$	$\overline{CE}$	$\overline{SI}$	$\overline{SO}$		$\overline{WRITE}$	$\overline{READ}$	ADDRESS BUS	
L	X	X	X		H	H	Hi-Z	Reset all counters to 0.
H	X	H	H		H	H	Hi-Z	No action
H	L	L	H	FULL = L	L	H	Write address from Ctr #1	Shift into FIFO stack (Write Cycle)
H	L	L	H	FULL = H	H	H	Hi-Z	Stack full (Write inhibited)
H	L	H	L	EMPTY = L	H	L	Read address from Ctr #2	Shift out of FIFO stack (Read Cycle)
H	L	H	L	EMPTY = H	H	H	Hi-Z	Stack empty (Read inhibited)
H	L	L	↓	Write cycle in progress	L	H	Write address from Ctr #1	Continue write cycle (until $\overline{SI}$ goes high)
H	L	↓	L	Read cycle in progress	H	L	Read address from Ctr #2	Continue read cycle (until $\overline{SO}$ goes high)
H	L	L	L	EMPTY = H	L	H	Write address from Ctr #1	Shift in (Read inhibited)
H	L	L	L	FULL = H	H	L	Read address from Ctr #2	Shift out (Write inhibited)
H	L	↑	H	Write cycle in progress	↑	H	Goes to Hi-Z	Increment write address counter #1 and status counter #3
H	L	H	↑	Read cycle in progress	H	↑	Goes to Hi-Z	Increment read address counter #2; decrement status counter #3
H	L	↑	L	Write cycle in progress (Note 1)	↑	↓	Changes to read address from Ctr #2	Increment write address counter #1 and status counter #3
H	L	L	↑	Read cycle in progress (Note 2)	↓	↑	Changes to write address from Ctr #1	Increment read address counter #2; decrement status counter #3
H	H	↓	H		H	H	Hi-Z	Chip disabled
H	H	H	↓		H	H	Hi-Z	Chip disabled
H	↑	L	X	FULL = L; write cycle begun (Note 1)	L	H	Write address from Ctr #1	Continue write cycle (until $\overline{SI}$ goes high)
H	↑	X	L	EMPTY = L; read cycle begun (Note 2)	H	L	Read address from Ctr #2	Continue read cycle (until $\overline{SO}$ goes high)
H	↓	L	L	FULL = L; EMPTY = L	—	—	—	This set of conditions should be avoided

NOTES

1. Write cycle will occur if either $\overline{SI}$ goes *low* before $\overline{SO}$ goes *low* or EMPTY = H when $\overline{SO}$ goes *low*.
2. Read cycle will occur if either $\overline{SO}$ goes *low* before $\overline{SI}$ goes *low* or FULL = H when $\overline{SI}$ goes

FIFO RAM CONTROLLER (FRC)

8X60

DC ELECTRICAL CHARACTERISTICS

CONDITIONS:

Commercial—
V_{CC} = 5.0V (±5%)
V_{BB} = 1.5V (±5%)[1]
0°C≤T_A≤70°C

Military—
V_{CC} = 5.0V (±10%)
V_{BB} = 1.5V (±10%)[1]
-55°C≤T_C≤125°C

ABSOLUTE MAXIMUM RATINGS

PARAMETER	DESCRIPTION	RATING	UNIT
V_{CC}	Power Supply Voltage	+7	Vdc
V_{BB}	Supply Voltage for Internal Circuits	+4	Vdc
V_{IN}	Input Voltage	+5.5	Vdc
V_O	Off-State Output Voltage	+5.5	Vdc
T_{STG}	Storage Temperature Range	-65 to +150	°C

PARAMETER	DESCRIPTION	TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNITS
			MIN	TYP[2]	MAX	MIN	TYP[2]	MAX	
V_{IH}	High level input voltage	Note 3	2.0						V
V_{IL}	Low level input voltage				0.8				V
V_{OH}	High level output voltage: All outputs except FULL and EMPTY	V_{CC} = Min; I_{OH} = -2.6mA	2.7	3.5					V
V_{OL}	Low level output voltage: Address Bus, $\overline{WRITE}$, $\overline{READ}$	V_{CC} = Min; I_{OL} = 16mA		0.38	0.5				V
	HALF FULL, FULL, and EMPTY	V_{CC} = Min; I_{OL} = 8mA		0.35	0.5				V
V_{CD}	Diode clamp voltage: All inputs except FULL and EMPTY	V_{CC} = Min; I_{CD} = -18mA	-1.5	-0.8					V
I_{IH}	High level input current: All inputs except FULL and EMPTY	V_{CC} = Max; V_{IH} = 2.7V		0.1	20				µA
	FULL and EMPTY	V_{CC} = Max; V_{IH} = 2.7V; Stack FULL or Stack EMPTY (Note 3)		-470	-750				µA
I_{IL}	Low level input current: All inputs except FULL and EMPTY	V_{CC} = Max; V_{IL} = 0.4V		-0.17	-0.4				mA
	FULL and EMPTY	V_{CC} = Max; V_{IL} = 0.4V; Stack FULL or Stack EMPTY		-1.12	-1.8				mA
I_{OH}	High level output current-FULL, EMPTY	V_{CC} = Min; V_{OH} = V_{CC} (min)		15	100				µA
I_{OZH}	High-Z output current (HIGH); Address Bus (Three-State)	V_{CC} = Max; V_{OUT} = 2.4V		0.9	20				µA
I_{OZL}	High-Z output current (LOW); Address Bus (Three-State)	V_{CC} = Max; V_{OUT} = 0.5V		0.6	-20				µA
I_I	Input leakage current; All inputs except FULL and EMPTY	V_{CC} = Max; V_{IN} = 5.5V		0.03	0.1				mA
I_{OS}	Short-circuit output current Address Bus and HALF FULL	V_{CC} = Max; V_{OH} = 0V	-15	-68	-100				mA
	$\overline{WRITE}$, $\overline{READ}$	V_{CC} = Max; V_{OH} = 0V	-40	-73	-100				mA
I_{CC}	Supply current from V_{CC}	V_{CC} = Max; Address Bus = High-Z		81	140				mA
I_{BB}	Supply current from V_{BB}	V_{BB} = Max		63	85				mA

NOTES

1. V_{BB} can be obtained from a regulated 1.5V supply, alternately, proper supply current (I_{BB}) can be obtained by connecting a 56-ohm (±5%, 0.5 W) resistor in series with V_{CC} as shown later in the APPLICATIONS diagram.
2. Typical limits are V_{CC} = 5.0V, T_A = 25°C.
3. Because of the internal pull-up resistor on the FULL and EMPTY pins, a negative current is required to force the required voltage.

FIFO RAM CONTROLLER (FRC) 8X60

AC ELECTRICAL CHARACTERISTICS

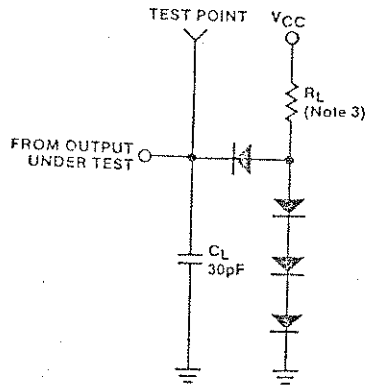
CONDITIONS: Commercial— V_{CC} = 5.0V (±5%), V_{BB} = 1.5V (±5%), 0°C≤T_A≤70°C; Military— V_{CC} = 5.0V (±10%), V_{BB} = 1.5V (±10%), -55°C≤T_C≤125°C; Loading—See TEST LOADING CIRCUITS

PARAMETERS	REFERENCES FROM	REFERENCES TO	TEST CONDITIONS	LIMITS (COMMERCIAL) MIN	TYP	MAX	LIMITS (MILITARY) MIN	TYP	MAX	UNITS
PULSE WIDTHS										
T_{LL} $\overline{SI}$ low	↓$\overline{SI}$	↑$\overline{SI}$	FULL = Low (override inhibit)	45	27					ns
T_{LH} $\overline{SI}$ high	↑$\overline{SI}$	↓$\overline{SI}$	Stack approaching FULL (Note 1)	25	13					ns
T_{DL} $\overline{SO}$ low	↓$\overline{SO}$	↑$\overline{SO}$	EMPTY = Low (override inhibit)	45	24					ns
T_{DH} $\overline{SO}$ high	↑$\overline{SO}$	↓$\overline{SO}$	Stack approaching EMPTY (Note 1)	30	16					ns
WRITE CYCLE TIMING										
T_{LA} Address stable delay	↓$\overline{SI}$	An	FULL = Low; $\overline{SO}$ = High		40	50				ns
T_{AW} Address lead time	An	↓$\overline{WRITE}$		5	11					ns
T_{LAW} $\overline{WRITE}$ output active delay	↓$\overline{SI}$	↓$\overline{WRITE}$	FULL = Low; $\overline{SO}$ = High	30	51	70				ns
T_{LW} $\overline{WRITE}$ output inactive delay	↑$\overline{SI}$	↑$\overline{WRITE}$			3	10				ns
T_{WA} Address lag time	↑$\overline{WRITE}$	An		25	34					ns
T_{LT} Address output disable	↑$\overline{SI}$	An(Hi-Z)			37	50				ns
T_{LF} FULL status active delay	↓$\overline{SI}$	↑FULL	Stack approaching FULL; $\overline{SO}$ = High		39	55				ns
T_{LE} EMPTY status inactive delay	↓$\overline{SI}$	↓EMPTY	Stack = EMPTY		40	65				ns
T_{HFH} HALF-FULL status active delay	↑$\overline{SI}$	↑HALF FULL	Stack approaching HALF FULL		30	45				ns
T_{DW} $\overline{WRITE}$ output active after read	↑$\overline{SO}$	↓$\overline{WRITE}$	Both $\overline{SI}$ & $\overline{READ}$ = Low		74	100				ns
READ CYCLE TIMING										
T_{DA} Address stable delay	↓$\overline{SO}$	An	EMPTY = Low; $\overline{SI}$ = High		40	50				ns
T_{AR} Address lead time	An	↓$\overline{READ}$		0	9					ns
T_{DAR} $\overline{READ}$ output active delay	↓$\overline{SO}$	↓$\overline{READ}$	EMPTY = Low; $\overline{SI}$ = High	30	48	70				ns
T_{DR} $\overline{READ}$ output inactive delay	↑$\overline{SO}$	↑$\overline{READ}$			5	10				ns
T_{RA} Address lag time	↑$\overline{READ}$	An		25	32					ns
T_{DT} Address output disable	↑$\overline{SO}$	An(Hi-Z)			37	50				ns
T_{DE} EMPTY status active delay	↓$\overline{SO}$	↑EMPTY	Stack approaching EMPTY; $\overline{SI}$=High		38	55				ns
T_{DF} FULL status inactive delay	↓$\overline{SO}$	↓FULL	Stack =FULL		38	55				ns
T_{HFL} HALF-FULL status inactive delay	↑$\overline{SO}$	↓HALF FULL	Stack exactly HALF FULL		54	75				ns
T_{LR} $\overline{READ}$ output active after write	↑$\overline{SI}$	↓$\overline{READ}$	Both $\overline{SO}$ & $\overline{WRITE}$ = Low		70	95				ns
CHIP ENABLE TIMING (WRITE)										
T_{HEW} Chip enable hold time[2]	↓$\overline{SI}$	↑$\overline{CE}$	FULL = Low; $\overline{SO}$ = High	10	1					ns
T_{SEW} Chip disable setup time[3]	↑$\overline{CE}$	↓$\overline{SI}$	FULL = Low; $\overline{SO}$ = High	10	1					ns
T_{PEW} Chip enable delay time	↓$\overline{CE}$	↓$\overline{WRITE}$	FULL = Low; $\overline{SI}$ = Low; $\overline{SO}$ = High		69	95				ns
CHIP ENABLE TIMING (READ)										
T_{HER} Chip enable hold time[2]	↓$\overline{SO}$	↑$\overline{CE}$	EMPTY = Low; $\overline{SI}$ = High	10	1					ns
T_{SER} Chip disable setup time[3]	↑$\overline{CE}$	↓$\overline{SO}$	EMPTY = Low; $\overline{SI}$ = High	10	1					ns
T_{PER} Chip enable delay time	↓$\overline{CE}$	↓$\overline{READ}$	EMPTY = Low; $\overline{SO}$ = Low; $\overline{SI}$ = High		64	95				ns
RESET TIMING										
T_{RR} $\overline{RESET}$ recovery	↑$\overline{RESET}$	↓$\overline{WRITE}$	$\overline{SI}$ = Low		57	80				ns
T_{RL} $\overline{RESET}$ pulse width (low)	↓$\overline{RESET}$	↑$\overline{RESET}$		15	8					ns
FULL/EMPTY OVERRIDE TIMING:										
T_{FW} Override Recovery for FULL	↓FULL	↓$\overline{WRITE}$	Stack = FULL; $\overline{SI}$ = Low; $\overline{SO}$ = High		70	100				ns
T_{ER} Override Recovery for EMPTY	↓EMPTY	↓$\overline{READ}$	Stack = EMPTY; $\overline{SO}$ = Low; $\overline{SI}$ = High		65	95				ns

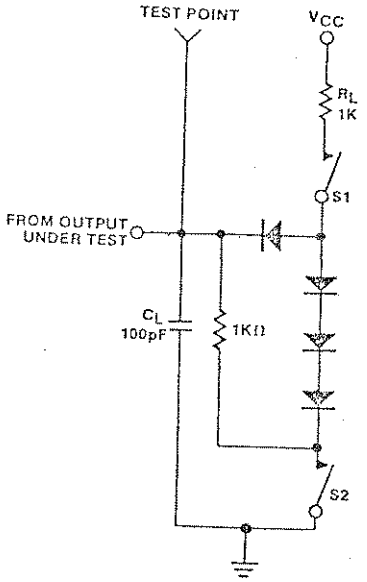
NOTES
1. Such that write/read request is inhibited after stack becomes full/empty.
2. The earliest rising edge of $\overline{CE}$ such that the $\overline{WRITE}$ or $\overline{READ}$ output always occurs.
3. The latest rising edge of $\overline{CE}$ such that the $\overline{WRITE}$ or $\overline{READ}$ output never occurs.

TEST LOADING CIRCUITS

APPLICABLE PINS: $\overline{WRITE}$ (7), $\overline{READ}$ (8), HALF FULL (10)

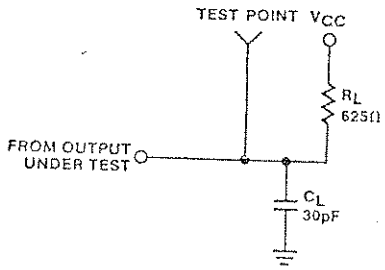


APPLICABLE PINS: A_n (15-20, 22-27)



OUTPUT STATE		SWITCH POSITION	
FROM	TO	S1	S2
Low	High	Closed	Closed
High	Low	Closed	Closed
High	HI-Z	Closed	Closed
Low	HI-Z	Closed	Closed
HI-Z	High	Open	Closed
HI-Z	Low	Closed	Open

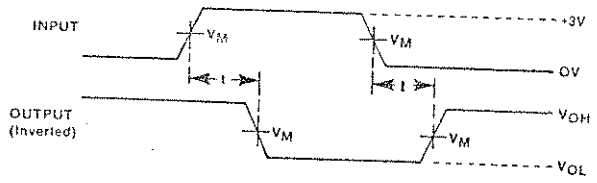
APPLICABLE PINS: FULL (9) AND EMPTY (11)



NOTES
1. In all cases C_L includes probe and jig capacitance.
2. All diodes are 1N916, 1N3064, or equivalent.
3. For $\overline{READ}$ and $\overline{WRITE}$ outputs, R_L = 280 ohms; for HALF FULL output, R_L = 2K ohms.

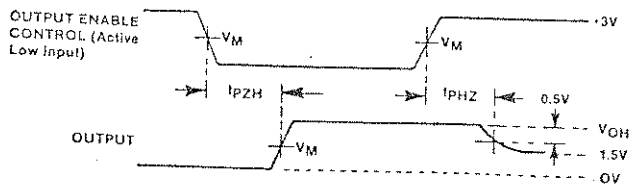
AC TEST WAVEFORMS

PROPAGATION DELAY
(Typical Example)

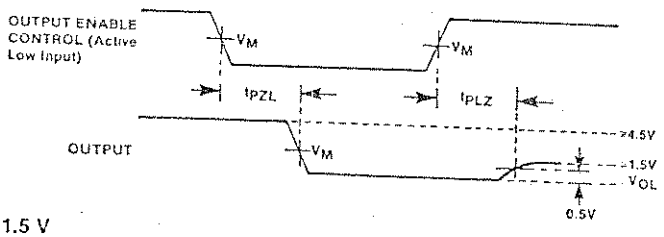


Note
Pulse widths and Setup/Hold times are measured using the same reference points as above waveform.

3-STATE ENABLE TIME TO LOW LEVEL AND DISABLE TIME FROM LOW LEVEL



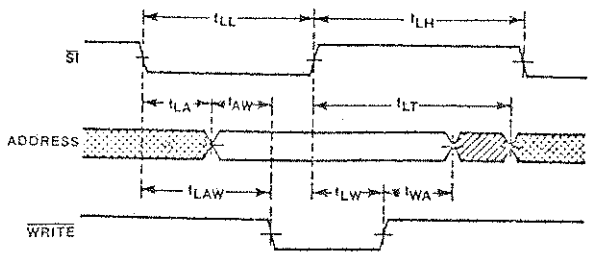
3-STATE ENABLE TIME TO HIGH LEVEL AND DISABLE TIME FROM HIGH LEVEL



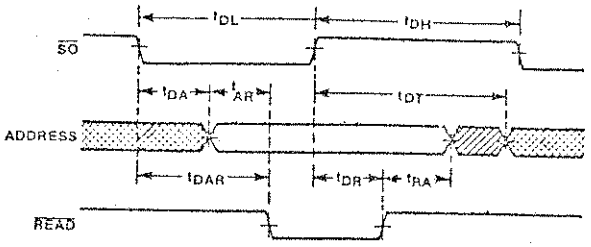
V_M = 1.5 V

TIMING DIAGRAMS

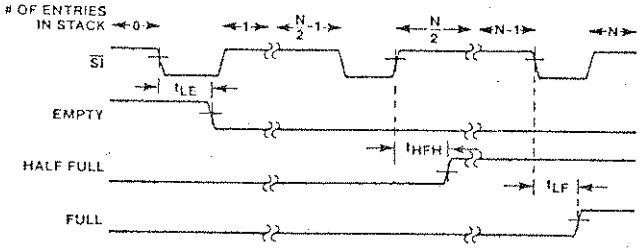
WRITE CYCLE TIMING



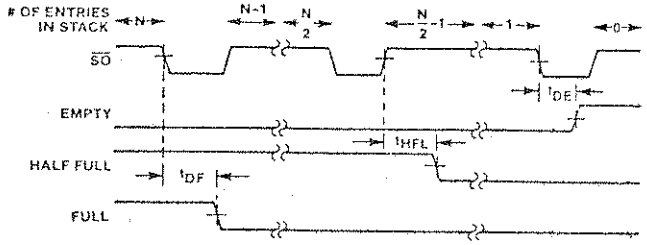
READ CYCLE TIMING



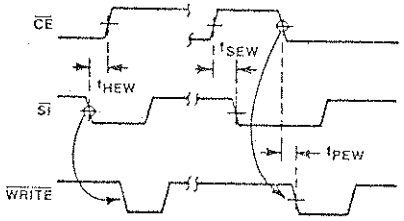
STATUS OUTPUT TIMING-WRITE



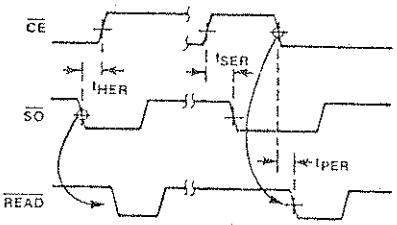
STATUS OUTPUT TIMING-READ



CHIP ENABLE TIMING WRITE*

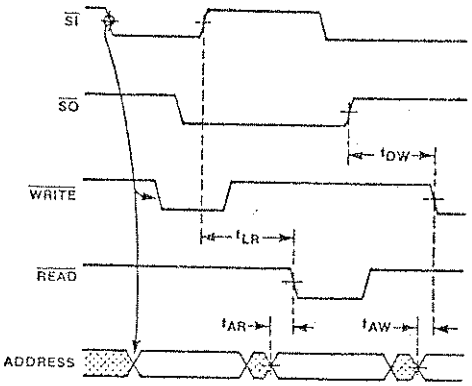


CHIP ENABLE TIMING READ*

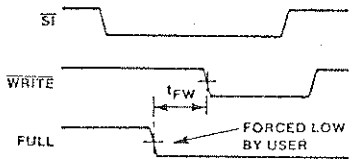


*The rising edge of $\overline{CE}$ should not occur within 10-nanoseconds before or after a falling edge of $\overline{SI}$ or $\overline{SO}$.

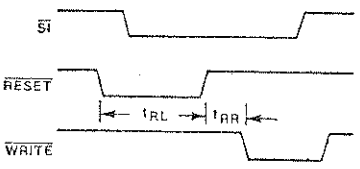
CHANGE OF CYCLE TIMING



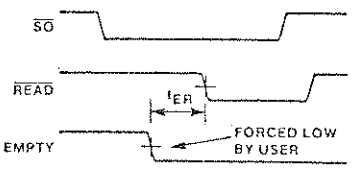
FULL OVERRIDE TIMING



RESET TIMING



EMPTY OVERRIDE TIMING



KEY

High-impedance state

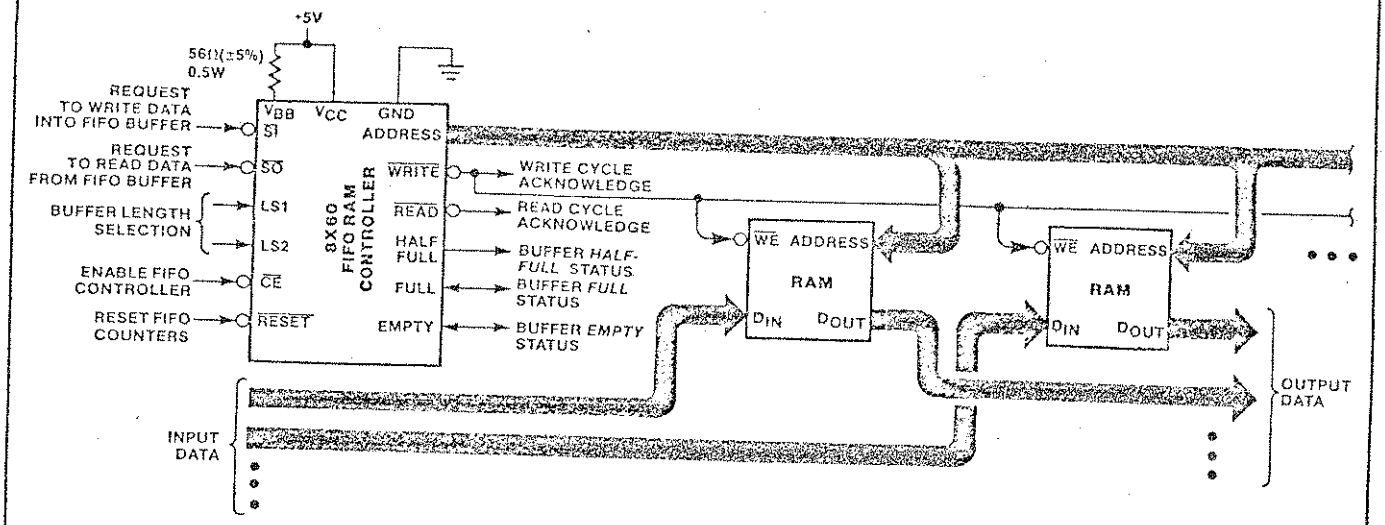
Changing data

FIFO RAM CONTROLLER (FRC)

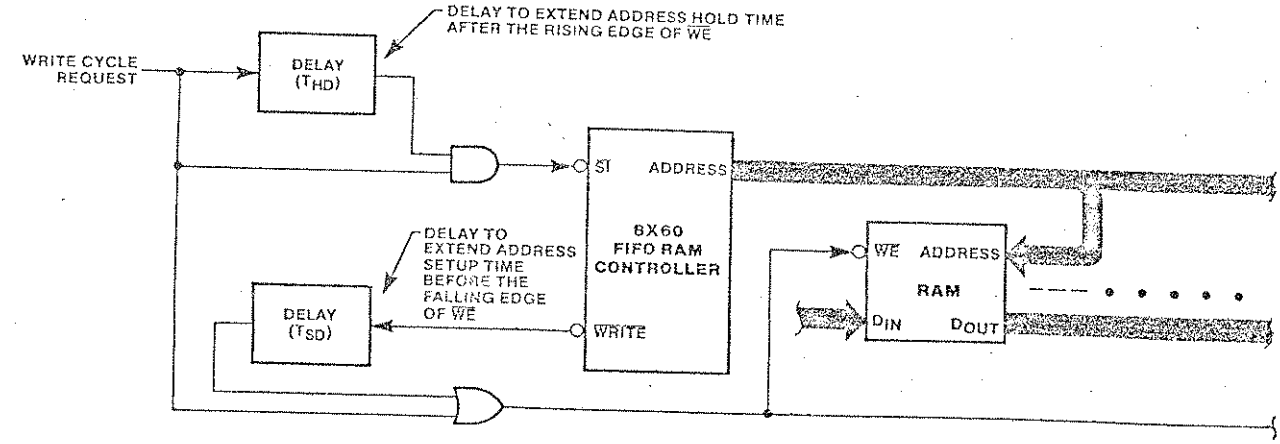
8X60

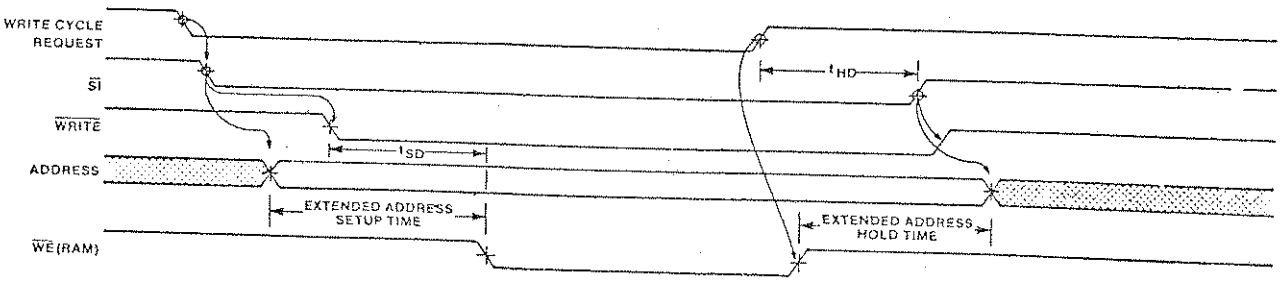
APPLICATIONS

IMPLEMENTATION OF A FIFO BUFFER USING THE 8X60 AND HIGH SPEED RAM



USING 8X60 WITH HIGH-DENSITY MOS RAMs





ORDERING INFORMATION

COMMERCIAL—

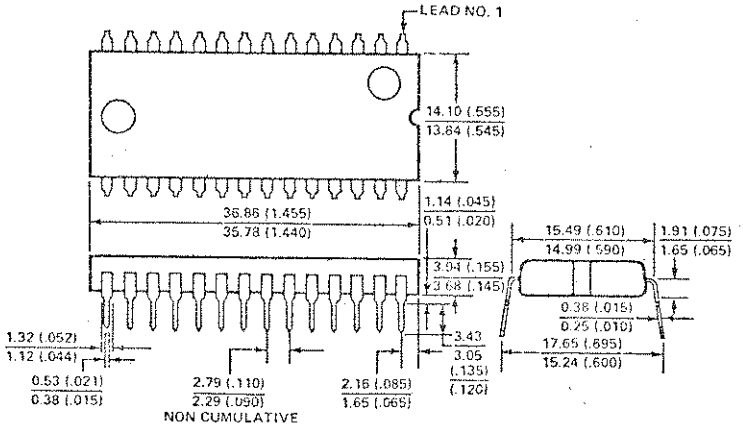
Order Number: N8X60N
Packaging information: Refer to Signetics price list
Supply Voltage: 5V (±5%)
Operating Temperature Range:
$0°C \leq T_A \leq 70°C$

MILITARY—

(Parts not Yet Available)

PACKAGE OUTLINE

N Package (Plastic)



Note

If the EMPTY or FULL status line is overridden (forced *low* externally) and transfers are performed beyond the normal buffer boundaries, the EMPTY status output may not occur when expected.

The sequencing of Status Counter #3 is not dependent upon the selected buffer length (LS1 and LS2). After overriding EMPTY or FULL, the 12-bit Status Counter may be displaced by a multiple of the selected buffer length, modulo 4096. The EMPTY status becomes active only when the Status Counter contains exactly the value "1" (with eleven leading zeroes). If the maximum buffer length (4096) is selected, the EMPTY status will always function properly.

The FULL status output is generated with respect to the selected buffer length (LS1 and LS2) and will always occur when expected.

National Semiconductor

MARCH 1982

DP8409 Multi-Mode Dynamic RAM Controller/Driver

General Description

Dynamic memory system designs, which formerly required several support chips to drive the memory array, can now be implemented with a single IC . . . the DP8409 Multi-Mode Dynamic RAM Controller/Driver. The DP8409 is capable of driving all 16k and 64k Dynamic RAMs (DRAMs) as well as 256k DRAMs. Since the DP8409 is a one-chip solution (including capacitive-load drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple-chip memory drive and control.

The DP8409's 8 modes of operation offer a wide selection of DRAM control capabilities. Memory access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing (either externally or automatically controlled) less complicated; and automatic memory initialization is both simple and fast.

The DP8409 is a 48-pin DRAM Controller/Driver with 9 multiplexed address outputs and 6 control signals. It consists of two 9-bit address latches, a 9-bit refresh counter, and control logic. All output drivers are capable of driving 500 pF loads with propagation delays of 25 ns. The DP8409 timing parameters are specified driving the typical load capacitance of 88 DRAMs, including trace capacitance.

The DP8409 has 3 mode-control pins: M2, M1, and M0, where M2 is in general $\overline{\text{REFRESH}}$. These 3 pins select 8 modes of operation. Inputs B1 and B0 in the memory access modes (M2 = 1), are select inputs which select one of four $\overline{\text{RAS}}$ outputs. During normal access, the 9 address outputs can be selected from the Row Address Latch or the Column Address Latch. During refresh, the 9-bit on-chip refresh counter is enabled onto the address bus and in this mode all $\overline{\text{RAS}}$ outputs are selected, while $\overline{\text{CAS}}$ is inhibited.

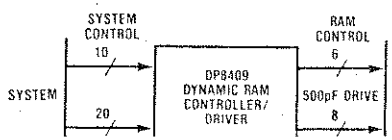
The DP8409 can drive up to 4 banks of DRAMs, with each bank comprised of 16k's, 64k's, or 256k's. Control signal outputs $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{WE}}$ are provided with the same drive capability. Each $\overline{\text{RAS}}$ output drives one bank of DRAMs so that the four $\overline{\text{RAS}}$ outputs are used to select the banks, while $\overline{\text{CAS}}$, $\overline{\text{WE}}$, and the multiplexed addresses can be connected to all of the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of the operating power) with the data outputs in TRI-STATE®. Only the bank with its associated $\overline{\text{RAS}}$ low will be written to or read from.

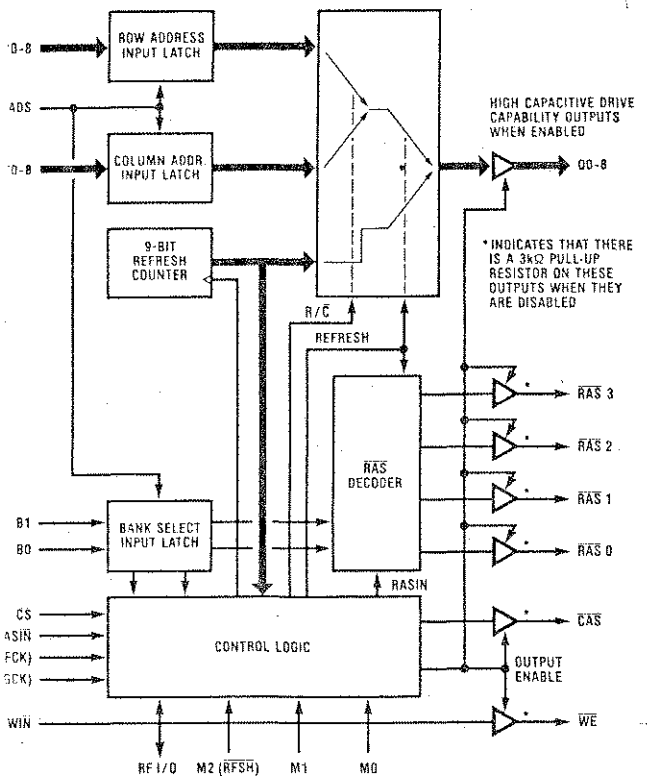
Operational Features

- All DRAM drive functions on one chip — minimizes skew on outputs, maximizes AC performance
- On-chip capacitive-load drives (specified to drive up to 88 DRAMs)
- Drives directly all 16k, 64k, and 256k DRAMs
- Capable of addressing 64k, 256k, or 1M words
- Propagation delays of 25 ns typical at 500 pF load
- $\overline{\text{CAS}}$ goes low automatically after column addresses are valid if desired
- Auto Access mode provides $\overline{\text{RAS}}$, row to column select, then $\overline{\text{CAS}}$ automatically and fast
- $\overline{\text{WE}}$ follows $\overline{\text{WIN}}$ unconditionally—offering READ, WRITE or READ-MODIFY-WRITE cycles
- On-chip 9-bit refresh counter with selectable End-of-Count (127, 255, or 511)
- End-of-Count indicated by RF I/O pin going low at 127, 255, or 511
- Low input on RF I/O resets 9-bit refresh counter
- $\overline{\text{CAS}}$ inhibited during refresh cycle
- Fall-through latches on address inputs controlled by ADS
- TRI-STATE outputs allow multi-controller addressing of memory
- Control output signals go high-impedance logic "1" when disabled for memory sharing
- Power-up: counter reset, control signals high, address outputs TRI-STATE, and End-of-Count set to 127

Mode Features

- 8 modes of operation: 3 access, 3 refresh, and 2 set-up
- 2 externally controlled modes: 1 access and 1 refresh (Modes 0, 4)
- 2 auto-access modes $\overline{\text{RAS}}$ → R/$\overline{\text{C}}$ → $\overline{\text{CAS}}$ automatic, with t_{RAH} = 20 or 30 ns minimum (Modes 5, 6)
- Auto-access mode allows Hidden Refreshing (Mode 5)
- Forced Refresh requested on RF I/O if no Hidden Refresh (Mode 5)
- Forced Refresh performed after system acknowledge of request (Mode 1)
- Automatic Burst Refresh mode stops at End-of-Count of 127, 255, or 511 (Mode 2)
- 2 All-$\overline{\text{RAS}}$ Access modes externally or automatically controlled for memory initialization (Modes 3a, 3b)
- Automatic All-$\overline{\text{RAS}}$ mode with external 8-bit counter frees system for other set-up routines (Mode 3a)
- End-of-Count value of Refresh Counter set by B1 and B0 (Mode 7)





DP8409 Functional Block Diagram

Table 1. DP8409 Mode Select Options

	M0	Mode of Operation	Conditions
	0	Externally Controlled Refresh	RF I/O = $\overline{EOC}$
	1	Auto Refresh — forced	RF I/O = Refresh Request ($\overline{RFRQ}$)
	0	Internal Auto Burst Refresh	RF I/O = $\overline{EOC}$
	1	All $\overline{RAS}$ Auto Write	RF I/O = $\overline{EOC}$; All $\overline{RAS}$ Active
	1	Externally Controlled All $\overline{RAS}$ Access	All $\overline{RAS}$ Active
	0	Externally Controlled Access	Active $\overline{RAS}$ defined by Table 2
	1	Auto Access, Slow t_{RAH}, Hidden Refresh	Active $\overline{RAS}$ defined by Table 2
	0	Auto Access, Fast t_{RAH}	Active $\overline{RAS}$ defined by Table 2
	1	Set End of Count	See Table 3 for Mode 7

Pin Definitions

V_{CC}, GND, GND — $V_{CC} = 5V \pm 5\%$. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are also two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 9 address bits change in the same direction simultaneously. A recommended solution would be a 1μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected close to pins 36 and 38 to reduce lead inductance.

R0-R8: Row Address Inputs.

C0-C8: Column Address Inputs.

Q0-Q8: Multiplexed Address Outputs — Selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.

$\overline{RASIN}$: Row Address Strobe Input — Enables selected $\overline{RAS}_n$ output when M2 ($\overline{RFSH}$) is high, or all $\overline{RAS}_n$ outputs when $\overline{RFSH}$ is low.

R/$\overline{C}$ (RFCK) — In Auto-Refresh Mode this pin is the external Refresh Clock Input: one refresh cycle has to be performed each clock period. In all other modes it is Row/Column Select Input: selects either the row or column address input latch onto the output bus.

$\overline{CASIN}$ (RGCK) — In Auto-Refresh Mode, Auto Burst Mode, and All-$\overline{RAS}$ Auto-Write Mode, this pin is the $\overline{RAS}$ Generator Clock Input. In all other modes it is $\overline{CASIN}$ (Column Address Strobe Input), which inhibits $\overline{CAS}$ output when high in Modes 4 and 3b. In Mode 6 it can be used to prolong $\overline{CAS}$ output.

ADS: Address (Latch) Strobe Input — Strobes Input Row Address, Column Address, and Bank Select Inputs into respective latches when high; Latches on high-to-low transition.

$\overline{CS}$: Chip Select Input — TRI-STATE®'s the Address Outputs and puts the control signal into a high-impedance logic "1" state when high (unless refreshing in one of the Refresh Modes). Enables all outputs when low.

M0, M1, M2: Mode Control Inputs — These 3 control pins determine the 8 major modes of operation of the DP8409 as depicted in Table 1.

RF I/O — The I/O pin functions as a Reset Counter Input when set low from an external open-collector gate, or as a flag output. The flag goes active-low in Modes 0 and 2 when the End-of-Count output is at 127, 255, or 511 (see Table 3). In Auto-Refresh Mode it is the Refresh Request output.

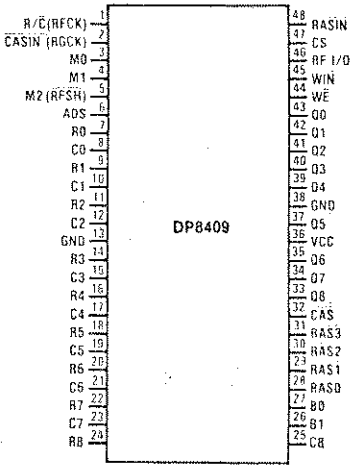
$\overline{WIN}$: Write Enable Input.

$\overline{WE}$: Write Enable Output — Buffered output from $\overline{WIN}$.

$\overline{CAS}$: Column Address Strobe Output — In Modes 3a, 5, and 6, $\overline{CAS}$ transitions low following valid column address. In Modes 3b and 4, it goes low after R/$\overline{C}$ goes low, or follows $\overline{CASIN}$ going low if R/$\overline{C}$ is already low. $\overline{CAS}$ is high during refresh.

$\overline{RAS}$ 0-3: Row Address Strobe Outputs — Selects a memory bank decoded from B1 and B0 (see Table 2), if

B0, B1: Bank Select Inputs — Strobed by ADS. Decoded to enable one of the $\overline{RAS}$ outputs when $\overline{RASIN}$ goes low. Also used to define End-of-Count in Mode 7 (Table 3).



Pin Configuration

Conditions for all Modes

Input Addressing

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after the valid address time, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

In normal memory access operation, $\overline{RASIN}$ and R/$\overline{C}$ are initially high. When the address inputs are enabled into the address latches, the row addresses appear on the Q outputs. The address strobe also inputs the bank-select address, (B0 and B1). If $\overline{CS}$ is low, all outputs are enabled. When $\overline{CS}$ is transitioned high, the address outputs go TRI-STATE® and the control outputs first go high through a low impedance, and then are held by an on-chip high impedance. This allows output paralleling with other DP8409s for multi-addressing. All outputs go active about 50 ns after the chip is selected again. If $\overline{CS}$ is high, and a refresh cycle begins, all the outputs become active until the end of the refresh cycle.

Drive Capability

The DP8409 has timing parameters that are specified with up to 600 pF loads. In a typical memory system this is equivalent to about 88, 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 or 22 bits, or two banks of 32 or 39 bits, or one bank of 64 or 72 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters, according to the graph of Figure 10. The AC performance parameters are specified with the typical load capaci-

k or 64k DRAMs

y 16k or 64k DRAMs. All 16k same configuration, including . Hence, In most applications, DRAMs are interchangeable chips), and the DP8409 can Figure 1a).

figurations for the 5V-only 64k column array with an on-RAM v by 512-column array with no and a 256-row by 256-column resh counter. The DP8409 can tions, and at the same time allows them all to be interchangeable (as shown in Figures 1b and 1c). Since the 9-bit on-chip refresh counter can be used as a 7-bit refresh counter for the 128-row configuration, or as an 8-bit refresh counter for the 256-row configuration, the on-RAM refresh counter (if present) is never used. As long as 128 rows are refreshed every 2 ms (i.e. 256 rows in 4 ms) all DRAM types are correctly refreshed.

This makes all the 64k DRAM configurations interchangeable when used with the DP8409, allowing maximum flexibility in the choice of DRAMs.

DP8409 Interface Between System & DRAM Banks

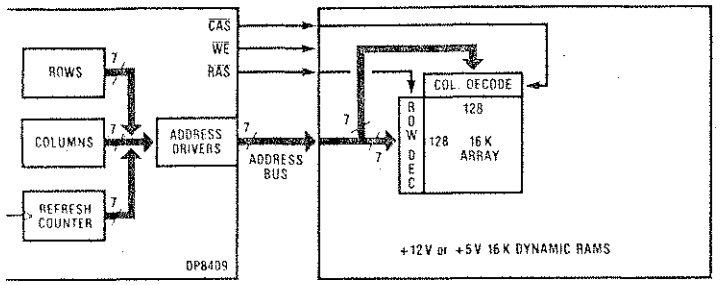
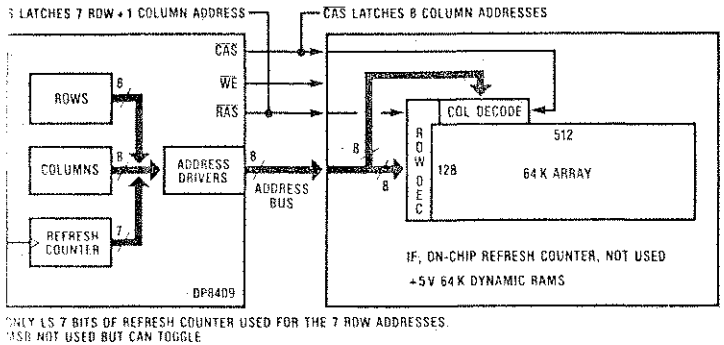
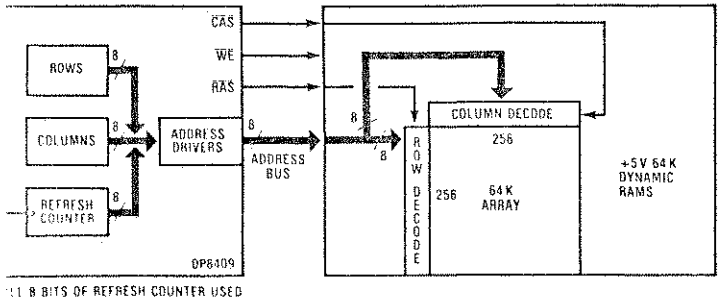


Figure 1a. DP8409 with any 16k DRAMS



ONLY LS 7 BITS OF REFRESH COUNTER USED FOR THE 7 ROW ADDRESSES.
MSB NOT USED BUT CAN TOGGLE

Figure 1b. DP8409 with 128 Row × 512 Column 64k DRAM



11 8 BITS OF REFRESH COUNTER USED

Figure 1c. DP8409 with 256 Row × 256 Column 64k DRAM

Read, Write, and Read-Modify-Write Cycles

The output signal, $\overline{WE}$, determines what type of memory access cycle the memory will perform. If $\overline{WE}$ is kept high while $\overline{CAS}$ goes low, a read cycle occurs. If $\overline{WE}$ goes low before $\overline{CAS}$ goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as $\overline{CAS}$ goes low. If $\overline{WE}$ goes low later than t_{CWD} after $\overline{CAS}$ goes low, first a read occurs and DO (DRAM output data) becomes valid; then data DI is written into the same address in the DRAM when $\overline{WE}$ goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by $\overline{WE}$, which follows $\overline{WIN}$.

Power-Up Initialize

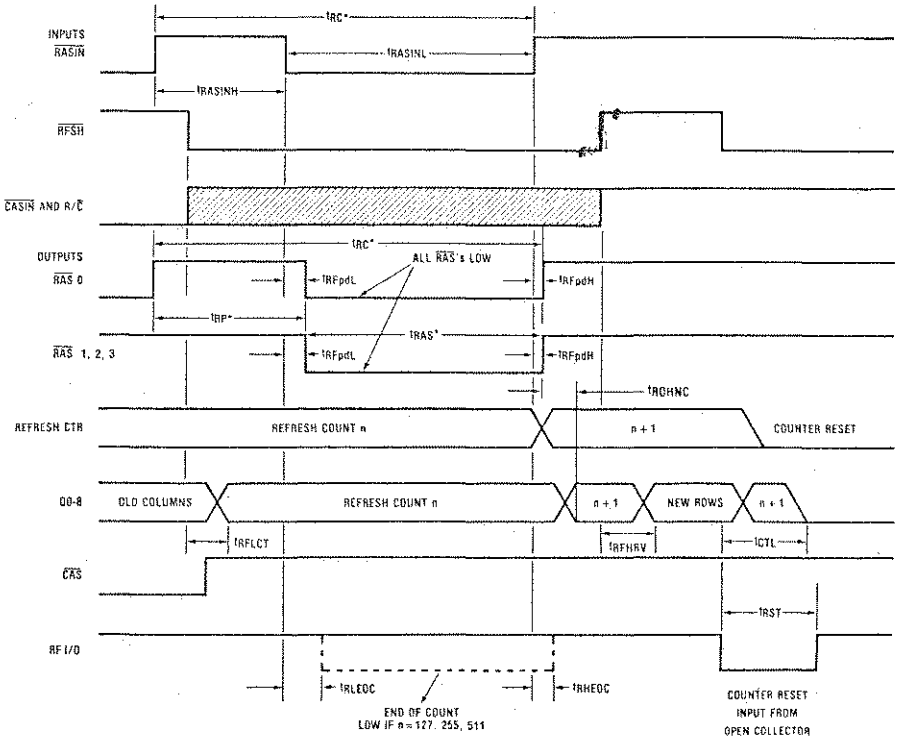
When V_{CC} is first applied to the DP8409, an initialize pulse clears the refresh counter, the internal control flip-flops, and sets the End-of-Count of the refresh counter to 127 (which may be changed via Mode 7). As V_{CC} increases to about 2.3 volts, it holds the output control signals at a level of one Schottky diode-drop below V_{CC}, and the output address to TRI-STATE®. As V_{CC} increases above 2.3 volts, control of these outputs is granted to the system.

DP8409 Functional Mode Descriptions

Mode 0 — Externally Controlled Refresh

Figure 2 is the Externally Controlled Refresh Timing. In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When $\overline{RAS}$ occurs, the enabled row in the DRAM is refreshed. In the Externally Controlled Refresh mode, all $\overline{RAS}$ outputs are enabled following $\overline{RASIN}$, and $\overline{CAS}$ is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either $\overline{RASIN}$ or $\overline{RFSH}$ goes low-to-high while the other is low. RF I/O goes low when the count is 127, 255, or 511, as set by End-of-Count (see Table 3), with $\overline{RASIN}$ and $\overline{RFSH}$ low. To reset the counter to all zeroes, RF I/O is set low through an external open-collector driver.

During refresh, $\overline{RASIN}$ and $\overline{RFSH}$ can transition low simultaneously because the refresh counter becomes valid on the output bus t_{RFLCT} after $\overline{RFSH}$ goes low, which is a shorter time than t_{RFPDL}. This means the counter address is valid on the Q outputs before $\overline{RAS}$ occurs on all $\overline{RAS}$ outputs, strobing the counter address into that row of all the DRAMs. Refer to Figure 2. To perform externally controlled burst refresh, $\overline{RFSH}$ initially can again have the same edge as $\overline{RASIN}$, but then can maintain a low state, since $\overline{RASIN}$ going low-to-high increments the counter (performing the burst refresh).



* INDICATES DYNAMIC RAM PARAMETERS

Figure 2. External Control Refresh Cycle (Mode 0)

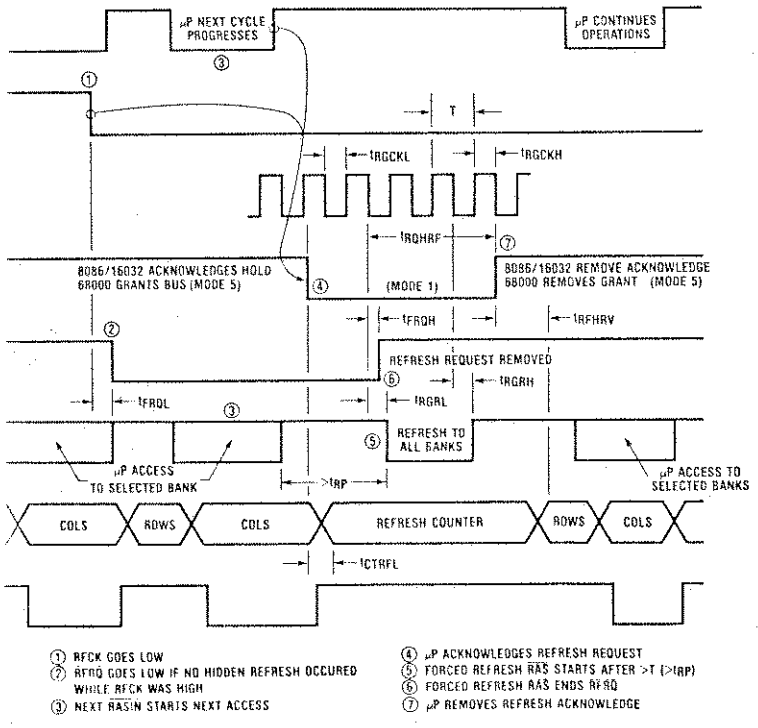
rced Refresh

pin becomes RFCK (refresh
, and $\overline{CAS}$ remains high. If
gh, then whenever M2 ($\overline{RFSH}$)
rolled refresh will occur and
$\overline{RASIN}$, strobing the refresh
RAMs. The RF I/O pin will
n set low externally through
e refresh counter resets as
trolled method may be pre-
e Automatic Access mode
orced refreshing is undesir-
necessary.

signal, one (and only one)
ice every RFCK cycle. Refer
fresh does not occur while
ien RF I/O ($\overline{Refresh\ Request}$)
RFCK goes low, indicating to
refresh is requested. The
d refresh to take place while
ire 3). The $\overline{Refresh\ Request}$
connected to a $\overline{Hold}$ or $\overline{Bus}$
n. The system acknowledges
hen ready, and outputs Hold
iest Acknowledge. If this is
4) pin, a forced-refresh cycle
DP8409, and $\overline{RAS}$ will be
four $\overline{RAS}$ outputs, to strobe
nts on the address outputs
ternal $\overline{RAS}$ Generator Clock

(RGCK) is required for this function. It is fed to the $\overline{CASIN}$ (RGCK) pin, and may be up to 10 MHz. Whenever M2 goes low (inducing a forced refresh), $\overline{RAS}$ remains high for one to two periods of RGCK, depending on when M2 goes low relative to the high-to-low triggering edge of RGCK; $\overline{RAS}$ then goes low for two periods, performing a refresh on all banks. In order to obtain the minimum delay from M2 going low to $\overline{RAS}$ going low, M2 should go low t_{RFSRG} before the next falling edge of RGCK. The $\overline{Refresh\ Request}$ on RF I/O is terminated as $\overline{RAS}$ begins, so that by the time the system has acknowledged the removal of the request and disabled its Acknowledge, (i.e., M2 goes high), Refresh $\overline{RAS}$ will have ended, and normal operations can begin again in the Automatic Access mode (Mode 5). If it is desired that Refresh $\overline{RAS}$ end in less than 2 periods of RGCK from the time $\overline{RAS}$ went low, then M2 may go high earlier than t_{FRQH} after RF I/O goes high and $\overline{RAS}$ will go high t_{RFRH} after M2.

To allow the forced refresh, the system will have been inactive for about 4 periods of RGCK, which can be as fast as 400 ns every RFCK cycle. To guarantee a refresh of 128 rows every 2 ms, a period of up to 16 μs is required for RFCK. In other words, the system may be down for as little as 400 ns every 16 μs, or 2.5% of the time. Although this is not excessive, it may be preferable to perform a Hidden Refresh each RFCK cycle, which is allowed while still in the Auto-Access mode, (Mode 5).



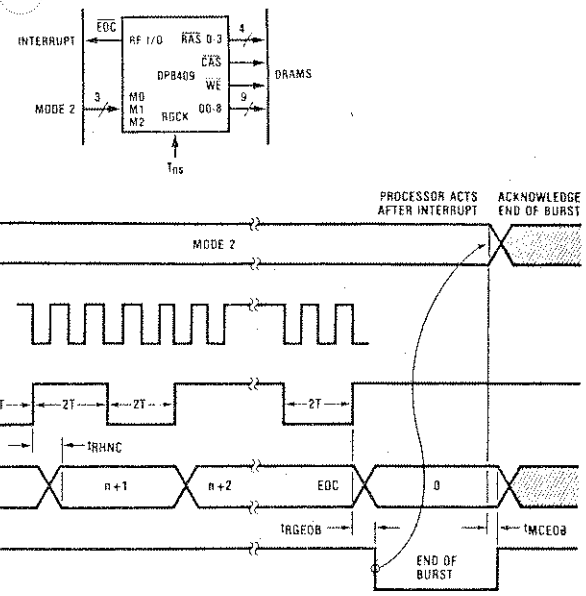


Figure 4. Auto-Burst Mode, Mode 2

Mode 2 — Automatic Burst Refresh

This mode is normally used before and/or after a DMA operation to ensure that all rows remain refreshed, provided the DMA transfer takes less than 2 ms (see Figure 4). When the DP8409 enters this mode, $\overline{CASIN}$ (RGCK) becomes the $\overline{RAS}$ Generator Clock (RGCK), and $\overline{RASIN}$ is disabled. $\overline{CAS}$ remains high, and RF I/O goes low when the refresh counter has reached the selected End-of-Count and the last $\overline{RAS}$ has ended. RF I/O then remains low until the Auto-Burst Refresh mode is terminated. RF I/O can therefore be used as an interrupt to indicate the End-of-Burst condition.

The signal on all four $\overline{RAS}$ outputs is just a divide-by-four of RGCK; in other words, if RGCK has a 100 ns period, $\overline{RAS}$ is high and low for 200 ns each cycle. The refresh counter increments at the end of each $\overline{RAS}$, starting from the count it contained when the mode was entered. If this was zero, then for a RGCK with a 100 ns period with End-of Count set to 127, RF I/O will go low after $128 \times 0.4\ \mu s$, or 51.2 μs. During this time, the system may be performing operations that do not involve DRAM. If all rows need to be burst refreshed, the refresh counter may be cleared by setting RF I/O low externally before the burst begins.

Burst-mode refreshing is also useful when powering down systems for long periods of time, but with data retention still required while the DRAMs are in standby. To maintain valid refreshing, power can be applied to the DP8409 (set to Mode 2), causing it to perform a complete burst refresh. When end-of-burst occurs (after 26 μs), power can then be removed from the DP8409 for 2 ms, consuming an average power of 1.3% of normal operating power. No control signal glitches occur when switching power to the DP8409.

Mode 3a — All-$\overline{RAS}$ Automatic Write

Mode 3a is useful at system initialization, when the memory is being cleared (i.e., with all-zeroes in the data field and the corresponding check bits for error detection and correction). This requires writing the same data to each location of memory (every row of each column of each bank). All $\overline{RAS}$ outputs are activated, as in refresh, and so are $\overline{CAS}$ and $\overline{WE}$. To write to all four banks simultaneously, every row is strobed in each column, in sequence, until data has been written to all locations.

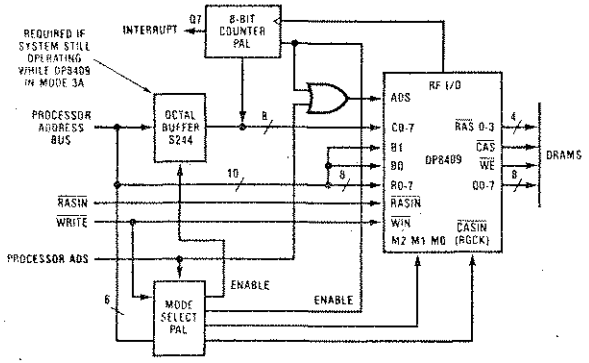
To select this mode, B1 and B0 must have previously been set to 00, 01, or 10 in Mode 7, depending on the DRAM size. For example, for 16k DRAMs, B1 and B0 are 00. For 64k DRAMs, B1 and B0 are 01, so that for the configuration of Figure 1b, the 8 refresh counter bits are strobed by $\overline{RAS}$ into the 7 row addresses and the ninth column address. After this Automatic-Write process, B1 and B0 must be set again in Mode 7 to 00 to set End-of-Count to 127. For the configuration of Figure 1c, B1 and B0 set to 01 will work for Automatic-Write and End-of-Count equals 255.

In this mode, R/$\overline{C}$ is disabled, $\overline{WE}$ is permanently enabled low, and $\overline{CASIN}$ (RGCK) becomes RGCK. RF I/O goes low whenever the refresh counter is 127, 255, or 511 (as set by End-of-Count in Mode 7), and the $\overline{RAS}$ outputs are active.

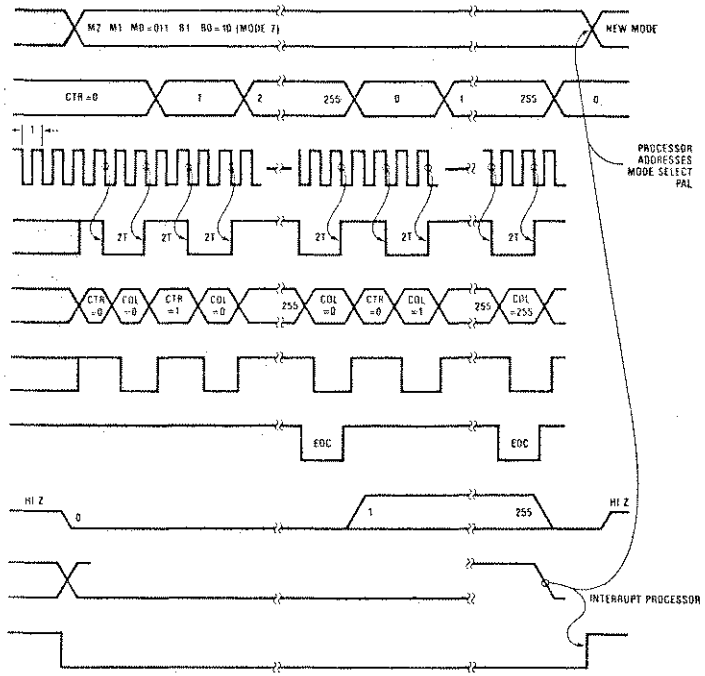
Referring to Figure 5a, an external 8-bit counter (for 64k DRAMs) with TRI-STATE® outputs is required and must be connected to the column address inputs. It is enabled only during this mode, and is clocked from RF I/O. The DP8409 refresh counter is used to address the rows, and the column address is supplied by the external counter. Every row for each column address is written to in all four banks. At the End-of-Count RF I/O goes low,

address, the refresh counter
ddress bus and all four $\overline{RAS}$
ddress into the DRAMs (see
f 30ns after $\overline{RAS}$ goes low
counter is disabled and the
is enabled onto the address
olumn address is valid, $\overline{CAS}$
trobing the column address
and $\overline{CAS}$ go high the refresh
ext row and the cycle repeats.
s mode, the data at DI (input
n into each row of the latched
$\overline{RAS}$ is high for two periods of
ds, giving a total write-cycle
ich is adequate for most 16k
row of a column, RF I/O incre-
to the next column address.

At the end of the last column address, an interrupt is generated from the external counter to let the system know that initialization has been completed. During the entire initialization time, the system can be performing other initialization functions. This approach to memory initialization is both automatic and fast. For instance, if four banks of 64k DRAMs are used, and RGCK is 100 ns, a write cycle to the same location in all four banks takes 400 ns, so the total time taken in initializing the 64k DRAMs is 65k × 400 ns or 26 ms. When the system receives the interrupt, the external counter must be permanently disabled. ADS and $\overline{CS}$ are interfaced by the system, and the DP8409 mode is changed. The interrupt must then be disabled.



a. DP8409 Extra Circuitry Required for All-$\overline{RAS}$ Auto Write Mode, Mode 3a



Mode 3b — Externally Controlled All $\overline{RAS}$ Write

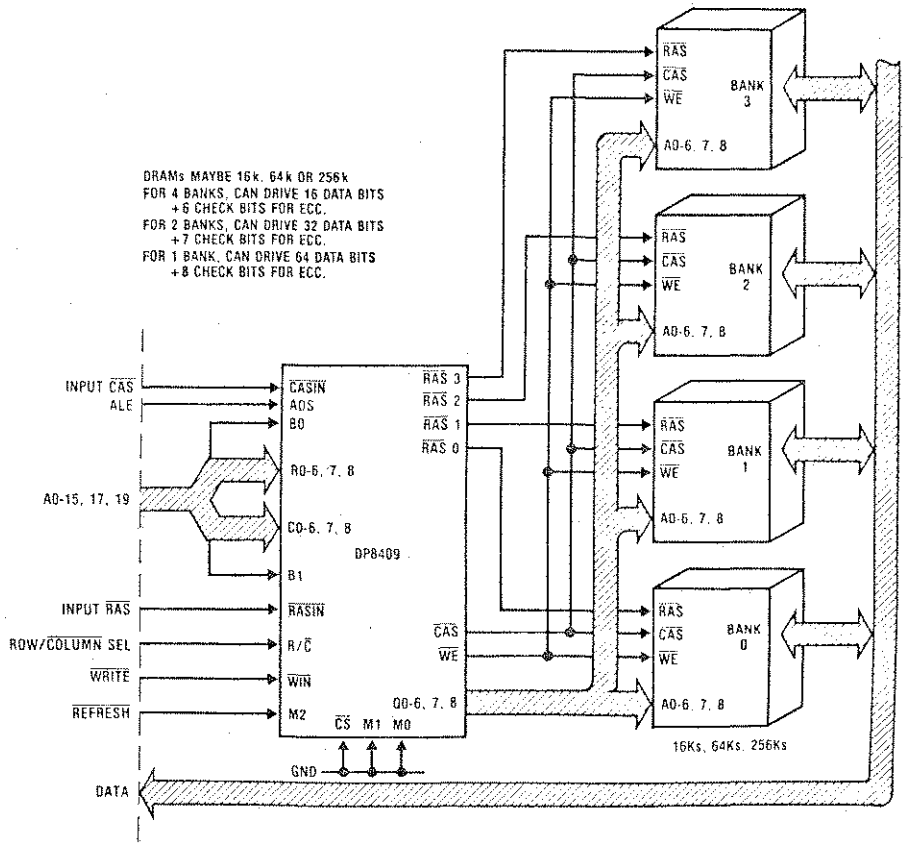
To select this mode, B1 and B0 must first have been set to 11 in Mode 7. This mode is useful at system initialization, but under processor control. The memory address is provided by the processor, which also performs the incrementing. All four $\overline{RAS}$ outputs follow $\overline{RASIN}$ (supplied by the processor), strobing the row address into the DRAMs. R/$\overline{C}$ can now go low, while $\overline{CASIN}$ may be used to control $\overline{CAS}$ (as in the Externally Controlled Access mode), so that $\overline{CAS}$ strobes the column address contents into the DRAMs. At this time $\overline{WE}$ should be low, causing the data to be written into all four banks of DRAMs. At the end of the write cycle, the input address is incremented and latched by the DP8409 for the next write cycle. This method is slower than Mode 3a since the processor must perform the incrementing and accessing. Thus the processor is occupied during RAM initialization, and is not free for other initialization operations. However, initialization sequence timing is under system control, which may provide some system advantage.

Mode 4 — Externally Controlled Access

This mode facilitates externally controlling all access-timing parameters associated with the DRAMs. The application of modes 0 and 4 are shown in Figure 6.

Output Address Selection

Refer to Figure 7a. With M2 ($\overline{RFSH}$) and R/$\overline{C}$ high, the row address latch contents are transferred to the multiplexed address bus output Q0-Q8, provided $\overline{CS}$ is set low. The column address latch contents are output after R/$\overline{C}$ goes low. $\overline{RASIN}$ can go low after the row addresses have been set up on Q0-Q8. This selects one of the $\overline{RAS}$ outputs, strobing the row address on the Q outputs into the desired bank of memory. After the row-address hold-time of the DRAMs, R/$\overline{C}$ can go low so that about 40 ns later column addresses appear on the Q outputs.



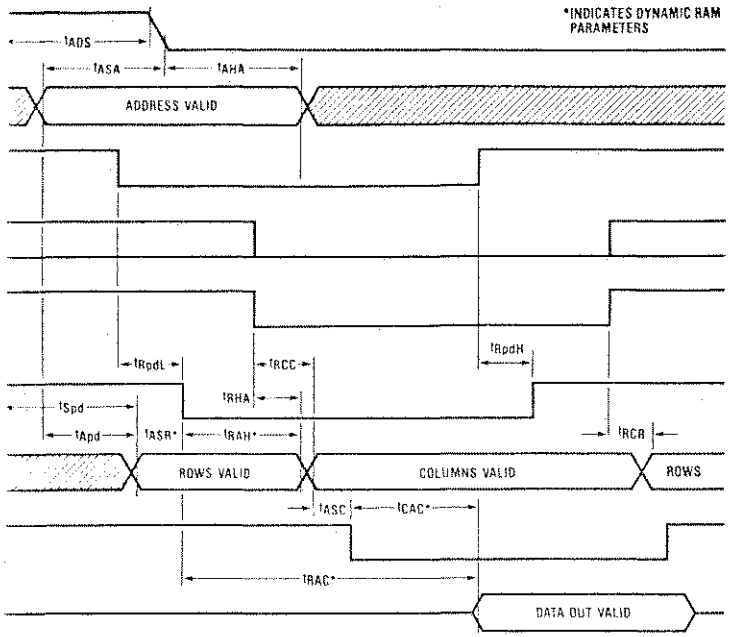
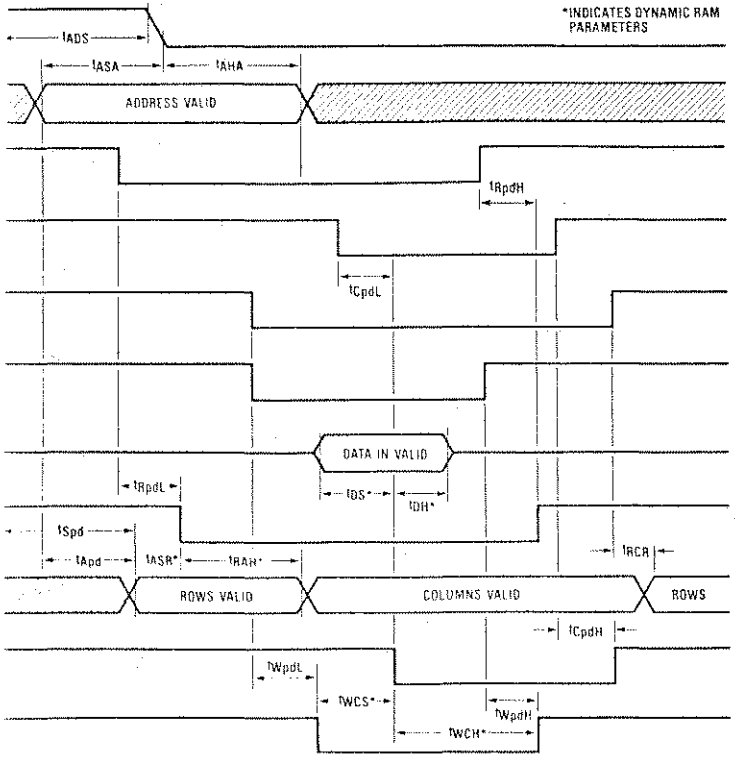


Figure 7a. Read Cycle Timing (Mode 4)



Automatic $\overline{CAS}$ Generation

In a normal memory access cycle $\overline{CAS}$ can be derived from inputs $\overline{CASIN}$ or R/$\overline{C}$. If $\overline{CASIN}$ is high, then R/$\overline{C}$ going low switches the address output drivers from rows to columns. $\overline{CASIN}$ then going low causes $\overline{CAS}$ to go low approximately 40 ns later, allowing $\overline{CAS}$ to occur at a predictable time (see Figure 7b). For maximum system speed, $\overline{CASIN}$ can be kept low, since $\overline{CAS}$ will automatically occur approximately 20 ns after the column addresses are valid, or about 60 ns after R/$\overline{C}$ goes low (see Figure 7a). Most DRAMs have a column address set-up time before $\overline{CAS}$ (t_{ASC}) of 0 ns or −10 ns. In other words, a t_{ASC} greater than 0 ns is safe. This feature reduces timing-skew problems, thereby improving access time of the system.

Fast Memory Access

For faster access time, R/$\overline{C}$ can go low a time delay ($t_{RPDL} + t_{RAH} - t_{RHA}$) after $\overline{RASIN}$ goes low, where t_{RAH} is the Row-Address hold-time of the DRAM.

Mode 5 — Automatic Access with Hidden Refresh

The Auto Access with Hidden Refresh mode has two advantages over the Externally Controlled Access mode, due to the fact that all outputs except $\overline{WE}$ are initiated from $\overline{RASIN}$. First, inputs R/$\overline{C}$ and $\overline{CASIN}$ are unnecessary and can be used for other functions (see Refreshing, below). Secondly, because the output control signals are derived internally from one input signal ($\overline{RASIN}$), timing-skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs. The automatic access features of Mode 5 (and Mode 6) of the DP8409 make DRAM accessing appear essentially "static".

Automatic Access Control

The major disadvantage of DRAMs compared to static RAMs is the complex timing involved. First, a $\overline{RAS}$ must occur with the row address previously set up on the multiplexed address bus. After the row address has been held for t_{RAH}, (the Row-Address hold-time of the DRAM), the column address is set up and then $\overline{CAS}$ occurs. This is all performed automatically by the DP8409 in this mode.

Provided the input address is valid as ADS goes low, $\overline{RASIN}$ can go low any time after ADS. This is because the selected $\overline{RAS}$ occurs typically 27 ns later, by which time the row address is already valid on the address output of the DP8409. The Address Set-Up time (t_{ASR}), is 0 ns on most DRAMs. The DP8409 in this mode (with ADS and $\overline{RASIN}$ edges simultaneously applied) produces a minimum t_{ASR} of 0 ns. This is true provided the input address was valid t_{ASA} before ADS went low (see Figure 8a).

Next, the row address is disabled after t_{RAH} (30 ns minimum); in most DRAMs, t_{RAH} minimum is less than 30 ns. The column address is then set up and t_{ASC} later, $\overline{CAS}$ occurs. The only other control input required is $\overline{WIN}$. When a write cycle is required, WIN must go low at least 30 ns before $\overline{CAS}$ is output low.

This gives a total typical delay from: input address valid to $\overline{RASIN}$ (15 ns); to $\overline{RAS}$ (27 ns); to rows held (50 ns); to columns valid (25 ns); to $\overline{CAS}$ (23 ns) = 140 ns (that is, 125 ns from $\overline{RASIN}$). All of these typical figures are for

This mode is therefore extremely fast. The external timing is greatly simplified for the memory system designer: the only system signal required is $\overline{RASIN}$.

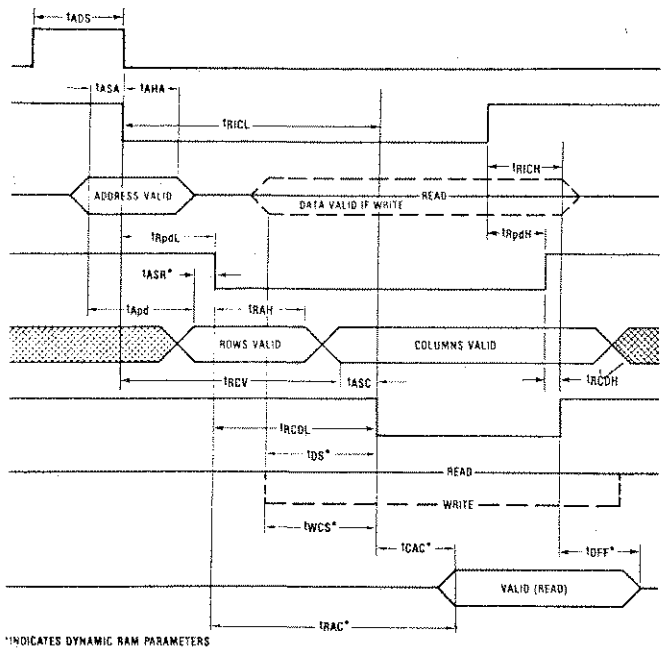
Refreshing

Because R/$\overline{C}$ and $\overline{CASIN}$ are not used in this mode, R/$\overline{C}$ becomes RFCK (refresh clock) and $\overline{CASIN}$ becomes RGCK ($\overline{RAS}$ generator clock). With these two signals it is possible to perform refreshing without extra ICs, and without holding up the processor.

One refresh cycle must occur during each refresh clock period and then the refresh address must be incremented to the next refresh cycle. As long as 128 rows are refreshed every 2 ms (one row every 16 μs), all 16k and 64k DRAMs will be correctly refreshed. The cycle time of RFCK must, therefore, be less than 16 μs. RFCK going high sets an internal refresh-request flip-flop. First the DP8409 will attempt to perform a hidden refresh so that the system throughput will not be affected. If, during the time RFCK is high, $\overline{CS}$ on the DP8409 goes high and $\overline{RASIN}$ occurs, a hidden refresh will occur. In this case, $\overline{RASIN}$ should be considered a common read/write strobe. In other words, if the processor is accessing elsewhere (other than the DRAMs) while RFCK is high, the DP8409 will perform a refresh. The refresh counter is enabled to the address outputs whenever $\overline{CS}$ goes high with RFCK high, and all $\overline{RAS}$ outputs follow $\overline{RASIN}$. If a hidden refresh is taking place as RFCK goes low, the refresh continues. At the start of the hidden refresh, the refresh-request flip-flop is reset so no further refresh can occur until the next RFCK period starts with the positive-going edge of RFCK. Refer to Figure 9.

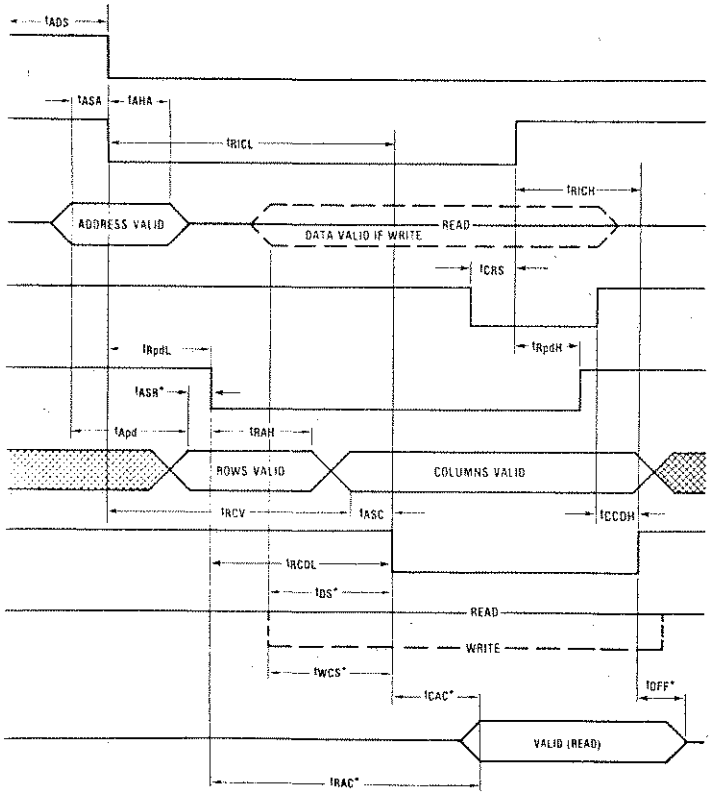
To determine the probability of a Hidden Refresh occurring, assume each system cycle takes 400 ns and RFCK is high for 8 μs, then the system has 20 chances to not select the DP8409. If during this time a hidden refresh did not occur, then the DP8409 forces a refresh while RFCK is low, but the system chooses when the refresh takes place. After RFCK goes low, (and the internal-request flip-flop has not been reset), RF I/O goes low indicating that a refresh is requested to the system. Only when the system acknowledges this request by setting M2 ($\overline{RFSH}$) low does the DP8409 initiate a forced refresh (which is performed automatically). Refer to Mode 1, and Figure 3. The internal refresh request flip-flop is then reset.

Figure 9 illustrates the refresh alternatives in Mode 5. If a hidden refresh has occurred and $\overline{CS}$ again goes high before RFCK goes low, the chip is deselected. All the control signals go high-impedance high (logic "1") and the address outputs go TRI-STATE® until $\overline{CS}$ again goes low. This mode (combined with Mode 1) allows very fast access, and automatic refreshing (possibly not even slowing down the system), with no extra ICs. Careful system design can, and should, provide a higher probability of hidden refresh occurring. The duty cycle of RFCK need not be 50-percent; in fact, the low-time should be designed to be a minimum. This is determined by the worst-case time (required by the system) to respond to the DP8409's forced-refresh request.

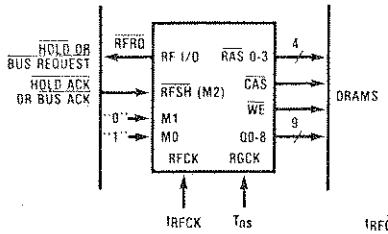


*INDICATES DYNAMIC RAM PARAMETERS

Figure 8a. Modes 5, 6 Timing ($\overline{\text{CASIN}}$ High in Mode 6)



CATES DYNAMIC RAM PARAMETERS



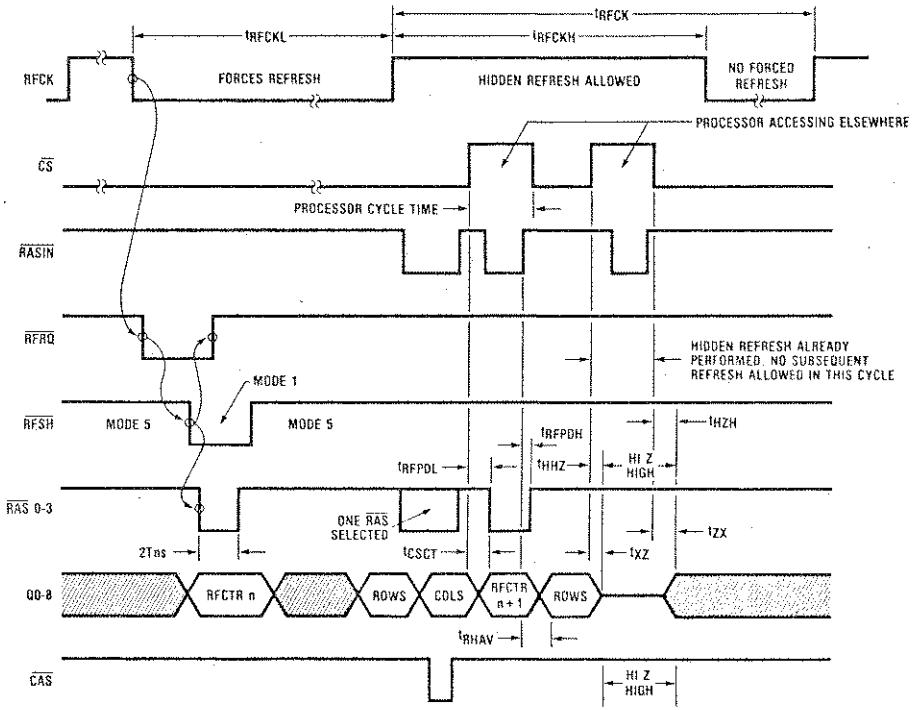


Figure 9. Hidden Refreshing (Mode 5) and Forced Refreshing (Mode 1) Timing

ory Bank Decode

	Enabled $\overline{RAS}_n$
	$\overline{RAS}_0$
	$\overline{RAS}_1$
	$\overline{RAS}_2$
	$\overline{RAS}_3$

n earlier than t_{CSRL} after $\overline{CS}$ he DP8409 interpreting the $\overline{RASIN}$ if no hidden refresh RFCK cycle. In this case, all for a short time. Thus, it is g Mode 5, $\overline{RASIN}$ should be $\overline{CS}$ goes low if a refresh is not

itic Access

similar to Mode 5, but has a num. It therefore can only be DRAMs (which have a t_{RAH} of s requiring fast access times; 105ns.

K) pin is not used, but $\overline{CASIN}$ l to allow an extended $\overline{CAS}$ minated. Refer to Figure 8b. cycle-times where $\overline{RAS}$ has to possible before the next $\overline{RAS}$ begins (to meet the precharge time, or t_{RP}, requirements of the DRAM). $\overline{CAS}$ may then be held low by $\overline{CASIN}$ to extend the data output valid time from the DRAM to allow the system to read the data. $\overline{CASIN}$ subsequently going high ends $\overline{CAS}$. If this extended $\overline{CAS}$ is not required, $\overline{CASIN}$ should be set high in Mode 6.

There is no internal refresh-request flip-flop in this mode, so any refreshing required must be done by entering Mode 0 or Mode 2.

Mode 7 — Set End-of-Count

The End-of-Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B1 and B0 (see Table 3). With B1 and B0 the same $\overline{EOC}$ is 127; with B1 = 0 and B0 = 1, $\overline{EOC}$ is 255; and with B1 = 1 and B0 = 0, $\overline{EOC}$ is 511. This selected value of $\overline{EOC}$ will be used until the next Mode 7 selection. At power-up the $\overline{EOC}$ is automatically set to 127 (B1 and B0 set to 11).

Table 3. Mode 7

Bank Select (Strobed by ADS)		End of Count
B1	B0	Selected
0	0	127
0	1	255
1	0	511
1	1	127

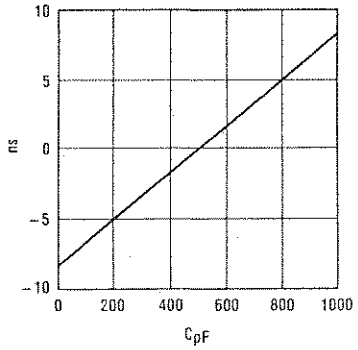


Figure 10. Change in Propagation Delay vs Loading Capacitance Relative to a 500pF Load

Absolute Maximum Ratings (Note 1)

Supply Voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150mA
Lead Temperature (Soldering, 10 seconds)	300°C

Operating Conditions

	Min.	Max.	Units
V_{CC} Supply Voltage	4.75	5.25	V
T_A Ambient Temperature	0	+70	°C

Electrical Characteristics $V_{CC} = 5.0V \pm 5\%$, $0°C \leq T_A \leq 70°C$ unless otherwise noted (Note 2)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
V_C	Input Clamp Voltage	V_{CC} = Min., I_C = −12mA		−0.8	−1.2	V
I_{IH1}	Input High Current for ADS, R/$\overline{C}$ only	V_{IN} = 2.5V		2.0	100	μA
I_{IH2}	Input High Current for All Other Inputs*	V_{IN} = 2.5V		1.0	50	μA
I_I RSI	Output Load Current for RF I/O	V_{IN} = 0.5V, Output High		−1.5	−2.5	mA
I_I CTL	Output Load Current for $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	V_{IN} = 0.5V, Chip Deselect		−1.5	−2.5	mA
I_{IL1}	Input Low Current for ADS, R/$\overline{C}$ only	V_{IN} = 0.5V		−0.1	−1.0	mA
I_{IL2}	Input Low Current for All Other Inputs*	V_{IN} = 0.5V		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	I_{OL} = 20mA		0.3	0.5	V
V_{OL2}	Output Low Voltage for RF I/O	I_{OL} = 10mA		0.3	0.5	V
V_{OH1}	Output High Voltage*	I_{OH} = −1mA	2.4	3.5		V
V_{OH2}	Output High Voltage for RF I/O	I_{OH} = −100μA	2.4	3.5		V
I_{1D}	Output High Drive Current*	V_{OUT} = 0.8V (Note 3)		−200		mA
I_{0D}	Output Low Drive Current*	V_{OUT} = 2.7V (Note 3)		200		mA
I_{OZ}	TRI-STATE® Output Current (Address Outputs)	$0.4V \leq V_{OUT} \leq 2.7V$, $\overline{CS}$ = 2.0V, Mode 4	−50	1.0	50	μA
I_{CC}	Supply Current	V_{CC} = Max.		250	325	mA

*Except RF I/O Output.

Switching Characteristics $V_{CC} = 5.0V \pm 5\%$, $0°C \leq T_A \leq 70°C$ unless otherwise noted (Notes 2, 4, and 5).

The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500pF; $\overline{RAS}$0–$\overline{RAS}$3, C_L = 150pF; $\overline{WE}$, C_L = 500pF; $\overline{CAS}$: C_L = 600pF, unless otherwise noted. See Figure 11 for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7kΩ unless otherwise noted.

Symbol	Access Parameter	Conditions	8409			8409 – 3			Units
			Min.	Typ.	Max.	Min.	Typ.	Max.	
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	95	125	160	95	125	185	ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	80	105	140	80	105	160	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	40	48	60	40	48	70	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	50	63	80	50	63	95	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		98	125		98	145	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b		78	105		78	120	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		27	40		27	40	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8a		40	65		40	65	ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8b	40	54	70	40	54	80	ns
t_{RAH}	Row Address Hold Time (Mode 5)	Figure 8a	30			30			ns
t_{RAH}	Row Address Hold Time (Mode 6)	Figures 8a, 8b	20			20			ns
t_{ASC}	Column Address Setup Time (Mode 5)	Figure 8a	8			8			ns
t_{ASC}	Column Address Setup Time (Mode 6)	Figures 8a, 8b	6			6			ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid	Figure 8a		90	120		90	140	ns

teristics (Cont'd)

ss Parameter	Conditions	8409			8409 – 3			Units
		Min.	Typ.	Max.	Min.	Typ.	Max.	
Address Valid (Mode 6)	Figures 8a, 8b		75	105		75	120	ns
lay	Figures 7a, 7b, 8a, 8b	20	27	35	20	27	40	ns
lay	Figures 7a, 7b, 8a, 8b	15	23	32	15	23	37	ns
Output Low Delay	Figures 7a, 7b, 8a, 8b		25	40		25	46	ns
Output High Delay	Figures 7a, 7b, 8a, 8b		25	40		25	46	ns
Address Output Low	Figures 7a, 7b		40	60		40	70	ns
Address Output High	Figures 7a, 7b		40	60		40	70	ns
me to ADS	Figures 7a, 7b, 8a, 8b	15			15			ns
e from ADS	Figures 7a, 7b, 8a, 8b	15			15			ns
ulse Width	Figures 7a, 7b, 8a, 8b	30			30			ns
t Delay	Figure 7b	15	25	30	15	25	35	ns
t Delay	Figure 7b	15	30	60	15	30	70	ns
e to $\overline{\text{RASIN}}$ High (Mode 6)	Figure 8b	35			35			ns
lay (R$\overline{\text{IC}}$ low in Mode 4)	Figure 7b	32	41	58	32	41	67	ns
lay (R$\overline{\text{IC}}$ low in Mode 4)	Figure 7b	25	39	50	25	39	60	ns
Column Address Valid	Figure 7a		40	58		40	67	ns
w Address Valid	Figures 7a, 7b		40	58		40	67	ns
d from Column Select	Figure 7a	10			10			ns
esh Parameter								
riod	Figure 2	100			100			ns
$\overline{\text{ASIN}}$ during Refresh	Figure 2	50			50			ns
lay during Refresh	Figures 2, 9	35	50	70	35	50	80	ns
lay during Refresh	Figures 2, 9	30	40	55	30	40	65	ns
nter Address Valid	$\overline{\text{CS}}$ = X, Figures 2,3,4		47	60		47	70	ns
w Address Valid	Figures 2, 3		45	60		45	70	ns
Count Valid	Figures 2, 4		30	55		30	55	ns
d-of-Count Low	C_L = 50pF, Figure 2			80			80	ns
d-of-Count High	C_L = 50pF, Figure 2			80			80	ns
d-of-Burst Low	C_L = 50pF, Figure 4			95			95	ns
End-of-Burst High	C_L = 50pF, Figure 4			75			75	ns
ulse Width	Figure 2	70			70			ns
unter Outputs All Low	Figure 2			100			100	ns
Width of RFCK	Figure 9	100			100			ns
nerator Clock	Figure 3	100			100			ns
Width Low of RGCK	Figure 3	35			40			ns
Width High of RGCK	Figure 3	35			40			ns
ced $\overline{\text{RFRQ}}$ Low	C_L = 50pF, Figure 3		20	30		20	30	ns
ced $\overline{\text{RFRQ}}$ High	C_L = 50pF, Figure 3		50	75		50	75	ns
S Low	Figure 3	50	65	95	50	65	95	ns
S High	Figure 3	40	60	85	40	60	85	ns
from $\overline{\text{RFSH}}$ $\overline{\text{RQST}}$ (RF I/O)	Figure 3	2T			2T			ns
S High (ending forced RFSH)	See Mode 1 Descrip.	55	80	110	55	80	125	ns
to RGCK Low (Mode 1)	See Mode 1 Descrip.	35			40			ns

Switching Characteristics (Cont'd)

Symbol	Refresh Parameter	Conditions	8409			8409 – 3			Units
			Min.	Typ.	Max.	Min.	Typ.	Max.	
t_{CSCT}	$\overline{\text{CS}}$ High to RFSH Counter Valid	Figure 9		55	70		55	75	ns
t_{RHAV}	$\overline{\text{RASIN}}$ High to Input Address Valid	Figure 9		55	95		55	95	ns
t_{CSRL}	$\overline{\text{CS}}$ Low to Access $\overline{\text{RASIN}}$ Low	See Mode 5 Descrip.	10			15			ns
	TRI-STATE® Parameter								
t_{ZH}	$\overline{\text{CS}}$ Low to Address Output High from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{HZ}	$\overline{\text{CS}}$ High to Address Output Hi-Z from High	C_L = 15pF, Figures 9, 12 R1 = 1k, S2 open		20	40		20	40	ns
t_{ZL}	$\overline{\text{CS}}$ Low to Address Output Low from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{LZ}	$\overline{\text{CS}}$ High to Address Output Hi-Z from Low	C_L = 15pF, Figures 9, 12, R2 = 1k, S1 open		25	50		25	50	ns
t_{HZH}	$\overline{\text{CS}}$ Low to Control Output High from Hi-Z High	Figures 9, 12 R2 = 750Ω, S1 open		50	80		50	80	ns
t_{HHZ}	$\overline{\text{CS}}$ High to Control Output Hi-Z High from High	C_L = 15pF, Figures 9, 12 R2 = 750Ω, S1 open		40	75		40	75	ns
t_{HZL}	$\overline{\text{CS}}$ Low to Control Output Low from Hi-Z High	Figure 12, S1, S2 open		45	75		45	75	ns
t_{LHZ}	$\overline{\text{CS}}$ High to Control Output Hi-Z High from Low	C_L = 15pF, Figure 12, R2 = 750Ω, S1 open		50	80		50	80	ns

Input Capacitance T_A = 25°C (Note 2)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
C_{IN}	Input Capacitance ADS, R/$\overline{\text{C}}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing this parameter. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, $t_R = t_F$ = 2.5ns, f = 2.5MHz, t_{PW} = 200ns. Input reference point on AC measurements is 1.5V. Output reference points are 2.7V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50pF.

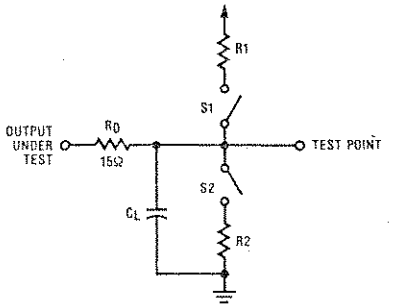


Figure 11. Output Load Circuit

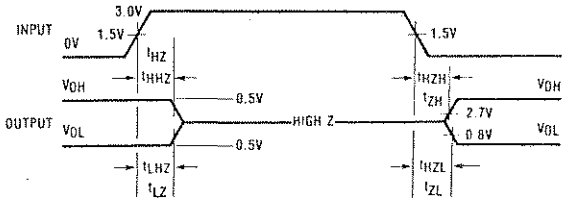


Figure 12. Waveform

ed, the DP8409 may be used
e 6.

efresh are required, then in
res the minimum of external
are ideal, as shown in Figure
provide proper arbitration be-
efresh. This chip supplies all
ls to the processor as well as
vo separate $\overline{CAS}$ outputs are
sing byte-writing. The refresh
down from either RGCK using
M74LS393 or better still, the
resh Timer. The DP84300 can
g from 15.4μs to 15.6μs based
0 MHz. Figure 13b shows the
nterfacing the DP8409 to dif-
ing the interface controller

quired, then Modes 5 and 6
o $\overline{CAS}$ of 125ns and 105ns,
e used for refresh.

If the system is complex, requiring automatic access and refresh, burst refresh, and all-banks auto-write, then more circuitry is required to select the mode. This may be accomplished by utilizing a PAL. The PAL has two functions. One as an address comparator, so that when the desired port address occurs (programmed in the PAL), the comparator gates the data into a latch, where it is connected to the mode pins of the DP8409. Hence the mode of the DP8409 can be changed as desired with one PAL chip merely by addressing the PAL location, and then outputting data to the mode-control pins. In this manner, all the automatic modes may be selected, assigning R/C as RFCK always, and $\overline{CASIN}$ as RGCK always. The output from RF I/O may be used as End-of-Count to an interrupt, or $\overline{Refresh\ Request}$ to $\overline{HOLD}$ or $\overline{BUS\ REQUEST}$. A complex system may use Modes 5 and 1 for automatic access and refresh, Modes 3a and 7 for system initialization, and Mode 2 (auto-burst refresh) before and after DMA.

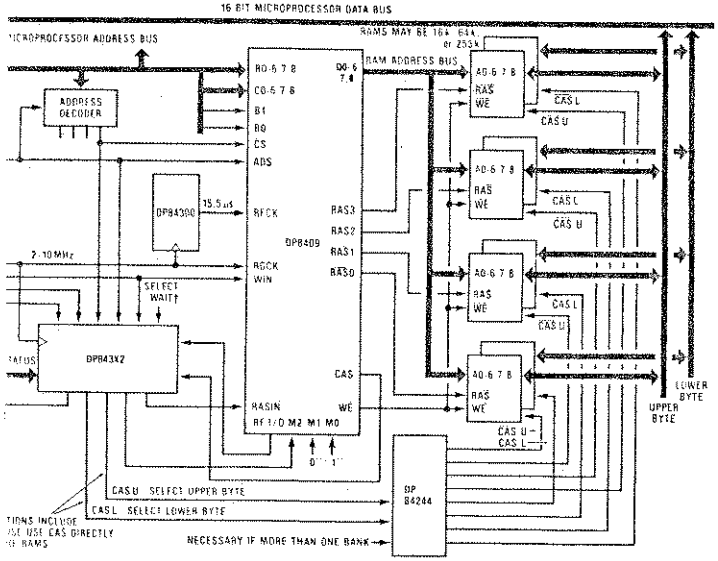


Figure 13a. Connecting the DP8409 between the 16-bit Microprocessor and Memory

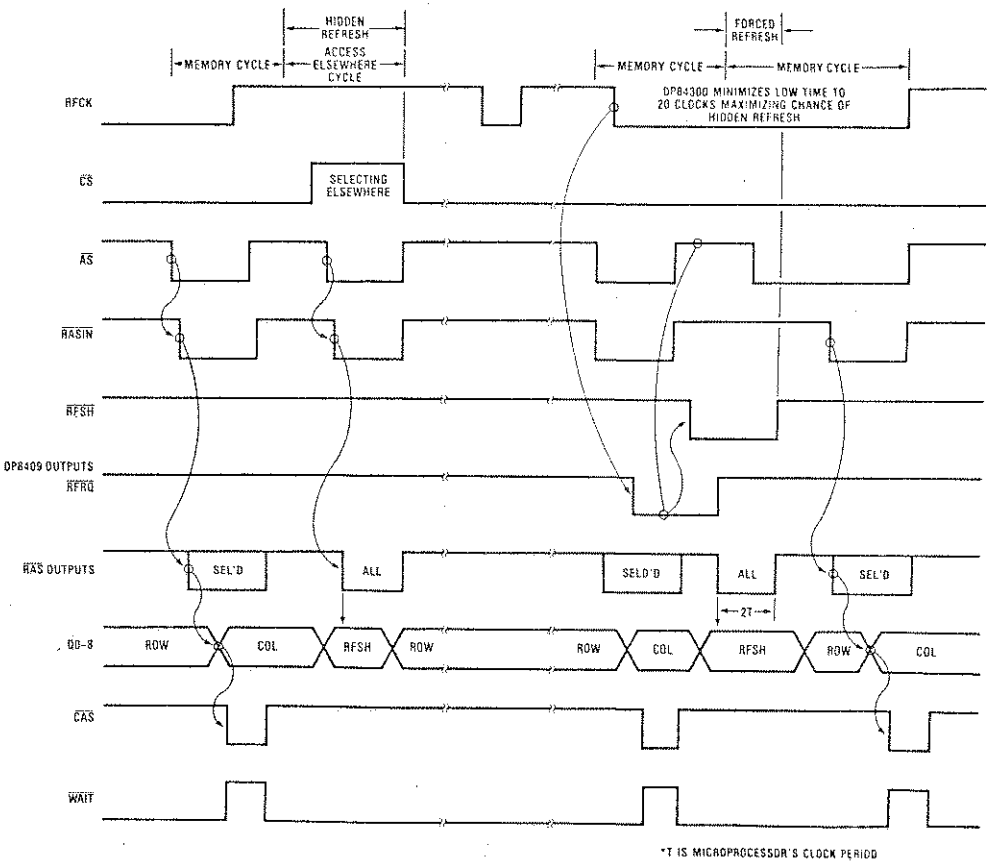
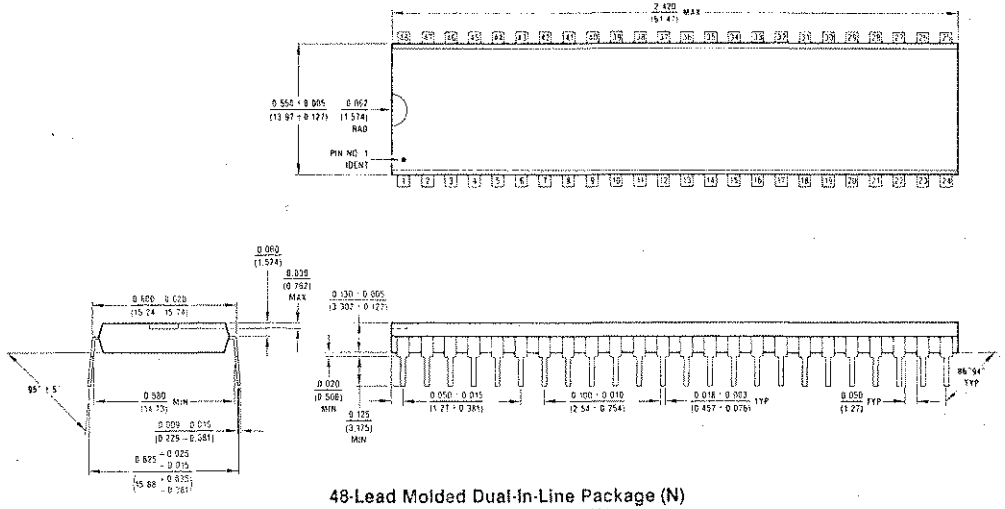
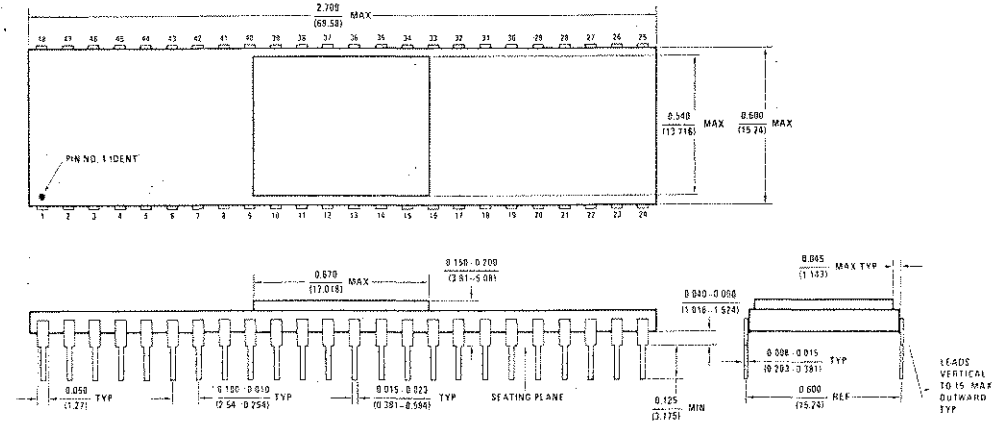


Figure 13b. DP8409 Auto Refresh

Physical Dimensions (inches/millimeters)



48-Lead Molded Dual-In-Line Package (N)
Order Number DP8409N
or Order Number DP8409N-3
NS Package N48A



48-Lead Dual-In-Line Package (D)
Order Number DP8409D
NS Package D48A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

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

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 737 5000

National Semiconductor GmbH
Furstenriederstrasse Nr 5I
8 Munchen 21
West Germany
Tel: (089) 58 30 51

NS Japan K K.
POB 4152
Shinjuku Center Building,
1-25-1 Nishishinjuku,
Shinjuku-ku,

National Semiconductor Hong Kong, Ltd.
1st Floor
Cheung Kong Electronic Bldg
4 Hong Yip Street

NS Electronics Do Brasil
Avda Brigadeiro Faria Lima 830
8 Andar
01452 Sao Paulo Brasil
Telex

NS Electronics Pty. Ltd.
Cnr. Stud Rd. & Mtn. Highway
Bayswater Victoria 3153
Australia
Tel: 03 729 6333

Am9511A

Arithmetic Processor

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- 2, 3 and 4MHz operation
- 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

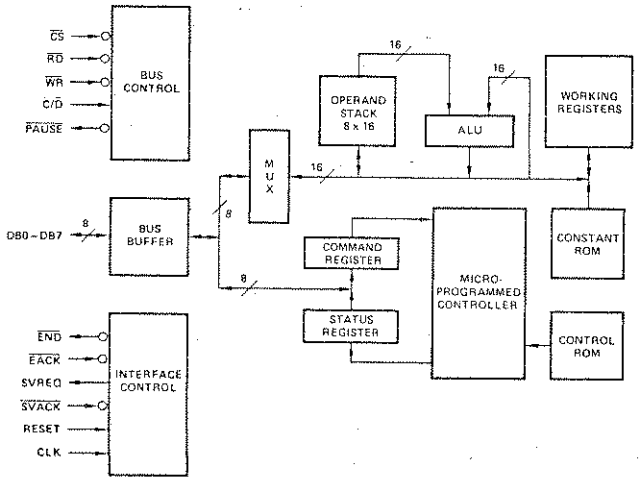
GENERAL DESCRIPTION

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.

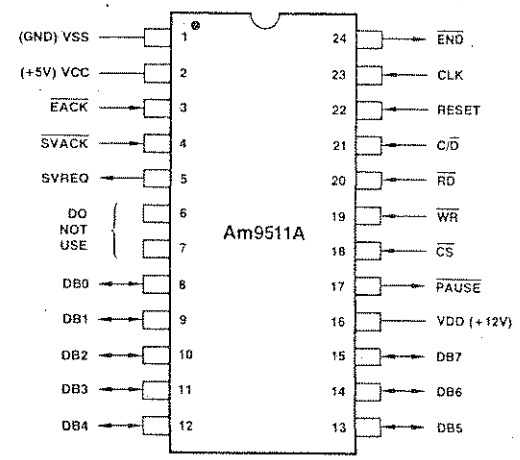
Transfers to and from the APU may be handled by the associated processor using conventional programmed I/O, or may be handled by a direct memory access controller for improved performance. Upon completion of each command, the APU issues an end of execution signal that may be used as an interrupt by the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

BLOCK DIAGRAM



MOS-046

CONNECTION DIAGRAM
Top View



Note: Pin 1 is marked for orientation.

MOS-047

ORDERING INFORMATION

Package Type	Ambient Temperature	Maximum Clock Frequency		
		2MHz	3MHz	4MHz
Hermetic DIP	0°C ≤ T_A ≤ +70°C	Am9511ADC	Am9511A-1DC	Am9511A-4DC
	−40°C ≤ T_A ≤ +85°C	Am9511ADI	Am9511A-1DI	
	−55°C ≤ T_A ≤ +125°C	Am9511ADM	Am9511A-1DM	

March 1982

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, Input)

This input when LOW makes the $\overline{END}$ output go LOW. As mentioned earlier LOW 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.

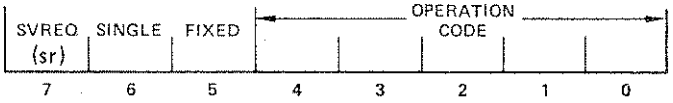
Communication between the external world and the Am9511A takes place on eight bidirectional input/output lines DB0 through DB7 (Data Bus). These signals are gated to the internal eight-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication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

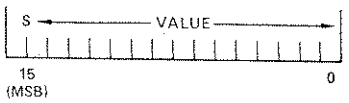
The following procedure must be followed to accomplish status register reading.

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).

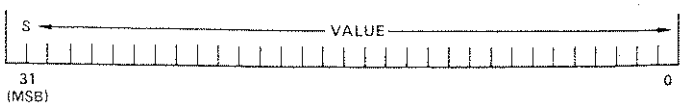
DATA FORMATS

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 accomodated by each of these formats is −32,767 to -32,767 for single precision and −2,147,483,647 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 (, 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 man- value between .5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

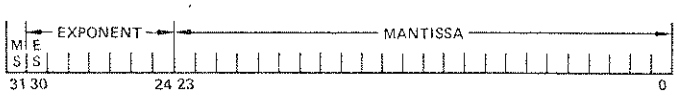
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

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

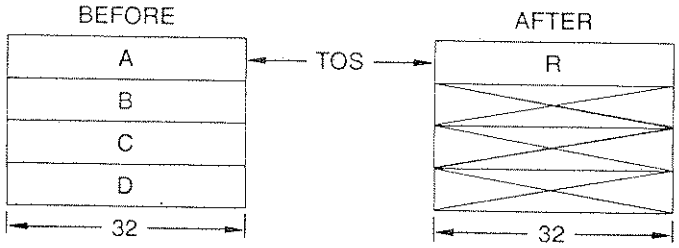
Execution Time: 6304 to 8284 clock cycles

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 2.0×10^{-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

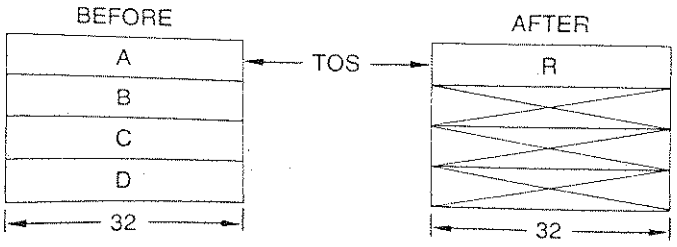
Execution Time: 6230 to 7938 clock cycles

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 4.0×10^{-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

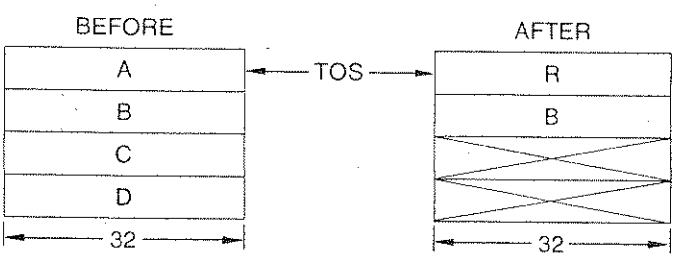
Execution Time: 4992 to 6536 clock cycles

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 3.0×10^{-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

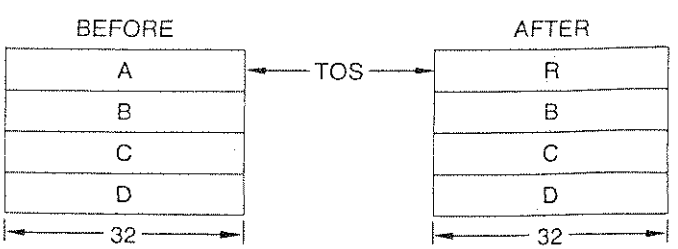
Hex Coding: B4 with sr = 1
34 with sr = 0

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

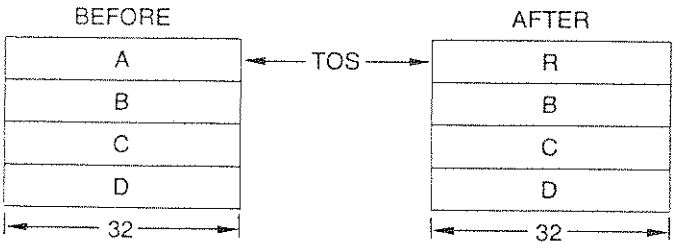
Description:

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

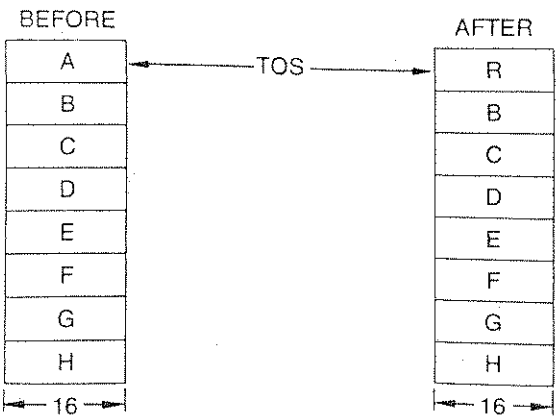
Description:

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:

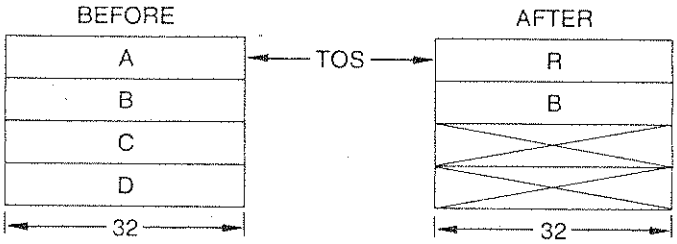
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point cosine of A. A is assumed to be in radians. Operands A, C and D are lost. B is unchanged.

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

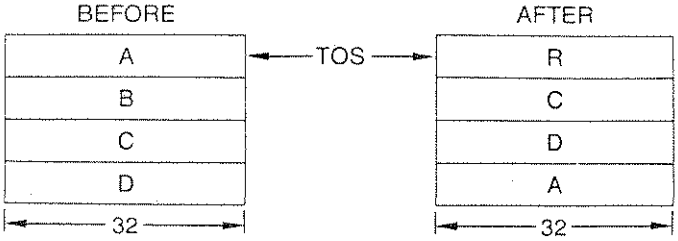
Description:

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

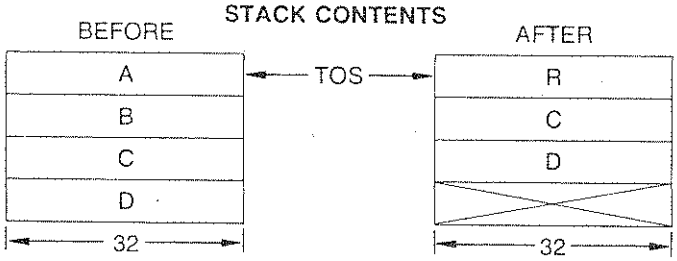
Hex Coding: AF with sr = 1
2F with sr = 0

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

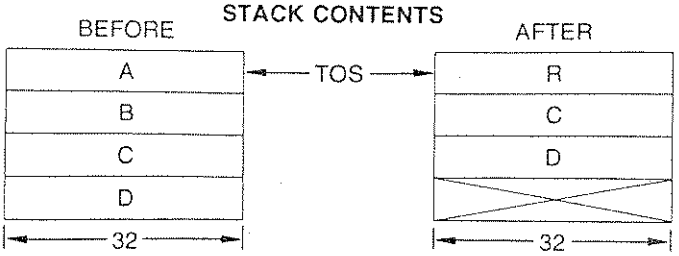
Hex Coding: AE with sr = 1
2E with sr = 0

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

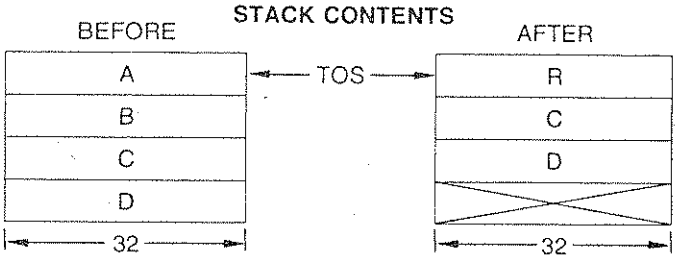
Hex Coding: B6 with sr = 1
36 with sr = 0

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

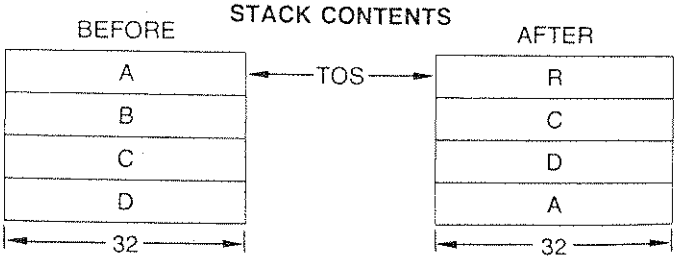
Hex Coding: AD with sr = 1
2D with sr = 0

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

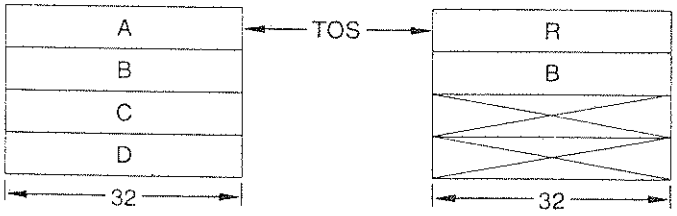
Execution Time: 3794 to 4878 clock cycles for $|A| \leq 1.0 \times 2^5$
34 clock cycles for $|A| > 1.0 \times 2^5$

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

STACK CONTENTS



FADD

32-BIT FLOATING-POINT ADD

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

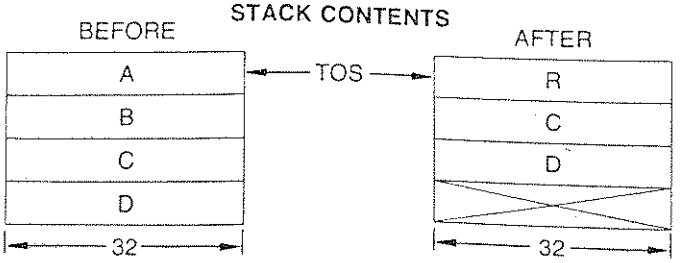
Hex Coding: 90 with sr = 1
10 with sr = 0

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

STACK CONTENTS



FDIV

32-BIT FLOATING-POINT DIVIDE

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

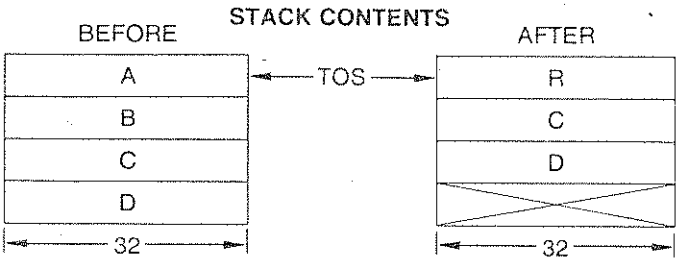
Hex Coding: 93 with sr = 1
13 with sr = 0

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

STACK CONTENTS



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

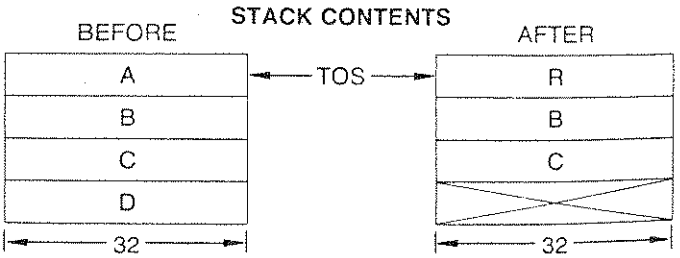
Execution Time: 90 to 336 clock cycles

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

STACK CONTENTS



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

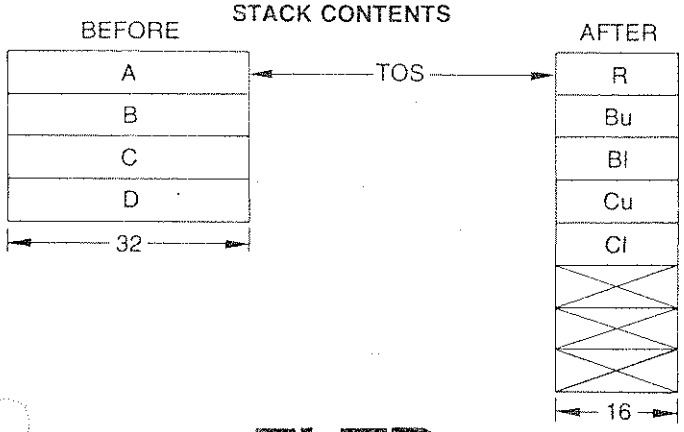
Description:

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 st. 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

STACK CONTENTS



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

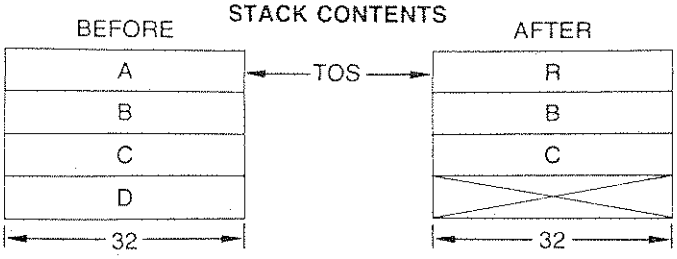
Execution Time: 56 to 342 clock cycles

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

STACK CONTENTS



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

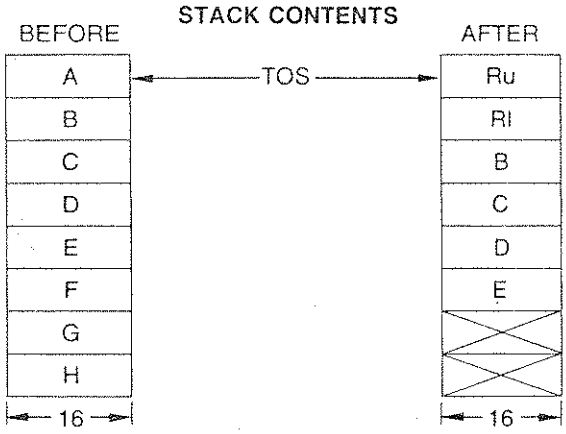
Execution Time: 62 to 156 clock cycles

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

STACK CONTENTS



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

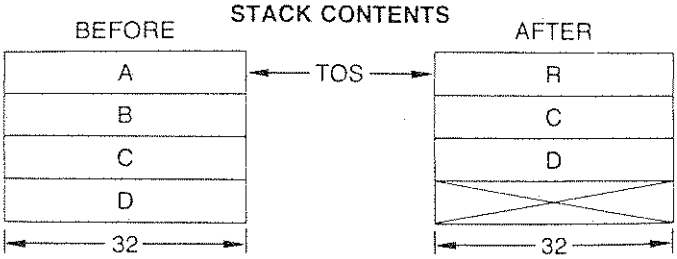
Description:

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

STACK CONTENTS



FSUB

32-BIT FLOATING-POINT SUBTRACTION

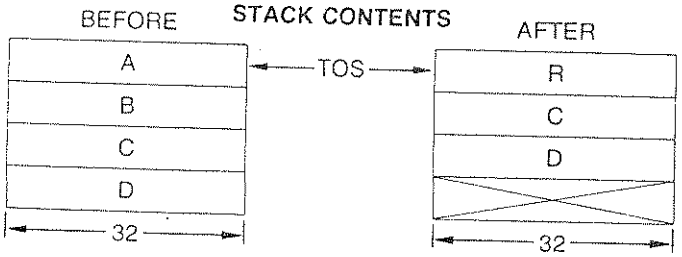
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	1

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

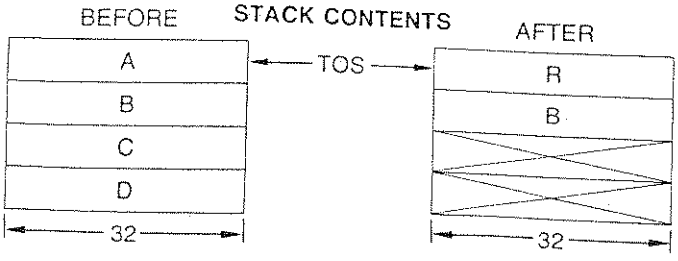
Hex Coding: 88 with sr = 1
08 with sr = 0

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

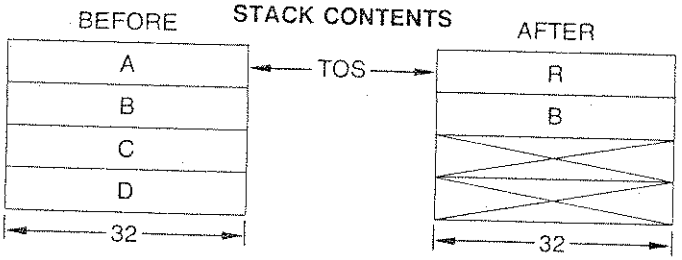
Hex Coding: 89 with sr = 1
09 with sr = 0

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

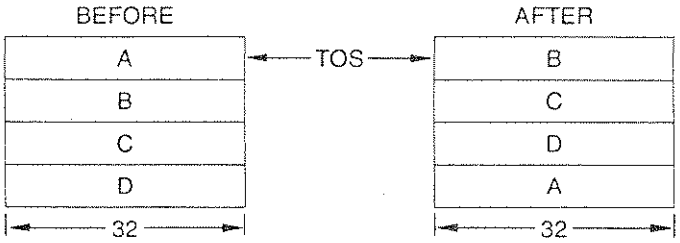
Hex Coding: B8 with sr = 1
38 with sr = 0

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

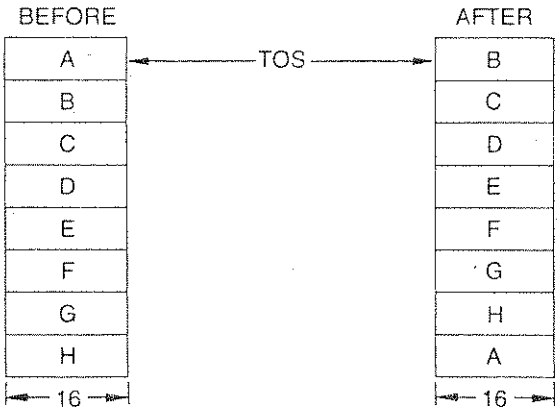
Hex Coding: F8 with sr = 1
78 with sr = 0

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

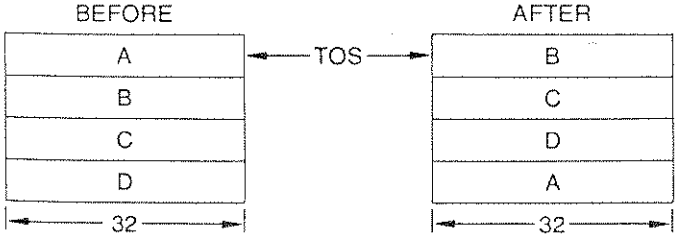
Hex Coding: 98 with sr = 1
18 with sr = 0

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

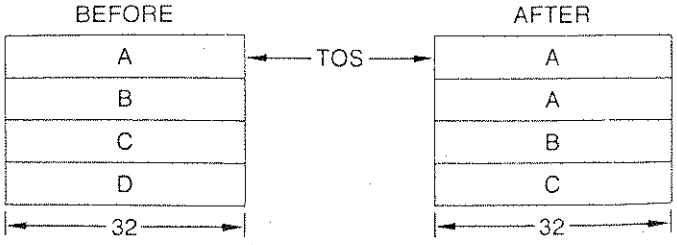
Hex Coding: B7 with sr = 1
37 with sr = 0

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

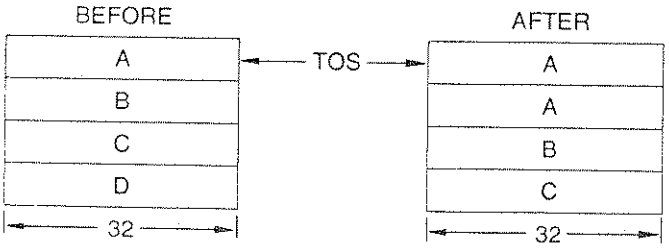
Hex Coding: 97 with sr = 1
17 with sr = 0

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

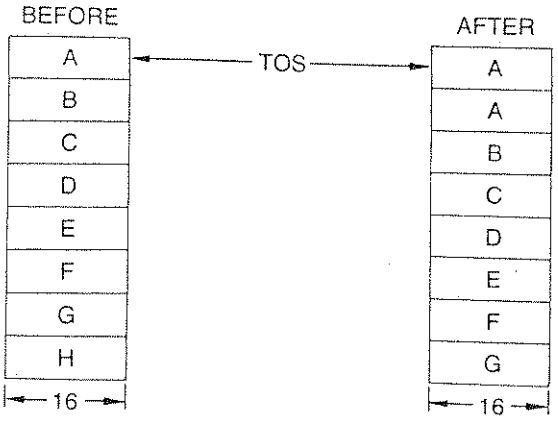
Hex Coding: F7 with sr = 1
77 with sr = 0

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

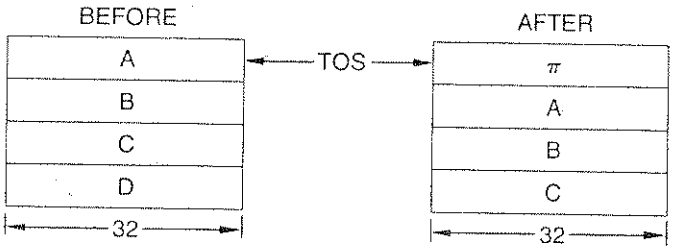
Hex Coding: 9A with sr = 1
1A with sr = 0

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.

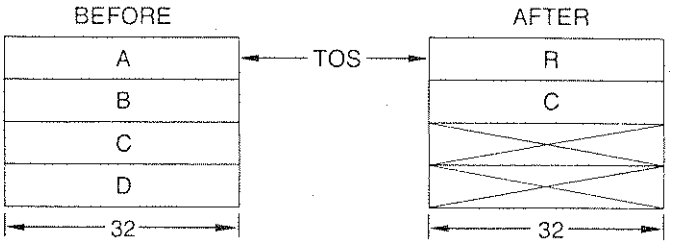
Accuracy: The error performance for PWR is a function of the LN and EXP performance as expressed by:

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

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

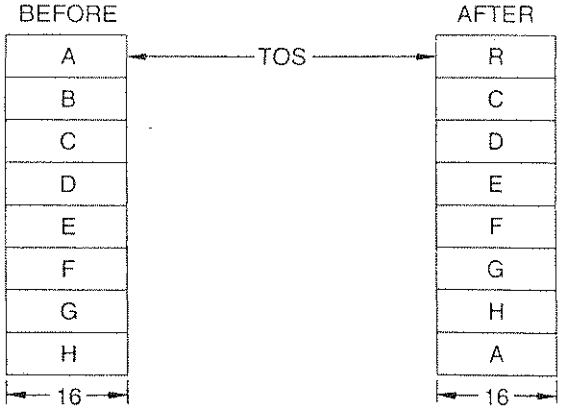
Description:

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

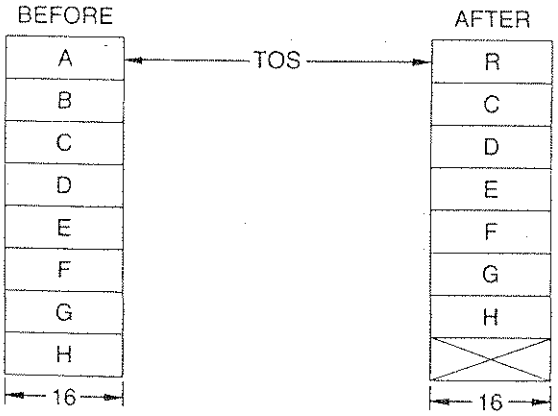
Hex Coding: EF with sr = 1
6F with sr = 0

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

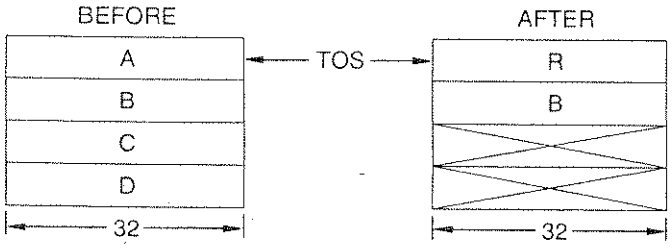
Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point sine of A. A is assumed to be in radians. Operands A, C and D are lost. Operand B is unchanged.

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 5.0×10^{-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

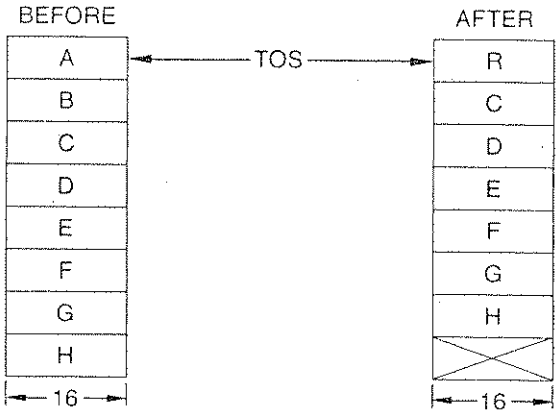
Execution Time: 84 to 94 clock cycles

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

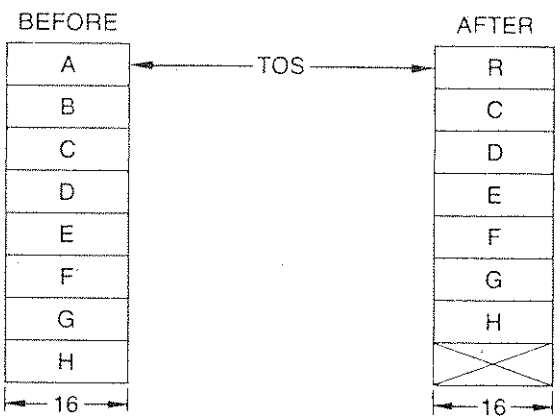
Description:

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

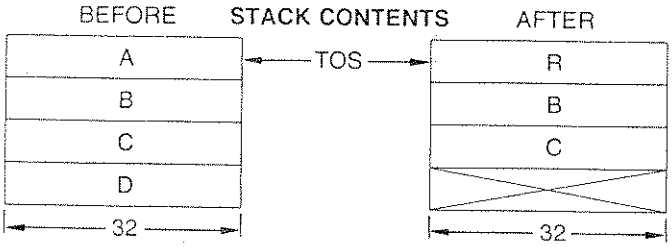
Execution Time: 782 to 870 clock cycles

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

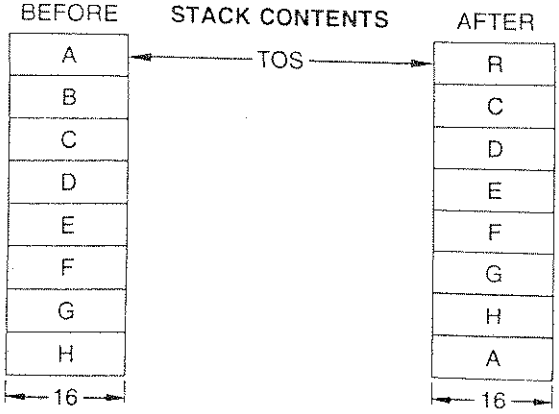
Execution Time: 30 to 32 clock cycles

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

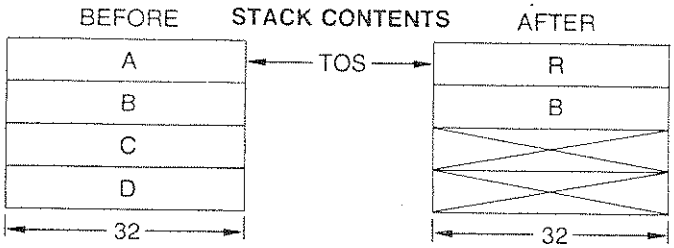
Description:

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

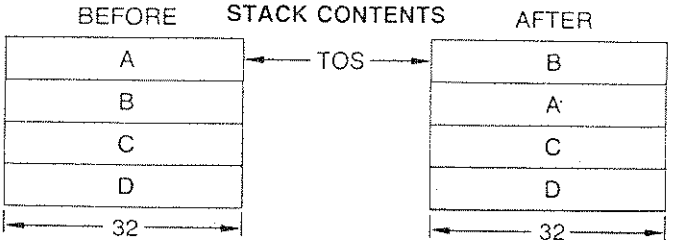
Hex Coding: B9 with sr = 1
39 with sr = 0

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

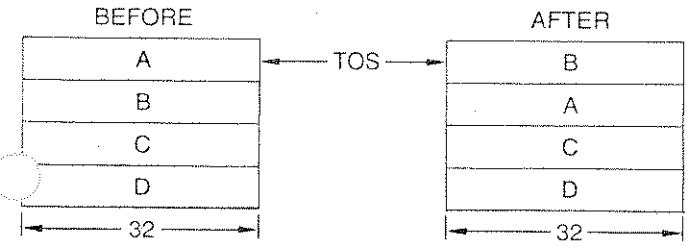
Hex Coding: 99 with sr = 1
19 with sr = 0

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

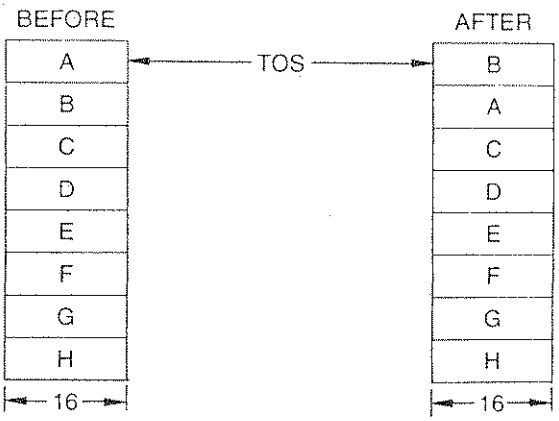
Hex Coding: F9 with sr = 1
79 with sr = 0

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
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
Am9511ADC	0°C ≤ T_A ≤ 70°C	0V	+5.0V ±5%	+12V ±5%
Am9511A-1DC	0°C ≤ T_A ≤ 70°C	0V	+5.0V ±5%	+12V ±5%
Am9511A-4DC	0°C ≤ T_A ≤ 70°C	0V	+5.0V ±5%	+12V ±5%
Am9511ADI	−40°C ≤ T_A ≤ 85°C	0V	+5.0V ±10%	+12V ±10%
Am9511A-1DI	−40°C ≤ T_A ≤ 85°C	0V	+5.0V ±10%	+12V ±10%
Am9511ADM	−55°C ≤ T_A ≤ 125°C	0V	+5.0V ±10%	+12V ±10%
Am9511A-1DM	−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
	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

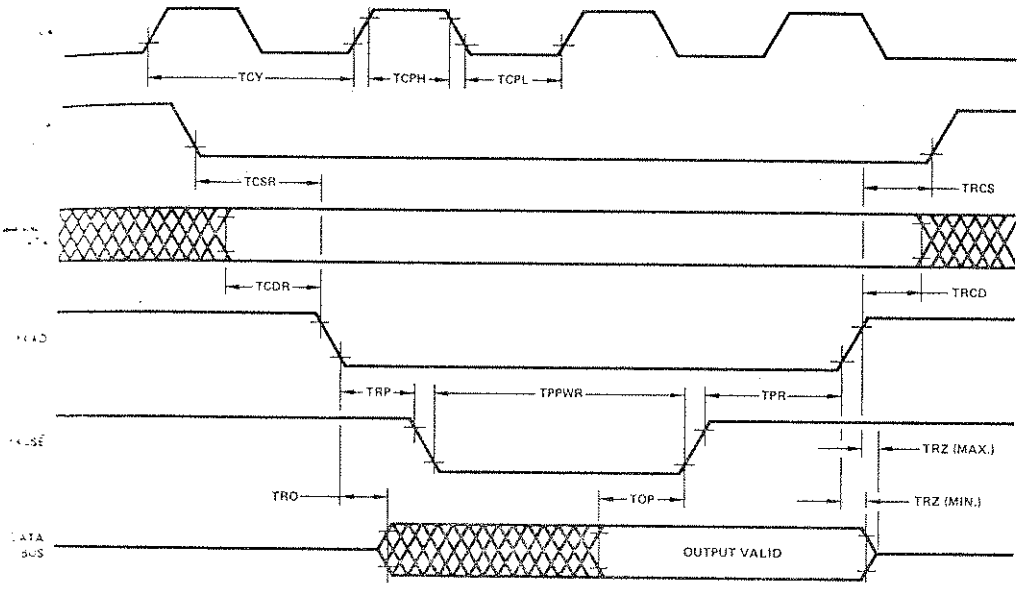
SWITCHING CHARACTERISTICS

Parameters	Description		Am9511A Min	Am9511A Max	Am9511A-1 Min	Am9511A-1 Max	Am9511A-4 Min	Am9511A-4 Max	Units
TAPW	$\overline{EACK}$ LOW Pulse Width		100		75		50		ns
TCDR	C/$\overline{D}$ to $\overline{RD}$ LOW Set-up Time		0		0		0		ns
TCDW	C/$\overline{D}$ to $\overline{WR}$ LOW Set-up Time		0		0		0		ns
TCPH	Clock Pulse HIGH Width		200		140		100		ns
TCPL	Clock Pulse LOW Width		240		160		120		ns
TCSR	$\overline{CS}$ LOW to $\overline{RD}$ LOW Set-up Time		0		0		0		ns
TCSW	$\overline{CS}$ LOW to $\overline{WR}$ LOW Set-up Time		0		0		0		ns
TCY	Clock Period		480	5000	320	3300	250	2500	ns
TDW	Data Bus Stable to $\overline{WR}$ HIGH Set-up Time		150		100 (Note 9)		100		ns
TEAE	$\overline{EACK}$ LOW to $\overline{END}$ HIGH Delay			200		175		150	ns
TEPW	$\overline{END}$ LOW Pulse Width (Note 4)		400		300		200		ns
TOP	Data Bus Output Valid to $\overline{PAUSE}$ HIGH Delay		0		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	3.5TCY+50	5.5TCY+200	ns
		Status	1.5TCY+50	3.5TCY+300	1.5TCY+50	3.5TCY+200	1.5TCY+50	3.5TCY+200	
TPPWW	$\overline{PAUSE}$ LOW Pulse Width Write (Note 8)			50		50		50	ns
TPR	$\overline{PAUSE}$ HIGH to $\overline{RD}$ HIGH Hold Time		0		0		0		ns
TPW	$\overline{PAUSE}$ HIGH to $\overline{WR}$ HIGH Hold Time		0		0		0		ns
TRCD	$\overline{RD}$ HIGH to C/$\overline{D}$ Hold Time		0		0		0		ns
TRCS	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH Hold Time		0		0		0		ns
TRO	$\overline{RD}$ LOW to Data Bus ON Delay		50		50		25		ns
TRP	$\overline{RD}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)		100	ns
TRZ	$\overline{RD}$ HIGH to Data Bus OFF Delay		50	200	50	150	25	100	ns
TSAPW	$\overline{SVACK}$ LOW Pulse Width		100		75		50		ns
TSAR	$\overline{SVACK}$ LOW to SVREQ LOW Delay			300		200		150	ns
TWCD	$\overline{WR}$ HIGH to C/$\overline{D}$ Hold Time		60		30		30		ns
TWCS	$\overline{WR}$ HIGH to $\overline{CS}$ HIGH Hold Time		60		30		30		ns
TWD	$\overline{WR}$ HIGH to Data Bus Hold Time		20		20		20		ns
TWI	Write Inactive Time	Command	3TCY		3TCY		3TCY		ns
		Data	4TCY		4TCY		4TCY		
TWP	$\overline{WR}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)		100	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 requirement (TWI) is observed and no previous command is being executed. TWI may be safely violated up to 500ns 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 the Am9511A-1DM.

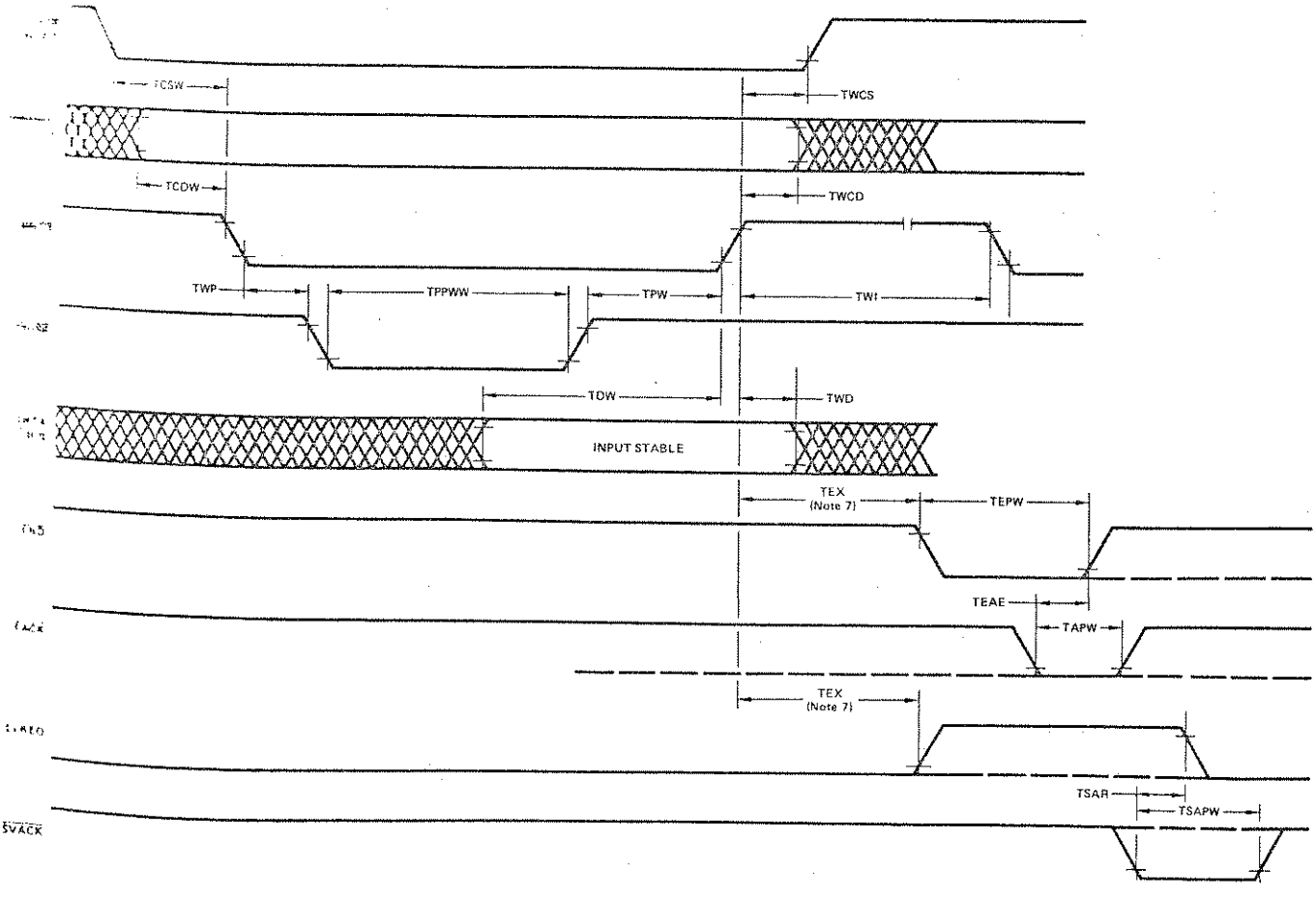
SWITCHING WAVEFORMS

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.

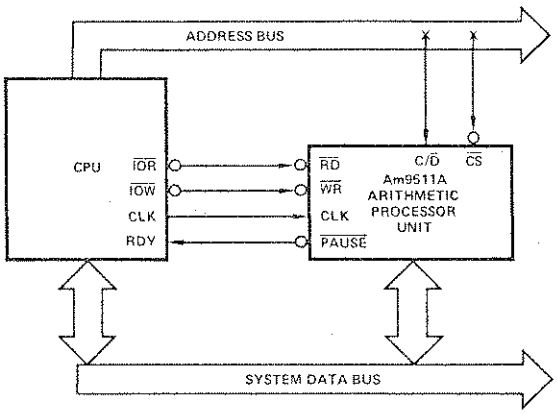


Figure 1. Am9511A Minimum Configuration Example.

MOS-050

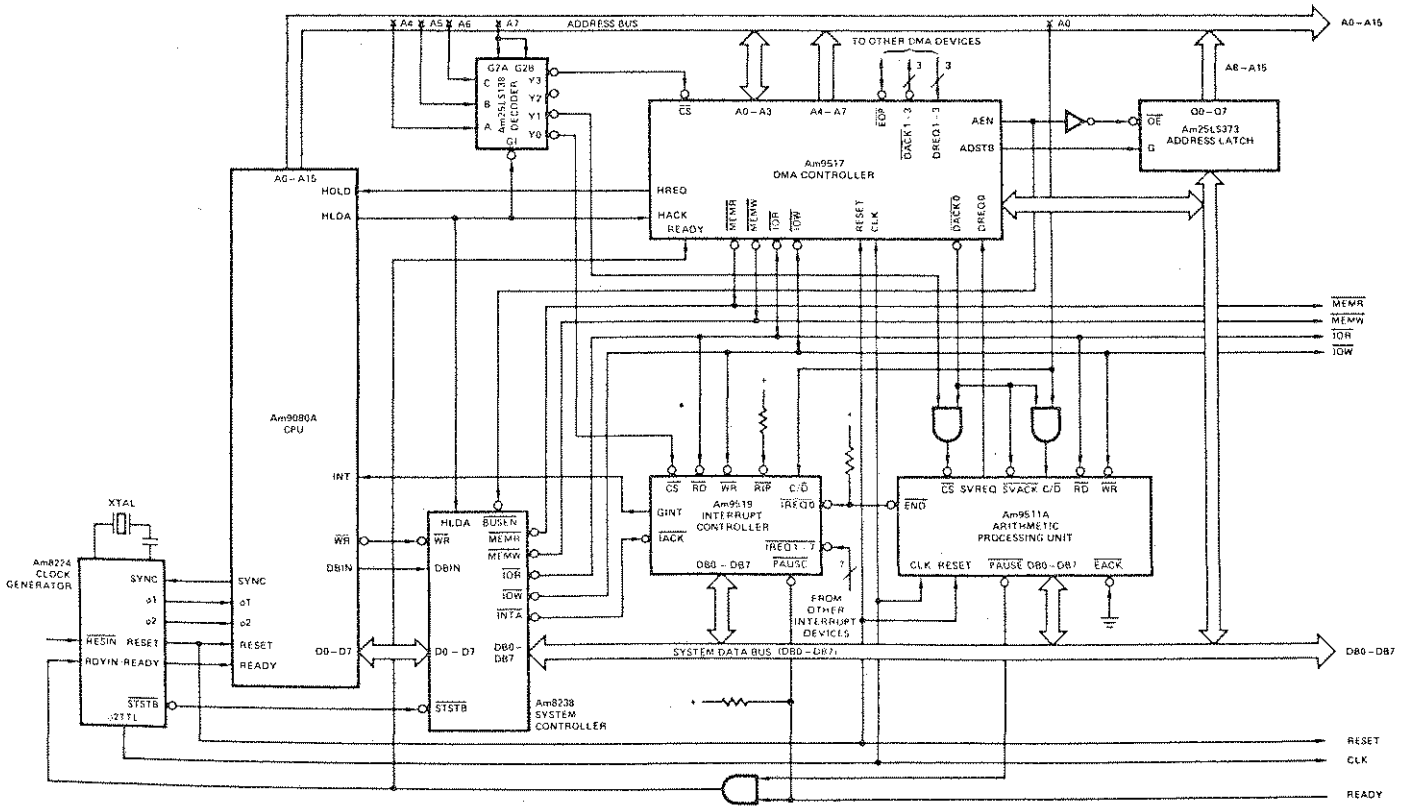
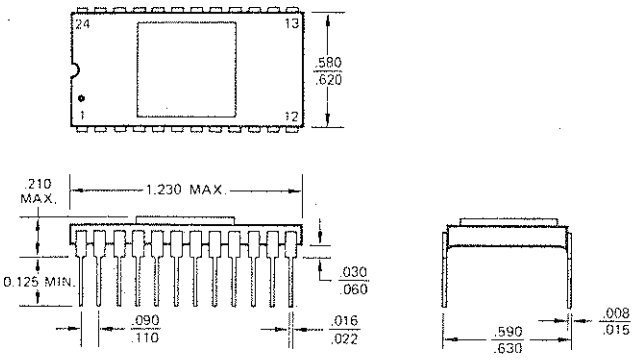


Figure 2. Am9511A High Performance Configuration Example.

MOS-051

PHYSICAL DIMENSIONS
Dual-In-Line

24-Pin Side-Brazed



The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs; 0.2% on Bipolar Logic & Interface; 0.3% on Linear, LSI Logic & other memories.

ADVANCED MICRO DEVICES

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