US005485614A

United States Patent [19]

Kocis et al.

[11] **Patent Number: 5,485,614**

[45] **Date of Patent: Jan. 16, 1996**

[54] **COMPUTER WITH POINTING DEVICE MAPPED INTO KEYBOARD**

[75] Inventors: **Thomas J. Kocis**; **Philip D. Chidester**, both of Austin, Tex.

[73] Assignee: **Dell USA, L.P.**, Austin, Tex.

[21] Appl. No.: **278,690**

[22] Filed: **Jul. 21, 1994**

Related U.S. Application Data

[63] Continuation of Ser. No. 813,379, Dec. 23, 1991, abandoned.

[51] **Int. Cl.**[6] **H01H 13/70**

[52] **U.S. Cl.** **361/680**; 395/500; 395/893; 345/145; 345/168; 341/22

[58] **Field of Search** 395/275, 375, 395/500, 800; 361/680; 345/145, 157, 163, 168

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,680,577	7/1987	Straayer et al.	345/160
4,786,894	11/1988	Furusawa et al.	345/160
4,903,222	2/1990	Carter et al.	364/708
4,931,781	6/1990	Miyakawa	345/160
4,974,183	11/1990	Miller	364/709.12
5,041,819	8/1991	Takeda	345/160
5,124,689	6/1992	Franz et al.	345/172
5,189,403	2/1993	Franz et al.	345/172
5,198,802	3/1993	Bertram et al.	345/146
5,245,321	9/1993	Franz et al.	340/711

OTHER PUBLICATIONS

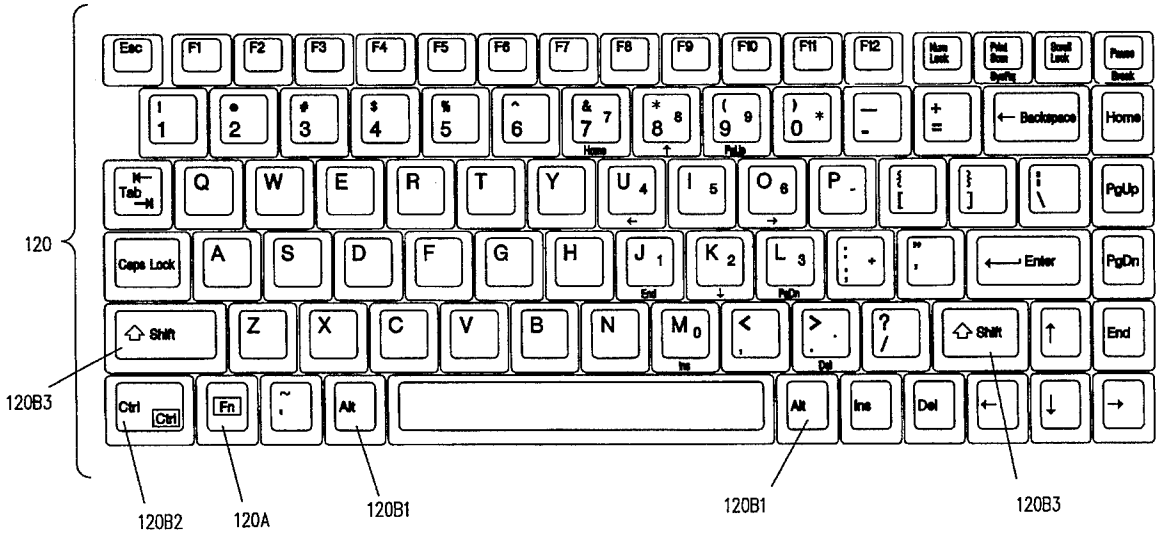
PC Magazine, Mar. 27, 1990.
WordPerfect 5.0 Manual, p. 104.

Primary Examiner—Kevin J. Teska
Assistant Examiner—Dan Fiul
Attorney, Agent, or Firm—Gary Stanford; Jeff Hood; James Huffman

[57] **ABSTRACT**

A small computer architecture in which the CPU can receive multiple kinds of interrupt signals, including one kind which is assigned to indicate the occurrence of a keystroke input and another kind which is assigned to indicate the occurrence of a pointing device input. However, the computer does not include any pointing device as such. Instead, the keyboard microprocessor (i.e. a microprocessor other than the CPU) monitors user keystrokes to the computer's keyboard, and: in response to simple keystrokes, or keystroke combinations which include one of the basic chording keys, the keyboard microprocessor sends a keystroke interrupt to the CPU; and in response to keystroke combinations which include an additional chording key, the keyboard microprocessor sends a pointing-device interrupt to the CPU, and provides a data output corresponding to an emulated pointing-device movement.

20 Claims, 72 Drawing Sheets



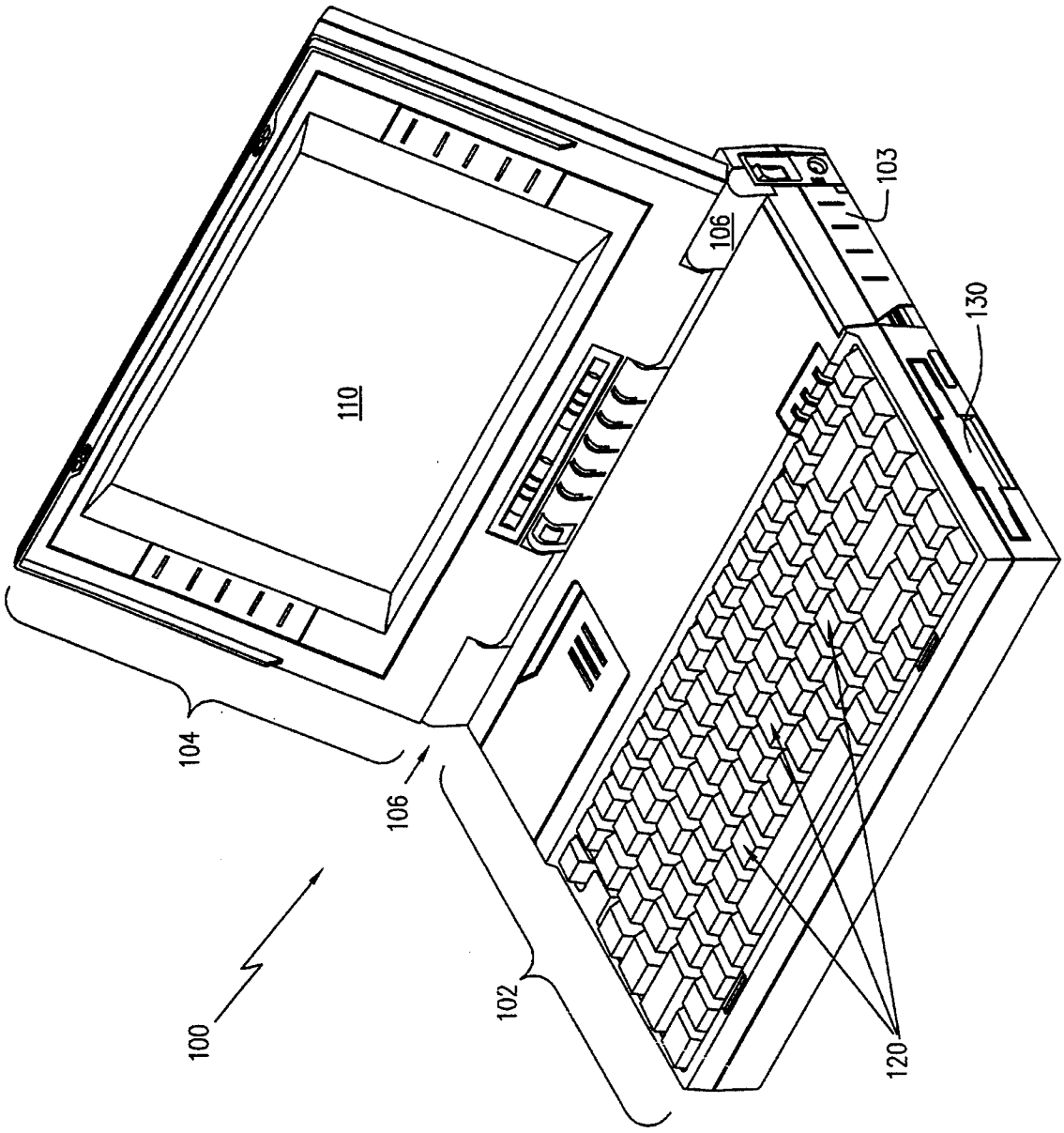


FIG. 1

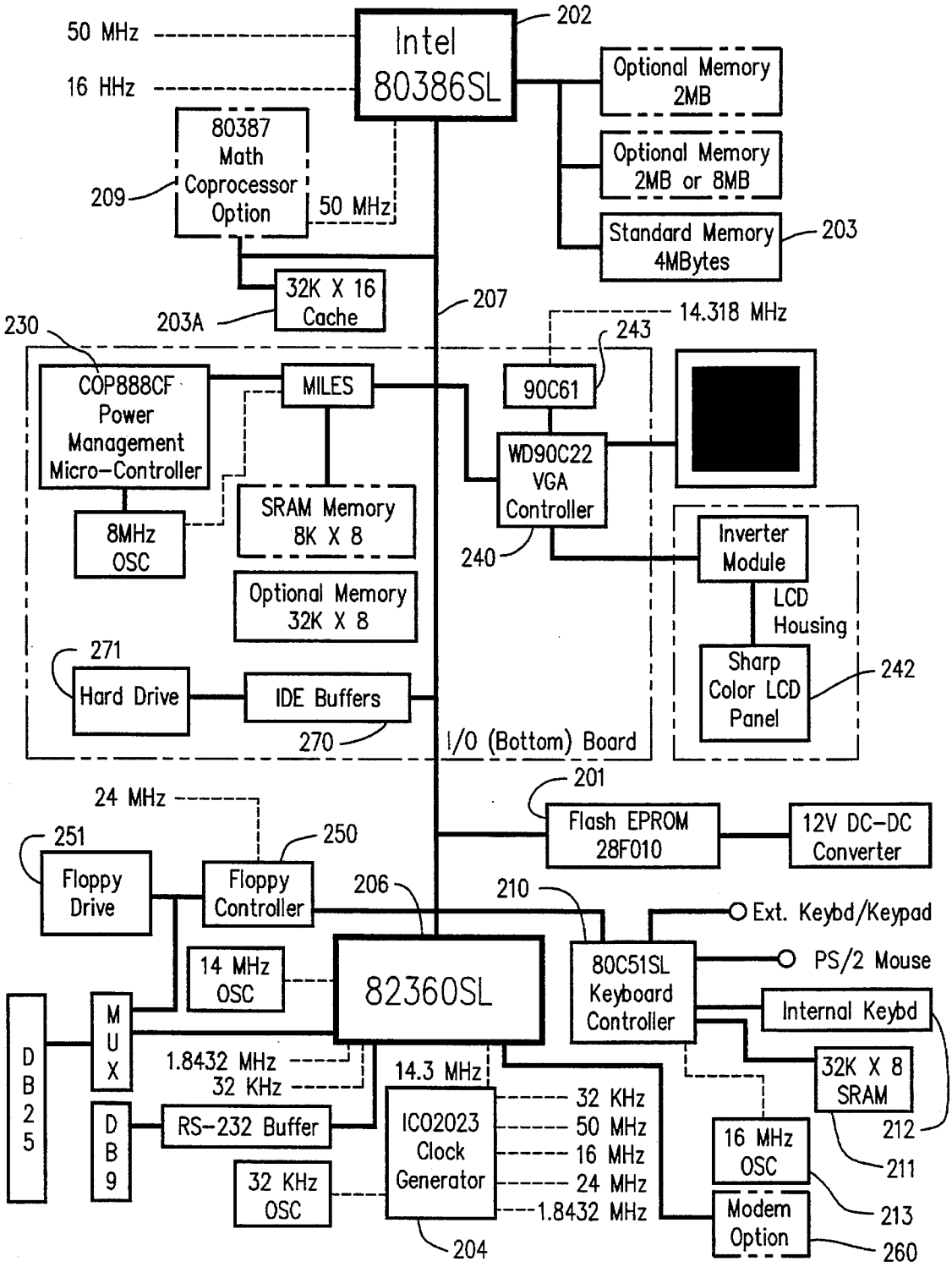

50 MHz
16 HHz
Intel 80386SL
202
Optional Memory 2MB
Optional Memory 2MB or 8MB
Standard Memory 4MBytes
203
80387 Math Coprocessor Option
209
50 MHz
32K X 16 Cache
203A
207
14.318 MHz
243
230
COP888CF Power Management Micro-Controller
MILES
90C61
WD90C22 VGA Controller
240
8MHz OSC
SRAM Memory 8K X 8
Optional Memory 32K X 8
Inverter Module
LCD Housing
Sharp Color LCD Panel
242
271
Hard Drive
IDE Buffers
270
I/O (Bottom) Board
201
Flash EPROM 28F010
12V DC-DC Converter
24 MHz
251
Floppy Drive
250
Floppy Controller
206
210
Ext. Keybd/Keypad
PS/2 Mouse
14 MHz OSC
82360SL
80C51SL Keyboard Controller
Internal Keybd
DB25
MUX
1.8432 MHz
32 KHz
14.3 MHz
32K X 8 SRAM
212
DB9
RS-232 Buffer
ICO2023 Clock Generator
32 KHz
50 MHz
16 MHz
24 MHz
1.8432 MHz
16 MHz OSC
211
32 KHz OSC
Modem Option
213
204
260

FIG. 2

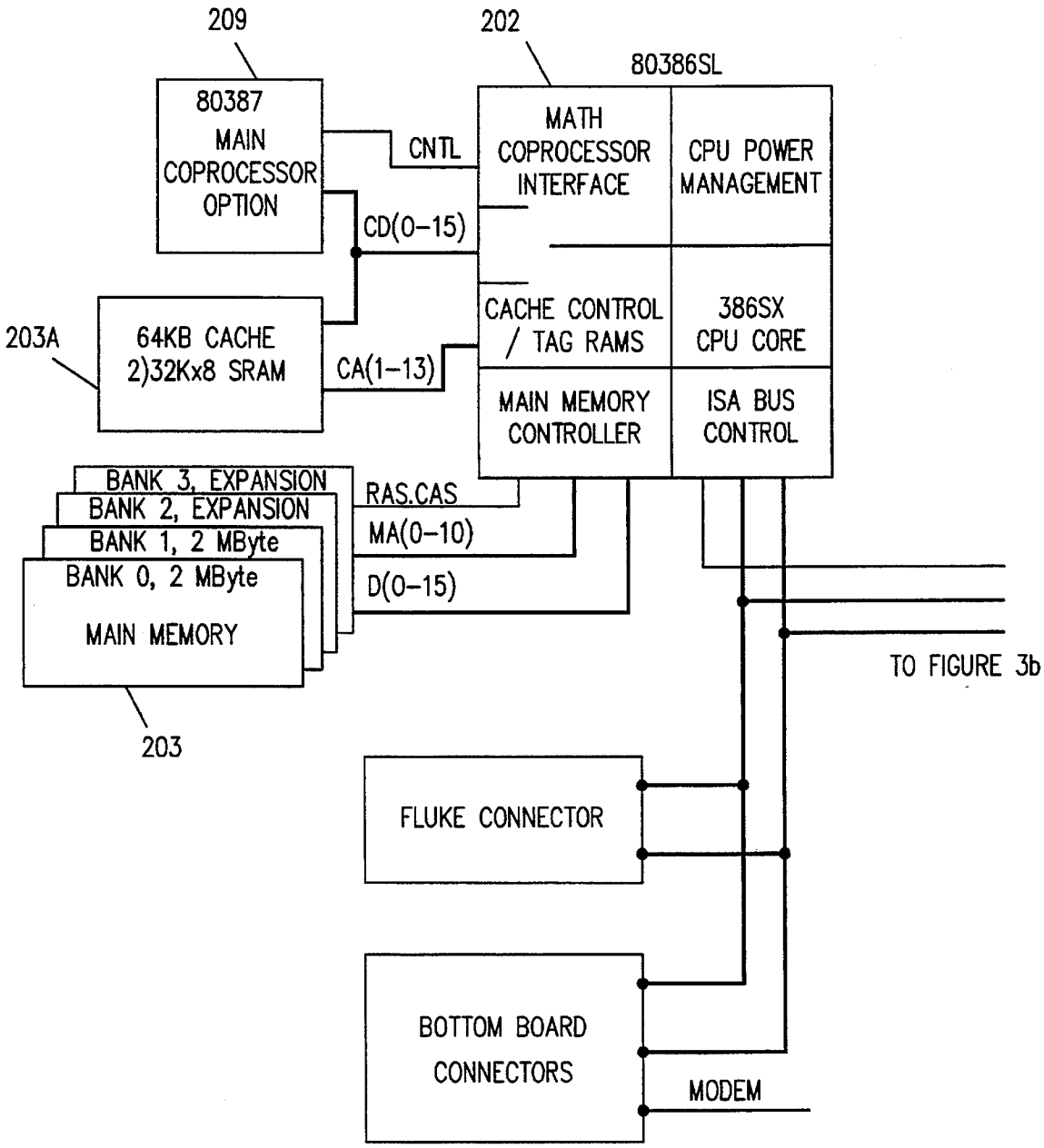

209
202
80386SL
80387
MAIN
COPROCESSOR
OPTION
CNTL
MATH
COPROCESSOR
INTERFACE
CPU POWER
MANAGEMENT
CD(0–15)
203A
64KB CACHE
2)32Kx8 SRAM
CA(1–13)
CACHE CONTROL
/ TAG RAMS
386SX
CPU CORE
MAIN MEMORY
CONTROLLER
ISA BUS
CONTROL
BANK 3, EXPANSION
BANK 2, EXPANSION
BANK 1, 2 MByte
BANK 0, 2 MByte
MAIN MEMORY
RAS.CAS
MA(0–10)
D(0–15)
TO FIGURE 3b
203
FLUKE CONNECTOR
BOTTOM BOARD
CONNECTORS
MODEM

FIG. 3a

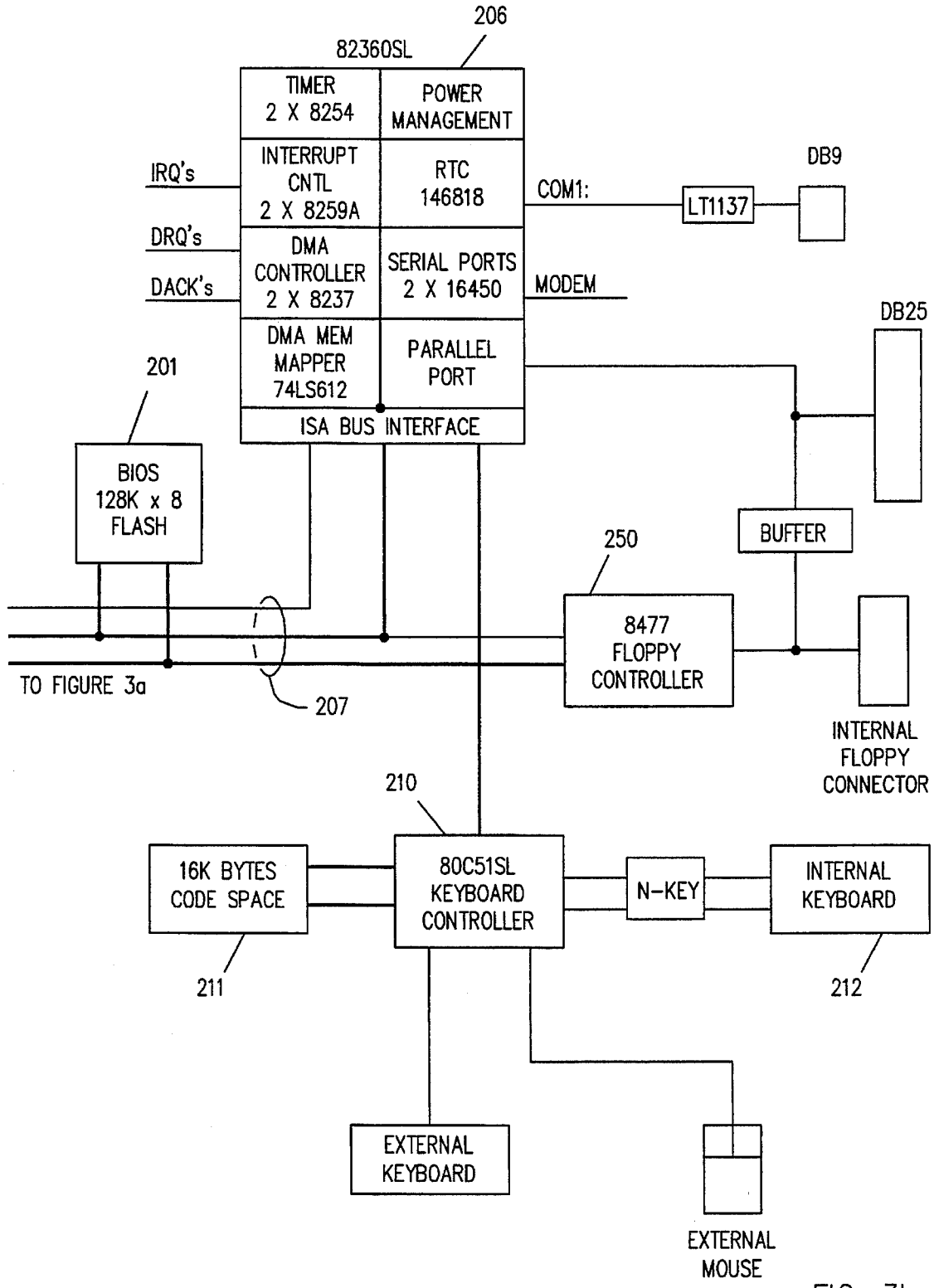

206
82360SL
TIMER
2 X 8254
POWER
MANAGEMENT
IRQ's
INTERRUPT
CNTL
2 X 8259A
RTC
146818
COM1:
LT1137
DB9
DRQ's
DMA
CONTROLLER
2 X 8237
SERIAL PORTS
2 X 16450
DACK's
MODEM
DB25
DMA MEM
MAPPER
74LS612
PARALLEL
PORT
201
ISA BUS INTERFACE
BIOS
128K x 8
FLASH
BUFFER
250
8477
FLOPPY
CONTROLLER
TO FIGURE 3a
207
INTERNAL
FLOPPY
CONNECTOR
210
16K BYTES
CODE SPACE
80C51SL
KEYBOARD
CONTROLLER
N-KEY
INTERNAL
KEYBOARD
211
212
EXTERNAL
KEYBOARD
EXTERNAL
MOUSE

FIG. 3b

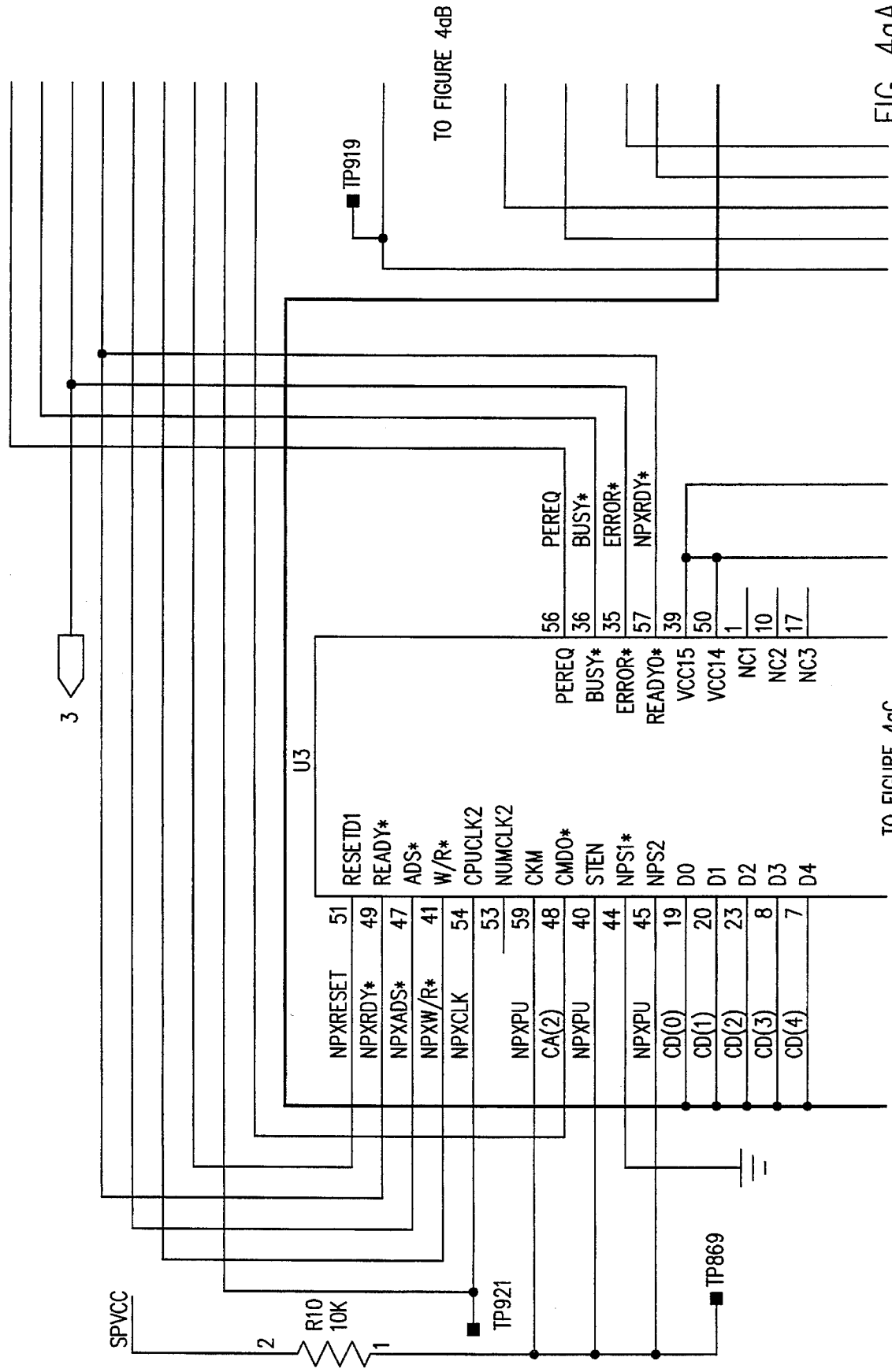

TO FIGURE 4aB
TP919
3
U3
PEREQ
BUSY*
ERROR*
NPXRDY*
56
36
35
57
39
50
1
10
17
PEREQ
BUSY*
ERROR*
READY0*
VCC15
VCC14
NC1
NC2
NC3
TO FIGURE 4aC
RESETD1
READY*
ADS*
W/R*
CPUCLK2
NUMCLK2
CKM
CMD0*
STEN
NPS1*
NPS2
D0
D1
D2
D3
D4
51
49
47
41
54
53
59
48
40
44
45
19
20
23
8
7
NPXRESET
NPXRDY*
NPXADS*
NPXW/R*
NPXCLK
NPXPU
CA(2)
NPXPU
NPXPU
CD(0)
CD(1)
CD(2)
CD(3)
CD(4)
SPVCC
R10
10K
2
1
TP921
TP869

FIG. 4aA

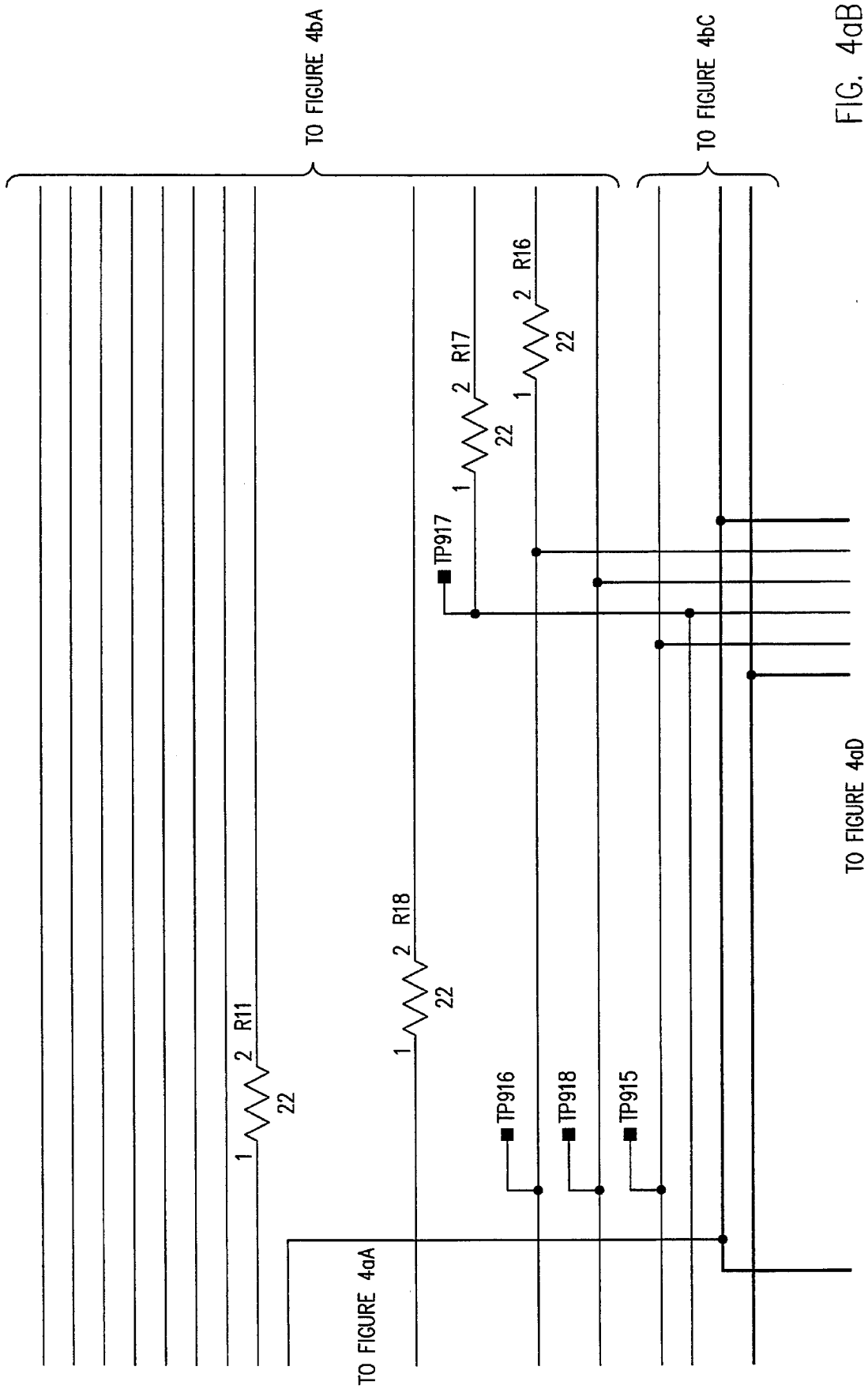

TO FIGURE 4bA
TO FIGURE 4bC
TO FIGURE 4aD
TO FIGURE 4aA
R11
R18
R17
R16
22
TP916
TP918
TP915
TP917
1
2

FIG. 4aB

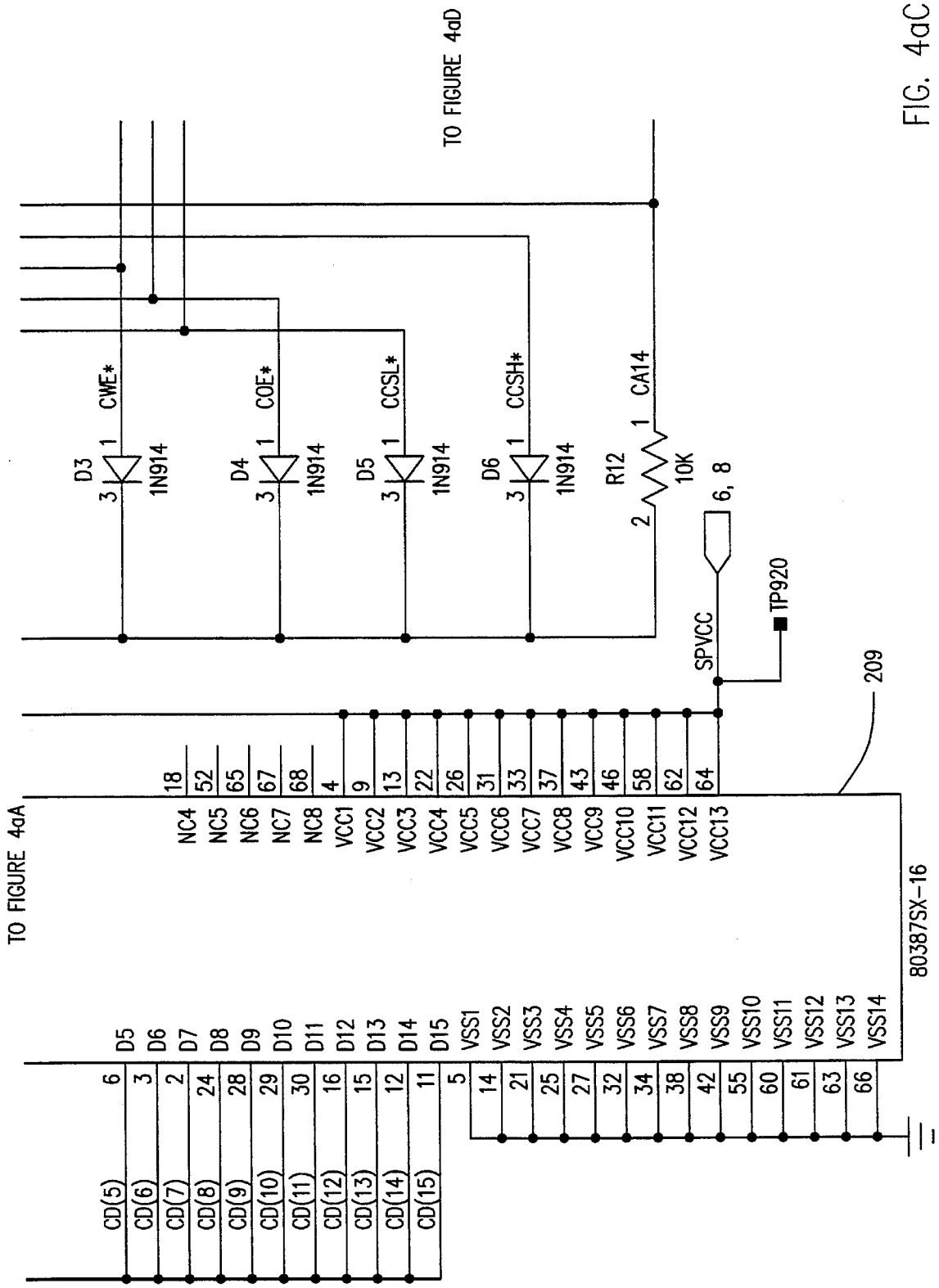

TO FIGURE 4aD
TO FIGURE 4aA
D3
1N914
CWE*
D4
1N914
COE*
D5
1N914
CCSL*
D6
1N914
CCSH*
R12
10K
CA14
6, 8
TP920
SPVCC
209
80387SX-16
NC4
NC5
NC6
NC7
NC8
VCC1
VCC2
VCC3
VCC4
VCC5
VCC6
VCC7
VCC8
VCC9
VCC10
VCC11
VCC12
VCC13
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
VSS1
VSS2
VSS3
VSS4
VSS5
VSS6
VSS7
VSS8
VSS9
VSS10
VSS11
VSS12
VSS13
VSS14
CD(5)
CD(6)
CD(7)
CD(8)
CD(9)
CD(10)
CD(11)
CD(12)
CD(13)
CD(14)
CD(15)

FIG. 4aC

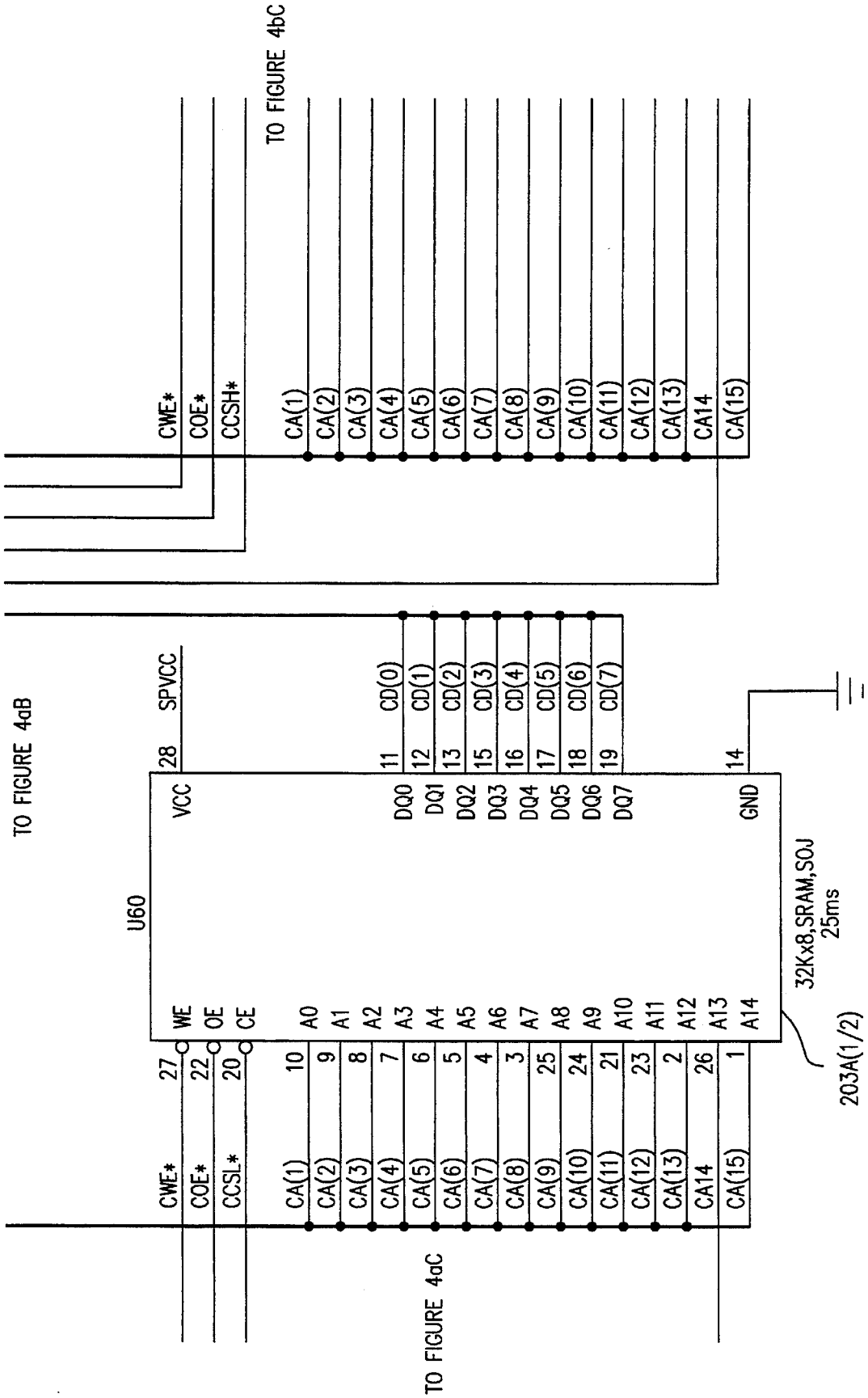

TO FIGURE 4bC
CWE*
COE*
CCSH*
CA(1)
CA(2)
CA(3)
CA(4)
CA(5)
CA(6)
CA(7)
CA(8)
CA(9)
CA(10)
CA(11)
CA(12)
CA(13)
CA14
CA(15)
TO FIGURE 4aB
SPVCC
28
VCC
U60
11 CD(0)
12 CD(1)
13 CD(2)
15 CD(3)
16 CD(4)
17 CD(5)
18 CD(6)
19 CD(7)
DQ0
DQ1
DQ2
DQ3
DQ4
DQ5
DQ6
DQ7
14
GND
32Kx8,SRAM,SOJ
25ns
203A(1/2)
WE
OE
CE
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
27
22
20
10
9
8
7
6
5
4
3
25
24
21
23
2
26
1
CWE*
COE*
CCSL*
CA(1)
CA(2)
CA(3)
CA(4)
CA(5)
CA(6)
CA(7)
CA(8)
CA(9)
CA(10)
CA(11)
CA(12)
CA(13)
CA14
CA(15)
TO FIGURE 4aC

FIG. 4aD

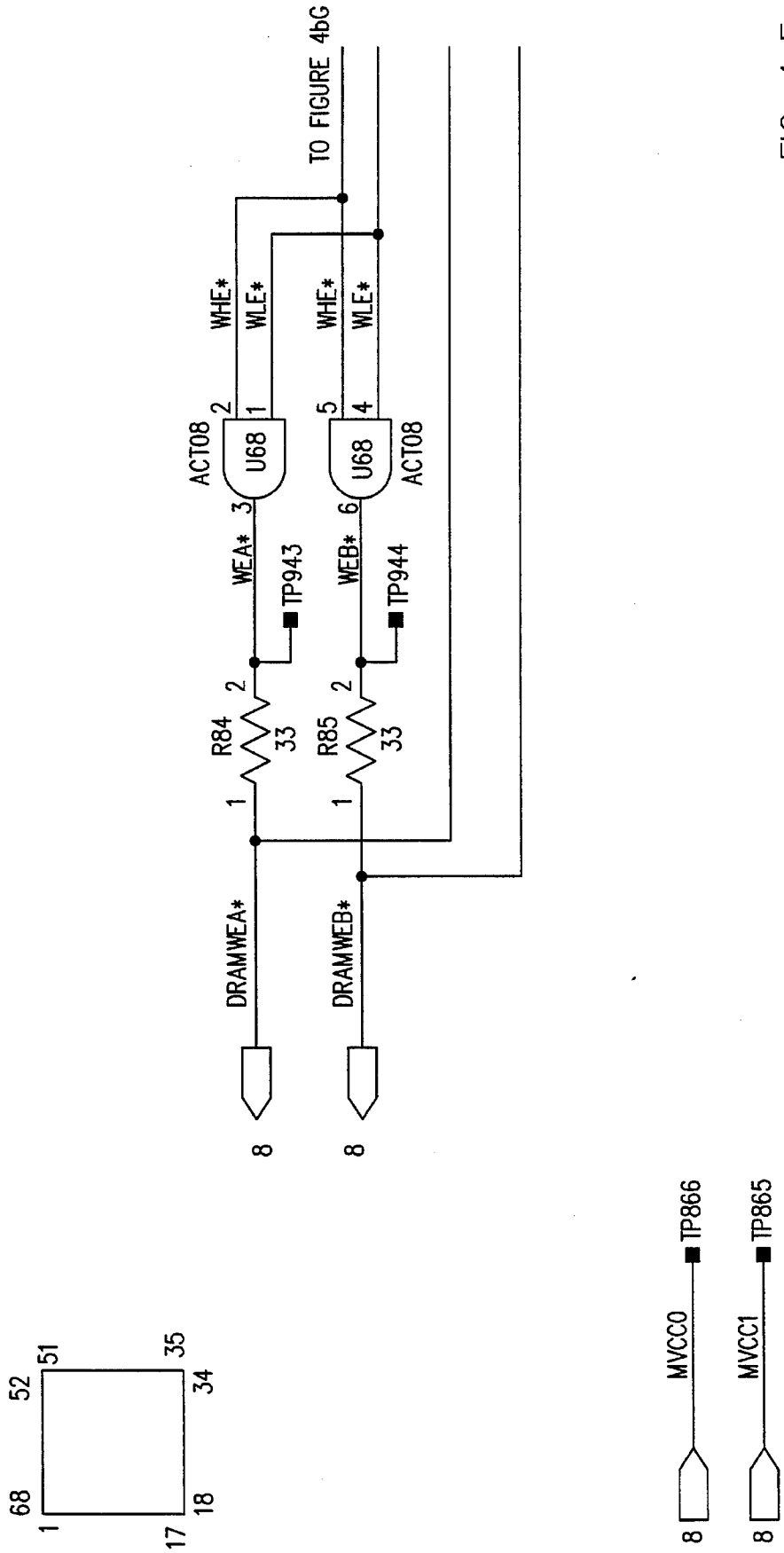

TO FIGURE 4bG
WHE*
WLE*
WHE*
WLE*
ACT08
U68
U68
ACT08
WEA*
WEB*
TP943
TP944
R84
33
R85
33
DRAMWEA*
DRAMWEB*
8
8
MVCC0
MVCC1
TP866
TP865
8
8
68
1
17
18
52
51
35
34

FIG. 4aE

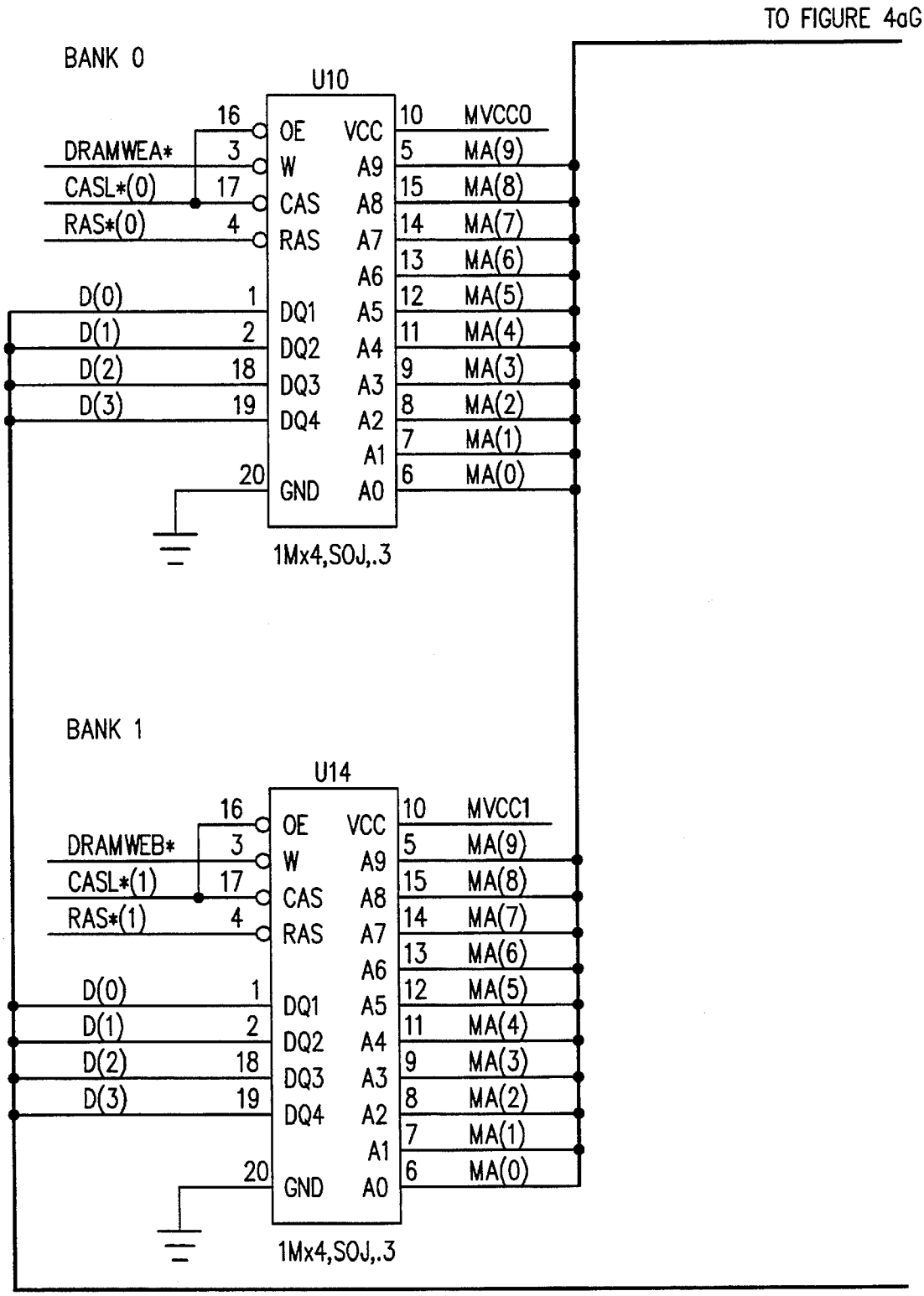

TO FIGURE 4aG
BANK 0
U10
16
OE
VCC
10
MVCC0
DRAMWEA*
3
W
A9
5
MA(9)
CASL*(0)
17
CAS
A8
15
MA(8)
RAS*(0)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(0)
1
DQ1
A5
12
MA(5)
D(1)
2
DQ2
A4
11
MA(4)
D(2)
18
DQ3
A3
9
MA(3)
D(3)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3
BANK 1
U14
16
OE
VCC
10
MVCC1
DRAMWEB*
3
W
A9
5
MA(9)
CASL*(1)
17
CAS
A8
15
MA(8)
RAS*(1)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(0)
1
DQ1
A5
12
MA(5)
D(1)
2
DQ2
A4
11
MA(4)
D(2)
18
DQ3
A3
9
MA(3)
D(3)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3

FIG. 4aF

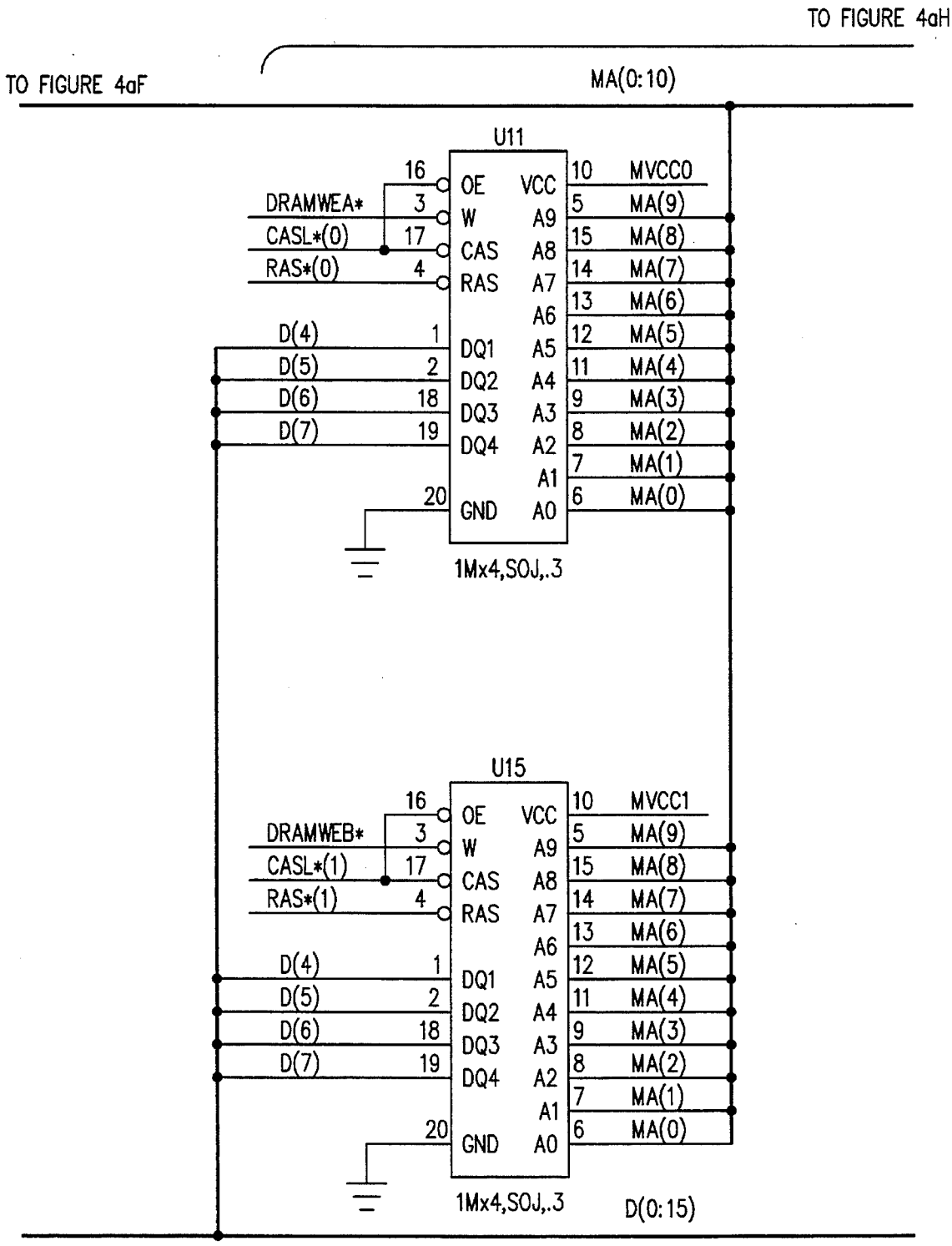

TO FIGURE 4aH
TO FIGURE 4aF
MA(0:10)
U11
16
OE
VCC
10
MVCC0
DRAMWEA*
3
W
A9
5
MA(9)
CASL*(0)
17
CAS
A8
15
MA(8)
RAS*(0)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(4)
1
DQ1
A5
12
MA(5)
D(5)
2
DQ2
A4
11
MA(4)
D(6)
18
DQ3
A3
9
MA(3)
D(7)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3
U15
16
OE
VCC
10
MVCC1
DRAMWEB*
3
W
A9
5
MA(9)
CASL*(1)
17
CAS
A8
15
MA(8)
RAS*(1)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(4)
1
DQ1
A5
12
MA(5)
D(5)
2
DQ2
A4
11
MA(4)
D(6)
18
DQ3
A3
9
MA(3)
D(7)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3
D(0:15)

FIG. 4aG

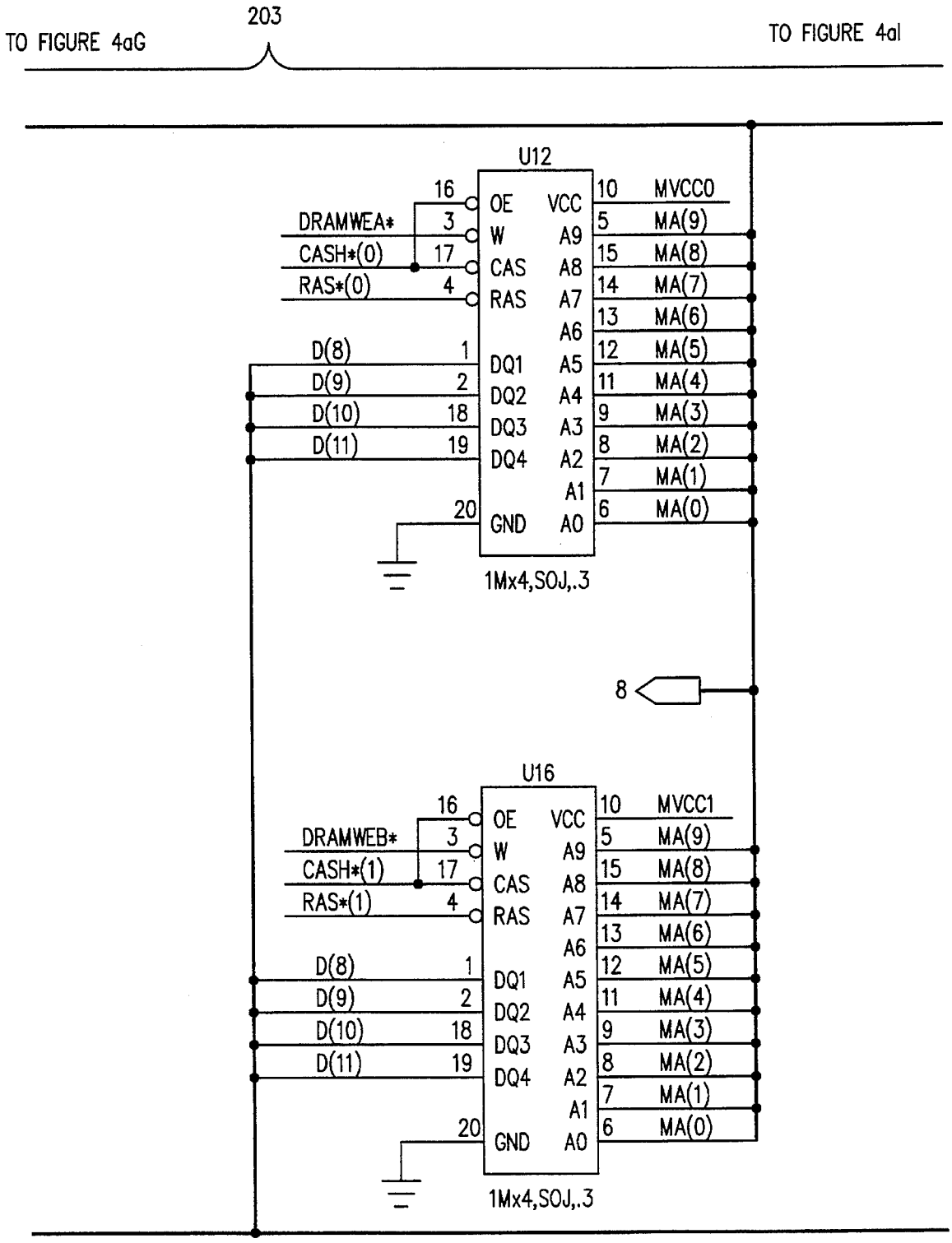

203
TO FIGURE 4aG
TO FIGURE 4al
U12
DRAMWEA*
CASH*(0)
RAS*(0)
D(8)
D(9)
D(10)
D(11)
OE
W
CAS
RAS
DQ1
DQ2
DQ3
DQ4
GND
VCC
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
MVCC0
MA(9)
MA(8)
MA(7)
MA(6)
MA(5)
MA(4)
MA(3)
MA(2)
MA(1)
MA(0)
1Mx4,SOJ,.3
8
U16
DRAMWEB*
CASH*(1)
RAS*(1)
D(8)
D(9)
D(10)
D(11)
MVCC1
MA(9)
MA(8)
MA(7)
MA(6)
MA(5)
MA(4)
MA(3)
MA(2)
MA(1)
MA(0)
1Mx4,SOJ,.3

FIG. 4aH

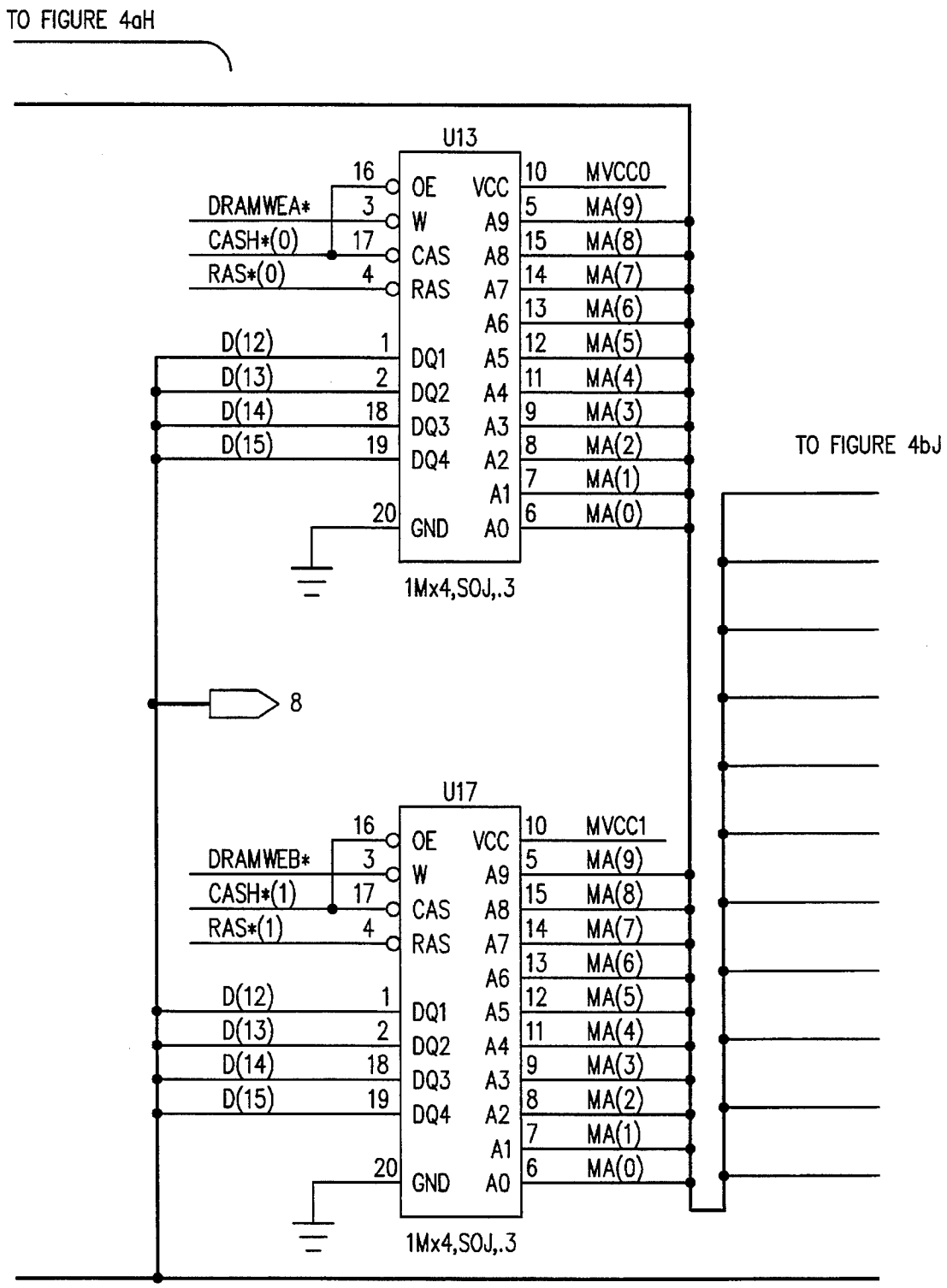

TO FIGURE 4aH
U13
16
OE
VCC
10
MVCC0
DRAMWEA*
3
W
A9
5
MA(9)
CASH*(0)
17
CAS
A8
15
MA(8)
RAS*(0)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(12)
1
DQ1
A5
12
MA(5)
D(13)
2
DQ2
A4
11
MA(4)
D(14)
18
DQ3
A3
9
MA(3)
D(15)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3
TO FIGURE 4bJ
8
U17
16
OE
VCC
10
MVCC1
DRAMWEB*
3
W
A9
5
MA(9)
CASH*(1)
17
CAS
A8
15
MA(8)
RAS*(1)
4
RAS
A7
14
MA(7)
A6
13
MA(6)
D(12)
1
DQ1
A5
12
MA(5)
D(13)
2
DQ2
A4
11
MA(4)
D(14)
18
DQ3
A3
9
MA(3)
D(15)
19
DQ4
A2
8
MA(2)
A1
7
MA(1)
20
GND
A0
6
MA(0)
1Mx4,SOJ,.3

FIG. 4al

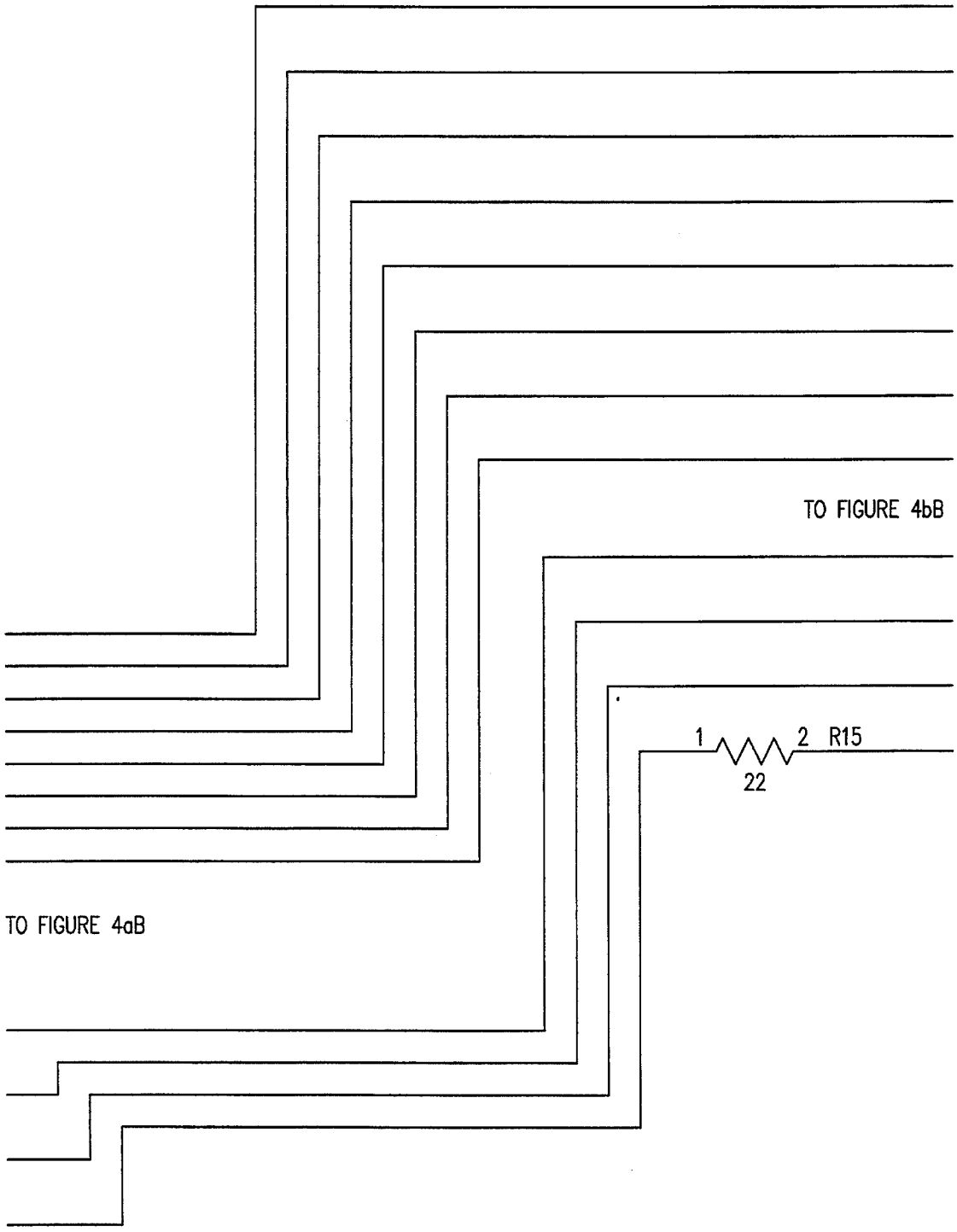


FIG. 4bA

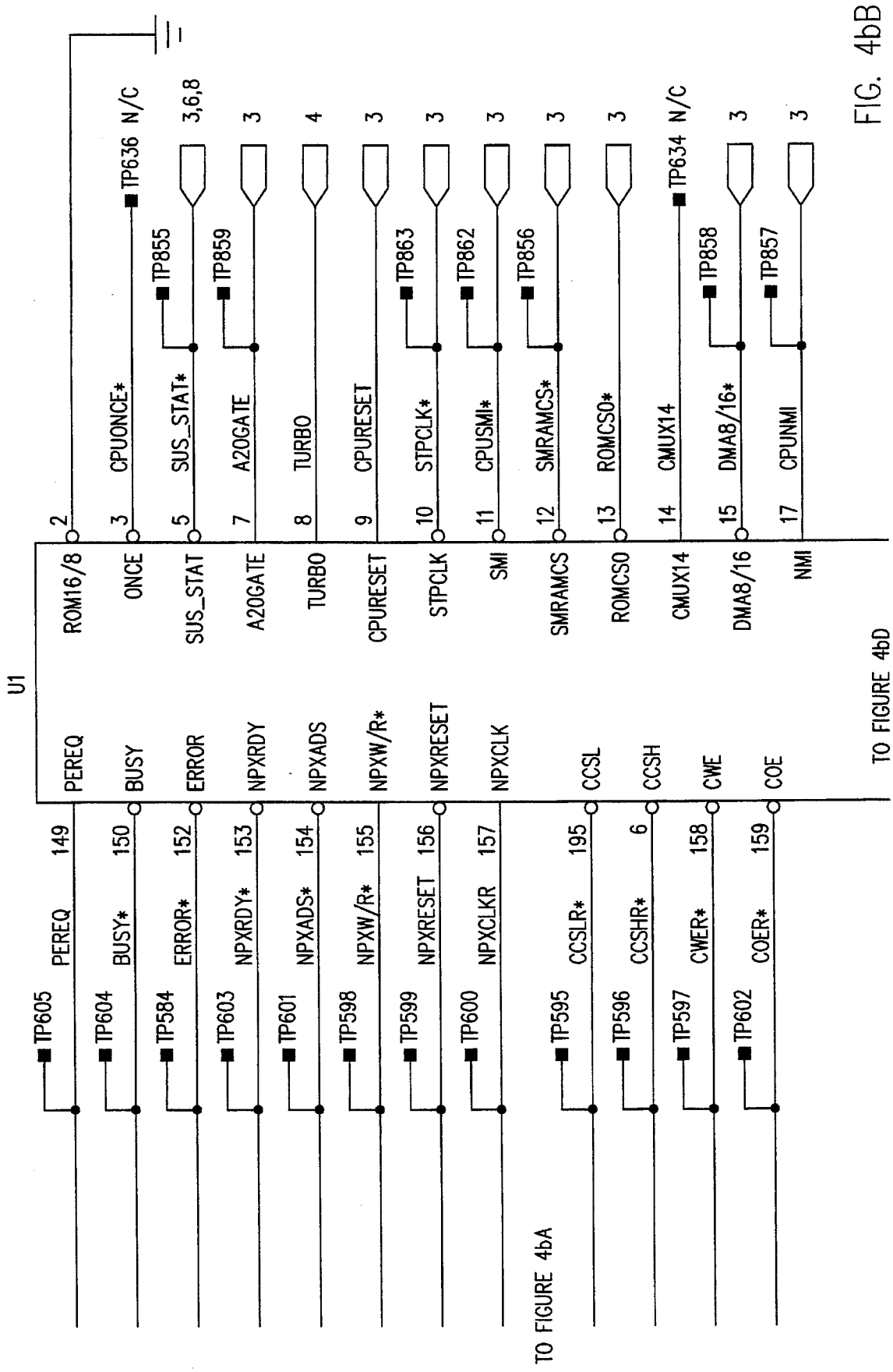

U1
TP605 PEREQ 149 PEREQ
TP604 BUSY* 150 BUSY
TP584 ERROR* 152 ERROR
TP603 NPXRDY* 153 NPXRDY
TP601 NPXADS* 154 NPXADS
TP598 NPXW/R* 155 NPXW/R*
TP599 NPXRESET 156 NPXRESET
TP600 NPXCLKR 157 NPXCLK
TP595 CCSLR* 195 CCSL
TP596 CCSHR* 6 CCSH
TP597 CWER* 158 CWE
TP602 COER* 159 COE
TO FIGURE 4bA
2 ROM16/8
3 CPUONCE* ONCE
TP636 N/C
5 SUS_STAT* SUS_STAT
TP855 3,6,8
7 A20GATE A20GATE
TP859 3
8 TURBO TURBO
4
9 CPURESET CPURESET
3
10 STPCLK* STPCLK
TP863 3
11 CPUSMI* SMI
TP862 3
12 SMRAMCS* SMRAMCS
TP856 3
13 ROMCS0* ROMCS0
3
14 CMUX14 CMUX14
TP634 N/C
15 DMA8/16* DMA8/16
TP858 3
17 CPUNMI NMI
TP857 3
TO FIGURE 4bD

FIG. 4bB

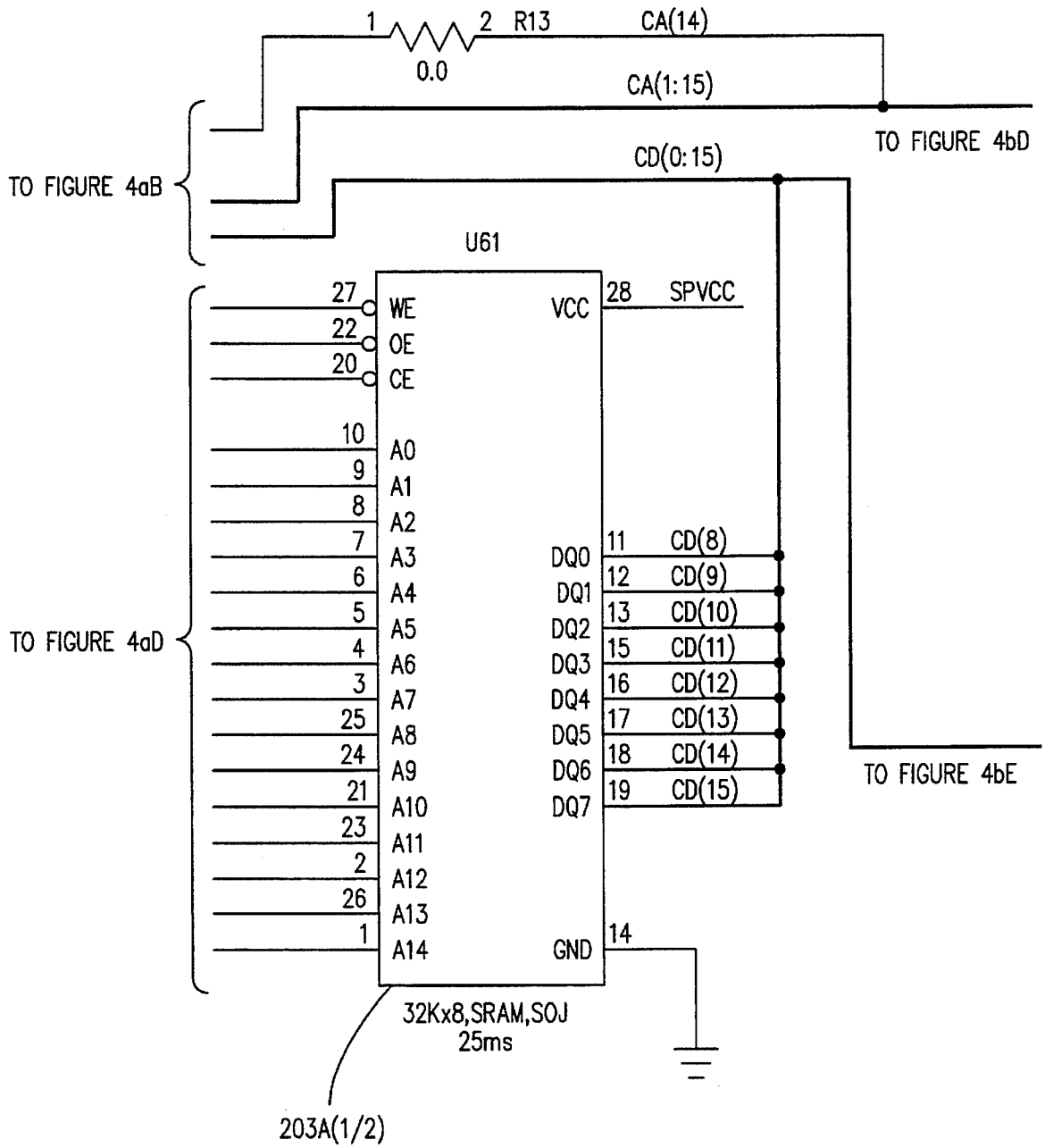

1
2 R13
0.0
CA(14)
CA(1:15)
TO FIGURE 4bD
CD(0:15)
TO FIGURE 4aB
U61
27 WE
22 OE
20 CE
VCC
28 SPVCC
10 A0
9 A1
8 A2
7 A3
6 A4
5 A5
4 A6
3 A7
25 A8
24 A9
21 A10
23 A11
2 A12
26 A13
1 A14
TO FIGURE 4aD
DQ0 11 CD(8)
DQ1 12 CD(9)
DQ2 13 CD(10)
DQ3 15 CD(11)
DQ4 16 CD(12)
DQ5 17 CD(13)
DQ6 18 CD(14)
DQ7 19 CD(15)
TO FIGURE 4bE
GND 14
32Kx8,SRAM,SOJ
25ns
203A(1/2)

FIG. 4bC

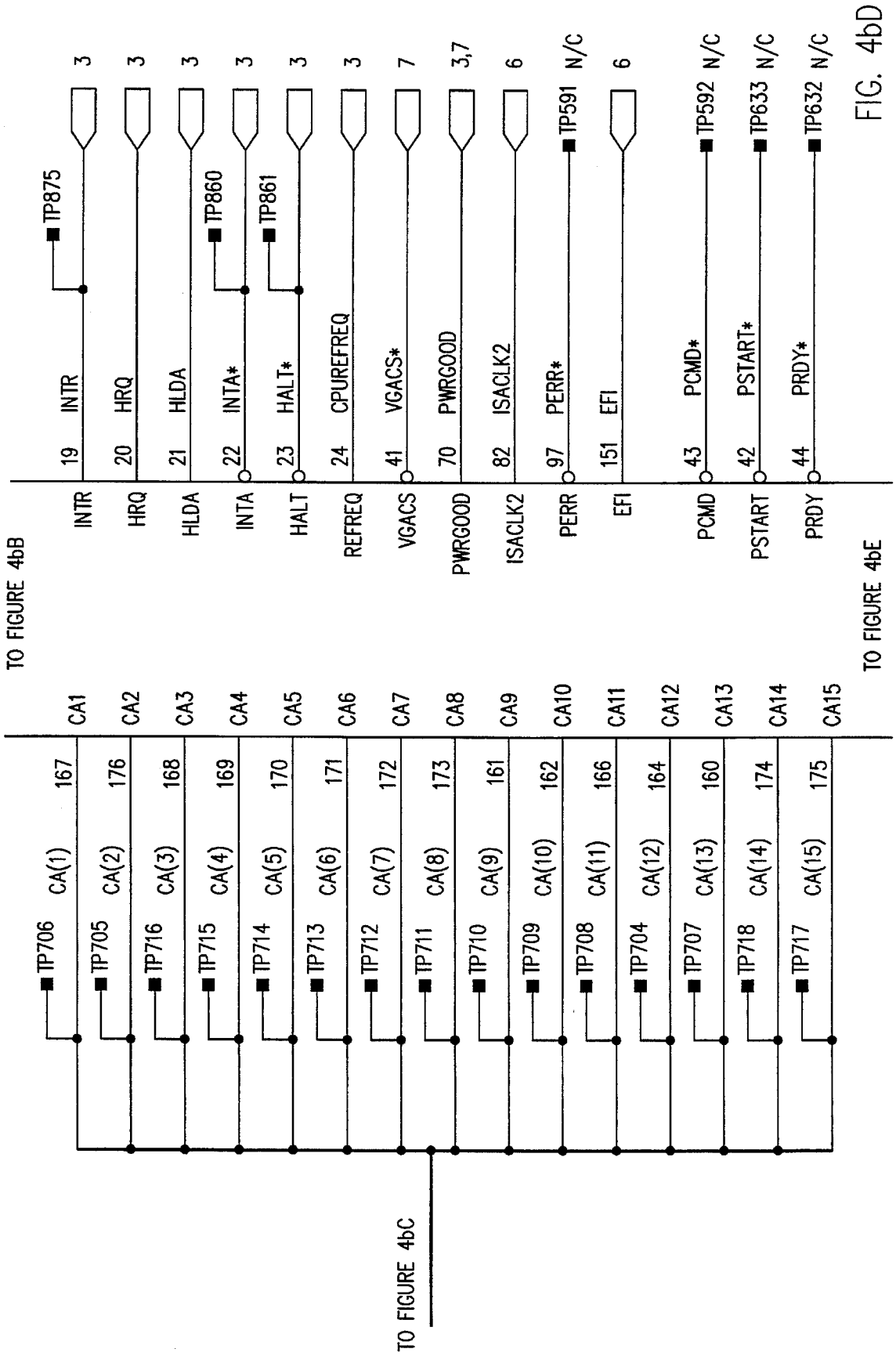

TO FIGURE 4bB
TP875
TP860
TP861
TP591
TP592
TP633
TP632
19 INTR
20 HRQ
21 HLDA
22 INTA*
23 HALT*
24 CPUREFREQ
41 VGACS*
70 PWRGOOD
82 ISACLK2
97 PERR*
151 EFI
43 PCMD*
42 PSTART*
44 PRDY*
3
3
3
3
3
3
7
3,7
6
N/C
6
N/C
N/C
N/C
INTR
HRQ
HLDA
INTA
HALT
REFREQ
VGACS
PWRGOOD
ISACLK2
PERR
EFI
PCMD
PSTART
PRDY
TO FIGURE 4bE
TP706 CA(1) 167 CA1
TP705 CA(2) 176 CA2
TP716 CA(3) 168 CA3
TP715 CA(4) 169 CA4
TP714 CA(5) 170 CA5
TP713 CA(6) 171 CA6
TP712 CA(7) 172 CA7
TP711 CA(8) 173 CA8
TP710 CA(9) 161 CA9
TP709 CA(10) 162 CA10
TP708 CA(11) 166 CA11
TP704 CA(12) 164 CA12
TP707 CA(13) 160 CA13
TP718 CA(14) 174 CA14
TP717 CA(15) 175 CA15
TO FIGURE 4bC

FIG. 4bD

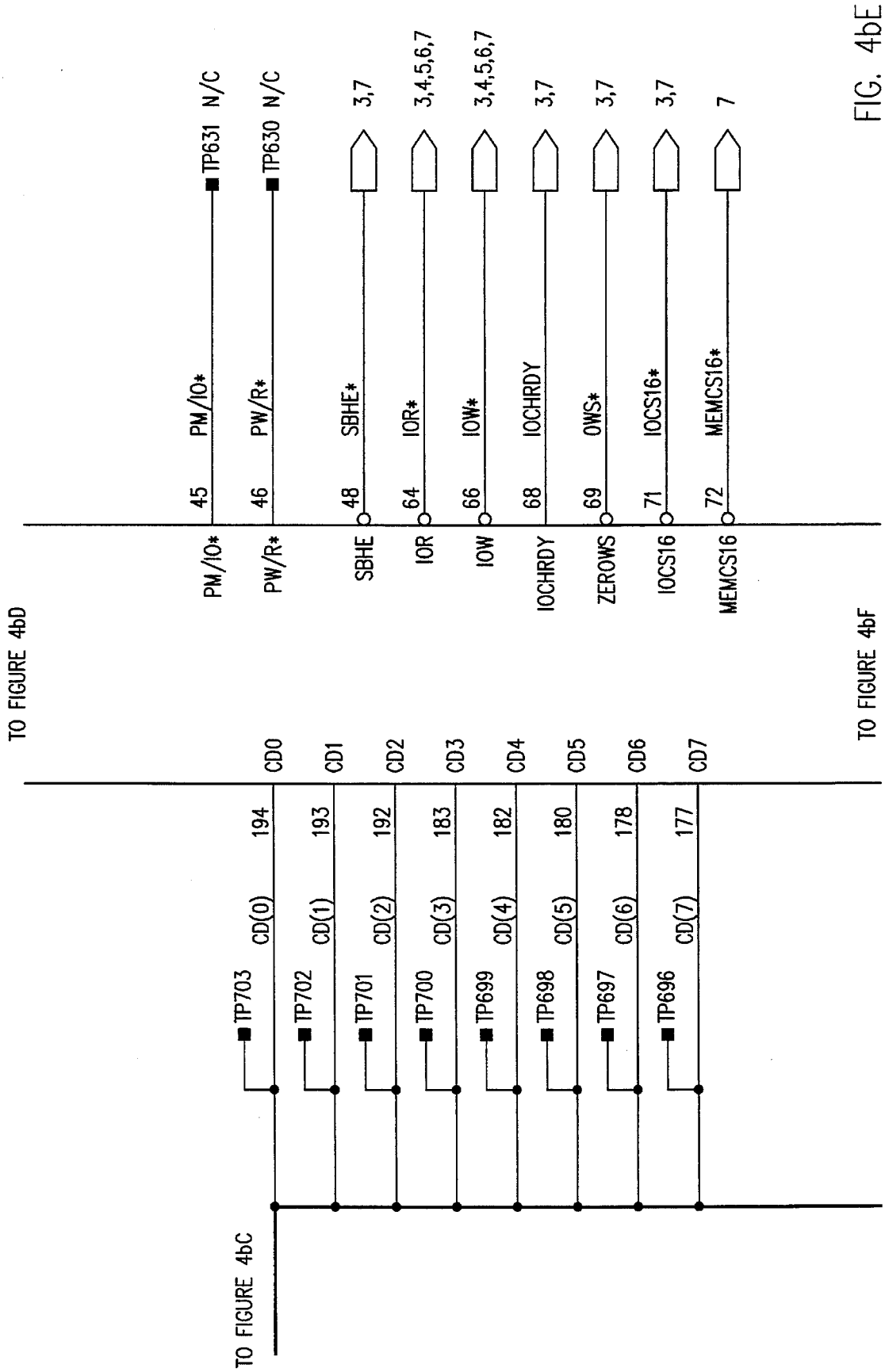

TO FIGURE 4bD
TO FIGURE 4bF
TO FIGURE 4bC
PM/IO*
PW/R*
SBHE
IOR
IOW
IOCHRDY
ZEROWS
IOCS16
MEMCS16
45
46
48
64
66
68
69
71
72
PM/IO*
PW/R*
SBHE*
IOR*
IOW*
IOCHRDY
OWS*
IOCS16*
MEMCS16*
TP631 N/C
TP630 N/C
3,7
3,4,5,6,7
3,4,5,6,7
3,7
3,7
3,7
7
CD0
CD1
CD2
CD3
CD4
CD5
CD6
CD7
194
193
192
183
182
180
178
177
CD(0)
CD(1)
CD(2)
CD(3)
CD(4)
CD(5)
CD(6)
CD(7)
TP703
TP702
TP701
TP700
TP699
TP698
TP697
TP696

FIG. 4bE

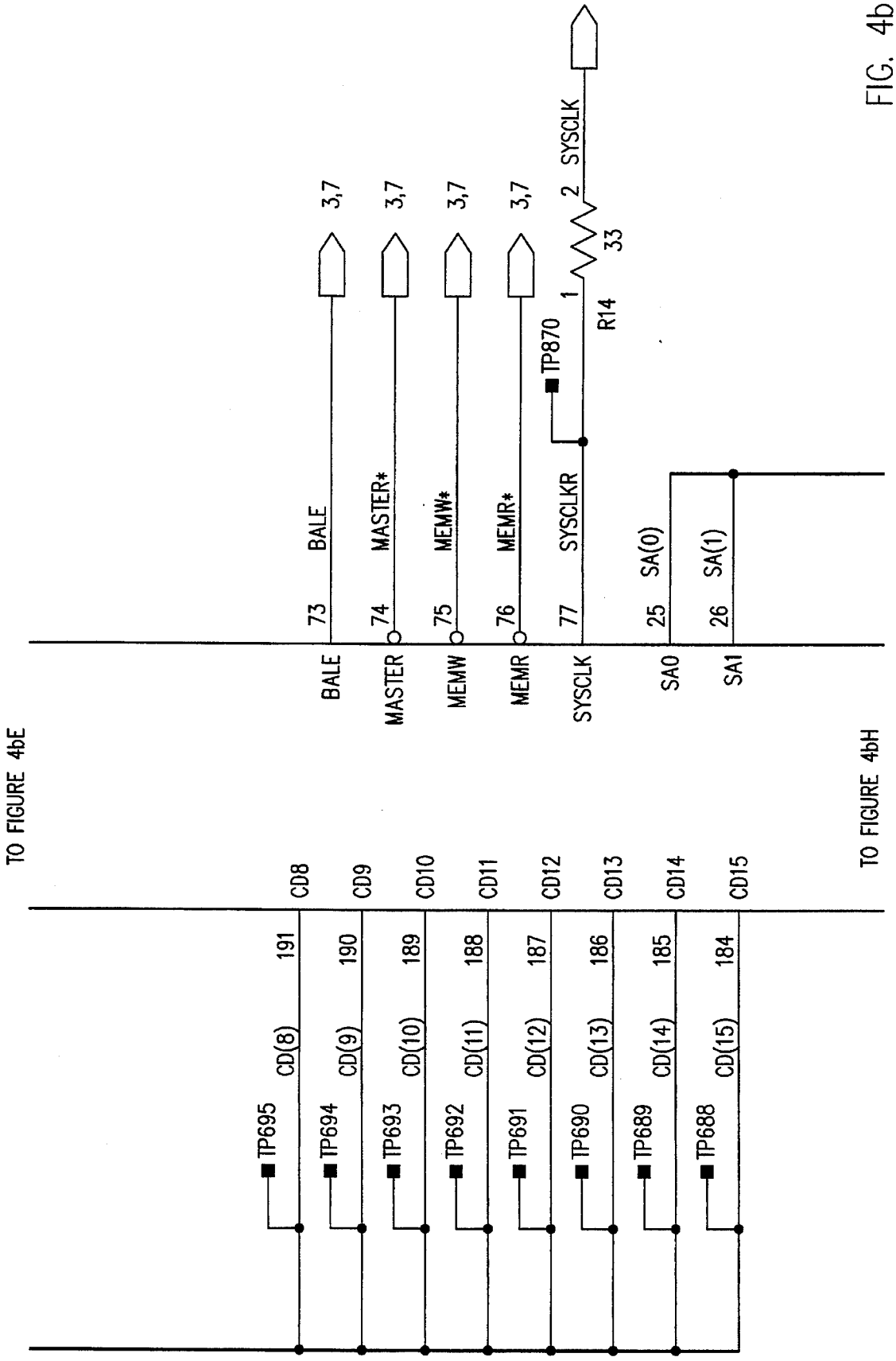

TO FIGURE 4bE
TP695
CD(8)
191
CD8
TP694
CD(9)
190
CD9
TP693
CD(10)
189
CD10
TP692
CD(11)
188
CD11
TP691
CD(12)
187
CD12
TP690
CD(13)
186
CD13
TP689
CD(14)
185
CD14
TP688
CD(15)
184
CD15
BALE
73
BALE
3,7
MASTER
74
MASTER*
3,7
MEMW
75
MEMW*
3,7
MEMR
76
MEMR*
3,7
SYSCLK
77
SYSCLKR
TP870
R14
1
33
2
SYSCLK
SA0
25
SA(0)
SA1
26
SA(1)
TO FIGURE 4bH

FIG. 4bF

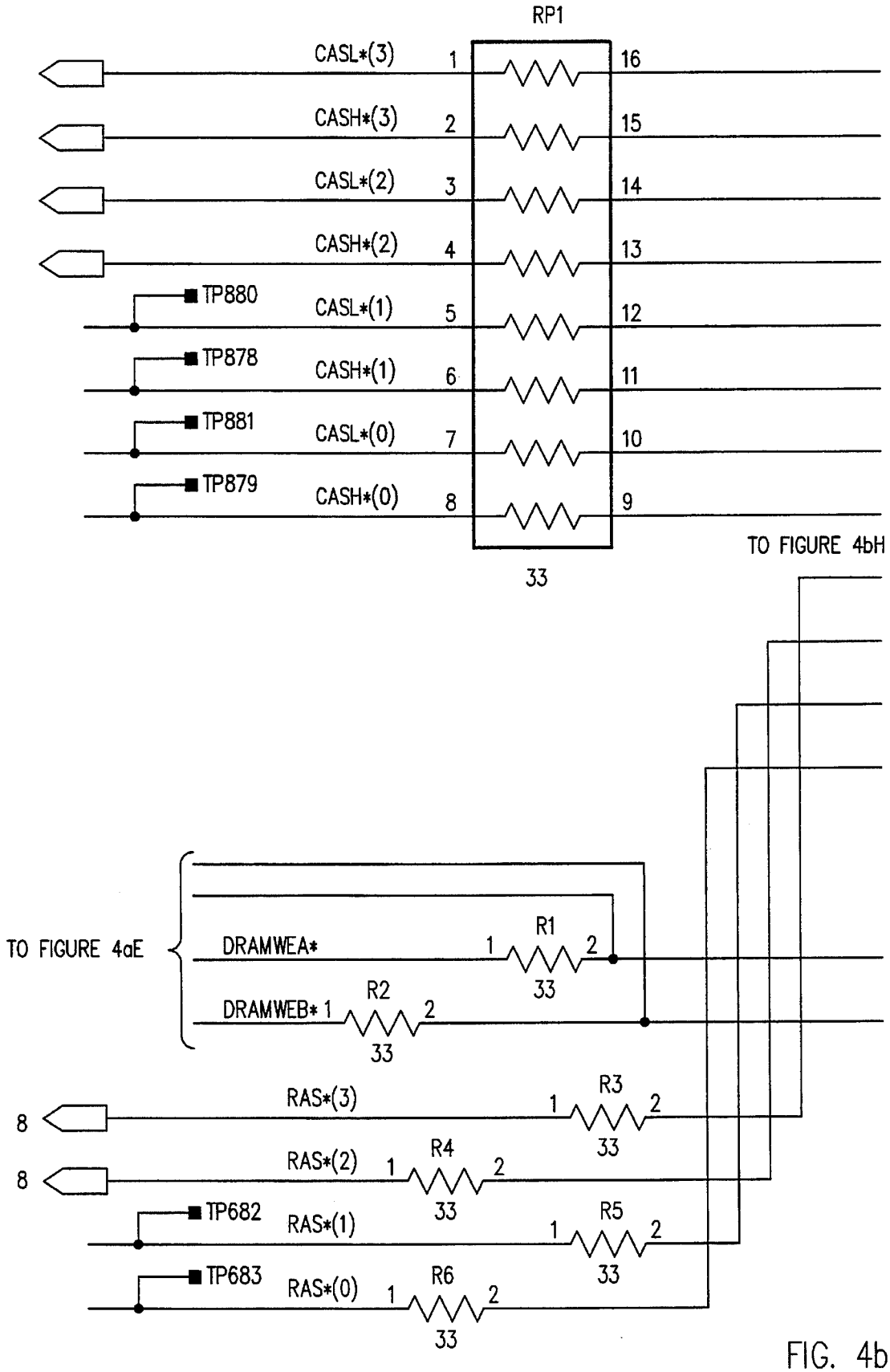

RP1
CASL*(3)
CASH*(3)
CASL*(2)
CASH*(2)
TP880
CASL*(1)
TP878
CASH*(1)
TP881
CASL*(0)
TP879
CASH*(0)
1
2
3
4
5
6
7
8
16
15
14
13
12
11
10
9
33
TO FIGURE 4bH
TO FIGURE 4aE
DRAMWEA*
DRAMWEB*
R1
R2
R3
R4
R5
R6
33
RAS*(3)
RAS*(2)
RAS*(1)
RAS*(0)
8
8
TP682
TP683

FIG. 4bG

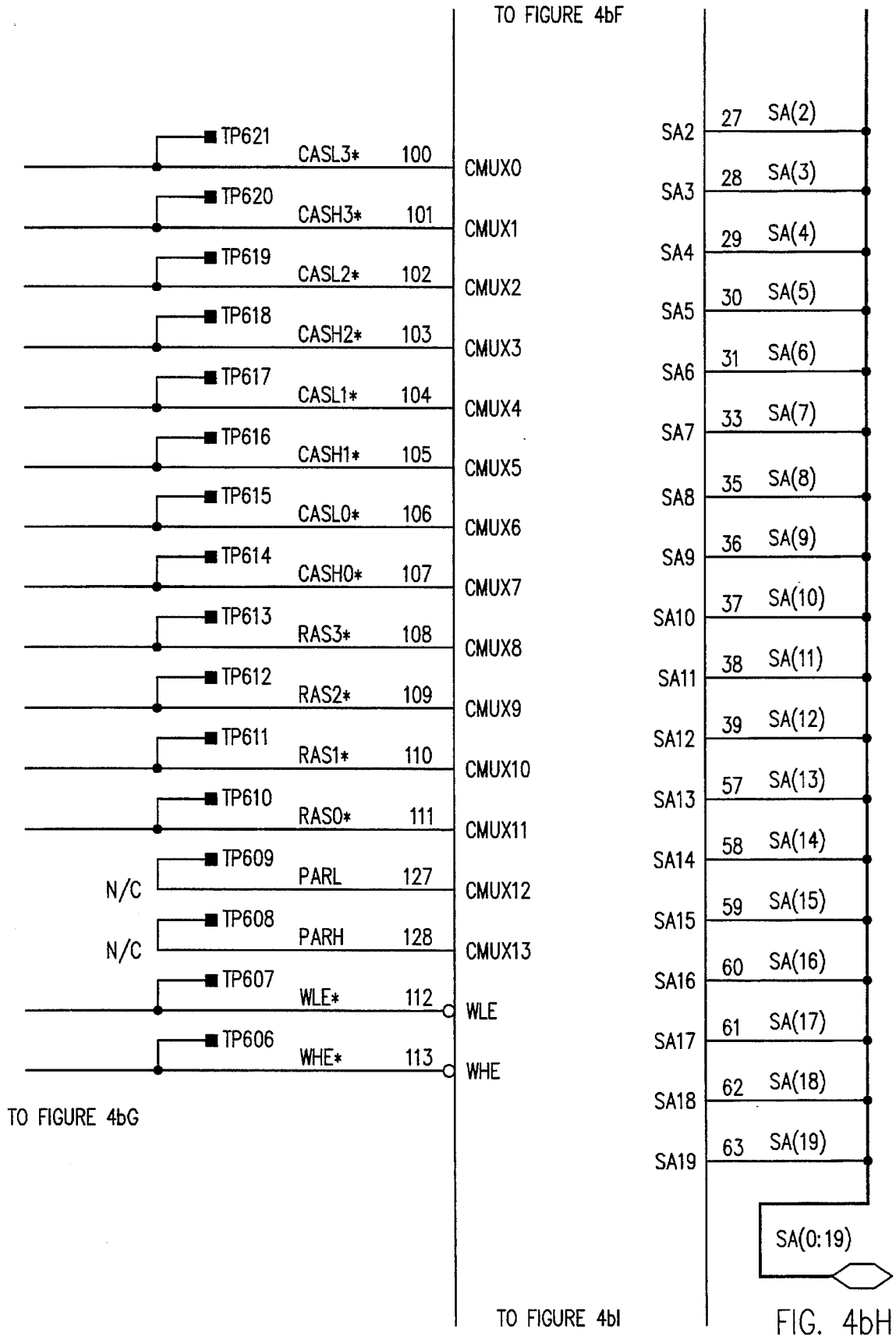

TO FIGURE 4bF
TP621 CASL3* 100 CMUX0
TP620 CASH3* 101 CMUX1
TP619 CASL2* 102 CMUX2
TP618 CASH2* 103 CMUX3
TP617 CASL1* 104 CMUX4
TP616 CASH1* 105 CMUX5
TP615 CASL0* 106 CMUX6
TP614 CASH0* 107 CMUX7
TP613 RAS3* 108 CMUX8
TP612 RAS2* 109 CMUX9
TP611 RAS1* 110 CMUX10
TP610 RAS0* 111 CMUX11
TP609 N/C PARL 127 CMUX12
TP608 N/C PARH 128 CMUX13
TP607 WLE* 112 WLE
TP606 WHE* 113 WHE
TO FIGURE 4bG
SA2 27 SA(2)
SA3 28 SA(3)
SA4 29 SA(4)
SA5 30 SA(5)
SA6 31 SA(6)
SA7 33 SA(7)
SA8 35 SA(8)
SA9 36 SA(9)
SA10 37 SA(10)
SA11 38 SA(11)
SA12 39 SA(12)
SA13 57 SA(13)
SA14 58 SA(14)
SA15 59 SA(15)
SA16 60 SA(16)
SA17 61 SA(17)
SA18 62 SA(18)
SA19 63 SA(19)
SA(0:19)
TO FIGURE 4bI

FIG. 4bH

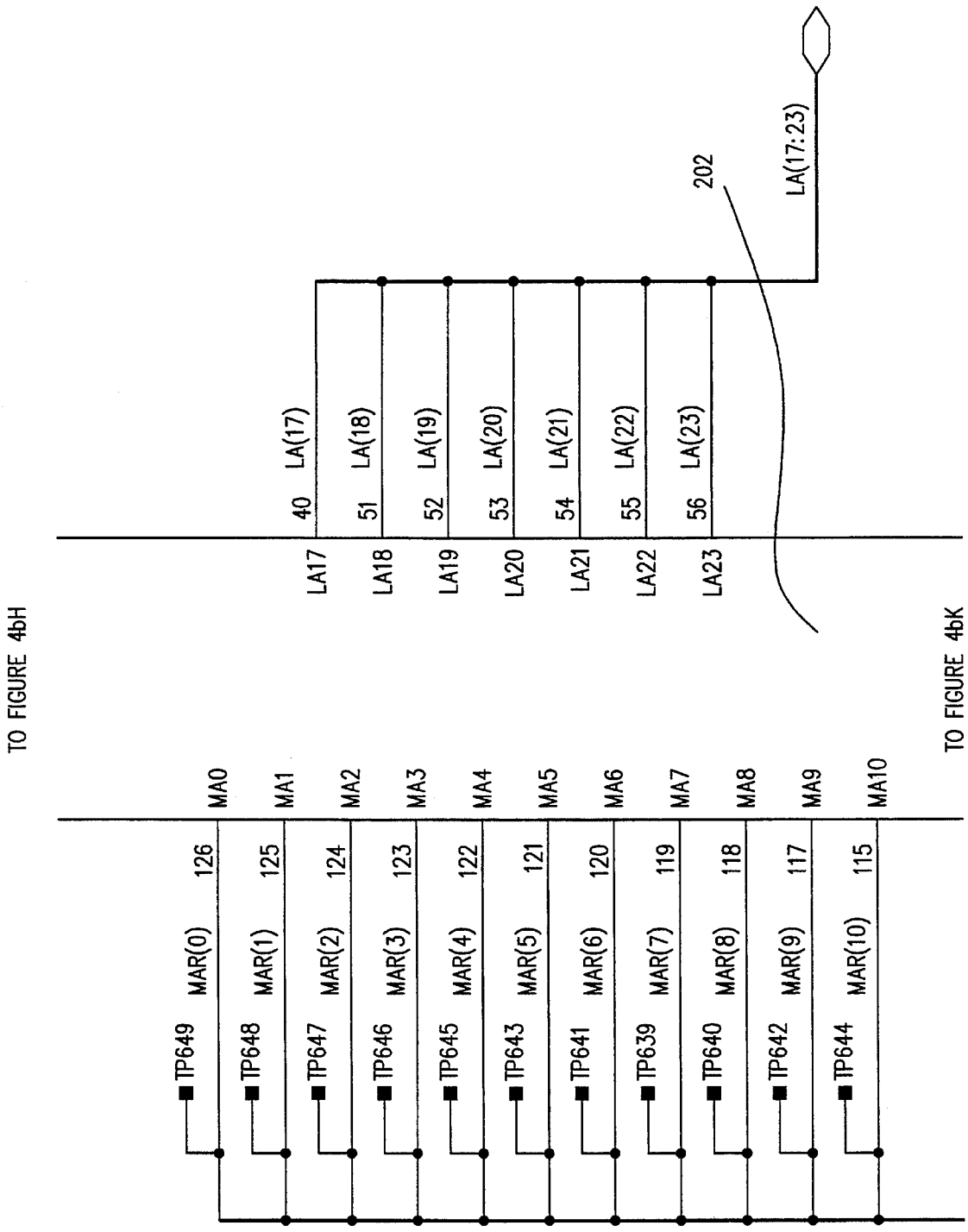

TO FIGURE 4bH
TP649
MAR(0)
126
MA0
TP648
MAR(1)
125
MA1
TP647
MAR(2)
124
MA2
TP646
MAR(3)
123
MA3
TP645
MAR(4)
122
MA4
TP643
MAR(5)
121
MA5
TP641
MAR(6)
120
MA6
TP639
MAR(7)
119
MA7
TP640
MAR(8)
118
MA8
TP642
MAR(9)
117
MA9
TP644
MAR(10)
115
MA10
LA17
LA18
LA19
LA20
LA21
LA22
LA23
40
LA(17)
51
LA(18)
52
LA(19)
53
LA(20)
54
LA(21)
55
LA(22)
56
LA(23)
202
LA(17:23)
TO FIGURE 4bK

FIG. 4bI

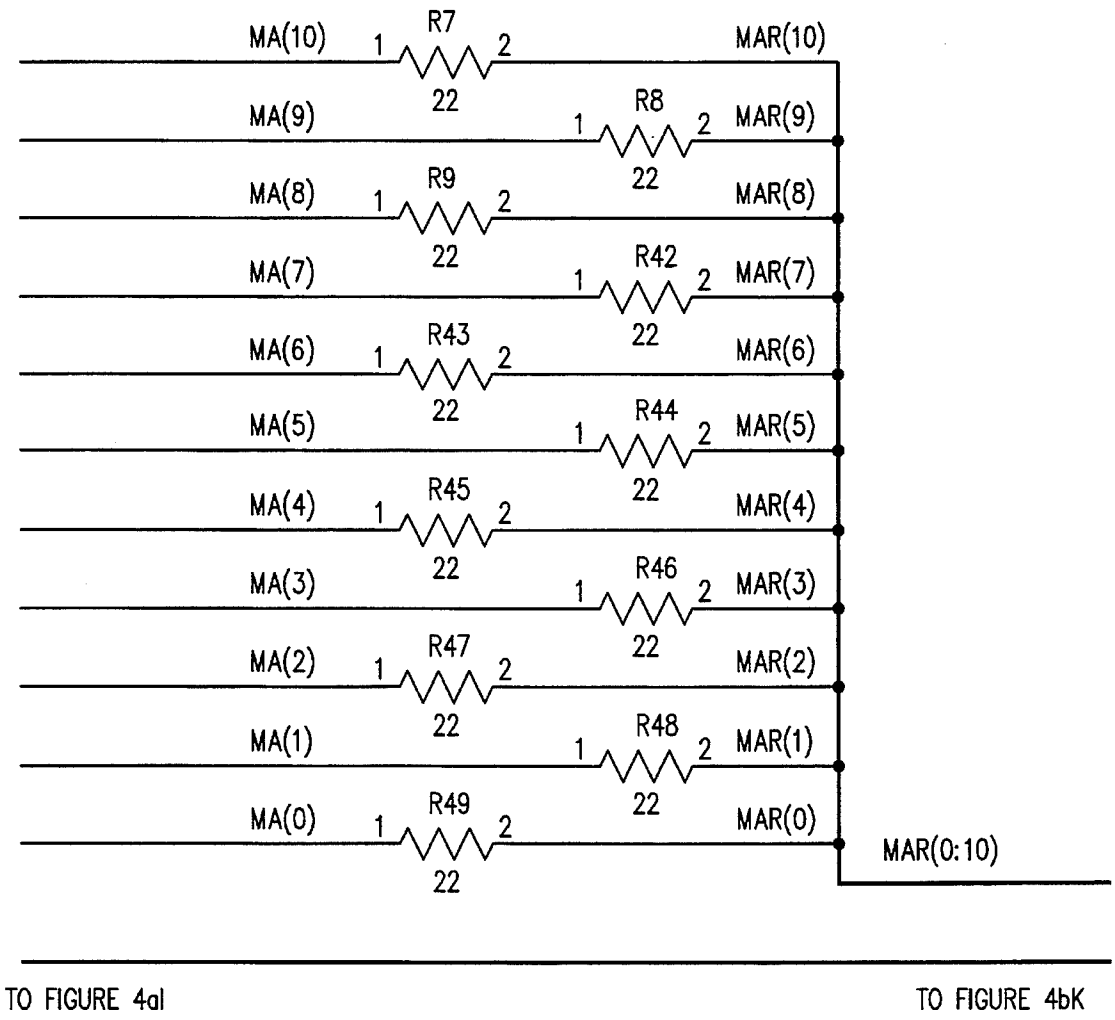

MA(10)
R7
1
2
22
MAR(10)
MA(9)
R8
1
2
22
MAR(9)
MA(8)
R9
1
2
22
MAR(8)
MA(7)
R42
1
2
22
MAR(7)
MA(6)
R43
1
2
22
MAR(6)
MA(5)
R44
1
2
22
MAR(5)
MA(4)
R45
1
2
22
MAR(4)
MA(3)
R46
1
2
22
MAR(3)
MA(2)
R47
1
2
22
MAR(2)
MA(1)
R48
1
2
22
MAR(1)
MA(0)
R49
1
2
22
MAR(0)
MAR(0:10)
TO FIGURE 4al
TO FIGURE 4bK

FIG. 4bJ

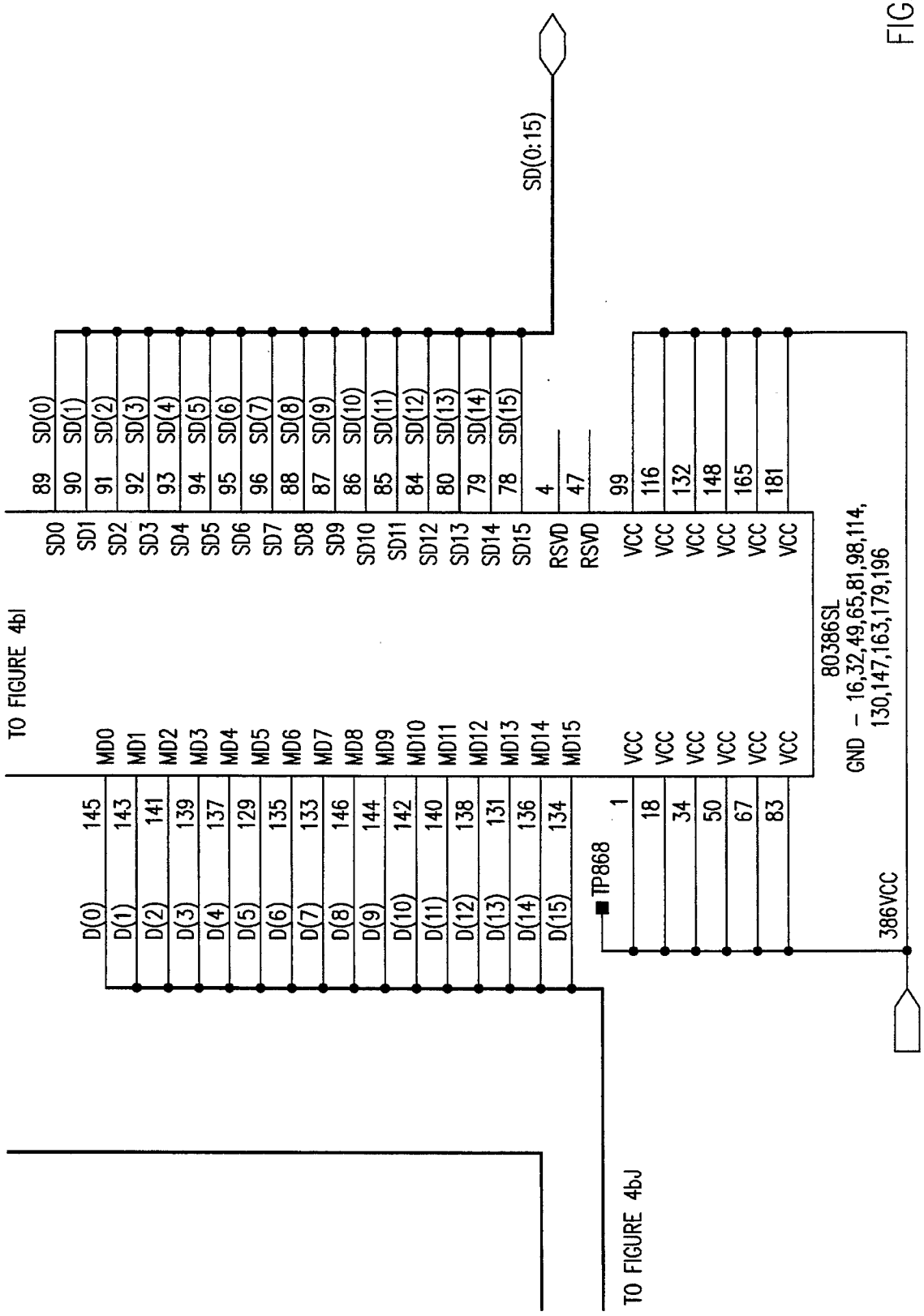

SD(0:15)
TO FIGURE 4bI
TO FIGURE 4bJ
80386SL
GND – 16,32,49,65,81,98,114,
130,147,163,179,196
386VCC
TP868
SD0 SD1 SD2 SD3 SD4 SD5 SD6 SD7 SD8 SD9 SD10 SD11 SD12 SD13 SD14 SD15 RSVD RSVD VCC VCC VCC VCC VCC VCC
89 90 91 92 93 94 95 96 88 87 86 85 84 80 79 78 4 47 99 116 132 148 165 181
MD0 MD1 MD2 MD3 MD4 MD5 MD6 MD7 MD8 MD9 MD10 MD11 MD12 MD13 MD14 MD15 VCC VCC VCC VCC VCC VCC
145 143 141 139 137 129 135 133 146 144 142 140 138 131 136 134 1 18 34 50 67 83
D(0) D(1) D(2) D(3) D(4) D(5) D(6) D(7) D(8) D(9) D(10) D(11) D(12) D(13) D(14) D(15)

FIG. 4bK

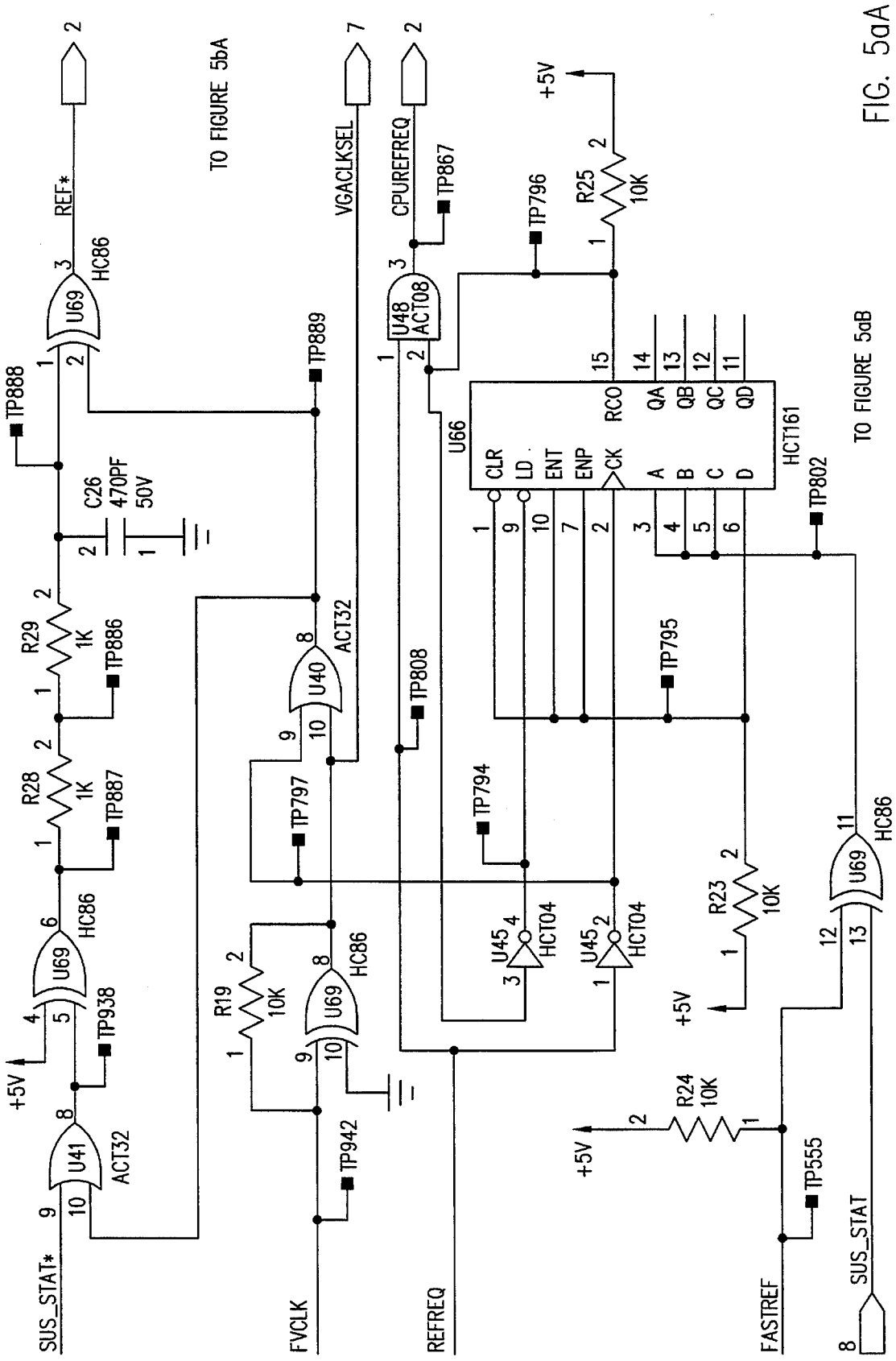

TO FIGURE 5bA
REF*
HC86
U69
TP888
C26
470PF
50V
R29
1K
TP886
R28
1K
TP887
TP938
+5V
U41
ACT32
SUS_STAT*
VGACLKSEL
TP889
U40
ACT32
TP797
R19
10K
TP942
FVCLK
CPUREFREQ
TP867
U48
ACT08
TP808
REFREQ
TP796
R25
10K
U66
CLR
LD
ENT
ENP
CK
RCO
QA
QB
QC
QD
A
B
C
D
HCT161
TP795
TP794
U45
HCT04
R23
10K
R24
10K
TP555
FASTREF
TP802
TO FIGURE 5aB
SUS_STAT

FIG. 5aA

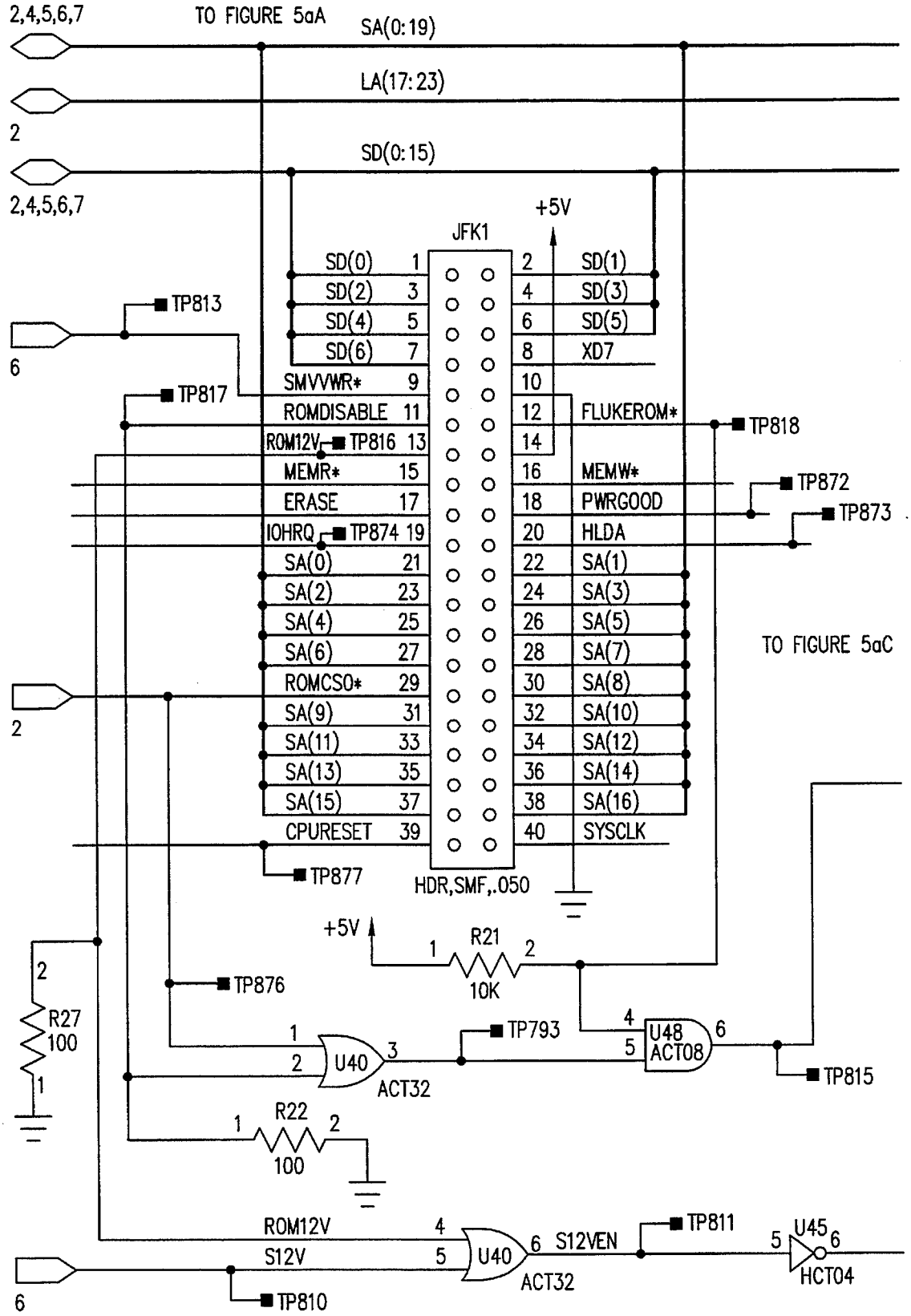

2,4,5,6,7
TO FIGURE 5aA
SA(0:19)
LA(17:23)
2
SD(0:15)
2,4,5,6,7
+5V
JFK1
SD(0) 1
SD(2) 3
SD(4) 5
SD(6) 7
2 SD(1)
4 SD(3)
6 SD(5)
8 XD7
TP813
6
TP817
SMVVWR* 9
ROMDISABLE 11
ROM12V TP816 13
MEMR* 15
ERASE 17
IOHRQ TP874 19
10
12 FLUKEROM*
14
16 MEMW*
18 PWRGOOD
20 HLDA
TP818
TP872
TP873
SA(0) 21
SA(2) 23
SA(4) 25
SA(6) 27
ROMCS0* 29
SA(9) 31
SA(11) 33
SA(13) 35
SA(15) 37
CPURESET 39
22 SA(1)
24 SA(3)
26 SA(5)
28 SA(7)
30 SA(8)
32 SA(10)
34 SA(12)
36 SA(14)
38 SA(16)
40 SYSCLK
TO FIGURE 5aC
2
TP877
HDR,SMF,.050
+5V
R21
10K
TP876
R27
100
U40
ACT32
TP793
U48
ACT08
TP815
R22
100
ROM12V
S12V
U40
ACT32
S12VEN
TP811
U45
HCT04
6
TP810

FIG. 5aB

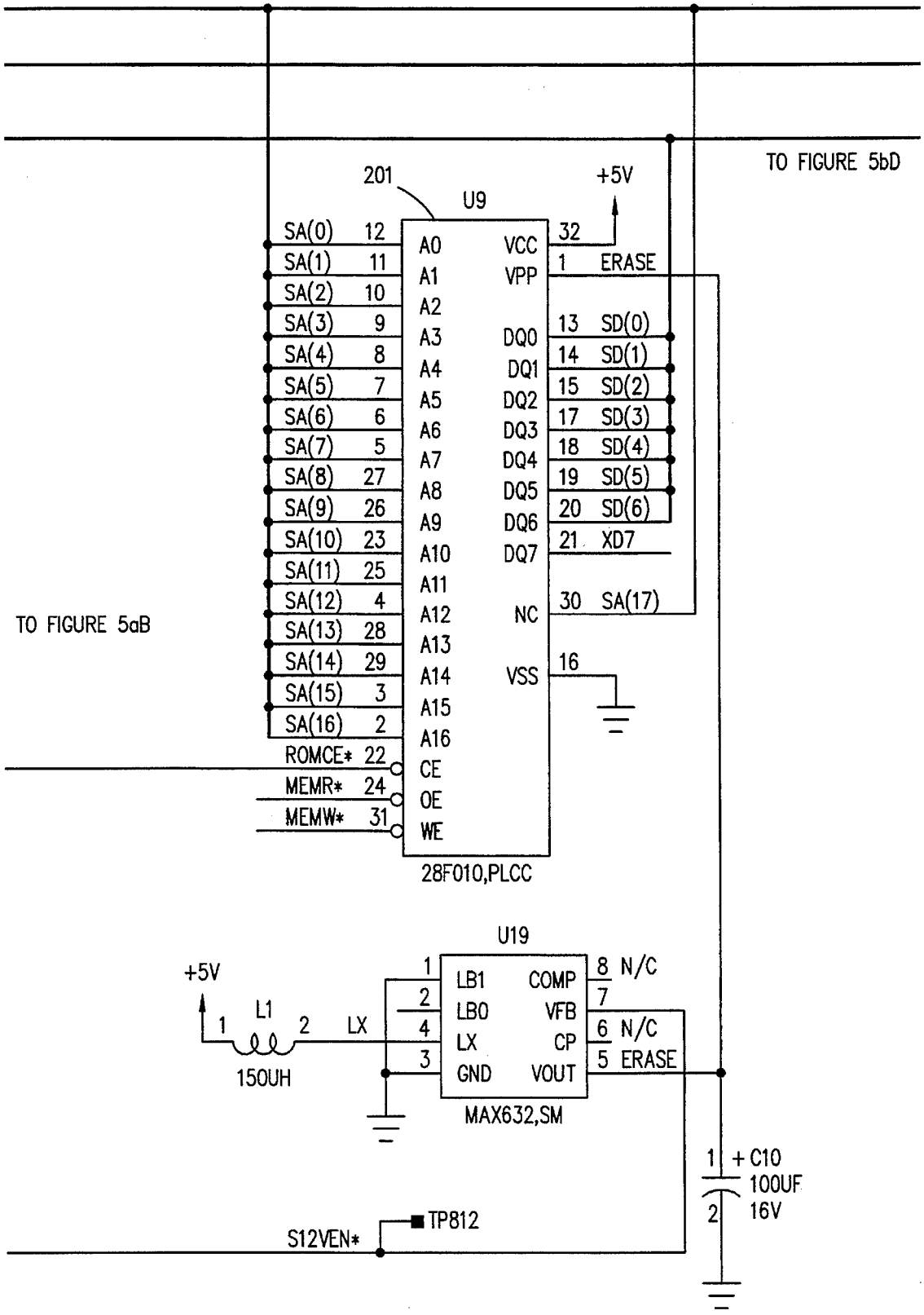

TO FIGURE 5bD
201
U9
+5V
SA(0) 12
SA(1) 11
SA(2) 10
SA(3) 9
SA(4) 8
SA(5) 7
SA(6) 6
SA(7) 5
SA(8) 27
SA(9) 26
SA(10) 23
SA(11) 25
SA(12) 4
SA(13) 28
SA(14) 29
SA(15) 3
SA(16) 2
ROMCE* 22
MEMR* 24
MEMW* 31
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
CE
OE
WE
VCC 32
VPP 1 ERASE
DQ0 13 SD(0)
DQ1 14 SD(1)
DQ2 15 SD(2)
DQ3 17 SD(3)
DQ4 18 SD(4)
DQ5 19 SD(5)
DQ6 20 SD(6)
DQ7 21 XD7
NC 30 SA(17)
VSS 16
28F010,PLCC
TO FIGURE 5aB
U19
+5V
L1
150UH
LX
LB1 1
LB0 2
LX 4
GND 3
COMP 8 N/C
VFB 7
CP 6 N/C
VOUT 5 ERASE
MAX632,SM
+C10
100UF
16V
TP812
S12VEN*

FIG. 5aC

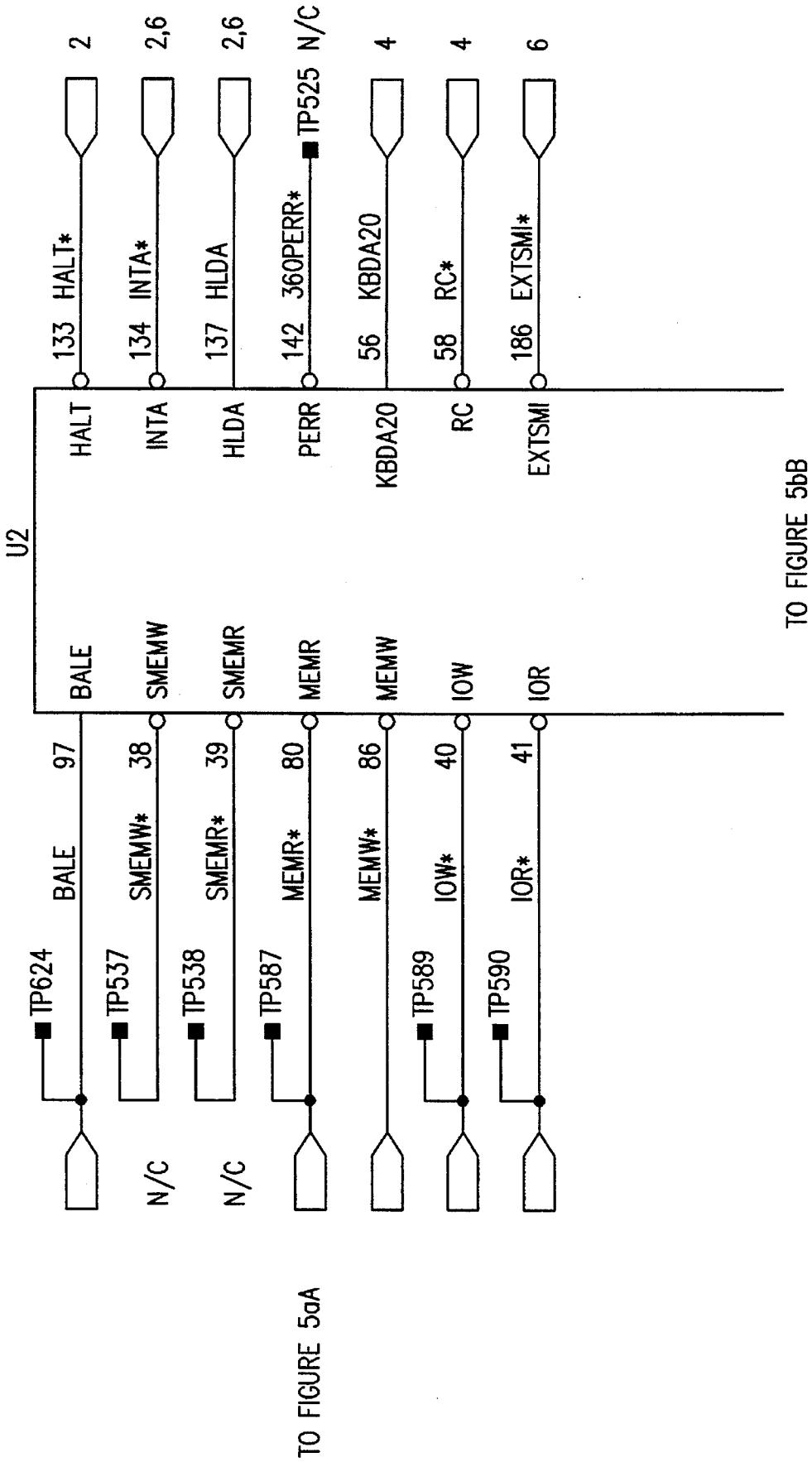

U2
133 HALT*
134 INTA*
137 HLDA
142 360PERR*
56 KBDA20
58 RC*
186 EXTSMI*
HALT
INTA
HLDA
PERR
KBDA20
RC
EXTSMI
2
2,6
2,6
N/C
TP525
4
4
6
TO FIGURE 5bB
BALE
SMEMW
SMEMR
MEMR
MEMW
IOW
IOR
97
38
39
80
86
40
41
BALE
SMEMW*
SMEMR*
MEMR*
MEMW*
IOW*
IOR*
TP624
TP537
TP538
TP587
TP589
TP590
N/C
N/C
TO FIGURE 5aA

FIG. 5bA

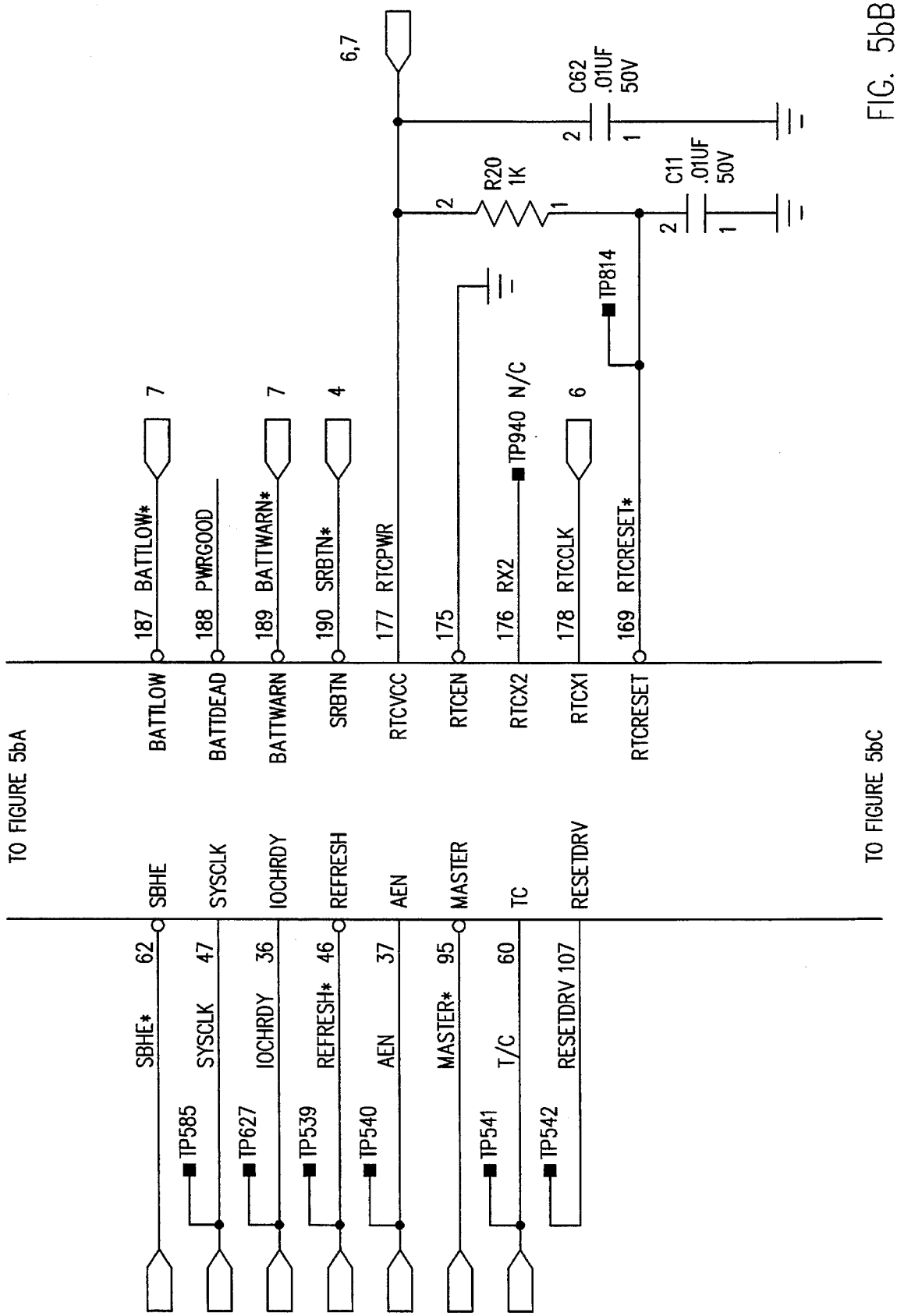

TO FIGURE 5bA
SBHE
SYSCLK
IOCHRDY
REFRESH
AEN
MASTER
TC
RESETDRV
BATTLOW
BATTDEAD
BATTWARN
SRBTN
RTCVCC
RTCEN
RTCX2
RTCX1
RTCRESET
TO FIGURE 5bC
62
47
36
46
37
95
60
107
SBHE*
SYSCLK
IOCHRDY
REFRESH*
AEN
MASTER*
T/C
RESETDRV
TP585
TP627
TP539
TP540
TP541
TP542
187 BATTLOW*
188 PWRGOOD
189 BATTWARN*
190 SRBTN*
177 RTCPWR
175
176 RX2
178 RTCCLK
169 RTCRESET*
7
7
4
6
TP940 N/C
TP814
6,7
R20
1K
C62
.01UF
50V
C11
.01UF
50V

FIG. 5bB

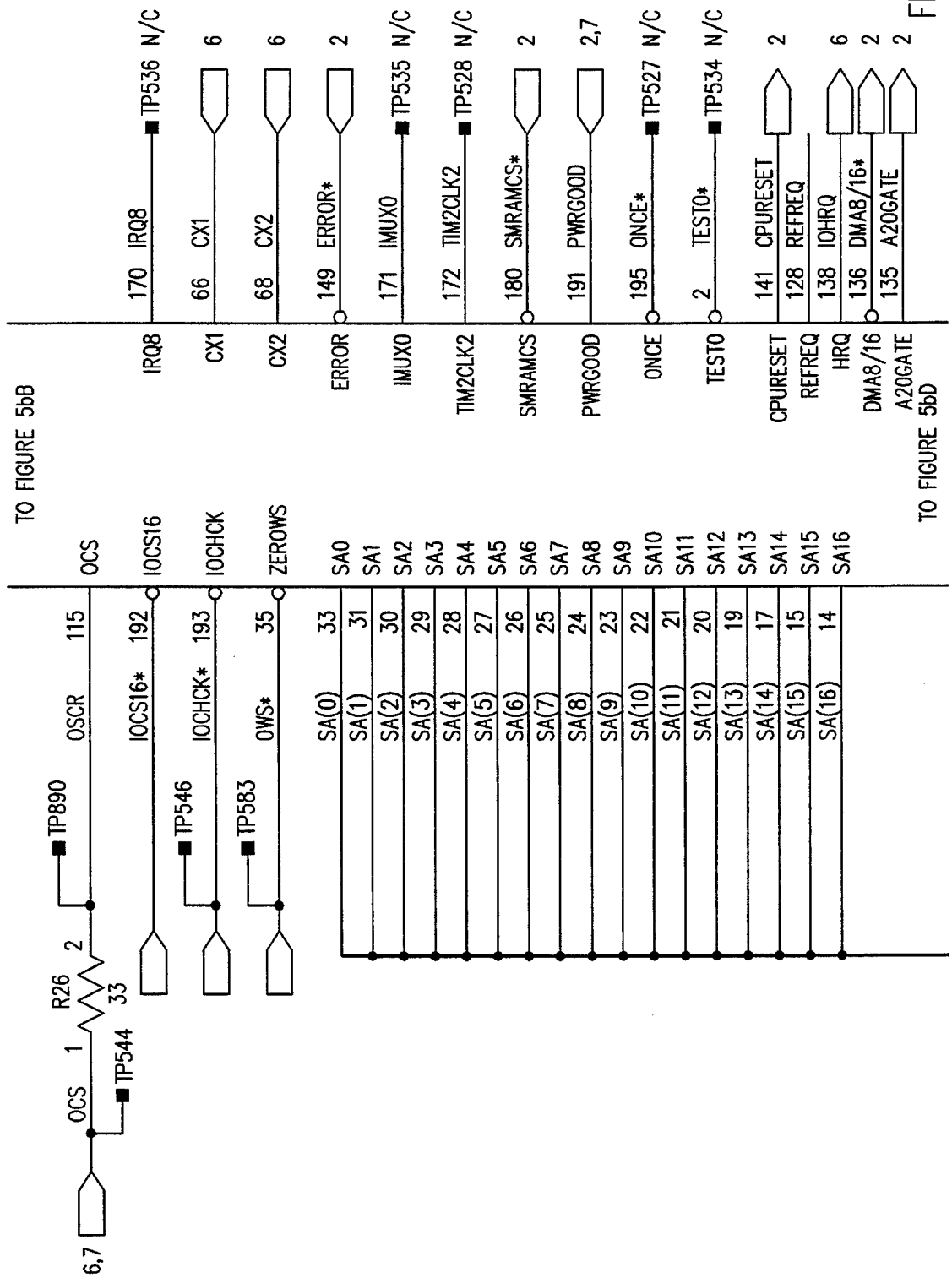

TO FIGURE 5bB
170 IRQ8
TP536 N/C
66 CX1
6
68 CX2
6
149 ERROR*
2
171 IMUX0
TP535 N/C
172 TIM2CLK2
TP528 N/C
180 SMRAMCS*
2
191 PWRGOOD
2,7
195 ONCE*
TP527 N/C
2 TEST0*
TP534 N/C
141 CPURESET
2
128 REFREQ
138 IOHRQ
6
136 DMA8/16*
2
135 A20GATE
2
IRQ8
CX1
CX2
ERROR
IMUX0
TIM2CLK2
SMRAMCS
PWRGOOD
ONCE
TEST0
CPURESET
REFREQ
HRQ
DMA8/16
A20GATE
TO FIGURE 5bD
OCS
IOCS16
IOCHCK
ZEROWS
SA0
SA1
SA2
SA3
SA4
SA5
SA6
SA7
SA8
SA9
SA10
SA11
SA12
SA13
SA14
SA15
SA16
115 OSCR
192 IOCS16*
193 IOCHCK*
35 OWS*
33 SA(0)
31 SA(1)
30 SA(2)
29 SA(3)
28 SA(4)
27 SA(5)
26 SA(6)
25 SA(7)
24 SA(8)
23 SA(9)
22 SA(10)
21 SA(11)
20 SA(12)
19 SA(13)
17 SA(14)
15 SA(15)
14 SA(16)
TP890
TP546
TP583
R26
2
33
1
TP544
OCS
6,7

FIG. 5bC

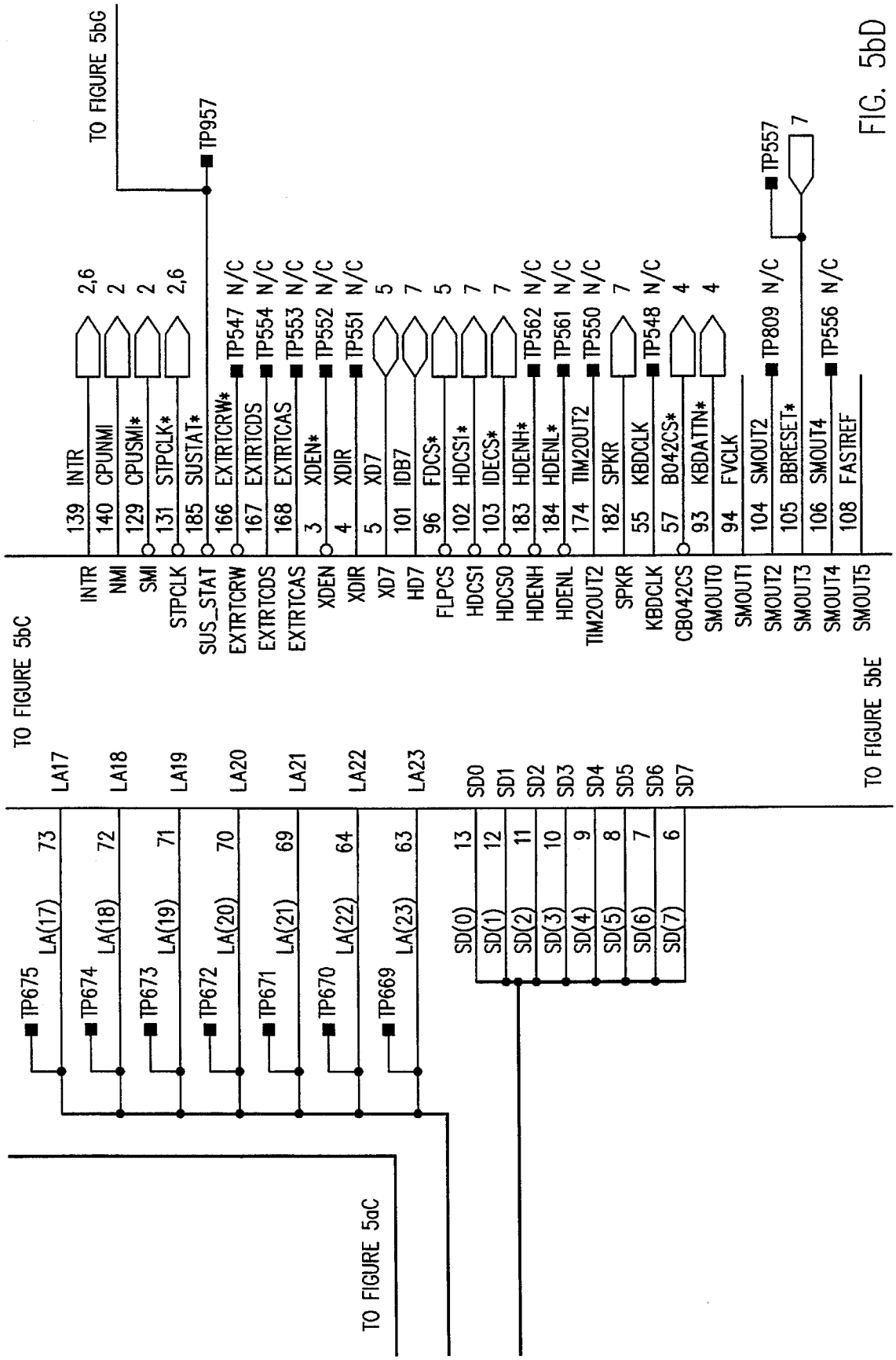

TO FIGURE 5bG
TP957
TP557
7
2,6
2
2
2,6
139 INTR
140 CPUNMI
129 CPUSMI*
131 STPCLK*
185 SUSTAT*
166 EXTRTCRW*
167 EXTRTCDS
168 EXTRTCAS
3 XDEN*
4 XDIR
5 XD7
101 IDB7
96 FDCS*
102 HDCS1*
103 IDECS*
183 HDENH*
184 HDENL*
174 TIM2OUT2
182 SPKR
55 KBDCLK
57 B042CS*
93 KBDATTN*
94 FVCLK
104 SMOUT2
105 BBRESET*
106 SMOUT4
108 FASTREF
TP547 N/C
TP554 N/C
TP553 N/C
TP552 N/C
TP551 N/C
5
7
5
7
7
TP562 N/C
TP561 N/C
TP550 N/C
7
TP548 N/C
4
4
TP809 N/C
TP556 N/C
INTR
NMI
SMI
STPCLK
SUS_STAT
EXTRTCRW
EXTRTCDS
EXTRTCAS
XDEN
XDIR
XD7
HD7
FLPCS
HDCS1
HDCS0
HDENH
HDENL
TIM2OUT2
SPKR
KBDCLK
CB042CS
SMOUT0
SMOUT1
SMOUT2
SMOUT3
SMOUT4
SMOUT5
TO FIGURE 5bC
TO FIGURE 5bE
LA17
LA18
LA19
LA20
LA21
LA22
LA23
SD0
SD1
SD2
SD3
SD4
SD5
SD6
SD7
TP675 LA(17) 73
TP674 LA(18) 72
TP673 LA(19) 71
TP672 LA(20) 70
TP671 LA(21) 69
TP670 LA(22) 64
TP669 LA(23) 63
SD(0) 13
SD(1) 12
SD(2) 11
SD(3) 10
SD(4) 9
SD(5) 8
SD(6) 7
SD(7) 6
TO FIGURE 5aC

FIG. 5bD

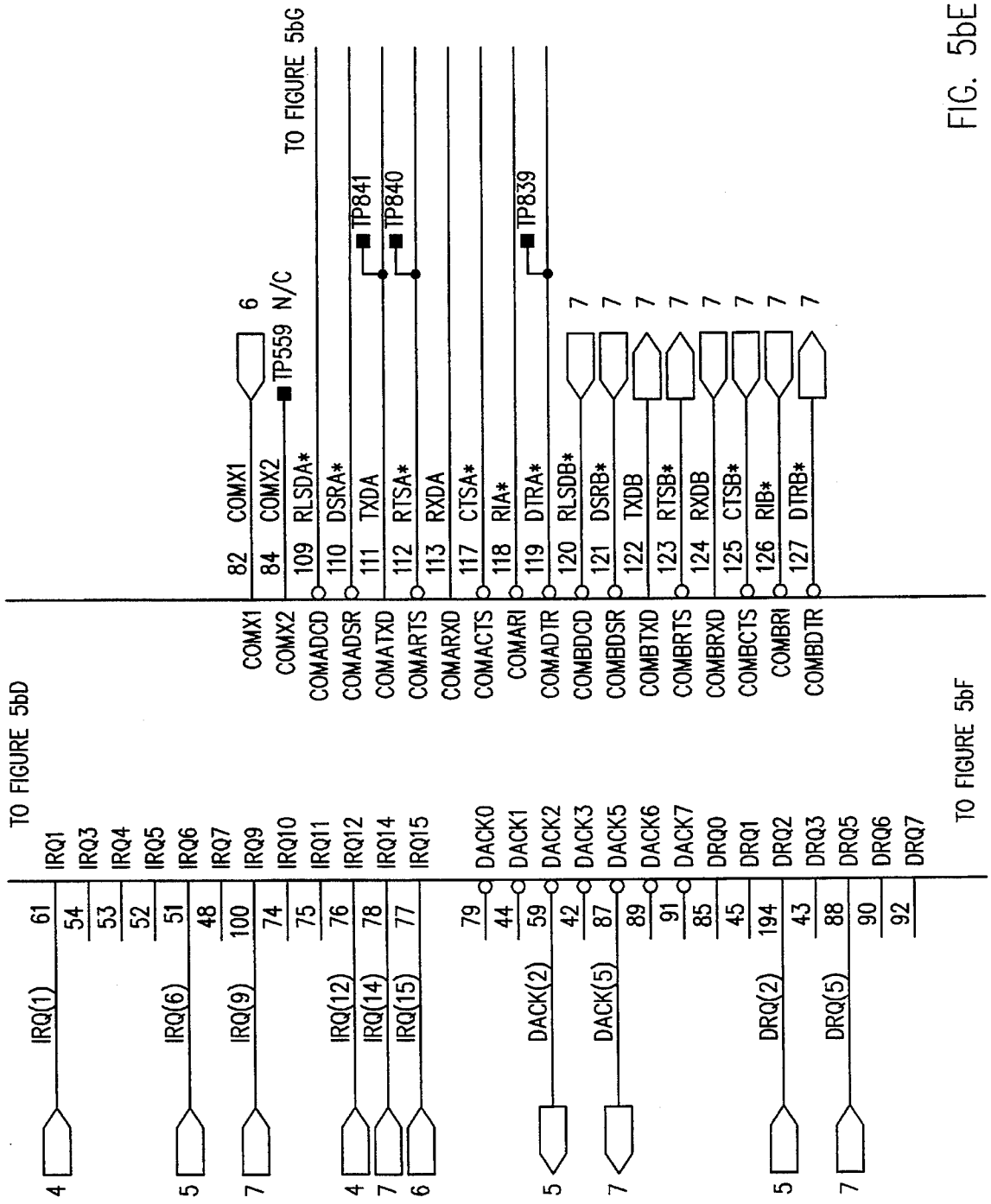

TO FIGURE 5bD
TO FIGURE 5bG
TO FIGURE 5bF
IRQ(1)
IRQ(6)
IRQ(9)
IRQ(12)
IRQ(14)
IRQ(15)
DACK(2)
DACK(5)
DRQ(2)
DRQ(5)
IRQ1
IRQ3
IRQ4
IRQ5
IRQ6
IRQ7
IRQ9
IRQ10
IRQ11
IRQ12
IRQ14
IRQ15
DACK0
DACK1
DACK2
DACK3
DACK5
DACK6
DACK7
DRQ0
DRQ1
DRQ2
DRQ3
DRQ5
DRQ6
DRQ7
COMX1
COMX2
COMADCD
COMADSR
COMATXD
COMARTS
COMARXD
COMACTS
COMARI
COMADTR
COMBDCD
COMBDSR
COMBTXD
COMBRTS
COMBRXD
COMBCTS
COMBRI
COMBDTR
TP559
N/C
TP841
TP840
TP839
RLSDA*
DSRA*
TXDA
RTSA*
RXDA
CTSA*
RIA*
DTRA*
RLSDB*
DSRB*
TXDB
RTSB*
RXDB
CTSB*
RIB*
DTRB*

FIG. 5bE

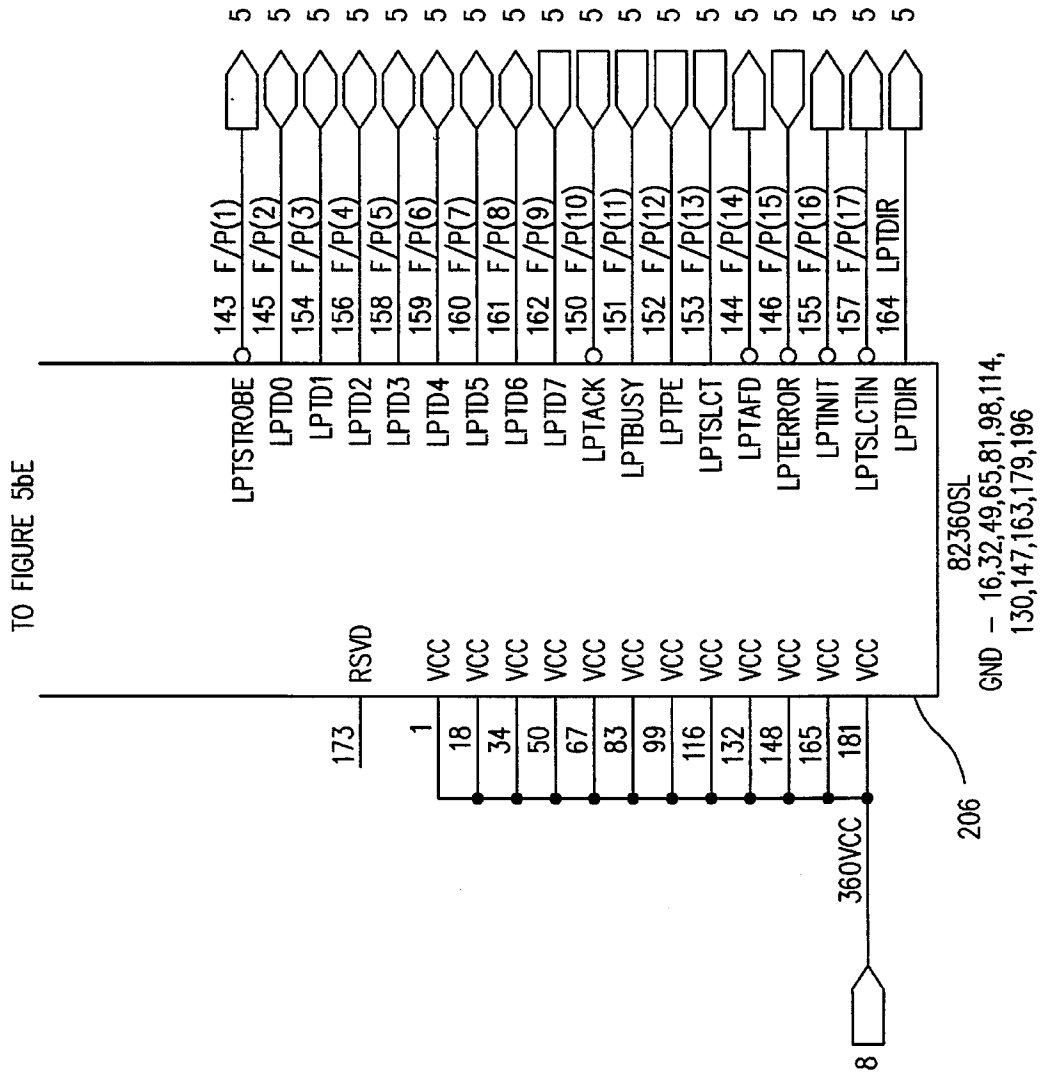

TO FIGURE 5bE
173
RSVD
1
VCC
18
34
50
67
83
99
116
132
148
165
181
360VCC
8
206
82360SL
GND – 16,32,49,65,81,98,114,
130,147,163,179,196
LPTSTROBE
LPTD0
LPTD1
LPTD2
LPTD3
LPTD4
LPTD5
LPTD6
LPTD7
LPTACK
LPTBUSY
LPTPE
LPTSLCT
LPTAFD
LPTERROR
LPTINIT
LPTSLCTIN
LPTDIR
143 F/P(1)
145 F/P(2)
154 F/P(3)
156 F/P(4)
158 F/P(5)
159 F/P(6)
160 F/P(7)
161 F/P(8)
162 F/P(9)
150 F/P(10)
151 F/P(11)
152 F/P(12)
153 F/P(13)
144 F/P(14)
146 F/P(15)
155 F/P(16)
157 F/P(17)
164 LPTDIR
5

FIG. 5bF

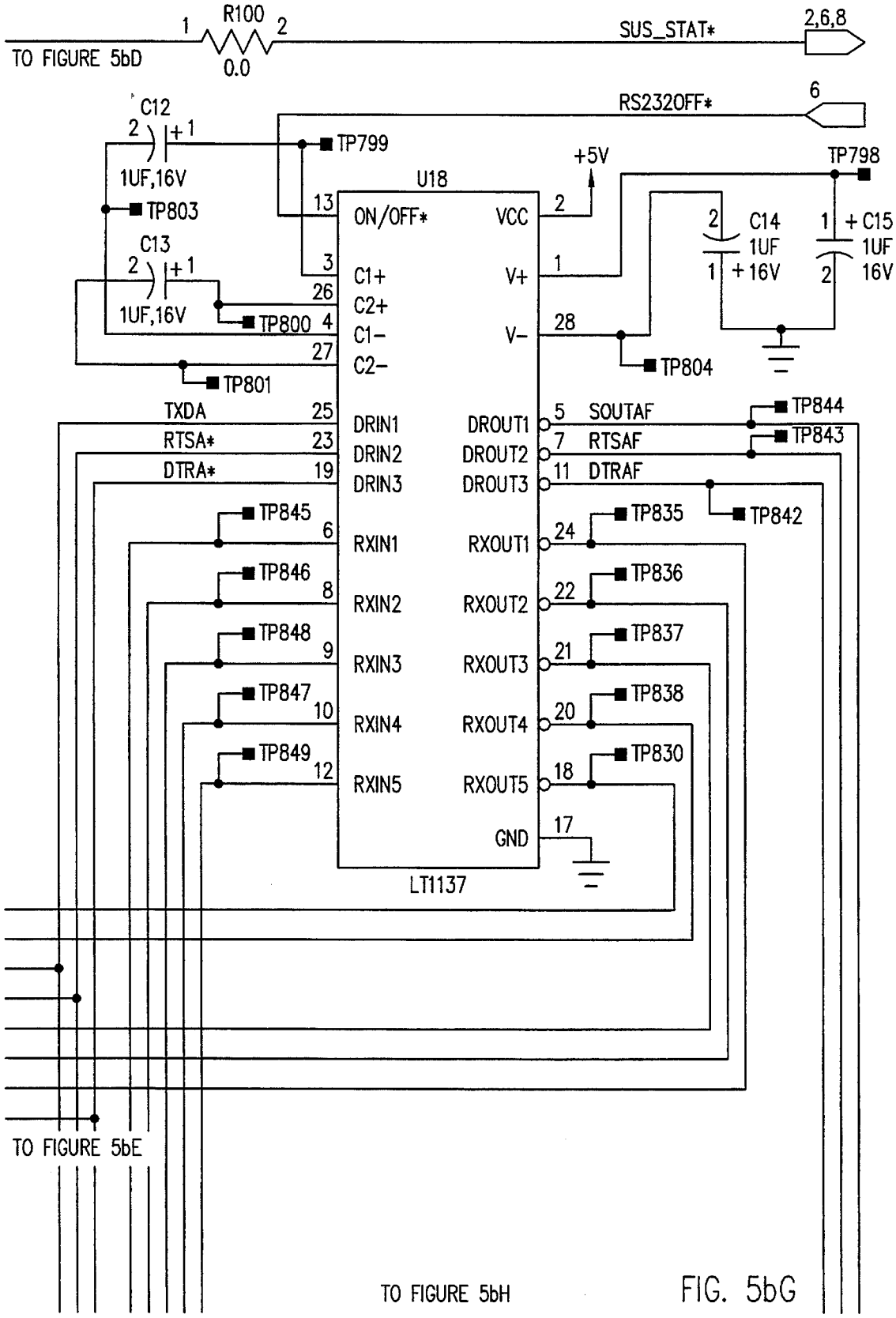

TO FIGURE 5bD
R100
0.0
SUS_STAT*
2,6,8
RS232OFF*
6
C12
1UF,16V
TP799
TP803
C13
1UF,16V
TP800
TP801
U18
ON/OFF*
C1+
C2+
C1-
C2-
VCC
V+
V-
+5V
TP798
C14
1UF
+16V
C15
1UF
16V
TP804
TXDA
RTSA*
DTRA*
DRIN1
DRIN2
DRIN3
RXIN1
RXIN2
RXIN3
RXIN4
RXIN5
DROUT1
DROUT2
DROUT3
RXOUT1
RXOUT2
RXOUT3
RXOUT4
RXOUT5
GND
SOUTAF
RTSAF
DTRAF
TP844
TP843
TP842
TP845
TP846
TP848
TP847
TP849
TP835
TP836
TP837
TP838
TP830
LT1137
TO FIGURE 5bE
TO FIGURE 5bH

FIG. 5bG

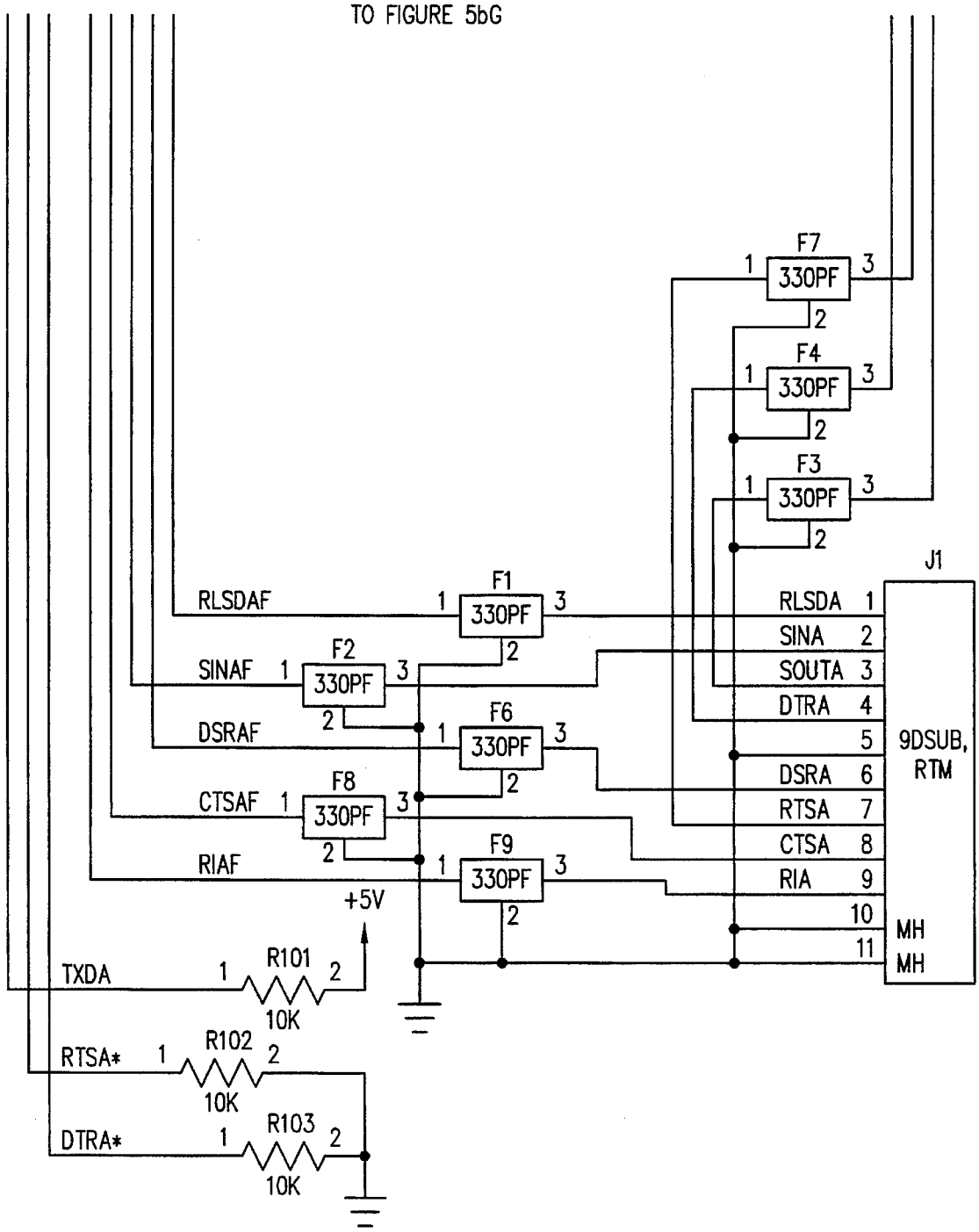

TO FIGURE 5bG
F7
330PF
F4
330PF
F3
330PF
F1
330PF
F2
330PF
F6
330PF
F8
330PF
F9
330PF
RLSDAF
SINAF
DSRAF
CTSAF
RIAF
+5V
TXDA
R101
10K
RTSA*
R102
10K
DTRA*
R103
10K
J1
RLSDA 1
SINA 2
SOUTA 3
DTRA 4
5
DSRA 6
RTSA 7
CTSA 8
RIA 9
10
11
MH
MH
9DSUB, RTM

FIG. 5bH

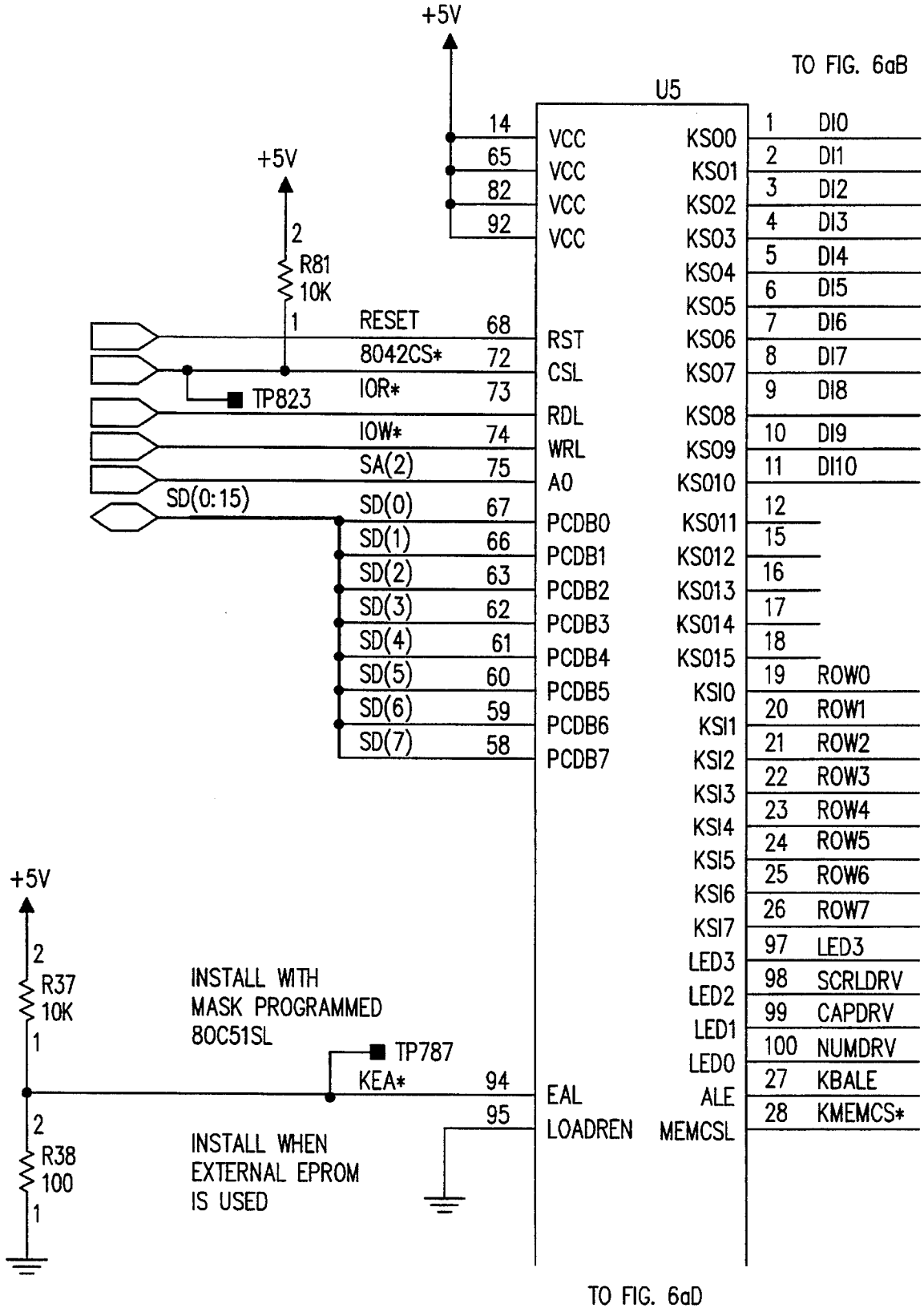

+5V
TO FIG. 6aB
U5
14 VCC
65 VCC
82 VCC
92 VCC
+5V
2
R81
10K
1
RESET 68 RST
8042CS* 72 CSL
IOR* 73 RDL
TP823
IOW* 74 WRL
SA(2) 75 A0
SD(0:15)
SD(0) 67 PCDB0
SD(1) 66 PCDB1
SD(2) 63 PCDB2
SD(3) 62 PCDB3
SD(4) 61 PCDB4
SD(5) 60 PCDB5
SD(6) 59 PCDB6
SD(7) 58 PCDB7
KS00 1 DI0
KS01 2 DI1
KS02 3 DI2
KS03 4 DI3
KS04 5 DI4
KS05 6 DI5
KS06 7 DI6
KS07 8 DI7
9 DI8
KS08
KS09 10 DI9
KS010 11 DI10
KS011 12
KS012 15
KS013 16
KS014 17
KS015 18
KSI0 19 ROW0
KSI1 20 ROW1
KSI2 21 ROW2
KSI3 22 ROW3
KSI4 23 ROW4
KSI5 24 ROW5
KSI6 25 ROW6
KSI7 26 ROW7
LED3 97 LED3
LED2 98 SCRLDRV
LED1 99 CAPDRV
LED0 100 NUMDRV
ALE 27 KBALE
MEMCSL 28 KMEMCS*
+5V
2
R37
10K
1
INSTALL WITH
MASK PROGRAMMED
80C51SL
TP787
KEA* 94 EAL
95 LOADREN
2
R38
100
1
INSTALL WHEN
EXTERNAL EPROM
IS USED
TO FIG. 6aD

FIG. 6aA

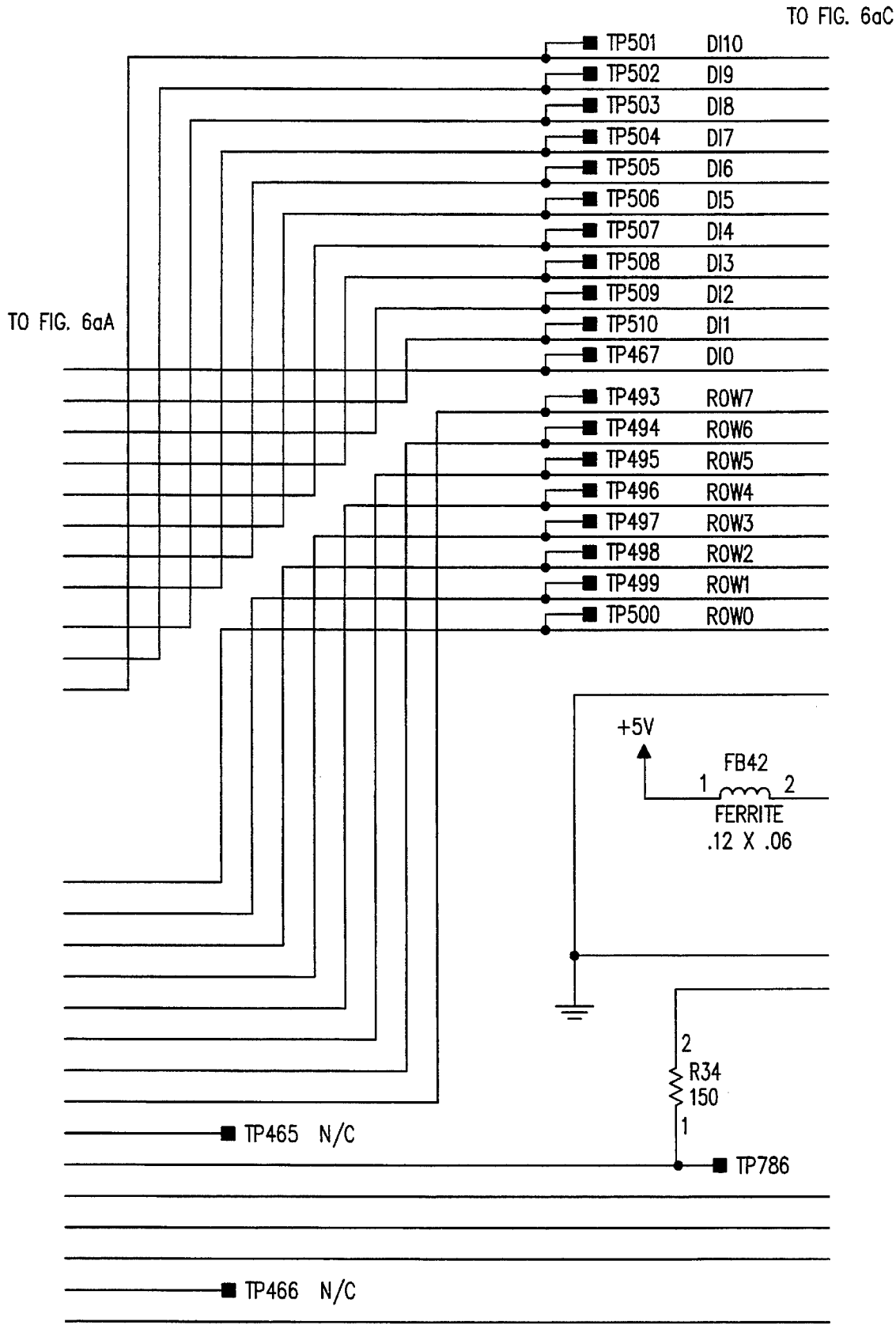

TO FIG. 6aC
TP501 DI10
TP502 DI9
TP503 DI8
TP504 DI7
TP505 DI6
TP506 DI5
TP507 DI4
TP508 DI3
TP509 DI2
TO FIG. 6aA
TP510 DI1
TP467 DI0
TP493 ROW7
TP494 ROW6
TP495 ROW5
TP496 ROW4
TP497 ROW3
TP498 ROW2
TP499 ROW1
TP500 ROW0
+5V
FB42
1
2
FERRITE
.12 X .06
2
R34
150
1
TP465 N/C
TP786
TP466 N/C

FIG. 6aB

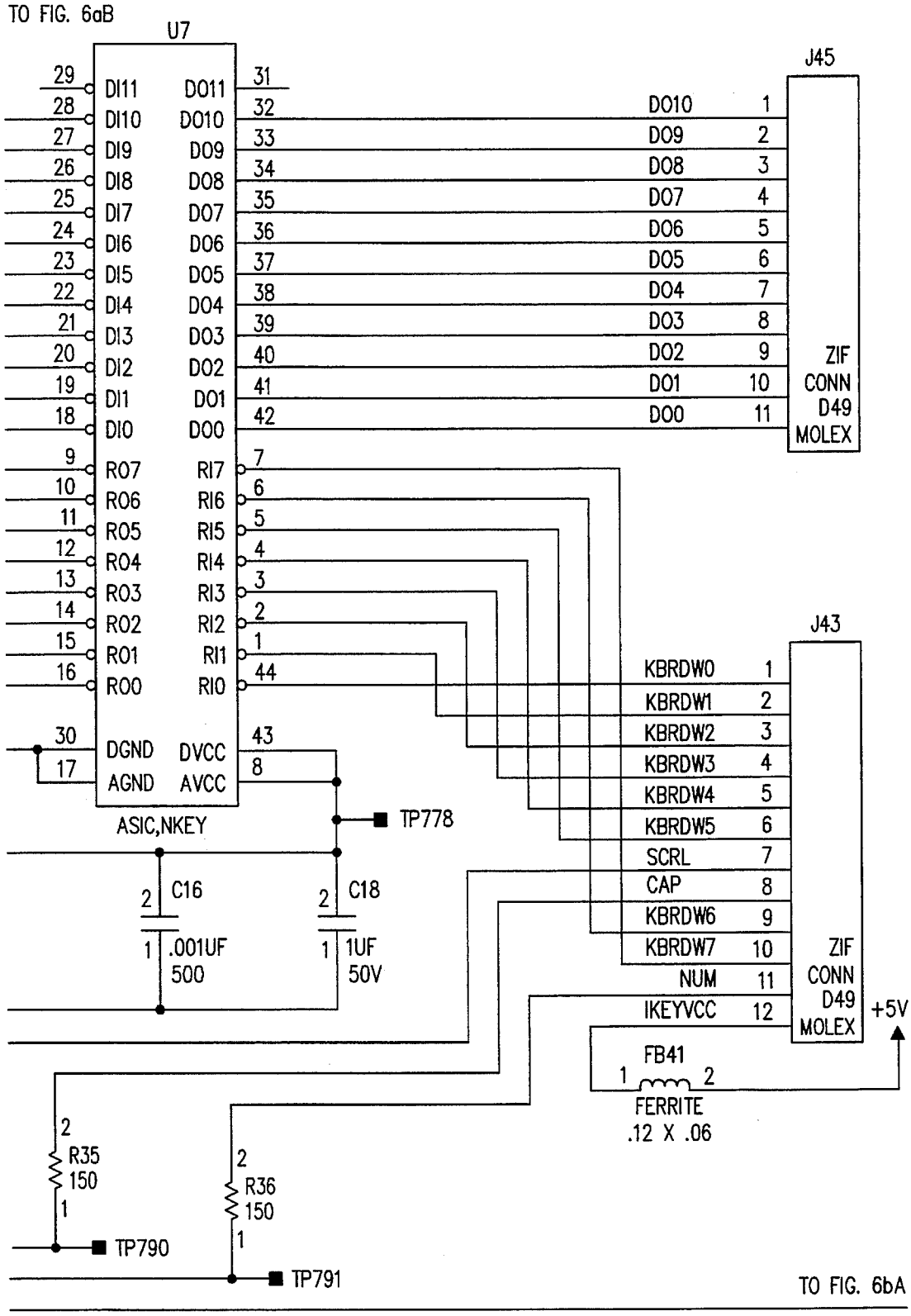

TO FIG. 6aB
U7
DI11
DI10
DI9
DI8
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
DO11
DO10
DO9
DO8
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO00
RO7
RO6
RO5
RO4
RO3
RO2
RO1
RO0
RI7
RI6
RI5
RI4
RI3
RI2
RI1
RI0
DGND
AGND
DVCC
AVCC
ASIC,NKEY
TP778
C16
.001UF
500
C18
1UF
50V
J45
ZIF
CONN
D49
MOLEX
J43
KBRDW0
KBRDW1
KBRDW2
KBRDW3
KBRDW4
KBRDW5
SCRL
CAP
KBRDW6
KBRDW7
NUM
IKEYVCC
+5V
FB41
FERRITE
.12 X .06
R35
150
R36
150
TP790
TP791
TO FIG. 6bA

FIG. 6aC

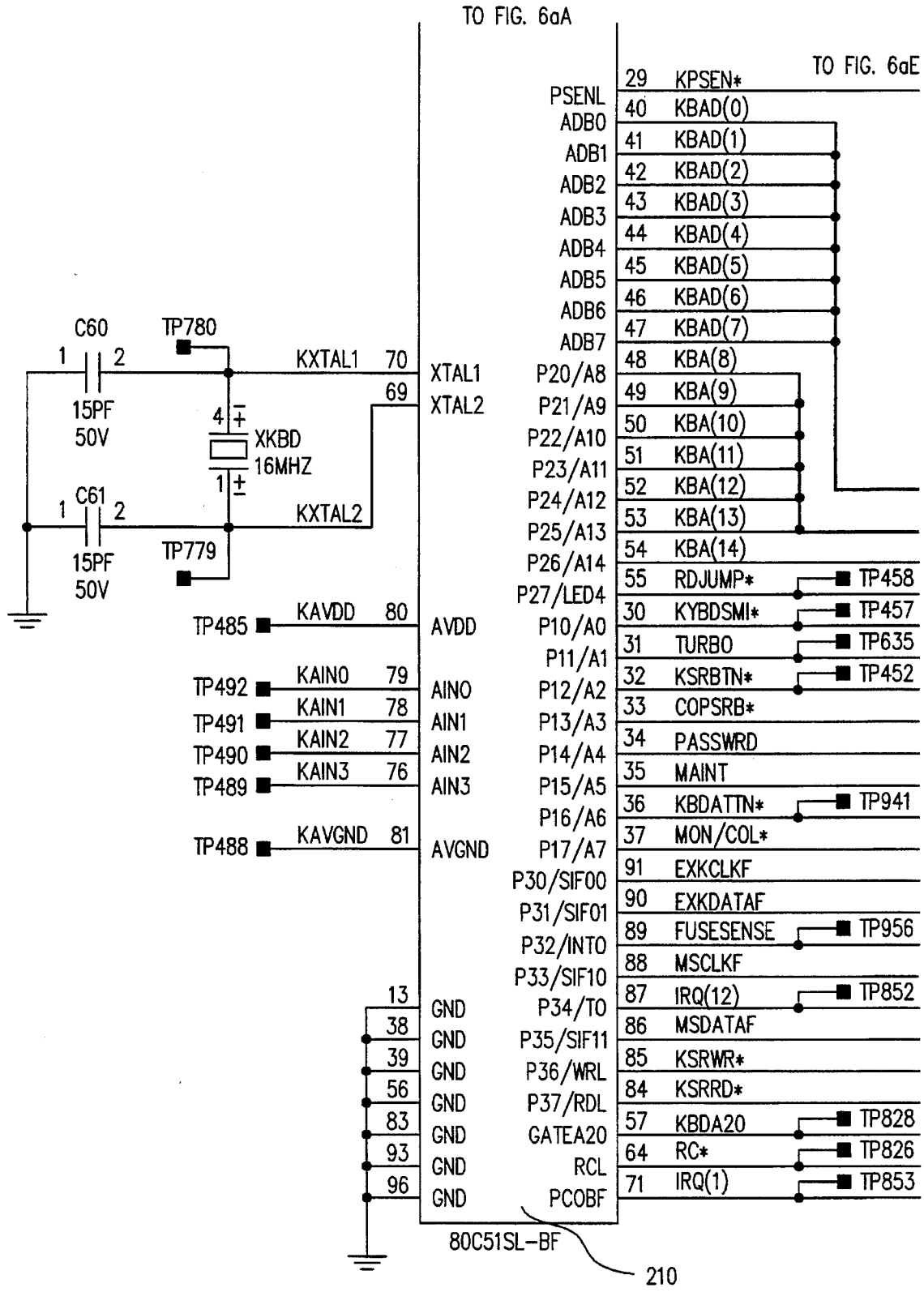

TO FIG. 6aA
TO FIG. 6aE
C60
TP780
15PF
50V
KXTAL1
70
XTAL1
69
XTAL2
XKBD
16MHZ
C61
KXTAL2
TP779
15PF
50V
TP485
KAVDD
80
AVDD
TP492
KAIN0
79
AIN0
TP491
KAIN1
78
AIN1
TP490
KAIN2
77
AIN2
TP489
KAIN3
76
AIN3
TP488
KAVGND
81
AVGND
13
38
39
56
83
93
96
GND
PSENL
ADB0
ADB1
ADB2
ADB3
ADB4
ADB5
ADB6
ADB7
P20/A8
P21/A9
P22/A10
P23/A11
P24/A12
P25/A13
P26/A14
P27/LED4
P10/A0
P11/A1
P12/A2
P13/A3
P14/A4
P15/A5
P16/A6
P17/A7
P30/SIF00
P31/SIF01
P32/INT0
P33/SIF10
P34/T0
P35/SIF11
P36/WRL
P37/RDL
GATEA20
RCL
PCOBF
29 KPSEN*
40 KBAD(0)
41 KBAD(1)
42 KBAD(2)
43 KBAD(3)
44 KBAD(4)
45 KBAD(5)
46 KBAD(6)
47 KBAD(7)
48 KBA(8)
49 KBA(9)
50 KBA(10)
51 KBA(11)
52 KBA(12)
53 KBA(13)
54 KBA(14)
55 RDJUMP*
30 KYBDSMI*
31 TURBO
32 KSRBTN*
33 COPSRB*
34 PASSWRD
35 MAINT
36 KBDATTN*
37 MON/COL*
91 EXKCLKF
90 EXKDATAF
89 FUSESENSE
88 MSCLKF
87 IRQ(12)
86 MSDATAF
85 KSRWR*
84 KSRRD*
57 KBDA20
64 RC*
71 IRQ(1)
TP458
TP457
TP635
TP452
TP941
TP956
TP852
TP828
TP826
TP853
80C51SL-BF
210

FIG. 6aD

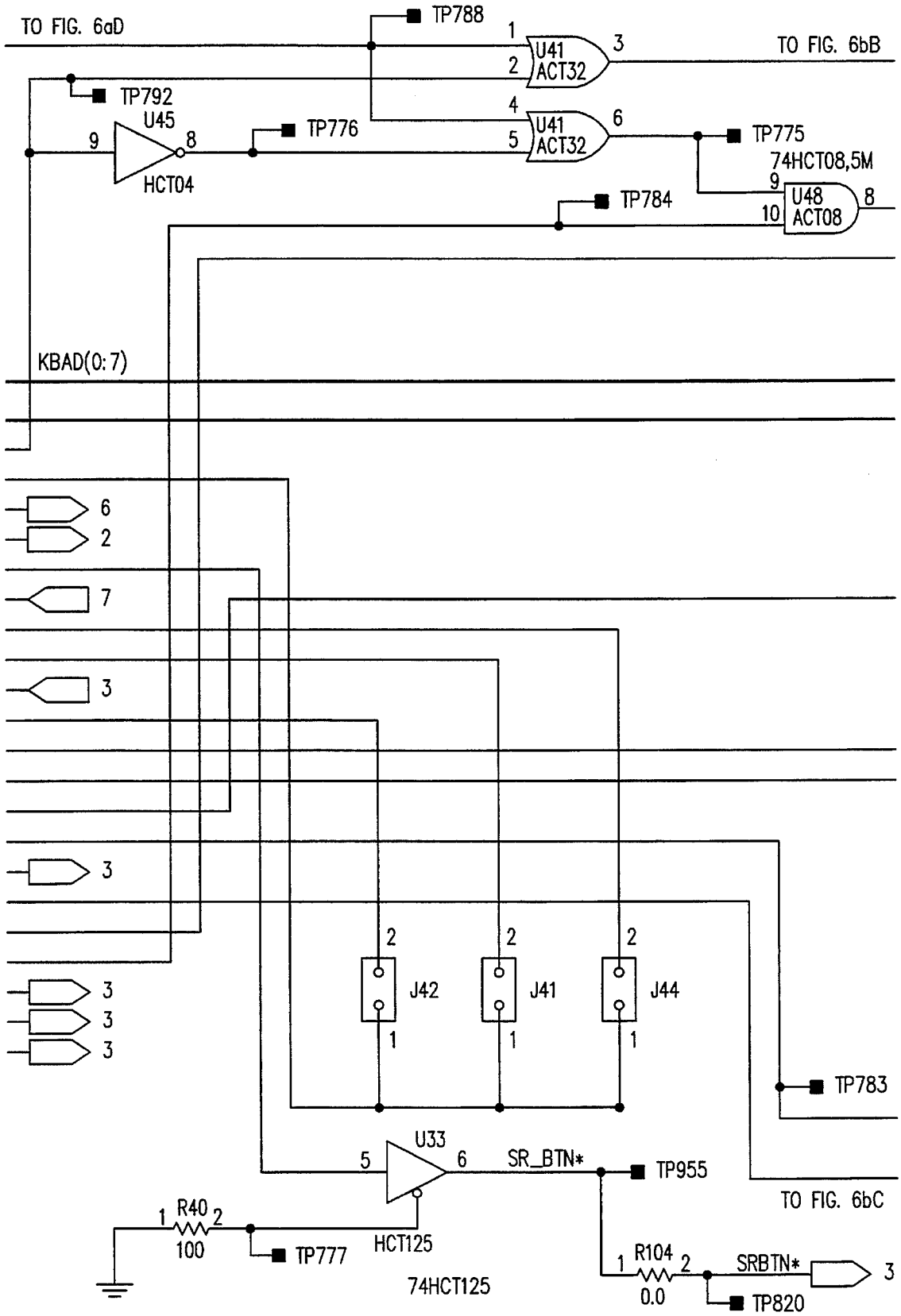

TO FIG. 6aD
TP788
U41
ACT32
TO FIG. 6bB
TP792
U45
HCT04
TP776
TP775
74HCT08,5M
U48
ACT08
TP784
KBAD(0:7)
J42
J41
J44
TP783
U33
SR_BTN*
TP955
TO FIG. 6bC
R40
100
TP777
HCT125
74HCT125
R104
0.0
SRBTN*
TP820

FIG. 6aE

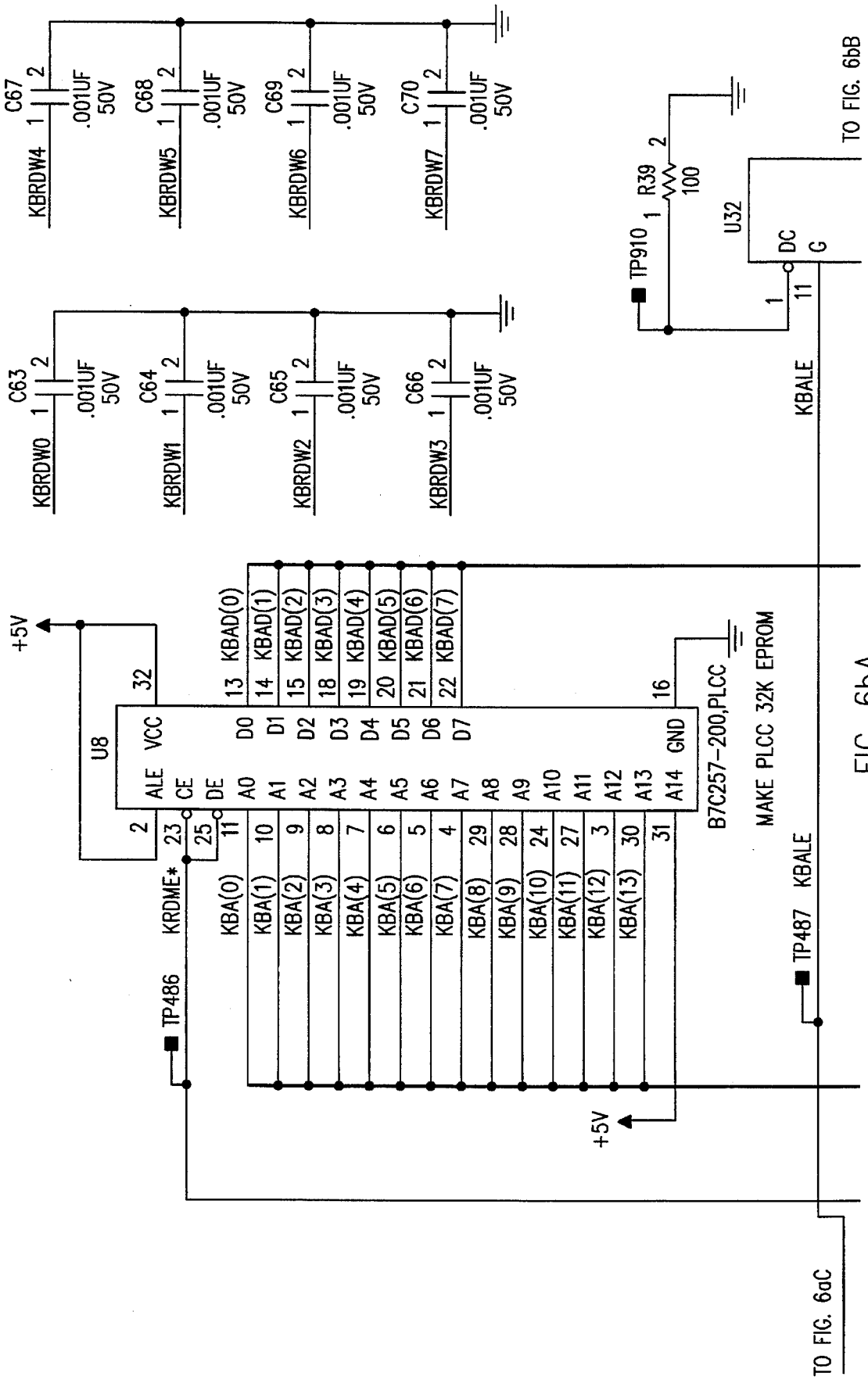

C67
.001UF
50V
KBRDW4
C68
.001UF
50V
KBRDW5
C69
.001UF
50V
KBRDW6
C70
.001UF
50V
KBRDW7
C63
.001UF
50V
KBRDW0
C64
.001UF
50V
KBRDW1
C65
.001UF
50V
KBRDW2
C66
.001UF
50V
KBRDW3
TP910
R39
100
U32
DC
G
TO FIG. 6bB
KBALE
+5V
U8
VCC
ALE
CE
DE
KRDME*
TP486
KBAD(0)
KBAD(1)
KBAD(2)
KBAD(3)
KBAD(4)
KBAD(5)
KBAD(6)
KBAD(7)
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
GND
KBA(0)
KBA(1)
KBA(2)
KBA(3)
KBA(4)
KBA(5)
KBA(6)
KBA(7)
KBA(8)
KBA(9)
KBA(10)
KBA(11)
KBA(12)
KBA(13)
B7C257-200,PLCC
MAKE PLCC 32K EPROM
TP487
KBALE
TO FIG. 6aC

FIG. 6bA

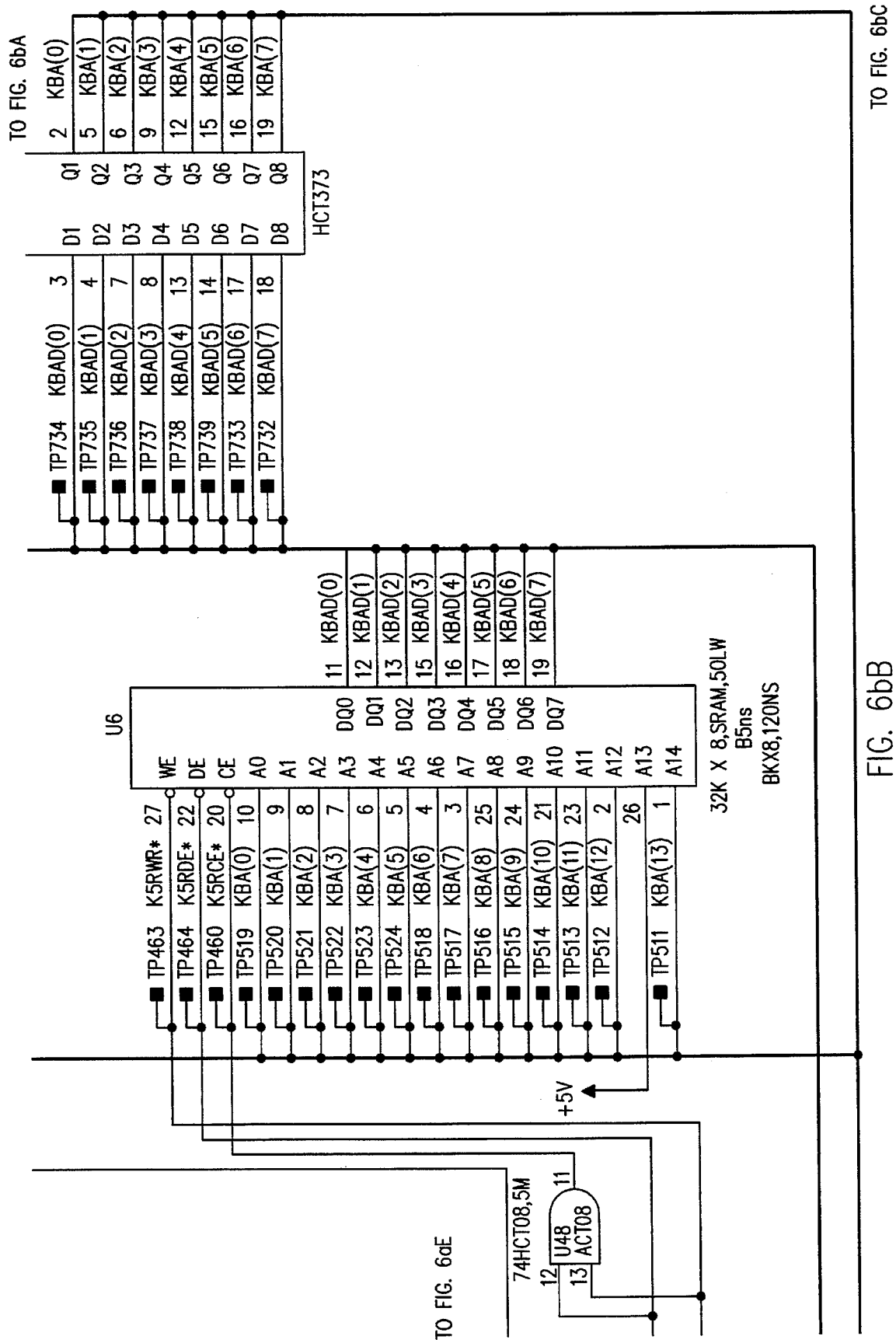

TO FIG. 6bA
TO FIG. 6bC
TO FIG. 6oE
HCT373
U6
32K X 8,SRAM,50LW
B5ns
BKX8,120NS
74HCT08,5M
U48
ACT08
+5V
TP734 KBAD(0) 3 D1 Q1 2 KBA(0)
TP735 KBAD(1) 4 D2 Q2 5 KBA(1)
TP736 KBAD(2) 7 D3 Q3 6 KBA(2)
TP737 KBAD(3) 8 D4 Q4 9 KBA(3)
TP738 KBAD(4) 13 D5 Q5 12 KBA(4)
TP739 KBAD(5) 14 D6 Q6 15 KBA(5)
TP733 KBAD(6) 17 D7 Q7 16 KBA(6)
TP732 KBAD(7) 18 D8 Q8 19 KBA(7)
TP463 K5RWR* 27 WE
TP464 K5RDE* 22 DE
TP460 K5RCE* 20 CE
TP519 KBA(0) 10 A0
TP520 KBA(1) 9 A1
TP521 KBA(2) 8 A2
TP522 KBA(3) 7 A3
TP523 KBA(4) 6 A4
TP524 KBA(5) 5 A5
TP518 KBA(6) 4 A6
TP517 KBA(7) 3 A7
TP516 KBA(8) 25 A8
TP515 KBA(9) 24 A9
TP514 KBA(10) 21 A10
TP513 KBA(11) 23 A11
TP512 KBA(12) 2 A12
26 A13
TP511 KBA(13) 1 A14
11 KBAD(0) DQ0
12 KBAD(1) DQ1
13 KBAD(2) DQ2
15 KBAD(3) DQ3
16 KBAD(4) DQ4
17 KBAD(5) DQ5
18 KBAD(6) DQ6
19 KBAD(7) DQ7
12 13 11

FIG. 6bB

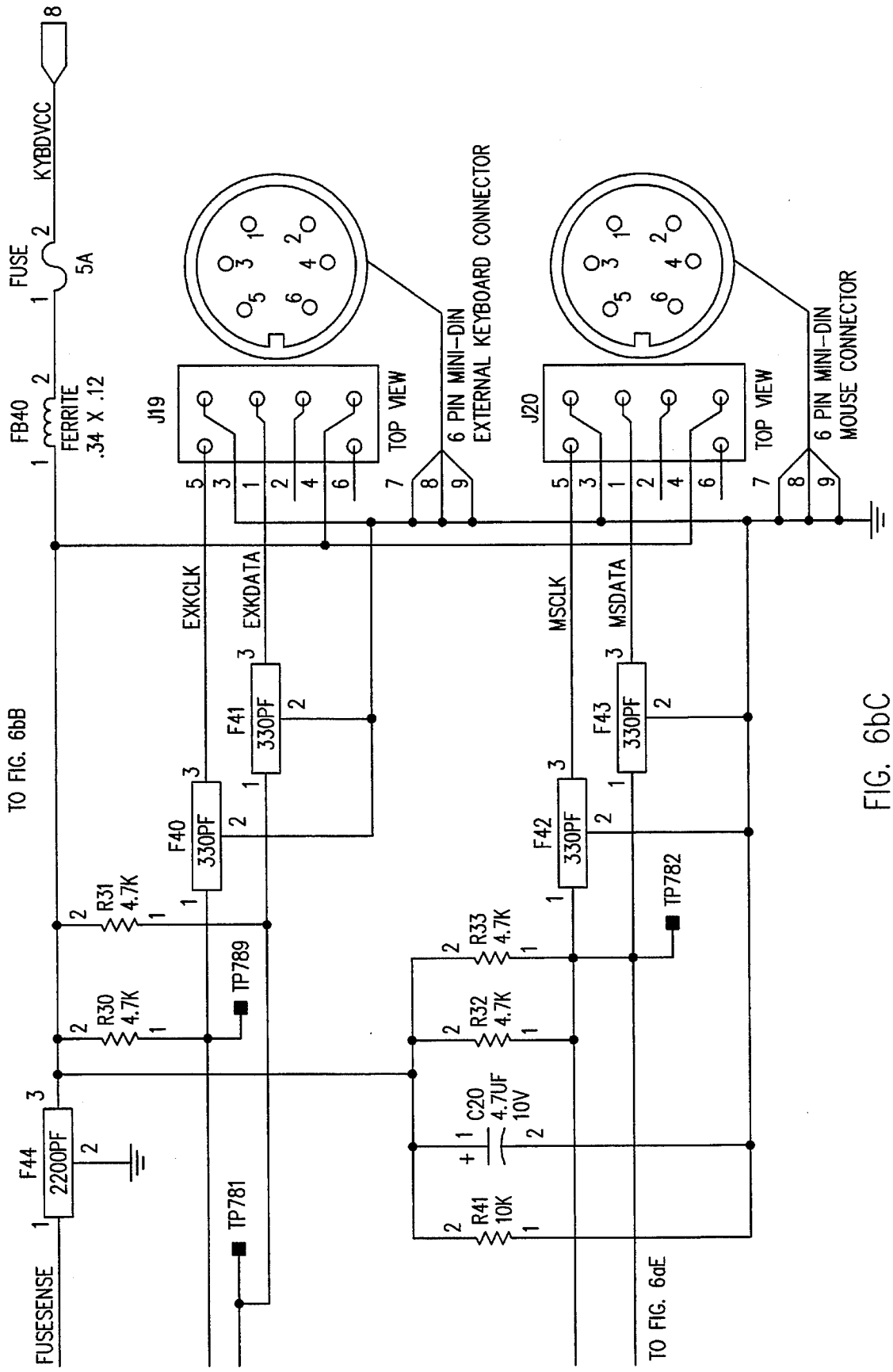

KYBDVCC
FUSE
5A
FB40
FERRITE
.34 X .12
J19
TOP VIEW
6 PIN MINI-DIN
EXTERNAL KEYBOARD CONNECTOR
J20
TOP VIEW
6 PIN MINI-DIN
MOUSE CONNECTOR
EXKCLK
EXKDATA
MSCLK
MSDATA
F41
330PF
F43
330PF
F40
330PF
F42
330PF
TO FIG. 6bB
R31
4.7K
R30
4.7K
TP789
R33
4.7K
R32
4.7K
TP782
C20
4.7UF
10V
F44
2200PF
TP781
R41
10K
FUSESENSE
TO FIG. 6aE

FIG. 6bC

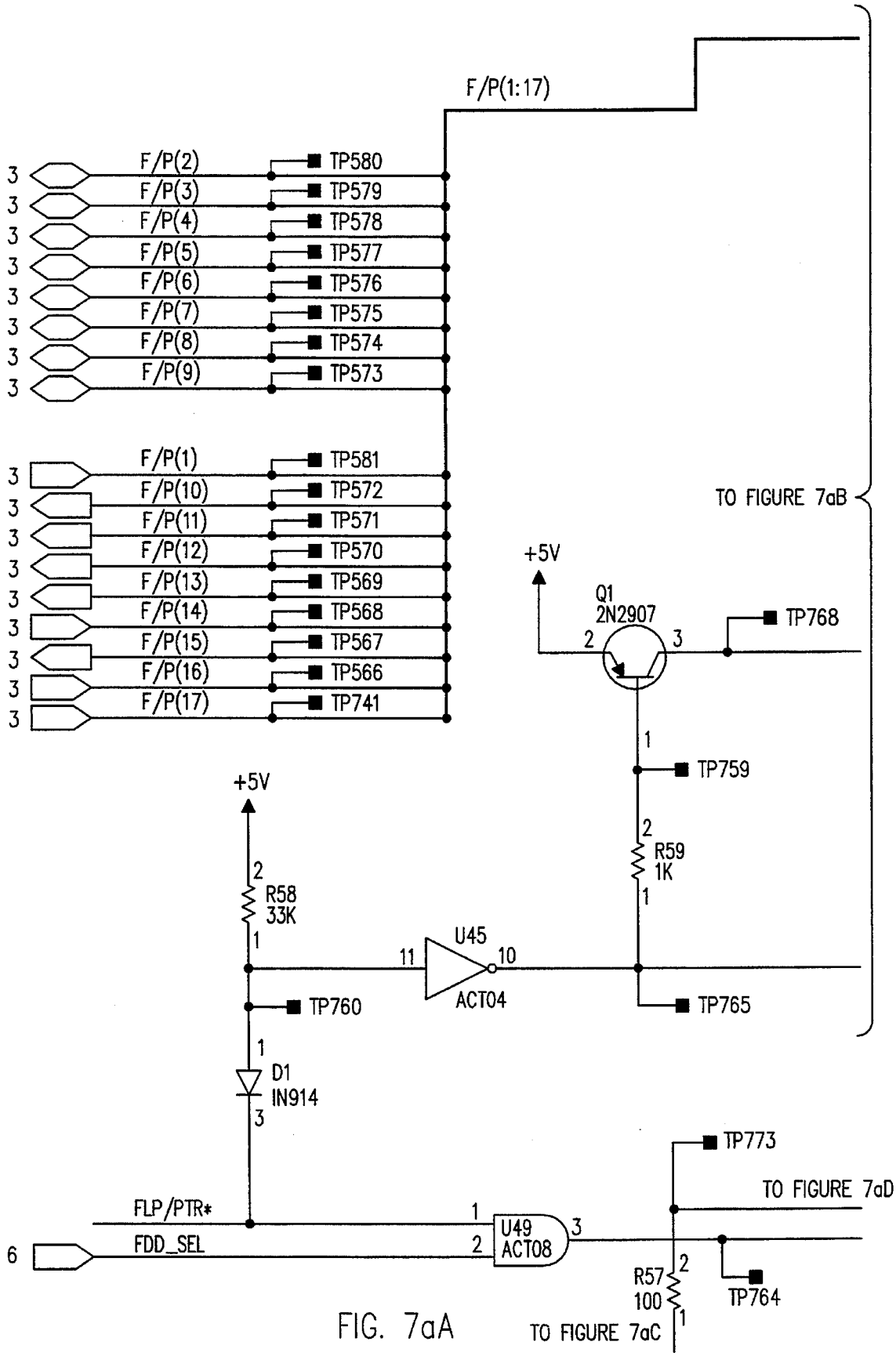

F/P(1:17)
F/P(2)
F/P(3)
F/P(4)
F/P(5)
F/P(6)
F/P(7)
F/P(8)
F/P(9)
TP580
TP579
TP578
TP577
TP576
TP575
TP574
TP573
F/P(1)
F/P(10)
F/P(11)
F/P(12)
F/P(13)
F/P(14)
F/P(15)
F/P(16)
F/P(17)
TP581
TP572
TP571
TP570
TP569
TP568
TP567
TP566
TP741
TO FIGURE 7aB
+5V
Q1
2N2907
TP768
TP759
R59
1K
R58
33K
U45
ACT04
TP760
TP765
D1
IN914
TP773
TO FIGURE 7aD
FLP/PTR*
FDD_SEL
U49
ACT08
R57
100
TP764
TO FIGURE 7aC

FIG. 7aA

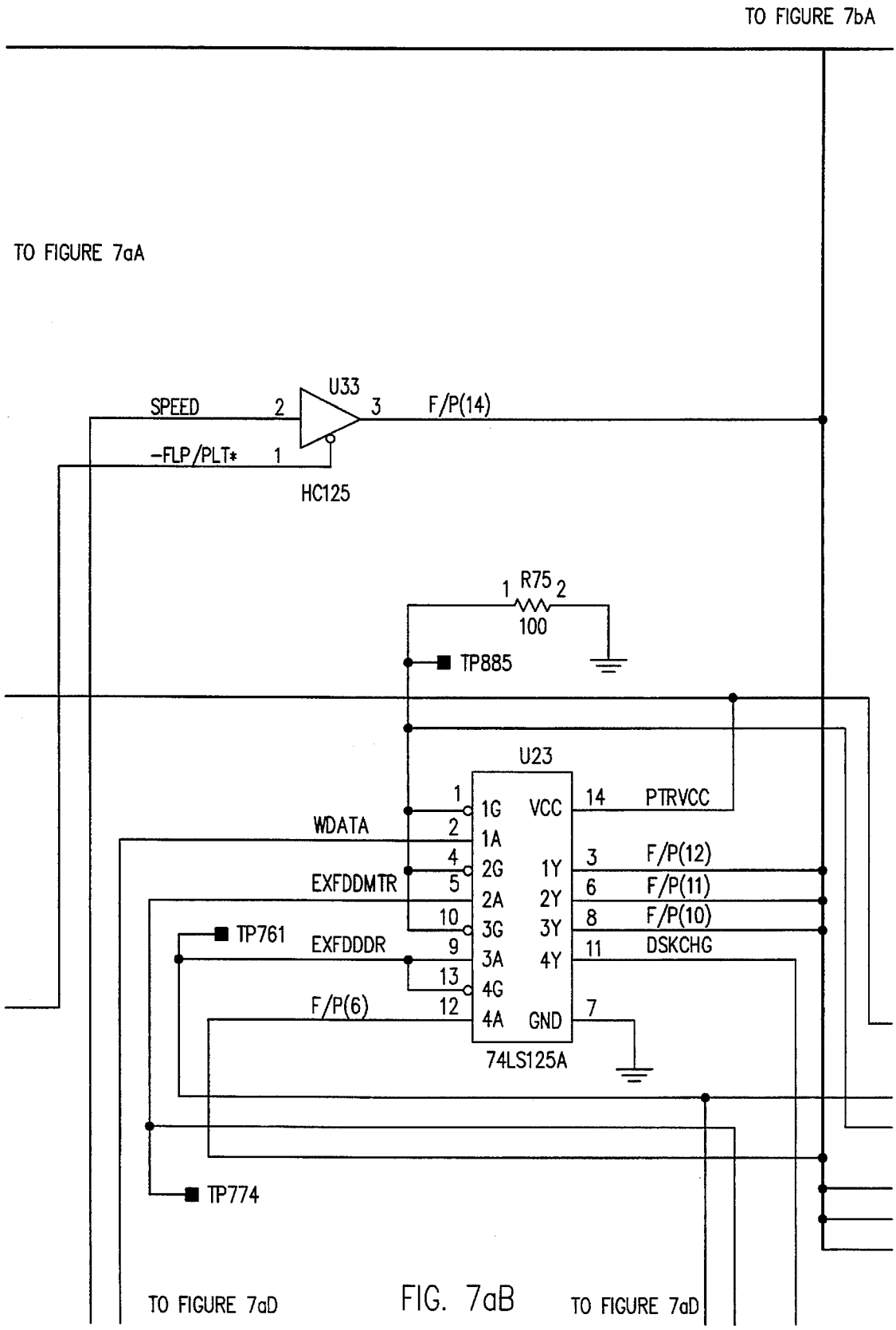

TO FIGURE 7bA
TO FIGURE 7aA
U33
SPEED
2
3
F/P(14)
-FLP/PLT*
1
HC125
R75
1
2
100
TP885
U23
1G
VCC
14
PTRVCC
WDATA
2
1A
4
2G
1Y
3
F/P(12)
EXFDDMTR
5
2A
2Y
6
F/P(11)
10
3G
3Y
8
F/P(10)
TP761
EXFDDDR
9
3A
4Y
11
DSKCHG
13
4G
F/P(6)
12
4A
GND
7
74LS125A
TP774
TO FIGURE 7aD
TO FIGURE 7aD

FIG. 7aB

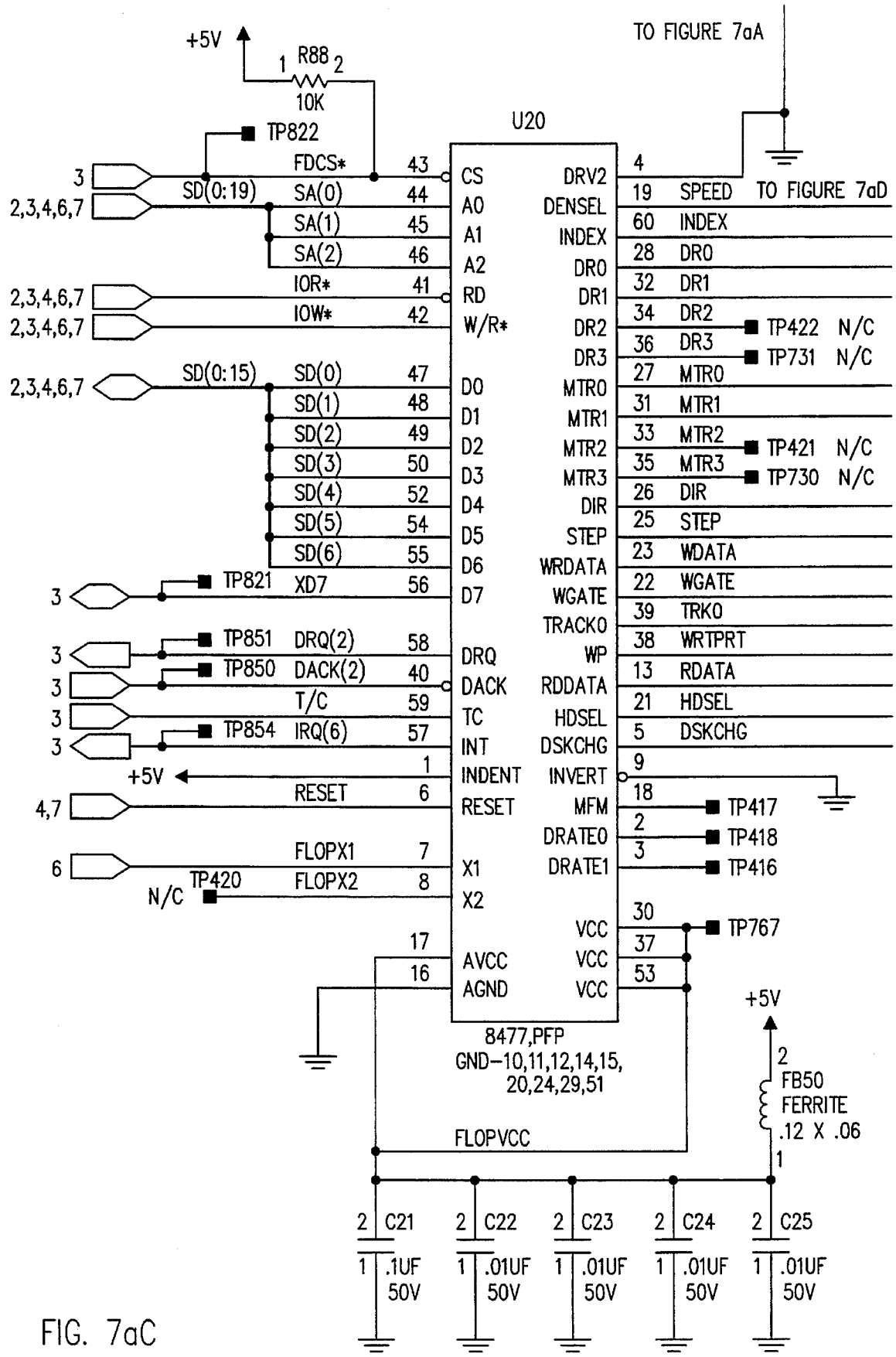

+5V
R88
10K
TO FIGURE 7aA
TP822
U20
FDCS*
SD(0:19)
SA(0)
SA(1)
SA(2)
IOR*
IOW*
SD(0:15)
SD(0)
SD(1)
SD(2)
SD(3)
SD(4)
SD(5)
SD(6)
TP821
XD7
TP851
DRQ(2)
TP850
DACK(2)
T/C
TP854
IRQ(6)
RESET
FLOPX1
FLOPX2
TP420
N/C
CS
A0
A1
A2
RD
W/R*
D0
D1
D2
D3
D4
D5
D6
D7
DRQ
DACK
TC
INT
INDENT
RESET
X1
X2
AVCC
AGND
DRV2
DENSEL
INDEX
DR0
DR1
DR2
DR3
MTR0
MTR1
MTR2
MTR3
DIR
STEP
WRDATA
WGATE
TRACK0
WP
RDDATA
HDSEL
DSKCHG
INVERT
MFM
DRATE0
DRATE1
VCC
VCC
VCC
SPEED
TO FIGURE 7aD
INDEX
DR0
DR1
DR2
DR3
MTR0
MTR1
MTR2
MTR3
DIR
STEP
WDATA
WGATE
TRK0
WRTPRT
RDATA
HDSEL
DSKCHG
TP422 N/C
TP731 N/C
TP421 N/C
TP730 N/C
TP417
TP418
TP416
TP767
8477,PFP
GND–10,11,12,14,15,
20,24,29,51
FB50
FERRITE
.12 X .06
FLOPVCC
C21 .1UF 50V
C22 .01UF 50V
C23 .01UF 50V
C24 .01UF 50V
C25 .01UF 50V

FIG. 7aC

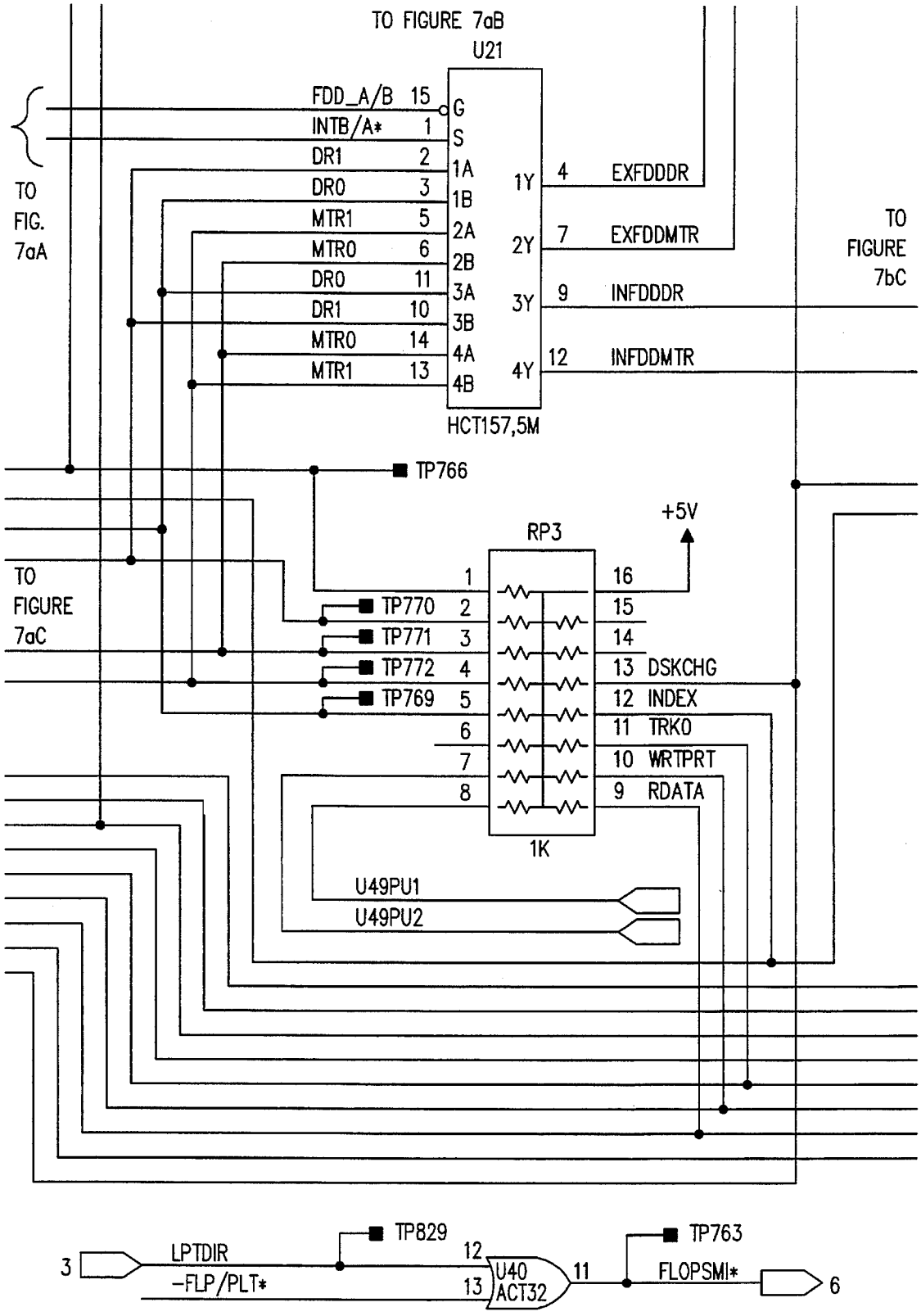

TO FIGURE 7aB
U21
FDD_A/B 15
INTB/A* 1
DR1 2
DR0 3
MTR1 5
MTR0 6
DR0 11
DR1 10
MTR0 14
MTR1 13
G
S
1A
1B
2A
2B
3A
3B
4A
4B
1Y 4 EXFDDDR
2Y 7 EXFDDMTR
3Y 9 INFDDDR
4Y 12 INFDDMTR
HCT157,5M
TO FIG. 7aA
TO FIGURE 7bC
TP766
+5V
RP3
TO FIGURE 7aC
TP770
TP771
TP772
TP769
13 DSKCHG
12 INDEX
11 TRK0
10 WRTPRT
9 RDATA
1K
U49PU1
U49PU2
TP829
3
LPTDIR
-FLP/PLT*
U40 ACT32
TP763
FLOPSMI*
6

FIG. 7aD

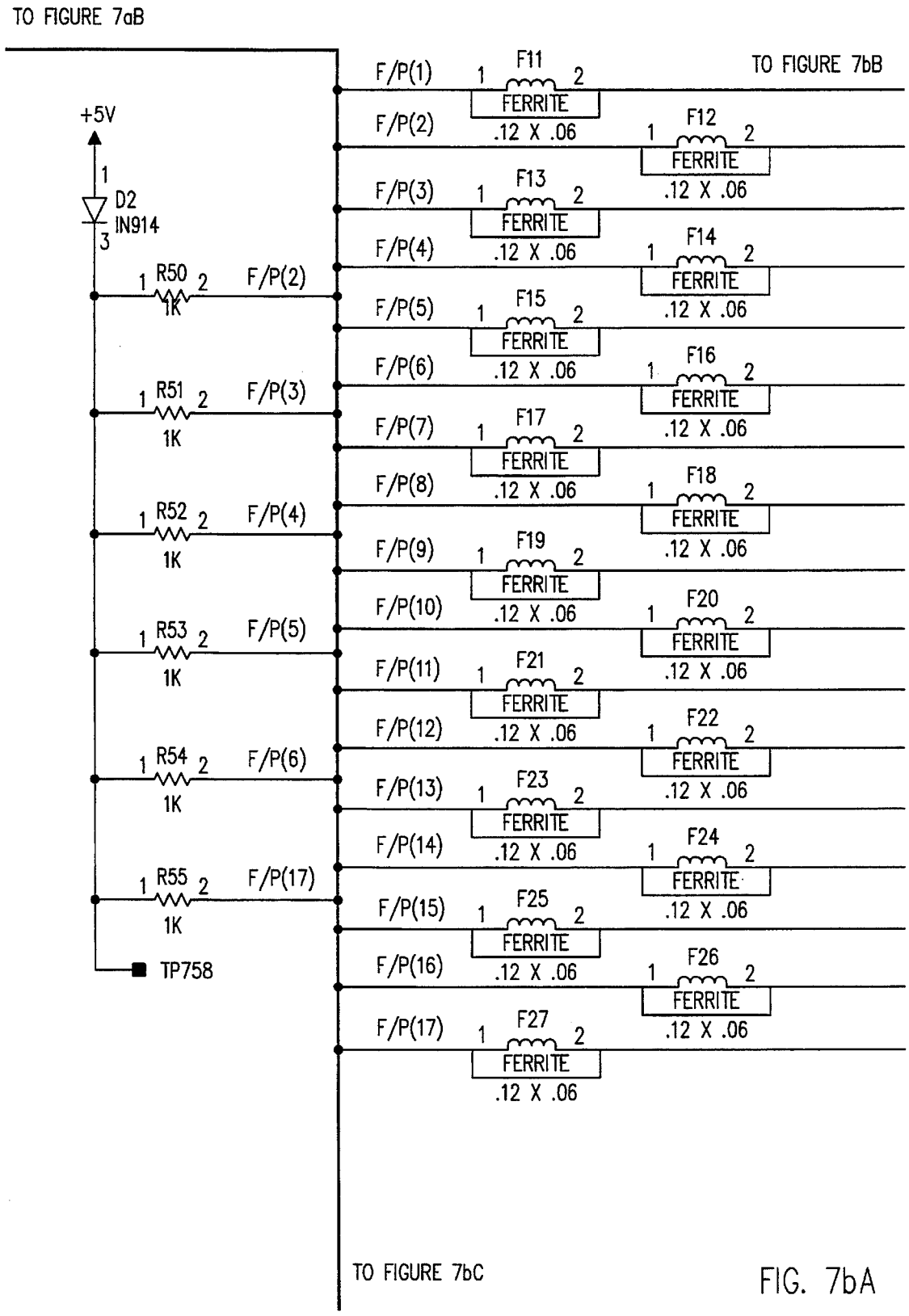

TO FIGURE 7aB
+5V
D2
IN914
R50
1K
R51
1K
R52
1K
R53
1K
R54
1K
R55
1K
TP758
F/P(1)
F/P(2)
F/P(3)
F/P(4)
F/P(5)
F/P(6)
F/P(7)
F/P(8)
F/P(9)
F/P(10)
F/P(11)
F/P(12)
F/P(13)
F/P(14)
F/P(15)
F/P(16)
F/P(17)
F11
F12
F13
F14
F15
F16
F17
F18
F19
F20
F21
F22
F23
F24
F25
F26
F27
FERRITE
.12 X .06
TO FIGURE 7bB
TO FIGURE 7bC

FIG. 7bA

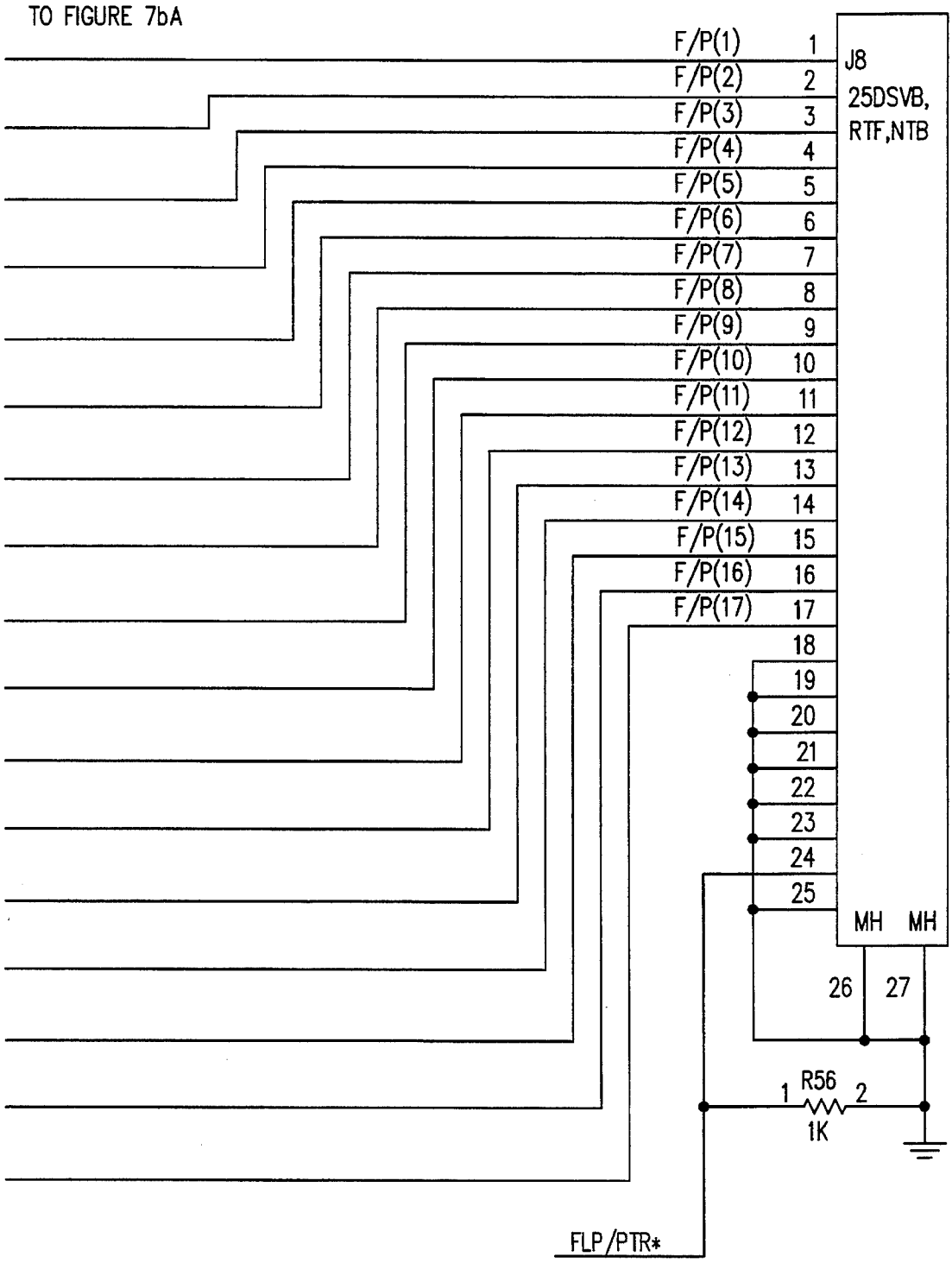

TO FIGURE 7bA
F/P(1)
F/P(2)
F/P(3)
F/P(4)
F/P(5)
F/P(6)
F/P(7)
F/P(8)
F/P(9)
F/P(10)
F/P(11)
F/P(12)
F/P(13)
F/P(14)
F/P(15)
F/P(16)
F/P(17)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
J8
25DSVB,
RTF,NTB
MH
MH
26
27
R56
1
2
1K
FLP/PTR*

FIG. 7bB

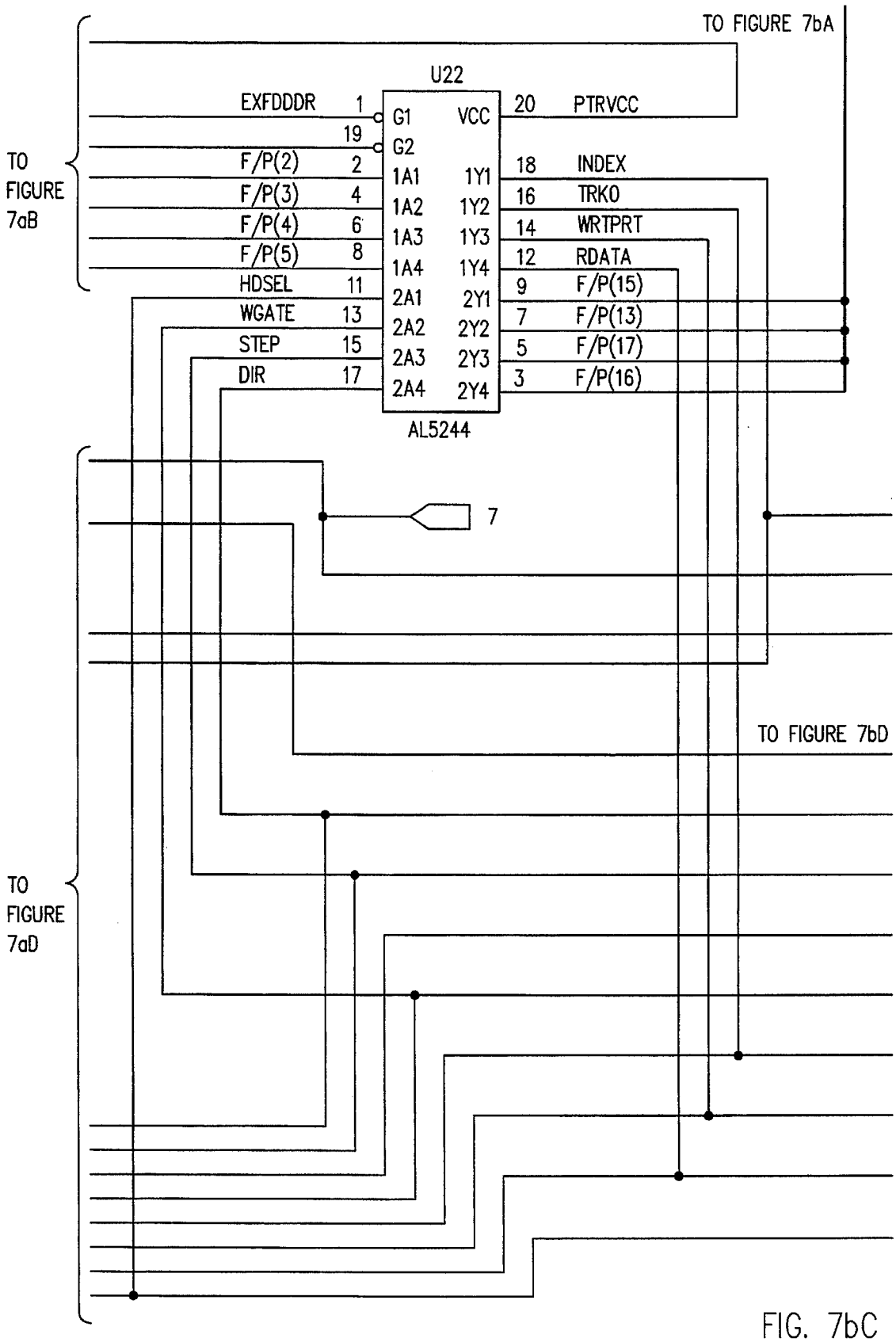
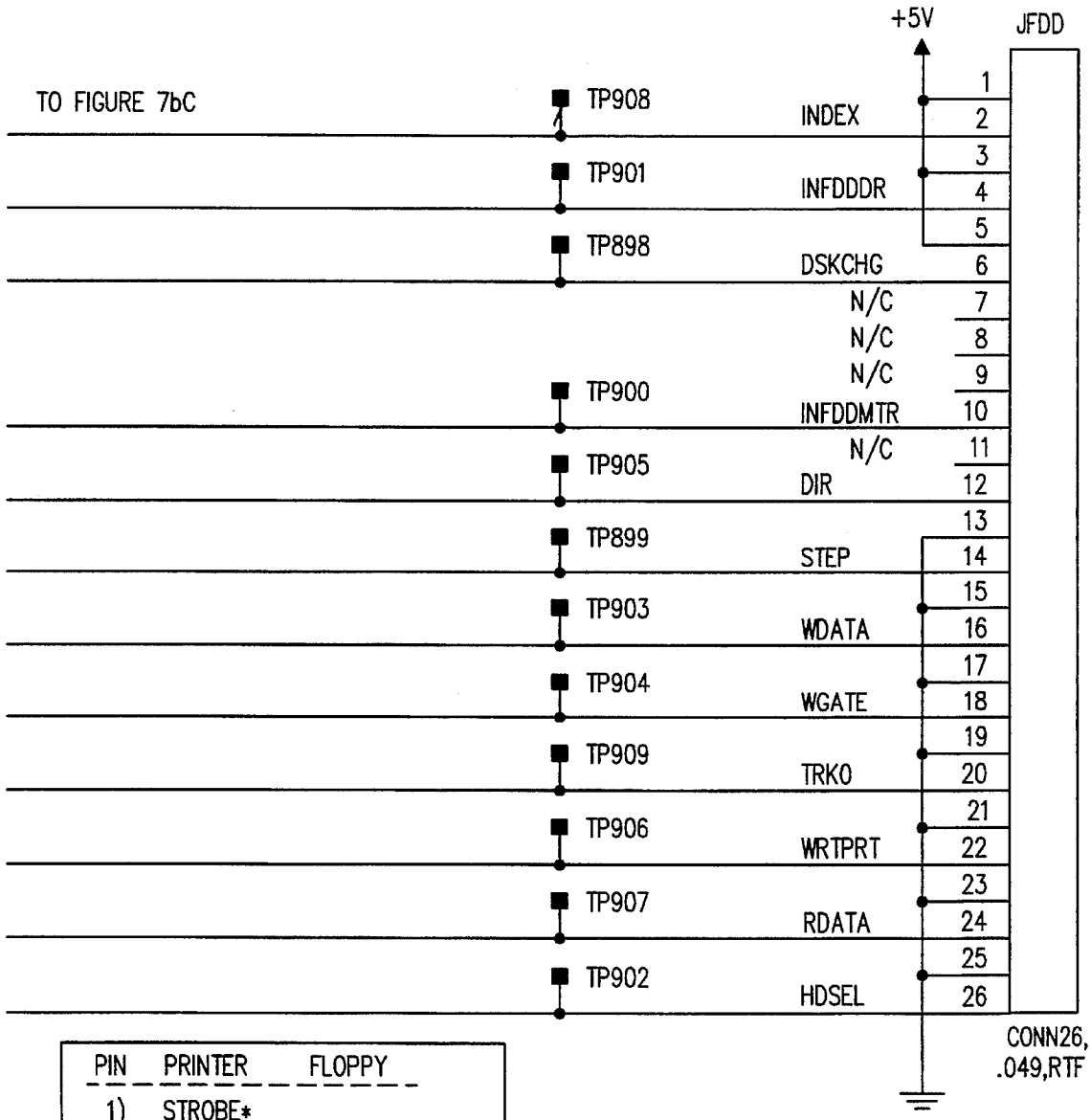

TO FIGURE 7bA
U22
EXFDDDR 1 G1
19 G2
VCC 20 PTRVCC
TO FIGURE 7aB
F/P(2) 2 1A1
F/P(3) 4 1A2
F/P(4) 6 1A3
F/P(5) 8 1A4
HDSEL 11 2A1
WGATE 13 2A2
STEP 15 2A3
DIR 17 2A4
1Y1 18 INDEX
1Y2 16 TRK0
1Y3 14 WRTPRT
1Y4 12 RDATA
2Y1 9 F/P(15)
2Y2 7 F/P(13)
2Y3 5 F/P(17)
2Y4 3 F/P(16)
AL5244
7
TO FIGURE 7bD
TO FIGURE 7aD

FIG. 7bC



PIN	PRINTER	FLOPPY
1)	STROBE*	
2)	DATA 0	INDEX
3)	DATA 1	TRACK 0
4)	DATA 2	WRITE PROTECT
5)	DATA 3	READ DATA
6)	DATA 4	DISK CHANGE
7)	DATA 5	TAPE SELECT
8)	DATA 6	
9)	DATA 7	
10)	ACK*	DRIVE SELECT
11)	BUSY	MOTOR ON
12)	PE	WRITE DATA
13)	SELECT	WRITE GATE
14)	AUTO FEED*	SPEED
15)	ERROR*	HEAD SELECT
16)	INIT*	DIR
17)	SEL IN*	STEP

FIG. 7bD

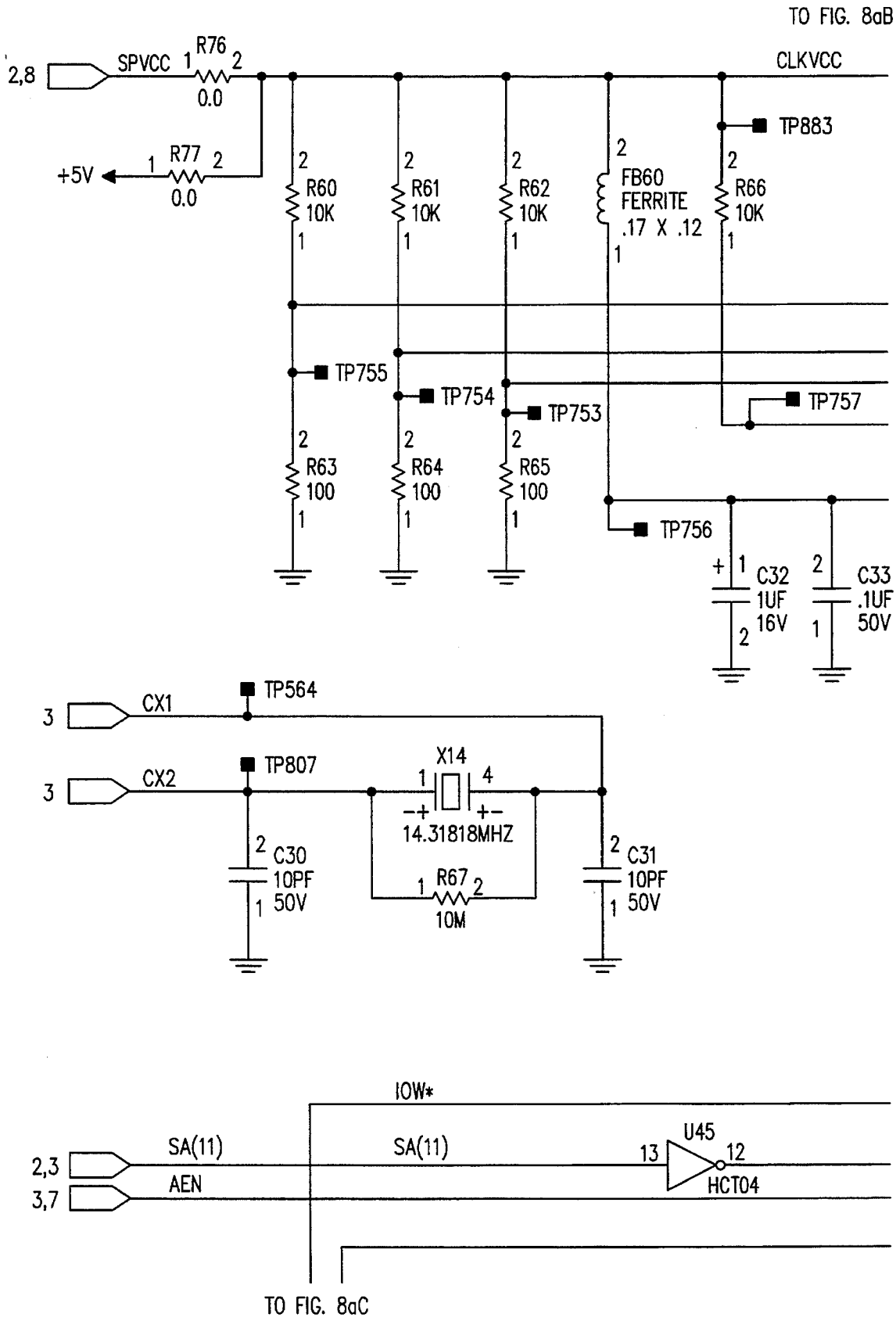

TO FIG. 8aB
2,8
SPVCC
R76
0.0
CLKVCC
TP883
+5V
R77
0.0
R60
10K
R61
10K
R62
10K
FB60
FERRITE
.17 X .12
R66
10K
TP755
TP754
TP753
TP757
R63
100
R64
100
R65
100
TP756
C32
1UF
16V
C33
.1UF
50V
CX1
TP564
CX2
TP807
X14
14.31818MHZ
C30
10PF
50V
R67
10M
C31
10PF
50V
IOW*
SA(11)
AEN
U45
HCT04
TO FIG. 8aC

FIG. 8aA

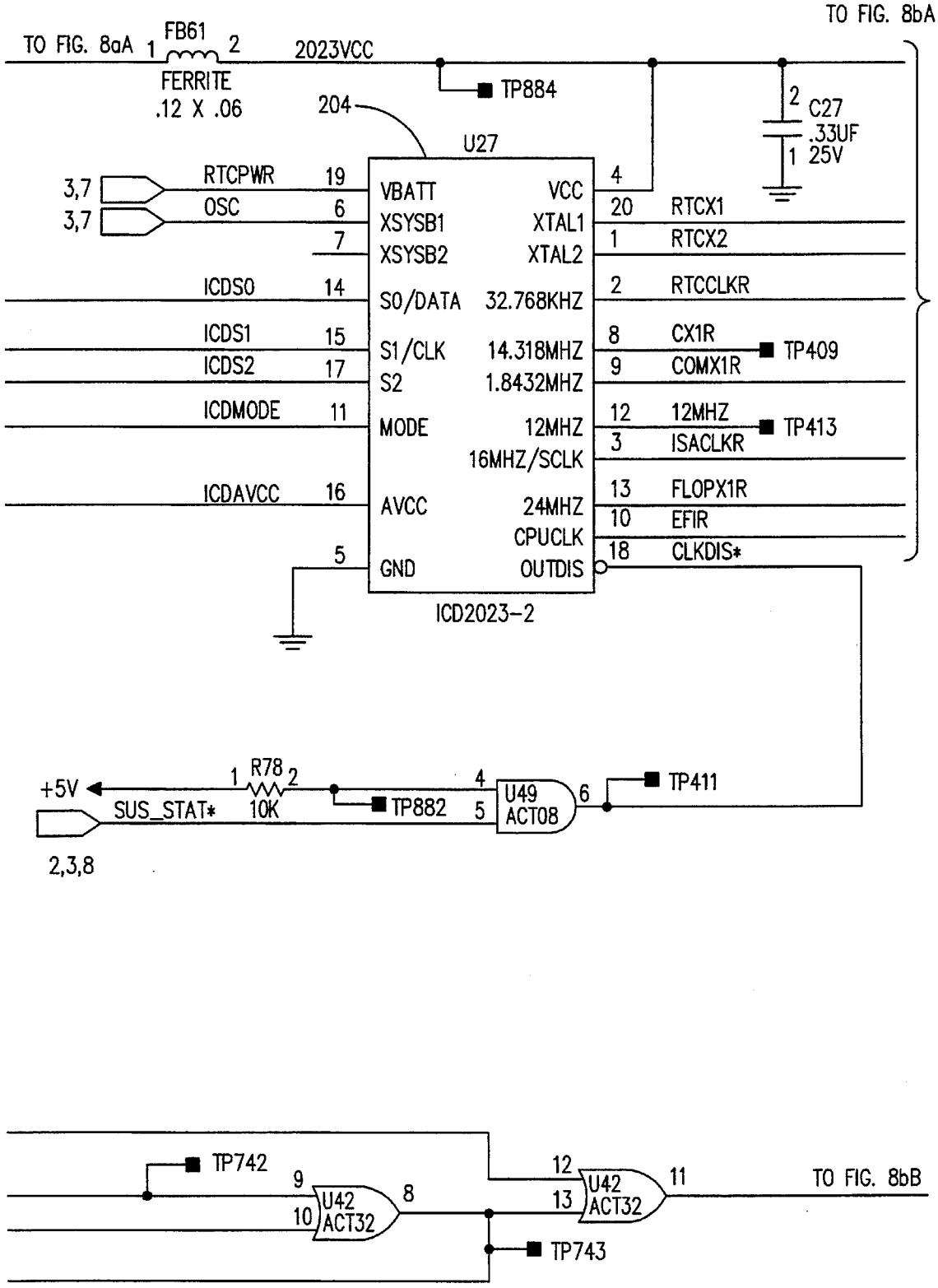

TO FIG. 8aA
FB61
FERRITE
.12 X .06
2023VCC
TP884
TO FIG. 8bA
C27
.33UF
25V
204
U27
3,7
RTCPWR
OSC
ICDS0
ICDS1
ICDS2
ICDMODE
ICDAVCC
VBATT
XSYSB1
XSYSB2
S0/DATA
S1/CLK
S2
MODE
AVCC
GND
VCC
XTAL1
XTAL2
32.768KHZ
14.318MHZ
1.8432MHZ
12MHZ
16MHZ/SCLK
24MHZ
CPUCLK
OUTDIS
RTCX1
RTCX2
RTCCLKR
CX1R
COMX1R
12MHZ
ISACLKR
FLOPX1R
EFIR
CLKDIS*
TP409
TP413
ICD2023-2
+5V
R78
10K
SUS_STAT*
TP882
U49
ACT08
TP411
2,3,8
TP742
U42
ACT32
TP743
TO FIG. 8bB

FIG. 8aB

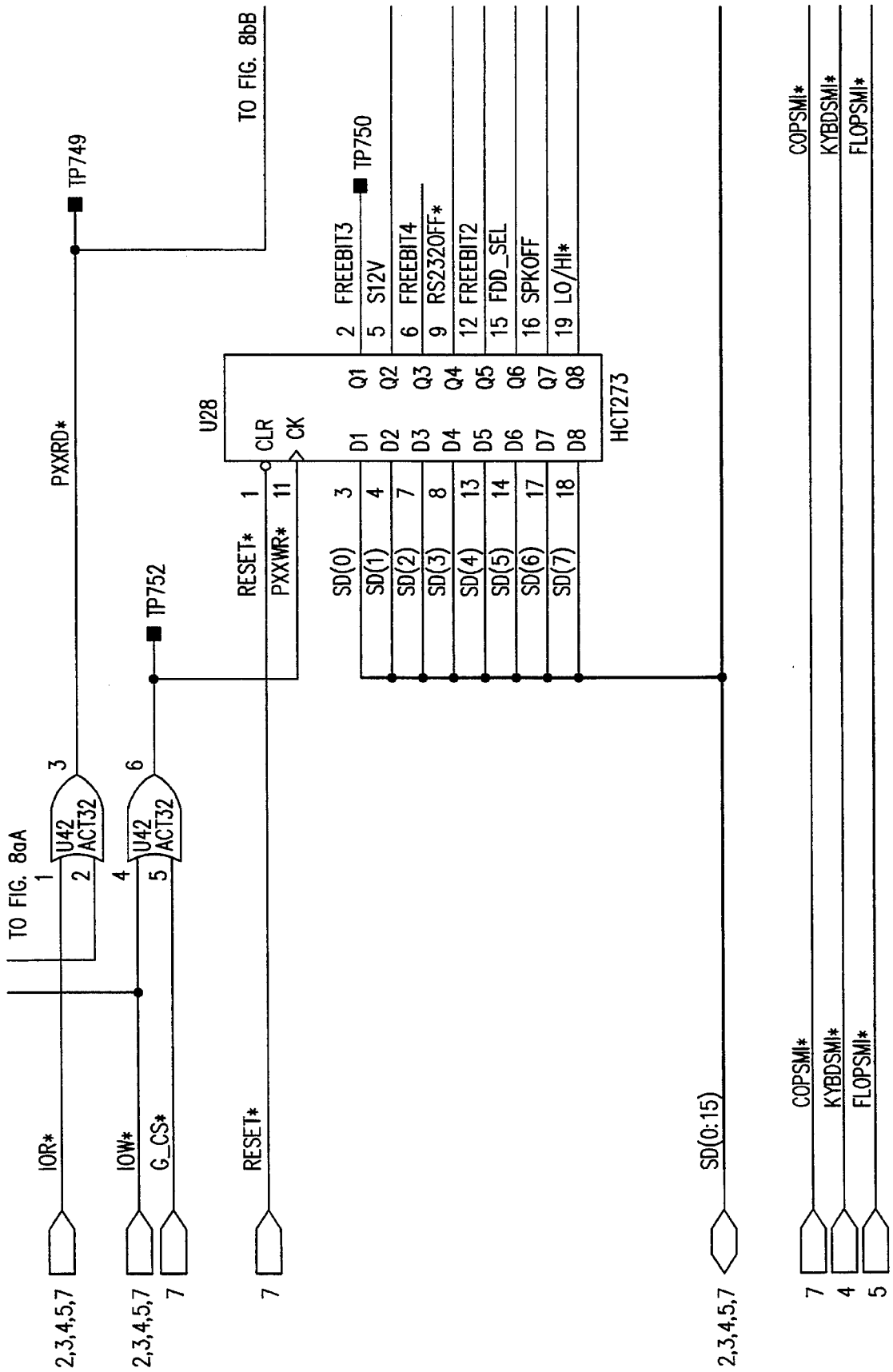

TO FIG. 8bB
TP749
TP750
TP752
PXXRD*
U28
HCT273
CLR
CK
RESET*
PXXWR*
SD(0)
SD(1)
SD(2)
SD(3)
SD(4)
SD(5)
SD(6)
SD(7)
D1
D2
D3
D4
D5
D6
D7
D8
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
FREEBIT3
S12V
FREEBIT4
RS232OFF*
FREEBIT2
FDD_SEL
SPKOFF
LO/HI*
COPSMI*
KYBDSMI*
FLOPSMI*
TO FIG. 8aA
U42 ACT32
IOR*
IOW*
G_CS*
RESET*
SD(0:15)
2,3,4,5,7
7
4
5

FIG. 8aC

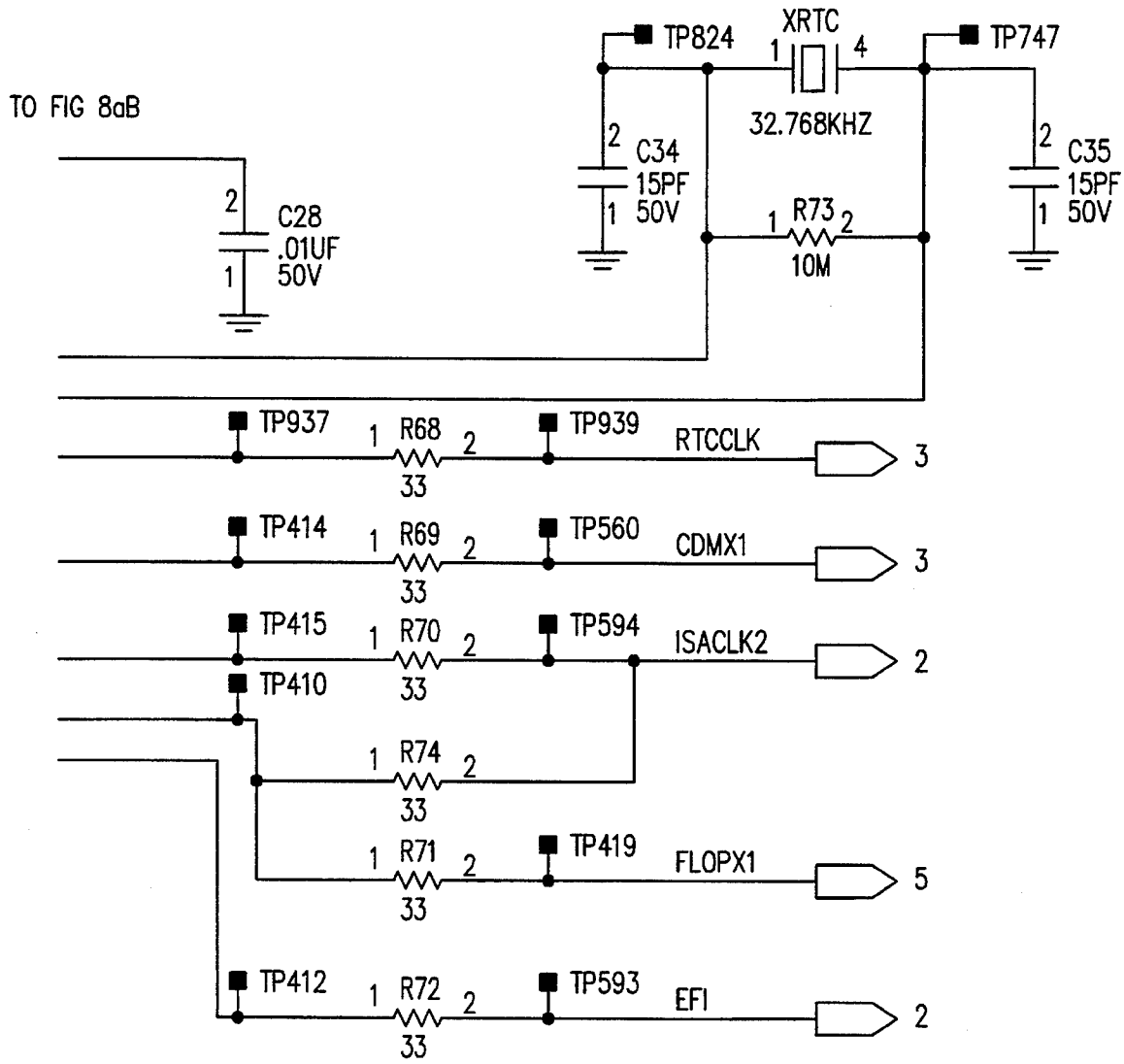


FIG. 8bA

TO FIG 8bB

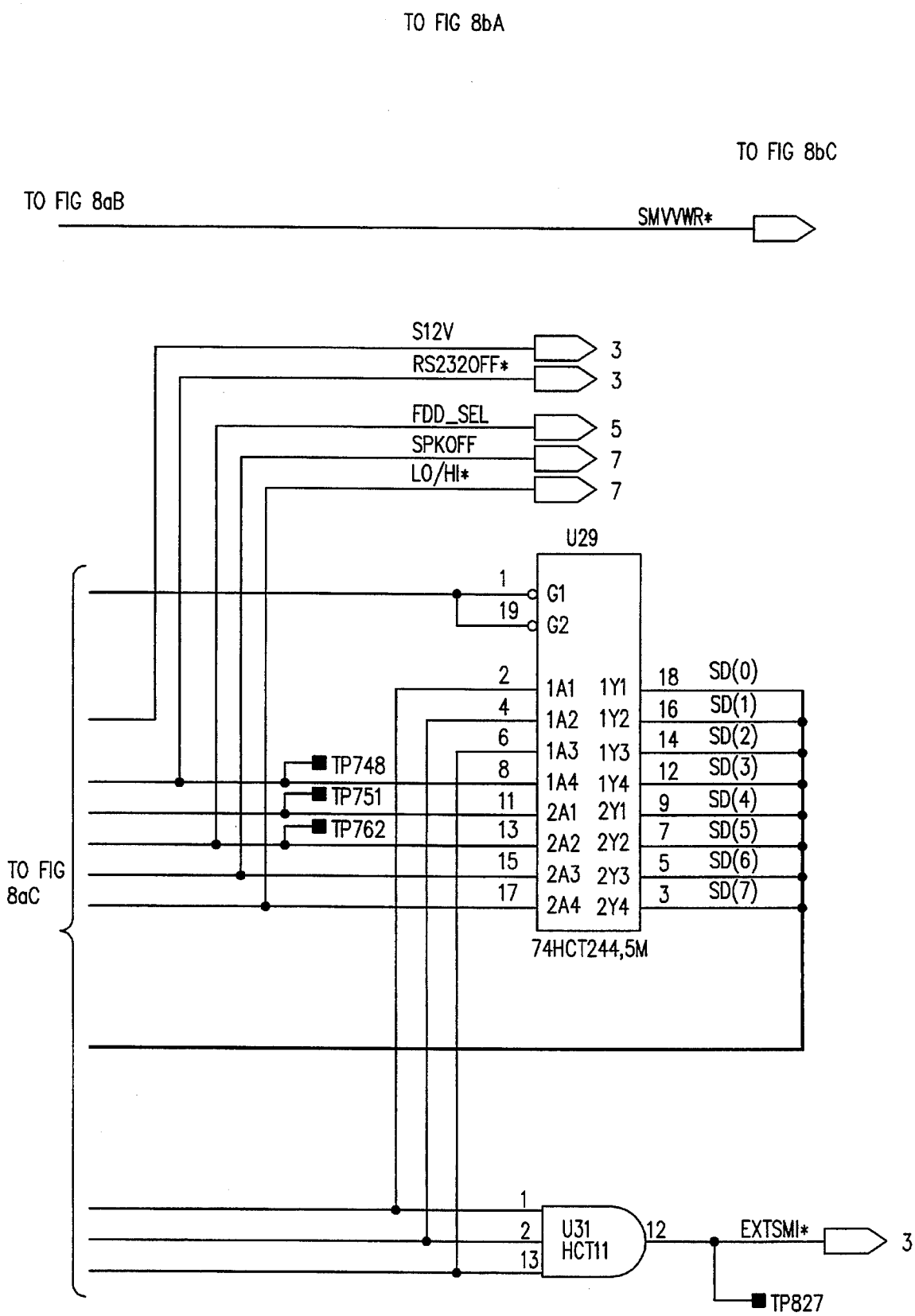

TO FIG 8bA
TO FIG 8bC
TO FIG 8aB
SMVWR*
S12V
RS232OFF*
FDD_SEL
SPKOFF
LO/HI*
U29
G1
G2
1A1 1Y1
1A2 1Y2
1A3 1Y3
1A4 1Y4
2A1 2Y1
2A2 2Y2
2A3 2Y3
2A4 2Y4
SD(0)
SD(1)
SD(2)
SD(3)
SD(4)
SD(5)
SD(6)
SD(7)
TP748
TP751
TP762
74HCT244,5M
TO FIG 8aC
U31 HCT11
EXTSMI*
TP827

FIG. 8bB

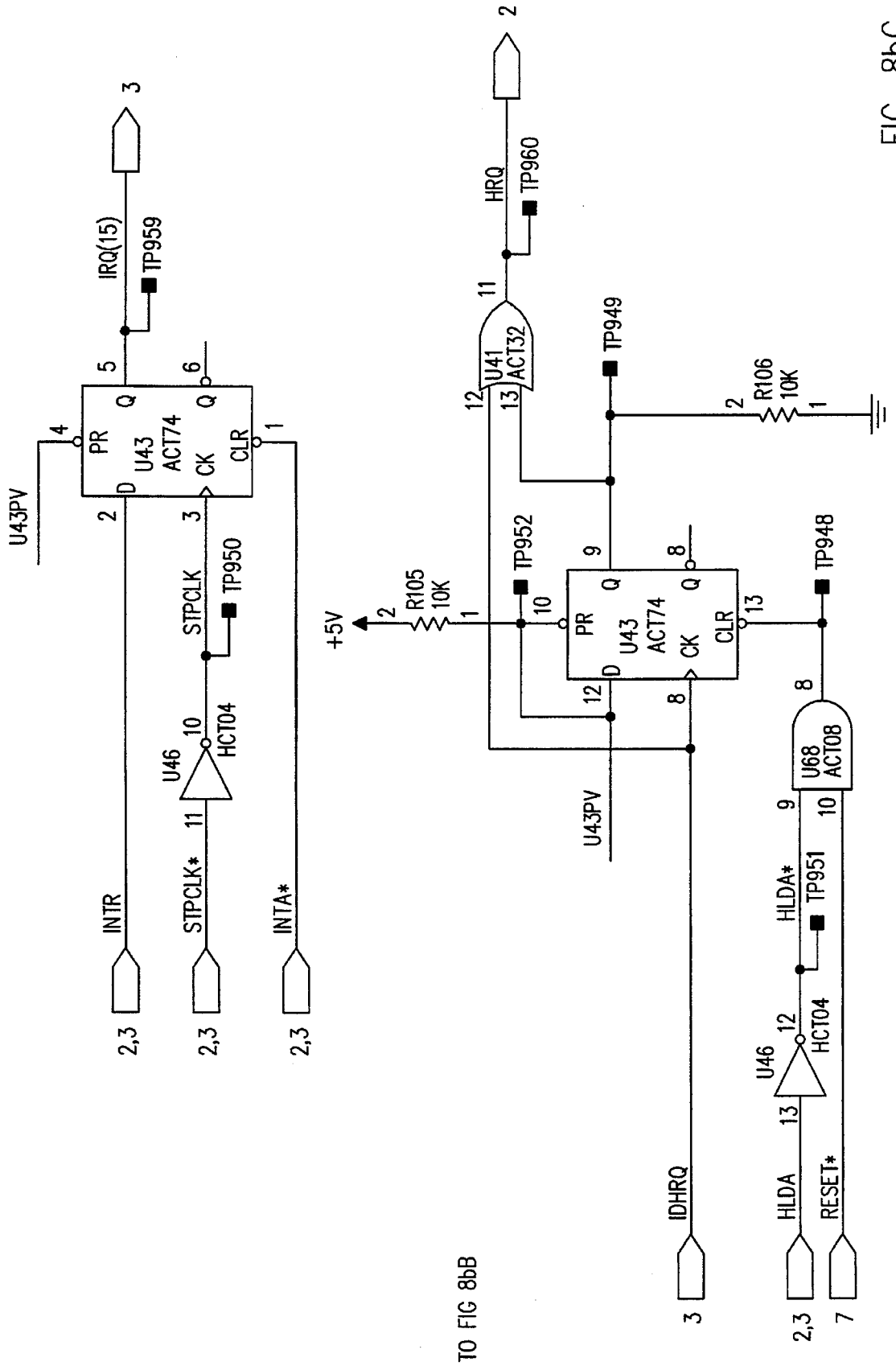
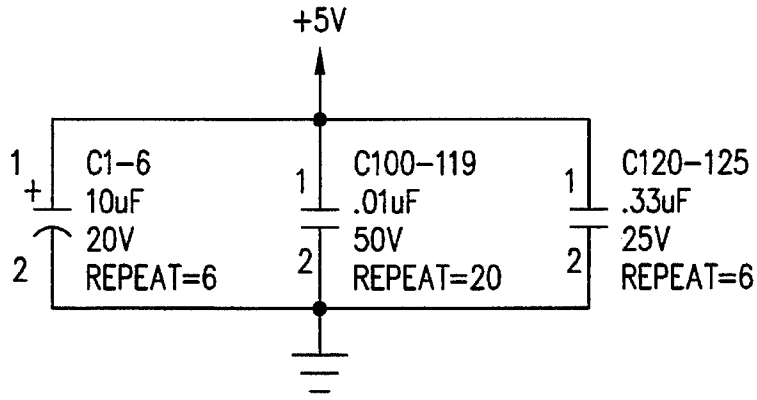
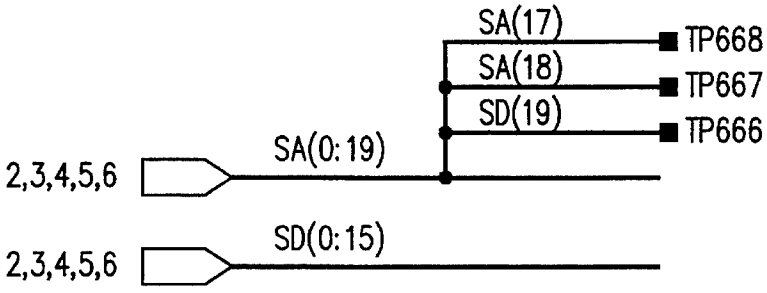

U43PV
PR
U43
ACT74
CK
CLR
D
Q
IRQ(15)
TP959
STPCLK
TP950
U46
HCT04
STPCLK*
INTR
INTA*
2,3
+5V
R105
10K
TP952
U41
ACT32
HRQ
TP960
TP949
R106
10K
TP948
U68
ACT08
HLDA*
TP951
HLDA
RESET*
IDHRQ
TO FIG 8bB

FIG. 8bC





TO FIGURE 9aB

FIG. 9aA

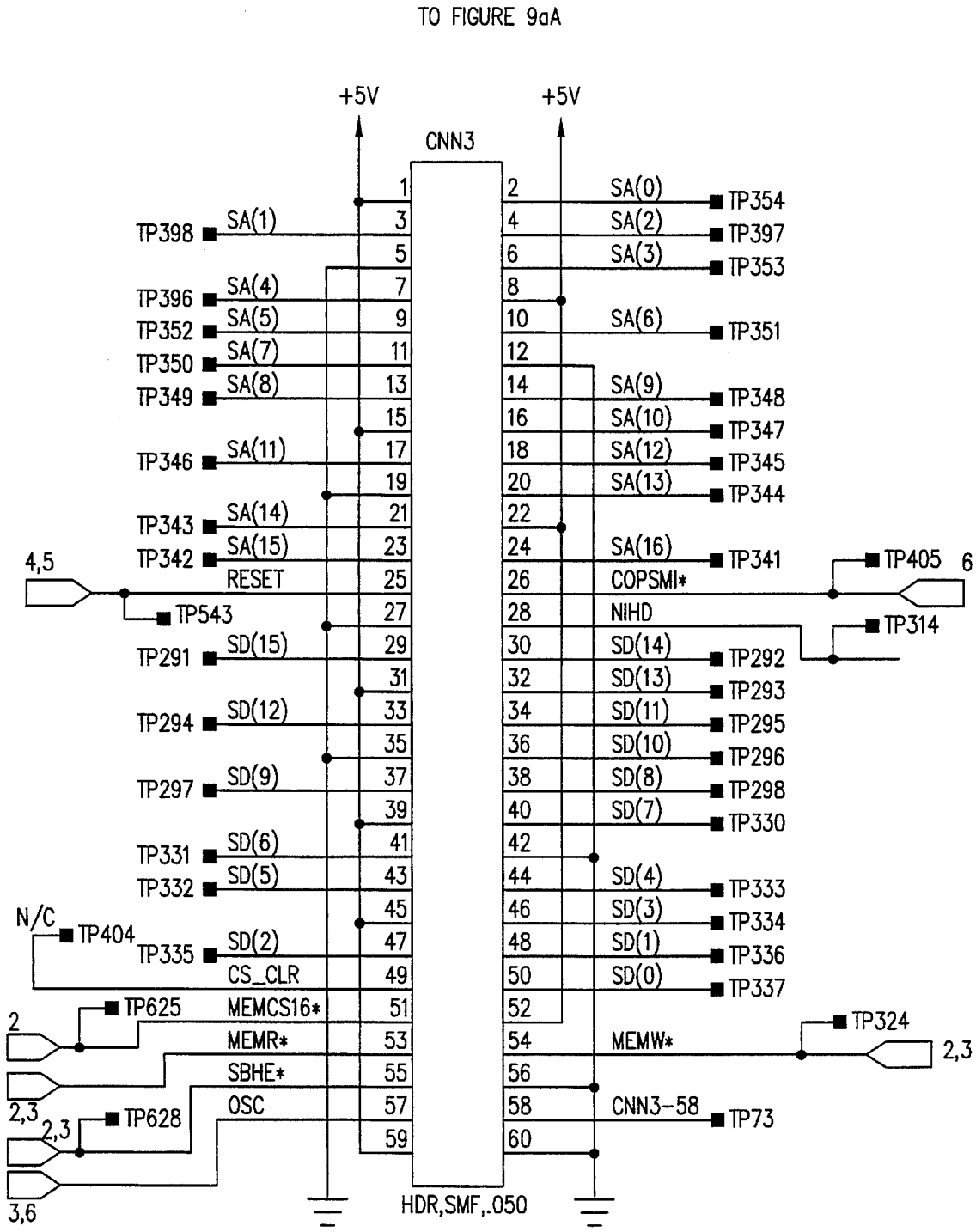

TO FIGURE 9aA
+5V
+5V
CNN3
TP398 SA(1)
TP396 SA(4)
TP352 SA(5)
TP350 SA(7)
TP349 SA(8)
TP346 SA(11)
TP343 SA(14)
TP342 SA(15)
4,5
RESET
TP543
TP291 SD(15)
TP294 SD(12)
TP297 SD(9)
TP331 SD(6)
TP332 SD(5)
N/C
TP404
TP335 SD(2)
CS_CLR
TP625
MEMCS16*
2
MEMR*
SBHE*
2,3
OSC
2,3
TP628
3,6
SA(0) TP354
SA(2) TP397
SA(3) TP353
SA(6) TP351
SA(9) TP348
SA(10) TP347
SA(12) TP345
SA(13) TP344
SA(16) TP341
TP405 6
COPSMI*
NIHD
TP314
SD(14) TP292
SD(13) TP293
SD(11) TP295
SD(10) TP296
SD(8) TP298
SD(7) TP330
SD(4) TP333
SD(3) TP334
SD(1) TP336
SD(0) TP337
TP324
MEMW*
2,3
CNN3-58
TP73
HDR,SMF,.050

FIG. 9aB

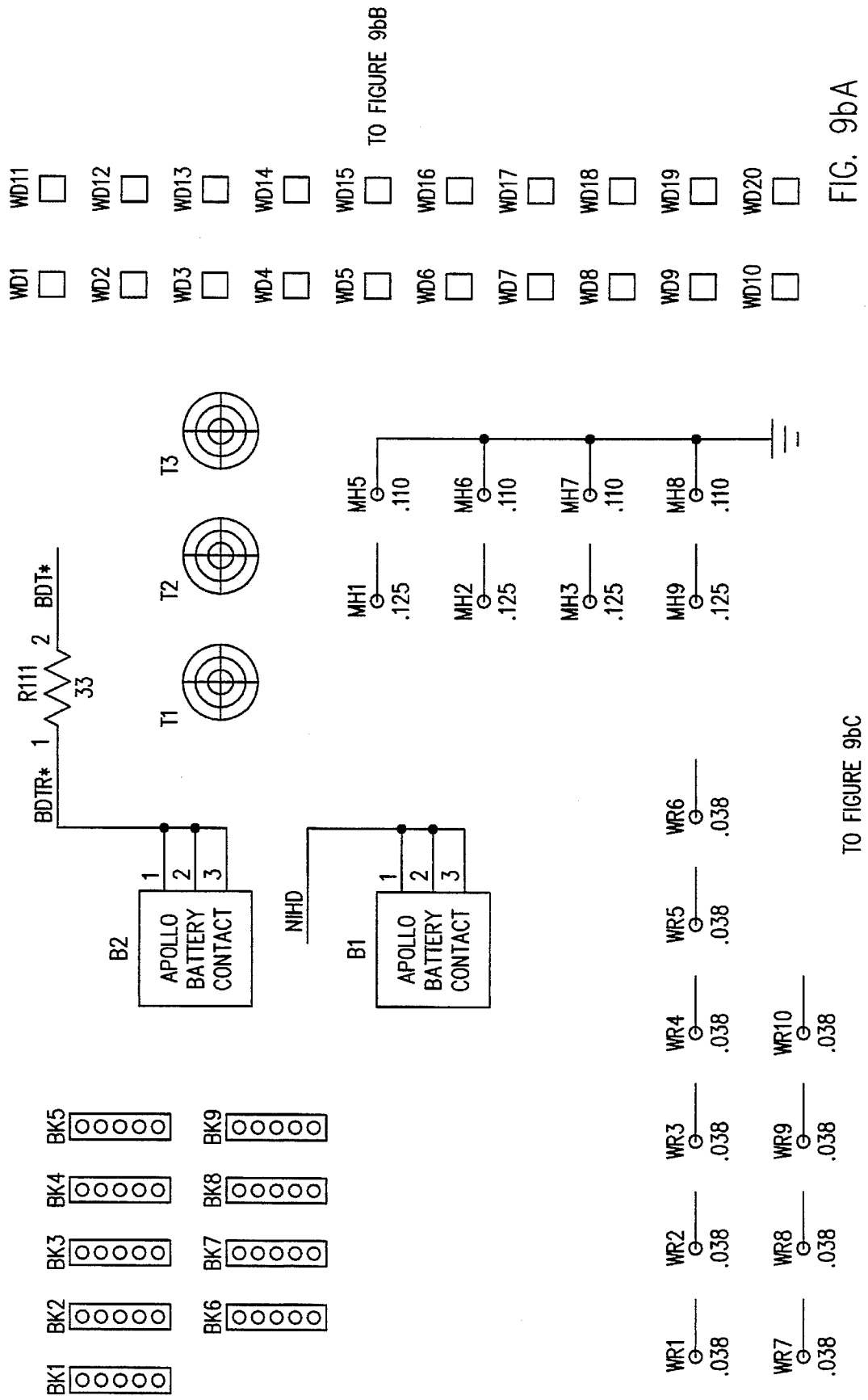

WD1
WD2
WD3
WD4
WD5
WD6
WD7
WD8
WD9
WD10
WD11
WD12
WD13
WD14
WD15
WD16
WD17
WD18
WD19
WD20
TO FIGURE 9bB
BDTR*
R111
33
BDT*
T1
T2
T3
B2
APOLLO BATTERY CONTACT
NIHD
B1
APOLLO BATTERY CONTACT
MH1 .125
MH2 .125
MH3 .125
MH9 .125
MH5 .110
MH6 .110
MH7 .110
MH8 .110
BK1
BK2
BK3
BK4
BK5
BK6
BK7
BK8
BK9
WR1 .038
WR2 .038
WR3 .038
WR4 .038
WR5 .038
WR6 .038
WR7 .038
WR8 .038
WR9 .038
WR10 .038
TO FIGURE 9bC

FIG. 9bA

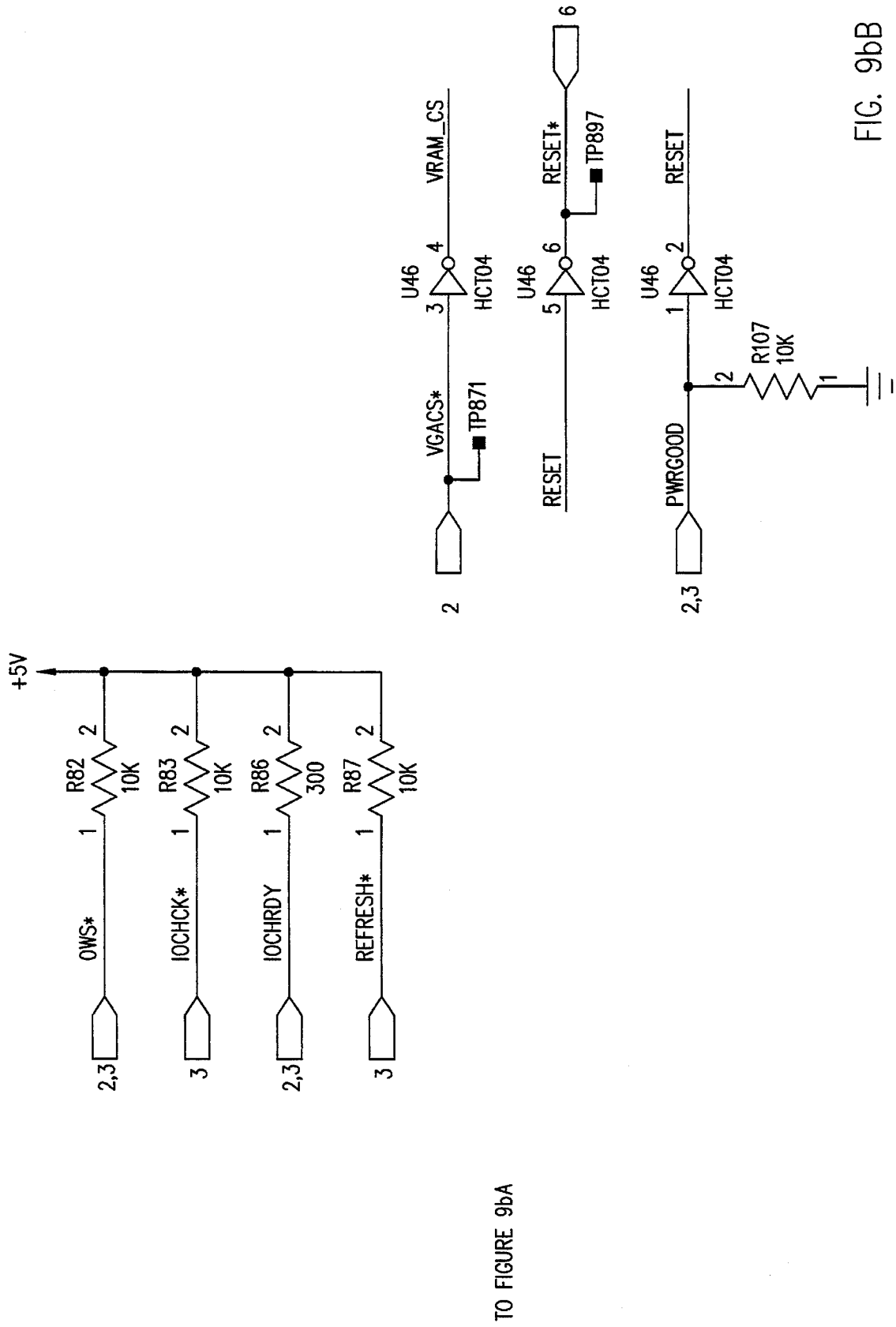

+5V
R82
10K
R83
10K
R86
300
R87
10K
OWS*
IOCHCK*
IOCHRDY
REFRESH*
2,3
3
2,3
3
VGACS*
TP871
U46
HCT04
VRAM_CS
RESET
RESET*
TP897
PWRGOOD
R107
10K
2
6
2,3
TO FIGURE 9bA

FIG. 9bB

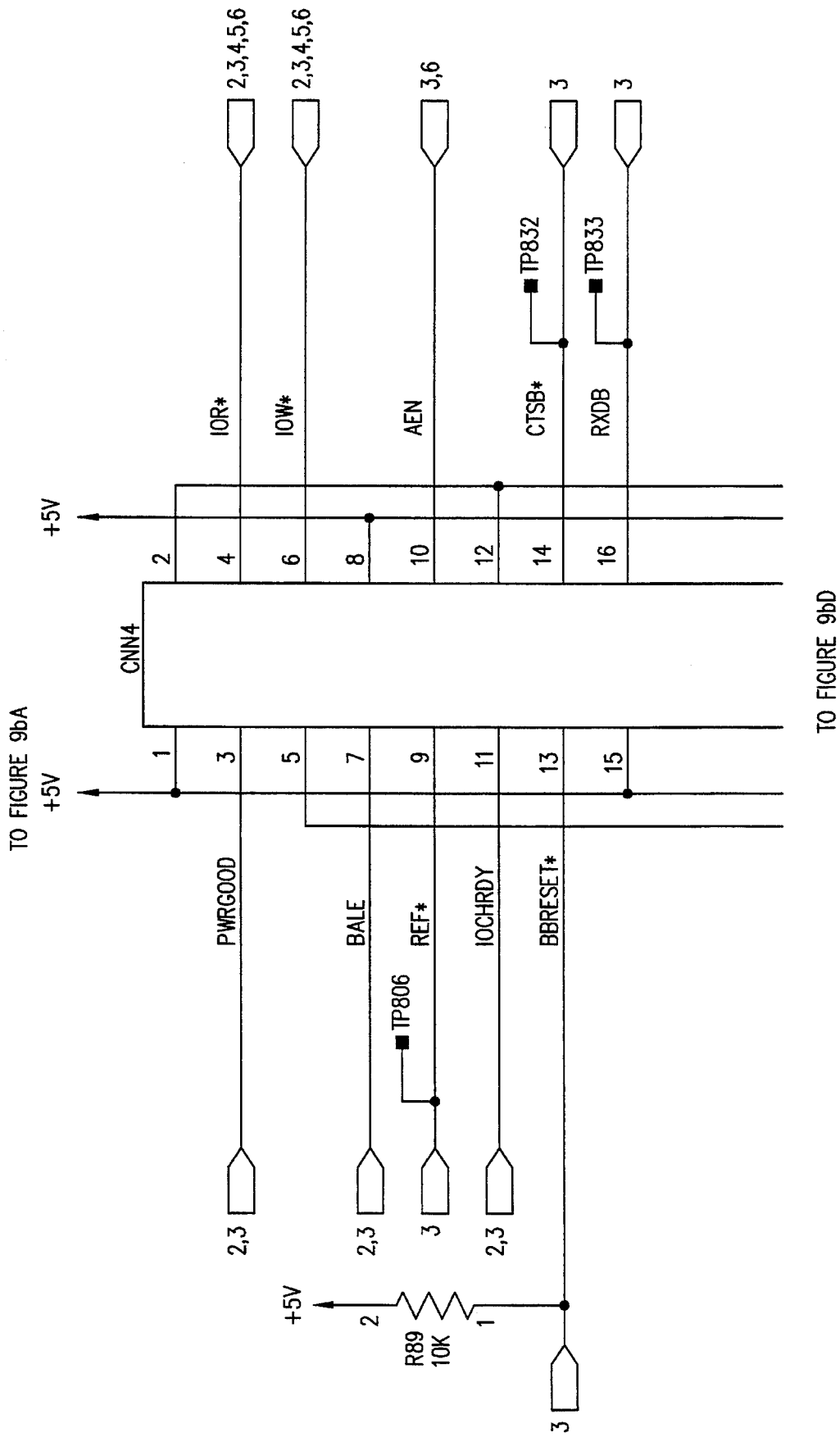

TO FIGURE 9bA
+5V
+5V
CNN4
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
PWRGOOD
IOR*
IOW*
BALE
REF*
AEN
IOCHRDY
CTSB*
BBRESET*
RXDB
TP806
TP832
TP833
2,3
2,3,4,5,6
2,3,4,5,6
3
3,6
3
3
3
+5V
R89
10K
TO FIGURE 9bD

FIG. 9bC

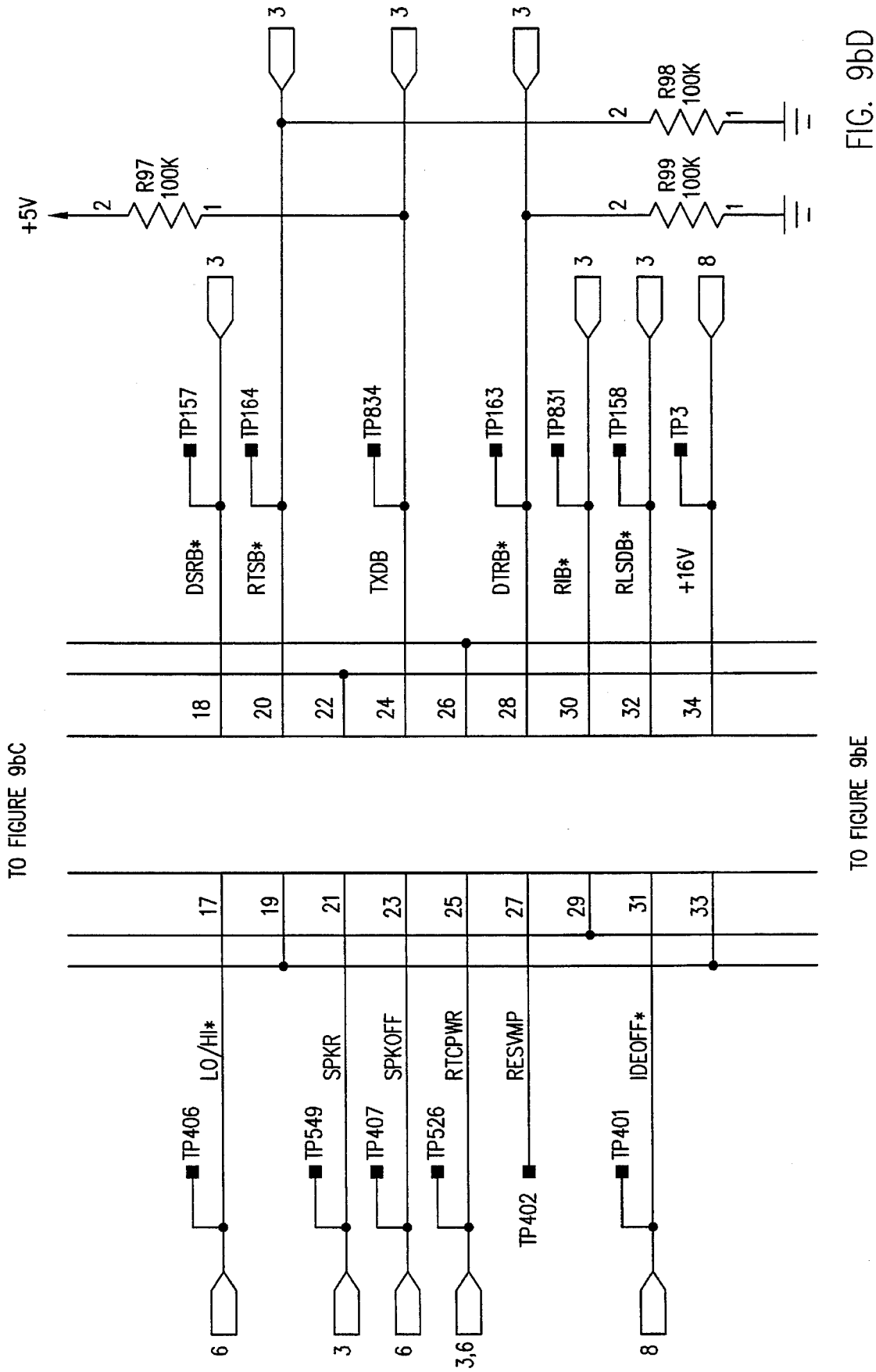

TO FIGURE 9bC
TO FIGURE 9bE
+5V
R97 100K
R98 100K
R99 100K
TP157
TP164
TP834
TP163
TP831
TP158
TP3
DSRB*
RTSB*
TXDB
DTRB*
RIB*
RLSDB*
+16V
TP406
TP549
TP407
TP526
TP402
TP401
LO/HI*
SPKR
SPKOFF
RTCPWR
RESVMP
IDEOFF*
18
20
22
24
26
28
30
32
34
17
19
21
23
25
27
29
31
33
6
3
6
3,6
8

FIG. 9bD

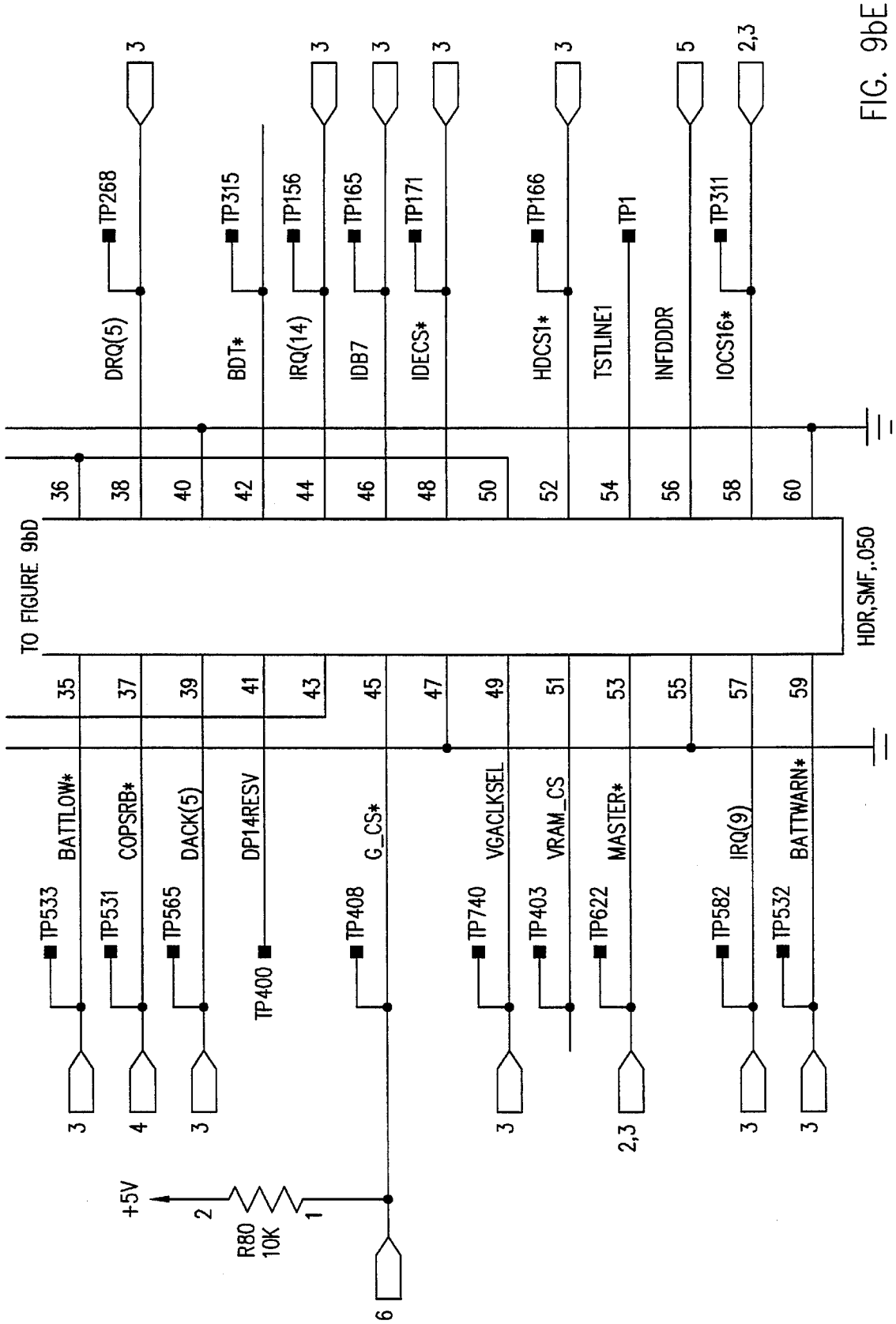

TO FIGURE 9bD
HDR,SMF,.050
DRQ(5)
BDT*
IRQ(14)
IDB7
IDECS*
HDCS1*
TSTLINE1
INFDDDR
IOCS16*
TP268
TP315
TP156
TP165
TP171
TP166
TP1
TP311
BATTLOW*
COPSRB*
DACK(5)
DP14RESV
G_CS*
VGACLKSEL
VRAM_CS
MASTER*
IRQ(9)
BATTWARN*
TP533
TP531
TP565
TP400
TP408
TP740
TP403
TP622
TP582
TP532
+5V
R80
10K

FIG. 9bE

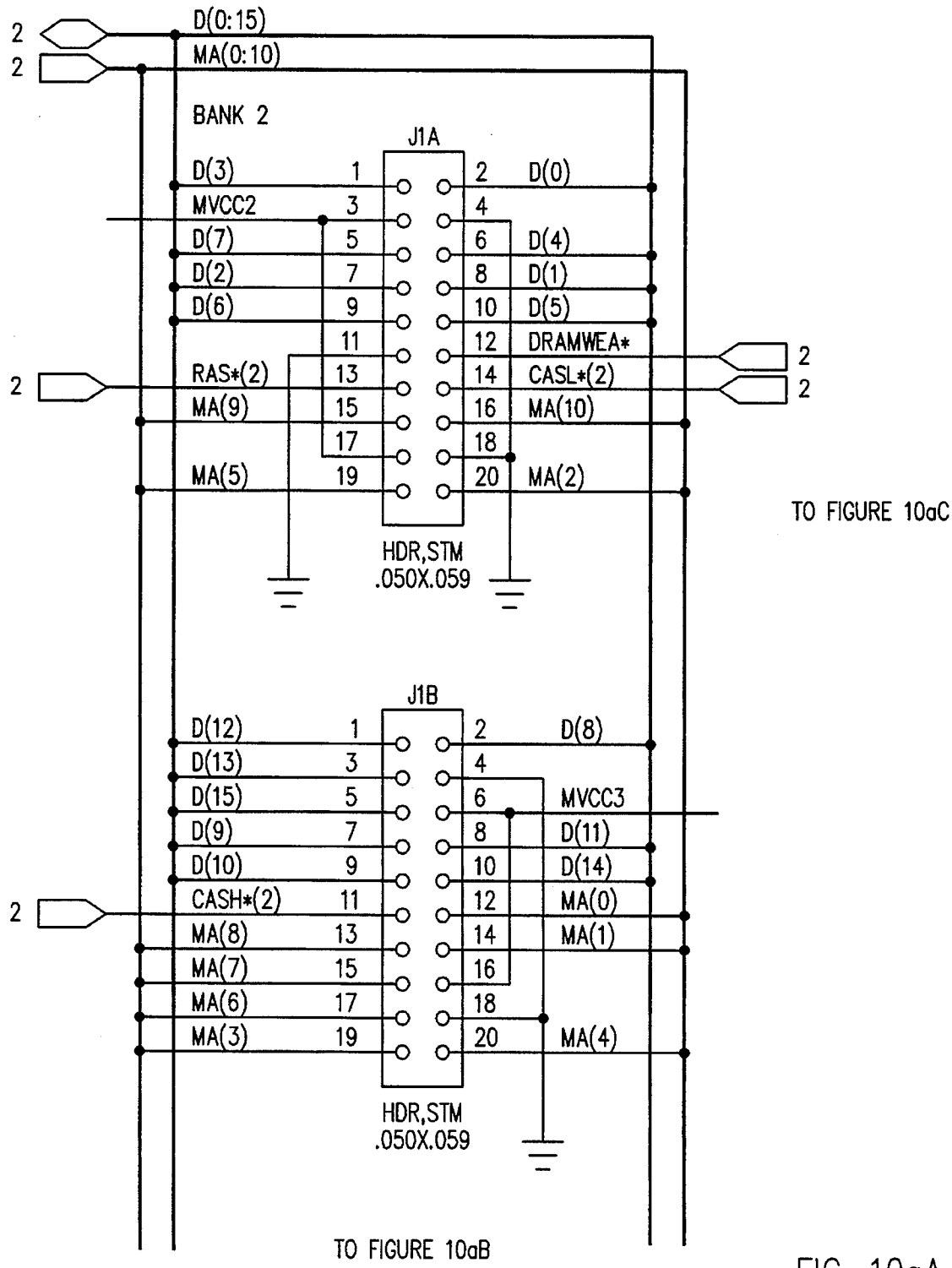
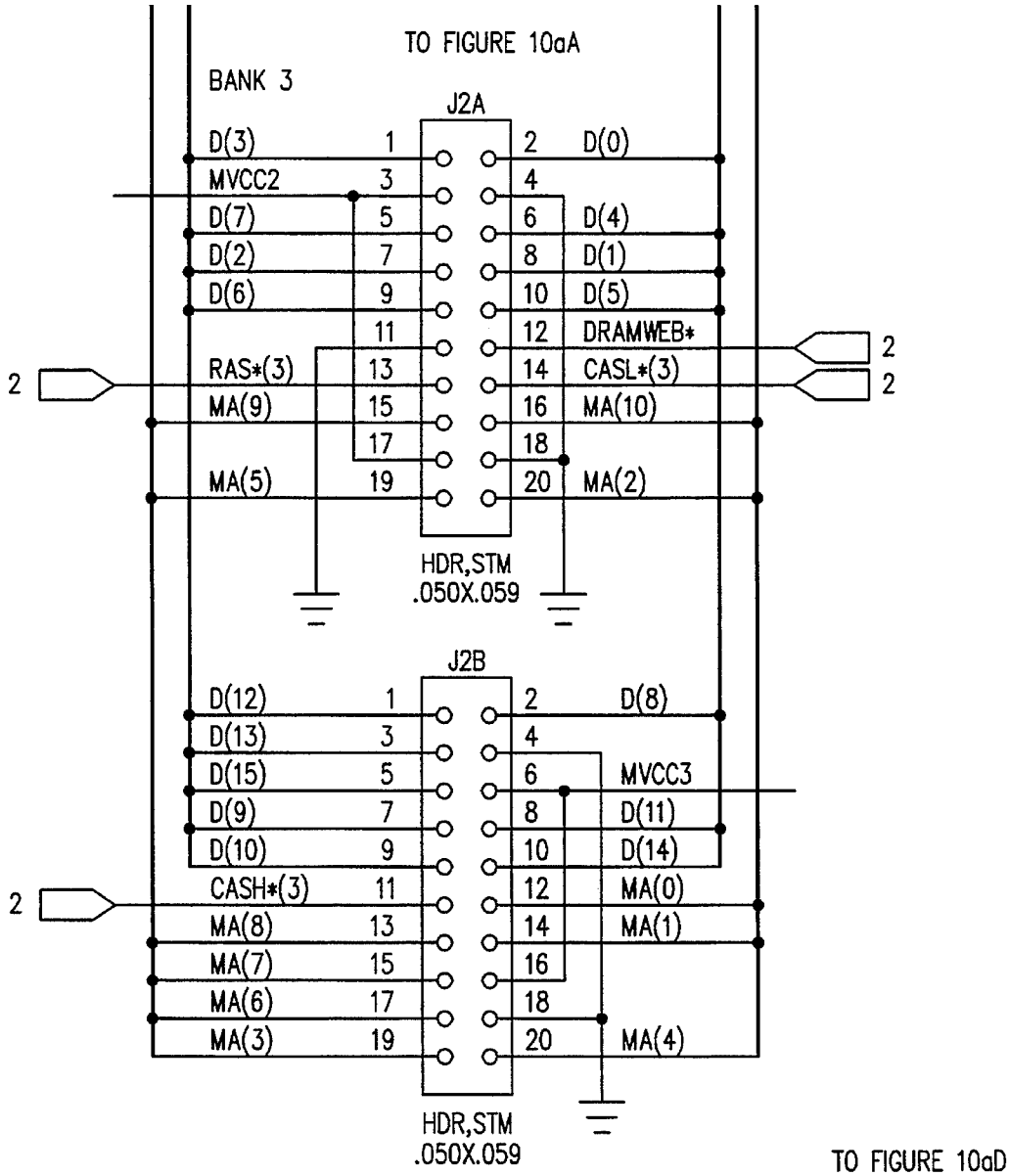

D(0:15)
MA(0:10)
BANK 2
J1A
D(3)
MVCC2
D(7)
D(2)
D(6)
RAS*(2)
MA(9)
MA(5)
D(0)
D(4)
D(1)
D(5)
DRAMWEA*
CASL*(2)
MA(10)
MA(2)
HDR,STM
.050X.059
TO FIGURE 10aC
J1B
D(12)
D(13)
D(15)
D(9)
D(10)
CASH*(2)
MA(8)
MA(7)
MA(6)
MA(3)
D(8)
MVCC3
D(11)
D(14)
MA(0)
MA(1)
MA(4)
HDR,STM
.050X.059
TO FIGURE 10aB

FIG. 10aA


TO FIGURE 10aA
BANK 3
J2A
D(3) 1
MVCC2 3
D(7) 5
D(2) 7
D(6) 9
11
RAS*(3) 13
MA(9) 15
17
MA(5) 19
2 D(0)
4
6 D(4)
8 D(1)
10 D(5)
12 DRAMWEB*
14 CASL*(3)
16 MA(10)
18
20 MA(2)
2
2
2
HDR,STM
.050X.059
J2B
D(12) 1
D(13) 3
D(15) 5
D(9) 7
D(10) 9
CASH*(3) 11
MA(8) 13
MA(7) 15
MA(6) 17
MA(3) 19
2 D(8)
4
6 MVCC3
8 D(11)
10 D(14)
12 MA(0)
14 MA(1)
16
18
20 MA(4)
2
HDR,STM
.050X.059
TO FIGURE 10aD

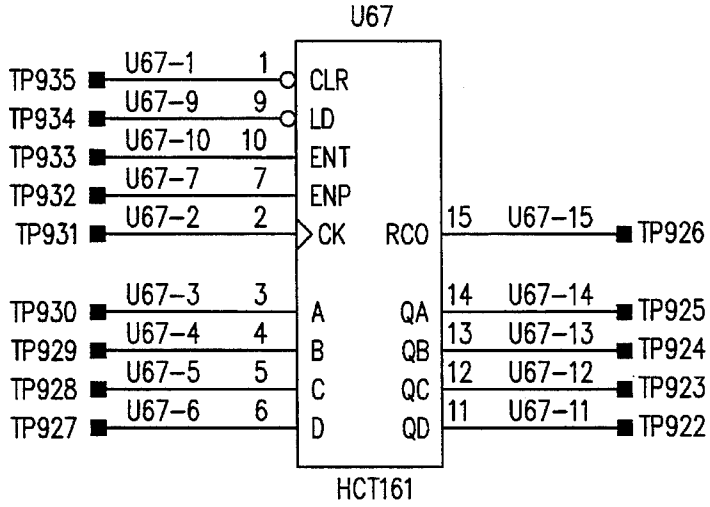

U67
TP935 U67-1 1 CLR
TP934 U67-9 9 LD
TP933 U67-10 10 ENT
TP932 U67-7 7 ENP
TP931 U67-2 2 CK
RCO 15 U67-15 TP926
TP930 U67-3 3 A
TP929 U67-4 4 B
TP928 U67-5 5 C
TP927 U67-6 6 D
QA 14 U67-14 TP925
QB 13 U67-13 TP924
QC 12 U67-12 TP923
QD 11 U67-11 TP922
HCT161

FIG. 10aB

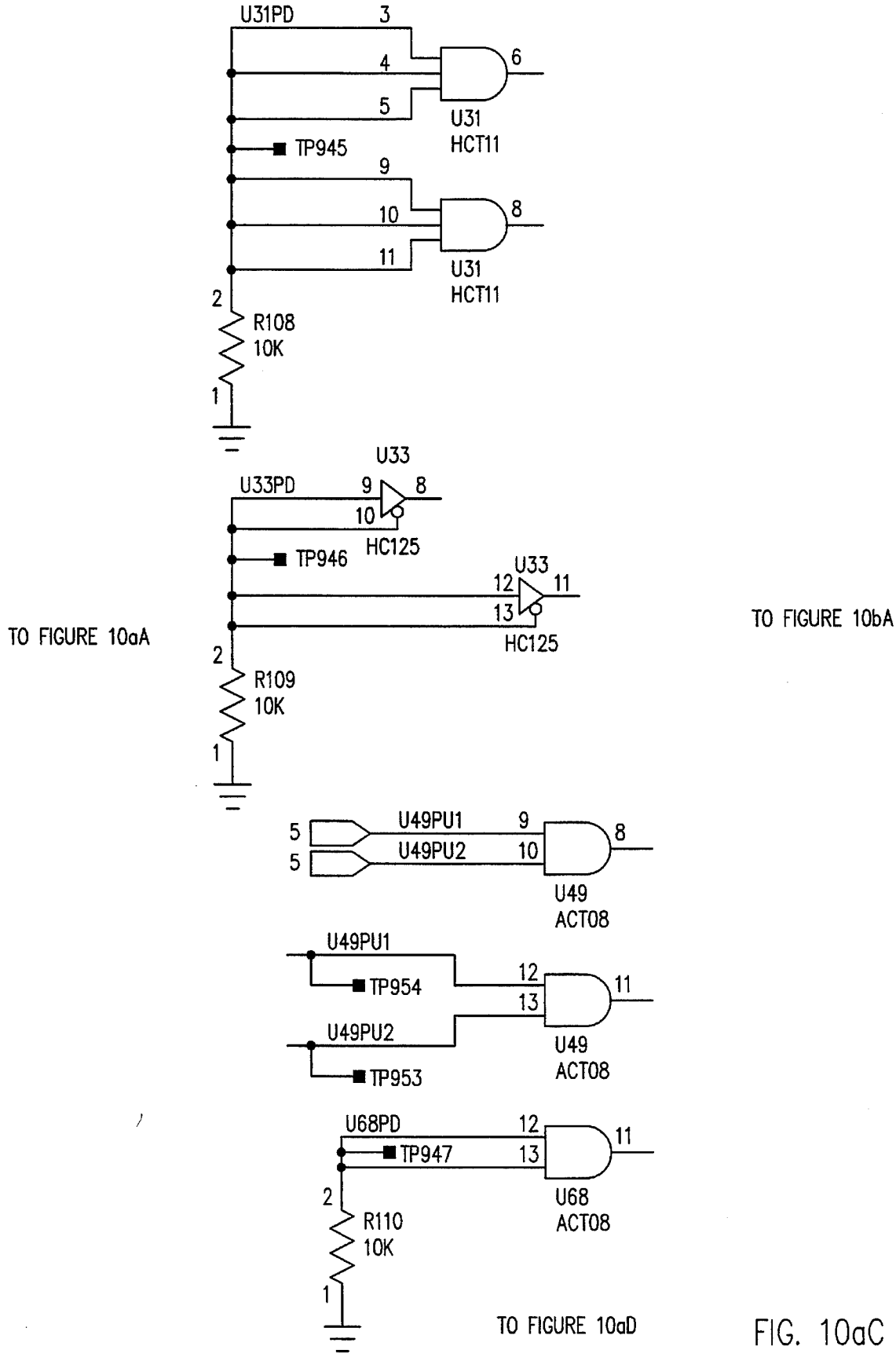
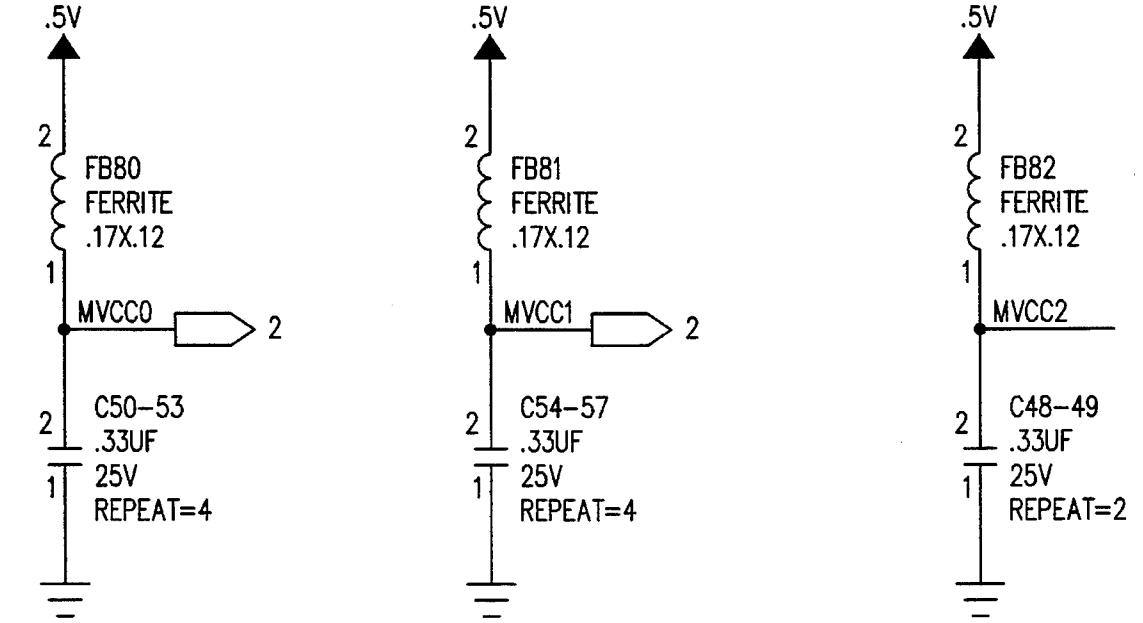

U31PD
3
4
5
6
U31
HCT11
TP945
9
10
8
11
U31
HCT11
2
R108
10K
1
U33
U33PD
9
8
10
HC125
TP946
U33
12
11
13
HC125
TO FIGURE 10aA
TO FIGURE 10bA
2
R109
10K
1
5
U49PU1
9
8
5
U49PU2
10
U49
ACT08
U49PU1
TP954
12
11
13
U49PU2
TP953
U49
ACT08
U68PD
12
11
TP947
13
2
R110
10K
U68
ACT08
1
TO FIGURE 10aD

FIG. 10aC

TO FIGURE 10aC



TO FIGURE 10aB

TO FIGURE 10bC

FIG. 10aD

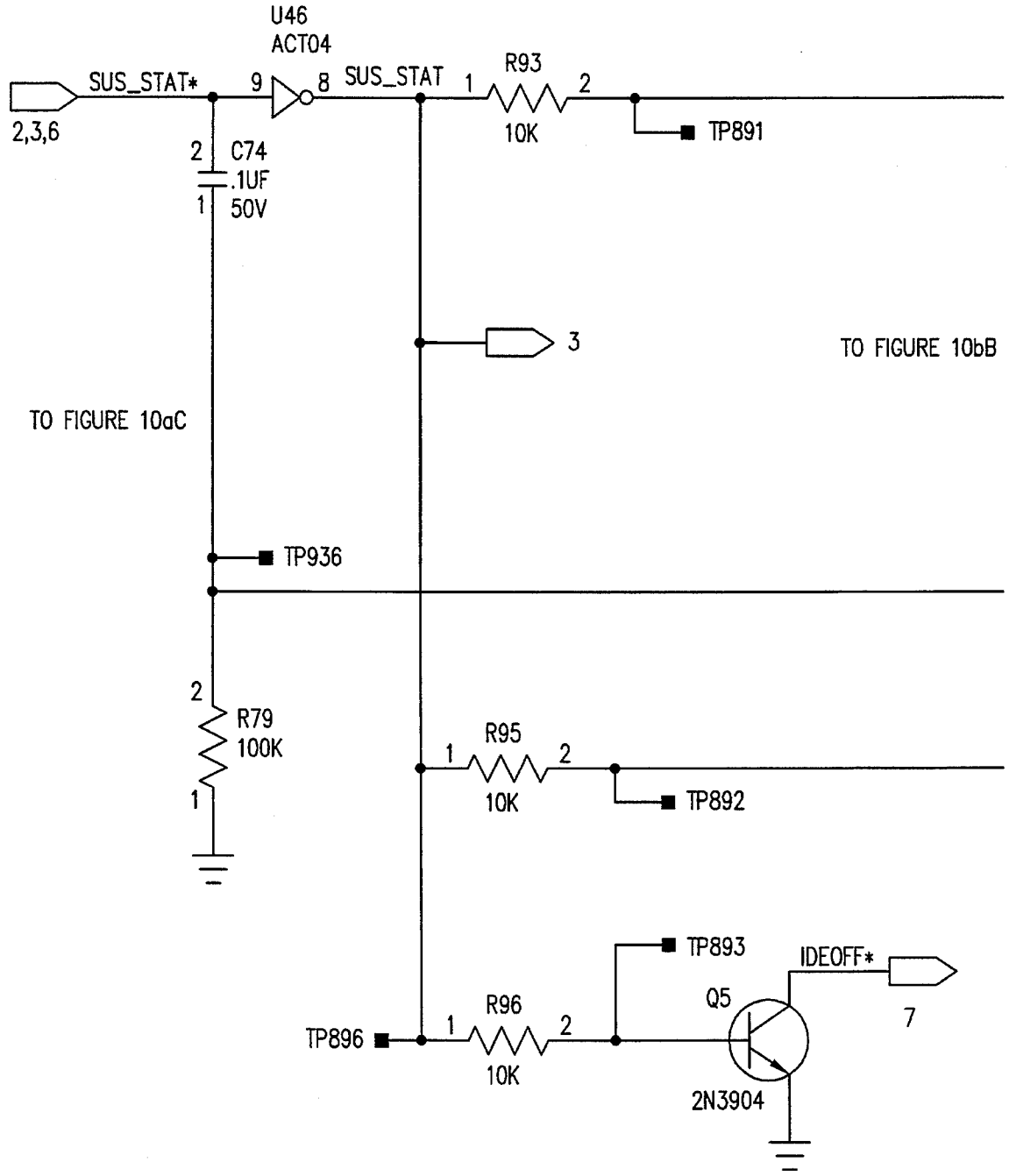

U46
ACT04
SUS_STAT*
9
8
SUS_STAT
1
R93
2
10K
TP891
2,3,6
2
C74
.1UF
1
50V
3
TO FIGURE 10bB
TO FIGURE 10aC
TP936
2
R79
100K
1
R95
1
2
10K
TP892
TP893
IDEOFF*
Q5
7
R96
TP896
1
2
10K
2N3904
TO FIGURE 10bC

FIG. 10bA

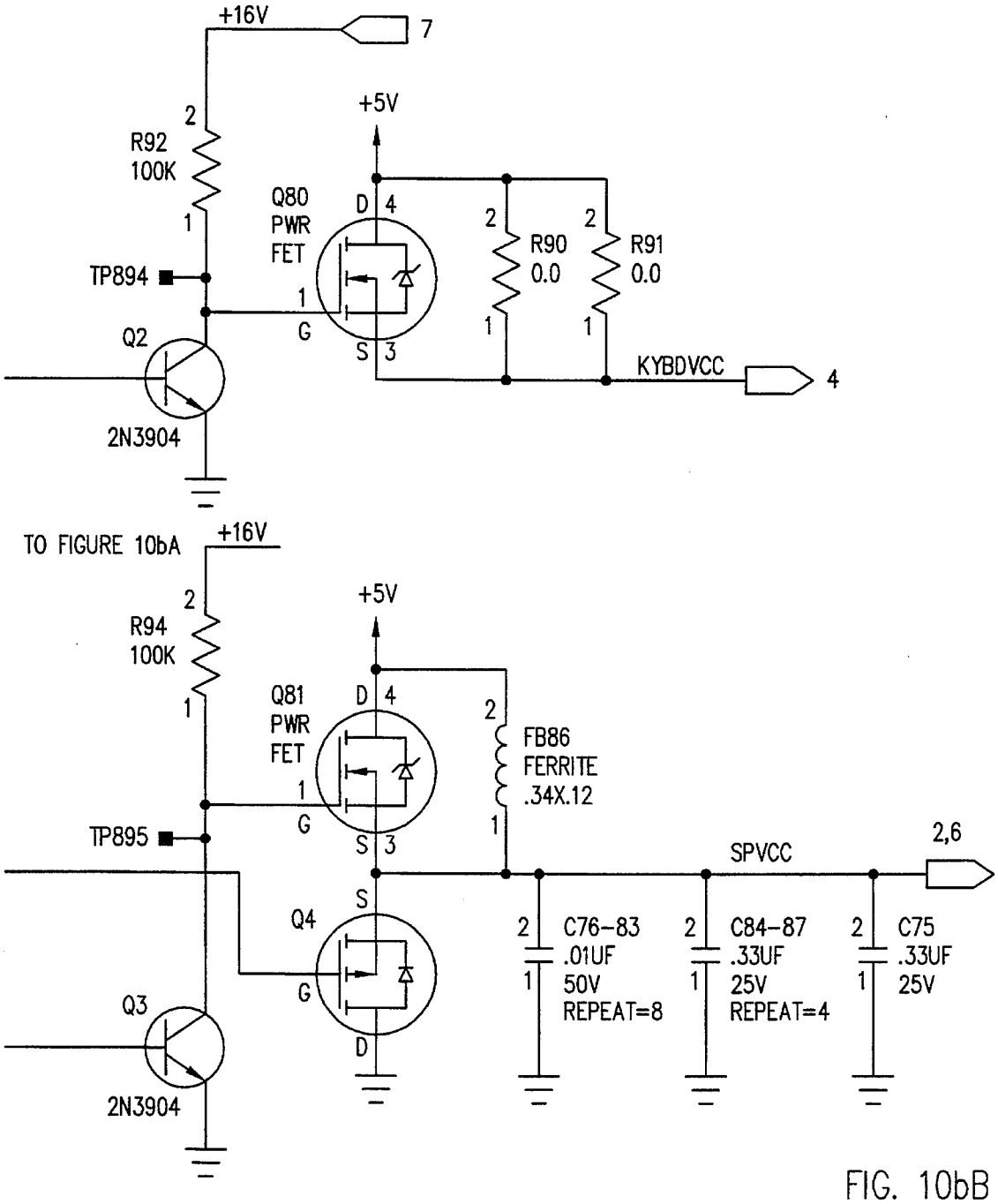
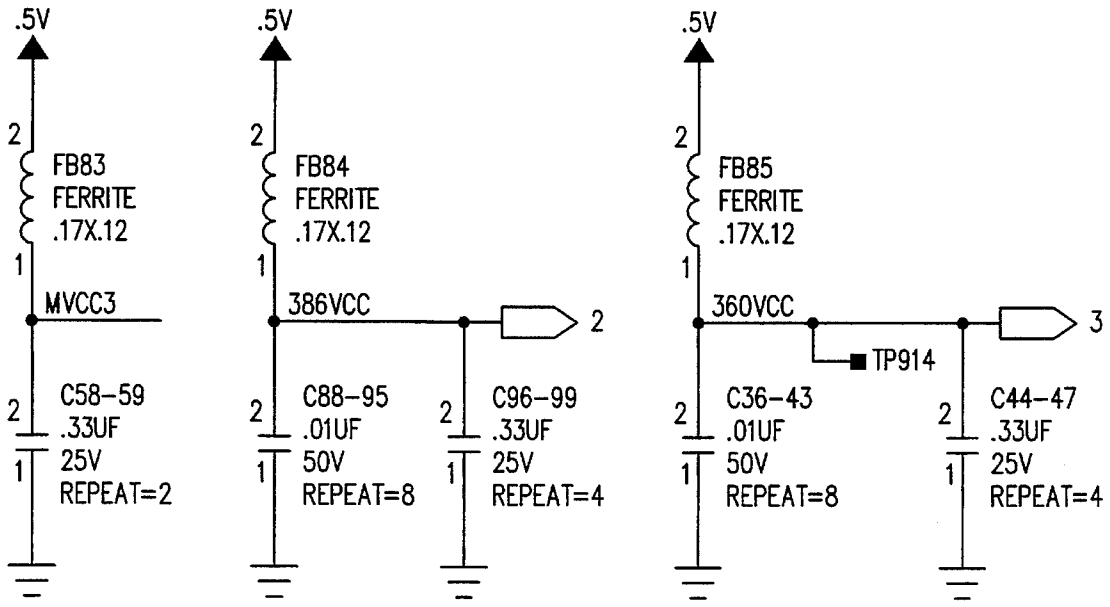

+16V
7
+5V
R92
100K
Q80
PWR
FET
D
4
R90
0.0
R91
0.0
TP894
G
S
3
Q2
2N3904
KYBDVCC
4
TO FIGURE 10bA
+16V
R94
100K
+5V
Q81
PWR
FET
FB86
FERRITE
.34X.12
TP895
SPVCC
2,6
Q4
C76-83
.01UF
50V
REPEAT=8
C84-87
.33UF
25V
REPEAT=4
C75
.33UF
25V
Q3
2N3904

FIG. 10bB

TO FIGURE 10bA



TO FIGURE 10aD

FIG. 10bC

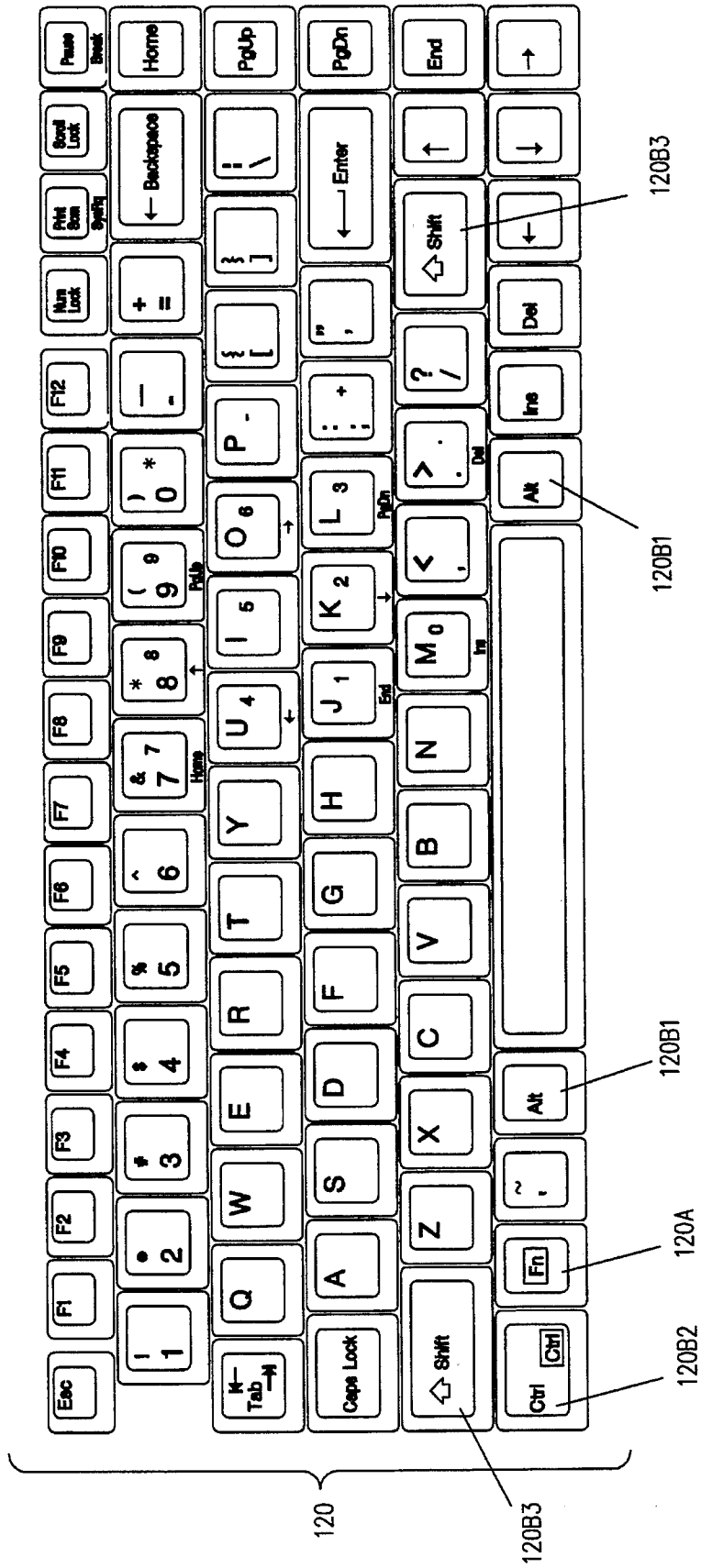


FIG. 11

COMPUTER WITH POINTING DEVICE MAPPED INTO KEYBOARD

This is a continuation of application Ser. No. 07/813,379 filed on Dec. 23, 1991 now abandoned.

PARTIAL WAIVER OF COPYRIGHT

All of the material in this patent application is subject to copyright protection under the copyright laws of the United States and of other countries. As of the first effective filing date of the present application, this material is protected as unpublished material.

Portions of the material in the specification and drawings of this patent application are also subject to protection under the maskwork registration laws of the United States and of other countries.

However, permission to copy this material is hereby granted to the extent that the owner of the copyright and maskwork rights has no objection to the facsimile reproduction by anyone of the patent document or patent disclosure, as it appears in the United States Patent and Trademark Office patent file or records, but otherwise reserves all copyright and maskwork rights whatsoever.

CROSS-REFERENCE TO OTHER APPLICATIONS

The following applications of common assignee contain at least some drawings in common with the present application, and are believed to have effective filing dates identical with that of the present application, and are all hereby incorporated by reference:

Ser. No. 07/813,380, filed Dec. 23, 1991, entitled "Computer with Transparent Power-Saving Manipulation of CPU Clock" (DC-264) now U.S. Pat. No. 5,369,771; and

Ser. No. 07/814,390, filed Dec. 23, 1991, entitled "Computer with Three-Processor Power-Management Architecture" (DC-265) now abandoned;

both of which are hereby incorporated by reference.

BACKGROUND AND SUMMARY OF THE INVENTION

The present invention relates to computer systems (particularly small computer systems), and to methods for using them.

The innovations disclosed in the present application provide computer systems (especially very small portable personal computers) which have advantageous new capabilities. To better explain the significance and advantages of these innovations, the following paragraphs will review some technological context. This technological context is not necessarily prior art, but is intended to help in pointing out the disclosed inventions.

Handling Keyboard Inputs

One of the basic choices in an interactive computer architecture is how to detect and respond to user inputs on the computer.

Evolution of the Industry-Standard Architecture

In the original IBM PC, the keyboard was located in a separate housing from the system chassis, and was connected to the chassis by a cable with a standard interface definition (which was changed in the transition from XT to

AT). The keyboard housing was powered, through its cable, from the system chassis. The keyboard housing included a separate microprocessor which was used to constantly monitor user keystrokes, and this processor relayed keystroke events to the CPU microprocessor through an interrupt-driven architecture. See generally Norton, "Inside the PC Keyboard," PC WORLD vol.1 no.4 pp.190–197 (1983), which is hereby incorporated by reference; Mefford, "An in-depth exploration of the PC keyboard and its interrupt service routines," MICROSOFT SYSTEMS JOURNAL vol.5 no.3 (May 1990) at pp.22ff, which is hereby incorporated by reference.

In this architecture, the keyboard microprocessor (which in fact is normally a microcontroller) monitors the individual keyswitches (essentially constantly) to detect any changes of state (i.e. whenever the user presses or releases any key). Whenever a keystroke is detected, the keyboard microprocessor sends a high-priority interrupt to the CPU.[1] When the CPU services this interrupt, it gets the key event code from the keyboard microcontroller, translates it to determine appropriate character and/or status codes, and loads this information into a software buffer. The application software can then read this software buffer, in due course, as needed.[2]

[1]This is INT9H.

[2]This can be done by a software interrupt (INT16H), or by a direct read to the address of the area reserved for the keyboard buffer.

This architecture has generally been retained through numerous stages of evolution of the industry-standard architecture (ISA).

In the industry-standard architecture, the keyboard microprocessor has generally been an Intel 8048, but several related chips have also been used.[3]

[3]See generally Levine, "Keyboard Mfrs. Scramble for Microcontrollers," ELECTRONIC NEWS vol.30 no.1482 p.50 (Jan. 30, 1984), which is hereby incorporated by reference. Other chips used have included the 8748 (which includes EPROM), the 8031 (ROM-less), the 8051 (mask-programmed), and the 8042 (the original keyboard controller whose interface is emulated by the more recent chips).

Recent Changes

The most significant departures from this keyboard interface architecture have occurred very recently. In laptop and smaller computers, the keyboard is often integrated into the same chassis as the CPU. Moreover, the low-power needs of such computers have forced reconsideration of all architectural blocks, including the keyboard architecture.

Thus, more recent hardware architectures have begun to handle keystroke inputs in a slightly different way. If the keyboard microprocessor is located in the system chassis, then the simple keystroke-monitoring functions can be combined with other functions too.

For example, the new Intel SL chipset[4] expects to use an 8051SL keyboard controller. This keyboard controller provides power-monitoring functions in addition to the basic keystroke-monitoring functions.

[4]See generally Wilson, "Intel wades into palmtop fray with 386 SL," Computer Design vol.29 no.21 pp.36ff (Nov. 1, 1990), which is hereby incorporated by reference; Martin, "Intel sampling "notebook" microprocessor set," EDN vol.35 no.21A pp.3ff (Oct. 18, 1990), which is hereby incorporated by reference. See also the Intel datasheets for the 386SL, 82360SL (combination peripheral-control and power-management subsystem chip), 8051SL (keyboard controller), and 82760SL (graphics controller), all of which are available from Intel and are hereby incorporated by reference. Many of these chips contain their own power-conservation features; for example, the 8051SL puts itself to sleep if the keyboard has not been touched for a certain length of time. See also the Intel datasheets for the 82347 (power management chip), 82341 (peripheral control), and 82343 and 82344 (AT interface chipset), which are all also hereby incorporated by reference.

Pointing Devices

For any graphical user interface, a simple keyboard is an inadequate channel for user input: a user also needs some

very rapid way to select a point from the two-dimensional display space.

A variety of pointing devices have been used in computer interfaces, including mice, touchpads, trackballs, joysticks, light pens, and touch-sensitive screens. All of these devices work well on the desktop where power and space are not major concerns. They do not work as well when used with a portable computer.

There are two categories of solutions for providing a pointing device in a portable computer. One is integrating a pointing device in the system unit. The other is using a third party pointing device like a mouse or a trackball which clips onto the side of the unit. Both of these solutions add cost, power, size, and weight to the total system solution.

Startup Software (POST, Bootstrap, etc.)

A computer system normally includes a number of complex hardware components (chips and subsystems). When power is first applied to a computer (or when the user triggers a reset after the system has locked up), the various hardware elements (chips and subsystems) will each have their own internal procedures (reset procedures) to regain a stable and known state. However, at some point (if the hardware is intact), these reset procedures will have ended, and at this point the CPU performs various important overhead tasks[5] under software control. This phase of operation is generally referred to as "POST" (Power-On-Self-Test).

[5]These include, for example, surveying the system configuration, performing sanity checks on system hardware, issuing diagnostic signals (such as sounding beeps through a speaker or turning on LEDs), and permitting the user to branch into an NVRAM configuration program.

After POST, a "bootstrap" program is run, to permit the CPU to begin execution of other software. For robustness, the POST and bootstrap software is normally stored in a read-only memory. The bootstrap program launches the CPU on execution of the primary operating system software;[6] the primary operating system can then be used by the user to launch an application program, either manually or automatically.

[6]Depending on how the system has been set up, the boot software may direct program execution into DOS, Unix, OS/2, a DOS variant, or another operating system. This is normally automatic and predetermined, but is manually user-selectable in some systems. However, the choice of operating system is not particularly relevant to the inventions described in the present application.

Bootstrap Programs

Any computer system must have some way to begin program execution after a cold start. The hardware architecture of a microprocessor (or other CPU) will normally provide for a "reset" operation, which places all of the hardware circuits in a known electrical state; but it is still necessary to start the CPU on execution of a desired program. For example, in the very early days of computing, some computer systems would be manually configured to read in a "bootstrap loader" program at startup. This bootstrap program was a simple program which loaded in, and started execution of, another sequence of instructions, which were the beginning of the desired program. Bootstrap programs are often referred to simply as "boot" software.

To give a more recent example, the Intel 80×86 microprocessors, after a hardware reset, will always attempt to begin program execution from a specific memory address.[7] Thus, if a branch (or conditional branch) instruction is found at this address, the microprocessor will continue its program execution from whatever address is specified.

[7]That is, the microprocessor will read the contents of that memory location, and will attempt to execute the bits it finds there as a microprocessor instruction. The specific memory location used by the 80×86 family is xxxFFFF0h, i.e. 16 bits below the top of the memory space. Other microprocessors may use a different starting address, but similar principles apply.

Thus, this initial target address is the entry point for every session of use. This address is normally used to enter execution of programs which must be run every time the computer is used.

"Basic Input/Output System" Software (BIOS)

The "basic input/output system" (BIOS) software contains frequently-used routines for interfacing to key peripherals, for interrupt handling, and so forth.[8] For system robustness, the BIOS software itself is normally packaged in nonvolatile memory[9] with other key pieces of overhead software, such as POST,[10] boot, and configuration management routines,[11] as well as a pointer to launch the computer into the operating system software.[12] (Thus, the term "BIOS" is often used more broadly, to refer to this whole collection of basic system routines in ROM or EPROM.)

[8]Thus, the BIOS software provides some degree of machine-independence. However, in PC-class computers, this independence is not fully exploited by the available commercial software. Many programs bypass the BIOS software, and directly access the underlying hardware addresses or devices. See generally Glass, "The IBM PC BIOS," Byte, April 1989, pp. 303ff.

[9]However, in 1991 IBM introduced a PS/2 system in which the BIOS is at least partially stored on disk.

[10]The POST routines provide an extensive check for hardware integrity.

[11]Packaging the BIOS, POST and boot routines in ROM makes a very robust firmware system. Short of hardware damage, it is very difficult for a user to distort the system to the point where it will not start up and run (if the operating system software is present).

However, this system also provides a considerable degree of flexibility. As the operating system starts up (after the POST and boot routines), the user can remap address pointers to revector BIOS calls away from the standard BIOS routines, if desired. (It is also common for users to map out the entire BIOS contents into fast RAM, for greater speed.)

[12]Depending on how the system has been set up, the BIOS software may direct program execution into DOS, Unix, PS/2, a DOS variant, or another operating system. However, the choice of operating system is not particularly relevant to the inventions described in the present application.

In many types of modern personal computers (and in all "IBM-compatible" personal computers), a key part of the system software is a "basic input/output system" (BIOS) program.[13] The BIOS program contains frequently-used routines for interfacing to key peripherals,[14] for interrupt handling, and so forth.[15]

[13]See generally, e.g., P. Norton, THE PETER NORTON PROGRAMMER'S GUIDE TO THE IBM PC (1985), which is hereby incorporated by reference.

[14]The term "peripheral" or "peripheral component" normally refers to those components of a computer system which are not on the motherboard, i.e. which must be addressed through a system bus or over a defined port. However, the usage of this term is somewhat variable; sometimes it is used to refer to any I/O device, or only to refer to components which are optional add-ons.

[15]Thus, the BIOS software provides some degree of machine-independence. However, in PC-class computers, this independence is not fully exploited by the available commercial software. Many programs bypass the BIOS software, and directly access the underlying hardware addresses or devices. See generally Glass, "The IBM PC BIOS," *Byte*, April 1989, pp. 303ff.

For system robustness, the BIOS software is normally packaged in a read-only-memory. In fact, it is normally packaged together with the startup software mentioned above. Thus, nowadays the term "BIOS" is often used, somewhat more broadly, to refer to this whole collection of basic system routines.

Operating System Software

The application software will normally interface to an operating system (such as DOS, DOS+Windows, OS/2, UNIX of various flavors, or UNIX plus X-windows). The operating system is a background software program[16] which

provides an application programming interface (API) for use by the application software. Thus, the programmers writing application software can write their software to fit the API, rather than having to find out and fit the peculiarities of each particular machine.[17]

[16]Some operating systems run continuously, or at least start up at regular intervals, even while an application program is running; other operating systems merely provide capabilities which can be called on by the application software.

[17]See, e.g., Quedens, "Windows virtual machine," *PC Tech Journal* vol.5, no. 10 pp.90, 92–93, 95, 97, 99–100, 102 (Oct. 1987), which is hereby incorporated by reference.

Device Driver Software

A device driver is a lower level of operating system software. Typically a device driver interfaces to the actual peripheral hardware components, and provides routines which application software can use to access the hardware components. Thus, the application software can simply make a call to an installed software subroutine, instead of having to find the specifications of each peripheral device and branch accordingly, whenever a peripheral I/O operation is needed. This permits application software to ignore the detailed specifications of peripheral hardware.

Normally device driver software must contain a description of each target hardware platform. Thus, the software must be revised repeatedly, for reasons which are beyond the control of the companies making peripherals.

In personal computers, installable device drivers were first introduced in DOS 2.0. The role of device drivers has since been expanded, in subsequently introduced operating systems.

In particular, OS/2 provided expanded support for device drivers, including a library of "DevHlp" routines which can be called by device drivers. See generally Duncan, "An examination of the DevHlp API (writing OS-2 bimodal device drivers)," 3 *Microsoft Systems Journal* no.2 (March 1988) at 39ff; Schmitt, "Designing drivers for OS/2: I," *PC Tech Journal* vol.5, no.12, p.164 (1987); and Schmitt, "Designing drivers for OS/2: II," *PC Tech Journal* vol.6, no.2 p.136–155 (Feb. 1988), all of which are hereby incorporated by reference.

Application Software

From a system designer's point of view, the application software is (subject only to the minimal constraints of the architectural standards) wholly unpredictable.[18] Common applications include spreadsheets, word processors and publishing programs, databases, games, project managers and a wide variety of others; but inevitably users will also run customized applications, and new types of applications.

[18]Many clever people are constantly looking for new ways to exploit the standard architecture, and many innovations continually result. Thus, hardware architects must expect that the application software will not only be unpredictable, but will be as unpredictable as possible.

Laptop and Smaller Computers

Portable personal computers were introduced in the early 1980s, and proved to be very useful and popular. As this market has developed, it has become increasingly clear that users strongly desire systems to have small volume, small weight, physical durability, and long battery-powered lifetime. Thus, small portable computers ("laptop" computers) have proven extremely popular during the late 1980s. Users continue to demand more features, longer time between recharges, and lower weight and volume. This combination of demands is difficult to meet. Moreover, as of 1990, another smaller generation of portable computers has begun to appear, referred to as "notebook" computers. This smaller form factor will only exacerbate the difficulty of the above tradeoffs.

Some Difficulties of Small Size

To meet the weight, volume, and power budgets of small portable computers, much effort has been made to re-engineer familiar components, within the limits of acceptable compromise with user comfort and convenience, to attain the needed portability. For example, almost no laptop or notebook computers have had more than two disk drives (typically one floppy drive and one hard drive); the keyboards are much smaller than would be optimal for touch typists; and the displays are much smaller than on common desktop models.

Where such compromises are needed to attain the overriding goal of portability, users readily accept them. However, if these compromises can be avoided some of the time, it is highly desirable to do so. Thus, for example, many users will "dock" their laptop or notebook computers, when on their desktops, in order to use better peripherals (faster, larger, more heavy-duty, and/or easier to use) than could be carried with the portable computer.

Portable computers are inherently more susceptible than desktop computers to accident, loss, and theft. Thus, if critical work is being done on a portable computer, backup is even more of a necessity than with a desktop computer.

The quantities of mass storage available on portables have steadily increased, but the cost per byte of the necessary ruggedized drives continues to be far above that of that of the drives normally used. This disparity seems likely to continue. Similarly, although some small portables use nonvolatile (or nonvolatized) solid-state memory to replace disk drives, the cost per byte of such memory is likely to continue to exceed that of conventional mass storage devices.

Laptops normally have a severely limited set of external ports. This limitation is imposed by several factors: first, each external connector takes up precious square inches of surface area. Second, each external connector is a point of vulnerability to electrostatic-discharge-induced component failure. Third, each external connector is a possible point of entry for dirt and moisture. Fourth, in calculating the worst-case power budget for a system, the possible power required by all connectors must be considered.

Similar problems arise from the need for bus extension. Internal space is not available for expansion cards, as in a normal personal computer; but needs for expansion still exist. Some small computers have brought out key bus lines into an external connector, but of course this is unwieldy. Other small computers have sent signals out over a port to an independently-powered extension bus controller with bus slots which would emulate the computer's internal bus.

Continuing Advantages of Stationary Personal Computers

The capabilities and cost of both stationary and portable computers have rapidly improved over the years. However, it is likely that stationary personal computers will always have some advantages over lightweight portables: it has always been true that, for a given price, stationary computers have always had more computing horsepower, larger mass storage, better displays, and better user interfaces (including keyboard, mouse, joystick and/or track ball input devices). These advantages are due to the necessary constraints (of

weight, volume, power budget, and shock-resistance, and environmental resistance) which necessarily must be met by a small portable computer, and need not be met by a stationary personal computer.

Use of Both a Portable and a Stationary Computer

As small portable computers become ever more common, an increasing number of users prefer to use two computers: one for their desktop, and one more for the road. This increasingly common pattern of usage is very convenient for users, but also generates some problems.

One problem which arises is loss of file coherency: when a user edits a file on his secondary machine, he must transfer that file back to his primary machine before he again edits the same file on the primary machine.

"Docking" a Portable Computer

Users find it very convenient to "dock" a portable computer on their desktop, to a full-size keyboard and/or display. When a portable computer is used in such a configuration, users will also wish to use many of the peripherals (such as printer, modem, fax machine, tape backup devices, or additional disk drives) which are easily available with a desktop computer.

This problem becomes more urgent as useful amounts of computing power become more available in physically small portable computers.

Thus, in general, as convenient as small portable computers are, there is some difficulty getting data into and out of them. Usually the only available data routes are through a modem or through the floppy drive.[19] This has been recognized as a problem, and many vendors have offered external drive connections for small portable computers.

[19]Some portable computers have LAN interface cards, but this is an expensive option, and requires a compatible LAN to interface to.

Innovative Computer System and Method

The disclosed inventions provide a portable-computer-plus-pointing-device solution that adds little or no additional cost, power consumption, size, or weight. The invention can be added to most portable computer systems by making only two alterations to the system design: 1) Additions must be made to the keyboard controller microcode, and 2) A pointing device interrupt output must be added to the Keyboard controller. Optionally, a key may be added to the keyboard to select pointing device mode.

In the presently preferred embodiment, a suitable additional key is available for other reasons. In this computer the keypad numeric keys[20] are overlaid onto certain of the standard alphabetic keys. That is, as may be seen from the layout of FIG. **1**, when the [Fn] key is pressed, pressing the J key will output a "1". Similarly, pressing [Fn]-K will output "2", etc.

[20]Most standard computer keyboards include not only a full typewriter-style keyboard, but also a ten-key keypad to the right of the typewriter-style keyboard. This keypad permits rapid one-hand entry of numbers.

The presently preferred embodiment provides a small computer architecture in which the CPU can receive multiple kinds of interrupt signals, including one kind which is assigned to indicate the occurrence of a keystroke input and another kind which is assigned to indicate the occurrence of a pointing device input. However, the computer does not include any pointing device as such. Instead, the keyboard microprocessor (i.e. a microprocessor other than the CPU) monitors user keystrokes to the computer's keyboard, and: in response to keystroke combinations which include the special function key (e.g., the [Fn] key) and a pointer key (e.g., an arrow key), the microprocessor sends a pointing device interrupt to the CPU, and provides data output corresponding to an emulated pointing device movement; and in response to simple keystrokes or keystroke combinations other than those described in the preceding paragraph, the microprocessor sends a keystroke interrupt to the CPU.

A substantial advantage of this architecture is that the pointing-device emulation can be made completely invisible to the application software. In many cases the user will not even have to install a device driver,[21] and in any case will not have to install a non-standard device driver.[22]

[21]In the presently preferred embodiment, the keyboard pointing inputs are translated to emulate a PS/2 mouse. This is a highly standard interface, recognized by most applications. See the IBM Technical Reference Manuals for the PS/2 Models 50, 70, and 80, which are all hereby incorporated by reference.

[22]For example, if the keyboard pointing device is configured to emulate a serial mouse which requires a device driver, then of course a device driver will be required for the keyboard pointing device operation too.

A further feature of the presently preferred embodiment is that one of the simple key combinations toggles the keyboard into (or out of) pointing-device-emulation mode.

A further feature of the presently preferred embodiment is that if the user prefers, he can disable the pointing-device emulation (either sometimes or always), in favor of an external pointing device. Thus, for example, users can use an ergonomically optimal mouse (or trackball, or joystick) on their home desk, while still retaining convenient usage of GUI software on the road.

The system of the presently preferred embodiment also includes a standard dedicated mouse port, and inputs to this port are handled by the same 8051SL chip which handles keyboard inputs. Thus, it is actually possible, in the presently preferred embodiment, for the user to provide mouse inputs both through the keyboard emulation and also through a mouse plugged into the mouse port. (However, in alternative embodiments such dual-mouse operation can of course be precluded.)

Many graphics-intensive software applications can be greatly accelerated by occasional use of character input. Thus, in many graphics-intensive environments, the user should ideally be able to switch rapidly back and forth between keyboard inputs and pointing-device inputs. However, this is not possible with the conventional external pointing devices: for example, with a conventional mouse, the user loses his keyboard position when he grabs for the mouse. Even with configurations which include a tiny trackball (or "inverted mouse") on the keyboard, a hand shift is typically necessary (or desirable) between trackball use and touch typing. However, the disclosed innovations enable a computer system in which no hand shift is necessary between touch-typing position and the pointing-device-input position. For some users this will be a significant ergonomic advantage.

BRIEF DESCRIPTION OF THE DRAWING

The present invention will be described with reference to the accompanying drawings, which show important sample embodiments of the invention and which are incorporated in the specification hereof by reference, wherein:

FIG. **1** shows a perspective view of the notebook computer **100** of the presently preferred embodiment in the open position.

FIGS. **2–10***bc* schematically illustrate the circuitry of computer **100**, in this sample embodiment.

FIG. 11 shows the keyboard layout used in the presently preferred embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The numerous innovative teachings of the present application will be described with particular reference to the presently preferred embodiment. However, it should be understood that this class of embodiments provides only a few examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily delimit any of the various claimed inventions. Moreover, some statements may apply to some inventive features but not to others.

FIG. 11 shows the keyboard layout of the presently preferred embodiment. Of course, other specific keyboard layouts can be used instead, and this specific layout, while illustrative of the presently preferred best mode, does not limit any of the claimed inventions.

Note that, in addition to the [Control], [Alt], and [Shift] keys, this keyboard also contains a fourth overlay key, the [Fn] key. The [Fn] key too can be used in chording combinations, but it is used in a different way than the [Control], [Alt], and [Shift] keys.

Note further that, as used herein, the terms "chording key," "special function key," and "pointer key" have special meanings. "Chording key" refers to the [Control], [Alt], or [Shift] keys which are usually pressed in combination with other keys to enable the latter keys to be used for multiple functions. The term "chording" derives its meaning from its usage in music where it refers to multiple tones sounded together in harmony to produce a certain effect. A "special function key" is a key such as the [Fn] key. A "pointer key" is a key such as (1) an arrow key which may be used to emulate directional movement of a pointing-device (e.g., a mouse), or (2) any other key which may be used to emulate a pointing-device button. Because the pointer key may often be used for other functions also, it is used to emulate mouse functions only while the special function key is also pressed.

A sample specific embodiment of this invention is implemented as follows, in a notebook computer with the keyboard shown in FIG. 1. The keyboard operates in two modes: keyboard mode and pointing device mode. It may operate in both modes at once. Keyboard mode operates identically to the way it works on the current Apollo system. When operating in pointing device mode, the keyboard emulates a standard pointing device. The operating system and application software are not aware that a standard pointing device is not present. The keyboard controller communicates with standard pointing device protocol and interrupt level.

Pointing device mode is selected when the operator depresses the [Fn] key, simultaneously with the one or a combination of the six pointer keys. The six pointer keys could be [–], [Alt], [up-arrow], [down-arrow], [left-arrow], and [right-arrow]. These keys were selected for the Apollo keyboard because they are located such that using them in combination would be both intuitive and easy. Other combinations could be selected if the design of the keyboard prevented these combinations from working, or if other selections were more desirable. The key or key-combinations would have the effects shown in the following table.

Keyboard Key Combination	Pointing Device Operation Emulated
[Fn]-[~]	button 1
[Fn]-[Alt]	button 2
[Fn]-[CapsLock]	lock/unlock pointing device mode: When pointing device mode is locked, the [Fn] does not need to be depressed as part of the combination
[Fn]-[up-arrow]	move pointer up
[Fn]-[down-arrow]	move pointer down
[Fn]-[left-arrow]	move pointer left
[Fn]-[right-arrow]	move pointer right
[Fn]-[up-arrow]-[right-arrow]	move pointer diagonal to top right
[Fn]-[up-arrow]-[left-arrow]	move pointer diagonal to top left
[Fn]-[down-arrow]-[right-arrow]	move pointer diagonal to bottom right
[Fn]-[down-arrow]-[left-arrow]	move pointer diagonal to bottom left
[Fn]-[~]-[up-arrow]	button 1 and move pointer up
[Fn]-[~]-[down-arrow]	button 1 and move pointer down
[Fn]-[~]-[left-arrow]	button 1 and move pointer left
[Fn]-[~]-[right-arrow]	button 1 and move pointer right
[Fn]-[~]-[up-arrow]-[right-arrow]	button 1 and move pointer diagonal to top right
[Fn]-[~]-[up-arrow]-[left-arrow]	button 1 and move pointer diagonal to top left
[Fn]-[~]-[down-arrow]-[right-arrow]	button 1 and move pointer diagonal to bottom right
[Fn]-[~]-[down-arrow]-[left-arrow]	button 1 and move pointer diagonal to bottom left
[Fn]-[Alt]-[down-arrow]	button 2 and move pointer up
[Fn]-[Alt]-[down-arrow]	button 2 and move pointer down
[Fn]-[Alt]-[left-arrow]	button 2 and move pointer left
[Fn]-[Alt]-[right-arrow]	button 2 and move pointer right
[Fn]-[Alt]-[up-arrow]-[right-arrow]	button 2 and move pointer diagonal to top right
[Fn]-[Alt]-[up-arrow]-[left-arrow]	button 2 and move pointer diagonal to top left
[Fn]-[Alt]-[down-arrow]-[right-arrow]	button 2 and move pointer diagonal to bottom right
[Fn]-[Alt]-[down-arrow]-[left-arrow]	button 2 and move pointer diagonal to bottom left

These particular emulation assignments are adapted to the geometry of the keyboard shown in FIG. 1. However, other emulation assignments can be used. In particular, if a keyboard with a different layout is used, it may be preferable to adjust some of the emulation assignments accordingly. A significant criterion for a pointing device is that the user should be able to move the cursor position at will while looking only at the screen, and not at the pointing device.

If another key is struck while a pointing device combination is also depressed, the keyboard controller will respond with both the normal keyboard key and the pointing device operation. This may not be possible in some circumstances due to the design of the keyboard. It is thought that the occurrences of such events will be very infrequent in practice. When such an event occurs the operator will have the option of locking the pointing device mode. This will reduce the number of keys which have to be depressed at one

time, and increase the likelihood that the desired operation will be recognized.

Typeamatic operation works with the combinations as well. For instance, pressing and holding the combination, [Fn]-[up-arrow] would cause the pointer to continue to move upward at a rate proportional to the typeamatic rate.

Hardware Implementation Details

Following is a sample specific implementation of hardware portions of the disclosed innovations. This example is intended to better illustrate the disclosed generally applicable innovations, and to comply with the best-mode requirements of US patent law; but of course this specific implementation is subject to change (and will certainly be changed as time goes by), and can be altered, as will be apparent to those skilled in the art, in many details.

The presently preferred embodiment will be described in its implementation in the Dell 325NC computer. This is a notebook computer, with an external closed size of about 8.5×11×2 inches.

FIG. **1** shows a perspective view of the notebook computer **100** of the presently preferred embodiment in the open position. Visible elements include case **102**, cover **104**, hinges **106**, display screen **110**, keyboard **120**, floppy disk drive **130**, and dust cover **103** (which covers the receptacle for the rechargeable battery pack).

FIG. **11** shows the keyboard layout used in the presently preferred embodiment. Note that, in addition to the usual chording keys ([Ctrl], [Alt], and [Shift]), this keyboard includes an additional chording, or special function key, labelled [Fn]. This key enables the keypad keys which are mapped into the keyboard, and also enables mouse emulation functions which are also mapped into the keyboard.

FIGS. **2**–**10***b* schematically illustrate the circuitry of computer **100**, in this sample embodiment. As shown in FIG. **2**, a processor chipset **202** and **206**, similar to the Intel 386SL Superset made of the 80386SL and 82360SL chips, is used. (Of course, a very wide variety of other chipsets can be used instead.) Chip **202** includes a 386SX processor core, and also includes ISA bus control logic (connected to ISA bus **207**). Chip **202** also includes memory controller logic (connected to main memory **203** by 22 address lines, 16 data lines, and RAS and CAS strobe signals). Chip **202** also includes cache control tag RAMs (connected to cache RAM **203**A). Chip **202** also includes math coprocessor interface logic (connected to optional math coprocessor **209**). Chip **202** also includes CPU power management logic. Chip **202** receives a clock input (16 MHz and 50 MHz in this example) from clock/timing generator **204**.

Chip **206** is connected to receive clock line SYSCLK from chip **202**. Chip **206** also includes timer logic (approximately equivalent to two 8254s), which is connected to receive a 14.31818 MHz clock signal from clock generator **204**. Chip **206** also includes interrupt controller logic (approximately equivalent to two 8259As). Chip **206** also includes DMA controller logic (approximately equivalent to two 8237s). Chip **206** also includes bus interface logic which is connected to the ISA bus **207**. Chip **206** also includes power management logic, and a real-time clock (approximately equivalent to a 146818), which is connected to receive a 32.768 KHz signal from clock generator **204**. Chip **206** also includes serial port control logic (approximately equivalent to two 16450s), which is connected to receive a 1.8432 MHz signal from clock generator **204**.

Flash EPROM **201** contains code for BIOS and for the operating system (e.g. DOS). This chip, in the presently preferred embodiment, is a 256K×8 memory.

A microcontroller **210** (an 80C51SL, in the presently preferred embodiment) receives keyboard inputs from keyboard **212**. Associated with microcontroller **210** are SRAM **211** (32K×8 in this sample embodiment) and resonant crystal oscillator **213** (16 MHz in this sample embodiment). Video signals from VGA controller **240** (which is a WD90C22 in the presently preferred embodiment), control the LCD display **242**. VGA controller **240** is also connected to programmable pixel clock **243**.

Also connected to bus **207** is a floppy disk drive controller **250** (connecting to floppy disk drive **251**) and an IDE interface **270**, which is connectible to the internal hard disk drive **271** (or to a semiconductor mass-memory drive emulation, if a diskless configuration is chosen). A power supply controlled by controller **230** is driven by a battery, and also may be powered by external power through an input socket. This power supply, in the presently preferred embodiment, provides output voltages of ±5V, ± 12V, and 18.6V. However, of course, lower logic supply voltages may be used in future embodiments.

The 82360SL chip connects to industry-standard connectors for serial ports (DB-9), parallel ports (DB-25), and optional modem unit **260**.

FIGS. **3***a*–*b* (which fit together to form one schematic diagram) indicate the functional portions of chips **202** and **206** and a slightly different selection of other items but does not include the items of FIG. **2** which are located on the bottom board (within broken line in the center of FIG. **2**), and FIGS. **4***a*–**10***b* illustrate the wiring in more detail. In particular, 80386SL chip **202** appears in FIG. **4***b* along with the cache **203**A, DRAM memory **203**, and math coprocessor **209** in FIG. **4***a*. 82360SL chip **206** appears in FIG. **5***b* along with EPROM **201** in FIG. **5***a*. Keyboard controller **210** is in FIG. **6***a* with the external keyboard connector and mouse connector in the lower righthand portion of FIG. **6***b*. Clock generator **204** appears in FIG. **8***a*.

The external clock is divided down to provide the CPU clock, any math coprocessor clock, and internal clock signals. The maximum CPU clock frequency is one half the EFI (external frequency input), but EFI may be further divided to generate the CPU clock: the contents of the CPUPWRMODE register within 386SX core may provide further division by 2, 4, or 8 (or even stopping the CPU clock). The CPU clock frequency sets the machine cycle time, and thus basically the CPU clock frequency determines the number of transistor switchings per second. The power consumed by 80386SL **202**, memory **203**, 82360SL **206**, . . . occurs primarily during transistor switchings; that is, the typical CMOS devices making up the circuitry have virtually zero static power consumption. Of course, power consumed by items such as display screen backlighting, DRAM refresh, and the real time clock will be independent of the machine cycle time; however, the bulk of the circuitry, which represents about half of the power in a backlit-screen system, will consume power roughly in proportion to the CPU clock frequency. System **100** also has APM advanced power management compatible BIOS.

Software Implementation Details

Following is a sample specific implementation of software portions of the disclosed innovations. This example is intended to better illustrate the disclosed generally appli-

cable innovations, and to comply with the best-mode requirements of US patent law; but of course this specific implementation is subject to change (and will certainly be changed as time goes by), and can be altered, as will be apparent to those skilled in the art, in many details.

The code portions set forth below provide the most relevant portions of the code used, but of course these code portions are used in combination with large additional portions of code. In the presently preferred embodiment, the software structure given below is used in combination with Phoenix™ BIOS code; but several vendors (such as AMI) offer BIOS software, and the disclosed code can be used with another vendor's BIOS code if desired. (Of course, minor adjustments would have to be made to the called routine names, etc.)

large additional portions of code. In the presently preferred embodiment, the software structure given below is used in combination with Phoenix™ BIOS code; but several vendors (such as AMI) offer BIOS software, and the disclosed code can be used with another vendor's BIOS code if desired. (Of course, minor adjustments would have to be made to the called routine names, etc.)

```
;*****************************************
;
; KEYBOARD MOUSE IMPLEMENTATION
;
;*****************************************

;*****************************************
;
; SET UP MOUSE BYTE IF Fn plus appropriate key
;
;*****************************************

ktl_procl:
      mov   a,r3                ;Silent Fn Test
      jb    acc.fn_silent_bit,befun
      mov   bMOUSE,#0           ;clear mouse byte
      jmp   not_fn        ;jump if Fn state off

befun:
;*********************************************
      call  check_mickey
      jc    jdone
      jmp   check_kbdsmi
jdone:
      jmp   done
;*********************************************
check_mickey:
      mov   a,bGHOST_CMD_NUM
      jnb   acc.bitEXT_MOUSE,int_mouse
      clr   c
      jmp   exit_mouse
int_mouse:
      mov   a,bMOUSE
```

```
        cjne  r1,#UP_ARROW,notmup
        setb  acc.5
        cjne  r2,#1,mup   ;jump if a make contact
        clr   acc.0
        jmp   notmup
mup:
        setb  acc.0
notmup:
        cjne  r1,#DOWN_ARROW,notmdown
        setb  acc.5
        cjne  r2,#1,mdown ;jump if a make contact
        clr   acc.1
        jmp   notmdown
mdown:
        setb  acc.1
notmdown:
        cjne  r1,#RIGHT_ARROW,notmright
        setb  acc.5
        cjne  r2,#1,mright      ;jump if a make contact
        clr   acc.2
        jmp   notmright
mright:
        setb  acc.2
notmright:
        cjne  r1,#LEFT_ARROW,notmleft
        setb  acc.5
        cjne  r2,#1,mleft ;jump if a make contact
        clr   acc.3
        jmp   notmleft
mleft:
        setb  acc.3
notmleft:
        cjne  r1,#LEFT_BUTTON,notmlbut
        cjne  r2,#1,mlbut ;jump if a make contact
        jnb   acc.6,notmlbut
        setb  acc.5
        clr   acc.6
        jmp   notmlbut
mlbut:
        jb    acc.6,notmlbut
        setb  acc.5
```

```
        setb  acc.6
notmlbut:
        cjne  r1,#RIGHT_BUTTON,notmrbut
        cjne  r2,#1,mrbut ;jump if a make contact
        jnb   acc.7,notmrbut
        setb  acc.5
        clr   acc.7
        jmp   notmrbut
mrbut:
        jb    acc.7,notmrbut
        setb  acc.5
        setb  acc.7
notmrbut:
        cjne  r1,#SLOW_MOUSE,notmfast
        setb  acc.5
        cjne  r2,#1,mfast ;jump if a make contact
        clr   acc.4
        jmp   notmfast
mfast:
        setb  acc.4
notmfast:
        clr   c
        jnb   acc.5,exit_mouse  ;exit if no change
        clr   acc.5
move_mouse:
        mov   bMOUSE,a
        %SET_MASK(bSERVICE,maskGHOSTCMD)  ;move that mouse!
        setb  c
exit_mouse:
        ret

;*************************************
;
; MOVE MOUSE ROUTINE - TO BE ADDED AS A TASK IN MAIN LOOP
;
;*************************************

BUMP_MOUSE:
        %LOCK_SCAN
        clr   %bitIRQ_PCI             ; disable commands from PC
```

```
        mov   b,bMOUSE
        mov   a,#008h
        jnb   b.bitMOUSE_LEFT_BUTTON,nolb
        setb  acc.0
nolb:
        jnb   b.bitMOUSE_RIGHT_BUTTON,norb
        setb  acc.1
norb:
        jnb   b.bitMOUSE_DOWN,nodn
        setb  acc.5
nodn:
        jnb   b.bitMOUSE_LEFT,nolf
        setb  acc.4
nolf:
        call  SENDAUX_DATA            ; SEND TO PC

        mov   DPTR,#regPC_STAT
        mov   r7,#0             ; retry counter
wait4MB0:
        movx  a,@DPTR
        jnb   acc.bitPCS_OBF,mb0done  ; Clear out output buffer
        djnz  r7,wait4MB0
        jmp   mb2done
mb0done:
        clr   a
        mov   b,bMOUSE
        jnb   b.bitMOUSE_RIGHT,noright
        mov   a,#003h
        jnb   b.bitMOUSE_SLOW,noright
        mov   a,#001h
noright:
        jnb   b.bitMOUSE_LEFT,noleft
        mov   a,#0fdh
        jnb   b.bitMOUSE_SLOW,noleft
        mov   a,#0ffh
noleft:
        call  SENDAUX_DATA            ; Send to PC

        mov   DPTR,#regPC_STAT
        mov   r7,#0
wait4MB1:
```

```
        movx   a,@DPTR
        jnb    acc.bitPCS_OBF,mb1done  ; Clear out output buffer
        djnz   r7,wait4MB1
        jmp    mb2done
mb1done:
        clr    a
        mov    b,bMOUSE
        jnb    b.bitMOUSE_UP,noup
        mov    a,#003h
        jnb    b.bitMOUSE_SLOW,noup
        mov    a,#001h
noup:
        jnb    b.bitMOUSE_DOWN,nodown
        mov    a,#0fdh
        jnb    b.bitMOUSE_SLOW,nodown
        mov    a,#0ffh
nodown:
        call   SENDAUX_DATA            ; Handle as normal mouse

        mov    DPTR,#regPC_STAT
        mov    r7,#0
wait4MB2:
        movx   a,@DPTR
        jnb    acc.bitPCS_OBF,mb2done  ; Clear out output buffer
        djnz   r7,wait4MB2
mb2done:

               setb  %bitIRQ_PCI       ; enable commands from PC

        mov    DPTR,#regPC_STAT
        movx   a,@DPTR
        clr    acc.bitPCS_OBF          ; Clear out output buffer
        movx   @DPTR,a

        %UNLOCK_SCAN

        jmp    main_service
```

Further Modifications and Variations

Further Modifications and Variations

It will be recognized by those skilled in the art that the innovative concepts disclosed in the present application can be applied in a wide variety of contexts. Moreover, the preferred implementation can be modified in a tremendous variety of ways. Accordingly, it should be understood that the modifications and variations suggested below and above are merely illustrative. These examples may help to show some of the scope of the inventive concepts, but these examples do not nearly exhaust the full scope of variations in the disclosed novel concepts.

Some innovative features of the preferred embodiment can be applied to non-portable computers. While such embodiments are less preferred, they may still provide some advantages derived from the disclosed innovative teachings.

For example, sequential logic can be used to trigger the pointing mode from other keystroke combinations which do not include the [Fn] key. This is less preferable, but could be used to attain some of the advantages of the claimed invention.

For another example, the pointing mode does not have to strictly emulate a target pointing device: other functions can be added, e.g. rapid change in the speed of cursor movement.

It should also be recognized that the inventions described are applicable to other architectures as well as to the particular style of architecture described. For example, an architecture does not have to be interrupt-driven to make use of the disclosed innovative ideas: these ideas can also be applied in architectures where a polled keyboard interface is used instead of the keyboard interrupt of the presently preferred embodiment. For another example, the disclosed inventions are not only applicable to uniprocessor architectures, but can also be applied to multiprocessor systems.

As will be recognized by those skilled in the art, the innovative concepts described in the present application can be modified and varied over a tremendous range of applications, and accordingly the scope of patented subject matter is not limited by any of the specific exemplary teachings given.

What is claimed is:

1. A computer system comprising:

at least one central processing unit (CPU) having a plurality of inputs, said CPU being configured to recognize a first combination of input signals as a keystroke interrupt and to recognize a second combination of input signals as a pointing-device interrupt;

a keyboard, having an array of binary keys including a special function key and a plurality of pointer keys; and

processor means, operatively connecting said keyboard to said CPU, to detect keystrokes of said keyboard and to drive said inputs of said CPU, said processor means enabling said keyboard to operate in a keyboard mode and in a pointing-device mode in which said pointer keys are used to enter pointing information, said keyboard normally operating in said keyboard mode and being actuated to said pointing-device mode responsive to actuation of said special function key, said processor means being configured and programmed such that in response to actuation of any key or combination of keys thereof while said keyboard is in said keyboard mode, said processor means drives said first combination of input signals to said CPU, and in response to actuation of one or more of said binary keys or one or more pointer keys while said keyboard is in said pointing-device mode, said processor means drives said second combination of input signals to said CPU, wherein said one or more of said binary keys emulates one or more buttons of a pointing device and said pointing keys emulate directional movement of a pointing device.

2. The system of claim **1**, wherein no joystick, mouse, or trackball is attached to said system.

3. The system of claim **1**, wherein said system does not include any pointing device other than said keys of said keyboard.

4. The system of claim **1**, wherein said system further includes a pointing device input connector which is operatively connected to provide input signals to said processor means, and wherein said processor means provides said second combination of input signals to said CPU in response to any input signals which may be received from a pointing device through said pointing device input connector.

5. The system of claim **1**, wherein said keyboard is configured such that all of said binary keys are readily accessible from a standard touch-typing hand position.

6. The system of claim **1**, wherein after a predetermined first sequence of keystrokes has occurred and until a predetermined second sequence of keystrokes has occurred, said processor means drives said second combination of input signals to said CPU, in response to actuation of one or more pointer keys without simultaneous actuation of said special function key.

7. The system of claim **1**, wherein said plurality of pointer keys of said keyboard include at least four arrow keys.

8. The system of claim **1**, further comprising nonvolatile memory which is connected to an address which is called by said CPU whenever said CPU emerges from reset, said nonvolatile memory containing bootstrap software and also containing basic system software which causes said CPU to recognize said first combination of signal inputs as a keystroke interrupt and to recognize said second combination of signal inputs as a pointing-device interrupt.

9. A computer system with an integrated typing and pointing capability, comprising:

at least one central processing unit (CPU) having a plurality of inputs, said CPU being configured to recognize a first combination of input signals as a keystroke input and to recognize a second combination of input signals as a pointing-device input;

a keyboard, having an array of binary keys including a special function key and a plurality of pointer keys, said keyboard normally operating in a keyboard mode and operating in a pointing-device mode responsive to actuation of said special function key; and

processor means, operatively connecting said keyboard to said CPU, to detect keystrokes on said keyboard and to drive said inputs of said CPU, said processor means being configured and programmed such that in response to actuation of any key or combination of keys thereof while said keyboard is in said keyboard mode, said processor means drives said first combination of input signals to said CPU, and in response to actuation of one or more of said binary keys or at least one of said pointer keys while said keyboard is in said pointing-device mode, said processor means drives said second combination of input signals to said CPU, wherein said one or more of said binary keys emulates a button of a pointing device and said pointing keys emulate directional movement of a pointing device.

10. The system of claim **9**, wherein no joystick, mouse, or trackball is attached to said system.

11. The system of claim **9**, wherein said system does not include any pointing device other than said keys of said keyboard.

12. The system of claim 9, wherein said system further includes a pointing device input connector which is operatively connected to provide input signals to said processor means, and wherein said processor means provides said second combination of input signals to said CPU in response to any signals which may be received from a pointing device through said pointing device input connector.

13. The system of claim 9, wherein said keyboard is configured such that all of said binary keys are readily accessible from a standard touch-typing hand position.

14. The system of claim 9, wherein after a predetermined first set of keystroke inputs has occurred and until a predetermined second set of keystroke inputs has occurred, said processor means drives said second combination of signal input lines, in response to actuation of one or more of said pointer keys without simultaneous actuation of said special function key.

15. The system of claim 9, wherein said plurality of pointer keys of said keyboard include at least four arrow keys.

16. A method of operating a computer system comprising a central processing unit (CPU) having a plurality of inputs and configured to recognize a first combination of input signals as a keyboard interrupt and a second combination of input signals as a pointing-device interrupt, a keyboard having an array of binary keys including a special function key and a plurality of pointer keys, and a processor coupled between said CPU and said keyboard for detecting keystrokes on said keyboard and driving said inputs of said CPU, the method comprising the steps of:

detecting keystrokes on said keyboard;

responsive to actuation of said special function key, causing said keyboard to operate in a pointing device mode in which pointing information may be entered;

otherwise causing said keyboard to operate in a keyboard mode;

responsive to actuation of any of said binary keys while said keyboard is operating in said keyboard mode, driving said first combination of input signals to said CPU; and

responsive to actuation of said binary keys while said keyboard is operating in said pointing-device mode, driving said second combination of signals to said CPU, wherein said second combination of signals emulates a button of a pointing device and directional movement of a pointing device.

17. The method of claim 16 further comprising the steps of, responsive to a first predetermined sequence of keystrokes being entered via said keyboard, driving said second combination of input signals to said CPU until a second predetermined sequence of keystroke is entered via said keyboard.

18. A computer system, comprising:

a central processing unit (CPU) configured to recognize a plurality of input types, including keyboard input and pointing-device input;

a keyboard having an array of binary keys including a plurality of pointing keys, said keyboard operating in keyboard mode or pointing-device mode; and

a processor operatively connecting said keyboard to said CPU, whereby if said keyboard is operating in said keyboard mode, said processor drives keyboard input signals to said CPU, and if said keyboard is operating in said pointing-device mode, said processor drives pointing-device input signals to said CPU, whereby when said keyboard is in pointing-device mode, at least one of said binary keys emulates a button of a pointing device and a plurality of said pointing keys emulate directional movement of a pointing device.

19. The system of claim 18, wherein said keyboard includes a special function key, said keyboard operating in pointing-device mode after actuation of said special function key and one or more of said pointer keys.

20. The system of claim 18, wherein after actuation of a first sequence of keystrokes said keyboard operates in said pointing-device mode until actuation of a second sequence of keystrokes.

* * * * *