TURBO MAC : HARDWARE MEMORY MAP

Burrell Smith and Brian Howard

17 October 1984

----- CONFIDENTIAL ---------- CONFIDENTIAL -----

Previous Document Versions: 19-Sep-84, 1Ø-Sep-1984.

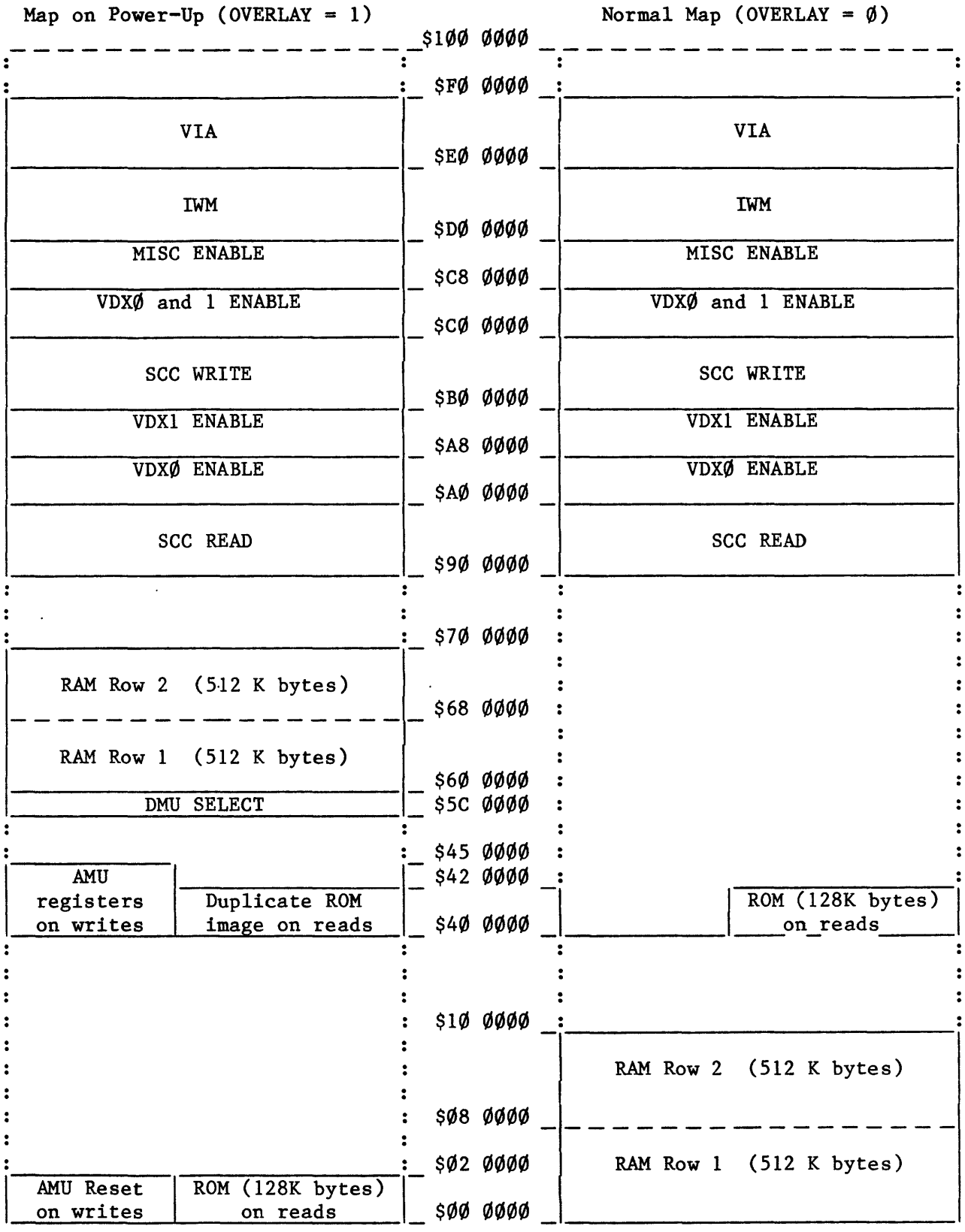
1. INTRODUCTION

The principle portions of Turbo Mac's address space consist of volatile read/write memory (RAM) and permanent read-only memory (ROM). In addition to RAM and ROM, several input/output functions are also selected using address lines, so that they appear to occupy portions of the Turbo Mac "memory". These include the 6522 Versatile Interface Adapter (VIA), the 853Ø Serial Communications Chip (SCC), the disk interface chip (IWM), the Address Management Unit (AMU), the Data Management Unit (DMU), and the Random Logic Control.

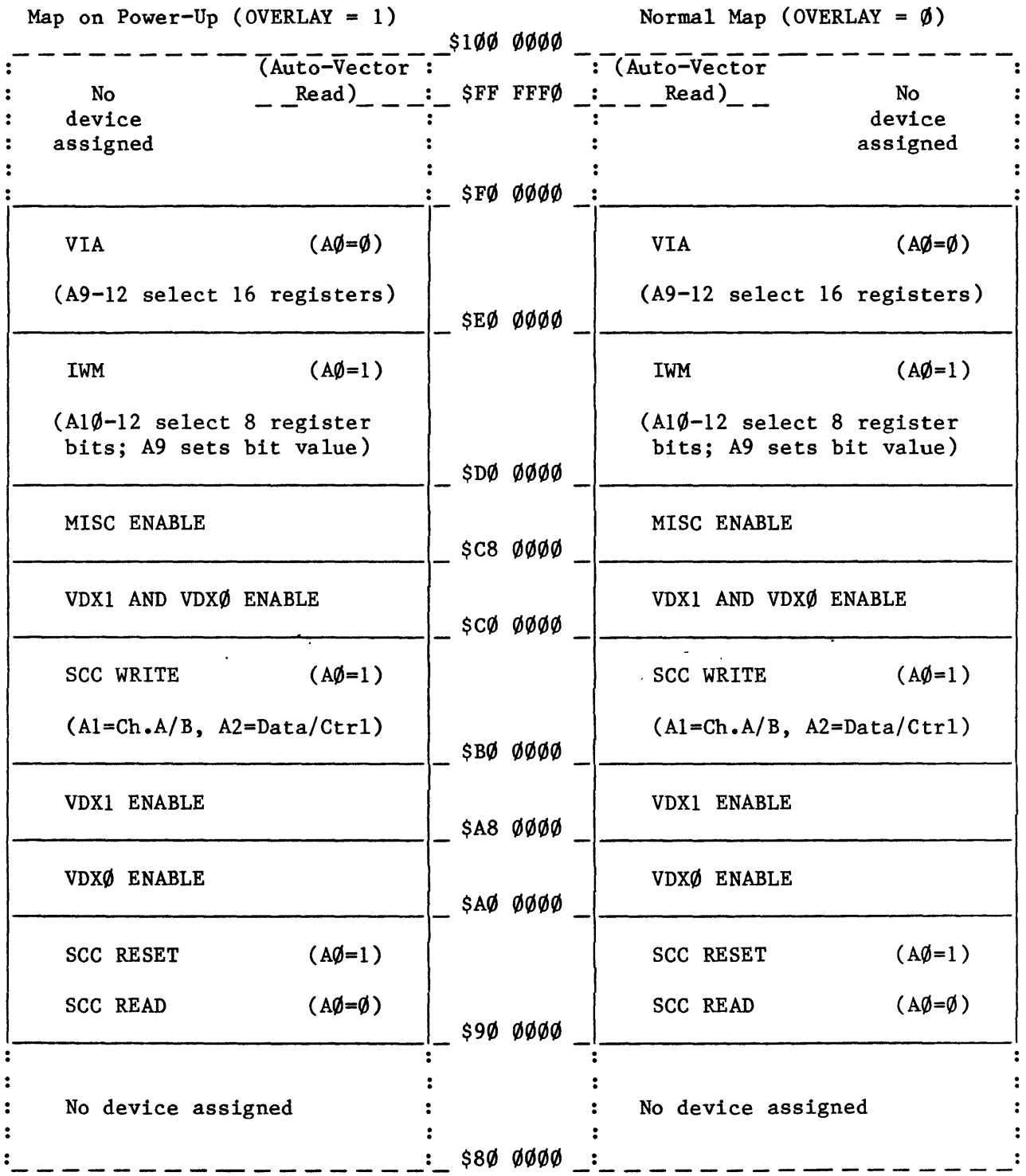
When the Turbo Mac is first turned on, ROM appears at the bottom (lowest addresses) portion of the address space. This is useful for the ROM-stored software which starts the system running. After startup, the OVERLAY signal from the VIA is changed to a low (zero), mapping RAM into its normal place at the bottom of the address space.

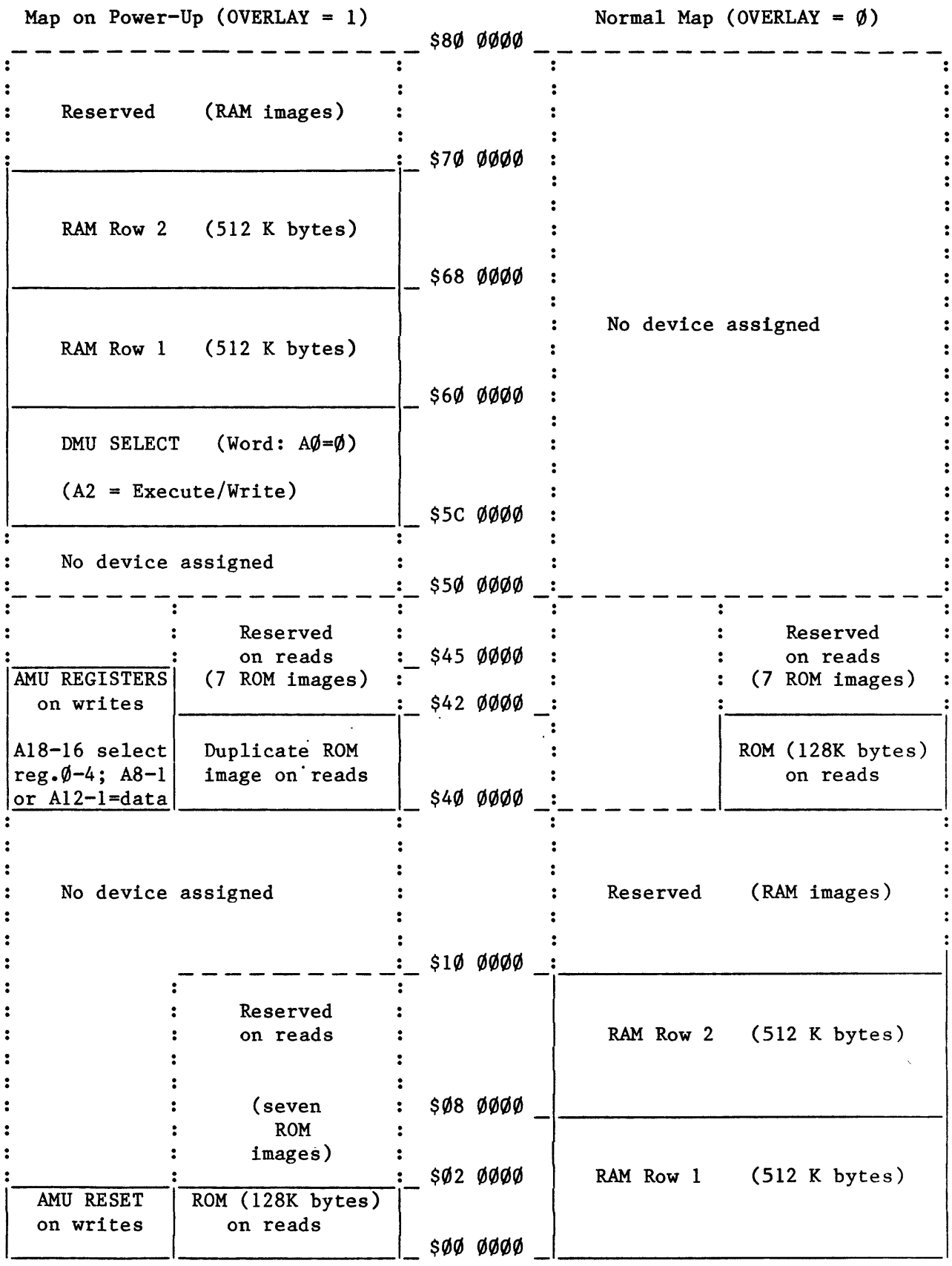
Selection of RAM, ROM, or other devices is done by from two to seven of the highest-order address lines, A23-A17. The VIA and IWM also use the four address lines A12-A9 for further internal decoding and register selection, while the SCC uses the three lowest-order address lines A2-AØ for internal decoding. When selecting certain AMU registers, the information on address lines A12-A1 or A8-A1 is used as data for the selected register. When selecting the DMU, address line A2 determines how the DMU control register is used.

2. THE TURBO MAC ADDRESS SPACE (Overview)



2.1 More Detailed Map of Turbo Mac Address Space (two pages)




Map on Power-Up (OVERLAY = 1)
Normal Map (OVERLAY = Ø)
$8Ø ØØØØ
Reserved (RAM images)
$7Ø ØØØØ
RAM Row 2 (512 K bytes)
$68 ØØØØ
RAM Row 1 (512 K bytes)
$6Ø ØØØØ
DMU SELECT (Word: AØ=Ø)
(A2 = Execute/Write)
$5C ØØØØ
No device assigned
$5Ø ØØØØ
Reserved on reads (7 ROM images)
$45 ØØØØ
AMU REGISTERS on writes
A18-16 select reg.Ø-4; A8-1 or A12-1=data
Duplicate ROM image on reads
$42 ØØØØ
ROM (128K bytes) on reads
$4Ø ØØØØ
No device assigned
Reserved (RAM images)
$1Ø ØØØØ
Reserved on reads
(seven ROM images)
$Ø8 ØØØØ
AMU RESET on writes
ROM (128K bytes) on reads
$Ø2 ØØØØ
RAM Row 2 (512 K bytes)
RAM Row 1 (512 K bytes)
$ØØ ØØØØ

2.2 ADDRESS LINE DECODING (two pages)

Address Lines

(Blank = Don't Care, x = Further Decoding)

Device Selected	23	22	21	2Ø	19	18	17	16	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø
RAM (OVERLAY=1)	Ø	1	1			<-	----	--->	----	----	512	K	bytes	(one			row)	---			----	----	----	->
						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
(OVERLAY=Ø)	Ø	Ø			<-	----	----	--->	----	----	1	M	bytes	(two			rows)	--			----	----	----	->
					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
ROM (OVERLAY=1) Reads, only	Ø	Ø	Ø	Ø				<-	----	----	----	----	128K	bytes			----	----	----	----	----	----	----	->
								x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
(Anytime) Reads, only	Ø	1	Ø	Ø																				
AMU Reset (OVERLAY=1) Writes, only	Ø	Ø	Ø	Ø	Ø	Ø	Ø																	
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AMU Register (OVERLAY=1) Writes, only	Ø	1	Ø	Ø	Ø	AMU reg. <-----> x	x	x				<- x	x	x	x	x	reg.Ø,2 x x x x x x x x				----> x x x x x x			
																<-	-- Data for reg.				----> 1,3,4			
DMU Select (OVERLAY=1)	Ø	1	Ø	1	1	1											Execute				/Write | x		Ø
Number of Bytes Decoded:	16 M	8 M	4 M	2 M	M	1 2 K	5 6 K	2 8 K	4 K	2 K	6 K	8	K	K	K	1 2	5 6	2 8	4	2	6	1	8	4 2

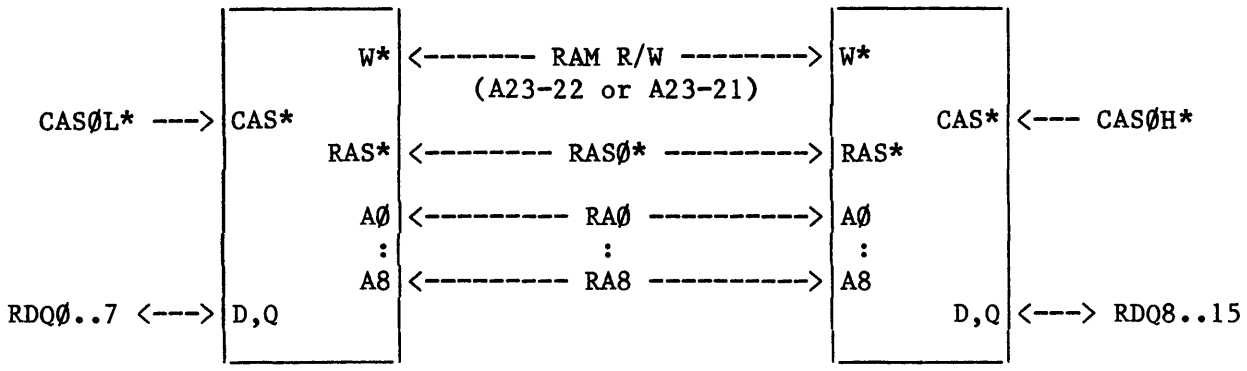
Address Lines

(Blank = Don't Care, x = Further Decoding)

Device Selected	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	Data/				Cmd		Ch. A/B	
SCC Read	1	0	0	1																	x	x	0	
SCC Reset (Anytime)	1	0	0	1																	x	x	1	
SCC Write (Anytime)	1	0	1	1																	x	x	1	
VDX0 Enable (Anytime)	1	0	1	0	0																			
VDX1 Enable (Anytime)	1	0	1	0	1																			
VDX0 & 1 En (Anytime)	1	1	0	0	0																			
MISC Enable (Anytime)	1	1	0	0	1																			
									IWM Reg. Bit <-				0..7 ----->			Bit |	Value							
IWM (Anytime)	1	1	0	1								x	x	x	x									1
											VIA <-		Reg. 0..15 ------->											
VIA (Anytime)	1	1	1	0								x	x	x	x									0
Number of Bytes Decoded:	16M	8M	4M	2M	1M	512K	256K	128K	64K	32K	16K	8K	4K	2K	1K	512	256	128	64	32	16	8	4	2

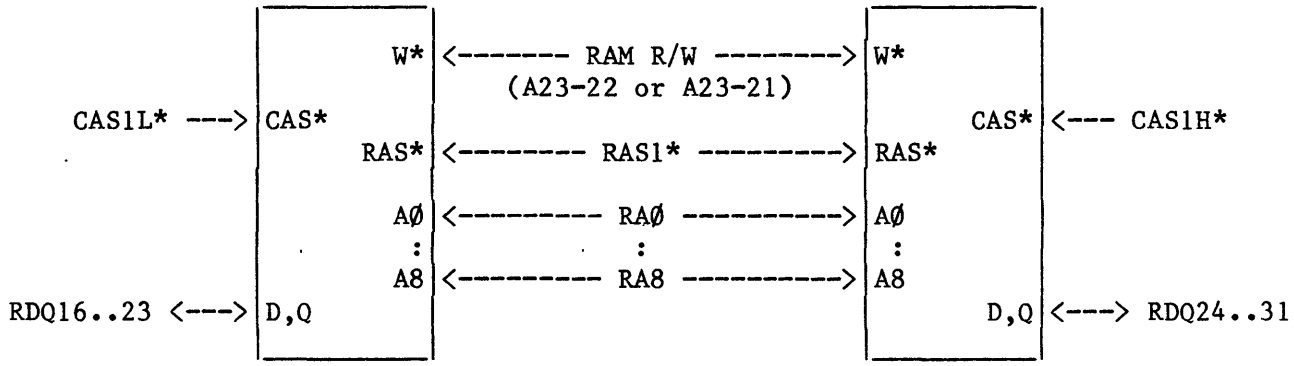
3. RAM

RAM Row #Ø



Eight 256K-bit DRAMs Eight 256K-bit DRAMs

RAM Row #1 (optional)



Eight 256K-bit DRAMs Eight 256K-bit DRAMs

3.1 Address Decoding to Activate RAMs

When RAM Addressed	Address Lines A23	A22	A21	A2Ø	Address Range
Startup: OVERLAY=1	Ø	1	1	x	$6Ø ØØØØ - $7F FFFF
Normal: OVERLAY=Ø	Ø	Ø	x	x	$ØØ ØØØØ - $3F FFFF

(Note: x indicates "don't care": either 1 or Ø)

3.2 Further RAM Address Decoding

The following table details RAM's use of address lines A19-AØ:

System Memory		Address Lines 19 18 17 16	15 14 13 12	11 1Ø 9 8	7 6 5 4	3 2 1 Ø
512K	Vid	x C1 CØ C8	C6 C5 C4 C3	C2 R8 R7 RØ	R6 R5 R4 R3	R2 R1 C7 B
	DMA	x C1 CØ C7	C6 C5 C4 C3	C2 R8 R7 RØ	R6 R5 R4 R3	R2 R1 C8 B
1 M	Vid	C8 C1 CØ W	C6 C5 C4 C3	C2 R8 R7 RØ	R6 R5 R4 R3	R2 R1 C7 B
	DMA	C8 C1 CØ C7	C6 C5 C4 C3	C2 R8 R7 RØ	R6 R5 R4 R3	R2 R1 W B

Note 1: Rn = row address, bit n; Cn = column address, bit n;
W = bank (RAM Row #Ø or Row #1); B = byte (even or odd);
x = don't care (ignored by decoding hardware).

Note 2: Vid = CPU references to top 128K bytes of memory;
DMA = CPU references to all remaining memory, and any DMA.

The following table organizes the same information in order of RAM addresses:

RAM Row Address	CPU or DMA Address	RAM Column Address	CPU or DMA Address
RA Ø	A 8	CA Ø	A 17
RA 1	A 2	CA 1	A 18
RA 2	A 3	CA 2	A 11
RA 3	A 4	CA 3	A 12
RA 4	A 5	CA 4	A 13
RA 5	A 6	CA 5	A 14
RA 6	A 7	CA 6	A 15
RA 7	A 9	CA 7	A1 (Vid) / A16 (DMA)
RA 8	A 1Ø	CA 8	A16 (Vid) / A1 (DMA) if 512 K, or A19 if 1 M Byte System

Note 2, above, explains "Vid" and "DMA"

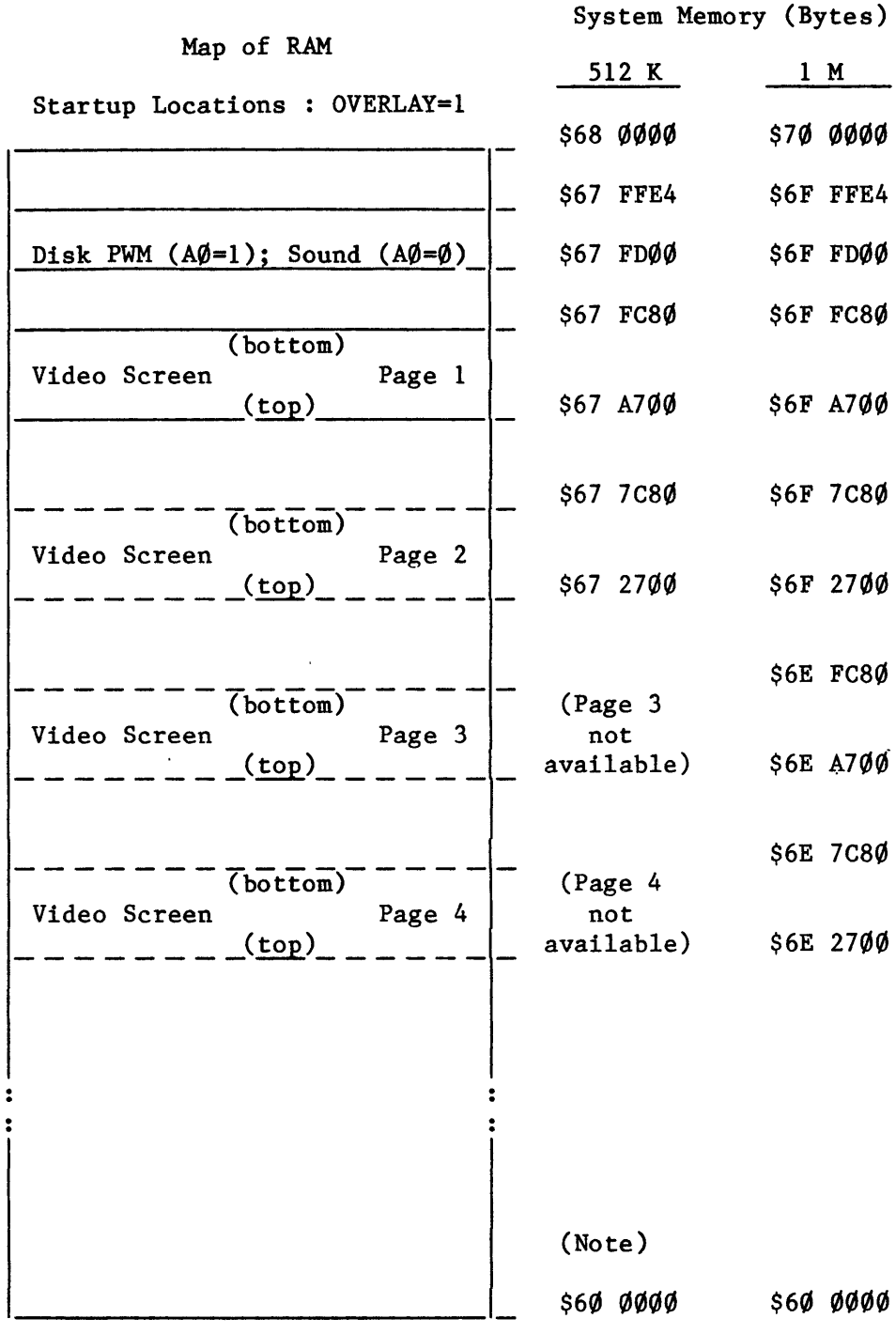
3.3 Some Useful RAM Addresses

RAM Size :	512K bytes (one row)	1 M bytes (two rows)	
Normal Addresses : (OVERLAY = Ø)	$ØØ ØØØØ - $Ø7 FFFF	$ØØ ØØØØ - $ØF FFFF	
Startup Addresses : (OVERLAY = 1)	$6Ø ØØØØ - $67 FFFF	$6Ø ØØØØ - $6F FFFF	
Video Screen : (OVERLAY = Ø)	(Top) - (Bottom)	(Top) - (Bottom)	
Page 1	$Ø7 A7ØØ - $Ø7 FC7F	$ØF A7ØØ - $ØF FC7F	(Each
Page 2	$Ø7 27ØØ - $Ø7 7C7F	$ØF 27ØØ - $ØF 7C7F	page:
Page 3	(not available)	$ØE A7ØØ - $ØE FC7F	$558Ø
Page 4	(not available)	$ØE 27ØØ - $ØE 7C7F	bytes)
Video Screen, Page 1, during startup : (OVERLAY = 1)	$67 A7ØØ - $67 FC7F	$6F A7ØØ - $6F FC7F	($558Ø bytes)
Sound Buffer : (OVERLAY = Ø)	$Ø7 FDØØ - $Ø7 FFE3	$ØF FDØØ - $ØF FFE3	($2E4 bytes)

(Note: Sound = high bytes, only)

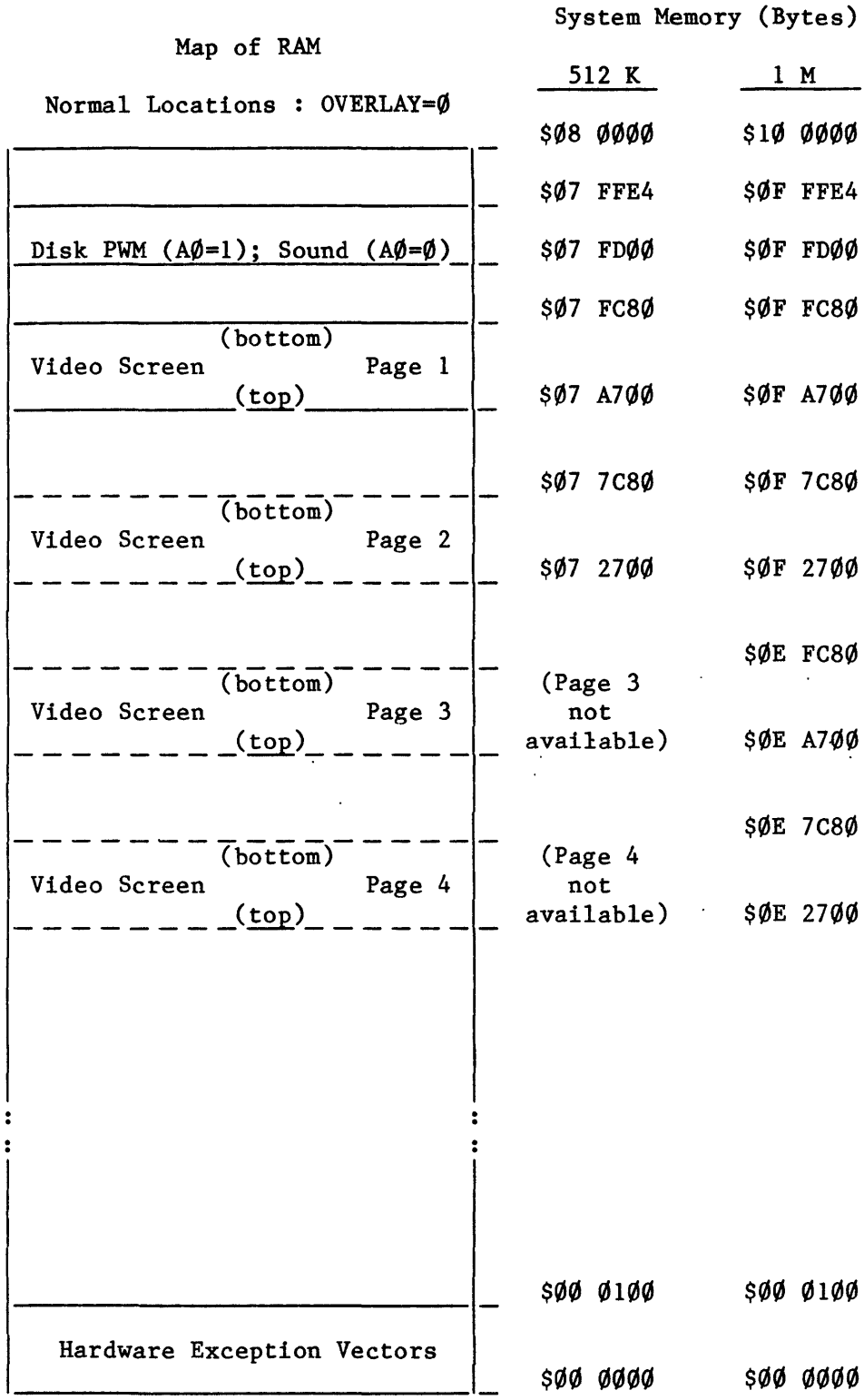
3.4 Use of RAM by Hardware

3.4.1 Map of RAM on Power-Up (OVERLAY = 1)



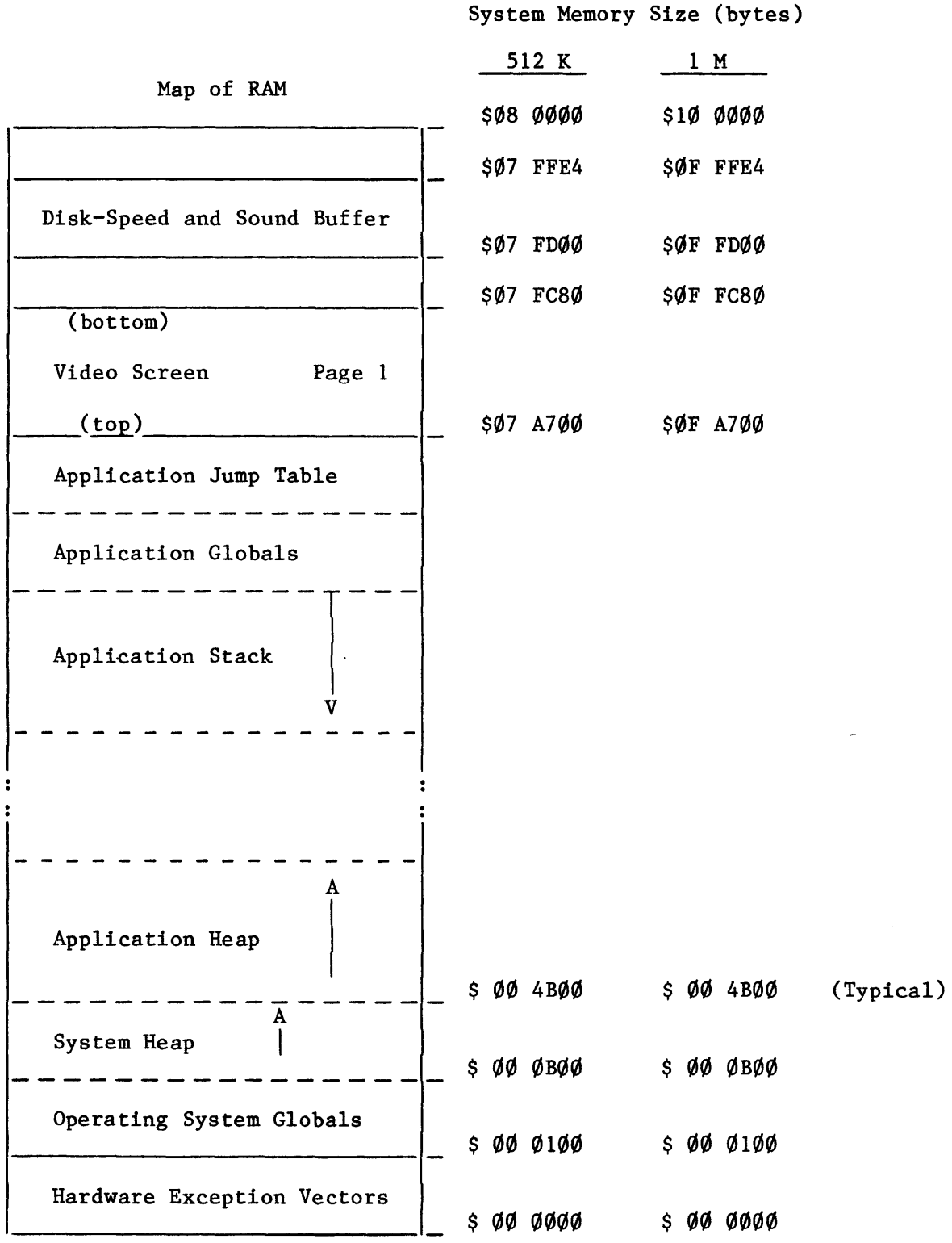
Note: the hardware exception vectors are always at $ØØØØØØ-$ØØØØFF. This places them in the ROM address space during startup (OVERLAY = 1).

3.4.2 Normal Map of RAM (OVERLAY = Ø)



3.5 Use of RAM by System and Application Software

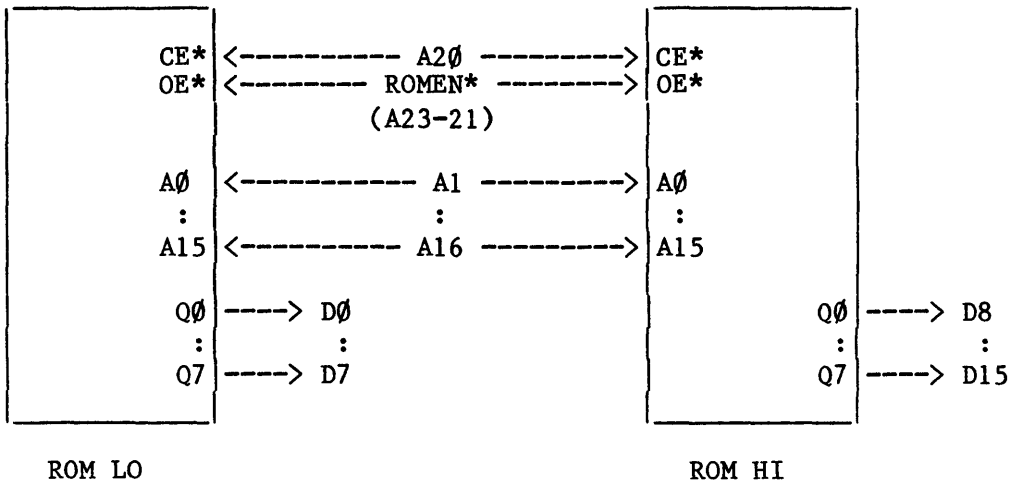
(Shown for a typical application, using page 1 video screen only.)



3.6 Hardware Exception Vectors (in RAM if OVERLAY = Ø)

Reset: Initial SSP	$ØØ ØØØØ
Reset: Initial PC	$ØØ ØØØ4
Bus Error	$ØØ ØØØ8
Address Error	$ØØ ØØØC
Illegal Instruction	$ØØ ØØ1Ø
Divide by Zero	$ØØ ØØ14
CHK Instruction	$ØØ ØØ18
TRAPV Instruction	$ØØ ØØ1C
Privilege Violation	$ØØ ØØ2Ø
Trace	$ØØ ØØ24
Line 1Ø1Ø Emulator	$ØØ ØØ28
Line 1111 Emulator	$ØØ ØØ2C
(Unassigned: Reserved)	$ØØ ØØ3Ø - $ØØ ØØ3B
Uninitialized Interrupt	$ØØ ØØ3C
(Unassigned: Reserved)	$ØØ ØØ4C - $ØØ ØØ5F
Spurious Interrupt	$ØØ ØØ6Ø
VIA Interrupt Auto-Vector	$ØØ ØØ64
SCC Interrupt Auto-Vector	$ØØ ØØ68
VIA+SCC (temp.) Auto-Vector	$ØØ ØØ6C
Interrupt Switch Auto-Vector	$ØØ ØØ7Ø
Int.Sw.+VIA Auto-Vector	$ØØ ØØ74
Int.Sw.+SCC Auto-Vector	$ØØ ØØ78
Int.Sw.+VIA+SCC Auto-Vector	$ØØ ØØ7C
TRAP Instruction Vectors	$ØØ ØØ8Ø - $ØØ ØØBF
(Unassigned: Reserved)	$ØØ ØØCØ - $ØØ ØØFF

4. ROM



4.1 Address Decoding to Activate ROMs

Note: ROM is activated whenever A2Ø=Ø and ROMEN*=Ø.

When ROM Addressed	Address Lines A23	A22	A21	A2Ø	Address Range
Startup: OVERLAY=1	Ø	Ø	Ø	Ø	$ØØ ØØØØ - $ØF FFFF
Anytime: OVERLAY=1 or Ø	Ø	1	Ø	Ø	$4Ø ØØØØ - $4F FFFF

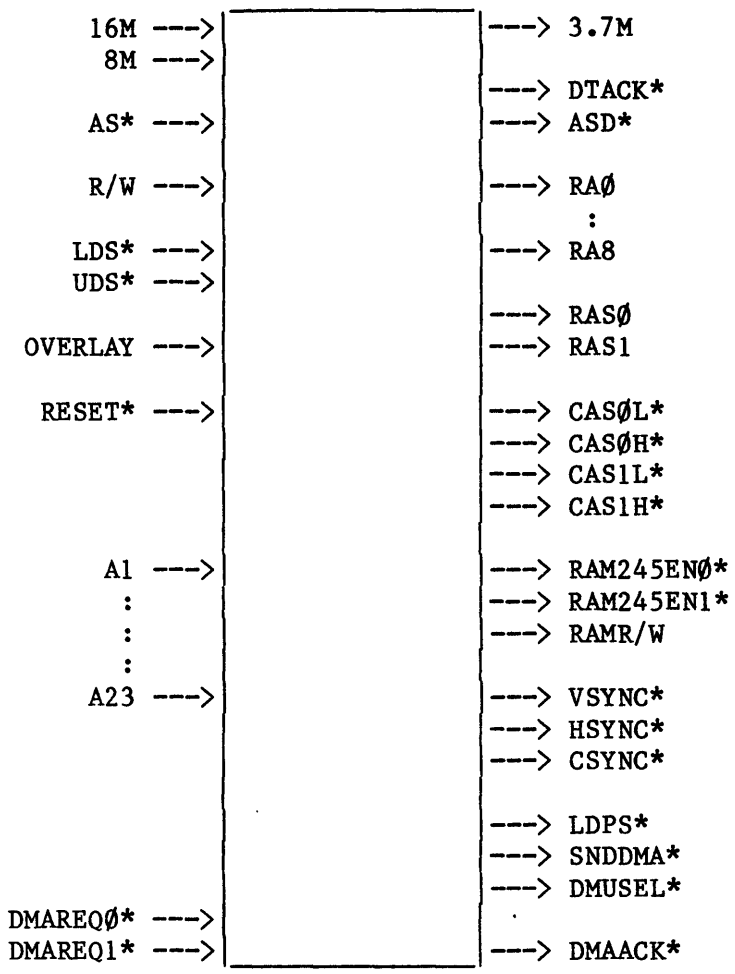
Note: Do not write to ROM. With OVERLAY=1, writing to ROM at $ØØØØØØ-$ØFFFFF will reset the AMU. With OVERLAY=1 or Ø, writing to ROM at $4ØØØØØ-$44FFFF will change the contents of the AMU registers.

4.2 Some Useful ROM Addresses

Note: these addresses are for a system with 128K bytes of ROM.

Startup ROM Addresses (OVERLAY = 1)		$ØØ ØØØØ - $Ø1 FFFF
	or	$4Ø ØØØØ - $41 FFFF (duplicate ROM image)
Normal ROM Addresses		$4Ø ØØØØ - $41 FFFF

5. AMU



AMU

5.1 Address Decoding to Activate AMU

Device Addressed	Address Lines A23	A22	A21	A2Ø	A19	A18	A17	Address Range
AMU Reset (Writes; OVERLAY=1)	Ø	Ø	Ø	Ø	Ø	Ø	Ø	$ØØ ØØØØ - $Ø1 FFFF
AMU Registers (Writes; OVERLAY=1)	Ø	1	Ø	Ø	Ø			$4Ø ØØØØ - $47 FFFF
AMU's DMU Select (OVERLAY=1)	Ø	1	Ø	1	1	1		$5C ØØØØ - $5F FFFF

Note: an attempt to read from the AMU Reset or AMU Registers memory area will not affect the AMU. Instead, ROM data will be read.

5.2 Further AMU Address Decoding

When writing to the AMU registers, the following address lines are also used:

A18-A16	Select one of the AMU registers, Ø..4
A12-A1	Data for AMU registers Ø and 2
A8-A1	Data for AMU registers 1, 3 and 4

When enabling the DMU Select line, address line A2 is also used:

A2 = Ø	Write a word of data into the DMU Control Register
A2 = 1	Execute the contents of the DMU Control Register as a command

The DMU Control Register is 16 bits wide, so word writes should be used (AØ=Ø).

5.3 Some Useful AMU Addresses

To Accomplish This Function:	Set OVERLAY=1 and Write to This Address:
Reset the AMU	$ØØ ØØØØ
Write into DMU Control Register	$5C ØØØØ
Execute Contents of DMU Control Register	$5C ØØØ4

Set DMA Start-Address of $Øv wxyz

For DMA Channel Ø :	($4Ø w'xyz)-2	and then	$41 ØØvw"
For DMA Channel 1 :	($42 w'xyz)-2	and then	$43 ØØvw"

(Note: if hex digit $w = msb bit2 bit1 lsb in binary, then $w" = msb bit2 bit1 Ø in binary and $w' = Ø Ø Ø lsb.)

Maximum DMA Address Range

For 512 K Byte System :	$ØØ ØØØØ - $Ø5 FFFF
For 1 M Byte System :	$ØØ ØØØØ - $ØD FFFF

Set the DMA Control Register

For 512 K Byte System:	Ch.Ø and 1 Read	$44 ØØ22
	Ch.Ø Write	$44 ØØ2Ø
	Ch.1 Write	$44 ØØØ2
For 1 M Byte System:	Ch.Ø and 1 Read	$44 Ø122
	Ch.Ø Write	$44 Ø12Ø
	Ch.1 Write	$44 Ø1Ø2

5.4 Information About AMU Registers

5.4.1 DMA Address Registers

Each DMA channel has an address counter which determines the source or destination address for the next 16-bit DMA transfer over that channel. The programmer sets the starting address into this counter by writing to two AMU registers: one sets bits 12...1 of the address and the other sets bits 2Ø...13. Thereafter, the address counter automatically increments its address by one 16-bit word on each DMAACK* for that channel.

When writing to the start-address registers, the register is selected by the high-order address lines, and the data for setting the register is contained in the low-order address lines. These low-order address lines are mapped into DMA start-address bits as follows:

CPU Hex Address	CPU Address Lines	Low-Order DMA Address Lines	DMA Hex Starting Address	High-Order DMA Address Lines
Digit Ø	AØ	(ignored: DMA AØ=Ø)		(ignored)
	A1	DMA A1	Digit Ø	DMA A13
	A2	DMA A2		DMA A14
	A3	DMA A3	Digit 3 (hi bits)	DMA A15
Digit 1	A4	DMA A4		DMA A16
	A5	DMA A5	Digit 1	DMA A17
	A6	DMA A6		DMA A18
	A7	DMA A7	Digit 4	DMA A19
Digit 2	A8	DMA A8		DMA A2Ø (must be Ø)
	A9	DMA A9	Digit 2	(ignored: DMA A21=Ø)
	A1Ø	DMA A1Ø		(ignored: DMA A22=Ø)
	A11	DMA A11	Digit 5	(ignored: DMA A23=Ø)
Digit 3	A12	DMA A12	Digit 3 (lsb)	
	A13	(ignored)		
	A14	(ignored)		
	A15	(ignored)		

Thus, to set a hex DMA starting address of $Øv wxyz , you must write (with OVERLAY=1) to two AMU registers, at the following addresses:

($4Ø w'xyz) - 2 and then $41 ØØvw" (DMA Channel Ø)

or

($42 w'xyz) - 2 and then $43 ØØvw" (DMA Channel 1)

where, if hex digit $w can be expressed as the four-bit binary number msb bit2 bit1 lsb , then $w" = msb bit2 bit1 Ø in binary and $w' = Ø Ø Ø lsb .

It is fairly important to write to the two DMA start-address registers in the manner shown, because the written data is clocked into the counters by the DMA clock. Writing any new value to the lower-bits register when the carry flag from the lower-bits counter was previously set (DMA A1..A12 were all 1's) will advance the upper-bits counter by 1 (at DMA A13: DMA address increases by 8K bytes). Writing to the upper-bits register will always advance the lower-bits counter by 1 (at DMA A1: lower-bits address portion increases by 2, but without carrying into the upper-bits address portion). The solution is to subtract 2 from the "correct" lower-bits address portion and write the result into the lower-bits register. Then, when you write into the upper-bits register, the lower-bits address portion will automatically be increased by 2, to become the "correct" address again.

The top 128K bytes of RAM are mapped differently from the rest of RAM memory, for the benefit of the video screens. This mapping is not convenient for DMA transfers, and the programmer should restrict DMA's to the portion of RAM memory below the top 128K bytes.

5.4.2 DMA Control Register

By writing to the AMU's DMA Control Register, the programmer can set various operating parameters.

AMU Register Number	Register Name	Description	Address Range for Writes
4	DMACTL	DMA Control Register (Values of A8-A1 set bits)	$44 ØØØØ - $44 Ø1FE

The bits in the DMA Control Register are assigned as follows:

Address Line	DMA Ctrl. Reg. Bit	AMU Signal Name	(Comment)
8	7	TWOROWS	(1 = two rows of DRAMs installed)
7	6		(Reserved)
6	5		(Reserved)
5	4	DMA1R/W	(1 = DMA Channel 1 set for reading)
4	3		(Reserved)
3	2		(Reserved)
2	1		(Reserved)
1	Ø	DMAØR/W	(1 = DMA Channel Ø set for reading)

Note: for a DMA channel, "reading" means a DMA transfer from a peripheral into the Mac; writing means a DMA transfer from the Mac to a peripheral.

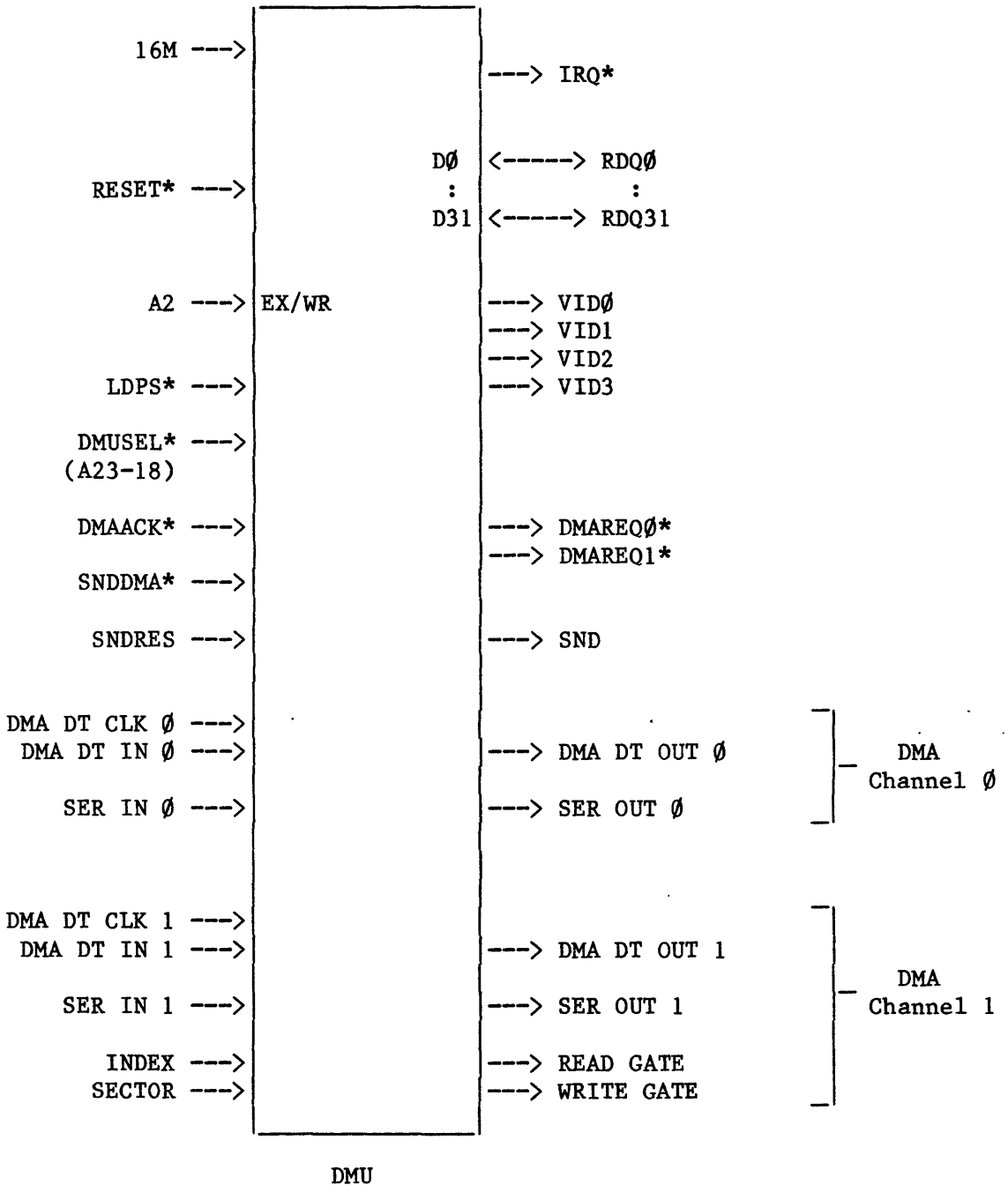
The correct hexadecimal address for writing to the DMA Control Register may be determined as follows:

$44 Øxyz

where the hexadecimal digits x, y, and z are determined as follows:

x: Memory Size	y: DMA Ch.1 R/W	z: DMA Ch.Ø R/W
Ø = 512K bytes	Ø = Write	Ø = Write
1 = 1 M bytes	2 = Read	2 = Read

6. DMU


16M --->
RESET* --->
A2 --->
LDPS* --->
DMUSEL* --->
(A23-18)
DMAACK* --->
SNDDMA* --->
SNDRES --->
DMA DT CLK Ø --->
DMA DT IN Ø --->
SER IN Ø --->
DMA DT CLK 1 --->
DMA DT IN 1 --->
SER IN 1 --->
INDEX --->
SECTOR --->
EX/WR
DØ
:
D31
<-----> RDQØ
:
<-----> RDQ31
---> IRQ*
---> VIDØ
---> VID1
---> VID2
---> VID3
---> DMAREQØ*
---> DMAREQ1*
---> SND
---> DMA DT OUT Ø
---> SER OUT Ø
---> DMA DT OUT 1
---> SER OUT 1
---> READ GATE
---> WRITE GATE
DMA Channel Ø
DMA Channel 1
DMU

6.1 Address Decoding to Activate DMU

Device Addressed	Address Lines A23	A22	A21	A2Ø	A19	A18	Address Range
DMU Select (OVERLAY=1)	Ø	1	Ø	1	1	1	$5C ØØØØ - $5C FFFF

6.2 Further DMU Address Decoding

When interacting with the DMU control register, address line A2 is also used:

A2 = Ø The DMU control register is selected for writing.

A2 = 1 The contents of the DMU control register are to be executed as a command.

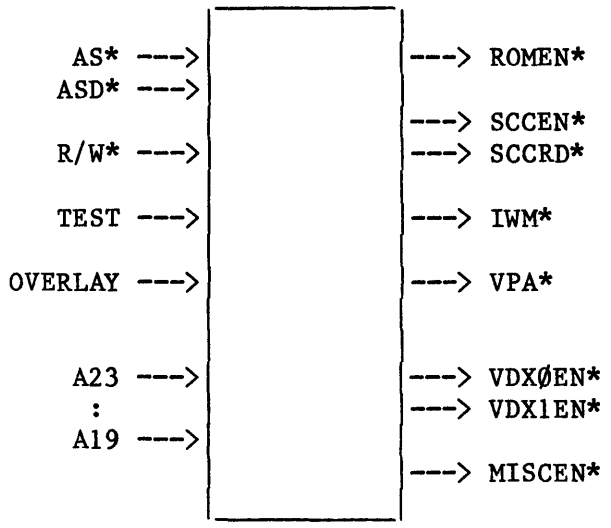
The DMU control register is 16 bits wide, so word writes should be used (AØ=Ø)

6.3 Some Useful DMU Addresses

Write into DMU Control Register	$5C ØØØØ
Execute the Contents of the DMU Control Register	$5C ØØØ4

7. RANDOM LOGIC CONTROLS (VDXØEN*, VDX1EN* and MISCEN*)

These signals are generated by the Off-Chip Decode PAL (OCD).



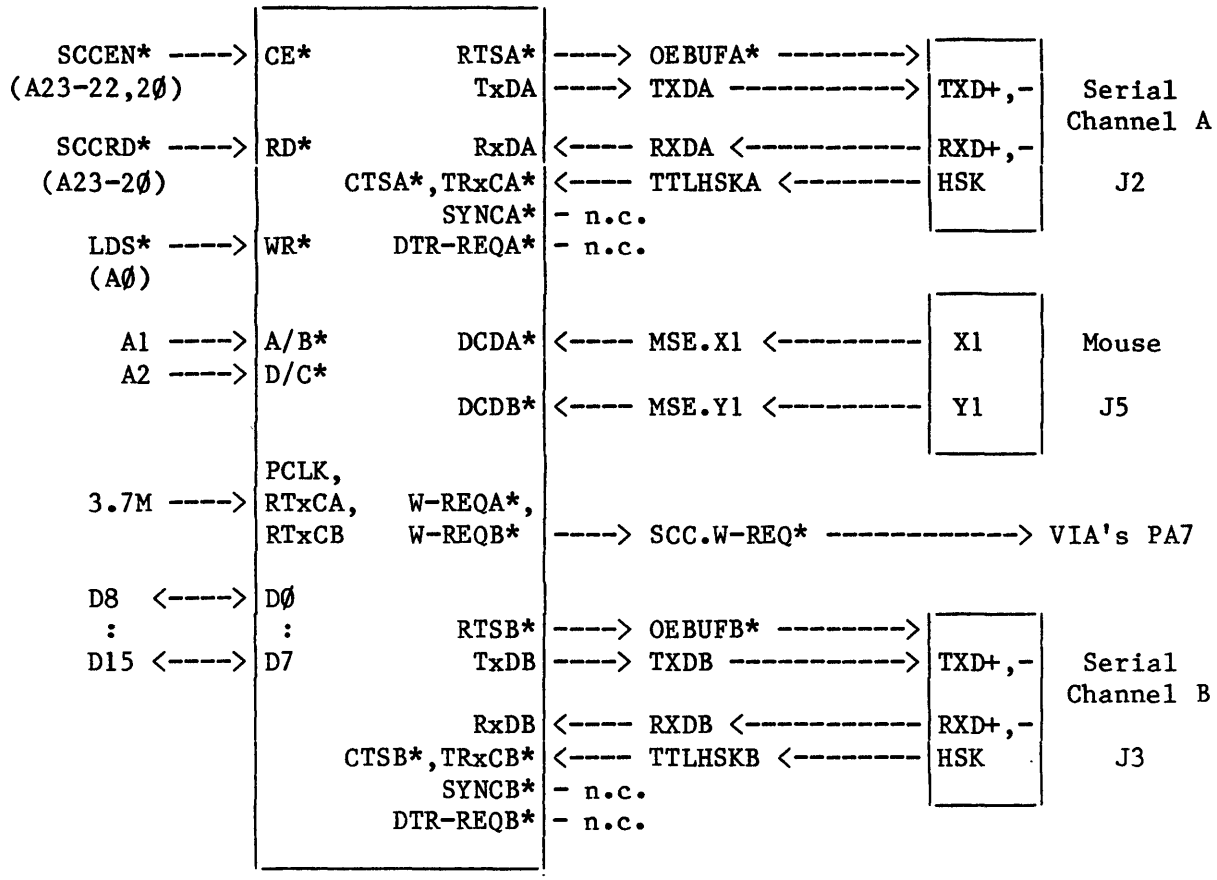
7.1 Address Decoding to Activate Random Logic Controls

Device Addressed	Address Lines A23	A22	A21	A2Ø	A19	Address Range	(Comments)
VDXØ Enable (VDXØEN*=Ø)	1	1	Ø	Ø	Ø	$CØ ØØØØ - $CF FFFF,	(Enables both VDXØ & VDX1)
	1	Ø	1	Ø	Ø	$AØ ØØØØ - $A7 FFFF	
VDX1 Enable (VDX1EN*=Ø)	1	1	Ø	Ø	Ø	$CØ ØØØØ - $C7 FFFF,	(Enables both VDXØ & VDX1)
	1	Ø	1	Ø	1	$A8 ØØØØ - $AF FFFF	
MISC Enable (MISCEN*=Ø)	1	1	Ø	Ø	1	$C8 ØØØØ - $CF FFFF	

7.2 Some Useful Random Logic Control Addresses

Enable VDXØ	$AØ ØØØØ	Video Expansion
Enable VDX1	$A8 ØØØØ	
Enable Both VDXØ and VDX1	$CØ ØØØØ	
Enable MISC	$C8 ØØØØ	Miscellaneous Expansion

8. SCC



SCC (853Ø)

8.1 Address Decoding to Activate SCC

Device Addressed	Address Lines A23	A22	A21	A2Ø	Address Range
SCC Read (SCCEN*=Ø, SCCRD*=Ø)	1	Ø	Ø	1	$9Ø ØØØØ - $9F FFFF
SCC Write (SCCEN*=Ø, SCCRD*=1)	1	Ø	1	1	$BØ ØØØØ - $BF FFFF

8.2 Further SCC Address Decoding

	A2	A1	AØ
LDS* = Ø	x	x	1
LDS* = 1	x	x	Ø
Channel A	x	1	x
Channel B	x	Ø	x
Data Register	1	x	x
Control Register	Ø	x	x

(Note: x indicates "don't care": either 1 or Ø)

READ
Byte Read:
AØ = Ø

The SCC uses the upper byte of the data bus, so use even (AØ=Ø) SCC READ addresses when reading the SCC. This sets LDS* high and the CPU reads data from D8-D15.

RESET
Byte Read:
AØ = 1

A byte access to any odd (AØ=1) SCC READ address sets LDS* and SCCRD* both low. This resets the SCC.

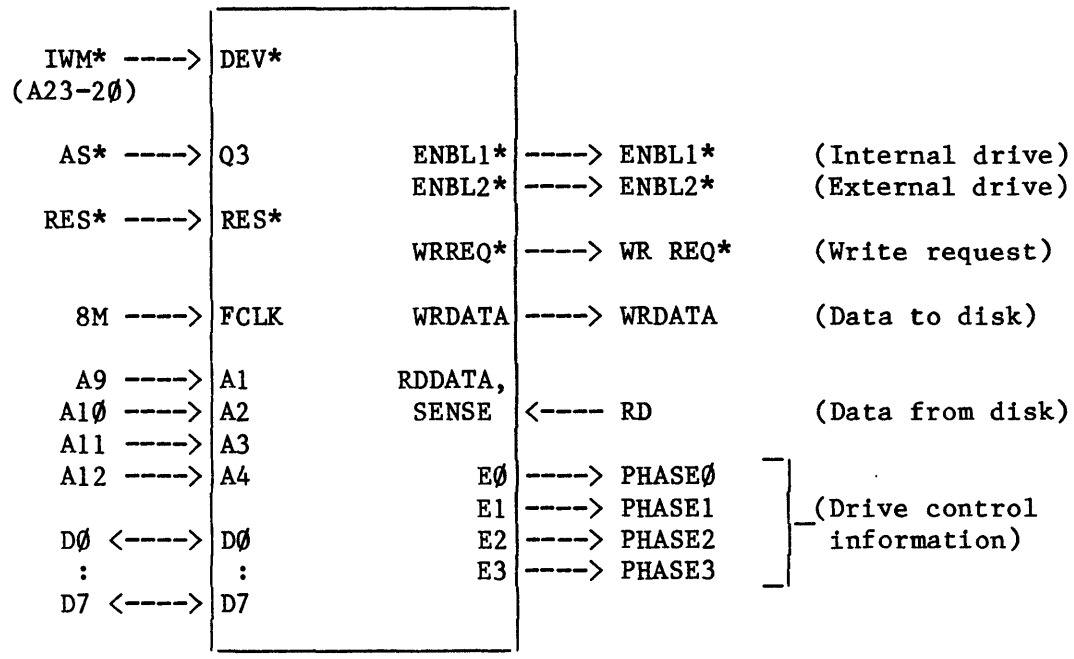
WRITE
Byte Write:
AØ = 1

When writing to the SCC, you must use odd SCC WRITE addresses (AØ=1), even though the SCC is on the upper byte of the data bus. This sets LDS* low, and the CPU writes the same byte of data to DØ-D7 and D8-D15, using a special feature of the 68ØØØ CPU: a write to the lower byte of the data bus also places the same data on the upper byte of the data bus.

8.3 Some Useful SCC Addresses

Channel A: Write to data register	$BF FFFF
Read from data register	$9F FFFE
Channel B: Write to data register	$BF FFFD
Read from data register	$9F FFFC
Channel A: Write to control register specified in Write Register Ø	$BF FFFB
Channel A: Read from control register specified in Write Register Ø	$9F FFFA
Channel B: Write to control register specified in Write Register Ø	$BF FFF9
Channel B: Read from control register specified in Write Register Ø	$9F FFF8
Reset SCC	$9F FFFF

9. IWM



IWM

9.1 Address Decoding to Activate IWM

Device Addressed	Address Lines A23	A22	A21	A2Ø	Address Range
IWM (IWM*=Ø)	1	1	Ø	1	$DØ ØØØØ - $DF FFFF

9.2 Further IWM Address Decoding

A12, A11, A1Ø — These three lines select one of the eight bits in the IWM State Register.

A9 ---- When DEV* (IWM*) goes low, the IWM State Register bit selected by A12-A1Ø is set to the "value" (1 or Ø) on line A9.

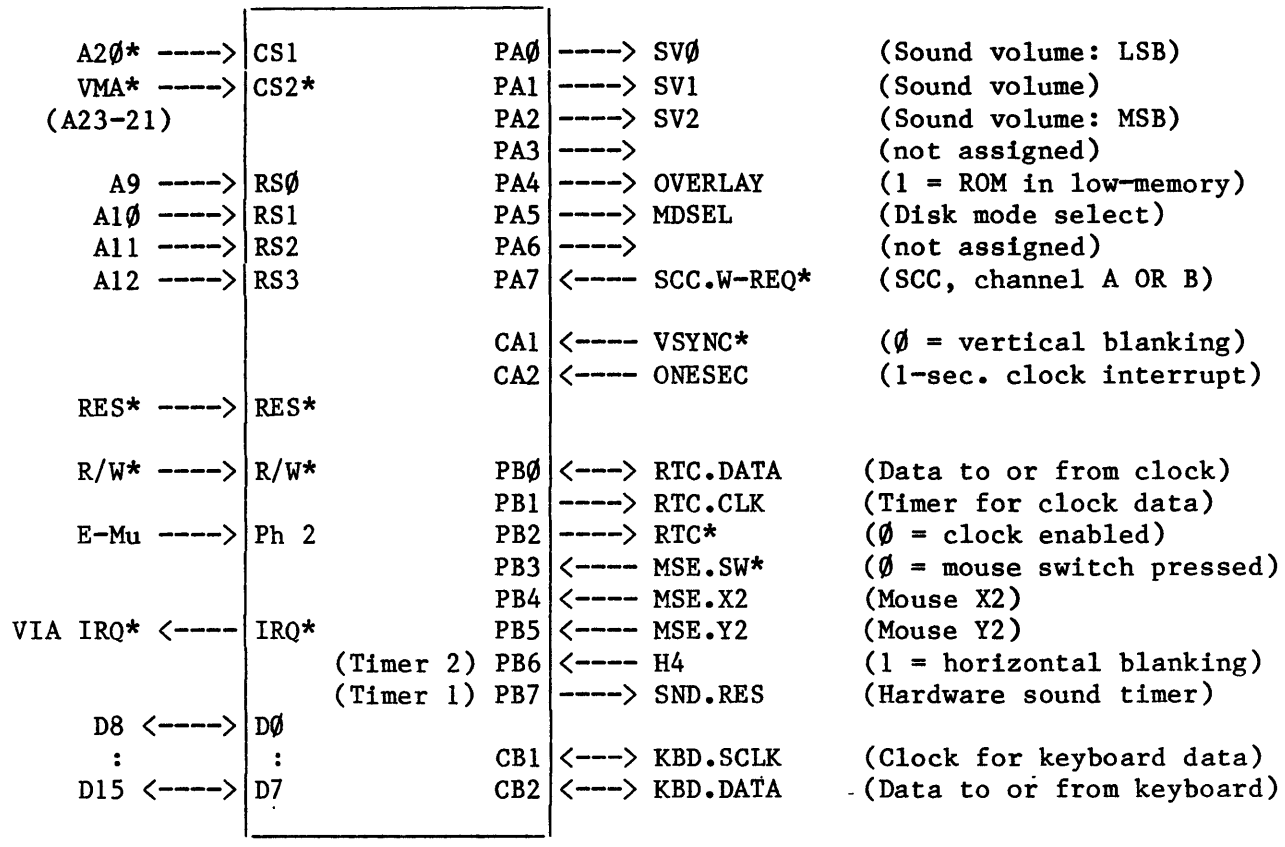
AØ = 1 The IWM uses the lower byte of the data bus, so use AØ = 1 .

9.3 Some Useful IWM Addresses

These are addresses which use A12-A9 to set individual bits in the IWM State Register.

State Register Bit	IWM Function		Turbo Mac Address	
Ø	Phase Ø:	Low	$DF E1FF	(Disk drive control information)
		High	$DF E3FF	
1	Phase 1:	Low	$DF E5FF	
		High	$DF E7FF	
2	Phase 2:	Low	$DF E9FF	
		High	$DF EBFF	
3	Phase 3:	Low	$DF EDFF	
		High	$DF EFFF	
4	Motor:	Off	$DF F1FF	(Disables all drives)
		On	$DF F3FF	(Enables selected drive)
5	Select:	Drive 1	$DF F5FF	(Selects internal drive)
		Drive 2	$DF F7FF	(Selects external drive)
6	Disk Q6:	Low	$DF F9FF	(IWM mode selects; called L6 and L7 in IWM document)
		High	$DF FBFF	
7	Disk Q7:	Low	$DF FDFF	
		High	$DF FFFF	

1Ø. VIA

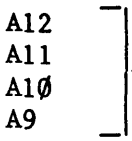


VIA (6522)

1Ø.1 Address Decoding to Activate VIA

Device Addressed	Address Lines A23	A22	A21	A2Ø	Address Range
VIA (VMA*=Ø, A2Ø*=1)	1	1	1	Ø	$EØ ØØØØ - $EF FFFF

1Ø.2 Further VIA Address Decoding



AØ = Ø The VIA uses the upper byte of the data bus, so use AØ = Ø .

1Ø.3 Some Useful VIA Addresses

These are addresses which use A12-A9 to select individual VIA registers.

Input or Output Register A	$EF FFFE	(Do NOT use I-O Register A with Handshake: $EFE3FE)
Input or Output Register B	$EF E1FE	
Data Direction Register A	$EF E7FE	(Ø-bits indicate inputs,
Data Direction Register B	$EF E5FE	while 1's are outputs)
Timer 1 Counter: Low Byte	$EF E9FE	(Associated with PB7)
High Byte	$EF EBFE	
Timer 1 Latch: Low Byte	$EF E9FE	
High Byte	$EF EBFE	
Timer 2 Counter: Low Byte	$EF F1FE	(Down-counter; may be
High Byte	$EF F3FE	associated with PB6)
Shift Register	$EF F5FE	(Shifts data into or out of VIA on CB2, clocked by Ph 2, Timer 2, or CB1)
Auxiliary Control Register	$EF F7FE	
Peripheral Control Register	$EF F9FE	
Interrupt Flag Register	$EF FBFE	
Interrupt Enable Register	$EF FDFE	

10.4 Turbo-Mac-Specific Information about VIA Registers

10.4.1 Port A Input, Output, and Data Direction Registers

Port A I-O Reg. Bit	VIA Line Name	Port A Data Direction	Computer Signal Name	(Comments)
7	PA7	<---- Input <----	SCC.W-REQ*	(SCC, channel A OR B)
6	PA6	----> Output ---->		(not assigned)
5	PA5	----> Output ---->	MDSEL	(Disk mode select)
4	PA4	----> Output ---->	OVERLAY	(1 = ROM in low-memory)
3	PA3	----> Output ---->		(not assigned)
2	PA2	----> Output ---->	SV2	(Sound volume: MSB)
1	PA1	----> Output ---->	SV1	(Sound volume)
0	PA0	----> Output ---->	SV0	(Sound volume: LSB)

Port A Data Direction Byte: $7F

10.4.2 Port B Input, Output, and Data Direction Registers

Port B I-O Reg. Bit	VIA Line Name	Port B Data Direction	Computer Signal Name	(Comments)
7 (Tmr1)	PB7	----> Output ------>	SND.RES	(Hardware sound timer)
6 (Tmr2)	PB6	<---- Input <------	H4	(1 = horizontal blanking)
5	PB5	<---- Input <------	MSE.Y2	(Mouse Y2)
4	PB4	<---- Input <------	MSE.X2	(Mouse X2)
3	PB3	<---- Input <------	MSE.SW*	(0 = mouse switch pressed)
2	PB2	----> Output ------>	RTC*	(Enables real-time clock)
1	PB1	----> Output ------>	RTC.CLK	(Timer for clock data)
0	PB0	<---> In or Out <-->	RTC.DATA	(Data to or from clock)

Port B Data Direction Byte,
when data is coming in from clock: $86
when data is going out to clock: $87

10.4.3 Control Registers

Peripheral Control Register Bit	VIA Line Controlled		Computer Signal or Interrupt Controlled	(Comments)
7				
6	---- CB2	<----->	KBD.DATA	(Data to or from keyboard)
5				
4	----- CB1	<----->	KBD.SCLK	(Clock for keyboard data)
3				
2	---- CA2	<-----	ONESEC	(1-second clock interrupt)
1				
0	----- CA1	<-----	VSYNC*	(0 = video vertical blanking)

10.4.4 Interrupt Flag and Enable Registers

Interrupt Flag Reg. Bit	VIA Function Flagged	Computer Signal Flagged	(Comments)
7	IRQ* (any enabled VIA interrupts)	Also sets VIA IRQ*	(CPU interrupt IPL0*)
6	Timer 1 (PB7)	SND.RES	(Hardware sound timer)
5	Timer 2		
4	CB1	KBD.SCLK	(Clock for keyboard data)
3	CB2	KBD.DATA	(Data to or from keyboard)
2	Shift Register	Eight bits of KBD.DATA Shifted	
1	CA1	VSYNC*	(Video vertical blanking)
0	CA2	ONESEC	(1-second clock interrupt)

The Interrupt Enable Register is arranged just like the Interrupt Flag Register except that bit 7 is "Set/Clear":

Bit 7 Value	Meaning of Values In Bits 6 Through 0
1	Each 1 enables the corresponding interrupt
0	Each 0 enables the corresponding interrupt

11. AUTO-VECTOR "READ" ADDRESSES

When servicing an interrupt, the CPU "reads" an address in the range $FFFFFØ - $FFFFFF . The exact address consists of ones on all address lines, except that address lines A3-A1 are determined by the three interrupt lines IPL2*-Ø*:

Interrupting Device	Interrupt Line	Address Line A3	A2	A1	Address "Read"
VIA	IPLØ*	Ø	Ø	1	$FF FFF3
SCC	IPL1*	Ø	1	Ø	$FF FFF5
VIA + SCC (Transient: Retry)	IPL1*+Ø*	Ø	1	1	$FF FFF7
Interrupt Switch	IPL2*	1	Ø	Ø	$FF FFF9
Int. + VIA	IPL2*+Ø*	1	Ø	1	$FF FFFB
Int. + SCC	IPL2*+1*	1	1	Ø	$FF FFFD
Int. + SCC + VIA	IPL2*+1*+Ø*	1	1	1	$FF FFFF

No device is activated, and any data "read" is ignored. The only response of the system is that device OCD sets the signal VPA* low. This in turn causes the CPU to set VMA* low and to jump through the appropriate auto-vector location in low memory.

When any address in the range $EØØØØØ-$FFFFFF is accessed, the OCD sets VPA* low, and the CPU responds by setting VMA* low. However, the CPU does not do an auto-vector jump unless the address was "read" by the CPU in servicing an interrupt.

12. SOME USEFUL DECODING EQUATIONS

Note: while some device functions are selected by address lines directly, others are selected by Turbo Mac signals which are internally decoded as follows:

RAMEN* = Ø when OVERLAY=1 and A23=Ø and A22=1 and A21=1
or when OVERLAY=Ø and A23=Ø and A22=Ø

RAMR/W* = Ø when ASEL1=Ø and ASELØ=Ø and DMAØR/W*=Ø (DMA Ch.Ø write)
or when ASEL1=Ø and ASELØ=1 and DMA1R/W*=Ø (DMA Ch.1 write)
or when ASEL1=1 and ASELØ=Ø and R/W*=Ø (CPU write)

	Bus Multiplexor Signals (in AMU)	
Device on Bus	ASEL1	ASELØ
DMA Ch. Ø	Ø	Ø
DMA Ch. 1	Ø	1
CPU	1	Ø
Video	1	1

RAM245ENØ* = Ø when RASØ*=Ø and VID/MU*=Ø (and stays low for CASØL,H*=Ø)
or when DMUSEL*=Ø

RAM245EN1* = Ø when RAS1*=Ø and VID/MU*=Ø (and stays low for CAS1L,H*=Ø)
or when DMUSEL*=Ø

```
ROMEN* = Ø       when  OVERLAY=1 and AS*=Ø and R/W*=1
                       and A23=Ø and A22=Ø and A21=Ø and A2Ø=Ø
                 or when  AS*=Ø and R/W*=1 and A23=Ø and A22=1 and A21=Ø and A2Ø=Ø


VDXØEN* = Ø      when  AS*=Ø and A23=1 and A22=Ø and A21=1 and A2Ø=Ø and A19=Ø
                 or when  AS*=Ø and A23=1 and A22=1 and A21=Ø and A2Ø=Ø and A19=Ø

VDX1EN* = Ø      when  AS*=Ø and A23=1 and A22=Ø and A21=1 and A2Ø=Ø and A19=1
                 or when  AS*=Ø and A23=1 and A22=1 and A21=Ø and A2Ø=Ø and A19=Ø


MISCEN* = Ø      when  AS*=Ø and A23=1 and A22=1 and A21=Ø and A2Ø=Ø and A19=1


DMUSEL* = Ø      when  MUSS=Ø and AS*=Ø and A23=Ø and A22=1 and A21=Ø
                       and A2Ø=1 and A19=1 and A18=1


SCC EN* = Ø      when  ASD*=Ø and A23=1 and A22=Ø and A2Ø=1

SCC RD* = Ø      when  AS*=Ø and A23=1 and A22=Ø and A21=Ø and A2Ø=1


IWM* = Ø         when  AS*=Ø and A23=1 and A22=1 and A21=Ø and A2Ø=1


VPA* = Ø         when  AS*=Ø and A23=1 and A22=1 and A21=1

VMA* = Ø         when  VPA*=Ø

A2Ø* = Ø         when  A2Ø=1
```