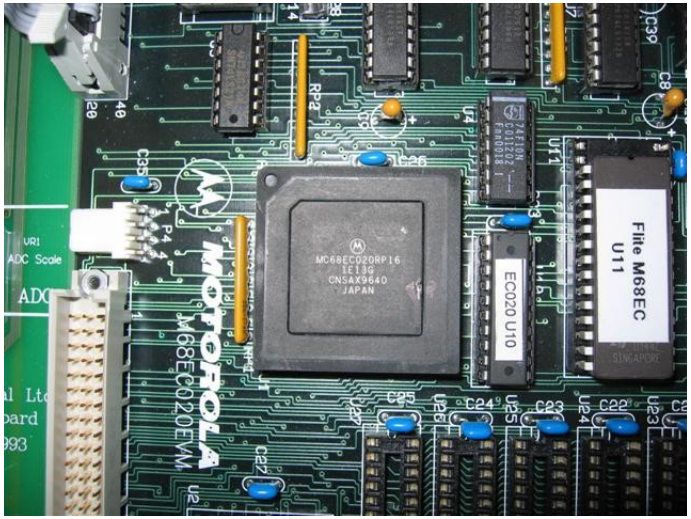
SYSC3601
Microprocessor Systems



Unit 9:

The Motorola 68000 μP

Topics/Reading

1. Overview of the 68000 μP

2. Programming model, assembly, addressing modes, stack

3. 68000 Hardware interfacing, bus arbitration

4. Read/Write cycles

5. Memory organization

6. Memory interfacing

7. I/O interfacing

8. Exception processing, hardware interrupts

Reading: Antonakos, chapters 1,2,3,4,7,8,9,12

Motorola 68000 μP

- 6800 μP - introduced in 1974, 8-bit.
- 68000 μP - introduced in Sept 1979.
  - NMOS (N-channel MOS technology)
  - 68K transistors?
  - 64 Pin DIP (8086 is 40) → No multiplexing.
  - Internal architecture is 32 bits (ALU is 16 bits wide).
  - 23 bit (physical) address bus A_1 - A_{23}. No A_0.
    - (24 bit effective address bus with LDS/UDS…)
  - 16 bit data bus.
  - Operands:
    - Bytes
    - Words (16-bits)
    - Long words (32-bits)

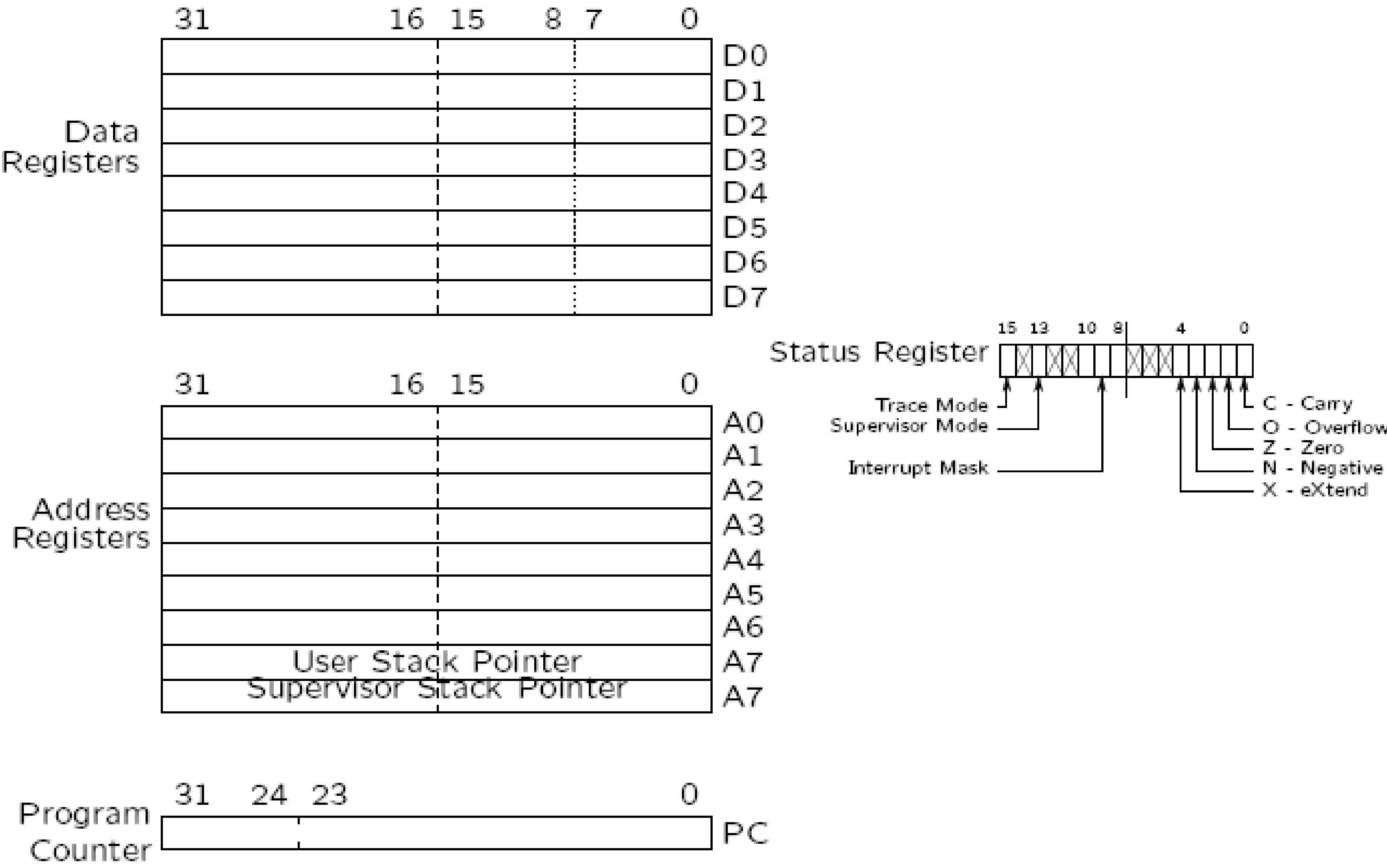
Motorola 68000 μP

- **68008** - 8 bit data bus, 20 bit address bus.
- **68010** - 1982. Added virtual memory support.
- **68020** - 1984. Fully 32 bit. 3 stage pipeline.
  - 256 byte cache. More addressing modes!
- **68030** - 1987. Integrated MMU into chip.
- **68040** - 1991. Harvard architecture with two 4KB caches. FP on chip. 6 stage pipeline.
- **68060** - 1994. Superscalar version . 10-stage pipeline. 2 integer, 1 fp unit. 8k caches. 4-5W.
- **Coldfire** - 1995. Embedded version. Stripped out funky addressing modes.

Motorola 68000 μP

- Used in:
  - Apple Macintosh (then PPC, now Intel!!).
  - Atari 520ST and 1040ST (Defunct).
  - Amiga (Defunct).
  - Early Sun workstations (Now SPARC).
  - NeXT (Defunct, purchased by Apple 1996, MAC OS X came from ‘NeXTStep’).
- 68000 architecture has user/supervisor modes.
  - protects operating system.
  - supports multitasking and multiprocessing.

Motorola 68000 µP – Programming Model



Data Registers

31 16 15 8 7 0

D0
D1
D2
D3
D4
D5
D6
D7

Address Registers

31 16 15 0

A0
A1
A2
A3
A4
A5
A6
A7 User Stack Pointer
A7 Supervisor Stack Pointer

Program Counter

31 24 23 0

PC

Status Register

15 13 10 8 4 0

Trace Mode
Supervisor Mode
Interrupt Mask

C - Carry
O - Overflow
Z - Zero
N - Negative
X - eXtend

Motorola 68000 μP - Assembler

- format: `INST SRC,DST`
- Prexes:
  - `%` binary
  - `$` Hexadecimal
  - `#` Immediate.
- Attach size to instruction, e.g.:
  - `move.b` byte
  - `move.w` word
  - `move.l` long word
- `` `() ``' refers to indirect addressing
  - (recall that Intel uses `` `[] ``').

Motorola 68000 μP – Assembler Examples

- `move.b #$F5,d1`
  - Store immediate (#) hex ($) byte (`.b`), `$F5` into destination `d1`. 8-bit transfer.
- `move.w (a2),d1`
  - stores contents of memory addressed by `a2` into data register `d1`. 16-bit transfer.
- `add.l d4,d5`
  - Add 32-bit contents of register `d4` to `d5` and store the results in `d5`. Set flags.

Motorola 68000 µP – Addressing Modes

- There are 14 different addressing modes (more with the 68020!)

Mode	Syntax
Data reg direct	d_n, n = 0..7
Addr reg direct	a_n, n = 0..7
Addr reg indirect	(a_n)
with Postincrement	$(a_n)+$
with Predecrement	$-(a_n)$
with Displacement	$d_{16}(a_n)$
with Index	$d_8(a_n,X_m)$ (X_m is any a_m or d_m)
Relative with offset	$d_{16}(PC)$
Relative with index and offset	$d_8(PC,X_n)$
Absolute short	< … > (16-bits sign-extended to 32) (for 000000-007FFF or FF8000-FFFFFF)
Absolute long	< … > (32-bits)
Immediate	#< … >
Quick immediate	#< … > (1 byte, sign-extend to 32)
Implied	Register specified as part of mnemonic

Motorola 68000 µP – Addressing Mode Examples

- `move.b #$6f,d0`
  - *Immediate*
- `move.w d3,d4`
  - *data reg direct*. Lower 16 bits of `d3` are copied to low 16-bits of `d4`, upper `d4` not changed
- `movea.l a5,a2`
  - *address reg direct*. Size **must be** `.w` or `.l`; `.w` implies sign extension
- `move.b (a0),d7`
  - *address register indirect*. Byte pointed at by `a0` copied to `d7`
- `move.w (a5)+,d2`
  - *post-increment*. Word pointed at by `a5` copied to `d2`, `a5` then incremented by two (`.w`)
- `move.b -(a2),d4`
  - *pre-decrement*. `a2` decremented by one, then byte pointed at by `a2` copied to `d4`

Motorola 68000 µP – Addressing Mode Examples

- `move.w $100(a0),d0`
  - *addr reg indirect with displacement*. Contents of memory at `a0+100`$_{16}$ copied to `d0`
- `move.b $08(a0,d1.w),d0`
  - *addr reg indir with index*. Note: can specify size here! Uses b$_{15}$-b$_{0}$ of `d1` only (addr=a0+d1.w+$08)
- `move.b $9AE0,d1`
  - *absolute short*. Sign extend to get data from address `$FF9AE0`.
- `move.b $2E0000,d4`
  - *Absolute long*
- `moveq #$2C,d3`
  - *Quick Immediate*. Byte only (data encoded within instruction word). Byte is sign-extended to 32 bits.

Motorola 68000 µP – Addressing Mode Examples

- Example: A sample assembler subroutine for the 68000:

Total: Find the sum of 16-bytes stored in memory.

```
        org     $8000          ;load program counter
total   clr.w    d0            ;clear D0.
        move.b   #16,d1        ;initialize counter
        movea.l #data,a0       ;init pointer to data
loop    add.b   (a0)+,d0       ;add byte, increment address
        subq.b   #1,d1         ;decrement counter
        bne     loop           ;test for zero, branch not equal.
        movea.l #sum,a1        ;load address to store result
        move.w   d0,(a1)       ;store sum at sum
        rts                    ;return from subroutine.

sum     dc.w     0             ;save room for result.
data    ds.b    16             ;save room for 16 data bytes.
        end
```

- Note:
  - `dc.w` - define a constant word, operand specifies the value to be written.
  - `ds.b` - define storage byte, operand specifies number of bytes, but not the contents

Motorola 68000 µP – Stack

- Address register `a7` is used to point to the stack.
- There are no push or pop instructions
  - – (except for `pea` - push effective address).
- A push is done with:

```
move.l d3,-(a7)
```

1. Decrement `a7` by four,
2. Write 32 bits to stack

- Actually implemented as:

1. Decrement `a7` by two.
2. Write low word of d3
3. Decrement `a7` by two.
4. Write high word of d3

Motorola 68000 μP – Stack

- A pop is done with

```
move.l (a7)+,d3
```

- Stack grows down (towards lower addresses)

- `pea` - Push Effective Address.

```
pea $40(a5)
```

  - Effective address is sum of `a5` and `$40`.
  - Result is pushed.

- `jsr, rts` - Subroutine calls
  - Push/pop program counter and branch.

Motorola 68000 μP – Instruction Set

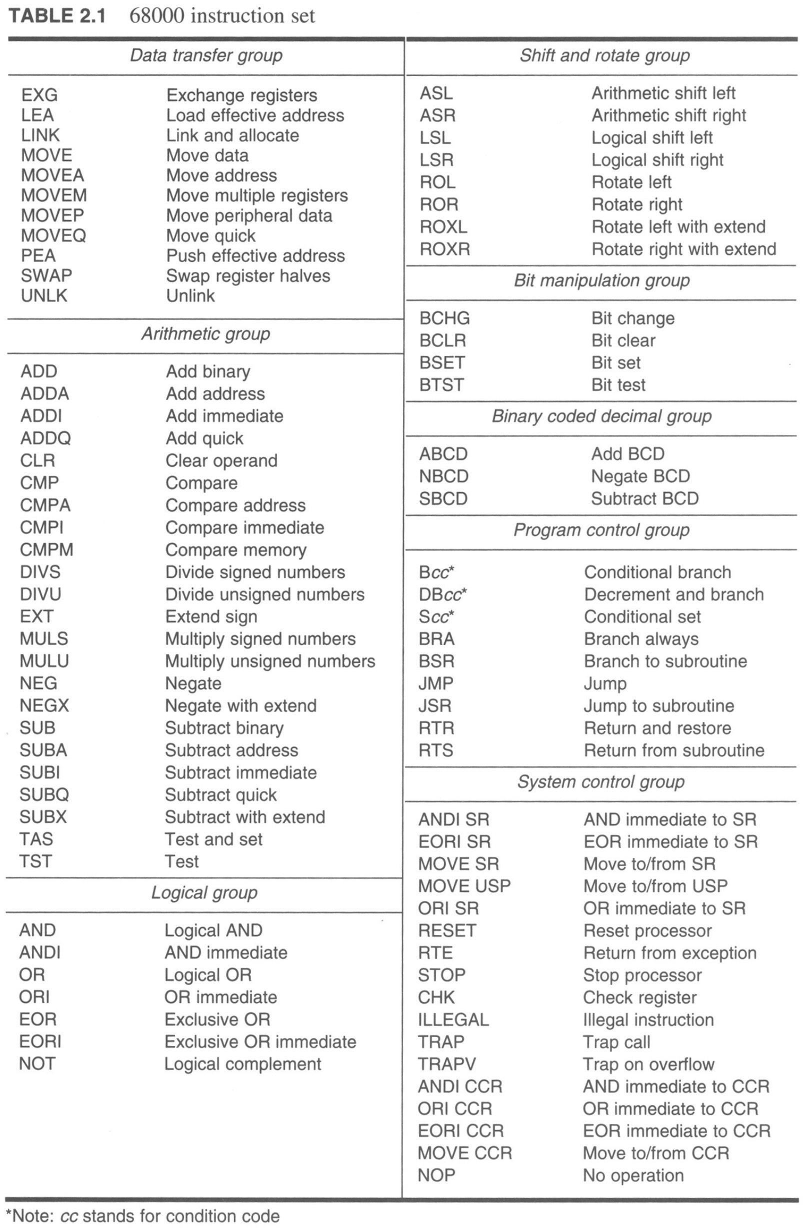


TABLE 2.1 68000 instruction set

Data transfer group	
EXG	Exchange registers
LEA	Load effective address
LINK	Link and allocate
MOVE	Move data
MOVEA	Move address
MOVEM	Move multiple registers
MOVEP	Move peripheral data
MOVEQ	Move quick
PEA	Push effective address
SWAP	Swap register halves
UNLK	Unlink

Arithmetic group	
ADD	Add binary
ADDA	Add address
ADDI	Add immediate
ADDQ	Add quick
CLR	Clear operand
CMP	Compare
CMPA	Compare address
CMPI	Compare immediate
CMPM	Compare memory
DIVS	Divide signed numbers
DIVU	Divide unsigned numbers
EXT	Extend sign
MULS	Multiply signed numbers
MULU	Multiply unsigned numbers
NEG	Negate
NEGX	Negate with extend
SUB	Subtract binary
SUBA	Subtract address
SUBI	Subtract immediate
SUBQ	Subtract quick
SUBX	Subtract with extend
TAS	Test and set
TST	Test

Logical group	
AND	Logical AND
ANDI	AND immediate
OR	Logical OR
ORI	OR immediate
EOR	Exclusive OR
EORI	Exclusive OR immediate
NOT	Logical complement

Shift and rotate group	
ASL	Arithmetic shift left
ASR	Arithmetic shift right
LSL	Logical shift left
LSR	Logical shift right
ROL	Rotate left
ROR	Rotate right
ROXL	Rotate left with extend
ROXR	Rotate right with extend

Bit manipulation group	
BCHG	Bit change
BCLR	Bit clear
BSET	Bit set
BTST	Bit test

Binary coded decimal group	
ABCD	Add BCD
NBCD	Negate BCD
SBCD	Subtract BCD

Program control group	
Bcc*	Conditional branch
DBcc*	Decrement and branch
Scc*	Conditional set
BRA	Branch always
BSR	Branch to subroutine
JMP	Jump
JSR	Jump to subroutine
RTR	Return and restore
RTS	Return from subroutine

System control group	
ANDI SR	AND immediate to SR
EORI SR	EOR immediate to SR
MOVE SR	Move to/from SR
MOVE USP	Move to/from USP
ORI SR	OR immediate to SR
RESET	Reset processor
RTE	Return from exception
STOP	Stop processor
CHK	Check register
ILLEGAL	Illegal instruction
TRAP	Trap call
TRAPV	Trap on overflow
ANDI CCR	AND immediate to CCR
ORI CCR	OR immediate to CCR
EORI CCR	EOR immediate to CCR
MOVE CCR	Move to/from CCR
NOP	No operation

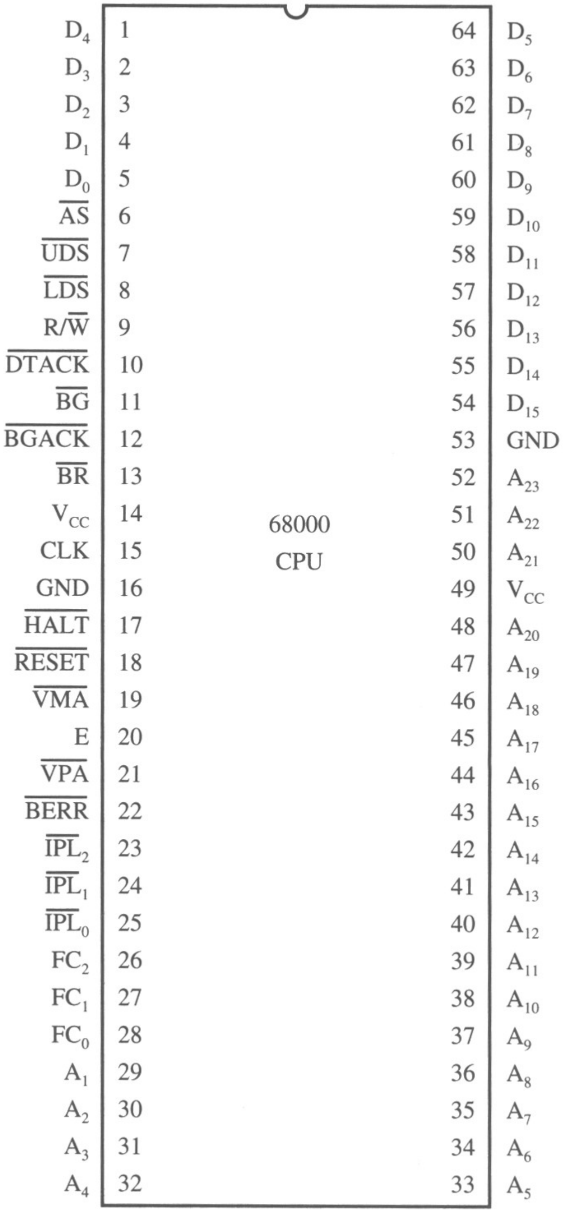
*Note: *cc* stands for condition code

Motorola 68000 µP – Hardware

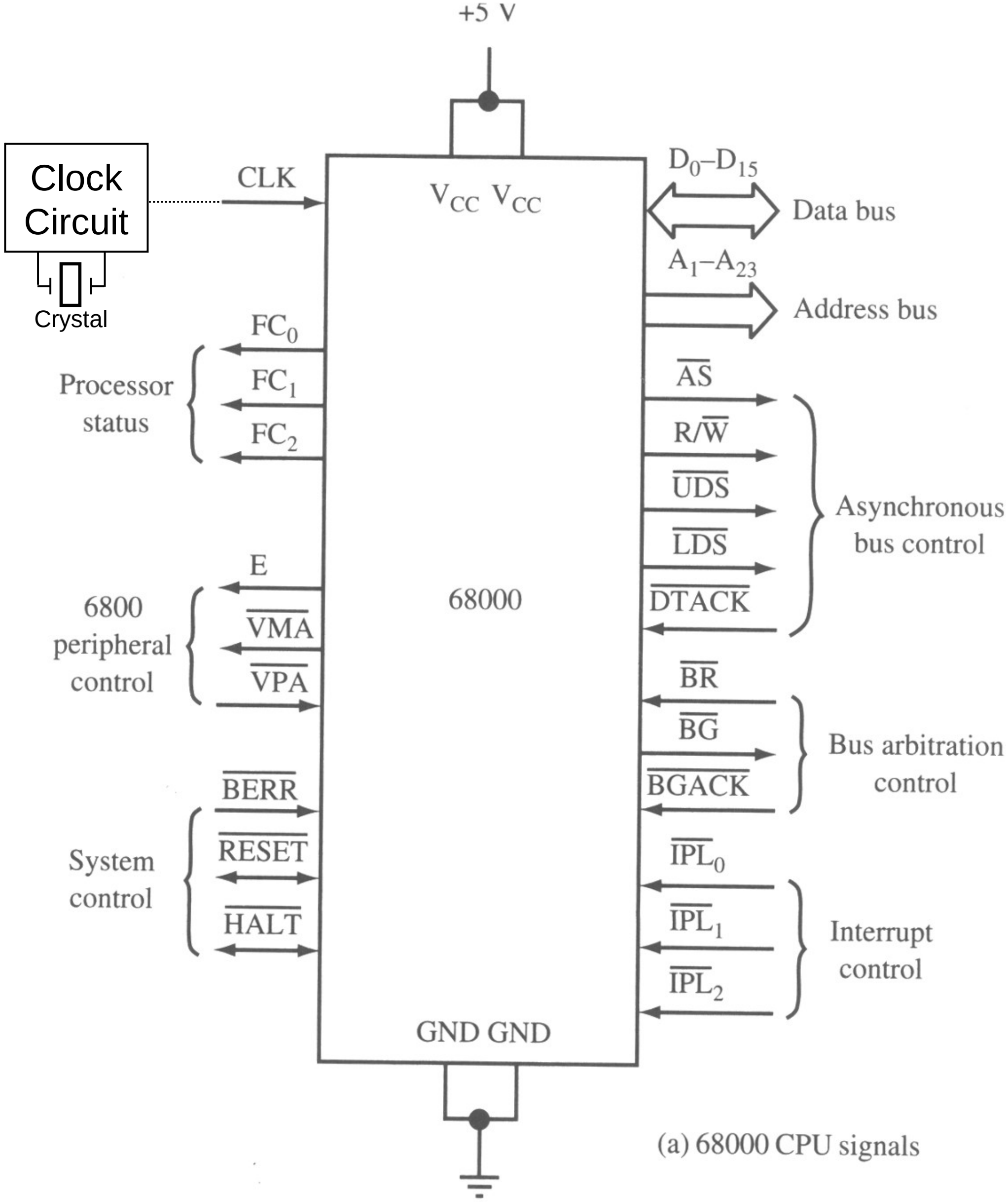
- 16-bit µP - 16-bit data bus (D_{15}-D_0).
  - Internal data paths are 32 bit
- 24-bit address bus.
  - ($\overline{\text{UDS}}$, $\overline{\text{LDS}}$, A_1-A_{23})
- No multiplexing of busses!
- Clock speeds of 4-12.5 MHz.
  - 10/12/16/20 MHz for CMOS version (1/10th power consumption)

Note that we will use ‘d7’ for data register d7 and ‘D7’ for data bus line D7. Likewise for a7 vs. A7.

Motorola 68000 µP – Hardware



D_4 1 | 64 D_5
D_3 2 | 63 D_6
D_2 3 | 62 D_7
D_1 4 | 61 D_8
D_0 5 | 60 D_9
$\overline{AS}$ 6 | 59 D_{10}
$\overline{UDS}$ 7 | 58 D_{11}
$\overline{LDS}$ 8 | 57 D_{12}
$R/\overline{W}$ 9 | 56 D_{13}
$\overline{DTACK}$ 10 | 55 D_{14}
$\overline{BG}$ 11 | 54 D_{15}
$\overline{BGACK}$ 12 | 53 GND
$\overline{BR}$ 13 | 52 A_{23}
V_{CC} 14 | 51 A_{22}
CLK 15 | 50 A_{21}
GND 16 | 49 V_{CC}
$\overline{HALT}$ 17 | 48 A_{20}
$\overline{RESET}$ 18 | 47 A_{19}
$\overline{VMA}$ 19 | 46 A_{18}
E 20 | 45 A_{17}
$\overline{VPA}$ 21 | 44 A_{16}
$\overline{BERR}$ 22 | 43 A_{15}
$\overline{IPL}_2$ 23 | 42 A_{14}
$\overline{IPL}_1$ 24 | 41 A_{13}
$\overline{IPL}_0$ 25 | 40 A_{12}
FC_2 26 | 39 A_{11}
FC_1 27 | 38 A_{10}
FC_0 28 | 37 A_9
A_1 29 | 36 A_8
A_2 30 | 35 A_7
A_3 31 | 34 A_6
A_4 32 | 33 A_5
68000
CPU



(a) 68000 CPU signals

Motorola 68000 µP – Asynchronous bus control

- $\overline{AS}$ Address strobe: valid address is on address bus.
- $R/\overline{W}$: for read, 0 for write.
- $\overline{UDS}$: Upper data strobe. Data on D_{15}-D_{8} (like $\overline{BHE}$).
- $\overline{LDS}$: Lower data strobe. Data on D_{7}-D_{0} (like $\overline{BLE}$).
- $\overline{DTACK}$: Data transfer acknowledge.
  - Signal by external hardware that µP may complete the current bus cycle.
  - During read, µP latches data when $\overline{DTACK}$ = 0.
  - During write, µP puts data on bus and keeps it there until $\overline{DTACK}$ = 0.

Motorola 68000 µP – Asynchronous bus control

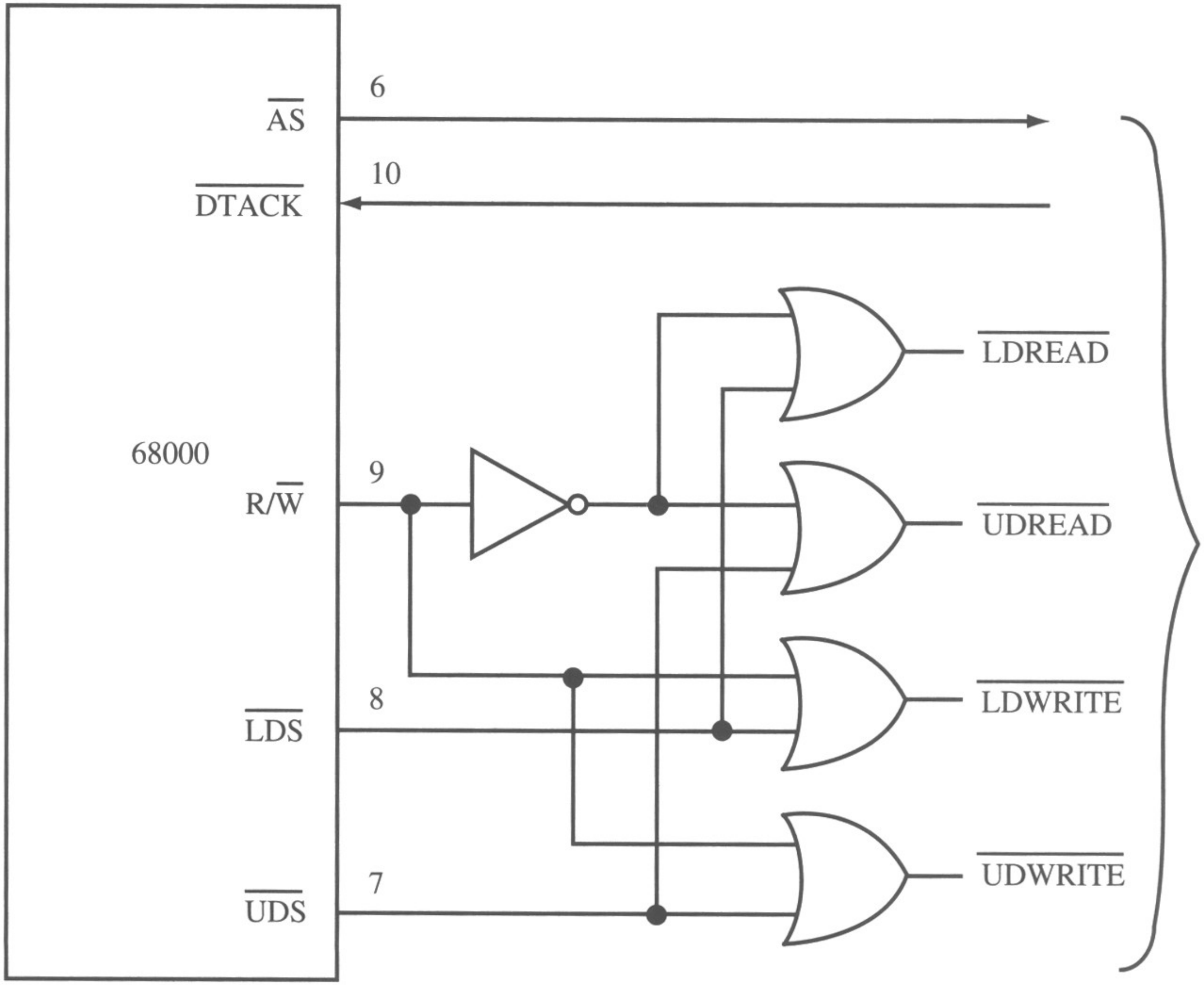


FIGURE 7.12 Decoding memory read/write signals

Motorola 68000 µP – Hardware

- System control
  - $\overline{\text{RESET}}$: reset the µP. (in)
  - $\overline{\text{HALT}}$: µP puts the busses into high-impedance state
    - (Equivalent to HOLD on 8086) (in/out).
  - $\overline{\text{BERR}}$: Bus error - illegal memory location (in)
    - YOU must generate this if $\overline{\text{DTACK}}$ or $\overline{\text{VPA}}$ never returns

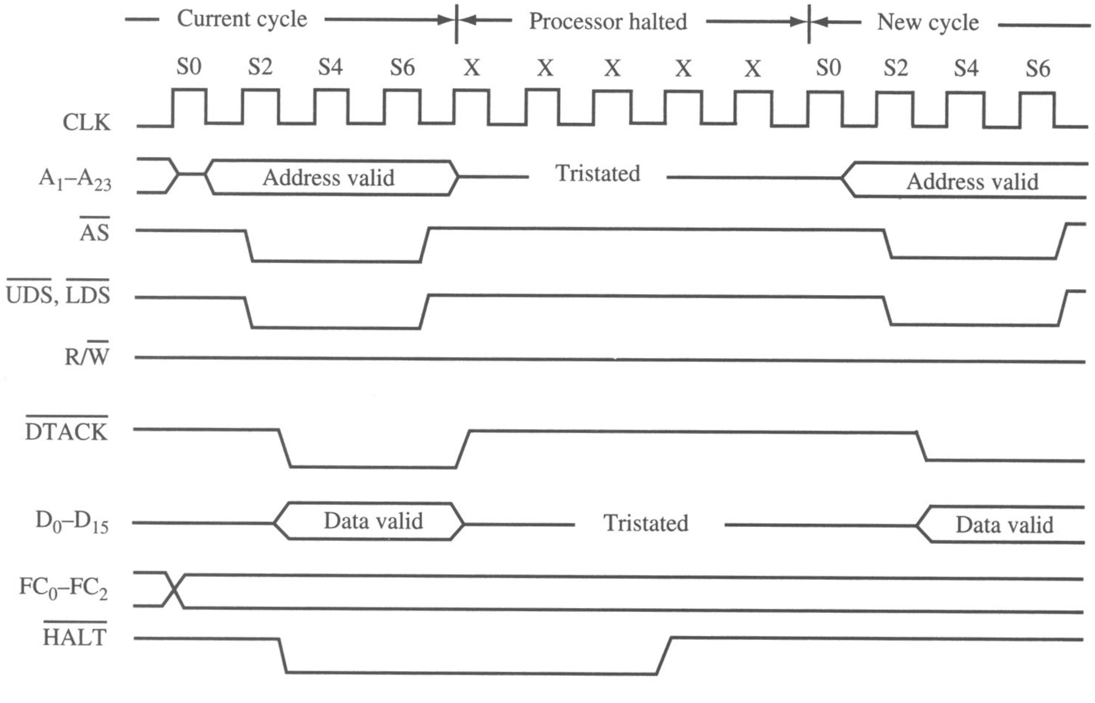
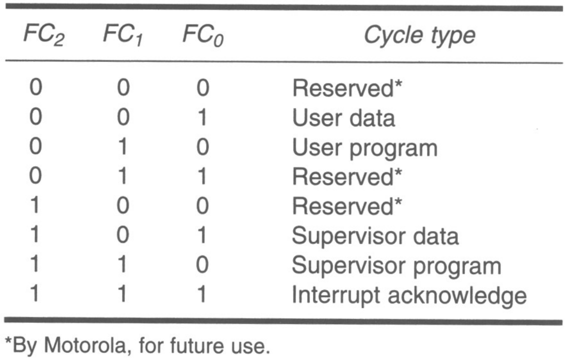


FIGURE 7.14 Processor HALT timing

Motorola 68000 µP – Hardware

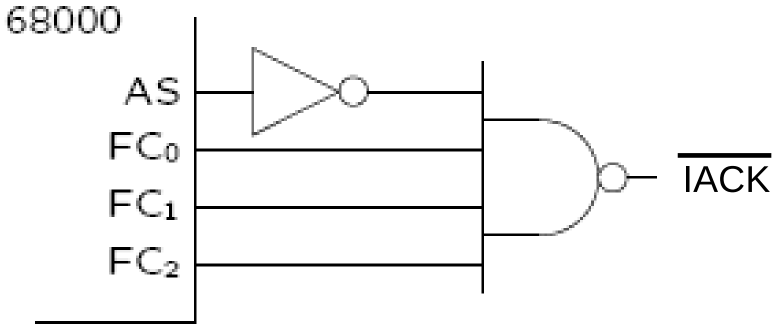
- FC_0, FC_1, FC_2:
  - Encoded processor states.
  - only valid with $\overline{AS} = 0$ (address strobe).
  - $FC_0FC_1FC_2 = 111$: interrupt acknowledge.

TABLE 7.1 Function code outputs



FC_2	FC_1	FC_0	*Cycle type*
0	0	0	Reserved*
0	0	1	User data
0	1	0	User program
0	1	1	Reserved*
1	0	0	Reserved*
1	0	1	Supervisor data
1	1	0	Supervisor program
1	1	1	Interrupt acknowledge

*By Motorola, for future use.



- $\overline{IPL_0}$, $\overline{IPL_1}$, $\overline{IPL_2}$
  - Encoded interrupt priority level.
  - Seven interrupt levels.
  - Level 7 (all zeros) is highest

Motorola 68000 μP – Bus arbitration

- Bus arbitration control:
  - $\overline{BR}$: Bus request (in)
  - $\overline{BG}$: Bus grant (out)
  - $\overline{BGACK}$: Bus grant acknowledgment (in)
- Used to place 68000 busses in high impedance state so that a peripheral can use the bus.
- Sequence:
  1. External device sets $\overline{BR} = 0$.
  2. 68000 sets $\overline{BG} = 0$.
  3. External device waits for $\overline{BG} = 0$, $\overline{AS} = 1$, $\overline{DTACK} = 1$, $\overline{BGACK} = 1$, then will set $\overline{BGACK} = 0$ to take control of busses.

Motorola 68000 μP – Bus arbitration

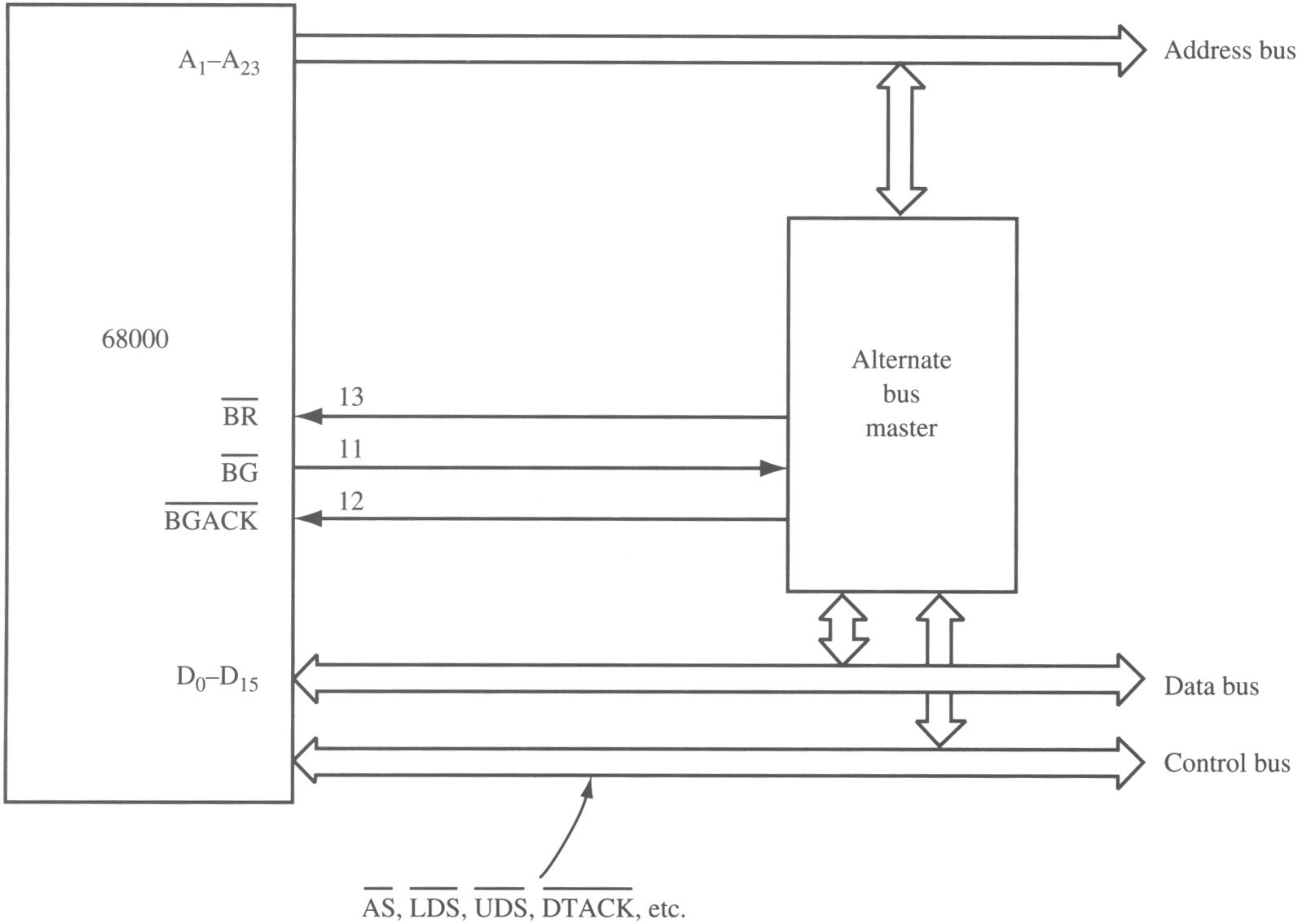
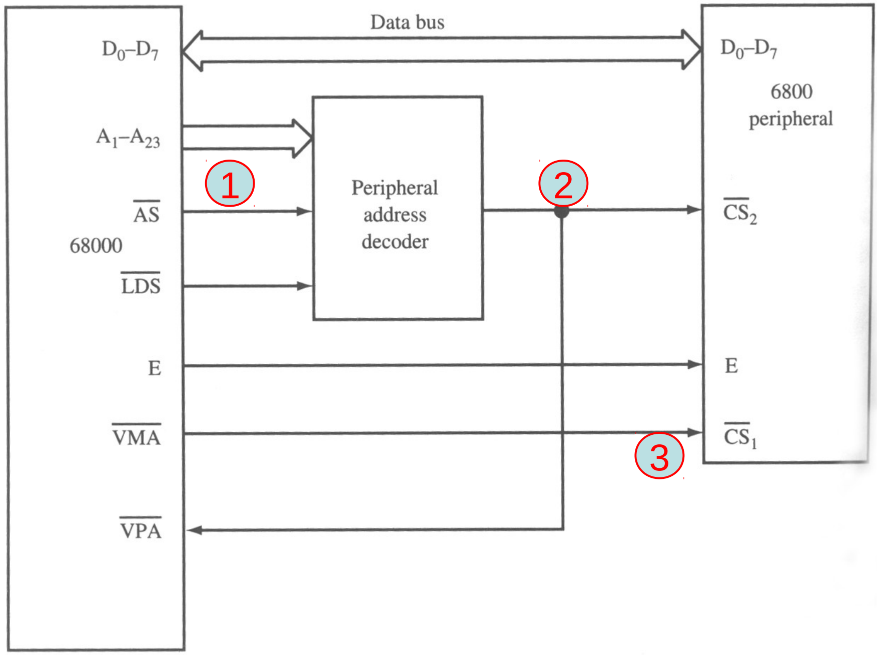


FIGURE 7.11 Bus arbitration logic block diagram

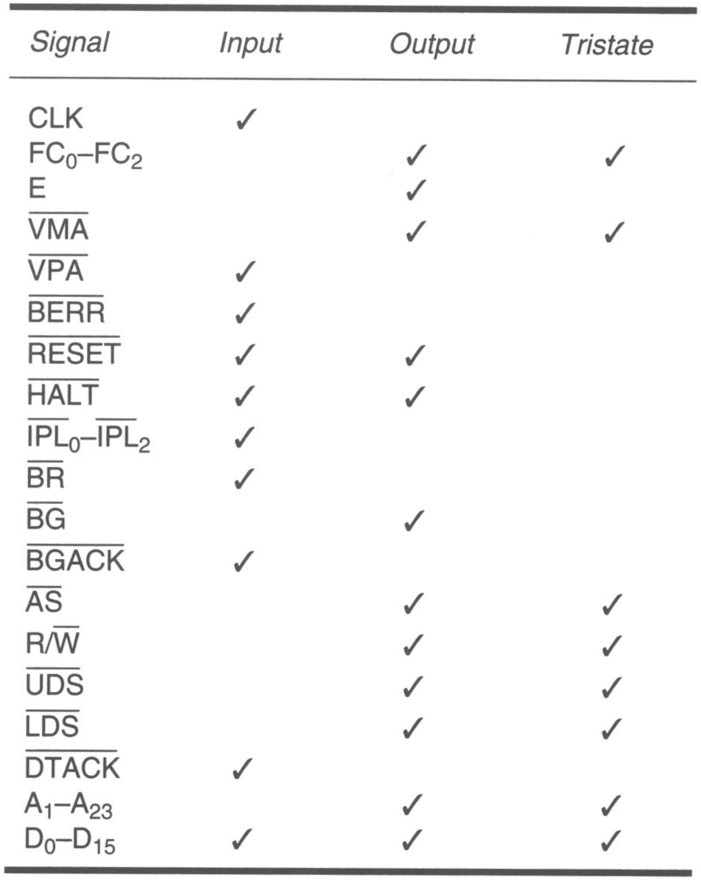
Motorola 68000 µP – Hardware

- Interface to 6800 peripherals:
  - E: Clock (out)
  - $\overline{\text{VPA}}$: Valid Peripheral address (in).
    - Should be asserted by interface circuitry whenever a 6800 peripheral has been selected.
  - $\overline{\text{VMA}}$: Valid Memory address (out).
    - Asserted by µP when internal clock is in synch with E-clock. Connected to second peripheral CS pin.



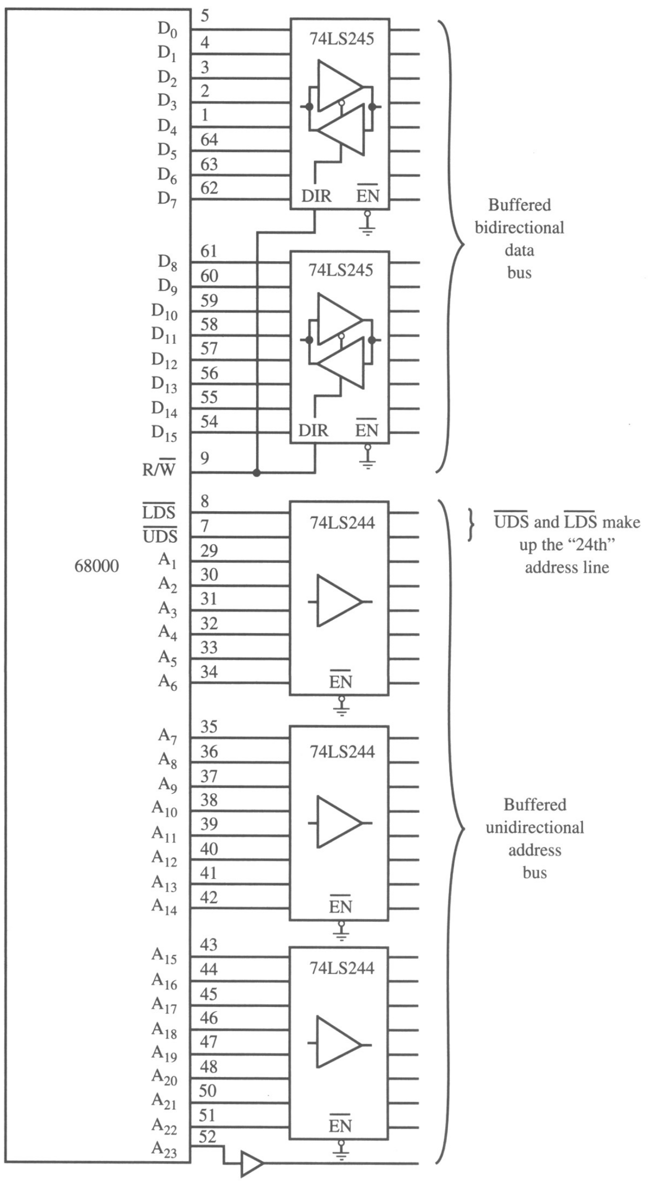
Motorola 68000 µP – Fully Buffered 68000

TABLE 7.3 Summary of 68000 signals



Signal	Input	Output	Tristate
CLK	✓		
FC_0–FC_2		✓	✓
E		✓	
$\overline{VMA}$		✓	✓
$\overline{VPA}$	✓		
$\overline{BERR}$	✓		
$\overline{RESET}$	✓	✓	
$\overline{HALT}$	✓	✓	
$\overline{IPL_0}$–$\overline{IPL_2}$	✓		
$\overline{BR}$	✓		
$\overline{BG}$		✓	
$\overline{BGACK}$	✓		
$\overline{AS}$		✓	✓
$R/\overline{W}$		✓	✓
$\overline{UDS}$		✓	✓
$\overline{LDS}$		✓	✓
$\overline{DTACK}$	✓		
A_1–A_{23}		✓	✓
D_0–D_{15}	✓	✓	✓

FIGURE 7.13 Buffering the address and data buses



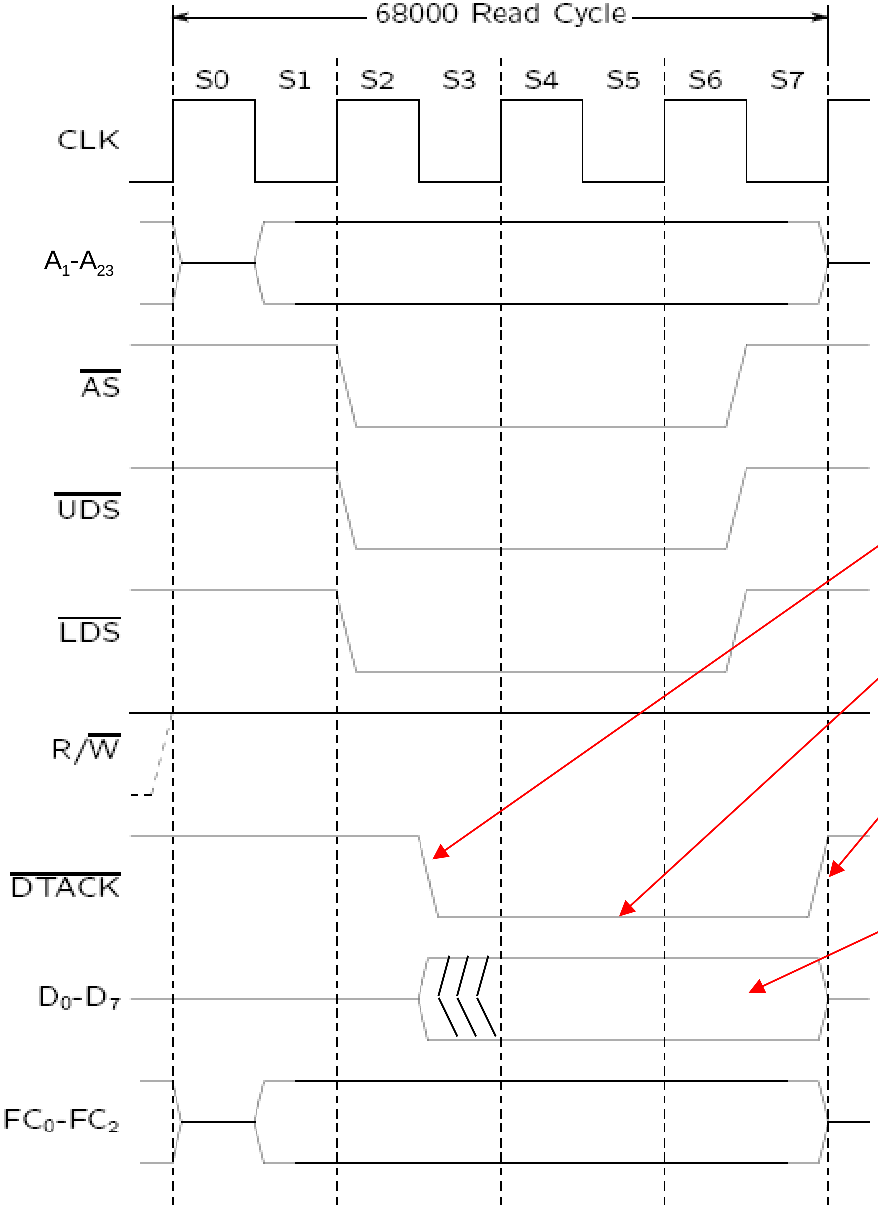
Motorola 68000 µP – Read/Write Bus Cycles

- The 68000µP uses $\overline{\text{AS}}=0$ to signal a memory access.
- When memory sees $\overline{\text{AS}}=0$, and it is the correct address, it responds by pulling $\overline{\text{DTACK}}$ low which tells the µP to proceed with the data transfer.
- Bus cycles are divided into a minimum of eight states, S0 - S7. Each state is 1/2 a clock cycle.

Motorola 68000 µP – Read Cycle

- **S0**: Address bus is in high impedance state, R/$\overline{\text{W}}$ is set to 1 (read operation).
- **S1**: Valid address appears on address bus.
- **S2**: $\overline{\text{AS}}$ goes low (valid address), $\overline{\text{LDS}}$ and $\overline{\text{UDS}}$ are set to the desired state.
- **S3**,**S4**: Minimum time given to memory to signal with $\overline{\text{DTACK}}$=0.
- **S5**: µP looks for $\overline{\text{DTACK}}$=0.
  - if $\overline{\text{DTACK}}$=1, insert two wait states then test $\overline{\text{DTACK}}$ again.
  - if $\overline{\text{DTACK}}$=0, continue with S6 and S7.
- **S6**: Nothing new happens.
- **S7**: Latch data into µP, set $\overline{\text{AS}}$, $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$ to 1. Memory releases $\overline{\text{DTACK}}$

Motorola 68000 µP – Read Cycle



ASYNCHRONOUS!
From memory device.

CHECK $\overline{\text{DTACK}}$
(S5)

$\overline{\text{DTACK}}$ released when
$\overline{\text{AS}}$ released

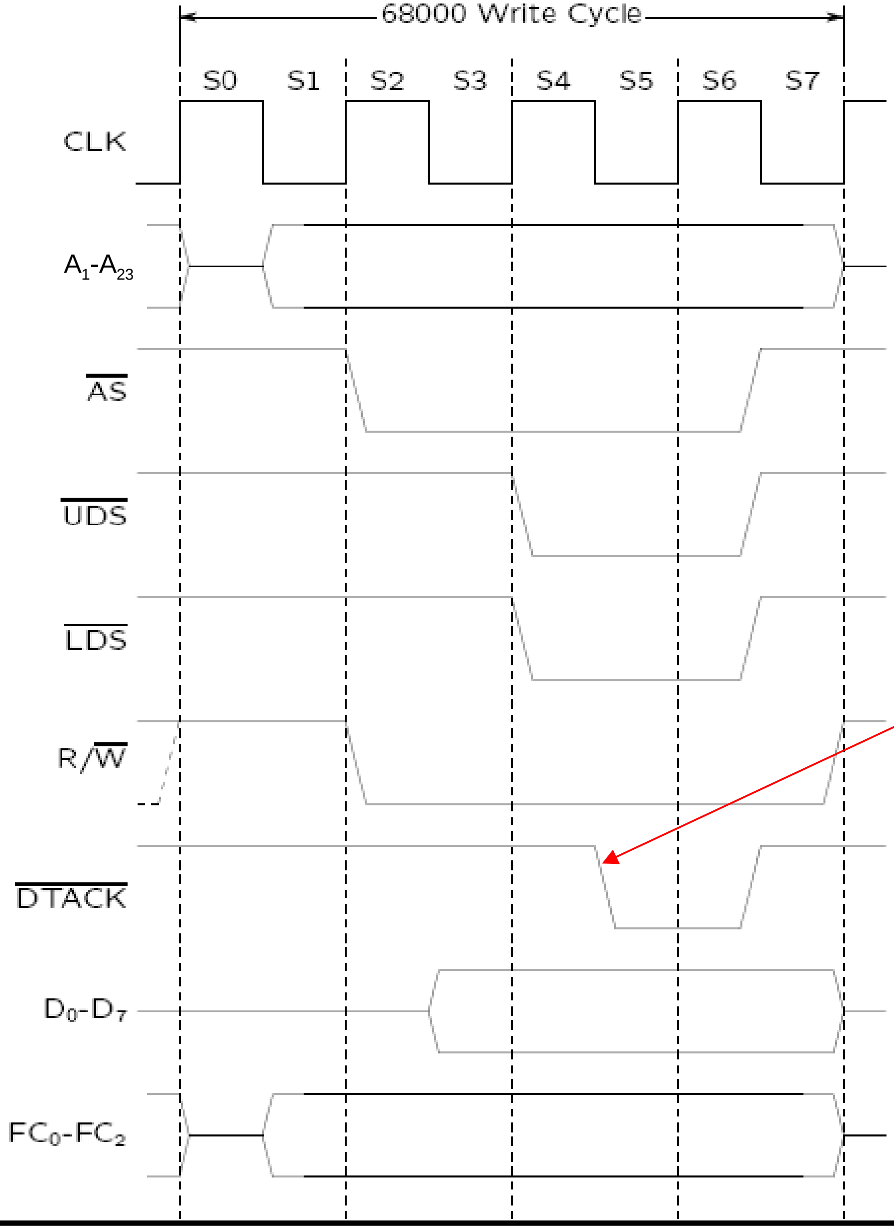
READ DATA
(end of S6)

Data may not be
avail when $\overline{\text{DTACK}}$ drops
Only guaranteeing that it
WILL be ready in time
(end of S6)

Motorola 68000 µP – Write Cycle

- **S0**: Address bus is in high impedance state.
- **S1**: Valid address appears on address bus.
- **S2**: $\overline{\text{AS}}$ goes low (valid address), R/$\overline{\text{W}}$ is set to 0 (write operation). ($\overline{\text{LDS}}$ and $\overline{\text{UDS}}$ are delayed to allow the bus transceivers to switch direction, and to allow the memory time to prepare.)
- **S3**: Valid data is placed on the bus by the µP.
- **S4**: $\overline{\text{LDS}}$ and $\overline{\text{UDS}}$ are set to the desired state.
- **S5**: µP looks for $\overline{\text{DTACK}}$=0.
  - – if $\overline{\text{DTACK}}$=1, insert two wait states then test $\overline{\text{DTACK}}$ again.
  - – if $\overline{\text{DTACK}}$=0, continue with S6 and S7.
- **S6**: Nothing new happens.
- **S7**: Set $\overline{\text{AS}}$, $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$ to 1. Memory releases $\overline{\text{DTACK}}$

Motorola 68000 μP – Write Cycle



ASYNCHRONOUS! From memory, can arrive any time before S5 without causing wait states

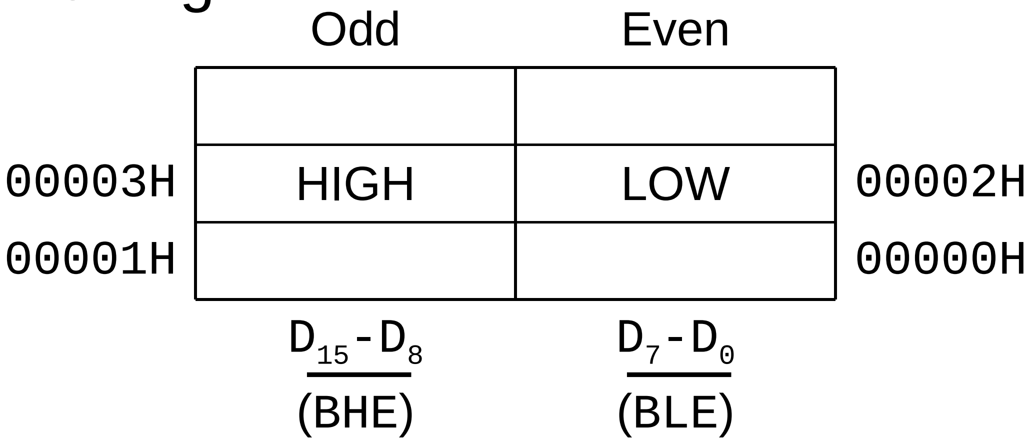
Motorola 68000 μP – Memory Organization

- 68000 is byte-addressable.
- 16-bit words and 32-bit long words must begin at an even address, otherwise address error.
- 68000 is a big-endian processor:
  - 16-bit words are stored with the lower-order byte (endian) in the higher-order (big) memory address.
  - (Recall that the 8086 is little-endian)
- Consequences:

	Bank	
	8086	68000
D_7-D_0	Even ($\overline{BLE}$)	Odd ($\overline{LDS}$)
D_{15}-D_8	Odd ($\overline{BHE}$)	Even ($\overline{UDS}$)

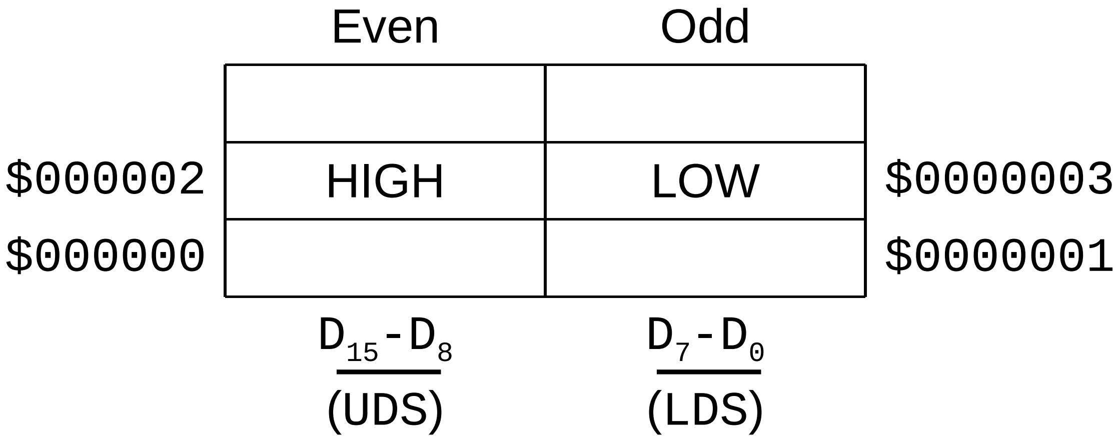
Motorola 68000 µP – Memory Organization

- 8086 memory is usually drawn with *odd* bank on left, and *even* bank on right.



	Odd	Even	
00003H	HIGH	LOW	00002H
00001H			00000H
	D_{15}-D_8	D_7-D_0	
	($\overline{\text{BHE}}$)	($\overline{\text{BLE}}$)	

- 68000 memory is usually drawn with *even* bank on left, and *odd* bank on right.



	Even	Odd	
$000002	HIGH	LOW	$0000003
$000000			$0000001
	D_{15}-D_8	D_7-D_0	
	($\overline{\text{UDS}}$)	($\overline{\text{LDS}}$)	

Motorola 68000 µP – Memory Organization

- Ex 68000 memory segment:

	Even (High)	Odd (Low)	
$00100A	C3	8F	$00100B
$001008	3F	6B	$001009
$001006	00	4A	$001007
$001004	23	08	$001005

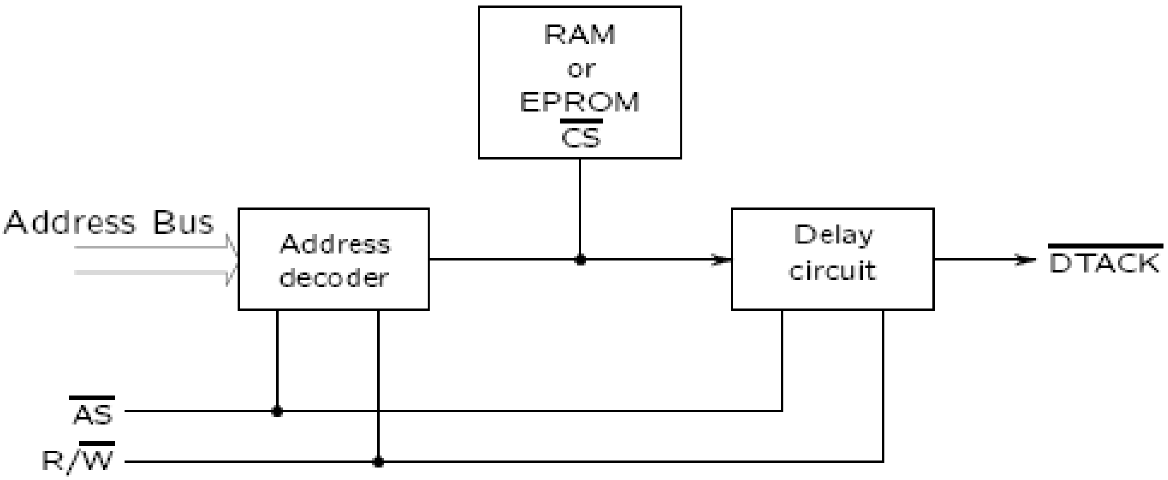
Byte@$1004	=	$23 (high half)
Byte@$1005	=	$08 (low half)
Word@$1006	=	$004A (both halves)
Word@$1008	=	$3F6B (both halves)
Word@$1009	=	Address Error
Long word@$1008	=	$3F6BC38F (both halves, twice)
Long word@$1005	=	Address Error

Motorola 68000 µP – Memory Interfacing

- Almost identical to the 8086 except:
  1. Switch even and odd banks
  2. Must generate $\overline{\text{DTACK}}$
  3. Must use $\overline{\text{AS}}$, R/$\overline{\text{W}}$, $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ for control.
- During a byte-read operation, the µP will select the correct half of the data bus depending on whether it's an even or odd address (similar to 8086). (However, most designs provide separate read strobes for even and odd banks.)
- Separate write strobes are required for even and odd banks so that data is not written to the wrong memory bank.

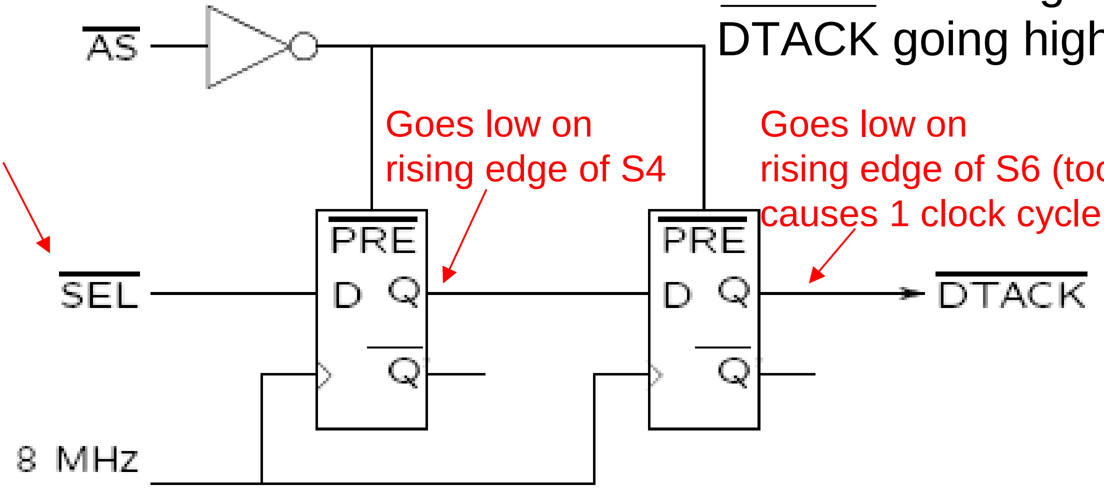
Motorola 68000 µP – Memory Interfacing

- Block diagram of $\overline{\text{DTACK}}$ circuit:



- $\overline{\text{DTACK}}$ delay generator:

Note that $\overline{\text{AS}}$ high leads to $\overline{\text{DTACK}}$ going high immediately



'Device select' signal from address decoder. (goes low end of S1)

Goes low on rising edge of S4

Goes low on rising edge of S6 (too late, causes 1 clock cycle delay)

'PRE' presets the flip-flop (sets it to 1)

Motorola 68000 μP – I/O Interfacing

- All I/O is memory-mapped.
- Decoding is the same as for memory.
- One still must generate $\overline{\text{DTACK}}$.

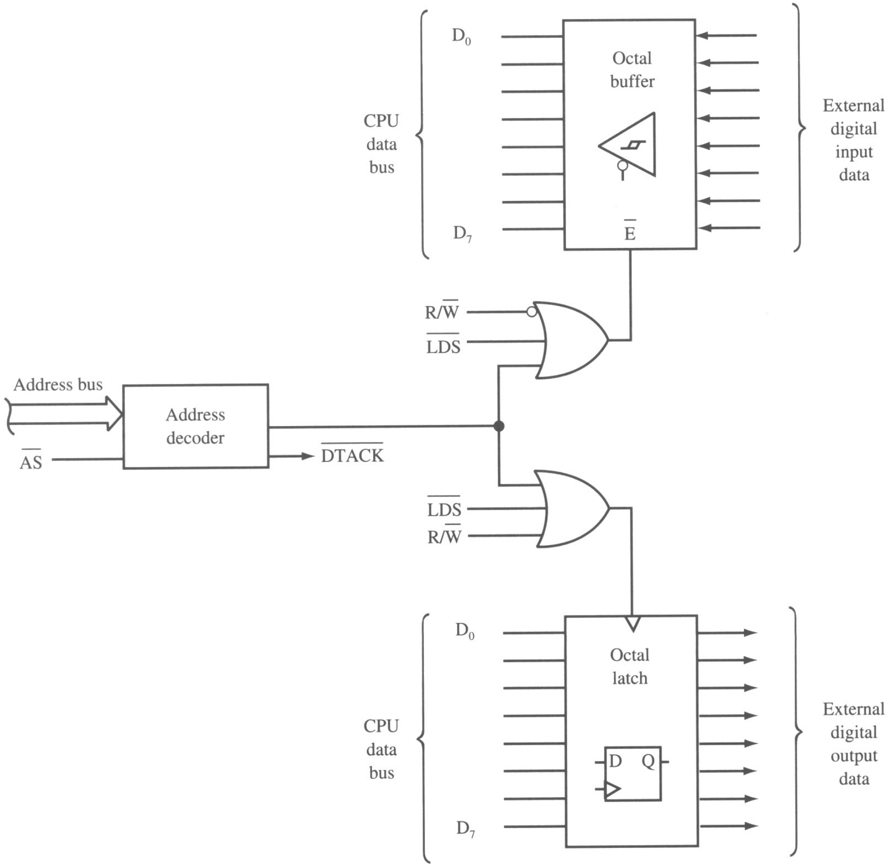
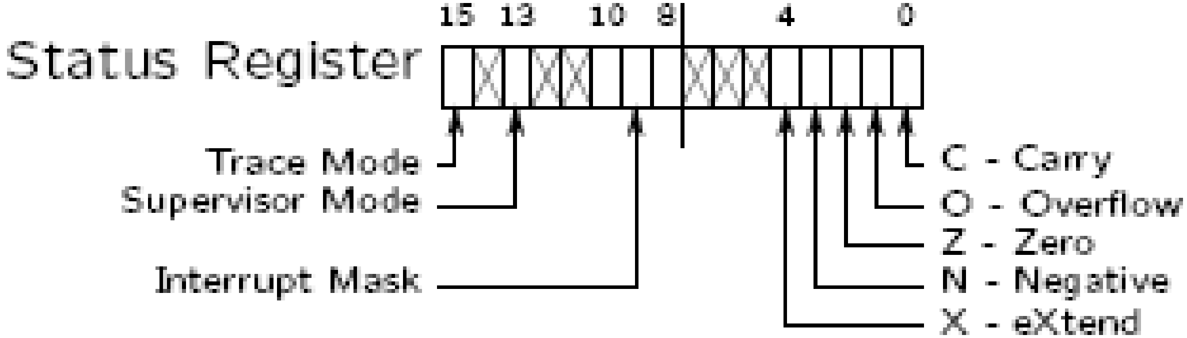


FIGURE 9.1 Memory-mapped I/O circuitry

Would require another buffer/latch pair for $\overline{\text{UDS}}$ for a 16-bit I/O interface. (connected to D_{15}-D_8).

Motorola 68000 µP – Exceptions (Interrupts)

- The 68000 has three execution states:
  1. Normal - running user program.
  2. Halted - not executing instructions. (perhaps because of a system failure such as a double bus fault, or due to $\overline{\text{HALT}}$ pin).
  3. Exception (processing) state - includes interrupts, but goes beyond the usual notion of interrupts.



Motorola 68000 μP – Exceptions (Interrupts)

- Two privilege states.
  - User and Supervisor.
  - Some instructions are only available in supervisor state.
    - `STOP`, `RESET`, `RTE`, `MOVE/AND/EOR/OR` to `SR`, `MOVE` to `USP`
  - Separate stack pointers.
  - Provides security for operating systems etc.
  - All exception processing is done in supervisor state.
  - The only way to get to supervisor state is through an exception (or reset).

Motorola 68000 μP – Exceptions (Interrupts)

- Can use privilege state for memory management
  - i.e. include FC2-pin in memory interface.

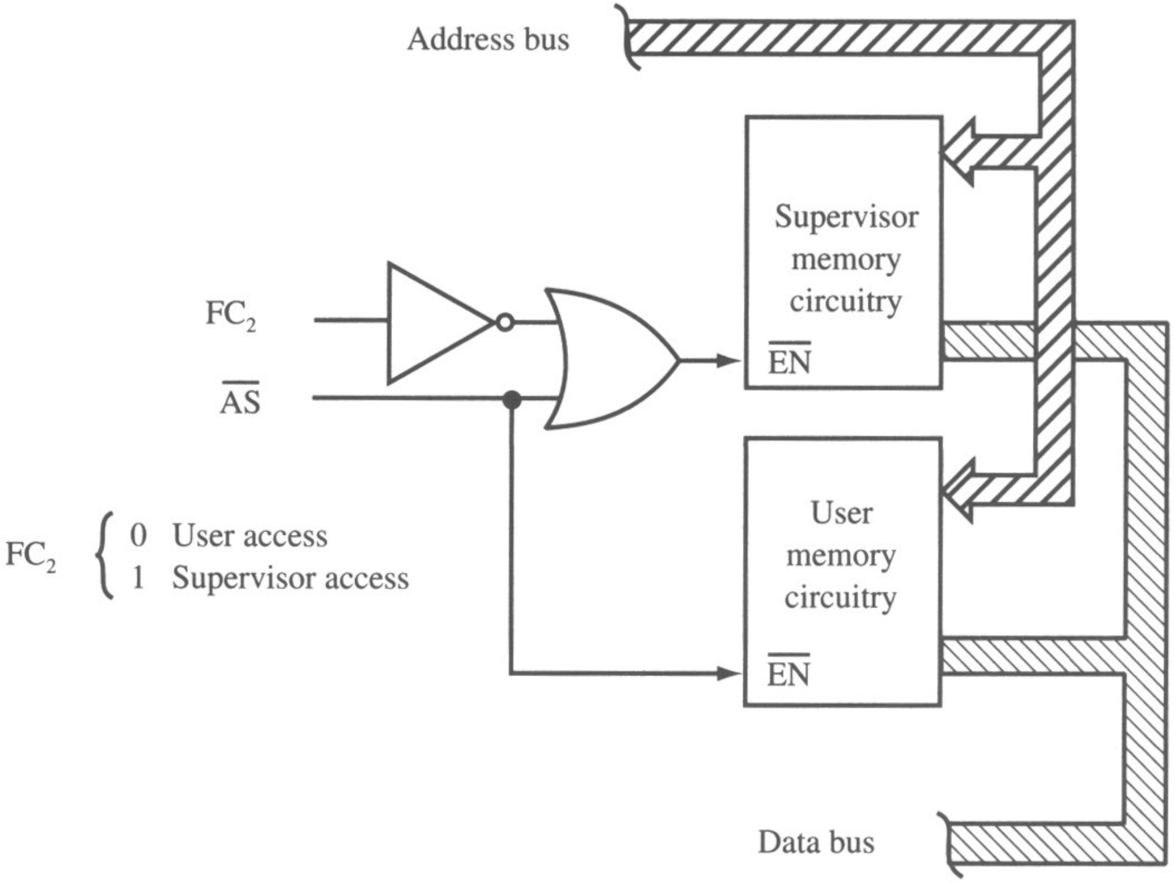
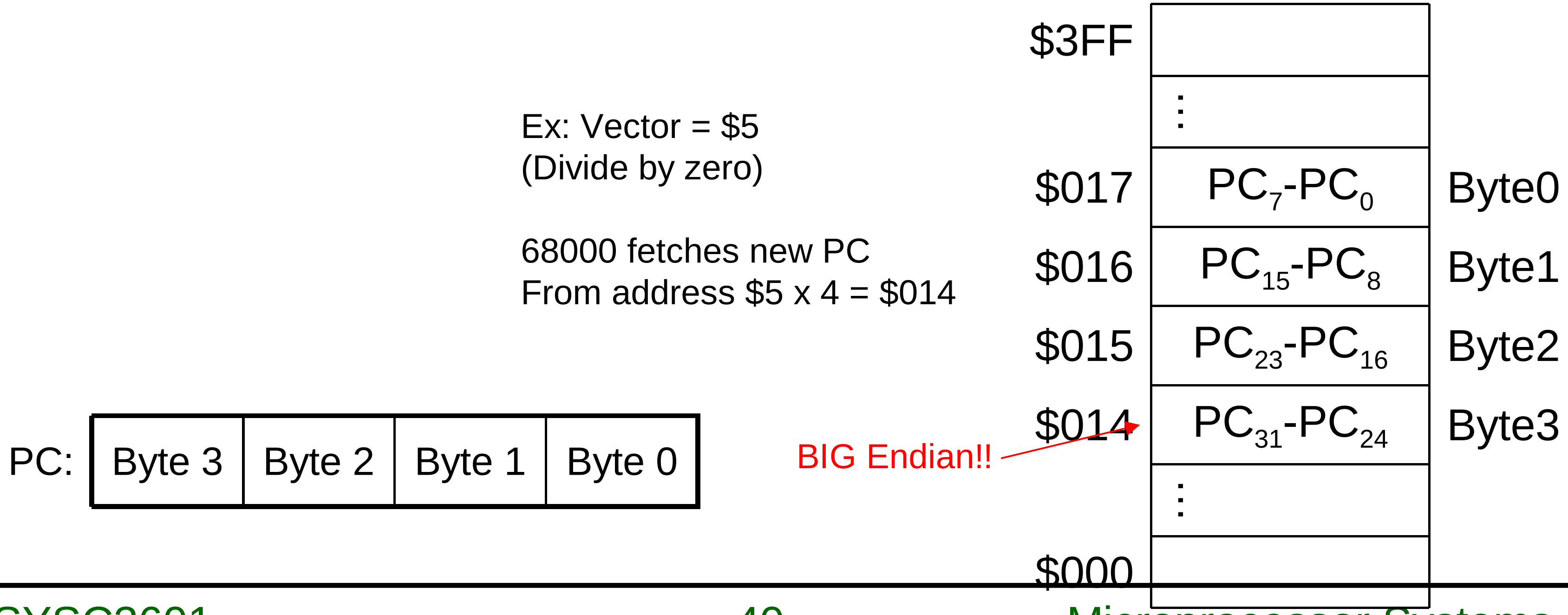


FIGURE 4.3 User/supervisor memory partitioning

Motorola 68000 µP – Exceptions (Interrupts)

- Interrupt Vectors and the vector table
  - A vector number is one byte (0-255)
  - Vector Table:
    - Occupies the first 1kbyte (`000-3FF`) of memory.
    - Each entry (except the first) is a 32-bit address pointing to the start of the exception handler code.



Ex: Vector = $5
(Divide by zero)

68000 fetches new PC
From address $5 x 4 = $014

$3FF		
	⋮	
$017	$PC_7\text{-}PC_0$	Byte0
$016	$PC_{15}\text{-}PC_8$	Byte1
$015	$PC_{23}\text{-}PC_{16}$	Byte2
$014	$PC_{31}\text{-}PC_{24}$	Byte3
	⋮	
$000		

BIG Endian!!

PC:				
	Byte 3	Byte 2	Byte 1	Byte 0

Motorola 68000 µP – Exceptions (Interrupts)

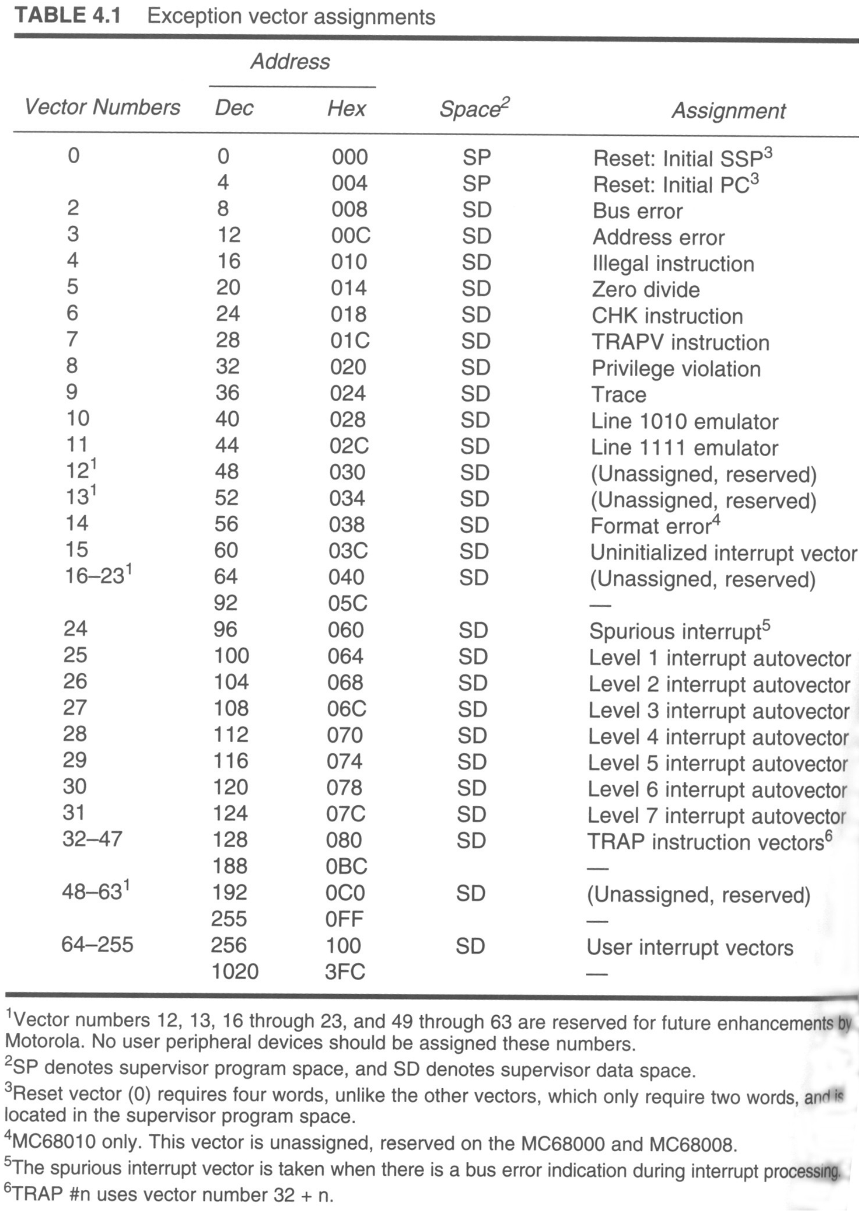


TABLE 4.1 Exception vector assignments

	Address			
Vector Numbers	*Dec*	*Hex*	*Space*[2]	*Assignment*
0	0	000	SP	Reset: Initial SSP[3]
	4	004	SP	Reset: Initial PC[3]
2	8	008	SD	Bus error
3	12	00C	SD	Address error
4	16	010	SD	Illegal instruction
5	20	014	SD	Zero divide
6	24	018	SD	CHK instruction
7	28	01C	SD	TRAPV instruction
8	32	020	SD	Privilege violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 emulator
11	44	02C	SD	Line 1111 emulator
12[1]	48	030	SD	(Unassigned, reserved)
13[1]	52	034	SD	(Unassigned, reserved)
14	56	038	SD	Format error[4]
15	60	03C	SD	Uninitialized interrupt vector
16–23[1]	64	040	SD	(Unassigned, reserved)
	92	05C		—
24	96	060	SD	Spurious interrupt[5]
25	100	064	SD	Level 1 interrupt autovector
26	104	068	SD	Level 2 interrupt autovector
27	108	06C	SD	Level 3 interrupt autovector
28	112	070	SD	Level 4 interrupt autovector
29	116	074	SD	Level 5 interrupt autovector
30	120	078	SD	Level 6 interrupt autovector
31	124	07C	SD	Level 7 interrupt autovector
32–47	128	080	SD	TRAP instruction vectors[6]
	188	0BC		—
48–63[1]	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64–255	256	100	SD	User interrupt vectors
	1020	3FC		—

[1] Vector numbers 12, 13, 16 through 23, and 49 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

[2] SP denotes supervisor program space, and SD denotes supervisor data space.

[3] Reset vector (0) requires four words, unlike the other vectors, which only require two words, and is located in the supervisor program space.

[4] MC68010 only. This vector is unassigned, reserved on the MC68000 and MC68008.

[5] The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.

[6] TRAP #n uses vector number 32 + n.

Motorola 68000 μP – Exceptions (Interrupts)

- Exception Processing Sequence

  1. Save the status register in a temporary register and set the S-bit so that the 68000 can enter supervisor mode.

  2. Get the vector number. There are several ways:

     (a) may be determined internally by processor.

     (b) external interrupts can be "auto-vectored" (to come).

     (c) external interrupts can provide a vector number (i.e. type) on D_7-D_0 during the interrupt acknowledge cycle.

Motorola 68000 μP – Exceptions (Interrupts)

3. Save processor information onto supervisor stack:
   (a) push PC low word.
   (b) push PC high word.
   (c) push status register (from saved temporary unmodified version).

4. Fetch new PC from the vector table.
5. Execute exception handler.
6. Return with RTE. Pops SR, then PC.

- Note: Exceptions can be nested – not masked automatically like 8086 does.

Motorola 68000 µP – Exceptions (Interrupts)

- 68000 Hardware Interrupts
  - Seven levels of external interrupts depending on $\overline{\mathtt{IPL2}}$, $\overline{\mathtt{IPL1}}$, and $\overline{\mathtt{IPL0}}$.
  - Level 0, all $\overline{\mathtt{IPLs}}$ = 1, no interrupt.
  - Level 7, all $\overline{\mathtt{IPLs}}$ = 0, highest priority (non-maskable).
  - Interrupt priority mask (bits 8, 9, and 10 of SR) is set to disable lower priority interrupts.

Example circuit to generate a level-7 interrupt using a single push-button.

We can develop more complex circuits to generate multiple interrupt levels depending on the source of the interrupt request.

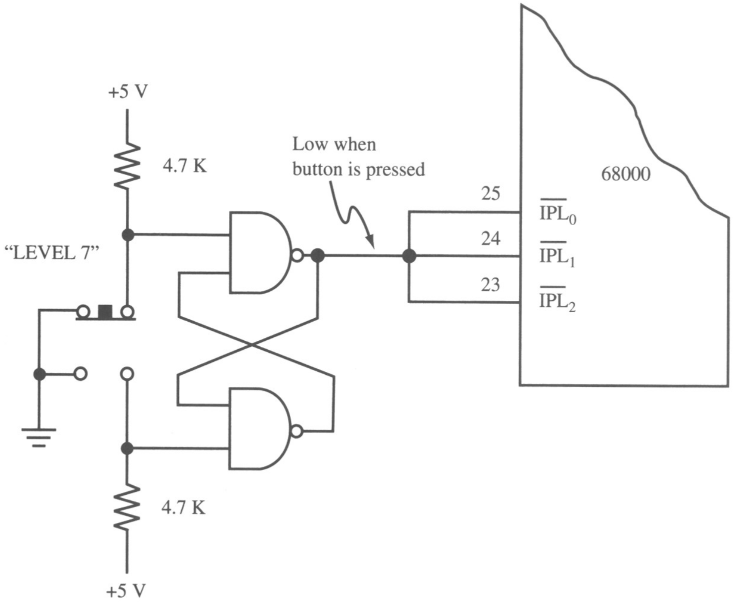

+5 V
4.7 K
"LEVEL 7"
Low when button is pressed
25
24
23
$\overline{\text{IPL}}_0$
$\overline{\text{IPL}}_1$
$\overline{\text{IPL}}_2$
68000
4.7 K
+5 V

FIGURE 7.10 Generating a level-7 interrupt with a pushbutton

Motorola 68000 µP – Exceptions (Interrupts)

- Interrupt Acknowledge Cycle
  - (asynchronous, hardware interrupt requests)

1. Device and interrupt logic set $\overline{\texttt{IPL2}}$, $\overline{\texttt{IPL1}}$and $\overline{\texttt{IPL0}}$.
2. µP completes current instruction.
3. µP enters interrupt acknowledge cycle.

   (a) FC2, FC1, FC0 = 111.

   (b) $\overline{\text{AS}}$ = 0, $\overline{\text{LDS}}$ = 0, R/$\overline{\text{W}}$ = 1.

   A_3, A_2, A_1 = requested interrupt level.

Motorola 68000 µP – Exceptions (Interrupts)

- Interrupt Acknowledge Cycle con’t

4. External logic may do one of two things:

(a) Supply a vector number.

- Place 8-bit vector number of $\mathtt{D_7}$-$\mathtt{D_0}$.
- pull $\overline{\mathtt{DTACK}}$ low.
- µP will read $\mathtt{D_7}$-$\mathtt{D_0}$.

(b) Request an “auto-vector".

- Pull $\overline{\mathtt{VPA}}$ low. Leave $\overline{\mathtt{DTACK}}$ high.
- µP generates its own vector based on interrupt level first supplied to $\overline{\mathtt{IPL}}$ inputs.
- autovectors point to locations `$064` through `$07F` in vector table.

– Autovectors should be used whenever 7 or less interrupt types are needed.

5. Proceed with exception handling steps from slide 42

Motorola 68000 µP – Exceptions (Interrupts)

he response to an interrupt is quite lengthy and complex:

. Resolve priorities from external interrupt request, present appropriate 3-bit code on IPL_{2-0}

. Monitor FC_{2-0} for intr acknowledge cycle.

- AS=0,
- R/W=1,
- LDS=0
- A_{3-1} = requested interrupt level

. Either:

3a) provide vector number on D_{7-0} and pull DTACK low,

OR

3b) request autovector by pulling VPA low.

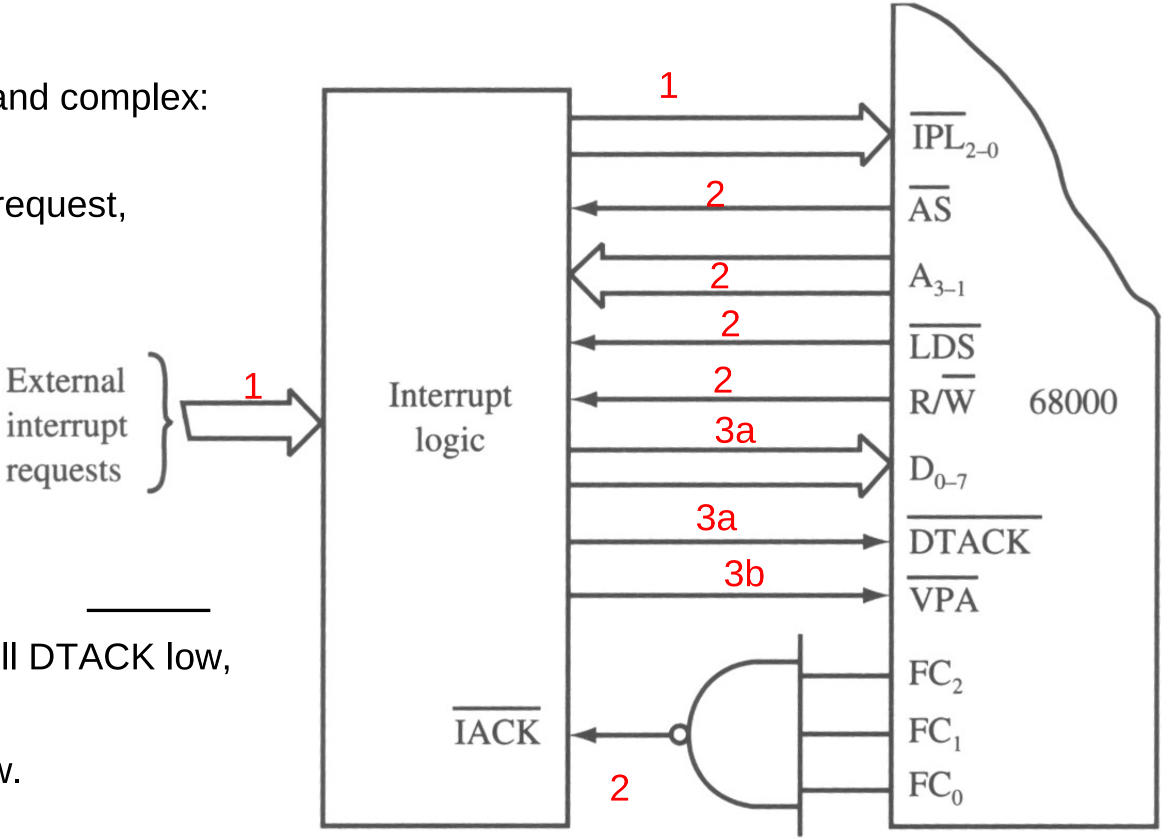


FIGURE 4.9 External interrupt circuitry block diagram

Motorola 68000 μP – Peripheral chips

68681 DUART

- Contains 2 UARTs, independently programmable
  - Both channels can provide simultaneous Tx/Rx.
- Interface using internal control/data/status registers selected via $RS_{4\text{-}1}$ pins.
- Also provides 6 parallel inputs and 8 parallel outputs
  - Can be used for handshaking signals or standard I/O pins.

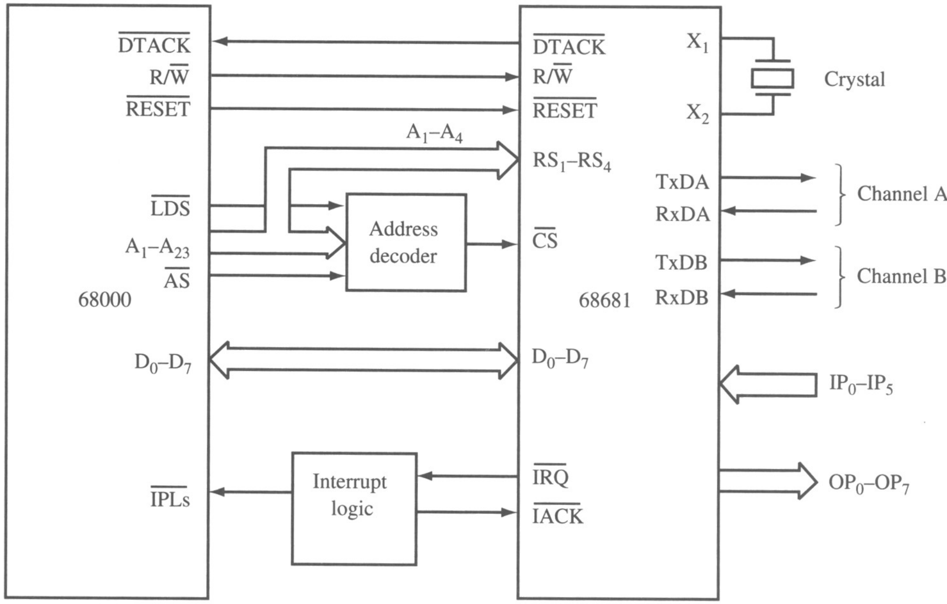
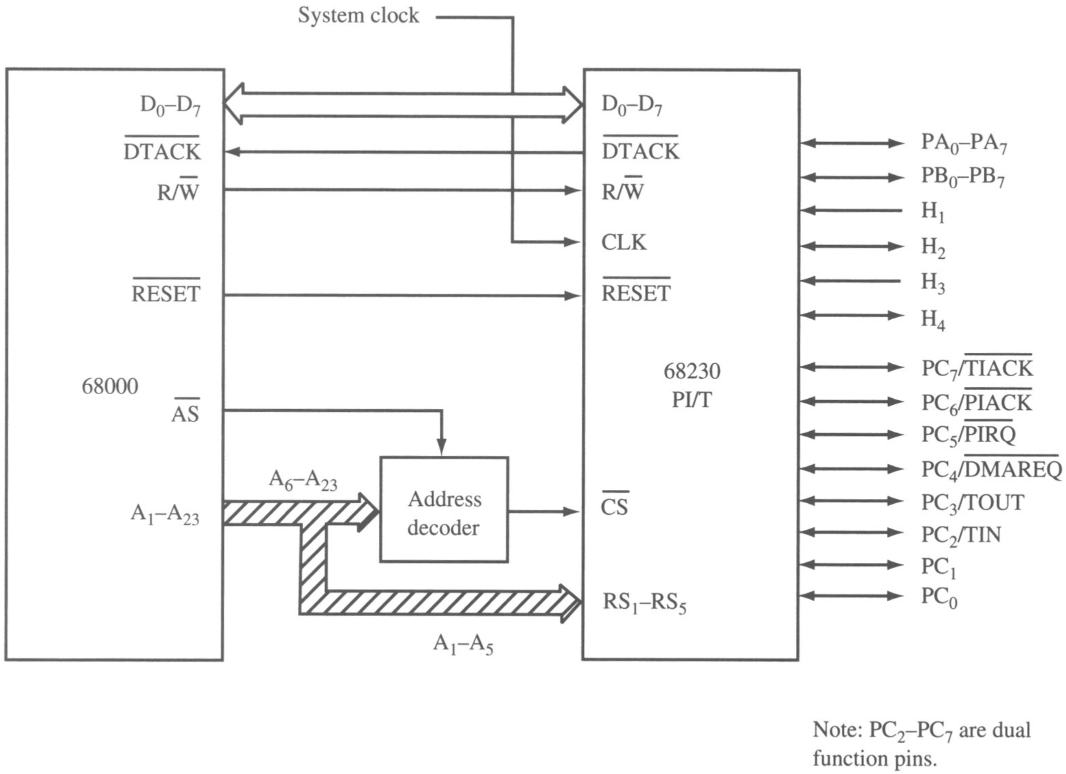


FIGURE 10.17 Using the 68681 in a 68000-based system

Motorola 68000 µP – Peripheral chips

68230 Parallel Interface/Timer (PI/T)

- Contains 3 8-bit parallel ports
  - Can be input, output, bidirectional
  - Ports A&B can form 16-bit port
- Also contains an internal 24 bit timer
  - Count down, square wave generator, etc.
- 23 internal data/control/status registers selected via $RS_{5\text{-}1}$ pins.
- $H_{4\text{-}1}$ pins are handshaking for ports A&B
  - Can cause interrupts
- PortC can be used as general I/O pins, interrupt request/acknowlege, timer inputs/outputs, or DMA request.



Note: PC_2–PC_7 are dual function pins.

FIGURE 10.25 Interfacing the 68230 PI/T

Motorola 68000 µP – Peripheral chips

Keyboard scanning with a 68230 PI/T

- Drive $PA_{3\text{-}0}$ in sequence
- Read $PB_{3\text{-}0}$ to check for key depressed

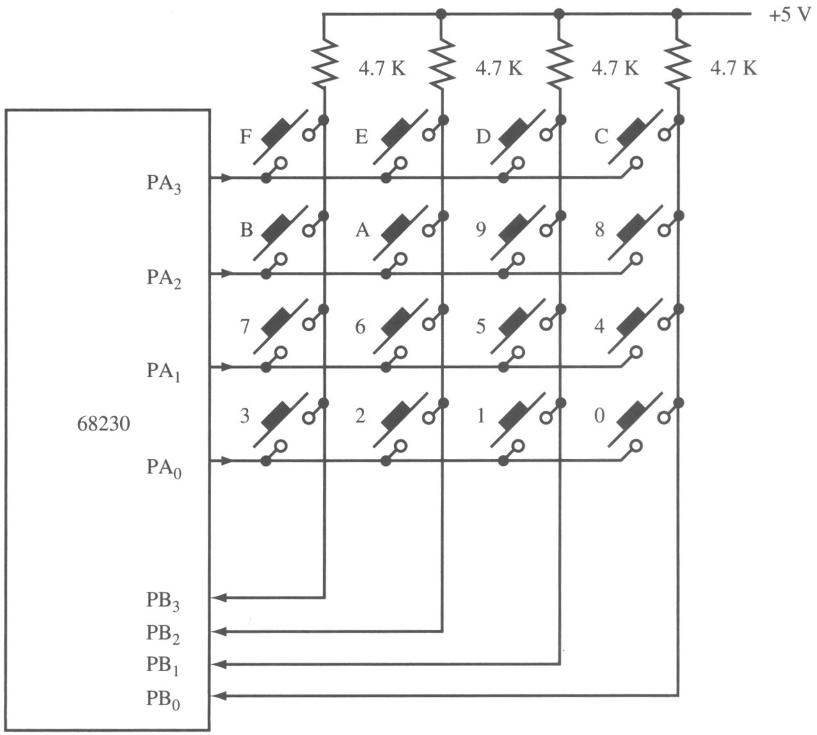


FIGURE 10.32 Sixteen-key keypad scanner using the 68230

TABLE 10.1 Keyboard scanning codes

Parallel outputs				
PA_3	*PA_2*	*PA_1*	*PA_0*	*Buttons scanned*
1	1	1	0	3, 2, 1, 0
1	1	0	1	7, 6, 5, 4
1	0	1	1	B, A, 9, 8
0	1	1	1	F, E, D, C

Motorola 68000 µP – Peripheral chips

4 Digit Display with a 68230 PI/T

- Select segments using $PA_{7\text{-}0}$
- Select display chip using $PB_{3\text{-}0}$
  - Have to scan through chips quickly to avoid flicker effect.
- Use timer to cause interrupts to cycle through chips.

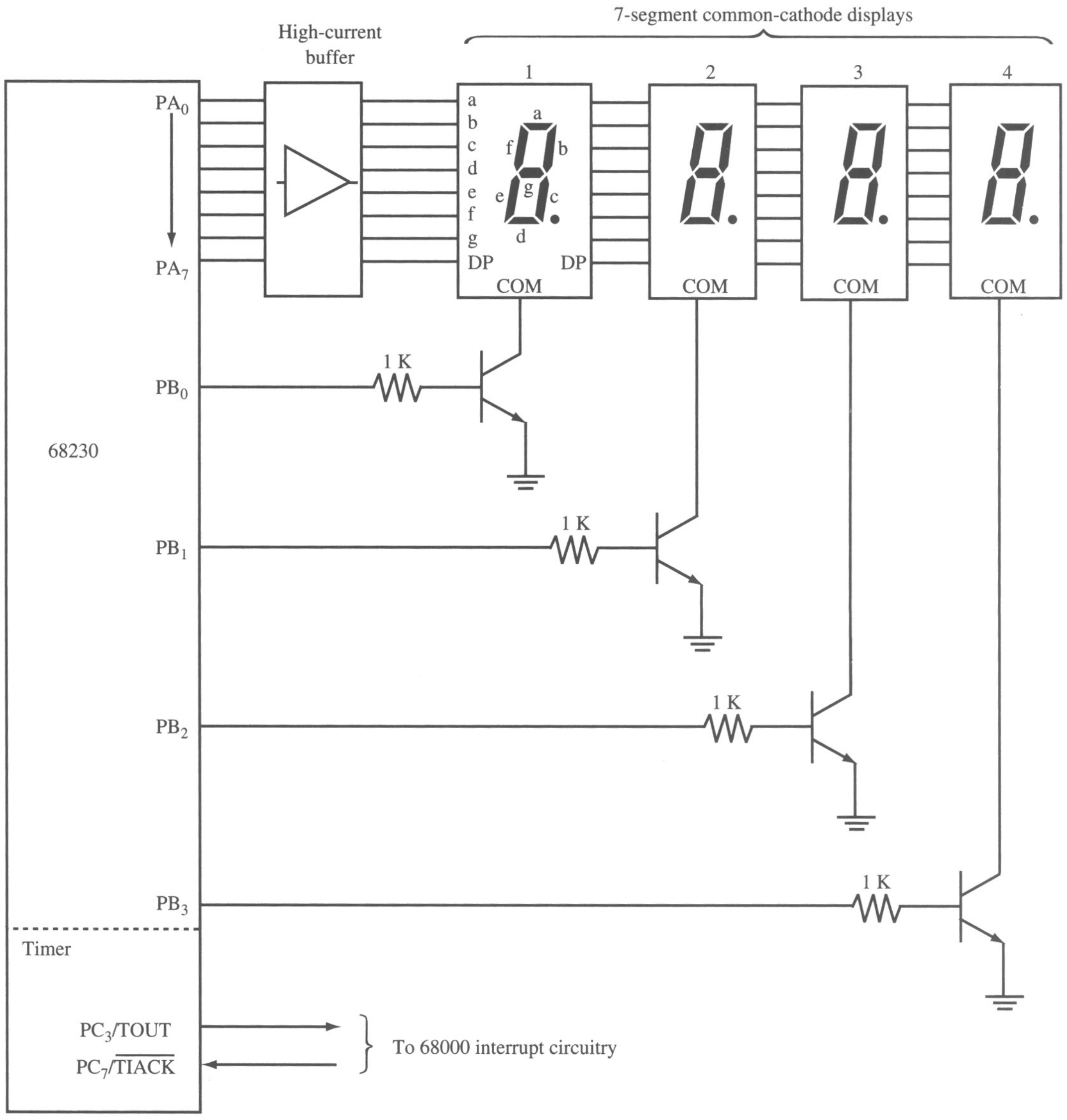
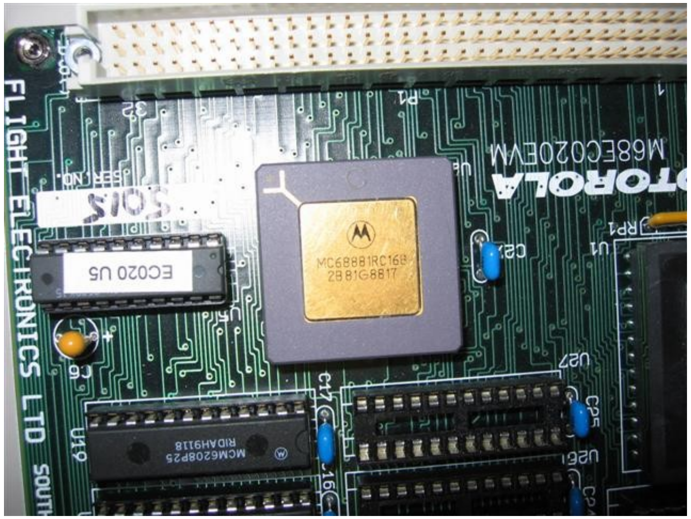


FIGURE 10.33 A four-digit multiplexed display

Motorola 68000 μP – Peripheral chips

68881 Math co-processor

- Similar to 8087
- 8 Internal 80-bit floating point registers
- 40 floating point instructions
- 32-bit data bus
  - Optimally used with 68020 (32-bit bus)
- A_0 and $\overline{\text{SIZE}}$ are used to configure the 68881 for the size of the μP's data bus.



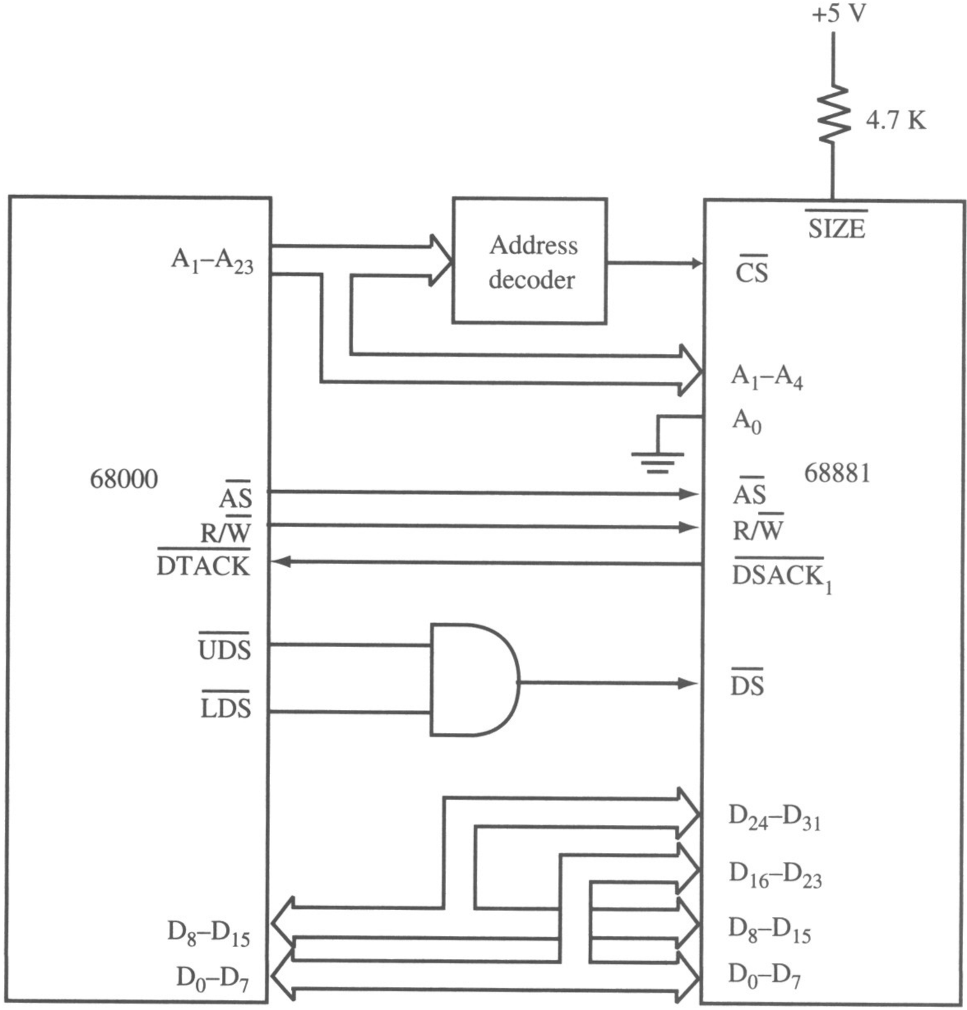


FIGURE 10.39 Floating-point coprocessor connections to the 68000

Motorola 68000 µP – Peripheral chips

INTEL 8279 Keyboard/Display Chip

- For non-68000 series chips, must generate $\overline{\text{DTACK}}$ signal using interface circuitry.
- Must also create separate $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals from joint R/$\overline{\text{W}}$

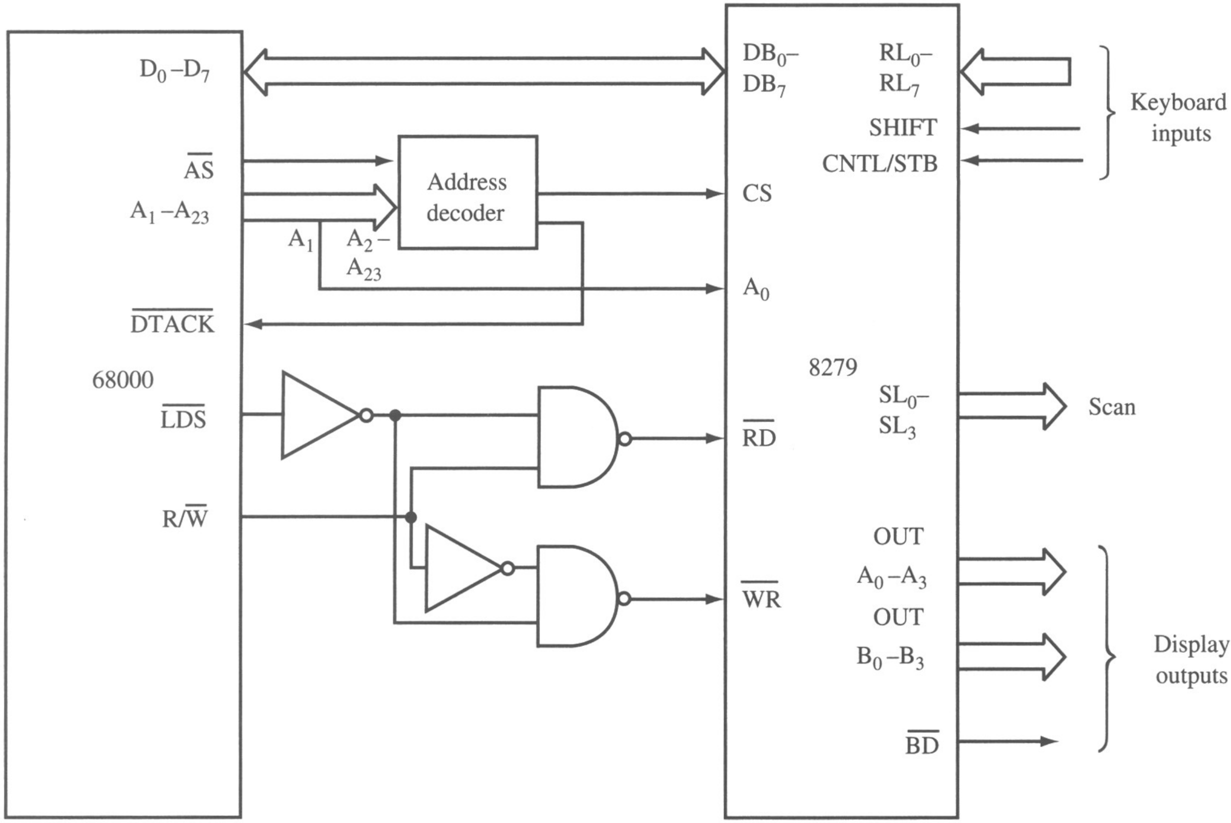


FIGURE 10.41 Interfacing the 8279 to the 68000

Other peripheral chips:

68153 BUS Interrupt Module
68440 Dual DMA Controller
6851 Memory Management Unit
68901 Multifunction Peripheral
68465 Floppy Disk Controller
68452 Bus Arbitration Module
68590 LAN Controller for Ethernet
68652 Multiprotocol Communications Controller
68824 Token-Passing Bus Controller
68486/68487 Raster Memory System
68184 Broadband Interface Controller