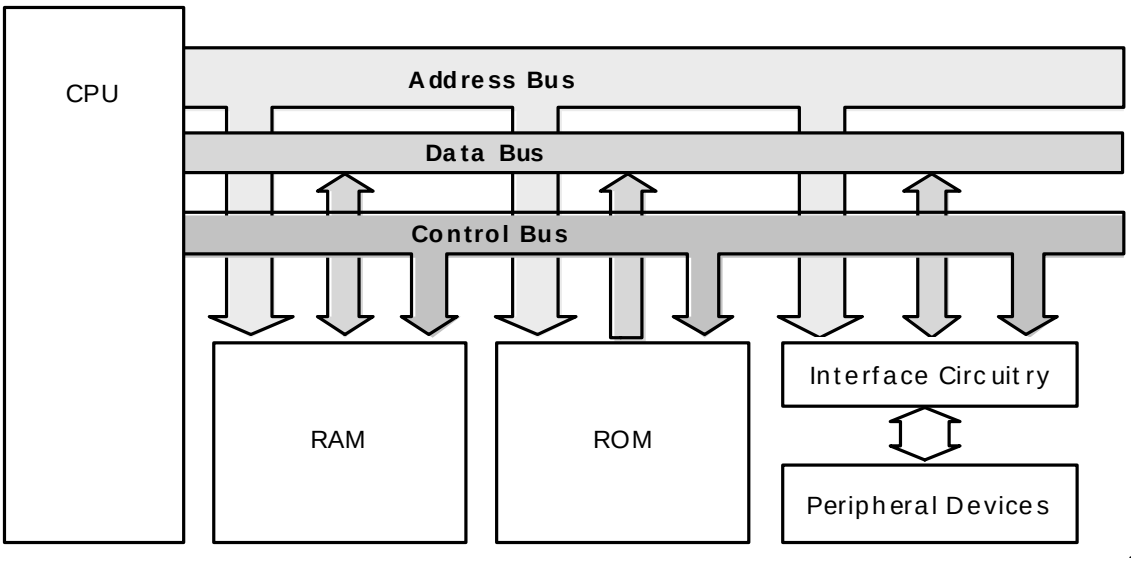
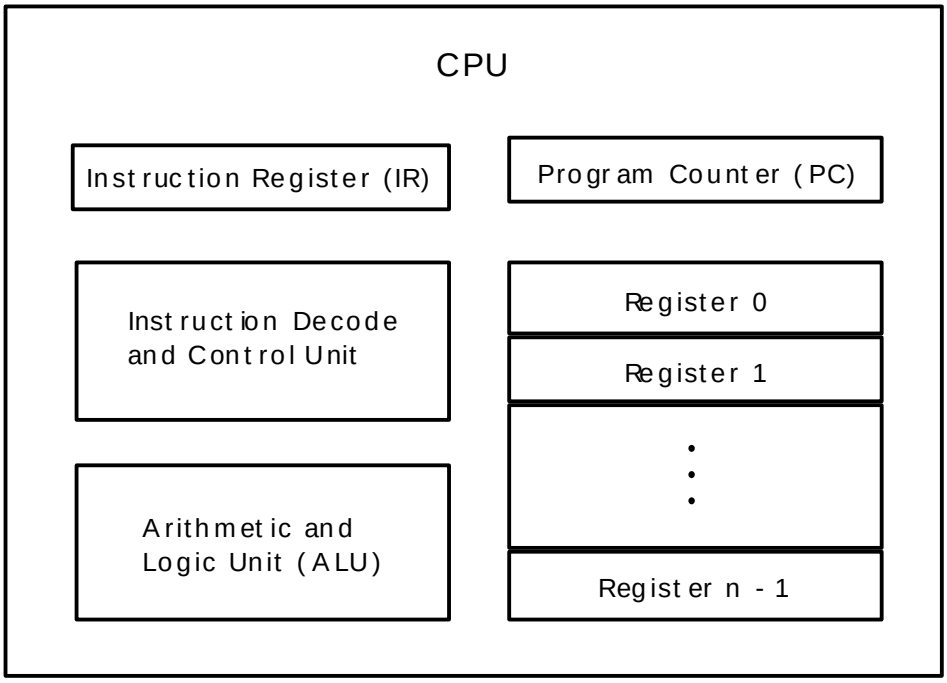
Microcomputer Block Diagram



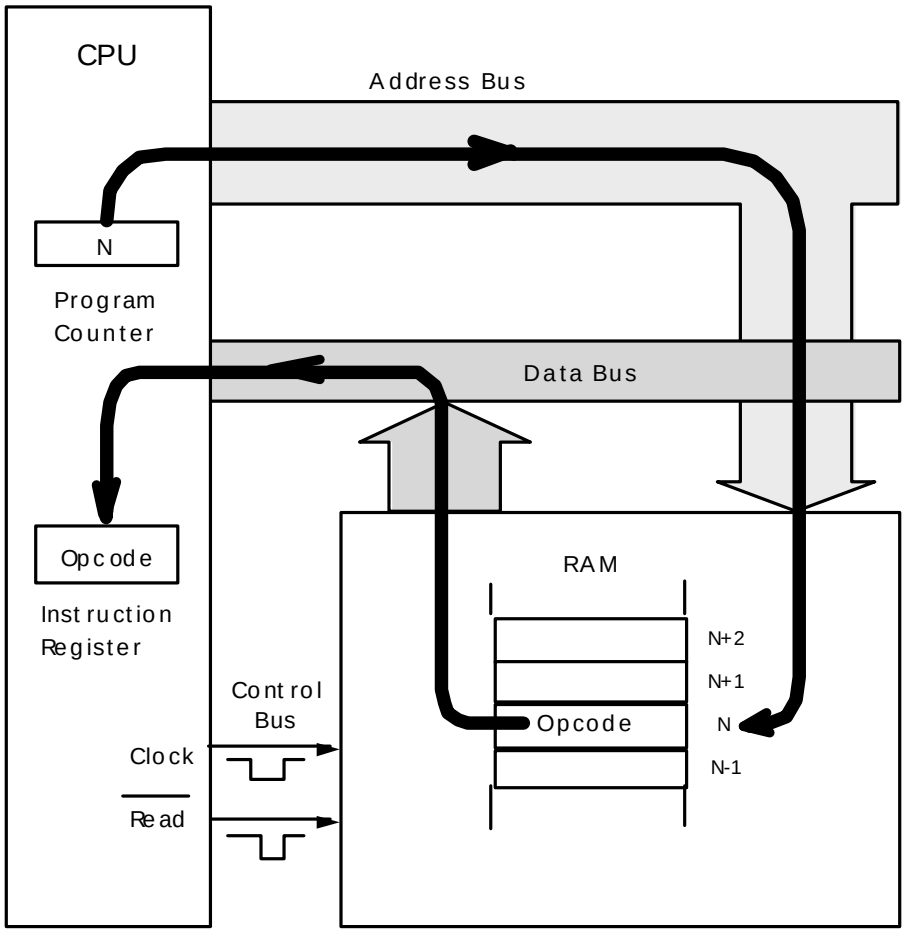
F1-1

CPU Functional Units



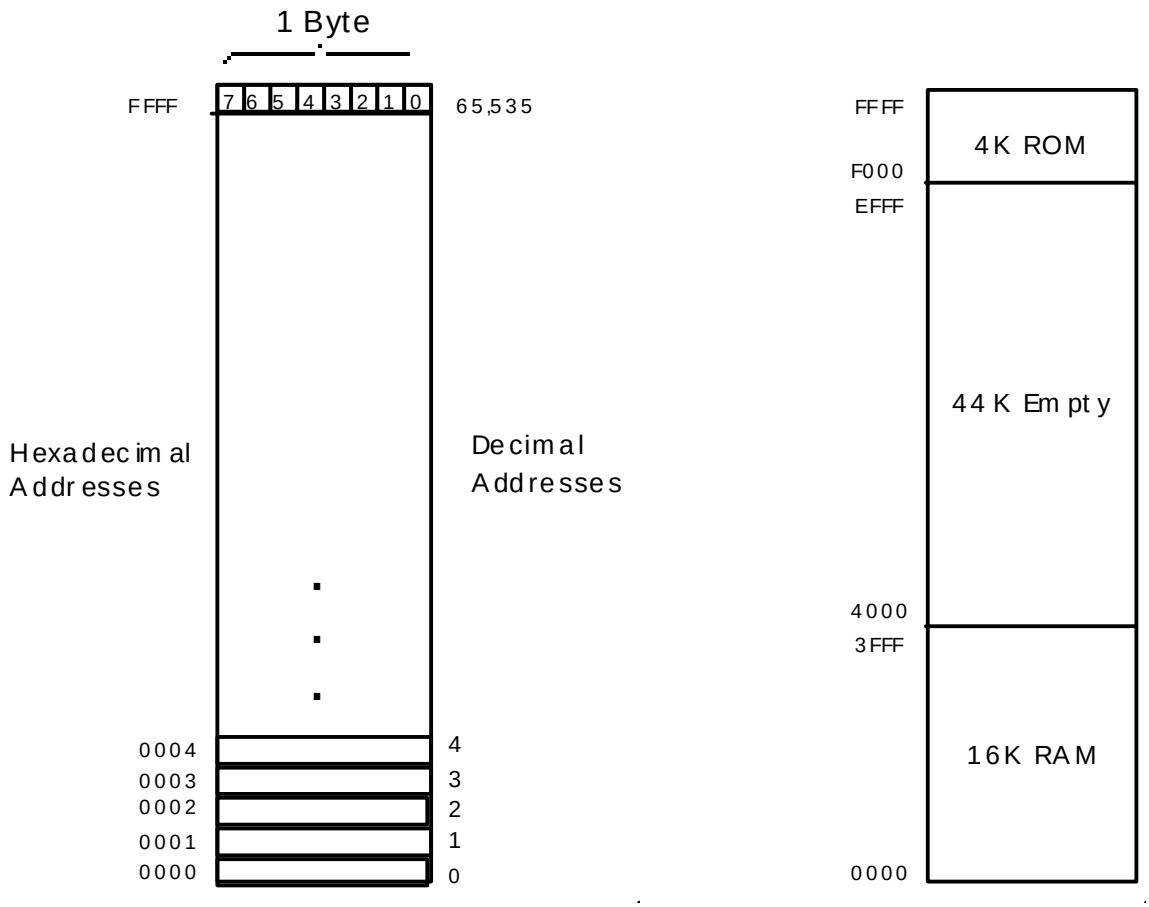
F1-2

Opcode Fetch



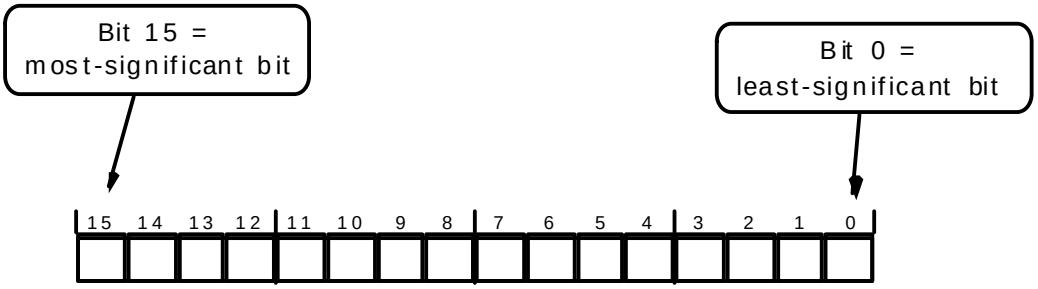
F1-3

Memory Maps

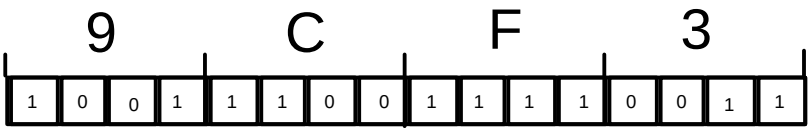


F1-6/7

16-Bit Addresses



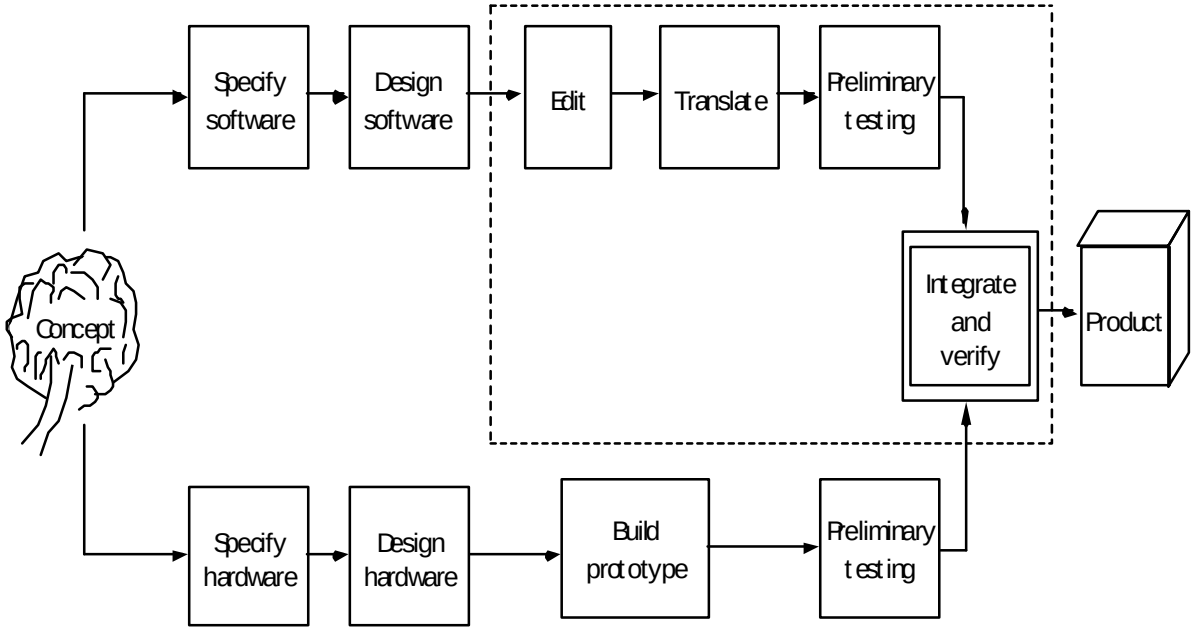
(a)



(b)

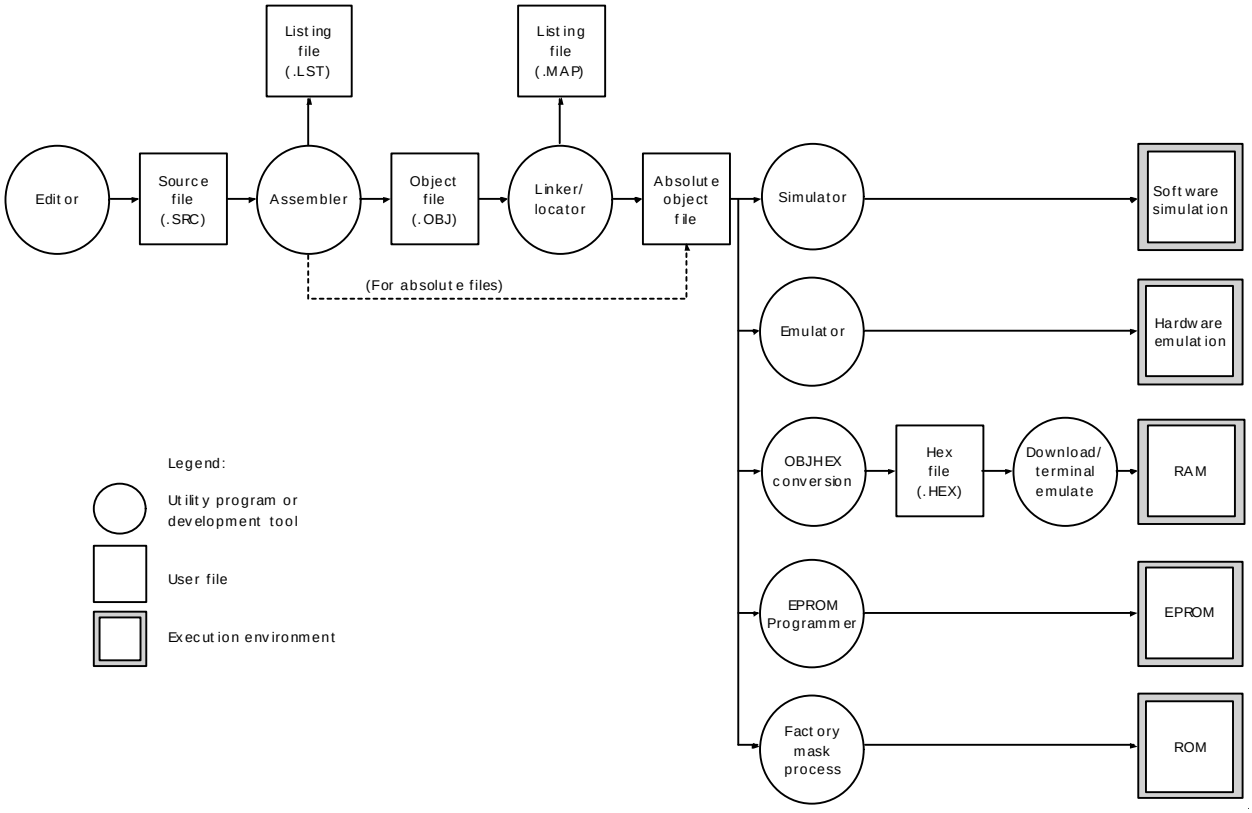
F1-8

The Development Cycle



1-11

Steps in the Development Cycle

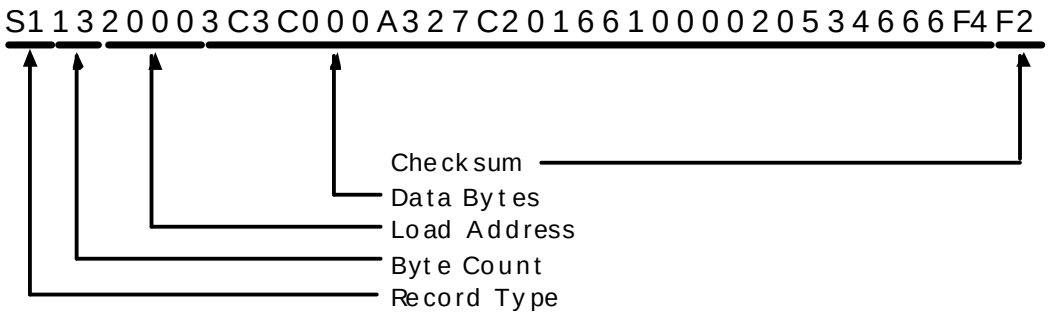


1-13

Motorola S-records

```
S00900006D796E616D656F
S11320003C3C000A327C20166100020534666F4F2
S11320101E3C00E44E4E53636F7474204D61634B59
S10C2020656E7A6965200D0A0061
S113202A1E3C00F81019670000064E4E60F64E7505
S9030000FC
```

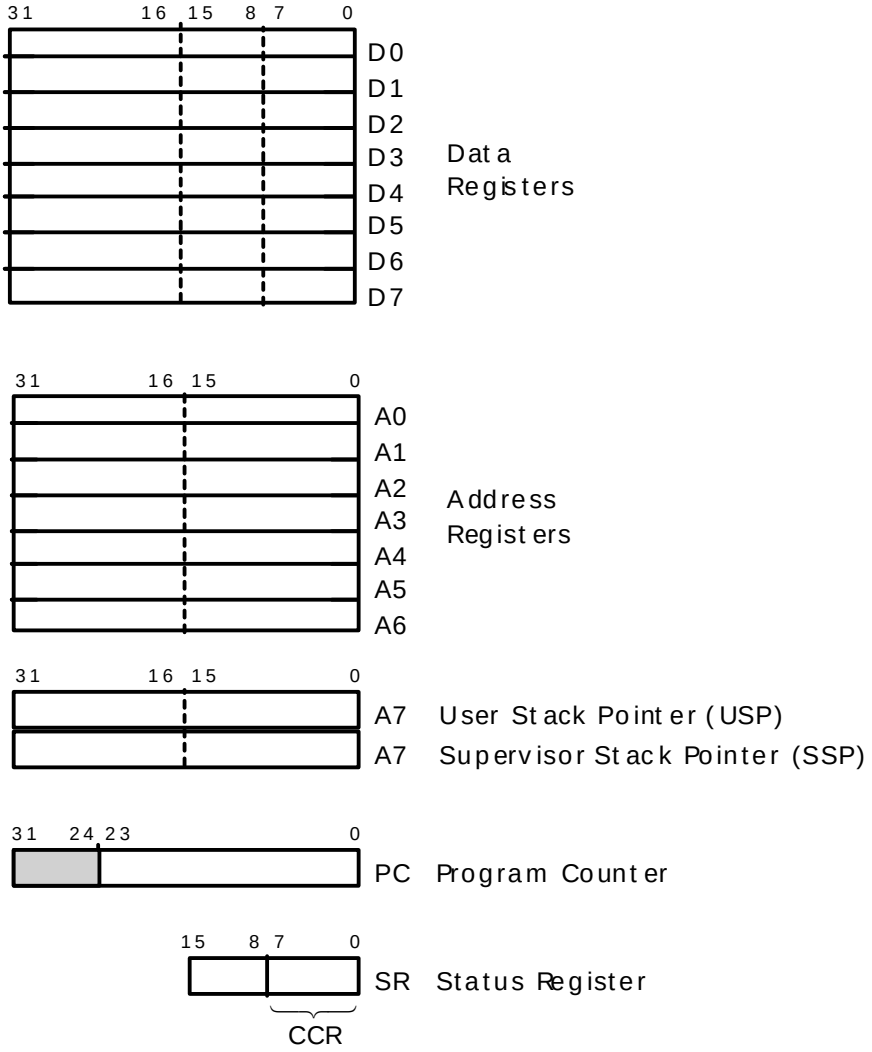
(a)



(b)

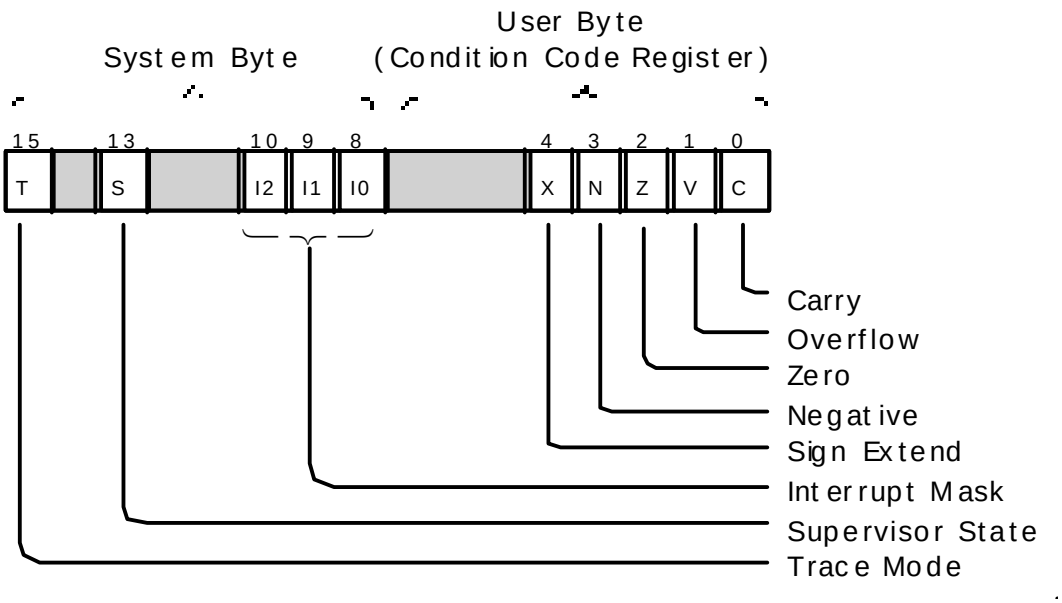
1-14

68000 Programmer's Model



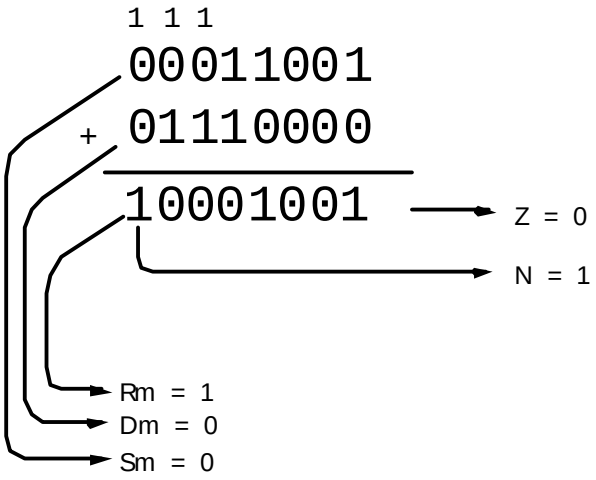
F2-1

68000 Status Register



F2-4

Condition Code Computation



$C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$

$= 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0$

$= 0$

$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$

$= 0 \cdot 0 \cdot 0 + 1 \cdot 1 \cdot 1$

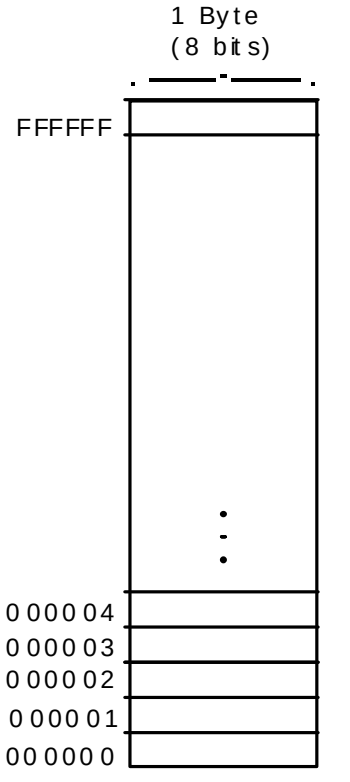
$= 1$

$X = C = 0$

F2-5

68000 Memory Map

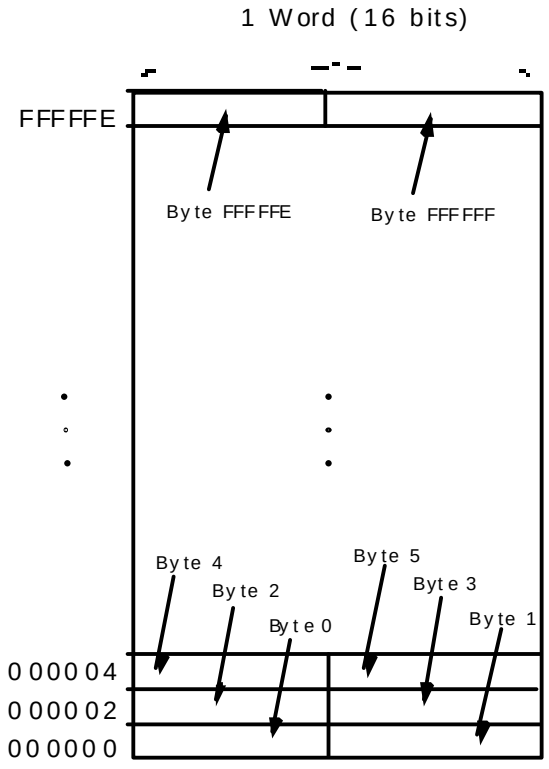
-- Byte View --



F2-7

68000 Memory Map

-- Word View --

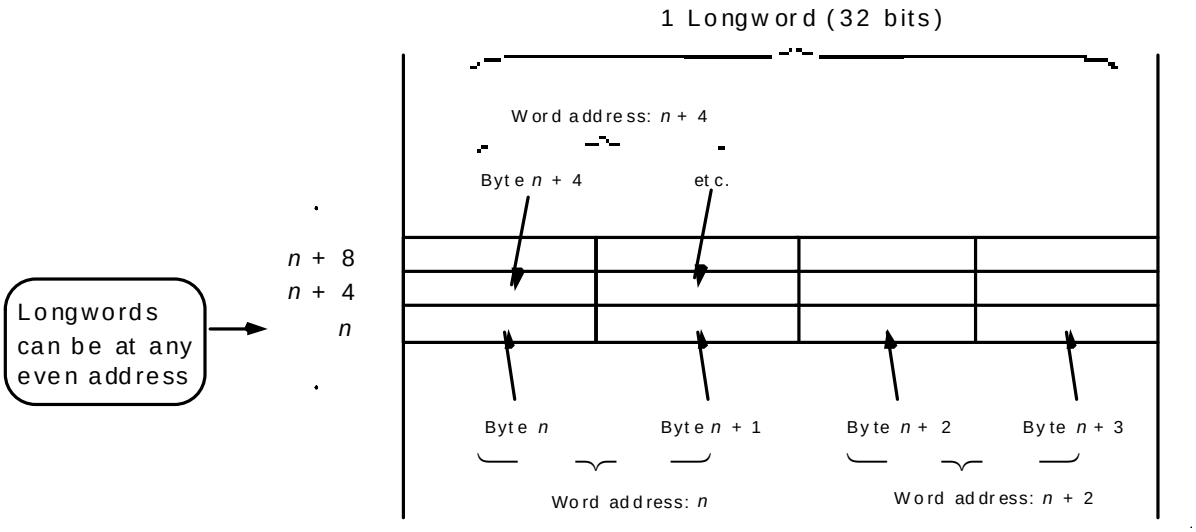


Note:
Even bytes correspond to *upper* bytes on the external data bus. *Odd* bytes correspond to *lower* bytes on the external data bus.

F2-8

68000 Memory Map

-- Longword View --



F2-9

68000 Addressing Modes

Mode	Assembler Syntax	Effective Address Generation
Data Register Direct	Dn	EA = Dn
Address Register Direct	An	EA = An
Absolute Short	xxx.W or <xxx	EA = (next word)
Absolute Long	xxx.L or >xxx	EA = (next two words)
Register Indirect	(An)	EA = (An)
Postincrement Register Indirect	(An)+	EA = (An), An ` An + N
Predecrement Register Indirect	-(An)	An ` An - N, EA = (An)
Register Indirect with Offset	d16(An)	EA = (An) + d16
Register Indirect with Index & Offset	d8(An,Xn)	EA = (An) + (Xn) + d8
PC Relative with Offset	d16(PC)	EA = (PC) + d16
PC Relative with Index and Offset	d8(PC,Xn)	EA = (PC) + (Xn) + d8
Immediate	#data	DATA = next word(s)
Implied Register	CCR, SR, USP, SSP, PC	EA = CCR, SR, USP, SSP, PC

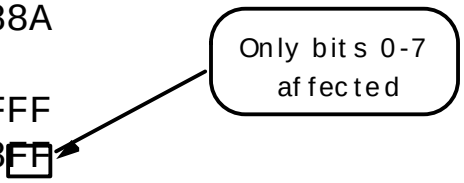
Notes:

EA = effective address
An = address register
Dn = data register
Xn = address or data register used as index register
CCR = condition code register
SR = status register
USP = user stack pointer
SSP = supervisor stack pointer
PC = program counter
( ) = contents of
d8 = 8-bit offset (displacement)
d16 = 16-bit offset (displacement)
N = 1 for byte, 2 for word, 4 for longword. (If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.)
` = is replaced by

T2-1

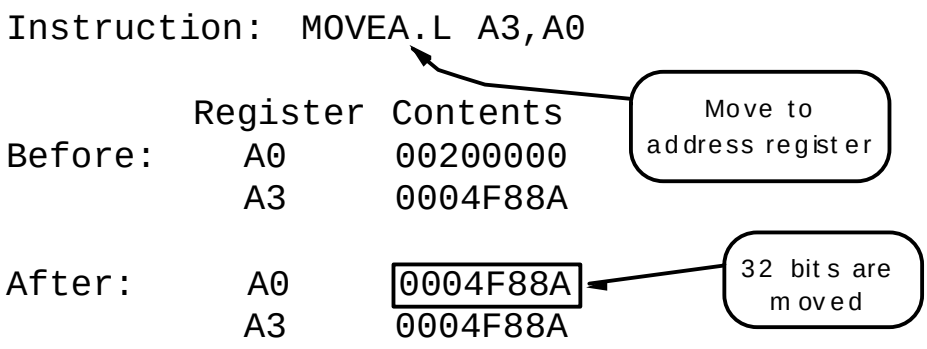
Data Register Direct

```
Instruction:  MOVE.B D0,D3

          Register  Contents
Before:      D0      10204FFF
             D3      1034F88A

After:       D0      10204FFF
             D3      1034F8FF
```



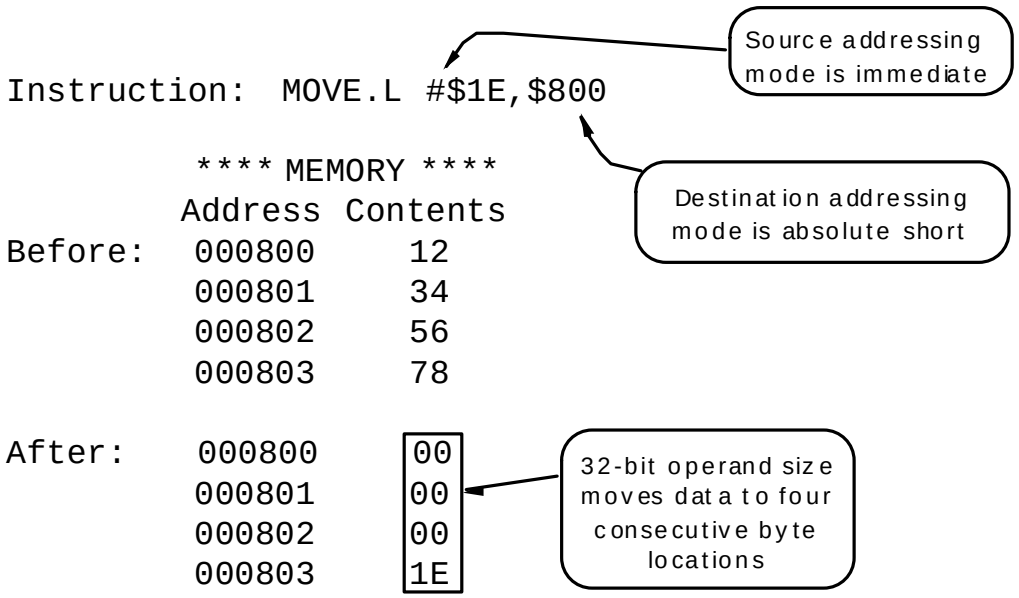
F2-10

Address Register Direct



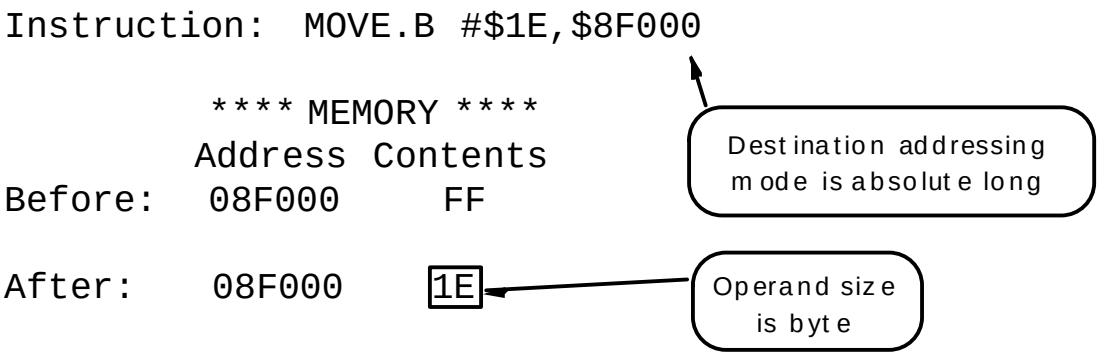
F2-11

Absolute Short



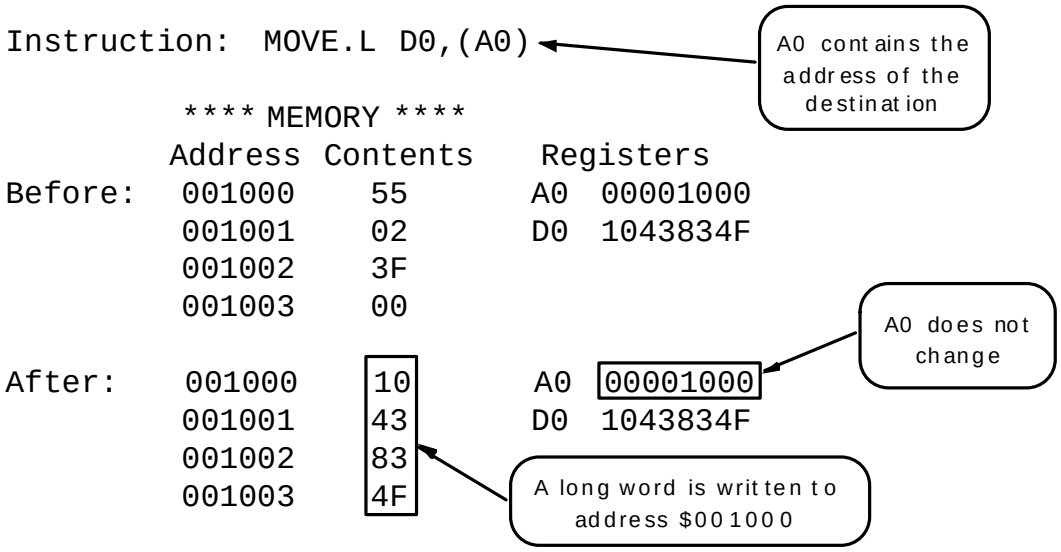
F2-13

Absolute Long



F2-14

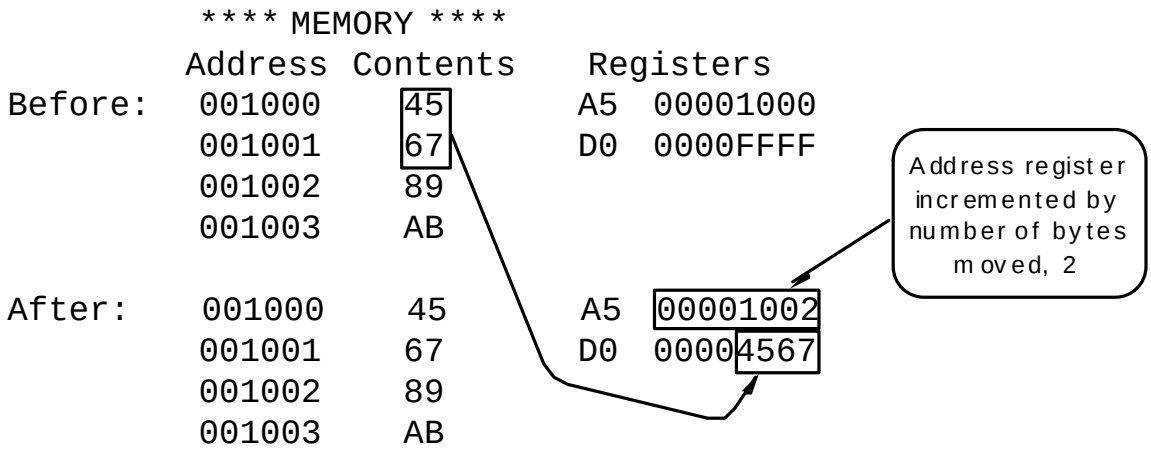
Register Indirect



F2-17

Postincrement Address Register Indirect

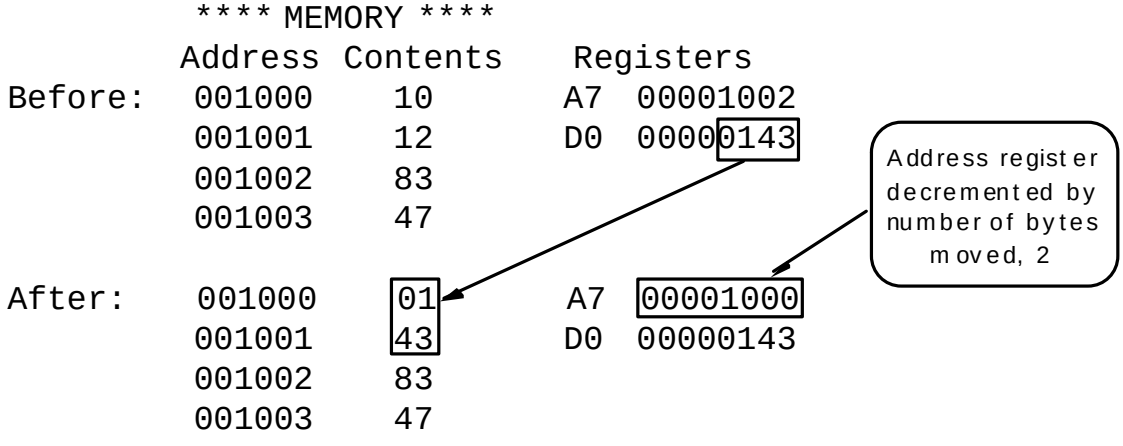
Instruction: MOVE.W (A5)+,D0



F2-18

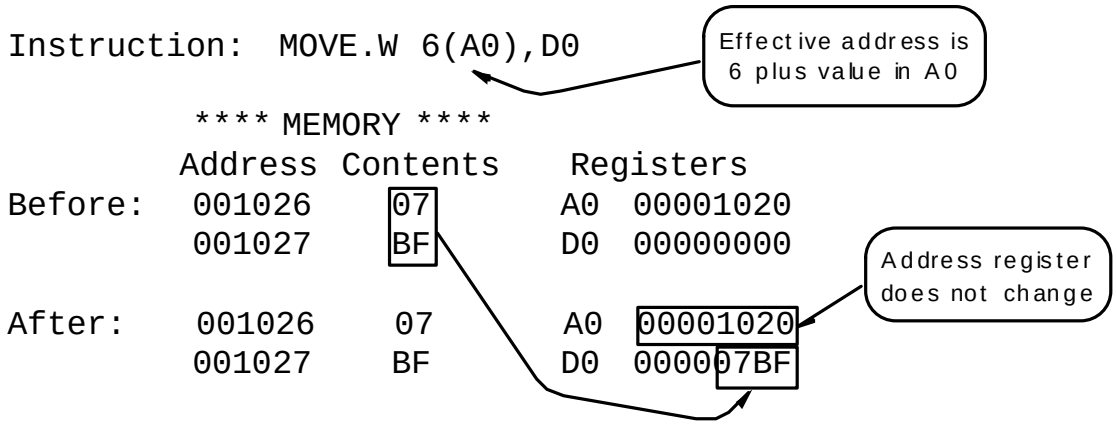
Predecrement Address
Register Indirect

Instruction: MOVE.W D0,-(A7)



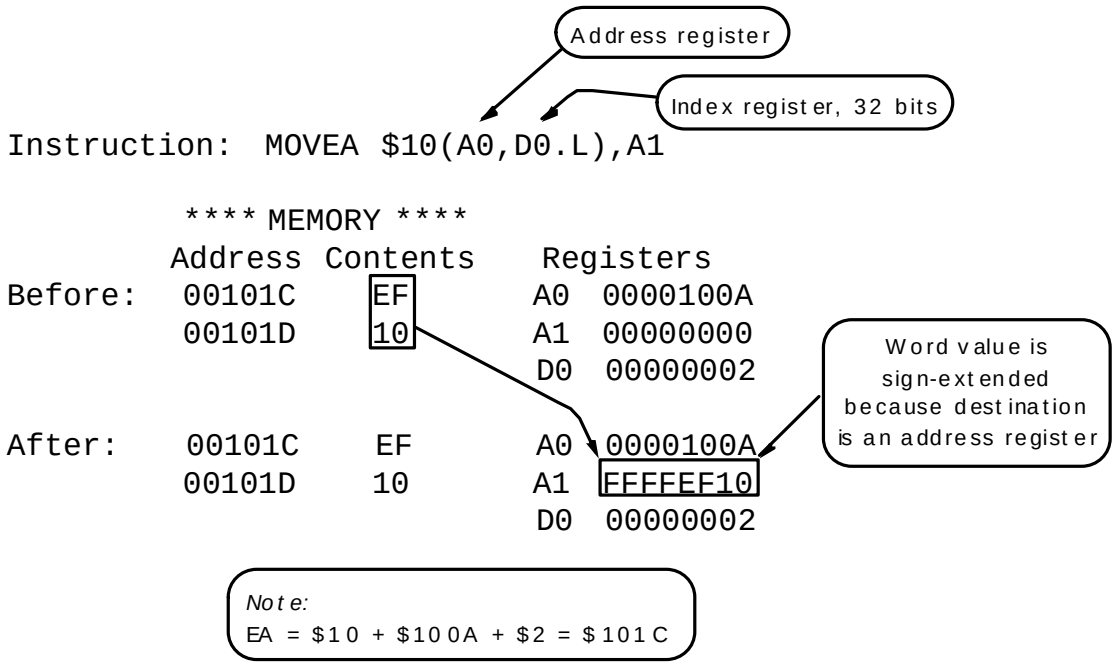
F2-19

Register Indirect With Offset



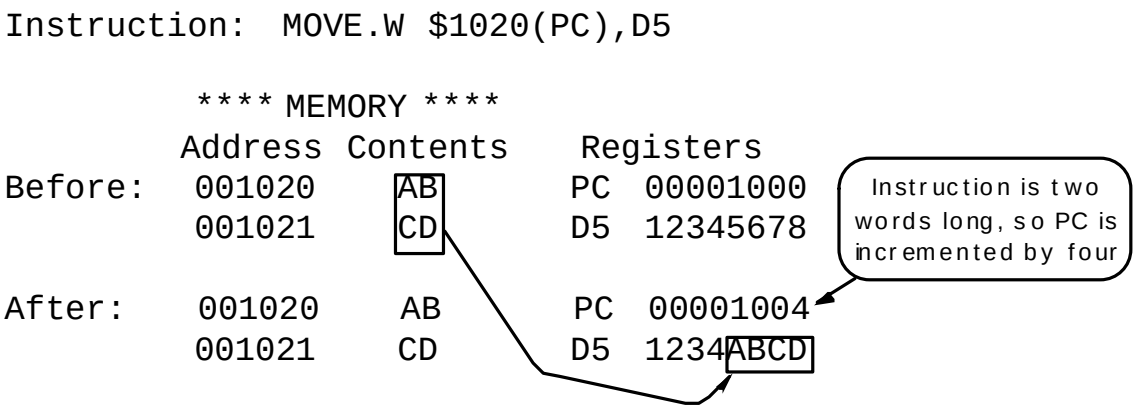
F2-20

Register Indirect
With Index and Offset



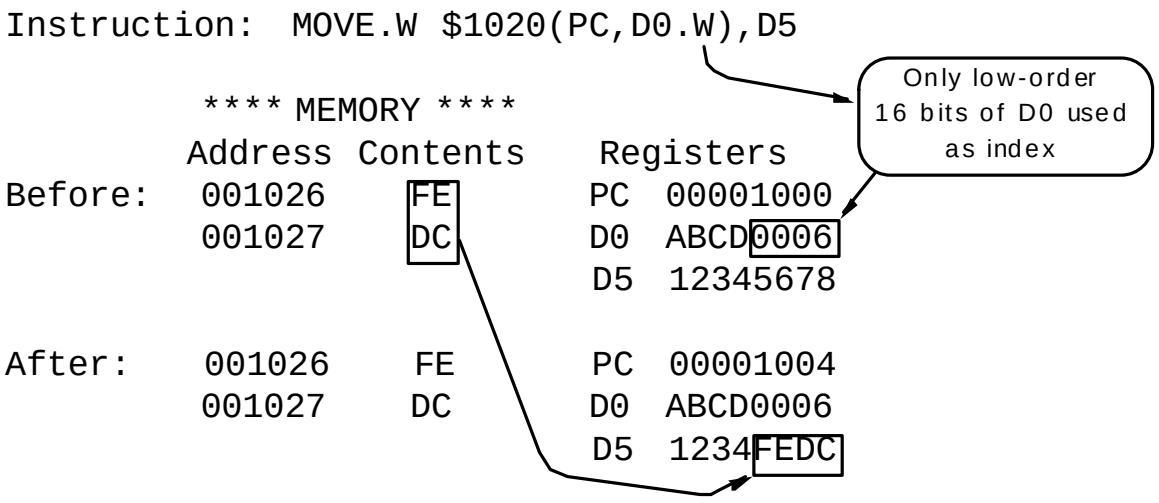
F2-22

PC-Relative
With Offset



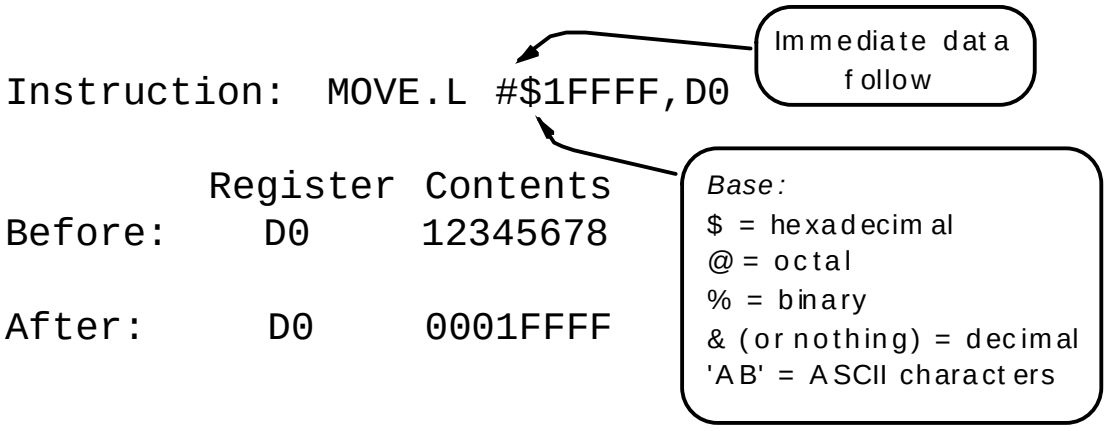
F2-23

PC-Relative
With Index and Offset



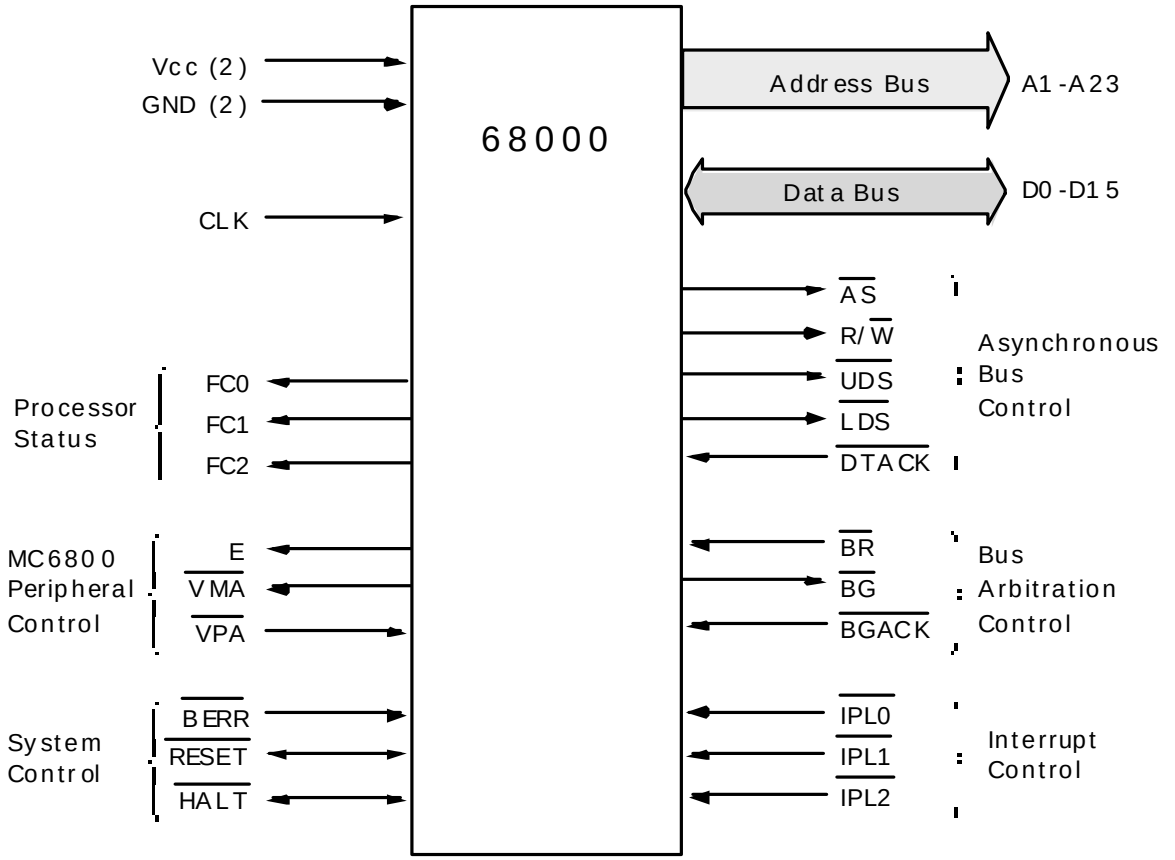
F2-26

Immediate



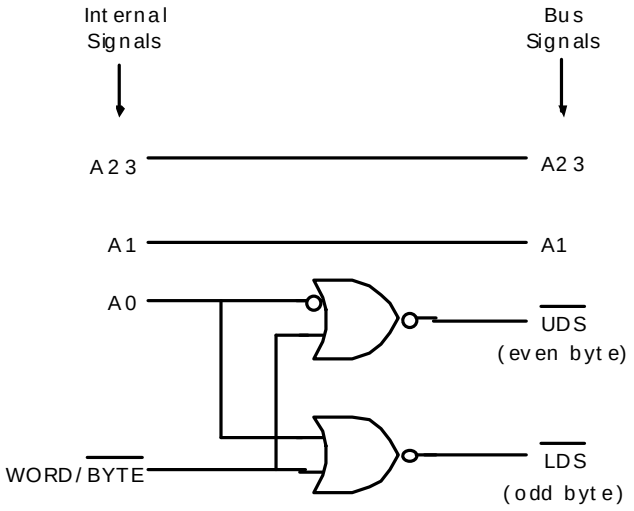
F2-28

68000 Signals



F2-33

Upper Data Strobe and Lower Data Strobe



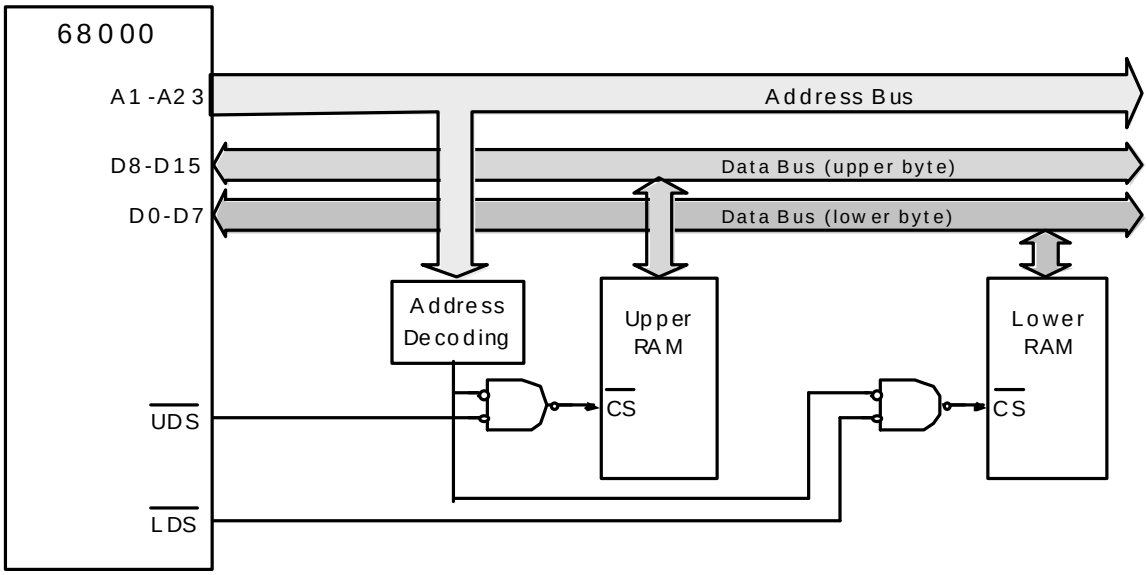
(a)

WORD/$\overline{\text{BYTE}}$	A0	$\overline{\text{UDS}}$	$\overline{\text{LDS}}$
1	X	0	0
0	0	0	1
0	1	1	0

(b)

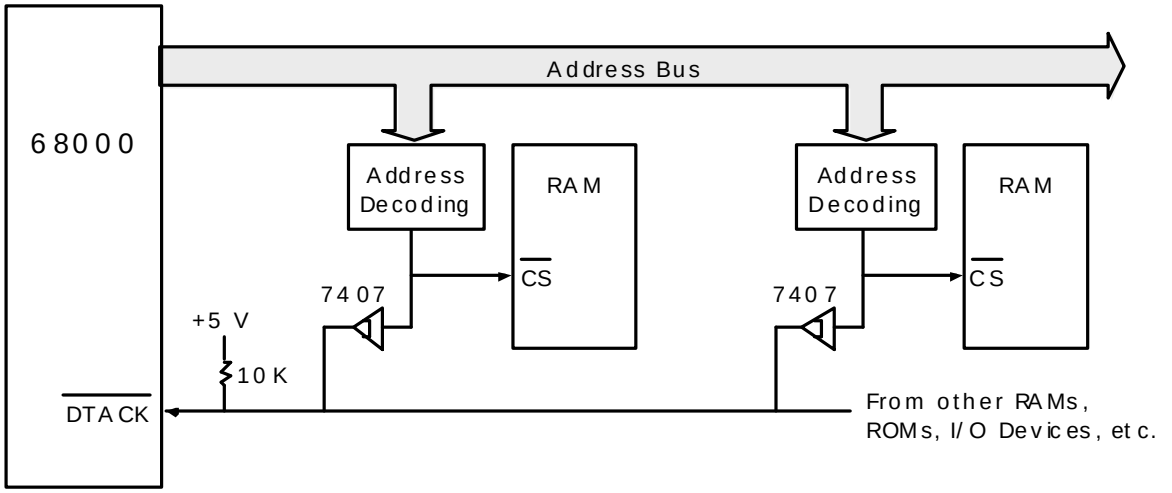
F2-34

Decoding with -UDS and -LDS



F2-36

Generation of -DTACK



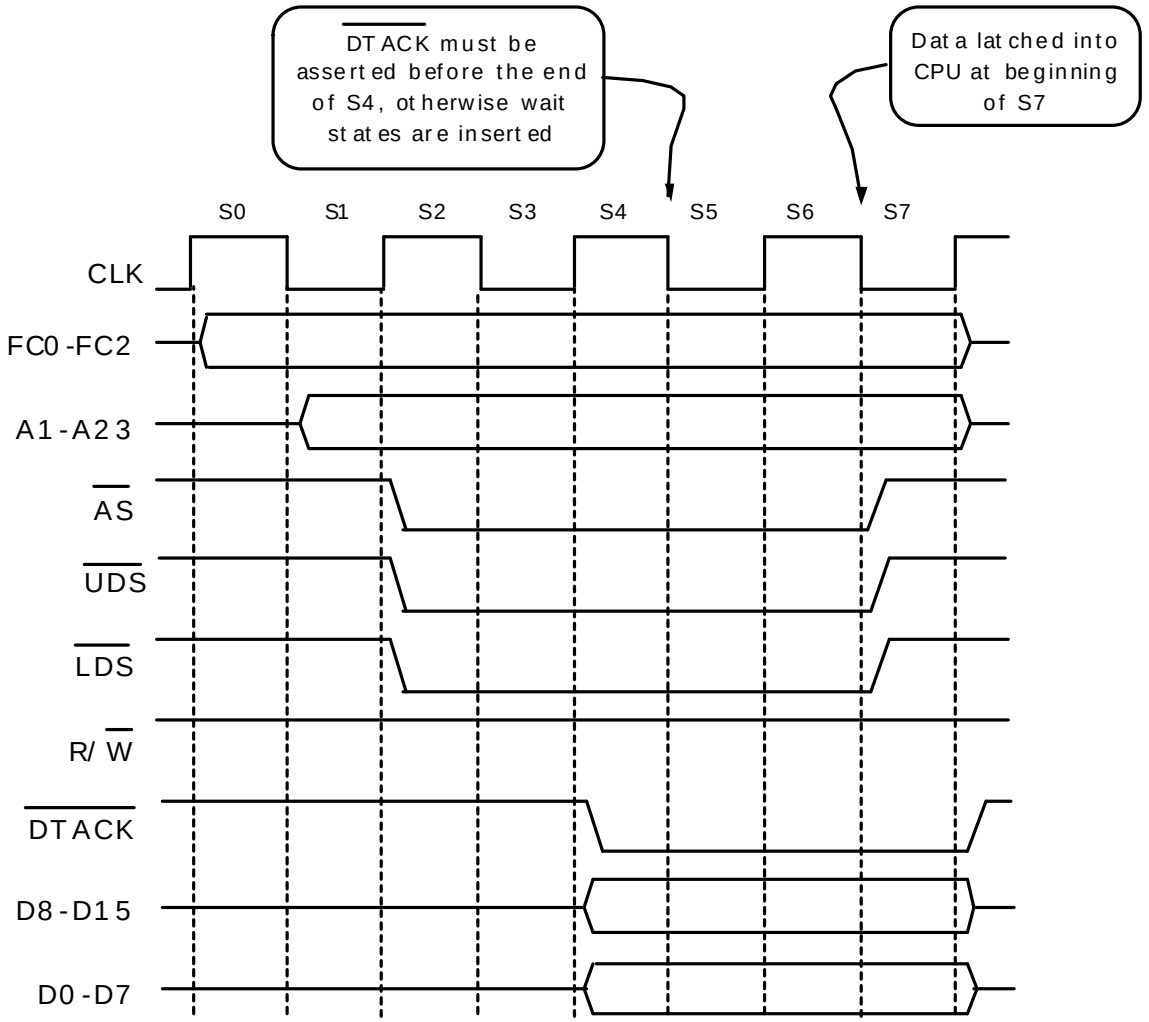
F2-36

Function Code Outputs

Function Code			Address Space Type
FC2	FC1	FC0	
0	0	0	(Undefined, reserved)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Undefined, reserved)
1	0	0	(Undefined, reserved)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	CPU Space (Interrupt Acknowledge)

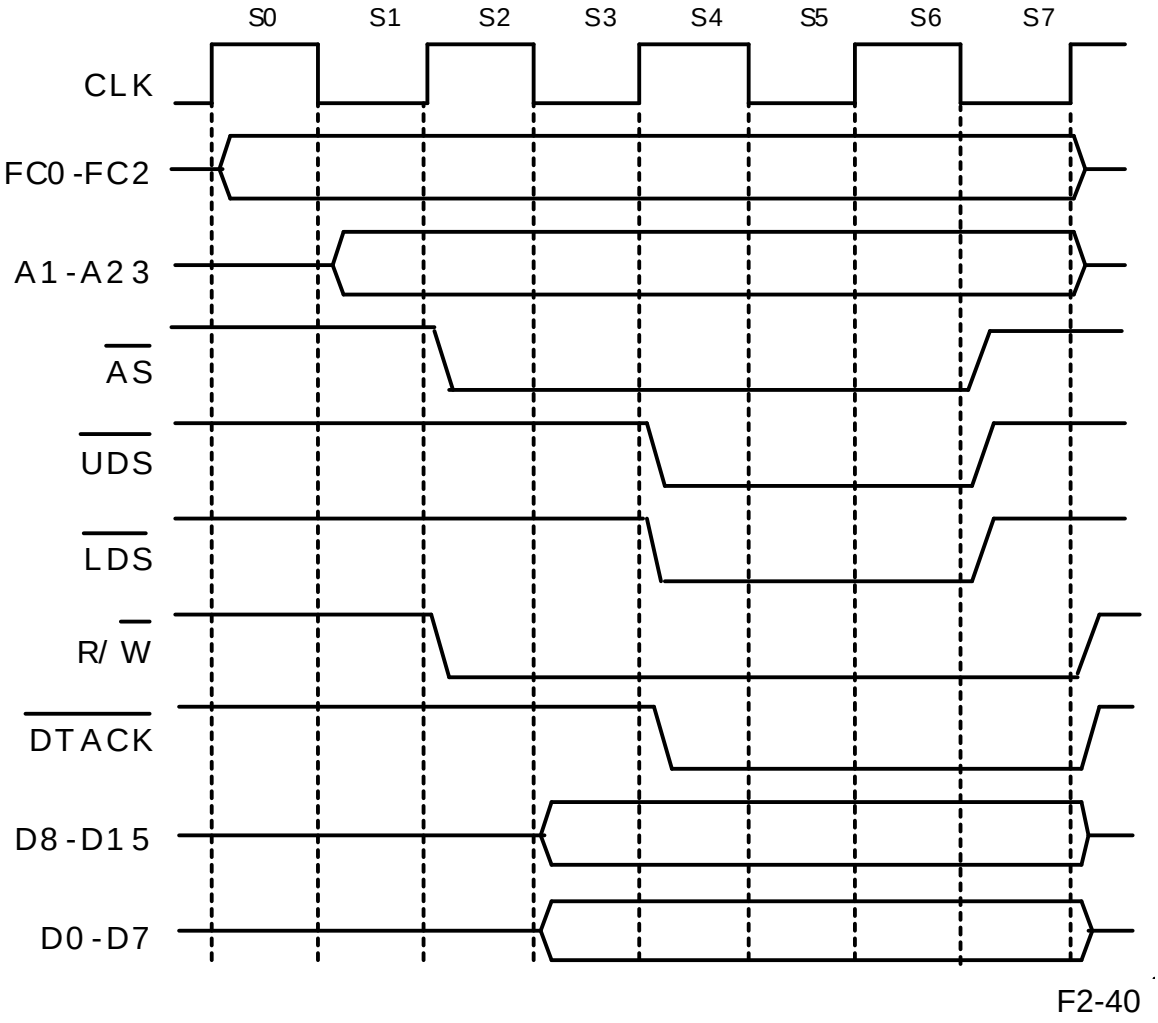
T2-3

Read Cycle Timing



F2-39

Write Cycle Timing



Data Movement Instructions

Instruction	Operation
EXG	Exchange registers
LEA	Load effective address
LINK	Link and allocate stack
MOVE	Move source to destination
MOVEA	Move source to address register
MOVEM	Move multiple registers
MOVEP	Move to peripheral
MOVEQ	Move short data to destination
PEA	Push effective address
UNLK	Unlink stack

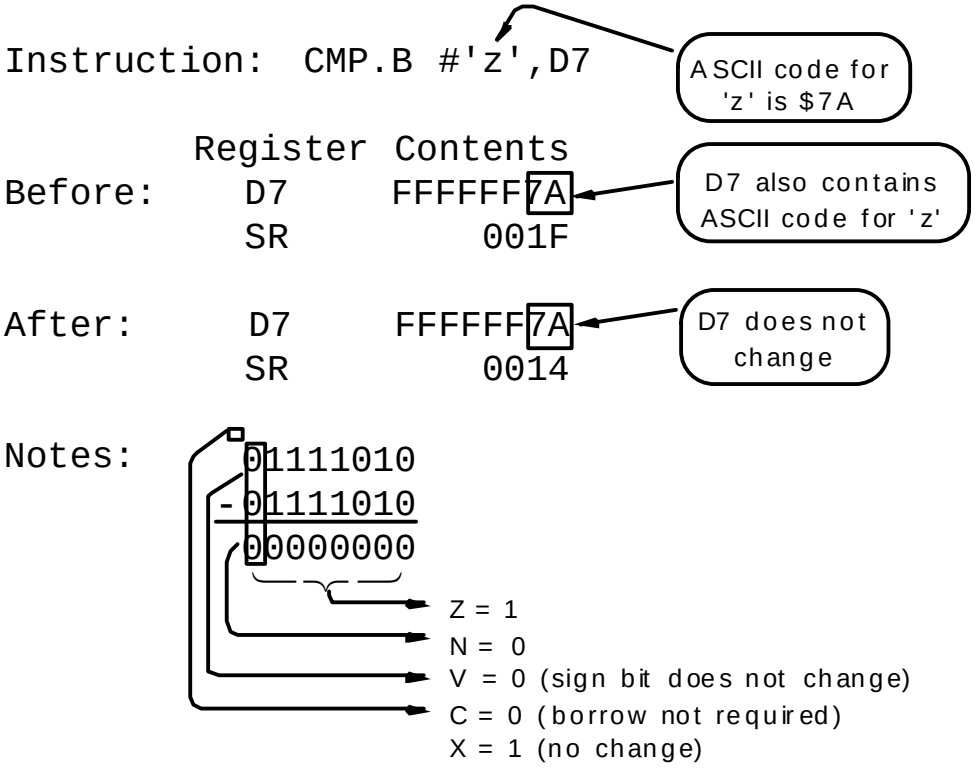
T3-3

Integer Arithmetic Instructions

Instruction	Operation
ADD	Add source to destination
ADDA	Add source to address register
ADDI	Add immediate data to destination
ADDQ	Add short data to destination
ADDX	Add with extend bit to destination
CLR	Clear operand
CMP	Compare source to destination
CMPA	Compare source to address register
CMPM	Compare memory
DIVS	Signed divide
DIVU	Unsigned divide
EXT	Sign extend
EXTB	Sign extend byte
MULS	Signed multiply
MULU	Unsigned multiply
NEG	Negate
NEGX	Negate with extend
SUB	Subtract source from destination
SUBA	Subtract source from address register
SUBI	Subtract immediate from destination
SUBQ	Subtract short from destination
SUBX	Subtract with extend bit from destination

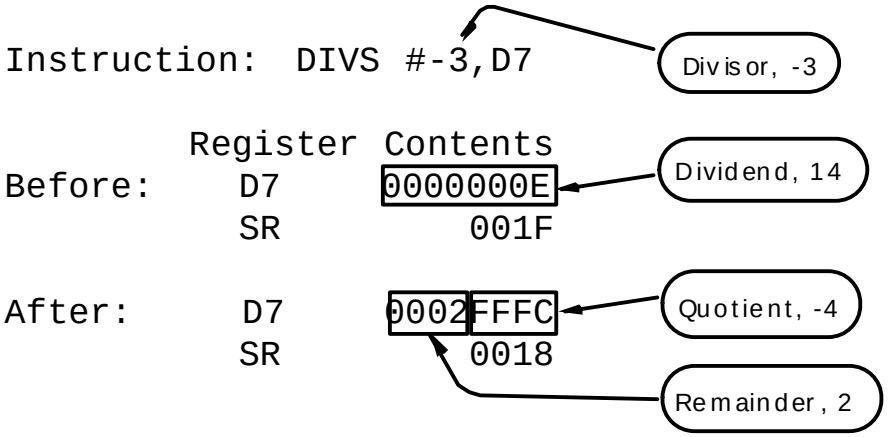
T3-4

CMP Example



F3-5

DIVS Example



Notes: 14 / -3 = -4 with a remainder of 2

F3-8

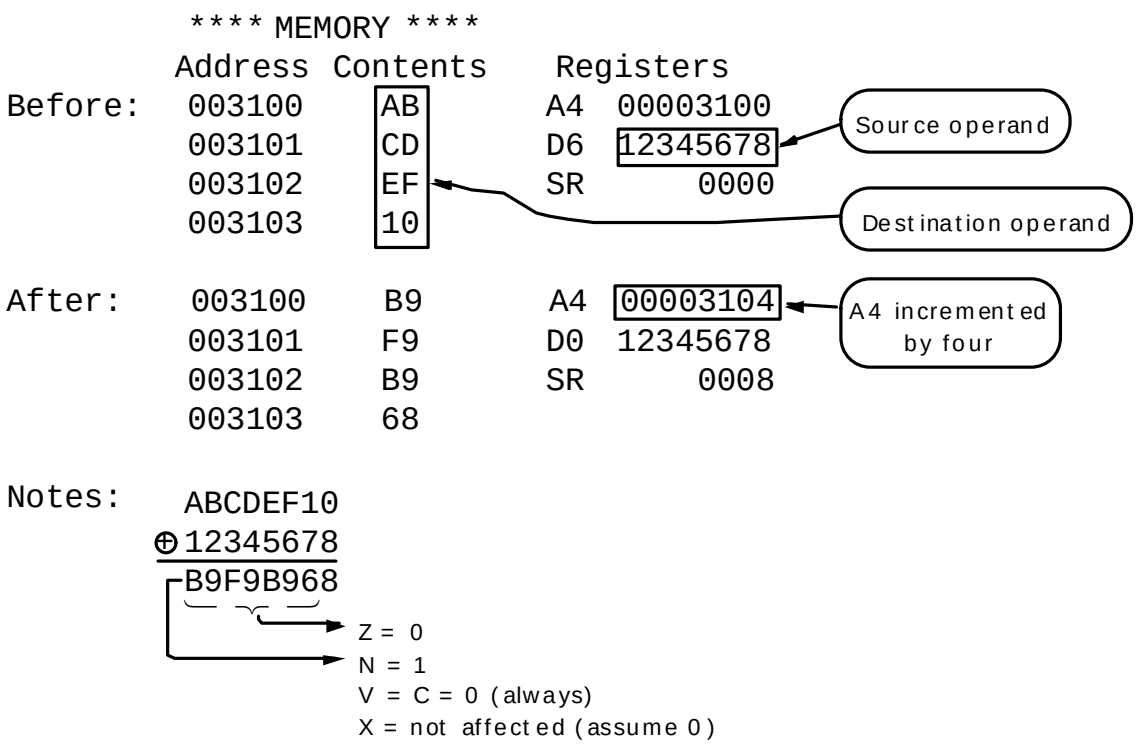
Boolean Instructions

Instruction	Operation
AND	AND source to destination
ANDI	AND immediate data to destination
EOR	Exclusive OR source to destination
EORI	Exclusive OR immediate data to destination
NOT	Complement destination
OR	OR source to destination
ORI	OR immediate data to destination
Scc	Test condition codes and set operand
TST	Test operand and set condition codes

T3-5

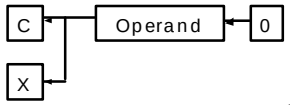
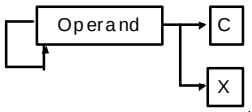
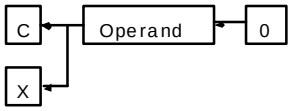
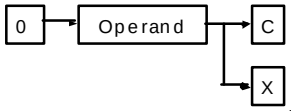
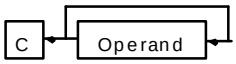
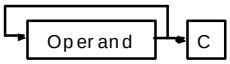
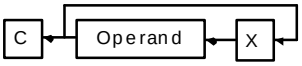
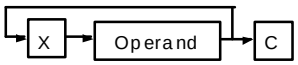
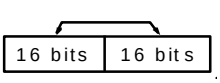
EOR Example

Instruction: EOR.L D6,(A4)+



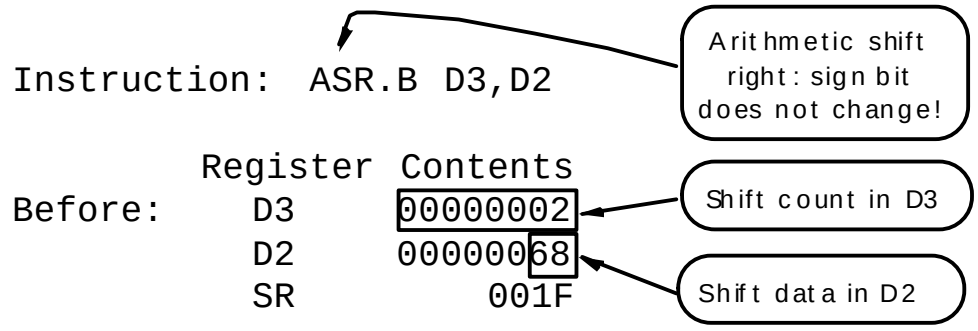
F3-10

Shift and Rotate Instructions

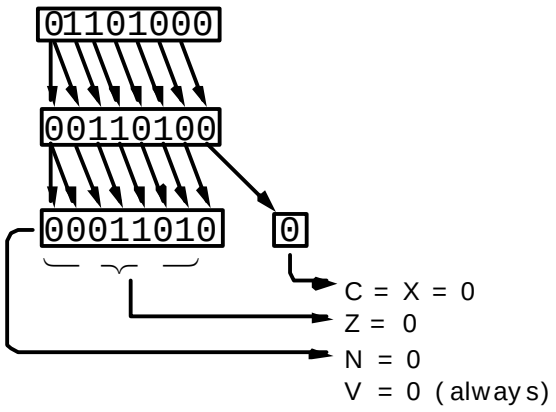
Instruction	Operation	Bit Movement
ASL	Arithmetic shift left	
ASR	Arithmetic shift right	
LSL	Logical shift left	
LSR	Logical shift right	
ROL	Rotate left	
ROR	Rotate right	
ROXL	Rotate left with extend bit	
ROXR	Rotate right with extend bit	
SWAP	Swap words of a longword	

T3-6

ASR Example



Notes:



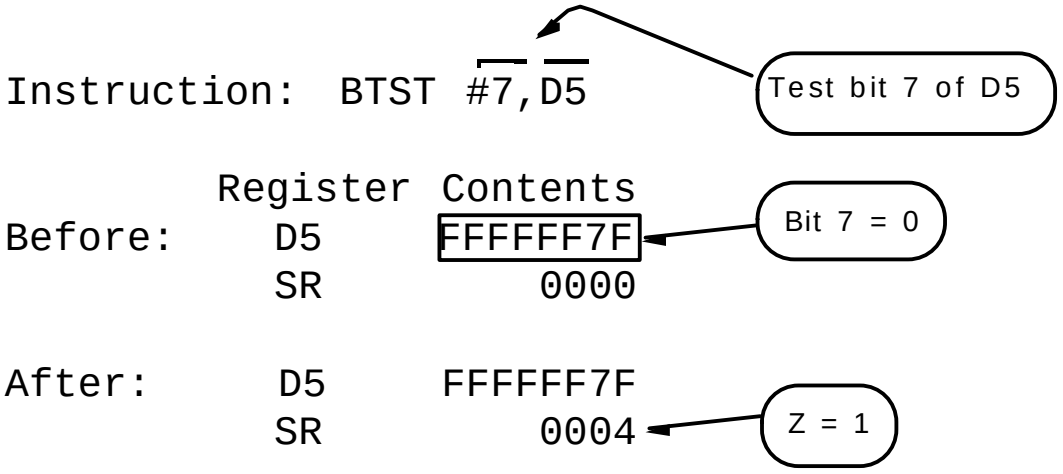
F3-11

Bit Manipulation Instructions

Instruction	Operation
BCHG	Change bit
BCLR	Clear bit
BSET	Set bit
BTST	Test bit

T3-7

BTST Example



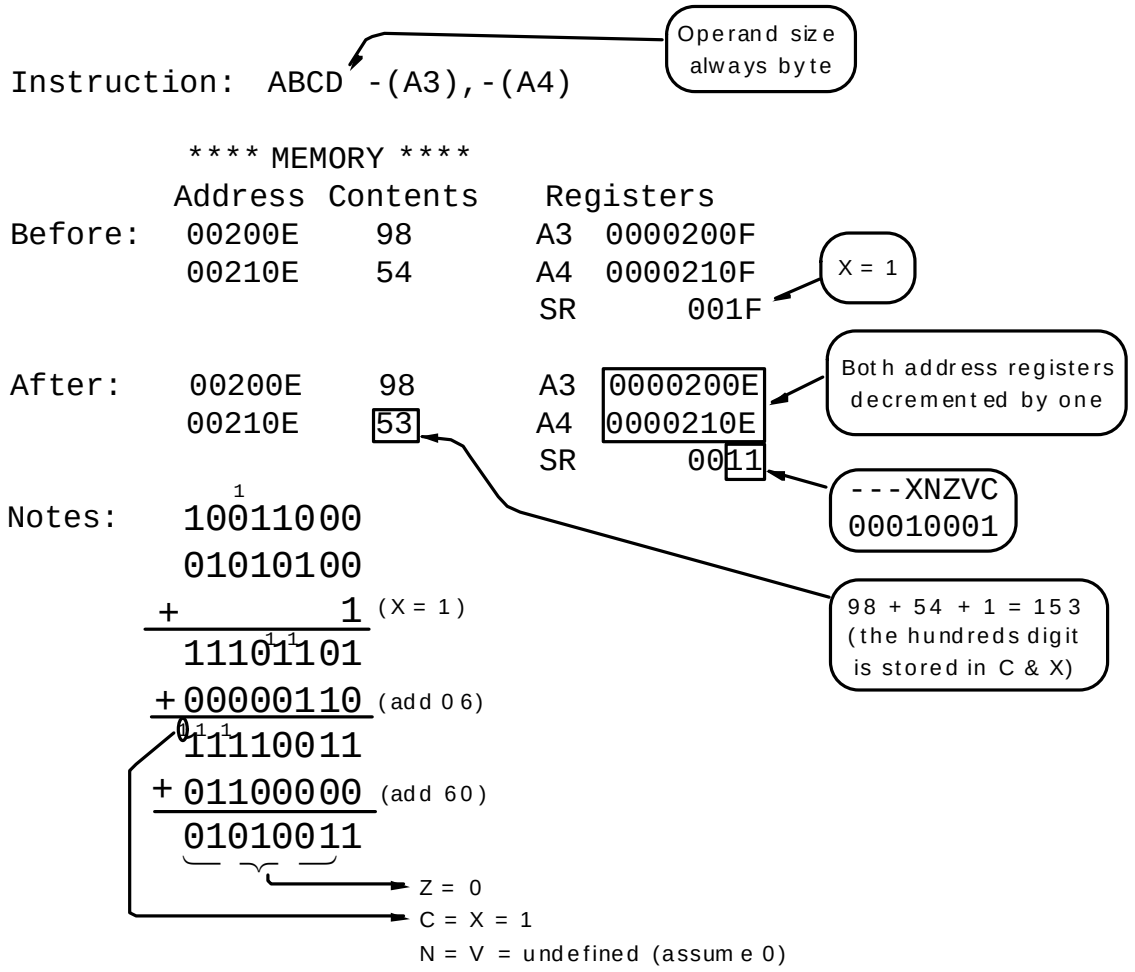
F3-12

Binary-Coded Decimal Instructions

Instruction	Operation
ABCD	Add source to destination
NBCD	Negate destination
SBCD	Subtract source from destination

T3-8

ABCD Example



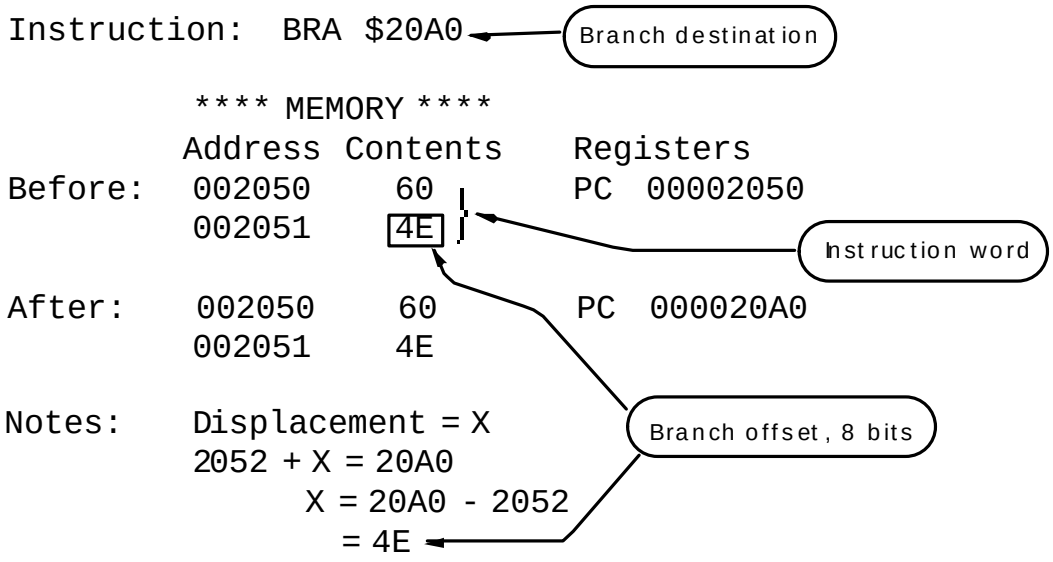
Program Flow Instructions

Instruction	Operation
Bcc	Branch conditionally
BRA	Branch always
BSR	Branch to subroutine
DBcc	Test, decrement, and branch
JMP	Jump to address
JSR	Jump to subroutine
NOP	No operation
RTE+	Return and deallocate stack
RTR	Return and restore condition codes
RTS	Return from subroutine

+privileged instruction

T3-9

BRA Example



F3-15

BSR/RTS Example

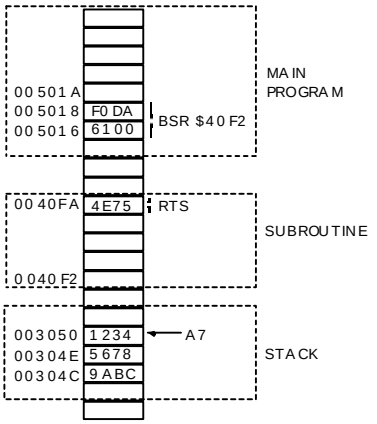
BSR $40F2

BEFORE

REGISTERS:

PC 005016
A7 00003050

MEMORY:

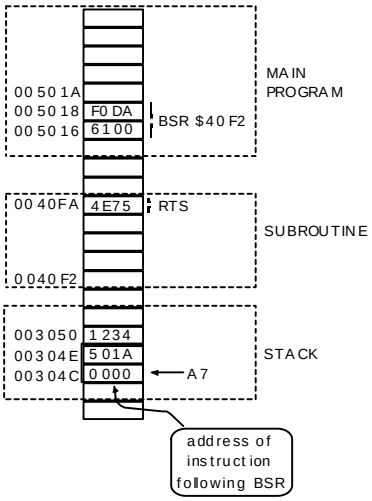


AFTER

REGISTERS:

PC 0040F2
A7 00003040

MEMORY:



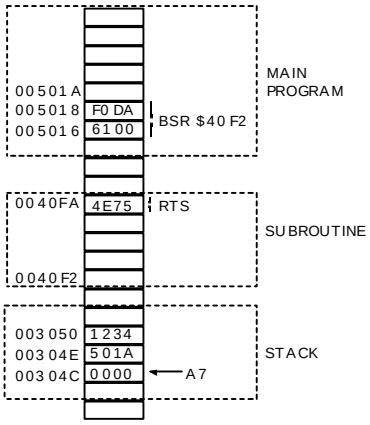
RTS

BEFORE

REGISTERS:

PC 0040FA
A7 00003040

MEMORY:



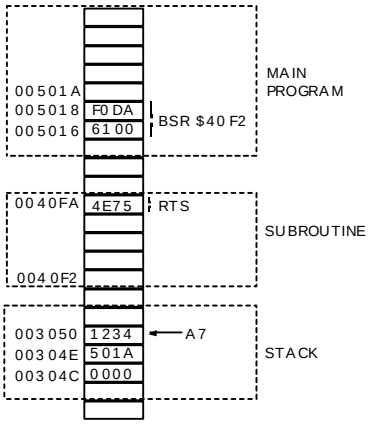
AFTER

REGISTERS:

PC 005018
A7 00003050

return to instruction following BSR

MEMORY:



F3-16

System Control Instructions

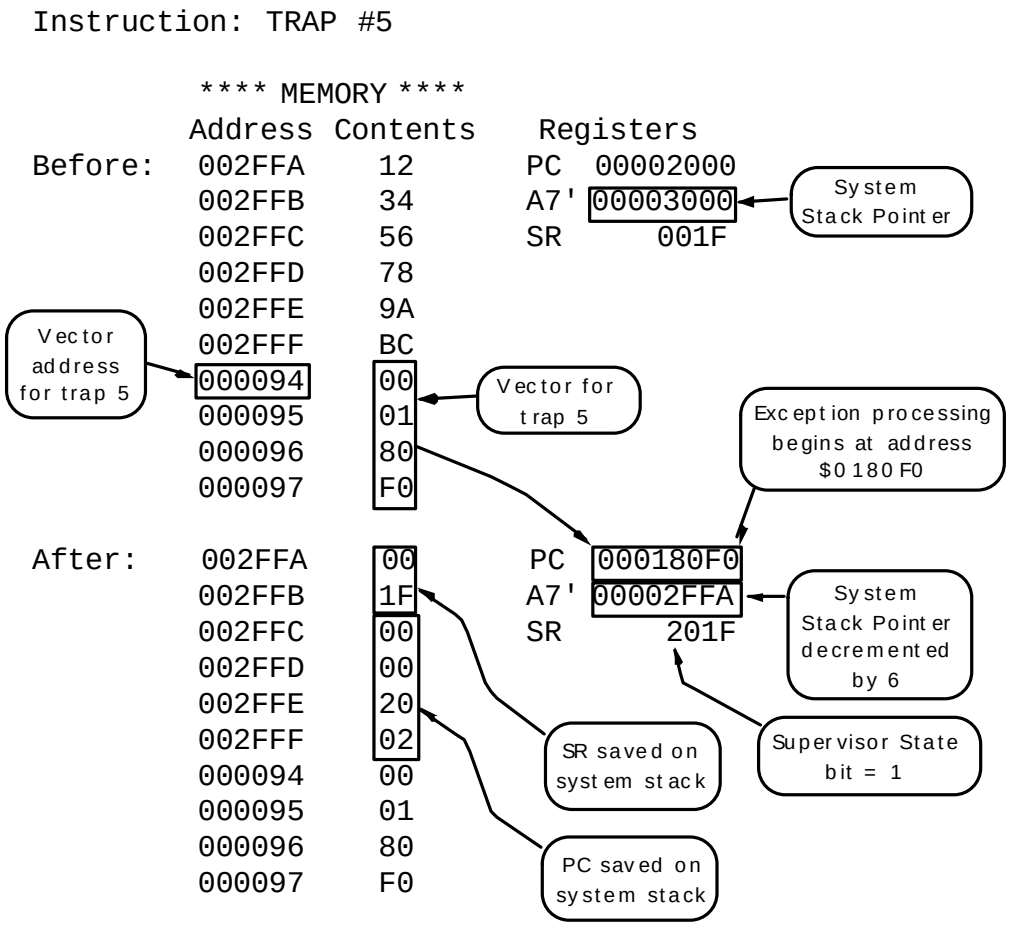
Instruction	Operation
ANDI^{++}	AND immediate to status register/condition code register
CHK	Trap on upper out-of-bounds operand
EORI^{++}	Exclusive OR immediate to status register/condition code register
ILLEGAL	Illegal instruction trap
MOVE^{++}	Move to/from status register/condition code register
ORI++	OR immediate to status register/condition code register
RESET^{+}	Assert RESET line
STOP^{+}	Stop processor
TAS	Test and set operand
TRAP	Trap unconditionally
TRAPV	Trap on overflow

$^{+}$privileged instruction

$^{++}$privileged instruction if SR specified

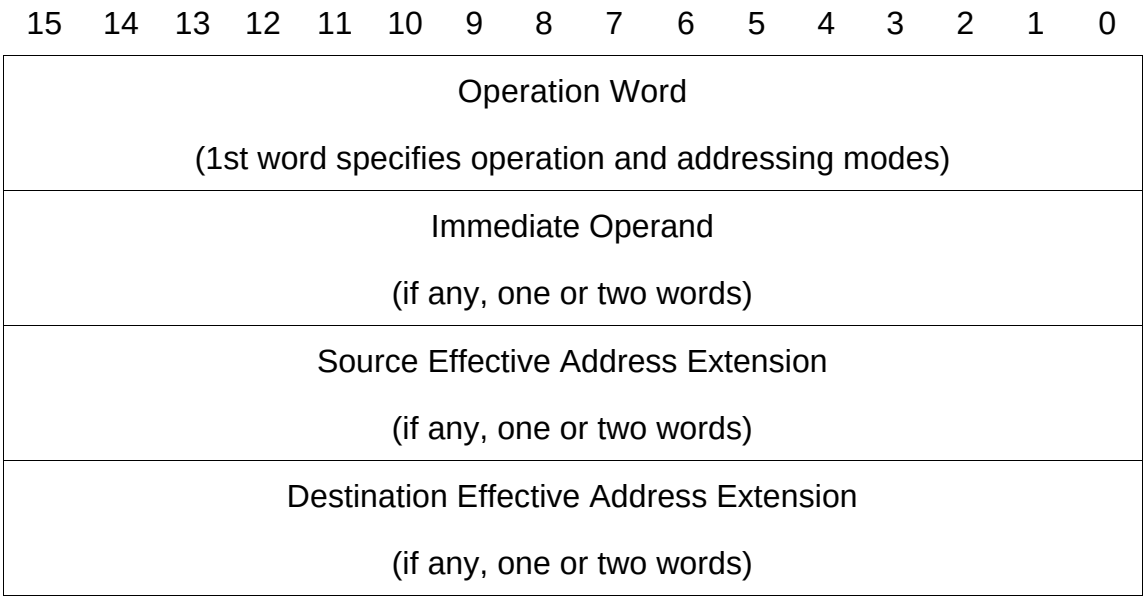
T3-10

TRAP Example



F3-18

68000 Instruction Format



F3-19

Effective Address Encoding

Addressing Mode	Mode Bits	Register Bits
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement†	101	register number
Address Register Indirect with Index*	110	register number
Absolute Short†	111	000
Absolute Long††	111	001
Program Counter with Displacement†	111	010
Program Counter with Index*	111	011
Immediate or Status Register†††	111	100

† One extension word required

†† Two extension words required

††† For Immediate addressing, one or two extension words required depending on the size of the operation

* One extension word required; see Table C-4 for the encoding

T3-11

68000 Condition Code Encoding

Mnemonic	Condition	Encoding	Test
T†	true	0000	1
F†	false	0001	0
HI	high	0010	$\overline{C} \bullet \overline{Z}$
LS	low or same	0011	C+Z
CC(HS)	carry clear	0100	$\overline{C}$
CS(LO)	carry set	0101	C
NE	not equal	0110	$\overline{Z}$
EQ	equal	0111	Z
VC††	overflow clear	1000	$\overline{V}$
VS††	overflow set	1001	V
PL	plus	1010	$\overline{N}$
MI	minus	1011	N
GE††	greater or equal	1100	$N \bullet V + \overline{N} \bullet \overline{V}$
LT††	less than	1101	$N \bullet \overline{V} + \overline{N} \bullet V$
GT††	greater than	1110	$N \bullet V \bullet \overline{Z} + \overline{N} \bullet \overline{V} \bullet \overline{Z}$
LE††	less or equal	1111	$Z + N \bullet \overline{V} + \overline{N} \bullet V$

† Not available for Bcc instruction

†† Twos complement arithmetic

• = Boolean AND

+ = Boolean OR

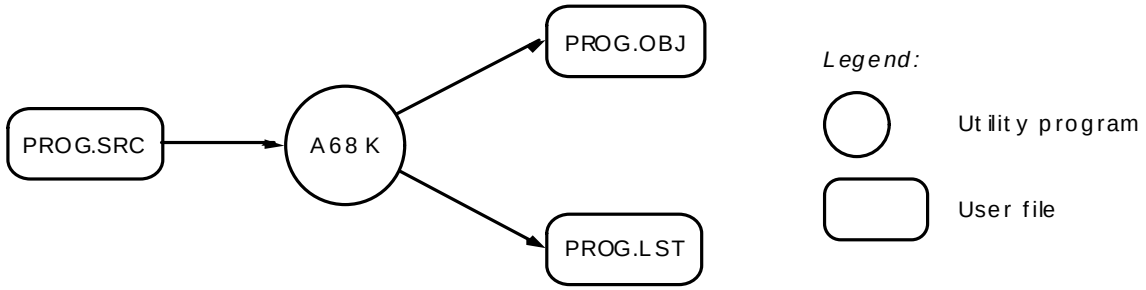
T3-12

Opcode Map

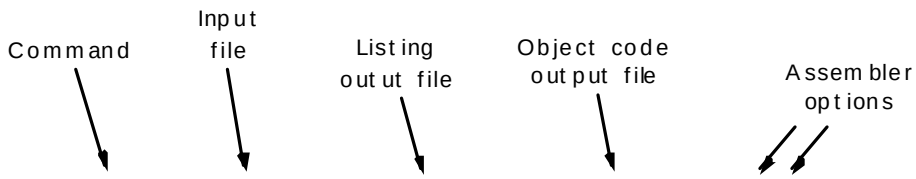
Bits 15 through 12	Operation
0000	Bit Manipulation/MOVEP/Immediate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/Scc/DBcc
0110	Bcc/BSR
0111	MOVEQ
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate
1111	(Unassigned)

T3-13

Assembler Operation



(a)



(b)

F4-1

Assembler Files

Label field | Mnemonic field | Operand field | Comment field (empty)

```
        ORG     $1000
PROG    LEA     $800,A0
        MOVE.B  #50,D0
        CLR.W   D7
LOOP    ADD.W   (A0)+,D7
        SUBQ.B  #1,D0
        BRA     *
        END
```

Source file

(a)

Line number | Address | Contents | Source file

```
1  00001000                        ORG     $1000
2  00001000 41F80800 PROG          LEA     $800,A0
3  00001004 103C0032               MOVE.B  #50,D0
4  00001008 4247                   CLR.W   D7
5  0000100A DE58     LOOP          ADD.W   (A0)+,D7
6  0000100C 5300                   SUBQ.B  #1,D0
7  0000100E 60FE                   BRA     *
8  00001010                        END
```

Listing file

(b)

F4-2

Listing Examples

```
 1  00001000                ORG     $1000
 2  00001000 6006           BRA     *+8             ;"*" location counter
 3  00001002 60FE           BRA     *               ;branch to itself
 4  00001004 6000FFFE HERE  BRA     HERE            ;branch to itself
 5  00001008 181B           MOVE.B  (A3)+,D4        ;indirect addressing
 6  00000064          COUNT EQU     100             ;equate symbol to value
 7  0000100A 3A3C0064       MOVE.W  #COUNT,D5       ;symbol as immed. data
 8  0000100E 3A3C0064       MOVE.W  #100,D5         ;decimal
 9  00001012 3A3C0064       MOVE.W  #$64,D5         ;hexadecimal
10  00001016 3A3C0064       MOVE.W  #144Q,D5        ;octal (A68K format)
11  0000101A 3A3C0064       MOVE.W  #%01100100,D5   ;binary
12  0000101E 3E3CFFFB       MOVE.W  #-5,D7          ;negative number, decimal
13  00001022 3E3CFFFB       MOVE.W  #$FFFB,D7       ;negative number, decimal
14  00001026 3A390000       MOVE    $F000,D5        ;data address
    0000102A F000
15  0000F000          PORT  EQU     $F000           ;equate symbol as address
16  0000102C 3A390000       MOVE    PORT,D5         ;data address (symbol)
    00001030 F000
17  00001032 4E71     BACK  NOP                     ;code address (NOP =
18  00001034 4E71           NOP                     ;   no operation)
19  00001036 67FA           BEQ     BACK
20  00001038                END
```

F4-4

Assemble-Time Operators

Operator†	Function	Precedence	Type
-	Unary minus	1	Unary
.NOT.	Logical NOT	1	Unary
.LOW.	Low byte	1	Unary
.HIGH.	High byte	1	Unary
.LWRD.	Low word	1	Unary
.HWRD.	High word	1	Unary
*	Multiplication	3	Binary
/	Division	3	Binary
+	Addition	4	Binary
-	Subtraction	4	Binary
.MOD.	Modulo	3	Binary
.SHR.	Logical shift right	3	Binary
.SHL.	Logical shift left	3	Binary
.AND.	Logical AND	5	Binary
.OR.	Logical OR	6	Binary
.XOR.	Logical XOR	6	Binary
.EQ.	Equal††	7	Binary
.NE.	Not Equal††	7	Binary
.GE.	Greater or equal††	7	Binary
.LE.	Less or equal††	7	Binary
.GT.	Greater than††	7	Binary
.LT.	Less than††	7	Binary
.UGT.	Unsigned greater than††	7	Binary
.ULT.	Unsigned less than††	7	Binary

†Operators apply to A68K. Different assemblers may support different operators.

††Relational operators return 1s (true) or 0s (false).

T4-1

Examples of Assemble-Time Operators

```
 1  00000064           COUNT    EQU      100
 2  00002000                    ORG      $2000
 3  00002000 3A3CFFFF           MOVE     #-1,D5
 4  00002004 3A3C0009           MOVE     #4+50/10,D5
 5  00002008 3A3C0001           MOVE     #25.mod.6,D5
 6  0000200C 3A3C0400           MOVE     #$8000.shr.5,D5
 7  00002010 3A3C0040           MOVE     #$45&$F0,D5
 8  00002014 3A3C0041           MOVE     #.high.'AB',D5
 9  00002018 3A3CFFFF           MOVE     #5.gt.4,D5
10  0000201C 3A3C0032           MOVE     #COUNT/2,D5
11  00002020                    END
```

F4-5

Assembler Directives

Directive	Operation	Syntax		
ORG	set program origin		ORG	value
EQU	equate value to symbol	symbol	EQU	value
END	end of source program		END	label
DC	define data constant	[label]	DC	number[,number][...]
DS	define RAM storage	[label]	DS	count
RSEG	begin relocatable segment		RSEG	name
EXTERN	define external symbol		EXTERN	symbol[,symbol][...]
PUBLIC	define public symbol		PUBLIC	symbol[,symbol][...]

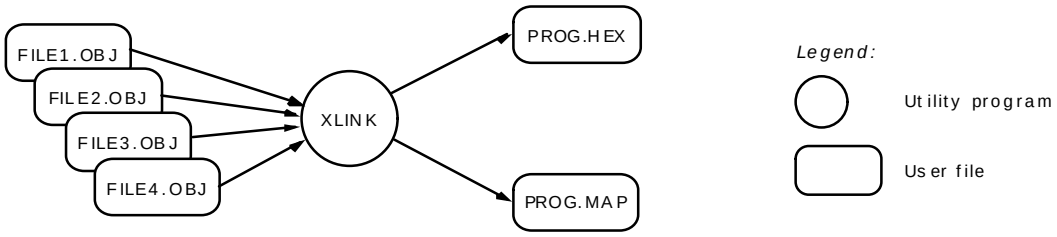
T4-2

Listing Examples

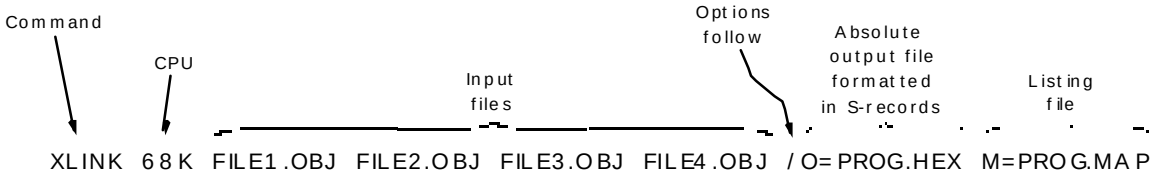
```
    Address  Contents  *************** CH4-6.SRC ********************
 1  00001000                 ORG      $1000
 2  00001000 1A3C0064 START  MOVE.B   #100,D5
 3  00000064          COUNT  EQU      100
 4  00002000                 ORG      $2000
 5  00002000 1A3C0064 HERE   MOVE.B   #COUNT,D5
 6  0000000D          CR     EQU      $0D         define a symbol
 7  00003000                 ORG      $3000       set origin
 8  00003000 0005FFFF NUM    DC       5,-1        word size default
 9  00003004 05FF     MORE   DC.B     5,-1        byte size constants
10  00003006 4A4F484E NAME   DC.B     'JOHN'      ASCII string
11  0000300A 0D00            DC.B     CR,0        CR is a symbol
12  00004001                 ORG      $4001
13  00004002 000F     VALUE  DC       15          decimal constant
14  00005000                 ORG      $5000
15  00005000 6100            DC.W     'a'
16  00000050          LENGTH EQU      80
17  00006000                 ORG      $6000
18  00006000          BUFFER DS.B     LENGTH
19  00006050          TEMP   DS.B     1
20  00007000                 ORG      $7000
21  00007000 7250            MOVE.L   #LENGTH,D1 use R1 as counter
22  00007002 327C6000        MOVEA    #BUFFER,A1 A1 points to buff
23  00007006 12FC0000 LOOP   MOVE.B   #0,(A1)+   clear location
24  0000700A 5301            SUBQ.B   #1,D1       done?
25  0000700C 66F8            BNE      LOOP        no: clear again
26  00000000                 RSEG     EPROM
27  00000000 1A3C002C BEGIN  MOVE.B   #44,D5
28  00000004                 END
```

F4-8

Linker Operation



(a)



(b)

F4-10

User Mode vs. Supervisor Mode

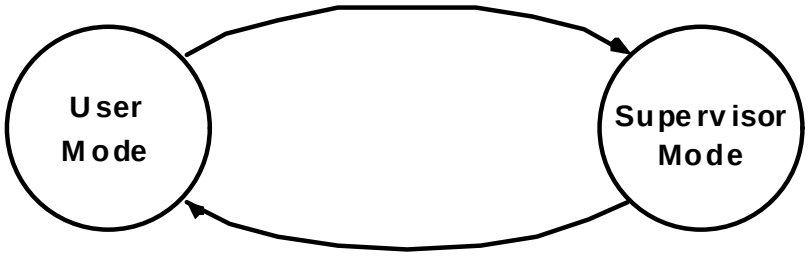
Feature	User Mode	Supervisor Mode
Entered by	Clearing S-bit in SR	Exception processing
FC2 =	0	1
Active stack pointer	USP	SSP
Other stacks using	A0 - A6	USP, A0-A6
SR access Read: Write:	 Entire SR CCR bits only	 Entire SR Entire SR

Instructions available	All except
	
	AND #data,SR
	EOR #data,SR
	MOVE <ea>,SR
	MOVE USP,An
	MOVE An,USP
	OR #data,SR
	RESET
	RTE
	STOP
		All

T6-1

Changing Between
User Mode and Supervisor Mode

Transition may occur only
during exception processing

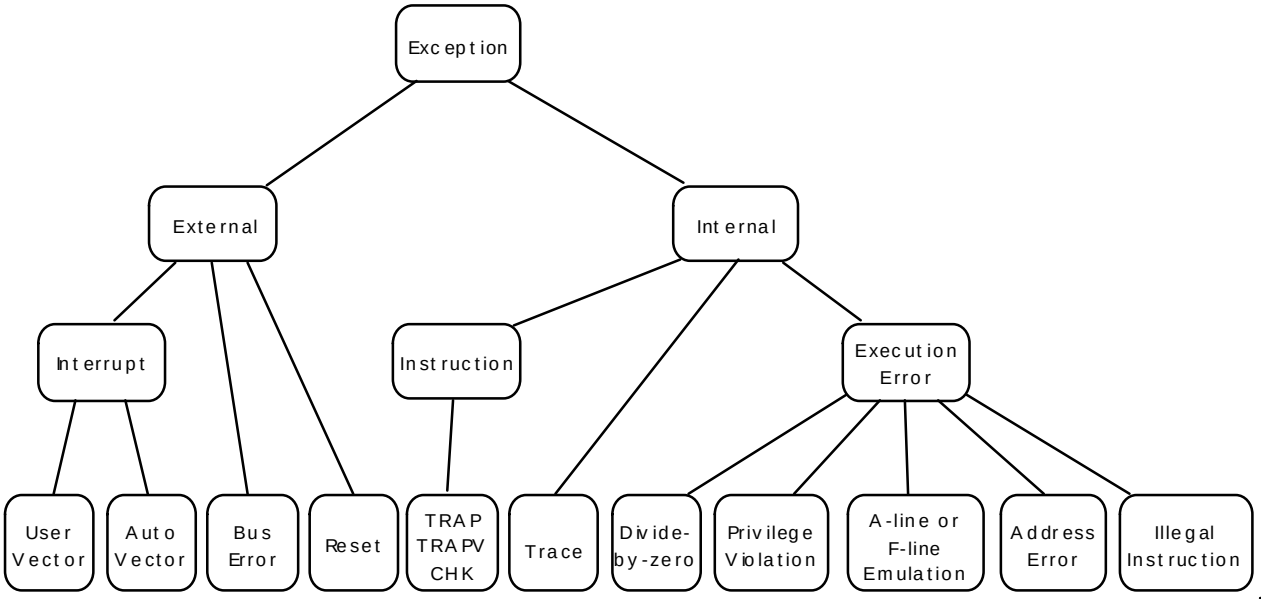


Transition may occur
through four instructions:

MOVE to SR
ANDI to SR
EOR to SR
RTE

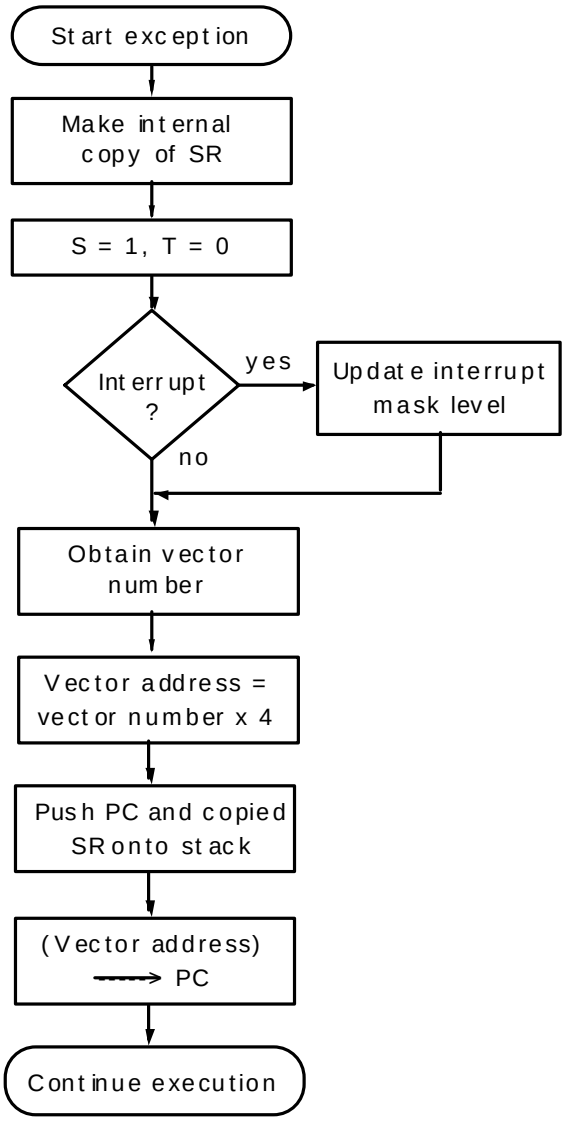
F6-1

Exception Tree



F6-2

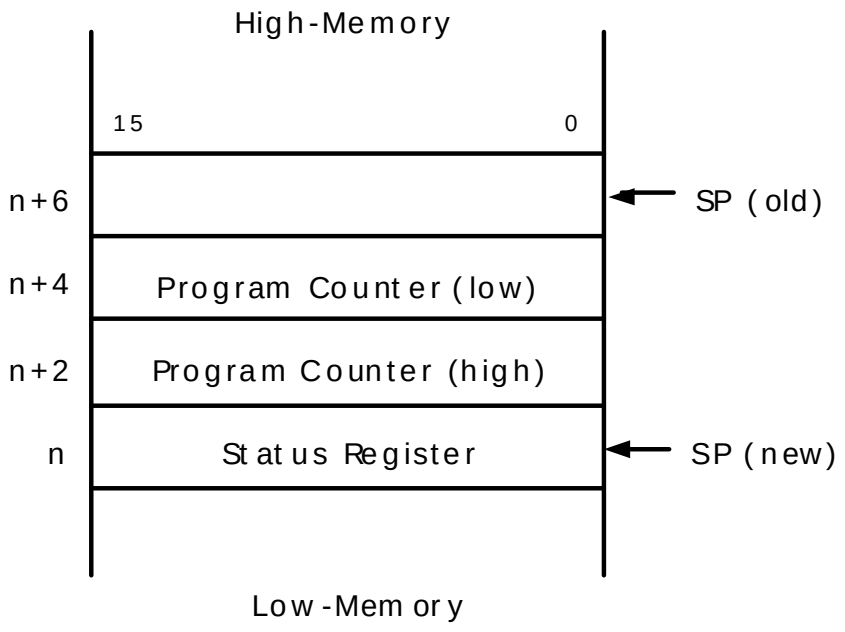
Exception Processing Sequence



F6-3

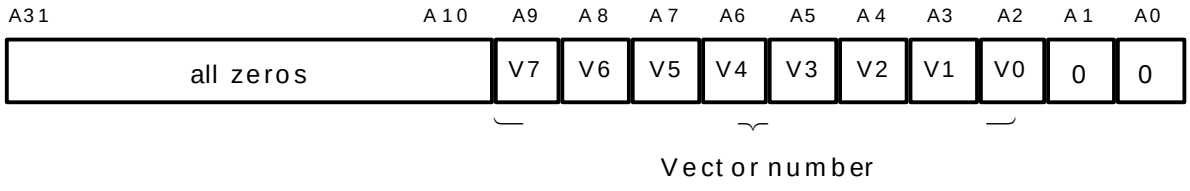
Stack Frame for Exceptions

(except bus error and address error)



F6-4

Exception Vector Address



F6-5

Reset and Exception Vector Assignments

Vector Number	Hexadecimal Address	Assignment
0	000	Reset SSP†
–	004	Reset PC†
2	008	Bus Error
3	00C	Address Error
4	010	Illegal instruction
5	014	Divide-by-zero
6	018	CHK instruction
7	01C	TRAPV instruction
8	020	Privilege violation
9	024	Trace
10	028	Line 1010 emulator
11	02C	Line 1111 emulator
12	030	(reserved)
13	034	(reserved)
14	038	Format error (68010)
15	03C	Uninitialized interrupt vector
16-23	040-05C	(reserved)
24	060	Spurious interrupt††
25	064	Level 1 interrupt autovector
26	068	Level 2 interrupt autovector
27	06C	Level 3 interrupt autovector
28	070	Level 4 interrupt autovector
29	074	Level 5 interrupt autovector
30	078	Level 6 interrupt autovector
31	07C	Level 7 interrupt autovector
32-47	080-0BC	TRAP instruction vectors†††
48-63	0C0-0FC	(reserved)
64-255	100-3FC	User interrupt vectors

† The reset vector is four words and resides in the supervisor program (SP) space. All other vectors reside in the supervisor data (SD) space.

†† The spurious interrupt vector is taken when there is a bus error during an interrupt acknowledge cycle.

††† Trap #n uses vector number 32 + n. See Table 6-5.

T6-2

Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception processing begins within two CPU cycles
1	Trace Interrupt Illegal Instruction Privilege Violation	Exception processing begins before the next instruction
2	TRAP, TRAPV CHK Zero Divide	Exception processing begins by normal instruction execution

T6-3

Traps vs. Subroutines

Features	Traps	Subroutines
Initiated from	user mode or supervisor mode	user mode or supervisor mode
Routine executes in	supervisor mode	user mode or supervisor mode
Registers saved	PC and SR	PC
Registers saved on	system stack	user stack or system stack
Routine ends with	RTE	RTS
Privilege state after is	user mode or supervisor mode	user mode or supervisor mode

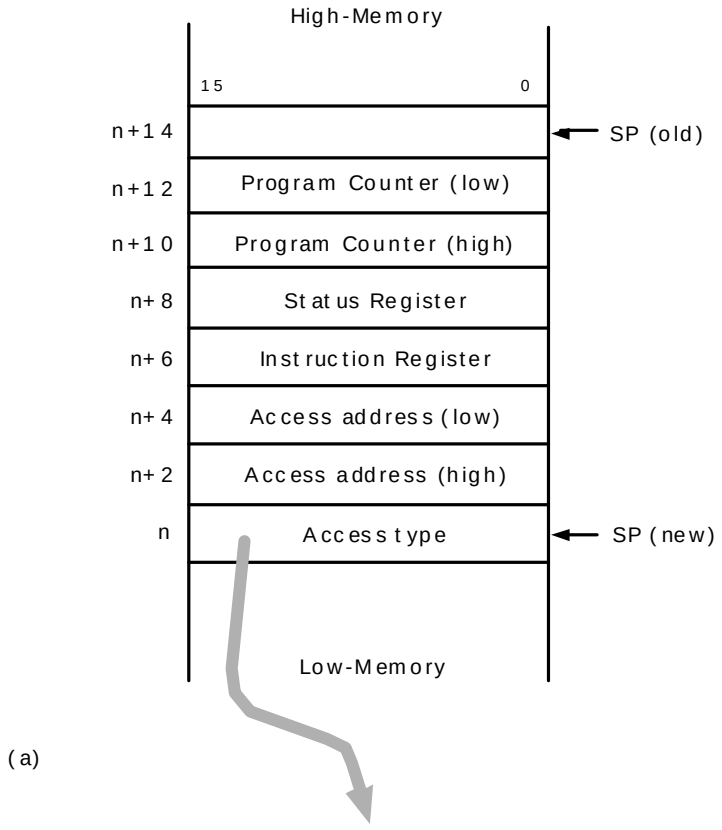
T6-4

Vector Assignments for TRAP Instructions

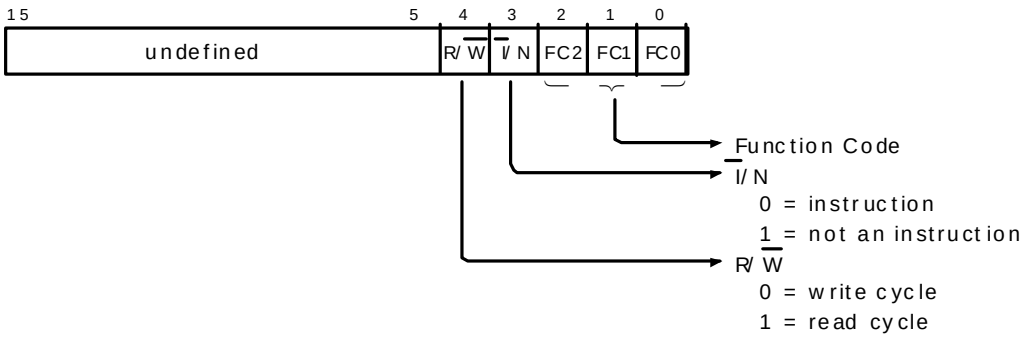
Instruction	Vector Number	Vector Address
TRAP #0	32	$000080
TRAP #1	33	$000084
TRAP #2	34	$000088
TRAP #3	35	$00008C
TRAP #4	36	$000090
TRAP #5	37	$000094
TRAP #6	38	$000098
TRAP #7	39	$00009C
TRAP #8	40	$0000A0
TRAP #9	41	$0000A4
TRAP #10	42	$0000A8
TRAP #11	43	$0000AC
TRAP #12	44	$0000B0
TRAP #13	45	$0000B4
TRAP #14	46	$0000B8
TRAP #15	47	$0000BC

T6-5

Stack Frame for
Bus Error and Address Error



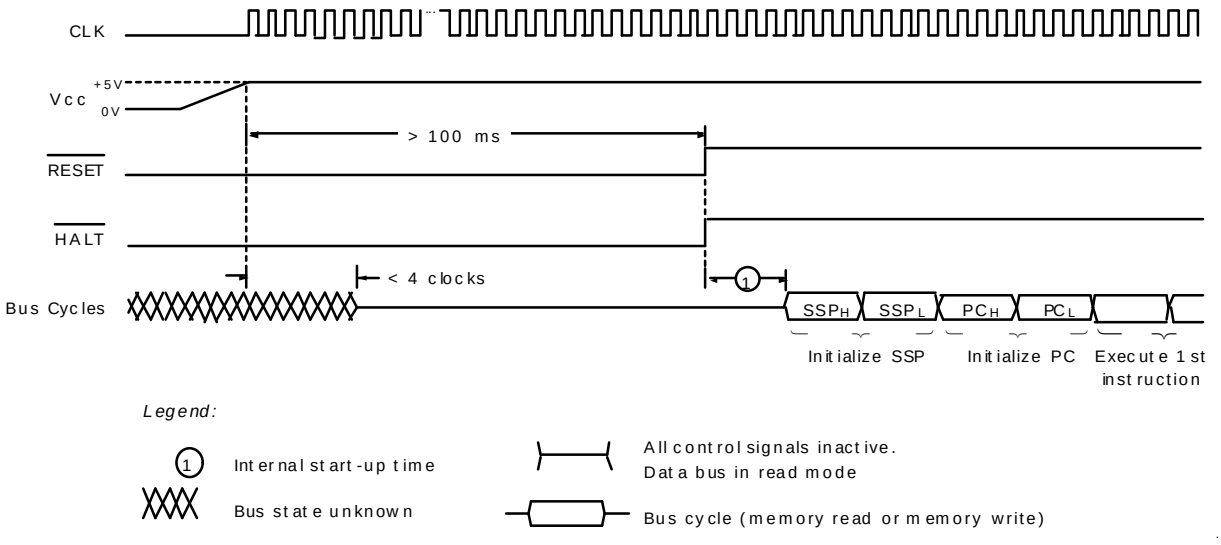
(a)



(b)

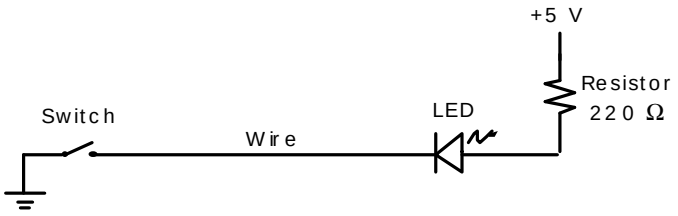
F6-6

Power-on Reset Timing



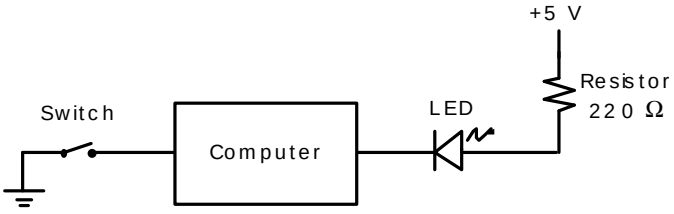
F6-7

A Switch as an Input Device and an LED as an Output Device



Switch	LED
OPEN	OFF
CLOSED	ON

(a)

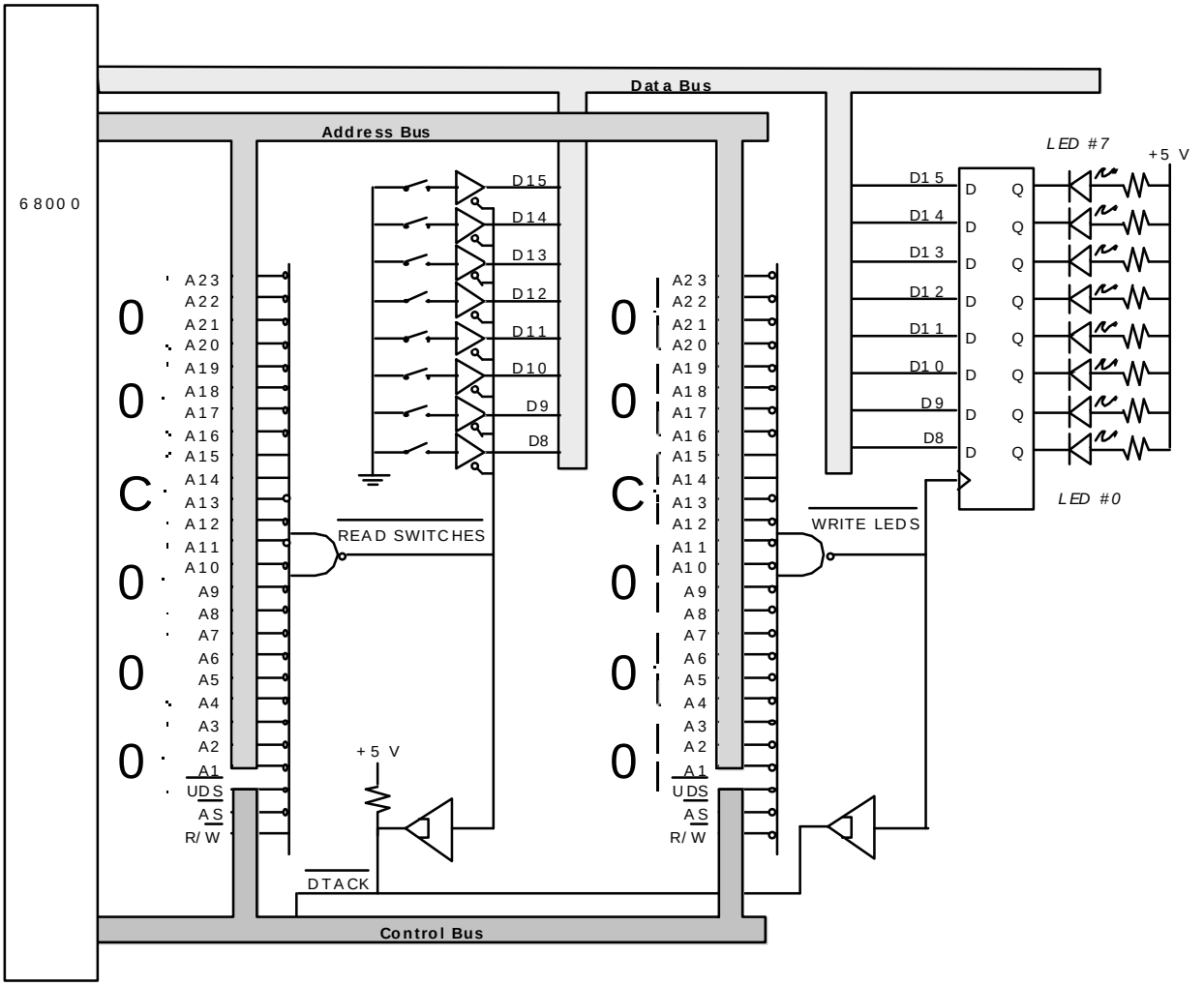


Switch	LED
OPEN	?
CLOSED	?

(b)

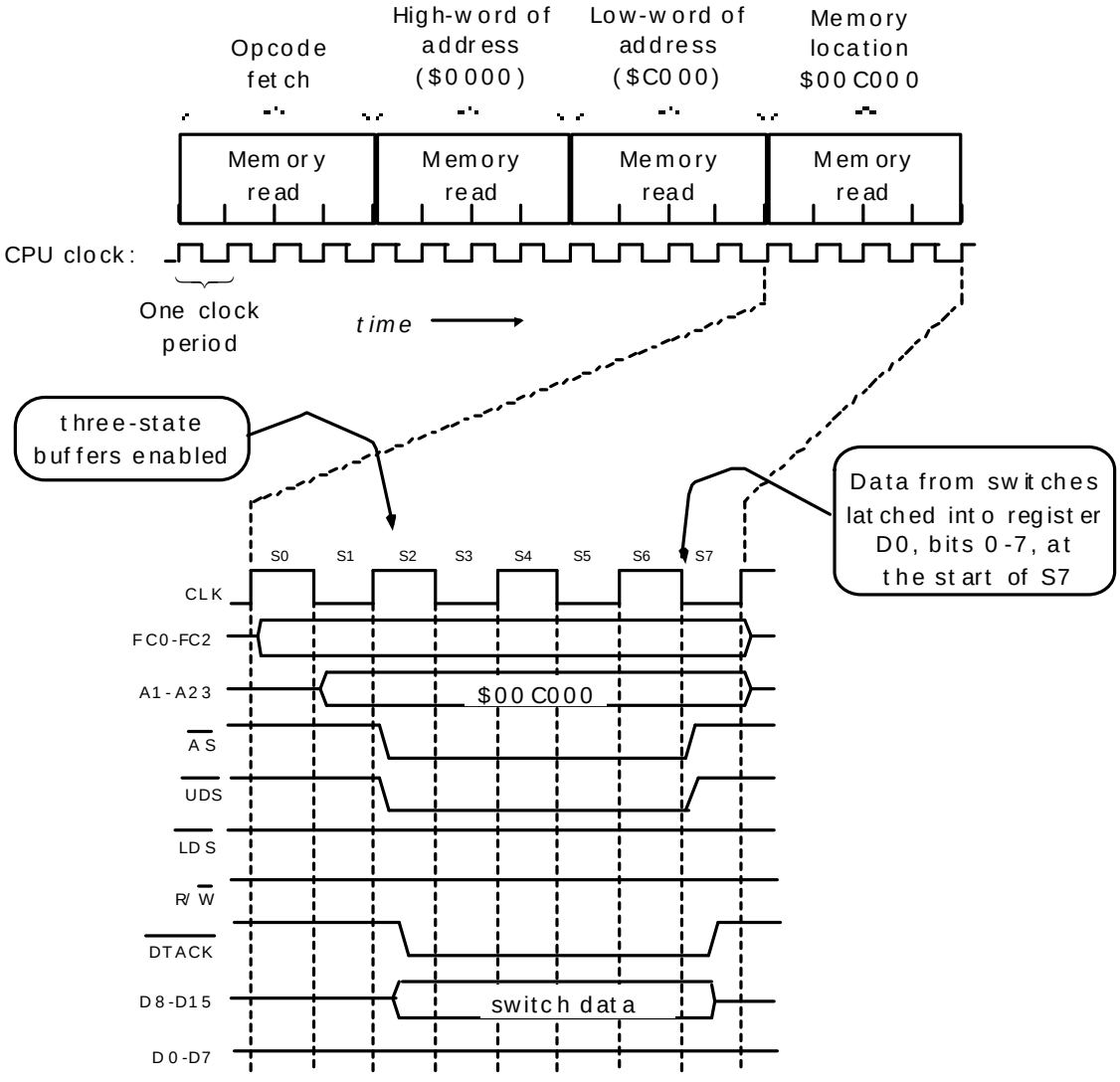
F7-1

Interface to Switches and LEDs (conceptual)



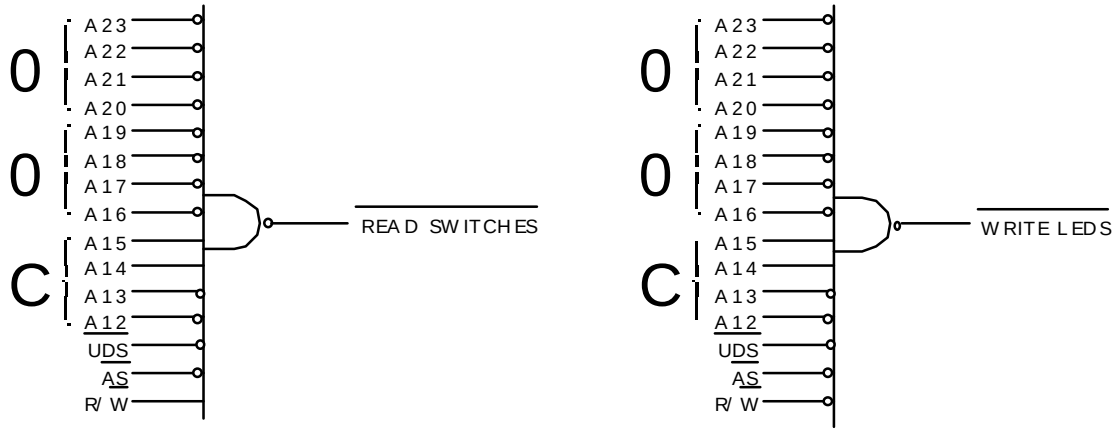
F7-2

Timing for MOVE.B $00C000,D0



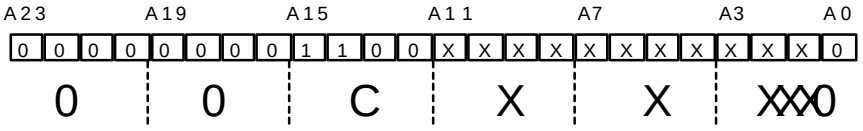
F7-3

Example of Partial Decoding



(a)

Address:

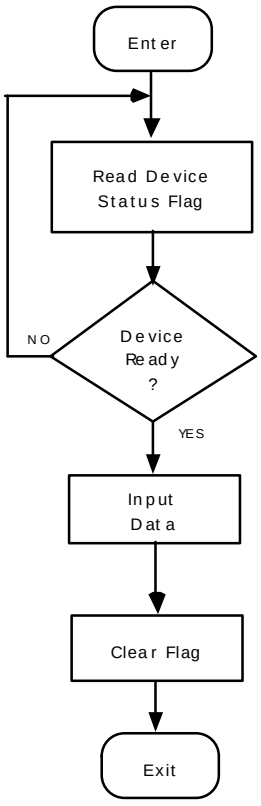


(b)

F7-5

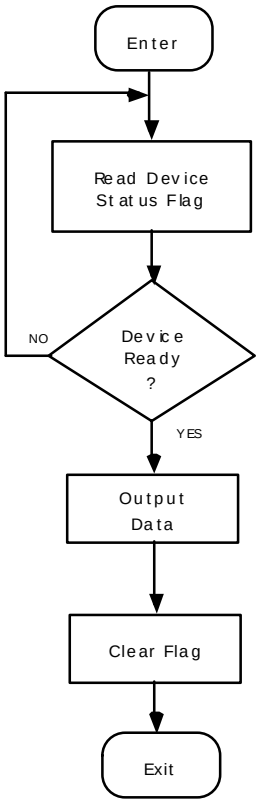
Flowcharts for Program-Conditional I/O

Input Flowchart:



(a)

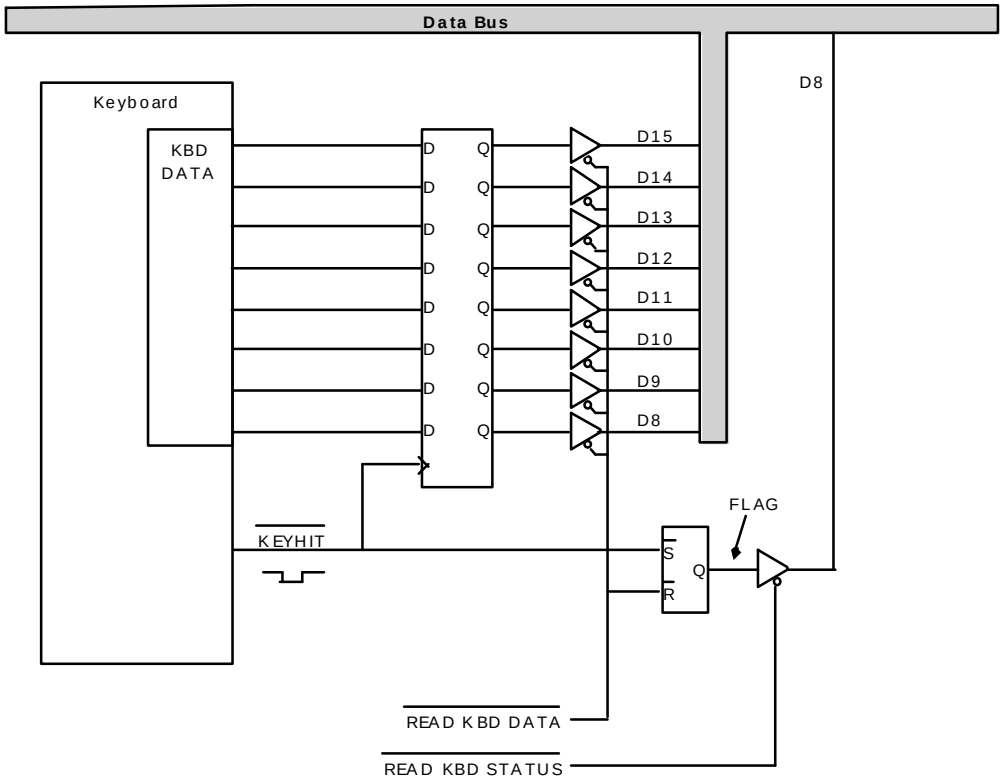
Output Flowchart:



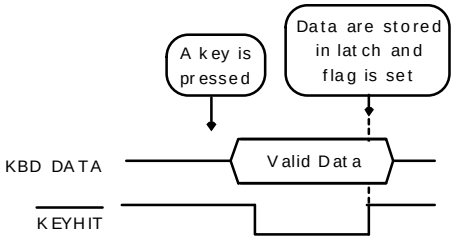
(b)

F7-7

Keyboard Interface



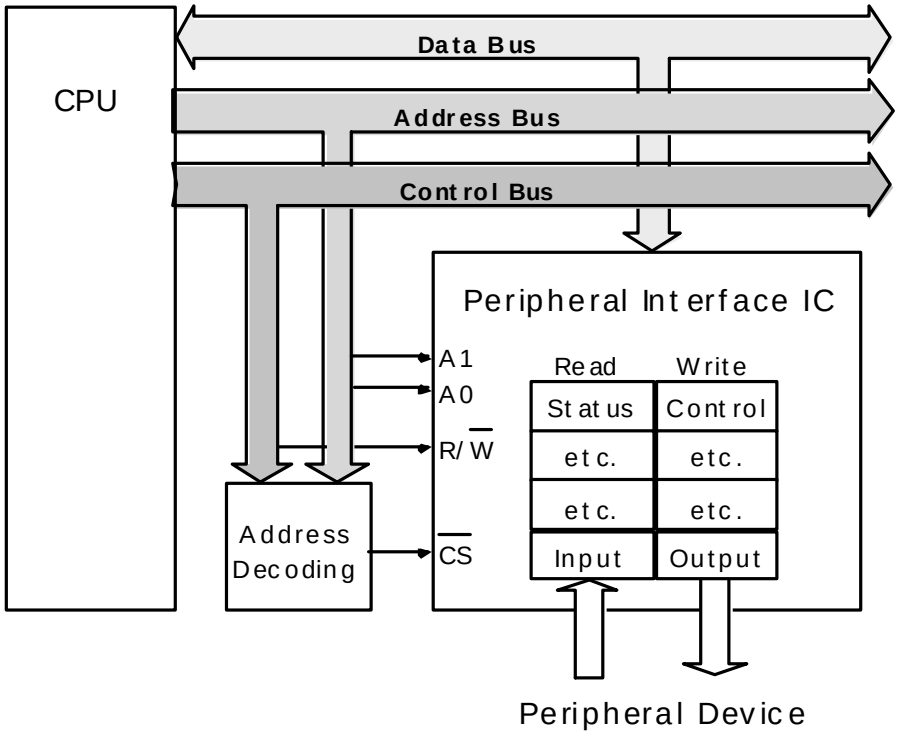
(a)



(b)

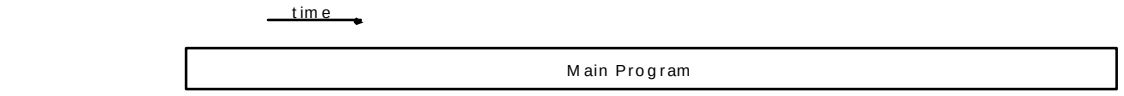
F7-6

Interface Using a Peripheral Interface IC

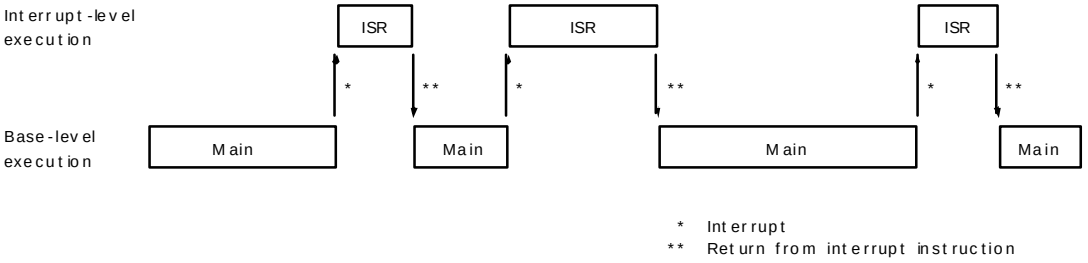


F7-9

Program Execution Without Interrupts or With Interrupts



(a)



(b)

F7-11

Interrupt Priority Conditions on -IPL2, -IPL1, and -IPL0

Signal			Interrupt	Condition	Maskable	Priority
$\overline{IP2}$	$\overline{IP1}$	$\overline{IP0}$				
1	1	1	0	No interrupt	-	-
1	1	0	1	Interrupt	Yes	Lowest
1	0	1	2	Interrupt	Yes	(etc.)
1	0	0	3	Interrupt	Yes	(etc.)
0	1	1	4	Interrupt	Yes	(etc.)
0	1	0	5	Interrupt	Yes	(etc.)
0	0	1	6	Interrupt	Yes	(etc.)
0	0	0	7	Interrupt	No	Highest

T7-3

Autovectors for Automatic IACK Cycles

Interrupt	Vector Address (Autovector)
0	-
1	$000064
2	$000068
3	$00006C
4	$000070
5	$000074
6	$000078
7	$00007C

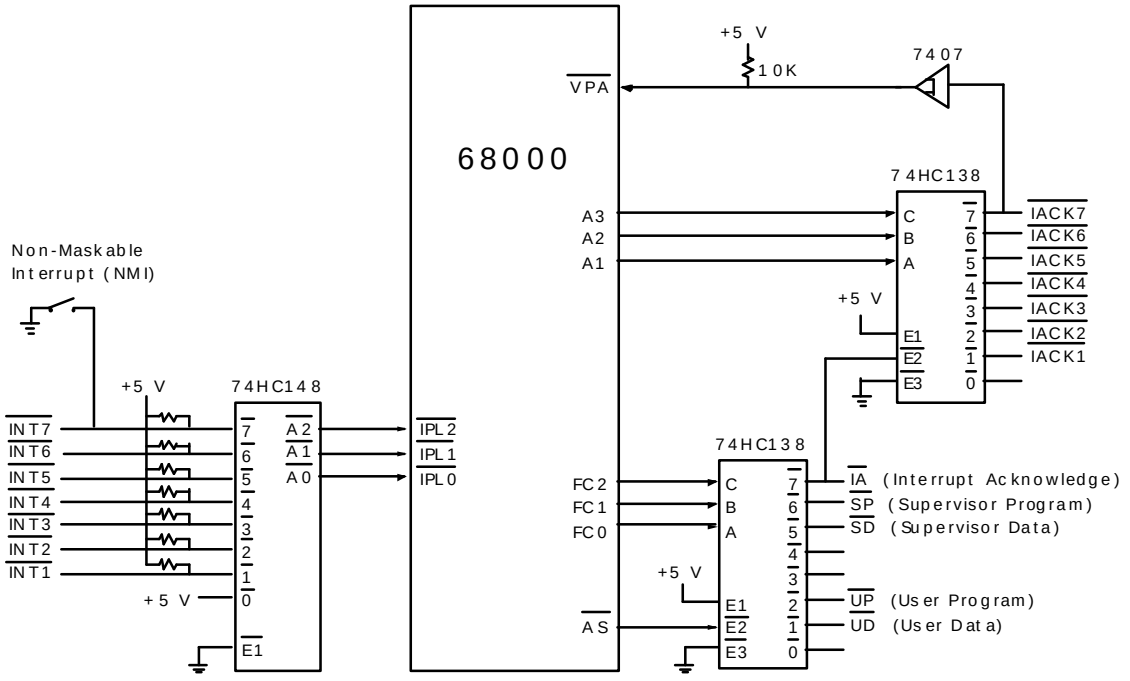
T7-4

Vector Addresses for User IACK Cycles

Vector Number	Vector Address
0	$000000
1	$000004
2	$000008
etc.	etc.
255	$0003FC

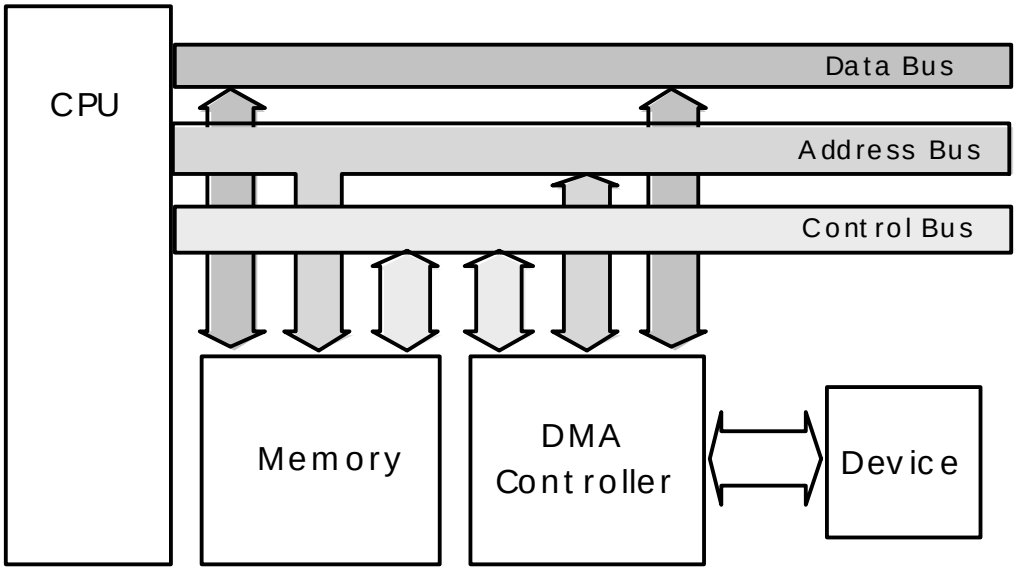
T7-5

Interrupt Circuitry



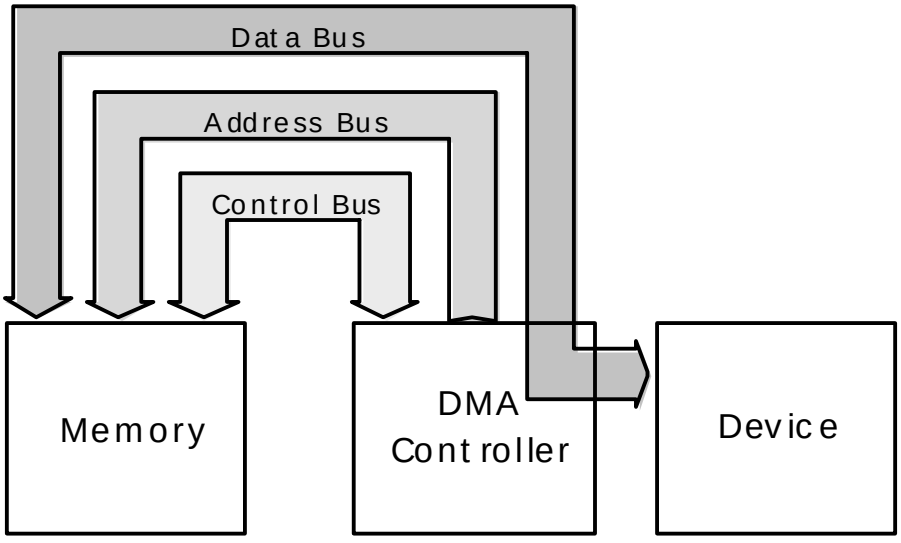
F7-12

Bus Connections for DMA Interface



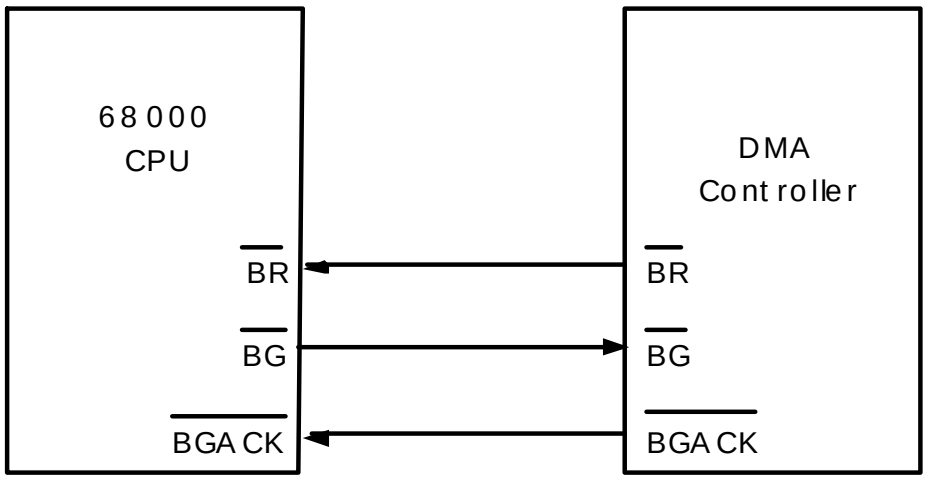
F7-13

Device-to-Memory Transfer Using DMA



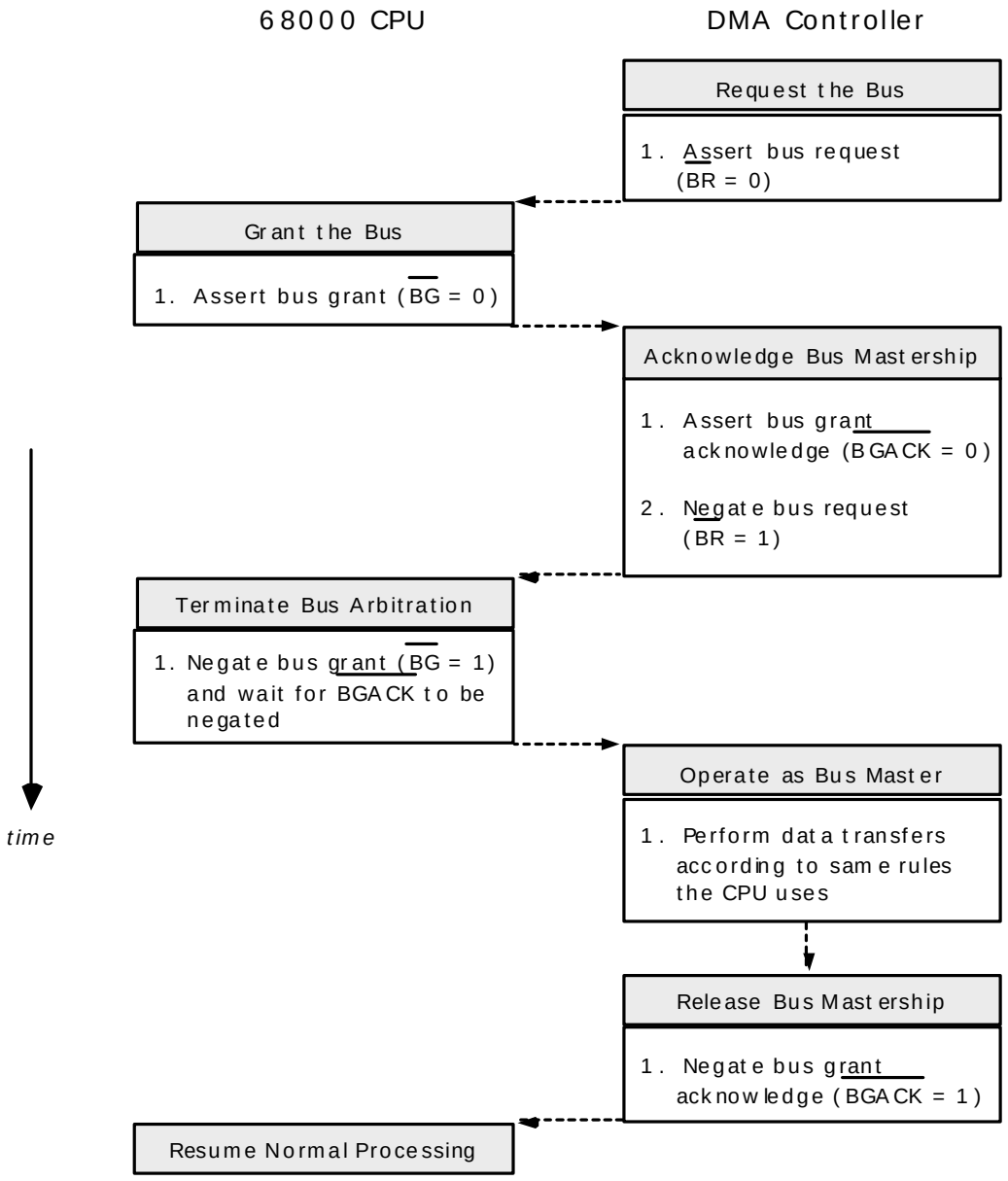
F7-14

68000 Bus Arbitration Control Signals



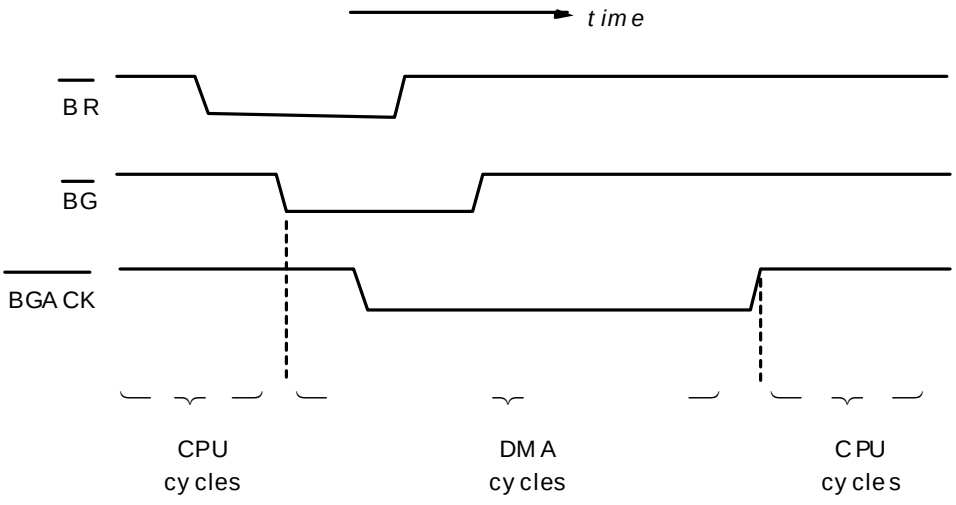
F7-15

Bus Arbitration



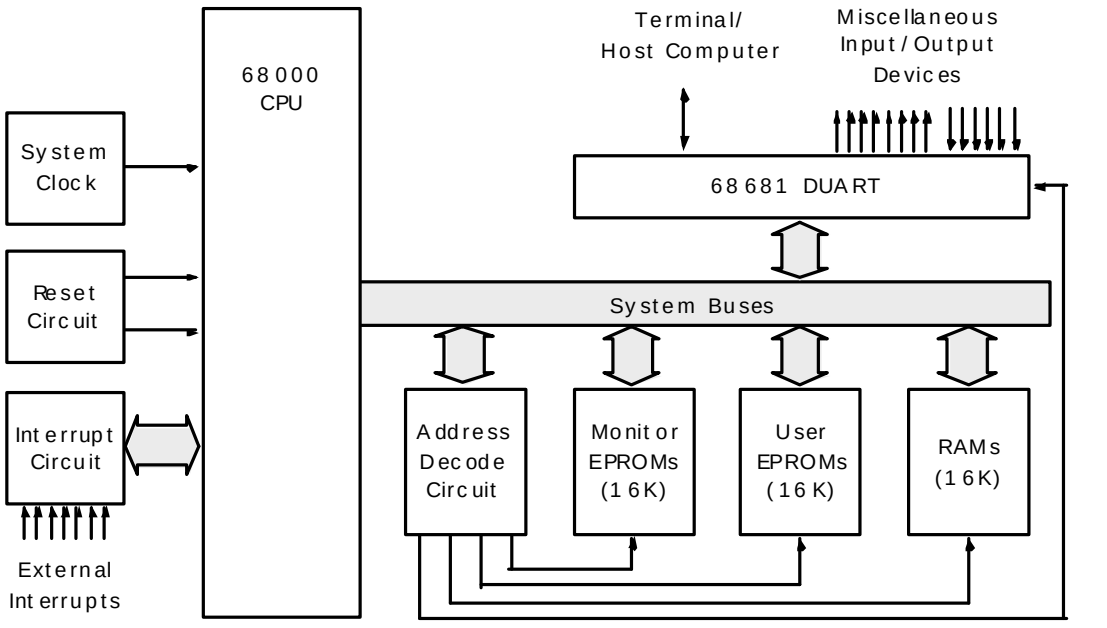
F7-16

Timing for Bus Arbitration



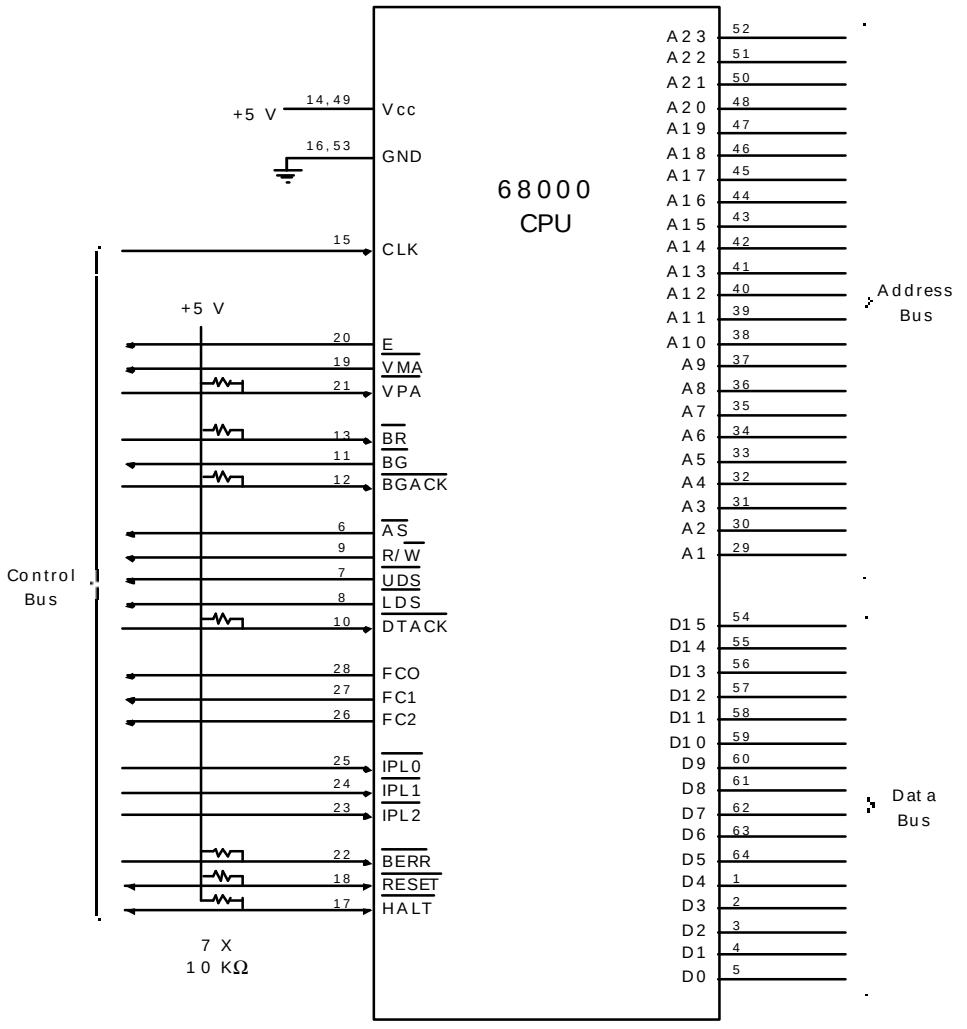
F7-17

Block Diagram of the 68KMB



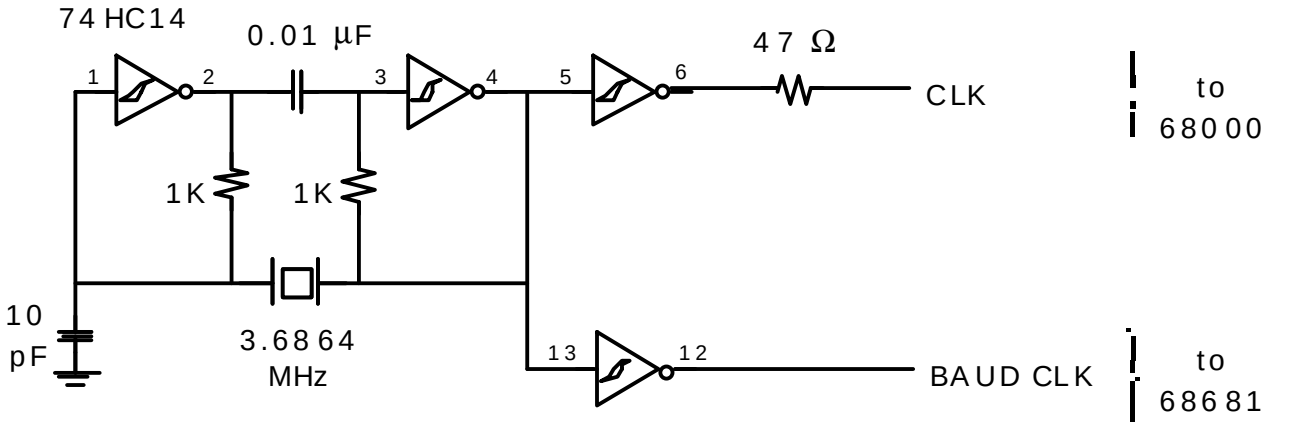
F8-3

68000 CPU



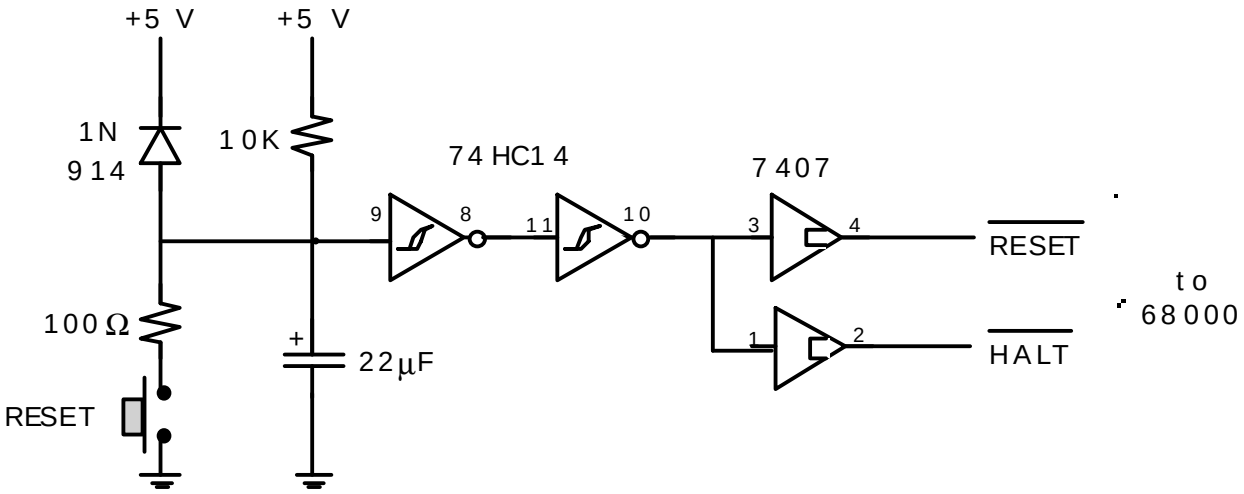
F8-4

68KMB Clock Circuit



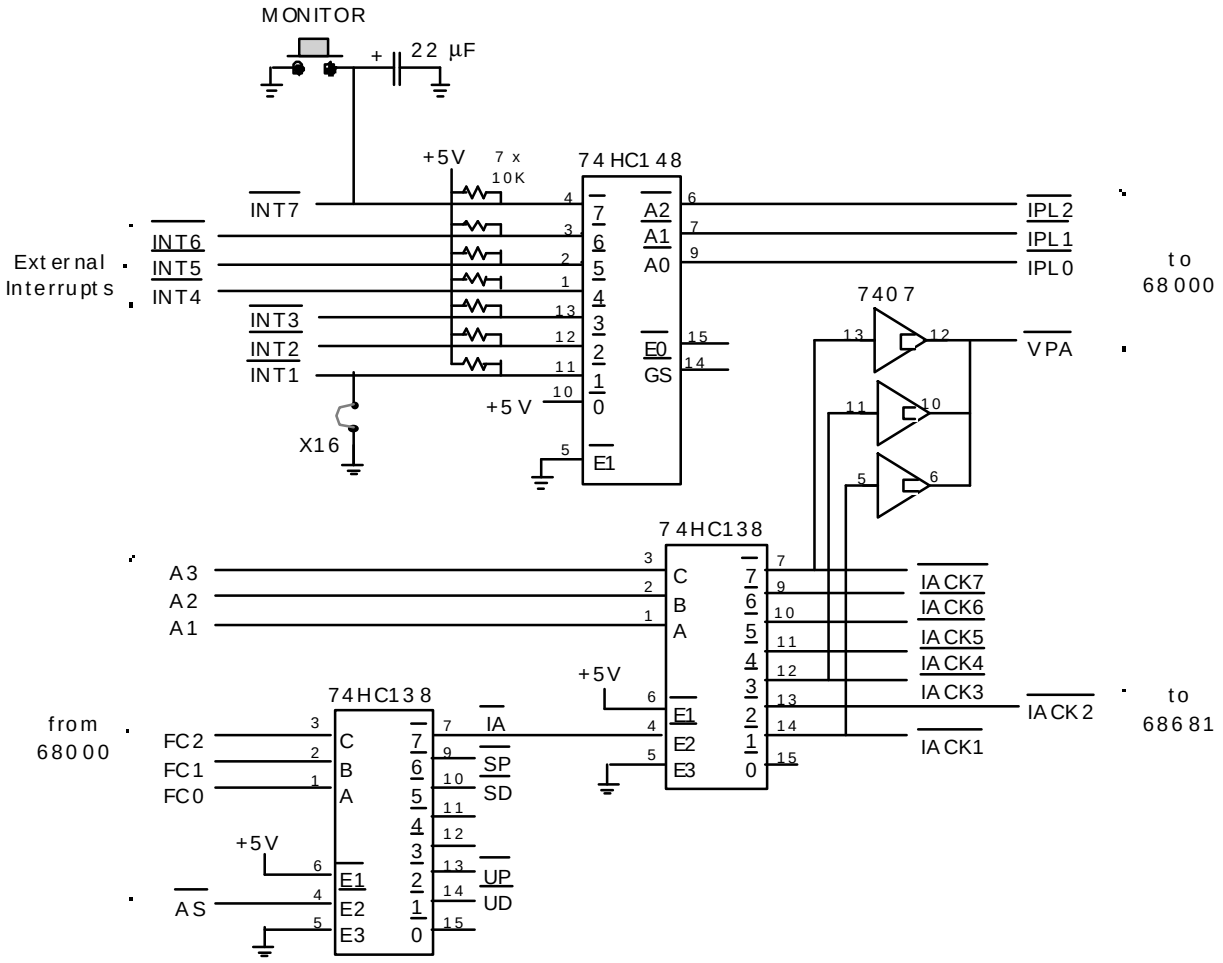
F8-6

68KMB Reset Circuit



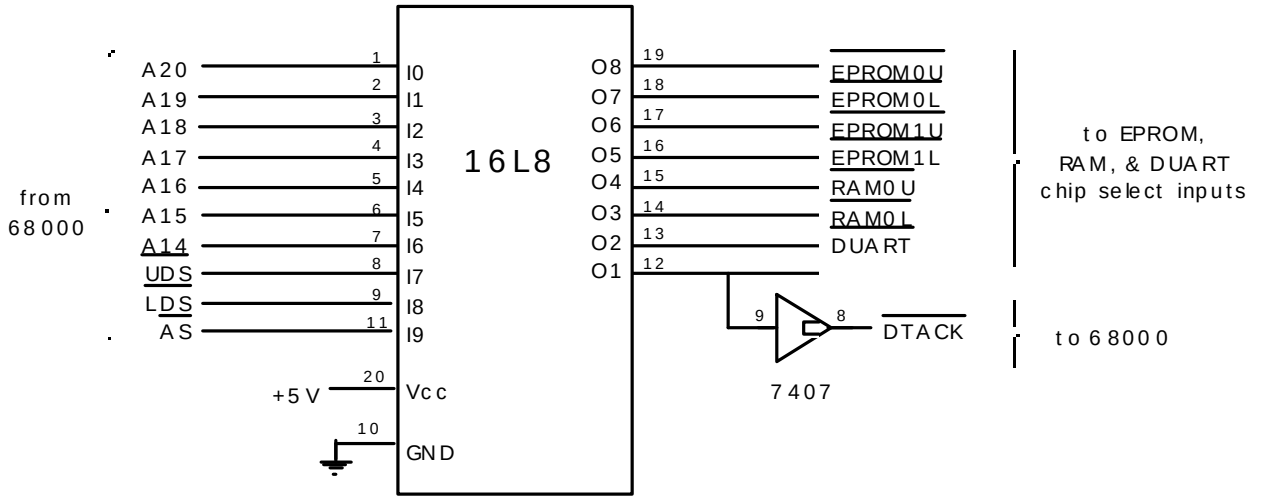
F8-7

68KMB Interrupt Circuit



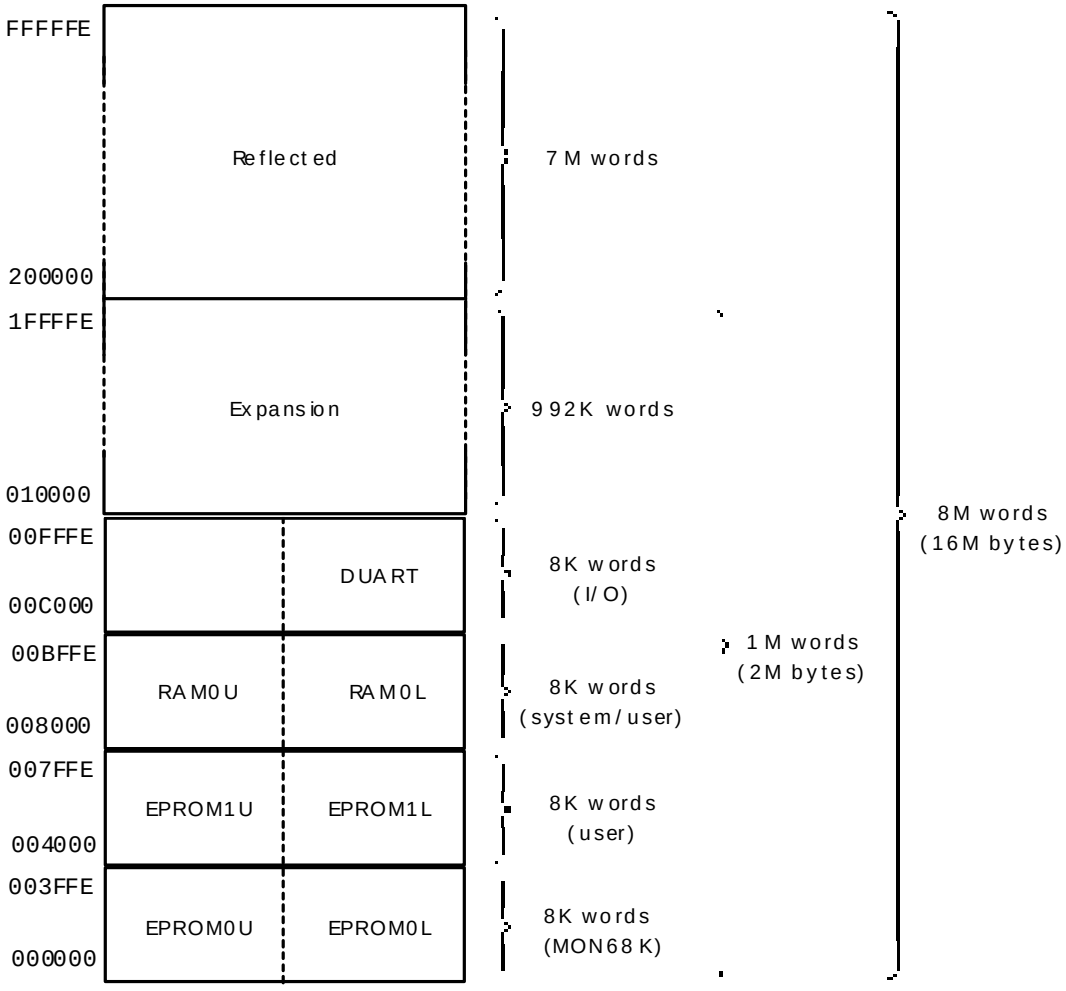
F8-8

68KMB Address Decoding



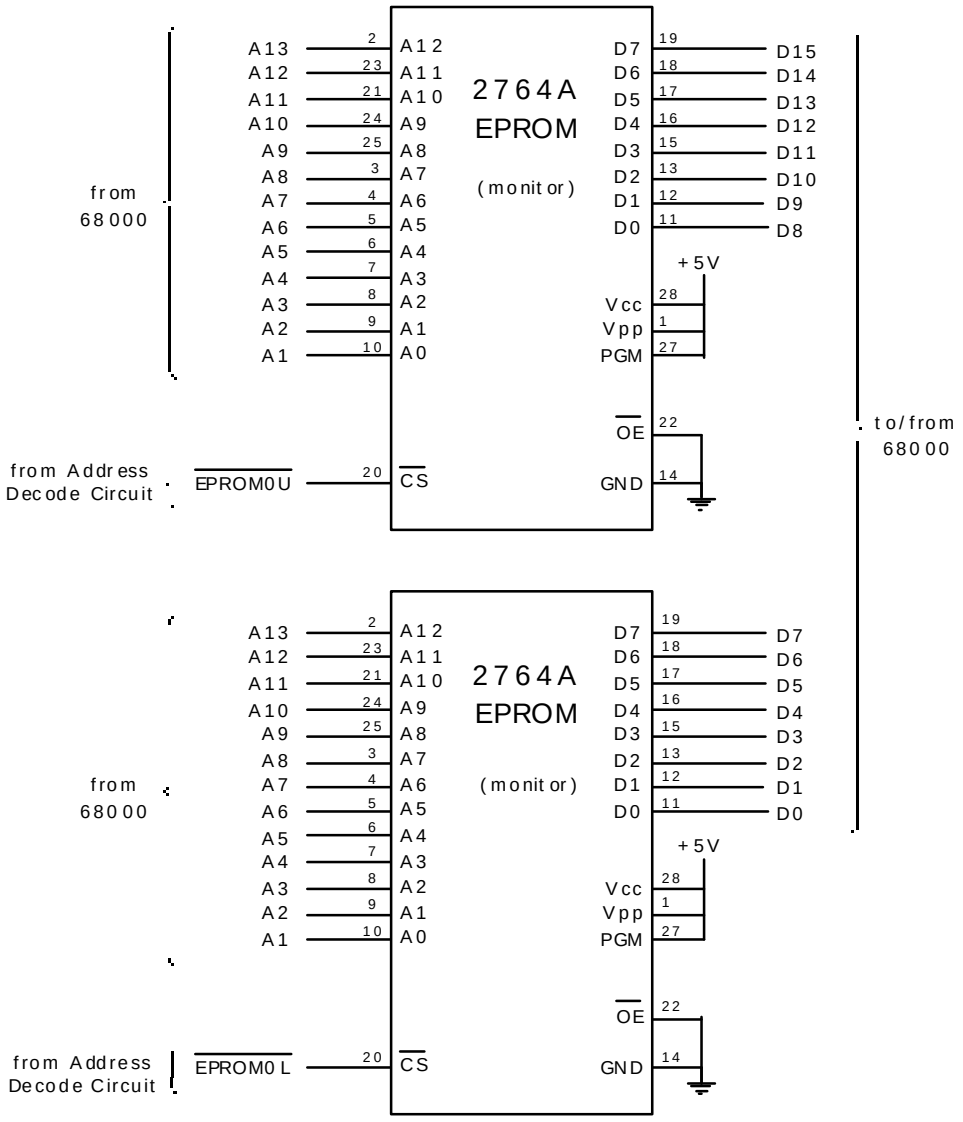
F8-9

68KMB Memory Map



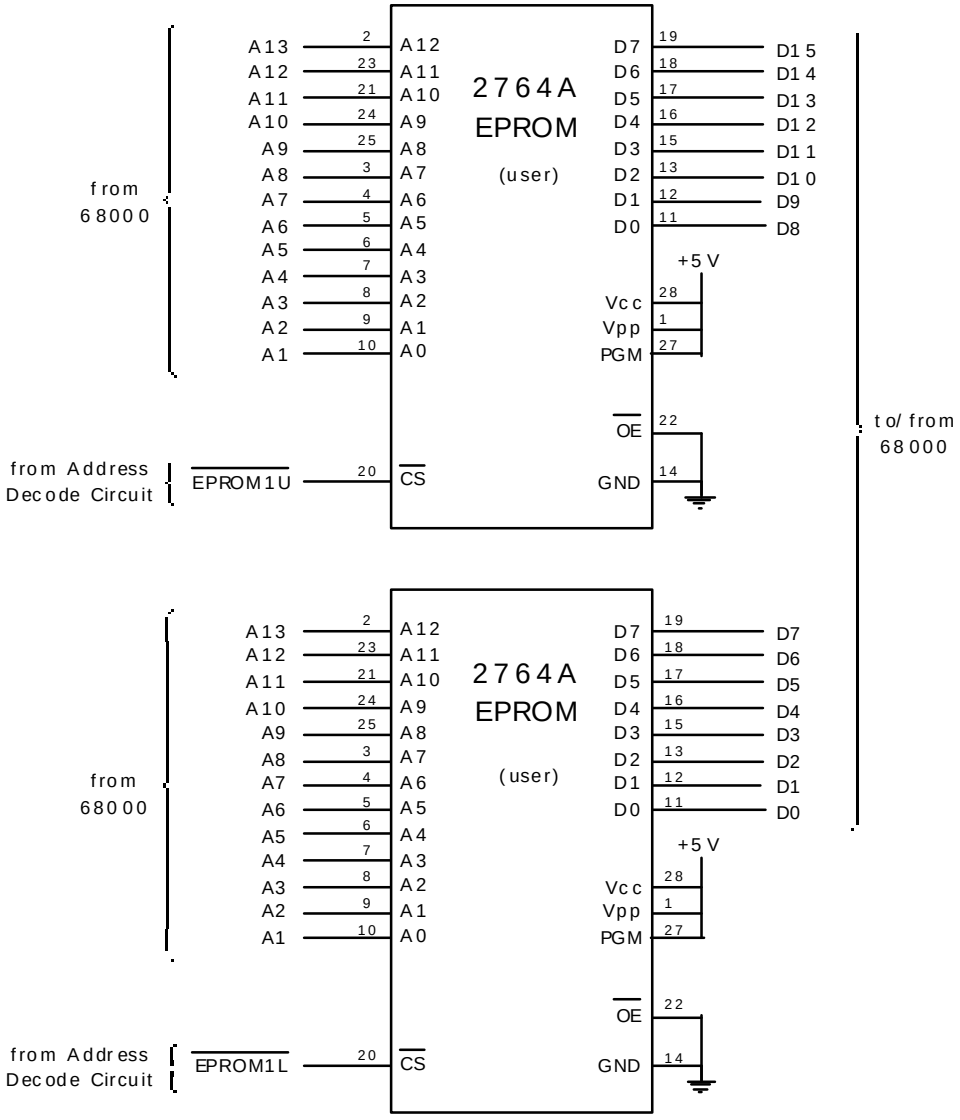
F8-10

Monitor EPROMs



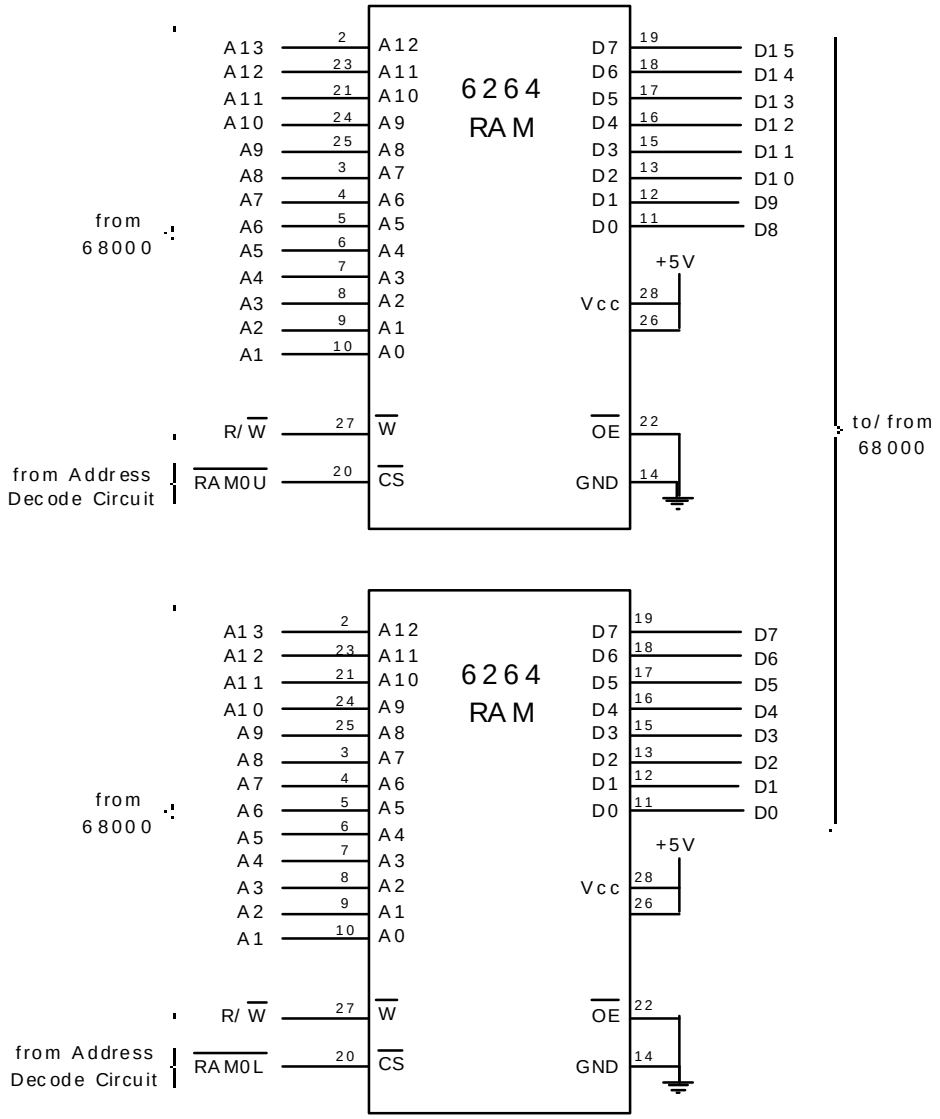
F8-12

User EPROMs



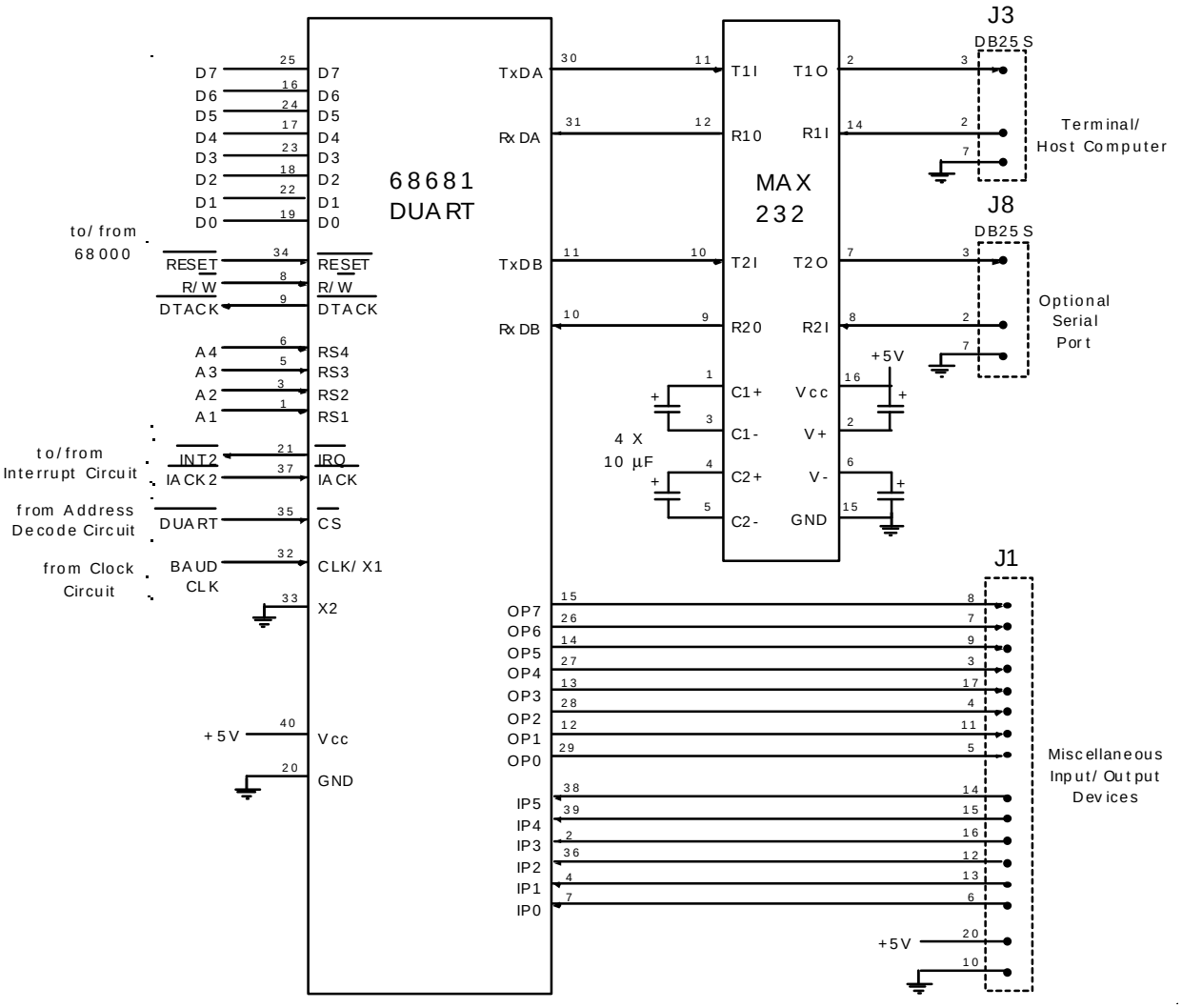
F8-13

System/User RAM



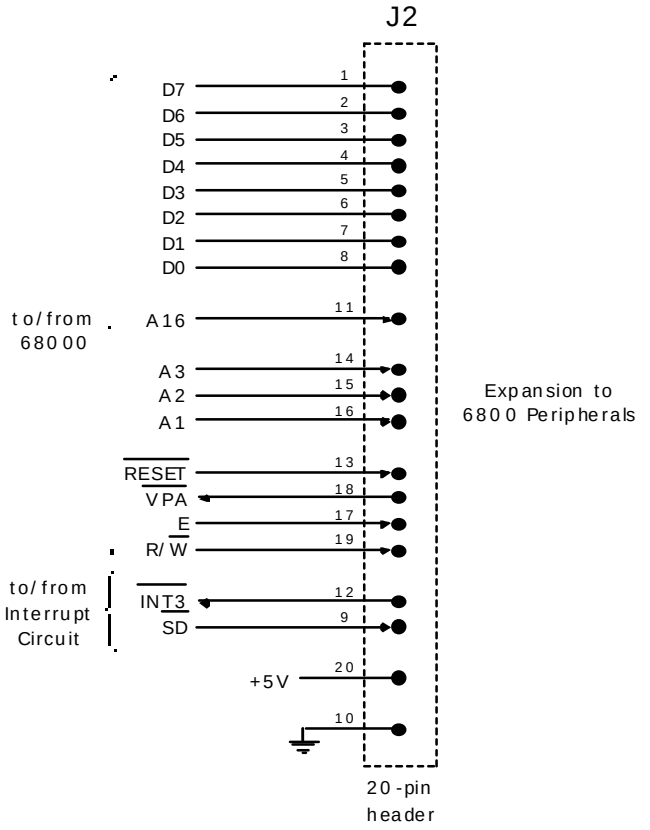
F8-14

68681 DUART



F8-15

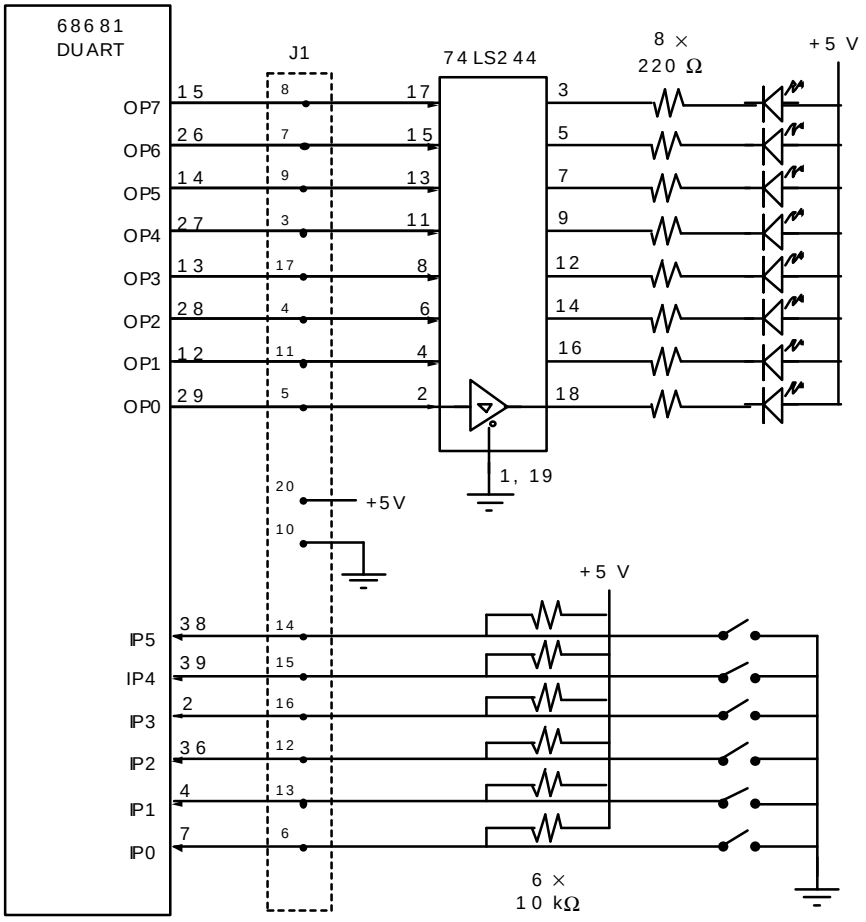
Expansion to 6800 Peripherals



F8-16

68681 Interface to LEDs and Switches

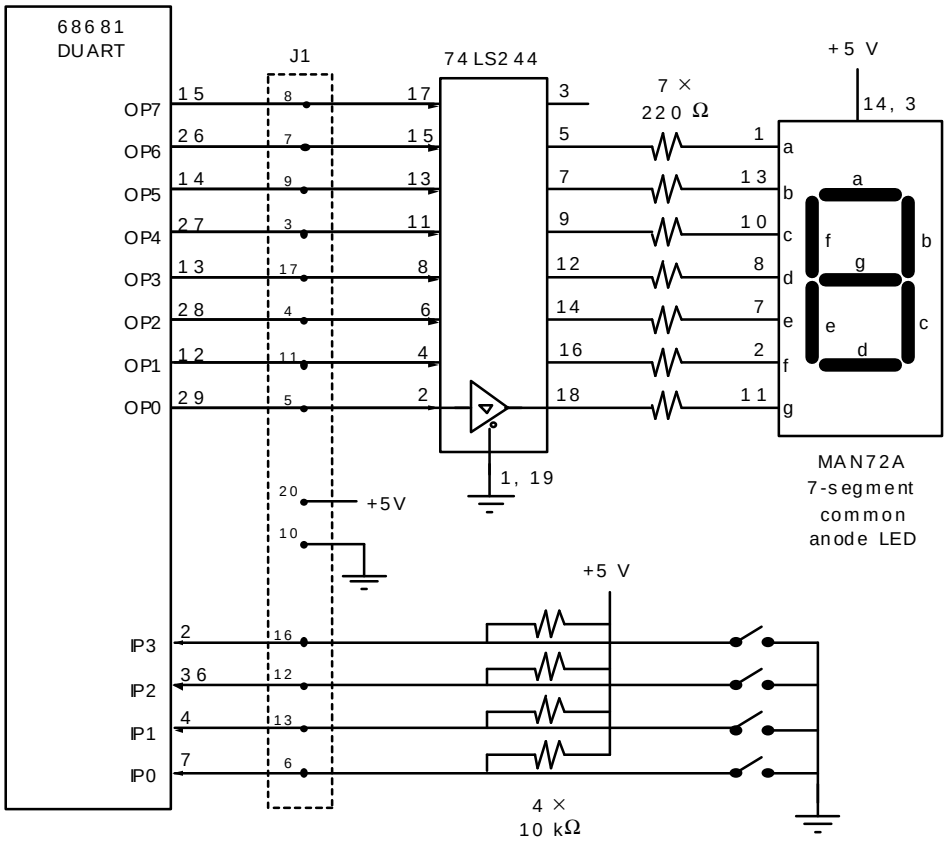
(I/O Board #1)



F9-3

Interface to Switches and 7-Segment LED

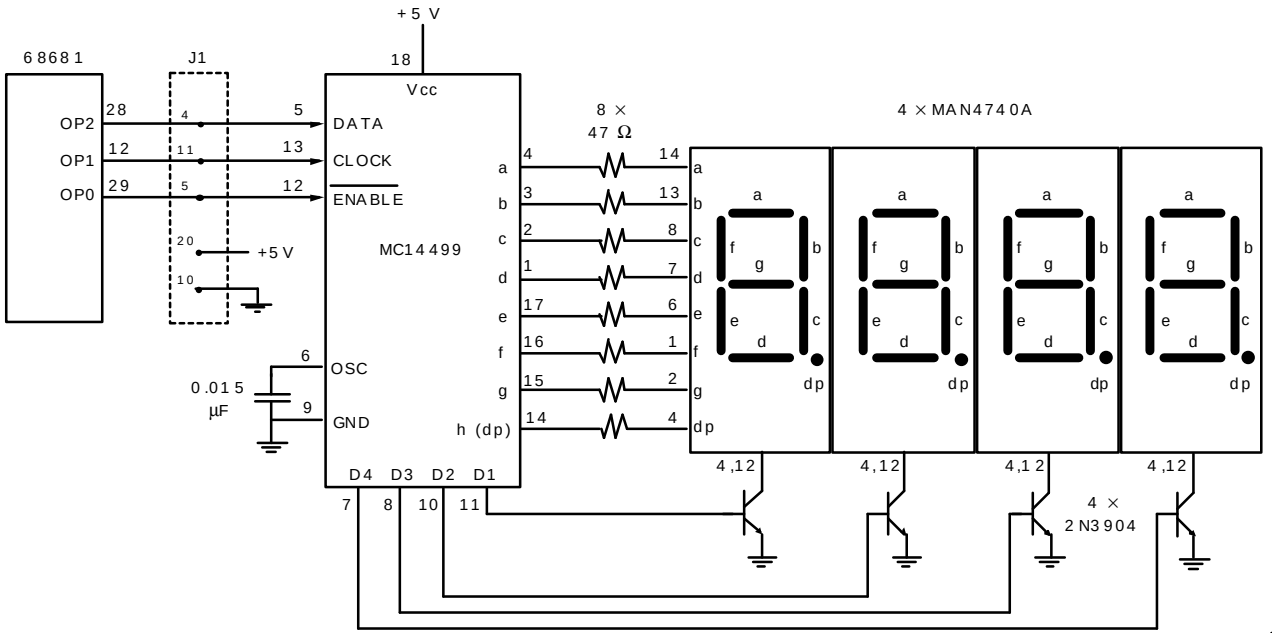
(I/O Board #2)



F9-4

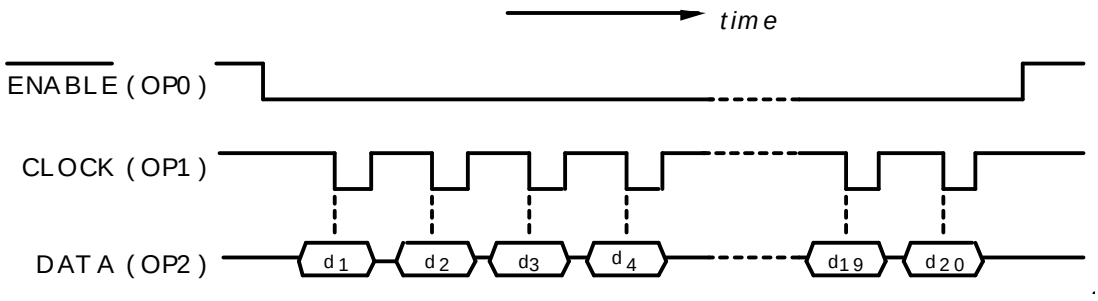
4-Digit 7-Segment Display

(I/O Board #3)



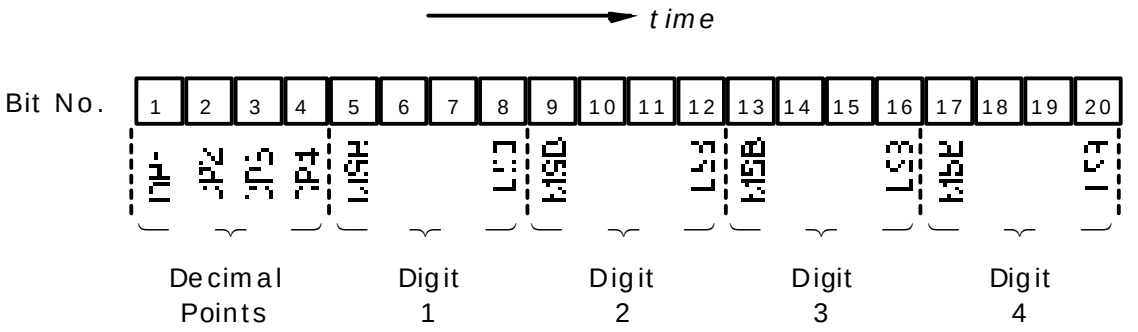
F9-6

MC14499 Timing



F9-7

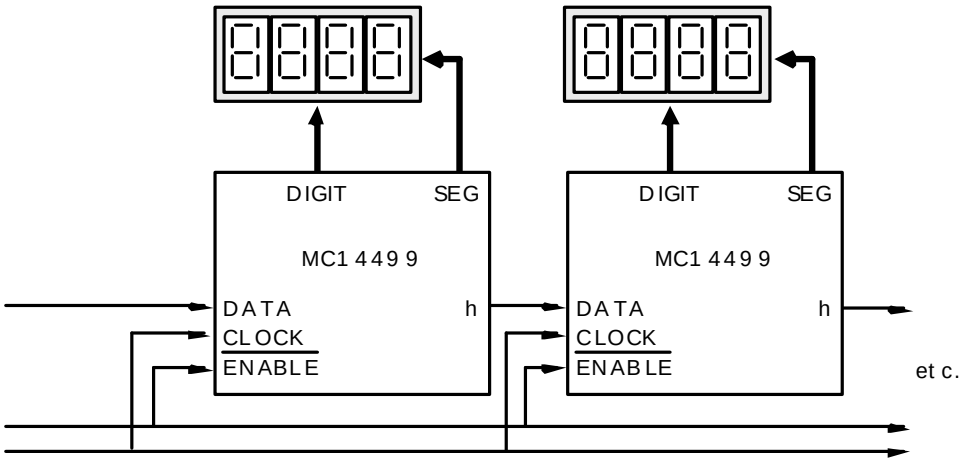
MC14499 Digit and Bit Sequence



F9-8

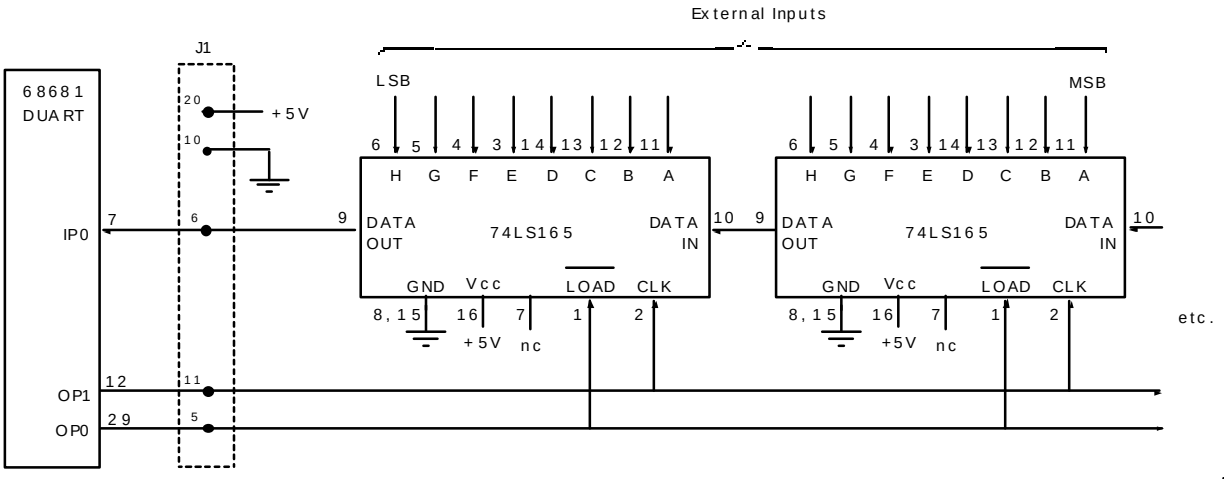
8-Digit 7-Segment Display

(I/O Board #4)



F9-9

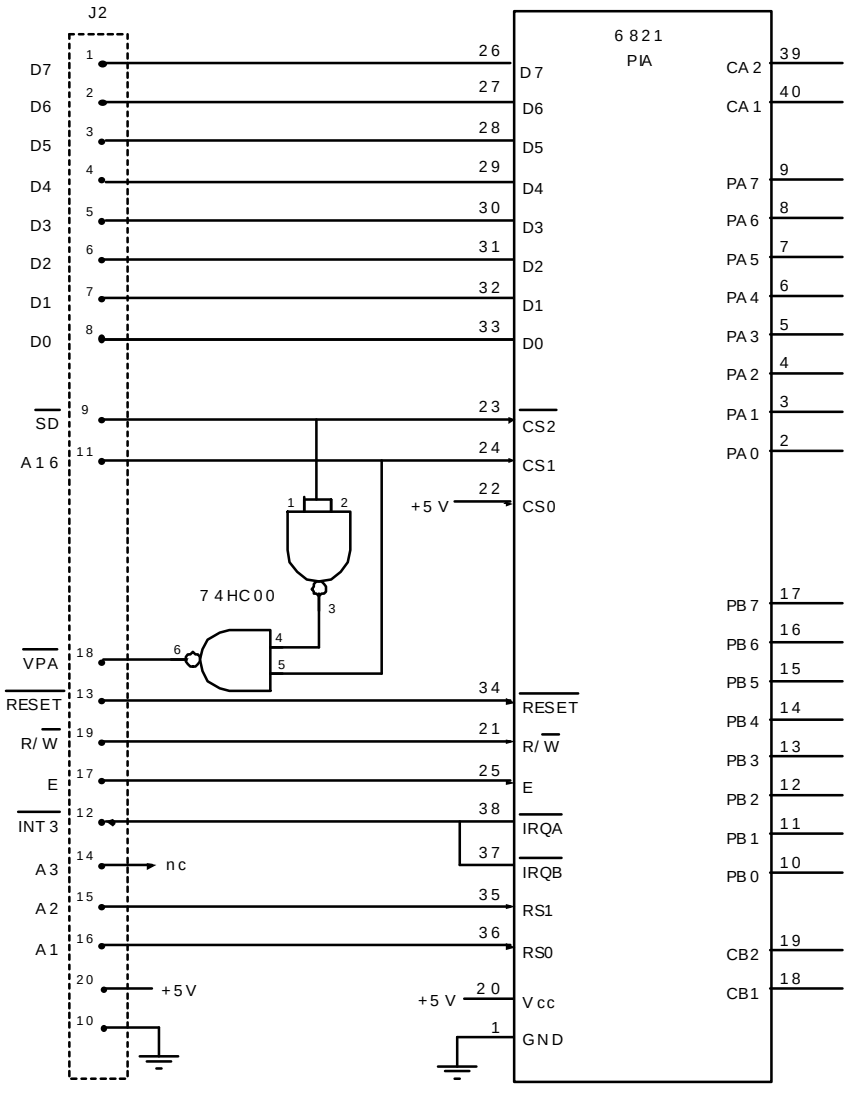
68681 Input Expansion
Using 74LS165s



F9-10

6821 Interface to the 68000

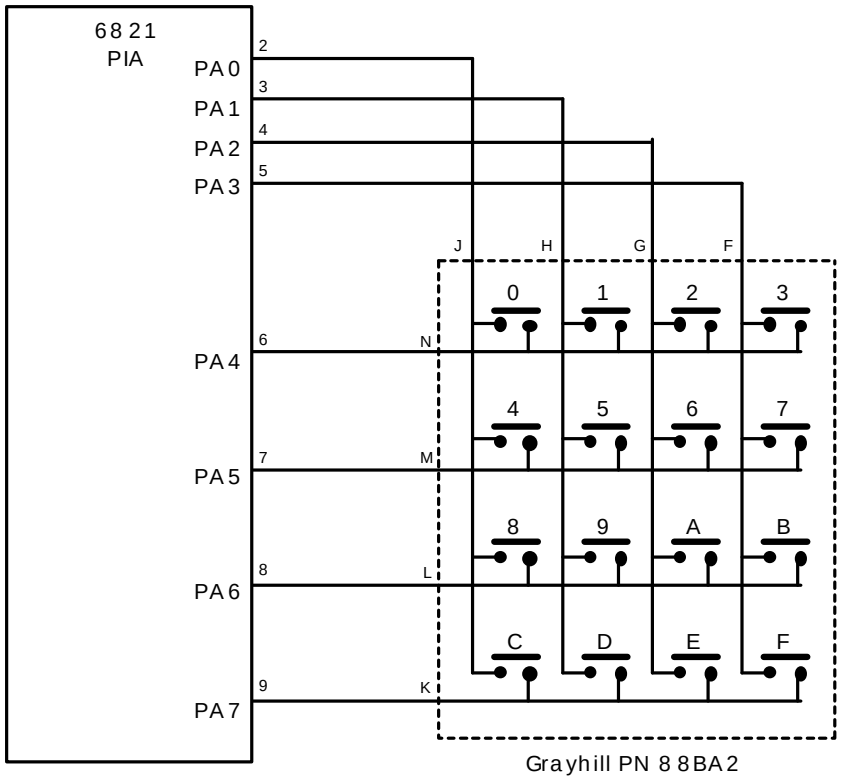
(I/O Boards #5 & #6)



F9-11

Keypad Interface to the 6821

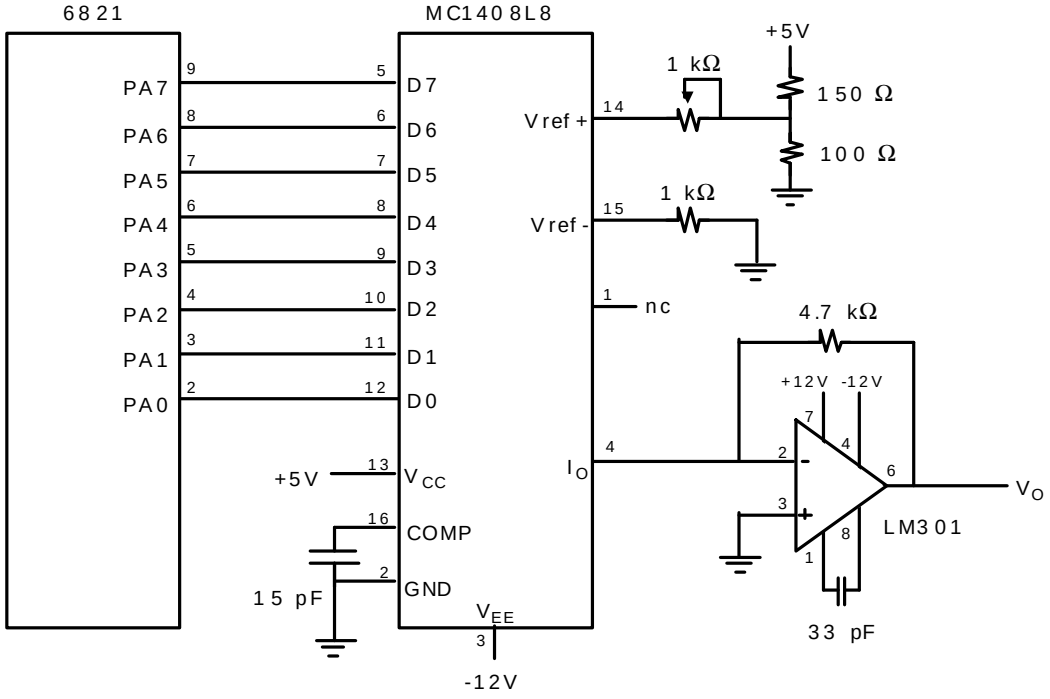
(I/O Board #5)



F9-12

Output to a MC1408L8 DAC

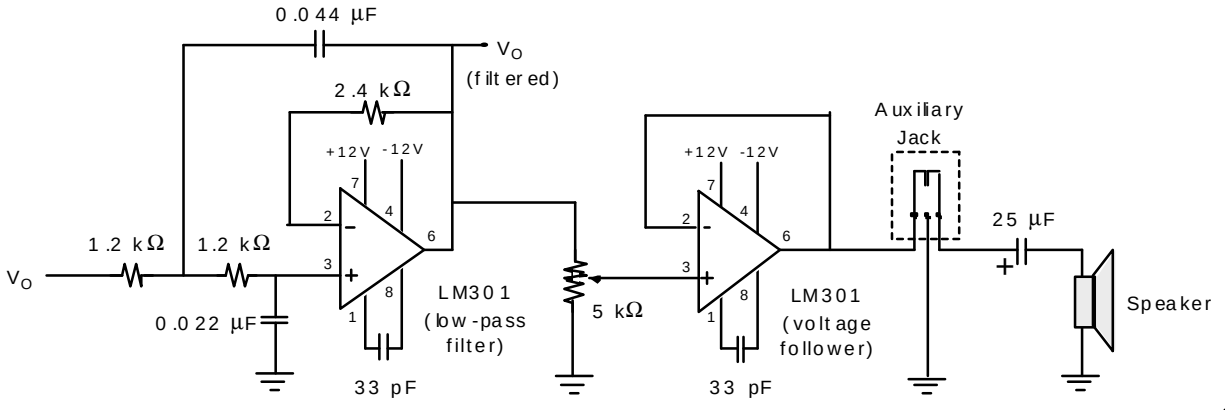
(I/O Board #6, 1 of 4)



F9-13

Low-Pass Filter and Audio Output

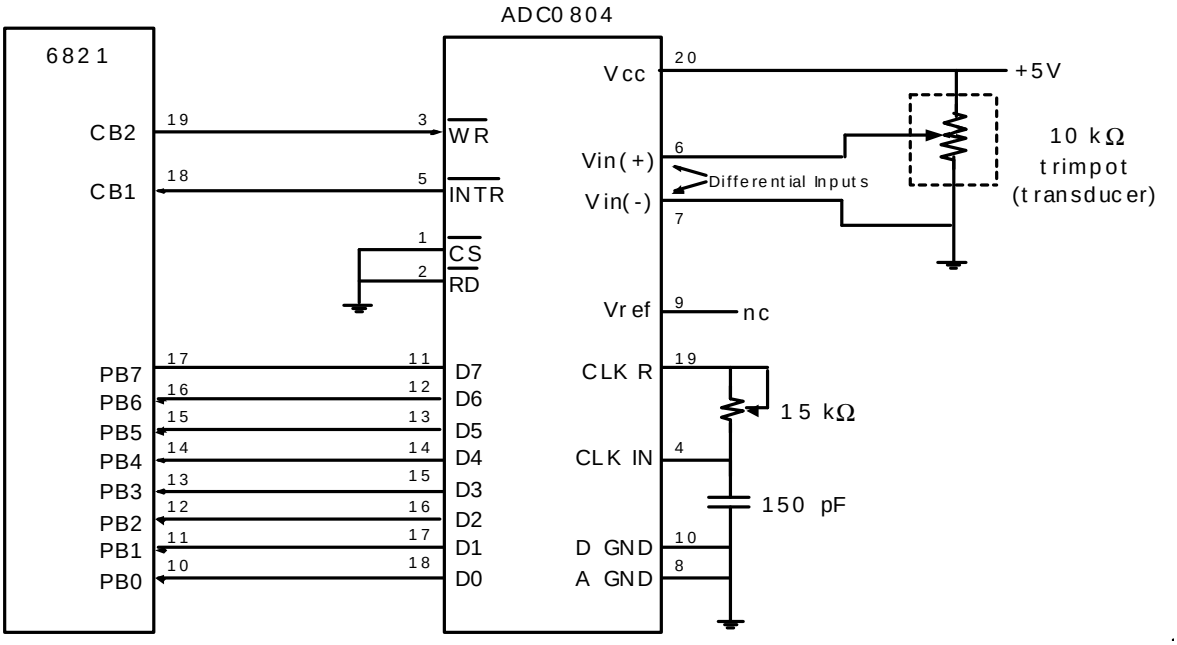
(I/O Board #6, 2 of 4)



F9-15

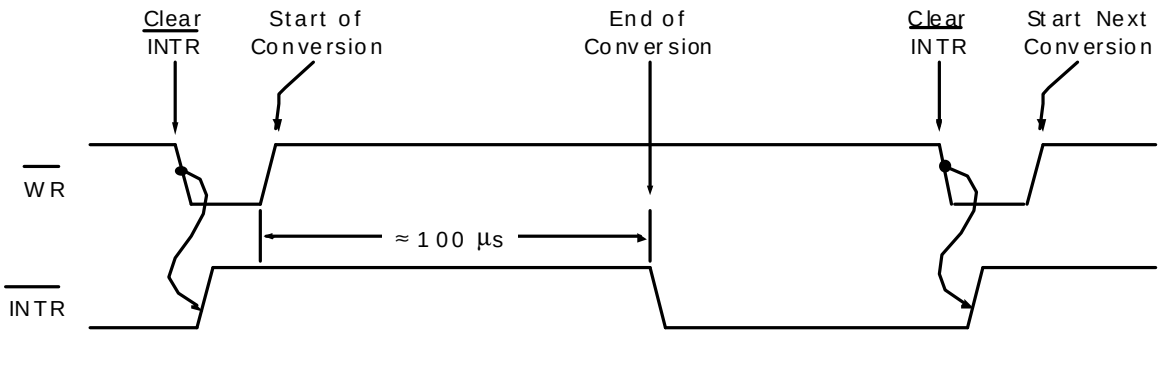
Interface to ADC0804 Analog-to-Digital Converter

(I/O Board #6, 3 of 4)



F9-16

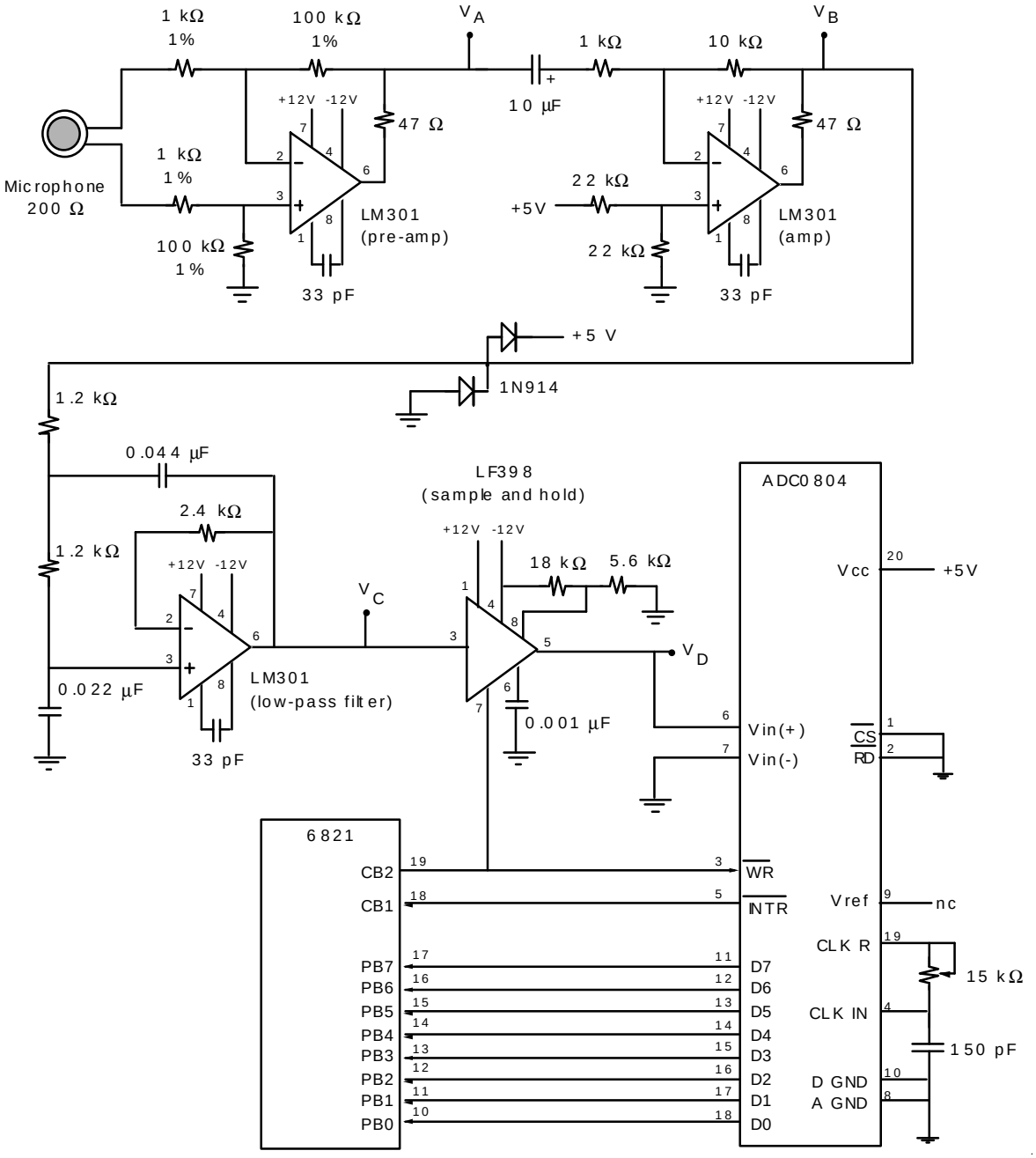
Timing for ADC0804 Conversions



F9-17

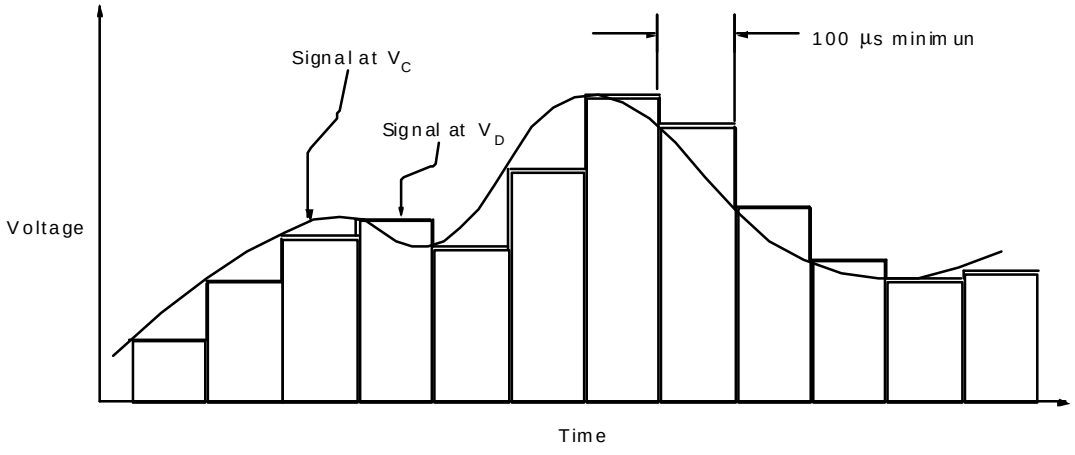
Microphone Input to the ADC0804

(I/O Board #6, 4 of 4)



F9-19

Sample-and-Hold Waveforms



F9-20

68000-Family Features

	48-pin 68008	52-pin 68008	68000	68010	68020	68030	68040
Data Bus (bits)	8	8	16	16	32	32	32
Address Bus (bits)	20	22	24	24	32	32	32
Data Cache (bytes)	–	–	–	–	–	256	4096
Instruction Cache (bytes)	–	–	–	–	256	256	4096
On-Chip Memory Management	No	No	No	No	No	Yes	Yes
On-Chip Floating-Point Unit	No	No	No	No	No	No	Yes

T10-1

Comparison of Five Recent Microprocessors†

	68040	80486	PowerPC	Pentium	Alpha 21064
Company	Motorola	Intel	IBM/Motorola	Intel	DEC
Introduced	1989	6/91	4/93	3/93	2/92
Architecture	CISC	CISC	RISC	CISC	RISC
Width (bits)	32	32	32	32	64
Registers (general/FP)	16/8	8/8	32/32	8/8	32/32
Multiprocessing Support?	No	No	Yes	Yes	Yes
Device Size (mm)	10.8 x 11.7	not available	11 x 11	17.2 x 17.2	15.3 x 12.7
Transistors (millions)	1.2	1.2	2.8	3.1	1.68
Clock (MHz)	25	50	80	66	200
SPECint 92††	21	27.9	85	67.4	130
SPECfp 92††	15	13.1	105	63.6	184
Peak Power (Watts)	6	5	9.1	16	30
Price ($US/1000 units)	$233	$432	$557	$898	$505

† Source: *IEEE Spectrum*, December 1993, p. 21

†† Integer and floating-point performance benchmarks

T10-10