

República Federativa do Brasil
Ministério do Desenvolvimento, Indústria
e do Comércio Exterior
Instituto Nacional da Propriedade Industrial.

(21) **PI 0806143-2 A2**

(22) Data de Depósito: 27/06/2008
(43) Data da Publicação: 08/05/2012
(RPI 2157)



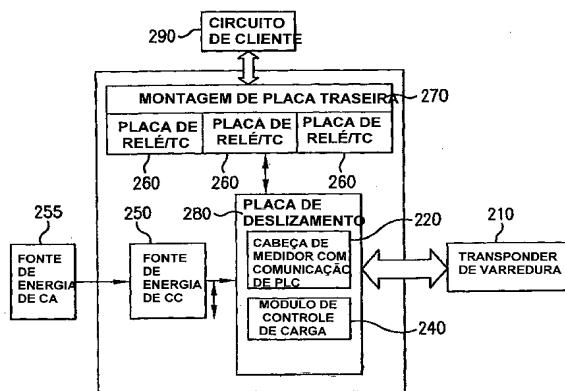
(51) *Int.Cl.:*
G01D 4/02
G01R 22/04
H04Q 9/02
H02J 13/00

(54) **Título:** APARELHOS PARA MEDIÇÃO DE MULTI-CANAL DE ELETRICIDADE

(73) **Titular(es):** Quadlogic Controls Corporation

(72) **Inventor(es):** Doron Shafrir, Robert Hayward, Sayre A. Swarztrauber, Siddharth Malik

(57) **Resumo:** APARELHOS PARA MEDIÇÃO DE MULTI-CANAL DE ELETRICIDADE. Em um aspecto, a invenção inclui um dispositivo para medir uso de eletricidade, incluindo: meio para desconexão remota por comunicação de linha de potência; meio para detecção de roubo de eletricidade; meio para detecção de fraude; e meio para detecção de tensão inversa. Em outro aspecto, a invenção inclui um aparelho para medição de multi-canal de eletricidade, incluindo: (a) uma cabeça de medidor operável para medir uso de eletricidade para uma pluralidade de linhas de consumidor de eletricidade; (b) um transponder operável para transmitir dados recebidos da cabeça de medidor por comunicação de linha de potência para um computador localizado remotamente, e para transmitir dados recebidos por comunicação de linha de potência do computador localizado remotamente para a cabeça de medidor; e (c) um módulo de controle de carga operável para atuar conexão e desconexão de cada um de uma pluralidade de relés, cada relé da pluralidade de relés correspondendo a uma da pluralidade de linhas de consumidor de eletricidade.



“APARELHOS PARA MEDIÇÃO DE MULTI-CANAL DE ELETRICIDADE”

Referência Cruzada a Pedidos Relacionados

Este pedido reivindica o benefício de Pedido de Patente Provisório US N° 60/737.580, depositado em 15 de novembro de 2005, Pedido de Patente Provisório US N° 60/739.375, depositado em 23 de novembro de 2005, e Pedido Provisório N° 60/813.901, depositado em 15 de junho de 2006, e é uma continuação em parte de Pedido de Patente US N° 11/431.849, depositado em 9 de maio de 2006, que é uma parte de Pedido de Patente US N° 11/030.417, depositado em 6 de janeiro de 2005 (agora Patente No. 7.054.770), que é uma parte de Pedido de Patente US N° 09/795.838, depositado em 28 de fevereiro de 2001 (agora Patente US N° 6.947.854). Os conteúdos inteiros de cada um desses pedidos estão incorporados aqui por referência.

Fundamento e Resumo

Uma concretização da presente invenção inclui um dispositivo de medição que está relacionado à família de medidores baseada em ASIC Quadlogic (veja Patente US N° 6.947.854, e Publicação de Pedido de Patente US N° 20060036388, os conteúdos inteiros de quais estão incorporados aqui por referência). Especificamente, esta concretização (referida aqui para conveniência como "Guarda de Energia") é um medidor de multi-canal que preferivelmente é capaz de prover muita da funcionalidade da família supracitada de medidores, e ademais provê as melhorias, características e componentes listados abaixo.

Usado em pelo menos uma concretização, um 'MiniCloset' é um dispositivo de medição de 24 canais que pode medir uso elétrico para até 24 clientes monofásicos, 12 clientes bifásicos, ou 8 clientes trifásicos. Preferivelmente conectado ao MiniCloset estão um ou mais Módulos de Controle de Carga (LCMs), discutidos abaixo.

Guarda de energia inclui preferivelmente um módulo de cabeça de medidor de MiniCloset e dois LCMs montados em uma caixa de aço. Relés que permitem para um cliente de eletricidade ser remotamente desconectado e reconectado, junto com transformadores de corrente, também
5 estão montados na caixa. Veja a Figura 1.

Na instalação, a linha de provisão de eletricidade de um cliente de eletricidade é derivada do alimentador elétrico principal, passado pelo aparelho de Guarda de Energia, e corre diretamente à casa do cliente. A construção e uso do Guarda de Energia serão aparentes àqueles qualificados
10 na arte na revisão da descrição abaixo e figuras relacionadas. Código de fonte é provido no Apêndice anexo.

Medidores de Guarda de Energia preferivelmente são operáveis para prover:

(A) Desconexão/Reconexão Remota: O medidor suporta
15 comunicação duplex total (bidirecional) por comunicação de linha de potência ("PLC") e pode ser equipado com relés operados remotamente (60 A, 100 A, ou 200 A) que permitem desconexão e reconexão de usuários elétricos remotamente.

(B) Prevenção de Roubo: O sistema é projetado com três
20 características específicas para prevenir roubo. Primeira, um aparelho de Guarda de Energia preferivelmente é instalado em um poste de utilidade acima das linhas de média tensão, fazendo difícil para clientes alcançarem e fraudarem. Segunda, porque não há nenhum fio de sinal adicional com o sistema (isto é, toda a comunicação é pela linha de potência), quaisquer fios
25 de comunicação cortados são imediatamente detectáveis. Quer dizer, se um fio de comunicação for cortado, o serviço é cortado, que é prontamente aparente. Uma terceira característica de prevenção de roubo é que o medidor pode ser usado para medir a energia de transformador a fim de validar os totais medidos de clientes individuais. Discrepâncias podem indicar roubo de

energia.

(C) Detecção de Fraude: O Guarda de Energia preferivelmente provê dois modos de detecção de fraude óptica. Cada unidade contém uma luz que reflete contra um pequeno adesivo como espelho. A ausência desta luz reflexiva indica que a caixa foi aberta. Esta detecção desconectará automaticamente todos os clientes medidos por aquela unidade de Guarda de Energia. Além disso, se o invólucro de Guarda de Energia for aberto e luz ambiente entrar, isto também desconectará automaticamente todos os clientes medidos por aquela unidade de Guarda de Energia. Estes dois modos de detecção de fraude são comprometidos continuamente e múltiplas vezes alternadas por segundo para segurança máxima.

(D) Detecção de Tensão Inversa: Em alguns casos, uma companhia de utilidade pode desconectar energia a um cliente individual e esse cliente é capaz de obter energia por uma alimentação alternativa. Se a utilidade fosse reconectar energia sob estas condições, dano poderia ocorrer ao equipamento de medição e/ou ao sistema de distribuição. Guarda de energia preferivelmente é capaz de detectar esta condição de falha. O Guarda de Energia pode detectar qualquer tensão que realimenta na desconexão aberta pelas linhas que conectam às instalações dos clientes. Se tensão for detectada, o firmware do Guarda de Energia prevenirá automaticamente a reconexão.

(E) Pré-pagamento: Pré-pagamento por energia pode ser feito por telefone, transação eletrônica, ou pessoalmente. A quantidade de kWh comprada é transmitida ao medidor e armazenada em sua memória. O medidor contará abaixo, mostrando quanta energia ainda está disponível antes de alcançar zero e desconectar. Contanto que o cliente continue comprando energia, não haverá nenhuma interrupção no serviço, e a companhia de utilidade terá um relatório de atividade diário.

(F) Limitação de Carga: Como uma alternativa à desconexão

por não pagamento ou parte de um sistema de pré-pagamento, medidores de Guarda de Energia podem permitir a utilidade limitar remotamente a potência entregue a um nível fixo, desconectando quando essa carga é excedida. Se o cliente exceder essa carga e for desconectado, o cliente pode reajustar um botão na unidade de exibição remota opcional para restabelecer carga contanto que a carga conectada seja menos do que o limite prefixado. Alternativamente, clientes podem chamar uma linha de serviço de utilidade elétrica por telefone para ter o serviço restabelecido. Esta característica permite a utilidades elétricas proverem eletricidade para sistemas críticos até mesmo, por exemplo, no caso de um cliente não pagante.

(G) Limitação de Consumo Mensal: Alguns clientes se beneficiam de taxas subsidiadas e são dados um consumo total máximo por mês. O firmware de Guarda de Energia é capaz de interromper energia quando um certo nível de consumo é alcançado. Porém, este tipo de programa é implementado melhor quando notificação avançada para clientes é provida. Isto pode ser alcançado tanto com uma exibição na casa, por meio de que uma mensagem ou série de mensagens notifica os clientes que sua taxa de consumo está chegando ao consumo projetado para o mês. Alternativamente (ou em conjunto), interrupções de serviço temporizadas podem ser programadas de forma que quando o limite está se aproximando, energia seja desconectada para períodos de tempo com incrementos cada vez mais longos para notificar os residentes. Estas interrupções planejadas em serviço atuam como uma advertência para clientes que seu limite está se aproximando de forma que eles tenham tempo para alterar seus padrões de consumo.

(H) Validação de Medidor: O módulo integrado do sistema preferivelmente é removível. Isto permite re-validação de laboratório fácil de precisão de medidor no evento de disputas de fatura de cliente.

(I) Benefícios Operacionais para Utilidade: O Guarda de Energia tem registros de evento e funções de diagnóstico a bordo, provendo

os técnicos de campo com uma riqueza de dados para comissionar e reparar os sistemas elétrico e de comunicação. Parâmetros de não faturamento incluem: ampères, volts, temperatura, distorção harmônica total, frequência, valores instantâneos de Watts, VARs e volt-ampères, V2 horas, I2 horas, fator de potência, e ângulo de fase.

Estas características e outras serão aparentes àqueles qualificados na arte depois de revisar as descrições anexas, código de software e esquemas.

Em um aspecto, a invenção inclui um dispositivo para medir uso de eletricidade, incluindo: meio para desconexão remota por comunicação de linha de potência; meio para detecção de roubo de eletricidade; meio para detecção de fraude; e meio para detecção de tensão inversa.

Em outro aspecto, a invenção inclui um aparelho para medição de multi-canal de eletricidade, incluindo: (a) uma cabeça de medidor operável para medir uso de eletricidade para uma pluralidade de linhas de consumidor de eletricidade; (b) um transponder em comunicação com a cabeça de medidor e operável para transmitir dados recebidos da cabeça de medidor por comunicação de linha de potência a um computador localizado remotamente, e para transmitir dados recebidos por comunicação de linha de potência do computador localizado remotamente à cabeça de medidor; e (c) um módulo de controle de carga em comunicação com a cabeça de medidor e operável para atuar conexão e desconexão de cada um de uma pluralidade de relés, cada relé da pluralidade de relés correspondendo a uma da pluralidade de linhas de consumidor de eletricidade.

Em várias concretizações: (1) o aparelho ademais inclui um detector de fraude em comunicação com a cabeça de medidor; (2) o detector de fraude inclui uma luz e uma superfície reflexiva, e a cabeça de medidor é operável para instruir o módulo de controle de carga para desconectar todas as linhas de cliente se o detector de fraude prover notificação que a luz não é

detectada refletindo da superfície reflexiva; (3) o aparelho ademais inclui uma caixa contendo a cabeça de medidor, o módulo de controle de carga e os relés, e em que o detector de fraude inclui um detector de luz ambiente entrando na caixa; (4) o aparelho ademais inclui uma caixa contendo a cabeça de medidor, o módulo de controle de carga e os relés, e em que a caixa é instalada em um poste de utilidade; (5) o aparelho ademais inclui meio para comparar energia de transformador à energia total usada pelas linhas de consumidor; (6) o aparelho ademais inclui meio para detectar fluxo de tensão inversa pelas linhas de consumidor; (7) o aparelho ademais inclui uma memória legível por computador em comunicação com a cabeça de medidor e um contador em comunicação com a cabeça de medidor, o contador correspondendo a uma linha de cliente e operável para contar abaixo uma quantidade de energia armazenada na memória, e a cabeça de medidor operável para enviar um sinal de desconexão ao módulo de controle de carga para desconectar a linha de cliente quando o contador alcança zero; (8) o aparelho ademais inclui uma memória legível por computador em comunicação com a cabeça de medidor, a memória operável para armazenar um limite de carga para uma linha de cliente, e a cabeça de medidor operável para enviar um sinal de desconexão ao módulo de controle de carga para desconectar a linha de cliente quando o limite de carga é excedido; (9) o aparelho ademais inclui uma memória legível por computador em comunicação com a cabeça de medidor, a memória operável para armazenar um limite de uso para uma linha de cliente, e a cabeça de medidor operável para enviar um sinal de desconexão ao módulo de controle de carga para desconectar a linha de cliente quando o limite de uso é excedido; (10) o transponder é operável para se comunicar com o computador localizado remotamente através de linhas de potência de média tensão; (11) o aparelho ademais inclui uma unidade de exibição em comunicação com a cabeça de medidor e operável para exibir dados recebidos da cabeça de medidor; (12) a unidade de exibição é operável para exibir

informação relativa a consumo de energia de um cliente; (13) a unidade de exibição é operável para exibir advertências relativas a uso de energia de um cliente ou roubo de energia suspeito; e (14) a unidade de exibição é operável para transmitir para dita cabeça de medidor informação entrada por um usuário.

Breve Descrição dos Desenhos

Figura 1 é um diagrama de bloco/fiação mostrando a conexão de concretizações preferidas.

Figura 2 é um diagrama de bloco mostrando a configuração física de concretizações preferidas.

Figuras 3A-3B são diagramas esquemáticos de uma placa de CPU preferida de um Transponder de Varredura e MiniCloset.

Figura 4 é um diagrama esquemático de uma fonte de energia de Transponder de Varredura preferida.

Figura 5 é um diagrama esquemático de uma fonte de energia de MiniCloset preferida.

Figura 6 é um diagrama esquemático de uma placa de circuito preferida para retornar informação de transformador de corrente a uma cabeça de medidor de MiniCloset.

Figuras 7A-7C são diagramas esquemáticos de uma placa de circuito de Módulo de Controle de Carga preferida.

Figuras 8A-8D são diagramas esquemáticos de uma placa de fonte de energia preferida que provê detecção de fraude óptica.

Figuras 9A-9C são diagramas esquemáticos de uma placa de conexão de Guarda de Energia preferida.

Figura 10 é um diagrama esquemático para uma placa de circuito de controle operável para prover controle de relé.

Figura 11 é um diagrama de montagem de base de Guarda de Energia preferida.

Figuras 12 e 13 são diagramas de barramentos de fase preferidos e construção dos mesmos.

Figura 14 é um diagrama descrevendo construção e montagem de estrutura de barra de neutro preferida.

5 Figura 15 descreve barras de transição preferidas;

Figura 16 descreve colocação preferida de barras de transição.

Figuras 17 e 18 descrevem construção de módulo de aceitante preferido.

10 Figura 19 descreve um módulo sensor de corrente e relé integrado preferido.

Figura 20 descreve uma vista explodida de um módulo sensor de corrente e relé integrado preferido.

Figura 21 mostra vistas explodidas de módulos de medição preferidos.

15 Figura 22 mostra os módulos de medição colocados em uma montagem de estrutura de EG e módulo de aceitante.

Figura 23 mostra uma vista explodida de uma concretização preferida de Guarda de Energia.

20 Figura 24 mostra uma vista explodida de uma montagem de EG preferida e montagem de base.

Figura 25 mostra uma disposição de EG preferida.

Figuras 26 e 27 são esquemas de módulo de medição preferidos.

25 Figura 28 tem esquemas preferidos para uma placa de lugar traseiro.

Figura 29 tem esquemas preferidos para uma placa de potência.

Figura 30 tem esquemas preferidos para uma placa de extensão de I/O.

Figura 31 tem esquemas preferidos para uma placa de CPU.

Figura 32 tem esquemas preferidos para um módulo de controle.

Figura 33 tem esquemas preferidos para circuitos de medição e fonte de energia para um módulo de exibição de cliente;

Figura 34 tem esquemas preferidos para uma placa de exibição para o CDM.

Figura 35 é um diagrama de bloco de uma extremidade dianteira analógica preferida para medição.

Figuras 36 e 37 descrevem implementações de DSP preferidas.

Figura 38 ilustra características de resposta a impulso e frequência de filtro em fase preferidas.

Figura 39 ilustra injetar sinais de PLC a harmônicos meio-ímpares de 60 Hz.

Figura 40 descreve 12 possíveis modos nos quais um quadro de FFT recebido por um medidor pode estar fora de fase com um quadro de FFT transmitido por transponder de varredura.

Figura 41 ilustra especificações de filtro de FIR preferidas.

Figura 42 descreve tensão e corrente resultando de uma FFT preferida.

Descrição Detalhada de Concretizações Preferidas

Em uma concretização, um aparelho de medição de Guarda de Energia inclui um MiniCloset (quer dizer, um aparelho de medição operável para medir uma pluralidade de linhas de cliente); um Transponder de Varredura; um ou mais relés operáveis para desconectar serviço para clientes selecionados; um Módulo de Controle de Carga; e meio de detecção de fraude óptica.

O MiniCloset e Transponder de Varredura referidos aqui são em grande parte os mesmos como descritos na Patente US N° 6.947.854.

Quer dizer, embora cada um tenha sido melhorado através dos anos, a funcionalidade e estrutura pertinentes a esta descrição podem ser tomadas para serem as mesmas como descrito nessa patente.

Um aspecto da invenção inclui tomar a funcionalidade de
5 medição de multicanal existente achada no MiniCloset e adicionar conexão e desconexão remotas por PLC. Prover tal funcionalidade adicional requerida adicionando novo hardware e software. O hardware adicionado inclui um Módulo de Controle de Carga (LCM) e relés de conexão/desconexão. Também adicionado eram circuitos de suporte para dirigir rastros de sinal
10 para e do processador de medidor principal - a Cabeça de Medidor de MiniCloset. As adições de software incluem módulos de código que se comunicam com o hardware adicionado, como descrito nas tabelas abaixo. Figura 1 é um diagrama de bloco de conexões de uma concretização preferida. Linhas de potência de média tensão A, B, C e N (neutro) alimentam
15 em Transformador de Distribuição 110. Linhas de baixa tensão conectam (por transformadores de corrente 120) o Transformador de Distribuição 110 à unidade de Guarda de Energia 140. A unidade de Guarda de energia 140 monitora os transformadores de corrente 120, e alimenta linhas de cliente monofásicas 1-24.

20 Figura 2 é um diagrama de bloco de estrutura preferida de uma unidade de Guarda de Energia 140.

Transponder de Varredura 210 é o coletor de dados preferido para a unidade 140, pode estar localizado externo ou dentro do MiniCloset, e pode ser o coletor de dados principal para mais de um MiniCloset de cada
25 vez. O Transponder de Varredura 210 preferivelmente: (a) verifica dados (cada comunicação preferivelmente começa com a verificação de relógio e identidade de medidor para assegurar integridade de dados); (b) coleta dados (coleta periodicamente um bloco de dados de cada unidade de medidor, com cada bloco contendo leituras de medidor previamente coletadas, leituras de

intervalo, e registros de evento); (c) armazena dados (preferivelmente os dados são armazenados em memória não volátil por um período especificado (por exemplo, 40 dias)); e (d) relata dados (tanto por PLC, modem de telefone, conexão de RS-232, ou outro meio).

5 A placa de deslizamento 280 inclui uma cabeça de medidor de Minicloset e um módulo de controle de carga 240, que provê os sinais de controle para ativar os relés. Toda a eletrônica preferivelmente é energizada por fonte de energia 250. A montagem de placa traseira 270 inclui múltiplos (por exemplo, 24) Transformadores de corrente e relés - agrupados, neste
10 exemplo, como três conjuntos de 8 TCs e relés. Cabos de cliente são conectados por fios pelos TCs e se conectam ao circuito nas instalações de cliente 290. O Transponder de Varredura remotamente localizado 210 acessa a cabeça de medidor de Guarda de Energia e se comunica de modo bidirecional usando comunicação de portadora de linha de potência.

15 O fluxo de sinal mostrado nas Figuras 1 e 2 é preferivelmente realizado implementando módulos de código de software diferentes que trabalham simultaneamente para habilitar a habilidade de conexão/desconexão remota no Minicloset. Estes módulos de software, providos no Apêndice abaixo, são:

Módulo de Código	Localização	Função
lcm.def e pic.def	Módulo de Controle de Carga	Atuar conexão e desconexão de relés
pulse.c e pulse.h	Cabeça de Medidor	Estabelecer Comunicação com LCM
picend.def e picvars.def	Cabeça de Medidor	Prover sinais de controle para LCM
pulselink.def e pulseoutm.c	Cabeça de Medidor	Provê LCM com pulsos a serem usados para conectar e desconectar relés

20 Figuras 3-10 são esquemas de componentes preferidos, como descrito abaixo. Os relés de conexão/desconexão preferidos são relés de série K850 KG, mas aqueles qualificados na arte reconhecerão que outros relés podem ser usados sem partir da extensão da invenção.

Figura	esquema	Detalhe

3	PCB 107D	Placa de CPU do Transponder de Varredura e MiniCloset
4	PCB 135C	Fonte de energia para Transponder de Varredura
5	PCB 144C	Fonte de energia para MiniCloset
6	PCB 146C	Esta placa trás de volta informação de Transformador de Corrente para cabeça de medidor de MiniCloset
7	PCB 160A	Placa para Módulo de Controle de Carga
8	PCB 170	Placa de fonte de energia de EG que adiciona capacidade para detecção de fraude óptica
9	PCB 171	Placa de conexão de EG. Uma placa com trilhas para rotear o sinal
10	PCB 172	Placa de circuitos de controle para controle de relé

Em outra concretização, a implementação de Guarda de Energia tira vantagem da semelhança de arquitetura de painéis de disjuntor tradicionais, com o ambiente de medição de multicanal. Em um painel de disjuntor, eletricidade é alimentada ao painel e distribuída entre vários circuitos de cliente por disjuntores que provêem a habilidade para conectar ou desconectar os circuitos de cliente.

No MiniCloset/Guarda de Energia, múltiplos transformadores de corrente medem a corrente em circuitos de cliente e trazem estes dados de volta a uma unidade de processamento central, onde as quantidades de medição são calculadas. Porém, o MiniCloset/Guarda de Energia tem várias diferenças principais com um painel de disjuntor. Por exemplo, enquanto os disjuntores são achados perto de instalações de cliente, o Guarda de Energia tipicamente é instalado perto do transformador de distribuição de utilidade. As vantagens oferecidas por esta concretização alternada serão aparentes àqueles qualificados na arte. Por exemplo, esta concretização oferece dimensões melhoradas e tamanho global sobre as concretizações discutidas acima. Espaço é sempre um constrangimento quando adições de equipamento são feitas a instalações elétricas existentes. Esta versão do Guarda de Energia ("EG"), com dimensões preferidas de 71,12 cm X 55,88 cm X 27,94 cm provê uma vantagem significativa em situações onde constrangimentos volumétricos existem.

A descrição seguinte inclui detalhes de construção preferidos, esquemas detalhados, e descrições de software. Como com as concretizações

discutidas acima, esta concretização é operável para prover operações de desconexão/conexão remotas, prevenir roubo, detectar fraude, detectar tensão inversa, executar pré-pagamento e limitar carga, e executar validação de medidor.

5 **Detalhes de Construção de EG Preferidos**

Nesta concretização, componentes primários do EG são:

1. Montagem de Base de Guarda de Energia
2. Montagem de Guarda de Energia
 - a. Barramentos de Fase e Barramentos de Neutro
 - b. Barras de transição
 - c. Módulo de Aceitante
3. Módulos de Medição de Guarda de Energia
 - a. Módulos de Medição
 - i. Módulos de Sensor de Corrente e Relé Integrados
4. Eletrônica de Guarda de Energia
 - a. PCB 203
 - b. PCB 204
 - c. PCB 234
 - d. PCB 235
 - e. PCB 202
 - f. PCB 210
 - g. PCB 230
 - h. PCB 206

Base de Montagem de EG

- 25 A base de EG inclui um fundo de invólucro com parafusos e arruelas de retenção como um mecanismo de travamento para a cobertura de topo de EG, que está conectada em um lado por dobradiças. Veja Figura 11. O fundo de invólucro provê roteamento para os cabos de cliente.

Montagem de EG - Barramentos de Fase

Três barramentos de fase de alumínio são colocados para o centro da montagem de Guarda de Energia e alternados. Veja Figuras 12 e 13. Estes provêm conexão aos módulos de medição de cliente pelo uso de barras de transição. Uma disposição de barramento alternada é descrita na Figura 13.

5 Barramentos são mostrados em preto.

Barras de Neutro

O EG preferivelmente inclui 4 barras de neutro que formam um quadro para montagem de EG, por esse meio provendo um caminho para a corrente de neutro. Isto é mostrado na Figura 14. O borne na barra cruzada
10 provê a alimentação de neutro do transformador de distribuição de utilidade. Também, há 2 barras de neutro de placa-mãe que levam a corrente de neutro ao módulo de controle.

Barras de Transição

As barras de transição completam a conexão mecânica e
15 elétrica entre os módulos de medição de cliente e os barramentos de fase. Veja a Figura 15. Uma barra de transição para fase A e C é mostrada na Figura 15A; uma barra de transição para fase B é mostrada na Figura 15B. Figura 16 mostra as barras de transição em preto.

Módulo de Aceitante

Um módulo de aceitante preferivelmente é feito de plástico e
20 aceita mecanicamente os módulos de medição que podem ser encaixados facilmente na montagem de EG. Cada EG tem 4 módulos de aceitante que são empilhados juntos e podem acomodar tanto 12 módulos de medição bifásicos ou 8 trifásicos. Veja Figura 17. O módulo de aceitante também provê uma
25 rota mecânica para a barra neutra de placa-mãe que conecta ao módulo de controle. Veja Figura 18.

Módulos de Medição de Cliente

Módulos de medição de cliente preferidos provêm a metrologia requerida para medir o consumo para um cliente monofásico,

bifásico ou trifásico. Um módulo individual funciona como um medidor independente completo que pode ser testado e avaliado como uma unidade de medição separada. Cada módulo preferivelmente inclui um módulo de sensor de corrente e relé integrado e eletrônica de metrologia, e provê uma conexão entre o circuito de cliente e os barramentos de fase. Figura 19 descreve um módulo de sensor de corrente e relé integrado preferido. Figura 20 descreve uma vista explodida de um módulo de sensor de corrente e relé integrado preferido.

Figura 21 mostra vistas explodidas de módulos de medição preferidos. Figura 22 mostra os módulos de medição (mostrados em preto) colocados na montagem de quadro de EG e módulo de aceitante.

Figura 23 mostra uma vista explodida de Guarda de Energia, e Figura 24 mostra uma vista explodida de uma Montagem de EG e Montagem de Base de EG preferidas.

Eletrônica

As caixas de Módulo de Controle preferivelmente incluem várias PCBs que funcionam simultaneamente para coletar dados de medição dos módulos de medição individuais e comunicar através de linhas de potência para transmitir estes dados a um dispositivo mestre, tal como um Transponder de Varredura ("ST").

Figura 25 mostra uma disposição de Guarda de Energia preferida para esta concretização. Cada linha de cliente tem um Módulo de Medição correspondente (PCB 203 e PCB 204, discutido abaixo) (esquemas mostrados nas Figuras 26 e 27).

Uma Placa de Colocação Atrás 2510 mostrada na Figura 25 (PCB 234; veja Figura 28 para diagrama de construção e esquemático) é o barramento comum que dirige sinais dentro do EG. Há dois tipos de opções de comunicação na Placa de Colocação Atrás 2510 para habilitar transferência de dados de Módulo de Controle 2520 para Módulos de Medição individuais

PCB 203. Isto pode ser feito tanto usando a opção de I2C de 2 fios ou a opção serial de 1 fio.

O Módulo de Controle 2520 inclui uma Placa de Potência (PCB 210; veja Figura 29 para esquemático) é a placa de fonte de energia que também tem os circuitos de transmissão e recepção de PLC nela. A Placa de Potência provê energia à placa de CPU e às placas de eletrônica 203. O Módulo de Controle 2520 também inclui uma Placa de Extensão de I/O (PCB 230; veja Figura 30 para esquemático) é uma placa com várias opções de extensão de I/O que habilitam comunicação de Módulos de Medição à placa de CPU.

Módulo de Controle 2520 também inclui uma Placa de CPU (PCB 202; veja Figura 31 para esquemático), que tem um processador de Processamento de Sinal Digital (DSP) a bordo.

Finalmente, Módulo de Controle 2520 inclui uma placa de roteamento (PCB 235; veja Figura 32 para esquemático) com trilhas e um coletor sem componentes eletrônicos nele.

Cada Módulo de Exibição de Cliente (CDM) 2530 é instalado nas instalações do cliente e pode se comunicar de modo bidirecional com o EG instalado no transformador de distribuição servindo o cliente. PLC bidirecional habilita comunicação de utilidade-cliente através de linhas de potência de baixa tensão e permite a utilidade enviar informação regular, advertências, informação especial sobre interrupções, etc., para o cliente. Cada CDM 2530 inclui uma combinação selecionada de medição e fonte de energia junto com circuitos de PLC na mesma placa (PCB 240; veja Figura 33 para esquemático). Cada CDM preferivelmente também tem uma placa de exibição de 9 dígitos (PCB 220; veja Figura 34 para esquemático). Esta exibição se comunica com EG e mostra informação sobre consumo, precauções, advertências e outras mensagens de utilidade.

Implementação de Hardware

Em uma concretização, o Guarda de Energia implementa Transformada de Fourier Rápida (FFT) no sinal de comunicação de PLC ambos no ST e no medidor, e para propósitos de medição executa análise de harmônico detalhada. Esta seção discute um esquema de implementação dos

5 Módulos de Medição, comunicação com Módulos de Controle e comunicação de PLC do Módulo de Controle com um Transponder de Varredura remotamente localizado.

O Módulo de Controle 2520 inclui fonte de energia e circuitos de PLC (PCB 210; veja Figuras 25 e 29); extensão de I/O (PCB 230; veja

10 Figura 30) e placa de CPU nomeada Medidor D (PCB 202; veja Figura 31). A fonte de energia provê energia ao medidor D e extensão de I/O e contém os circuitos de transmissor e receptor de PLC. PCB 235 provê um roteamento de trilha e conexão de coletor entre várias placas.

O Módulo de Medição pode ter duas versões: bifásica ou

15 trifásica. A versão bifásica pode ser programada por software para funcionar como um único medidor bifásico ou dois medidores monofásicos. A versão bifásica inclui um medidor B2 (PCB 203, esquemático mostrado na Figura 26), enquanto a versão trifásica inclui um medidor B3 (PCB 204 esquemático mostrado na Figura 27). Os medidores B atuam como escravos para o

20 medidor D no Módulo de Controle 2520. Os medidores D e B podem se comunicar por um protocolo de ASCII serial.

Os vários medidores B são interconectados por BPB 2510 a 2520, que provê energia, uma referência de 1 Hz e comunicações seriais para o medidor D. A máquina de DSP preferida para o medidor B é o chip

25 56F8014VFAE Freescale. O microprocessador preferido usado para implementar a CPU no medidor D é o entre a família de ColdFire Integrated Microprocessadores, MCF5207. O uso de um processador específico é determinado por requisitos de RAM e Flash ditados pela versão de medidor. Uma fonte de energia separada e placa de LCD completam a porção

eletrônica do medidor D como um produto. À parte de atuar como um mestre para medidores B, o medidor D também é um medidor trifásico e mede a saída de transformador total em que o EG está instalado. Como uma característica anti-roubo, este total é comparado com o consumo total relatado pelos vários medidores B.

$$\sum_{n=1}^{24} kWh_n = \text{Saída de Transformador Total}$$

A constituição de fluxos de sinal é como segue:

B2: Duas tensões, Duas correntes e Nenhum Canal de Portadora de Linha de Potência (PLC).

B3: Três tensões, Três correntes e Nenhum Canal de PLC.

D: Três tensões, Três correntes e um Canal de PLC.

Cada fluxo tem um circuito associado para efetuar amplificação analógica e 'antialiasing'.

Específico para o medidor D é a implementação preferida de:

- Uma Malha Travada por Fase (PLL) para travar a amostragem dos fluxos de sinal a um múltiplo da linha de A/C entrante (amostragem síncrona para a linha de potência).

- Um Oscilador Controlado por Tensão (VCO) a 90-100 MHz controlado por processador de DSP por dois módulos de PWM acionando diretamente o relógio de sistema, conseqüentemente fazendo o DSP coerente com o PLL.

- Um detector de fase síncrono que só responde ao fundamental da onda de frequência de linha entrante e não a seus harmônicos.

- Opção para executar esquemas de modulação de FSK e PSK.

Cada canal de medição e comunicação preferivelmente inclui circuitos analógicos de extremidade dianteira seguidos pelo processamento de sinal. Único aos circuitos analógicos é um filtro 'anti-aliasing' com ganho fixo que provê rastreamento de temperatura de primeira ordem, conseqüentemente

eliminando a necessidade para recalibrar medidores quando derivas de temperatura são encontradas. Isto é discutido a seguir, e então uma implementação de processamento de sinal preferida é discutida.

Cadeia de Sinal Analógico de Tensão e Corrente

5 A extremidade dianteira analógica para canais de tensão (corrente) inclui elementos sensores de tensão (corrente) e um atenuador programável, seguido por um filtro 'antialiasing'. O atenuador reduz o nível de sinal entrante de forma que nenhum corte ocorra depois do filtro 'anti-aliasing'. O filtro 'anti-aliasing' de ganho constante restabelece o sinal a valor
10 total na entrada do Conversor Analógico para Digital (ADC). Para medição, o filtro 'anti-aliasing' corta fora frequências acima de 5 kHz. As entradas são então alimentadas no ADC que faz parte do DSP. Veja Figura 35, que é um diagrama de bloco de uma extremidade dianteira analógica preferida para medição.

15 Enquanto uma implementação típica incluiria um Amplificador de Ganho Programável (PGA) seguido por um filtro 'anti-aliasing' de baixo ganho, a invenção, nesta concretização, implementa um atenuador programável seguido por um filtro de ganho fixo grande. Além disso, a implementação de ambos os filtros 'anti-aliasing' em um único chip é
20 a mesma usando os mesmos Amplificadores Operacionais Quádruplos junto com resistores de 25 ppm e capacitores de NPO/COG. Esta implementação única casando os filtros 'anti-aliasing' assegura que as derivas de fase encontradas em ambos os canais de tensão e corrente sejam exatamente idênticas e conseqüentemente precisão do cálculo de potência (dado pelo
25 produto de V e I) não é comprometido. Isto provê um meio para ambos canais de V e I rastreamos derivas de temperatura até primeira ordem sem recalibrar o medidor.

Em contraste, usando um PGA junto com um filtro de ganho baixo não pode rastrear o deslocamento de fase nos sinais V e I introduzidos

devido à temperatura. Isto é porque o deslocamento de fase introduzido por PGA é uma função do ganho.

Cadeia de Sinal Digital de Tensão, Corrente e PLC

Figura 36 é um diagrama de bloco da placa de PCB 202; as funções de cada bloco serão aparentes àqueles qualificados na arte. Figura 36 mostra uma implementação de DSP preferida.

Esta concretização preferivelmente usa uma PLL para travar a amostragem dos fluxos de sinal a um múltiplo da frequência de linha de A/C entrante. Na concretização discutida acima, a amostragem é a uma taxa assíncrona à linha de potência. No medidor D, há um VCO a 90-100 MHz que é controlado pela máquina de DSP por dois módulos de PWM. O VCO excita diretamente o relógio de sistema do chip de DSP (desabilitando a PLL interna), assim o DSP se torna uma parte integrante da PLL. Travar o relógio de sistema do DSP à linha de potência facilita o alinhamento da amostragem para a forma de onda da linha de potência. O detector de fase deveria funcionar assim para só responder ao fundamental da onda de 60Hz entrante e não a seus harmônicos. Figura 37 é um diagrama de bloco desta implementação de DSP preferida.

Um DSP BIOS ou código de comutação de contexto voluntário provê três pilhas, cada uma para fundo, comunicações de PLC e comunicações seriais. O pequeno micro se comunica com o DSP usando um excitador I2C. O circuito integrado MSP430F2002 mede as fontes de energia, porta de fraude, temperatura e tensão de bateria. As tarefas do MSP430F2002 incluem:

- i. manter um RTC;
- ii. medir a tensão de bateria;
- iii. medir a temperatura;
- iv. medir a fonte de energia +U;
- v. reajustar o DSP na interrupção parcial;

vi. prover um circuito de vigilância adicional; e

vii. prover uma referência de 1 segundo para entrar no DSP para uma referência de tempo para medir a referência de 1 segundo contra o relógio de sistema do VCO.

5 Cadeia de Sinal de Comunicação de PLC de Medidor D

Uma instalação típica consiste em múltiplos EGs e STs se comunicando através das linhas de potência. O medidor D se comunica de modo bidirecional com um Transponder de Varredura localizado remotamente pelo transformador de distribuição. Para habilitar isto, esta

10 concretização usa uma banda de 10-25 kHz para comunicação de PLC. O sinal de PLC é amostrado a cerca de 240 kHz ($212 * 60$), síncrono com tensão de linha, seguindo o que um filtro de Resposta a Impulso Finita (FIR) é aplicado para dizimar os dados. Especificações de FIR preferidas são dadas abaixo: Banda de 10-25 kHz

Número de Derivações	65
Atenuação de Banda de Parada	71,23 dB
Frequência Superior de Banda Passante	25 kHz
Frequência Inferior de Banda de Parada	35 kHz
Entrada Amostrada	$60 * 4096$
Saída Mostrada	$30 * 2048$

15 Veja Figura 38 para características preferidas de resposta de frequência de filtro em fase e resposta a impulso.

Depois que a dizimação é feita a 60 kHz ($2^{11} * 30$), uma FFT de 2048 pontos é então executada nos dados dizimados. A taxa de dados é assim determinada ser 30 baud dependendo da escolha de filtros de FIR. Toda

20 FFT produz dois bits aproximadamente a todo 66 ms ao usar FIR na banda de 10-25 kHz para se comunicar por transformadores de distribuição.

Para evitar o problema de se comunicar na presença de ruído de linha, esta concretização preferivelmente implementa uma técnica única

para comunicação robusta e segura. Isto é feito injetando sinais de PLC a frequências que são harmônicos meio-ímpares da frequência de linha (60Hz). Isto é discutido abaixo, para uma concretização usando um espectro de ruído típico achado em linhas de CA na gama de 12-12,2 kHz. Figura 39 ilustra

5 injetar sinais de PLC a harmônicos meio-ímpares de 60 Hz. Desde que FFT é feita a todo 30 Hz e os harmônicos são separados por 60 Hz, os bits de dados residem na caixa correspondendo ao 201,5° e 202,5° harmônico de 60 Hz na Figura 39. O algoritmo considera estas duas caixas de frequências e compara a amplitude do sinal nos dois para determinar 1 ou 0. Este esquema de FSK

10 usa duas frequências e produz uma taxa de dados de 30 baud. Alternativamente, QFSK, que usa 4 frequências, pode ser implementado para produzir 60 baud.

Ao atravessar por transformadores, preferivelmente ambos STs e medidores D executam FFT no PLC e sinais de dados a todo 30 Hz em uma

15 gama de 10-25 kHz. Porque as Malhas Travadas em Fase (PLLs) implementadas ambos no ST e no medidor D estão travadas à linha, os quadros de dados estão sincronizados à frequência de linha (60 Hz) igualmente. Porém, os quadros de dados podem se deslocar em fase devido a:

1. várias configurações de transformador que podem existir no
- 20 caminho entre o ST e medidor (delta -ípsilon, etc.); e
2. um deslocamento em fase devido ao fato que STs estão travados em uma fase particular, enquanto medidores mono e polifásicos podem ser energizados por outras fases.

A relação de sinal para ruído (SNR) é maximizada quando o

25 quadro de dados de medidor e quadros de dados de ST estão alinhados perto de perfeição. Do ponto de vista de um medidor, isto requer receber sinal de PLC de todos os possíveis STs que pode "ouvir", decodificar o sinal, verificar por SNR alinhando quadros de dados, e então responder ao ST que está produzindo SNR máxima. Figura 40 descreve os 12 possíveis modos nos

quais o quadro de FFT recebido pelo medidor pode estar fora de fase com quadro de FFT de ST. Linhas pontilhadas correspondem a uma rotação de 30 graus considerando um transformador delta no caminho de sinal entre ST e o medidor.

5 Além disso, porque os quadros de dados estão disponíveis a todo 30 Hz em uma linha de 60 Hz, há duas possibilidades correspondendo às 2 possíveis fases obtidas dividindo 60 Hz por 2. Conseqüentemente, há 24 modos que os quadros de dados de medidor podem estar desalinhados com quadros de dados de ST.

10 Em cada quadro do ST, há um número integral ímpar de ciclos da frequência de portadora. Desde que o esquema de modulação preferido é Chaveamento por Deslocamento de Frequência (FSK), se houver n ciclos para transmitir bit 1, bit 0 é transmitido usando $n+2$ ciclos da frequência de portadora. Fica vital para o medidor reconhecer seus próprios 2 ciclos de 60
15 Hz a fim de ser capaz de decodificar seus bits de dados que estão disponíveis a todo 1/30 de um segundo.

 Se o medidor D decodificar sinais com quadros de dados desalinhados, há energia que transborda nas frequências adjacentes (meio-ímpares separadas). Se o nível de sinal que cai na caixa de frequência
20 "adjacente" for menos do que o fundo de ruído, o sinal pode ser decodificado corretamente. Porém, se o derramamento for mais do que o fundo de ruído, a habilidade para distinguir entre 1 e 0 diminui, e conseqüentemente a SNR global cai, resultando em um erro em decodificação. Em conclusão:

 a. Se os quadros estiverem desalinhados, contaminação de bits
25 de dados ocorre e a SNR degrada.

 b. No evento que a frequência muda e há quadros de dados desalinhados, há uma quantia significativa de energia que transborda nas caixas de FFT adjacentes, conseqüentemente interferindo com os outros STs no sistema que se comunicam usando frequências nessas caixas específicas.

Uma vez que o deslocamento de relógio seja determinado correspondendo à SNR mais alta, o medidor então se trava até que uma mudança significativa em relação de SNR seja encontrada pelo medidor, em qual caso o processo repete.

5 Implementação de Medição em Medidor D e B usando FFT

10 Enquanto versões do medidor B e do medidor D executam medição, o medidor D também é responsável por coletar a informação de medição dos vários medidores B por PCB 234. Cada fluxo de dados nos medidores tem um circuito associado para efetuar amplificação analógica e 'anti-aliasing'. Cada uma das seções de extremidade dianteira analógica tem um atenuador programável que é controlado pelo código de nível mais alto. O fluxo de dados é amostrado a 60 kHz ($2^{10} * 60$) e então um filtro de FIR é aplicado para dizimar o fluxo de dados a ~ 15 kHz ($2^8 * 60$). Especificações de filtro preferidas são mostradas na tabela abaixo e Figura 41.

Número de Derivações	29
Atenuação de Banda de Parada	80,453dB
Frequência Superior de Banda Passante	3kHz
Frequência Inferior de Banda de Parada	12kHz

15 Desde que só os dados até 3 kHz são de interesse, preferivelmente uma faixa de 3-12 kHz no FIR de dizimação é usada com taxa de amostra de ~ 15 KHZ. As frequências de 0-3 kHz ou 12-15 kHz são mapeadas em 0-3 kHz. Uma FFT real é executada para produzir 2 fluxos de dados que podem ser decompostos ademais em 4 fluxos de dados: Tensão
20 Real e Imaginária e Corrente Real e Imaginária. Isto é alcançado adicionando e subtraindo frequências de espelho positivas e negativas para as partes real e imaginária, respectivamente. Desde que o sinal 'aliased' na gama de 12-15 kHz cai abaixo de 80 dB, a precisão é alcançada usando o filtro de FIR acima discutido. Alternativamente, uma FFT complexa de 256 pontos pode ser
25 executada em toda fase do fluxo de dados dizimado. Isto produz 2 pares de

fluxos de dados: uma parte real, que é a tensão, e uma parte imaginária, que é a corrente. Esta abordagem requer uma FFT complexa de 256 pontos a todo 16,667 milissegundos.

Os resultados de executar qualquer FFT são a tensão e corrente mostradas na Figura 42, onde a notação $V_{m,n}$ denota o m-ésimo harmônico do n-ésimo número de ciclo. Por exemplo, V_{11} e I_{11} correspondem ao fundamental do primeiro ciclo, e V_{21} e I_{21} para o primeiro harmônico do primeiro ciclo, etc., como mostrado na Figura 42, que descreve quadros de FFT para tensão, indicando os harmônicos.

As partes real e imaginária do conteúdo harmônico de qualquer k-ésimo ciclo são dadas por:

$$V_{mk} = \text{Re}(V_{mk}) + i\text{Im}(V_{mk}); m = 1, \dots, M$$

$$I_{mk} = \text{Re}(I_{mk}) + i\text{Im}(I_{mk}); k = 1, \dots, n$$

A parte imaginária de tensão é a medida de falta de sincronização entre a PLL e a frequência de linha. A fim de calcular quantidades de medição, os cálculos são feitos no domínio de tempo. No domínio de tempo, a funcionalidade de FFT oferece a flexibilidade para calcular quantidades de medição tanto usando só o fundamental ou incluindo os harmônicos. Usando a forma complexa da tensão e corrente obtidas da FFT, as quantidades de medição são calculadas como:

$$P = V_{mk} * I_{mk}^*$$

$$W = \text{Re}(P) = \text{Re}(V_{mk} * \text{Re}(I_{mk}) + \text{Im}(I_{mk}) * \text{Im}(V_{mk}))$$

$$\text{Var} = \text{Im}(P) = \text{Re}(I_{mk}) * \text{Im}(V_{mk}) - \text{Re}(V_{mk}) * \text{Im}(I_{mk})$$

$$\text{Fator de Potência} = W/P$$

Porém, nas fórmulas anteriores, quando os harmônicos estão incluídos (V_{mk} e I_{mk} ; $m = 1, \dots, M$; $k = 1, \dots, n$), todas as quantidades de medição incluem os efeitos de harmônicos. Por outro lado, quando só o fundamental é usado (V_{1k} e I_{1k}), todas as quantidades calculadas representam só a contribuição de 60 Hz. Como um exemplo, nós mostramos os cálculos só quando o fundamental é usado para executar cálculos. Só V_1 e I_1 são usados de

todos os quadros de dados de FFT. As quantidades seguintes são calculadas para um dado conjunto de N quadros e uma frequência de linha de f_{line} :

$$kWh = \sum_{i=1}^N [\text{Re}(V_{1i}) * \text{Re}(I_{1i}) + \text{Im}(V_{1i}) * \text{Im}(I_{1i})] * \Delta t_i * 10^{-3}$$

$$kVAR = \sum_{i=1}^N [\text{Re}(I_{1i}) * \text{Im}(V_{1i}) - \text{Re}(V_{1i}) * \text{Im}(I_{1i})] * \Delta t_i * 10^{-3}$$

$$kVAh = \sum_{i=1}^N |V_{1i}| * |I_{1i}| * \Delta t_i * 10^{-3}; \Delta t = 1/f_{line}$$

$$V^2h = \sum_{i=1}^N |V_{1i}|^2 * \Delta t_i$$

$$I^2h = \sum_{i=1}^N |I_{1i}|^2 * \Delta t_i$$

O fator de potência de deslocamento é dado por:

$$\cos(\theta) = \left| \frac{W}{VA} \right|;$$

Onde W e VA incluem só os fundamentais e

$VA_1 = V_1\text{RMS} * I_1\text{RMS}$; onde

$$V_1\text{RMS} = \sqrt{\sum_{n=1}^N |V_{1,n}|^2} \text{ \& } I_1\text{RMS} = \sqrt{\sum_{n=1}^N |I_{1,n}|^2}$$

para N ciclos.

Esta flexibilidade para tanto incluir ou excluir os harmônicos ao calcular quantidades de medição se traduz a uma melhoria significativa sobre as capacidades oferecidas pela concretização acima descrita. Ainda outra característica oferecida por esta concretização é o cálculo de Distorção Harmônica Total (THD). A THD é a medição da distorção harmônica presente, e é definida como a relação da soma das potências de todos os componentes harmônicos para a potência do fundamental. Para o n-ésimo ciclo, isto é avaliado como:

$$VTHD_n = \frac{\sqrt{\sum_{m=2}^M V_{mn}^2}}{V_{1n}} \text{ \& } ITHD_n = \frac{\sqrt{\sum_{m=2}^M I_{mn}^2}}{I_{1n}}$$

V_{mn} (I_{mn}) é o m-ésimo harmônico do n-ésimo ciclo obtido da

FFT, onde

$$V_{m,n}^2 = \text{Re}(V_{m,n})^2 + \text{Im}(V_{m,n})^2 \text{ e } I_{m,n}^2 = \text{Re}(I_{m,n})^2 + \text{Im}(I_{m,n})^2.$$

Módulo de Exibição de Cliente

O módulo de exibição de cliente é instalado nas instalações de cliente, se comunica com o Guarda de Energia perto do transformador, e inclui: PCB 240, fonte de energia e circuitos de PLC (veja a Figura 33); e PCB 220, mostrador de LCD (veja a Figura 34). Em uma concretização, a unidade de exibição de cliente instalada na residência do cliente é um unidade de PLC bidirecional que se comunica com EG. Por exemplo, não só pode a utilidade enviar mensagens, o cliente também pode pedir uma verificação de consumo com o EG instalado no poste.

Enquanto certas concretizações específicas da invenção foram descritas aqui para propósitos ilustrativos, a invenção não está limitada aos detalhes específicos, dispositivos representativos, e exemplos ilustrativos mostrados e descritos aqui. Várias modificações podem ser feitas sem partir do espírito ou extensão da invenção definida pelas reivindicações anexas e seus equivalentes.

APÊNDICE

LCM.DEF

```

; QLC LCM (Load Control Module) program for PIC 16C63A;
;
;    include 'C:\pictools\16c63a.inc'
;    include 'C:\pictools\16c63.inc'      ;
; Configuration bits
;
;    FUSES _BODEN_OFF ; Brown-out reset disabled
;    FUSES _CP_OFF    ; No code protect
;    FUSES _PWRTE_OFF ; Disable power-up timer
;    FUSES _WDT_ON    ; Watchdog Timer Enabled
;    FUSES _RC_OSC    ; Oscillator = RC
; Program Parameter Equates
;
Software_type equ    7      ; Which PIC program is this?
;                          ; 7 = LCM program
Software_version equ  2      ; Version #
;
LCM_base_addr equ    16     ; Address of first LCM
;
Ram_start      equ    020h  ; Beginning of available RAM
;
RESET_VECTOR   equ    0     ; Location of reset vector
;
ROM_start      equ    008h  ; first available program location
;
Num_start_bits equ    12    ; Number of start bits
;
;    Port definitions
;
comm_port      equ    PORTA
;
data_out       equ    3
data_in        equ    4
clock_in       equ    5
;
ctl_port equ    PORTA
;
driver_enable  equ    0      ; output bit to turn on the drivers
;
Driver_enable_mask equ    1 << driver_enable
;
;    Initialization values for I/O ports
;
init_TRISA     equ    00110000b ; Data direction for port A
;
init_TRISB     equ    000h      ; B port data direction - all ports output
init_TRISC     equ    000h      ; C port data direction - all out

```

```

read_TRISC    equ    01fh                ; C port data direction for jumper read - 0-4 in

Init_comm_port_value equ    1 << data_out + Driver_enable_mask

option_reg_val equ    10001111b ; Option register init value

;          bit 7 = 1:    Port B pullups disabled
;          bit 6 = 0:    Int on falling edge
;          bit 5 = 0:    Timer 0 uses inst clock
;          bit 4 = 0:    Timer 0 counts on rising edge
;          bit 3 = 1:    Prescaler used by WDT
;          bit 2,1,0 = 111: Prescaler divide by 128

; NOTE - interrupts and timer 0 are not used by this program

ms_preset     equ    700    ; Time set for millisecond delay
;

; RAM locations

        ORG    Ram_start

Status_reg    ds    4        ; status register
        ORG    Status_reg
Status_MSB    ds    1

Clk_timeout_flag    equ    7        ; Timeout waiting for comm clock edge
bad_reset_flag      equ    6        ; Non-POR, non-WDT reset encountered
POR_flag            equ    5        ; Power-on reset detected
WDT_reset_flag      equ    4        ; WDT reset, not from sleep

Status_LSB         ds    1

CPU_reset_flag      equ    7
Need_param_refresh_flag equ    6
Parity_err_flag      equ    5

Global_cmd_timer    ds    1

Jumper_state        ds    1        ; current state of input jumpers

Parameter_reg       ds    4        ; Communication parameters
                    org    Parameter_reg
                    ds    1        ;
                    ds    2        ;
                    ds    1        ;

;
; Fast communication variables
;

Comm_state          ds    1        ; Stored state of fast comm uart
Wstate              ds    1        ; Stored state of fast comm word handler
Comm_bit_count      ds    1        ; Bit counter for fast comm
Comm_buffer          ds    1        ; fast comm word buffer

```

```

Work_addr      ds    1      ; Address read from jumpers
Temp           ds    1      ; temporary register
Tpar           ds    1      ; Transverse parity
Word_count     ds    1      ; Counter for sending and receiving 5-bit words
Command        ds      1      ; Command received over comm link
Register_ID    ds    1      ; Register specified in comm command
PIC_addr       ds    1      ; Address of this PIC chip
Comm_buf       ds    5      ; Communication buffer

Input_hold     ds    1      ; Hold register for comm rcv value

Output_bit_reg ds    4      ; Hold register for output bit mask

Flags          ds    1      ; Program control flags

Clk_wait_flag  equ    7      ; indicates which clock edge we are waiting for
Global_cmd_flag equ    6      ; Processing global command

scratch        ds    1      ; Work register
Timeout_ctr    ds    2      ; Counter for pulse duration
Millisecs      ds    1      ; Counter for pulse duration milliseconds

Clk_timeout_ctr      ds    2      ; counter for clock timeout
NPR_timer            ds    2      ; counter for need parameter refresh

;

      ORG    0a0h

; Bank 1 RAM locations

      IF      $ - 1 and 0100h
error - RAM overflow
      ENDIF

;

      ORG    RESET_VECTOR
      jmp    start      ; Jump to first program location

;
;      Start of program space
;
      ORG    ROM_start

;
;      Initialization
;
Start
      setb    RP0      ; point to upper register bank
      jnb    NOT_POR,got_POR      ; Power on reset, set flags
      clrb    RP0      ; point to RAM bank 0
      jnb    NOT_TO,WDT_reset      ; Watchdog reset
      setb    status_MSB.bad_reset_flag      ; Indicate unknown reset condition
      jmp    Do_initialize

```

```

WDT_reset
    setb    Status_MSB.WDT_reset_flag
    jmp     do_initialize
got_POR
    clrb    RPO                                ; point to RAM bank 0
    mov     Status_MSB,#1 << POR_flag ; Set POR flag
    clr     Status_LSB
Do_initialize
    mov     FSR,#Ram_start+2                ; Point past status flags
:clr_loop
    clr     INDF                            ; Clear RAM location
    inc     FSR                             ; Increment pointer
    jz      Init_state                      ; If zero, done
    mov     W,FSR                          ; get pointer
    and     W,#7fh                         ; look at 7 LSB's
    jnz     :clr_loop                      ; Not end of segment, continue
    add     FSR,#20h                        ; Point to next segment
    jmp     :clr_loop                      ; Not last segment, continue
Init_state
    setb    Status_LSB.CPU_reset_flag      ; Indicate CPU reset
    setb    Status_LSB.Need_param_refresh_flag ; Indicate parameter refresh needed
    mov     comm_port,#Init_comm_port_value ; initialize comm port state
    mov     Comm_state,#Edge_wait
    mov     Comm_bit_count,#Num_start_bits ; Initialize fast comm state
machine
    clr     PORTB
    clr     PORTC
    jmp     Main_loop                      ; start running main program

;
; Comm Receive routines
;
Edge_wait
    mov     !OPTION,#option_reg_val        ; Initialize Watchdog timer
    setb    comm_port.data_out             ; Set option reg
    dec     Comm_bit_count                 ; send a 1
    jz      See_edge_one                   ; decrement bit count
    jb      Input_hold.data_in,Set_edge_wait ; if zero, looking for a 1
    jmp     Comm_ret                       ; Looking for 0, reset bit count if 1
See_edge_one
    jnb     Input_hold.data_in,Set_edge_wait ; looking for 1, reset if 0
    mov     Wstate,#Rcv_addr               ; First word is address
    mov     Tpar,#01fh                     ; Init parity
    jmp     Rcv_next_word
Rcv_stop
    mov     Comm_state,#Edge_wait
    mov     Comm_bit_count,#Num_start_bits ; Preset state to wait for edge
    jnb     Input_hold.data_in,Parity_error ; stop bit must be 1
    jmp     Cmd_exec                       ; Done, process received buffer
Rcv_bits
    clc                                     ; copy ...

```



```

        snb     Input_hold.data_in      ; ... input bit ...
        setc                                         ; ... to carry
        rl      Comm_buffer             ; shift buffer
        jmp     Rcv_word_handler        ; Check for complete word
;
; Comm Transmit routines
;
Send_edge
    dec     Comm_bit_count              ; decrement bit count
    jz      :One                        ; if zero, send a 1
    clrb    comm_port.data_out          ; send a 0
    jmp     Comm_ret
:One
    setb    comm_port.data_out          ; send a 1
    jmp     Do_xmit_word

Send_bits
    rl      Comm_buffer                 ; put xmit bit into carry
    jc      :one                        ; send a '0'
    clrb    comm_port.data_out          ; send a '1'
    jmp     :check
:one
    setb    comm_port.data_out          ; send a '1'
:check
    jmp     Xmit_word_handler

;
; Comm receive word handler routines
;
Rcv_word_handler
    djnz    Comm_bit_count,Comm_ret     ; If word not complete, continue
    and     Comm_buffer,#03fh           ; only 6 bits
    clc
    mov     W,>>Comm_buffer              ; get 5 MSBs of received word
    call    parity_lookup
    xor     W,Comm_buffer                ; compare with received word
    jnz     Parity_error
    rr      Comm_buffer
    and     Comm_buffer,#01fh            ; extract 5-bit value
    xor     Tpar,Comm_buffer             ; calculate total parity
    mov     PCLATH,#Rcv_addr<
    mov     W,Wstate
    jmp     W                            ; Execute word handler routine

Rcv_addr
    clrb    Flags.Global_cmd_flag ; Clear global command flag
    mov     W,Comm_buffer          ; Get received address
    mov     PIC_addr,W             ; save it
    xor     W,#1fh                 ; is it global command
    jnz     See_if_us              ; No, check for address match
    setb    Flags.Global_cmd_flag ; Set global command flag
    jmp     Its_us                 ; process rest of command
See_if_us

```

```

mov    !PORTC,#read_TRISC ; Switch C port bit 0-4 to input
clrb   ctl_port.driver_enable ; Disable output drivers, enable jumper read
mov    temp,#9
:loop
  djnz  temp,:loop          ; Wait at least 20 uSec
  mov   W,/PORTC            ; Get inverse of jumpers in W
  mov   Work_addr,W         ; Save jumper value
  setb  ctl_port.driver_enable ; Enable output drivers
  mov   !PORTC,#init_TRISC ; Restore C port status
  mov   Jumper_state,Work_addr ; Save jumper state
  and   Work_addr,#03h      ; Ignore all but 2 LSBs
  add   Work_addr,#LCM_base_addr ; Add base offset
  cjne  Work_addr,PIC_addr,Set_edge_wait
Its_us
  mov   Wstate,#Rcv_cmd          ; point to next routine
  jmp   Rcv_next_word

Rcv_cmd
  mov   Command,Comm_buffer      ; save command
  jnb   Command.4,Comm_get_fast  ; Fast read command, skip Reg ID
  mov   Wstate,#Rcv_reg_ID      ; point to next routine
  jmp   Rcv_next_word

Comm_get_fast
  mov   Wstate,#Rcv_tpar        ; point to next routine
  jmp   Rcv_next_word

Rcv_reg_ID
  mov   register_ID,Comm_buffer  ; save register number
  mov   Wstate,#Rcv_tpar        ; point to next routine
  jnb   Command.3,Rcv_next_word ; If no value, done
  mov   word_count,#7           ; Init word count
  mov   Wstate,#Rcv_value       ; point to next routine
  jmp   Rcv_next_word

Rcv_value
  rl    Comm_buffer
  rl    Comm_buffer
  rl    Comm_buffer              ; Shift rcvd bits to MSBs
  mov   Comm_bit_count,#5       ; shift 5 bits
:shift_loop
  rl    Comm_buffer
  rl    Comm_buf+4
  rl    Comm_buf+3
  rl    Comm_buf+2
  rl    Comm_buf+1
  rl    Comm_buf                ; Shift a bit from Comm_buf to comm_buf
  djnz  Comm_bit_count,:shift_loop
  djnz  word_count,Rcv_next_word ; any more words?
  mov   Wstate,#Rcv_tpar        ; point to next routine
  jmp   Rcv_next_word

Rcv_tpar
  mov   W,Tpar                  ; Get result of parity calculations
  jnz   Parity_error            ; If not 0, parity error
  jmp   Set_rcv_stop            ; set up to receive stop bit

```

```

;
;      Comm transmit word handler routines
;
Xmit_word_handler
    djnz    Comm_bit_count,Comm_ret          ; If word not complete, continue
Do_xmit_word
    mov     PCLATH,$$<
    mov     W,Wstate
    jmp     W                                ; Execute word handler routine

Send_addr
    mov     Wstate,#Send_cmd>                ; point to next word routine
    mov     W,PIC_addr                       ; get address
    jmp     Send_next_word

Send_cmd
    mov     Wstate,#Send_value>              ; point to send value
    jnb     Command.4,Value_is_next          ; Is it fast read command?
    mov     Wstate,#Send_reg_ID>            ; point to next word routine
Value_is_next
    mov     W,Command                        ; get command
    jmp     Send_next_word

Send_reg_ID
    mov     Wstate,#Send_value>              ; point to next word routine
    mov     W,Register_ID                   ; get Register ID
    jmp     Send_next_word

Send_value
    mov     Wstate,#Next_value>              ; point to next word routine
    mov     Word_count,#7                   ; value is 7 words long

Next_value
    mov     Comm_bit_count,#5
:loop
    rl      comm_buf+4
    rl      comm_buf+3
    rl      Comm_buf+2
    rl      Comm_buf+1
    rl      Comm_buf
    rl      Comm_buffer
    djnz    Comm_bit_count,:loop
    mov     W,Comm_buffer                    ; Get 5-bit word
    djnz    word_count,Send_next_word        ; if not last word, all set
    mov     Wstate,#Send_Tpar>              ; send parity next
    mov     W,Comm_buffer
    jmp     Send_next_word

Send_Tpar
    mov     Wstate,#Send_stop                ; point to next word routine
    mov     W,Tpar                          ; get transverse parity

```

```

        jmp     Send_next_word

Send_stop
        jmp     Set_edge_wait           ; Go back to receive mode

;
;
;     Process received buffer
;
        org     100h

Cmd_exec
Process_cmd
        cjbe    Command,#7h,Fast_register_read    ; is it fast read?
        cje     Command,#10h,read_register        ; is it read register command?
        cje     Command,#18h,write_register       ; is it write register?
        cje     Command,#19h,reset_status_bits    ; is it reset status?
        jmp     Cmd_process_done

Fast_register_read
        mov     register_ID,command    ; Register number is same as command code

Read_register
        mov     W,#27
        mov     W,register_ID-W        ; Get index of register to read
        jnc     Cmd_process_done       ; if negative, not a valid register
        mov     Temp,W                 ; save index
        mov     PCLATH,#$<            ; set up High-order PC bits
        mov     W,Temp                 ; get index
        jmp     PC+W
        jmp     cpy_output_reg
        jmp     cpy_reg_28
        jmp     cpy_reg_29
        jmp     cpy_reg_30

Cpy_reg_31                                ; Reg 31 = Status register
        mov     Comm_buf,Status_reg
        mov     Comm_buf+1,Status_reg+1
        mov     Comm_buf+2,Status_reg+2
        mov     Comm_buf+3,Status_reg+3
        jmp     Setup_reply

Cpy_reg_30                                ; Reg 30 = Serial number
        call    serno
        mov     comm_buf+3,W
        call    serno+1
        mov     comm_buf+2,W
        call    serno+2
        mov     comm_buf+1,W
        call    serno+3
        mov     comm_buf,W
        jmp     Setup_reply

Cpy_reg_29                                ; Reg 29 = Software type and version
        mov     Comm_buf,#Software_type
        mov     Comm_buf+1,#Software_version
        mov     Comm_buf+2,#0
        mov     Comm_buf+3,#0          ; Send Software type and version

```

```

        jmp      Setup_reply

Cpy_reg_28                                ; Reg 28 = Parameter reg
        mov     Comm_buf,Parameter_reg
        mov     Comm_buf+1,Parameter_reg+1
        mov     Comm_buf+2,Parameter_reg+2
        mov     Comm_buf+3,Parameter_reg+3
        jmp     Setup_reply

Cpy_output_reg                            ; Reg 27 = Output state
        mov     Comm_buf,Output_bit_reg
        mov     Comm_buf+1,Output_bit_reg+1
        mov     Comm_buf+2,Output_bit_reg+2
        mov     Comm_buf+3,Output_bit_reg+3
        jmp     Setup_reply

Write_register
        mov     Comm_bit_count,#5                                ; need to shift buffer by 5
bits
:loop
        rl      comm_buf+4
        rl      comm_buf+3
        rl      comm_buf+2
        rl      comm_buf+1
        rl      comm_buf
        djnz    Comm_bit_count,:loop
        cjne    Register_ID,#28,See_write_output_reg ; Not reg 28, see if output reg
        mov     Parameter_reg,Comm_buf
        mov     Parameter_reg+1,Comm_buf+1
        mov     Parameter_reg+2,Comm_buf+2
        mov     Parameter_reg+3,Comm_buf+3                ; Copy received data to reg
        clrb    Status_LSB.Need_param_refresh_flag        ; Reset need parameter refresh flag
        jmp     Cpy_reg_28                                ; send contents back in reply

See_write_output_reg
        cjne    Register_ID,#27,Cmd_process_done          ; if not output reg, ignore
        mov     Output_bit_reg,Comm_buf
        mov     Output_bit_reg+1,Comm_buf+1
        mov     Output_bit_reg+2,Comm_buf+2
        mov     Output_bit_reg+3,Comm_buf+3                ; Copy received data to reg
        call    Update_outputs
        jmp     Cpy_output_reg                            ; send contents back in reply

Reset_status_bits
        mov     Comm_bit_count,#5                                ; need to shift buffer by 5
bits
:loop
        rl      comm_buf+4
        rl      comm_buf+3
        rl      comm_buf+2
        rl      comm_buf+1
        rl      comm_buf
        djnz    Comm_bit_count,:loop
        mov     W,/Comm_buf+1                                ; get byte
        and     Status_reg+1,W                                ; Clear bits
        /       mov     W,/Comm_buf                            ; get MSB

```

```

        and    Status_reg,W                ; Clear bits
        mov    Register_ID,#31             ; indicate status reg in reply
        jmp    Cpy_reg_31

Setup_reply
        jb     Flags.Global_cmd_flag,Set_edge_wait ; If it was global, don't reply
        mov    W,Status_LSB                ; Get status LSB
        mov    Comm_buf+4,W                ; preset stored flags
        and    W,#01fh                     ; Turn off 3 MSBs
        or     W,Status_MSB                 ; Combine 5 LSbs with Status_MSB
        sz
        setb   Comm_buf+4.7                ; If any other flag bits set, indicate in

MSB
        mov    Tpar,#1fh                   ; Initialize parity word
        mov    Comm_state,#Send_edge       ; Go to send_edge state
        mov    Wstate,#Send_addr           ; Point to next routine
        mov    Comm_bit_count,#Num_start_bits ; Init number of bits
        jmp    Comm_ret

Cmd_process_done
        jmp    Set_edge_wait                ; No reply needed, done

;
; Exit routines for comm
;

Parity_error
        setb   status_LSB.Parity_err_flag ; Indicate parity error

Set_edge_wait
        mov    Comm_state,#Edge_wait       ; Go to edge wait state
        mov    Comm_bit_count,#Num_start_bits ; Init number of bits
        jmp    Comm_ret

Set_send_reply
        mov    Comm_state,#Send_edge       ; Go to send edge state
        mov    Comm_bit_count,#Num_start_bits
        jmp    Comm_ret

Rcv_next_word
        mov    Comm_state,#Rcv_bits        ; set up to receive 6-bit word
        mov    Comm_bit_count,#6
        jmp    Comm_ret

Send_next_word
        and    W,#01fh                     ; only 5 bits
        xor    Tpar,W                       ; update parity calculation
        call   parity_lookup
        mov    Comm_buffer,W                ; Save word with parity (6 bits)
        rl     Comm_buffer
        rl     Comm_buffer                  ; move to MSBs
        mov    Comm_state,#Send_bits       ; set up to send 6-bit word
        mov    Comm_bit_count,#6
        jmp    Comm_ret

Set_rcv_stop

```

```

    mov    Comm_state,#Rcv_stop      ; point to next routine
    jmp    Comm_ret

    ORG    200h

Serno
;    retw   05ah,092h,05fh,000h
;    retw   0ffh,0ffh,0ffh,0ffh
    ds     4
    ret

;
;
;    Returns 6-bit value corresponding to passed 5-bit value with odd parity
;
parity_lookup
    mov     Temp,W                    ; Save index
    mov     PCLATH,#:table<          ; Set up high-order bits
    mov     W,Temp                    ; get index
    jmp     pc+w

:table
    retw    1,2,4,7,8,11,13,14,16,19,21,22,25,26,28,31,32,35,37,38,41,42,44,47,49,50,52,55,56,59,61,62

;
;
;    Sets output bits according to received data
;    Comm_buf = Pulse duration in milliseconds
;
;    0:      Continuous output (infinite duration)
;    1-255:  Set output state for specified duration
;             After duration is over, turn all outputs off
;
;
;    Comm_buf+1[3] = Bit map of output states - MSB first
;    bits 23-18:  Unused
;    bits 17-0:   Outputs 18-1
;
;
Update_outputs
    mov     W,Comm_buf+3              ; get lsbs
    and     W,#03h                    ; only need 2 bits
    mov     Temp,W
    add     Temp,W                     ; Shift into bits 1 and 2 (X1 and X2)
    mov     W,PORTA                   ; Get present status of port A pins
    and     W,#0f9h                   ; clear output control bits
    or      W,Temp                     ; Set desired output state
    mov     PORTA,W                   ; output new value
    rr      comm_buf+1
    rr      comm_buf+2
    rr      comm_buf+3
    rr      comm_buf+1
    rr      comm_buf+2
    rr      comm_buf+3                ; shift buffer left 2 bits
    mov     PORTB,comm_buf+2
    mov     PORTC,comm_buf+3
    mov     !PORTA,#init_TRISA

```

```

    mov    !PORTB,#init_TRISB
    mov    !PORTC,#init_TRISC
    mov    Millisecs,Comm_buf    ; first byte is pulse duration
    jz     done_output_set

ms_loop
    mov    Timeout_ctr,#ms_preset<
    mov    Timeout_ctr+1,#ms_preset>

:loop
    djnz   Timeout_ctr+1,:loop
    djnz   Timeout_ctr,:loop
    djnz   Millisecs,ms_loop      ; delay for pulse duration
    and    PORTA,#0f9h           ; turn off X1 and X2
    clr    PORTB
    clr    PORTC                 ; turn off other outputs
done_output_set
    ret

;
; Main Program start
;

Main_loop
    mov    !PORTA,#Init_TRISA    ; Initialize port A direction
    mov    !PORTB,#Init_TRISB    ; Initialize port B direction
    mov    !PORTC,#Init_TRISC    ; Init port C direction
    mov    Clk_timeout_ctr,#12    ; preset ...
    clr    Clk_timeout_ctr+1      ; ... clock timeout counter
    clrb   Flags.Clk_wait_flag    ; Indicate waiting for clock low

Clkwait_loop
    djnz   scratch,Clk_cont
    clr    WDT                    ; Reset watchdog
    djnz   Clk_timeout_ctr+1,Clk_cont
    djnz   Clk_timeout_ctr,Clk_cont
    jb     Status_MSB.POR_flag,got_POR    ; Ignore clock timeout until POR
flag_cleared
    setb   Status_MSB.Clk_timeout_flag    ; Indicate timeout on comm clock
    jmp    do_initialize

Clk_cont
    jb     Comm_port.clock_in,:high    ; clock high

:low
    jb     Flags.Clk_wait_flag,Clkwait_loop
    mov    Input_hold,Comm_port        ; Save Data port value
    setb   Flags.Clk_wait_flag         ; Indicate waiting for clock high
    jmp    Clkwait_loop

:high
    jnb    Flags.Clk_wait_flag,Clkwait_loop
    djnz   NPR_timer+1,:cont            ; Dec LSB of need parm timer
    djnz   NPR_timer,:cont              ; Dec MSB
    setb   Status_LSB.Need_param_refresh_flag ; set flag every 64k clocks

:cont
    mov    PCLATH,#0                  ; Set up high-order bits for routines in page 0
    mov    W,Comm_state                ; get vector
    jmp    W..                          ; Execute comm state machine

Comm_ret
    jmp    Main_loop

```


PIC.DEF

```

/*
include file for serial communication with PIC chips
*/

#pragma switch(ALL,FREQ)

switch (PIC_serial_status)
{
case sending_bits:
case sending_st5_bits:

#ifdef VATEST

    out_bit_value = (PICxmitBuf.bits.bufLong[0] and 0x80000000) != 0; /* out bit value =
MSB */
    PICxmitBuf.bits.bufLong[0] = PICxmitBuf.bits.bufLong[0] << 1;
    if (PICxmitBuf.bits.bufLong[1] and 0x80000000)
        PICxmitBuf.bits.bufLong[0]++;
    PICxmitBuf.bits.bufLong[1] = PICxmitBuf.bits.bufLong[1] << 1;
    if (PICxmitBuf.bits.bufLong[2] and 0x80000000)
        PICxmitBuf.bits.bufLong[1]++;
    PICxmitBuf.bits.bufLong[2] = PICxmitBuf.bits.bufLong[2] << 1; /* Shift 96-bit
buffer */

#else

    out_bit_value = 0;
    shift_LSB_ptr = andPICxmitBuf.bits.bufByte[12]; /* point past least significant word
of 96-bit register */
    asm(" MOVE.L _shift_LSB_ptr,A0", /* Get pointer to buffer, clear carry */
        " ROXL.W -(A0)", /* rotate left and decrement pointer */
        " ROXL.W -(A0)", /* rotate left and decrement pointer */
        " ROXL.W -(A0)", /* rotate left and decrement pointer */
        " ROXL.W -(A0)", /* rotate left and decrement pointer */
        " ROXL.W -(A0)", /* rotate left and decrement pointer */
        " ROXL.W -(A0)", /* 96 bits have been shifted, MSB is in carry
*/
        " ROXL.W _out_bit_value" /* Put MSB in out_bit_value */
    );

#endif

break;
case do_send_cmd:
    bit_count = NUM_START_BITS-1;
    PIC_serial_status = sending_start;
    out_bit_value = 0;
    break;
case do_PIC_start :
    bit_count = 3 * BITS_PER_SEC;
    out_bit_value = 1;
    PIC_serial_status = sending_PIC_start;
    break;
case sending_start:
    out_bit_value = 0;
    break;

```

```

    case sending_stop:
    case sending_start_1:
        out_bit_value = 1;
        break;
    default:
        out_bit_value = 1;
    }

    if(out_bit_value)
        fs1004.io.pulseOut2Off = 1;
    else
        fs1004.io.pulseOut2Off = 0;          /* Send bit out */
    in_bit_value = fs1004.io.pulseIn1State;   /* Get rcvd bit value */
    fs1004.io.pulseOut1Off = 1;              /* Send clock rising edge */

    switch (PIC_serial_status)
    {
    case waiting_for_start:
        if (in_bit_value == 0)
            zero_count++;
        else
        {
            if (zero_count == (NUM_START_BITS-1))
            {
                PIC_serial_status = rcving_bits;
                bit_count = PICrcvBuf.bitCount + 1;
            }
            else
                zero_count = 0;
        }
        break;
    case rcving_bits:

#ifdef VATEST

        PICrcvBuf.bits.bufLong[0] = PICrcvBuf.bits.bufLong[0] << 1;
        if (PICrcvBuf.bits.bufLong[1] and 0x80000000)
            PICrcvBuf.bits.bufLong[0]++;
        PICrcvBuf.bits.bufLong[1] = PICrcvBuf.bits.bufLong[1] << 1;
        if (PICrcvBuf.bits.bufLong[2] and 0x80000000)
            PICrcvBuf.bits.bufLong[1]++;
        PICrcvBuf.bits.bufLong[2] = (PICrcvBuf.bits.bufLong[2] << 1) +
            in_bit_value;
    #else

        shift_LSB_ptr = andPICrcvBuf.bits.bufByte[12]; /* point past least significant word
of 96-bit register */
        asm(" MOVE.L _shift_LSB_ptr,A0",          /* Get pointer to buffer, clear carry */
            " ROXL.W -(A0)",                      /* rotate left and decrement pointer */
            " ROXL.W -(A0)",                      /* rotate left and decrement pointer */
            " ROXL.W -(A0)",                      /* rotate left and decrement pointer */
            " ROXL.W -(A0)",                      /* rotate left and decrement pointer */
            " ROXL.W -(A0)",                      /* rotate left and decrement pointer */
            " ROXL.W -(A0)"                      /* 96 bits have been shifted, LSb is unknown
*/

```

```

    );
    PICrcvBuf.bits.bufByte[11] =
        (PICrcvBuf.bits.bufByte[11] and 0xfe) | in_bit_value;    /* add in received bit */
#endif

    break;
default:
    {
    }
}

if (--bit_count <= 0)
{
    switch (PIC_serial_status)
    {
        case rcving_bits:
            PIC_serial_status = rcving_stop;
            bit_count = 1;
            break;
        case sending_bits:
            PIC_serial_status = sending_stop;
            bit_count = 1;
            break;
        case sending_stop:
            if (PICxmitBuf.PIC_addr == GLOBAL_PIC_ADDR)
                PIC_serial_status = idle;
            else
            {
                PIC_serial_status = waiting_for_start;
                bit_count = PIC_wait_limit;
                zero_count = 0;
            }
            break;
        case sending_PIC_start:
            PIC_serial_status = idle;
            break;
        case sending_start:
            PIC_serial_status = sending_start_1;
            bit_count = 1;
            break;
        case sending_start_1:
            PIC_serial_status = sending_bits;
            bit_count = PICxmitBuf.bitCount;
            break;
        case waiting_for_start:
            PICrcvBuf.flags |= rcv_timeout_flag | rcv_data_ready_flag;
            PIC_serial_status = idle;
            break;
        case rcving_stop:
            if (in_bit_value == 0)
                PICrcvBuf.flags |= rcv_bad_length_flag;
            PIC_serial_status = idle;
            PICrcvBuf.flags |= rcv_data_ready_flag;
            bit_count = 1;
    }
}

```

```
        break;  
    default:  
        PIC_serial_status = idle;  
    }  
}
```

```
#pragma switch(ALL,NOFREQ)
```

PULSE.C

```

#include "mtrlink.def"
/* #include "flash.def" */
#define PSTRU_DEFINED
#include "pulselink.def"

#include "pulse.h"
#include "ufloat.h"

#include "log.h"

#include "alarm.h"

#include "copymem.h"

#include "plcctrl.h"

#include <stdio.h>

#ifdef fakeMC5
    #undef NUMPH
    ~ #define NUMPH 24
#endif

/* define WDT flags for the PIC communication routine */
#pragma region ("ram=WDTFflags")
short int WDTpulseSec ;
#pragma region ("ram=ram")

#pragma region ("data=secondBack")
void (*pulseSecondp)(void)=pulseSecond;
#pragma region ("data=data")

#ifdef IS_MC5
#pragma region ("data=everySubsecond")
#ifdef IS_RSM
void (*pulseSubsecp)(void)=pulseSubsecond ;
#else
void (*pulseSubsecp)(void)=pulseService;
#endif
#pragma region ("data=data")
#endif

#pragma region ("data=powerUp")
void (*pulseStartupp)(void)=pulseStartup ;
#pragma region ("data=data")

#pragma region ("data=dayBack")
void (*pulseDayp)(void)=pulseDay ;
#pragma region ("data=data")

```

```

/* clear out PIC */
void pulseColdstart()
{
    int i;
    for (i=0; i<num_PICs; i++)
        if (PIC_status[i].software_type == 3)
        {
            PIC_status[i].clear_pulse_regs = TRUE;
            PIC_status[i].get_status = TRUE;
        }
    clear_state = SEND_GLOBAL_CLR;
}

void pulseStartup(void)
{
    int i;

#ifdef IS_RSM
    int work_PIC_addr;
#endif
#ifdef IS_MC5
    int work_num_pulses;
#endif

#ifdef VATEST
    /* test !!!!! */
    temp_command = SYSTEM_CONTROL;
    temp_reg_ID = 0;
    temp_command_data = READ_ENCODER_ID;
    temp_flags = rcv_data_ready_flag;
#endif

    for (i=0; (!dont_clear_PIC_stats) andand (i<(2*NUMPH)); i++)
    {
        pulseData[i].error_cnt = 0;
        pulseData[i].ID_field[0]=0;
        pulscData[i].reading = 0;
    }

    for (i=0; i<MAX_PICS; i++)
        PIC_status[i].PIC_addr = 0;

    hold_num_pulse_ctrs = releaseCodep->option.numPulseCounters;
    if (hold_num_pulse_ctrs > NUMPH)
        num_pulse_ctrs = NUMPH;
    else
        num_pulse_ctrs = hold_num_pulse_ctrs;

    num_PICs = 1;

#ifdef IS_MC5
    PIC_status[0].PIC_addr = MC5_MUX_PIC_ADDR; /* If MC5, add PDM PIC(s) to PIC
table */

```

```

if(int_configtmessg.configLTM[PULSE1].accum)
{
    work_num_pulses = num_pulse_ctrs;
    work_PIC_addr = 0;
    while (work_num_pulses > 0)
    {
        PIC_status[num_PICs++].PIC_addr = work_PIC_addr++; /* Pulse counters for Q13
*/
        work_num_pulses -= 4;
    }
}
if(int_configtmessg.configLTM[PULSE2].accum)
{
    work_num_pulses = num_pulse_ctrs;
    work_PIC_addr = 6;
    while (work_num_pulses > 0)
    {
        PIC_status[num_PICs++].PIC_addr = work_PIC_addr++; /* Pulse counters for Q14
*/
        work_num_pulses -= 4;
    }
}
num_LCMs = releaseCodep->option.LCM_flags and 0x03;
work_PIC_addr = LCM_BASE_ADDR;
for (i=0;i<num_LCMs;i++)
{
    PIC_status[num_PICs++].PIC_addr = work_PIC_addr++; /* LCM's */
}
#else
#ifdef IS_ST5
    PIC_status[0].PIC_addr = ST5_MUX_PIC_ADDR; /* If ST5, add mux to PIC table */
    ATM_work = releaseCodep->option.couplers_mask;
    work_PIC_addr = ATM_BASE_ADDR;
    for (i=0;i<4;i++)
    {
        if ((ATM_work and 0x08) != 0)
        {
            /* Bit is set in mask, add ATM PIC to table */
            PIC_status[num_PICs++].PIC_addr = work_PIC_addr;
        }
        work_PIC_addr++;
        ATM_work = ATM_work << 1;
    }
}
#else
    if (num_pulse_ctrs == 0)
        num_PICs = 0;
#endif
#endif
for (i=0; i<MAX_PICS; i++)
{
    /* Build PIC table */
    PIC_status[i].get_serno = TRUE;
    PIC_status[i].get_version = TRUE;
    PIC_status[i].get_status = FALSE;
    PIC_status[i].reset_status = FALSE;
    PIC_status[i].update_parameter = FALSE;
}

```

```

if (!dont_clear_PIC_stats)
{
    PIC_status[i].PIC_reset_count = 0;
    PIC_status[i].comm_error_count = 0;
    PIC_status[i].PIC_data_err_count = 0;
    PIC_status[i].software_type = 0;
    PIC_status[i].software_version = 0;
    PIC_status[i].serial_number = 0;
}

PIC_status[i].curr_pulse_reg = 0;

#ifdef IS_MCS

    if (PIC_status[i].PIC_addr >= ST5_MUX_PIC_ADDR)
        PIC_status[i].reply_wait_limit = DIRECT_PIC_WAIT_LIMIT;
    else
        PIC_status[i].reply_wait_limit = BUFFERED_PIC_WAIT_LIMIT;
#else
    PIC_status[i].reply_wait_limit = DIRECT_PIC_WAIT_LIMIT;
#endif

    if (PIC_status[i].PIC_addr <= MAX_PULSE_PIC_ADDR)
        PIC_status[i].num_pulse_regs = PULSES_PER_PIC;
    else
        PIC_status[i].num_pulse_regs = 0;

}

current_PIC_index = 0;
rcv_PIC_index = 0;

if (dont_clear_PIC_stats)
    dont_clear_PIC_stats = FALSE; /* Clear flag */
else
{
    PIC_bad_addr_count = 0;
    PICrcvBuf.flags = 0;
    PIC_serial_status = do_PIC_start;
    PIC_clk_state = clk_low; /* Initialize serial comm variables */
}

clear_state = NO_GLOBAL_SEND;

pulseOutMode = releaseCode.option.picMode; /* for compatibility with old code */

if (startup.coldStart)
{
    pulseColdstart();
}
}

```



```
const unsigned char parity6[32] =
{1,2,4,7,8,11,13,14,16,19,21,22,25,26,28,31,32,35,37,38,41,42,44,47,49,50,52,55,56,59,61,62
};
```

```
void pulseDay()
```

```
{
    dont_clear_PIC_stats = TRUE; /* Tell pulseStartup not to clear error counts */
    rebuild_timer = REBUILD_DELAY; /* Set up for delayed table rebuild */
}
```

```
void pulseSecond()
```

```
{
#ifdef IS_ST5
    pulseService();
#endif
    if (Comm_background_flags.do_rebuild)
    {
        pulseStartup();
        Comm_background_flags.do_rebuild = FALSE;
    }
    if (Comm_background_flags.do_alarm)
    {
        putAlarm(andPIC_alarm,4);
        Comm_background_flags.do_alarm = FALSE;
    }
}
```

```
/* set_PLC_relays function */
/* Called from PLC routines to request update of */
/* PLC port multiplexer PIC chip on ST5 power board */
```

```
#ifdef IS_ST5
```

```
void set_PLC_relays(int xmitRelay,int rcvRelay)
```

```
{
    while (MUX_control.active_flag || MUX_control.request_flag)
    {
        ccwait();
    }
    MUX_control.xmit_mask = xmitRelay;
    MUX_control.rcv_mask = rcvRelay;
    MUX_control.done_flag = FALSE;
    MUX_control.request_flag = TRUE;
    if ((MUX_control.xmit_mask != MUX_control.last_xmit_mask) ||
        (MUX_control.rcv_mask != MUX_control.last_rcv_mask))
    {
        while (!MUX_control.done_flag andand
            !(MUX_control.active_flag andand (PIC_serial_status == rcving_bits)))
        {
            ccwait();
        }
    }
}
```

```

#endif

void pulseService(void)
{
    BOOLEAN    need_pulse_read,
               copy_rcvd_data,
               rcv_buf_err;

#ifdef IS_MCS
    BOOLEAN
        got_ver_3,
        got_ver_3_3;

    int      j;

    unsigned char *work_char_ptr;

    unsigned long work_bit_mask;
#endif

    int      work_pulse_index;
    phaccum accumXfer;
    int i;

    if (num_PICs == 0)
        PIC_serial_status = idle;

    if (PIC_serial_status == idle)
    {
        if (!WDTpulseSec)
            WDTpulseSec=1; /*Tell WDT controller that we are OK*/
    }

    if (hold_num_pulse_ctrs != releaseCodep->option.numPulseCounters)
    {
        /* hdwr -n has changed */
        dont_clear_PIC_stats = FALSE;
        rebuild_timer = 1; /* Force immediate table rebuild */
    }

    if ((PIC_serial_status == idle) andand
        (rebuild_timer != 0) andand
        (~rebuild_timer == 0))
    {
        Comm_background_flags.do_rebuild = TRUE;
    }
    else if ((PIC_serial_status == idle) andand
             (num_PICs != 0) andand
             (pulseOutMode != PIC_MC_SERIAL) andand
             (!Comm_background_flags.do_rebuild)) /* Serial processing enabled? */
    {
        if (PICrcvBuf.flags and rcv_data_ready_flag)
        {
            /* Process received message */
            PICrcvBuf.flags and= ~rcv_data_ready_flag;

```

```

rcv_buf_err = !decode_rcv_buffer();

if ((PIC_comm_mon.control_flags and RCV_DATA_RDY) == 0)
{
    PIC_comm_mon.rcv_buf = PICrcvBuf;
    PIC_comm_mon.control_flags |= RCV_DATA_RDY;
}

if ((ext_comm.control_flags and CPY_RCV_DATA) != 0)
{
    ext_comm.rcv_buf = PICrcvBuf;
    ext_comm.control_flags and= ~CPY_RCV_DATA;
    ext_comm.control_flags |= RCV_DATA_RDY;
}

#ifdef IS_ST5
if (ATM_control.active_flag)
{
    ATM_control.result_flags = PICrcvBuf.flags;
    if (rcv_buf_err)
    {
        if (ATM_control.state != WAIT_ALIGN)
        {
            ATM_control.error_flag = TRUE;
            ATM_control.active_flag = FALSE;
            ATM_control.done_flag = TRUE;
        }
    }
}
else
{
    switch (ATM_control.state)
    {
        case DO_PRESET:
        {
            ATM_control.xmit_level = PICrcvBuf.command_data >> 24;
            ATM_control.coupler_level = (PICrcvBuf.command_data >> 16) and 0x0ff;
            ATM_control.active_flag = FALSE;
        }
        break;
        case SET_ALIGN:
        {
            ATM_control.state = WAIT_ALIGN;
        }
        break;
        case WAIT_ALIGN:
        {
            ATM_control.optimum_cap_code = PICrcvBuf.command_data and
ATM_RC_CAP_MASK;
            ATM_control.optimum_reading = PICrcvBuf.command_data >> 16;
            ATM_control.result_flags |= (PICrcvBuf.command_data and 0xe000);
            /* Copy flags from reply */
            if ((ATM_control.result_flags and 0xc000) != 0)
            {
                /* Analog levels out of range */
                ATM_control.error_flag = TRUE;
            }
        }
    }
}

```

```

        if(ATM_control.optimum_reading < ATM_MIN_READING)
        {
            ATM_control.result_flags |= 0x1000; /* Indicate reading too low */
            ATM_control.error_flag = TRUE;
        }
        ATM_control.active_flag = FALSE;
    }
    break;
case DO_PLC_SET:
case DO_ATM_DISCONNECT:
    {
        ATM_control.active_flag = FALSE;
    }
    break;
default:
    {
        ATM_control.active_flag = FALSE;
        ATM_control.error_flag = TRUE;
    }
    break;
}
}
if(!ATM_control.active_flag)
    ATM_control.done_flag = TRUE;
}

if(MUX_control.active_flag)
{
    MUX_control.result_flags = PICrcvBuf.flags;
    if(rcv_buf_err)
    {
        MUX_control.error_flag = TRUE;
        MUX_control.last_xmit_mask = 0;
        MUX_control.last_rcv_mask = 0;
    }
    else
    {
        MUX_control.last_xmit_mask = PICrcvBuf.command_data and 0xff;
        MUX_control.last_rcv_mask = (PICrcvBuf.command_data >> 8) and 0xff;
    }
    MUX_control.active_flag = FALSE;
    MUX_control.done_flag = TRUE;
}

#endif

    if(rcv_buf_err)
    {
#ifdef IS_ST5
        if(!ATM_control.active_flag || (ATM_control.state != WAIT_ALIGN))
#endif
        {
            last_comm_err_flag = PICrcvBuf.flags;
            copymem(sizeof(PICbitBuf), (char *) (andHoldrcvBuf), (char *)
(andPICrcvBuf));
            if((PICrcvBuf.flags and rcv_bad_addr_flag) != 0)

```

```

{      /* past end of valid PICs, address error */
PIC_bad_addr_count++;
set_PIC_alarm(PIC_addrerr_0,PICrcvBuf.PIC_addr,
              (PICxmitBuf.PIC_addr << 8)+PICxmitBuf.command);
}
else
{      /* PIC address was valid, but comm failed */
PIC_status[rcv_PIC_index].comm_error_count++;
if (PIC_status[rcv_PIC_index].comm_error_count > 2)
{
set_PIC_alarm(PIC_comerr_0,PIC_status[rcv_PIC_index].PIC_addr,
              PICrcvBuf.flags);
}

/* Lost contact with PIC, read serial number and software version */
PIC_status[rcv_PIC_index].get_serno = TRUE;
PIC_status[rcv_PIC_index].get_version = TRUE;
}
}
}
else
{
/* Valid message received */
if ((PICrcvBuf.command == READ_REGISTER) andand (PICrcvBuf.reg_ID ==
STATUS_REG))
{
process_status_bits();
}
else if (PICrcvBuf.command == SYSTEM_CONTROL)
{
if (PICrcvBuf.reg_ID == STATUS_REG)
{
/* Response from reset status bits cmd */
PIC_status[rcv_PIC_index].reset_status = FALSE;
process_status_bits();
}
}
#ifdef IS_MC5
else if ((PIC_status[rcv_PIC_index].software_type == 4) andand
((PIC_status[rcv_PIC_index].last_cmd_data and 0xffff0000) ==
READ_ENCODER_ID))
{
/* response from encoder read ID cmd */
work_pulse_index = (PICrcvBuf.PIC_addr * PULSES_PER_PIC) +
PICrcvBuf.reg_ID;
i = (PIC_status[rcv_PIC_index].last_cmd_data and 0x0000ff00) >> 8;
work_char_ptr = (unsigned char *) andPICrcvBuf.command_data;
for (j=0; j<4; j++)
{
if ((*work_char_ptr == 0) || (i >= ID_FIELD_LENGTH))
{
j = 10; /* indicate end of message found */
}
else
{
pulseData[work_pulse_index].ID_buffer[i] = *work_char_ptr;
i++;
work_char_ptr++;
}
}
}
}

```

```

        if (j>5)
        {
            /* found end of ID field from remote encoder */
            for (i=0; i<ID_FIELD_LENGTH; i++)
            {
                pulseData[work_pulse_index].ID_field[i] =
pulseData[work_pulse_index].ID_buffer[i];
                pulseData[work_pulse_index].ID_buffer[i] = 0;
            }
        }
    }
    else if ((PICrcvBuf.command == READ_REGISTER) andand (PICrcvBuf.reg_ID
== ERROR_REG))
    {
        if (PIC_status[rcv_PIC_index].software_type == 4)
        {
            /* Remote encoder interface module */
            work_pulse_index = (PICrcvBuf.PIC_addr * PULSES_PER_PIC) +
(PICrcvBuf.command_data >> 24);
            if (work_pulse_index < (2 * NUMPH))
            {
                pulseData[work_pulse_index].error_cnt++;
                for (i=0; i<ID_FIELD_LENGTH;
pulseData[work_pulse_index].ID_buffer[i++] = 0);
            }
        }
    }
#endif
}
    else if ((PICrcvBuf.command == WRITE_REGISTER) andand (PICrcvBuf.reg_ID
== PARM_REG))
    {
        PIC_status[rcv_PIC_index].update_parameter = FALSE;
    }
    else if ((PICrcvBuf.command == READ_REGISTER) andand (PICrcvBuf.reg_ID
== VERS_REG))
    {
        PIC_status[rcv_PIC_index].get_version = FALSE;
        PIC_status[rcv_PIC_index].software_type =
(PICrcvBuf.command_data and 0xff000000) >> 24;
        PIC_status[rcv_PIC_index].software_version =
(PICrcvBuf.command_data and 0x00ff0000) >> 16;
        PIC_status[rcv_PIC_index].loop_rate =
(PICrcvBuf.command_data and 0x0000ff00) >> 8;
    }
    else if ((PICrcvBuf.command == READ_REGISTER) andand (PICrcvBuf.reg_ID
== SERNO_REG))
    {
        PIC_status[rcv_PIC_index].get_serno = FALSE;
        PIC_status[rcv_PIC_index].serial_number = PICrcvBuf.command_data;
    }
    :: else if (((PICrcvBuf.command == READ_REGISTER) || (PICrcvBuf.command <=
MAX_FAST_READ))
        andand (PICrcvBuf.reg_ID <= MAX_PULSE_REG))
    {
        /* Pulse reading */
        work_pulse_index = (PICrcvBuf.PIC_addr * PULSES_PER_PIC) +
PICrcvBuf.reg_ID;

```

```

if (work_pulse_index < (NUMPH * 2))
{
    pulseData[work_pulse_index].reading = PICrcvBuf.command_data;
    copy_rcvd_data = FALSE;
    if (PIC_status[rcv_PIC_index].get_version)
    {
    }
    else if (PIC_status[rcv_PIC_index].software_type == 4)
    {
        /* Remote encoder interface module */
        copy_rcvd_data = TRUE;
        accumXfer.low = pulseData[work_pulse_index].reading;
    }
    else if (PIC_status[rcv_PIC_index].software_type == 3)
    {
        /* Remote pulse counter module */
        copy_rcvd_data = TRUE;
    }

    if (!PIC_status[rcv_PIC_index].get_serno)
    {
        sprintf((char *) pulseData[work_pulse_index].ID_field, "S%08ldP%d",
            PIC_status[rcv_PIC_index].serial_number,
            PICrcvBuf.reg_ID);
    }

    if (releaseCode.option.countEveryEdge)
    {
        accumXfer.low = pulseData[work_pulse_index].reading;
    }
    else
    {
        accumXfer.low = pulseData[work_pulse_index].reading >> 1;
    }
}
accumXfer.high = 0L;
#ifdef VATEST
    if (copy_rcvd_data)
    {
#ifdef IS_MC5
        /* It is an RSM or ST5, so mapping for pulse 1 thru 4 is M1Q13, M1Q14, M2Q13,
        M2Q14 */
        ph[0][work_pulse_index/2][(work_pulse_index&0x01)?PULSE2:PULSE1].accum =
        ph[1][work_pulse_index/2][(work_pulse_index&0x01)?PULSE2:PULSE1].accum =
            accumXfer;

    #else
        /* It is an MC5, so mapping for pulse 1 thru 48 is M1Q13..M24Q13,
        M1Q14..M24Q14 */
        ph[0][work_pulse_index%NUMPH][(work_pulse_index/NUMPH)?PULSE2:PULSE1].accu
m =

```

```

ph[1][work_pulse_index%NUMPH][((work_pulse_index/NUMPH)?PULSE2:PULSE1].accum
m =
        accumXfer;
#endif
    }

#endif
    }

    if ((PICrcvBuf.flags and (PIC_error_flag | PIC_invalid_data_flag)) != 0)
        PIC_status[rcv_PIC_index].get_status = TRUE;
    if ((PICrcvBuf.flags and PIC_need_refresh_flag) != 0)
        PIC_status[rcv_PIC_index].update_parameter = TRUE;
    /* check status flags */
}

/* Decide what command to send */

PICxmitBuf.flags = 0;

#ifndef IS_ST5
/* MC5 or RSM, check for cold-start processing to clear pulse registers */

if (clear_state == SEND_GLOBAL_CLR)
{
    /* Cold start, send global clr */
    PICxmitBuf.PIC_addr = GLOBAL_PIC_ADDR;
    PICxmitBuf.command = CLEAR_PULSE_REGISTERS;
    format_xmit_buffer();
    clear_state = SENDING_GLOBAL_CLR;
    PIC_serial_status = do_send_cmd; /* Start UART */
}
else if (clear_state == SENDING_GLOBAL_CLR)
{
    clear_state = GLOBAL_CLR_SENT;
}
#else

/* ST5, check for ATM (Automatic Tuning Module) operations */

if (ATM_control.request_flag)
{
    ATM_control.request_flag = FALSE;
    ATM_control.error_flag = FALSE;
    ATM_control.active_flag = TRUE;
    ATM_control.state = ATM_DONE;
    PICxmitBuf.PIC_addr = ATM_control.ATM_number + ATM_BASE_ADDR;

    switch (ATM_control.operation)
    {
        case SET_PRE_ALIGN:

```



```

        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = OUTPUT_REG;
        PICxmitBuf.command_data = ATM_control.start_cap_code and
ATM_RC_CAP_MASK;
        PICxmitBuf.command_data |= ATM_PRESET_CODE;
        ATM_control.state = DO_PRESET;
        break;

    case DO_ALIGN:
        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = CONTROL_REG;
        if (ATM_control.start_cap_code > ATM_control.end_cap_code)
            ATM_control.end_cap_code = ATM_control.start_cap_code;
        PICxmitBuf.command_data = ATM_control.start_cap_code and
ATM_RC_CAP_MASK;
        PICxmitBuf.command_data = (PICxmitBuf.command_data << 12) |
            (ATM_control.end_cap_code and ATM_RC_CAP_MASK) |
            (ATM_MEAS_TIME << 24);
        ATM_control.state = SET_ALIGN;
        ATM_control.align_timer = ATM_control.end_cap_code -
ATM_control.start_cap_code;
        ATM_control.align_timer *= ((ATM_DISC_TIME + ATM_MEAS_TIME + 4) *
(1.25 / 62.5));
        ATM_control.align_timer += 13; /* Time in 1/64 sec to wait before checking
result */
                                   /* Number of steps * time per step + 25% + .2 second */
        break;

    case SET_NORMAL_PLG:
        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = OUTPUT_REG;
        PICxmitBuf.command_data = ATM_control.optimum_cap_code and
ATM_RC_CAP_MASK;
        PICxmitBuf.command_data |= ATM_PLG_CODE;
        ATM_control.state = DO_PLG_SET;
        break;

    case DISCONNECT_COUPLER:
        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = OUTPUT_REG;
        PICxmitBuf.command_data = ATM_control.optimum_cap_code and
ATM_RC_CAP_MASK;
        PICxmitBuf.command_data |= ATM_DISC_CODE;
        ATM_control.state = DO_ATM_DISCONNECT;
        break;

    default:
    case READ_XMIT_LEVEL:
        ATM_control.error_flag = TRUE;
        ATM_control.done_flag = TRUE;
        ATM_control.active_flag = FALSE;
        break;
}

```

```

if (ATM_control.active_flag)
{
    format_xmit_buffer();
    PIC_wait_limit = DIRECT_PIC_WAIT_LIMIT;
    PIC_serial_status = do_send_cmd;    /* Start UART */
}

}

else if (ATM_control.active_flag) :
{
    if ((ATM_control.state != WAIT_ALIGN) ||
        (-ATM_control.align_timer <= 0))
    {
        ATM_control.error_flag = TRUE;
        ATM_control.done_flag = TRUE;
        ATM_control.active_flag = FALSE;
    }
    else
    {
        /* poll ATM for auto-tune end */
        PICxmitBuf.command = 0;
        format_xmit_buffer();
        PIC_wait_limit = DIRECT_PIC_WAIT_LIMIT;
        PIC_serial_status = do_send_cmd;    /* Start UART */
    }
}

else if (MUX_control.request_flag)
{
    MUX_control.request_flag = FALSE;
    MUX_control.error_flag = FALSE;
    if ((pulseOutMode == PIC_ST_SERIAL))
    {
        PICxmitBuf.bits.bufByte[0] = ((MUX_control.rcv_mask and 0x03) << 5) |
            ((MUX_control.xmit_mask and 0x7f) >> 2);

        PICxmitBuf.bits.bufByte[1] = ((MUX_control.xmit_mask and 0x03) << 6);

        bit_count = 11;
        PIC_serial_status = sending_st5_bits;    /* Start UART */
        MUX_control.last_xmit_mask = MUX_control.xmit_mask;
        MUX_control.last_rcv_mask = MUX_control.rcv_mask;
        MUX_control.done_flag = TRUE;
    }
    else
    {
        MUX_control.active_flag = TRUE;
        PICxmitBuf.PIC_addr = ST5_MUX_PIC_ADDR;
        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = OUTPUT_REG;
        PICxmitBuf.command_data = (MUX_control.rcv_mask << 8) +
MUX_control.xmit_mask;
        format_xmit_buffer();
        PIC_wait_limit = DIRECT_PIC_WAIT_LIMIT;
        PIC_serial_status = do_send_cmd;    /* Start UART */
    }
}

```

```

    }
}

#endif

else if ((ext_comm.control_flags and (XMIT_DATA_RDY | XMIT_REQUEST)) ==
        (XMIT_DATA_RDY | XMIT_REQUEST))
{
    /* Send command for external routine */
    PICxmitBuf = ext_comm.xmit_buf;
    ext_comm.control_flags and= ~XMIT_REQUEST;
    ext_comm.control_flags |= CPY_RCV_DATA;

    /* Look up correct PIC_status index for requested PIC */
    for (i=0; ((i<MAX_PICS) andand (PIC_status[i].PIC_addr != PICxmitBuf.PIC_addr));
    i++);

    if (i < num_PICs)
    {
        PIC_status[i].last_cmd_data = PICxmitBuf.command_data;
        PIC_status[i].last_command = PICxmitBuf.command;
    }

#ifdef IS_MC5
    if (ext_comm.xmit_buf.PIC_addr < ST5_MUX_PIC_ADDR)
        PIC_wait_limit = BUFFERED_PIC_WAIT_LIMIT;
    else
#endif
        PIC_wait_limit = DIRECT_PIC_WAIT_LIMIT;

    format_xinit_buffer();
    PIC_serial_status = do_send_cmd;      /* Start UART */
}

else if ((ext_comm.control_flags and SUPPRESS_NORMAL_COMM) == 0)
#ifdef IS_ST5
    andand (--ST5_PIC_delay_ctr <= 0))
    {
        ST5_PIC_delay_ctr = ST5_PIC_DELAY; /* Slow down routine communications in
the ST5 */
    }
#else
    )
{
#endif
    PICxmitBuf.PIC_addr = PIC_status[current_PIC_index].PIC_addr;
    need_pulse_read = TRUE; /* Set defaults */

#ifdef IS_ST5

    if ((pulseOutMode == PIC_ST_SERIAL) andand
        (PICxmitBuf.PIC_addr == ST5_MUX_PIC_ADDR))
    {
    }
    else

```

```
#endif
```

```
if (PIC_status[current_PIC_index].force_pulse_read andand
(!PIC_status[current_PIC_index].get_version) andand
(PIC_status[current_PIC_index].PIC_addr < ST5_MUX_PIC_ADDR) )
{
    PIC_status[current_PIC_index].force_pulse_read = FALSE;
}
else
{
    PIC_status[current_PIC_index].force_pulse_read = TRUE;
```

```
#ifndef IS_ST5
```

```
if (PIC_status[current_PIC_index].clear_pulse_regs andand
(clear_state == GLOBAL_CLR_SENT) )
{
    /* Global clear sent, get status */
    need_pulse_read = FALSE;
    PICxmitBuf.command = READ_REGISTER;
    PICxmitBuf.reg_ID = STATUS_REG;
    format_xmit_buffer();
}
else
```

```
#endif
```

```
if (PIC_status[current_PIC_index].get_version)
{
    need_pulse_read = FALSE;
    PICxmitBuf.command = READ_REGISTER;
    PICxmitBuf.reg_ID = VERS_REG;
    format_xmit_buffer();
}
else if (PIC_status[current_PIC_index].get_serno)
{
    need_pulse_read = FALSE;
    PICxmitBuf.command = READ_REGISTER;
    PICxmitBuf.reg_ID = SERNO_REG;
    format_xmit_buffer();
}
else if (PIC_status[current_PIC_index].update_parameter)
{
    PICxmitBuf.command = WRITE_REGISTER;
    if (PIC_status[current_PIC_index].software_type == 3)
    {
        /* Pulse ctr */

        /* Pulse ctr parameters: */
        /* Power-On sample rate: 42/sec */
        /* Power-off sample rate: 1/sec */
        /* Days to count if revving pulses: */
        /* RSM: 5 days */
        /* PDM: 35 days */
        /* Days to count if no pulses rcvd: */
        /* RSM: 1 day */
        /* PDM: 3 days */

        /* on RSM the 2 lsb of */
    }
```

```

        /* load_shed_mask[0] are sent to */
        /* the PIC output bits */
#ifdef IS_RSM
        PICxmitBuf.command_data = 0x01150507 and
        (load_shed_state[0] | 0xffffffffc);
#else
        PICxmitBuf.command_data = 0x0115230d;
#endif
    }
#ifdef IS_MC5
    else if (PIC_status[current_PIC_index].software_type == 4)
    {
        /* Remote encoder reader */
        PICxmitBuf.command_data = READ_ENCODER_PARM;
        if ((releaseCodep->option.couplers_mask and 0x01) == 0)
        {
            /* If mask bit 0 is 0, no touch-pad compatibility */
            PICxmitBuf.command_data and= 0xff00ffff;
        }
    }
    else if (PIC_status[current_PIC_index].software_type == 5)
    {
        /* MC5 mux */
        got_ver_3 = got_ver_3_3 = FALSE;

        for (i=0; i<num_PICs; i++)
        {
            if (PIC_status[i].get_version)
            {
                got_ver_3 = TRUE;
                got_ver_3_3 = TRUE;
            }
            else if (PIC_status[i].software_type == 3)
            {
                got_ver_3 = TRUE;
                if (PIC_status[i].software_version == 3)
                    got_ver_3_3 = TRUE;
            }
        }
        if (got_ver_3)
        {
            if (got_ver_3_3 andand
                ((releaseCodep->option.couplers_mask and 0x02) != 0))
            {
                PICxmitBuf.command_data = 0x0c007000; /* 32 bps, periodic wake-up */
            }
        }
        else
        {
            PICxmitBuf.command_data = 0x0c007080; /* 32 bps, no wake-up */
        }
    }
    else
        PICxmitBuf.command_data = 0x02030080; /* 205 bps, no wake-up */
}

#endif
#endif IS_ST5

```

```

else if (PIC_status[current_PIC_index].software_type == 8)
{
    /* Auto tune module */
    PICxmitBuf.command_data = 0x02030000 | (ATM_DISC_TIME << 8) |
    (ATM_RC_XMIT_ON >> 8) | ATM_NUM_CAP_BITS;
    /* xmit offset = 2, cplr offset = 3, */
    /* relay delay = ATM_DISC_TIME, Xmit mask = ATM_RC_XMIT_ON, */
    /* number of capacitor relays = ATM_NUM_CAP_BITS */
}
else if (PIC_status[current_PIC_index].software_type == 6)
{
    /* ST5 mux */
    PICxmitBuf.command_data = 0x00000000;
}
#endif

else if (PIC_status[current_PIC_index].software_type == 7)
{
    /* Load Control Module */
    PICxmitBuf.command_data = 0x00000000;
}
else
{
    PIC_status[current_PIC_index].update_parameter = FALSE;
    /* unknown software type, cancel command */
}

if (PIC_status[current_PIC_index].update_parameter)
{
    need_pulse_read = FALSE;
    PICxmitBuf.reg_ID = PARM_REG;
    format_xmit_buffer();
}
}

else if (PIC_status[current_PIC_index].reset_status)
{
    need_pulse_read = FALSE;
    PICxmitBuf.command = SYSTEM_CONTROL;
    PICxmitBuf.command_data =
    (PIC_status[current_PIC_index].status_reg_value and 0xffbf);
    PICxmitBuf.command_data = PICxmitBuf.command_data << 16; /* Move
status bits to MSb's */
    PICxmitBuf.reg_ID = STATUS_REG;
    format_xmit_buffer();
}
else if (PIC_status[current_PIC_index].get_status)
{
    need_pulse_read = FALSE;
    PICxmitBuf.command = READ_REGISTER;
    PICxmitBuf.reg_ID = STATUS_REG;
    format_xmit_buffer();
}
}

#ifdef IS_MC5
if (need_pulse_read andand (PIC_status[current_PIC_index].software_type == 4))
{
    need_pulse_read = FALSE;
    PICxmitBuf.command = SYSTEM_CONTROL;

```

```

if (++PIC_status[current_PIC_index].curr_ID_reg >= PULSES_PER_PIC)
    PIC_status[current_PIC_index].curr_ID_reg = 0;

if ( (((PIC_status[current_PIC_index].PIC_addr * PULSES_PER_PIC) +
    PIC_status[current_PIC_index].curr_ID_reg) % 24) >= num_pulse_ctrs)
    PIC_status[current_PIC_index].curr_ID_reg = 0;

PICxmitBuf.reg_ID = PIC_status[current_PIC_index].curr_ID_reg;
work_pulse_index = (PIC_status[current_PIC_index].PIC_addr *
PULSES_PER_PIC) + PICxmitBuf.reg_ID;
for (i=0; i<ID_FIELD_LENGTH; i++)
{
    if (pulseData[work_pulse_index].ID_buffer[i] == 0)
    {
        j = i - 1;
        i = ID_FIELD_LENGTH;
    }
}

if (j<0)
{
    j = 0;
}
else if (j > (ID_FIELD_LENGTH-4))
{
    j = ID_FIELD_LENGTH-4;
}

PICxmitBuf.command_data = READ_ENCODER_ID | (j << 8);
format_xmit_buffer();
}
#endif

}

if (need_pulse_read)
{
#ifdef IS_MC5
    if (PIC_status[current_PIC_index].software_type == 7)
    {
        i = PIC_status[current_PIC_index].PIC_addr - LCM_BASE_ADDR; /* get
LCM # */
        work_bit_mask = (load_shed_state[0] >> (9 * i)) and 0x000001ff;
        PICxmitBuf.command_data = 0;
        for (i=0; i<9; i++)
        {
            if ((work_bit_mask and 0x100) != 0)
                PICxmitBuf.command_data = (PICxmitBuf.command_data << 2) + 1; /* lsb
turns relay on */
            else
                PICxmitBuf.command_data = (PICxmitBuf.command_data << 2) + 2; /* msb
turns relay on */
            work_bit_mask = work_bit_mask << 1;
        }
        PICxmitBuf.command_data |= 0x32000000; /* Set pulse time = 50 mS */
    }
}

```

```

        PICxmitBuf.command = WRITE_REGISTER;
        PICxmitBuf.reg_ID = OUTPUT_REG;
        format_xmit_buffer();
        need_pulse_read = FALSE;
    }
    else
#endif
        if (PIC_status[current_PIC_index].num_pulse_regs == 0)
        {
#ifdef IS_ST5
            PICxmitBuf.command = READ_REGISTER;
            PICxmitBuf.reg_ID = VERS_REG;
            format_xmit_buffer();
            need_pulse_read = FALSE;
#endif
        }
        else
        {
#ifdef IS_MC5
            if ( (((PIC_status[current_PIC_index].PIC_addr * PULSES_PER_PIC) +
                PIC_status[current_PIC_index].curr_pulse_reg) % 24) >=
num_pulse_ctrs)
                PIC_status[current_PIC_index].curr_pulse_reg = 0;
#endif
            if (PIC_status[current_PIC_index].curr_pulse_reg <= MAX_FAST_READ)
                PICxmitBuf.command = PIC_status[current_PIC_index].curr_pulse_reg;
            else
                PICxmitBuf.command = READ_REGISTER;

            PICxmitBuf.reg_ID = PIC_status[current_PIC_index].curr_pulse_reg;
            format_xmit_buffer();
            need_pulse_read = FALSE;
            if (++PIC_status[current_PIC_index].curr_pulse_reg >=
PIC_status[current_PIC_index].num_pulse_regs)
            {
                PIC_status[current_PIC_index].curr_pulse_reg = 0;
#ifdef IS_RSM
                PIC_status[current_PIC_index].update_parameter = TRUE;
#endif
            }
        }
    }

    PIC_status[current_PIC_index].last_cmd_data = PICxmitBuf.command_data;
    PIC_status[current_PIC_index].last_command = PICxmitBuf.command;

    PIC_wait_limit = PIC_status[current_PIC_index].reply_wait_limit;

    rev_PIC_index = current_PIC_index;

    if (++current_PIC_index == num_PICs)
        current_PIC_index = 0;

    if (!need_pulse_read)
        PIC_serial_status = do_send_cmd;    /* Start UART */

```



```

    }
}

#ifdef IS_RSM /* In RSM, generate serial stream from subseconds */

void pulseSubsecond(void)
{
    if (num_PICs != 0)
    {
        if (PIC_clk_state == clk_high)
        {
            /* Falling edge, just set clock low */
            fs1004.io.pulseOut1Off = 0;
            PIC_clk_state = clk_low;
        }
        else
        {
            PIC_clk_state = clk_high;
        }
    }
}

#include "pic.def" /* Do PIC serial comm on clock rising edge */

}
}

#endif

void process_status_bits(void)
{
    unsigned int    work_pulse_index, work_status_bits;

    PIC_status[rcv_PIC_index].get_status = FALSE;
    PIC_status[rcv_PIC_index].status_reg_value = PICrcvBuf.command_data >> 16;
    PIC_status[rcv_PIC_index].time_since_global = (PICrcvBuf.command_data >> 8) and
    0xff;

    if (PIC_status[rcv_PIC_index].status_reg_value != 0)
    {
        PIC_status[rcv_PIC_index].reset_status = TRUE;
        if ((PIC_status[rcv_PIC_index].status_reg_value and 0x8f10) != 0)
        {
            PIC_status[rcv_PIC_index].PIC_data_err_count++; /* PIC detected bad data */
            set_PIC_alarm(PIC_flags_0, PIC_status[rcv_PIC_index].PIC_addr,
                PIC_status[rcv_PIC_index].status_reg_value);
            work_status_bits = (PIC_status[rcv_PIC_index].status_reg_value >> 8) and 0x0f;
            work_pulse_index = PIC_status[rcv_PIC_index].PIC_addr * PULSES_PER_PIC;
            while (work_status_bits > 0)
            {
                if (work_status_bits and 0x01)
                    pulseData[work_pulse_index].error_cnt++;
            }
        }
    }
}

```

```

        work_status_bits = work_status_bits >> 1;
        work_pulse_index++;
    }

    if (((PIC_status[rcv_PIC_index].status_reg_value and 0x20) != 0) andand
        !PIC_status[rcv_PIC_index].clear_pulse_regs)
    {
        PIC_status[rcv_PIC_index].comm_error_count++; /* PIC detected comm parity err
*/
        if (PIC_status[rcv_PIC_index].comm_error_count > 2)
        {
            set_PIC_alarm(PIC_comerr_0, PIC_status[rcv_PIC_index].PIC_addr,
                PICrcvBuf.flags | 0x0800);
        }
    }
    if (((PIC_status[rcv_PIC_index].status_reg_value and 0x3080) != 0)
    {
        PIC_status[rcv_PIC_index].PIC_reset_count++; /* PIC was reset */
        if (PIC_status[rcv_PIC_index].PIC_reset_count > 1)
        {
            set_PIC_alarm(PIC_reset_0, PIC_status[rcv_PIC_index].PIC_addr,
                PIC_status[rcv_PIC_index].status_reg_value);
        }
    }
}

if (PIC_status[rcv_PIC_index].clear_pulse_regs andand
    (clear_state == GLOBAL_CLR_SENT))
{
    PIC_status[rcv_PIC_index].clear_pulse_regs = FALSE;
    if (((PIC_status[rcv_PIC_index].status_reg_value and 0x0f) != 0x0f) andand
        (PIC_status[rcv_PIC_index].software_type == 3))
    {
        PIC_status[rcv_PIC_index].PIC_data_err_count++; /* Clear cmd failed */
        set_PIC_alarm(PIC_flags_0, PIC_status[rcv_PIC_index].PIC_addr,
            PIC_status[rcv_PIC_index].status_reg_value);
    }
}
else if (((PIC_status[rcv_PIC_index].status_reg_value and 0x0f) != 0) andand
    ((PIC_status[rcv_PIC_index].status_reg_value and 0xf10) == 0))
{
    PIC_status[rcv_PIC_index].PIC_data_err_count += 1000; /* Spurious clr cmd */
    set_PIC_alarm(PIC_flags_0, PIC_status[rcv_PIC_index].PIC_addr,
        PIC_status[rcv_PIC_index].status_reg_value);
}
}

void set_PIC_alarm(alarmcodes pass_alarm_codes,
    int pass_PIC_addr,
    unsigned int pass_alarm_data)
{
    if (!Comm_background_flags.do_alarm)
    {
        PIC_alarm.alarm_codes = pass_alarm_codes + pass_PIC_addr;
    }
}

```

```

    PIC_alarm.ack = pass_alarm_data;
    Comm_background_flags.do_alarm = TRUE;
}

}

unsigned char workTpar;

BOOLEAN decode_rcv_buffer(void)
{
    int i;
    unsigned int workInt;

    workTpar = 0x1f;

    if (PICrcvBuf.flags == 0)
    {
        work_PIC_addr = Get5();
        if (PICrcvBuf.flags == 0)
            PICrcvBuf.PIC_addr = work_PIC_addr; /* Valid address received */
        PICrcvBuf.command = Get5();
        if (PICrcvBuf.command > 0x0f)
            PICrcvBuf.reg_ID = Get5();
        else
            PICrcvBuf.reg_ID = PICrcvBuf.command;
        PICrcvBuf.command_data = 0;
        for (i=0; i<6; i++)
            PICrcvBuf.command_data = (PICrcvBuf.command_data << 5) + Get5();
        workInt = Get5();
        PICrcvBuf.command_data = (PICrcvBuf.command_data << 2) + (workInt >> 3);
        PICrcvBuf.flags |= workInt and 0x07;
        workInt = Get5();
        if (workTpar != 0)
            PICrcvBuf.flags |= rcv_tpar_error_flag;
    }
    /* Look up correct PIC_status index for data in rcv buffer */
    for (i=0; ((i<MAX_PICS) andand (PIC_status[i].PIC_addr != PICrcvBuf.PIC_addr)); i++);

    if (i >= num_PICs)
        PICrcvBuf.flags |= rcv_bad_addr_flag; /* past end of valid PICs, address error */

    if (i == MAX_PICS)
        rcv_PIC_index = 0;
    else
        rcv_PIC_index = i;

#ifdef VATEST
    #if VATEST==1
        PICrcvBuf.command = temp_command;
        PICrcvBuf.reg_ID = temp_reg_ID;
        PICrcvBuf.command_data = temp_command_data;
        PICrcvBuf.flags = temp_flags;
    #endif
#endif
}

```

```

return((PICrcvBuf.flags and
~(PIC_error_flag | PIC_need_refresh_flag | PIC_invalid_data_flag)) == 0);
}

void format_xmit_buffer(void)
{
    int i,j,ShiftCount;
    unsigned int WorkInt;
    unsigned long WorkData;

    PICxmitBuf.bitCount = 0;
    PICxmitBuf.bits.bufLong[0] = 0;
    PICxmitBuf.bits.bufLong[1] = 0;
    PICxmitBuf.bits.bufLong[2] = 0; /* Clear xmit bit buffer */

    workTpar = 0x1f; /* Initializa Tpar */

    Stuff6(PICxmitBuf.PIC_addr);
    Stuff6(PICxmitBuf.command);
    if (PICxmitBuf.command > 0x0f)
    {
        Stuff6(PICxmitBuf.reg_ID);
        if (PICxmitBuf.command > 0x17)
        {
            WorkData = PICxmitBuf.command_data;
            ShiftCount = 32;
            for (i = 0, i < 7; i++)
            {
                WorkInt = 0;
                for (j = 0; j < 5; j++)
                {
                    WorkInt = WorkInt << 1;
                    if (WorkData and 0x80000000)
                        WorkInt += 1;
                    WorkData = WorkData << 1;
                    if (--ShiftCount == 0)
                    {
                        WorkInt = (WorkInt << 3) + (PICxmitBuf.flags and 0x07);
                        j = 5;
                    }
                }
                Stuff6(WorkInt);
            }
        }
    }
    Stuff6(workTpar); /* Buffer set, set up UART control */
    PICrcvBuf.bitCount = 66;

    if ((PIC_comm_mon.control_flags and XMIT_DATA_RDY) == 0)
    {
        PIC_comm_mon.xmit_buf = PICxmitBuf;
        PIC_comm_mon.control_flags |= XMIT_DATA_RDY;
    }
}

```

```

PICrcvBuf.flags = 0;
PICrcvBuf.PIC_addr = PICxmitBuf.PIC_addr; /* preset address of remote PIC */
}

```

```

unsigned int Get5(void)

```

```

{
    int      byte_index, bit_index;
    unsigned int work_result;

    work_result = 0;
    if ((PICrcvBuf.flags and
        ~(PIC_error_flag | PIC_need_refresh_flag | PIC_invalid_data_flag)) == 0)
    {
        byte_index = 96 - PICrcvBuf.bitCount;
        bit_index = byte_index and 0x07;
        byte_index = byte_index >> 3;
        work_result = (PICrcvBuf.bits.bufByte[byte_index] << 8) +
        PICrcvBuf.bits.bufByte[byte_index+1];
        work_result = work_result >> (10 - bit_index);
        work_result and= 0x3f;
        if (parity6[work_result >> 1] != work_result)
        {
            work_result = 0;
            PICrcvBuf.flags |= rcv_parity_err_flag;
        }
    }
    else
    {
        work_result = work_result >> 1;
        workTpar ^= work_result;
        PICrcvBuf.bitCount -= 6;
        if (PICrcvBuf.bitCount < 0)
            PICrcvBuf.flags |= rcv_bad_length_flag;
    }
}
return(work_result);
}

```

```

void Stuff6(int pass_data)

```

```

{
    int byte_index, bit_index;
    workTpar ^= pass_data;
    byte_index = PICxmitBuf.bitCount >> 3;
    bit_index = PICxmitBuf.bitCount and 0x07;
    pass_data = parity6[pass_data] << (10 - bit_index);
    PICxmitBuf.bits.bufByte[byte_index] |= pass_data >> 8;
    PICxmitBuf.bits.bufByte[byte_index+1] |= pass_data and 0xff;
    PICxmitBuf.bitCount += 6;
}

```

PULSE.H

```

/*
*/

#ifndef BOOLEAN_DEFINED
#define BOOLEAN_DEFINED
typedef int BOOLEAN;
#endif

/* subsecond processing for pulse counters */
void pulseSubsecond(void);

/* Daily PIC table rebuild */
void pulseDay(void);

/* Main PIC communication routine */
void pulseService(void);

/* PIC comm routine - once-per-second code */
void pulseSecond(void);

/* Subroutines for PIC communication */
void Stuff6(int pass_data);
void format_xmit_buffer(void);
BOOLEAN decode_rcv_buffer(void);
void process_status_bits(void);
unsigned int Get5(void);

void set_PIC_alarm(alarmcodes pass_alarm_type,
                  int pass_PIC_addr,
                  unsigned int pass_alarm_data);

/* Startup code for PIC communication */
void pulseStartup(void);

/* Routine to clear pulse registers after cold start */
void pulseColdstart(void);

/* defined in pt.c */
#define RELAY_A 0
#define RELAY_B 1
void pulseOut(int picNbr,int relayNbr,int ONoff);
void ptEveryMinute(void);
void ptEverySecond(void);
void set_PLC_relays(int xmitRelay,int rcvRelay);

```

PICEND.DEF

```
/*
include file at end of mtrsamp to control MC and ST pic chips from mtrsamp interrupt
*/
```

```
#ifdef IS_MC5
```

```
if (pulseOutMode==PIC_MC_SERIAL)
{
asm (" MOVE.L #0x80400E,A0");
asm (" AND.W #0xFCFF,(A0)"); /* Clear clock and data ports */
}
#endif
```

```
#ifdef IS_ST5
```

```
asm (" MOVE.L #0x80400E,A0");
asm (" AND.W #0xFEFF,(A0)"); /* Insert clock falling edge */
```

```
if (PIC_serial_status == sending_st5_bits)
{
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" NOP");
asm (" MOVE.L #0x80400E,A0");
asm (" OR.W #0x0100,(A0)"); /* If sending bits, activate clock*/
}
#endif
```

PICVARS.DEF

```
/*
local variables to control MC and ST pic chips from mtrsamp interrupt
meter/c/lib/picvars.def
*/
```

```
unsigned long bitOneFlag;
```

PULSELINK.DEF

```

/* determine if variables are external or defined here */
#undef ref
#ifdef PSTRU_DEFINED
#define ref
#else
#define ref extern
#endif

```

```

#if (NUMPH > 6)
#define IS_MC5
#else
#ifdef MAX_SCAN_METERS
#define IS_ST5
#else
#define IS_RSM
#endif
#endif

```

```

/* Values for hdwr -p (pulseOutMode) */

```

```

#define STANDARD_PIC_COMM 0

```

```

#if 0 /* remove these equates from mtrlink.def */
#define PIC_PULSE_SERIAL 1
#define PIC_MC_SERIAL 2
#define PIC_ST_SERIAL 3
#endif /* then uncomment them here */

```

```

/* #define VATEST 1 */

```

```

#ifdef VATEST

```

```

#undef IS_MC5
#undef IS_RSM
#undef IS_ST5

```

```

#define IS_RSM
#undef NUMPH
#define NUMPH 3
#undef MAX_SCAN_METERS

```

```

#endif

```

```

/* Uncomment for water meters (Oakville Hydro) */
#define READ_ENCODER_ID 0x49ff0000
#define READ_ENCODER_PARM 0x0312500c

```

```

/* Uncomment for gas meters (Sonix)
#define READ_ENCODER_ID 0x49fe0389

```



```

#define READ_ENCODER_PARM 0x0300500c
    */

#define ID_FIELD_LENGTH 13

#define PULSES_PER_PIC    4

#ifdef IS_MC5
#define BITS_PER_SEC      822
#define MAX_PICS         16
#else
#ifdef IS_ST5
#define BITS_PER_SEC      942
#define MAX_PICS          5
#else
#define BITS_PER_SEC     32
#define MAX_PICS          1
#endif
#endif

#define DIRECT_PIC_WAIT_LIMIT 24
#define BUFFERED_PIC_WAIT_LIMIT 10 * BITS_PER_SEC

#define NUM_START_BITS    12

#ifdef IS_ST5
#define REBUILD_DELAY 30000
#else
#define REBUILD_DELAY 1000
#endif

typedef
    struct {
        unsigned long    serial_number;
        unsigned long    last_cmd_data;
        unsigned char     PIC_addr;
        unsigned char     software_type;
        unsigned char     software_version;
        unsigned char     loop_rate;
        unsigned int      status_reg_value;
        unsigned char     time_since_global;
        unsigned char     num_pulse_regs;
        unsigned char     curr_pulse_reg;
        unsigned char     curr_ID_reg;
        unsigned char     last_command;
        unsigned int      reply_wait_limit;
        unsigned int      PIC_data_err_count;
        unsigned int      PIC_reset_count;
        unsigned int      comm_error_count;
        unsigned          force_pulse_read: 1;
        unsigned          get_serno       : 1;
        unsigned          get_version     : 1;
        unsigned          get_status      : 1;
        unsigned          reset_status    : 1;
    }

```

```

        unsigned    update_parameter: 1;
        unsigned    clear_pulse_regs: 1;
    }
        PIC_status_t;

typedef
    enum {
        sending_bits,
        sending_st5_bits,
        rcving_bits,
        do_PIC_start,
        sending_PIC_start,
        do_send_cmd,
        sending_start,
        sending_start_1,
        sending_stop,
        waiting_for_start,
        rcving_stop,
        idle
    }
        PIC_s_stat_t;

typedef
    enum {
        SET_PRE_ALIGN,
        DO_ALIGN,
        SET_NORMAL_PLG,
        DISCONNECT_COUPLER,
        READ_XMIT_LEVEL
    }
        ATM_op_t;

typedef
    enum {
        DO_PRESET,
        DO_PLG_SET,
        SET_ALIGN,
        WAIT_ALIGN,
        DO_ATM_DISCONNECT,
        ATM_DONE
    }
        ATM_state_t;

typedef
    struct{
        unsigned char  PIC_addr;
        unsigned char  command;
        unsigned char  reg_ID;
        unsigned long  command_data;
        unsigned int   flags;
        int            bitCount;
        union {
            unsigned long bufLong[3];
            unsigned char bufByte[12];
        }
    }

```

```

    } bits;
}
PICbitBuf;

#ifndef BOOLEAN_DEFINED
#define BOOLEAN_DEFINED
typedef int BOOLEAN;
#endif

/* Definition of bits in Flags field of PIC comm buffers */

/* NOTE- Top 4 bits (0xf000) reserved for use by ATM code */

#define rcv_bad_addr_flag    0x0100
#define rcv_tpar_error_flag  0x0080
#define rcv_data_ready_flag  0x0040
#define rcv_bad_length_flag  0x0020
#define rcv_timeout_flag     0x0010
#define rcv_parity_err_flag  0x0008
#define PIC_error_flag       0x0004
#define PIC_need_refresh_flag 0x0002
#define PIC_invalid_data_flag 0x0001

/* Command codes for PIC communication */

#define MAX_FAST_READ          0x07
#define CLEAR_PULSE_REGISTERS 0x08
#define READ_REGISTER          0x10
#define WRITE_REGISTER         0x18
#define SYSTEM_CONTROL         0x19

/* Register ID codes for PIC communication */

#define MAX_PULSE_REG          15
#define CONTROL_REG            16
#define ERROR_REG              26
#define OUTPUT_REG             27
#define PARM_REG               28
#define VERS_REG               29
#define SERNO_REG              30
#define STATUS_REG             31

/* Special-purpose PIC addresses */

#define GLOBAL_PIC_ADDR        31
#define MC5_MUX_PIC_ADDR       30
#define ST5_MUX_PIC_ADDR       29

#define LCM_BASE_ADDR          16 /* First LCM is at address 16 */
#define ATM_BASE_ADDR          20 /* First ATM is at address 20 */
#define MAX_PULSE_PIC_ADDR     15 /* pulse counters are from 0 to 15 */

/* Bit usage in the ATM relay control word */

```

```

#define ATM_RC_XMIT_ON    0x00008000
#define ATM_RC_MEAS_XMIT  0x00004000
#define ATM_RC_MEAS_CPLR  0x00002000
#define ATM_RC_RES_SHORTED 0x00001000

#define ATM_RC_CAP_MASK   0x000003ff
#define ATM_NUM_CAP_BITS  10

#define ATM_PRESET_CODE   ATM_RC_XMIT_ON | ATM_RC_MEAS_XMIT |
ATM_RC_MEAS_CPLR
#define ATM_PLC_CODE      ATM_RC_XMIT_ON | ATM_RC_RES_SHORTED
#define ATM_DISC_CODE     0

/* Other ATM control equates */

#define ATM_MEAS_TIME     20L    /* mSec to wait between cap change and ADC
reading */
#define ATM_DISC_TIME     20L    /* mSec to wait before switching cap relays */
#define ATM_MIN_READING   0x0038 /* If reading at end of autotune less than this, fail
*/

/* Equates for ST5 comm */

#define ST5_PIC_DELAY     66

#pragma region ("ram=ram")

ref unsigned short      current_PIC_index, rcv_PIC_index;
ref PIC_status_t  PIC_status[MAX_PICS];

struct
{
    unsigned long  reading;
    int            error_cnt;
    unsigned char  ID_field[ID_FIELD_LENGTH];
    unsigned char  ID_buffer[ID_FIELD_LENGTH];
    } ref    pulseData[2*NUMPH];

struct
{
    PICbitBuf      xmit_buf;
    PICbitBuf      rcv_buf;
    int            control_flags;
    } ref    ext_comm, PIC_comm_mon;

struct
{
    ATM_op_t      operation;
    unsigned int  ATM_number;
    ATM_state_t   state;
    unsigned      request_flag : 1;
    unsigned      done_flag : 1;
    unsigned      error_flag : 1;
    unsigned      active_flag : 1;
    unsigned int  start_cap_code;
    unsigned int  end_cap_code;
    unsigned int  optimum_cap_code;

```

```

unsigned int optimum_reading;
unsigned int xmit_level;
unsigned int coupler_level;
unsigned int result_flags;
long int align_timer;
    } ref ATM_control;

struct
{
    unsigned request_flag : 1;
    unsigned done_flag : 1;
    unsigned error_flag : 1;
    unsigned active_flag : 1;
    unsigned int xmit_mask;
    unsigned int rcv_mask;
    unsigned int result_flags;
    unsigned int last_xmit_mask;
    unsigned int last_rcv_mask;
    } ref MUX_control;

struct
{
    unsigned do_rebuild : 1;
    unsigned do_alarm : 1;
    } ref Comm_background_flags;

/* Bit definitions in ext_buf.control_flags */
#define SUPPRESS_NORMAL_COMM 0x0001
#define XMIT_REQUEST 0x0002 /* External routine requests send of xmit_buf */
#define CPY_RCV_DATA 0x0004 /* External cmd was sent, copy rcvd buffer to
rcv_buf */
#define RCV_DATA_RDY 0x0008 /* Got reply from external cmd, reply is in
rcv_buf */
#define XMIT_DATA_RDY 0x0010 /* xmit_buf has copy of last cmd sent */

ref PICbitBuf PICxmitBuf, PICrcvBuf, HoldrcvBuf;
ref PIC_s_stat_t PIC_serial_status;
ref unsigned int num_PICs;
ref unsigned int num_LCMs;
ref unsigned int rebuild_timer;
ref unsigned int ATM_work;
ref unsigned int work_PIC_addr;
ref unsigned int num_pulse_ctrs;
ref unsigned int hold_num_pulse_ctrs;
ref unsigned int bit_count;
ref unsigned int out_bit_value;
ref unsigned char *shift_LSB_ptr;
ref unsigned char in_bit_value;
ref unsigned int PIC_wait_limit;
ref unsigned int zero_count;
ref unsigned int last_comm_err_flag;
ref BOOLEAN dont_clear_PIC_stats;
ref int ST5_PIC_delay_ctr;
ref unsigned int PIC_bad_addr_count;
ref alarmstru PIC_alarm;

#ifdef VATEST

```

```

/* test !!!! */
ref int          temp_command;
ref int          temp_reg_ID;
ref long         temp_command_data;
ref int          temp_flags;
/* test !!!! */
#endif

ref enum {
    clk_high,
    clk_low
} PIC_clk_state;

ref enum {
    NO_GLOBAL_SEND,
    SEND_GLOBAL_CLR,
    SENDING_GLOBAL_CLR,
    GLOBAL_CLR_SENT
} clear_state;

/* Old equates maintained for compatibility */
#if 0
#define RELAY_A_BIT 1
#define RELAY_B_BIT 2
#define IDLE 0
#define TX 1
#define RX 2
#define DATA_READY 3
#define WAKE 4
#define PIC_RESET 5

#define WAKE_COUNT_RELOAD (64*4) /* Four Seconds */
#define RESET_COUNT_RELOAD 10 /* 1/4 Seconds */
#define RELAY_A_OFF_CMD 5 /*active low trigger*/
#define RELAY_A_ON_CMD 4
#define RELAY_B_OFF_CMD 7
#define RELAY_B_ON_CMD 6
#define CLEAR_ACC_CMD 8
#define SUBACC_CMD 12
#define ECHO_CMD 14
#define SLEEP_CMD 15
#define PULSE_SERIAL_COM_MAX 15
#define WAKE_CMD 16
#define RESET_CMD 17
#define PULSE_COM_NUM 18
#define NO_PULSE_BIT_DATA 0xFFFFFFFF
#define RELAY_CMD_MIN 4
#define RELAY_CMD_MAX 7
#endif

```

PULSEOUTM.DEF

```

#include "clklink.def"
#include "mtrlink.def"

#include "plc.def"
#include "serlink.def"
#include "pulselink.def"

#ifdef NO_PULSES

#include "scan.h"
#include "pulse.h"
#include "log.h"

void pulseout_second_back(void);
void pulseout_ram_init1(void);
ADDRFN pulseoutaddrfn ;

#pragma region ("data=ramInit1")
void (*pulseout_ram_init1p)(void)=pulseout_ram_init1;
#pragma region ("data=data")

#pragma region ("data=secondBack")
void (*pulseout_second_backp)(void)=pulseout_second_back ;
#pragma region ("data=data")

ADDRFN_RET pulseoutaddrfn(reg8stru reg8)
{
    /*: write the load shed event to flash */
    putEvent(Load_SHED_EVENT,1,andreg8.ulong);

    return(0);
}

void pulseout_ram_init1()
{
    addrfn_tbl[GET_FROM_SLAVE][PULSEOUT_STATE] = pulseoutaddrfn ;
}
void pulseout_second_back()
{
}
#endif

```

REIVINDICAÇÕES

1. Aparelho para medição de multi-canal de eletricidade, caracterizado pelo fato de que compreende:

5 uma cabeça de medidor que compreende um ou mais pontos medidores, a cabeça de medidor operável para medir uso de eletricidade para uma pluralidade de linhas de consumidor de eletricidade;

10 um transponder em comunicação por comunicação de linha de potência com dita cabeça de medidor e operável para transmitir dados recebidos de dita cabeça de medidor para um computador localizado remotamente, e para transmitir à cabeça de medidor, por comunicação de linha de potência, dados recebidos do dito computador localizado remotamente; e

15 um ou mais módulos de controle de carga em comunicação com dita cabeça de medidor e operáveis para atuar cada um de uma pluralidade de relés, cada relé de dita pluralidade de relés operável para serviço de conectar e desconectar em uma da pluralidade de linhas de consumidor de eletricidade.

20 2. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda um detector de fraude em comunicação com dita cabeça de medidor.

25 3. Aparelho de acordo com reivindicação 2, caracterizado pelo fato de que dito detector de fraude inclui uma luz e uma superfície reflexiva, e em que dita cabeça de medidor é operável para instruir dito um ou mais módulos de controle de carga para desconectar todas de ditas linhas de cliente se dito detector de fraude prover notificação que dita luz não é detectada refletindo de dita superfície reflexiva.

4. Aparelho de acordo com reivindicação 2, caracterizado pelo fato de que compreende ainda uma caixa contendo dita cabeça de medidor, dito um ou mais módulos de controle de carga, e ditos relés, e em que dito

detector de fraude inclui um detector de luz ambiente entrando na caixa.

5. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda um dispositivo para comparar energia de transformador à energia total usada por ditas linhas de consumidor.

5 6. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda um dispositivo para detectar fluxo de corrente inversa por ditas linhas de consumidor.

10 7. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor e um contador em comunicação com dita cabeça de medidor, dito contador correspondendo a uma linha de cliente e operável para contar regressivamente uma quantidade de energia armazenada em dita memória, e dita cabeça de medidor operável para enviar um sinal de desconexão para um módulo de controle de carga correspondente
15 para desconectar dita linha de cliente quando dito contador alcança zero.

20 8. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor, dita memória operável para armazenar um limite de carga para uma linha de cliente, e dita cabeça de medidor operável para enviar um sinal de desconexão para um módulo de controle de carga correspondente para desconectar dita linha de cliente quando dito limite de carga é excedido.

25 9. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor, dita memória operável para armazenar um limite de uso para uma linha de cliente, e dita cabeça de medidor operável para enviar um sinal de desconexão para um módulo de controle de carga correspondente para desconectar dita linha de cliente quando dito limite de uso é excedido.

10. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que dito transponder é operável para se comunicar com dito computador localizado remotamente através de linhas de potência de média tensão.

5 11. Aparelho de acordo com reivindicação 1, caracterizado pelo fato de que compreende ainda uma unidade de exibição em comunicação com dita cabeça de medidor e operável para exibir dados recebidos de dita cabeça de medidor.

10 12. Aparelho de acordo com reivindicação 11, caracterizado pelo fato de que dita unidade de exibição é operável para exibir informação relativa a consumo de energia de um cliente.

 13. Aparelho de acordo com reivindicação 11, caracterizado pelo fato de que dita unidade de exibição é operável para exibir advertências relativas a uso de energia de um cliente ou a suspeito roubo de energia.

15 14. Aparelho de acordo com reivindicação 11, caracterizado pelo fato de que dita unidade de exibição é operável para transmitir para dita cabeça de medidor informação entrada por um cliente.

20 15. Aparelho para medição remota de multi-canal de eletricidade usando comunicação de linha de potência, caracterizado pelo fato de que compreende:

uma pluralidade de pontos medidores localizados em um lado secundário de um transformador e operáveis para medir separadamente uso de eletricidade para cada uma de uma pluralidade de linhas de consumidor de eletricidade;

25 o aparelho em comunicação direta com um transponder localizado em um lado primário do transformador e operável para transmitir dados para e receber dados do transponder por comunicação de linha de potência direta, o transponder operável para transmitir dados para e para receber dados de um computador localizado remotamente;

o aparelho operável para controlar um ou mais módulos de controle de carga operáveis para atuar conexão e desconexão de cada um de uma pluralidade de relés, cada relé da pluralidade de relés correspondendo a uma da pluralidade de linhas de consumidor de eletricidade; e

5 uma caixa contendo a pluralidade de pontos medidores, o módulo de controle de carga, e os relés, em que o transformador de distribuição converte voltagens de distribuição de média tensão em voltagens de baixa tensão apropriadas para fornecer energia aos consumidores, e em que o aparelho é operável para injetar sinais em e receber sinais de linhas de
10 potência de baixa tensão que abastecem consumidores com energia elétrica; ditos sinais provendo comunicação de via dupla entre a cabeça de medidor e o transponder, e atravessando o transformador de distribuição.

16. Aparelho de acordo com reivindicação 15, caracterizado
15 pelo fato de que compreende ainda um detector de fraude em comunicação com dita cabeça de medidor.

17. Aparelho de acordo com reivindicação 16, caracterizado
 pelo fato de que dito detector de fraude inclui uma luz e uma superfície reflexiva, e em que dita cabeça de medidor é operável para instruir dito
20 módulo de controle de carga para desconectar todas de ditas linhas de cliente se dito detector de fraude prover notificação que dita luz não é detectada refletindo de dita superfície reflexiva.

18. Aparelho de acordo com reivindicação 16, caracterizado
 pelo fato do dito detector de fraude incluir um detector de luz ambiente entrando na caixa.

19. Aparelho de acordo com reivindicação 15, caracterizado
25 pelo fato de que compreende ainda um dispositivo para comparar energia de transformador à energia total usada por ditas linhas de consumidor.

20. Aparelho de acordo com reivindicação 15, caracterizado
 pelo fato de que compreende ainda um dispositivo para detectar fluxo de

tensão inversa por ditas linhas de consumidor.

21. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor e um contador em comunicação com dita cabeça de medidor, dito contador correspondendo a uma linha de cliente e operável para contar regressivamente uma quantidade de energia armazenada em dita memória, e dita cabeça de medidor operável para enviar um sinal de desconexão ao módulo de controle de carga para desconectar dita linha de cliente quando dito contador alcança zero.

22. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor, dita memória operável para armazenar um limite de carga para uma linha de cliente, e dita cabeça de medidor operável para enviar um sinal de desconexão para o módulo de controle de carga para desconectar dita linha de cliente quando dito limite de carga é excedido.

23. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de que compreende ainda uma memória legível por computador em comunicação com dita cabeça de medidor, dita memória operável para armazenar um limite de uso para uma linha de cliente, e dita cabeça de medidor operável para enviar um sinal de desconexão para o módulo de controle de carga para desconectar dita linha de cliente quando dito limite de uso é excedido.

24. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de que dito transponder é operável para se comunicar com dito computador localizado remotamente através de linhas de potência de média tensão.

25. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de que compreende ainda uma unidade de exibição em comunicação

com dita cabeça de medidor e operável para exibir dados recebidos de dita cabeça de medidor.

26. Aparelho de acordo com reivindicação 25, caracterizado pelo fato de que dita unidade de exibição é operável para exibir informação relativa a consumo de energia de um cliente.

27. Aparelho de acordo com reivindicação 25, caracterizado pelo fato de que dita unidade de exibição é operável para exibir advertências relativas a uso de energia de um cliente ou a suspeito roubo de energia.

28. Aparelho de acordo com reivindicação 25, caracterizado pelo fato de que dita unidade de exibição é operável para transmitir para dita cabeça de medidor informação entrada por um cliente.

29. Aparelho de acordo com reivindicação 15, caracterizado pelo fato do transponder ser operável para se comunicar com o computador localizado remotamente via pelo menos um de: comunicações de rádio, fibra óptica, e linhas telefônicas.

30. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de ser operável para transmitir dados diretamente ao transponder a frequências dentro da faixa de 10 a 25 kHz.

31. Aparelho de acordo com reivindicação 15, caracterizado pelo fato de ser operável para transmitir dados diretamente ao transponder a frequências correspondendo a harmônicos meio ímpares de 60 Hz.

32. Aparelho para medição de multi-canal de eletricidade, caracterizado pelo fato de que compreende:

um módulo de controle em comunicação com um circuito secundário de um transformador de distribuição, em que o transformador de distribuição converte voltagens de distribuição de média tensão em voltagens de baixa tensão apropriadas para fornecer energia aos consumidores;

o módulo de controle em comunicação direta com um transponder em comunicação com um circuito primário do transformador e

operável para transmitir dados para e receber dados do transponder por comunicação de linha de potência direta, através do transformador de distribuição, o transponder operável para transmitir dados para e para receber dados de um computador localizado remotamente;

5 uma pluralidade de módulos de medição em comunicação com o módulo de controle, cada módulo de medição operável para medir uso de eletricidade em uma de uma pluralidade de linhas de consumidor de eletricidade alimentadas pelo circuito secundário do transformador de distribuição;

10 um ou mais relés em comunicação com o módulo de controle e operáveis para atuar conexão e desconexão de eletricidade às linhas de consumidor de eletricidade; e

 uma caixa contendo o módulo de controle, os módulos de medição e os relés, em que o aparelho é operável para injetar sinais em e
15 receber sinais de linhas de potência de baixa tensão que abastecem consumidores com energia elétrica;

 ditos sinais provendo comunicação de via dupla entre a cabeça de medidor e o transponder, e atravessando o transformador de distribuição para se comunicar, pelo menos parcialmente, sobre linhas de potência de
20 média tensão, com o computador localizado remotamente.

33. Aparelho de acordo com reivindicação 32, caracterizado pelo fato do um ou mais relés serem operáveis para controlar cada fase das linhas de consumidor de eletricidade.

34. Aparelho de acordo com reivindicação 32, caracterizado
25 pelo fato do transponder ser operável para transmitir dados para e receber dados de um computador localizado remotamente via um ou mais transponders intermediários.

35. Aparelho para medição remota de multi-canal de eletricidade usando comunicação de linha de potência, caracterizado pelo fato

de que compreende:

uma pluralidade de pontos medidores localizados em um lado secundário de um transformador e operáveis para medir separadamente uso de eletricidade para cada uma de uma pluralidade de linhas de consumidor de eletricidade;

o aparelho em comunicação com um transponder localizado em um lado primário do transformador e operável para transmitir dados para e receber dados do transponder por comunicação de linha de potência, o transponder operável para transmitir dados para e para receber dados de um computador localizado remotamente;

um ou mais módulos de controle de carga em comunicação com a cabeça de medidor e operáveis para atuar cada um de uma pluralidade de relés, cada relé da pluralidade de relés operável para serviço de conectar e desconectar em uma da pluralidade de linhas de consumidor de eletricidade; e

uma caixa contendo a pluralidade de pontos medidores, o módulo de controle de carga, e os relés, em que o transformador de distribuição converte voltagens de distribuição de média tensão em voltagens de baixa tensão apropriadas para fornecer energia aos consumidores, e em que o aparelho é operável para injetar sinais em e receber sinais de linhas de potência de baixa tensão que abastecem consumidores com energia elétrica; ditos sinais provendo comunicação de via dupla entre a cabeça de medidor e o transponder, e atravessando o transformador de distribuição.

36. Aparelho para medição de multi-canal de eletricidade, caracterizado pelo fato de que compreende:

uma cabeça de medidor localizada em um lado secundário de um transformador e operável para medir separadamente uso de eletricidade para cada uma de uma pluralidade de linhas de consumidor de eletricidade;

a cabeça de medidor em comunicação direta com um transponder localizado em um lado primário do transformador operável para

transmitir dados para e receber dados do transponder por comunicação de linha de potência direta, o transponder operável para transmitir dados para e para receber dados de um computador localizado remotamente;

5 um módulo de controle de carga em comunicação com a cabeça de medidor e operável para atuar conexão e desconexão de cada um de uma pluralidade de relés, cada relé da pluralidade de relés correspondendo a uma da pluralidade de linhas de consumidor de eletricidade; e

uma caixa contendo a cabeça de medidor, o módulo de controle de carga, e os relés.

10 37. Aparelho para medição de multi-canal de eletricidade, caracterizado pelo fato de que compreende:

um módulo de controle em comunicação com um circuito secundário de um transformador de distribuição;

15 o módulo de controle em comunicação direta com um transponder em comunicação com um circuito primário do transformador e operável para transmitir dados para e receber dados do transponder por comunicação de linha de potência direta, através do transformador de distribuição, o transponder operável para transmitir dados para e para receber dados de um computador localizado remotamente;

20 uma pluralidade de módulos de medição em comunicação com o módulo de controle, cada módulo de medição operável para medir uso de eletricidade em uma de uma pluralidade de linhas de consumidor de eletricidade alimentadas pelo circuito secundário do transformador de distribuição;

25 um ou mais relés em comunicação com o módulo de controle e operáveis para atuar conexão e desconexão de eletricidade para as linhas de consumidor de eletricidade; e

uma caixa contendo o módulo de controle, os módulos de medição, e os relés.

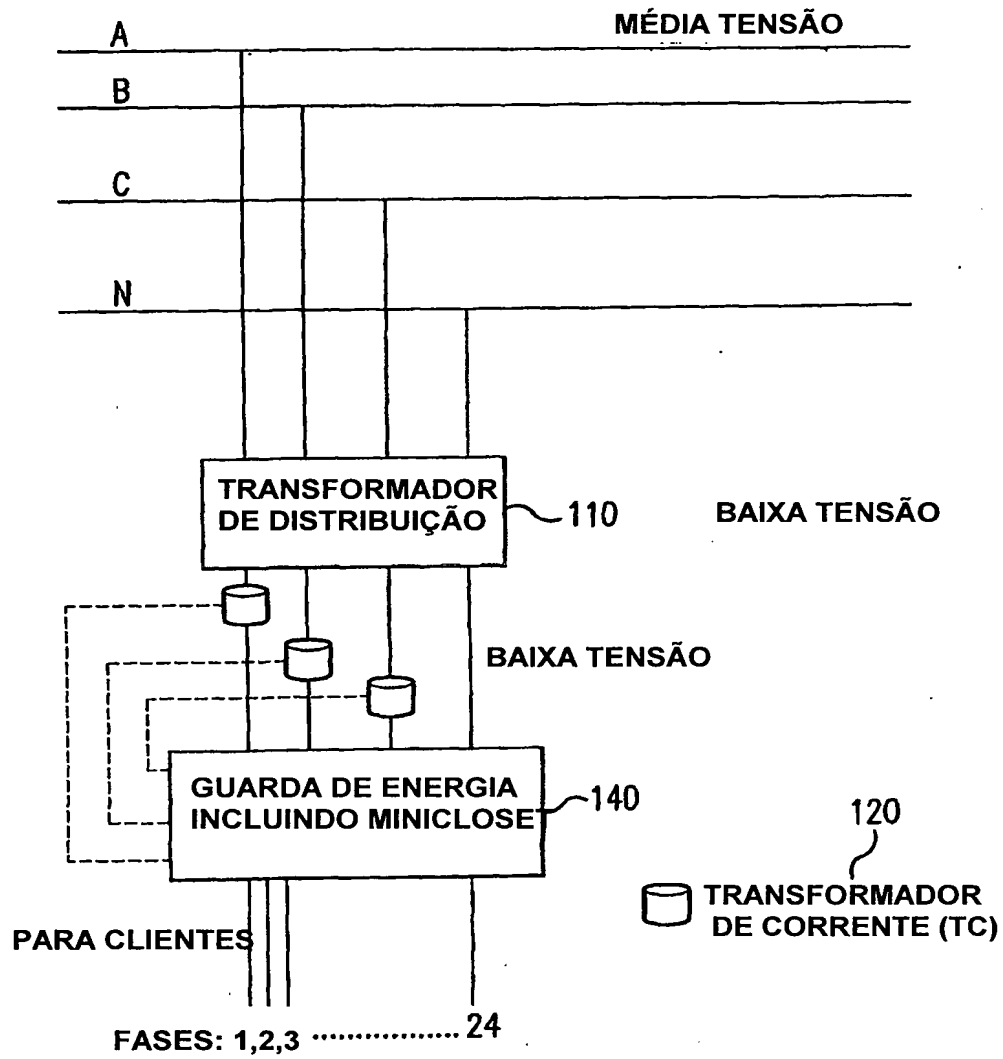


FIG.1

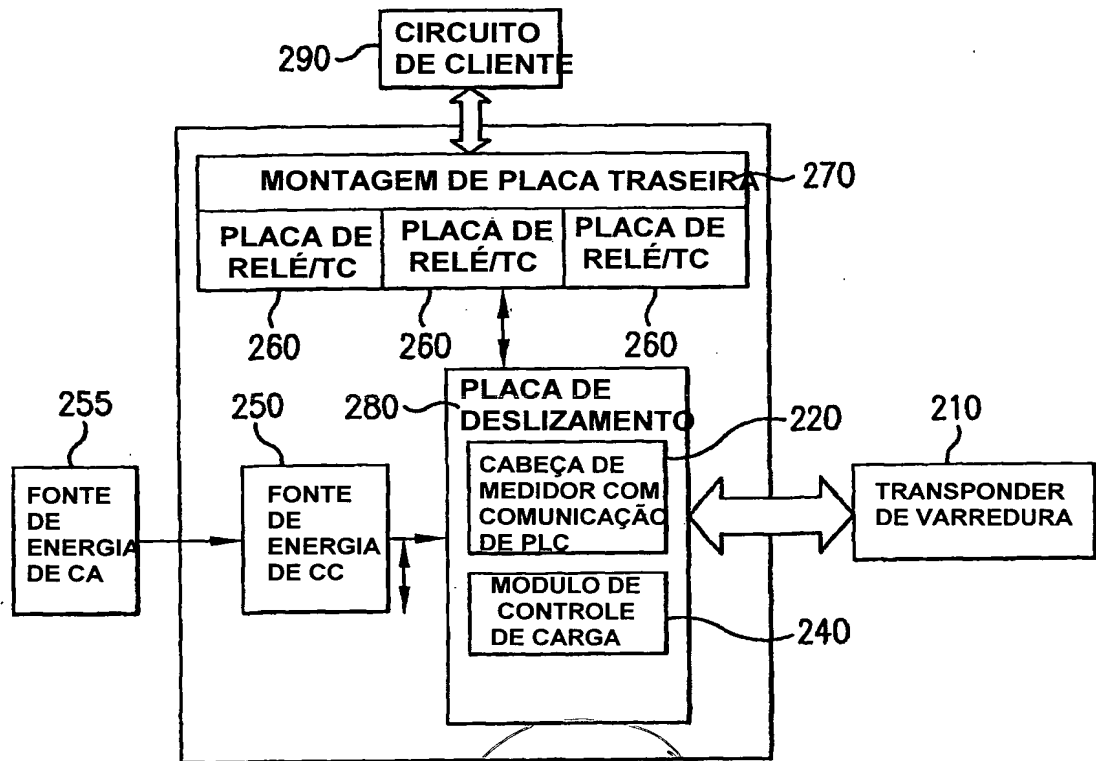


FIG. 2

FIG.3A-1	FIG.3A-2
FIG.3A-3	FIG.3A-5
FIG.3A-6	FIG.3A-8
FIG.3A-9	FIG.3A-11
FIG.3A-4	FIG.3A-7
FIG.3A-10	

FIG.3A

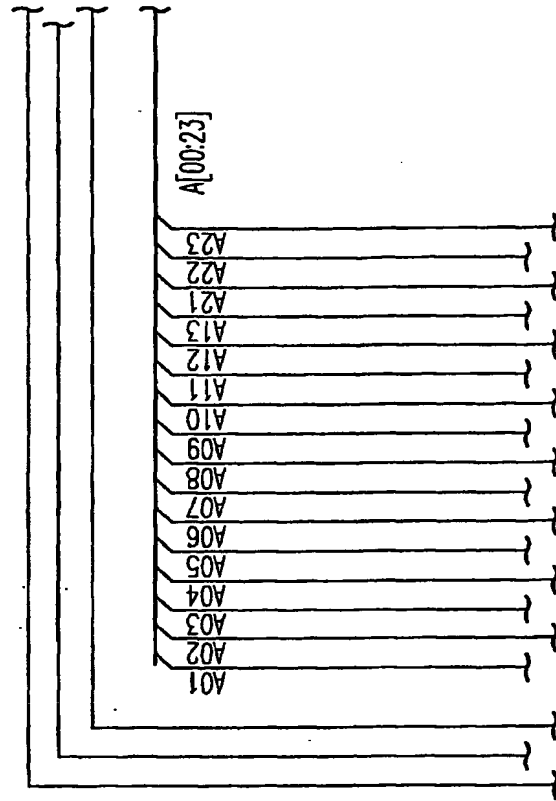


FIG.3A-1

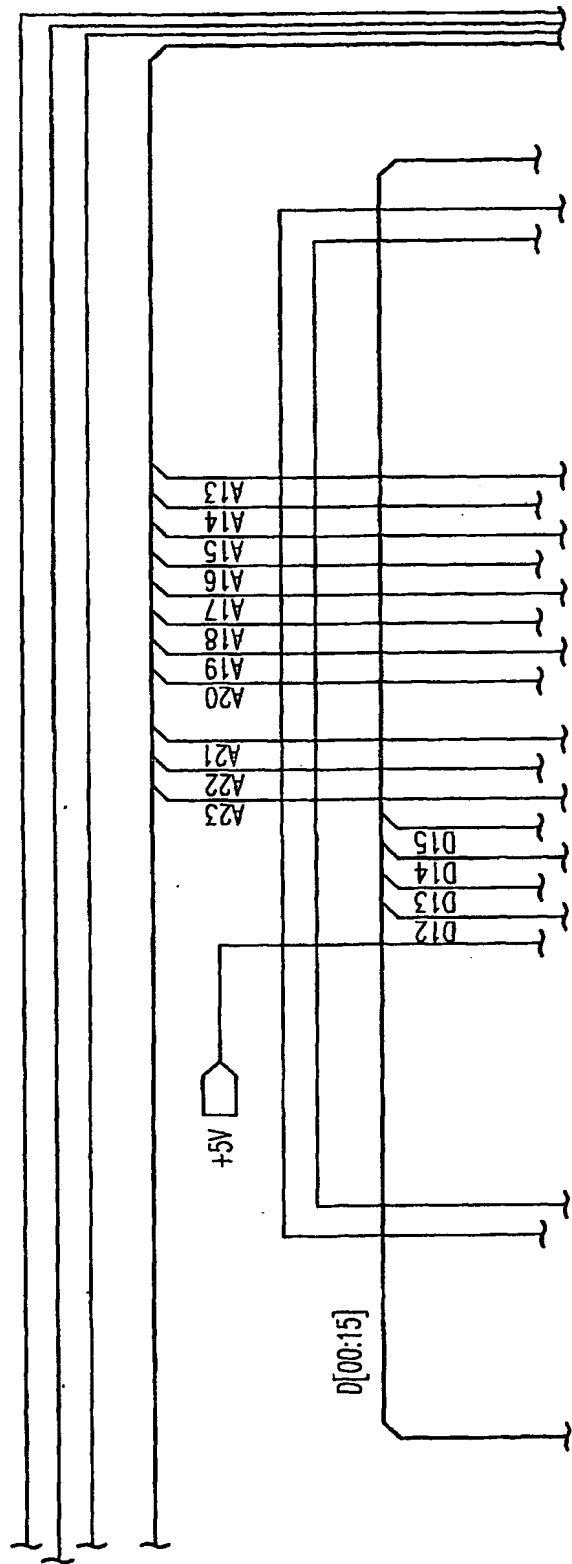


FIG. 3A-2



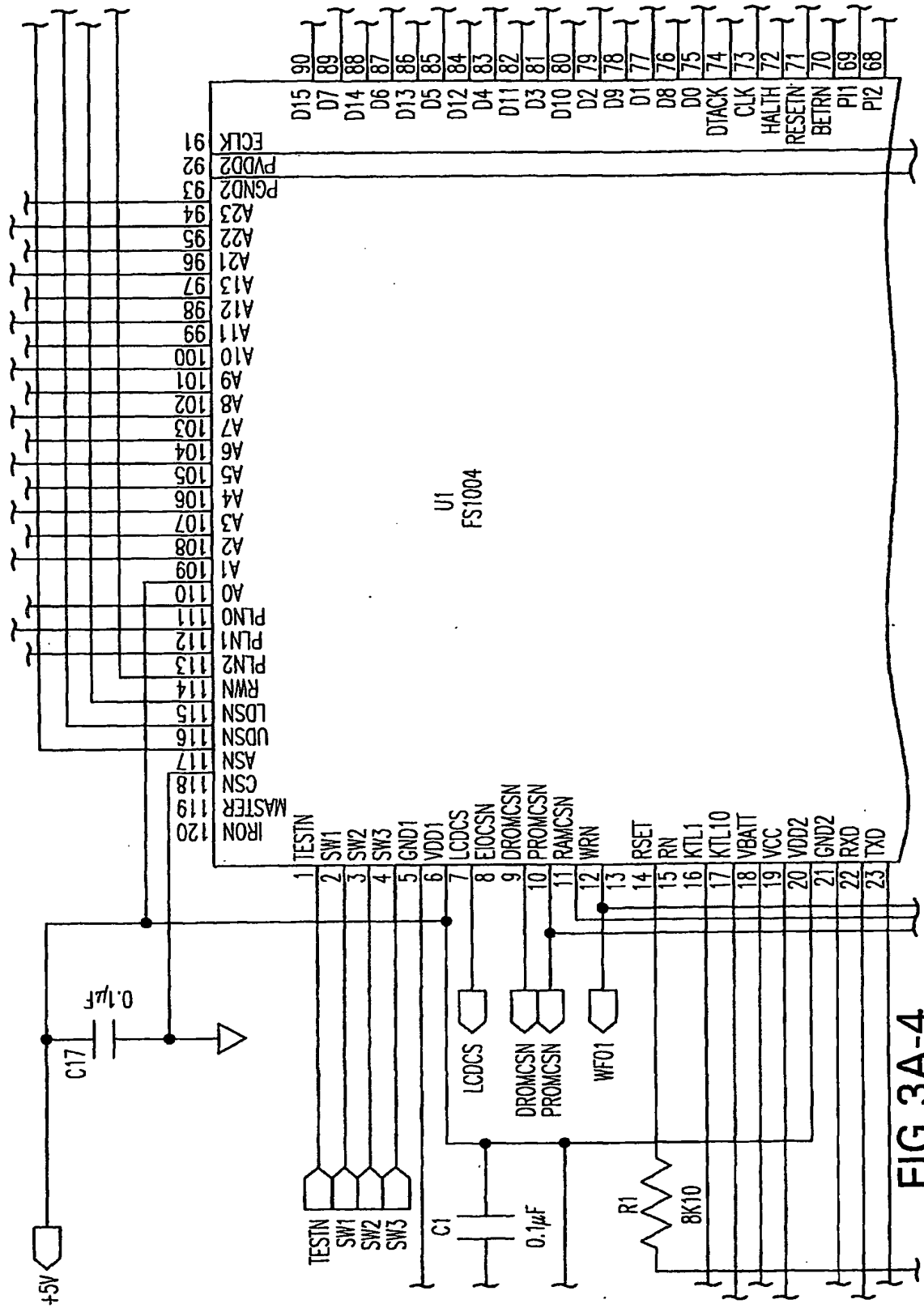


FIG.3A-4

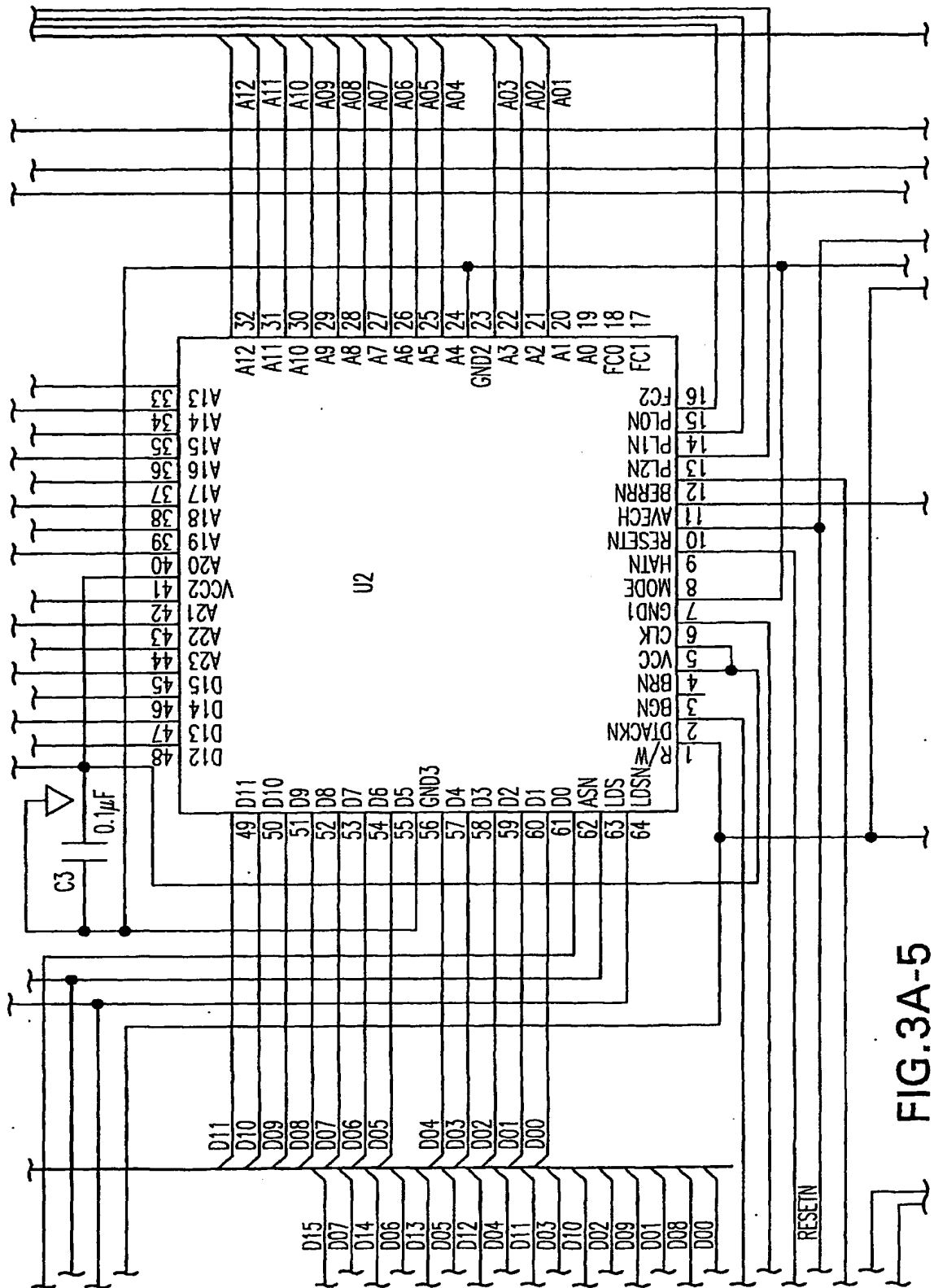


FIG.3A-5

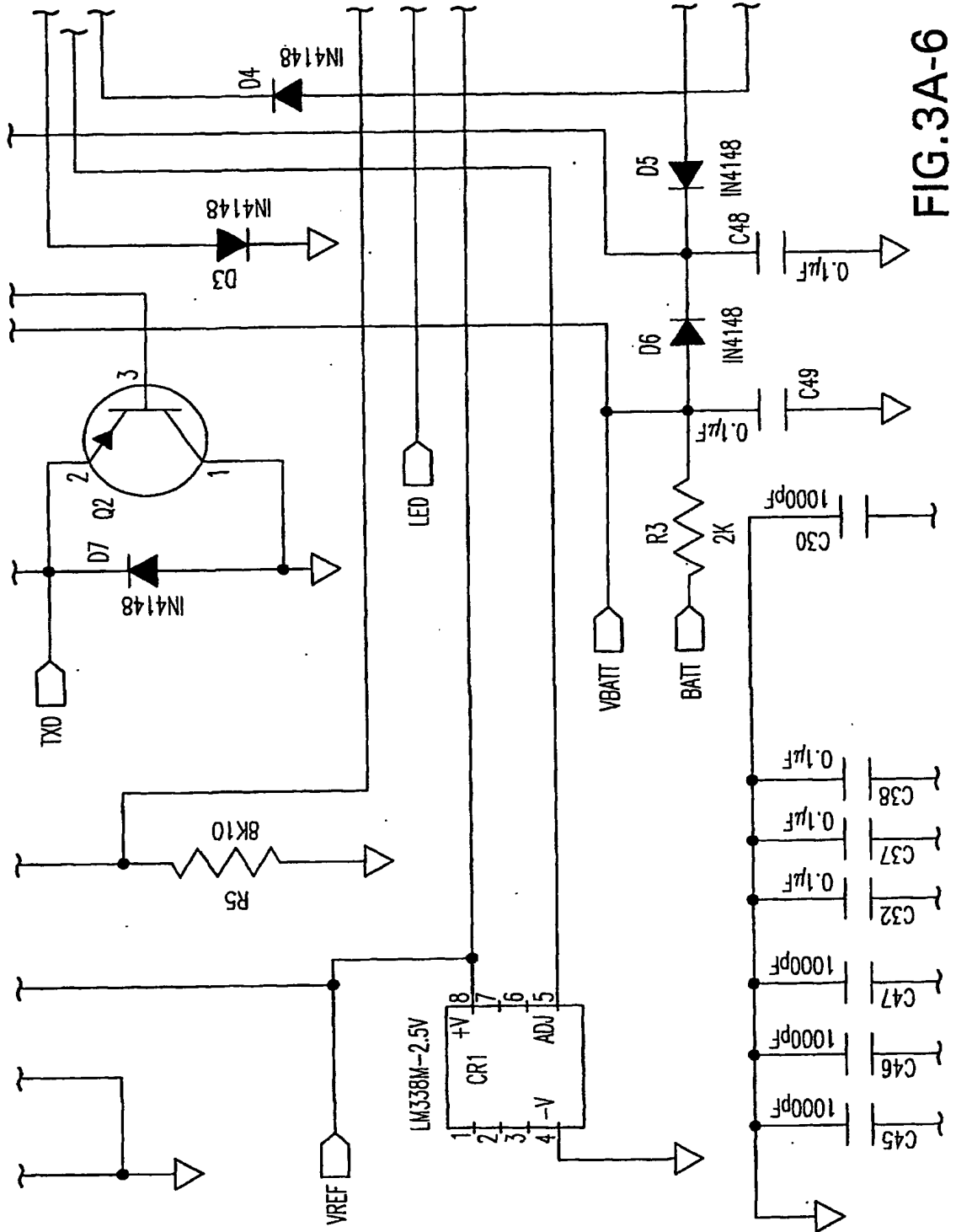


FIG. 3A-6



FIG. 3A-7



FIG. 3A-8

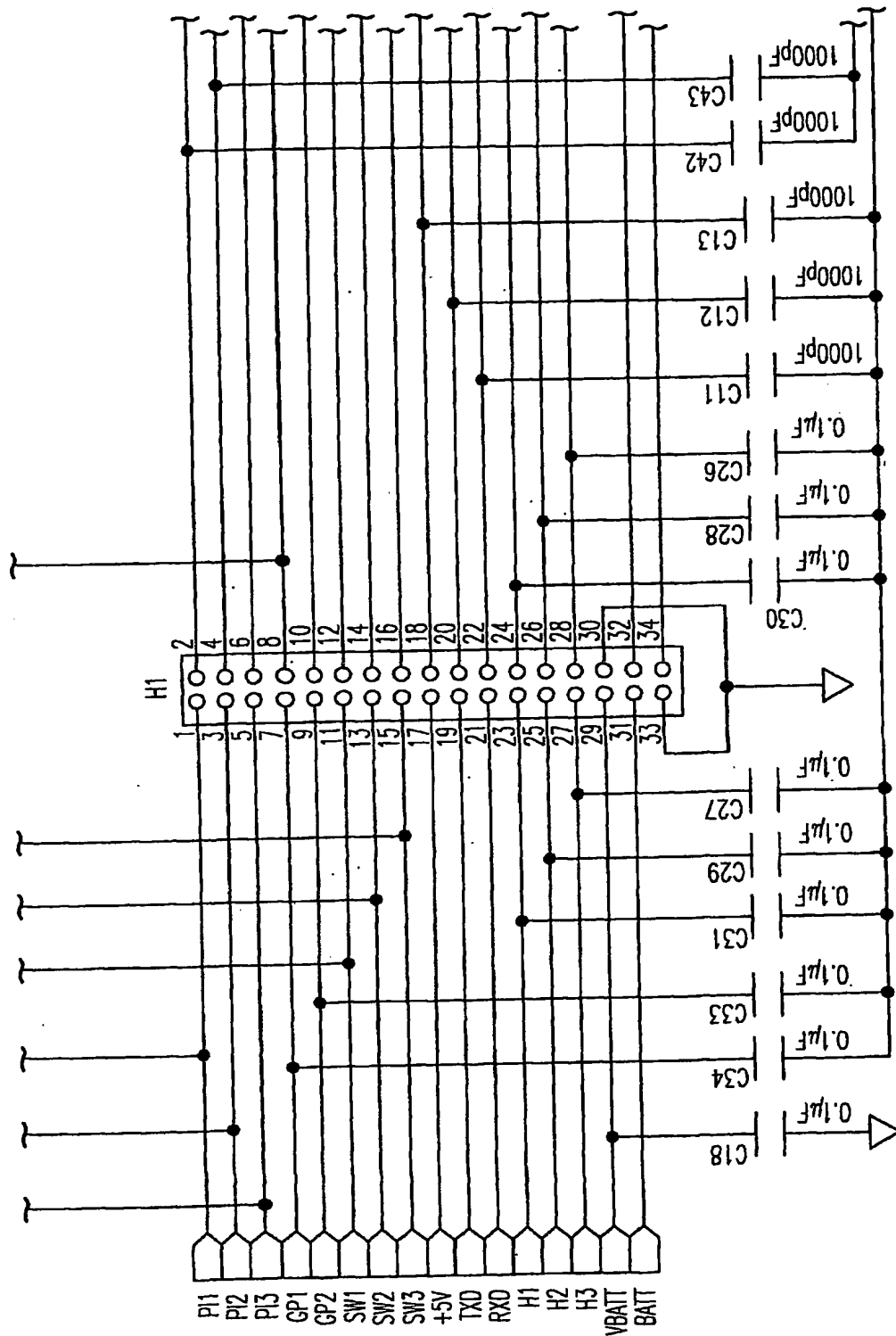


FIG.3A-9

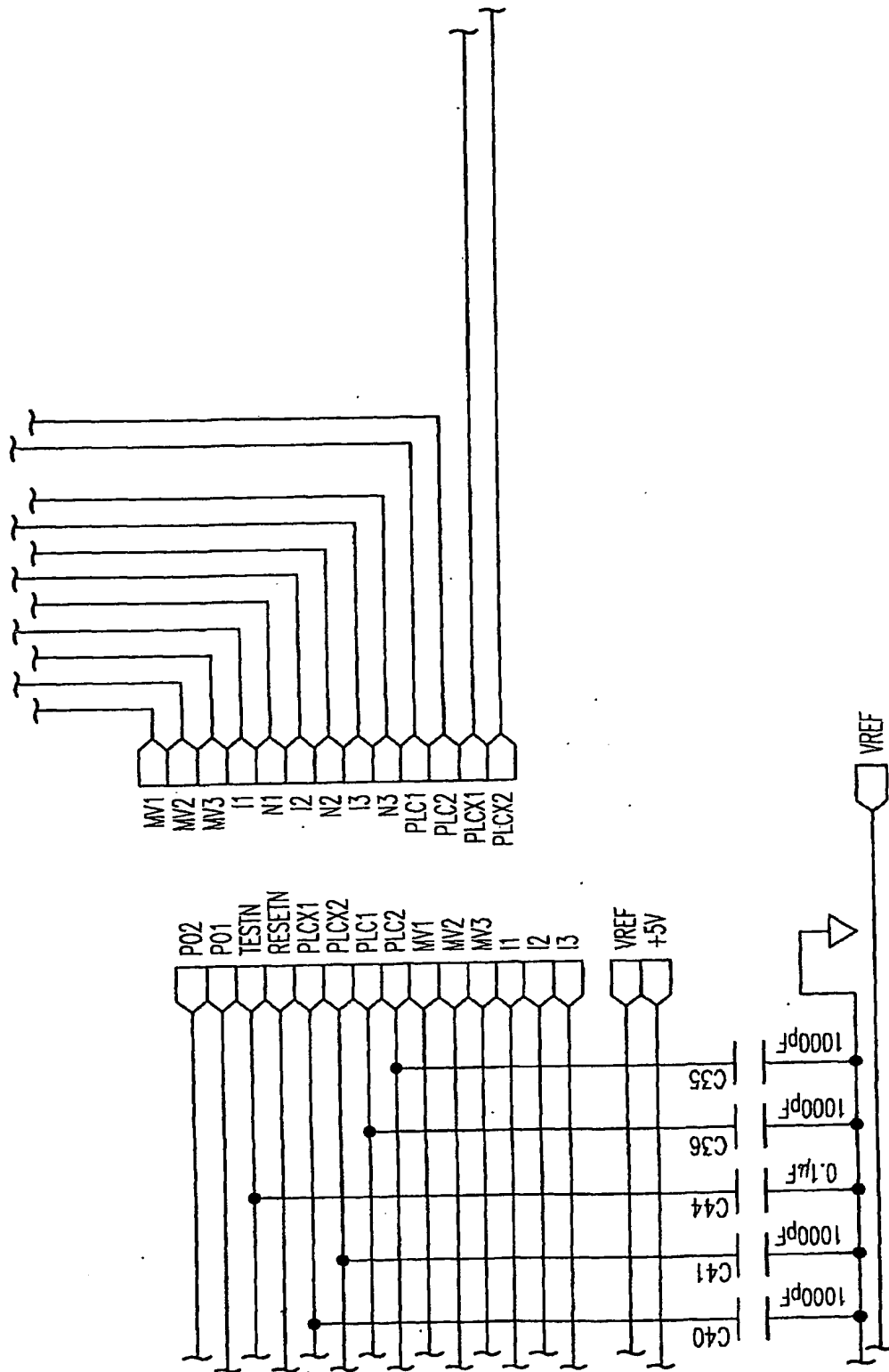


FIG.3A-10

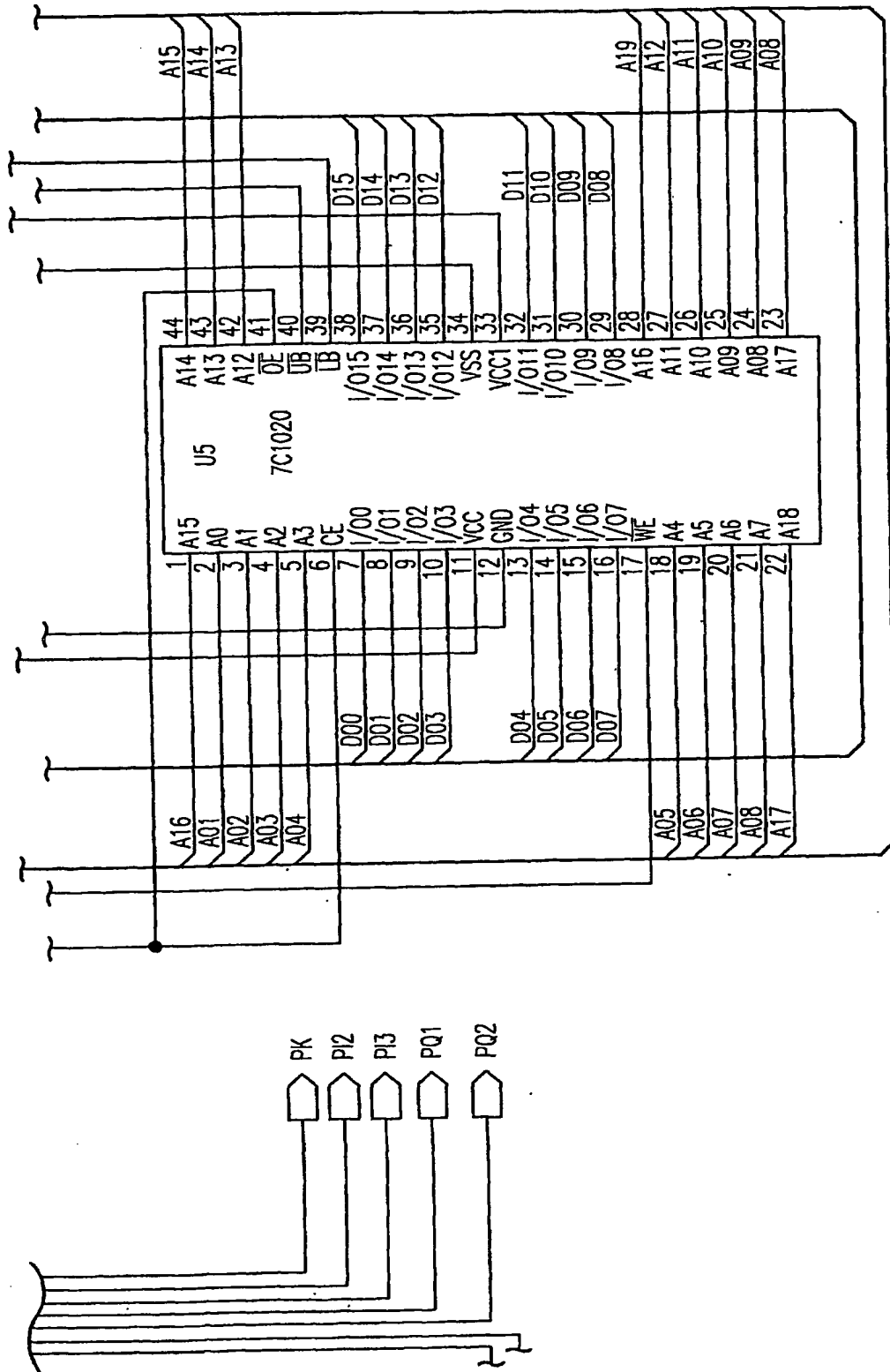


FIG.3A-11

FIG.3B-1	FIG.3B-2	FIG.3B-3	FIG.3B-4	FIG.3B-5
	FIG.3B-6	FIG.3B-7	FIG.3B-8	

FIG.3B

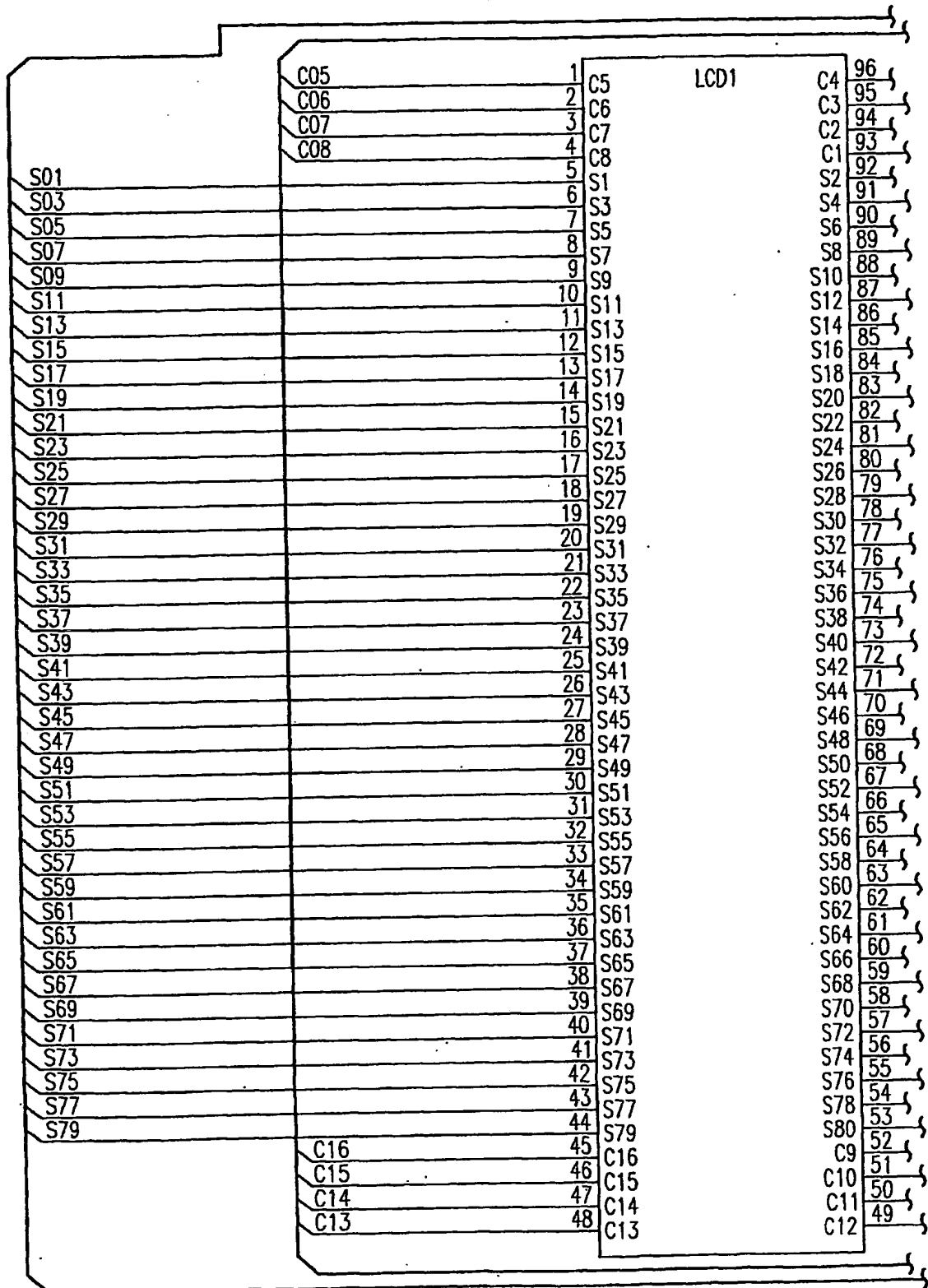


FIG.3B-1

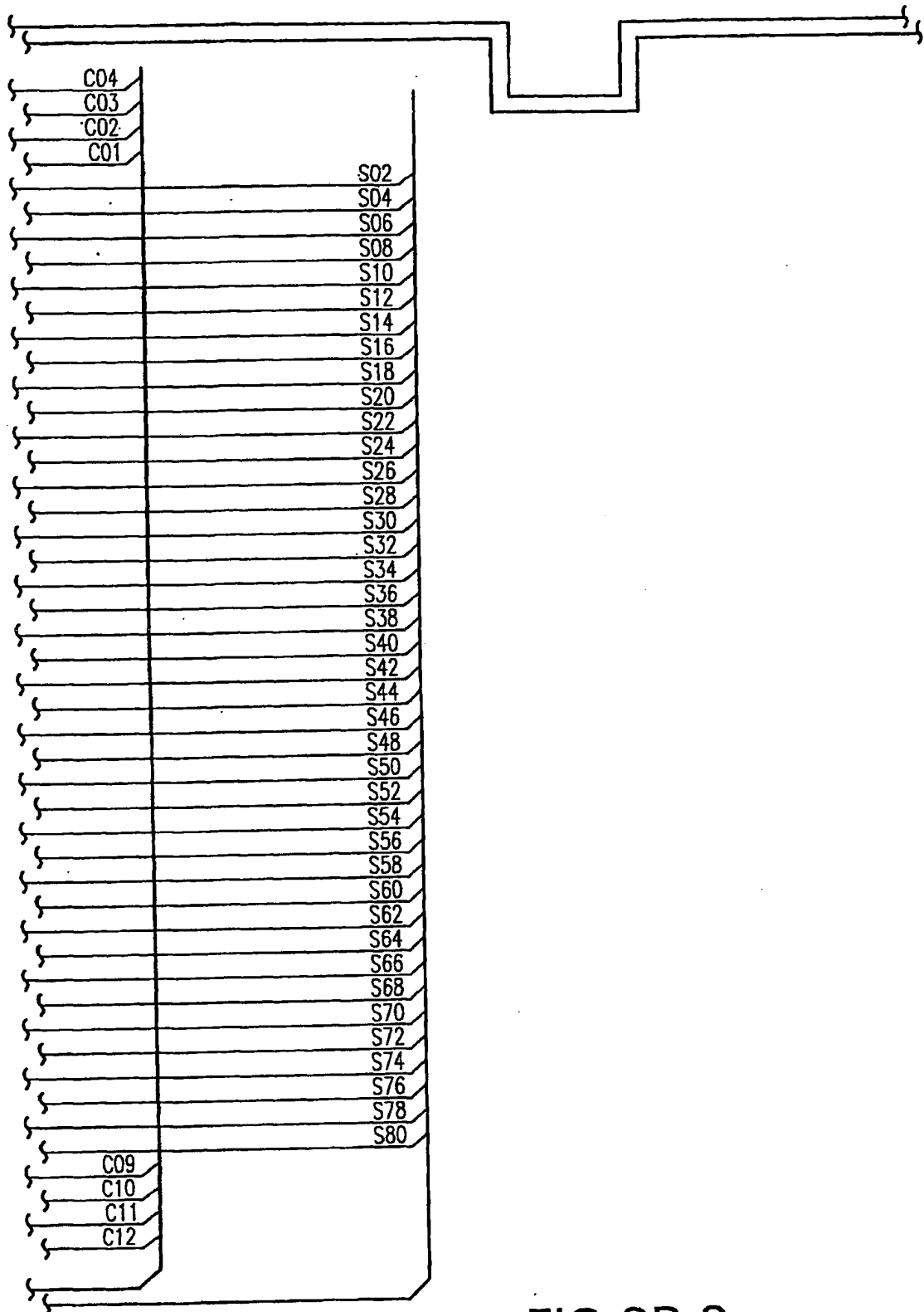


FIG.3B-2

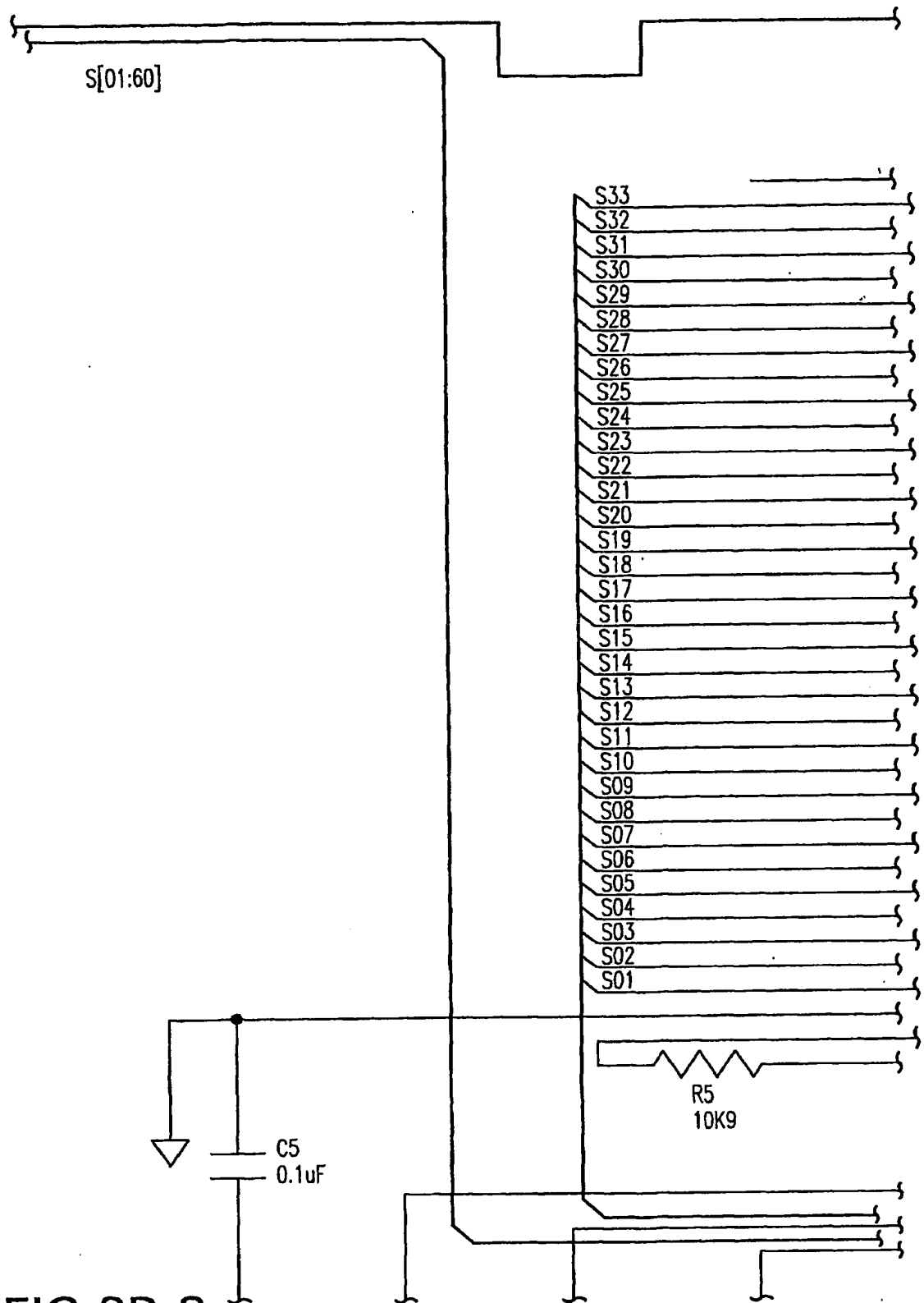
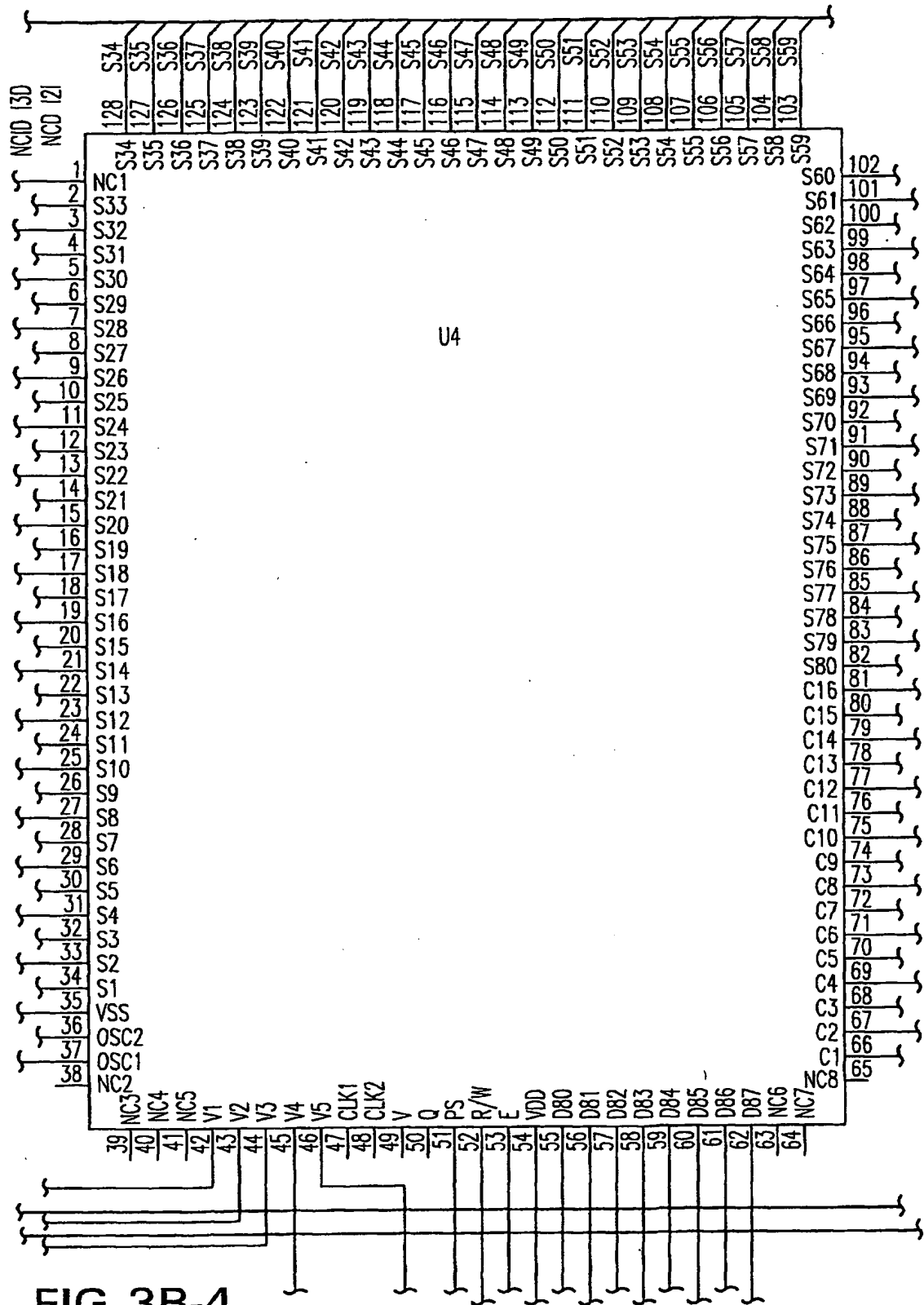


FIG.3B-3



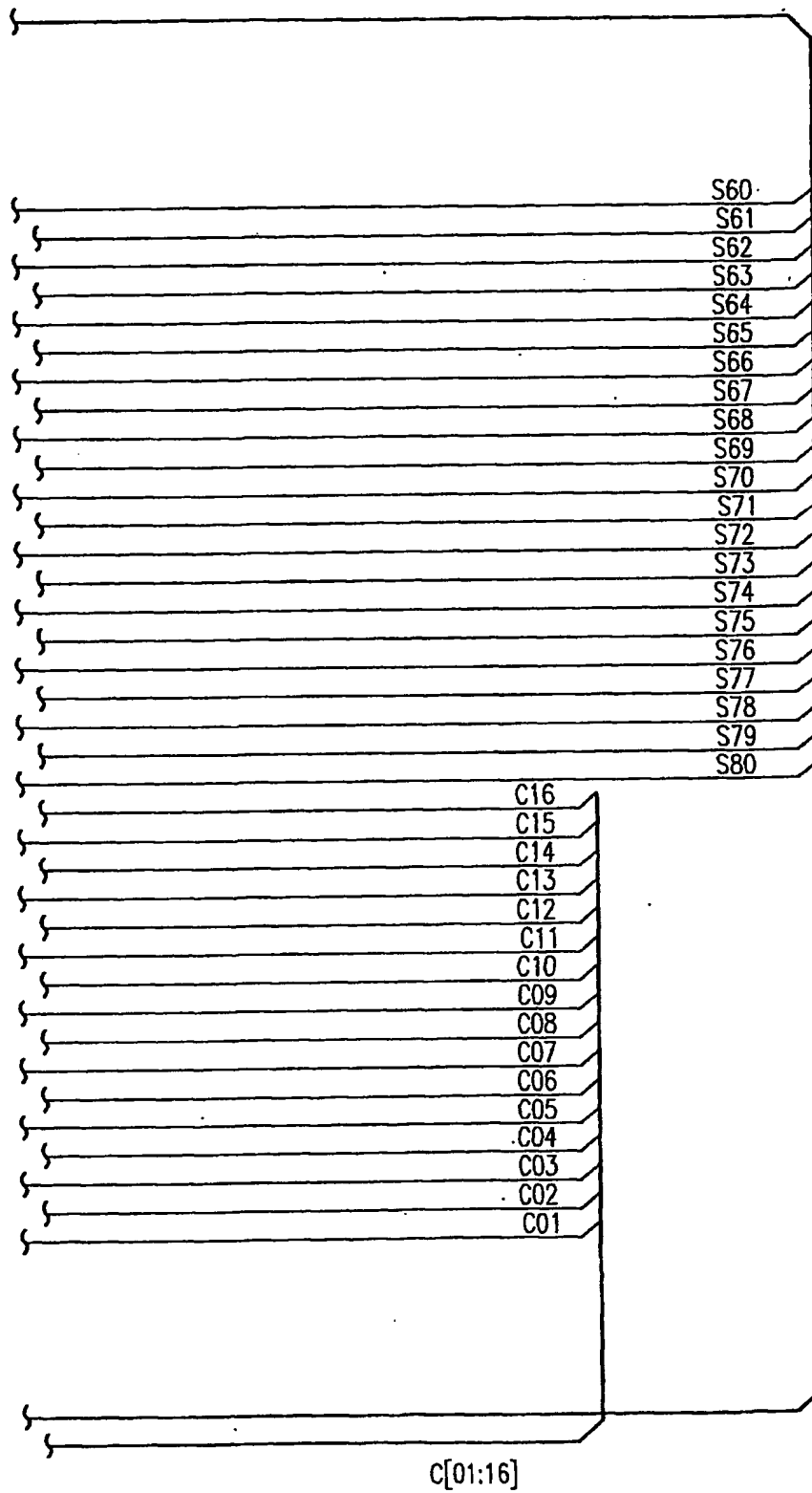


FIG.3B-5

OPCIONAL

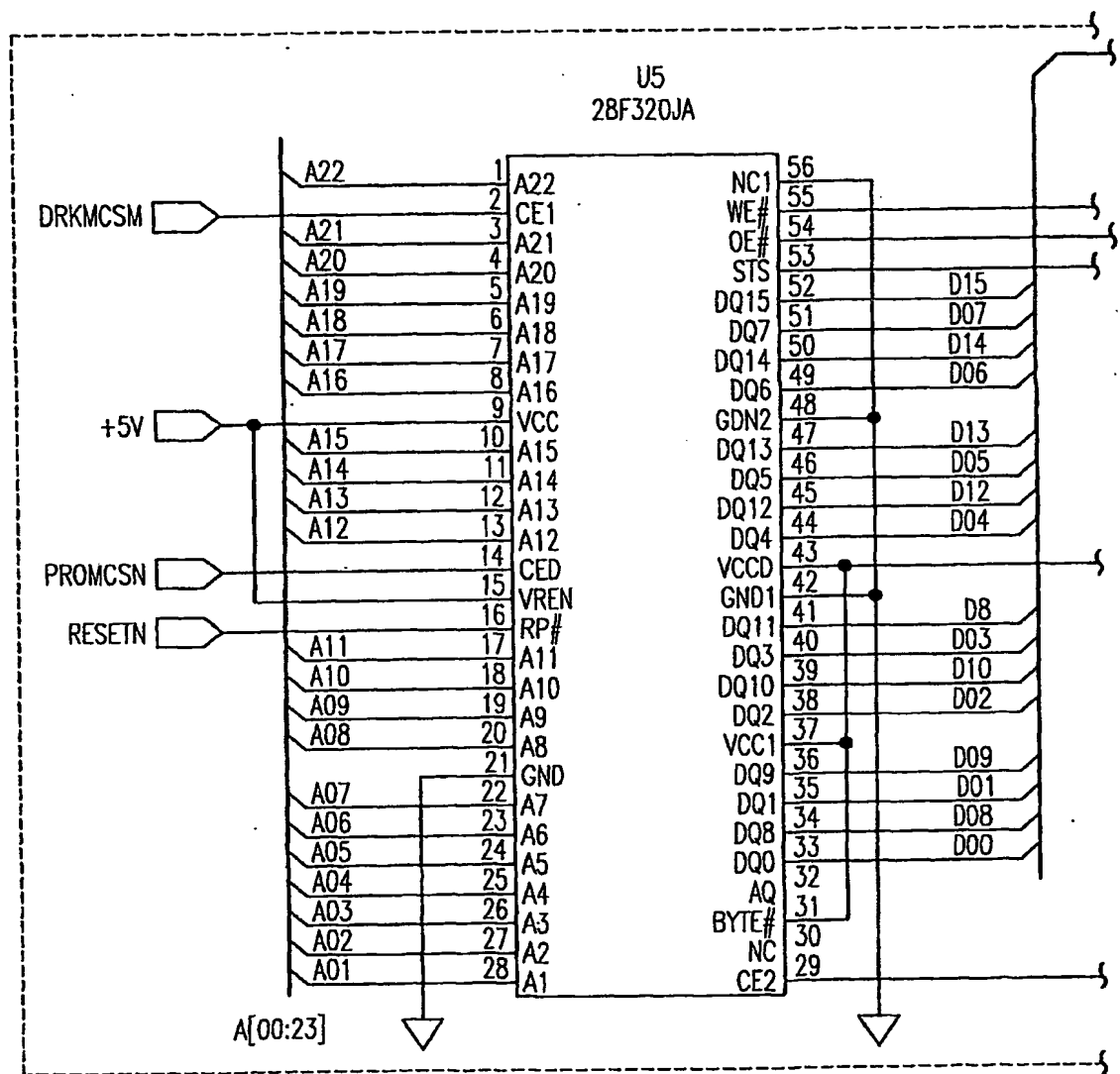


FIG.3B-6

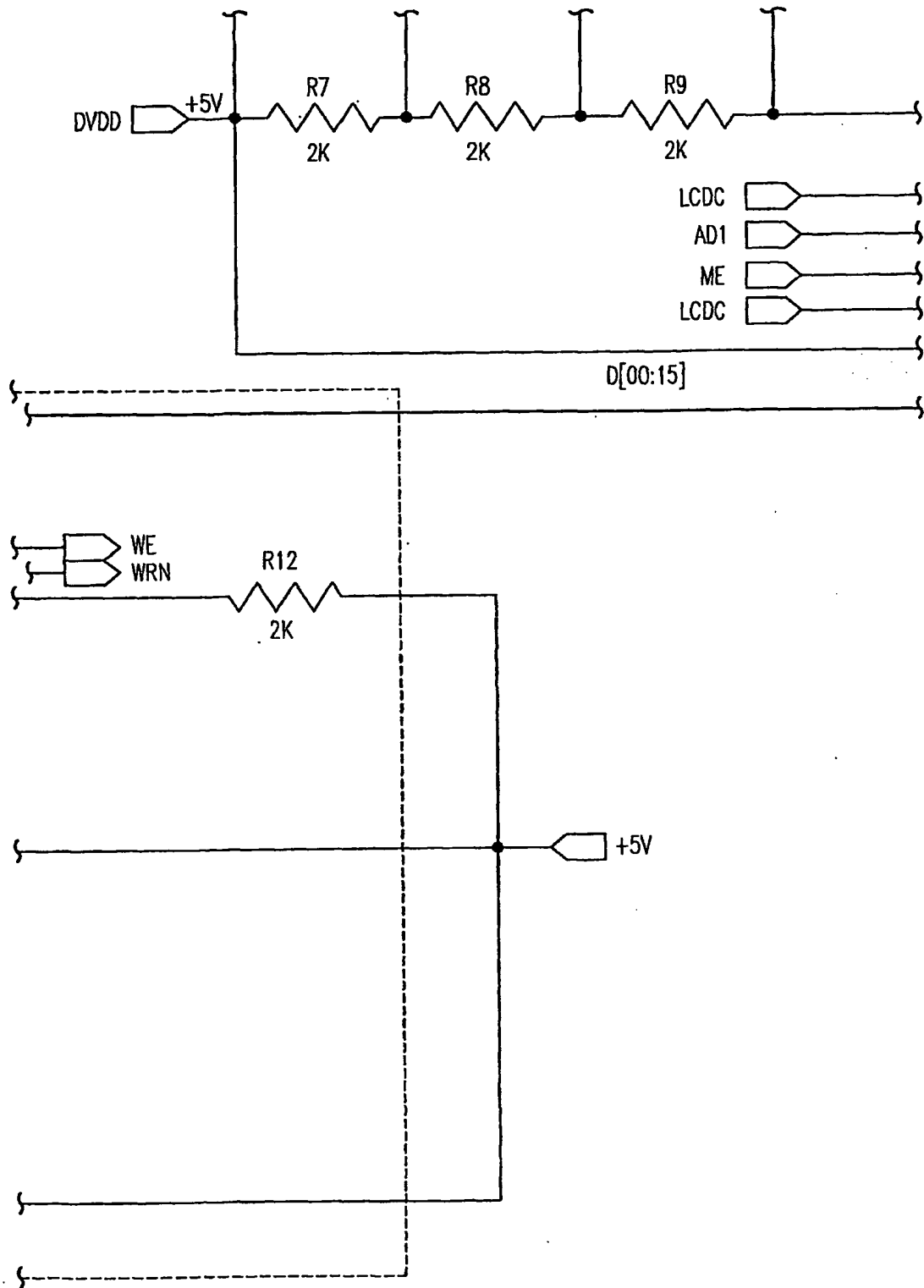


FIG.3B-7

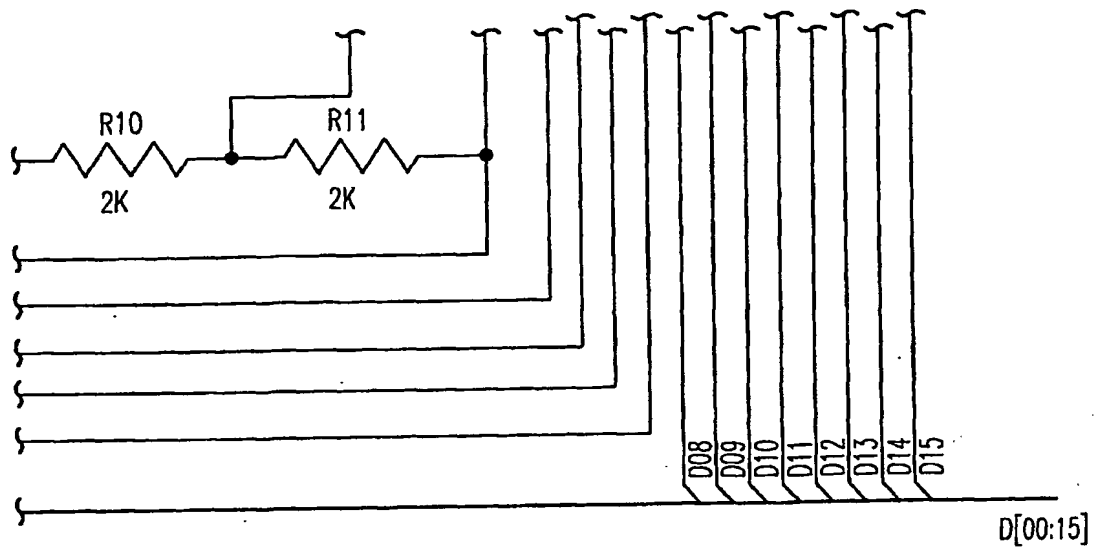


FIG.3B-8

FIG.4A	FIG.4B	FIG.4C
FIG.4D	FIG.4E	FIG.4F
FIG.4G	FIG.4H	FIG.4I
FIG.4J	FIG.4K	FIG.4L

FIG.4

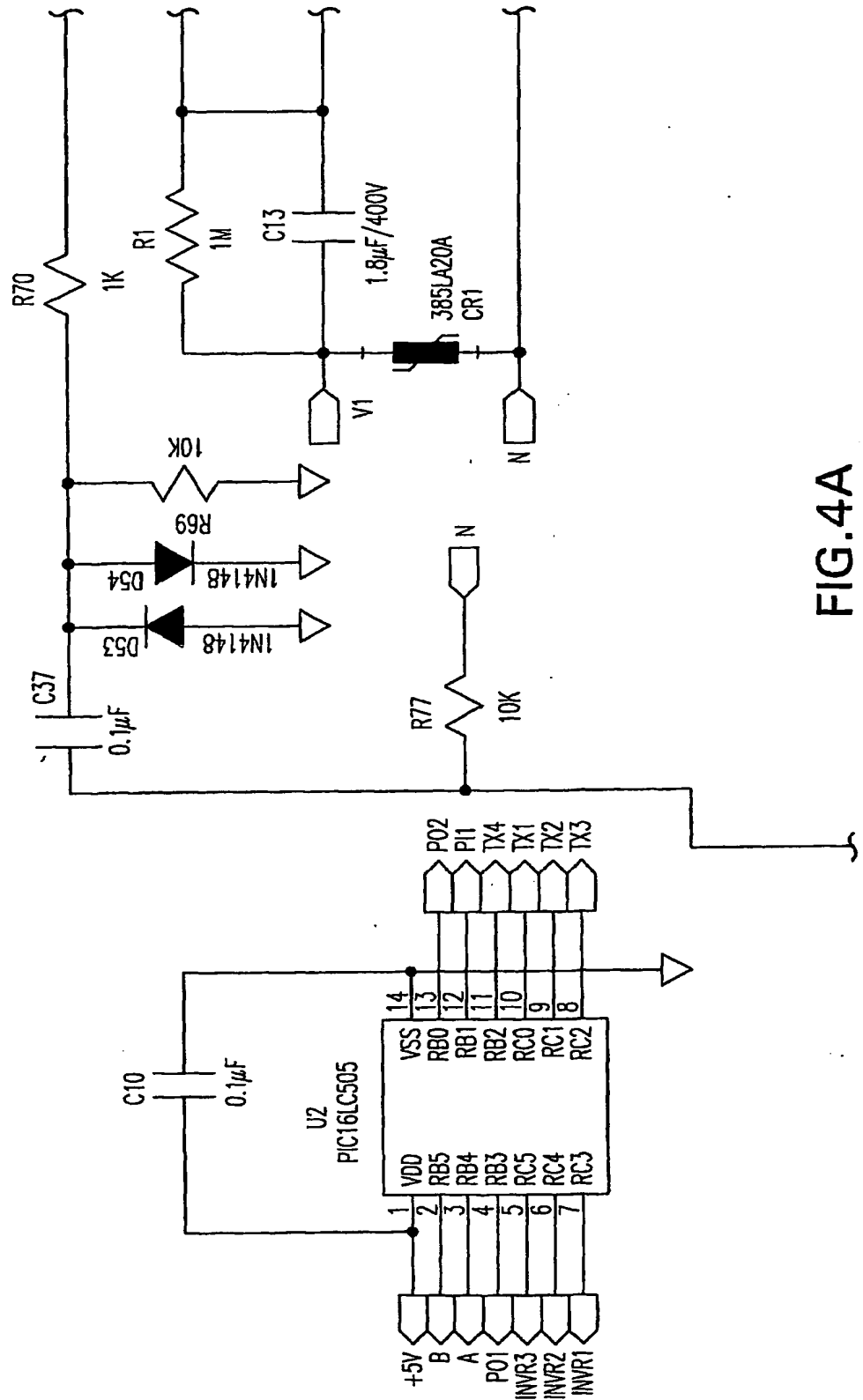


FIG. 4A

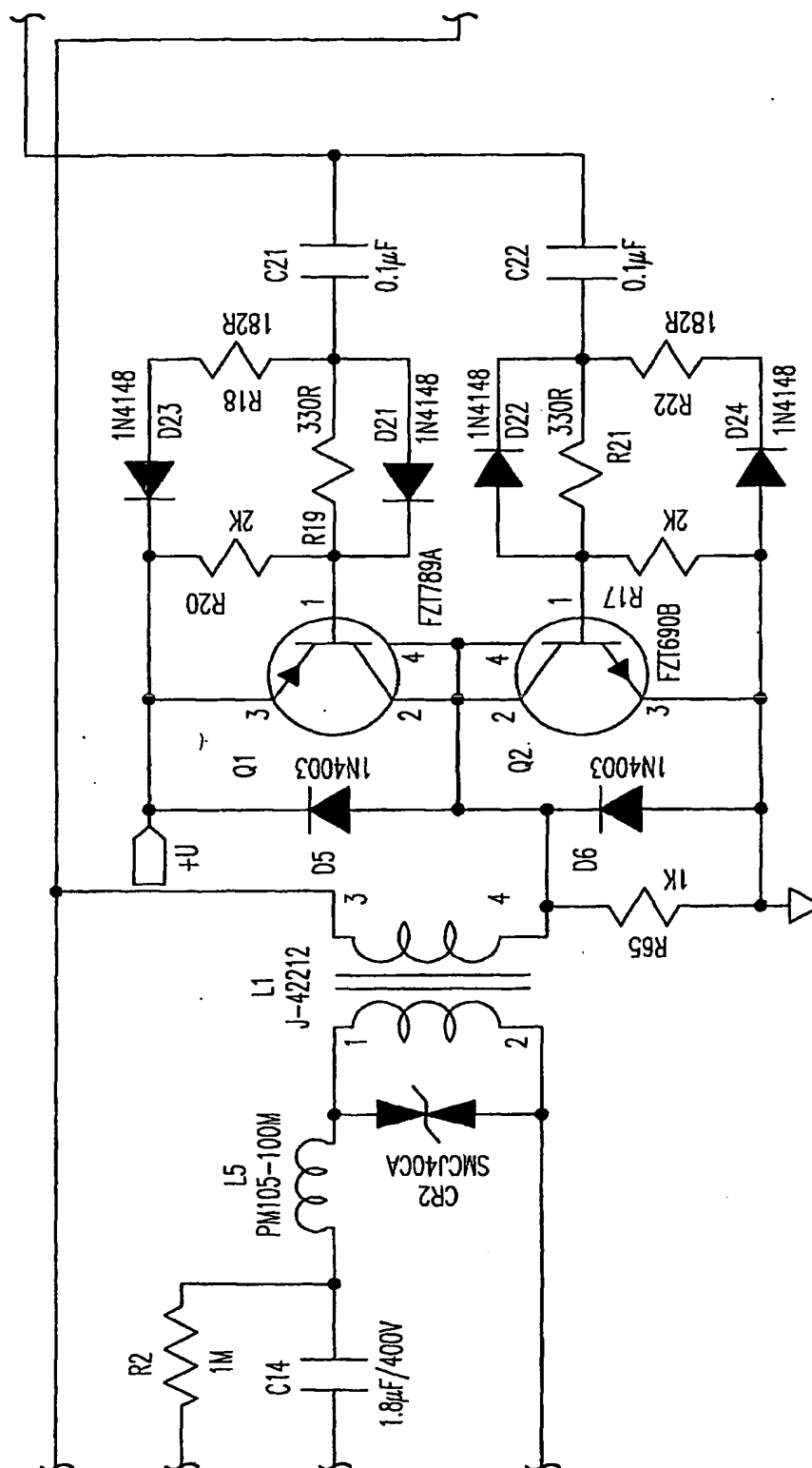


FIG. 4B

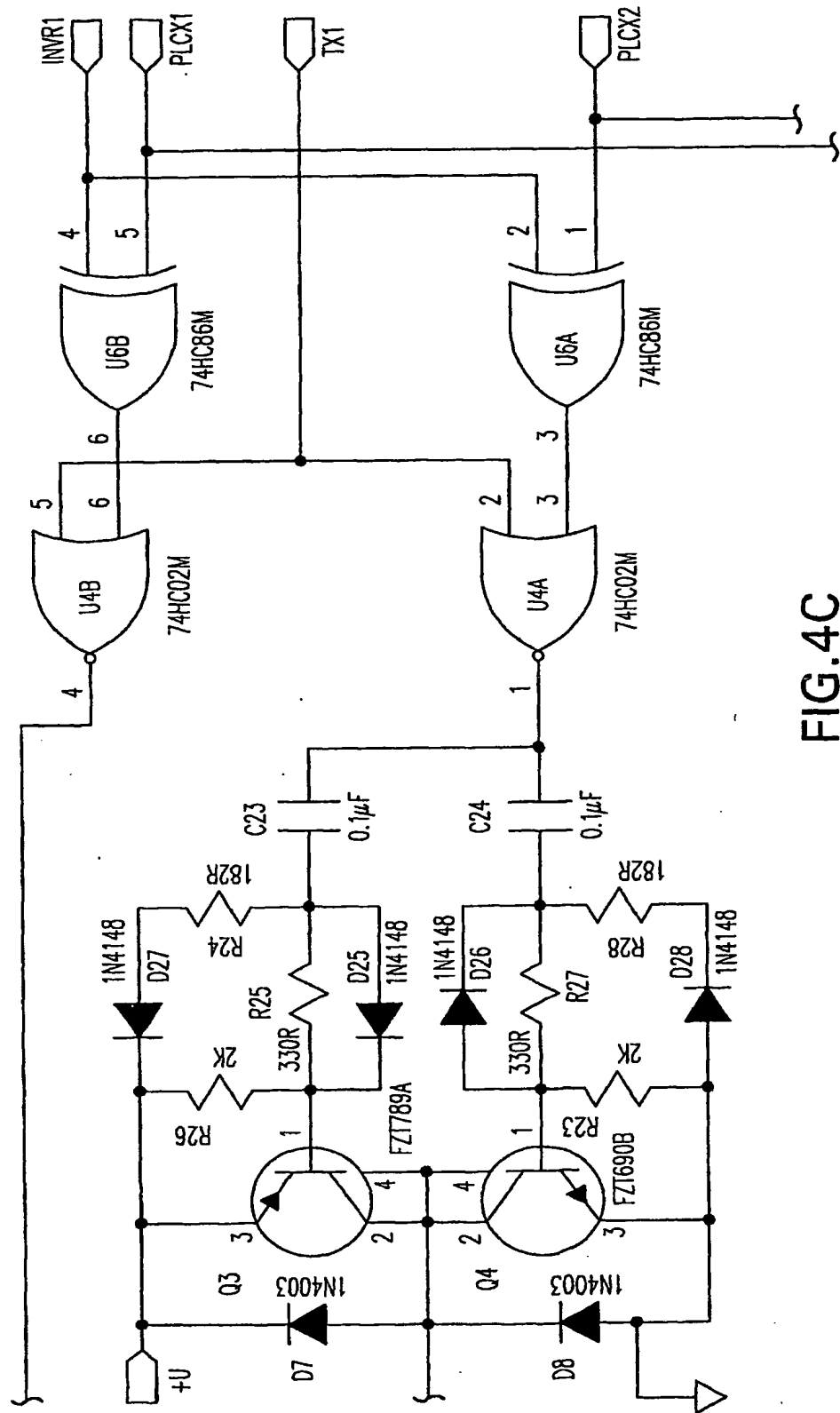


FIG. 4C

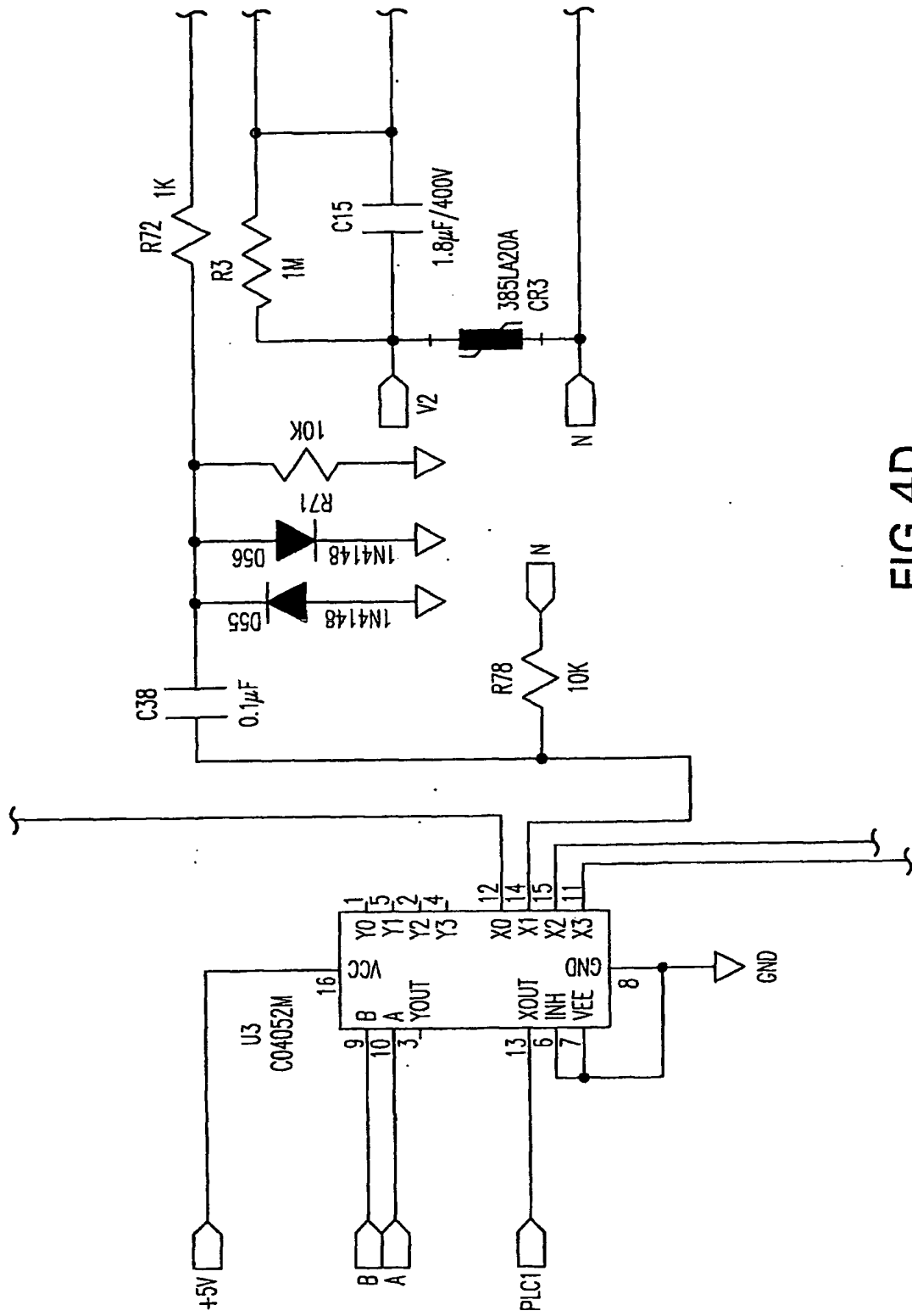


FIG. 4D

FIG. 4E

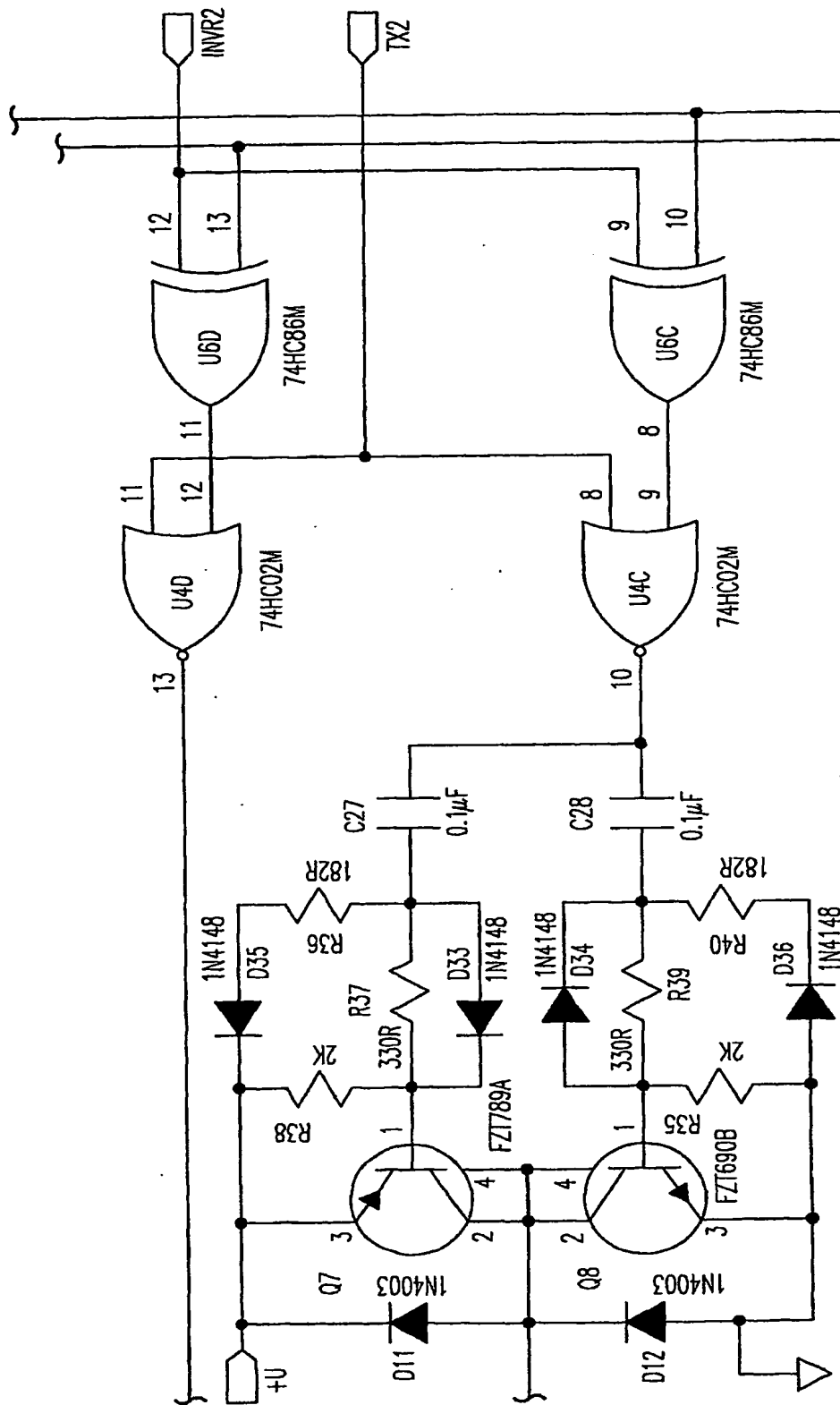


FIG. 4F

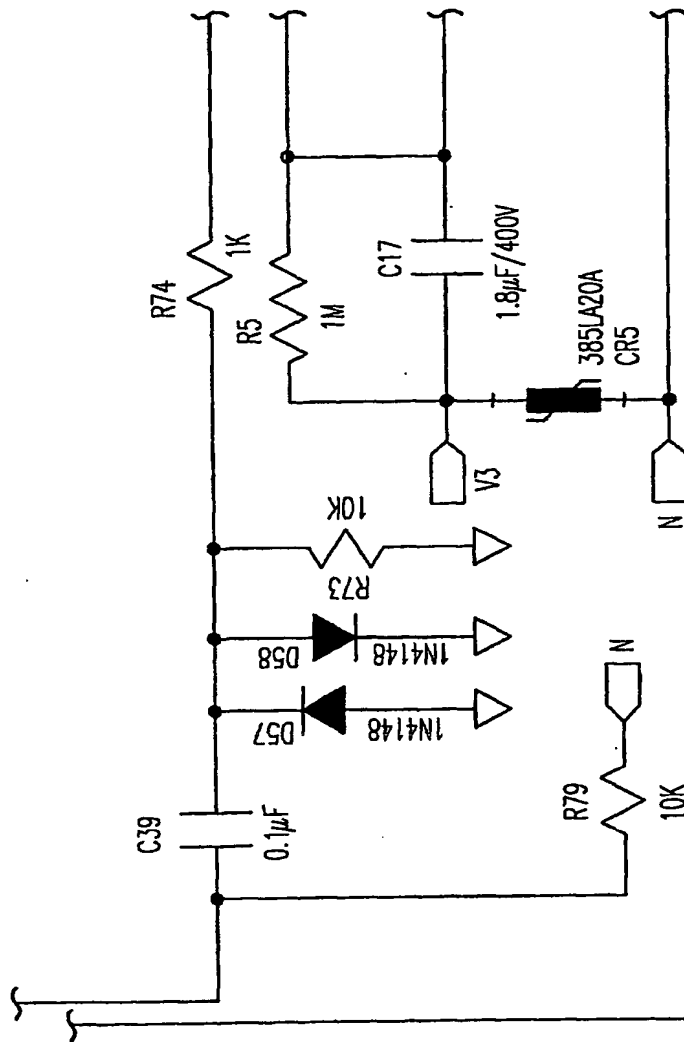


FIG. 4G

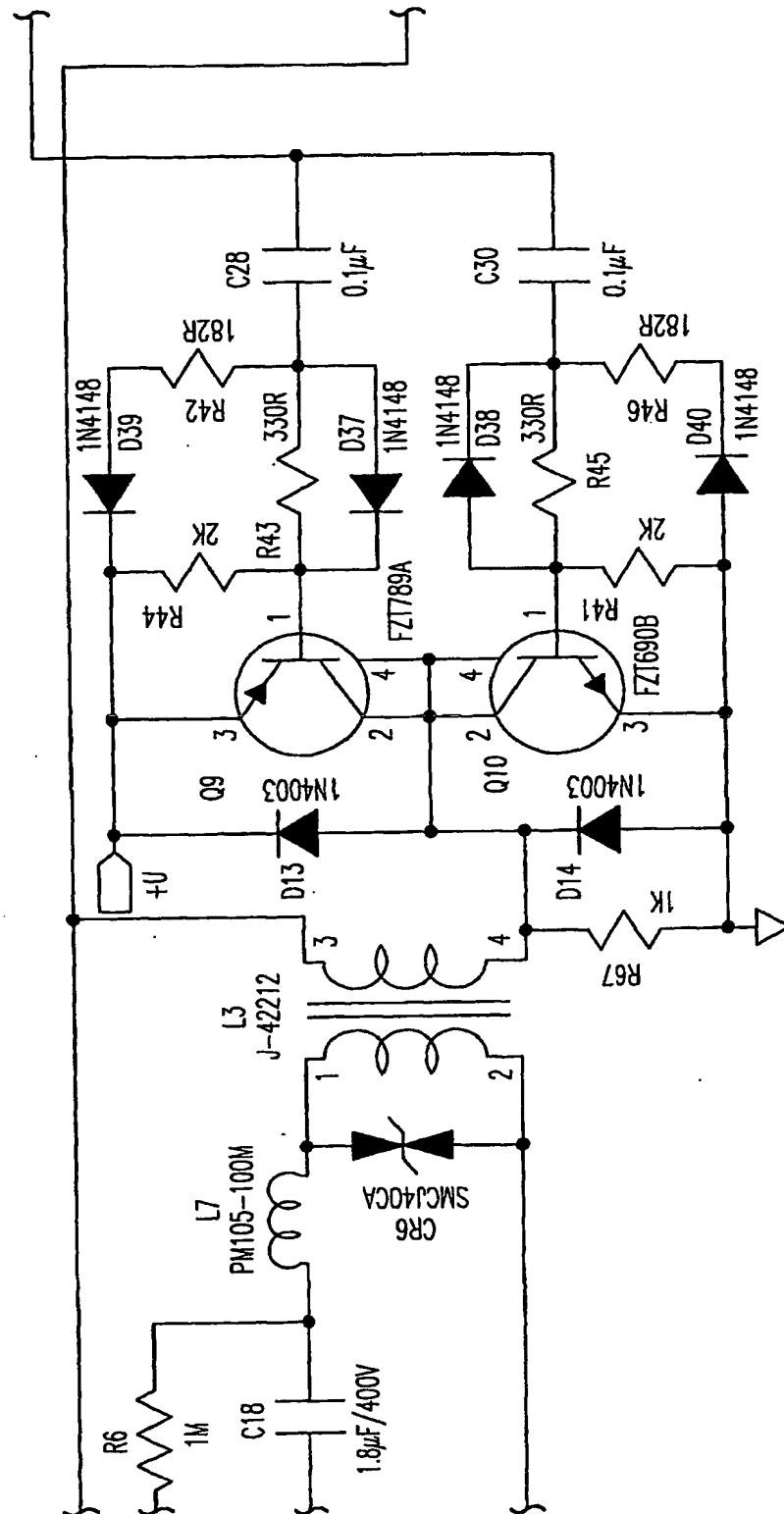


FIG. 4H

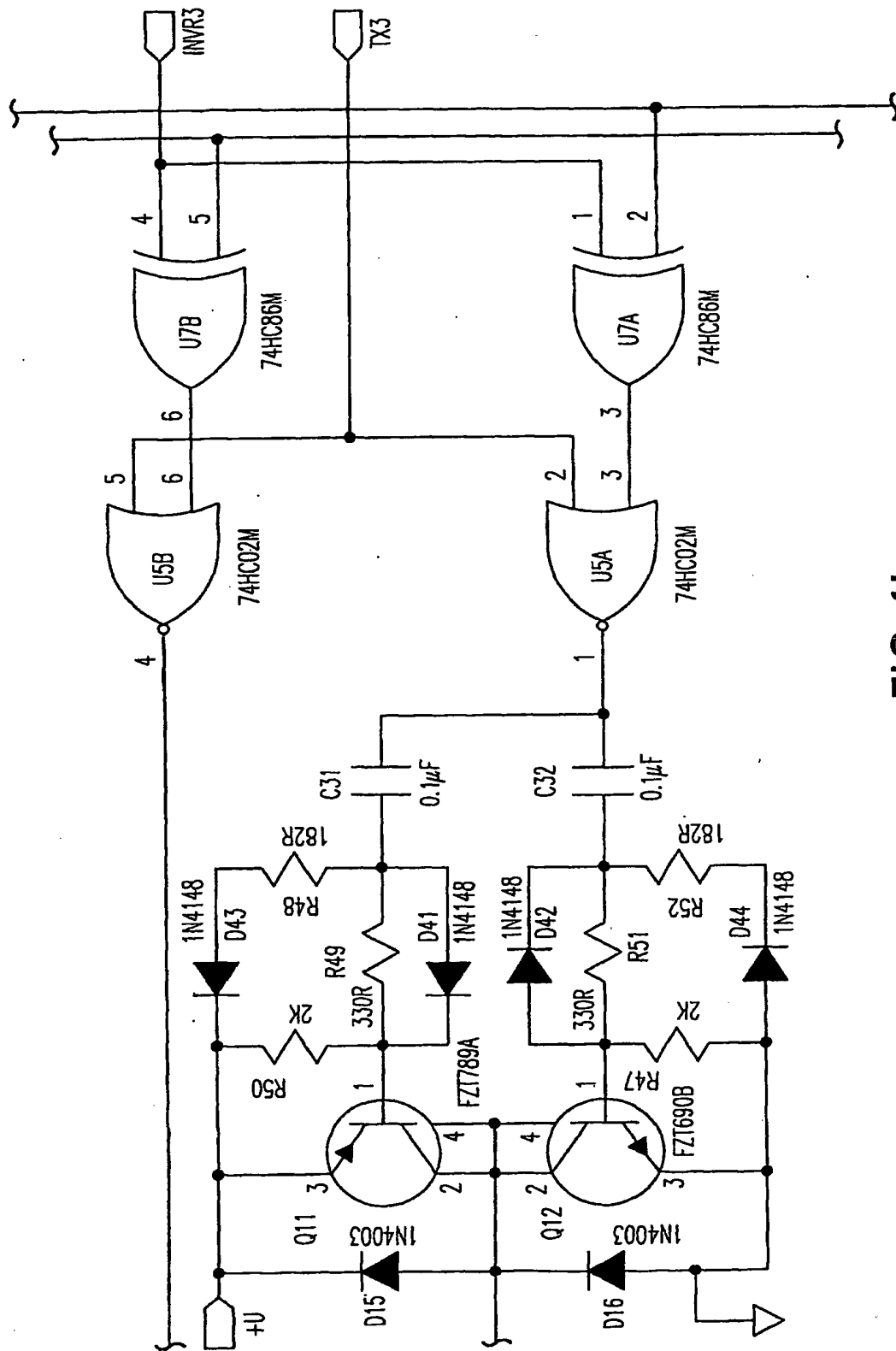


FIG.4I

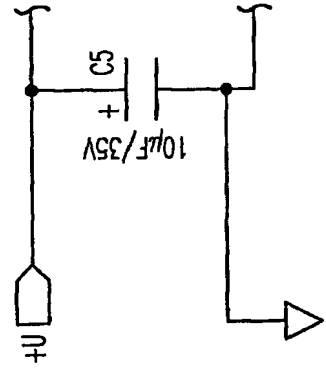
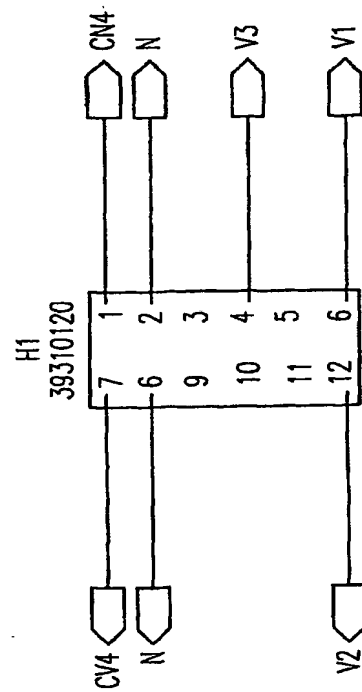
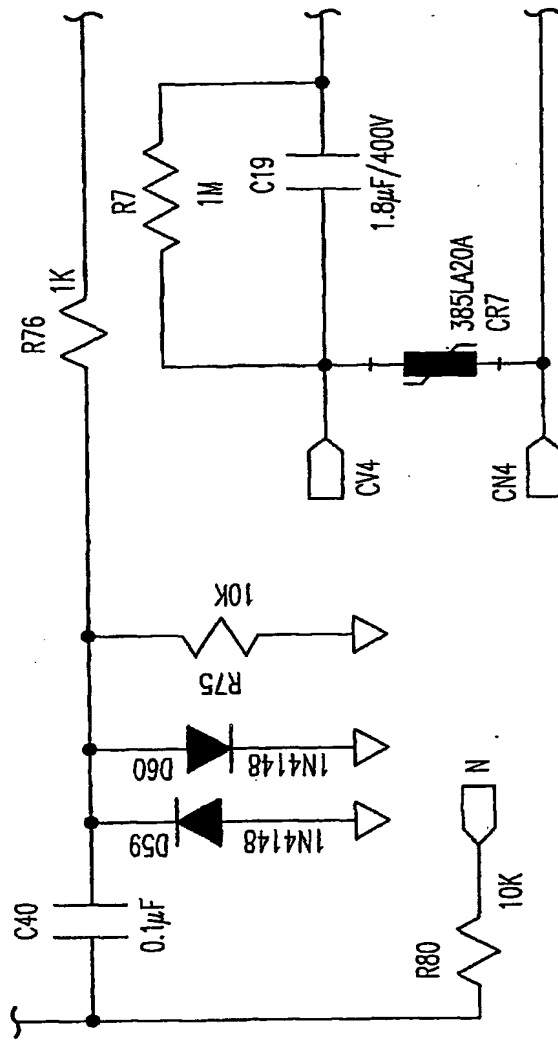
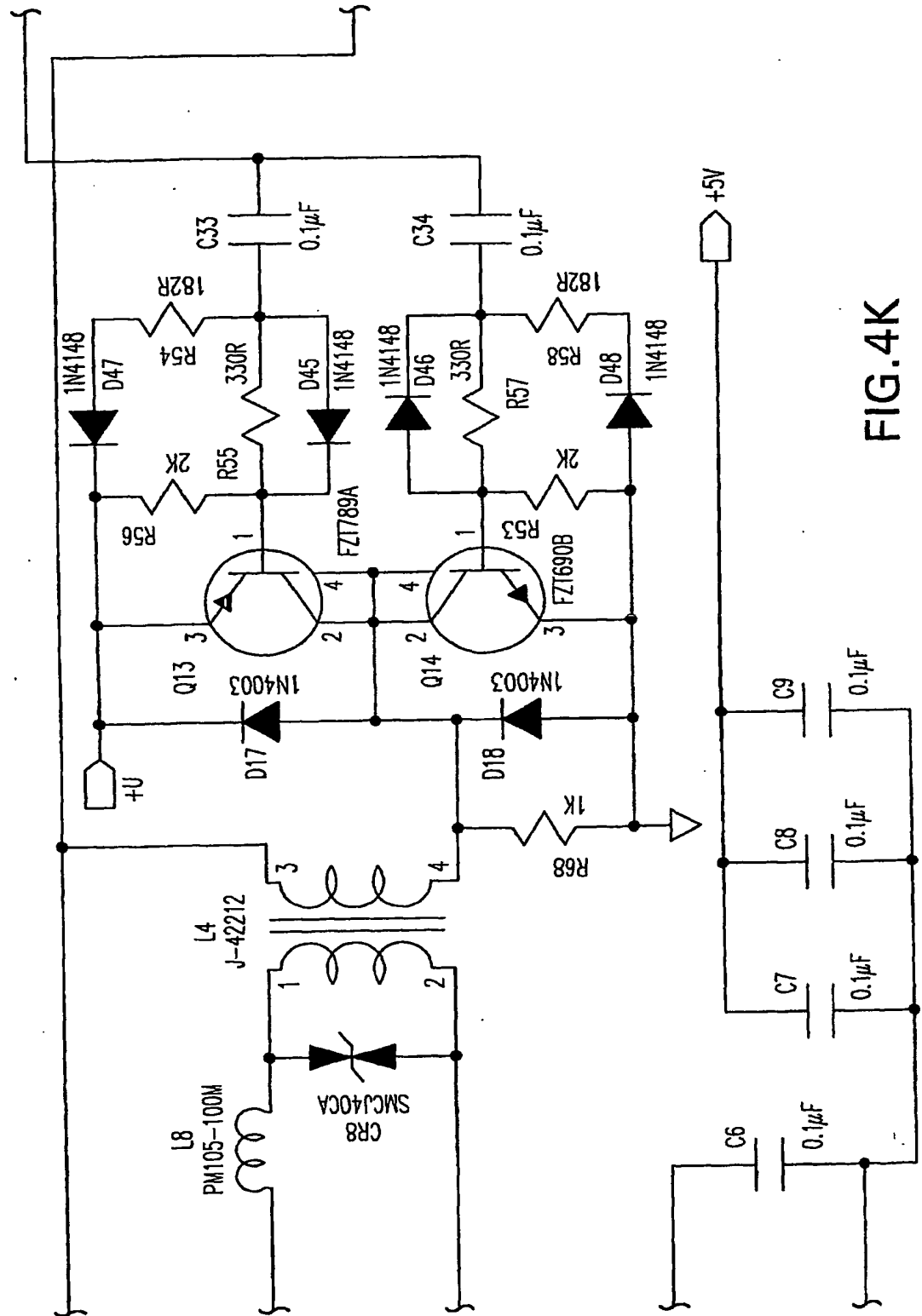


FIG.4J



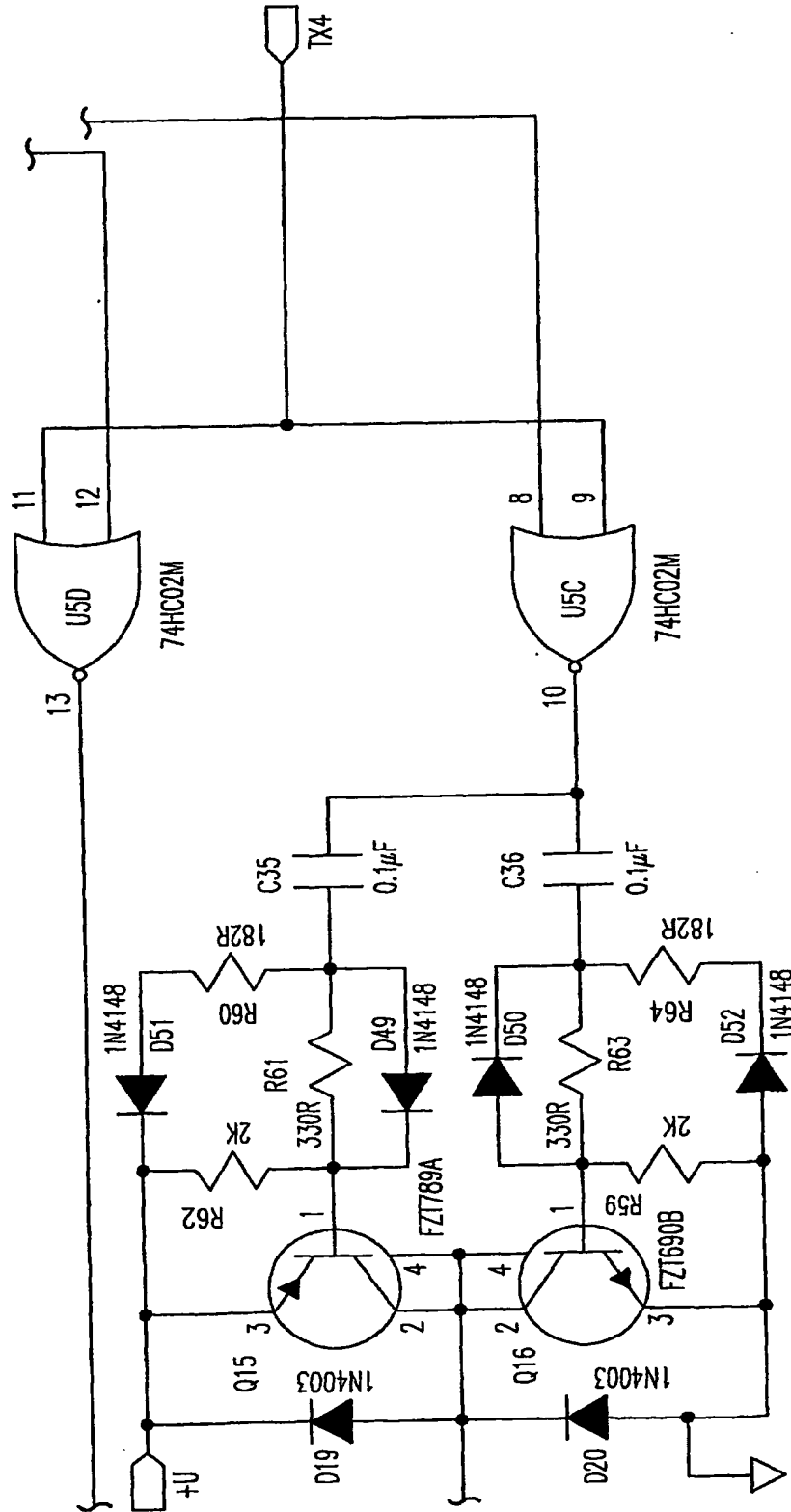


FIG.4L

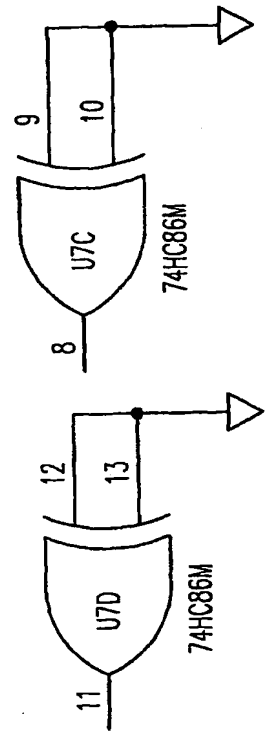
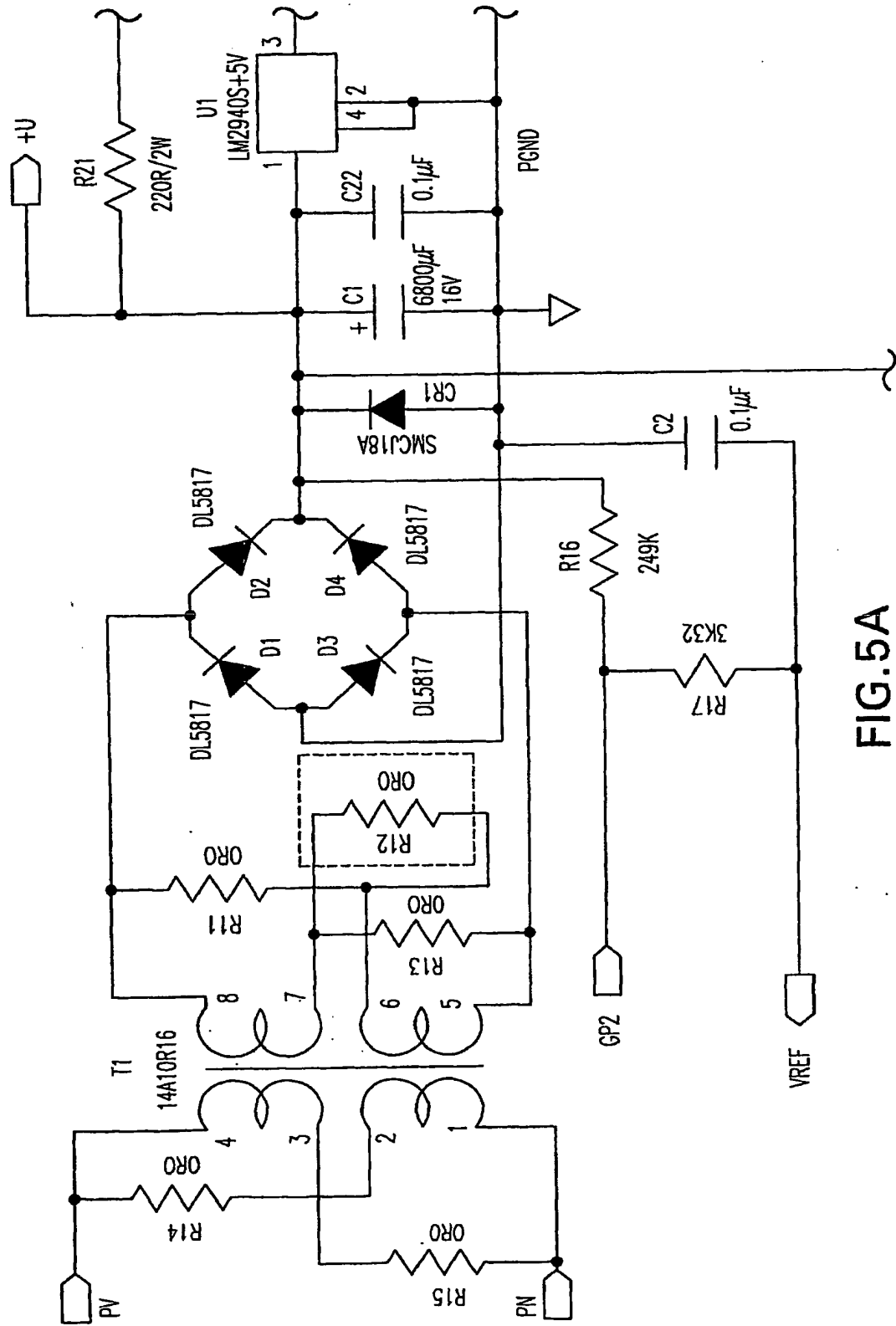


FIG.5A	FIG.5B	FIG.5C
FIG.5D	FIG.5E	FIG.5F
FIG.5G	FIG.5H	FIG.5I

FIG.5



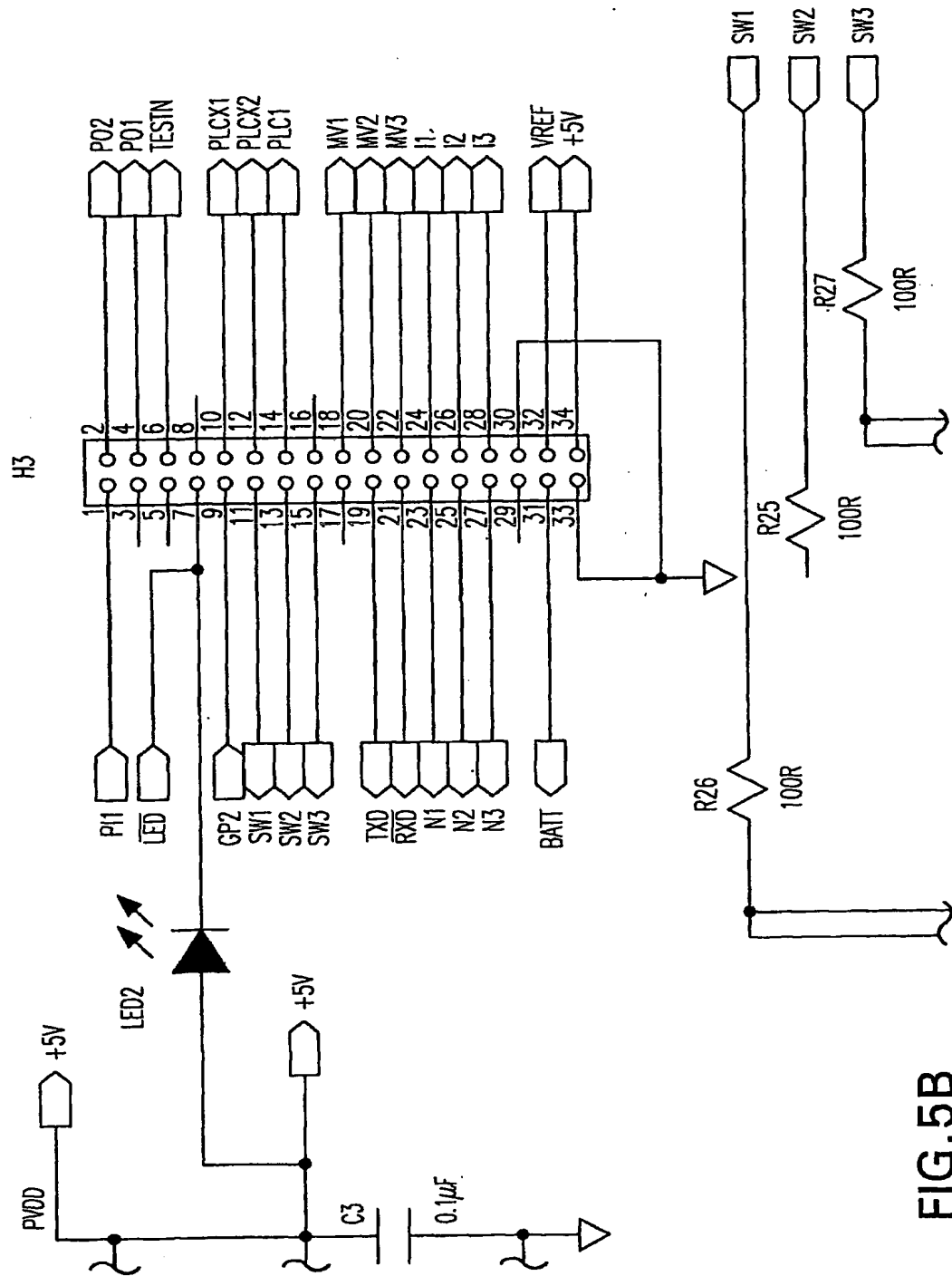


FIG. 5B

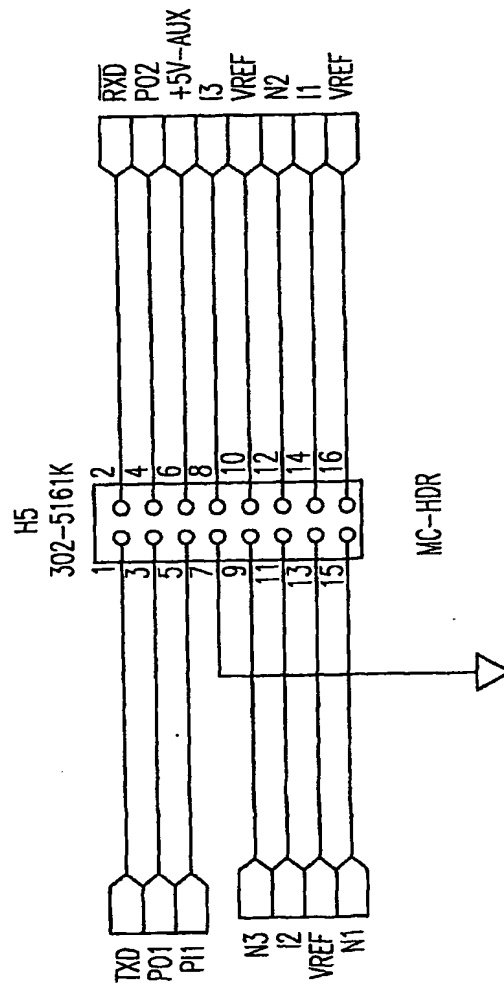


FIG.5C

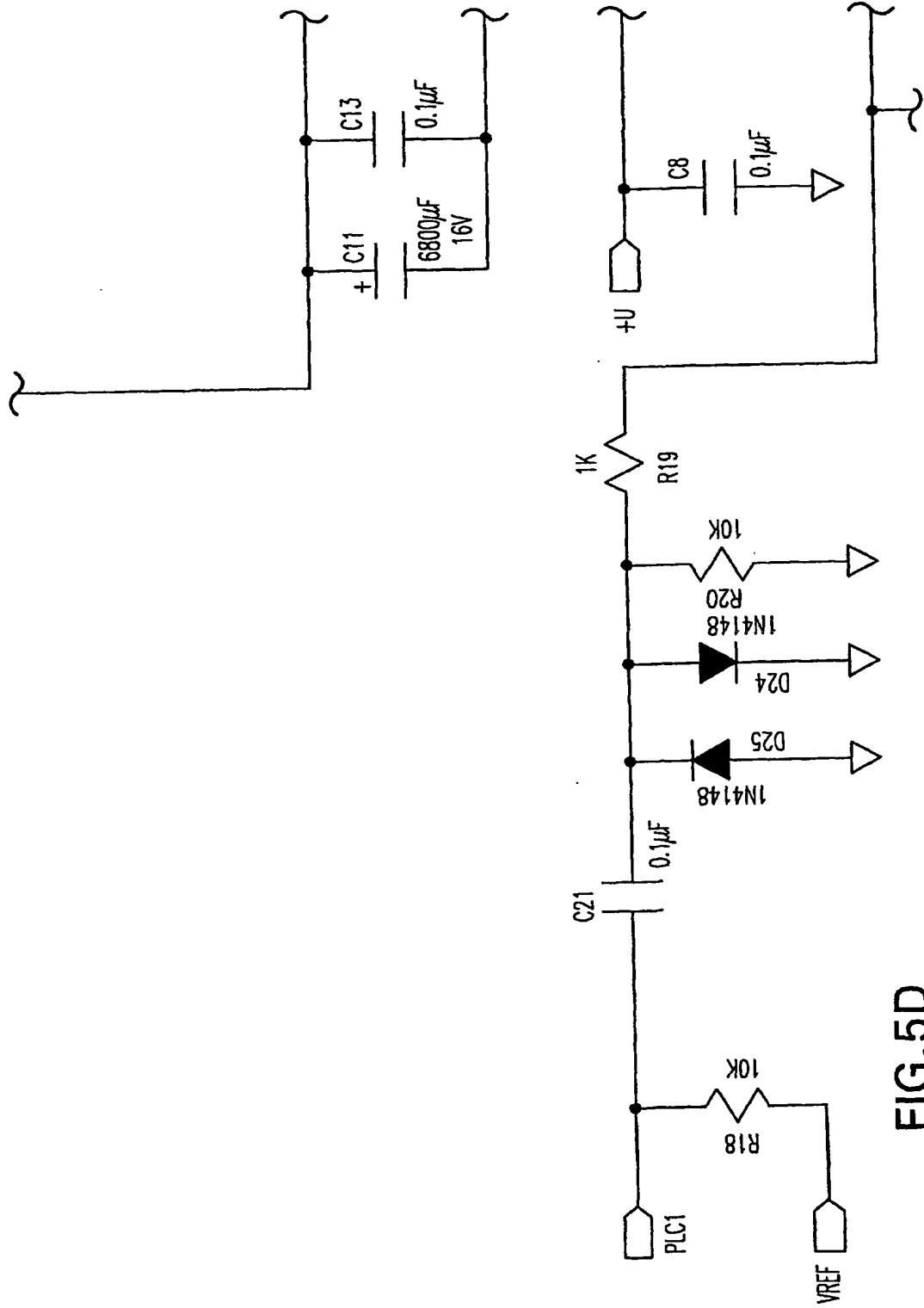


FIG.5D

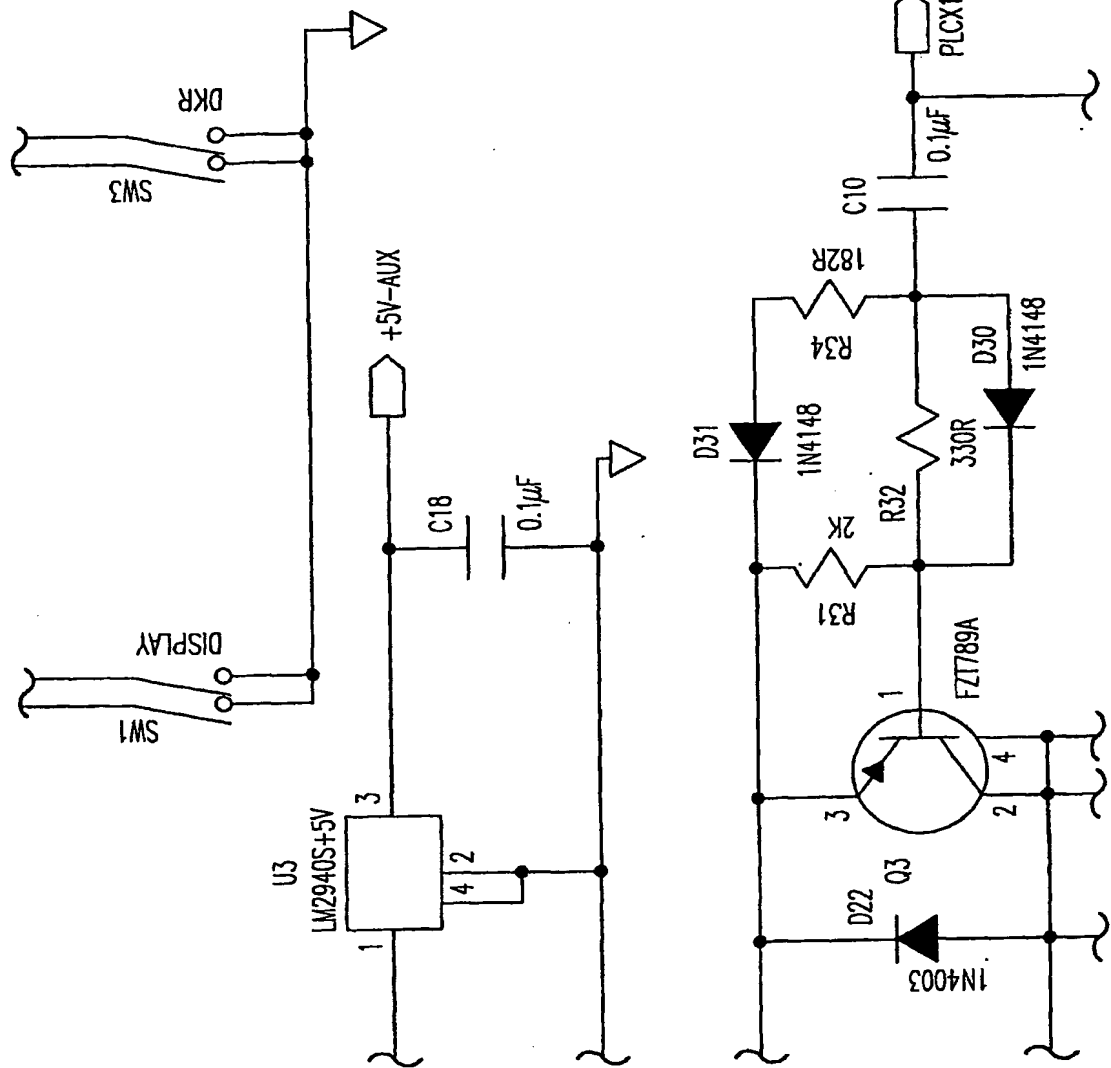


FIG.5E

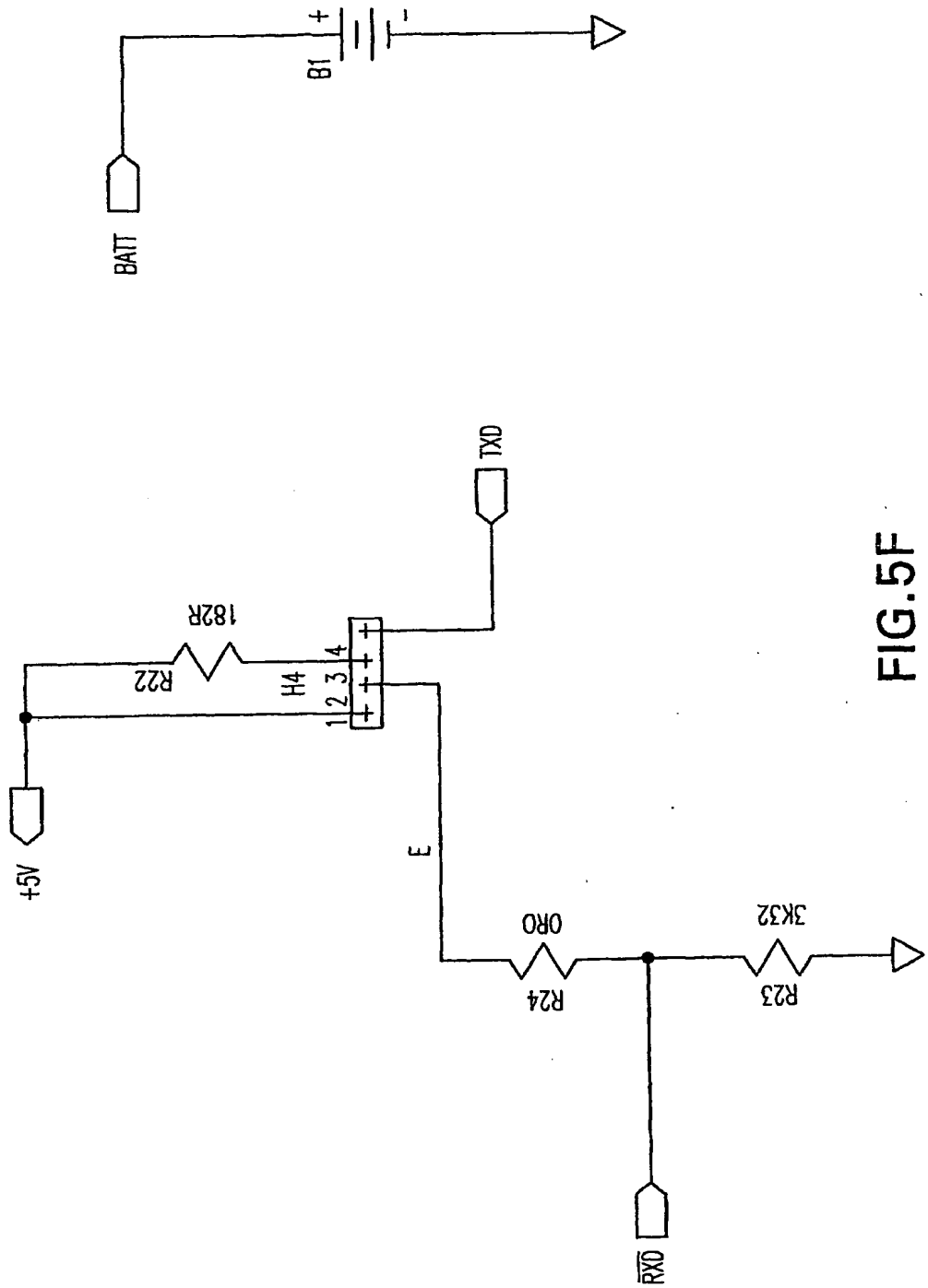


FIG. 5F

FIG. 5G

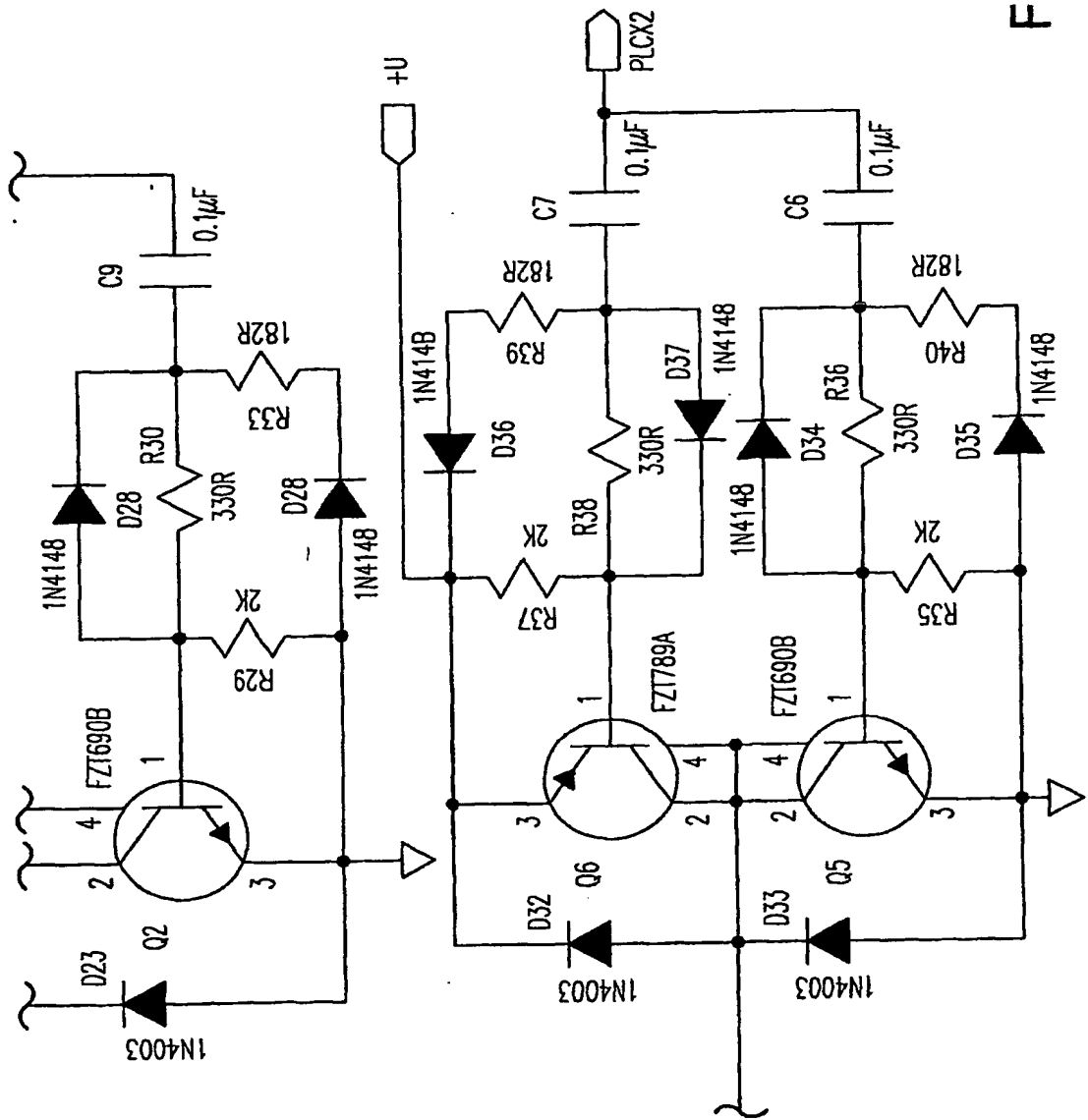


FIG. 5H

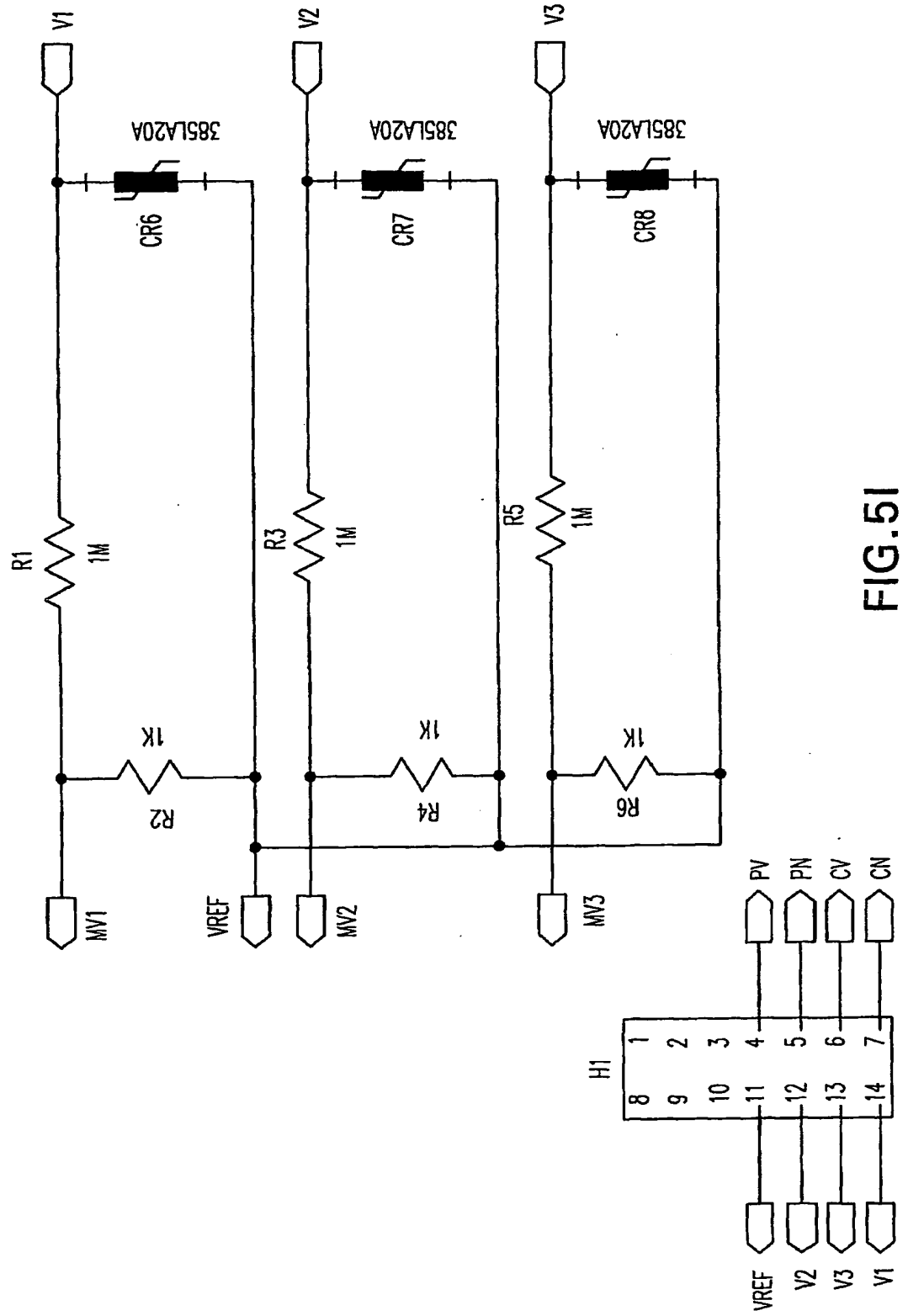
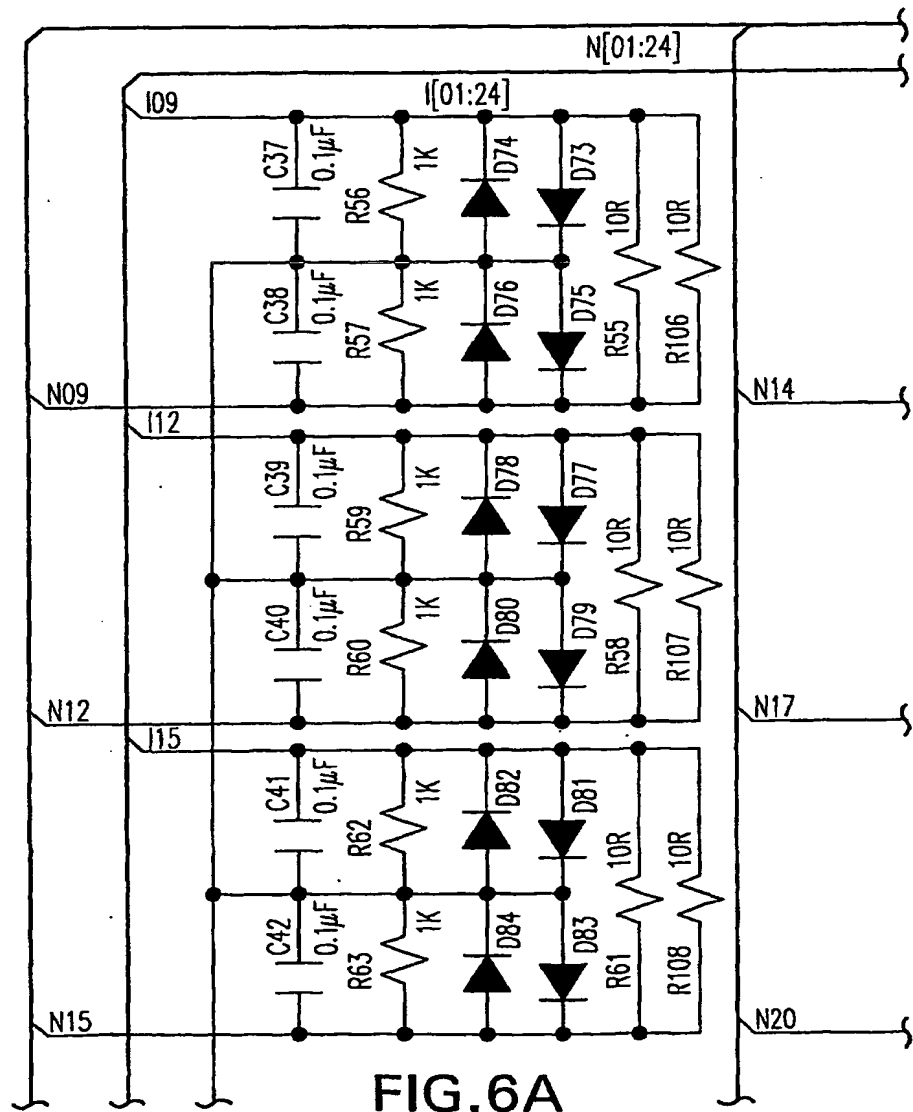
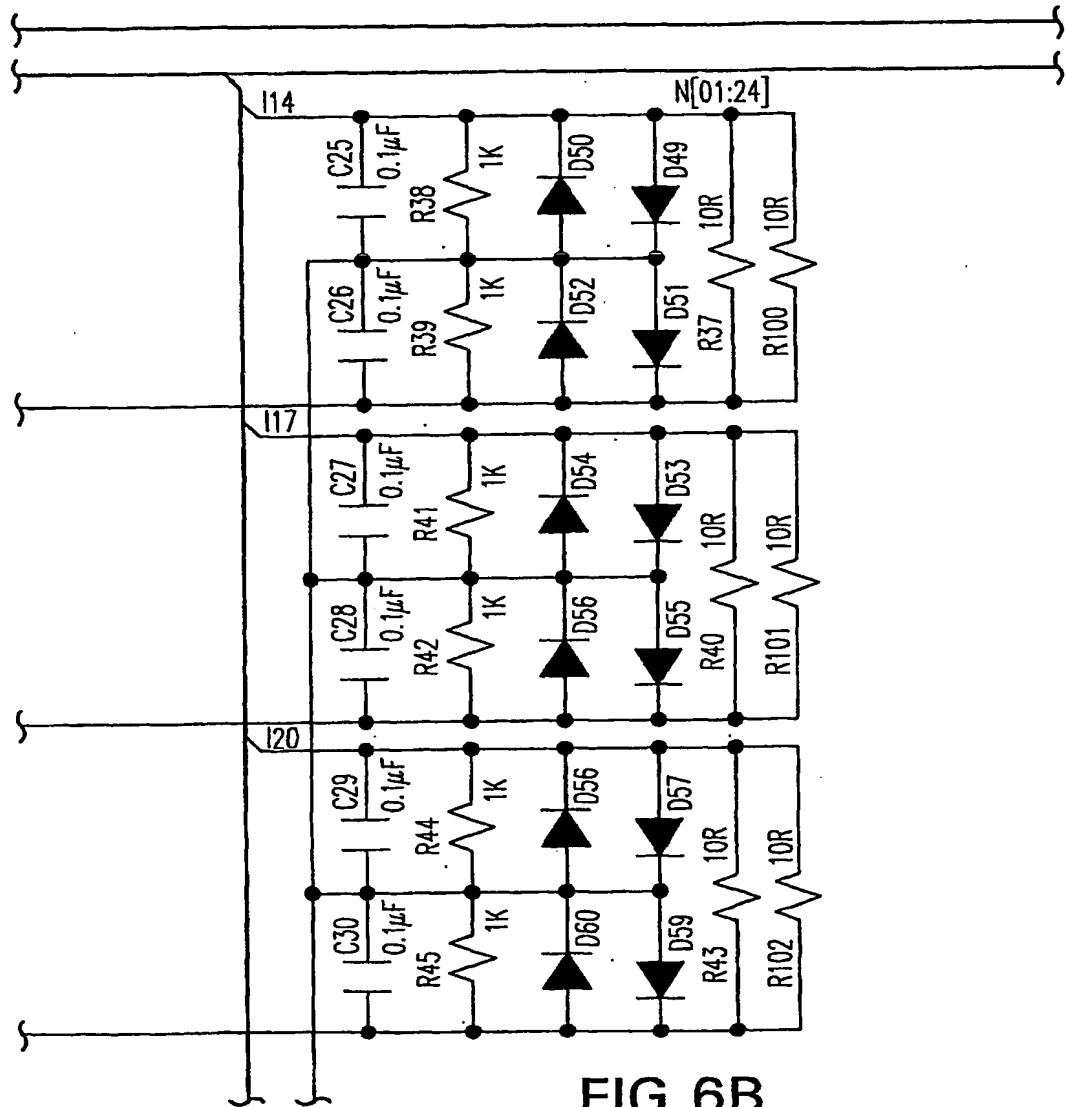


FIG.5I

FIG.6A	FIG.6B	FIG.6C	FIG.6D	FIG.6E	FIG.6F	FIG.6G
FIG.6H	FIG.6I	FIG.6J	FIG.6K	FIG.6L	FIG.6M	FIG.6N
FIG.6O	FIG.6P	FIG.6Q	FIG.6R	FIG.6S	FIG.6T	FIG.6U

FIG.6





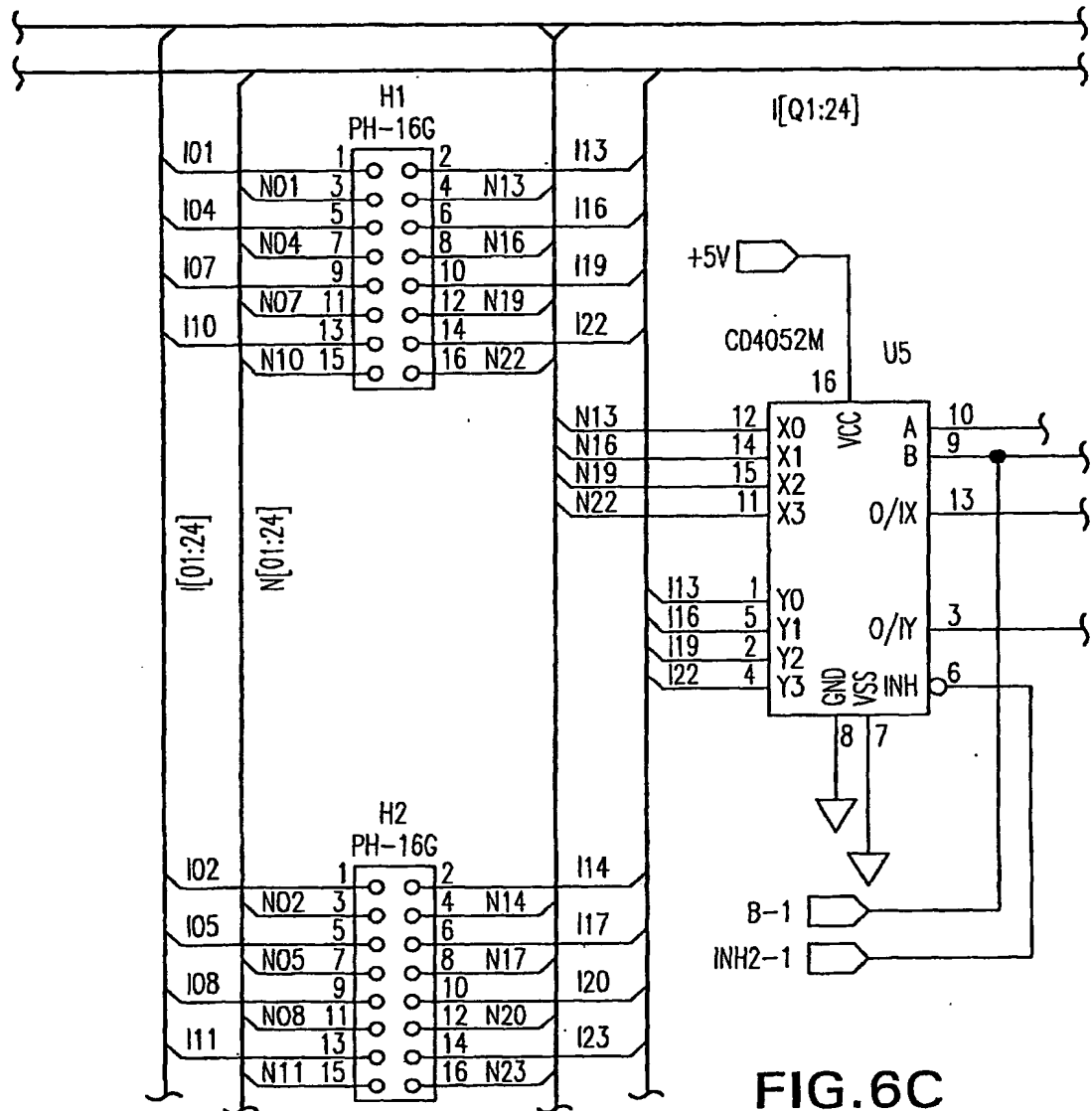


FIG.6C

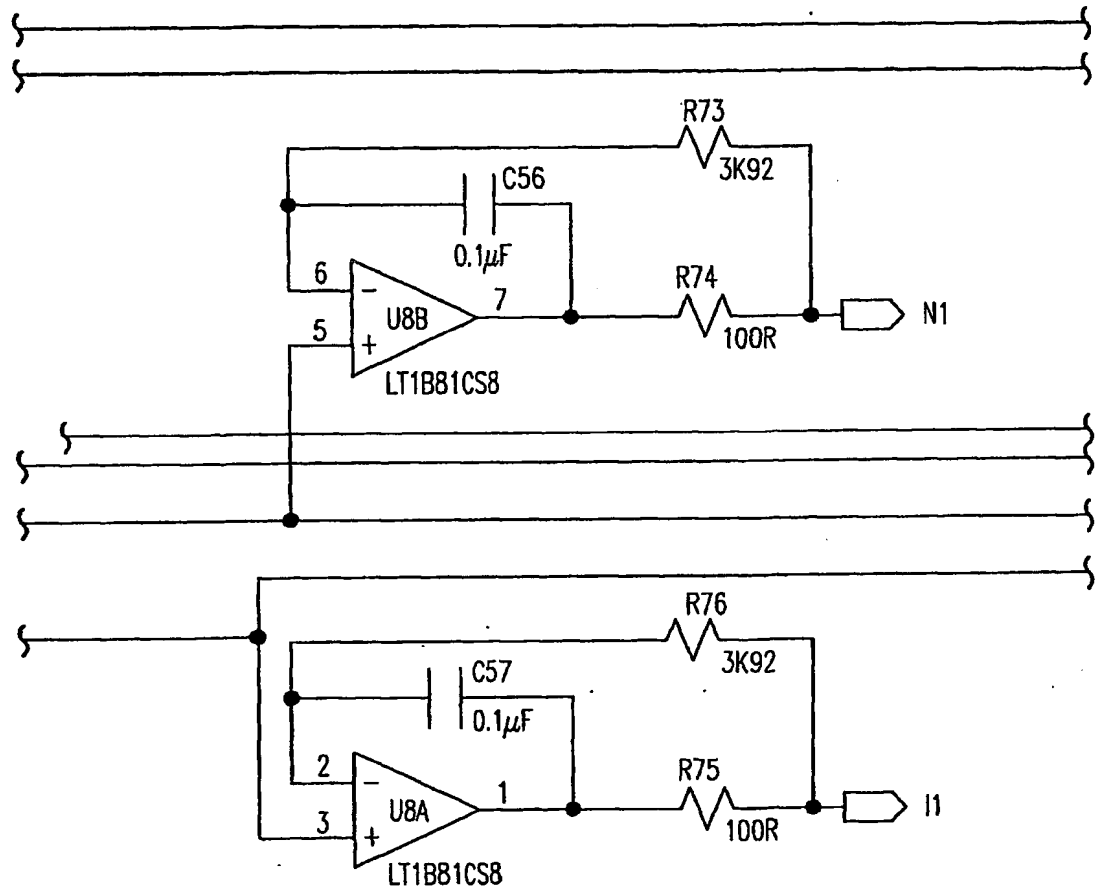


FIG.6D

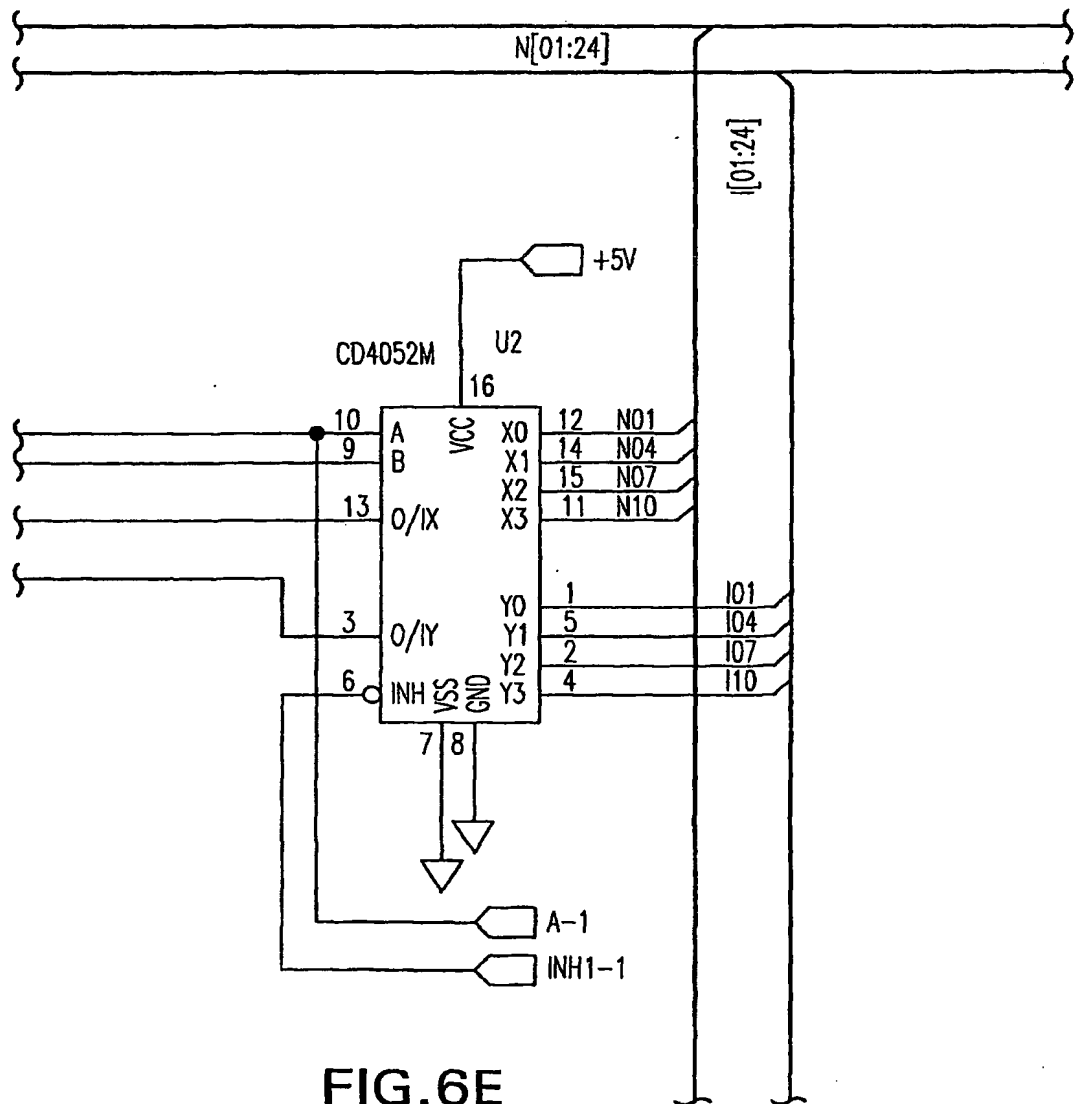
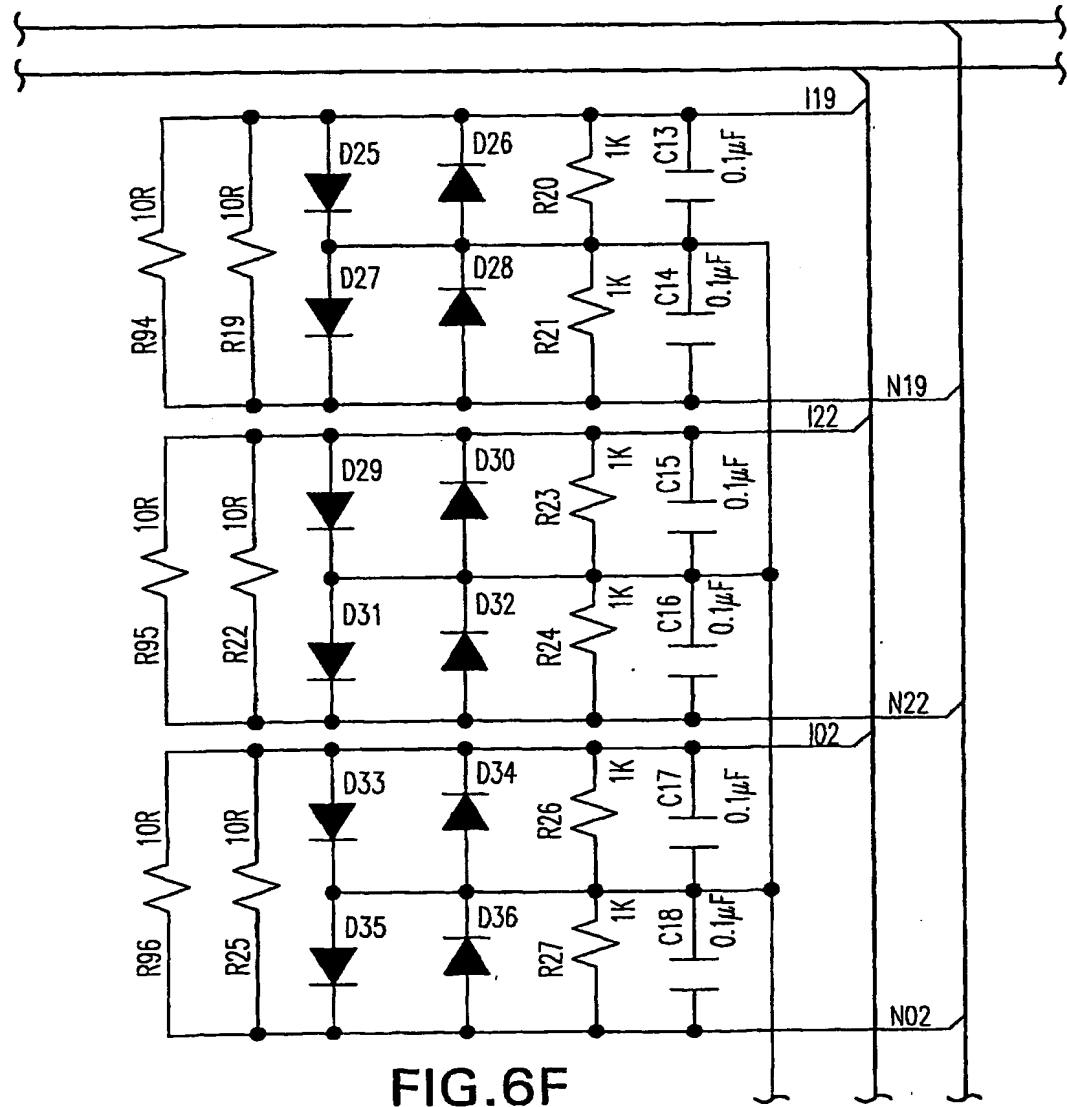
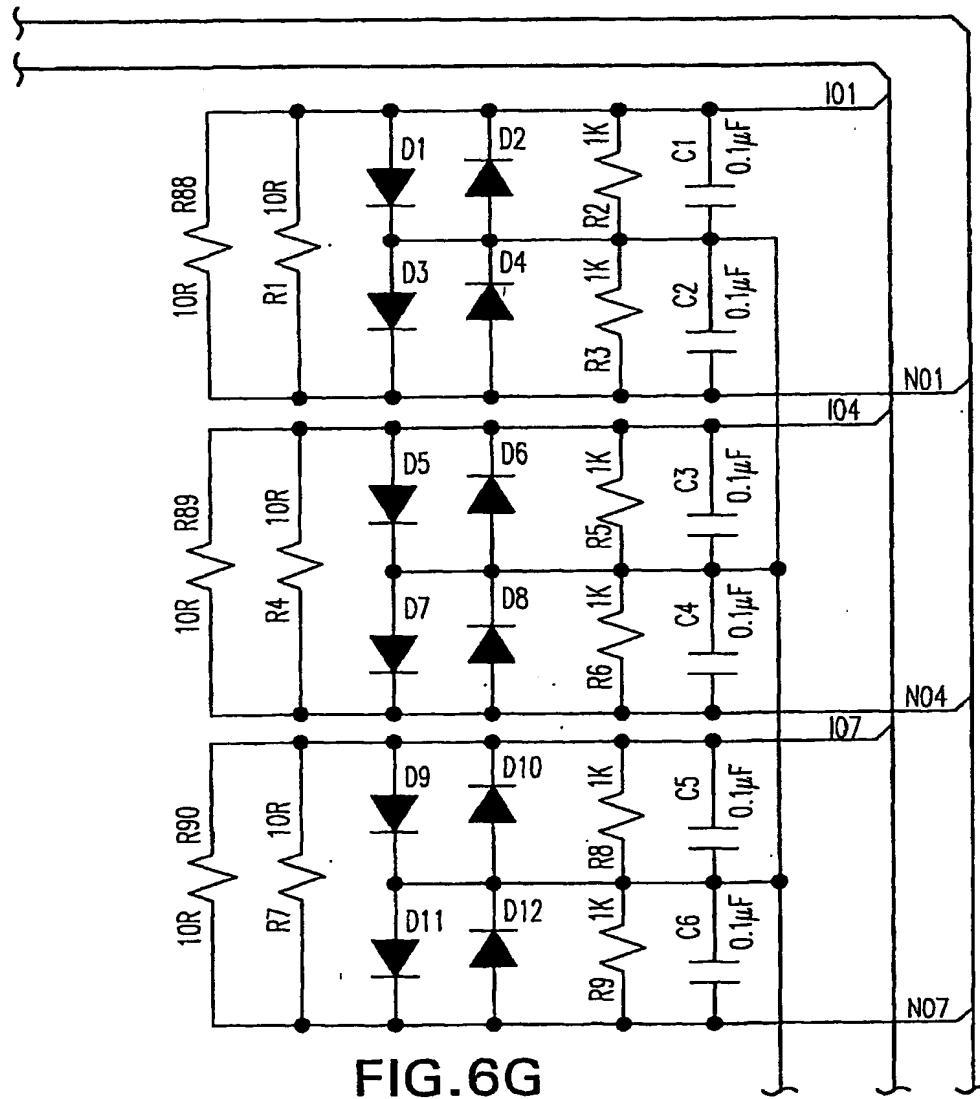


FIG.6E





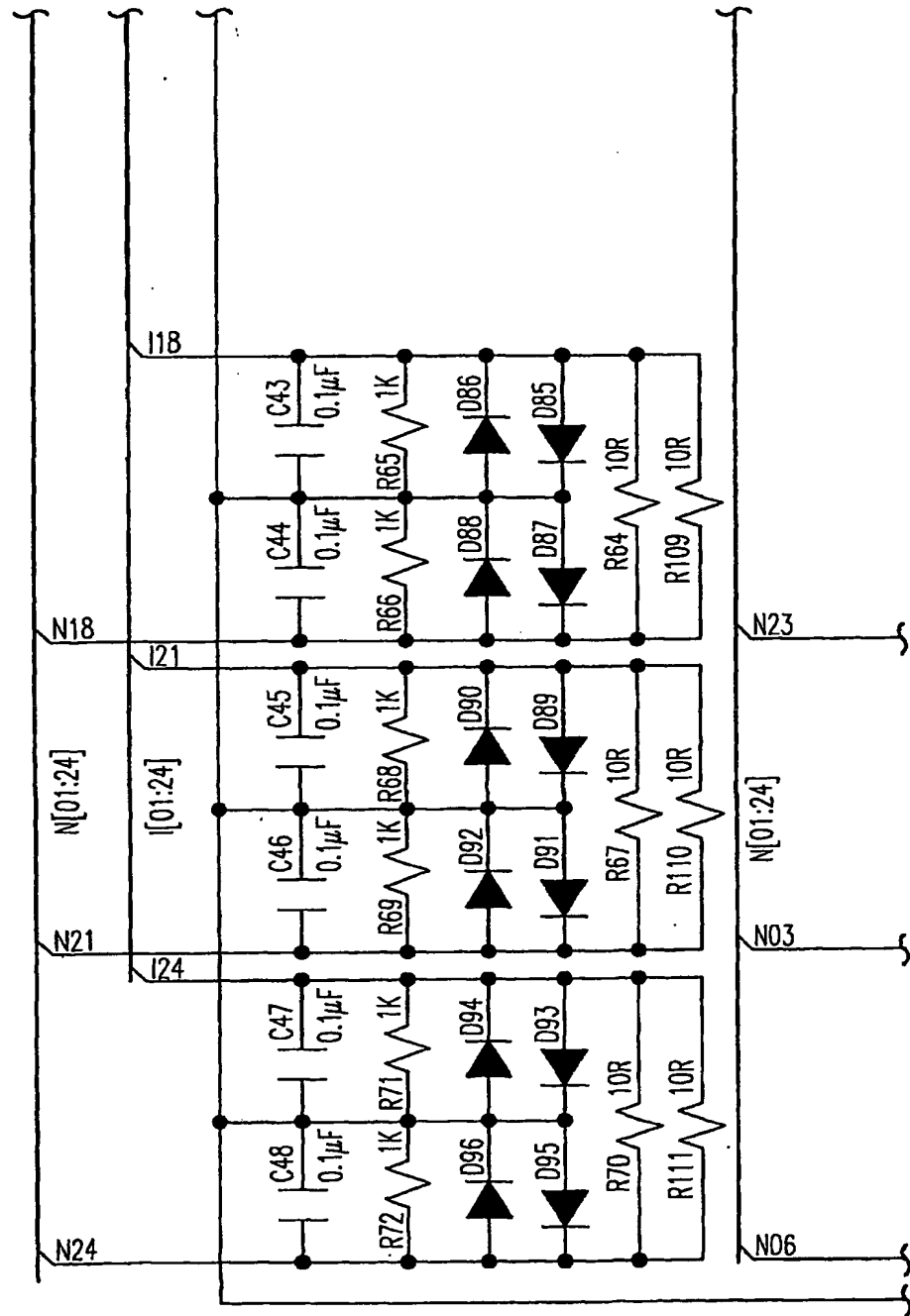


FIG.6H

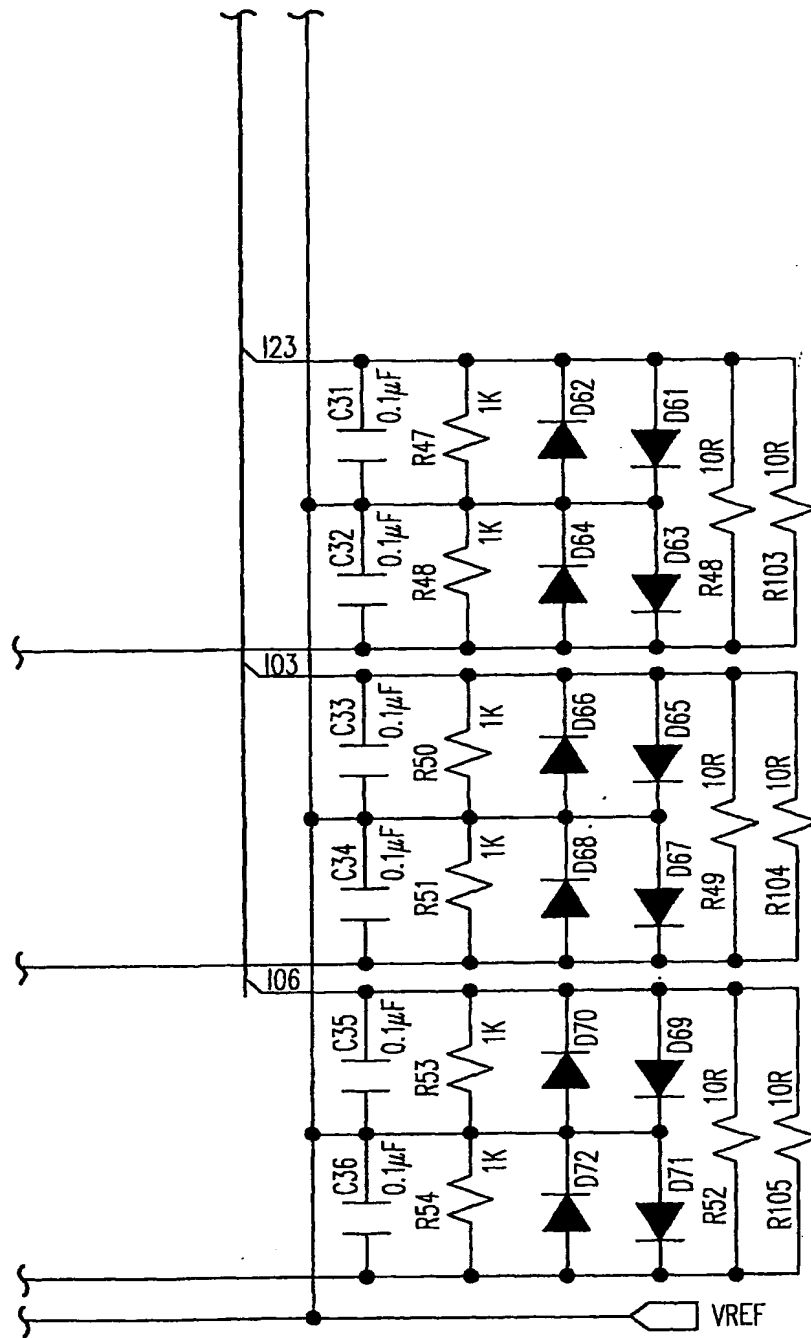
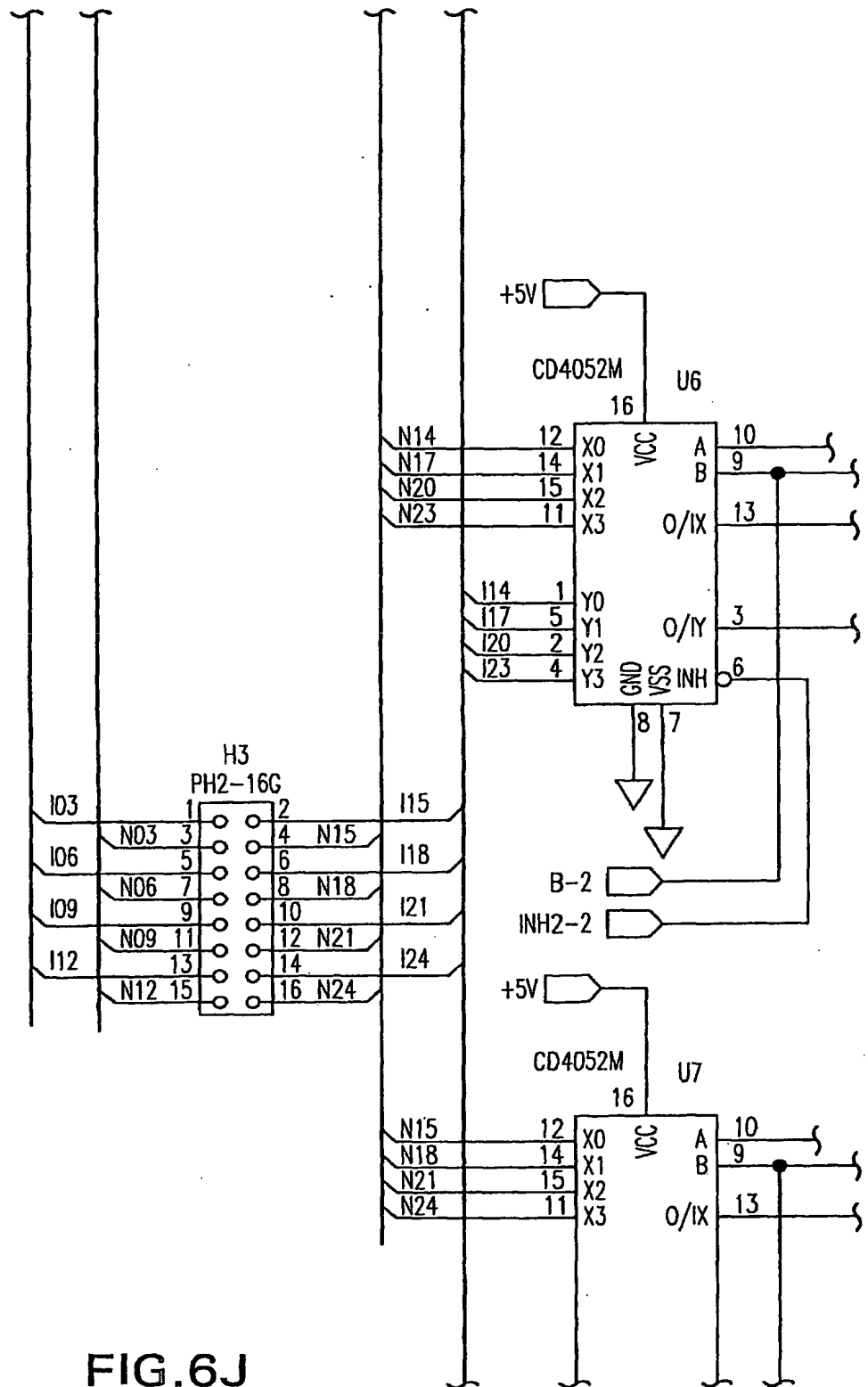


FIG.6I



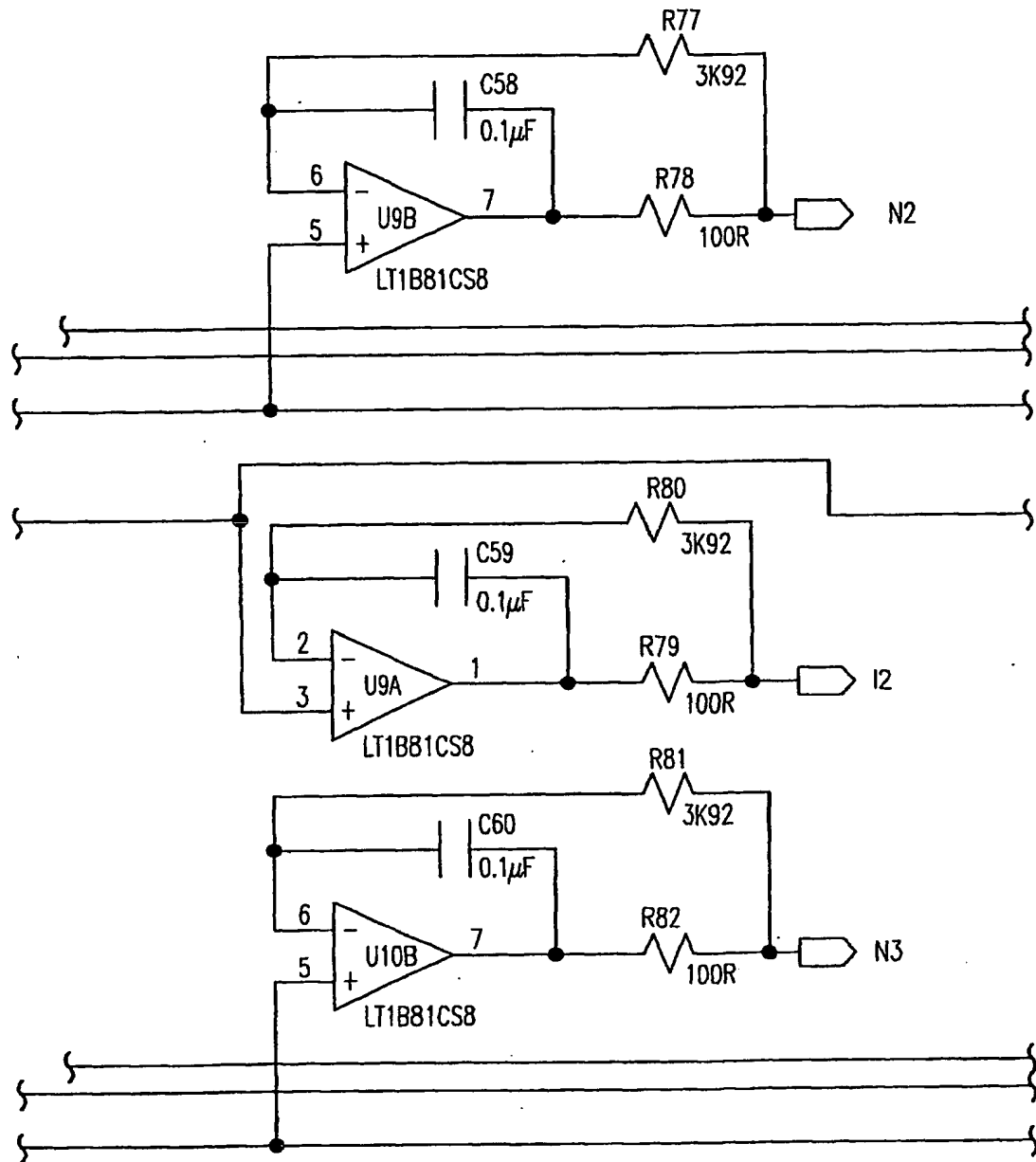
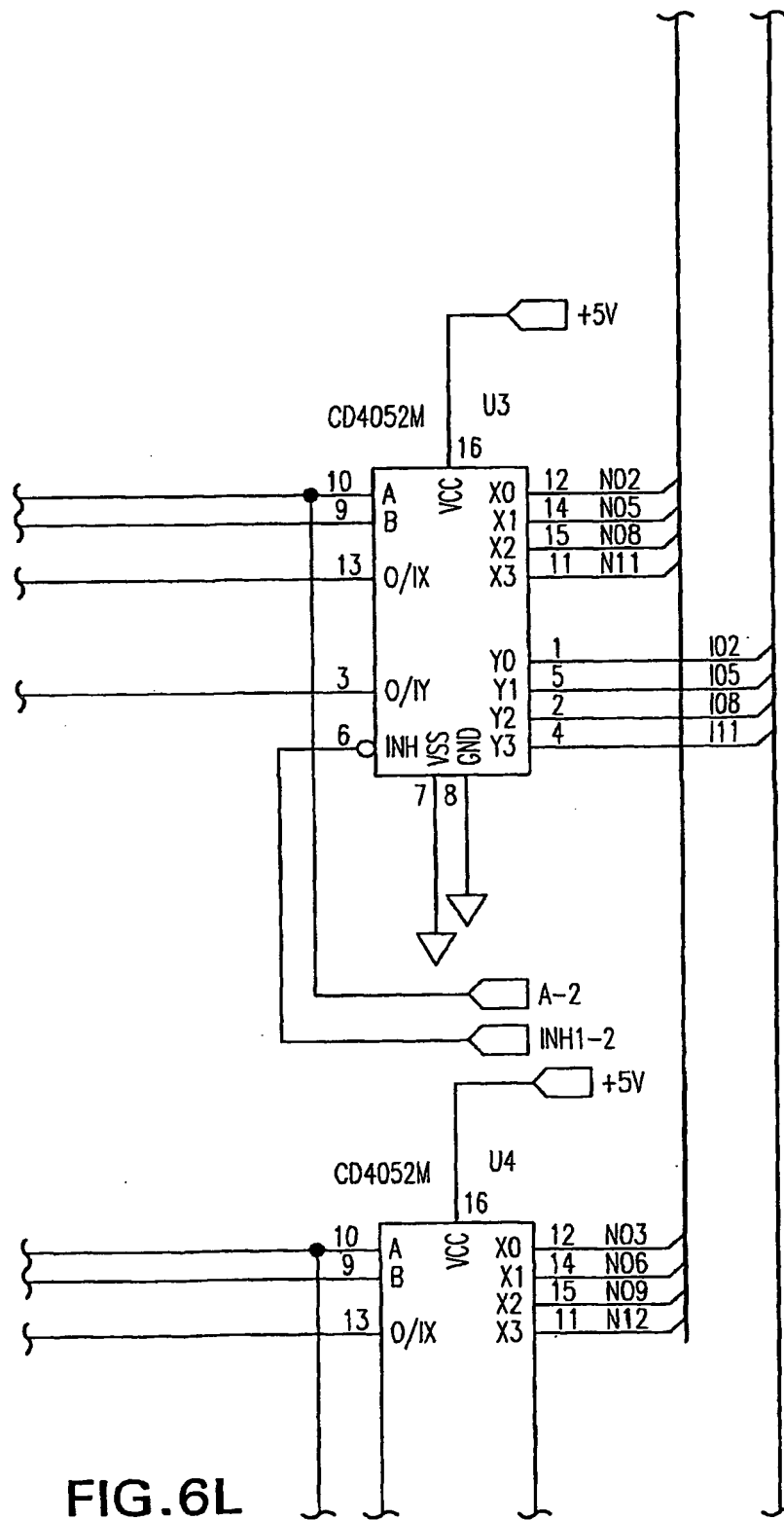


FIG.6K



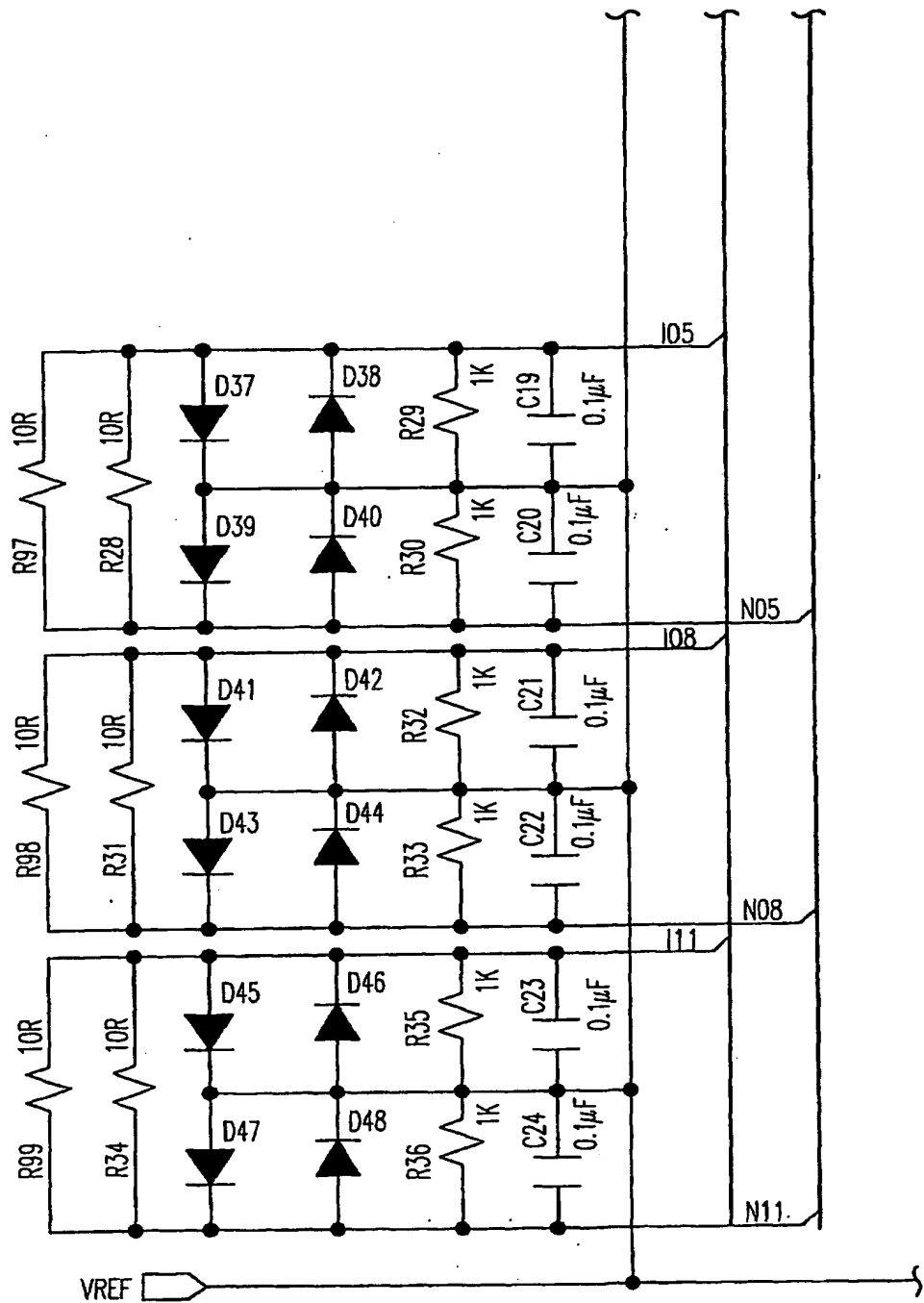


FIG.6M

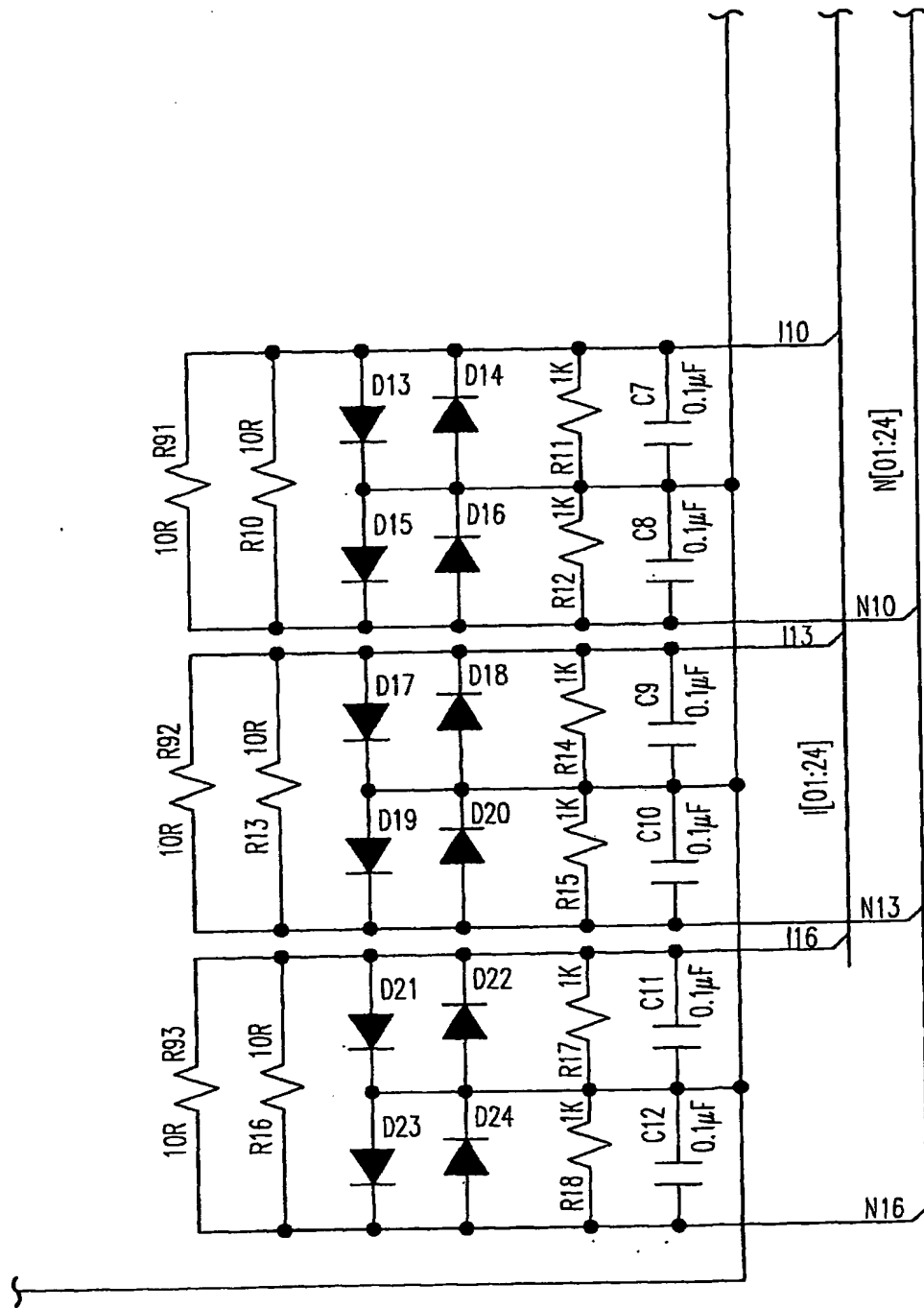


FIG.6N

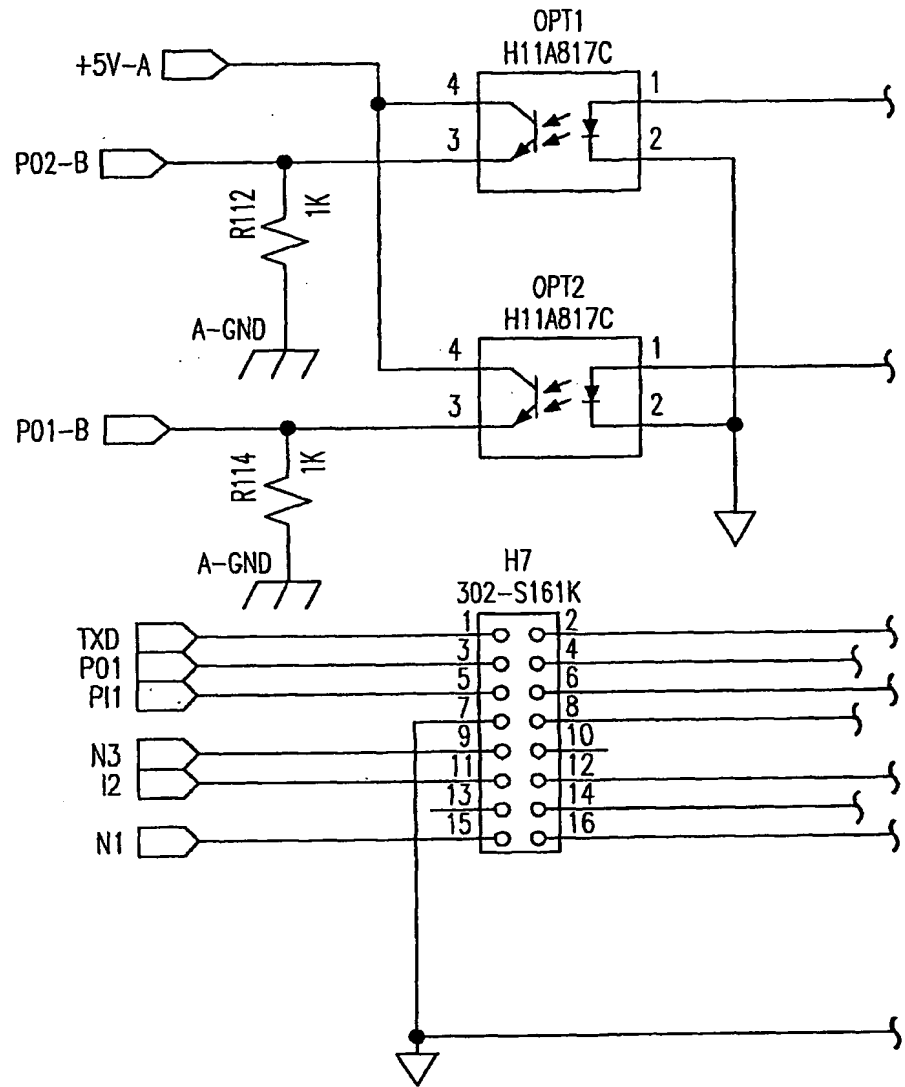


FIG. 60

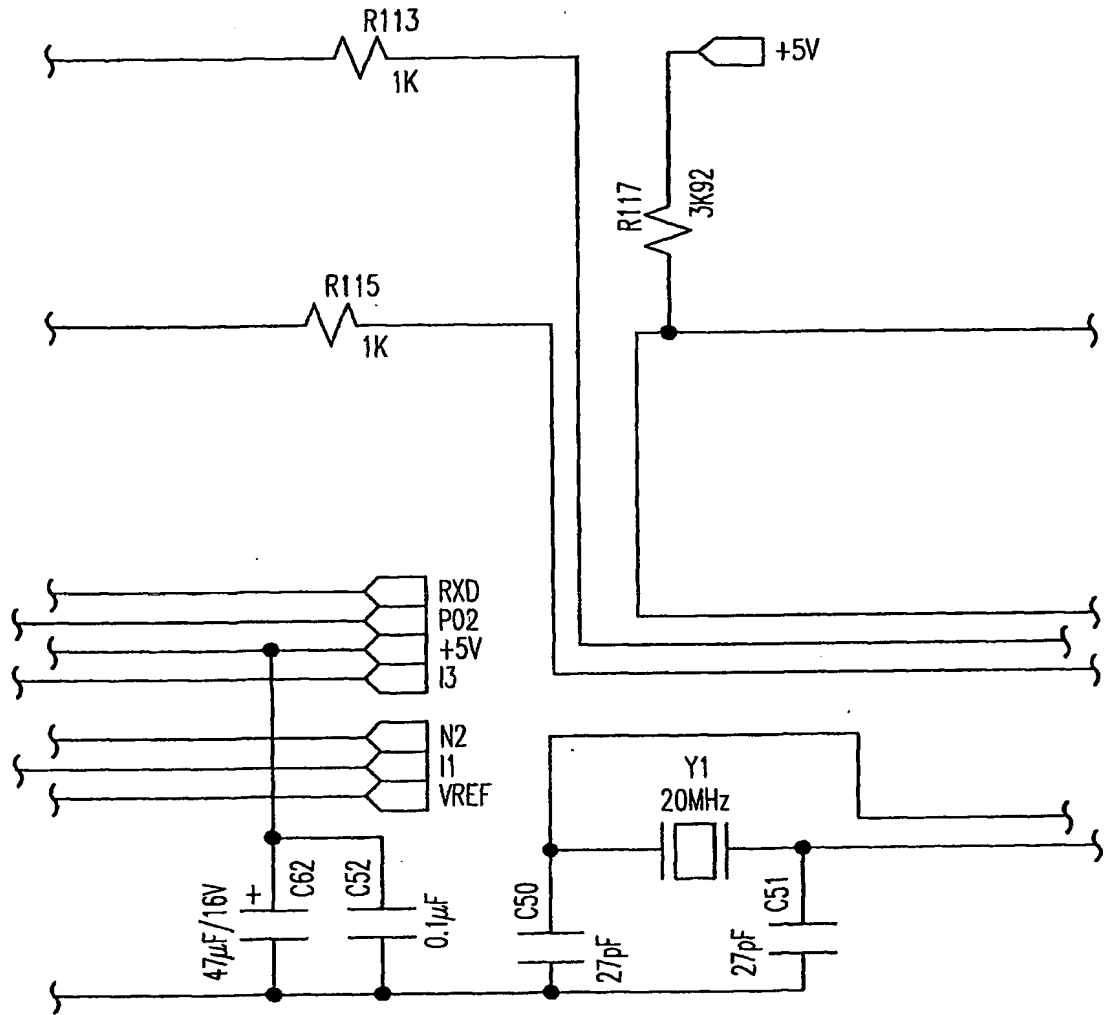


FIG.6P

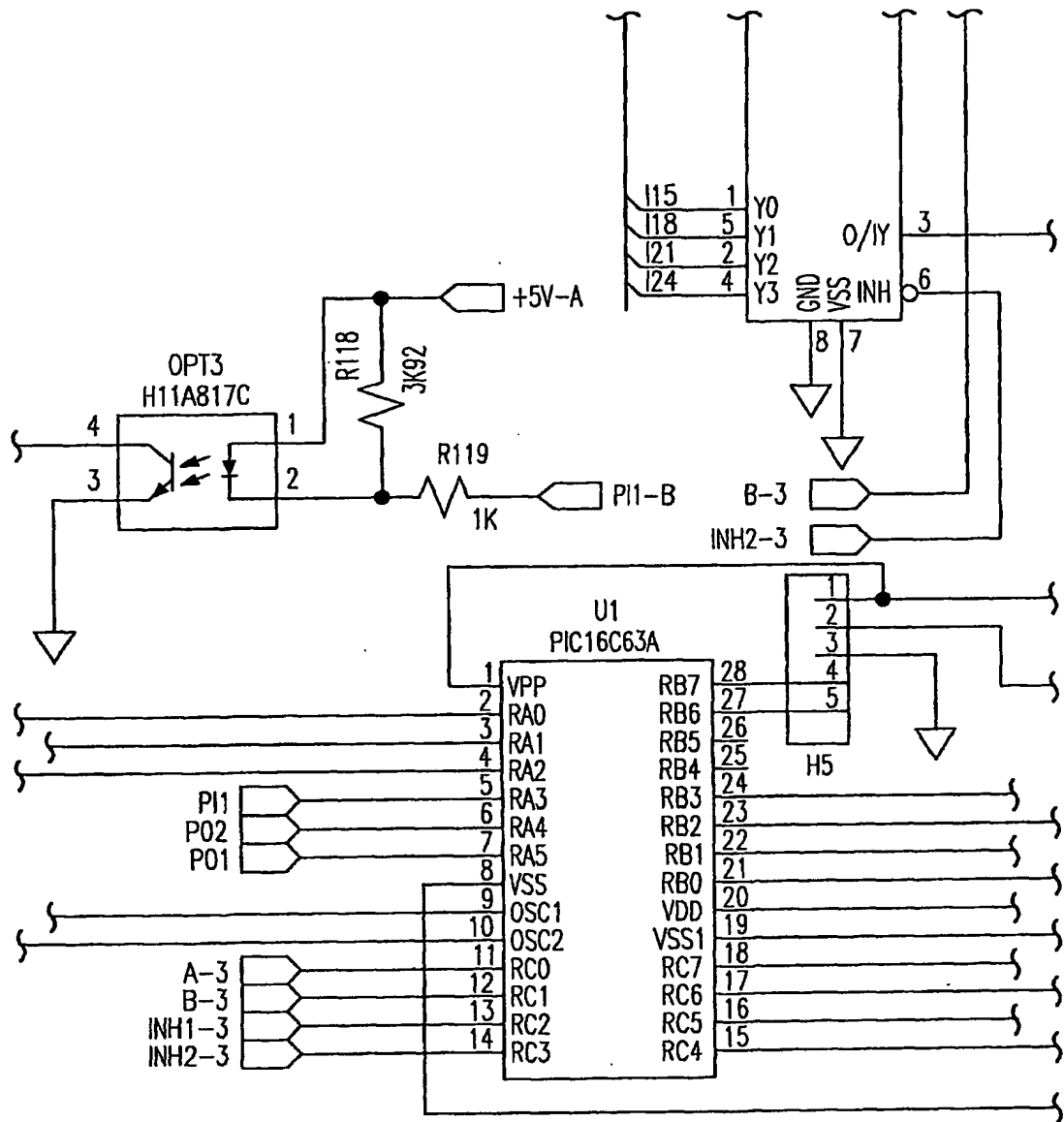


FIG.6Q

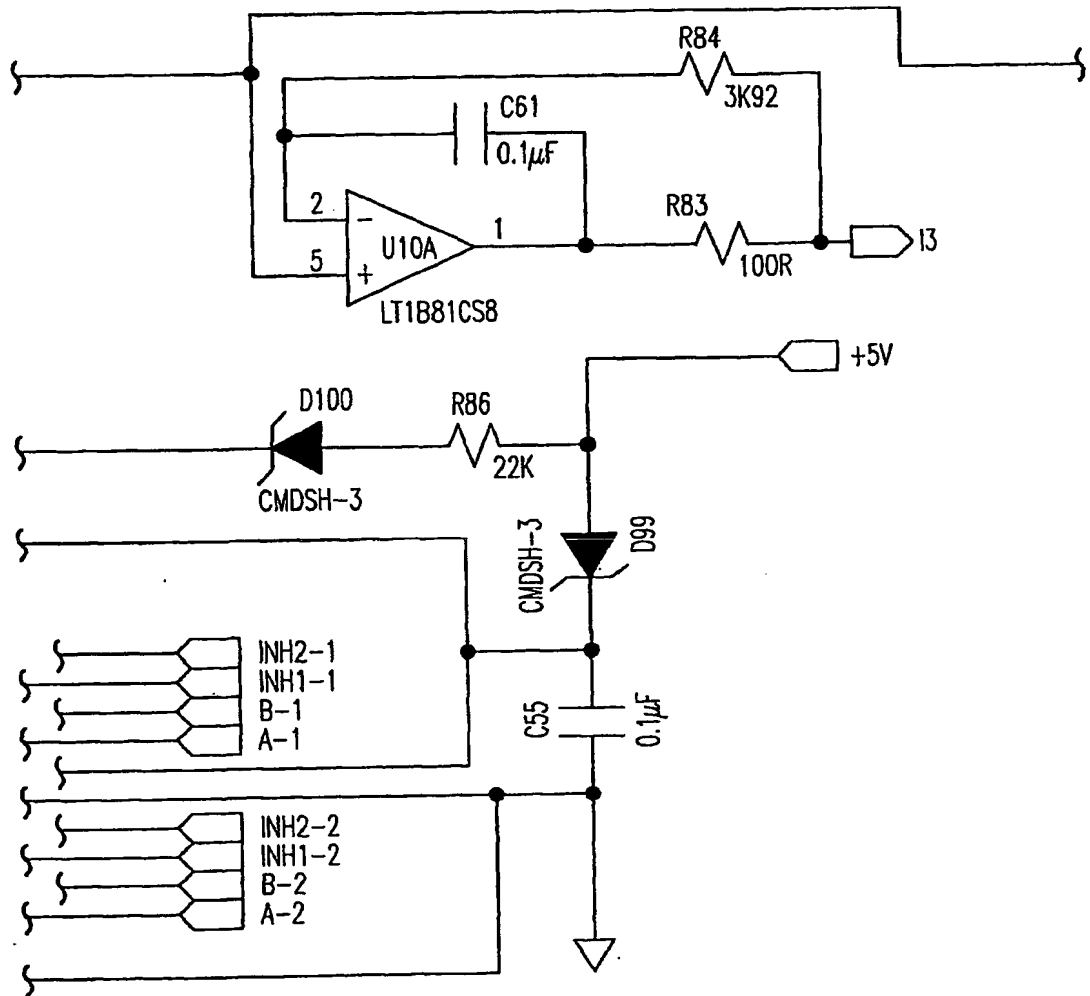


FIG. 6R

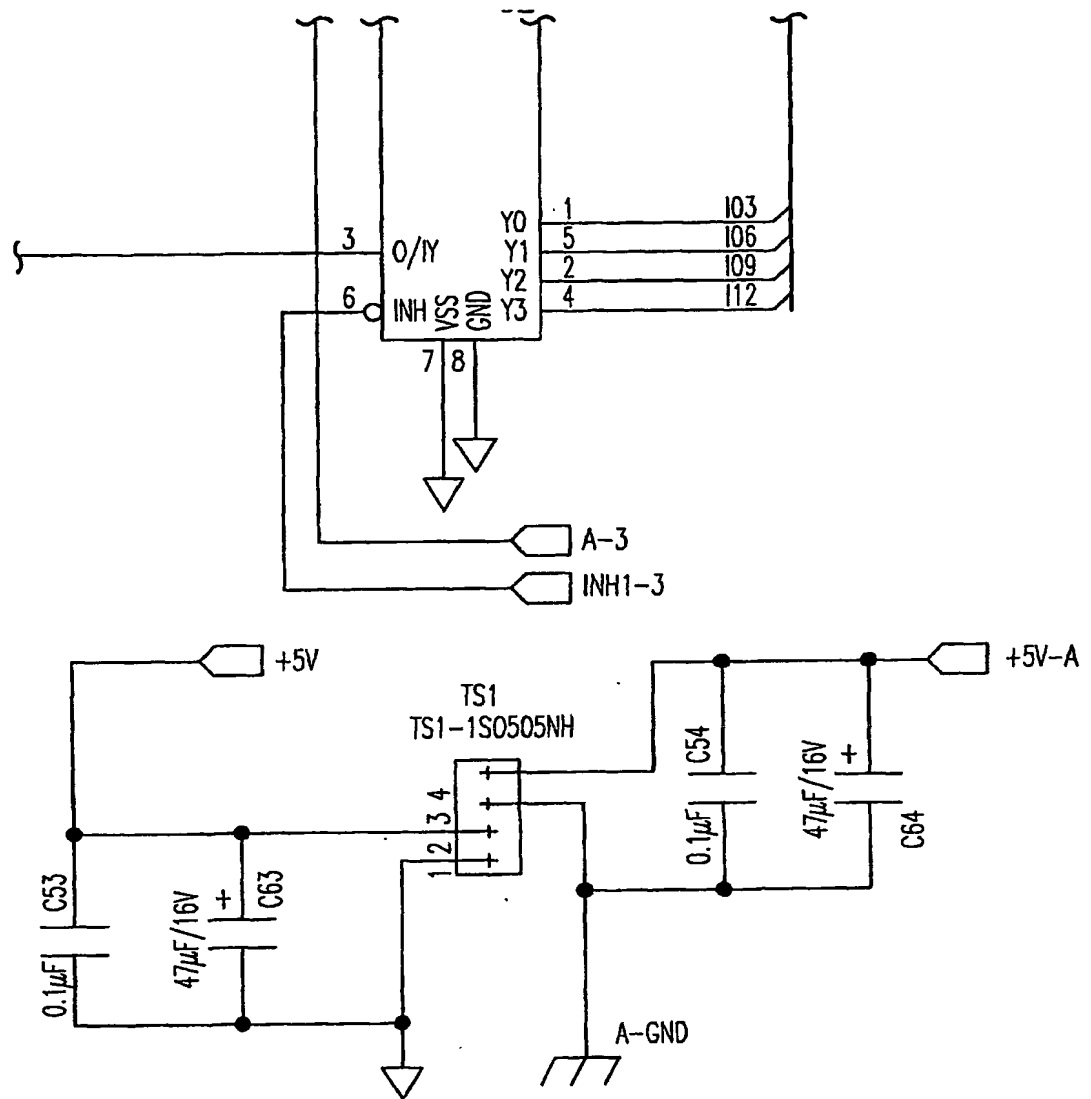


FIG.6S

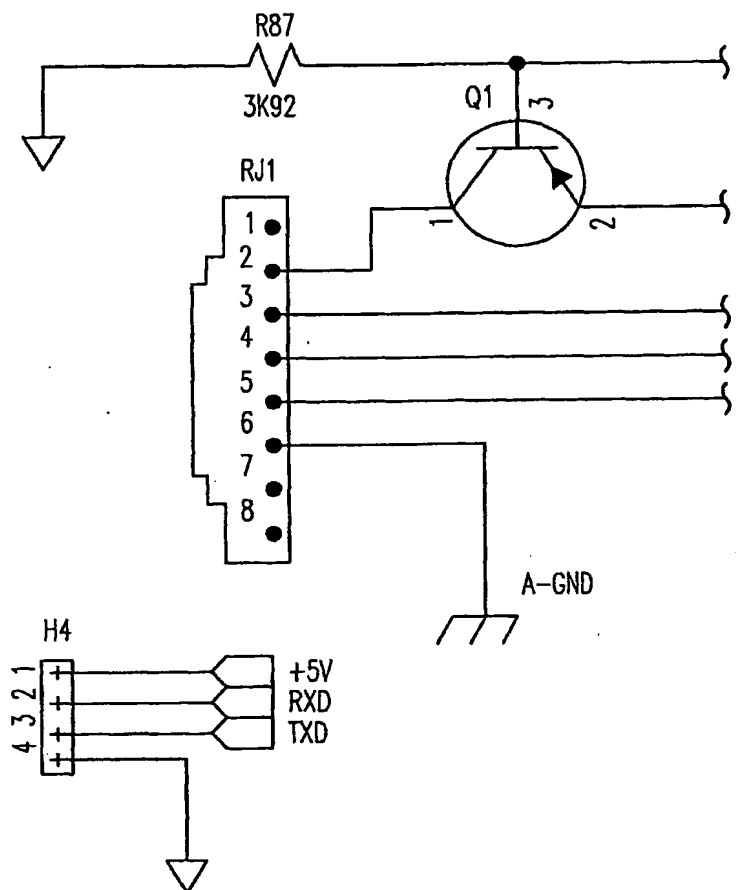


FIG.6T

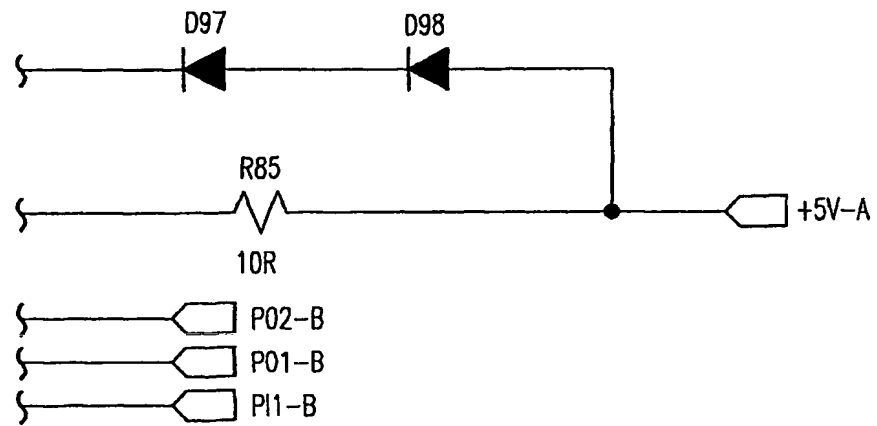


FIG.6U

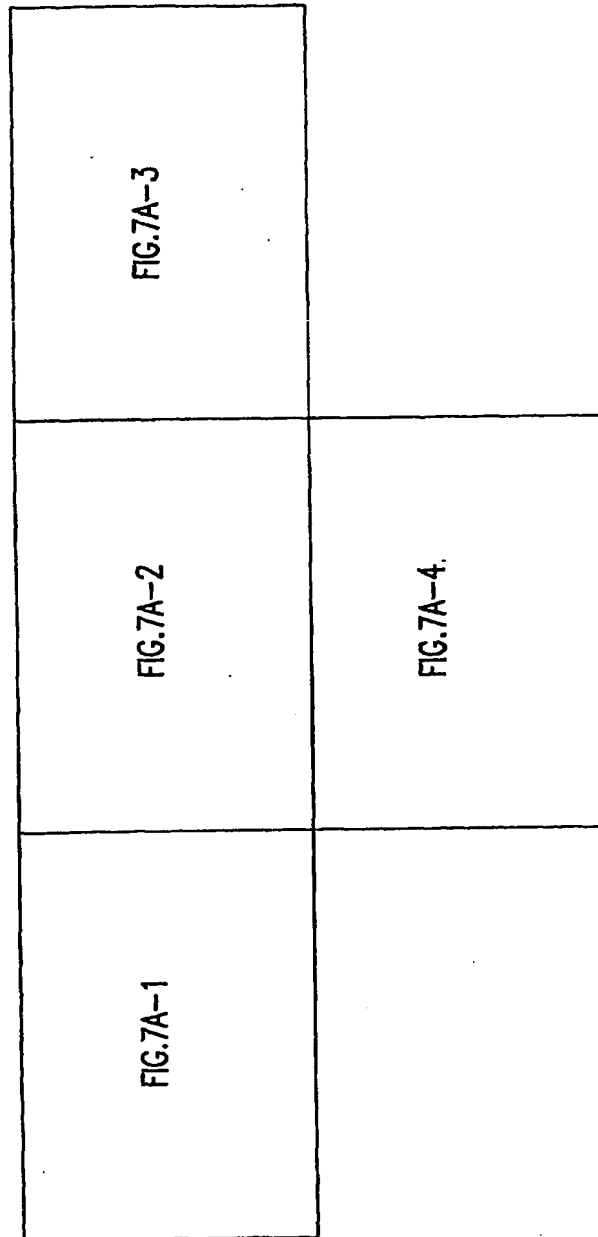
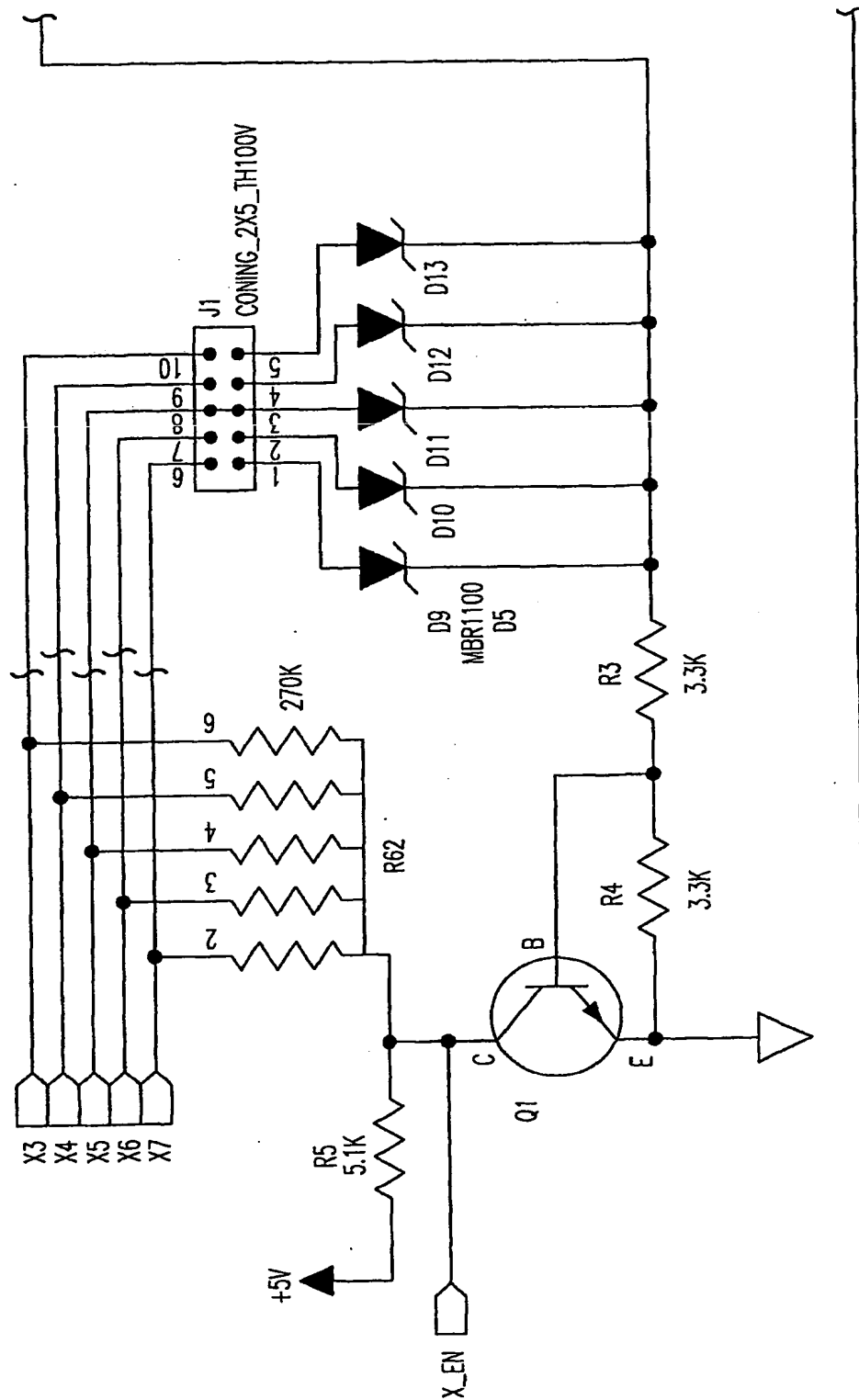
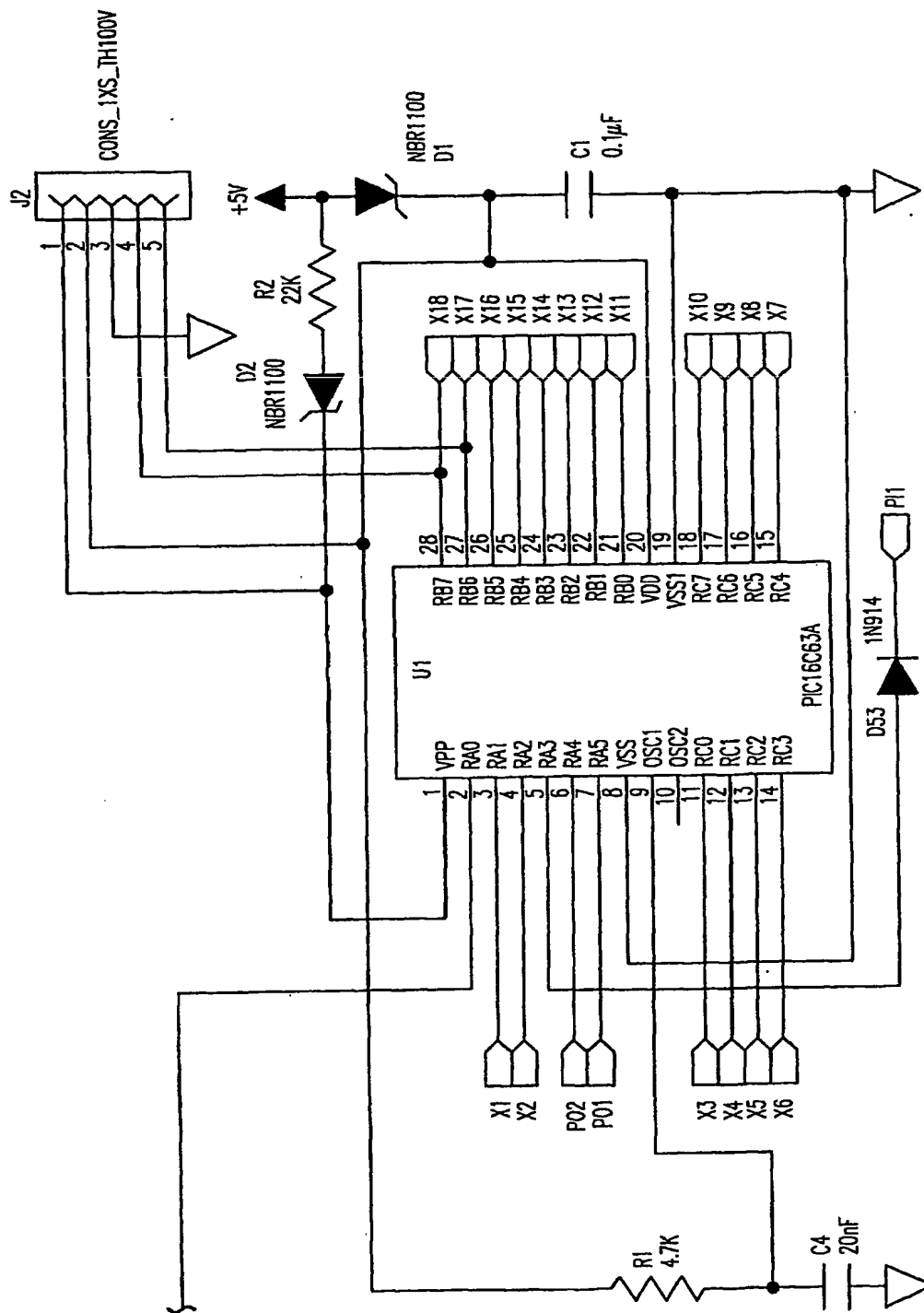


FIG.7A





A- +5V CIRCUITO DE CONTROLE

+12V CIRCUITO DE EXCITAÇÃO

FIG.7A-2

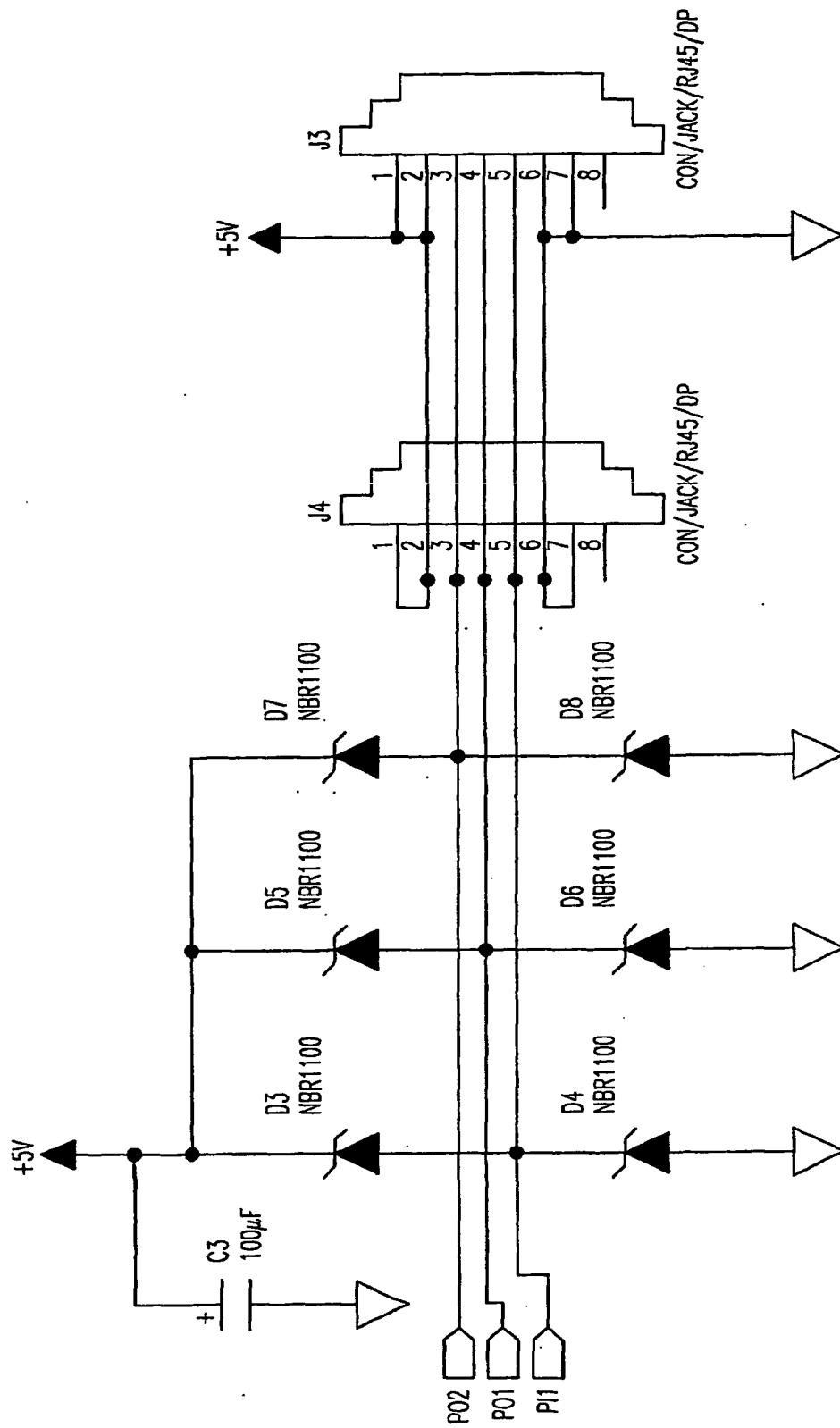


FIG.7A-3

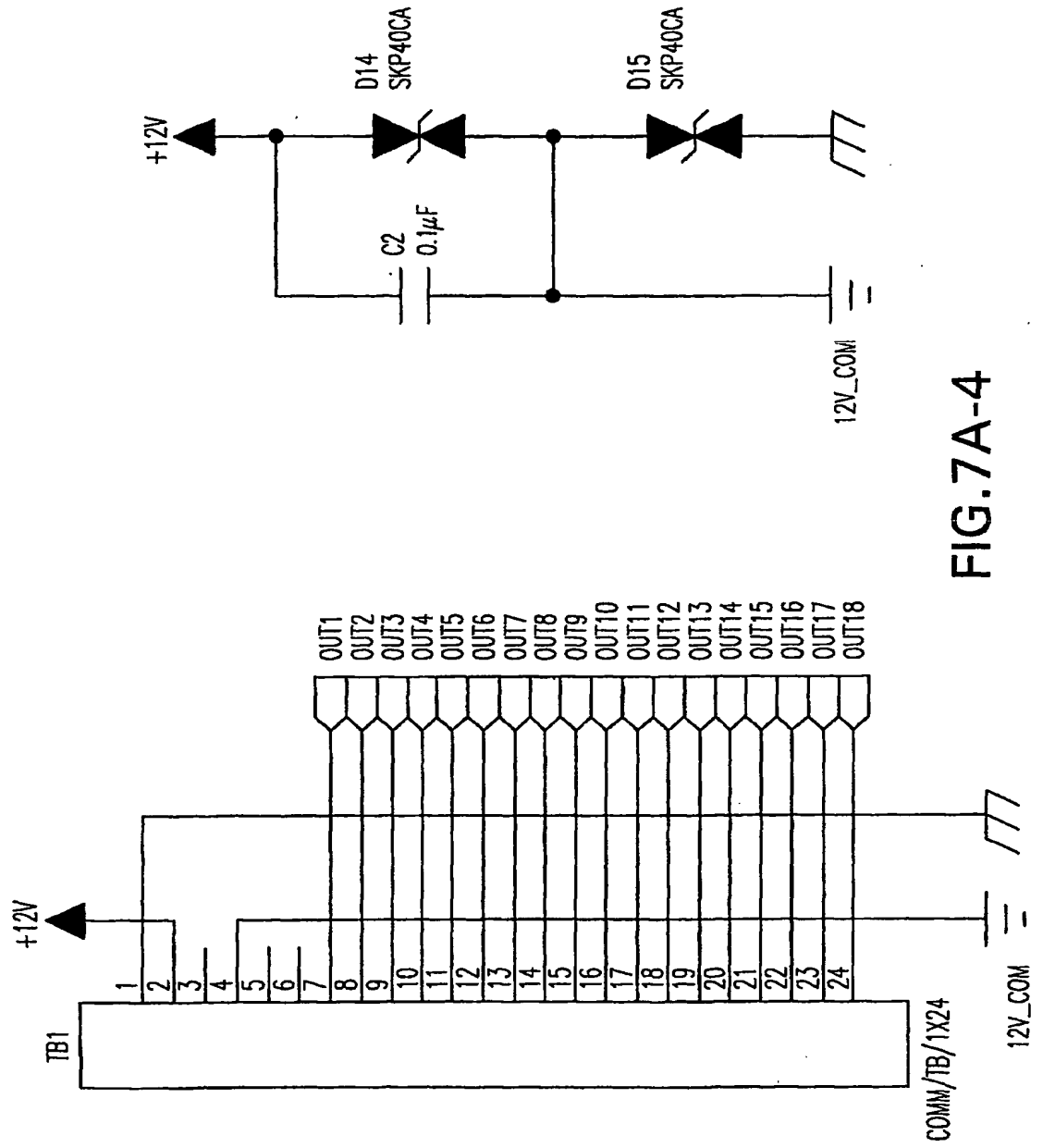


FIG. 7A-4

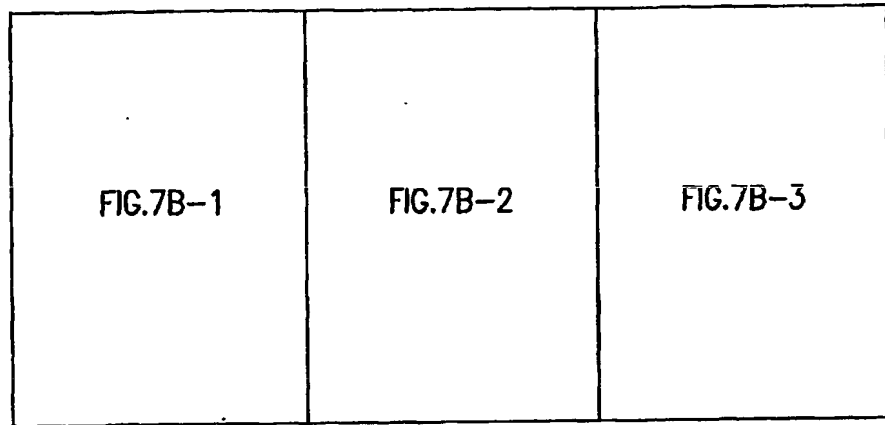


FIG.7B

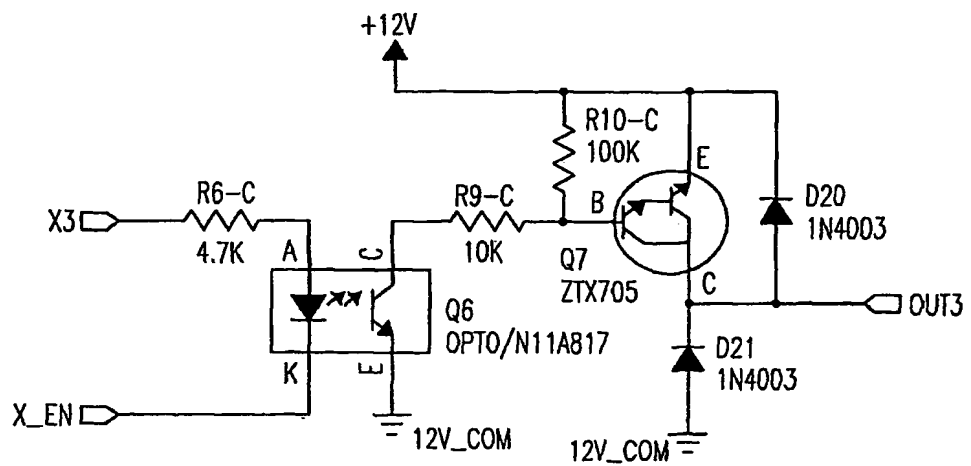
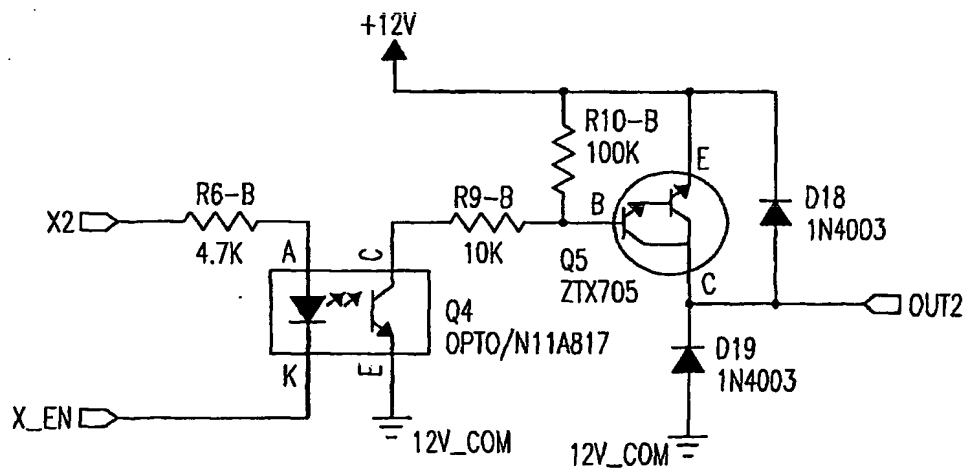
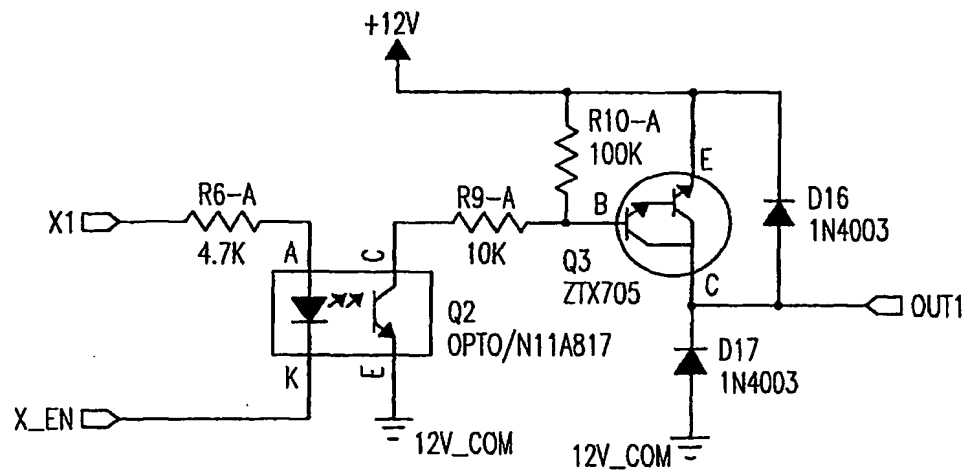


FIG.7B-1

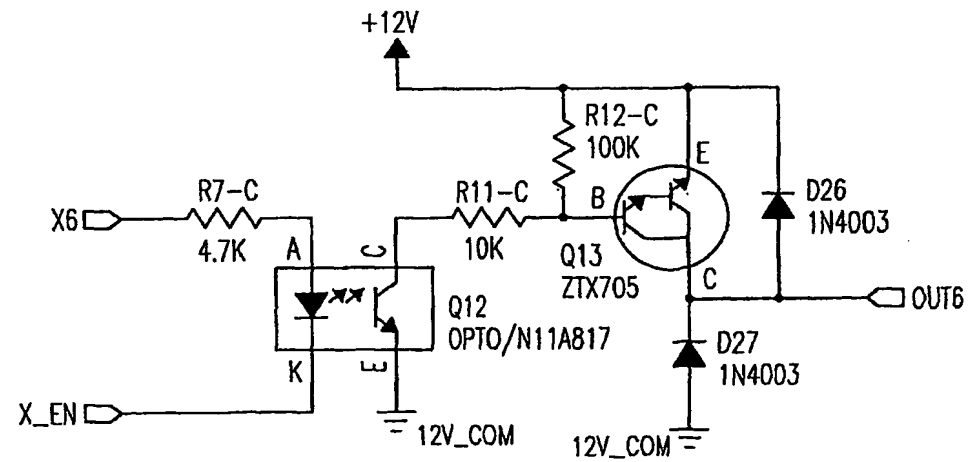
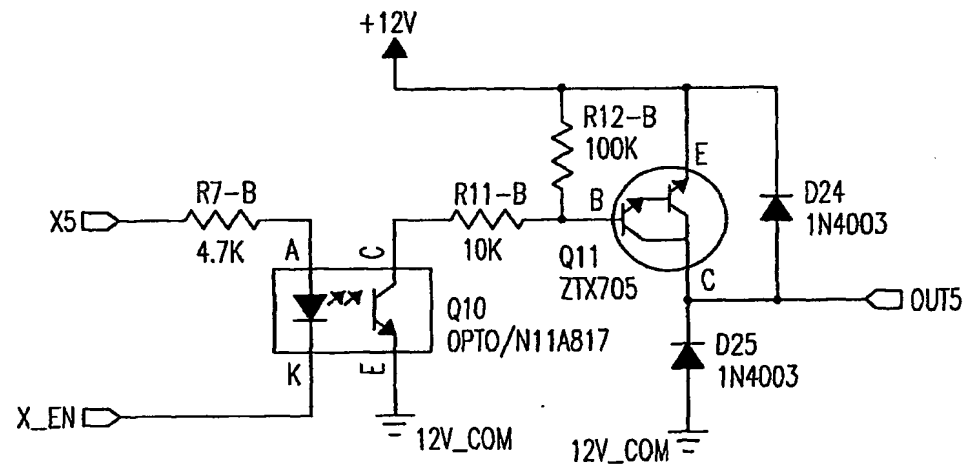
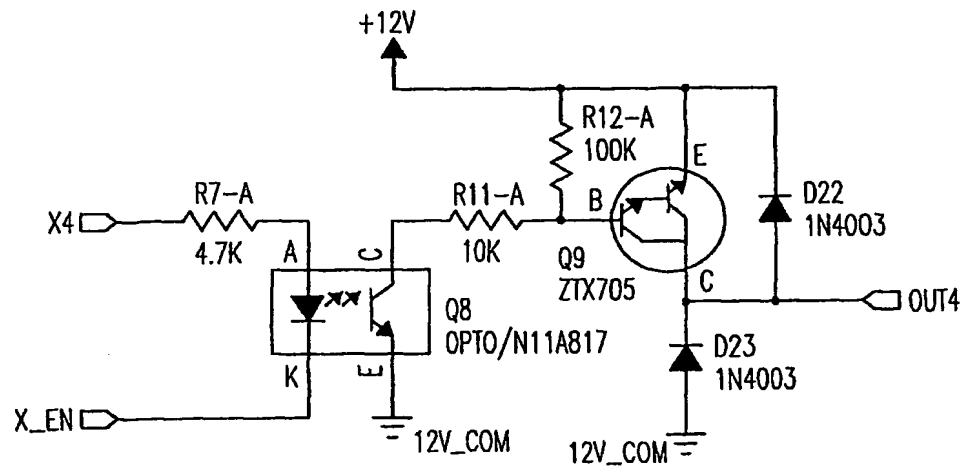


FIG.7B-2

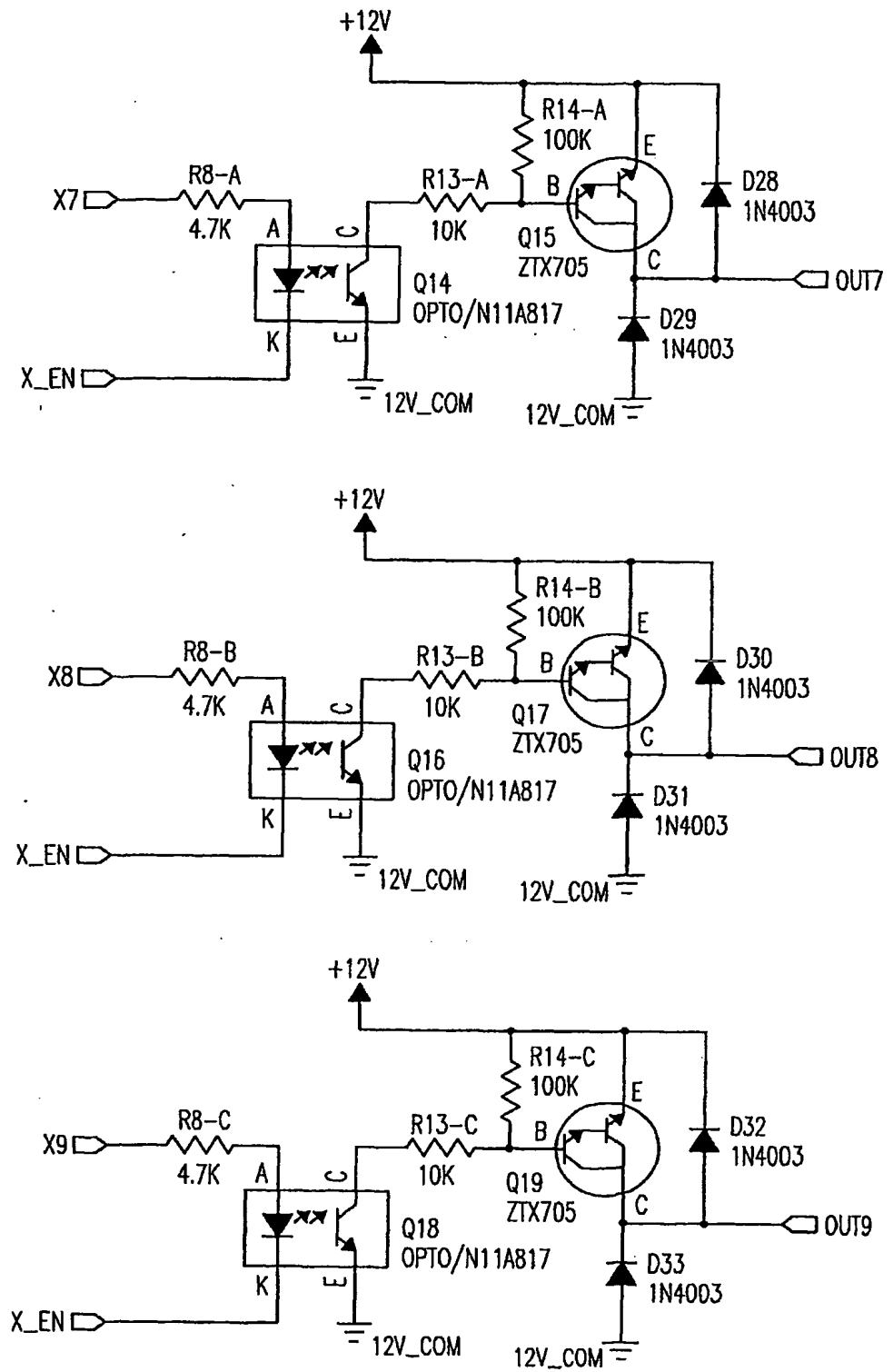


FIG.7B-3

FIG.7C-1	FIG.7C-2	FIG.7C-3

FIG.7C

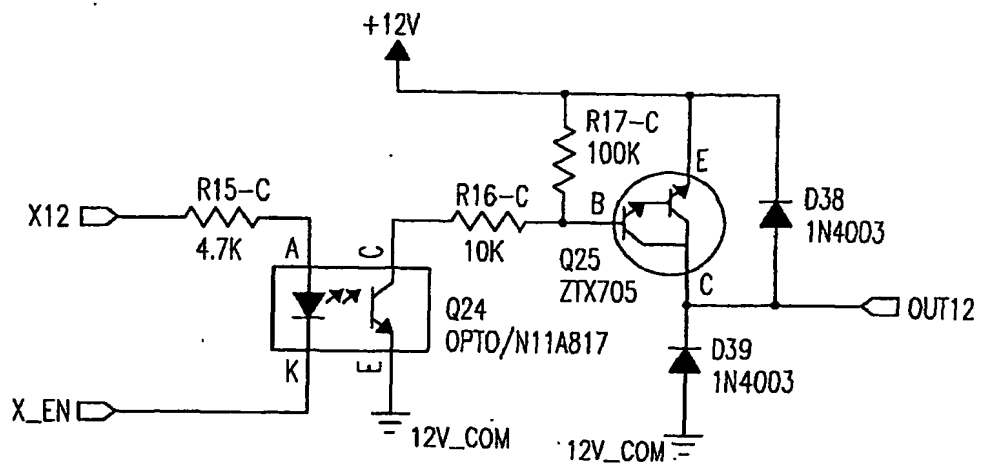
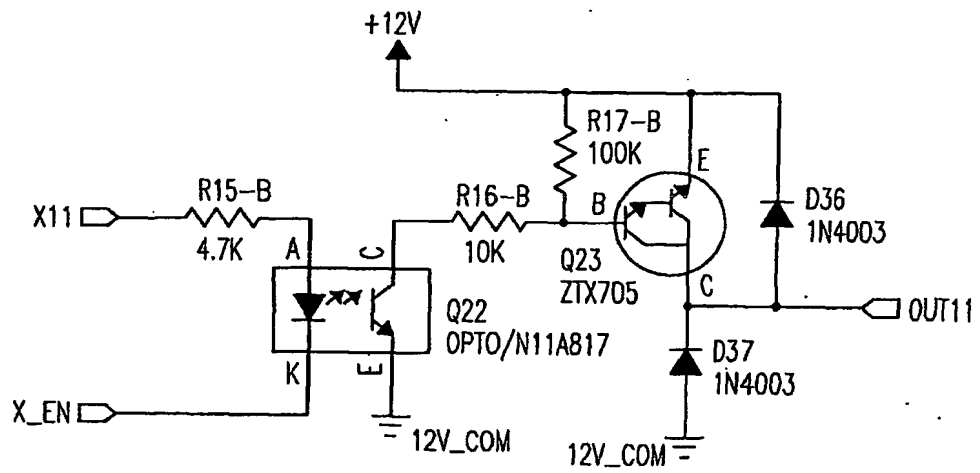
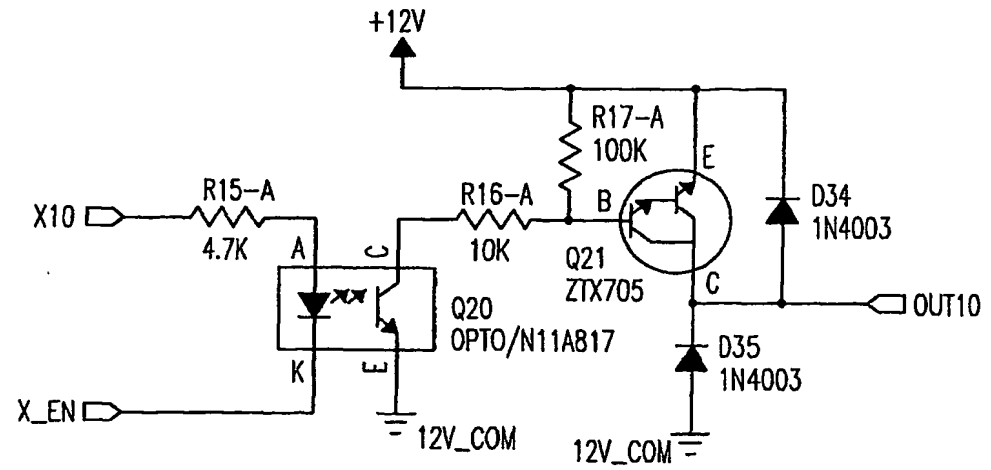


FIG.7C-1

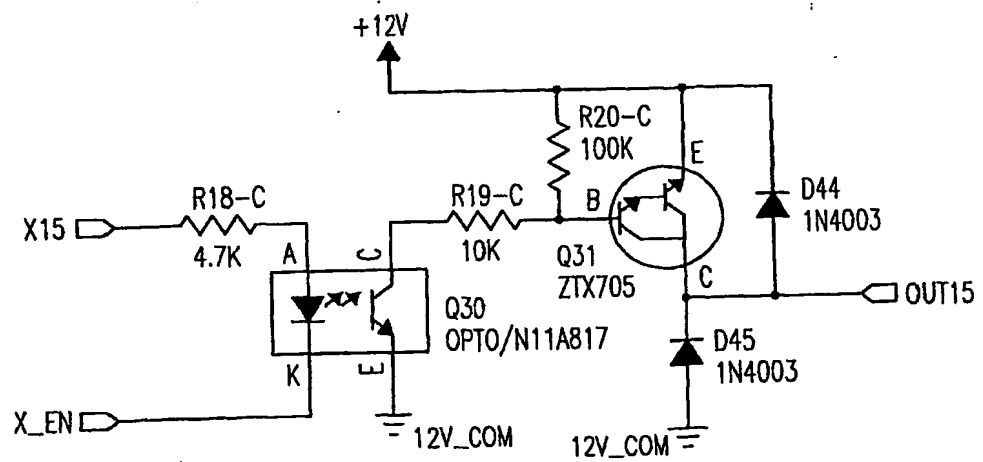
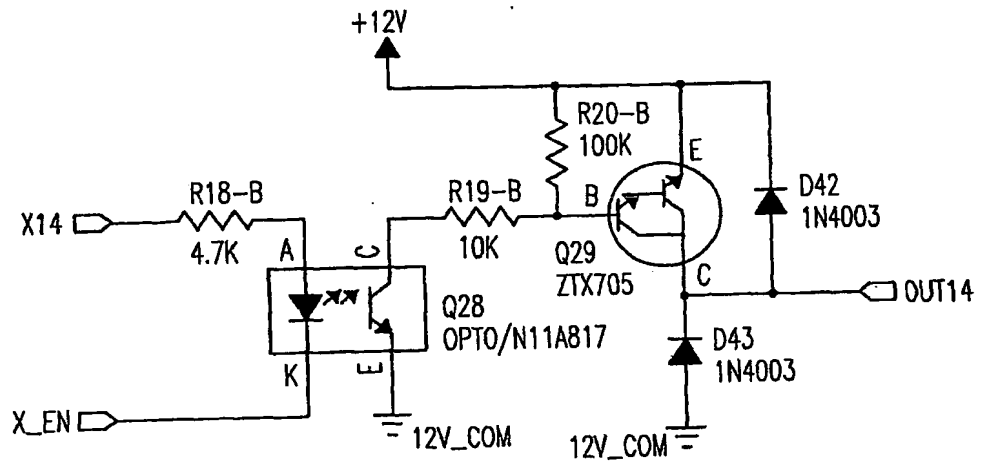
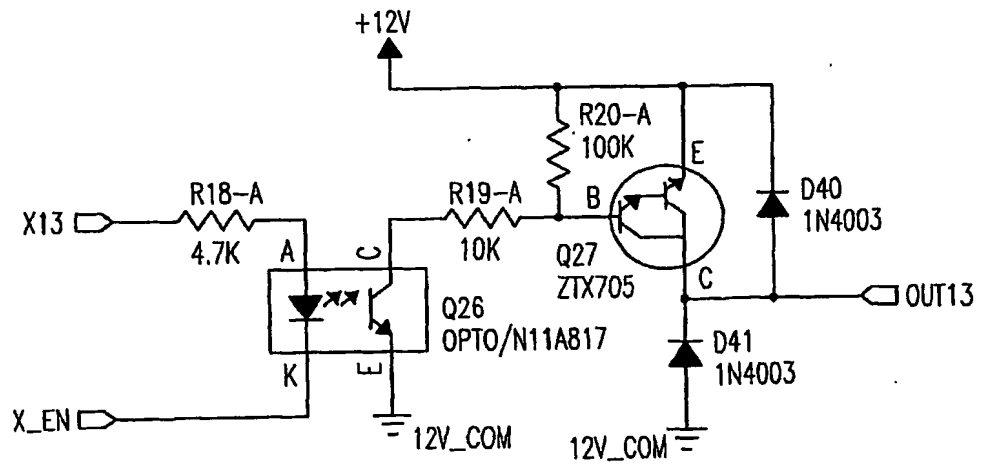


FIG.7C-2

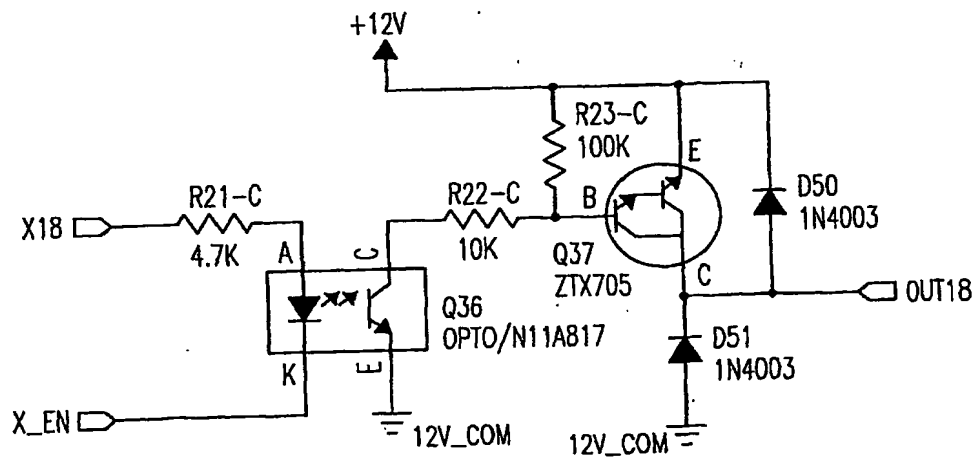
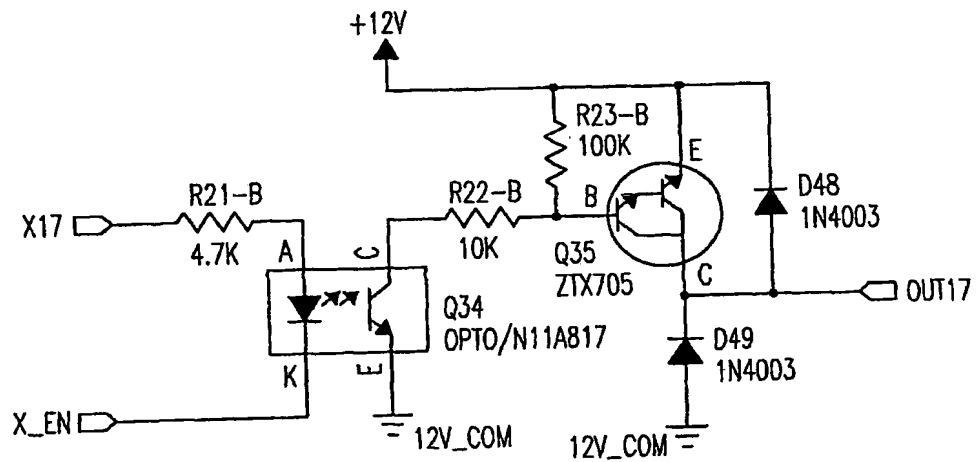
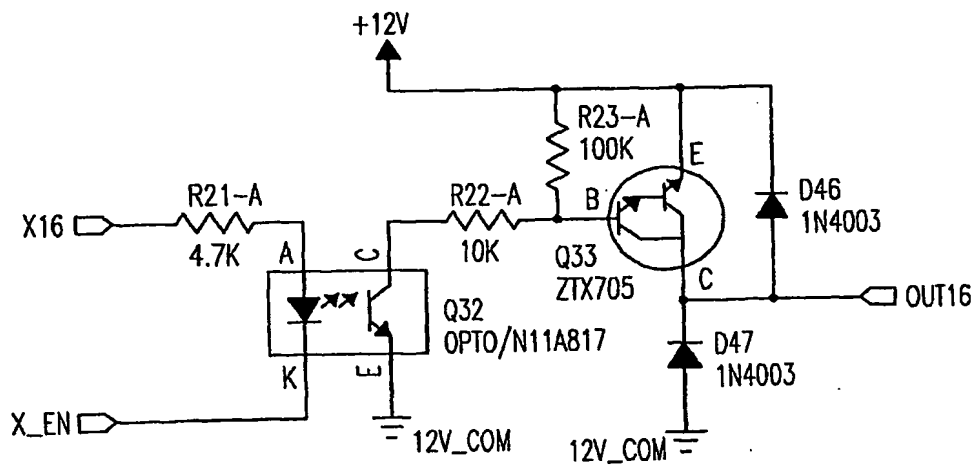


FIG.7C-3

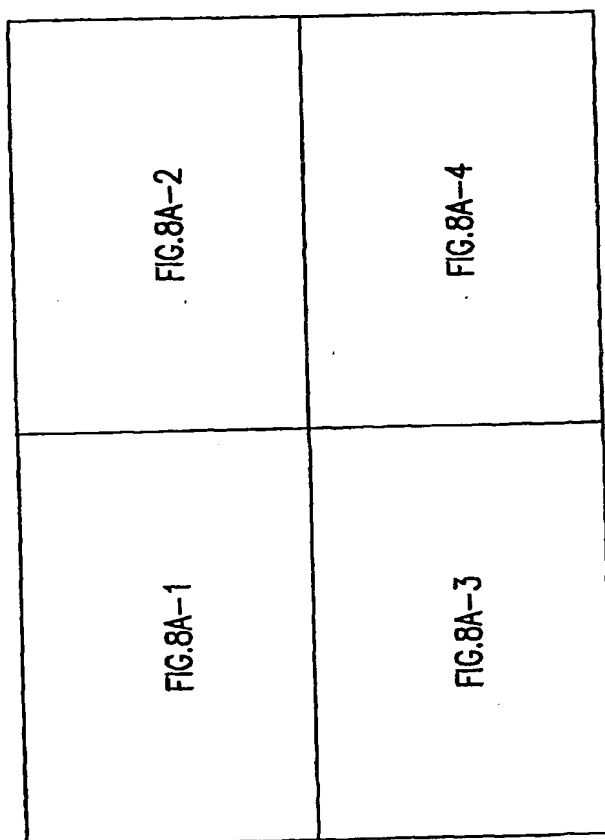


FIG.8A

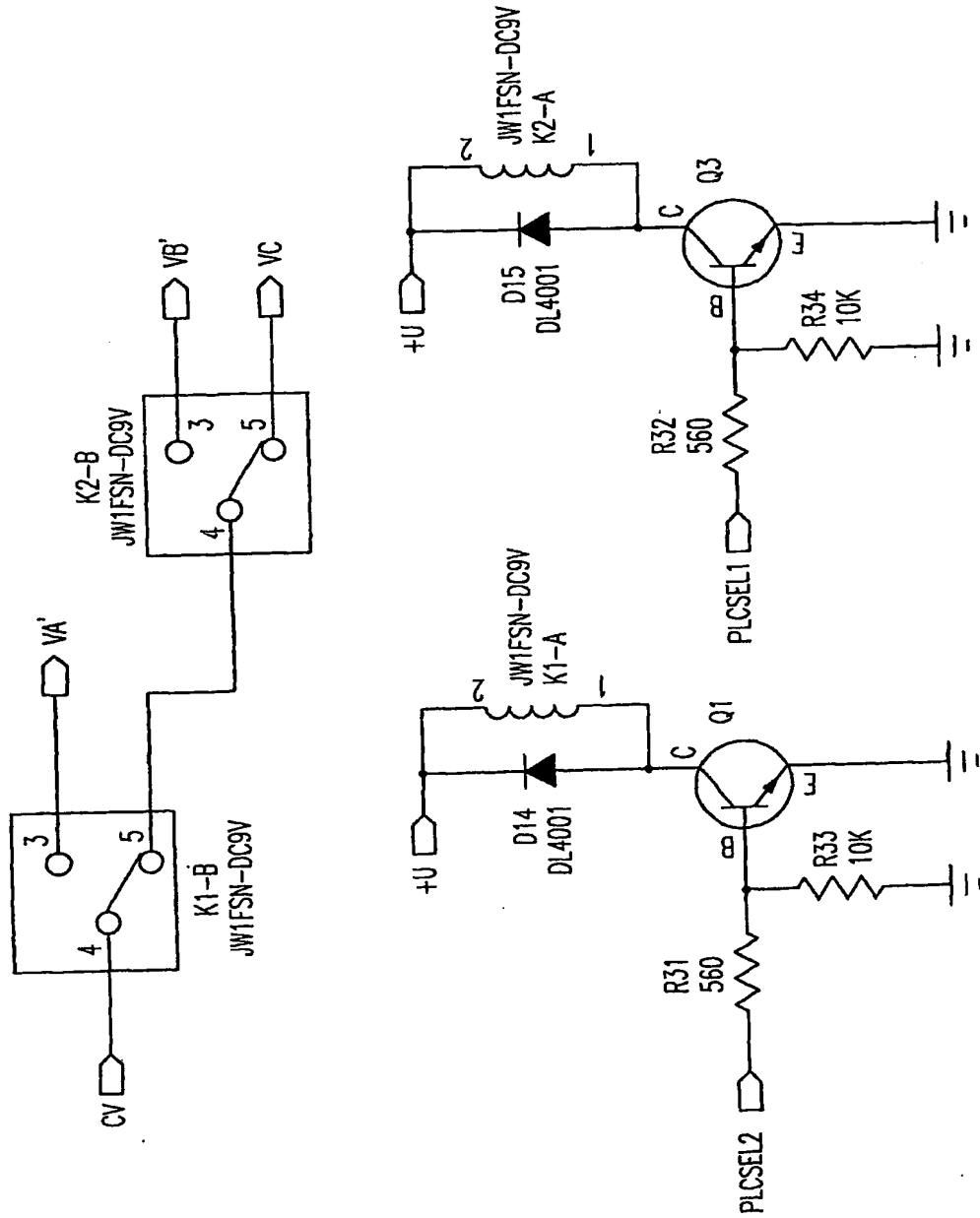


FIG.8A-1

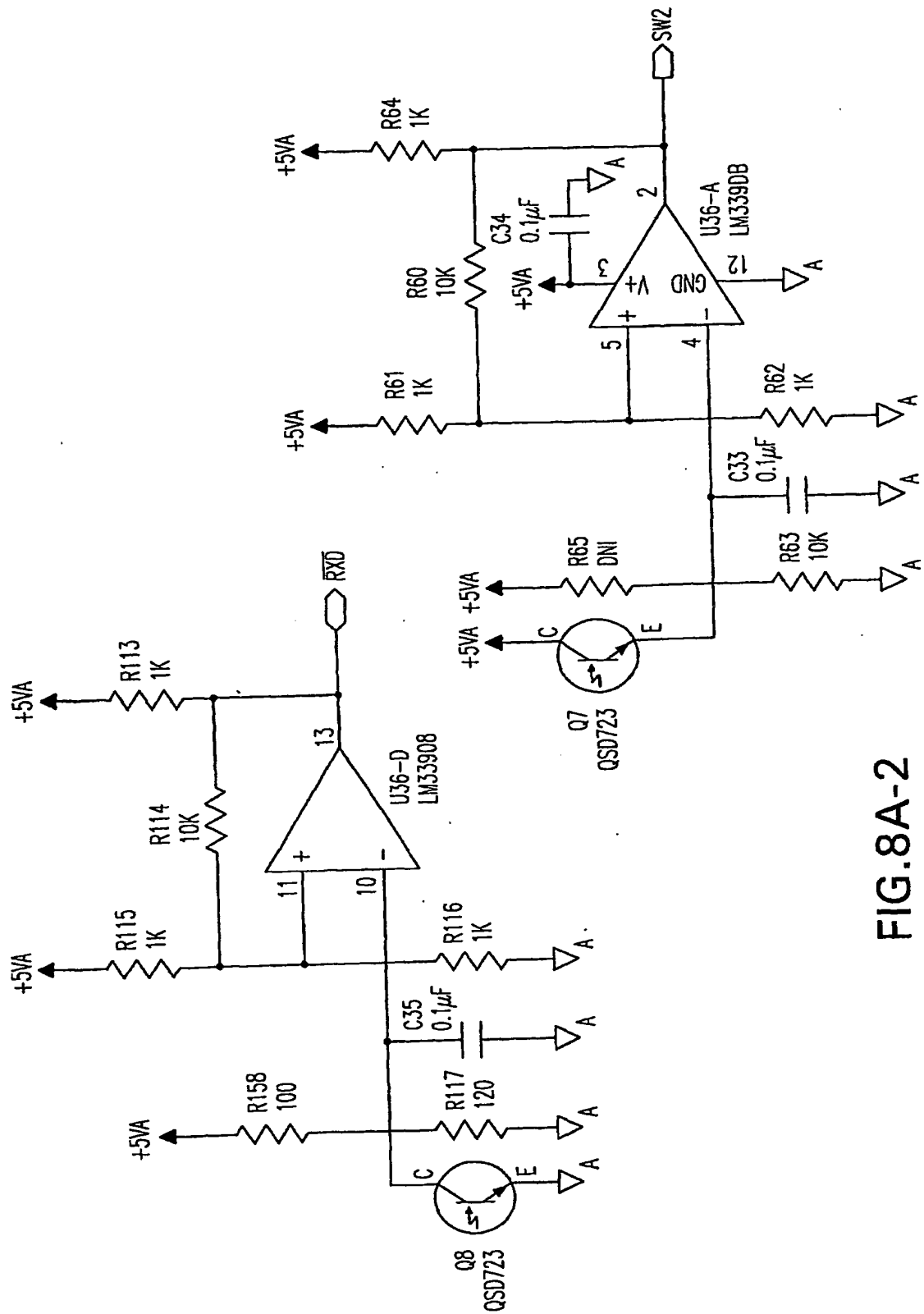
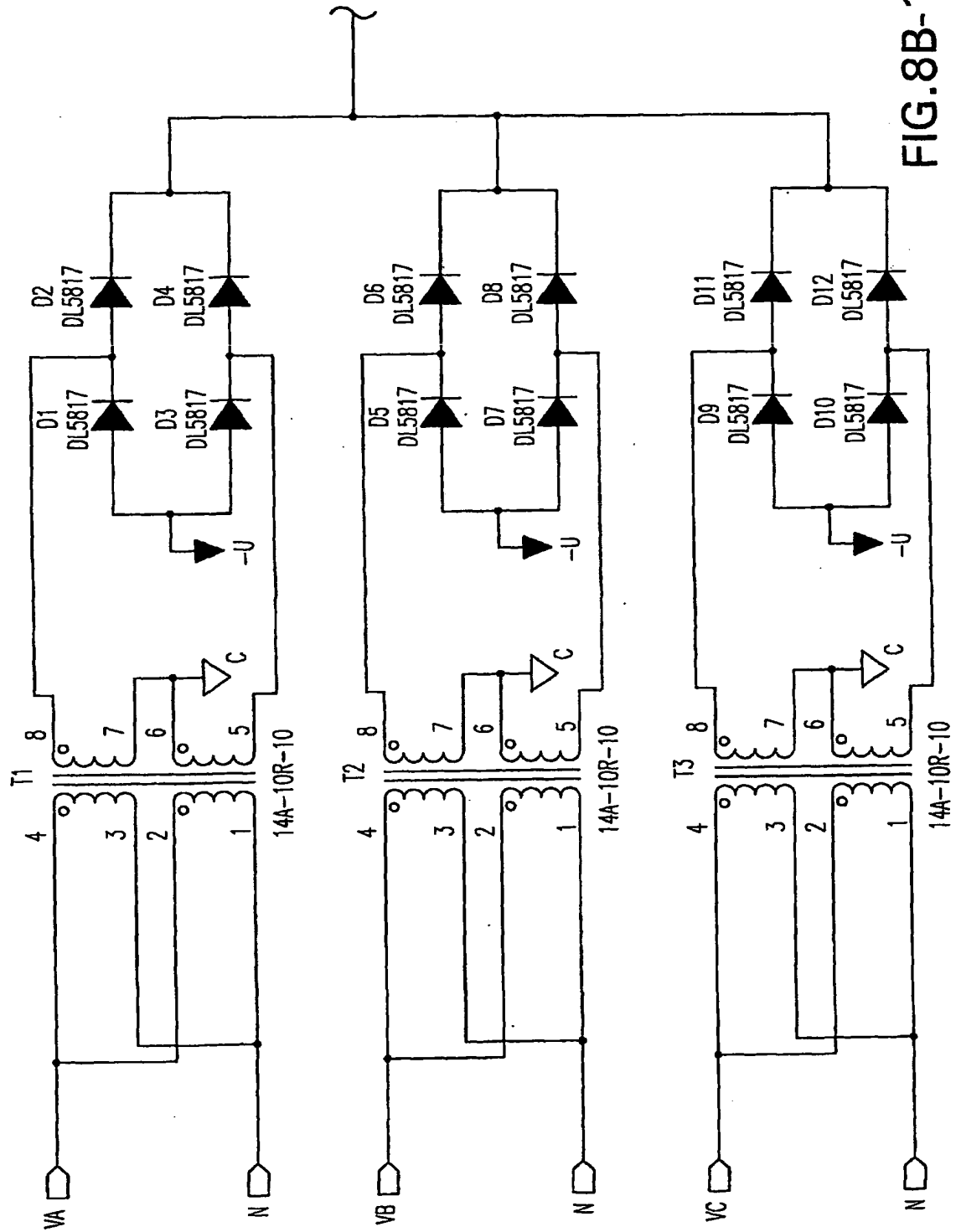


FIG. 8A-2

FIG.8B-1	FIG.8B-2
FIG.8B-3	FIG.8B-4

FIG.8B



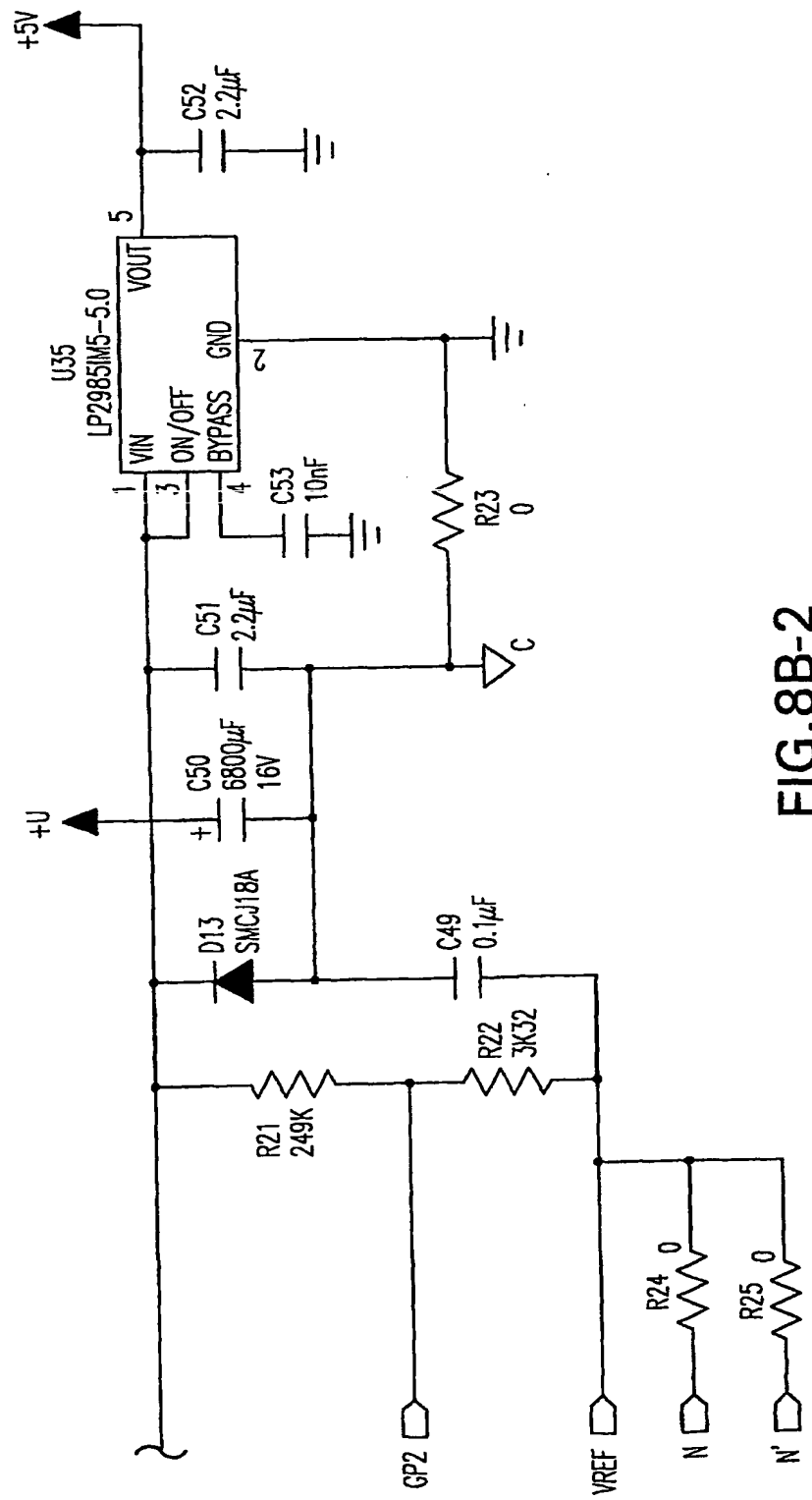


FIG.8B-2

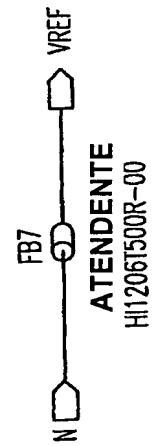
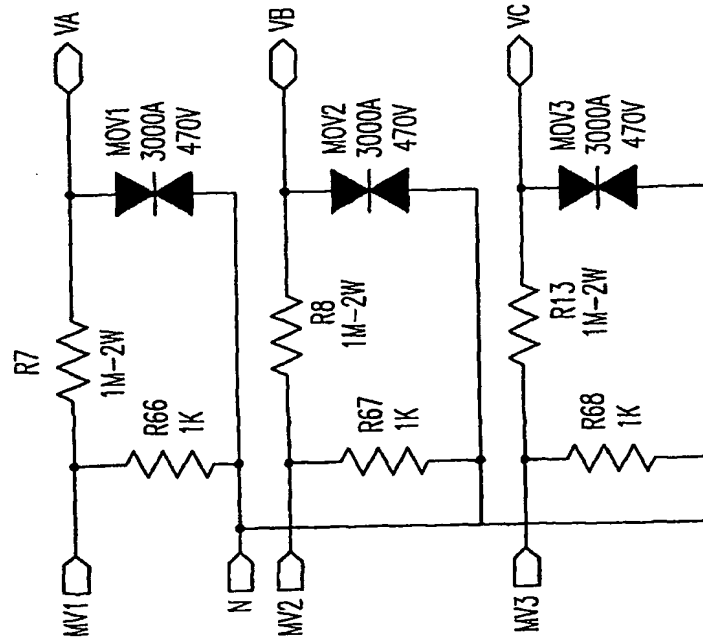
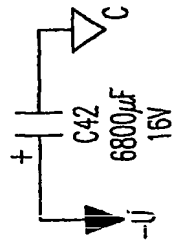


FIG. 8B-3

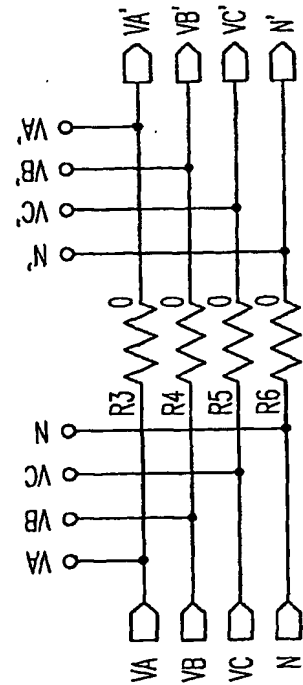


FIG. 8B-4

FIG.8C-1	FIG.8C-2	FIG.8C-3	FIG.8C-4	FIG.8C-5
FIG.8C-6	FIG.8C-7	FIG.8C-8	FIG.8C-9	FIG.8C-10

NOTA: PARA ENTRADAS 1 E 2 SE REFIRA A H4 DA FIGURA 3

FIG.8C

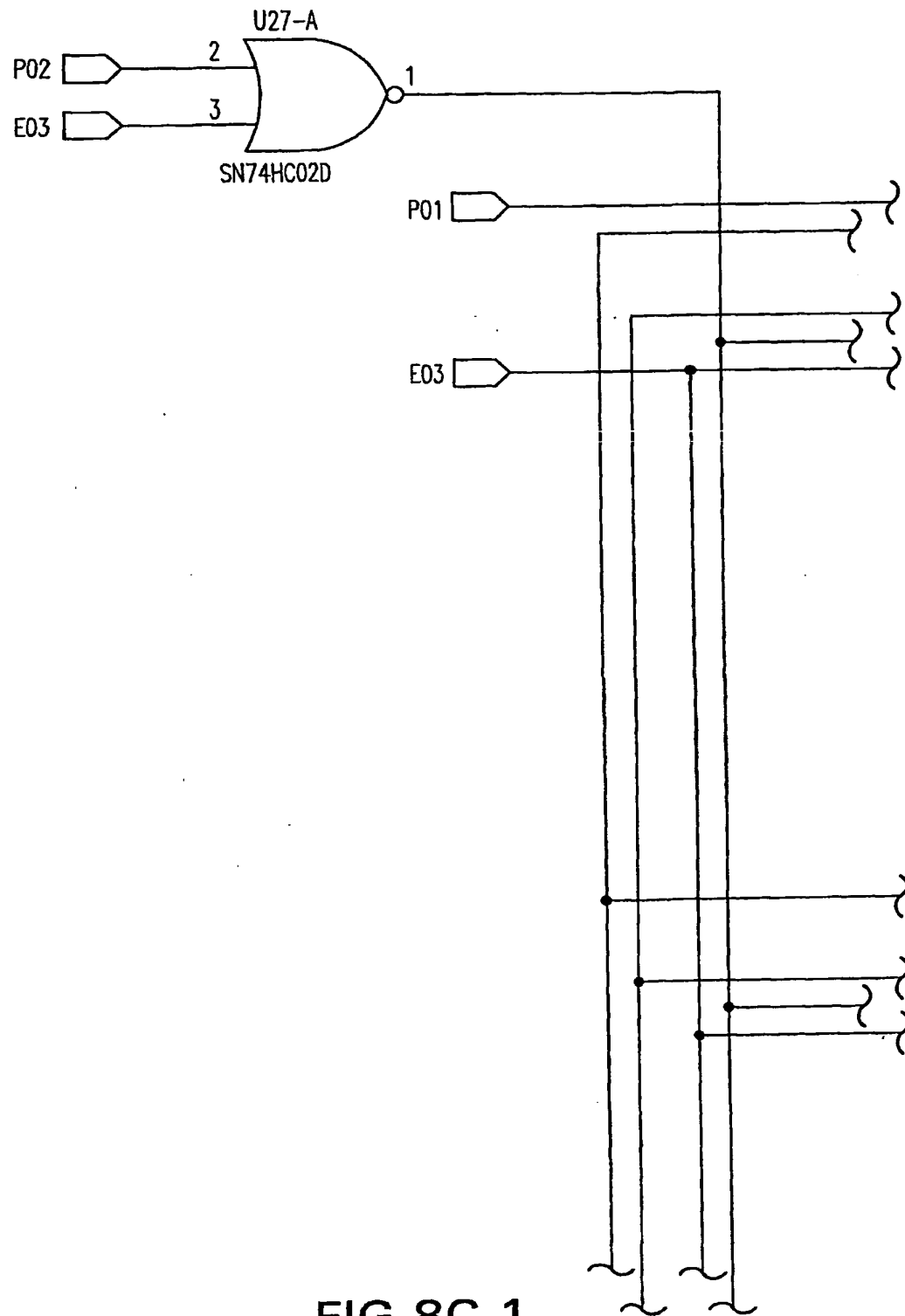


FIG.8C-1

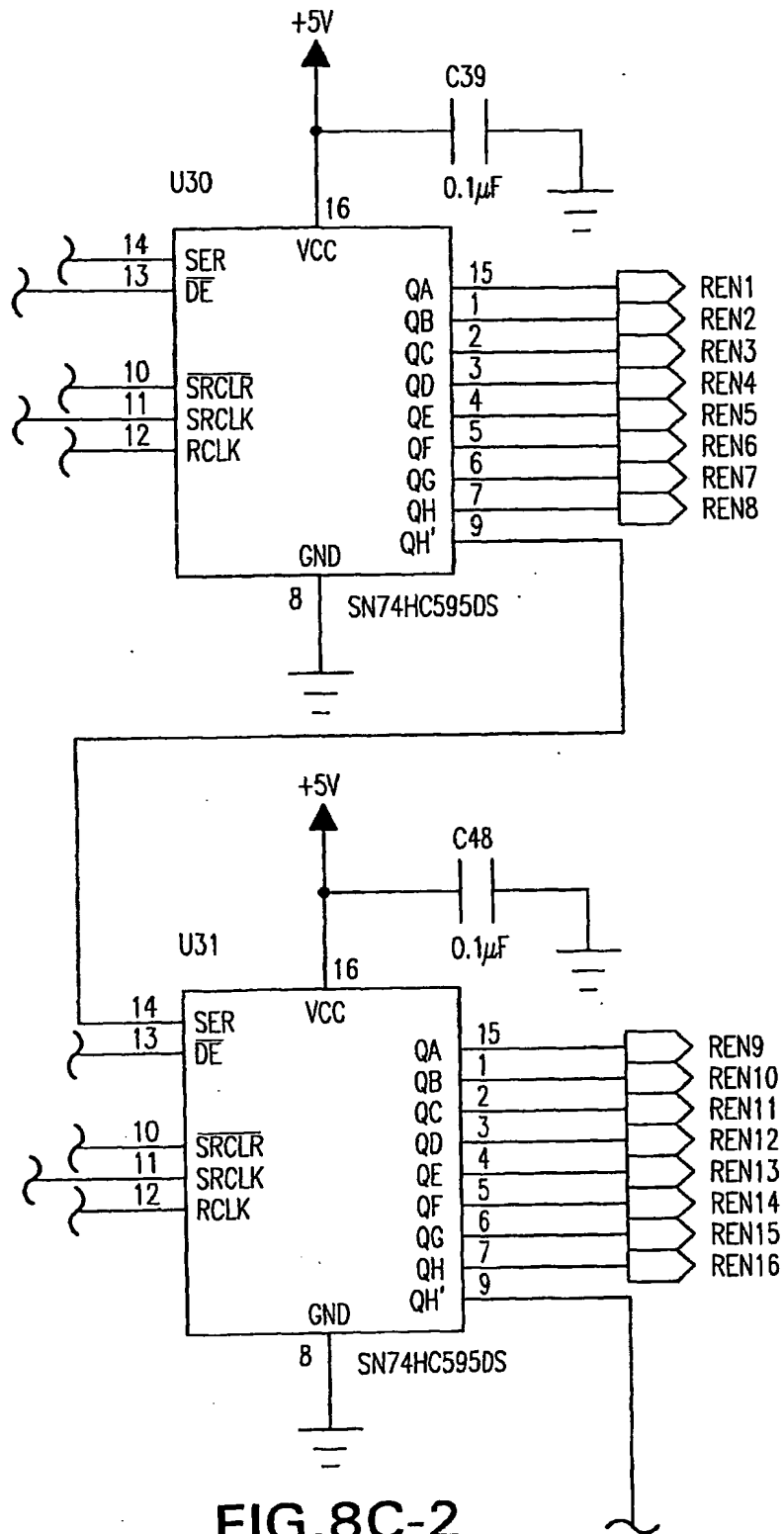


FIG.8C-2

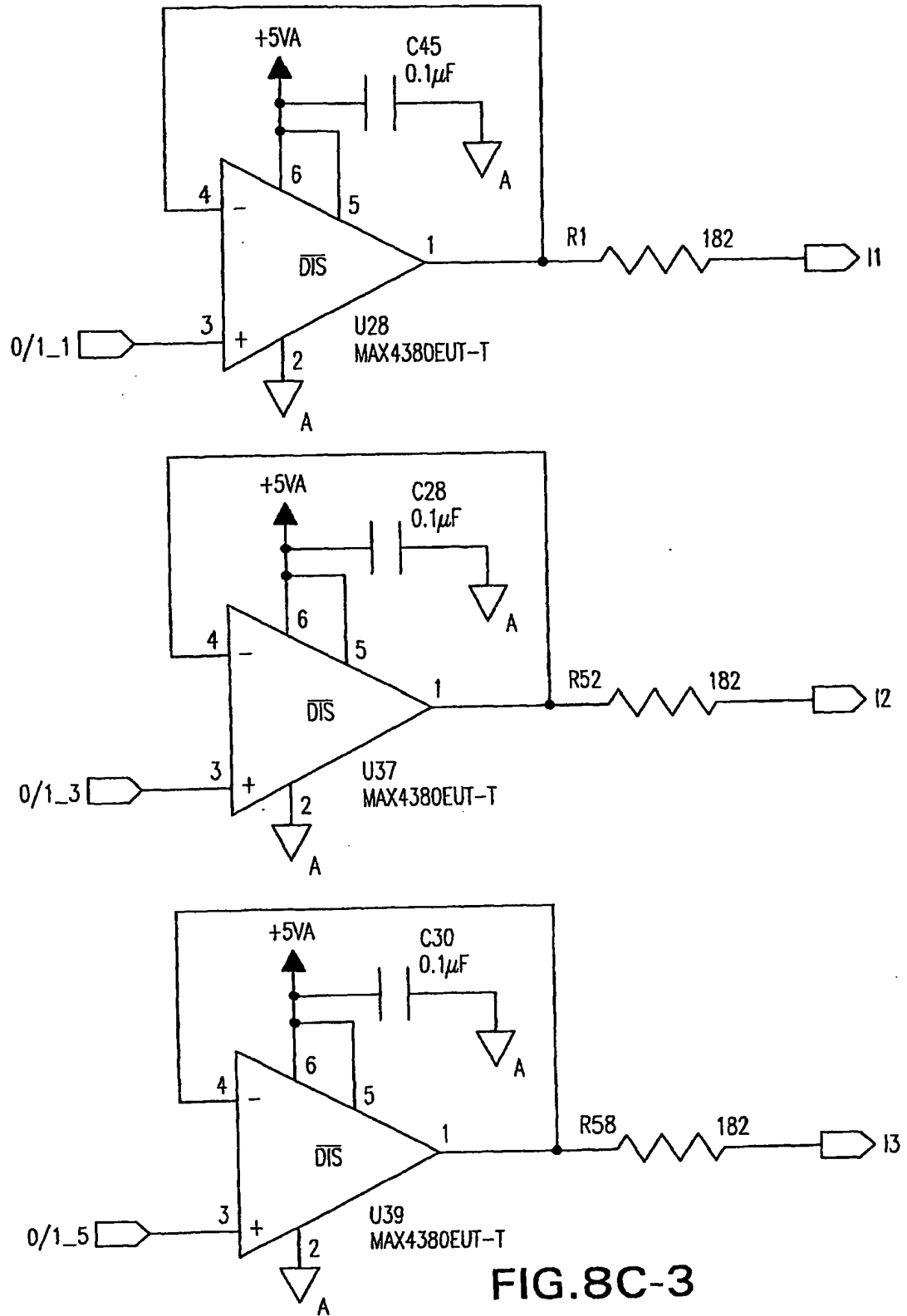


FIG.8C-3

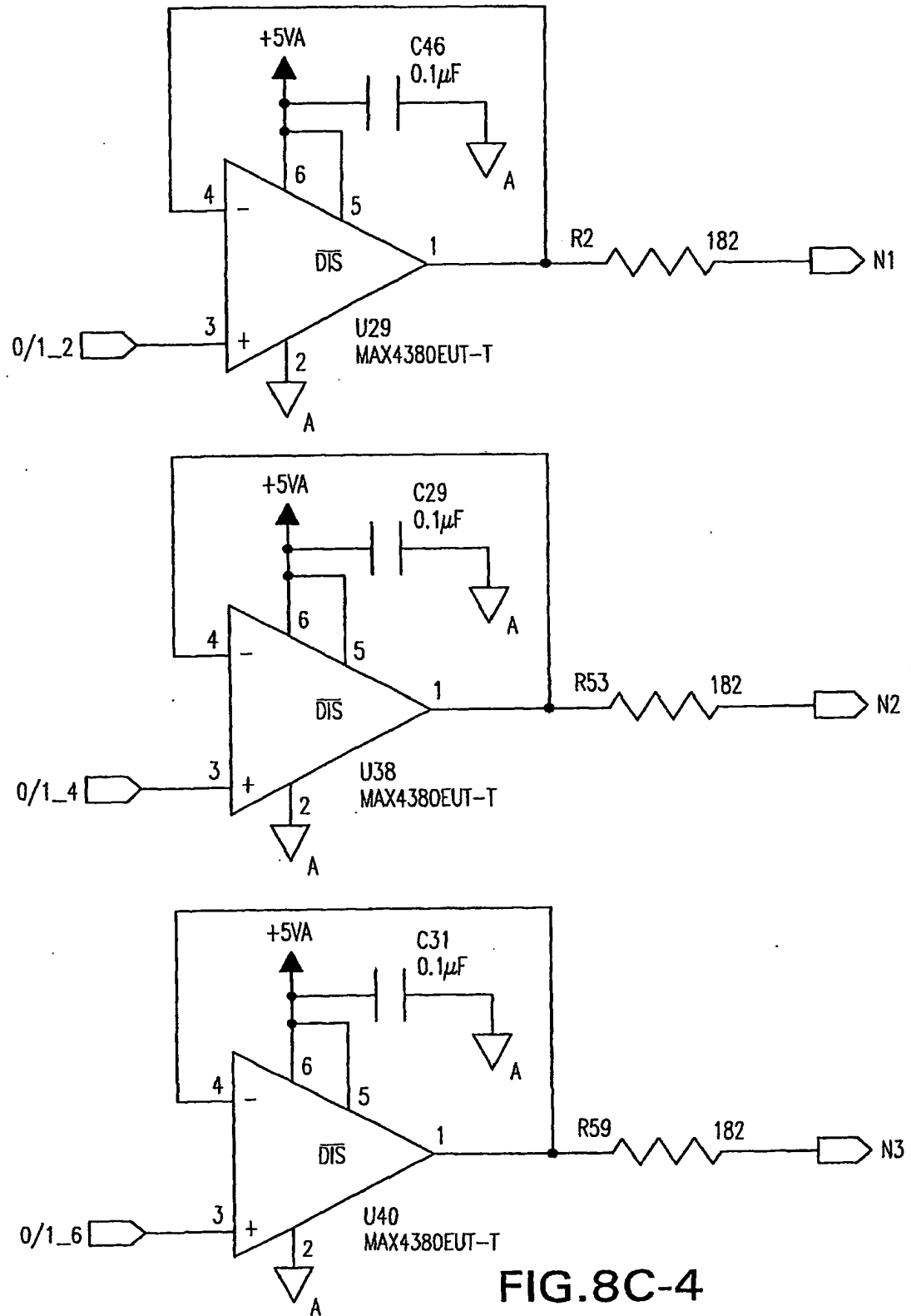


FIG.8C-4

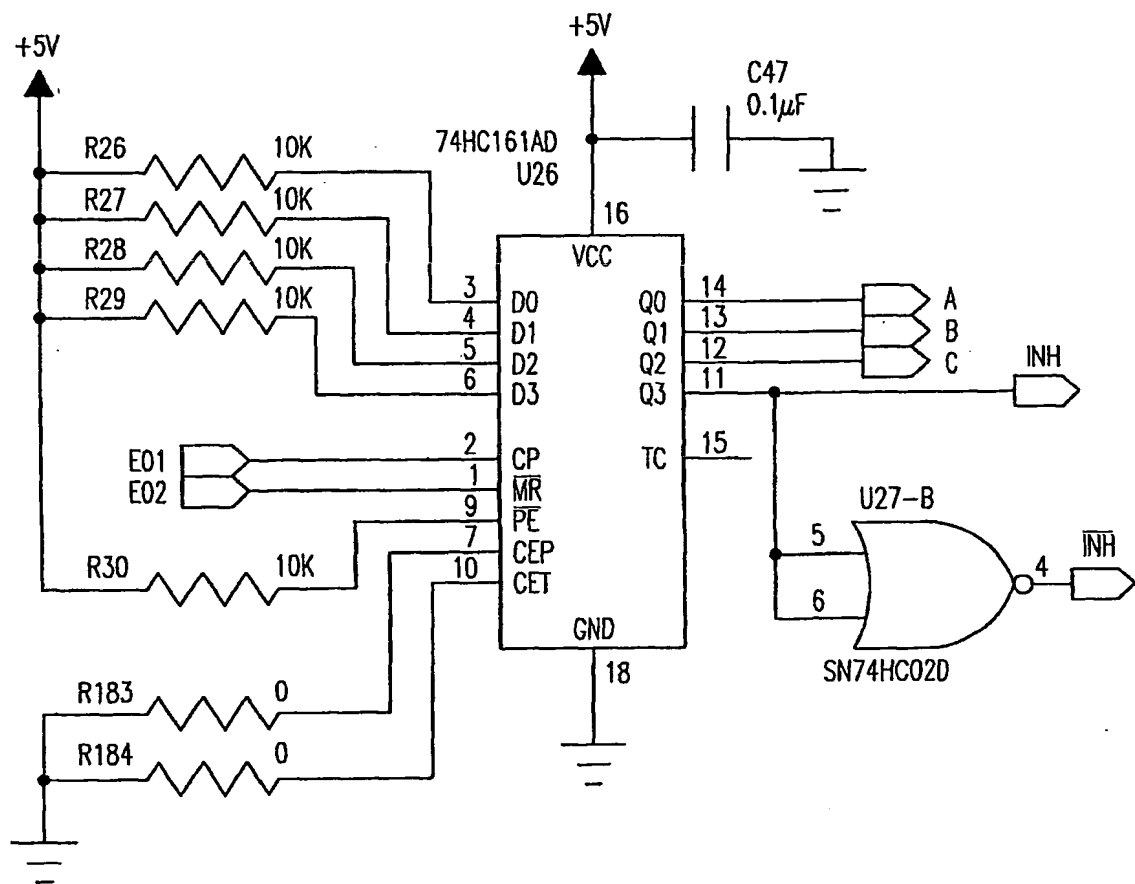


FIG.8C-5

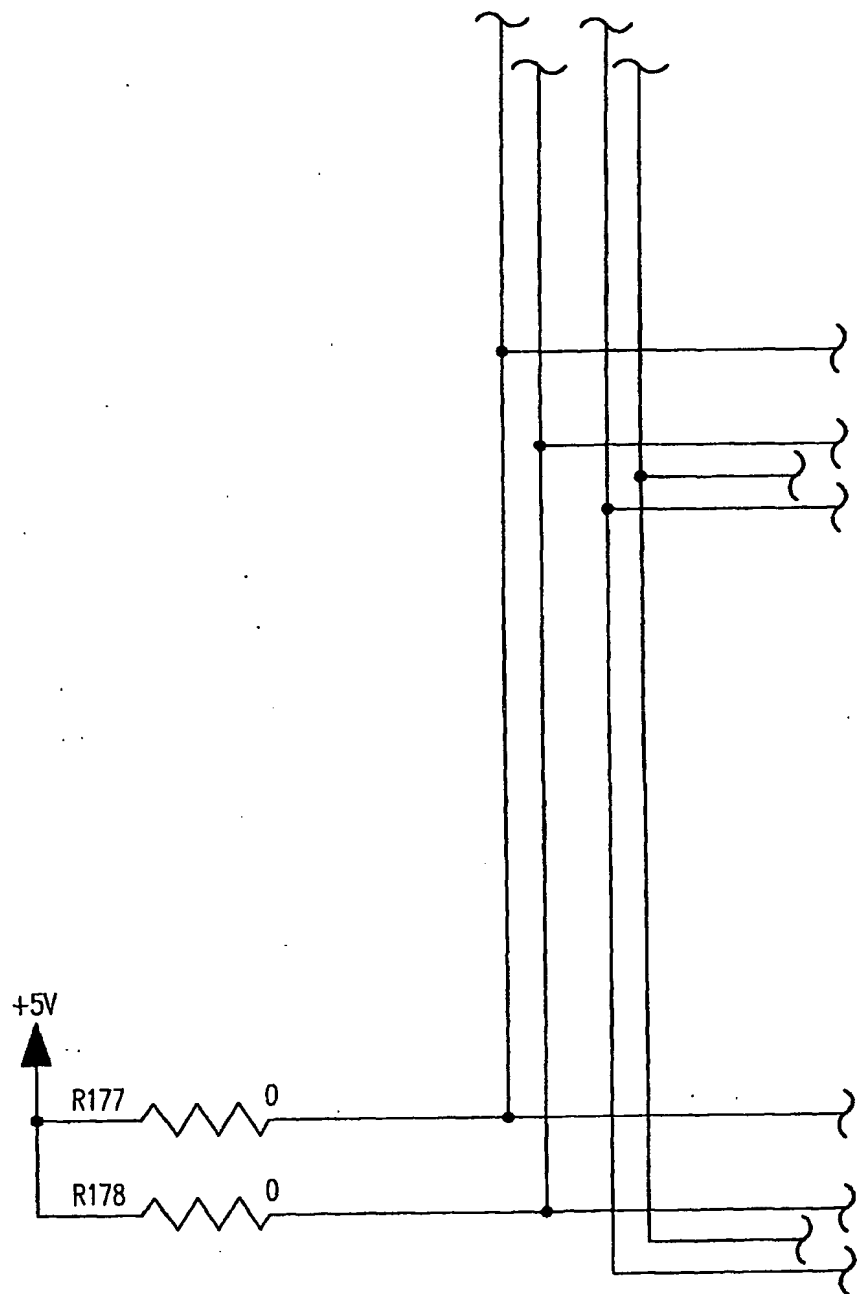


FIG.8C-6

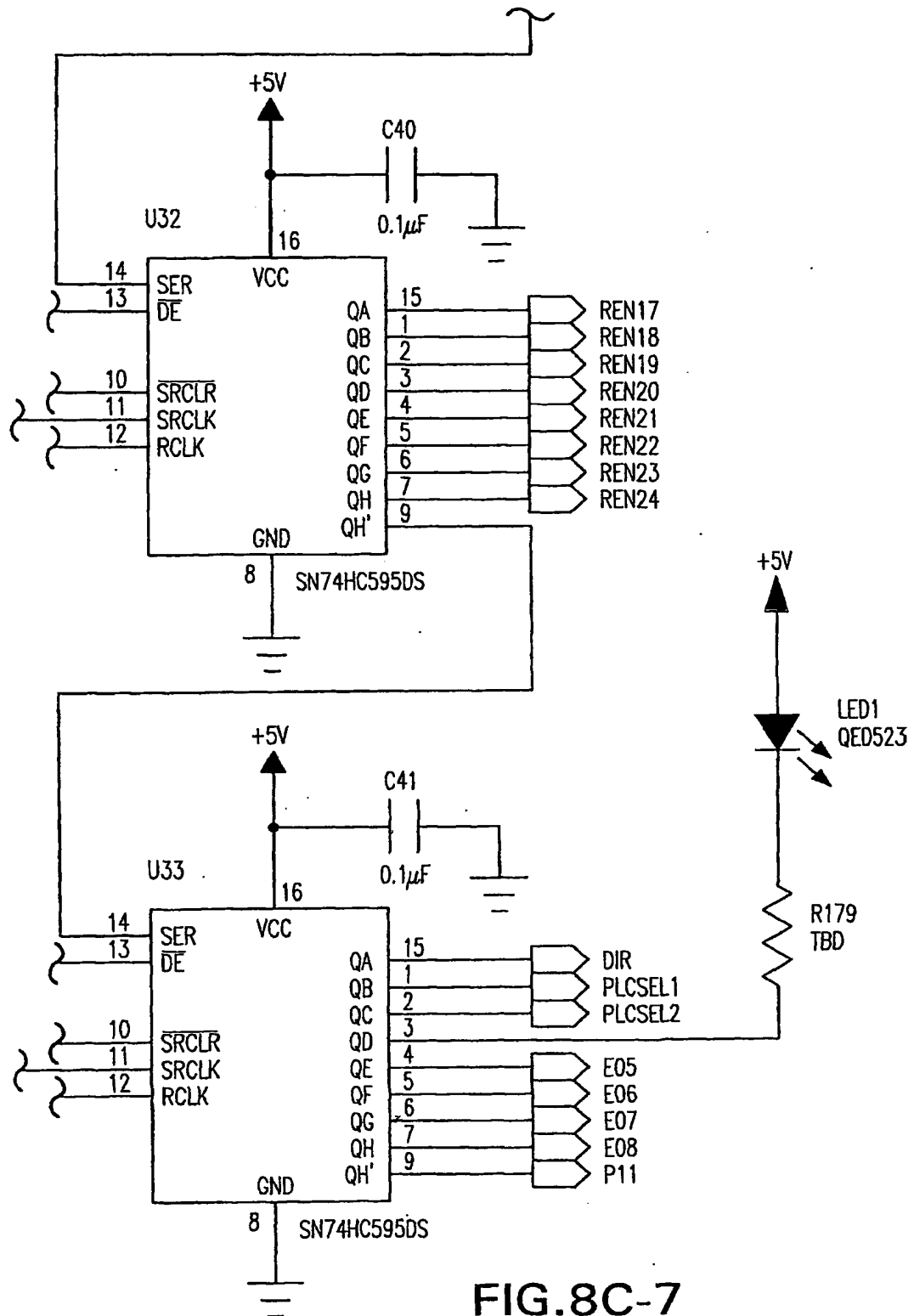


FIG.8C-7

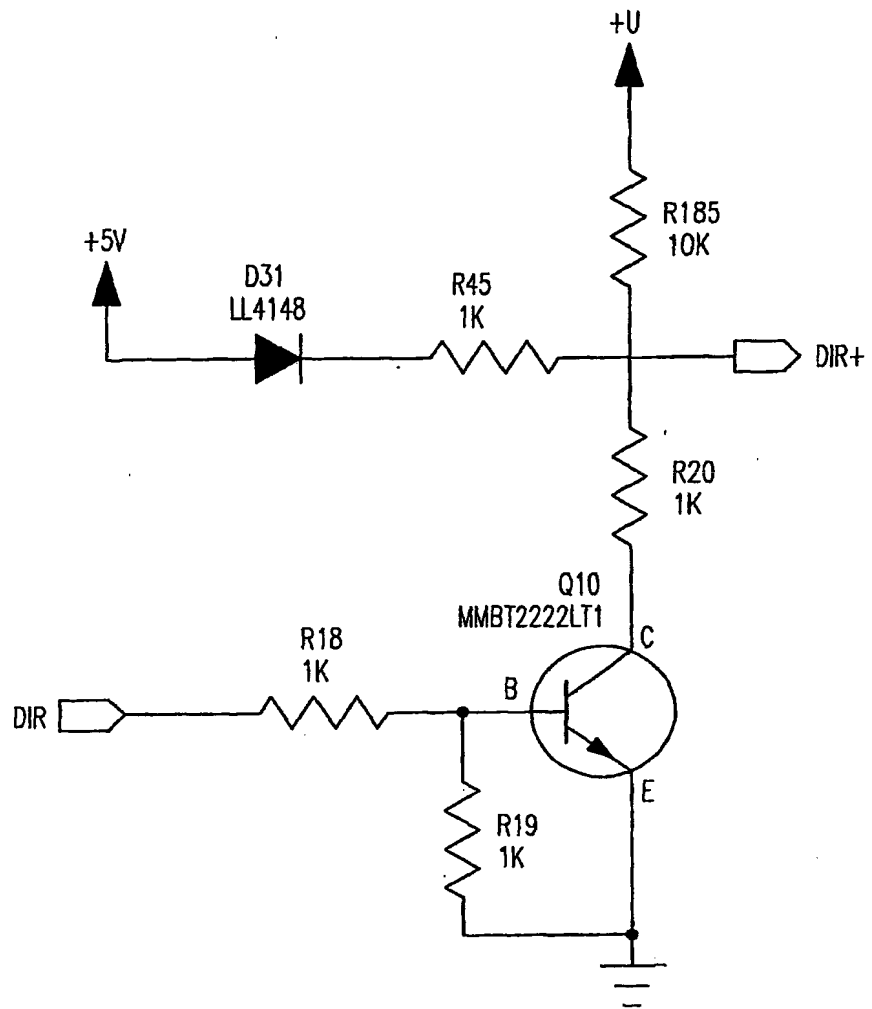


FIG.8C-8

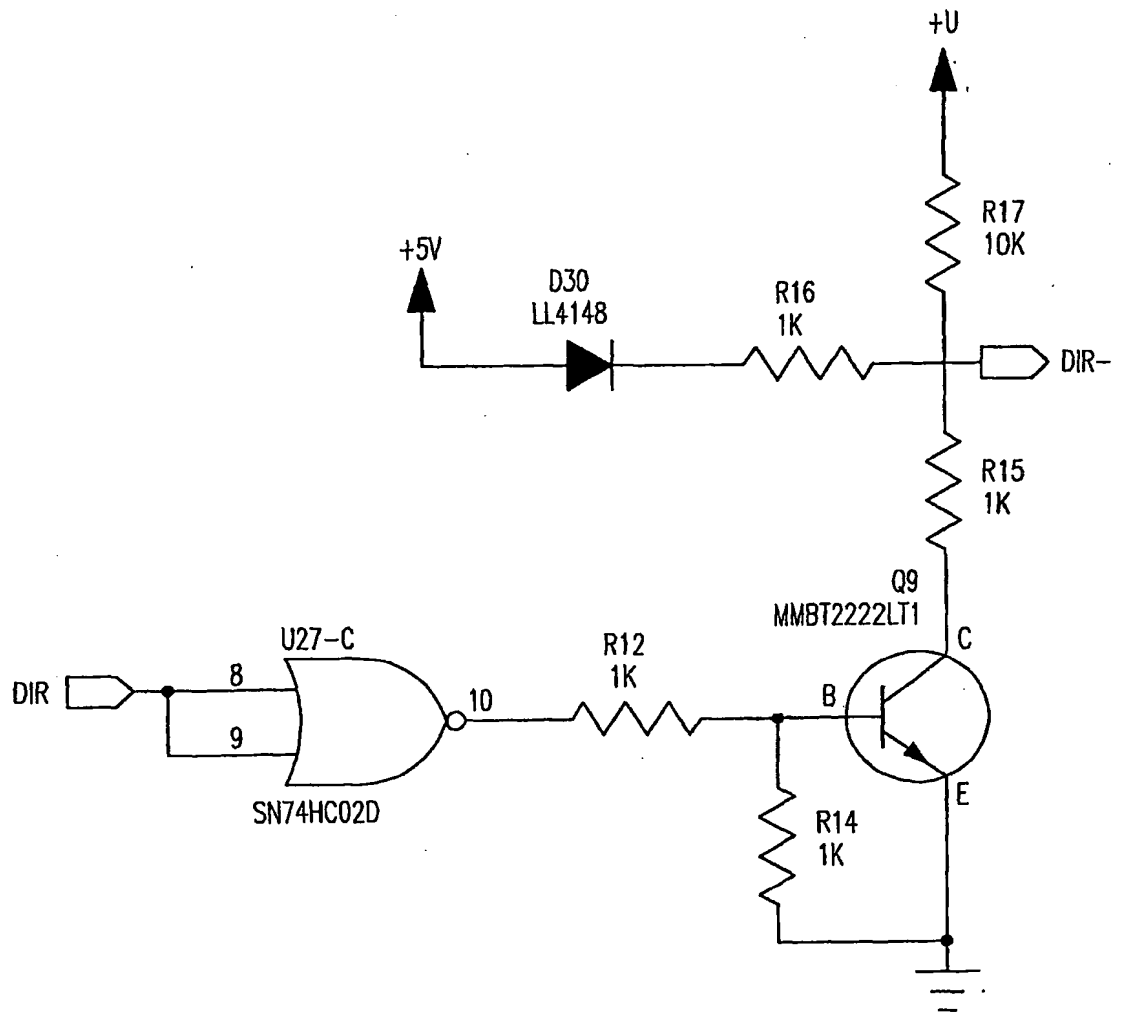


FIG. 8C-9

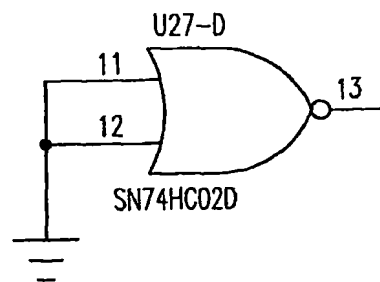
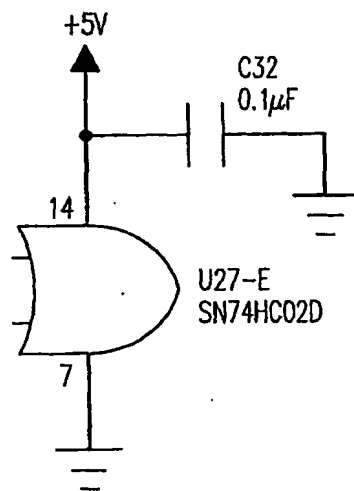


FIG.8C-10

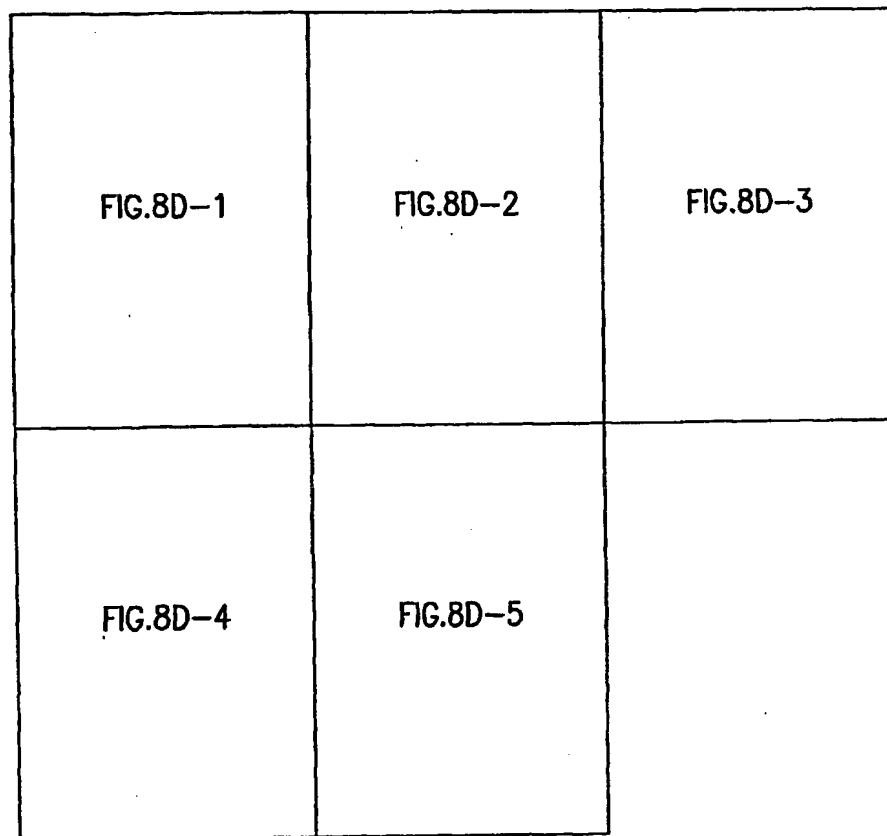
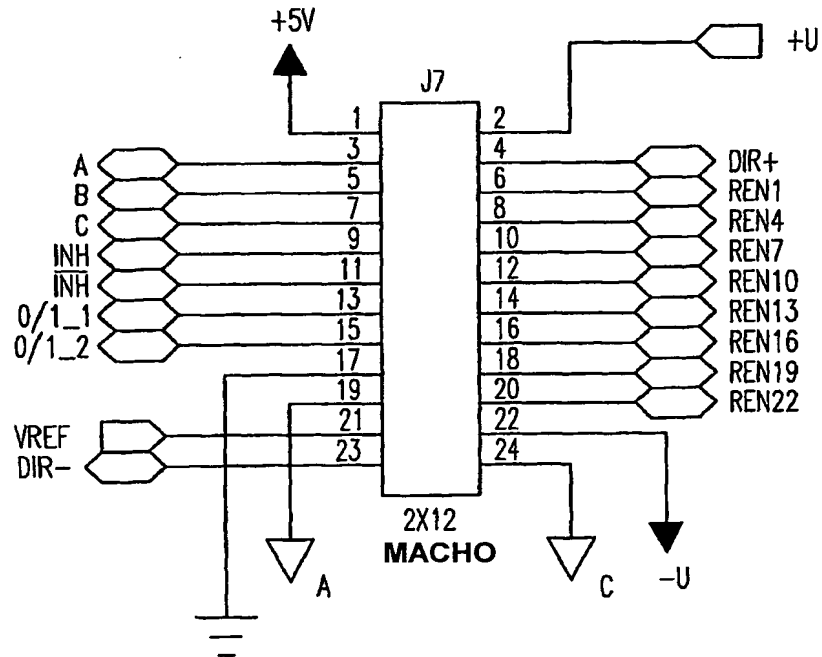


FIG.8D

FORA PARA PLACA RJ FASE A



FORA PARA PLACA RJ FASE B

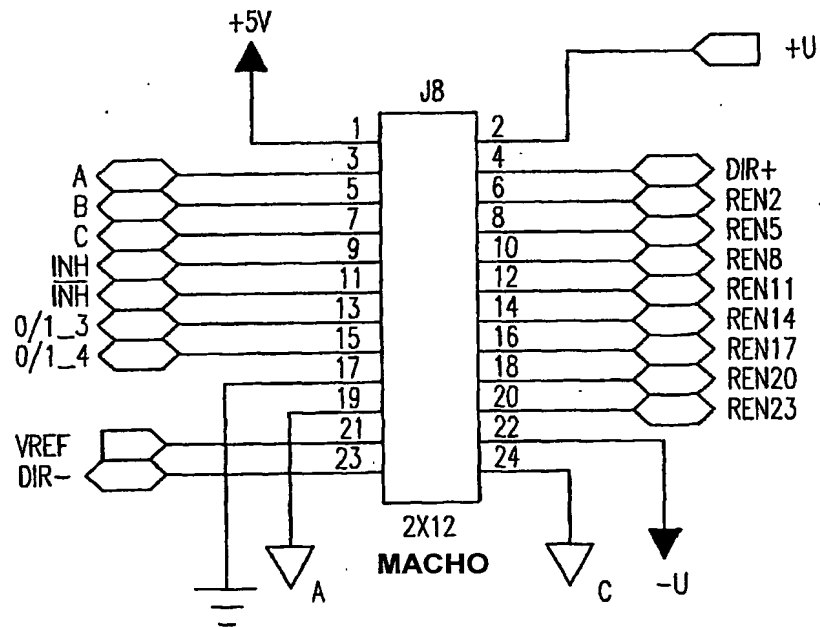
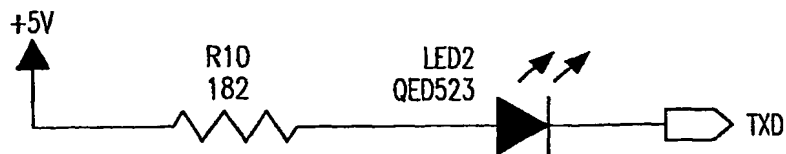


FIG.8D-1



DE PLACA DE MOSTRADOR

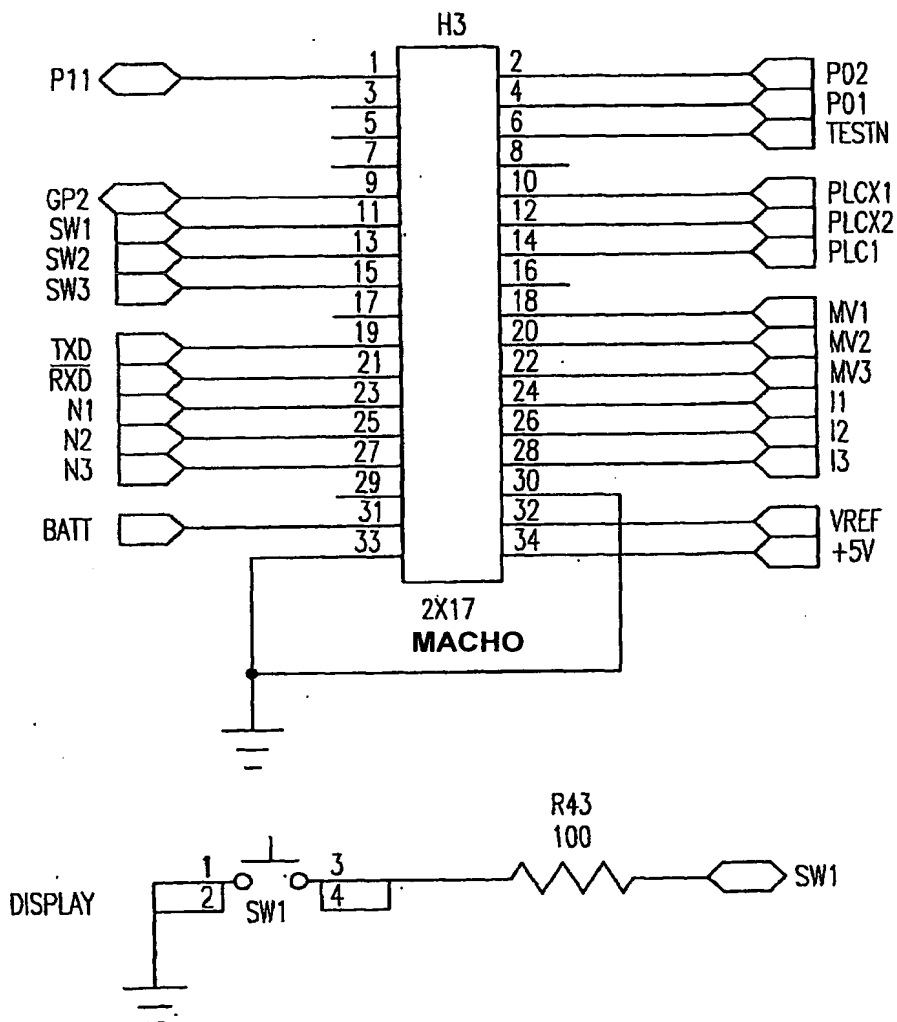


FIG.8D-2

PARA PLACA DE OPÇÃO RJ11

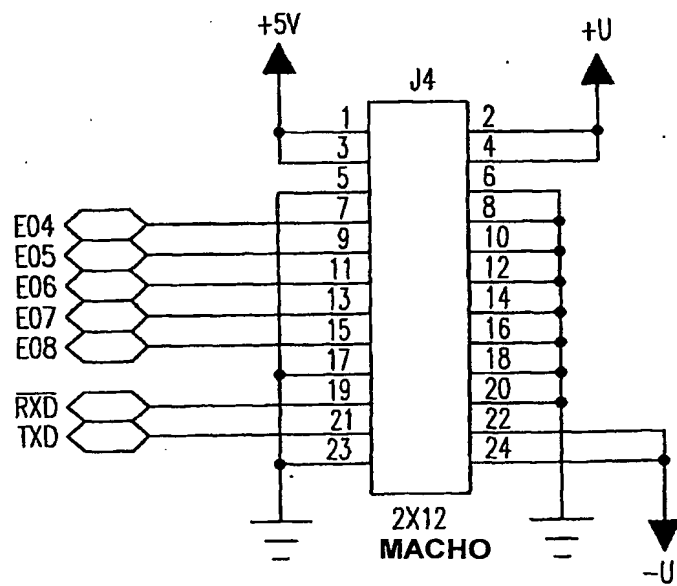


FIG.8D-3

FORA PARA PLACA RJ FASE C

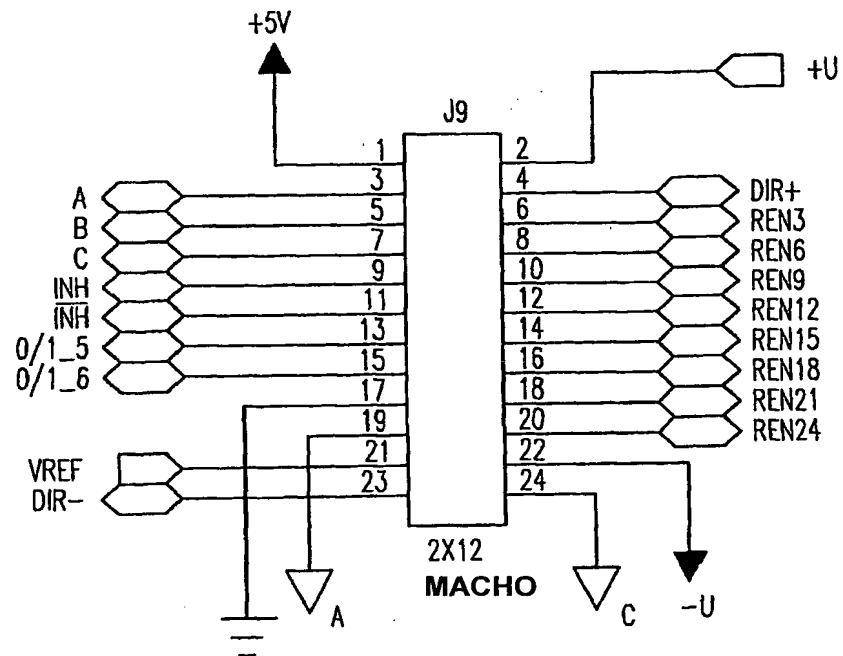


FIG.8D-4

DE PLACA DE MOSTRADOR

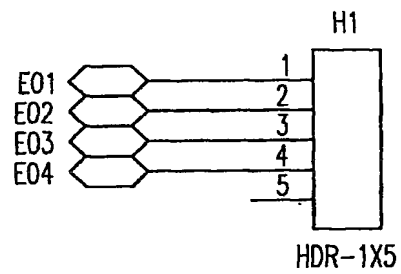


FIG.8D-5

FIG.9A-1	FIG.9A-2	FIG.9A-3
FIG.9A-4	FIG.9A-5	FIG.9A-6
FIG.9A-7	FIG.9A-8	FIG.9A-9

FIG.9A

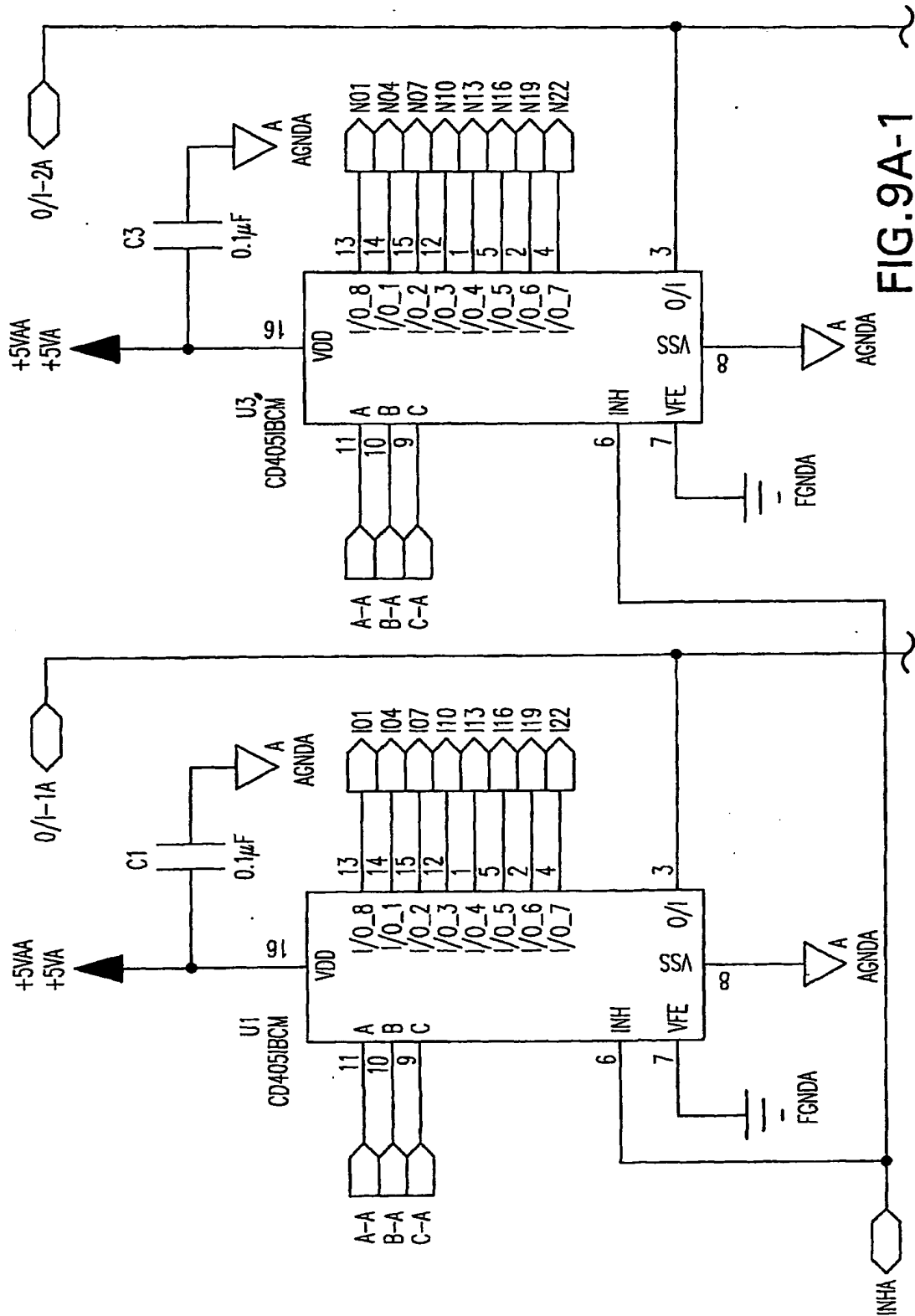


FIG.9A-1

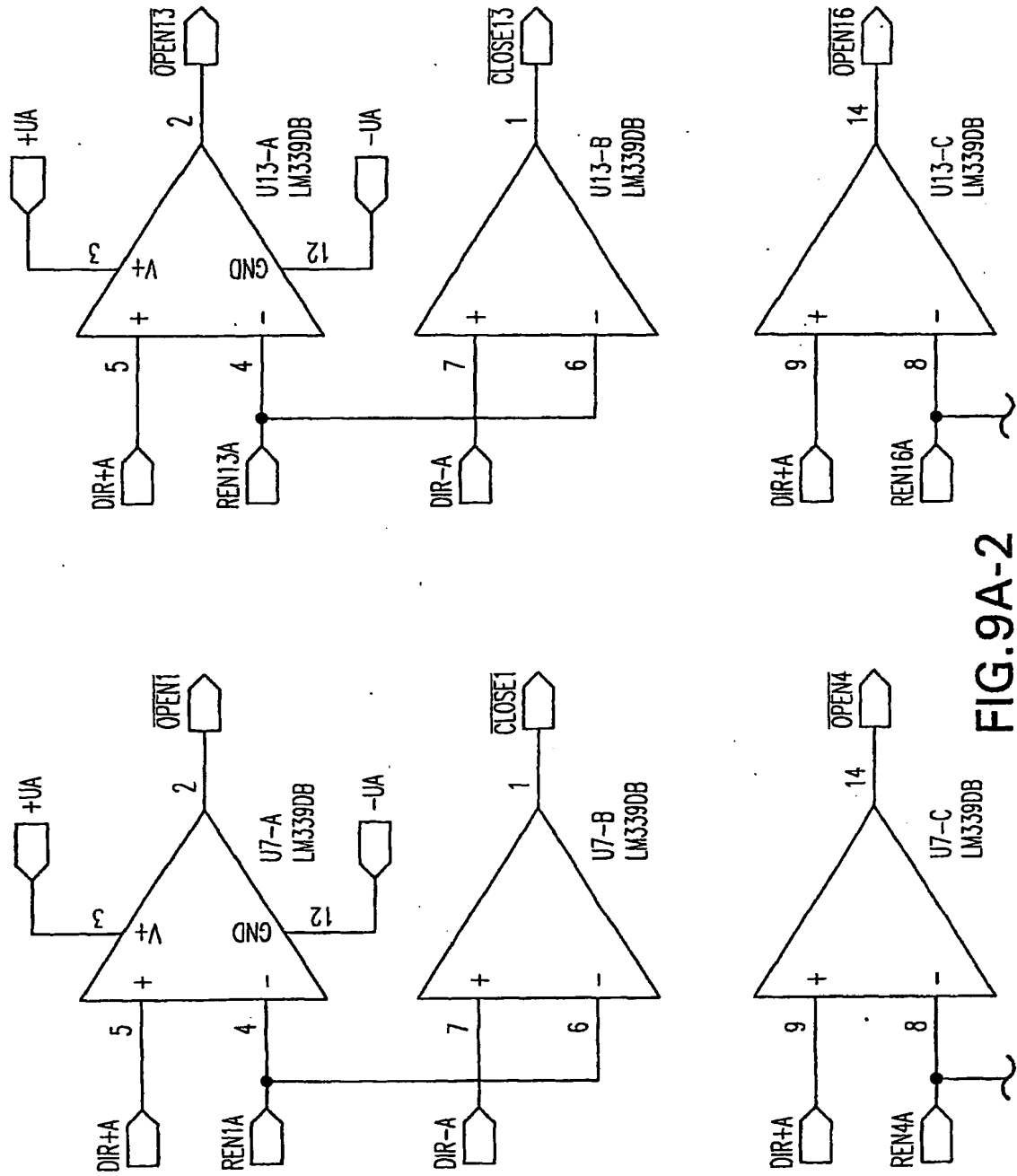


FIG. 9A-2

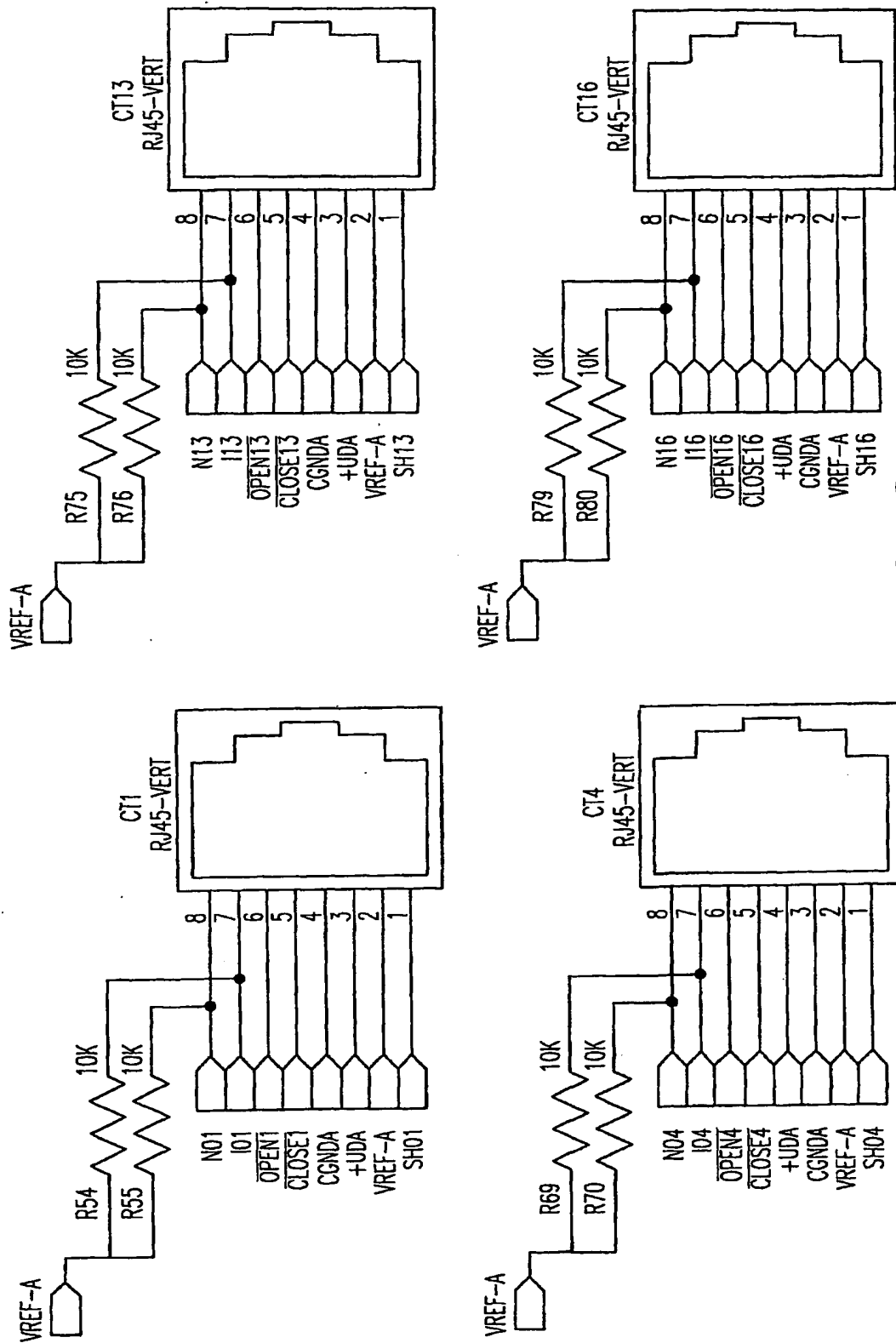


FIG.9A-3

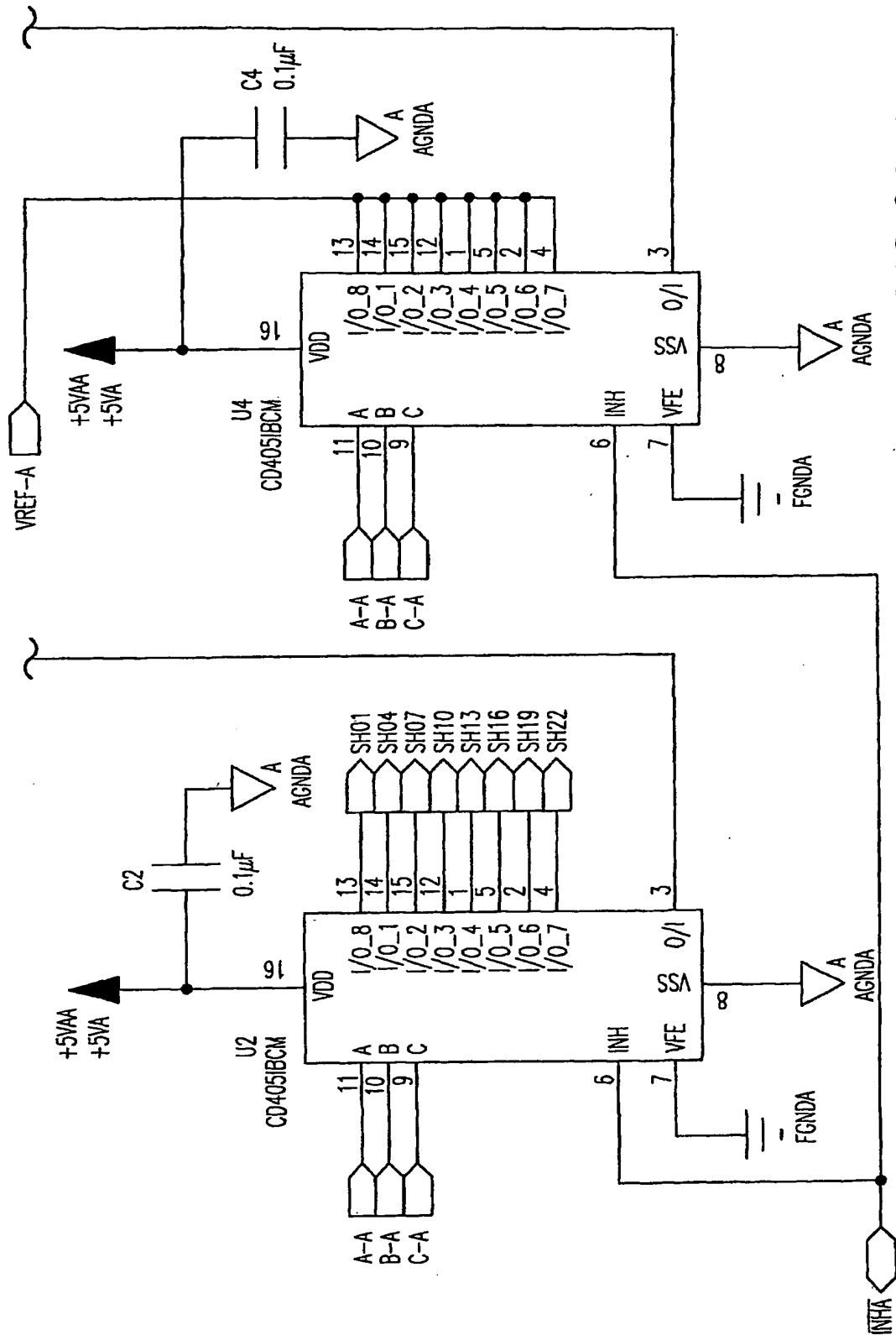


FIG.9A-4

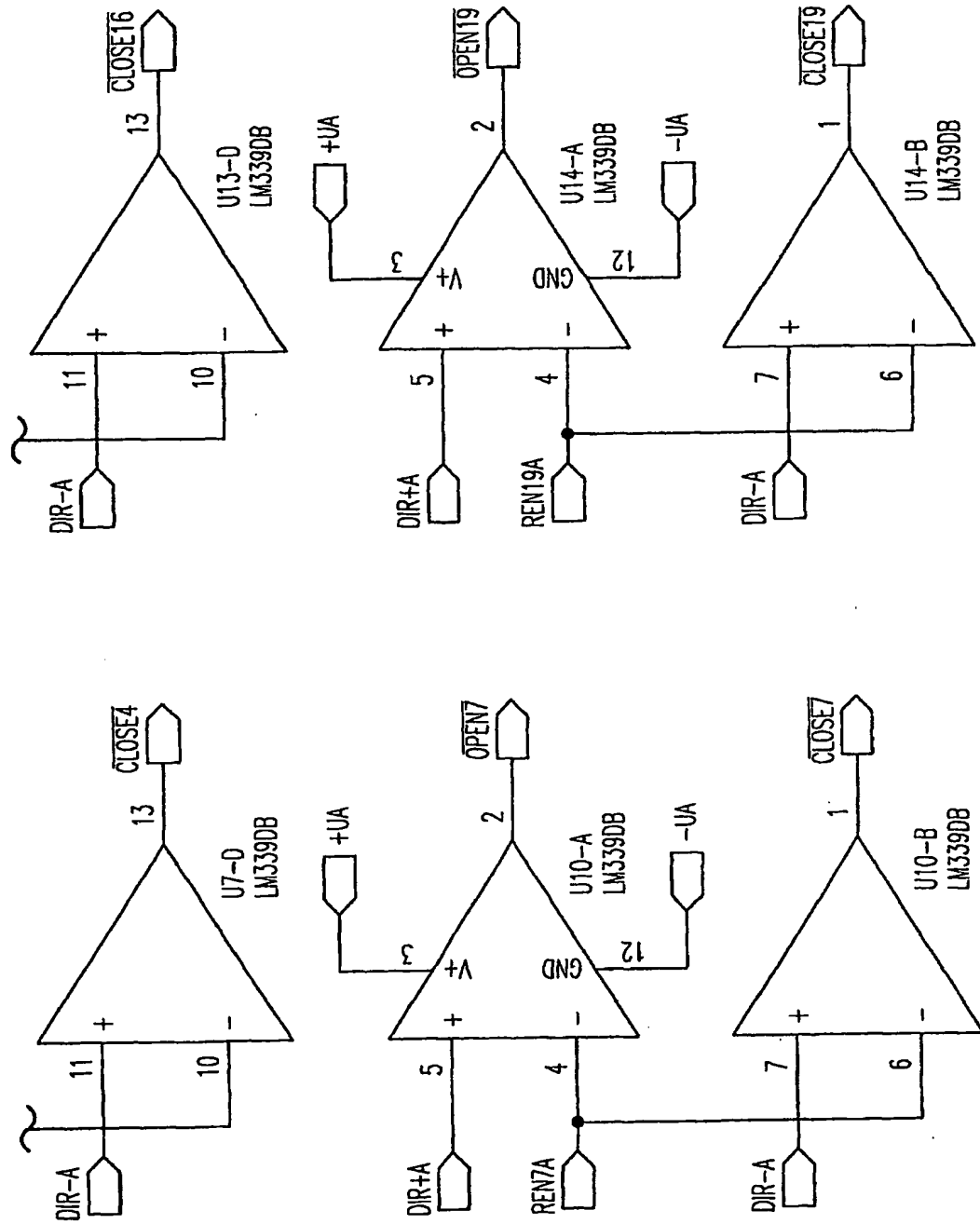


FIG.9A-5

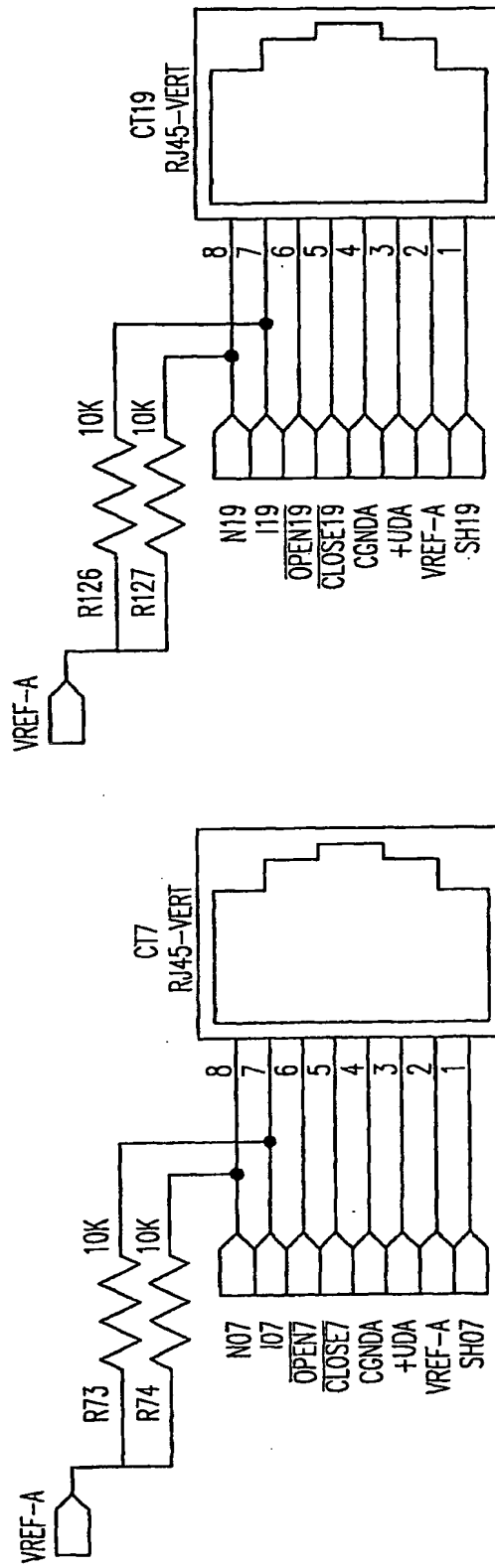


FIG. 9A-6

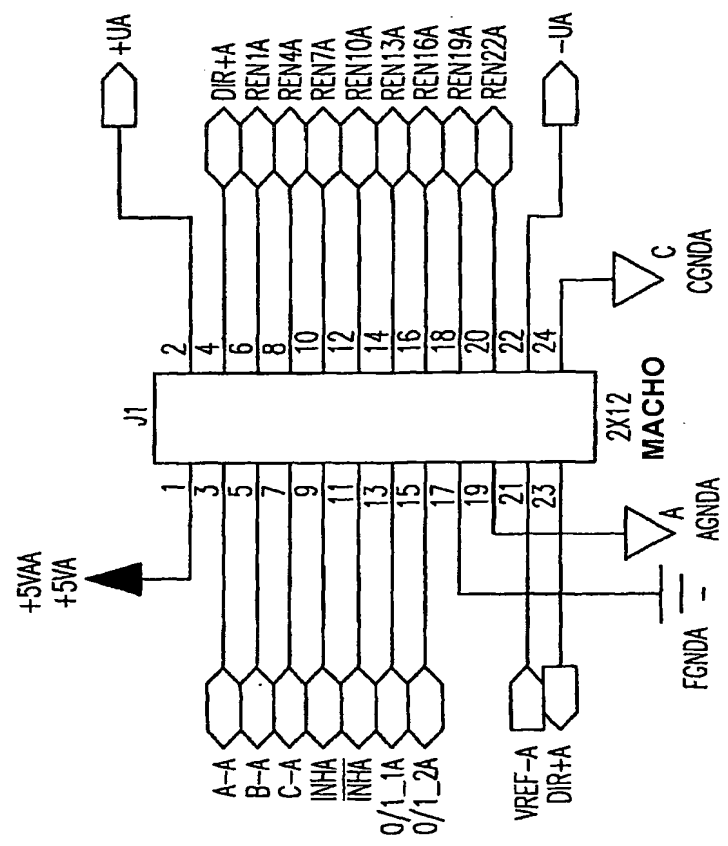
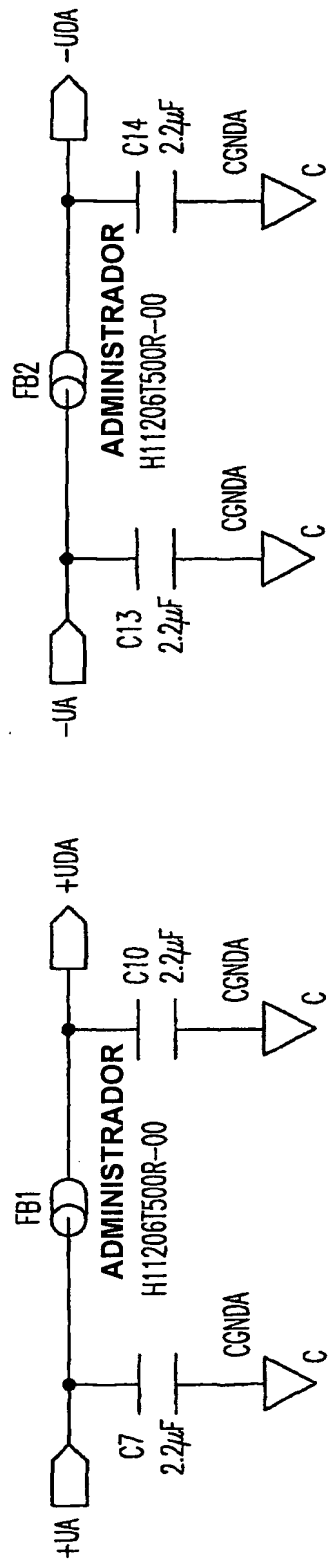


FIG.9A-7

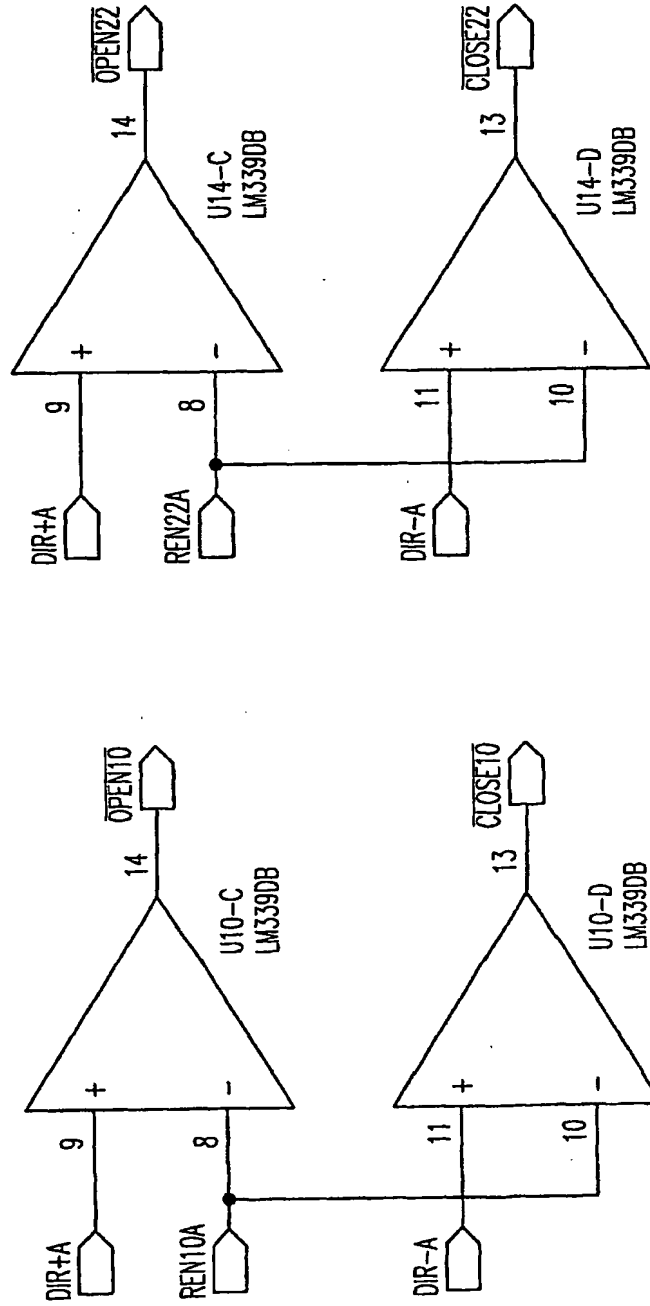


FIG.9A-8

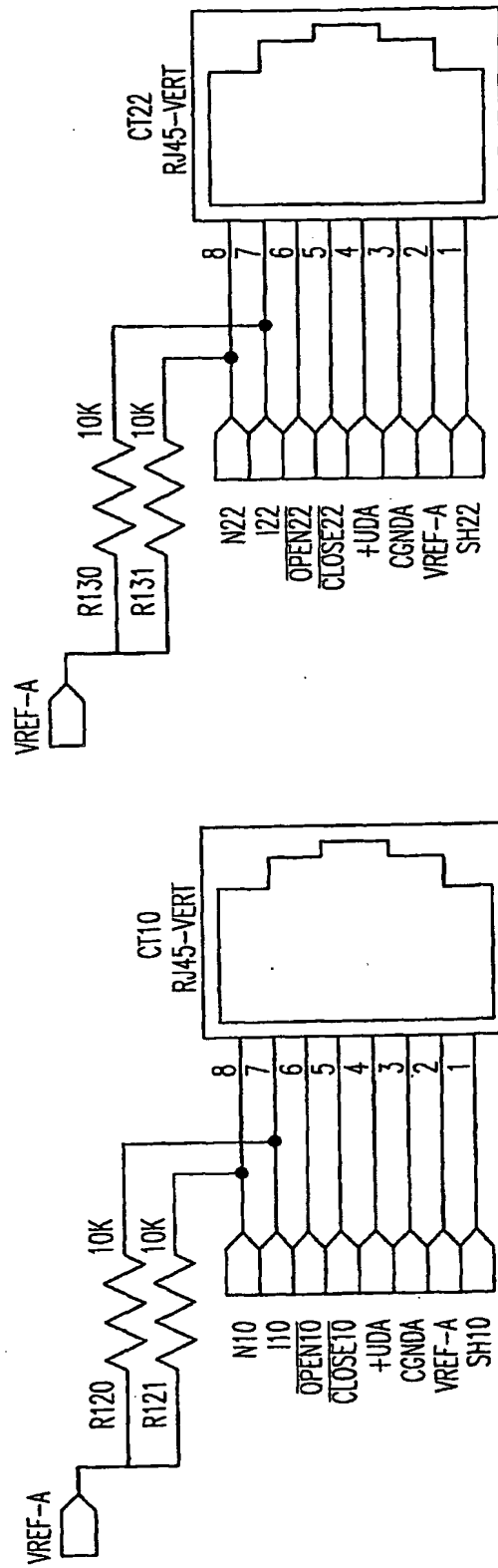


FIG. 9A-9

FIG.9B-1	FIG.9B-2	FIG.9B-3
FIG.9B-4	FIG.9B-5	FIG.9B-6
FIG.9B-7	FIG.9B-8	FIG.9B-9

FIG.9B

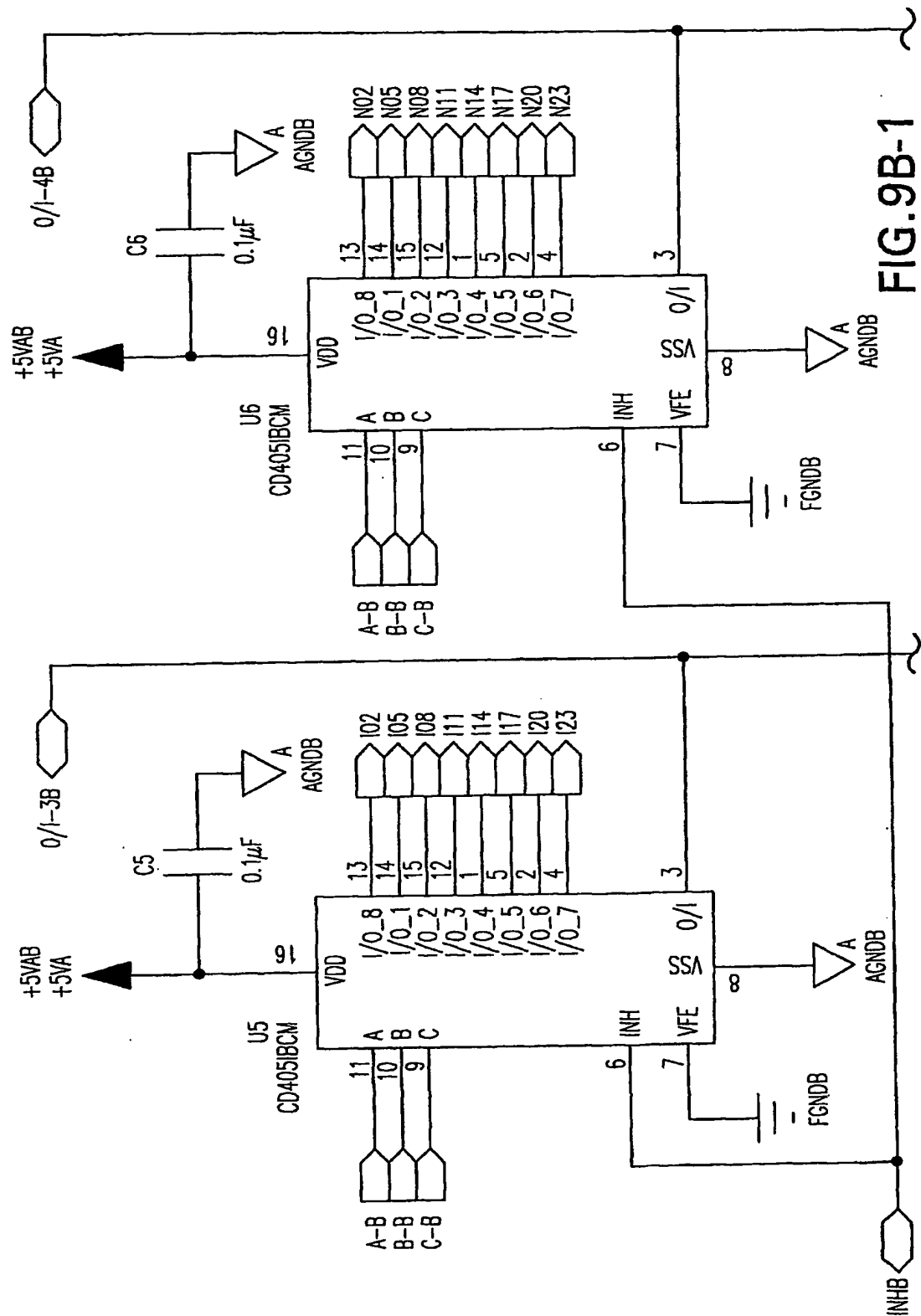


FIG. 9B-1

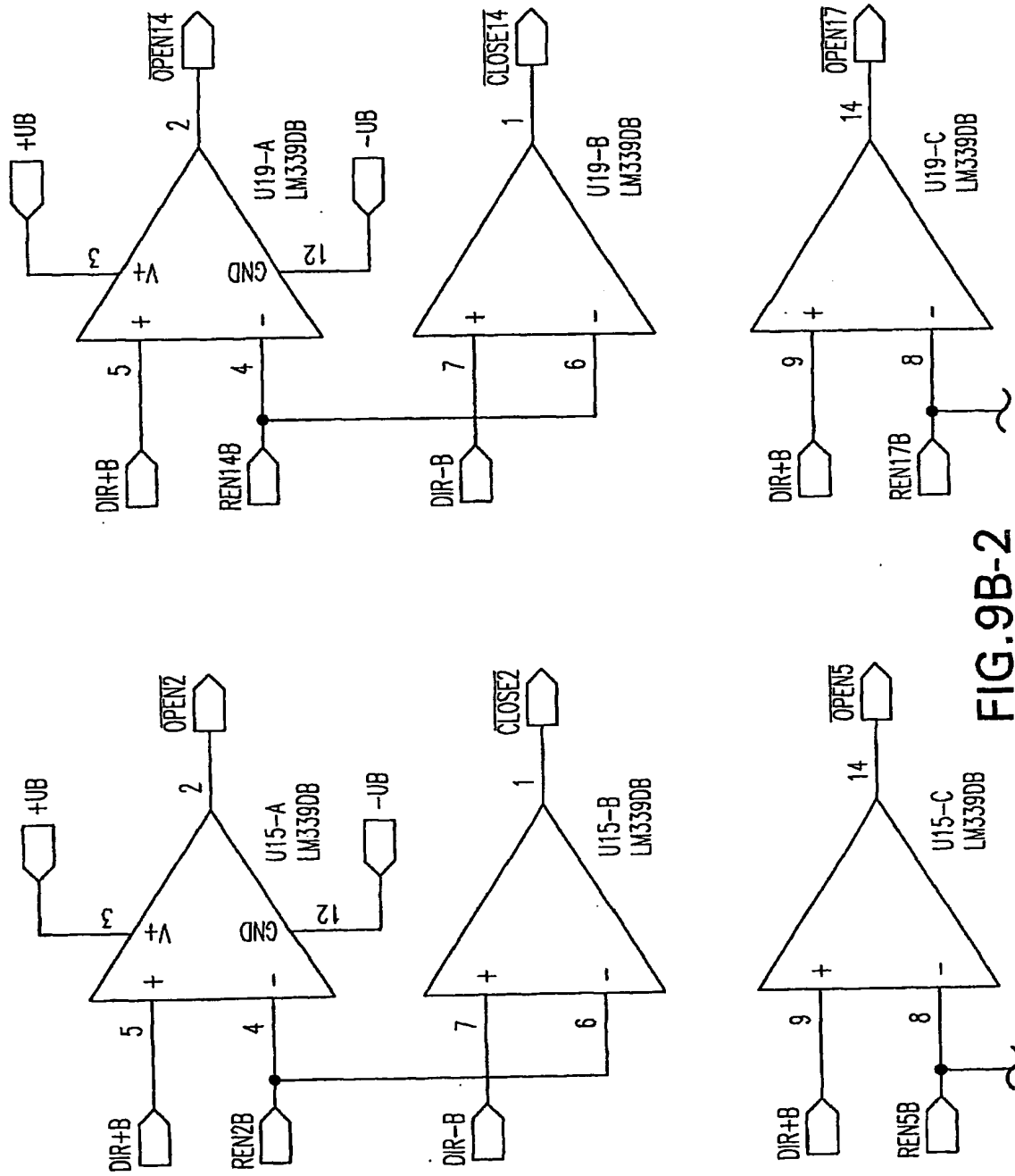


FIG. 9B-2

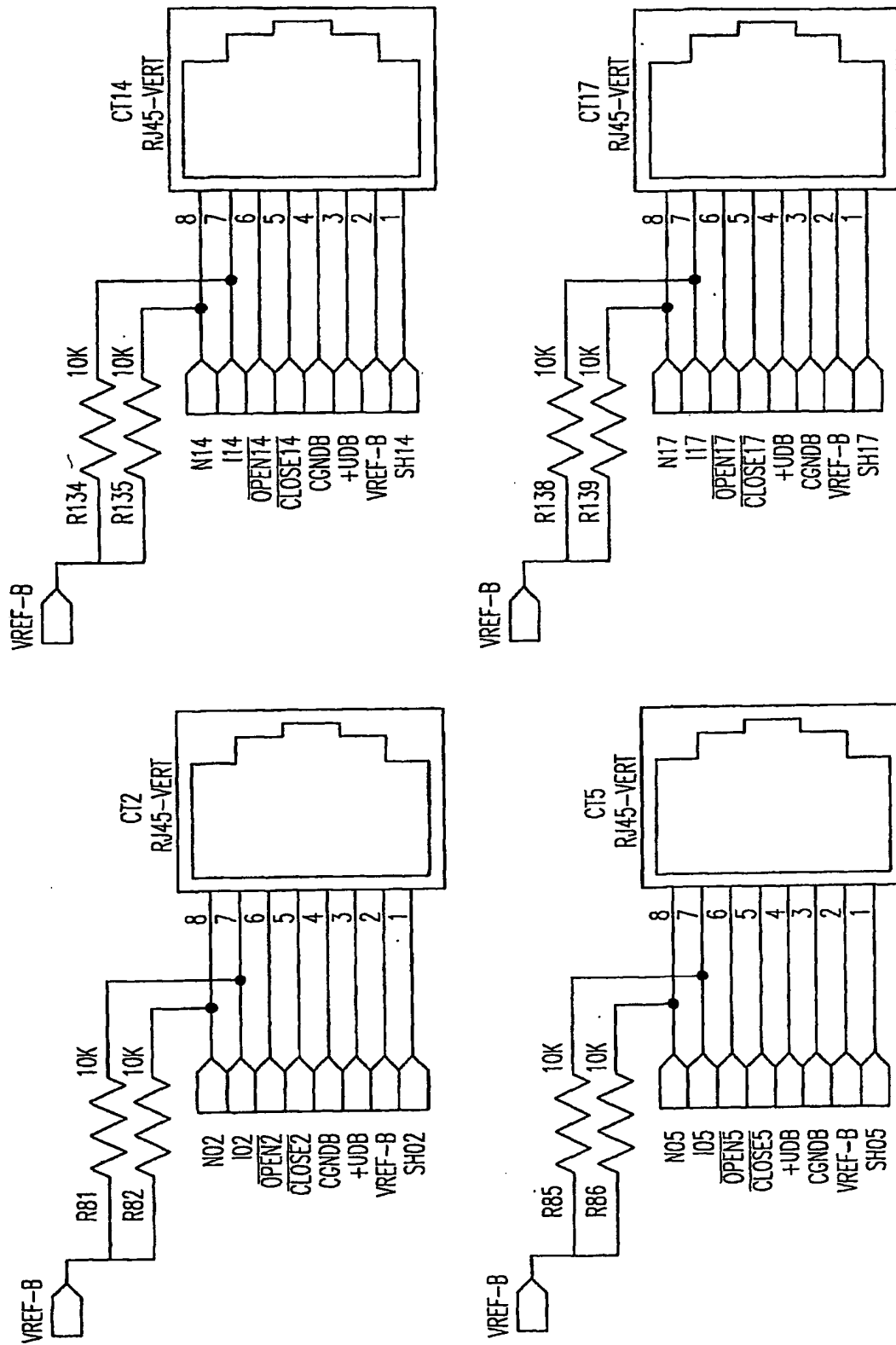


FIG.9B-3

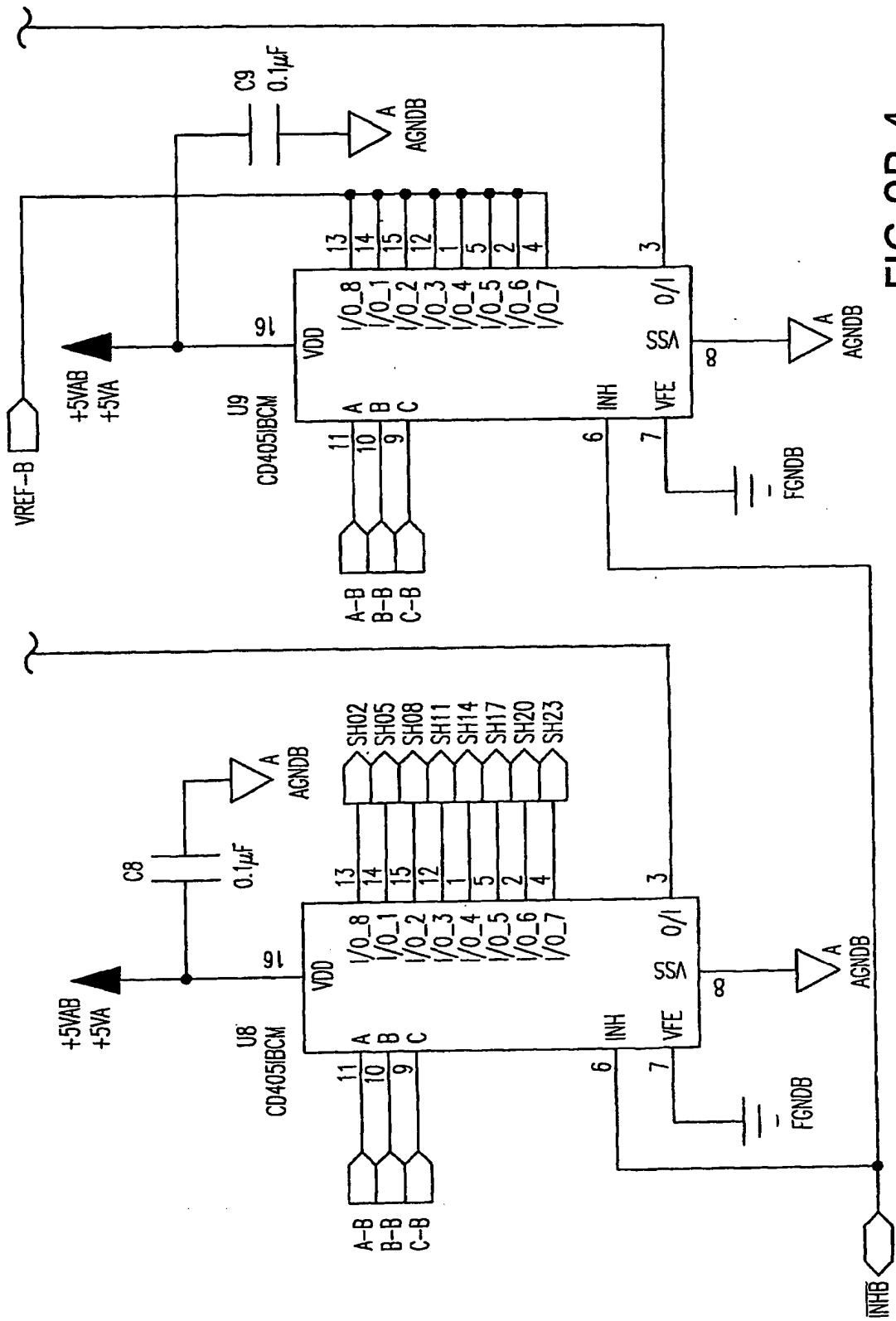


FIG.9B-4

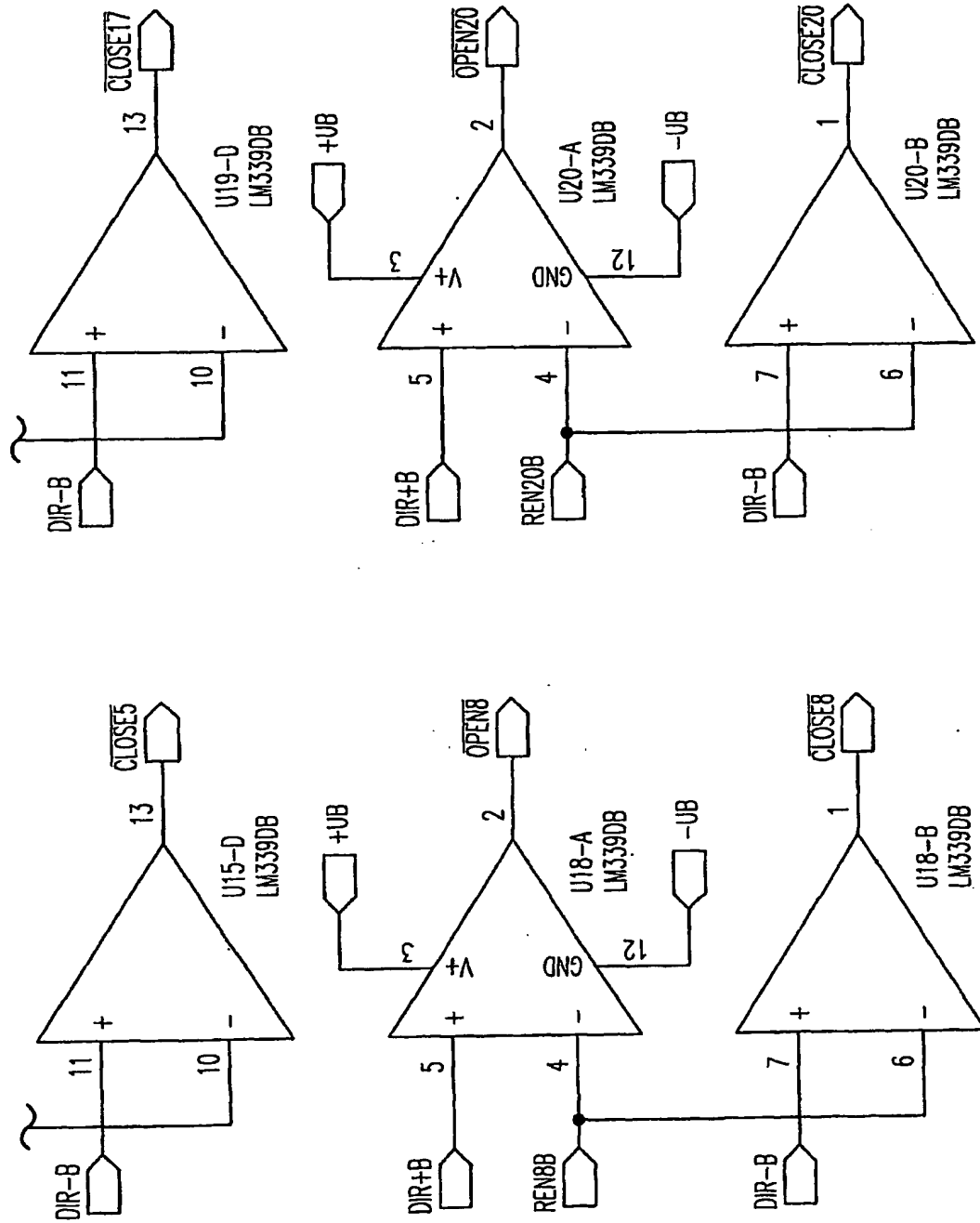


FIG. 9B-5

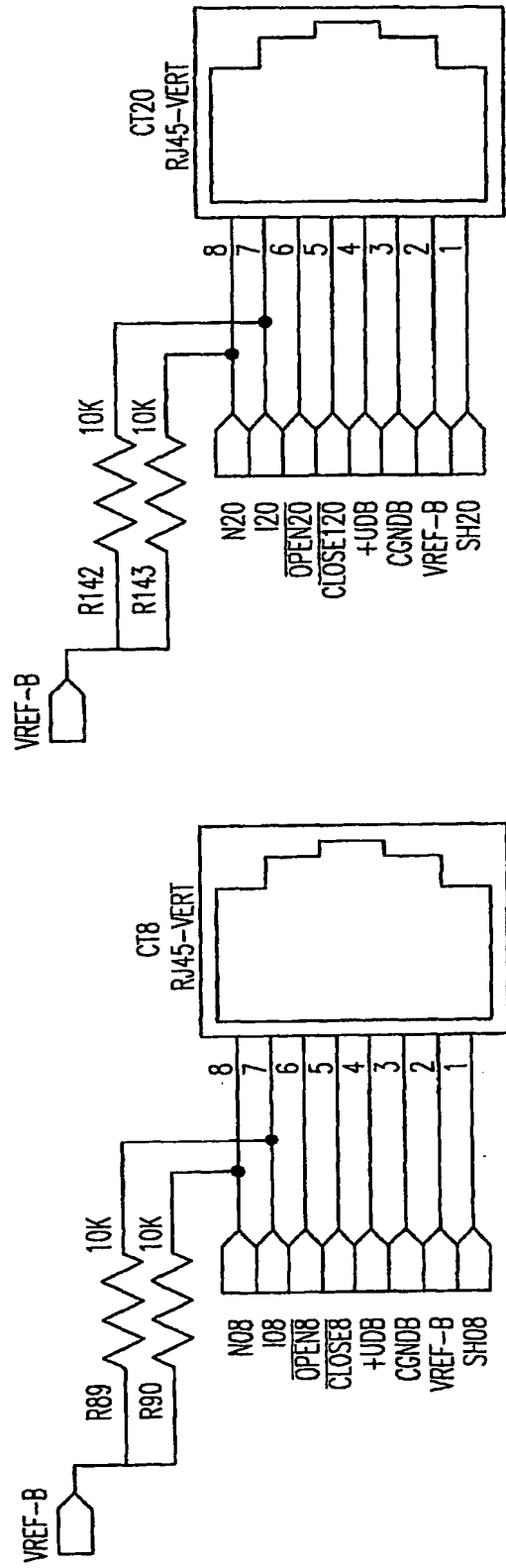


FIG.9B-6

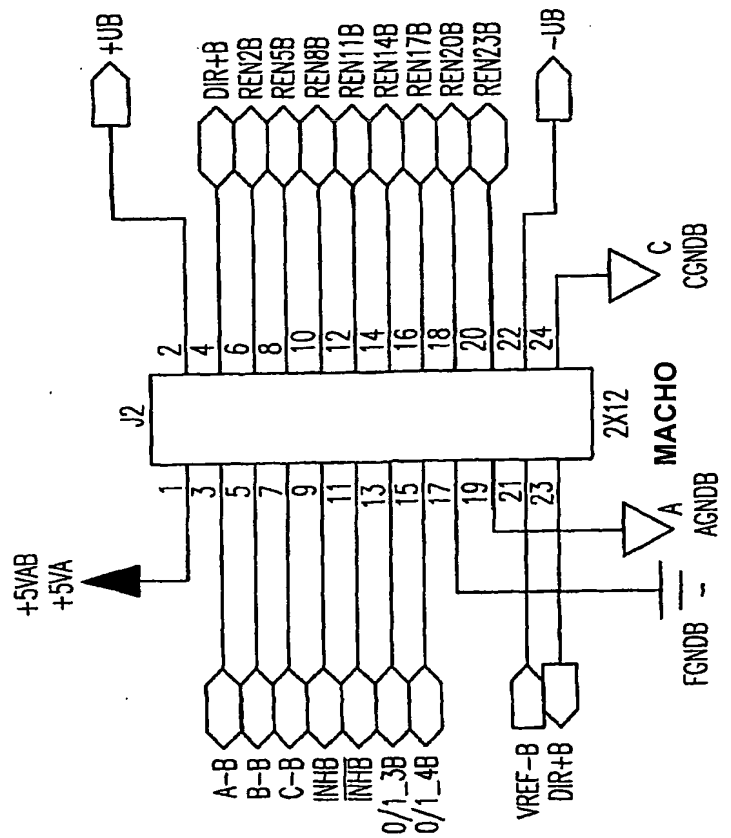
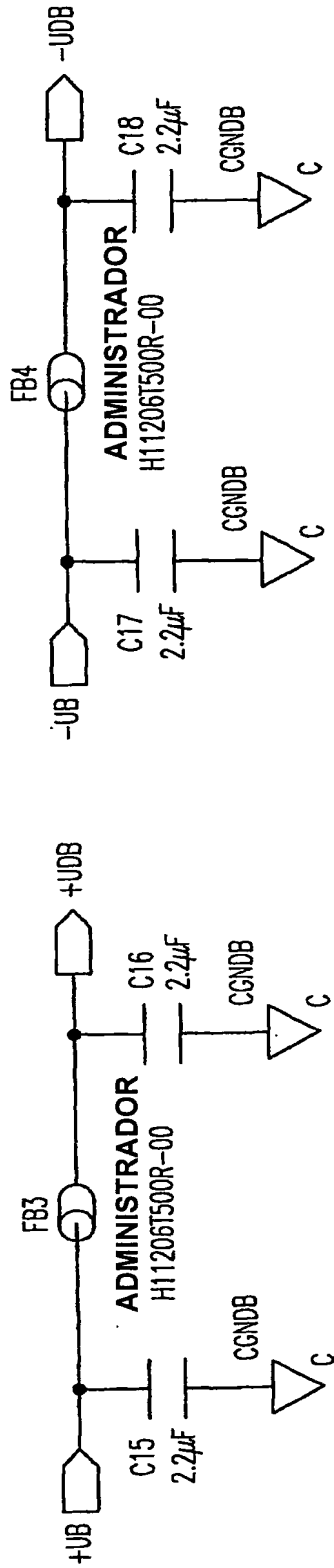


FIG.9B-7

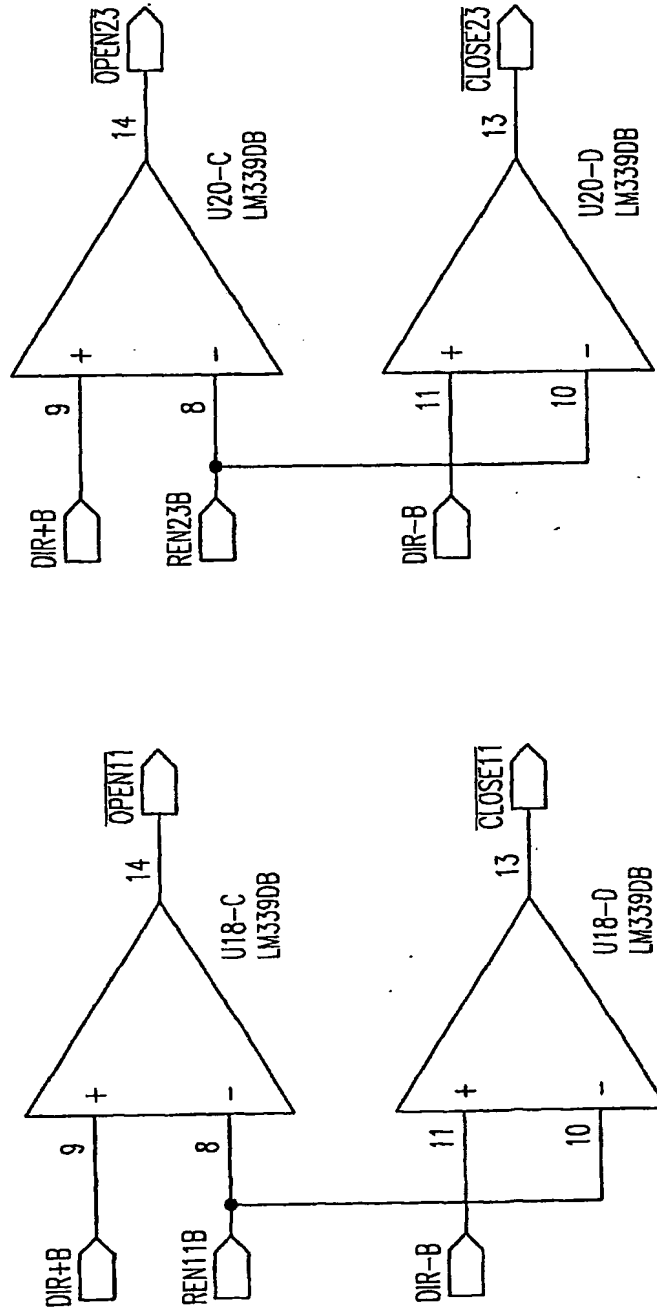


FIG. 9B-8

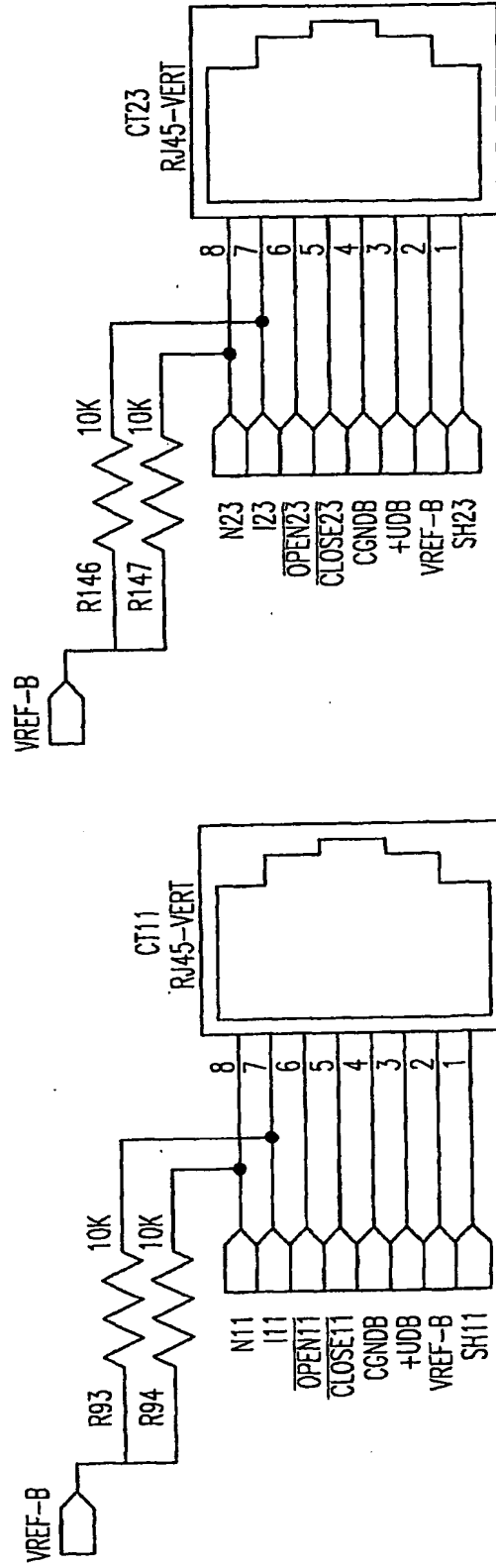


FIG.9B-9

FIG.9C-1	FIG.9C-2	FIG.9C-3
FIG.9C-4	FIG.9C-5	FIG.9C-6
FIG.9C-7	FIG.9C-8	FIG.9C-9

FIG.9C

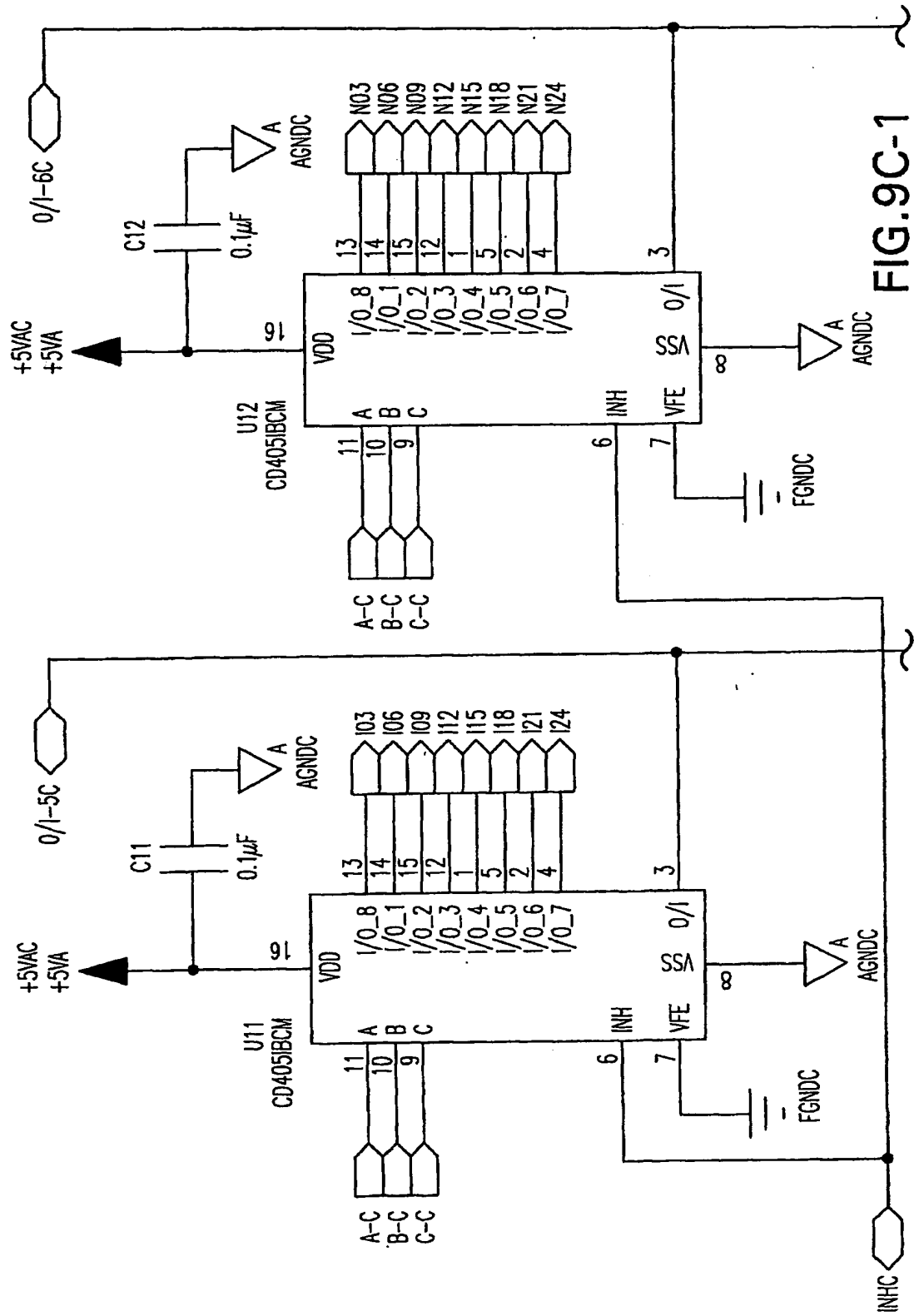


FIG.9C-1

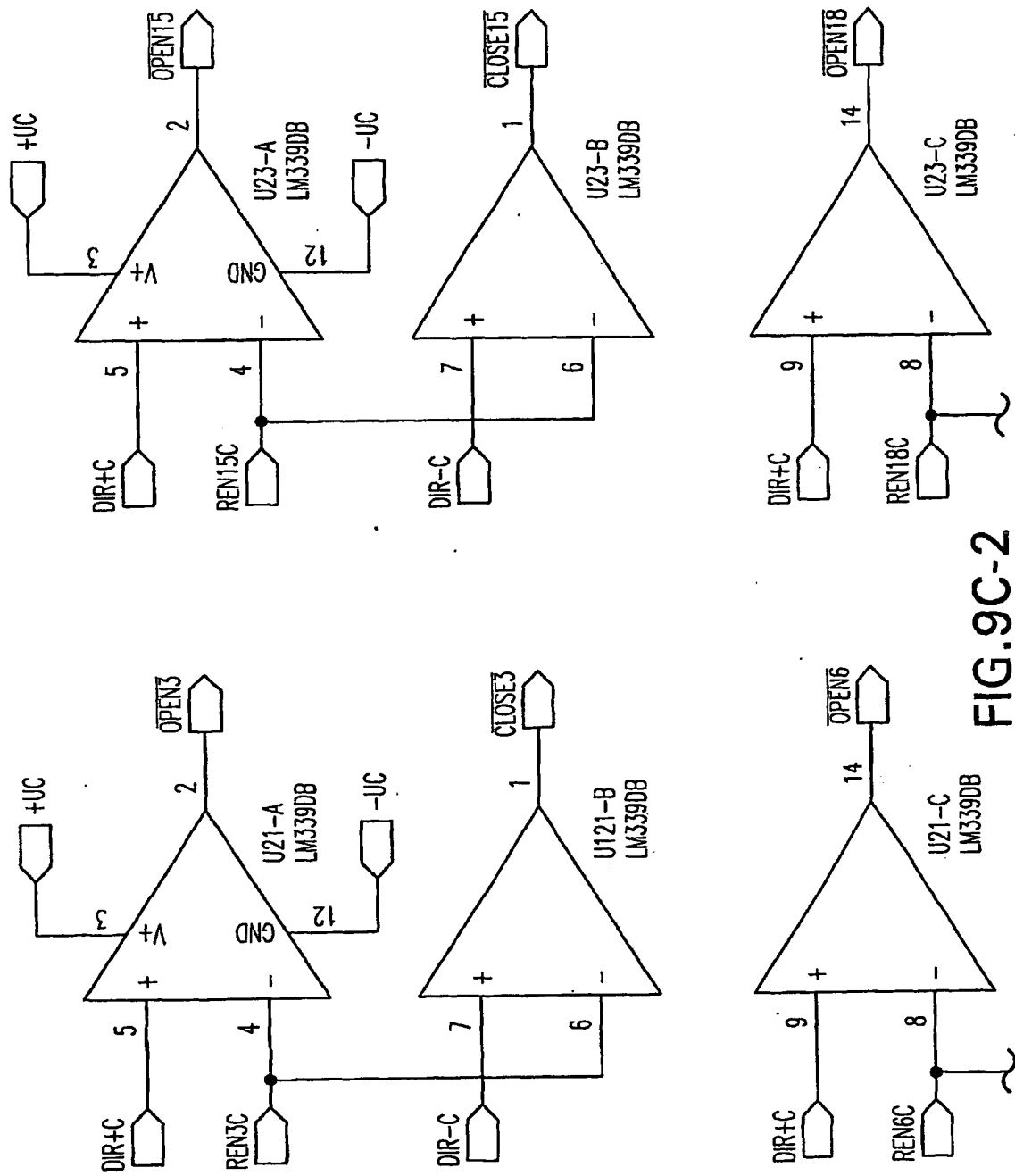


FIG. 9C-2

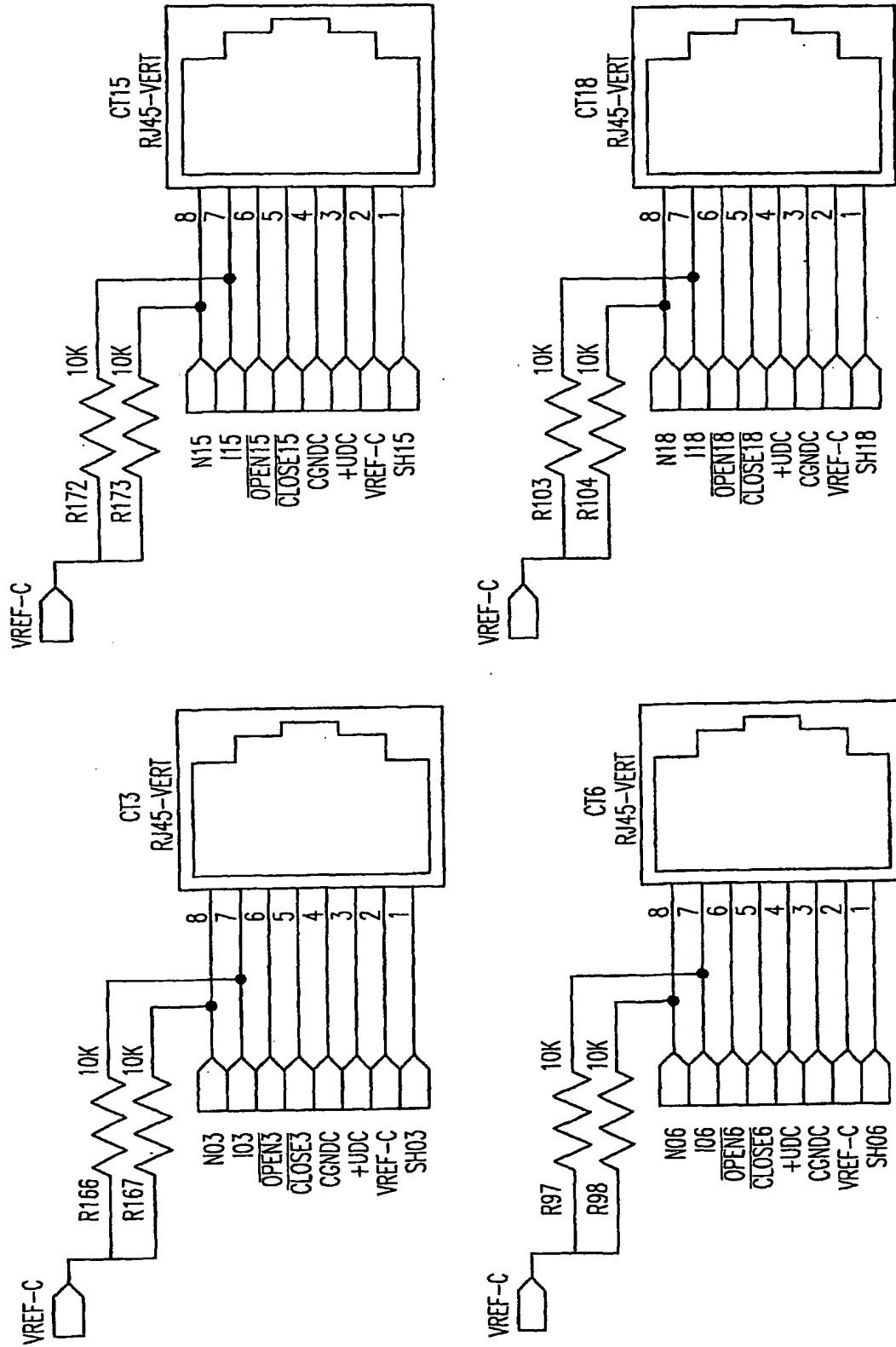


FIG.9C-3

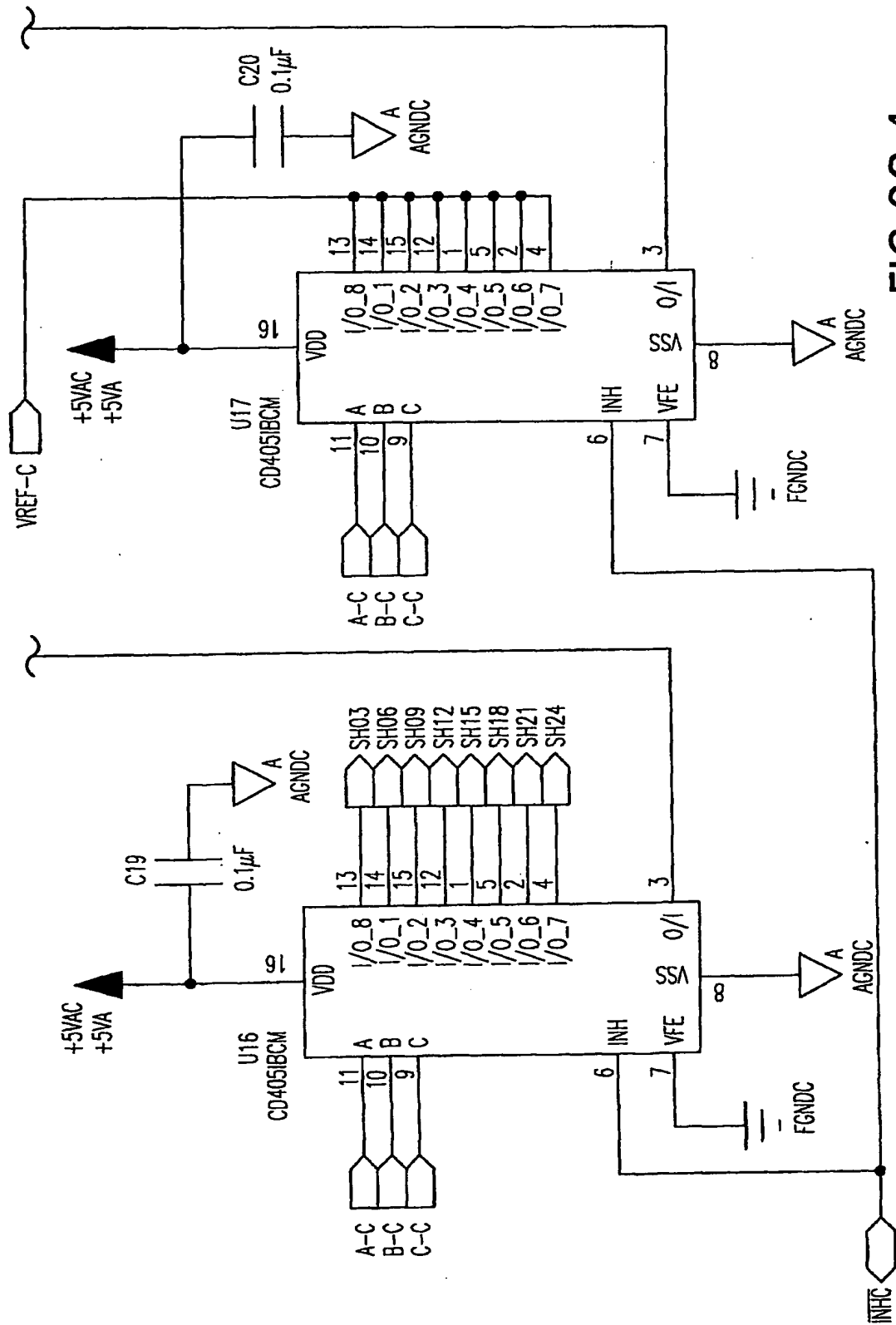


FIG. 9C-4

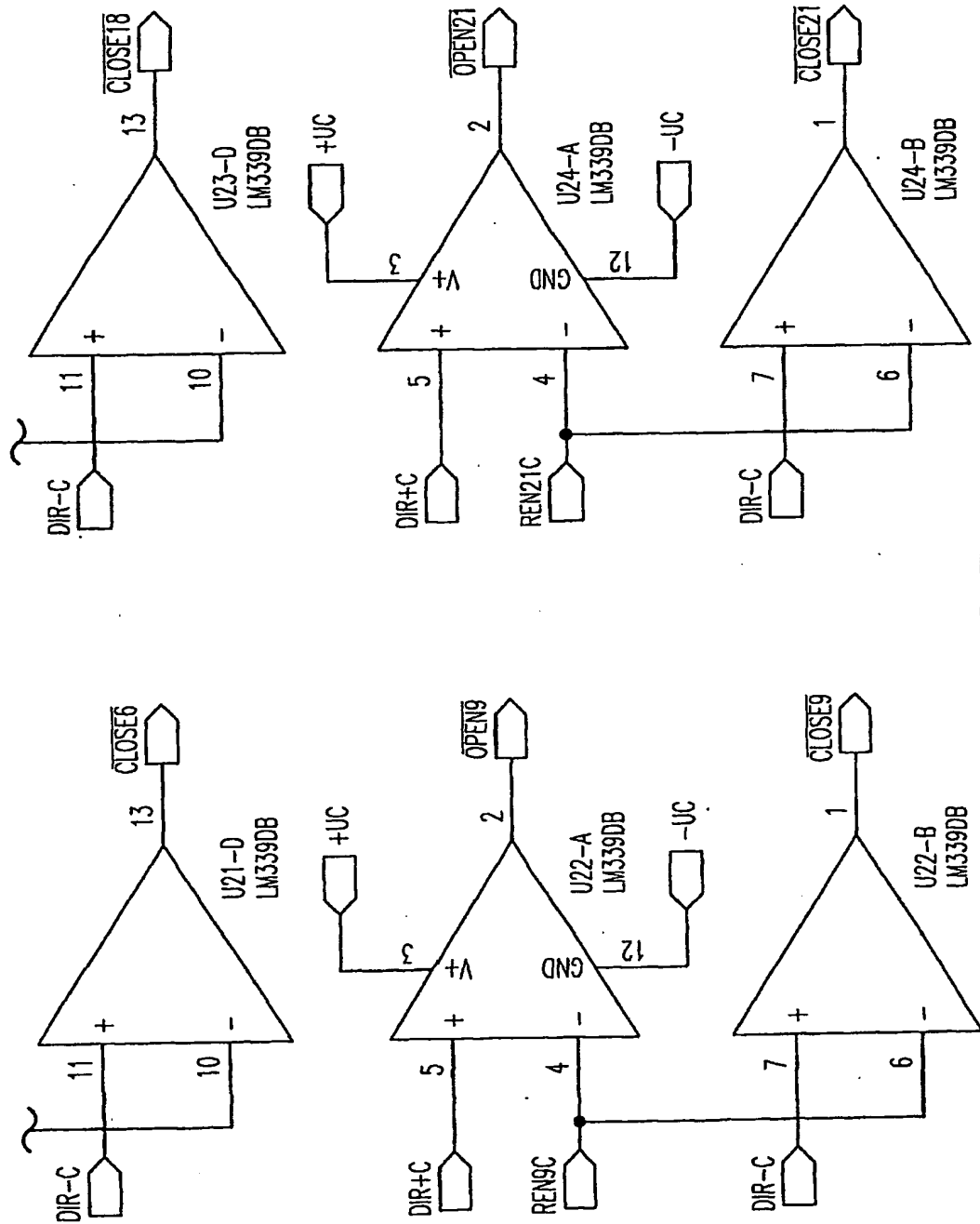


FIG. 9C-5

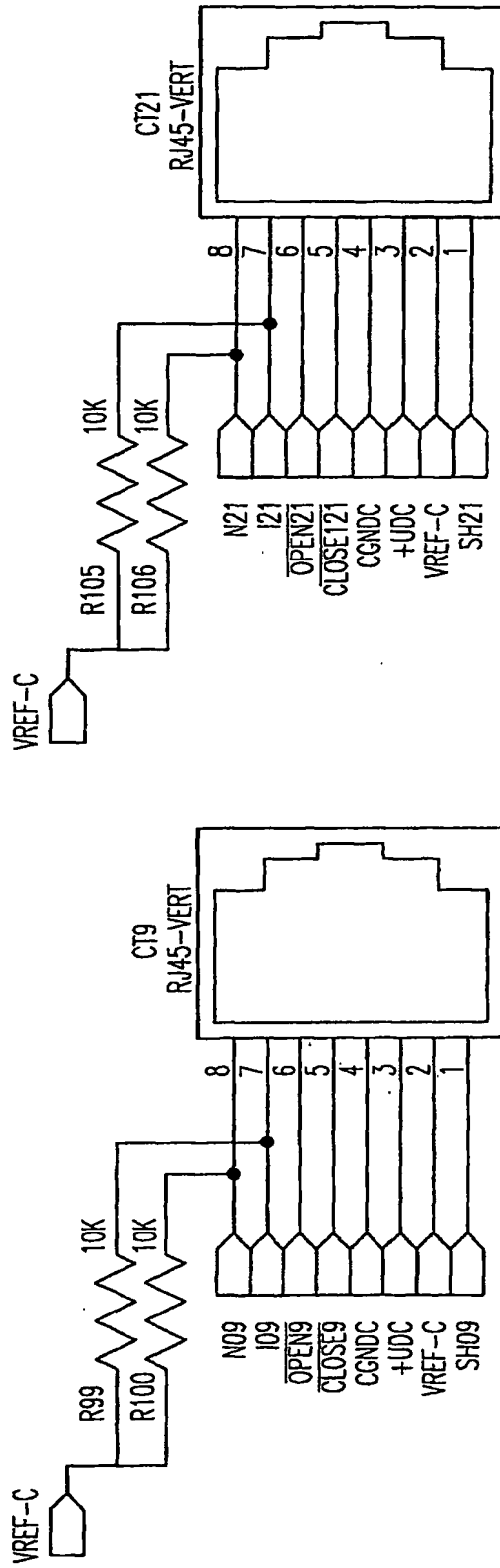


FIG.9C-6

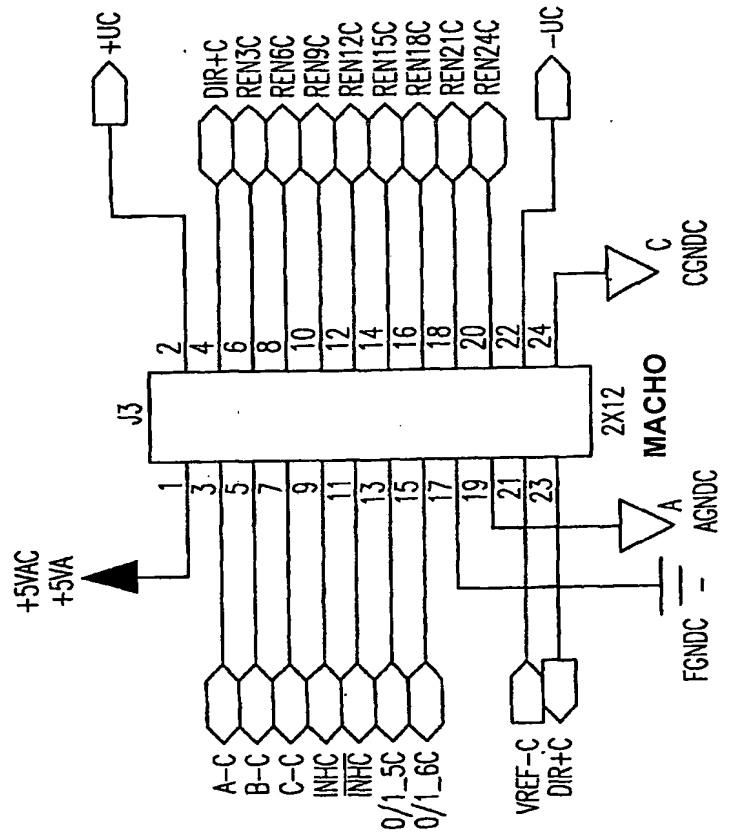
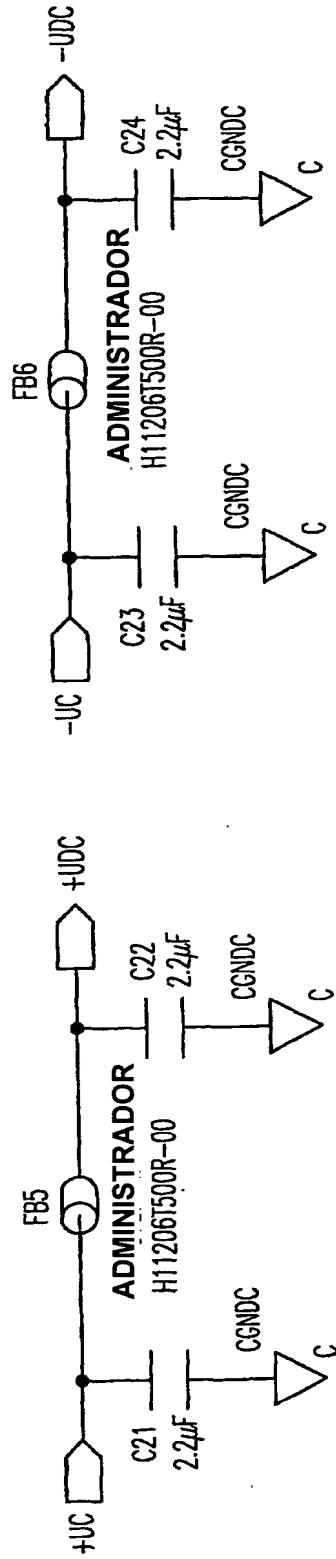


FIG.9C-7

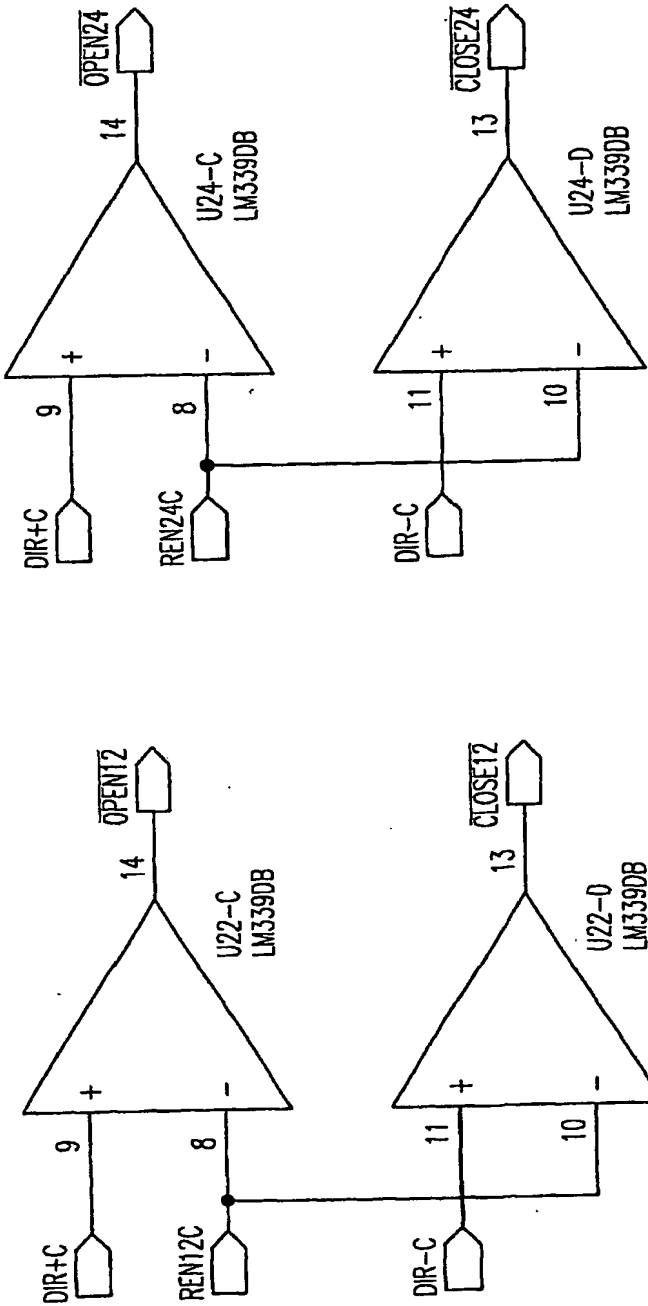


FIG.9C-8

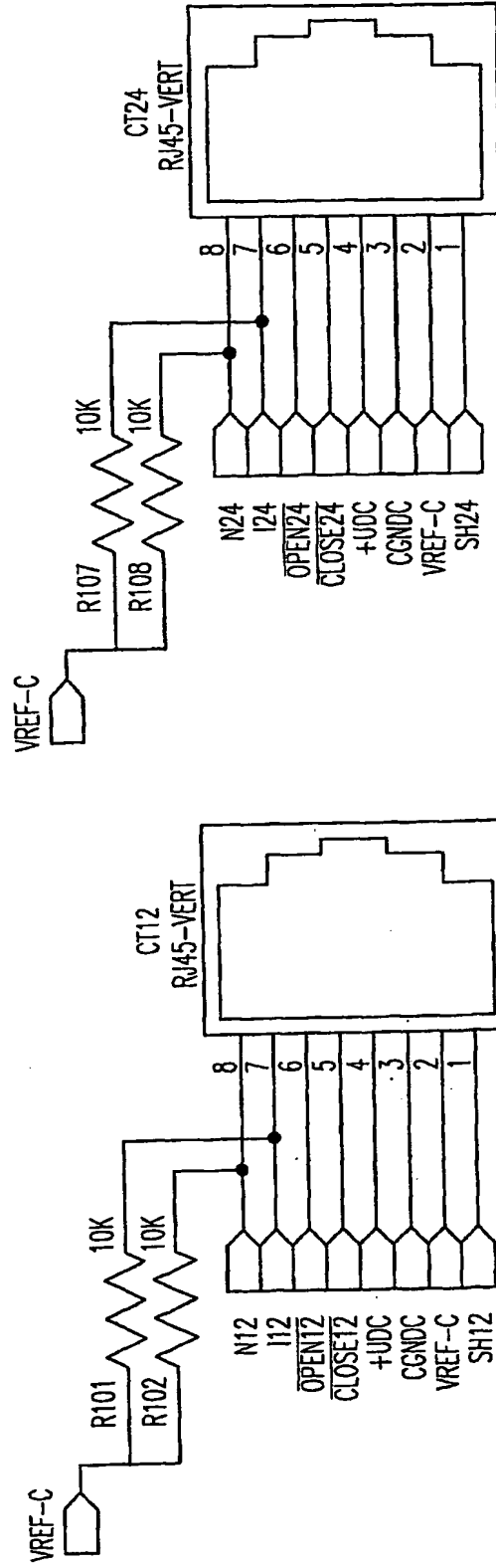


FIG.9C-9

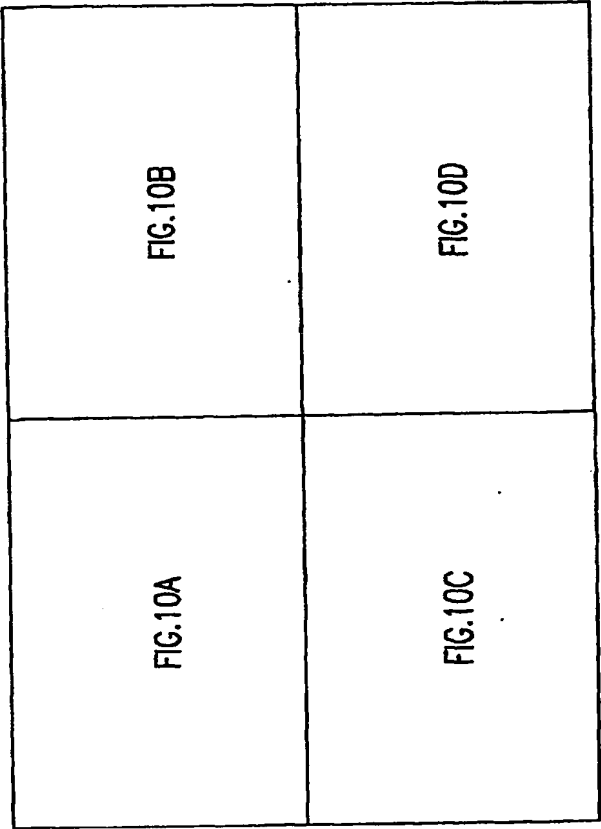


FIG.10

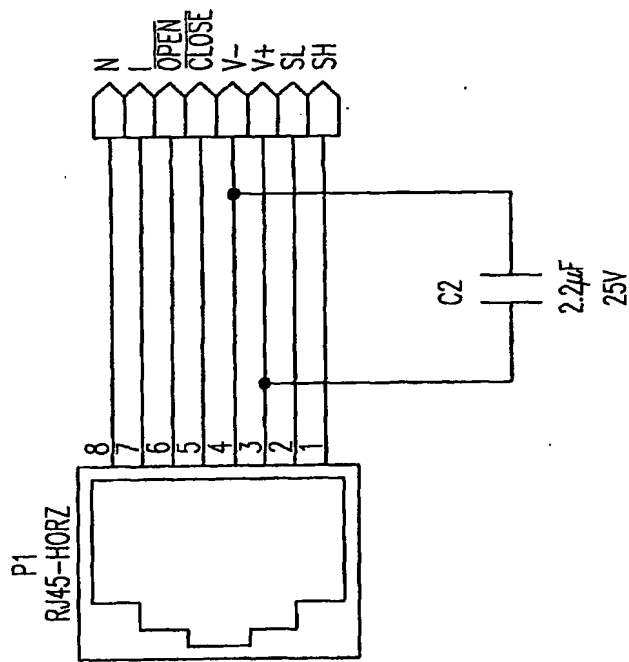


FIG.10A

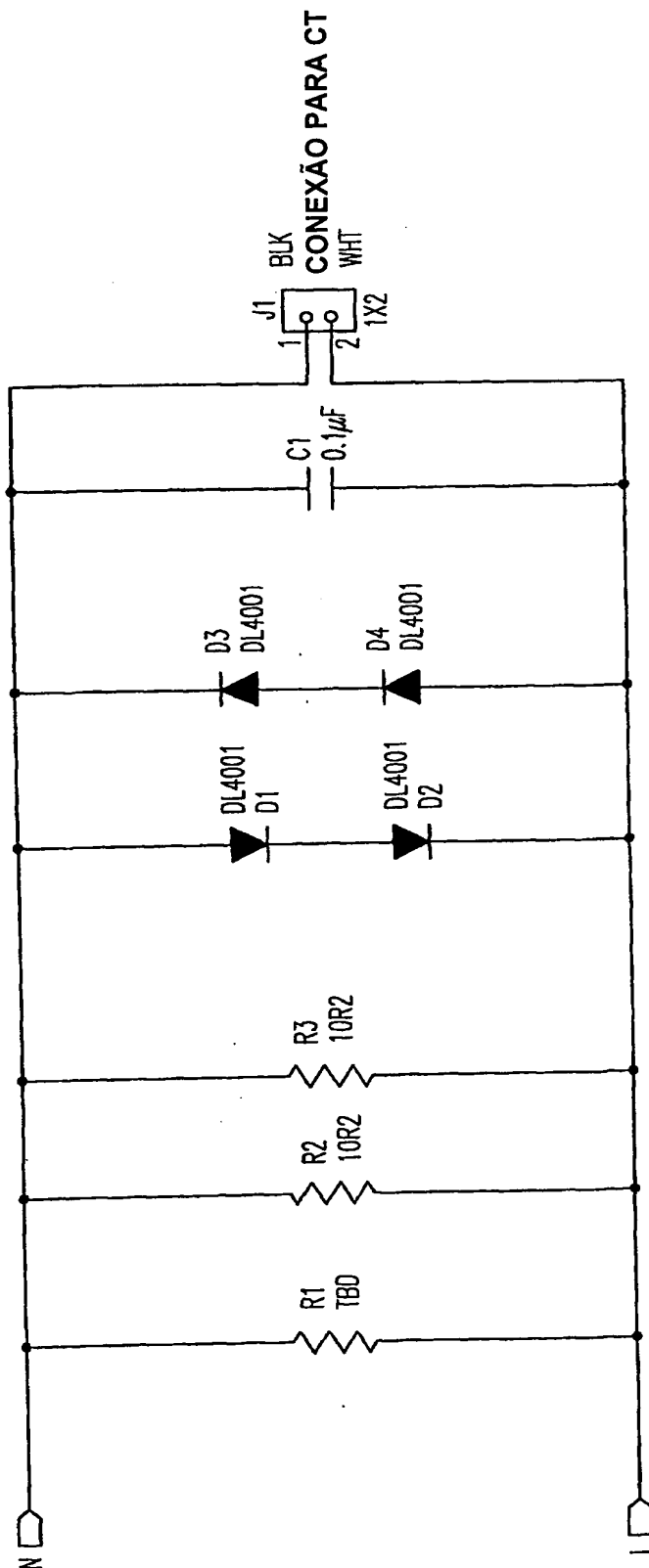


FIG.10B

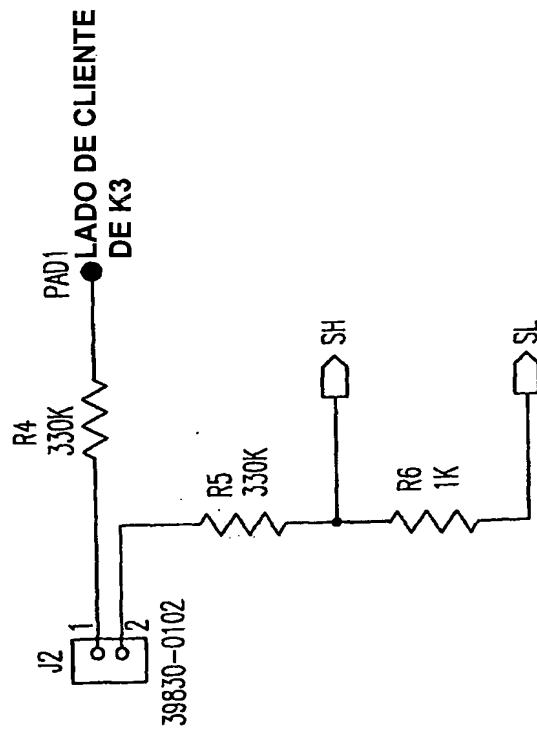


FIG.10C

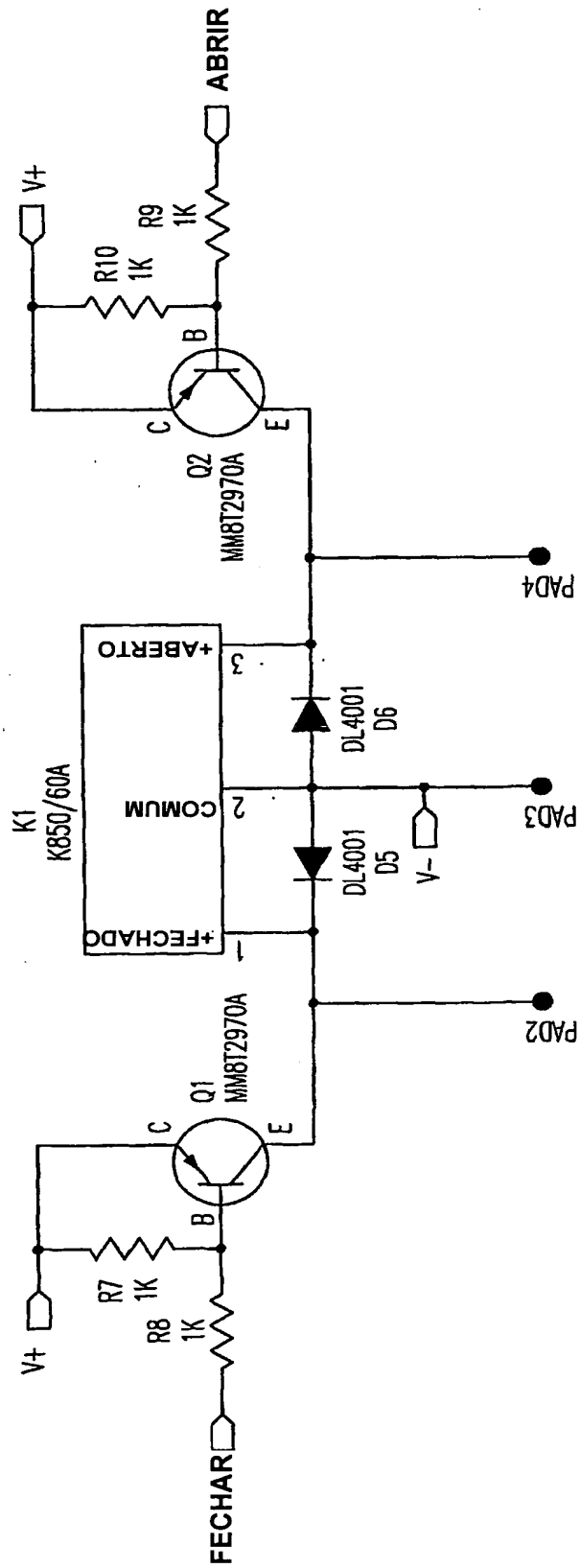


FIG. 10D

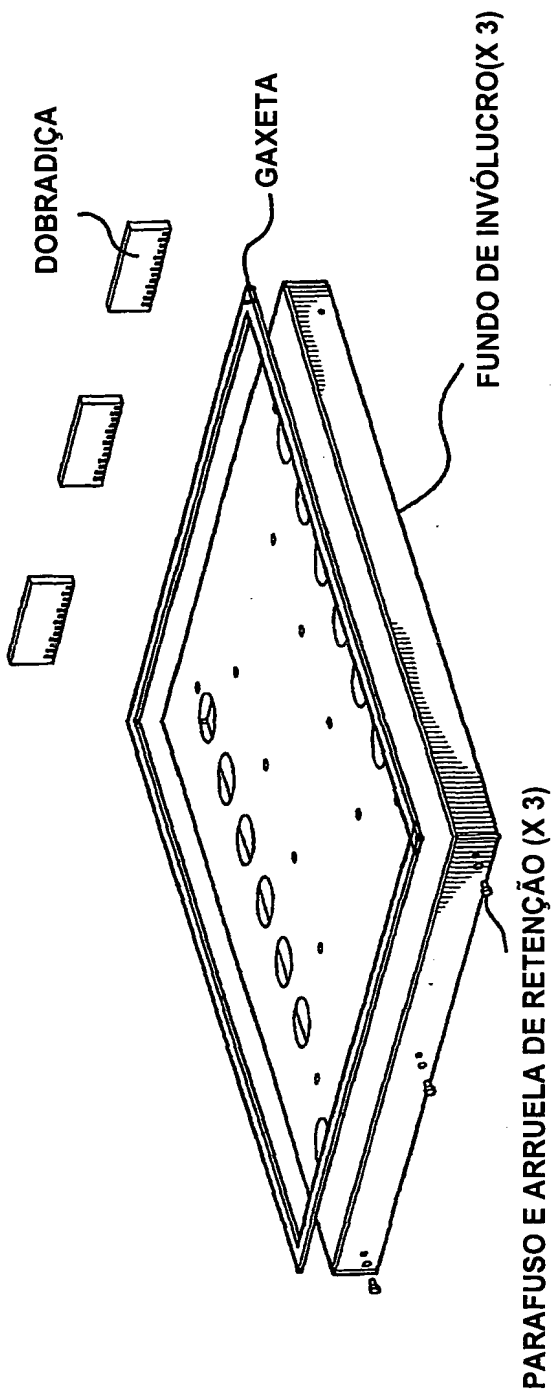


FIG.11

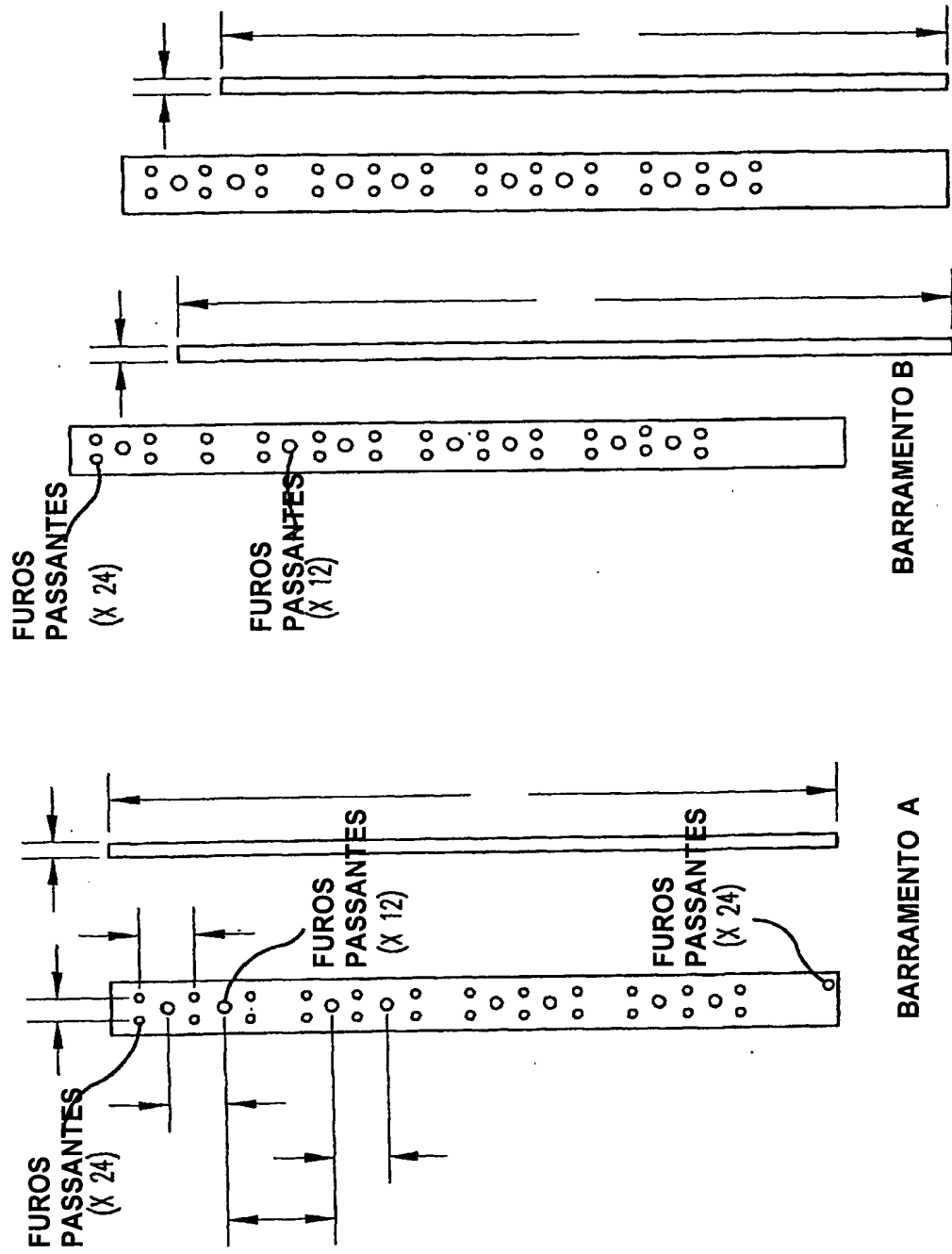


FIG.12

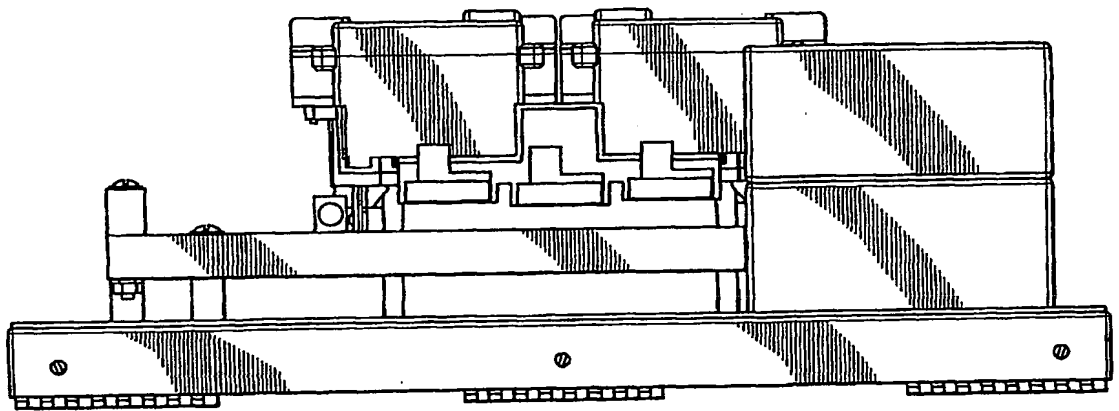


FIG.13

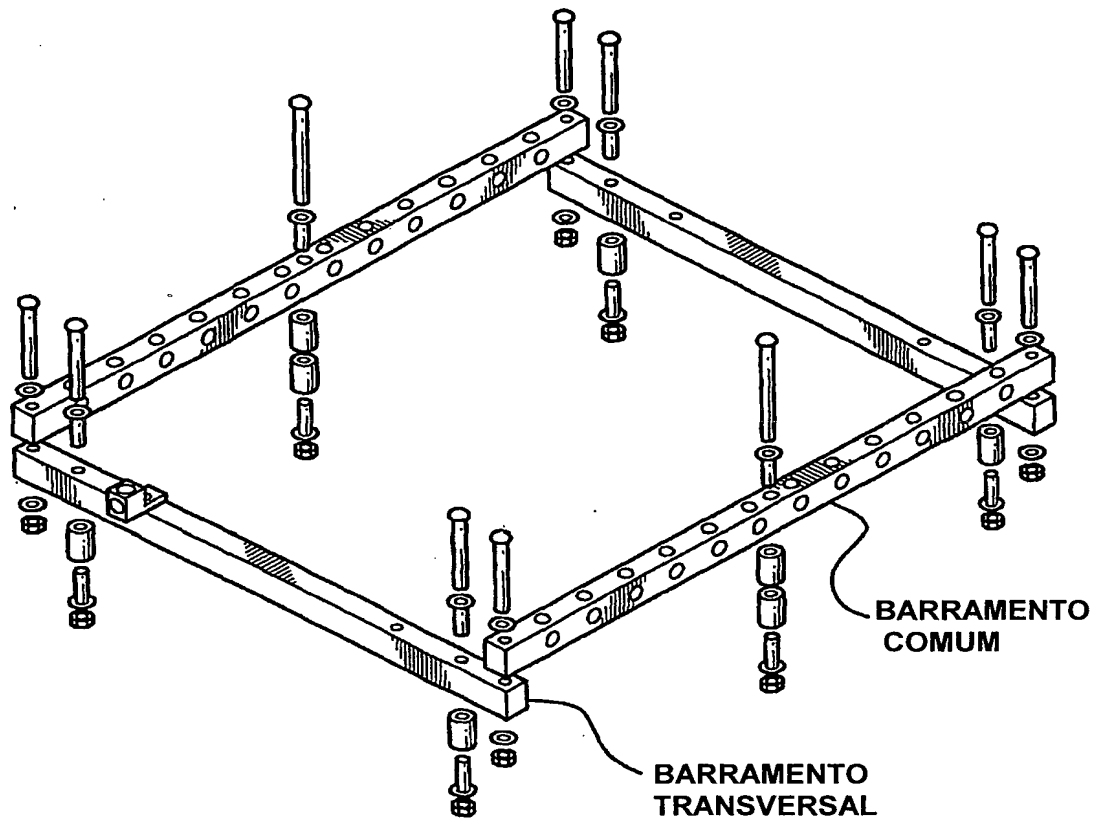


FIG.14

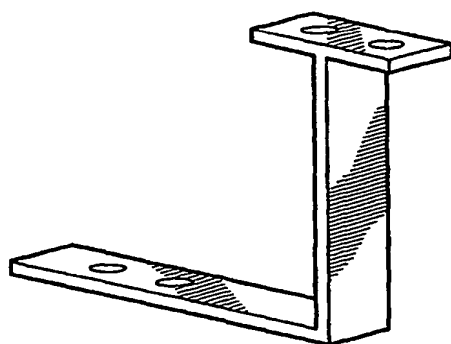


FIG. 15A

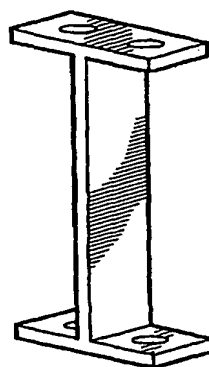


FIG. 15B

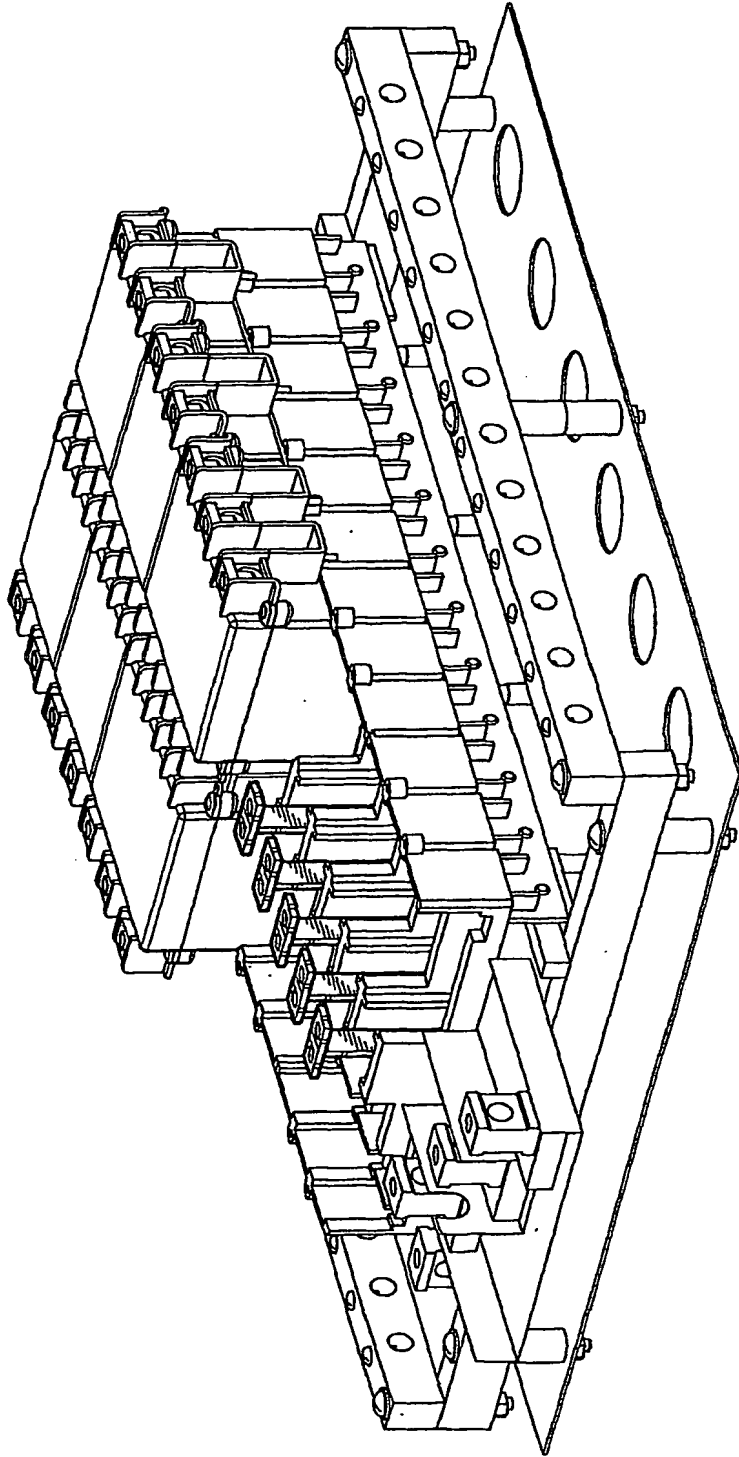


FIG.16

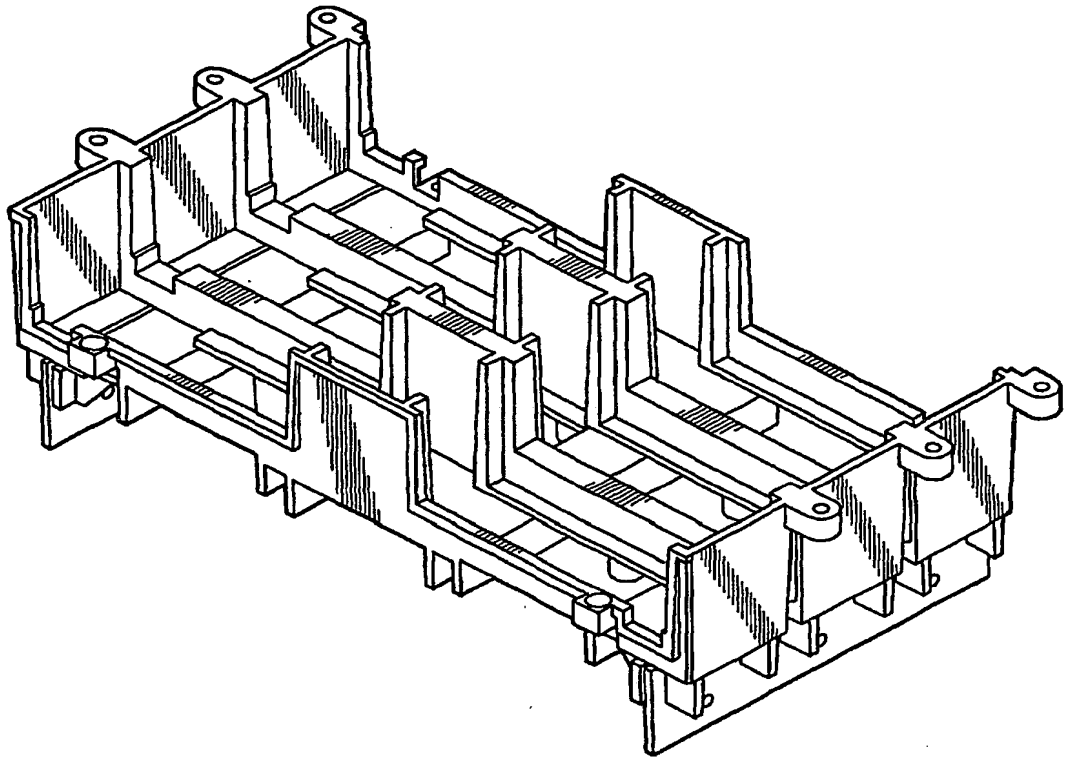


FIG.17

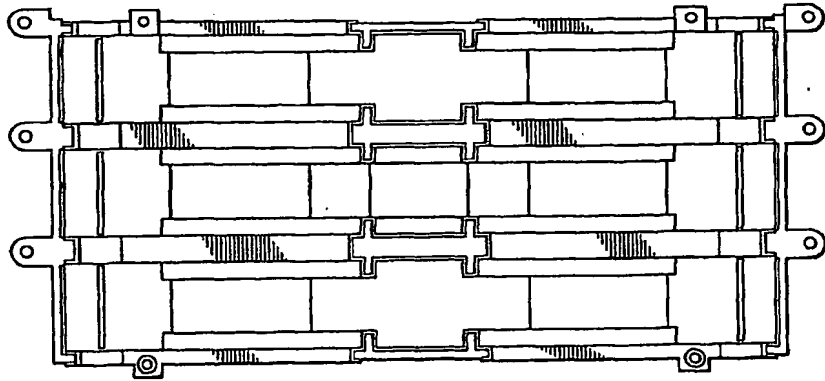


FIG. 18A

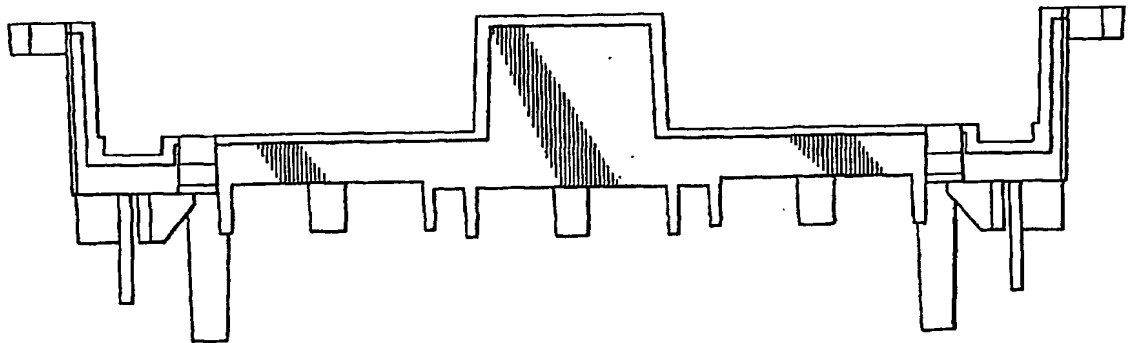


FIG. 18B

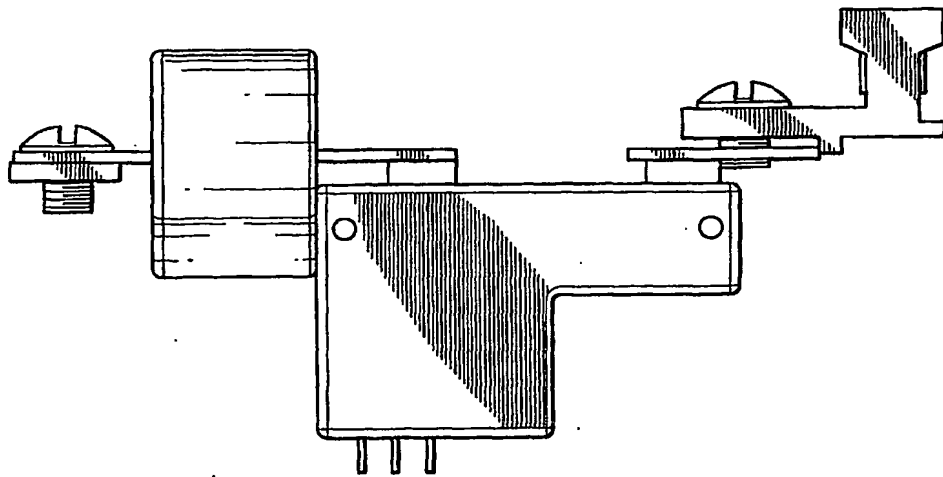


FIG. 19

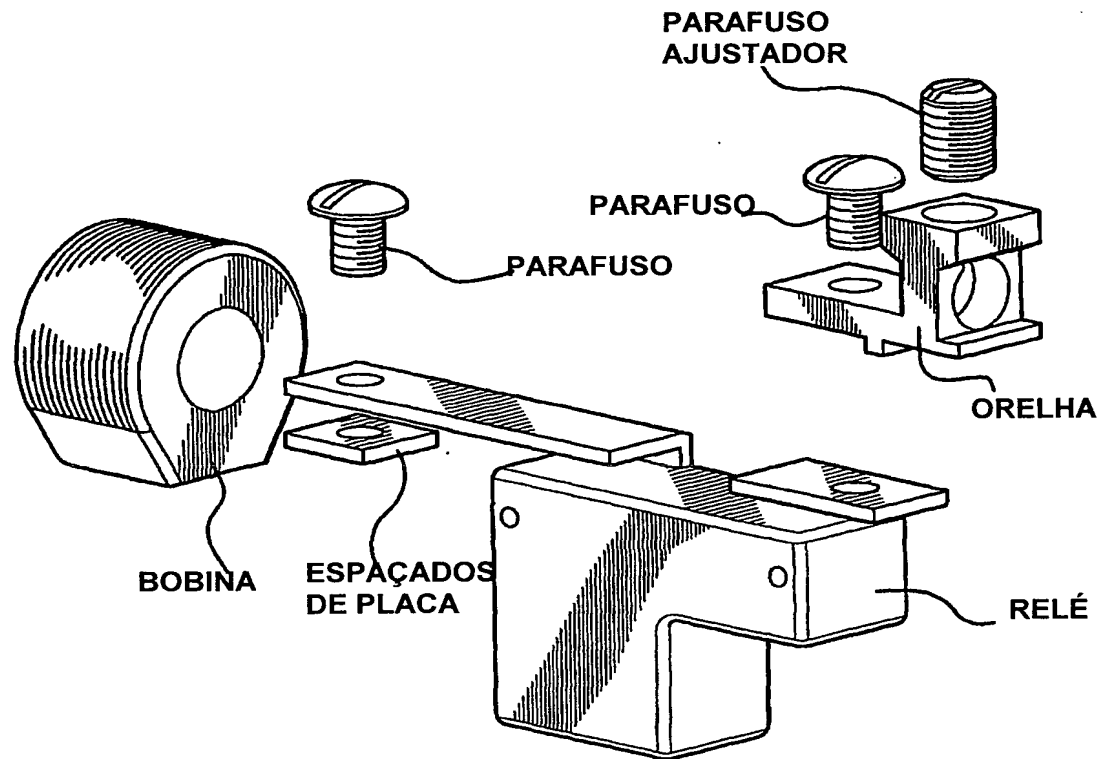


FIG.20

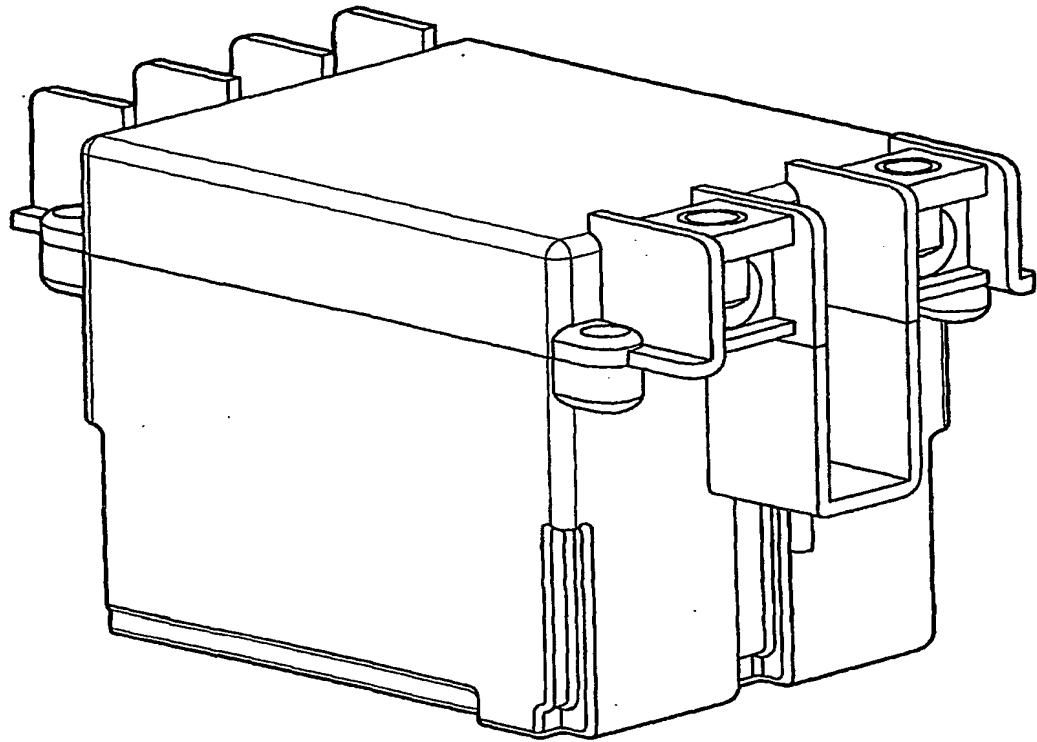
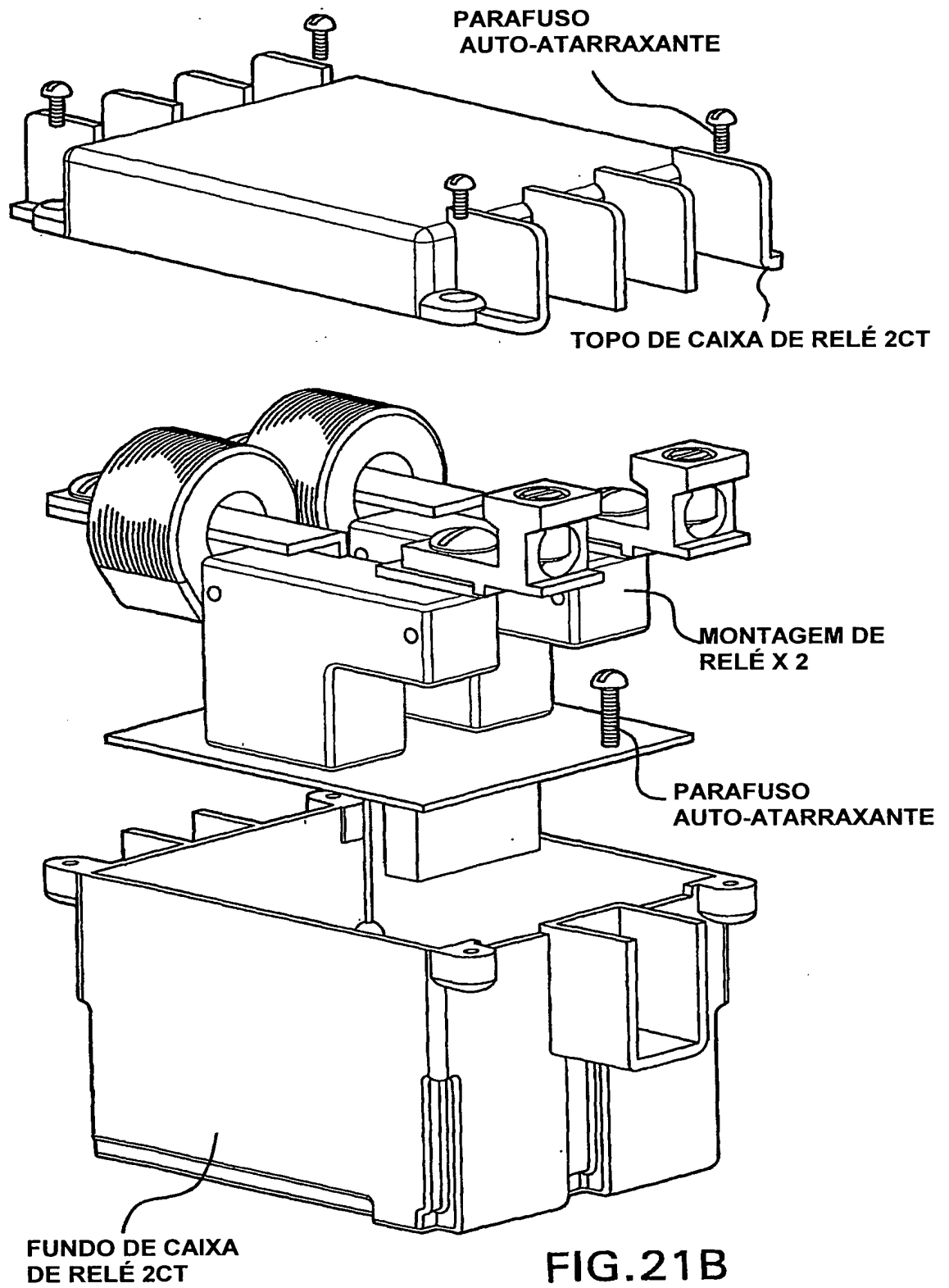


FIG.21A



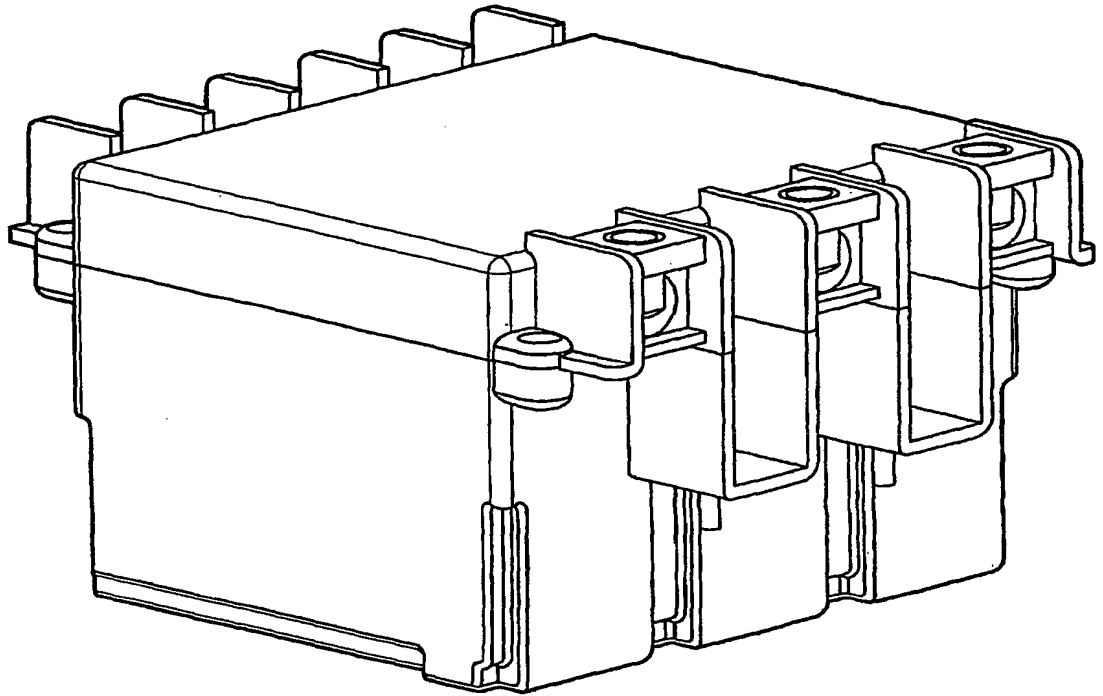
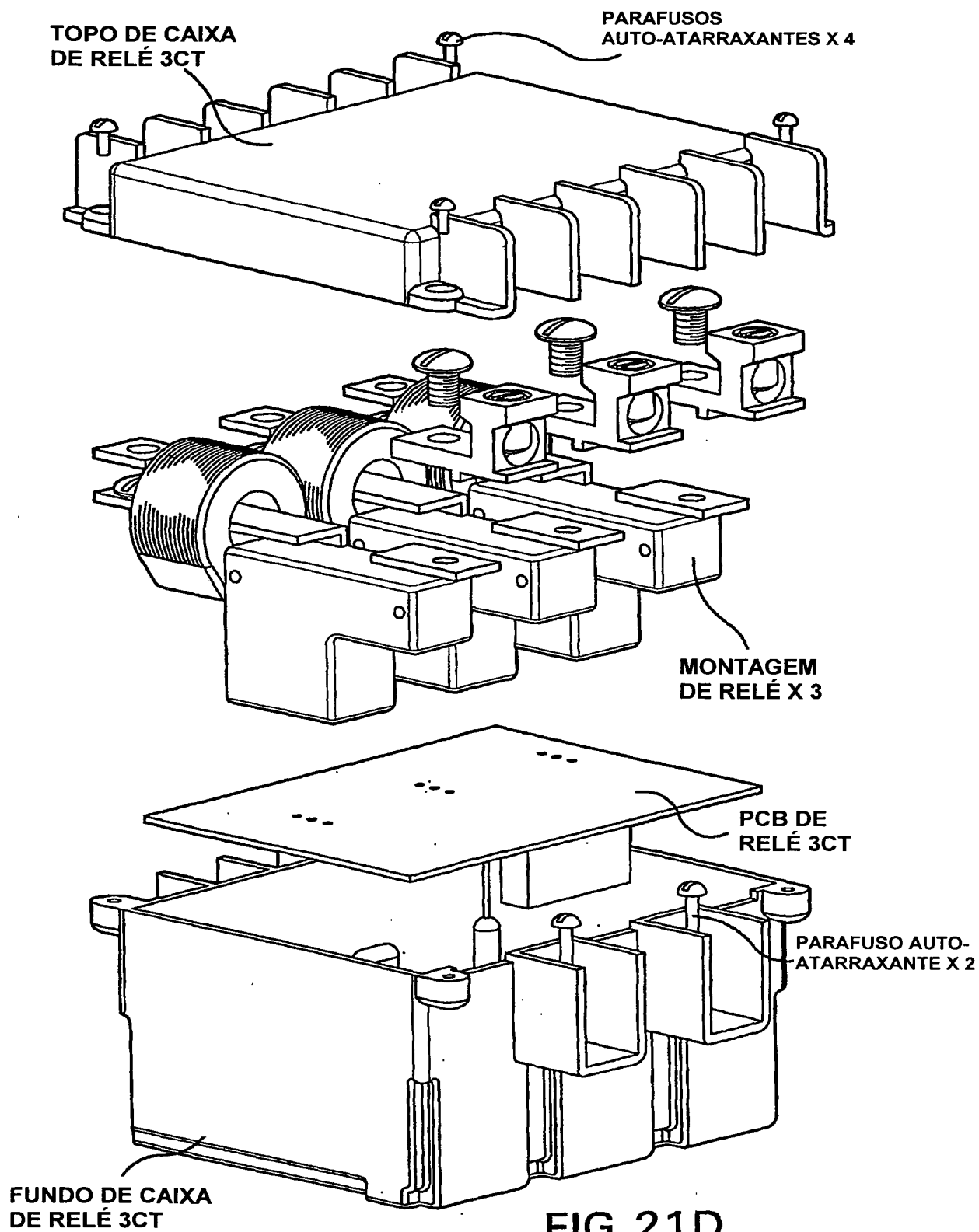


FIG.21C



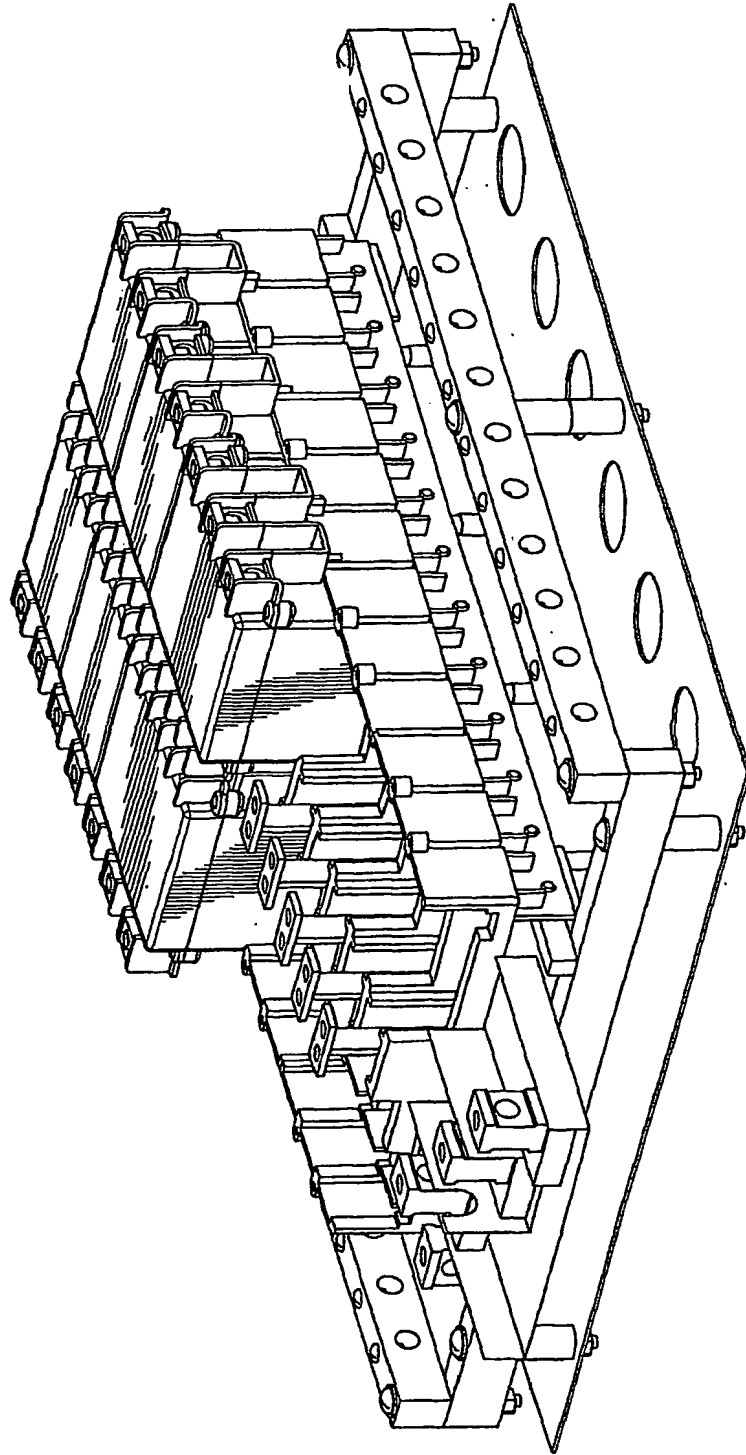
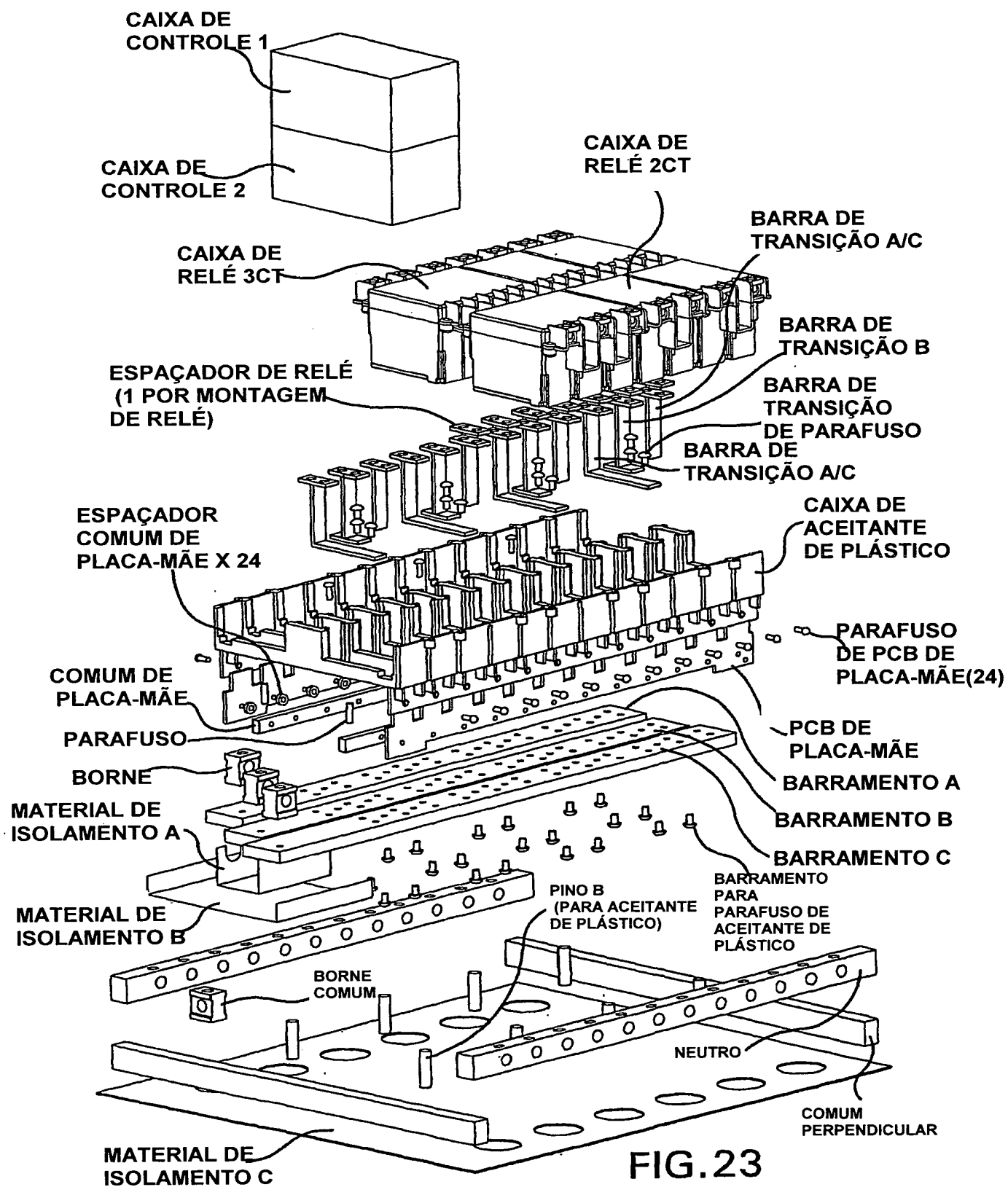


FIG.22



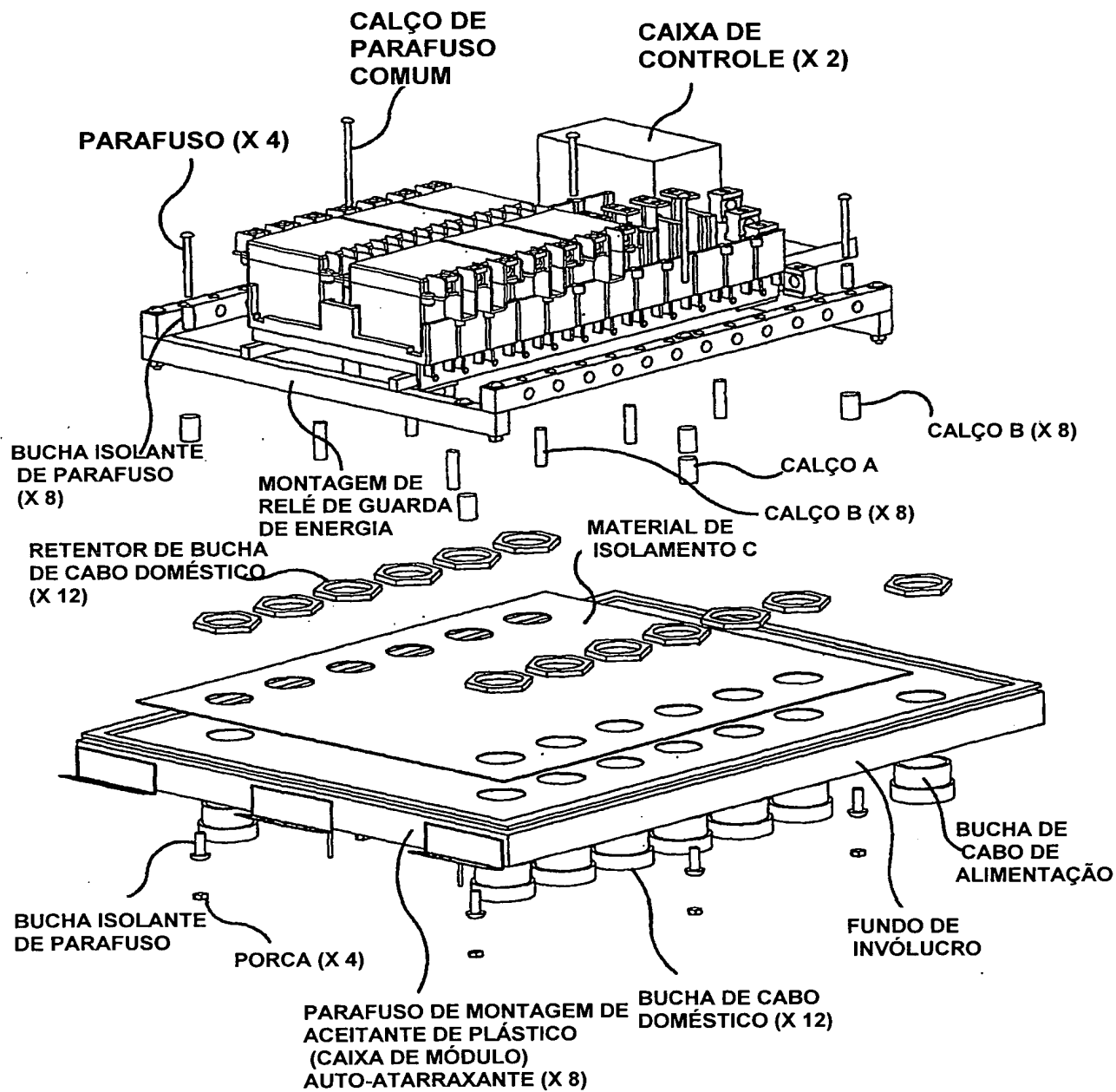
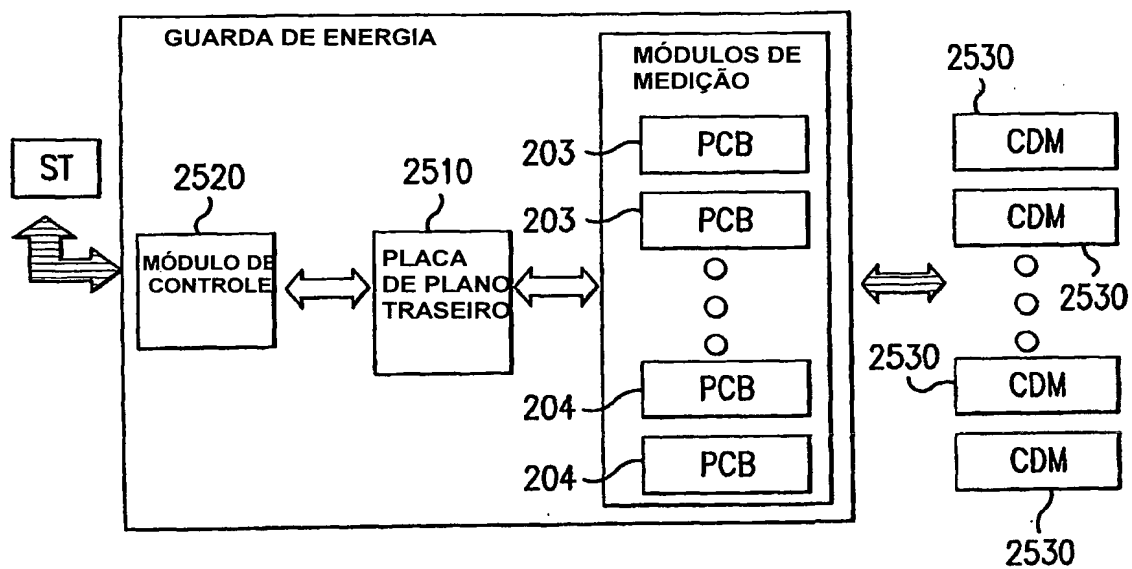


FIG.24



ST: TRANSPONDER DE VARREDURA INSTALADO REMOTO A EG



COMUNICAÇÃO DE PLC

FIG.25

CPU PCB203

FIG.26A-1	FIG.26A-2
FIG.26A-3	FIG.26A-4

FIG.26A

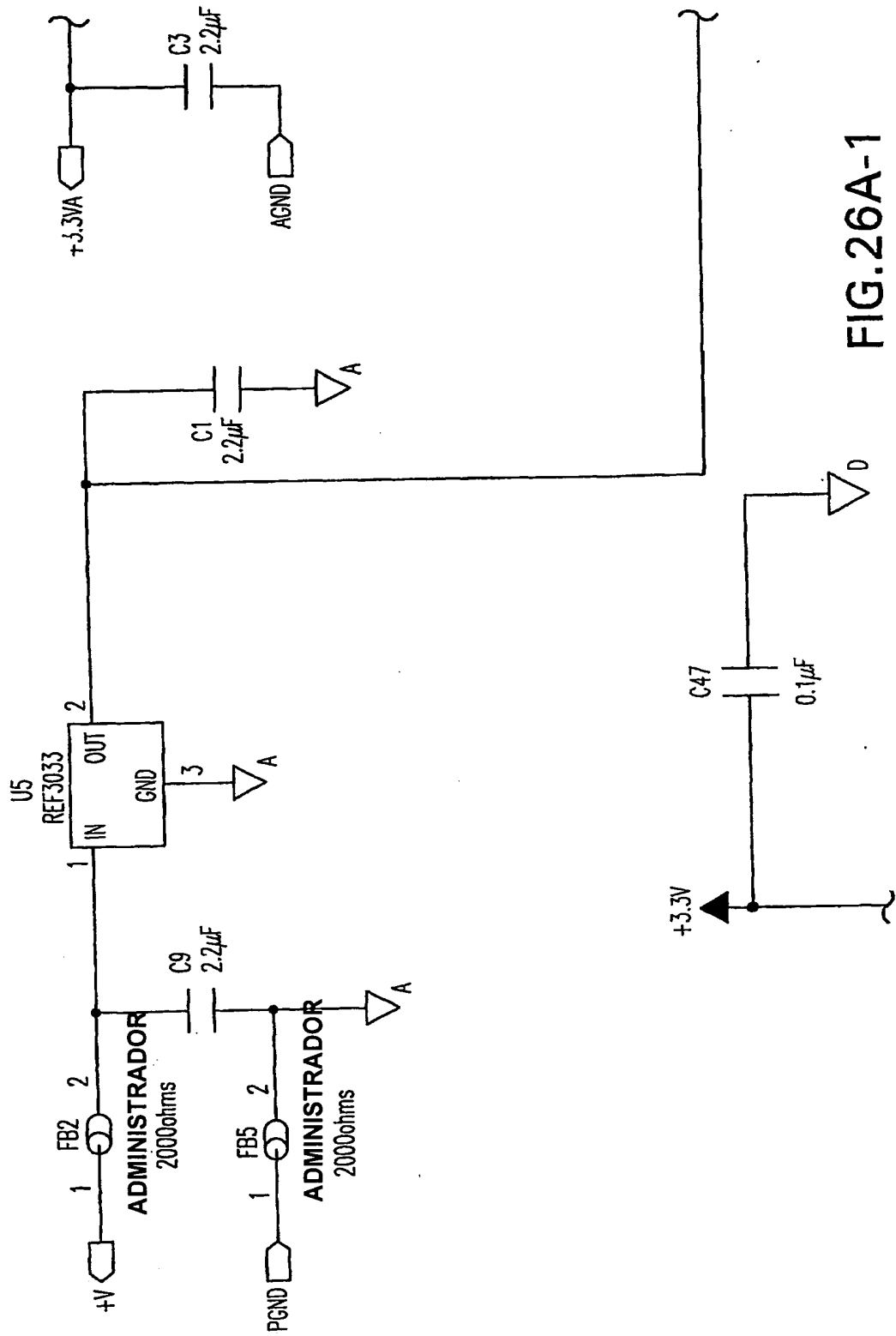


FIG. 26A-1

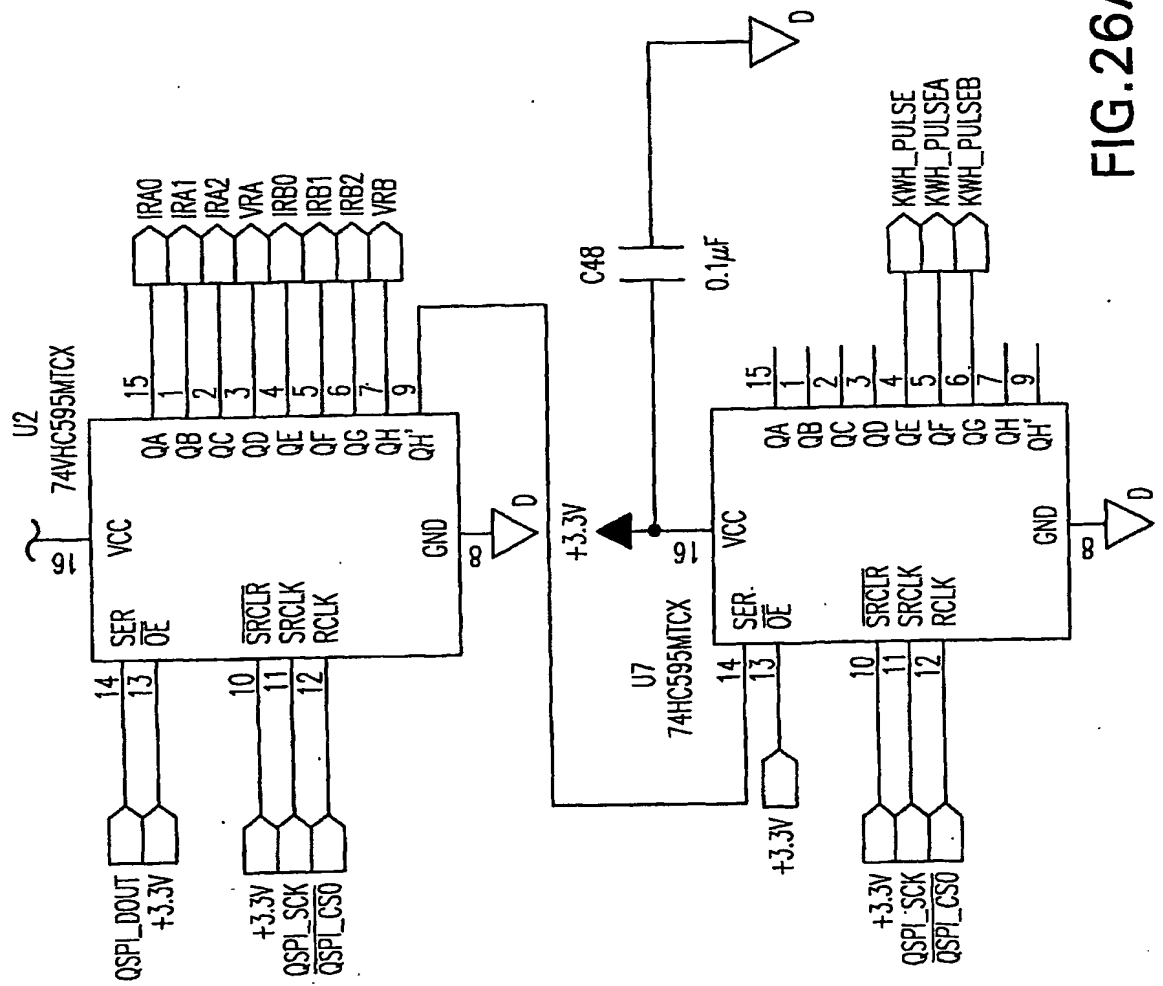


FIG.26A-3

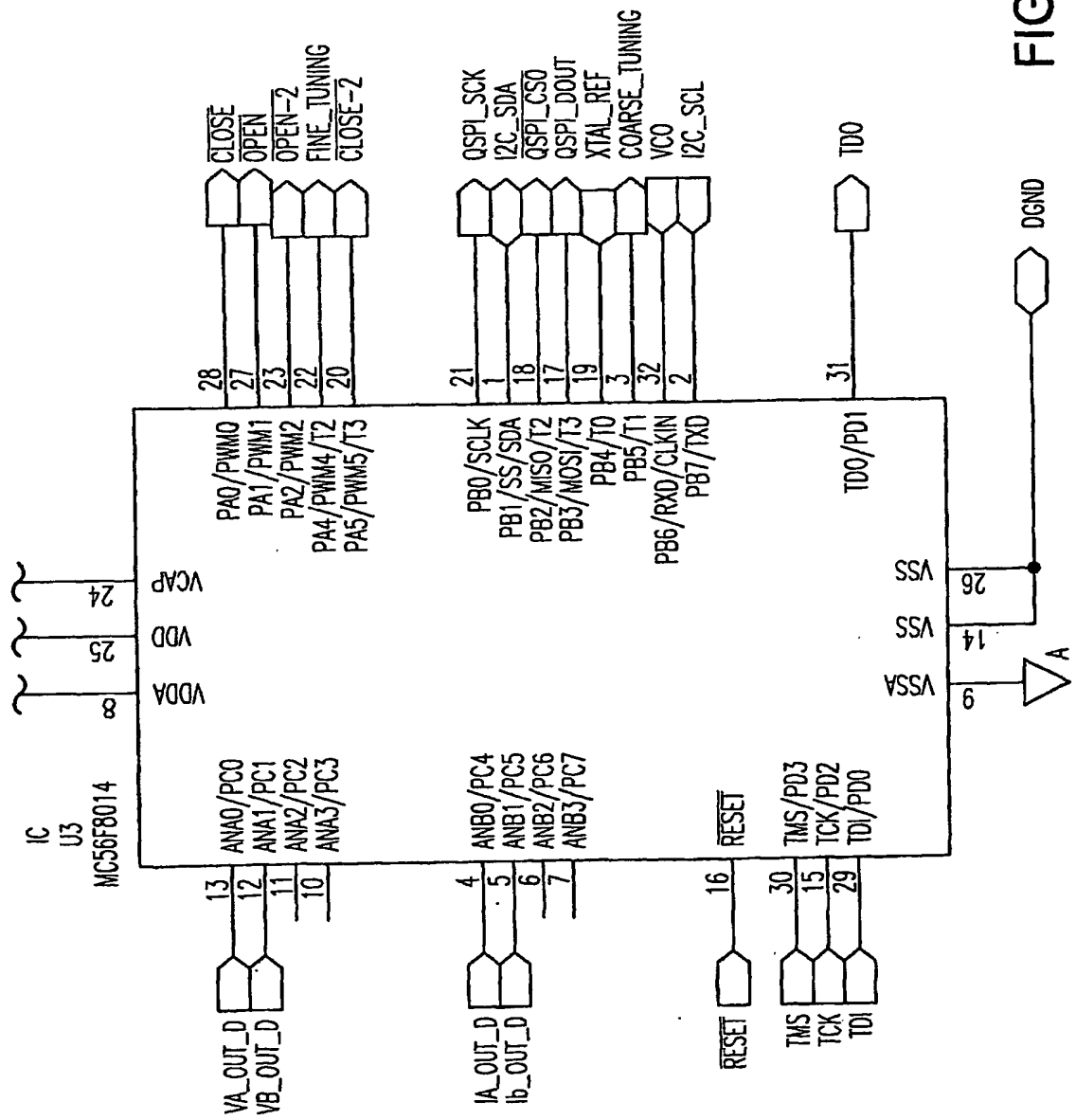


FIG.26A-4

MEDIDOR I PCB203

FIG.26B-1	FIG.26B-2	FIG.26B-3
FIG.26B-4	FIG.26B-5	FIG.26B-6

FIG.26B

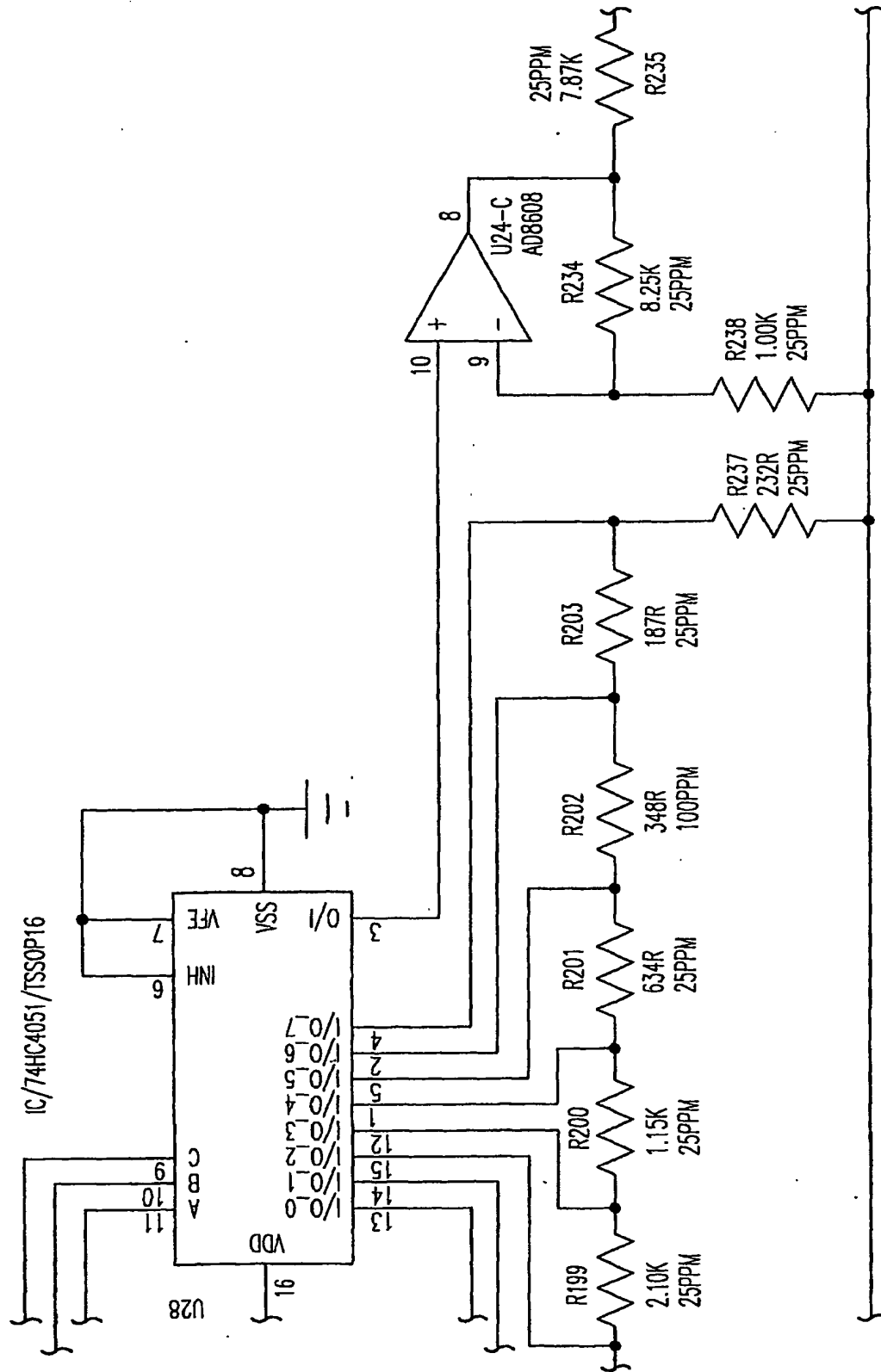


FIG. 26B-2

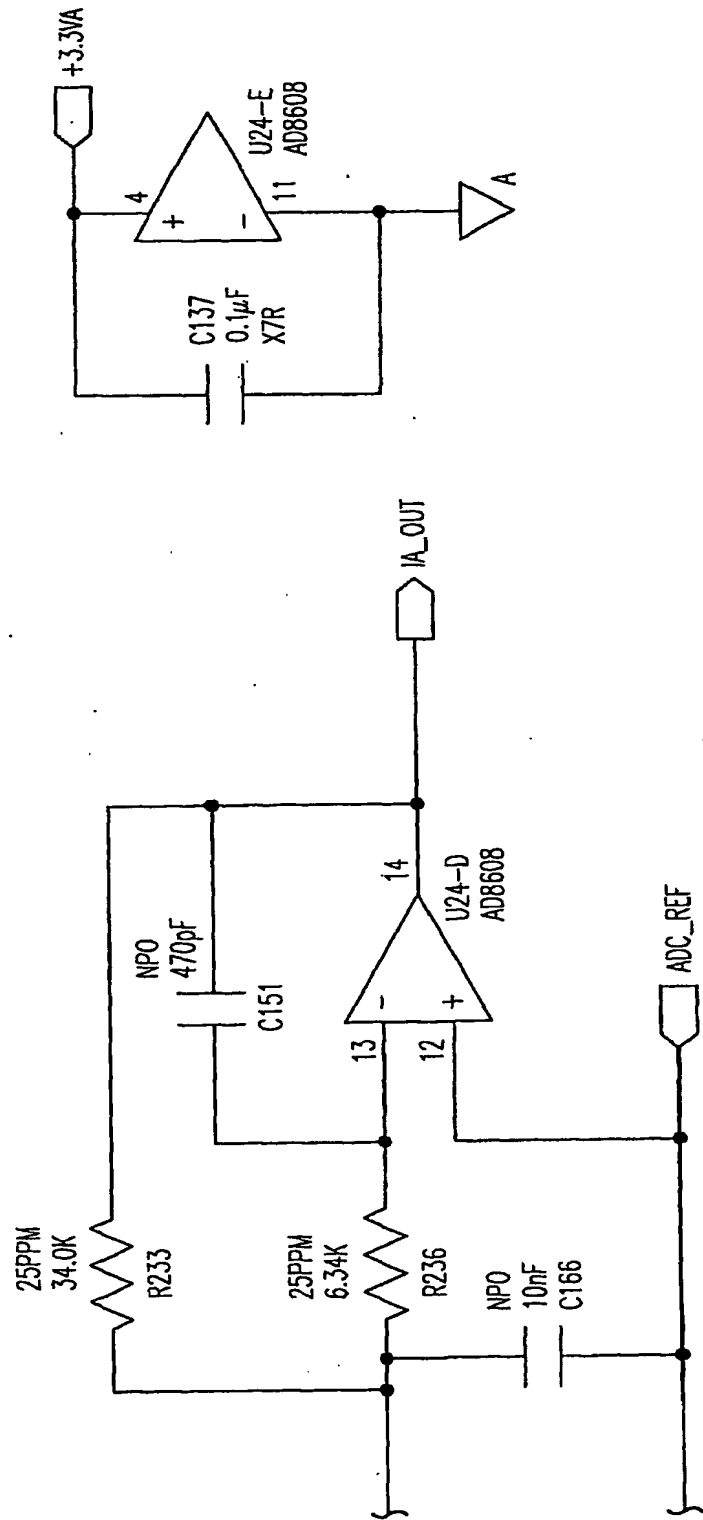


FIG. 26B-3

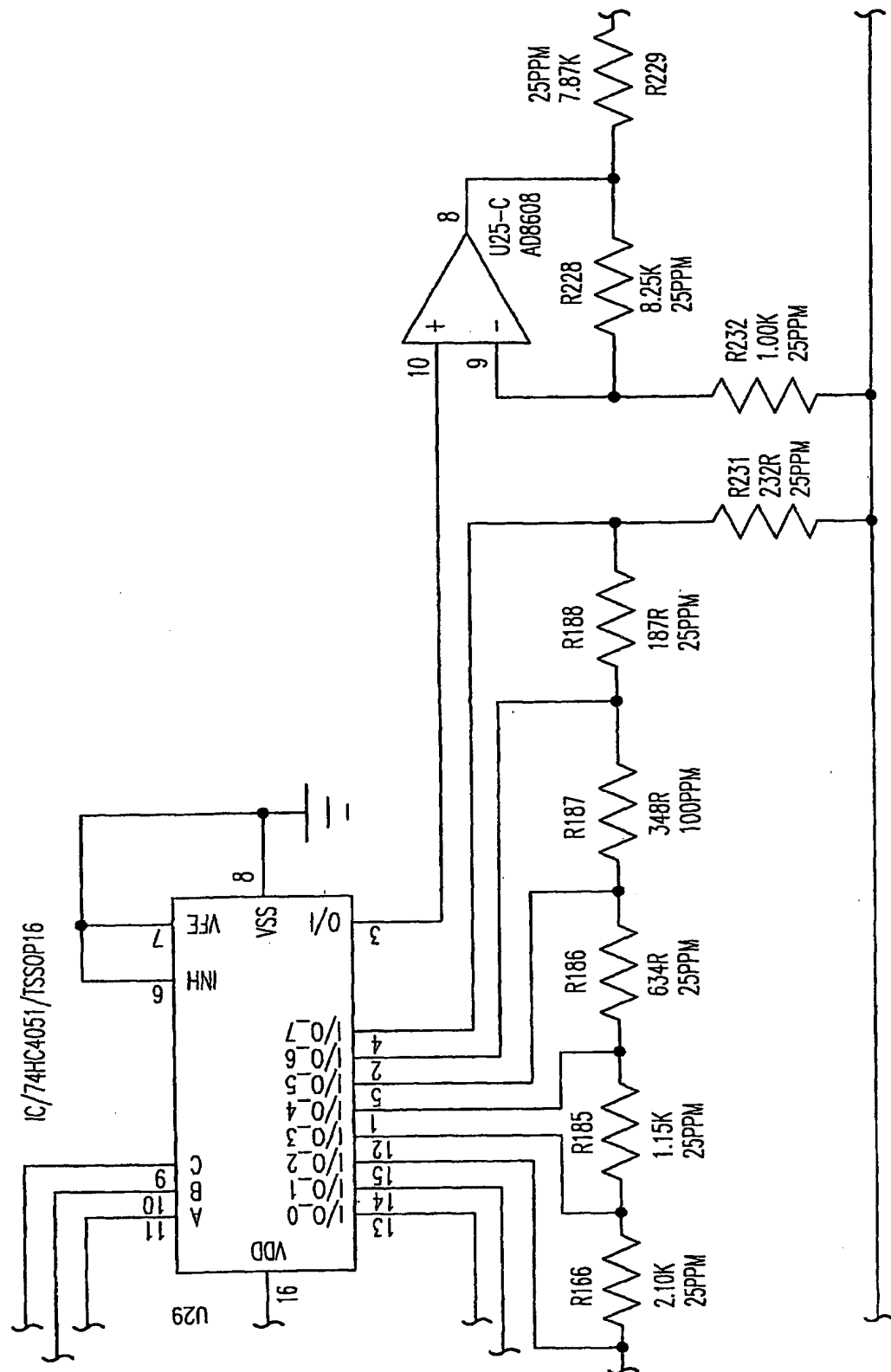


FIG.26B-5

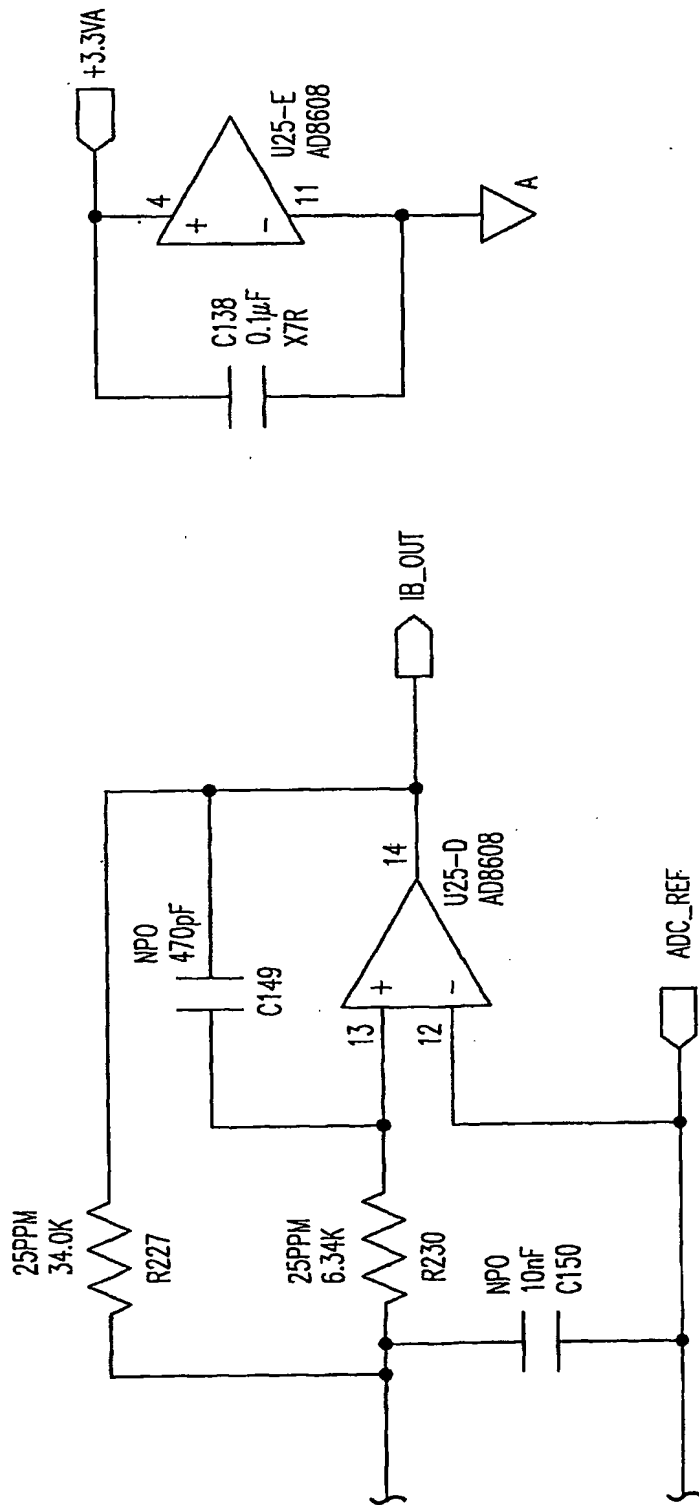


FIG. 26B-6

MEDIDOR V PCB203

FIG.26C-1	FIG.26C-2
FIG.26C-3	FIG.26C-4

FIG.26C

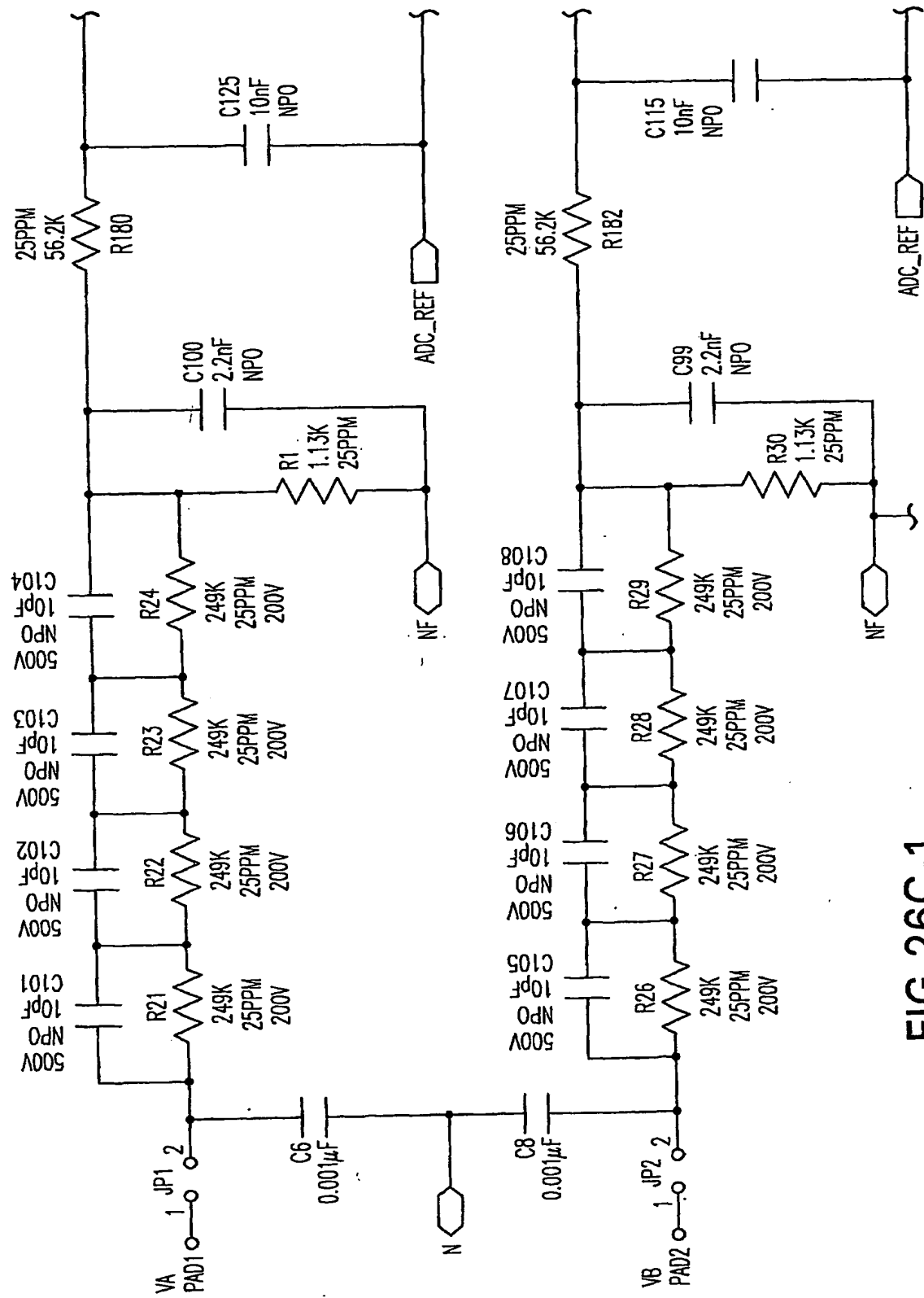


FIG. 26C-1

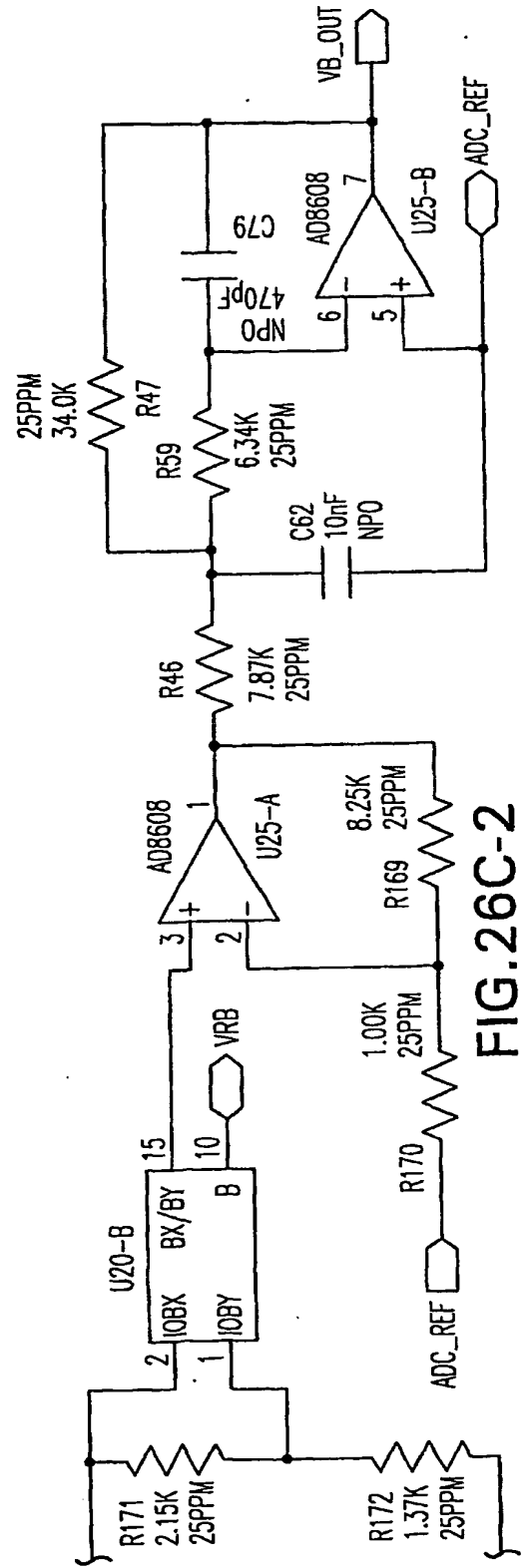
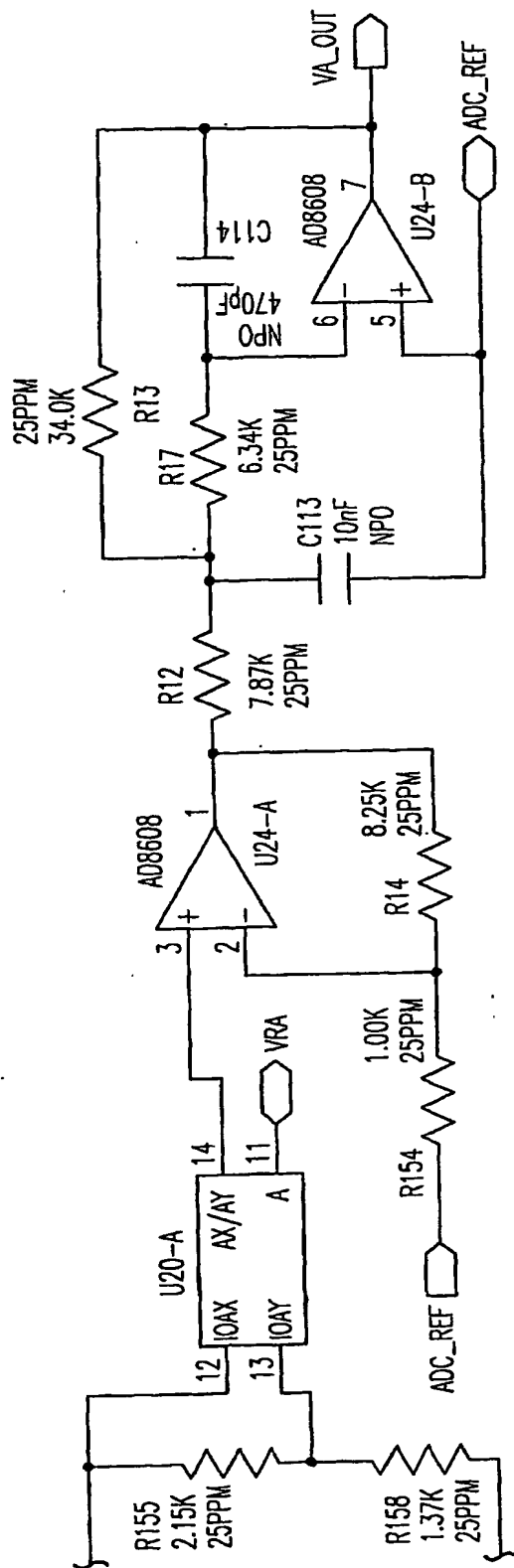


FIG. 26C-2

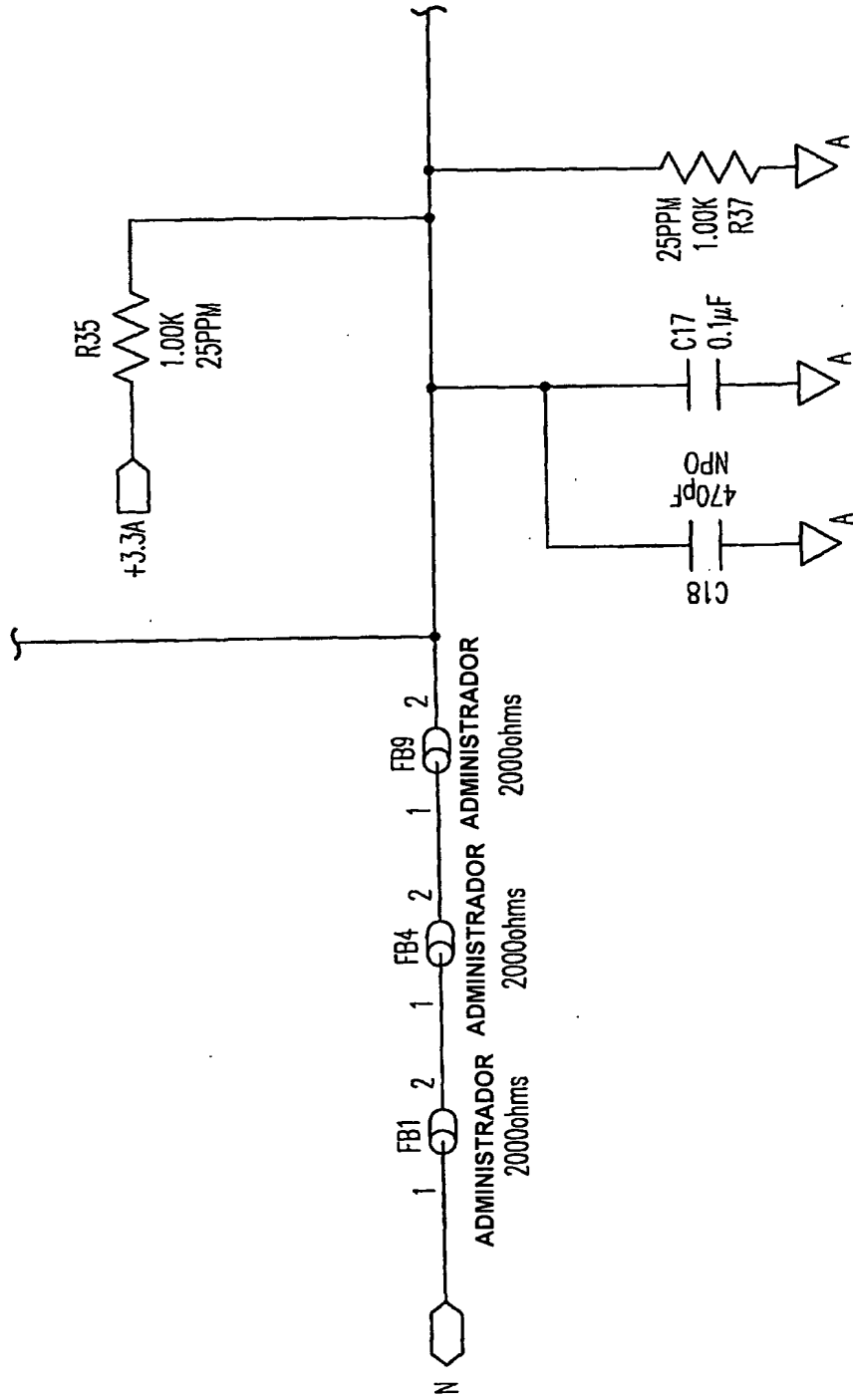


FIG.26C-3

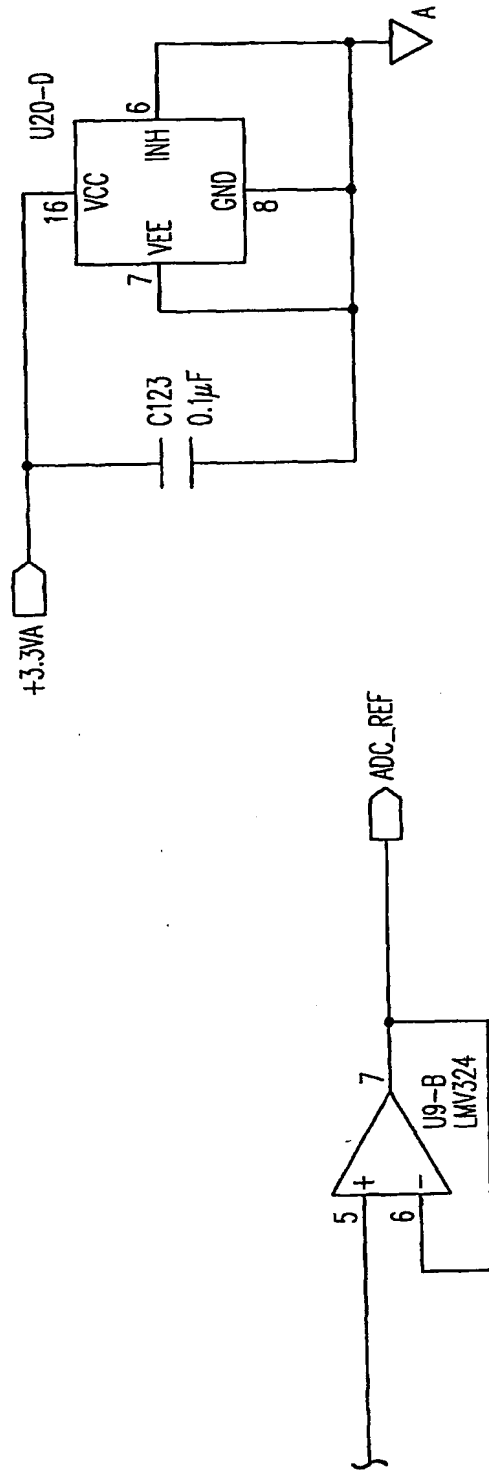
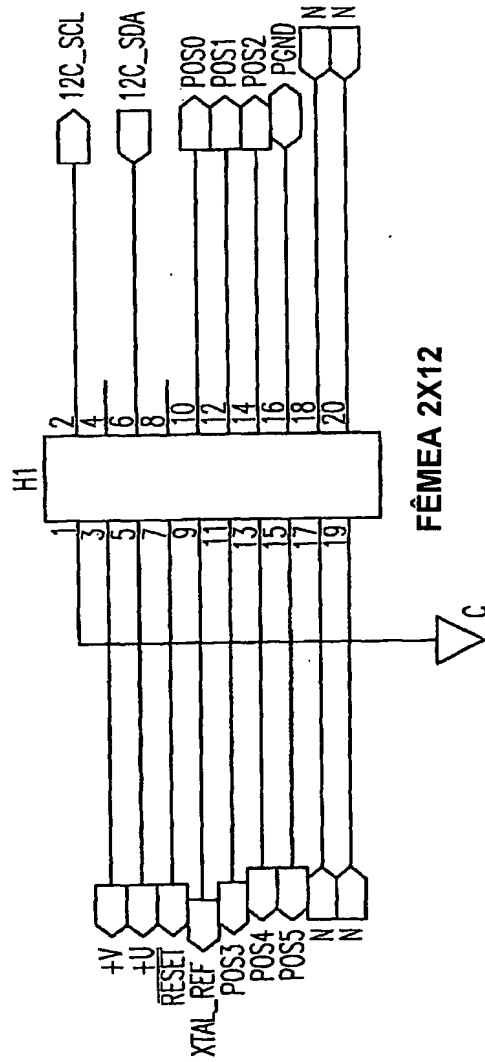


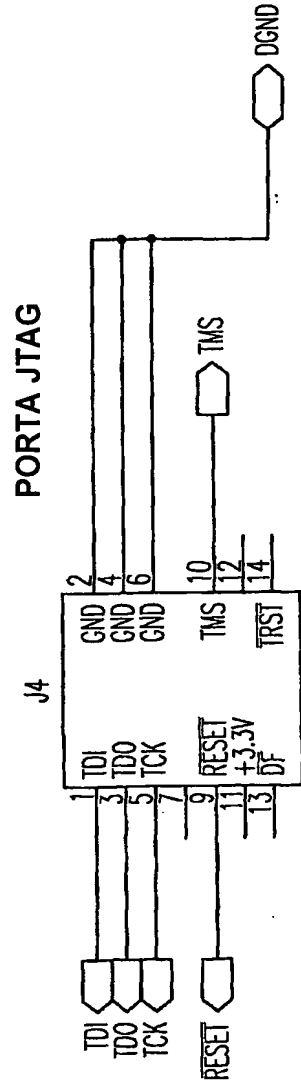
FIG.26C-4

CONECTOR FÊMEA DE BORDA DE 20 PINOS



FÊMEA 2X12

COLETOR DE TESTE JTAG



PORTA JTAG

MTG1
1
4-40 HDWR

MTG2
1
4-40 HDWR

MACHO 2X7

CONECTOR DE BORDA PCB203

FIG.26D

MEDIDOR DE LÁMINA EG 24 PCB 203

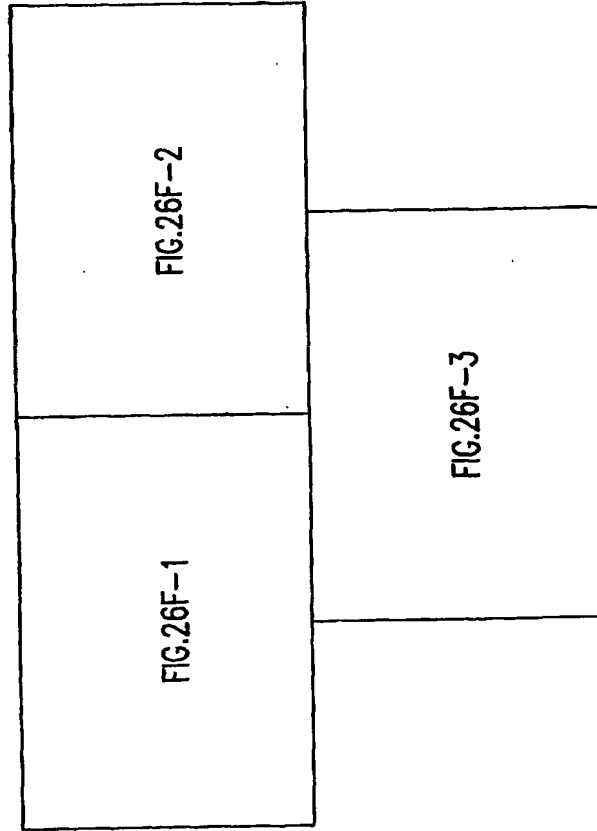


FIG.26F

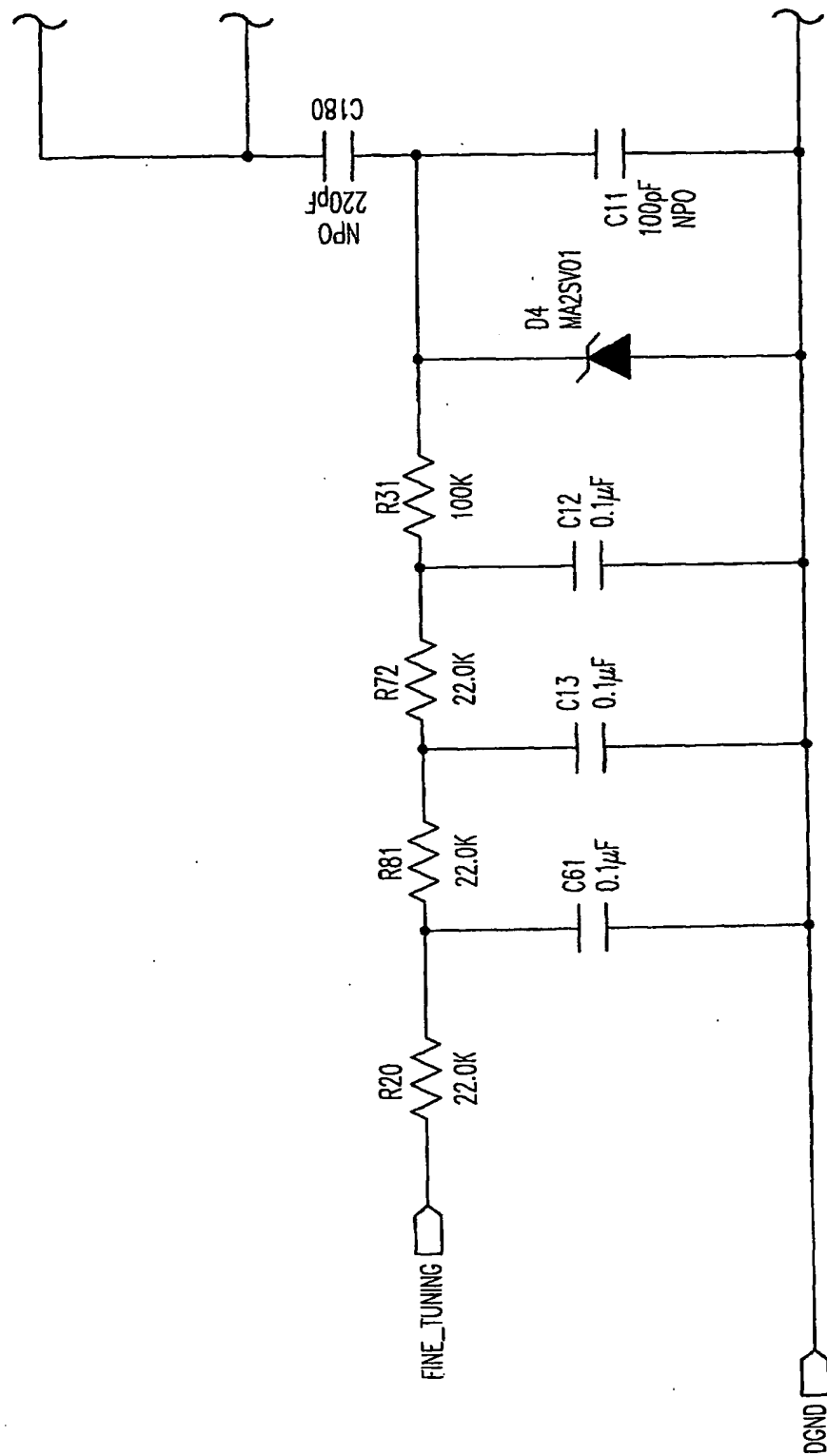


FIG. 26F-1

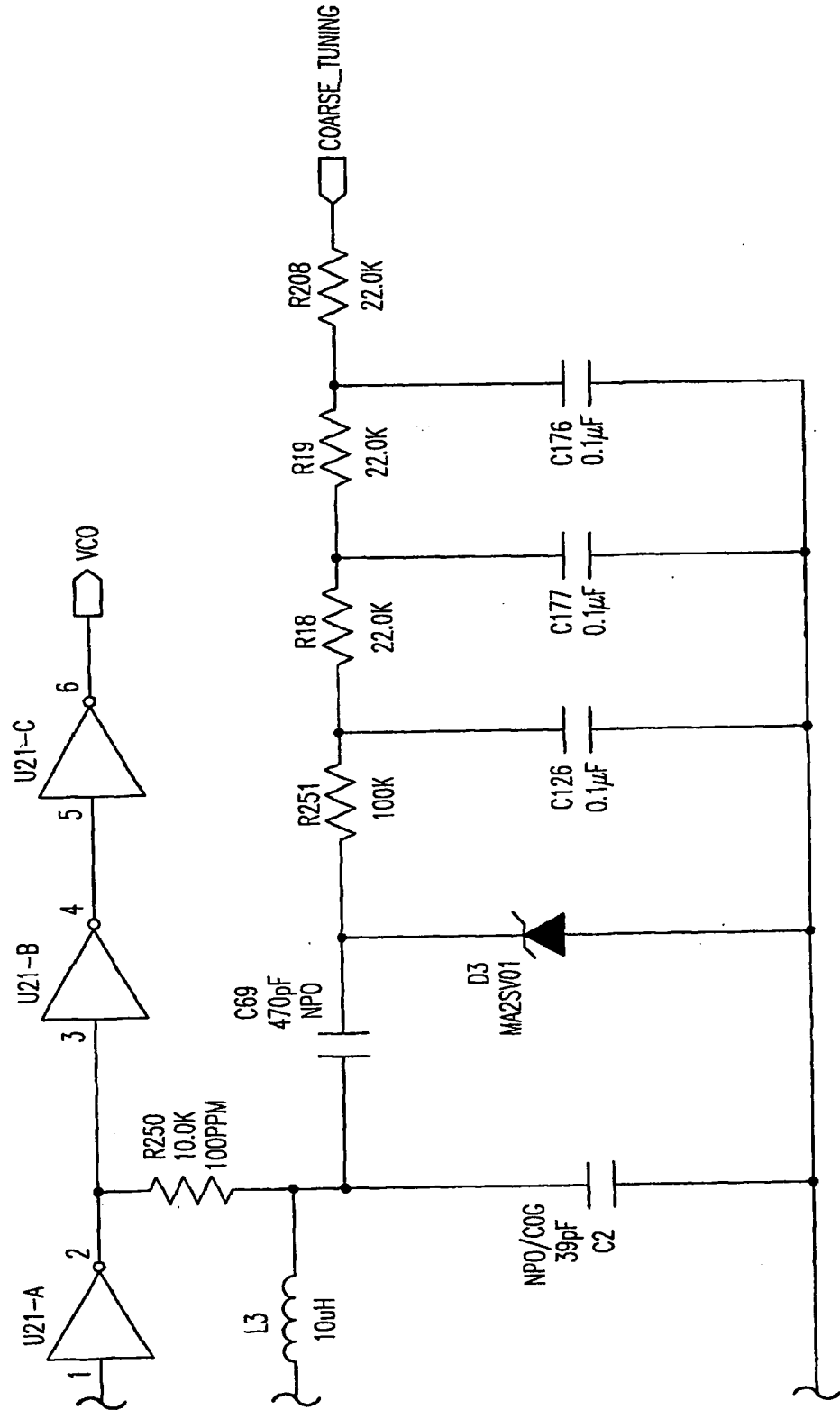


FIG. 26F-2

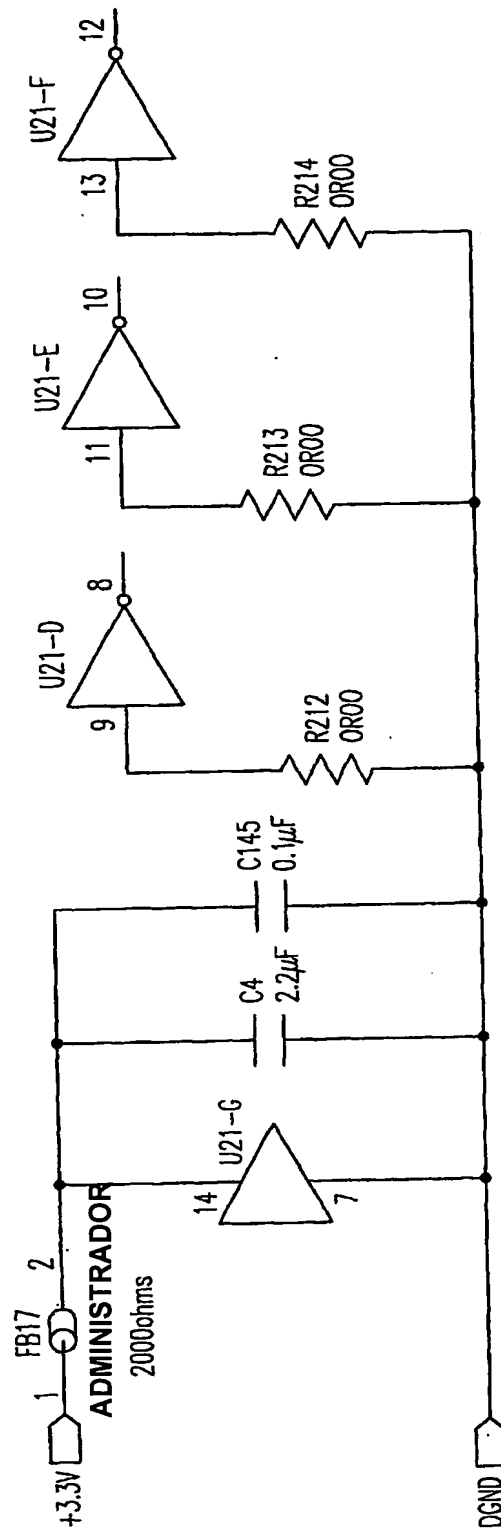


FIG.26F-3

MEDIDOR DE LÁMINA EG 24 PCB 203

FIG.26G-1	FIG.26G-2
FIG.26G-3	FIG.26G-4

FIG.26G

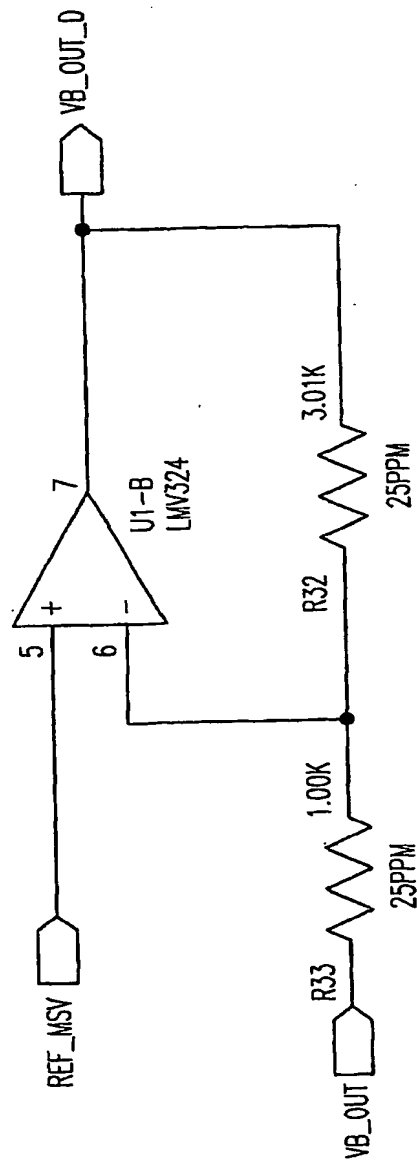
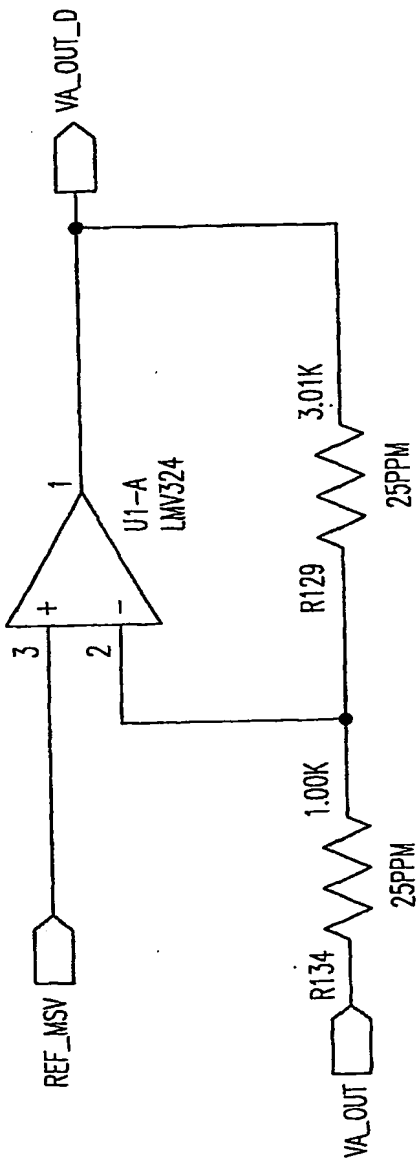


FIG.26G-1

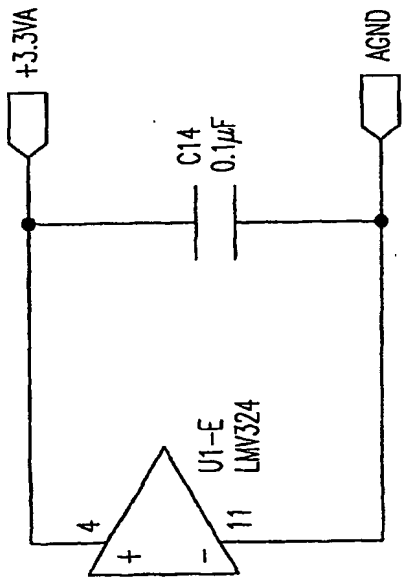


FIG. 26G-2

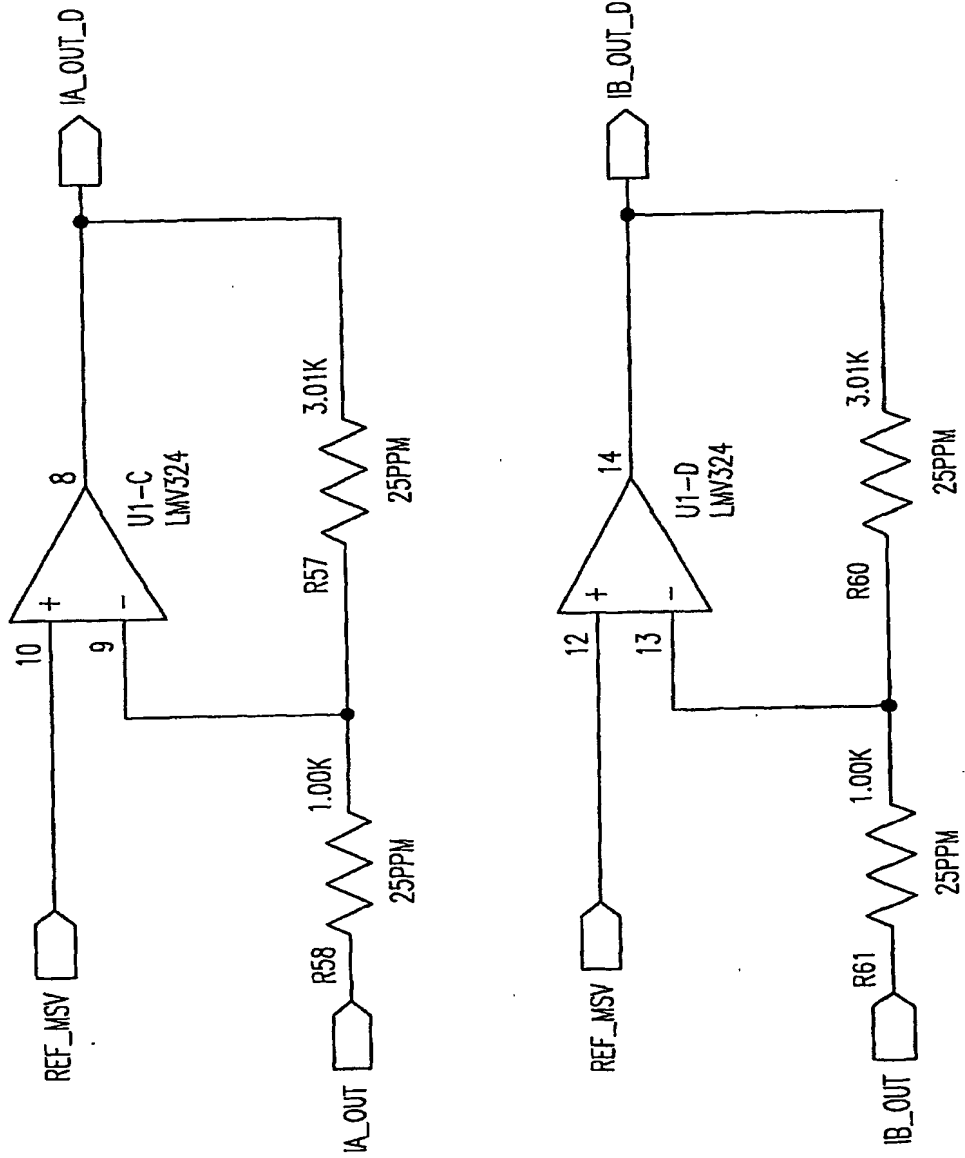


FIG.26G-3

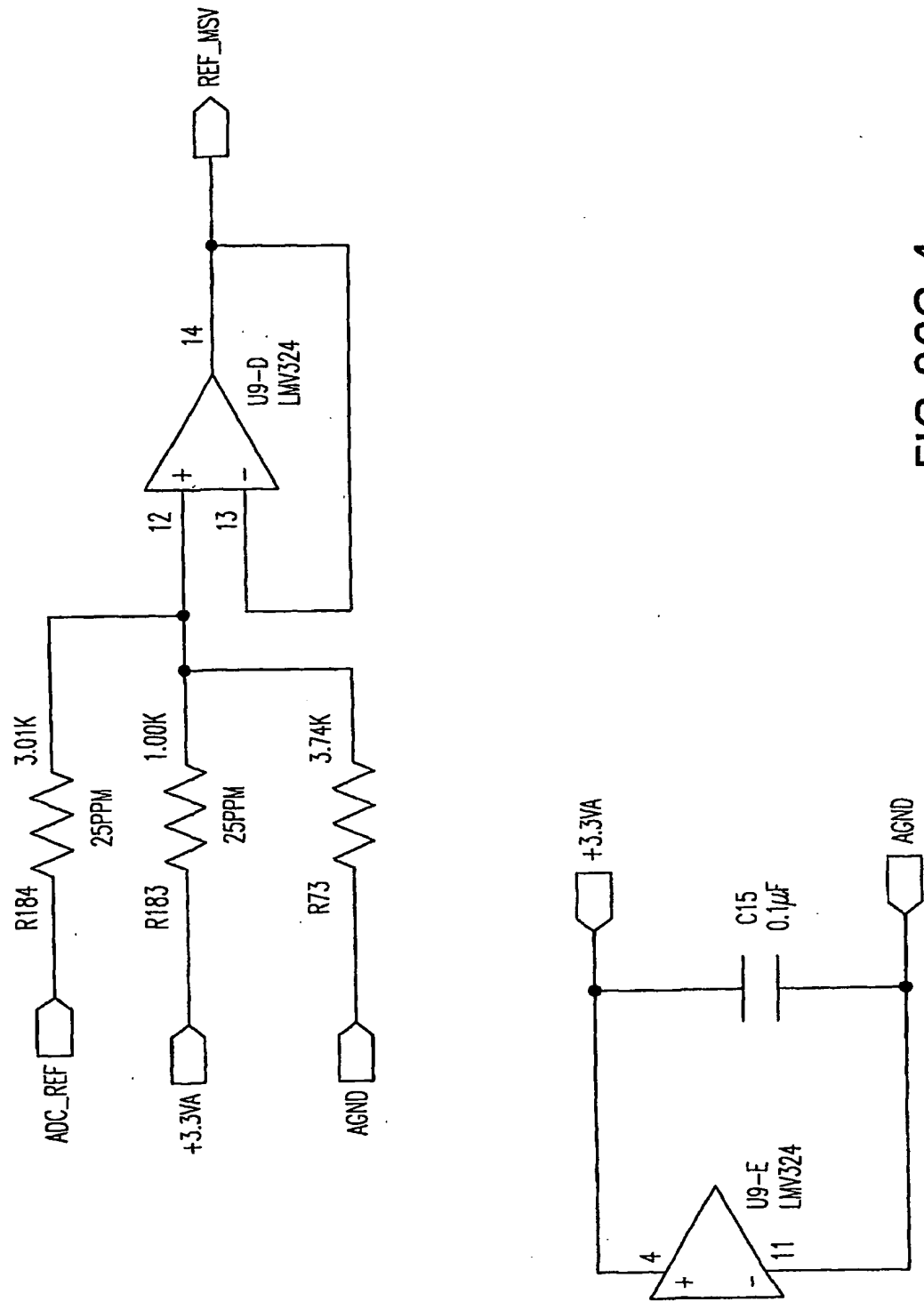


FIG. 26G-4

MEDIDOR DE LÂMINA EG 24 PCB 203

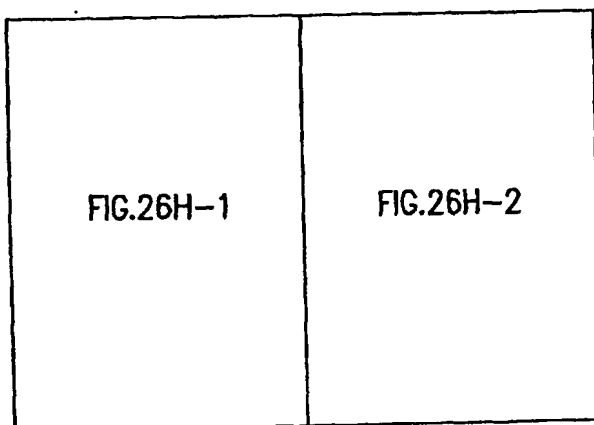


FIG.26H

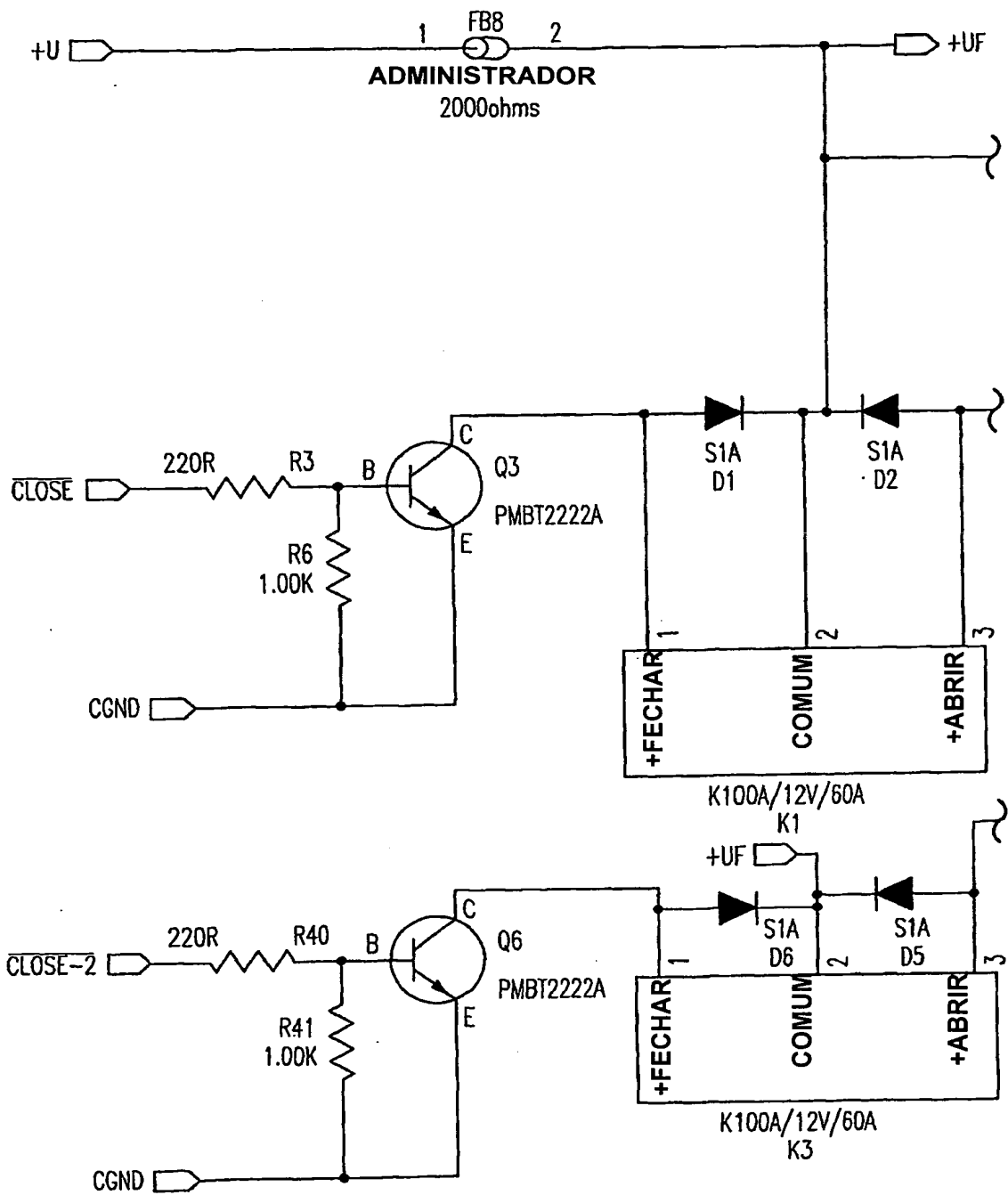


FIG.26H-1

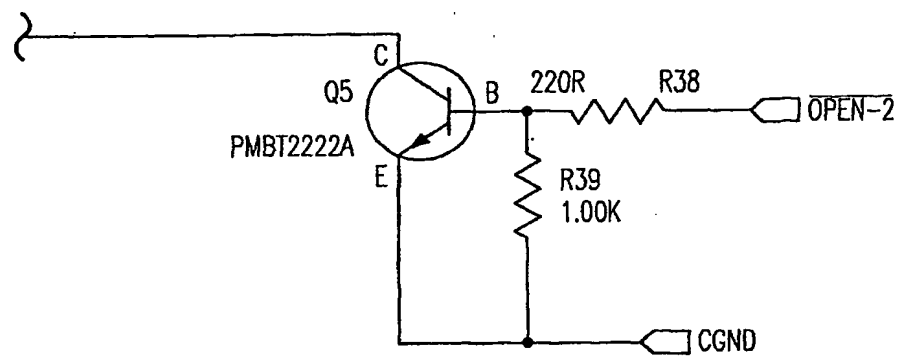
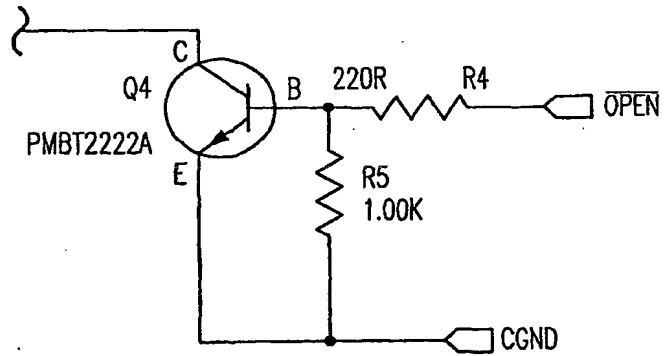
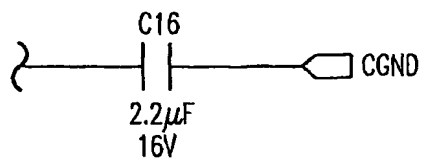


FIG.26H-2

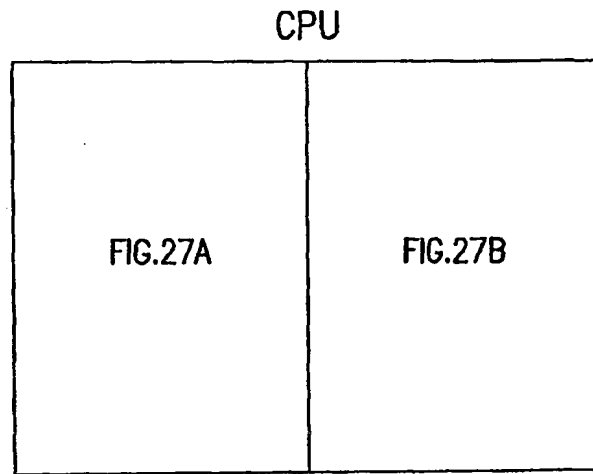


FIG.27

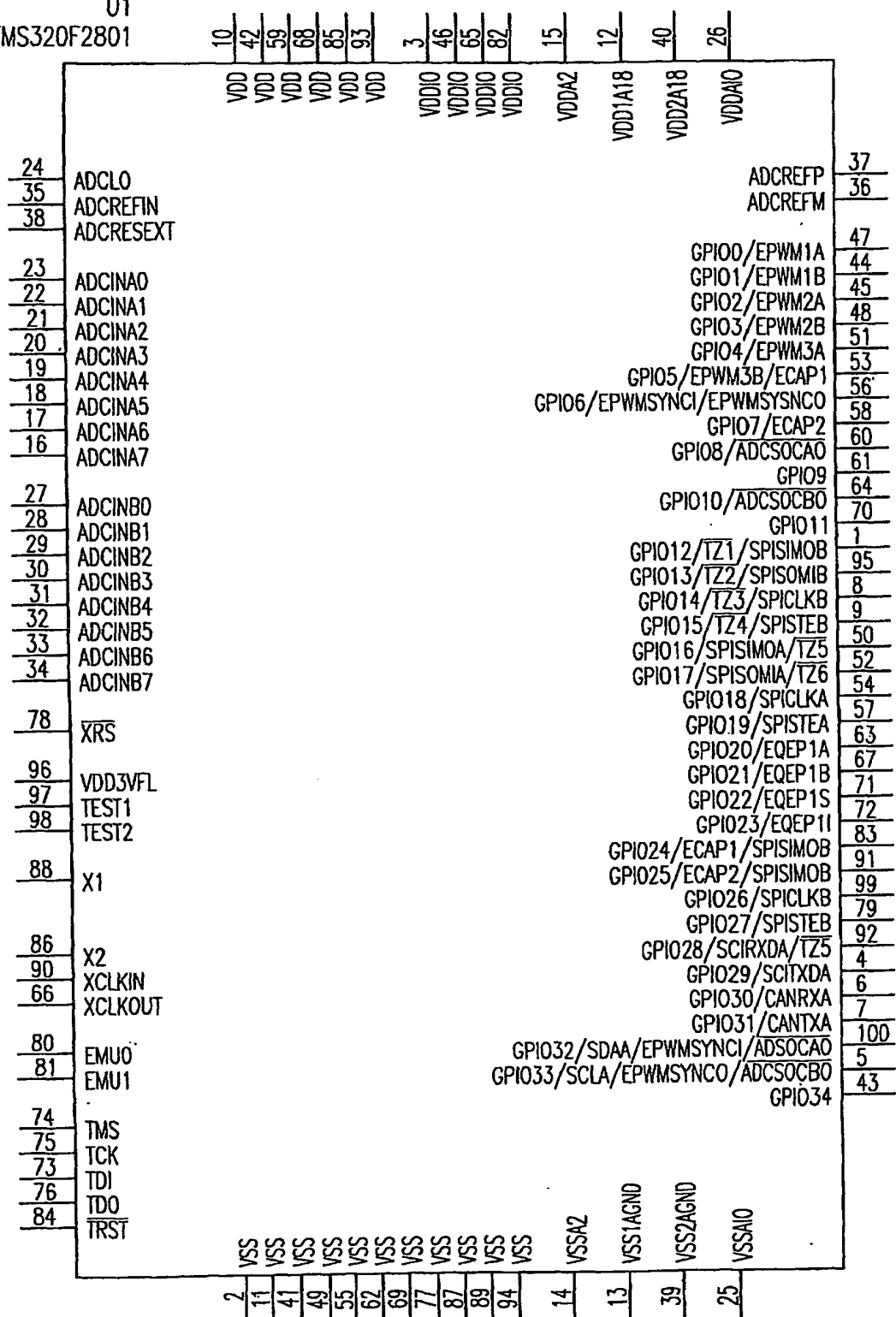


FIG.27A

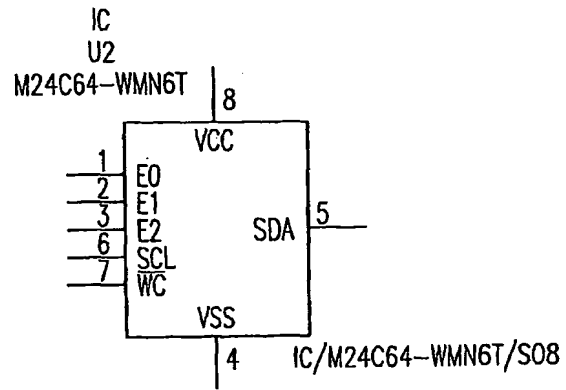


FIG.27B

HDRS PARA LÂMINAS PCB234

FIG.28B-1	FIG.28B-2
FIG.28B-3	FIG.28B-4
FIG.28B-5	FIG.28B-6

FIG.28B

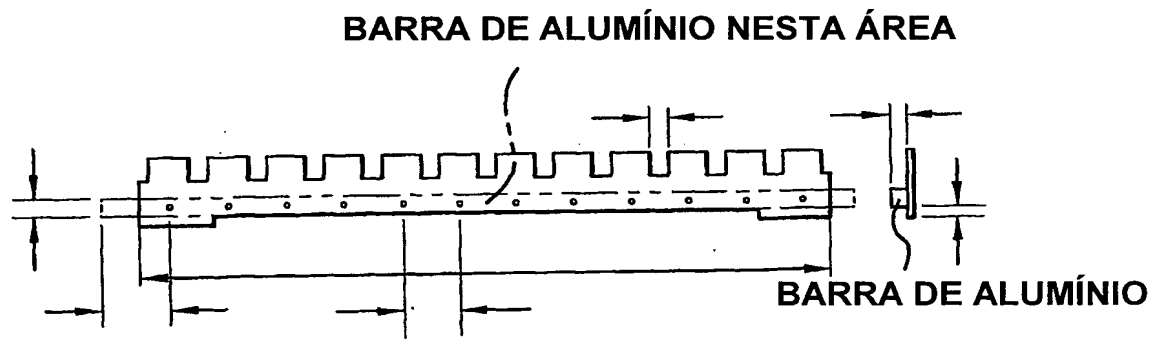


FIG.28A

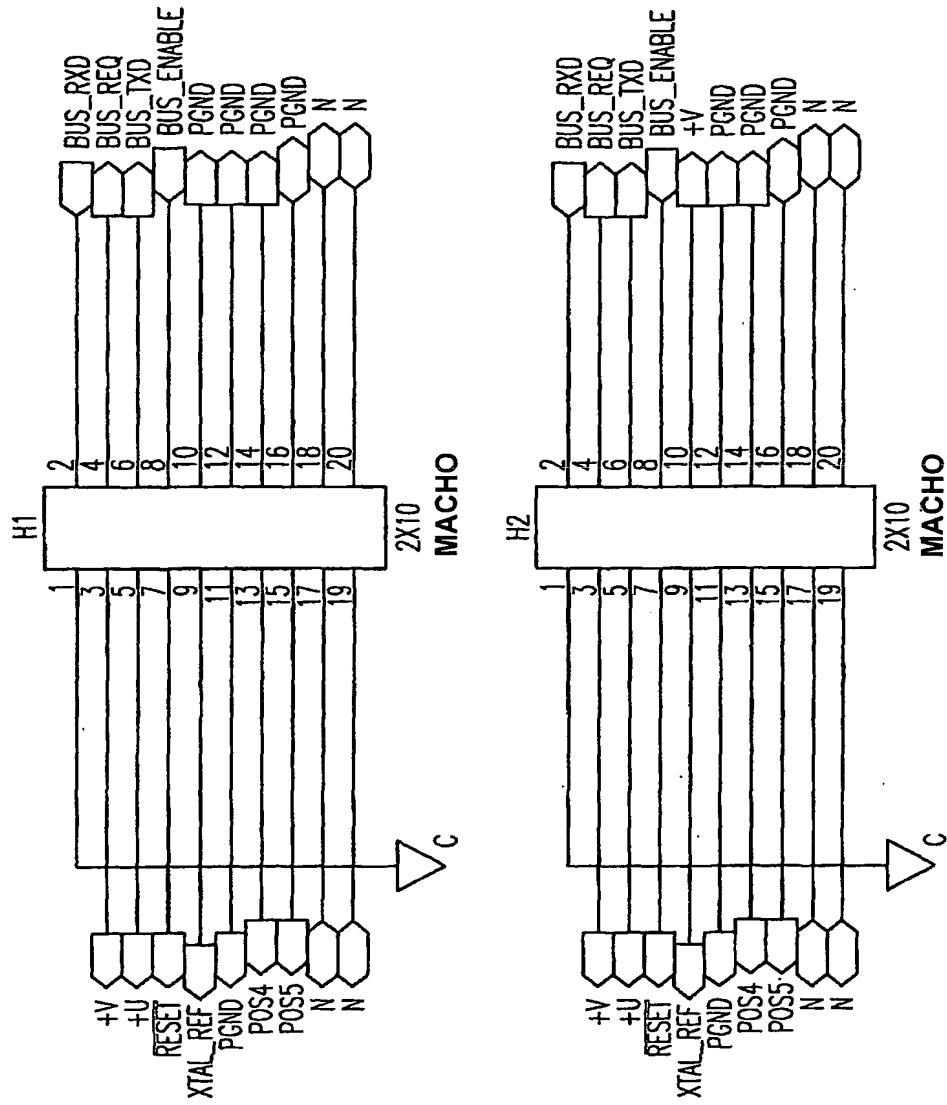


FIG.28B-1

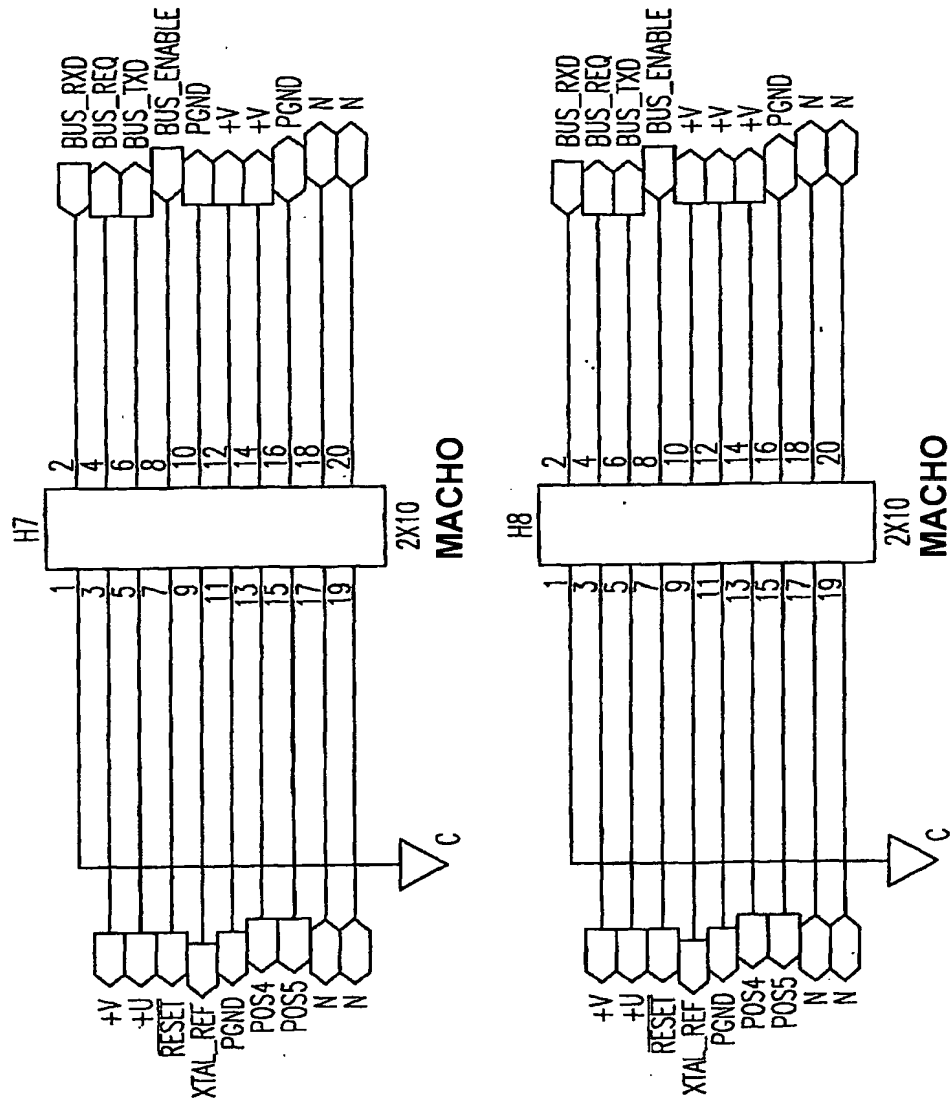


FIG.28B-2

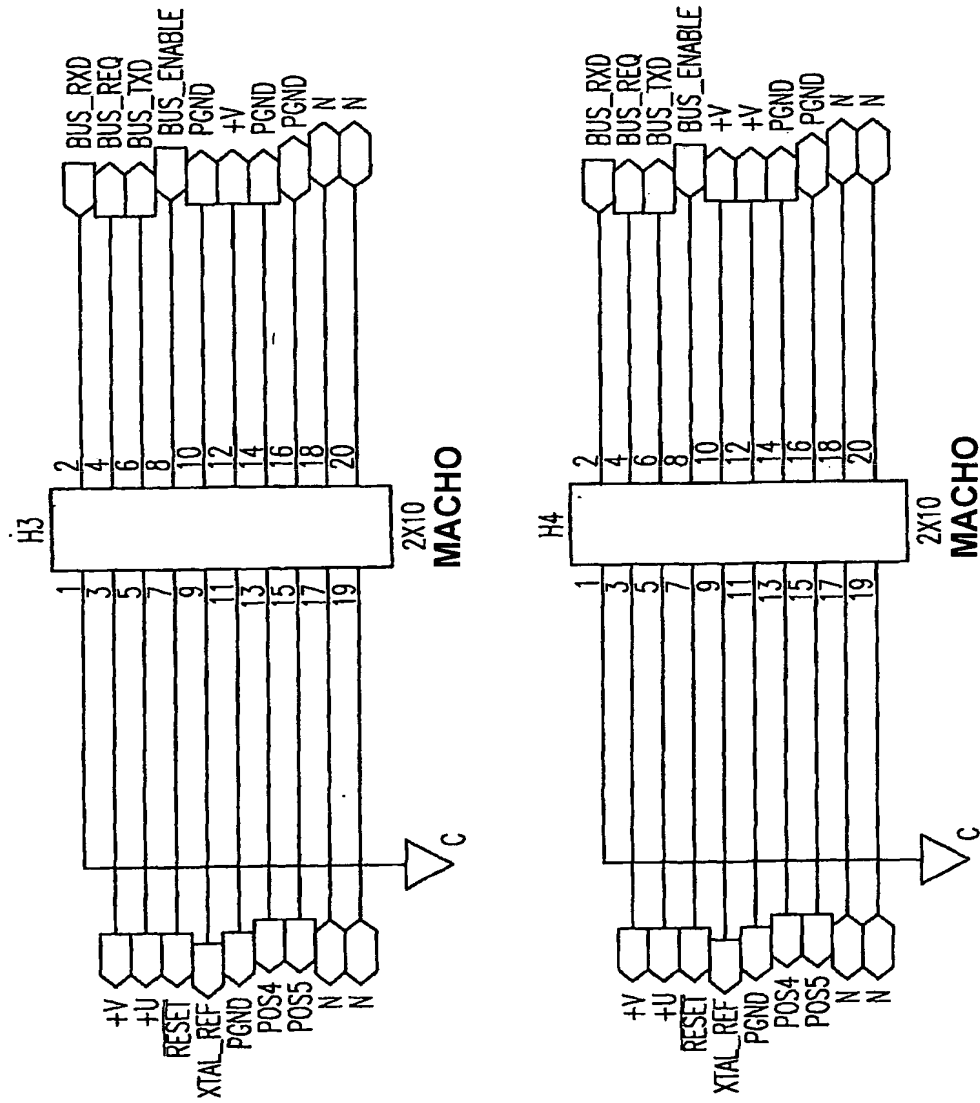


FIG.28B-3

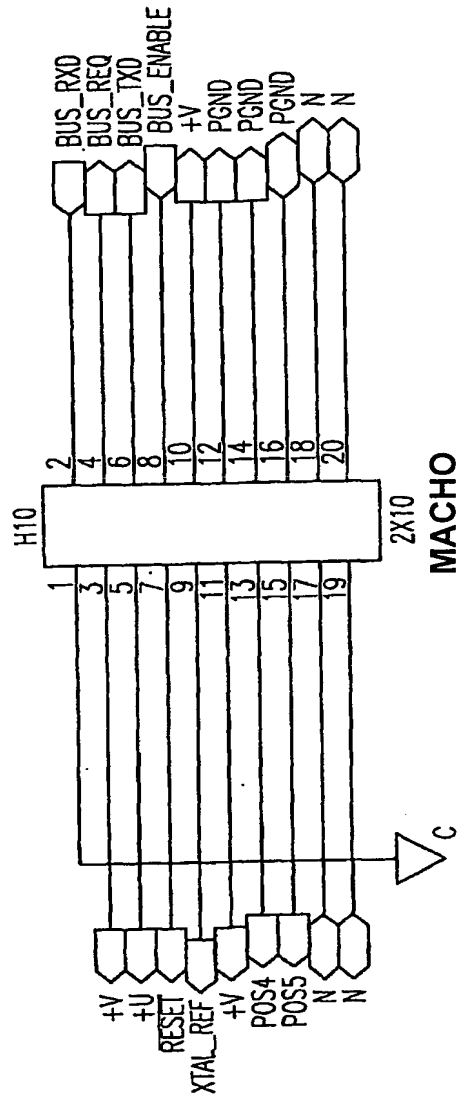
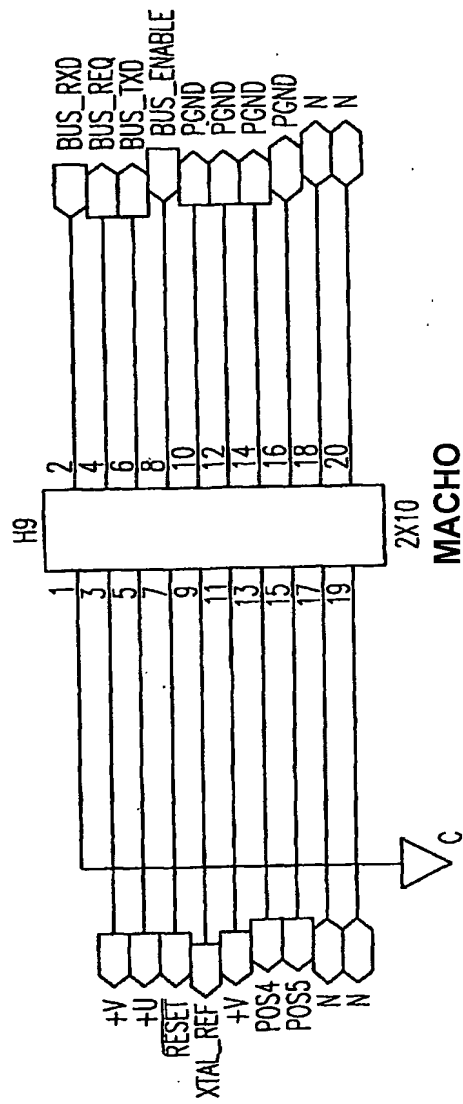


FIG.28B-4

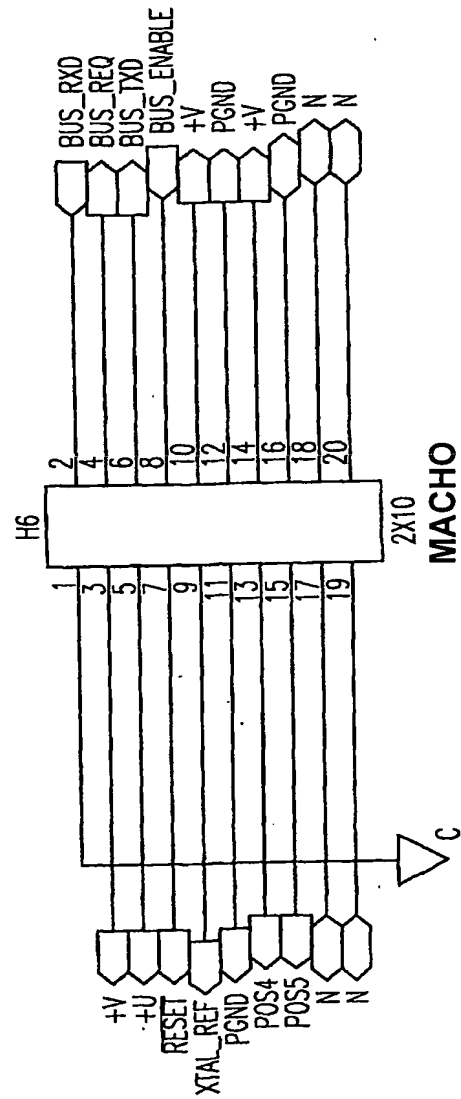
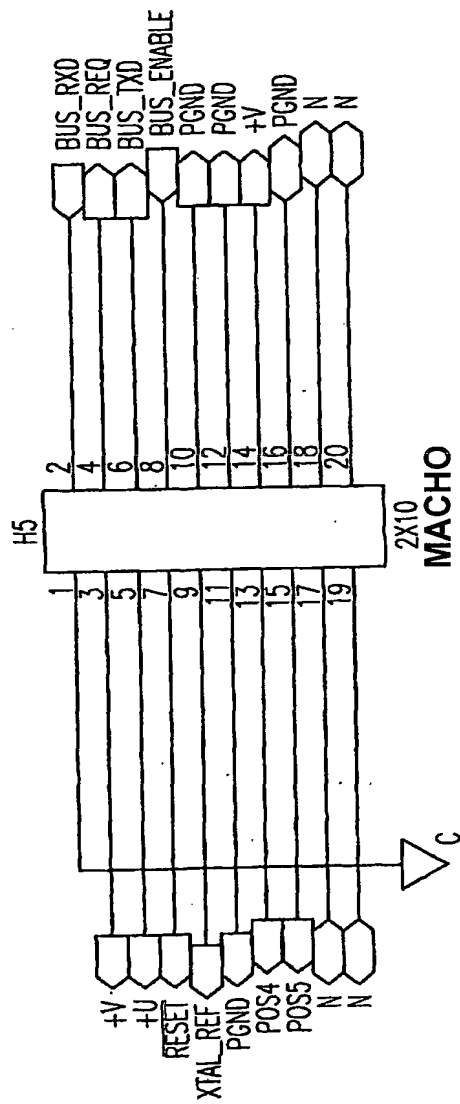


FIG.28B-5

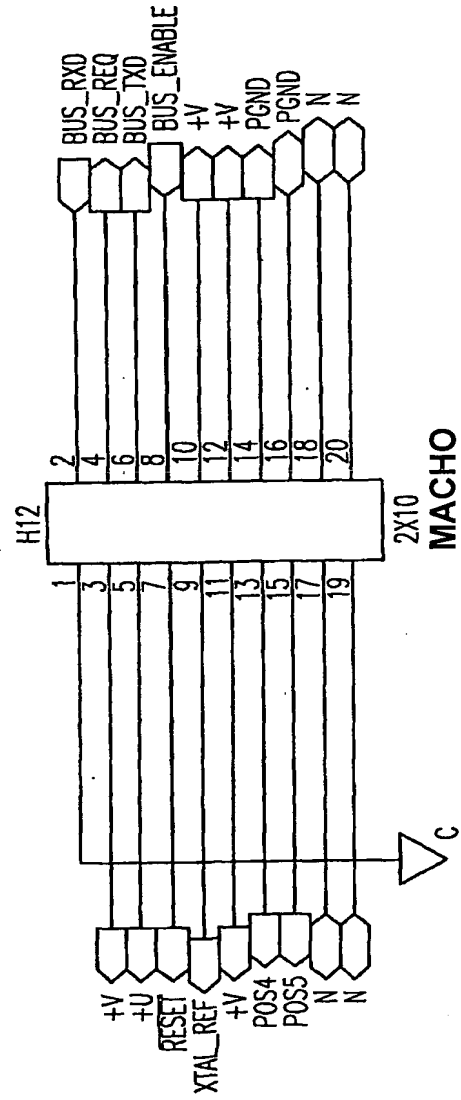
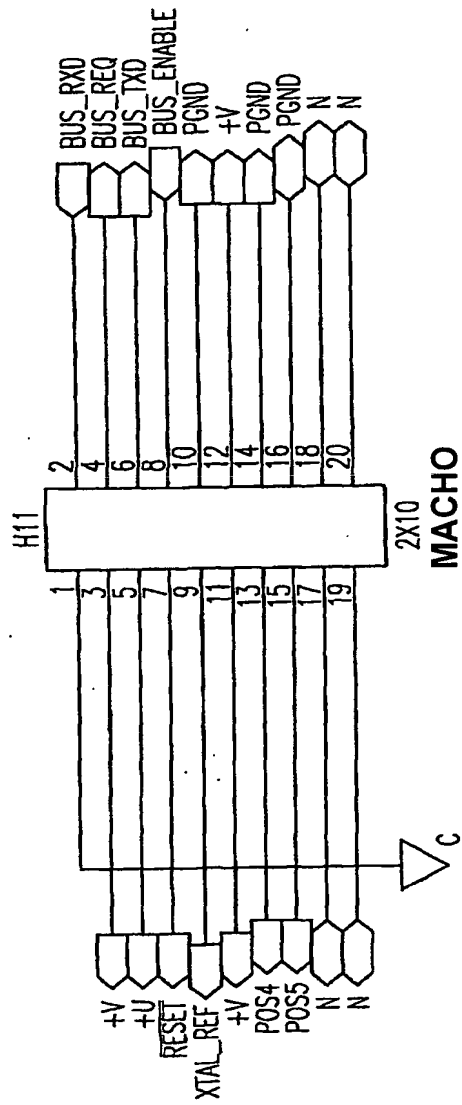


FIG.28B-6

COLETORES PCB234

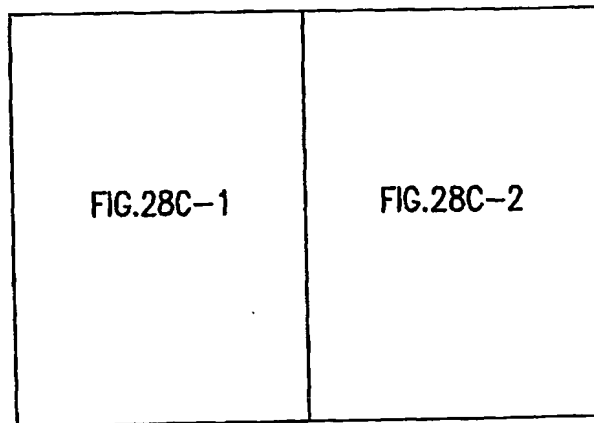
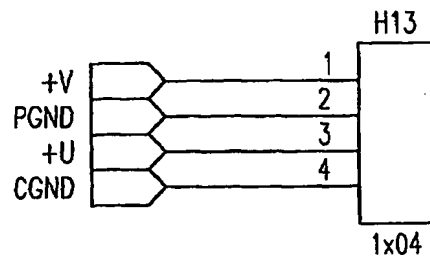
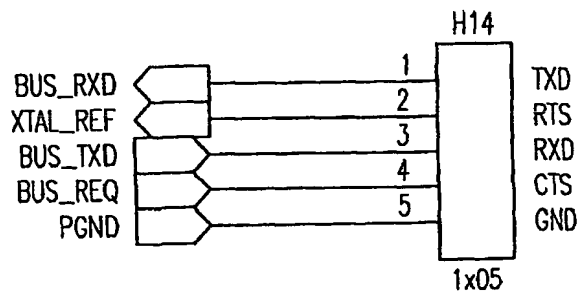
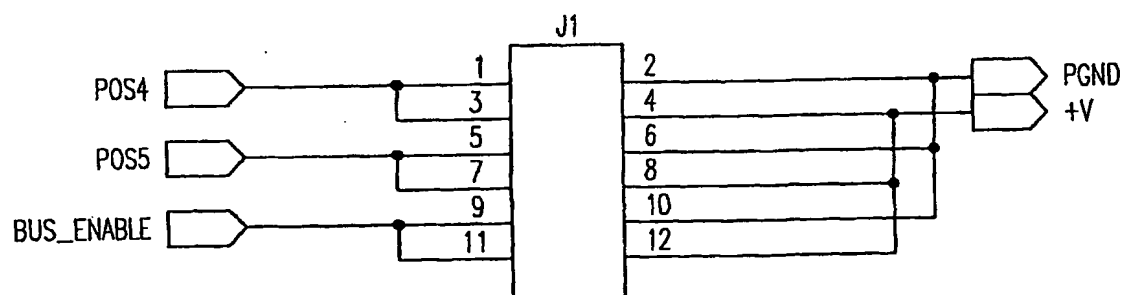
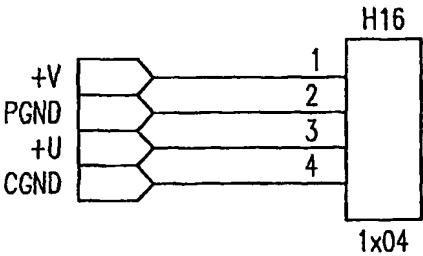


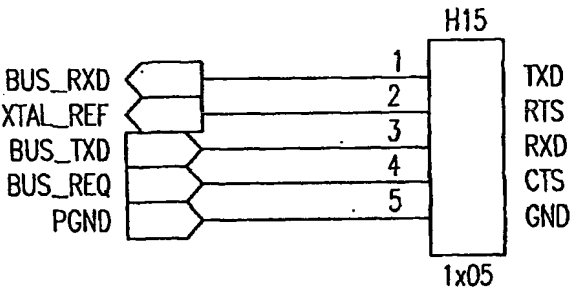
FIG.28C

COLETOR DE POTÊNCIA**COLETOR RS232****PONTES****MACHO 2X6 VERTICAL****FIG.28C-1**

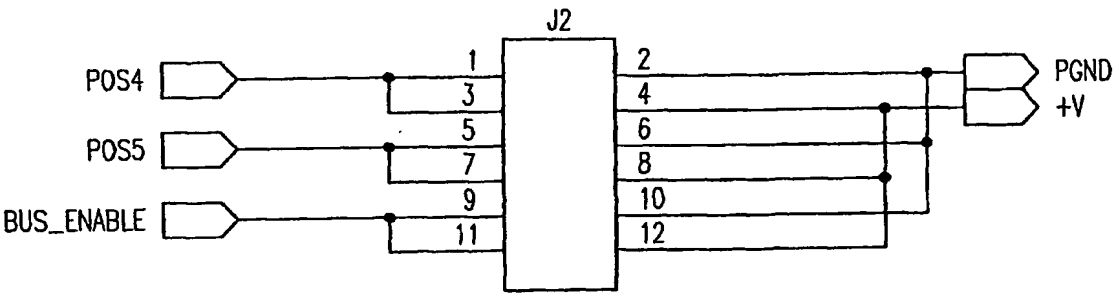
COLETOR DE POTÊNCIA



COLETOR RS232



PONTES



MACHO 2X6 VERTICAL

FIG.28C-2

UNIDADE DE FONTE DE ENERGIA SCH210-1

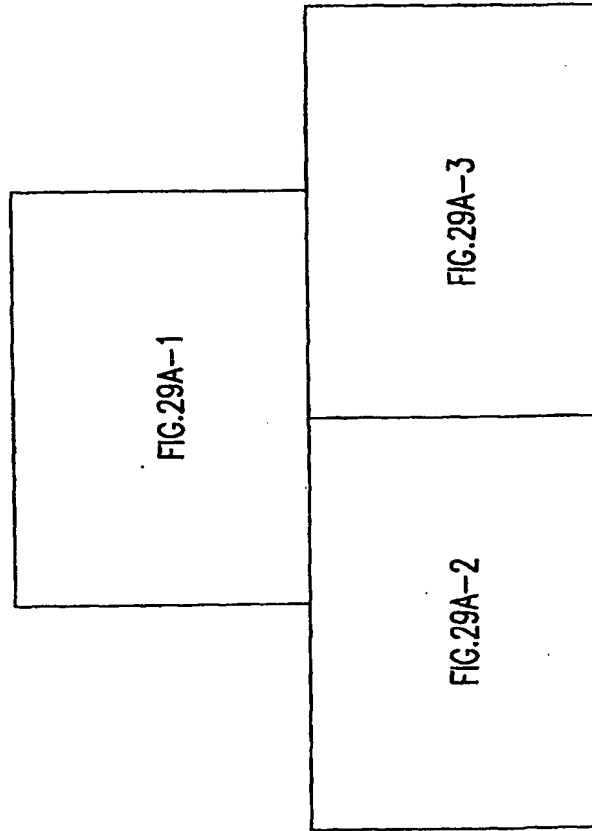


FIG.29A

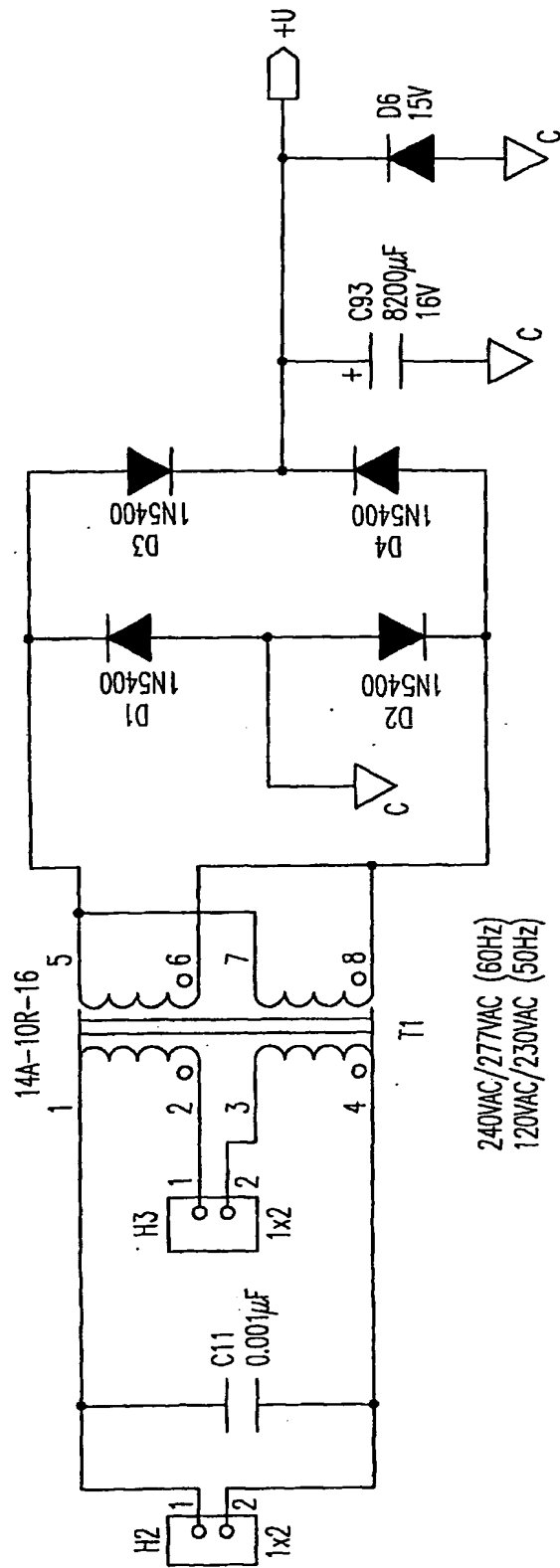


FIG. 29A-1

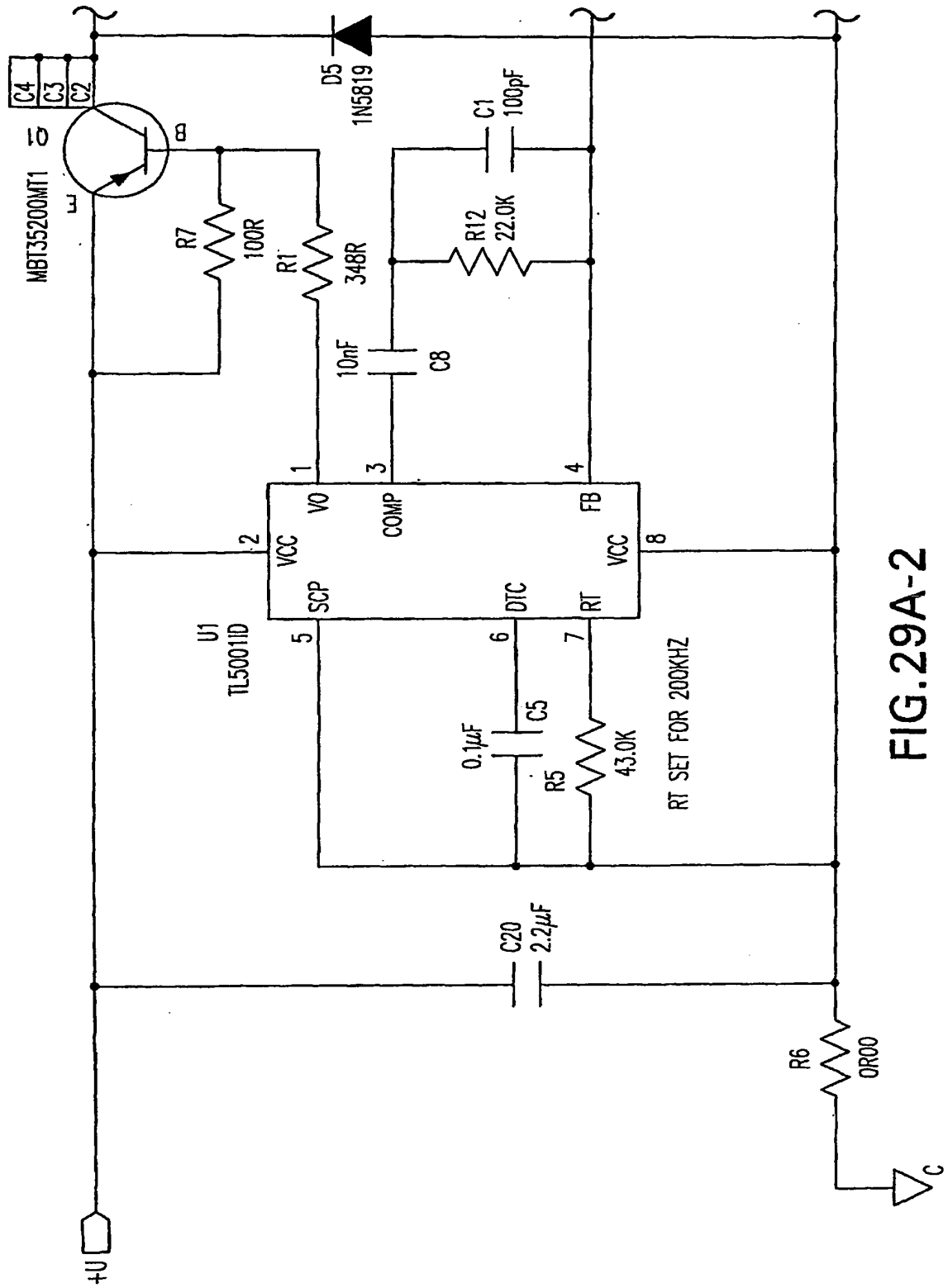


FIG.29A-2

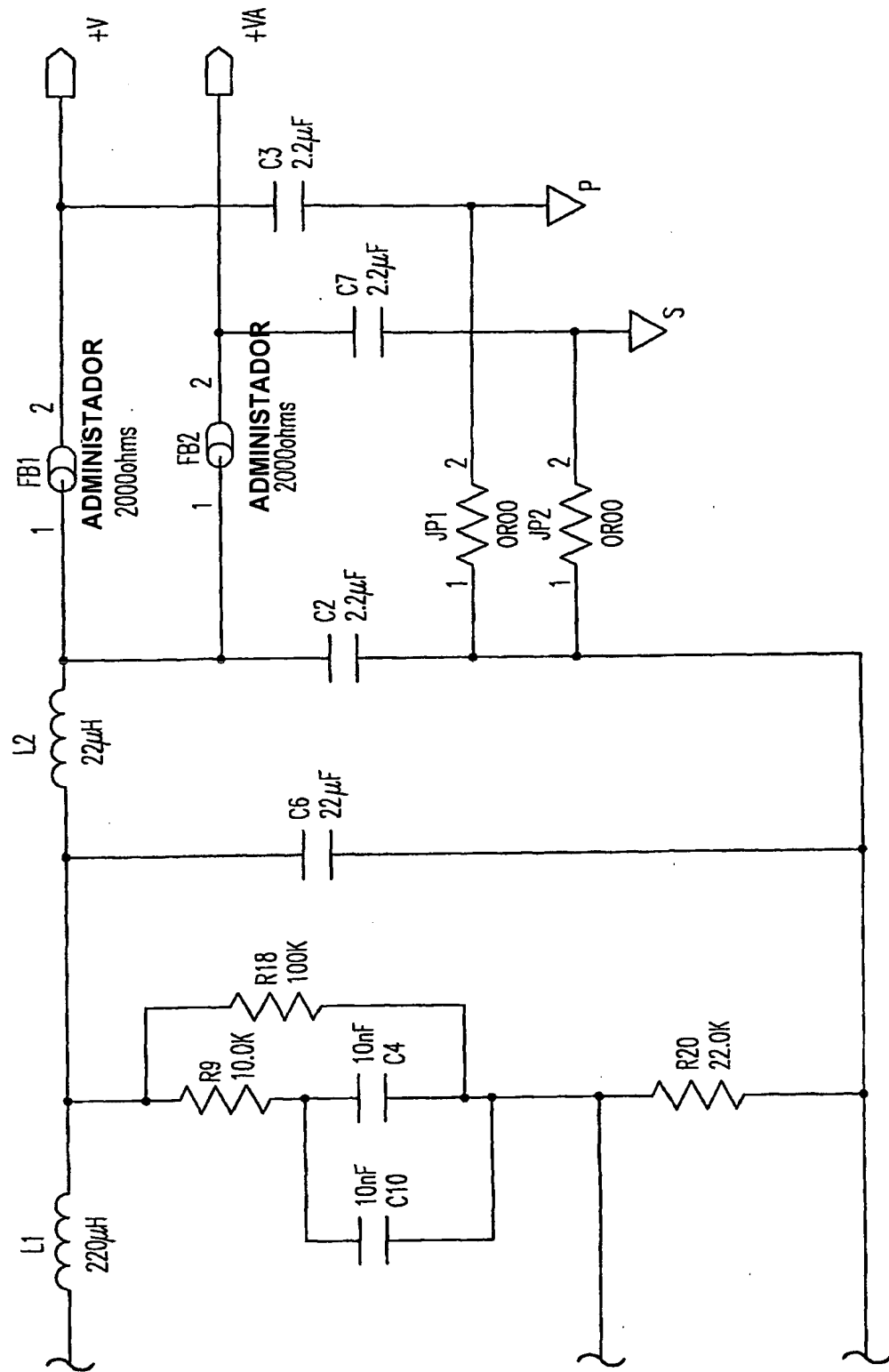


FIG. 29A-3

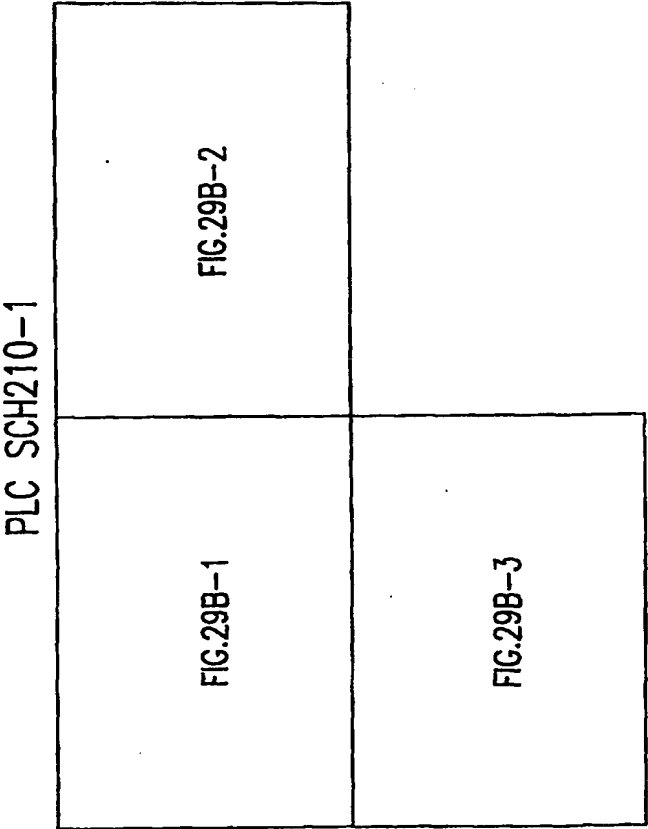


FIG.29B

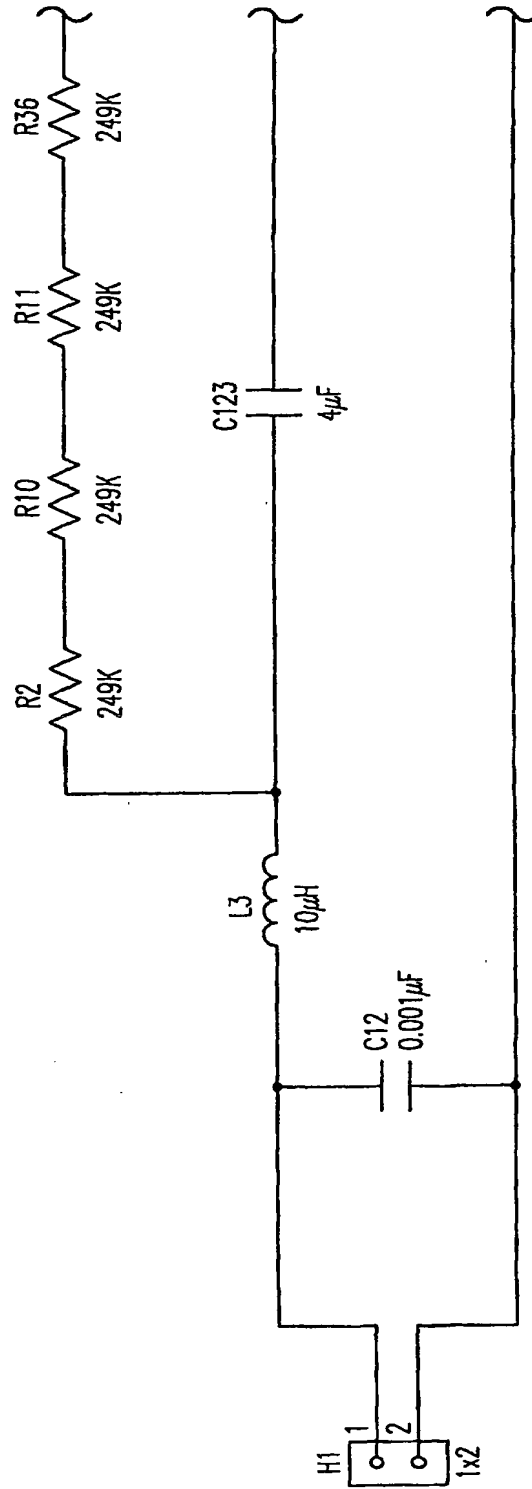


FIG. 29B-1

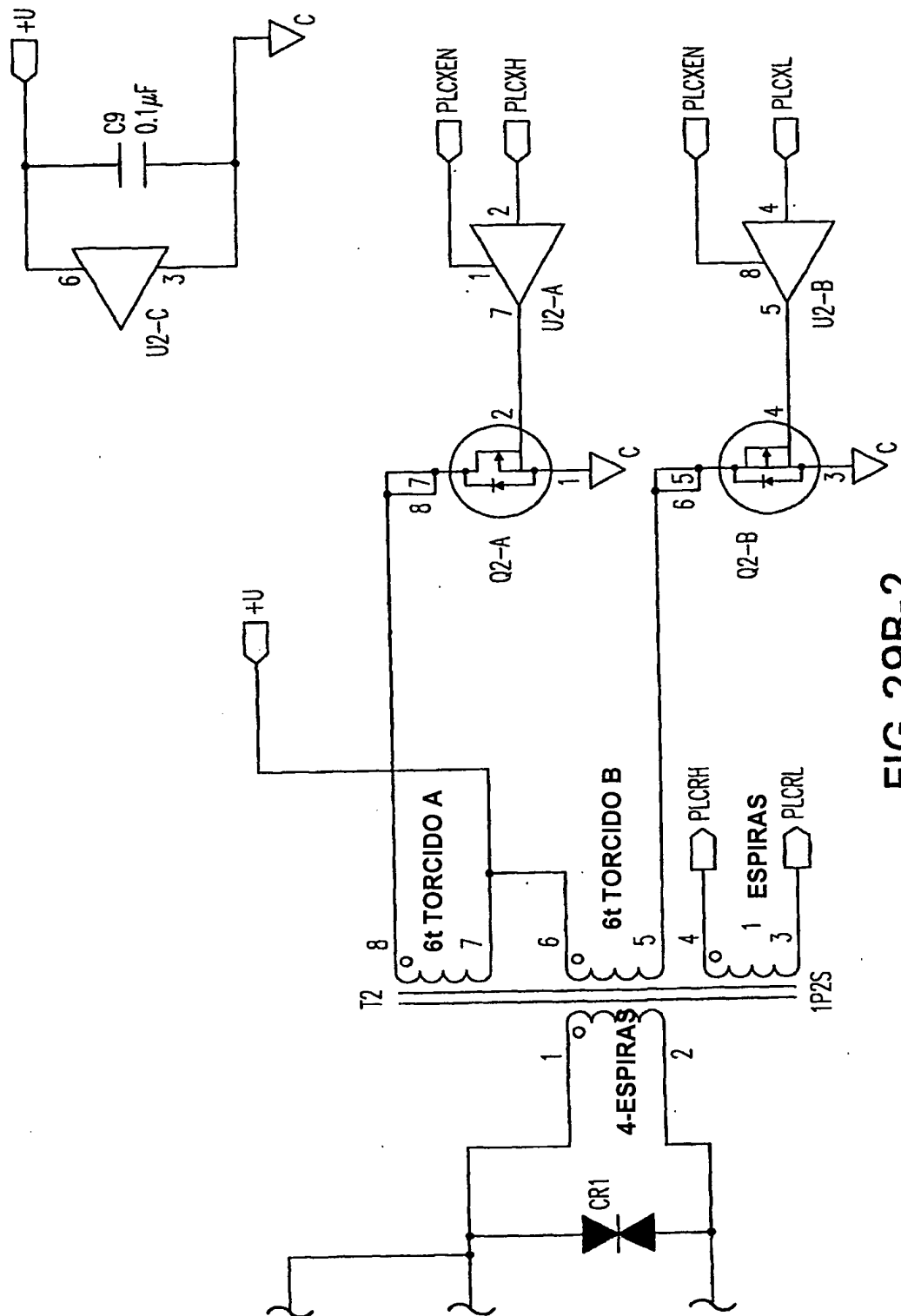


FIG.29B-2

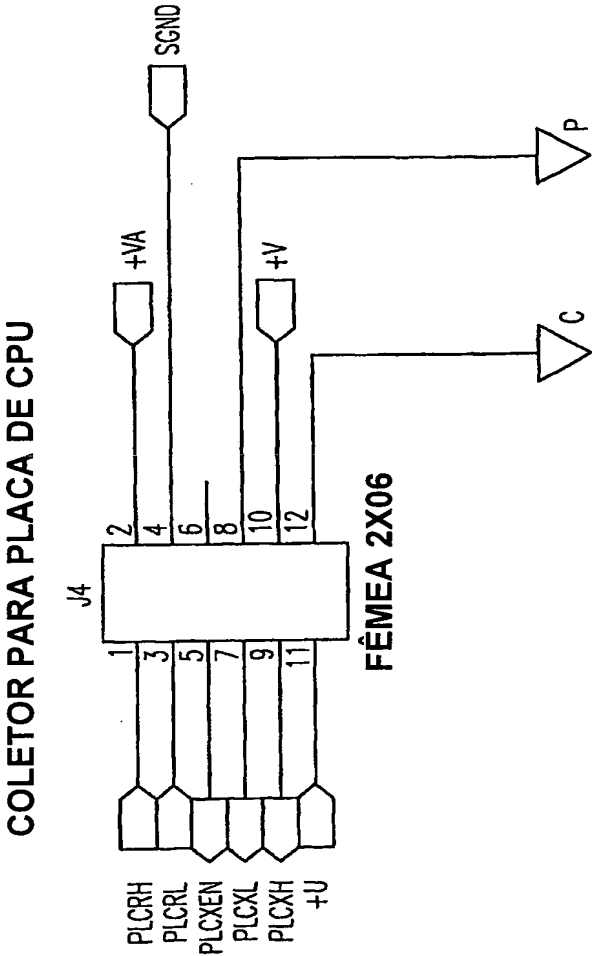


FIG.29B-3

FONTE DE ENERGIA PCB230

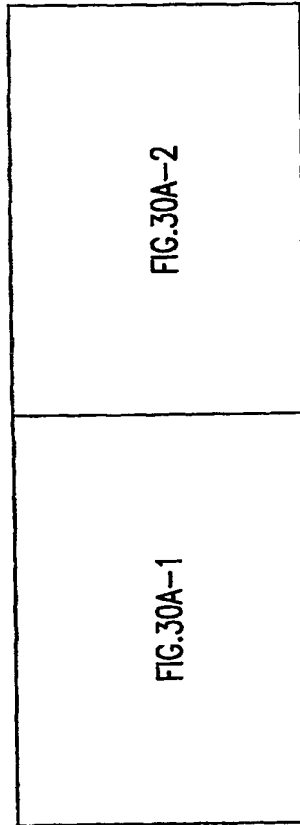


FIG.30A

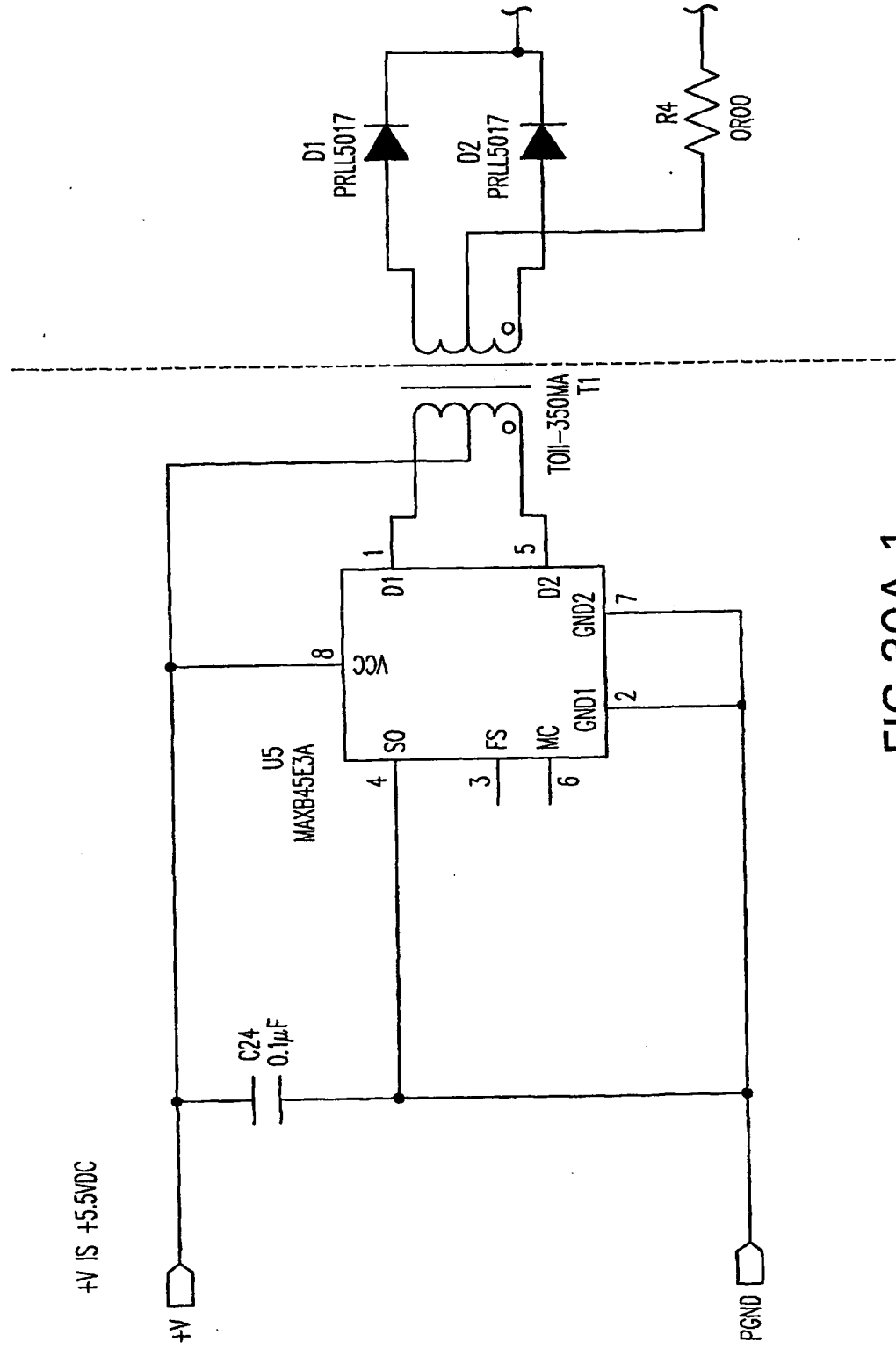


FIG.30A-1

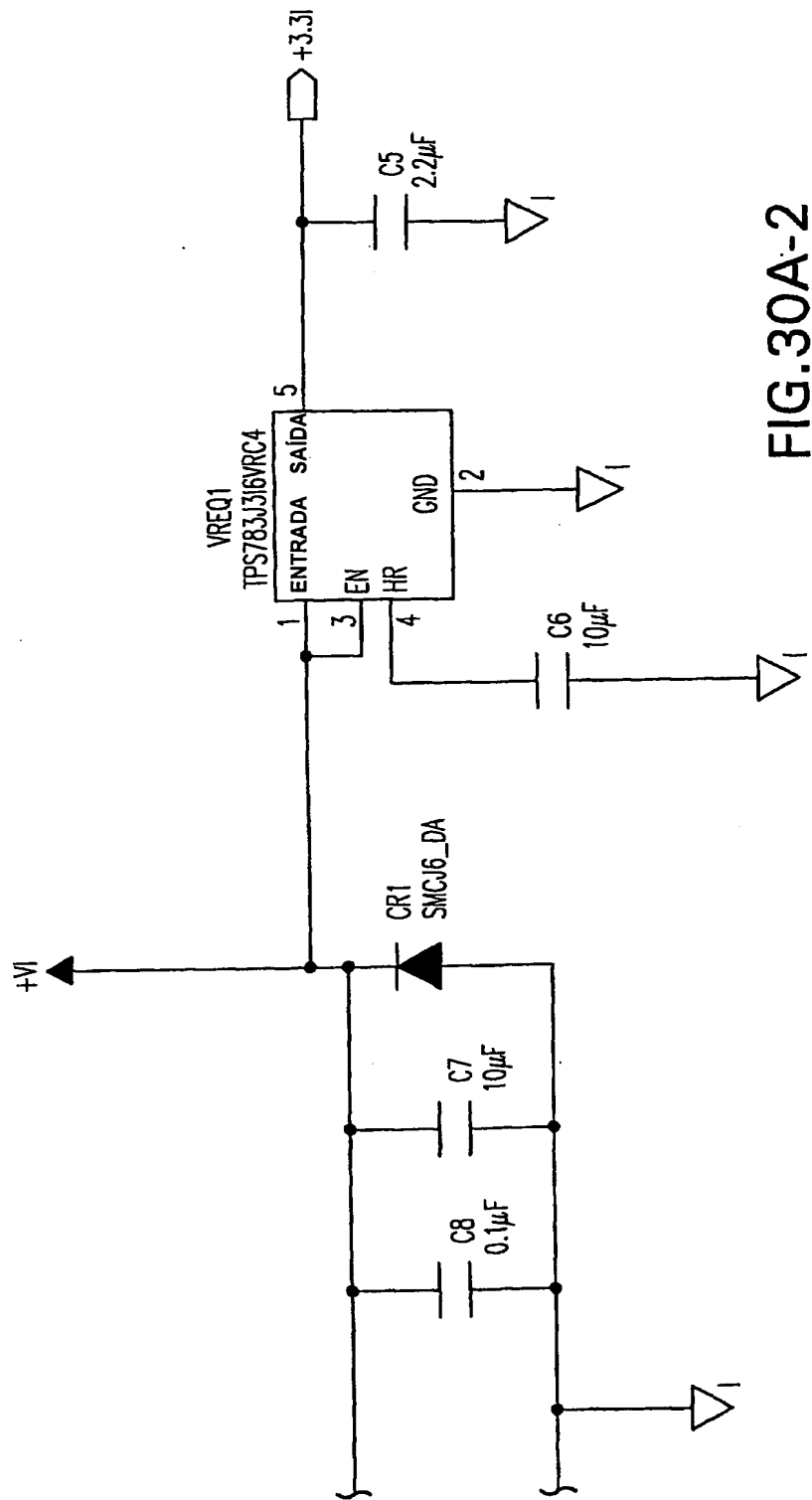
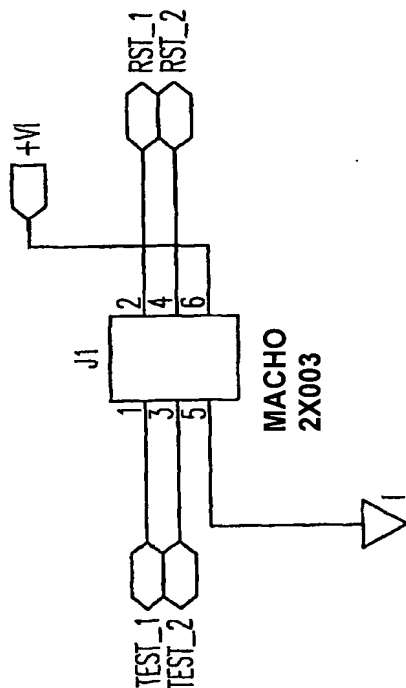
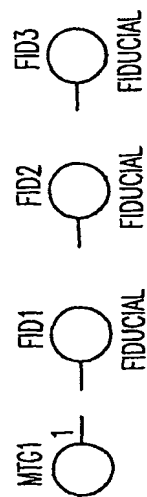
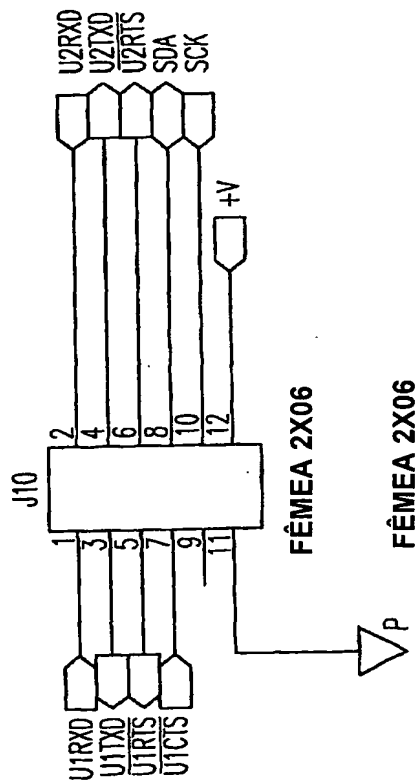


FIG.30A-2

COLETOR PARA PROGRAMAR CHIPS MSP430



COLETOR PARA PLACA DE CPU



PLACA DE IO SCH230-1

FIG.30B

PLACA DE PC DE I/O PCB230-1

FIG.30C-1	FIG.30C-2
FIG.30C-3	FIG.30C-4

FIG.30C

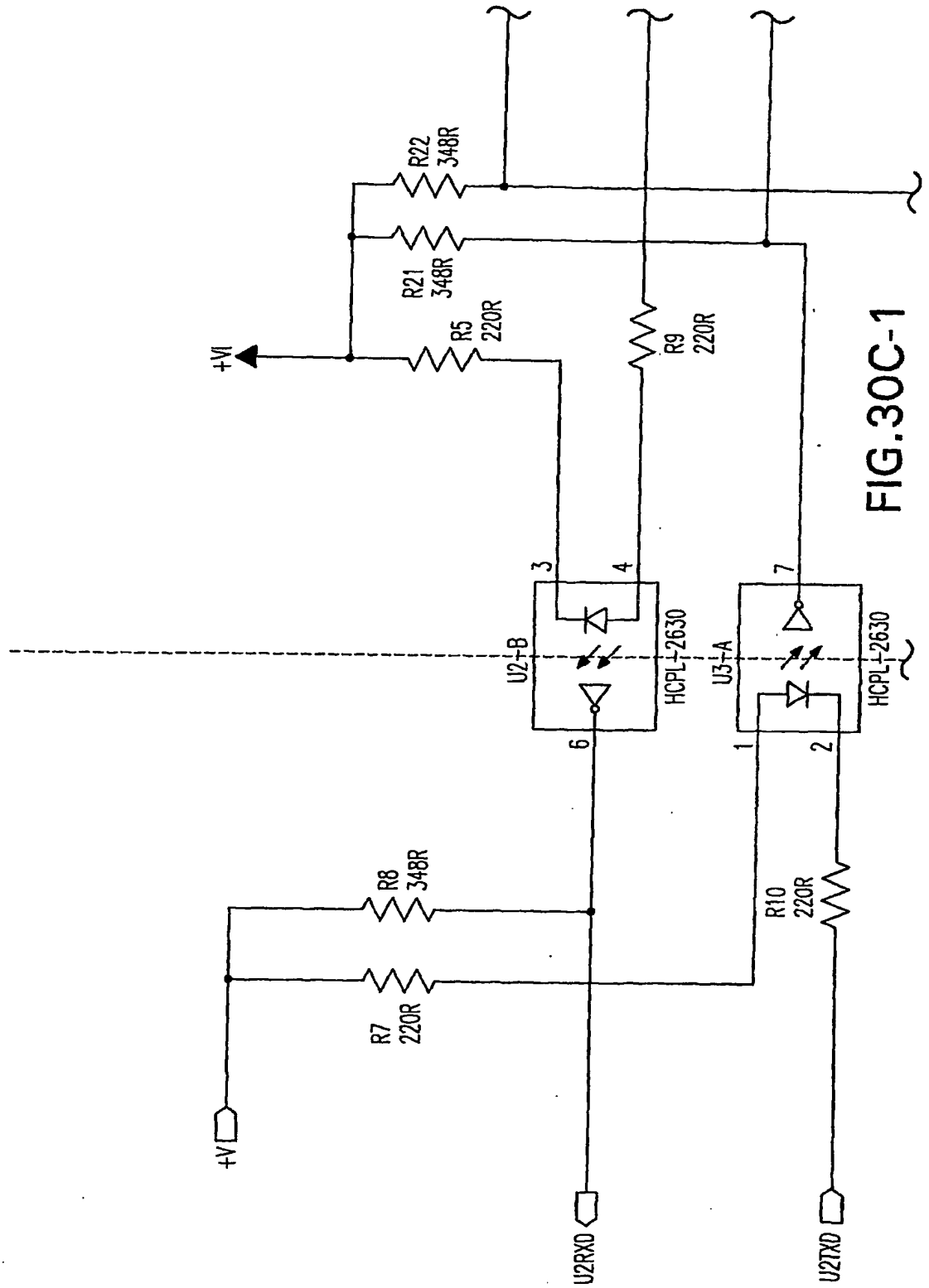
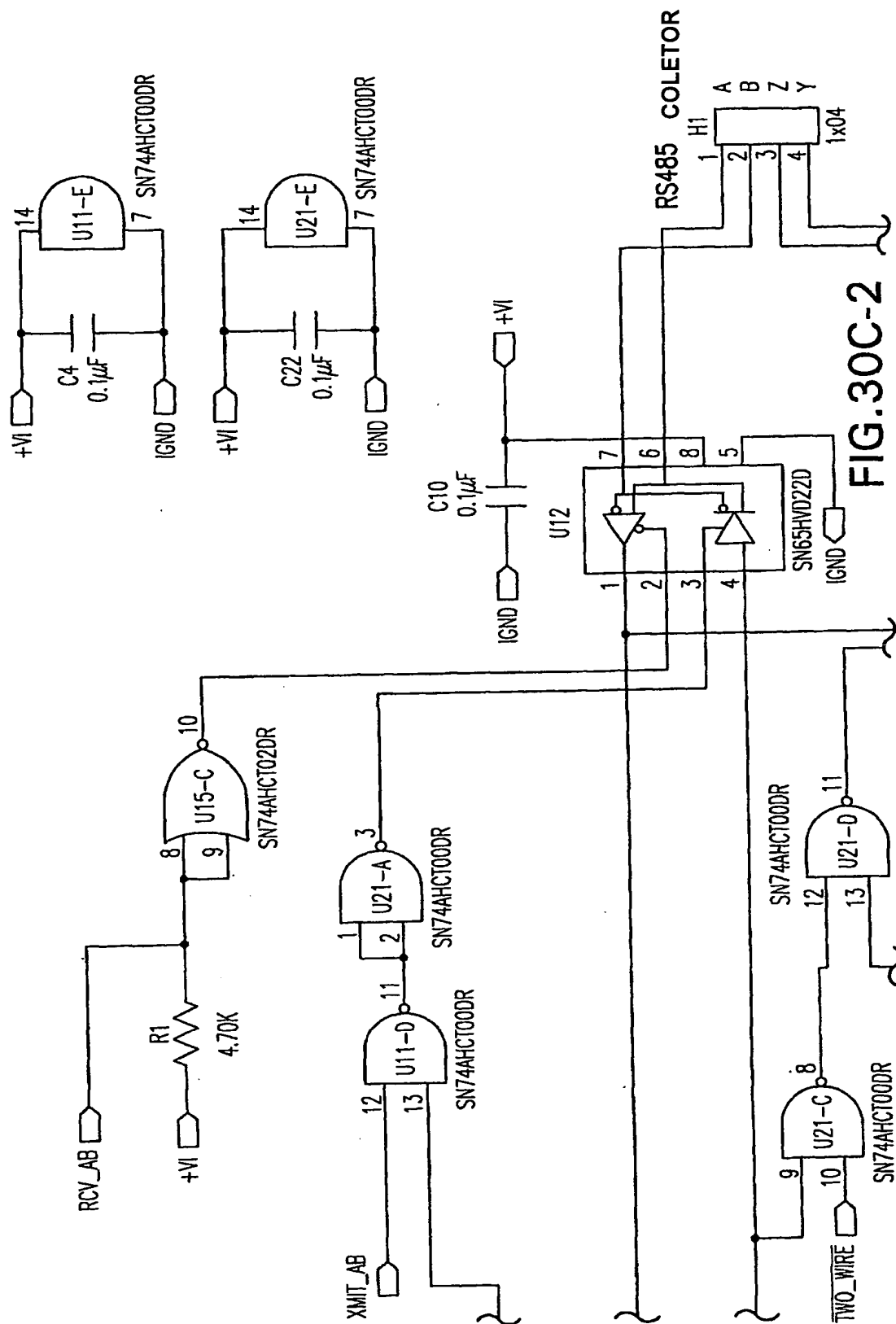


FIG.30C-1



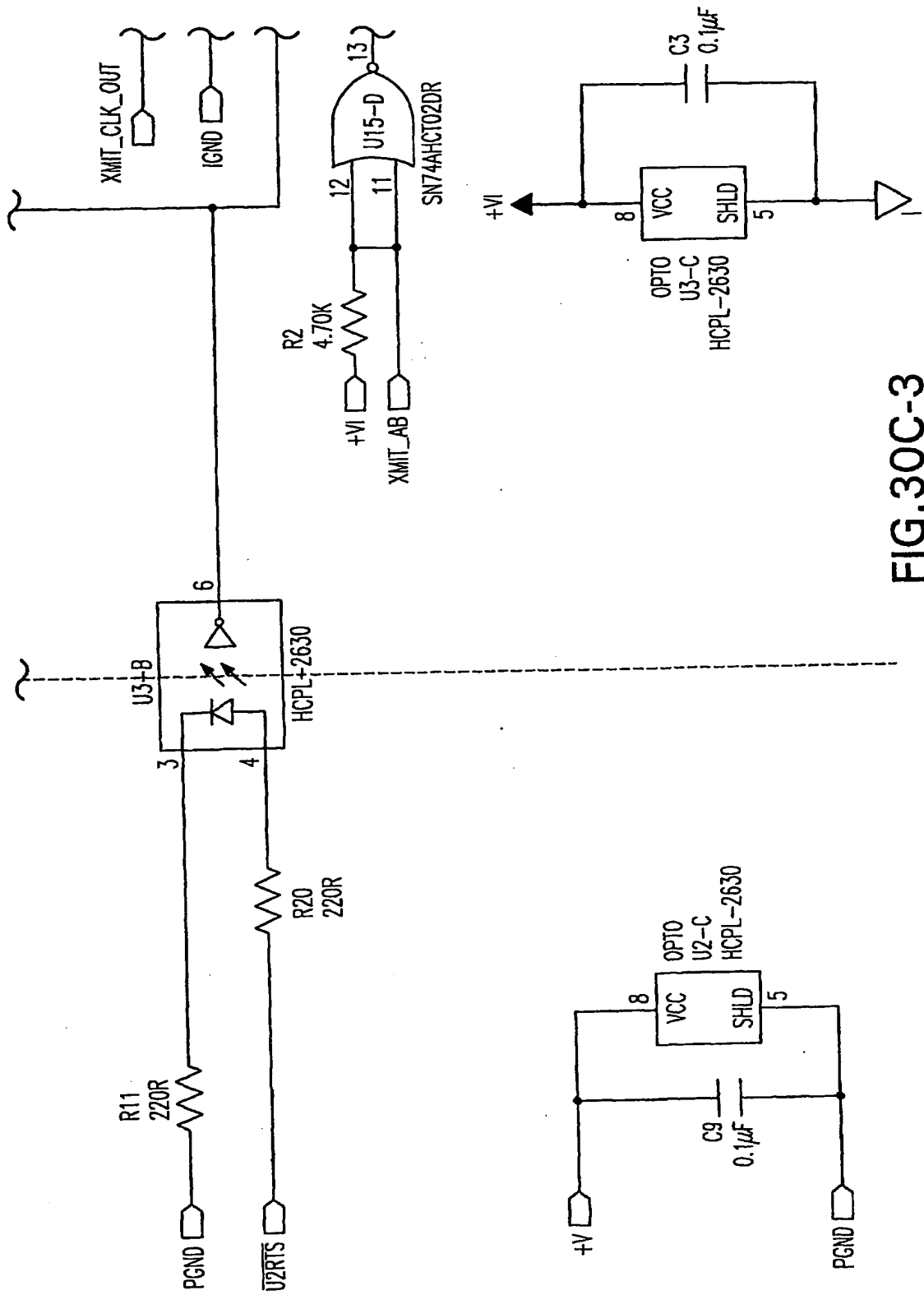


FIG. 30C-3

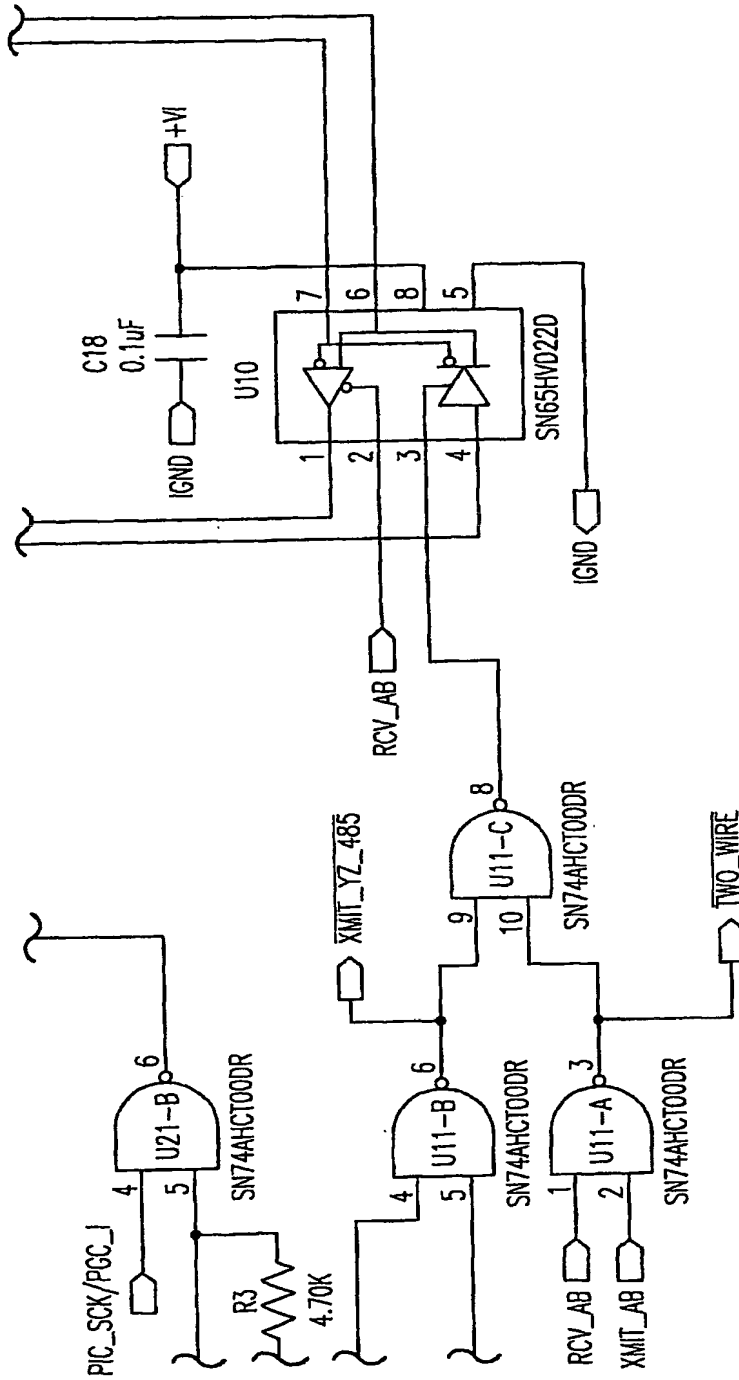


FIG.30C-4

PLACA DE PC DE I/O PCB230-1

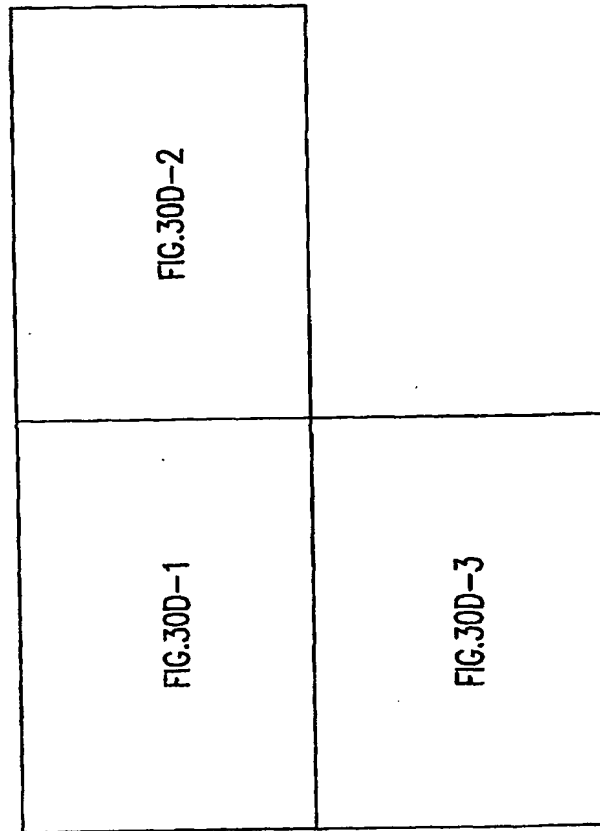


FIG.30D

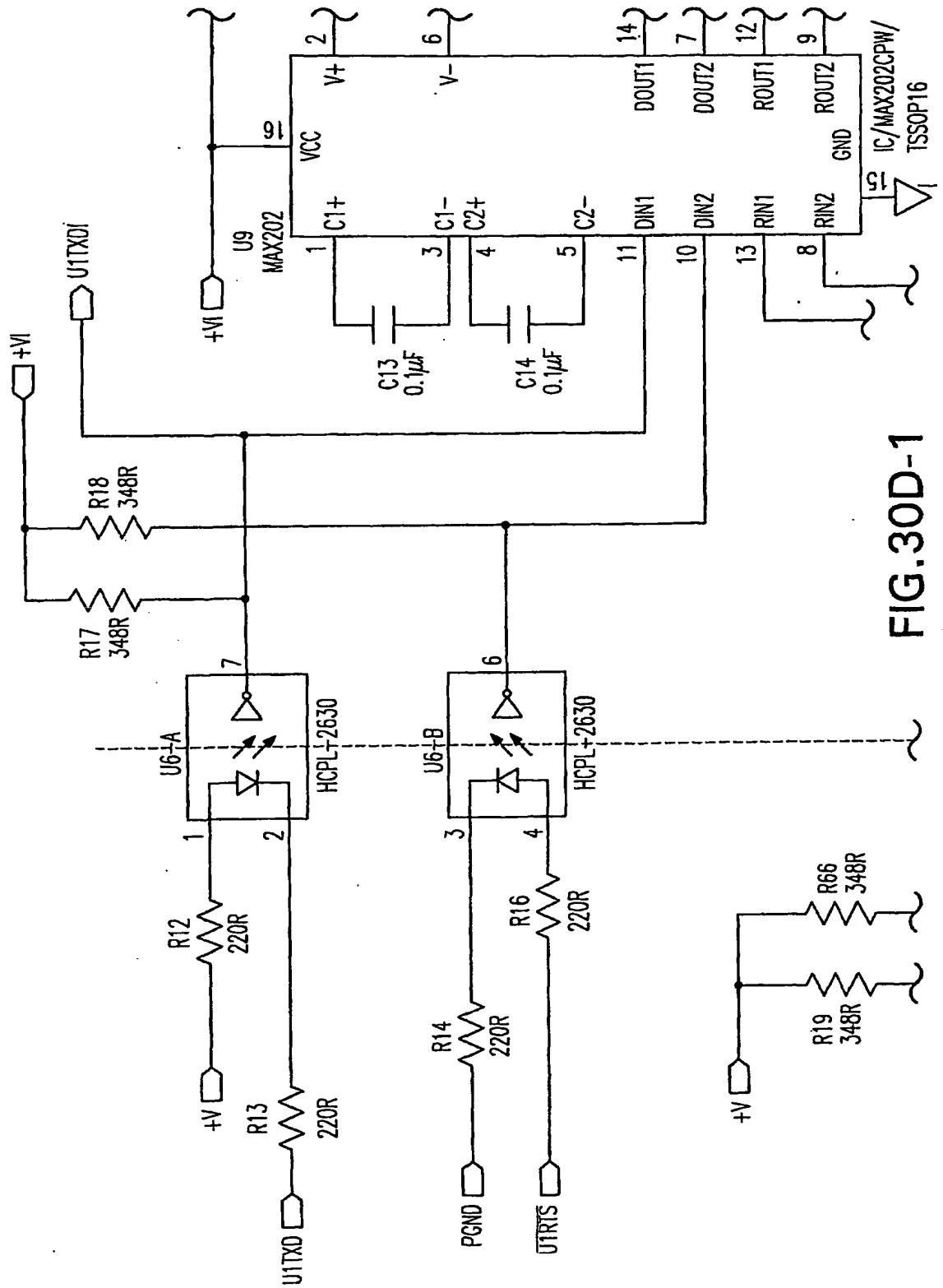


FIG.30D-1

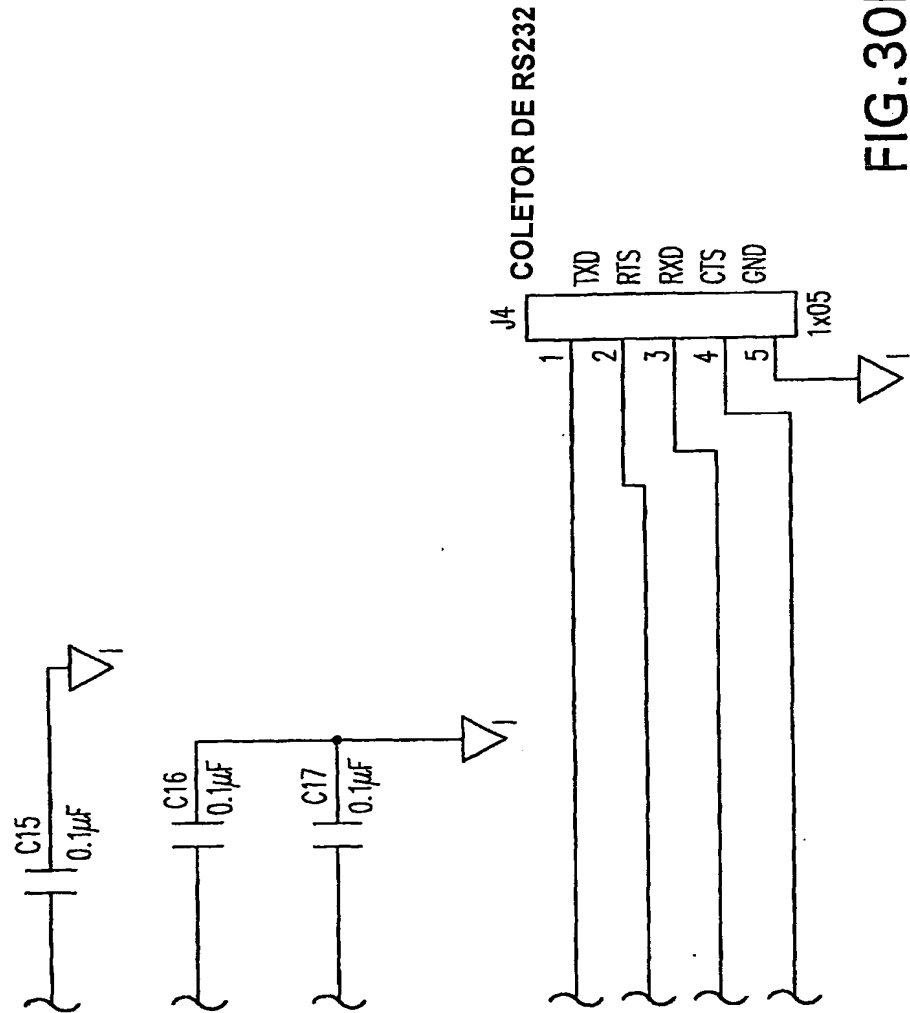


FIG.30D-2

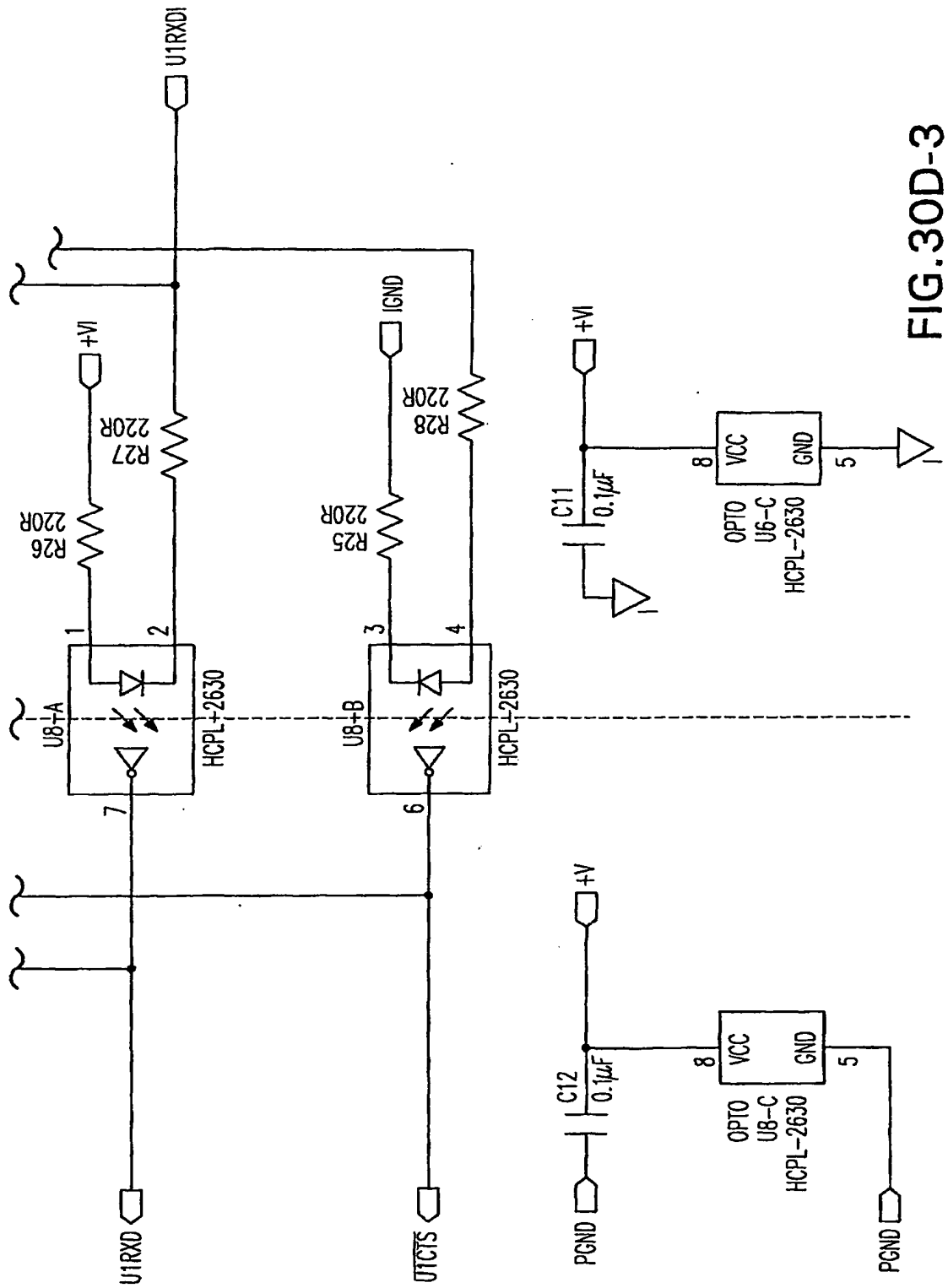


FIG.30D-3

SAÍDA DE RELÉ PCB230

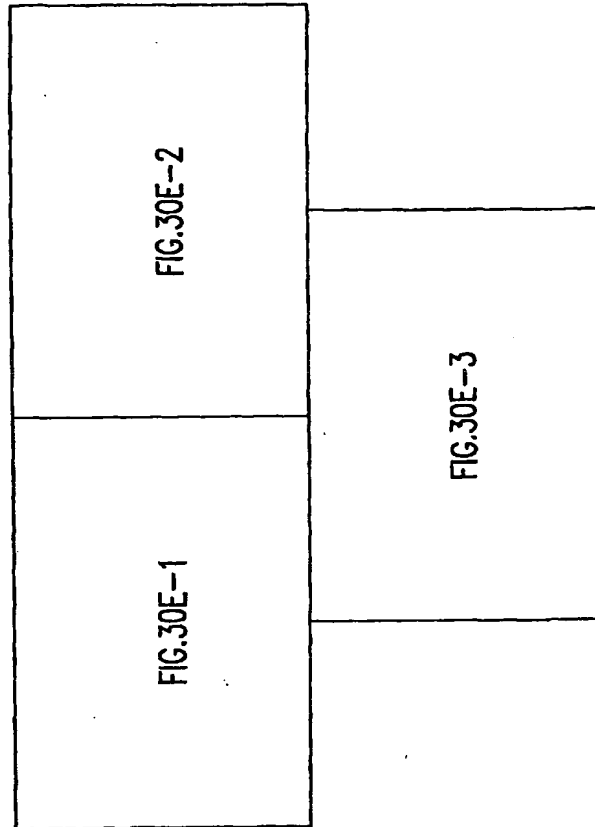


FIG.30E

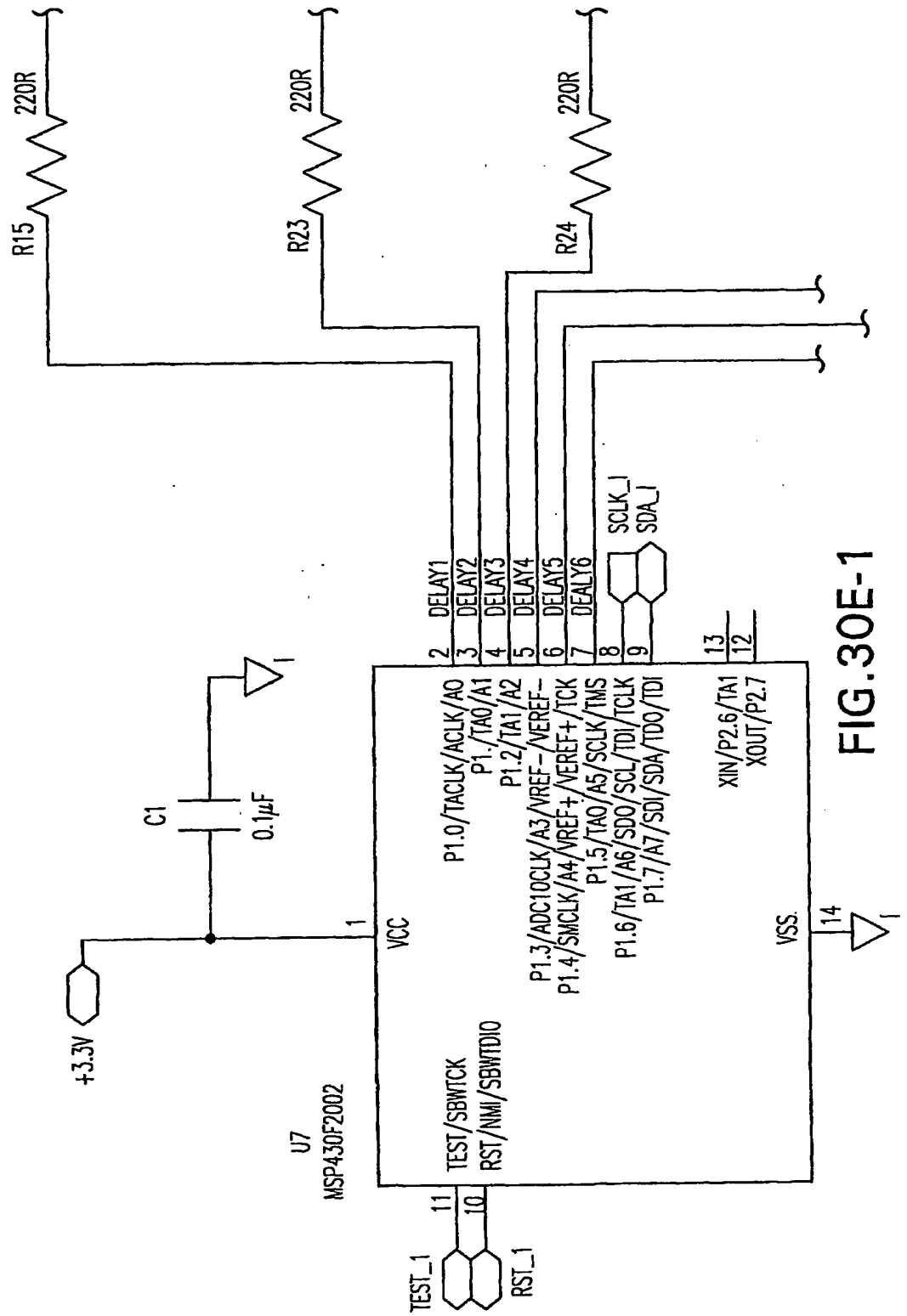


FIG.30E-1

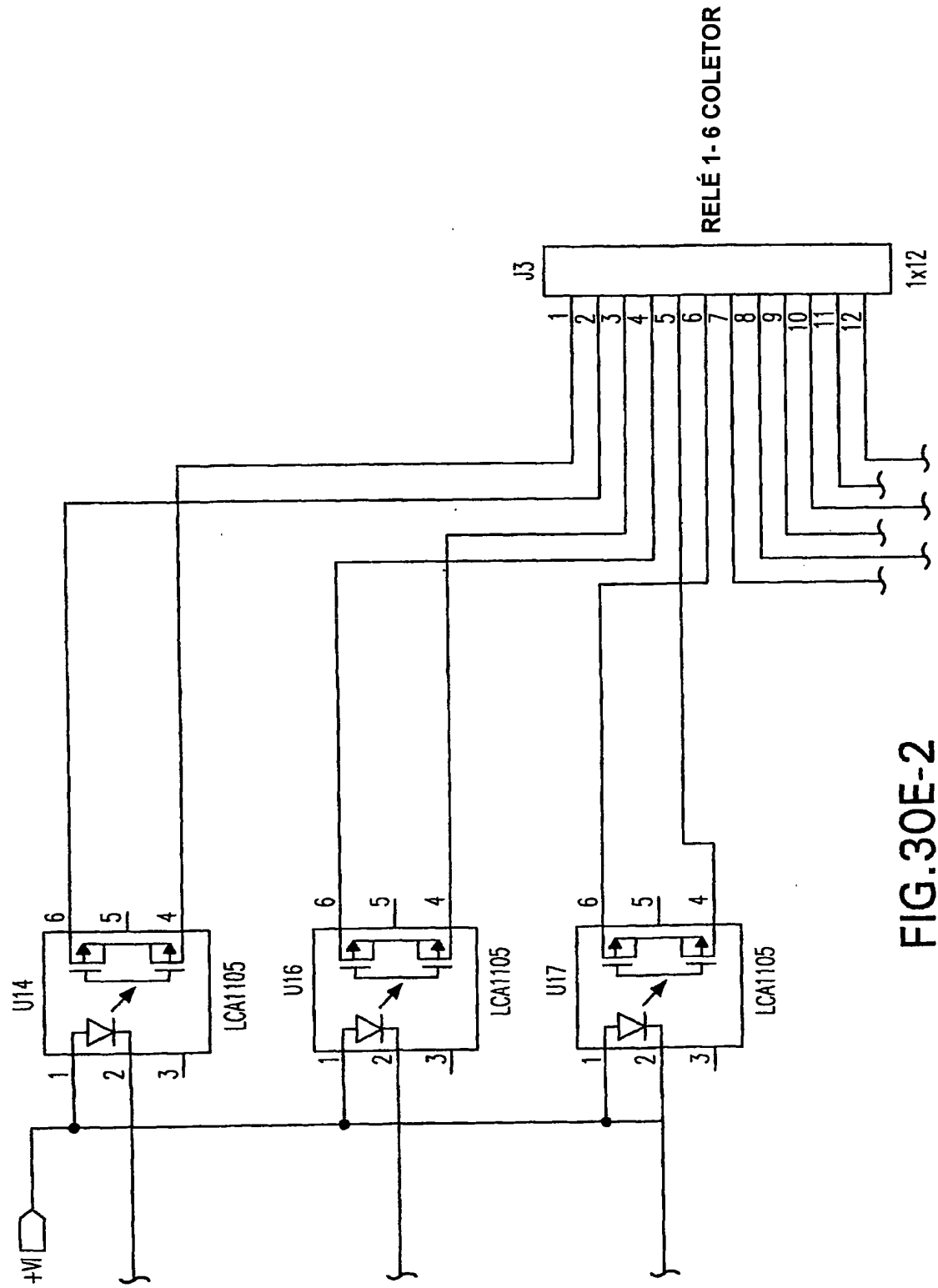


FIG. 30E-2

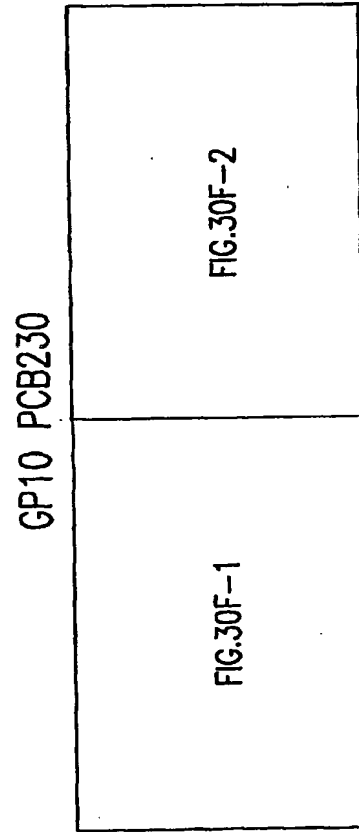


FIG.30F

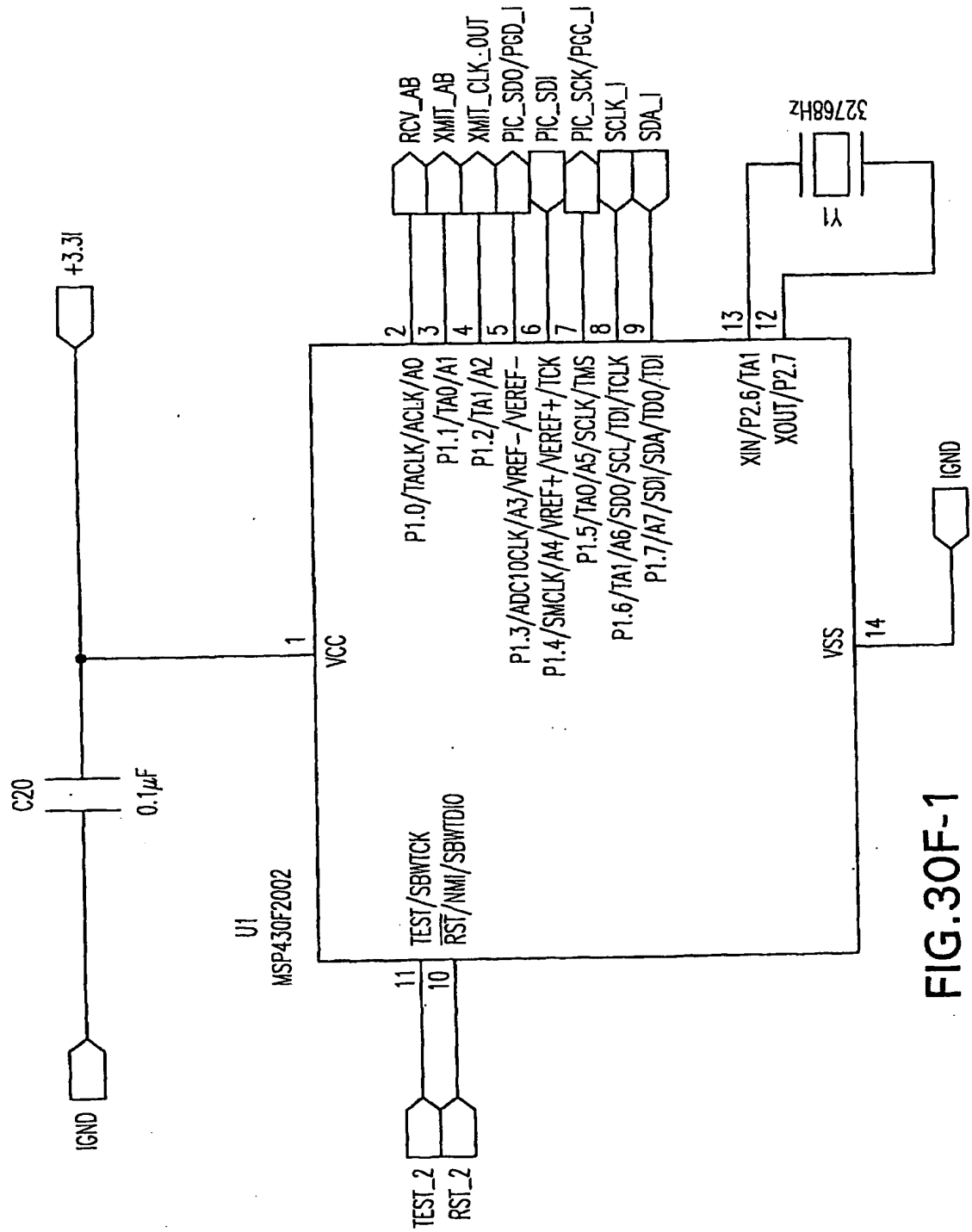


FIG.30F-1

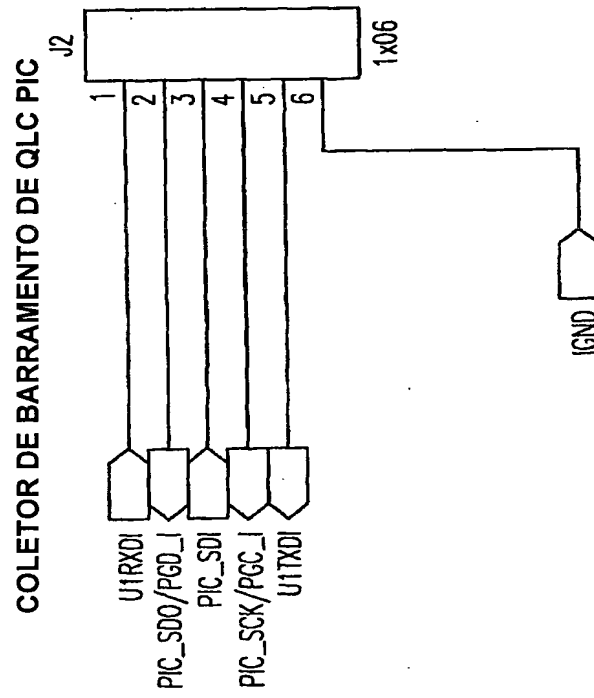


FIG.30F-2

12C PCB230

FIG.30G-1	FIG.30G-2
FIG.30G-3	FIG.30G-4

FIG.30G

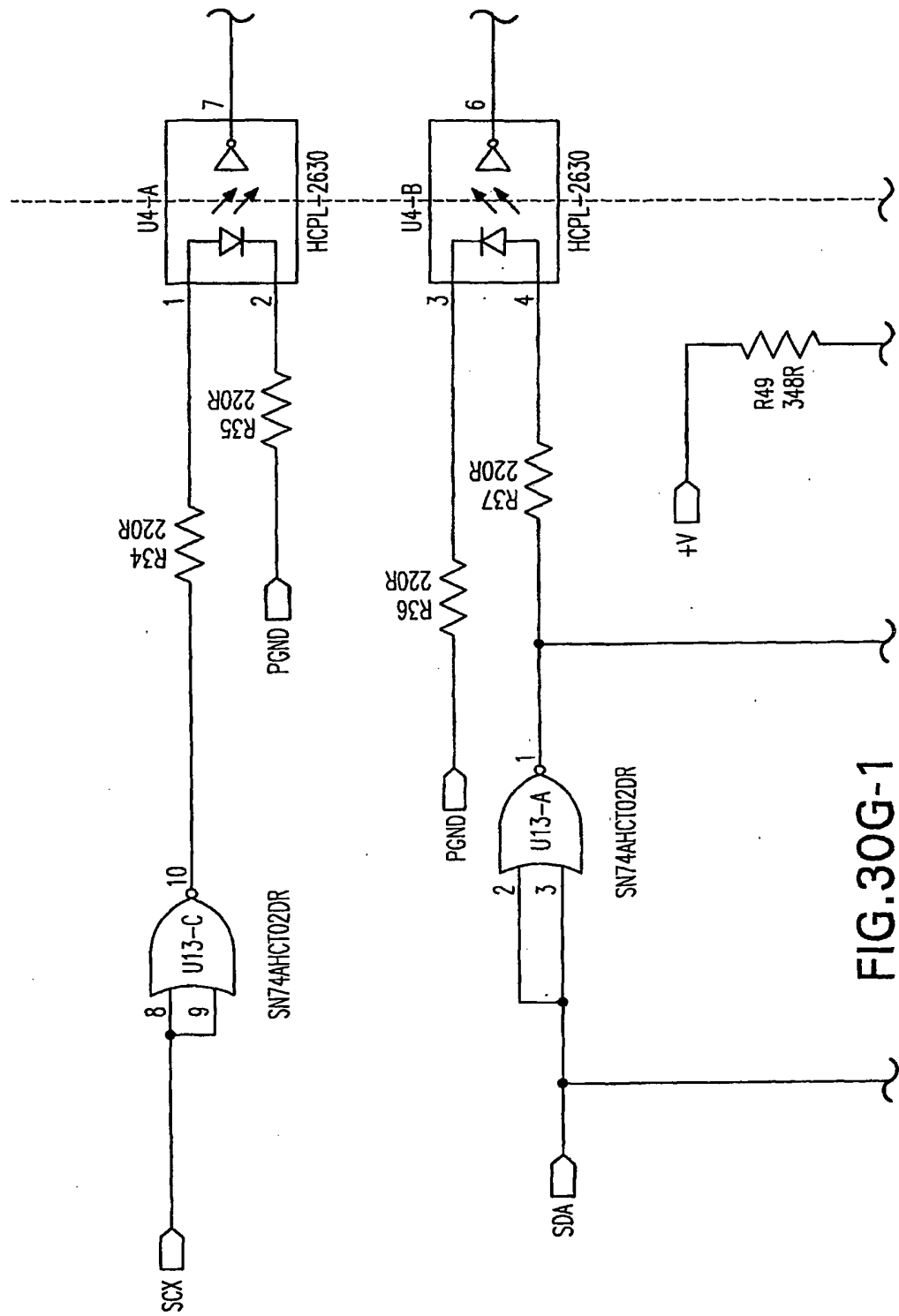


FIG. 30G-1

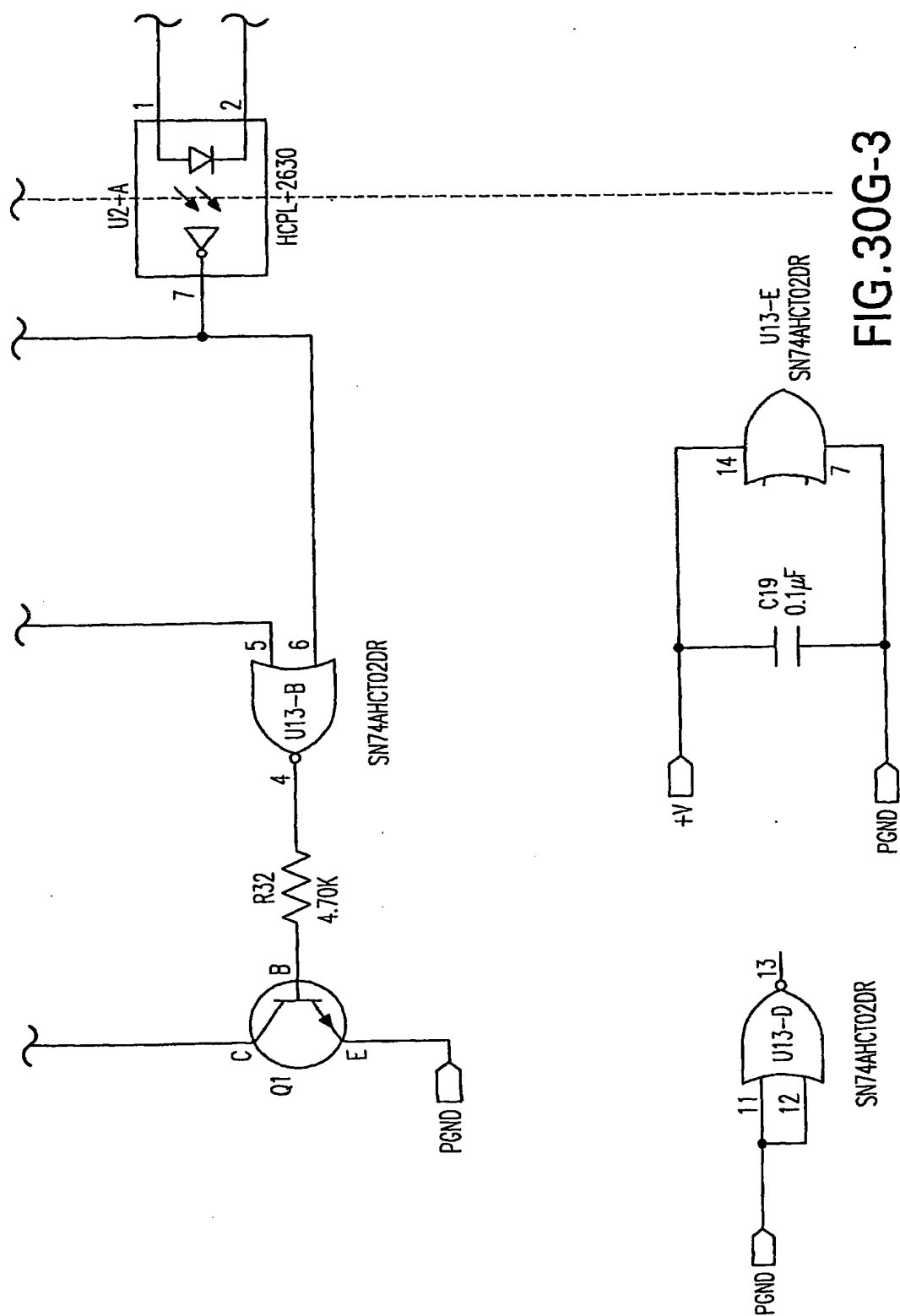




FIG. 30G-4

CPU SCH202-3

FIG.31A-1	FIG.31A-2	FIG.31A-3	FIG.31A-4
FIG.31A-5	FIG.31A-6	FIG.31A-7	FIG.31A-8

FIG.31A

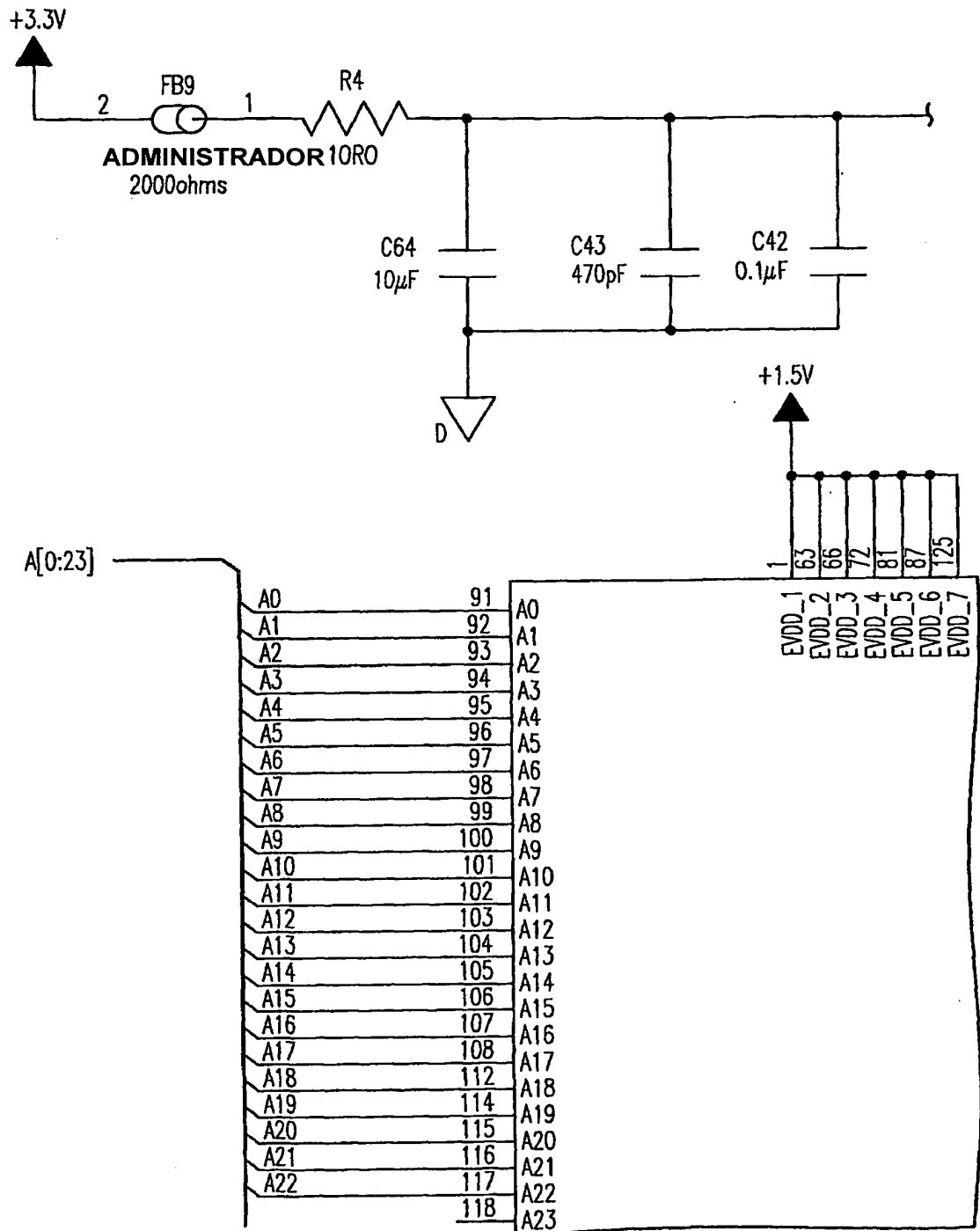


FIG.31A-1

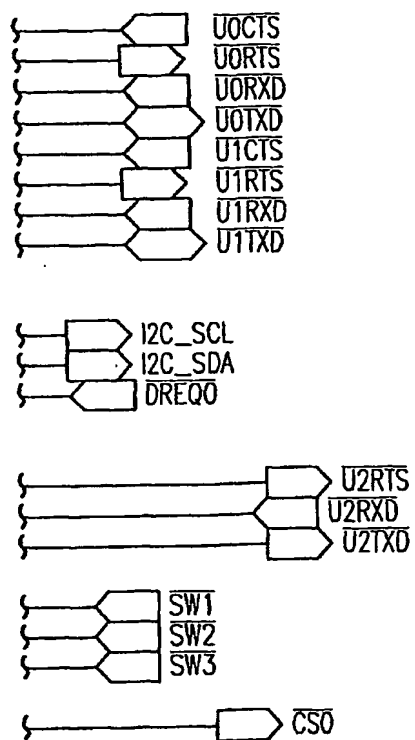


FIG.31 A-3

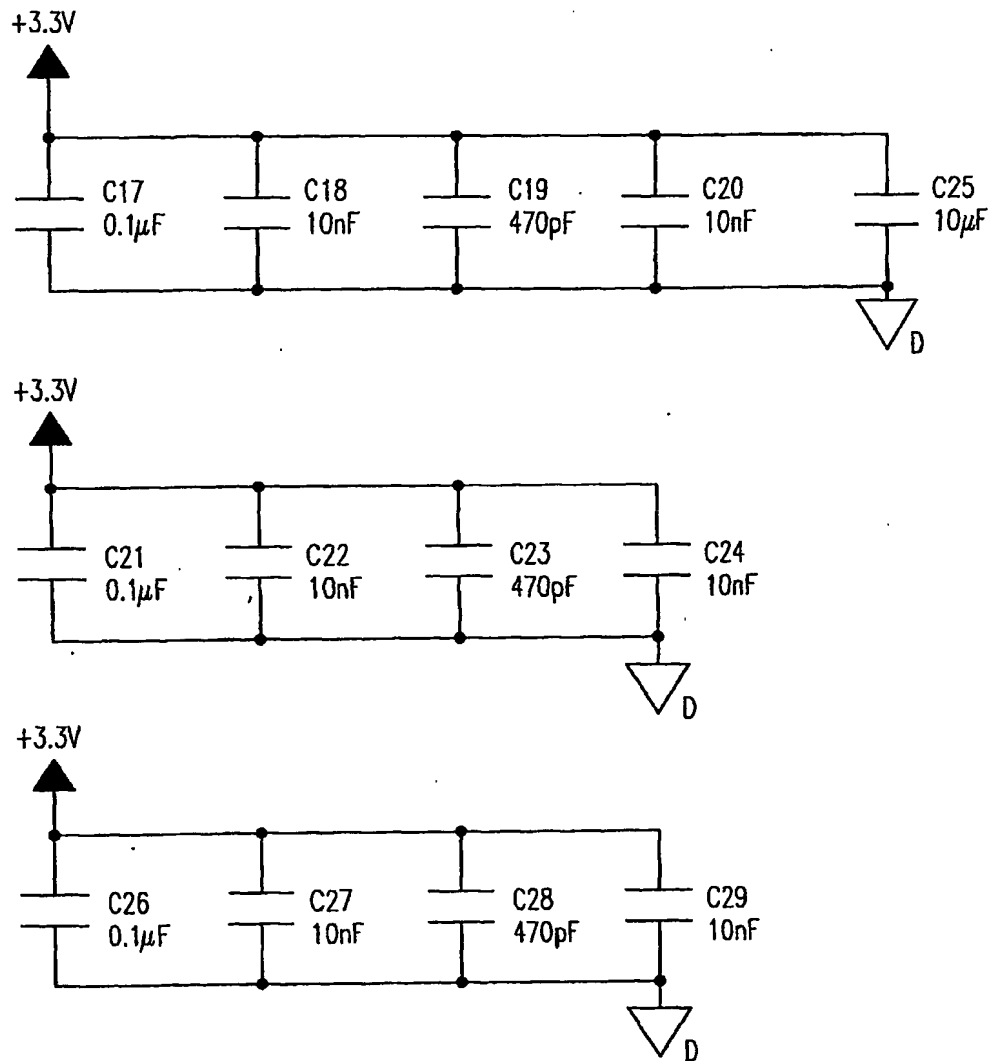


FIG.31A-4

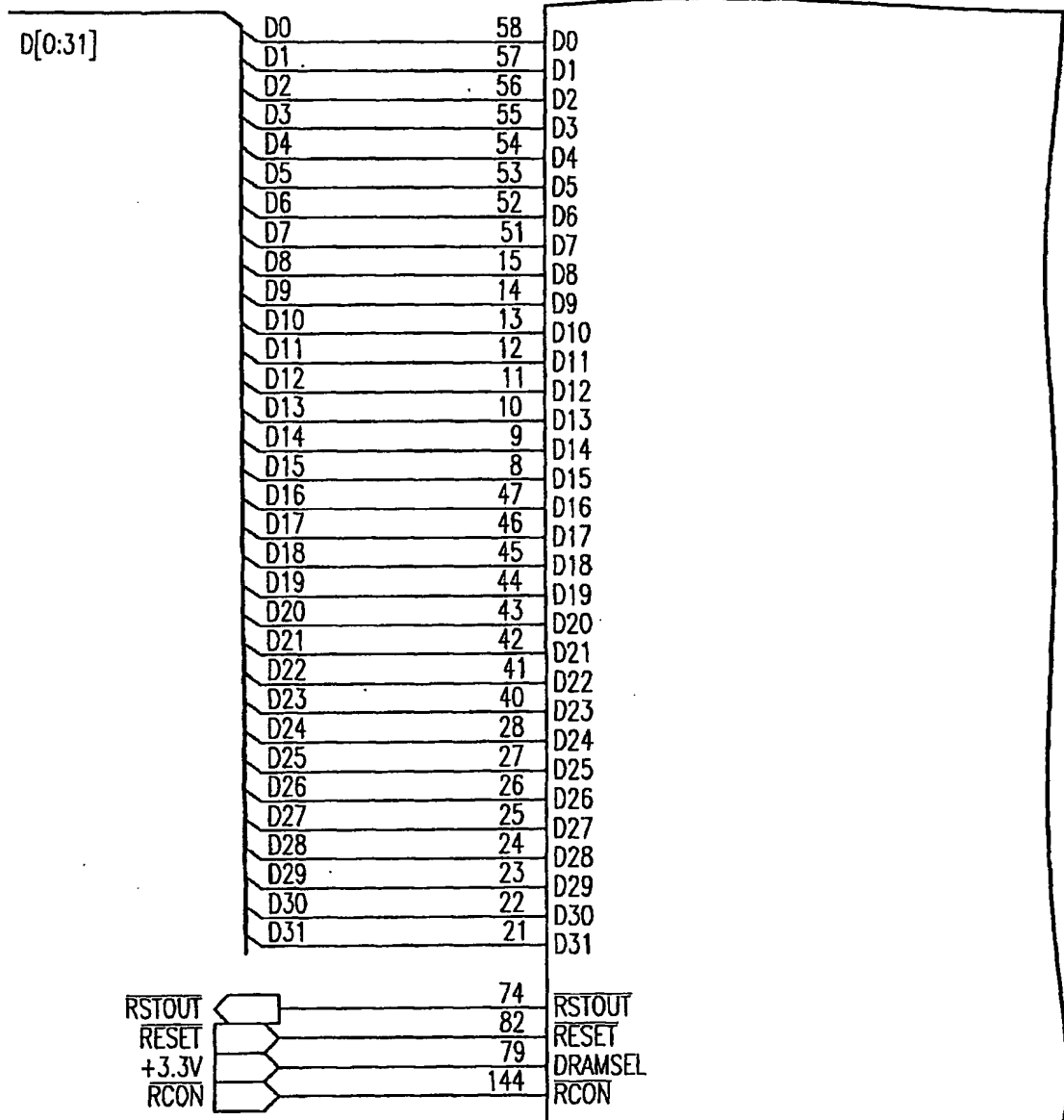


FIG.31A-5

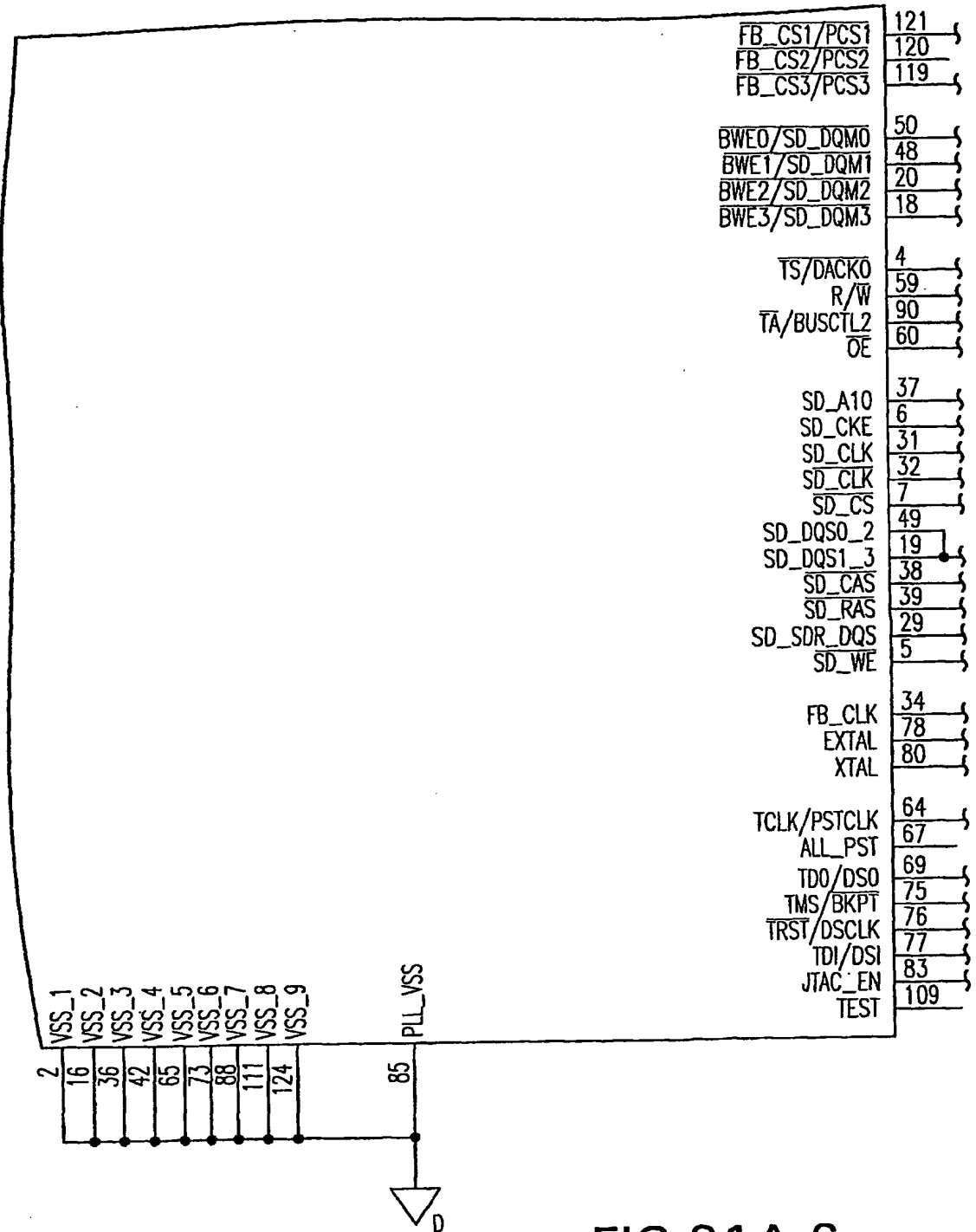


FIG.31A-6

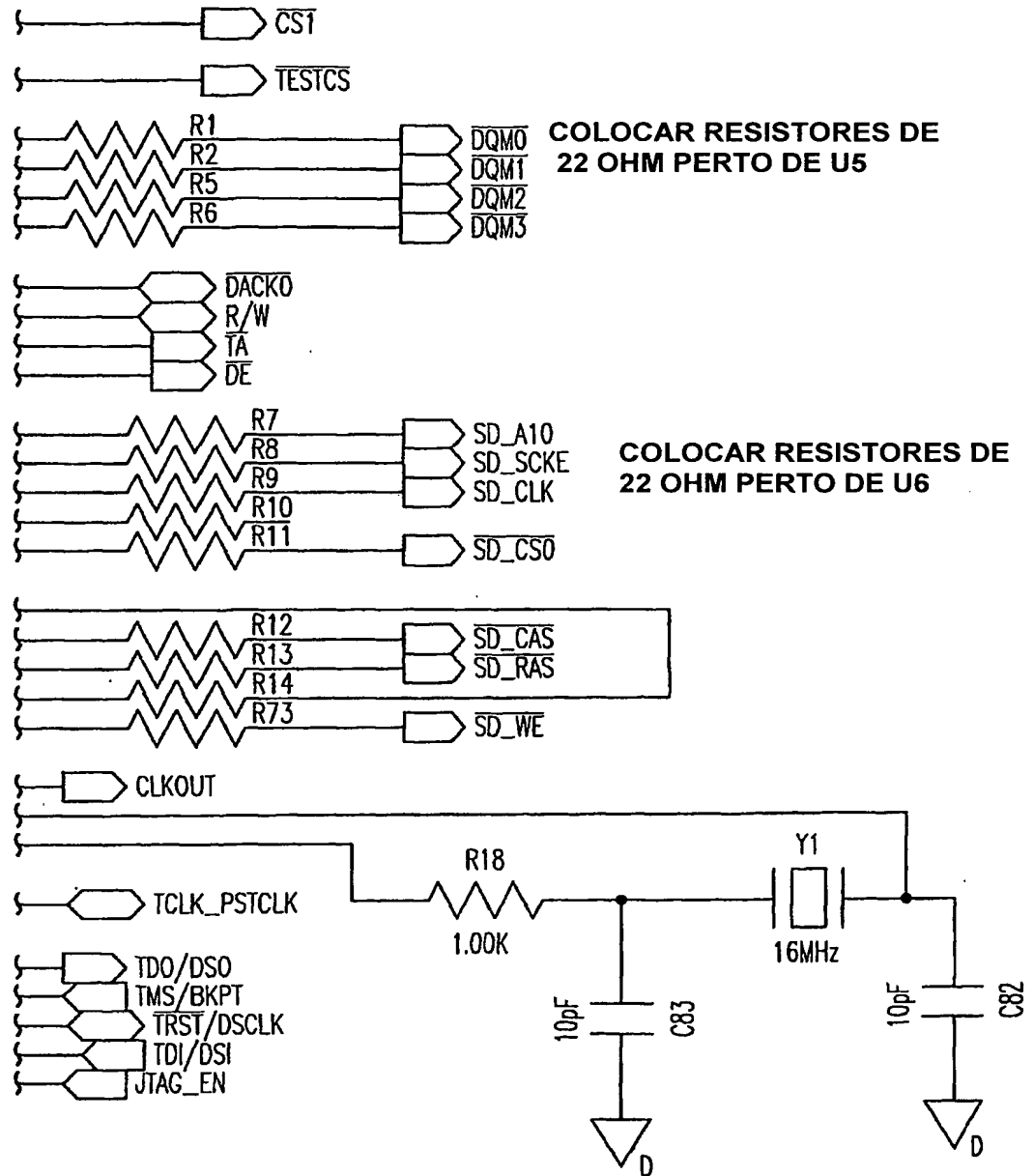


FIG.31A-7

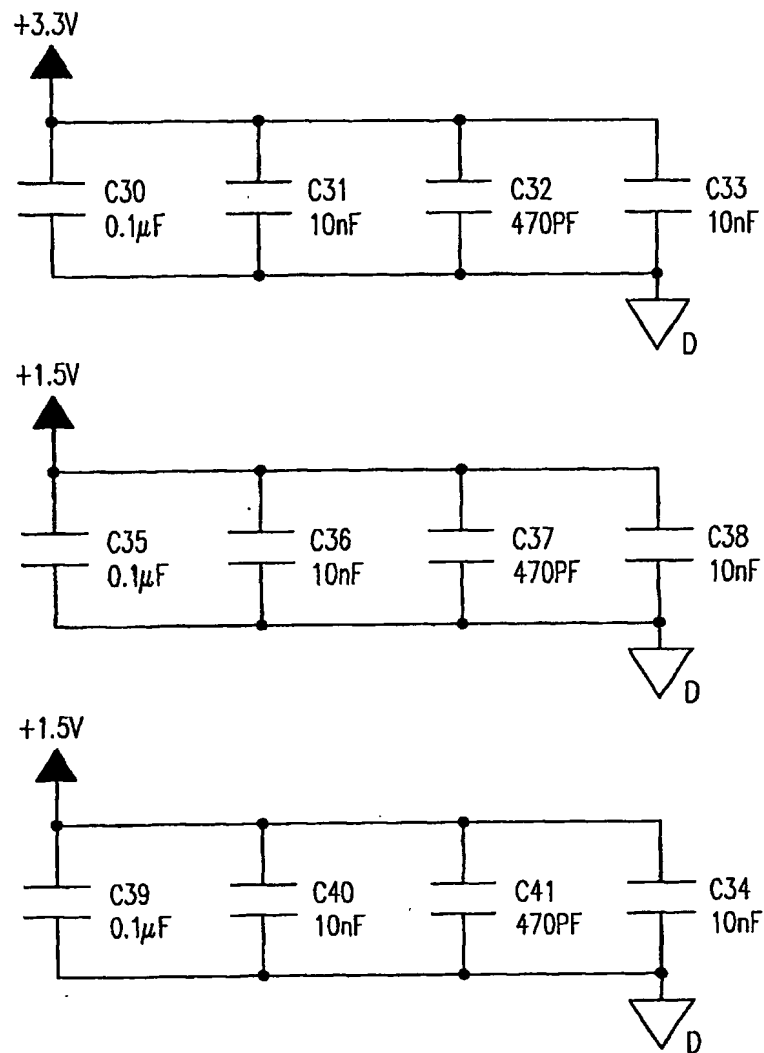


FIG.31A-8

UNIDADE DE FONTE DE ENERGIA SCH202-3

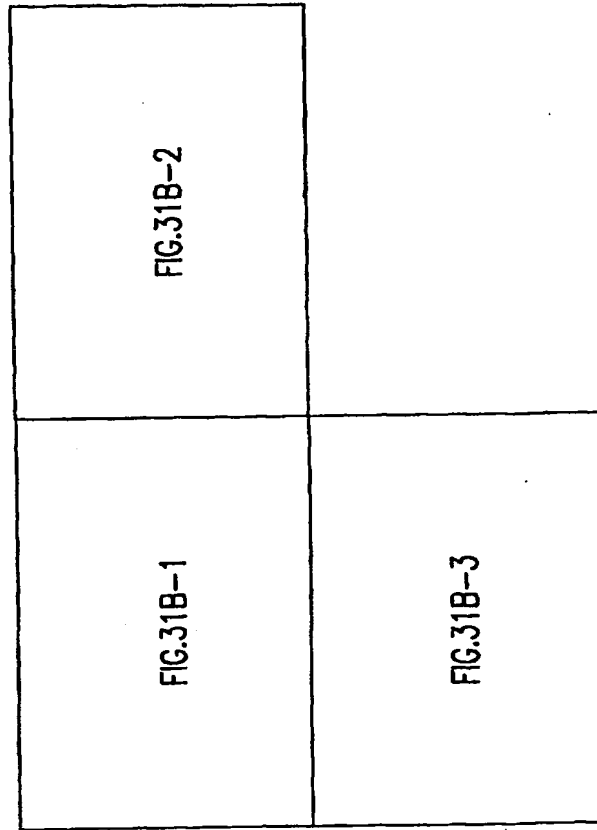


FIG.31B

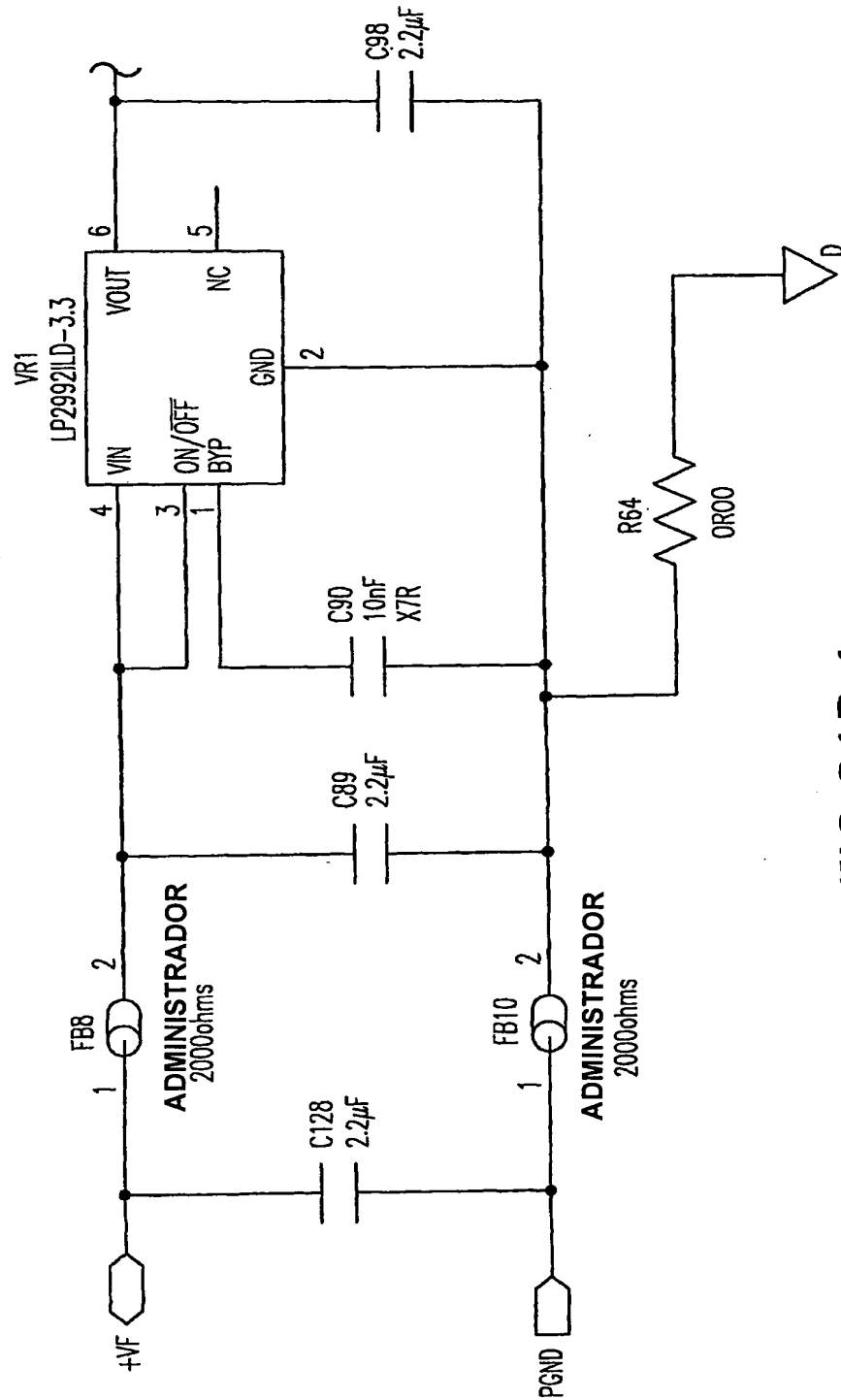


FIG.31B-1

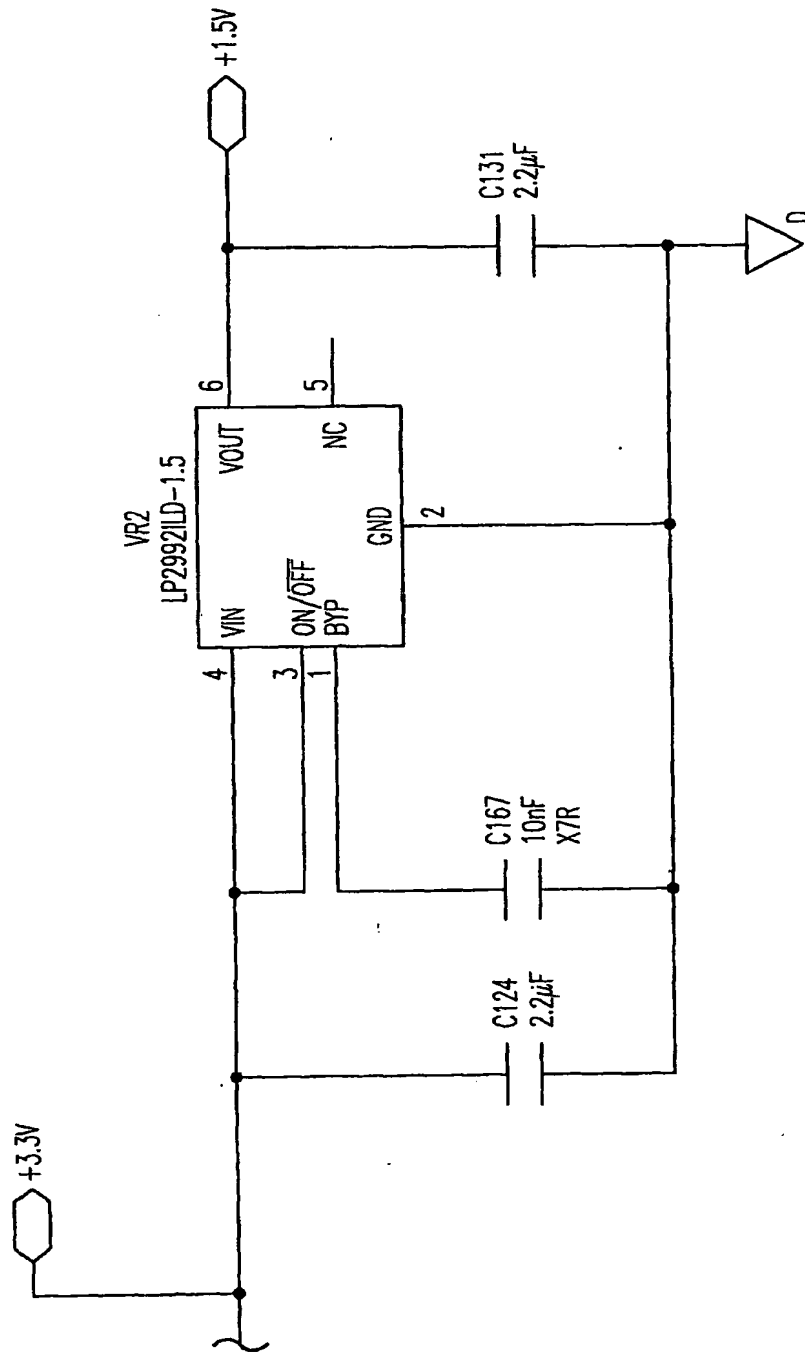
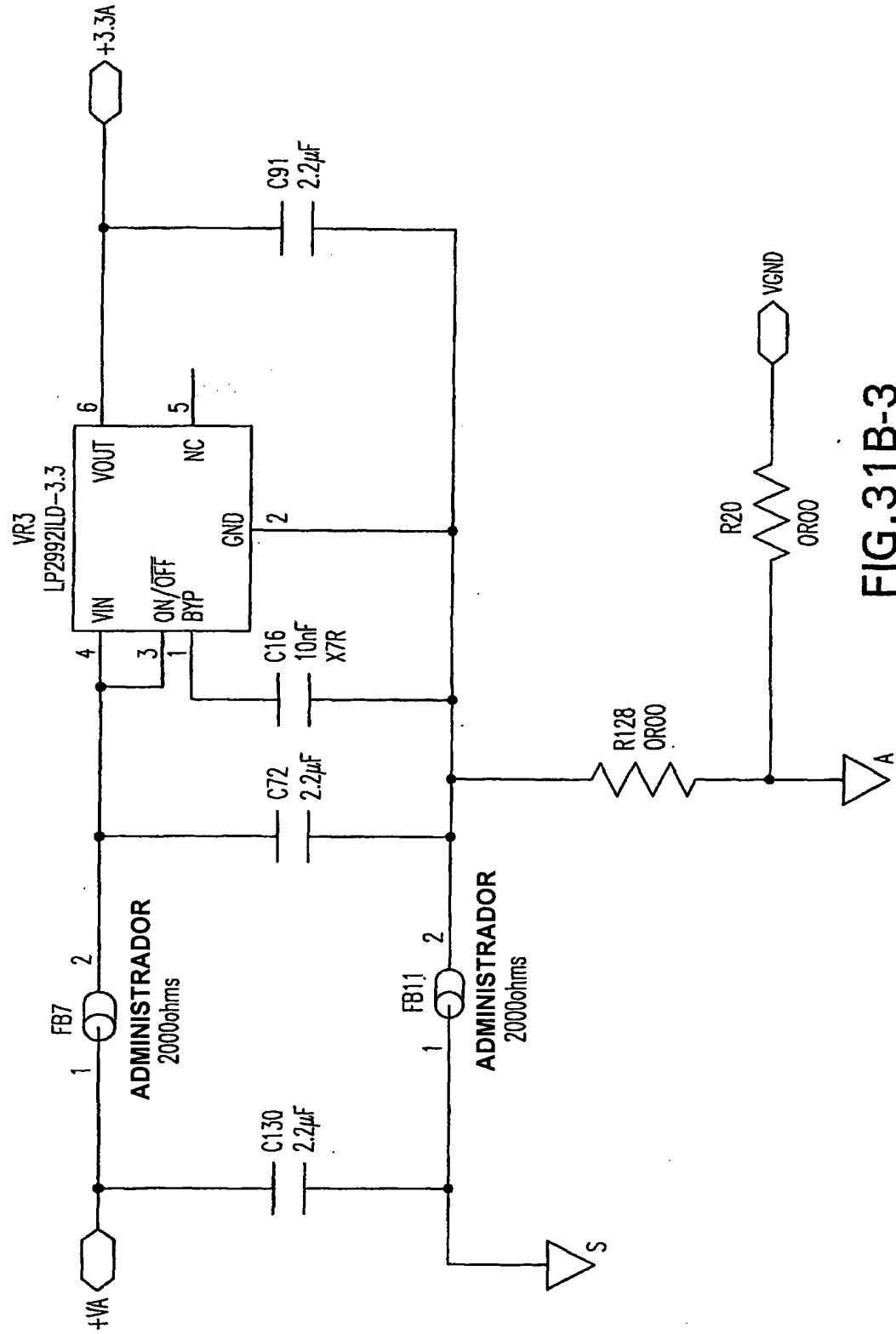


FIG.31B-2



RTC/FRAUDE TEMP/REAJUSTE PCB202-3

FIG.31C-1	FIG.31C-2	FIG.31C-3
FIG.31C-4	FIG.31C-5	FIG.31C-6

FIG.31C

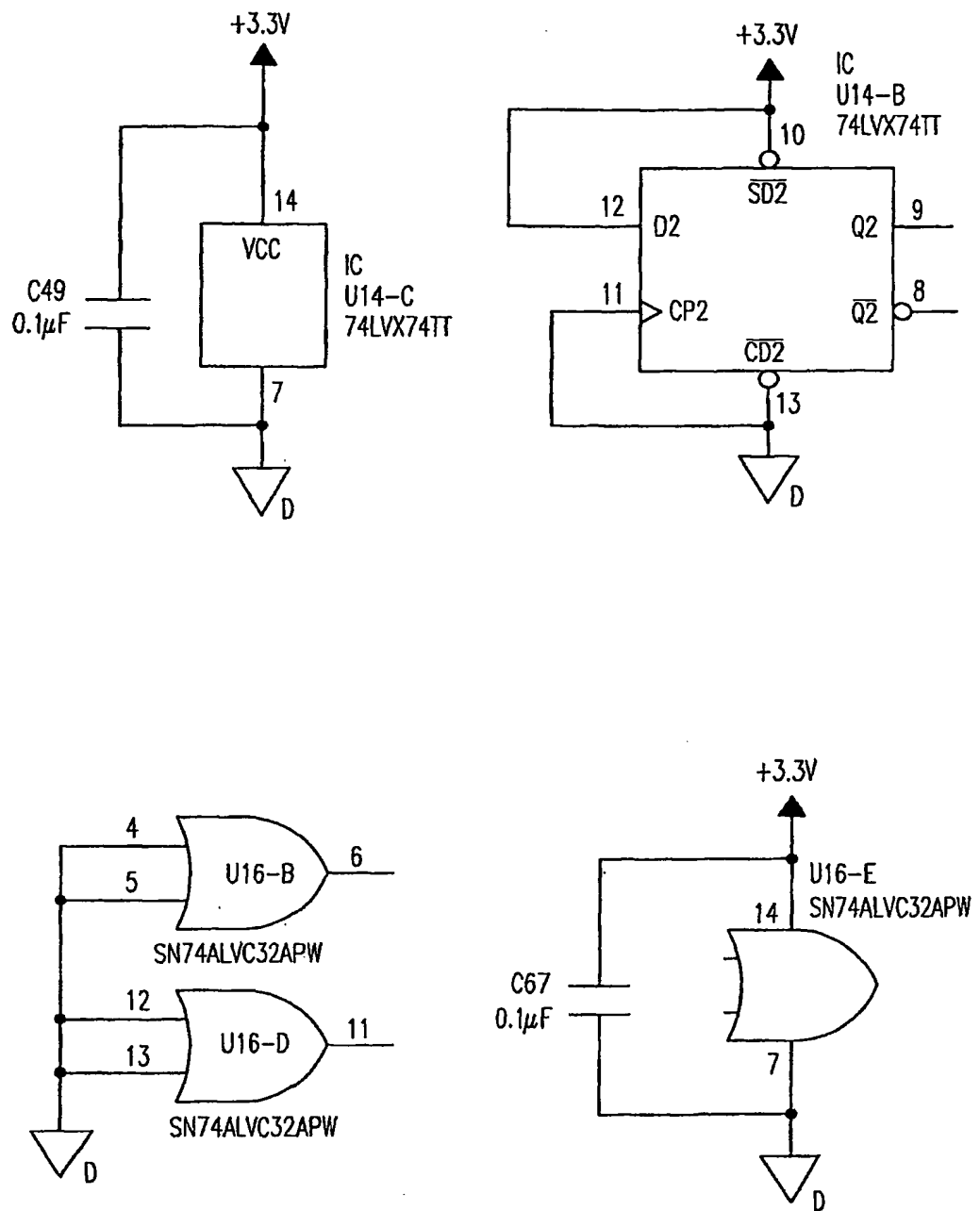


FIG.31C-1

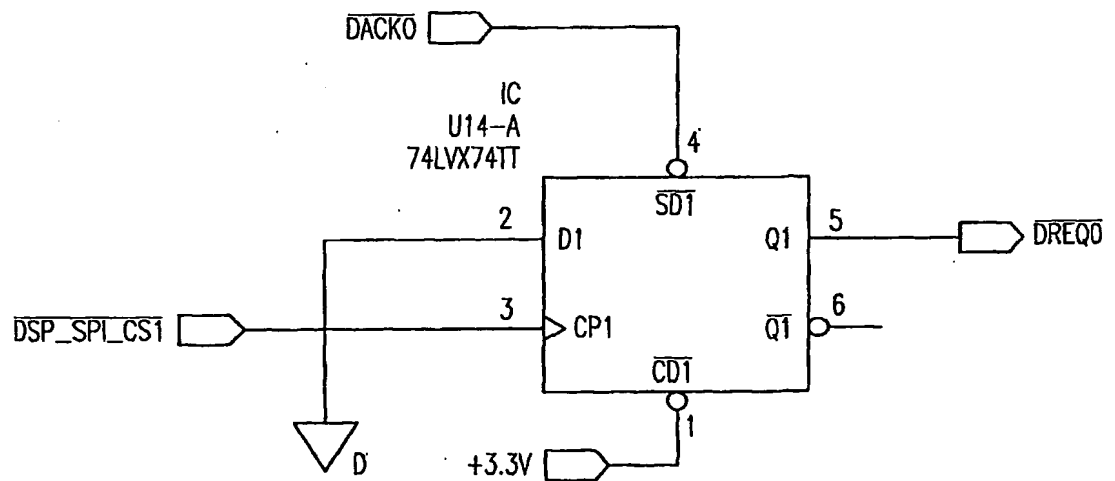
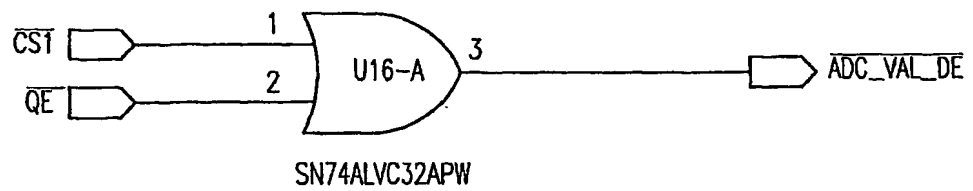
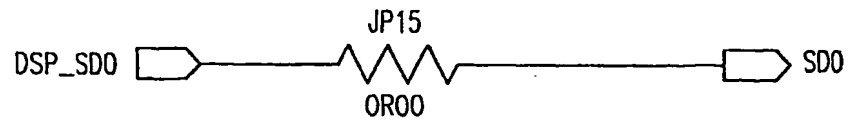


FIG.31C-2

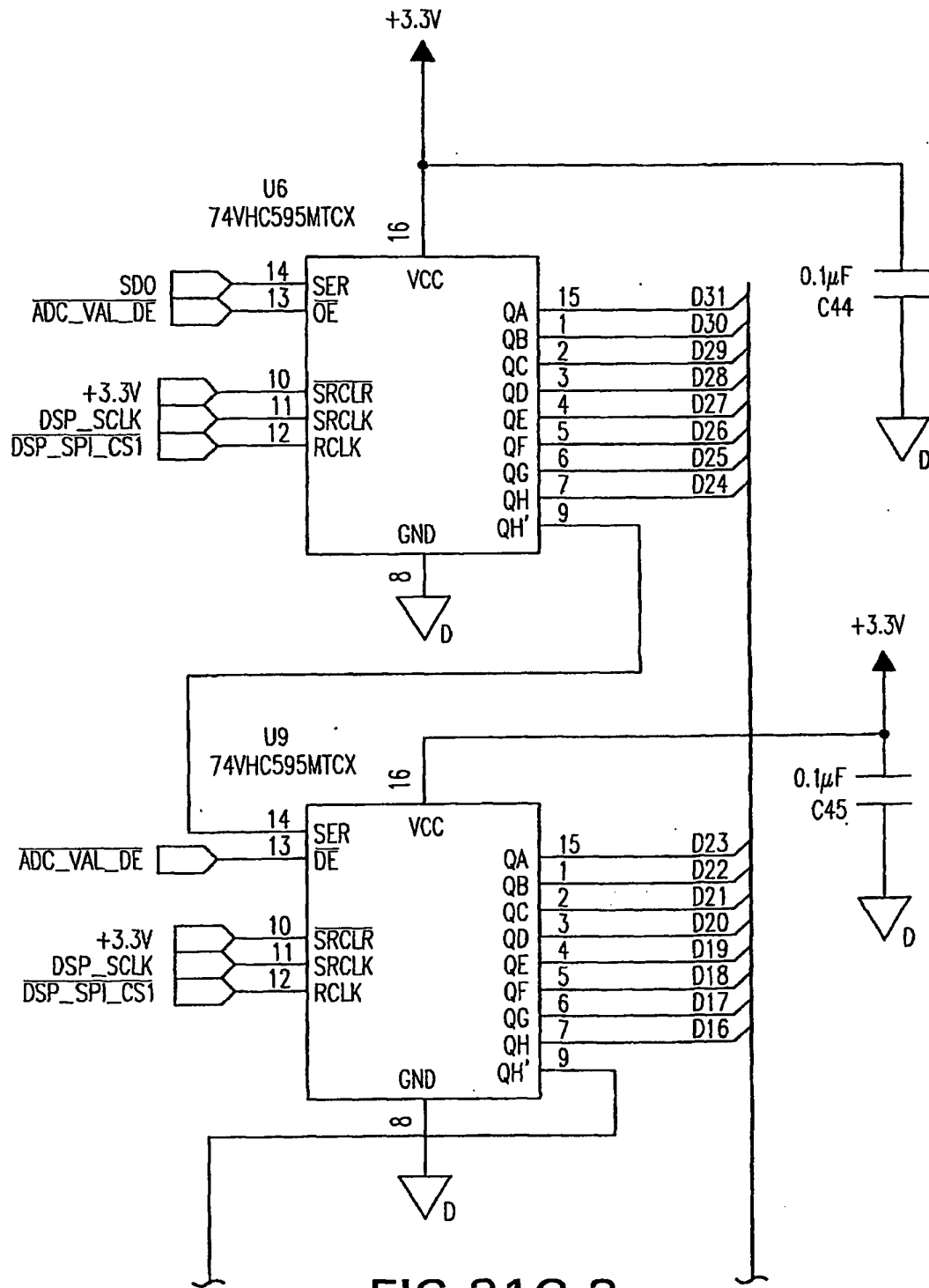


FIG.31C-3

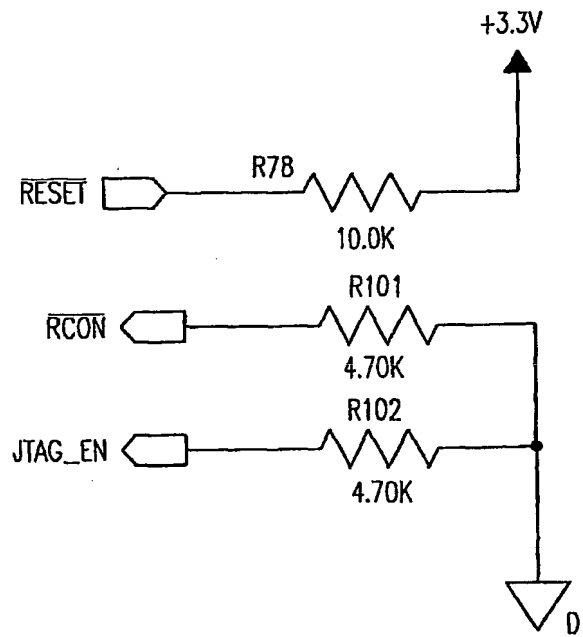


FIG.31C-4

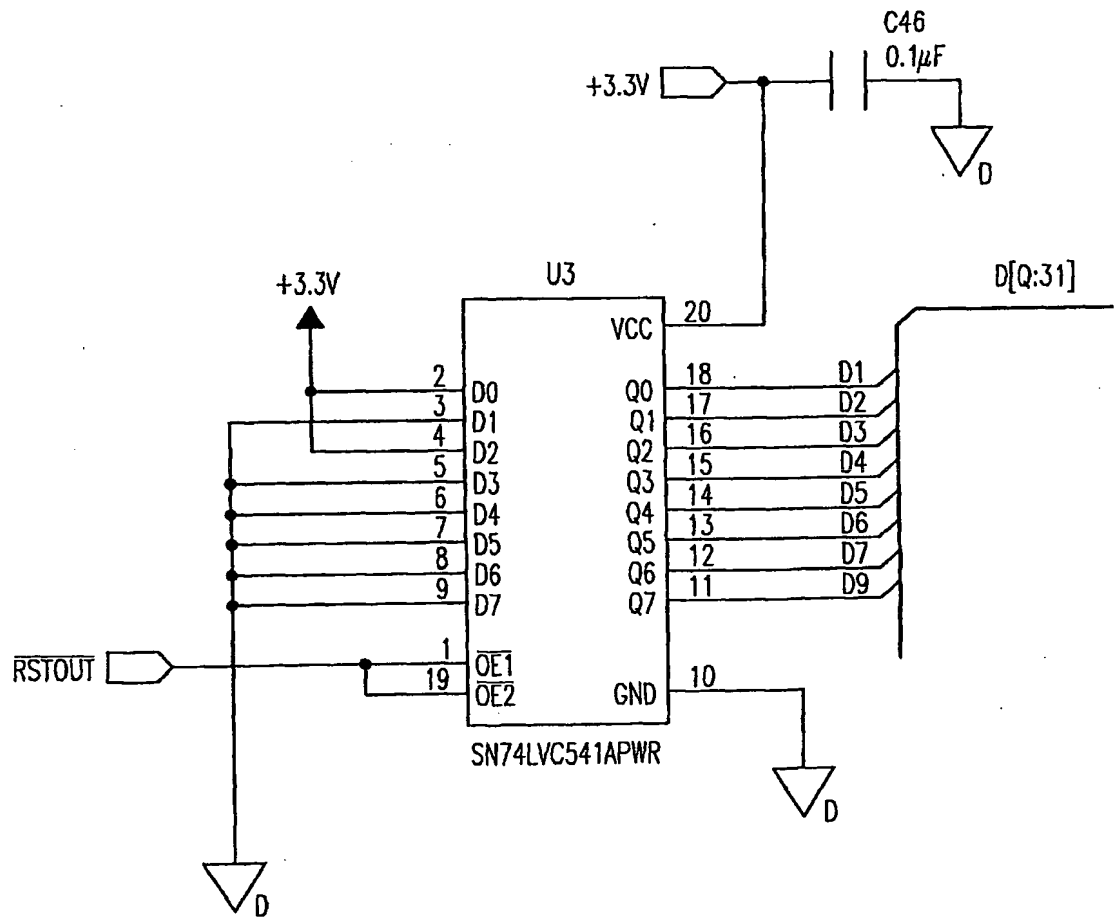


FIG.31C-5

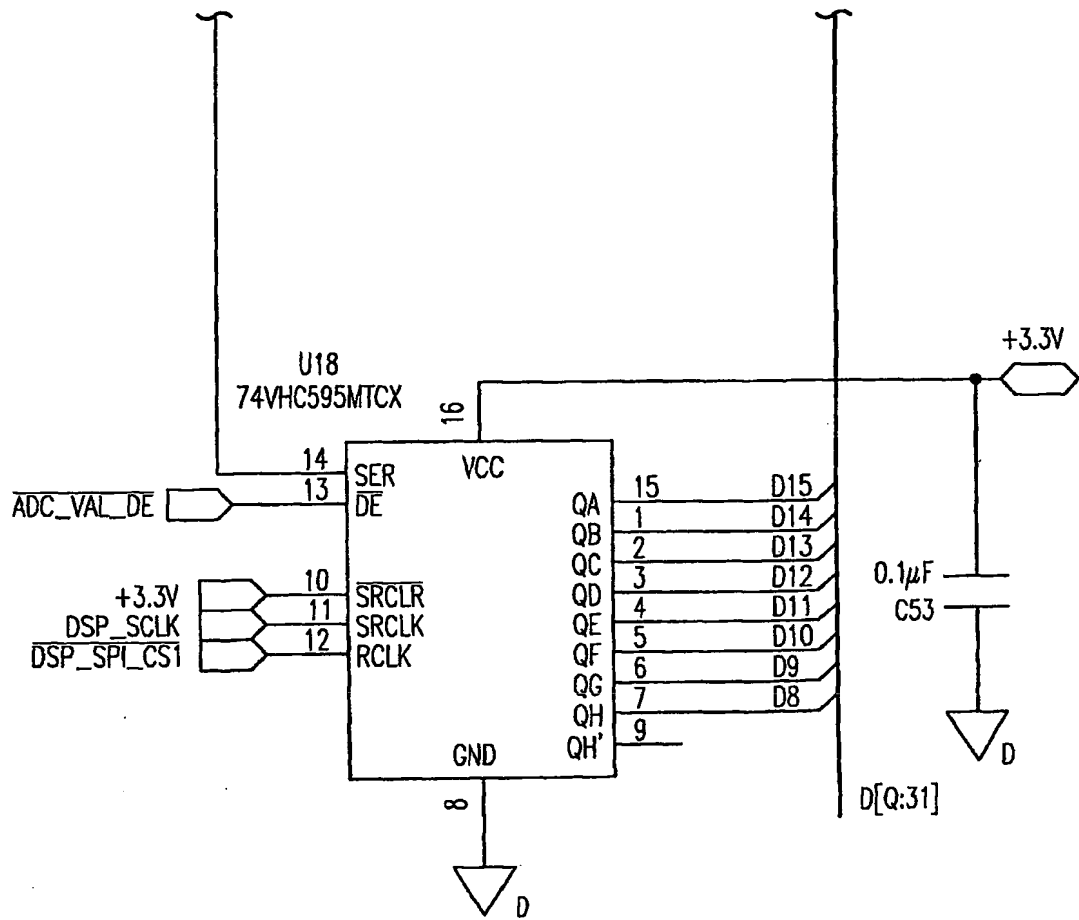


FIG.31C-6

RTC/FRAUDE TEMPI/REAJUSTE PCB202-3

FIG.31D-1	FIG.31D-2
FIG.31D-3	FIG.31D-4

FIG.31D

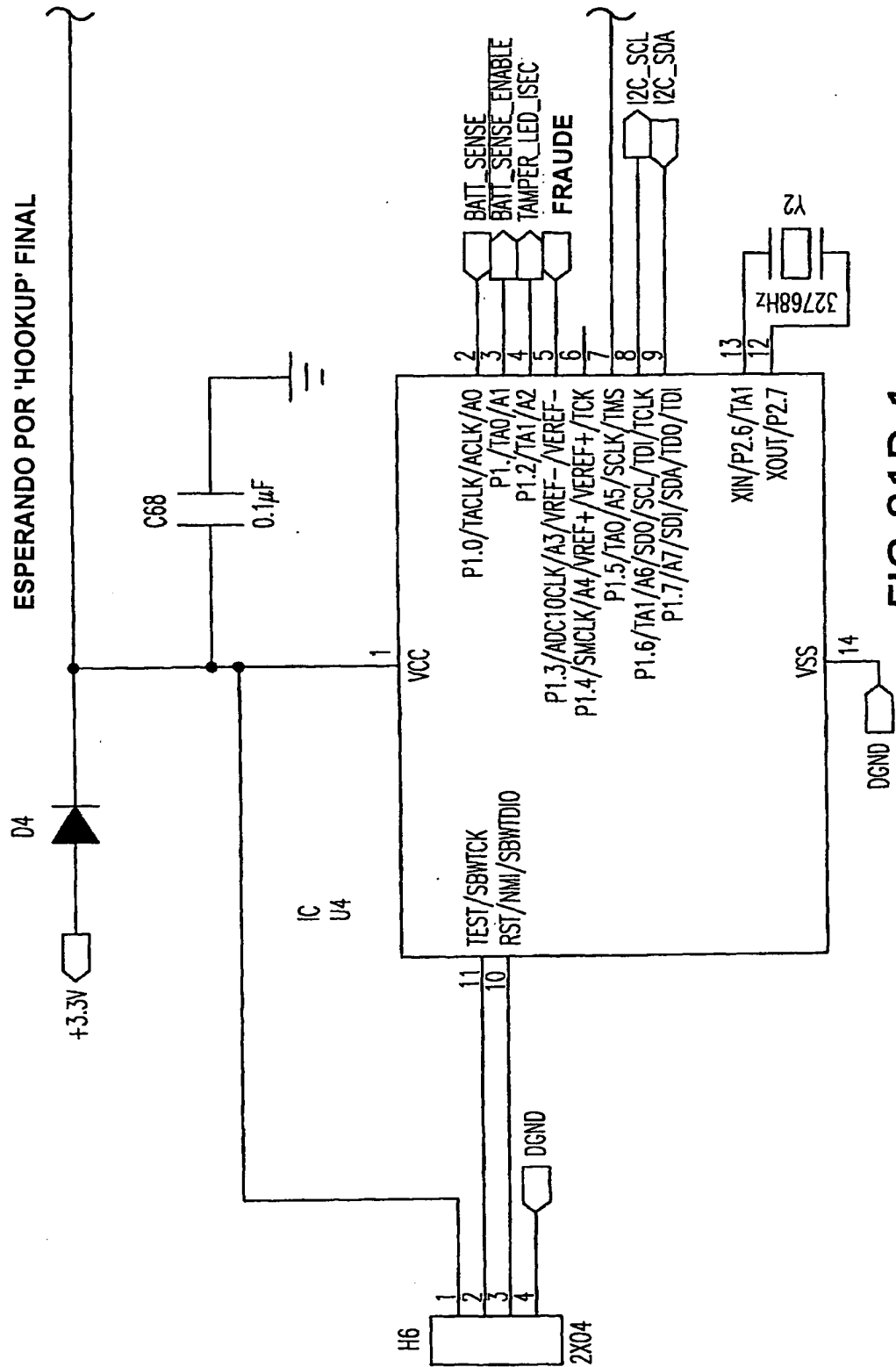


FIG.31D-1

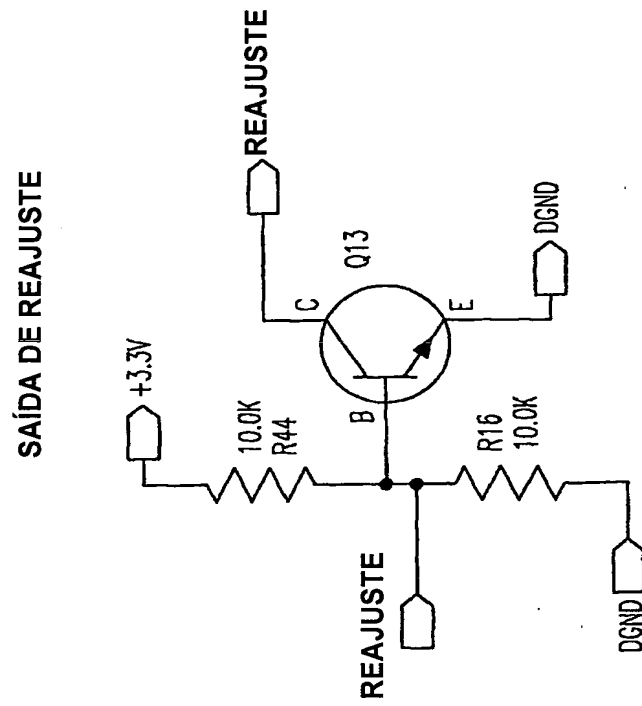


FIG.31D-4

MEMÓRIA FLASH SCH202-3

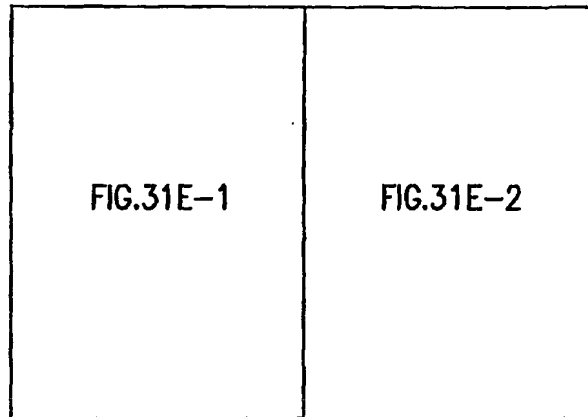


FIG.31E

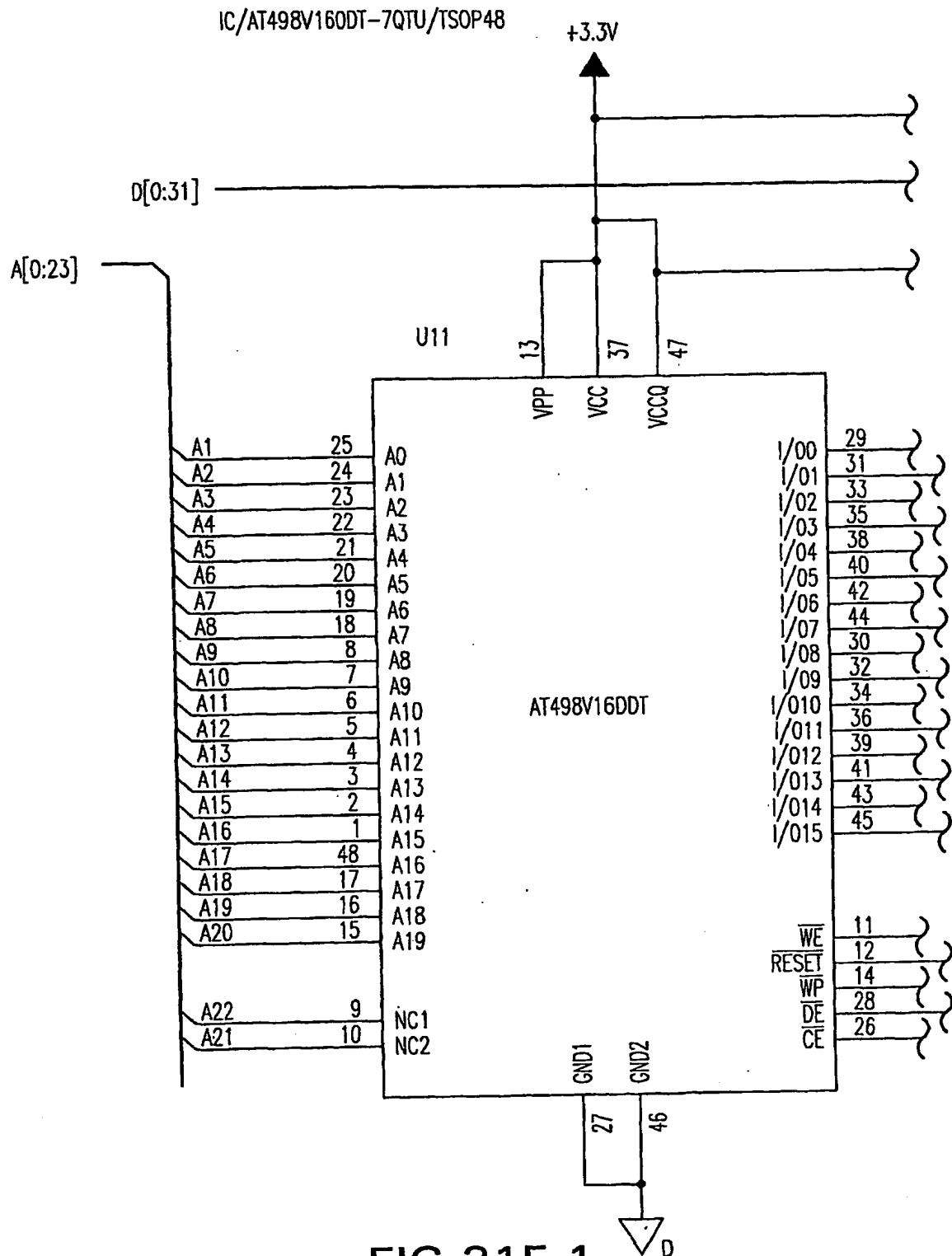


FIG.31E-1

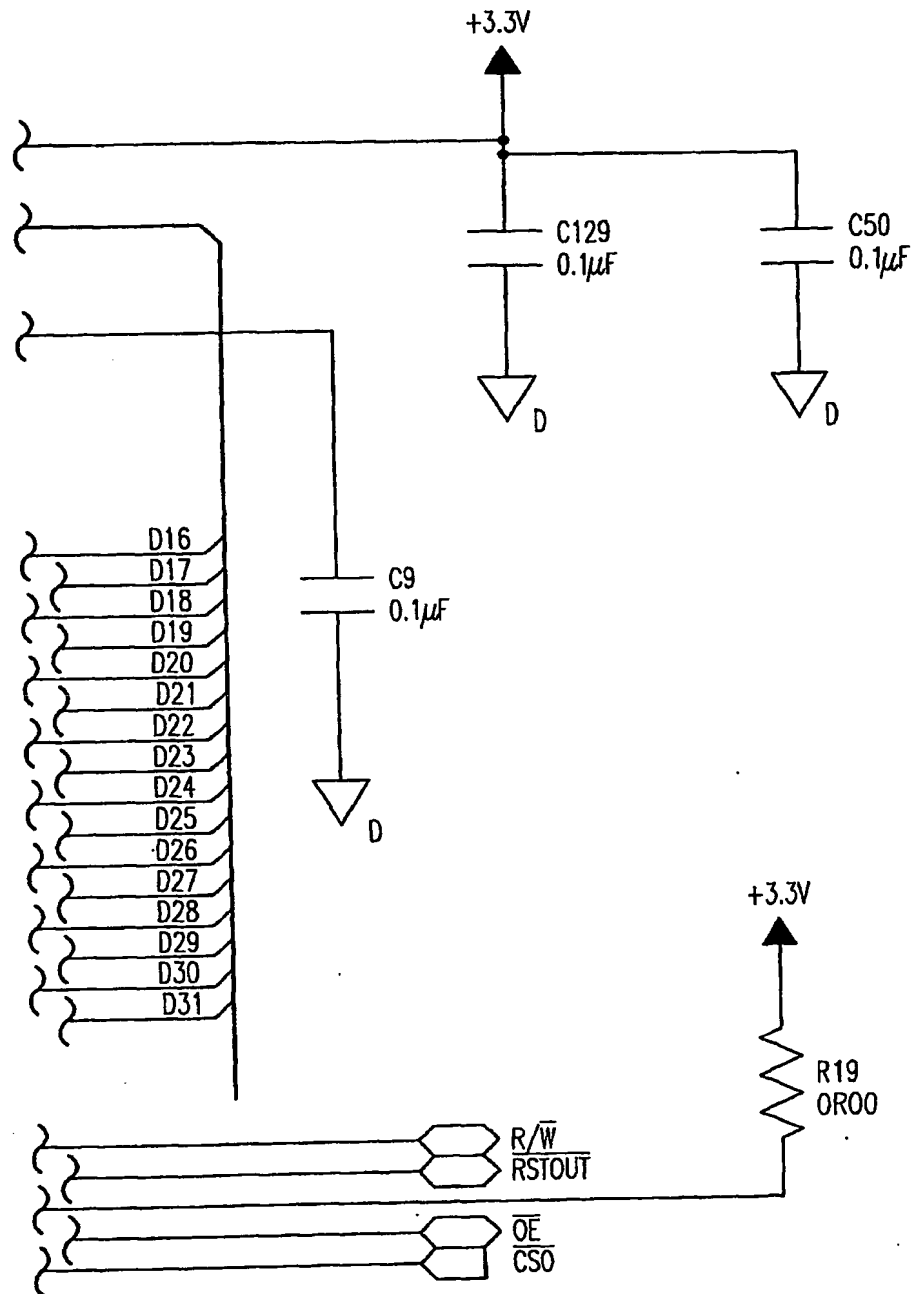


FIG.31E-2

MEMÓRIA FLASH SCH202-3

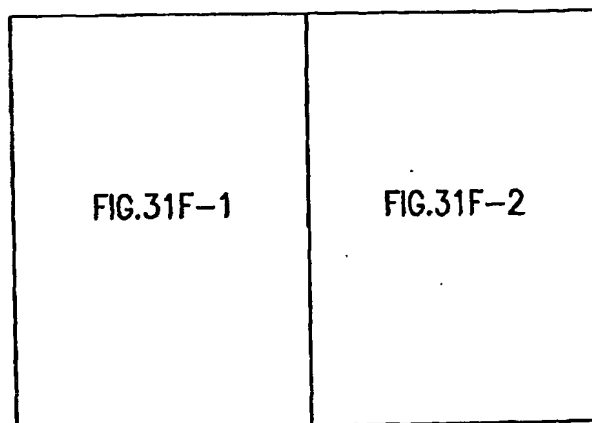


FIG.31F

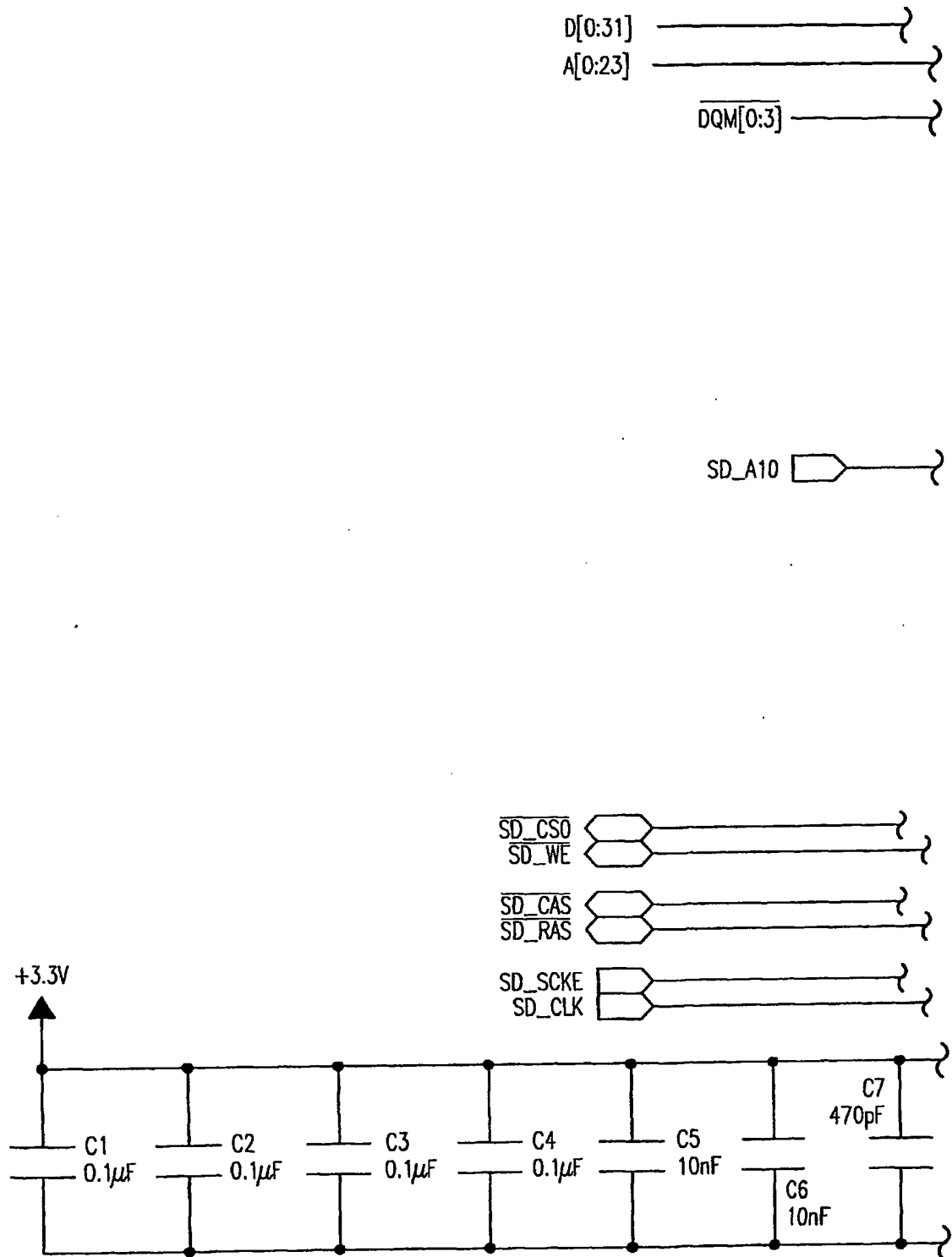


FIG.31F-1

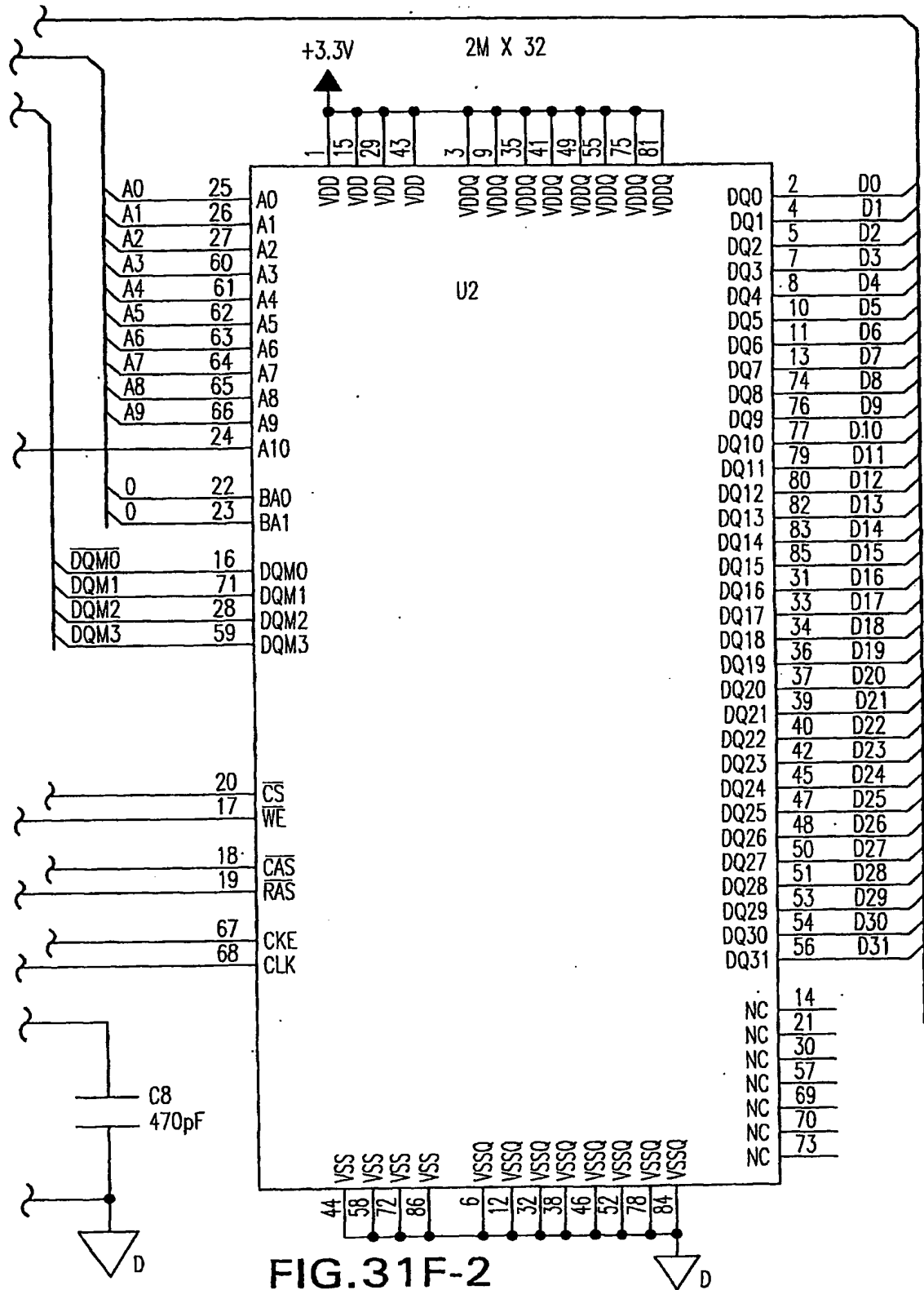


FIG.31F-2

AMPLIFICADORES OPERACIONAIS PCB202-3

FIG.31G-1	FIG.31G-2	FIG.31G-3
FIG.31G-4	FIG.31G-5	

FIG.31G

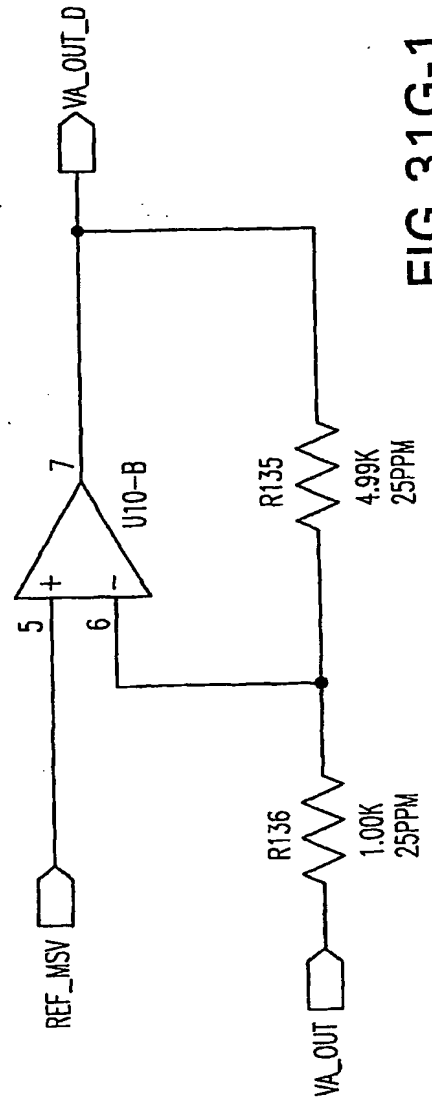
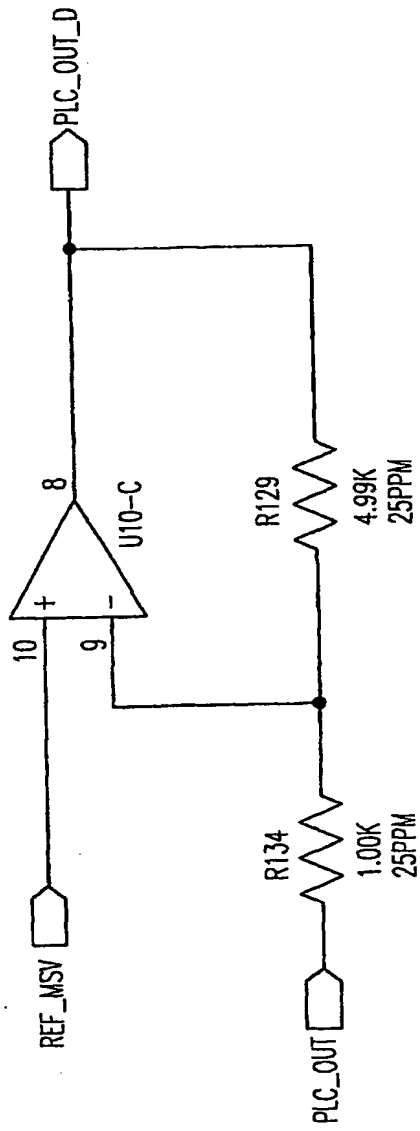


FIG.31G-1

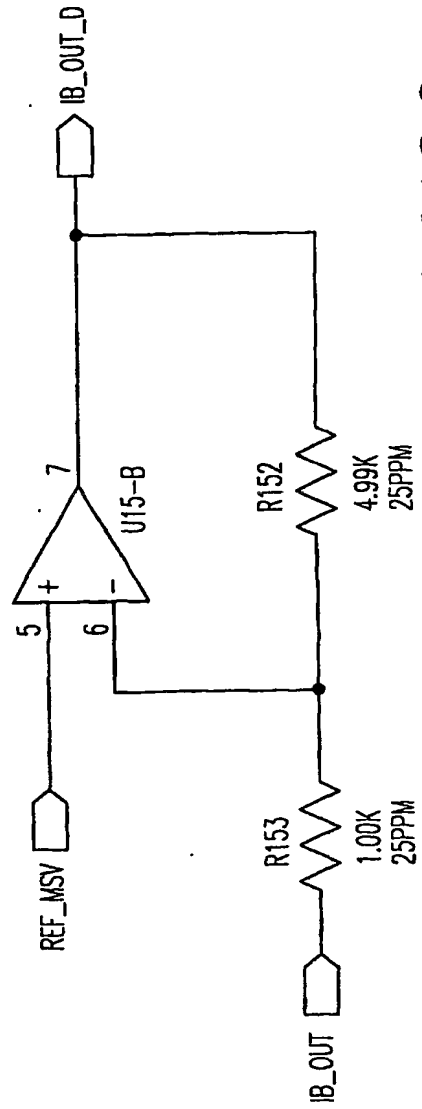
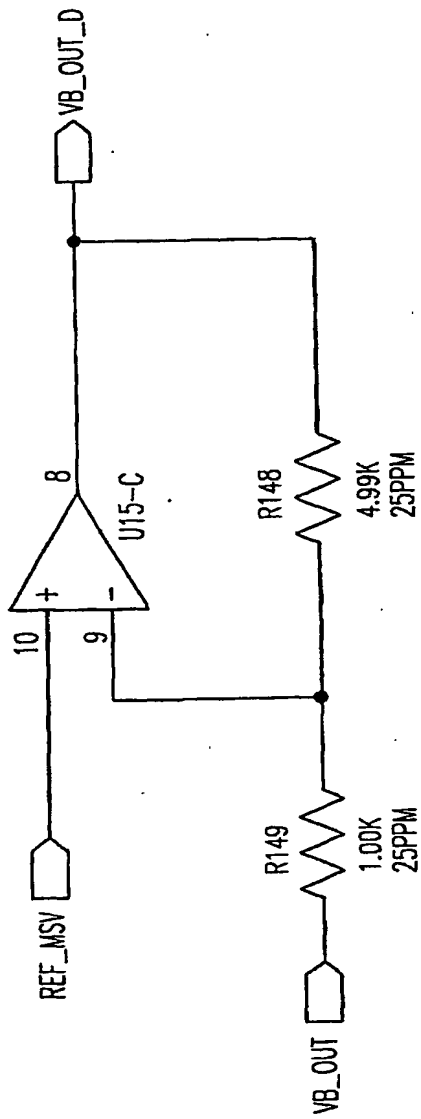
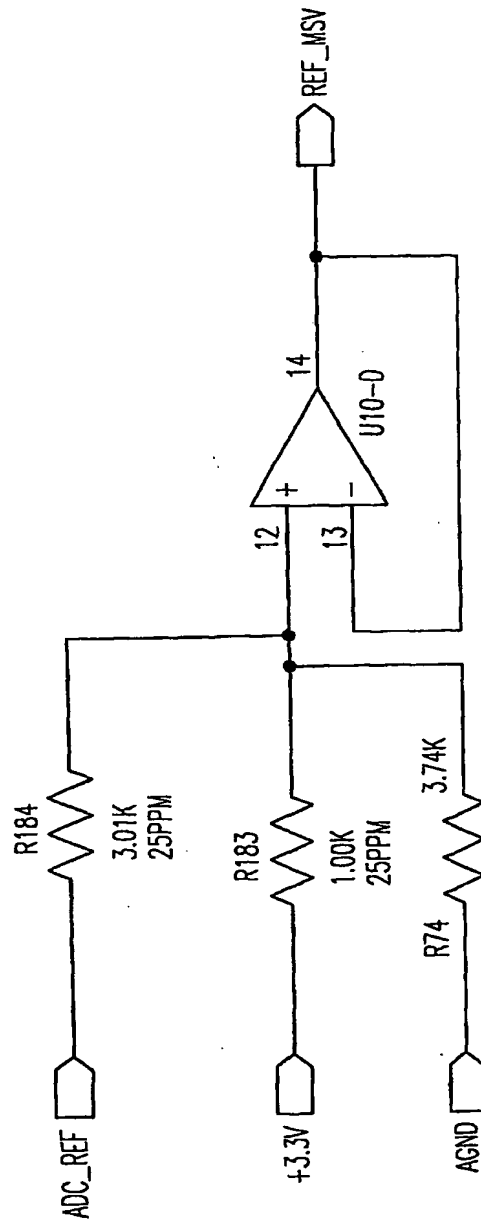


FIG.31G-2



REFERÊNCIA EXTERNA

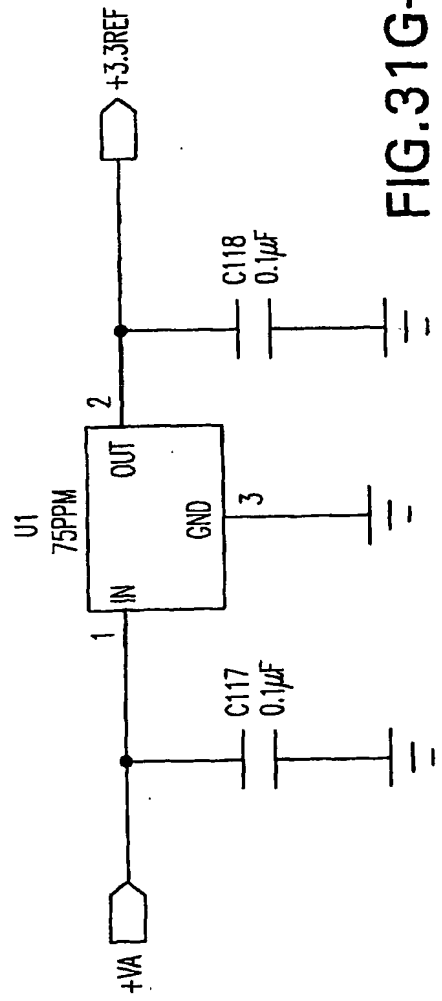


FIG.31G-3

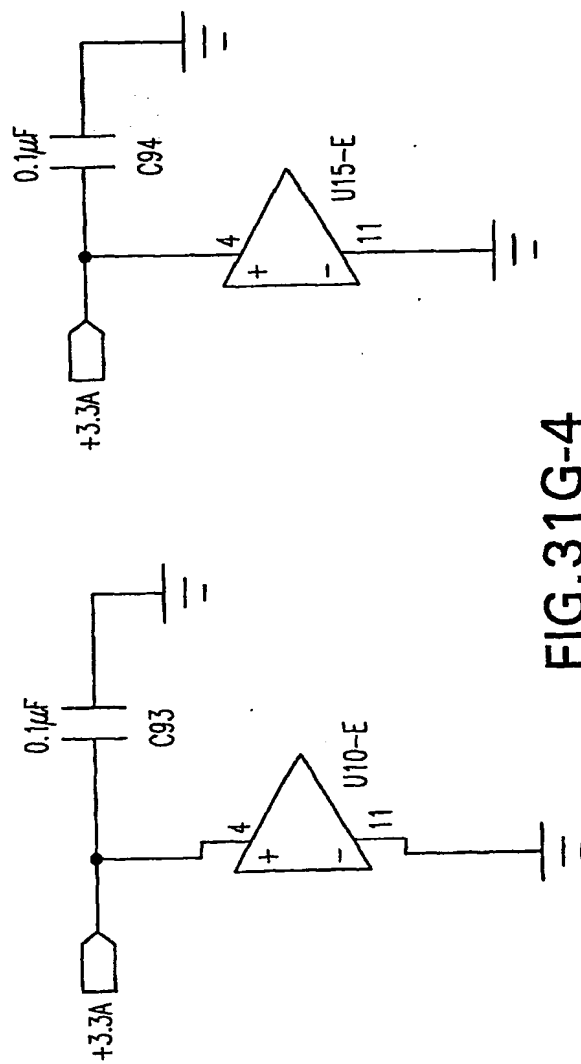
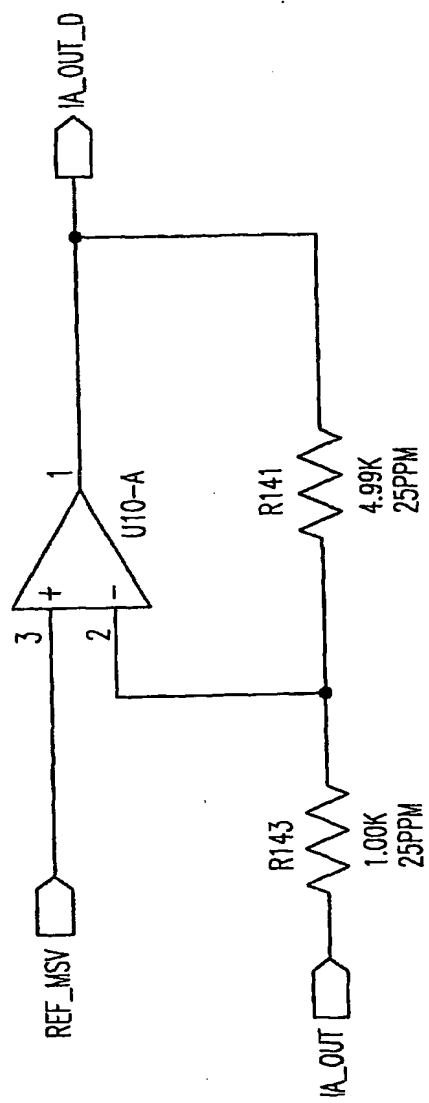


FIG.31G-4

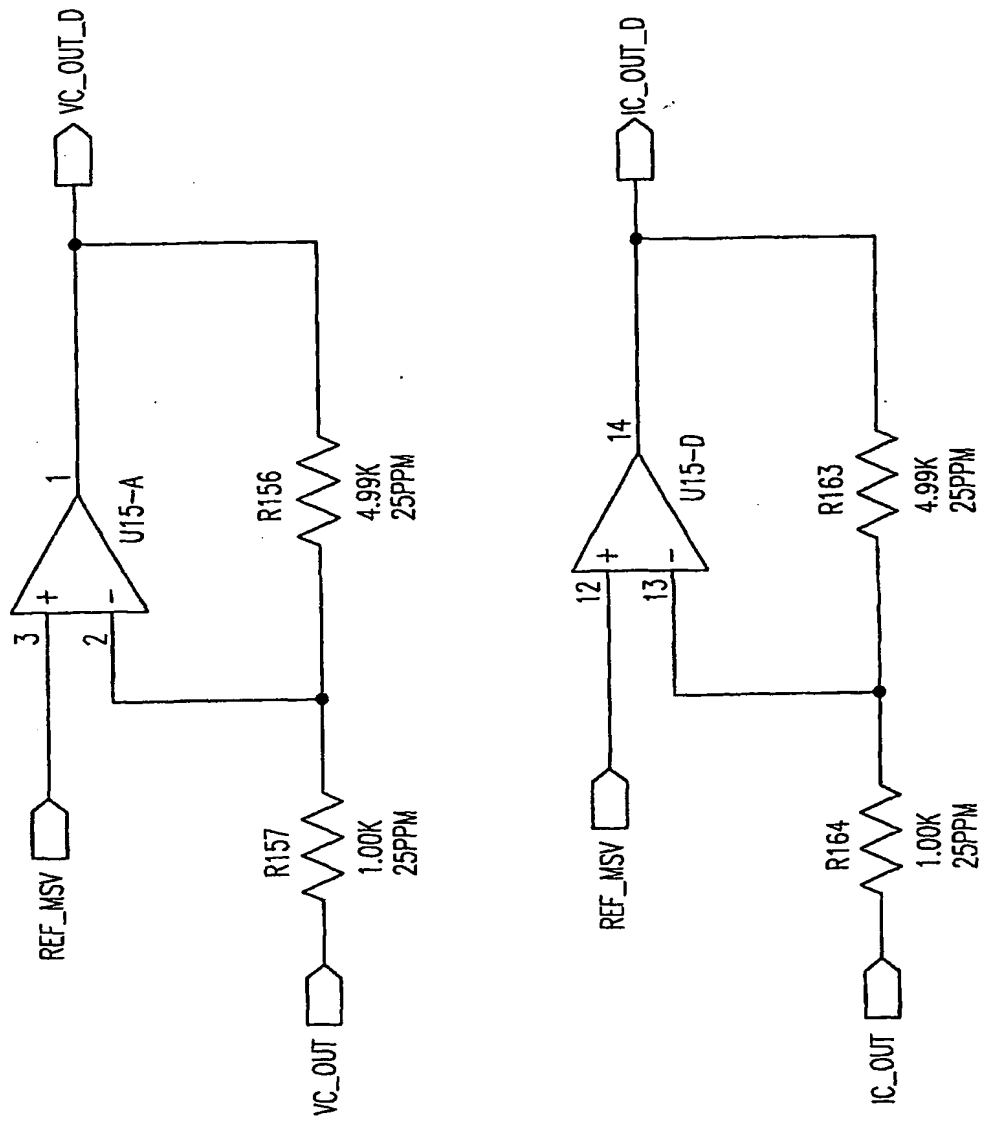


FIG.31G-5

MEDIDOR I PCB202-3

FIG.31H-1	FIG.31H-2	FIG.31H-3
FIG.31H-4	FIG.31H-5	FIG.31H-6
FIG.31H-7	FIG.31H-8	FIG.31H-9

FIG.31H

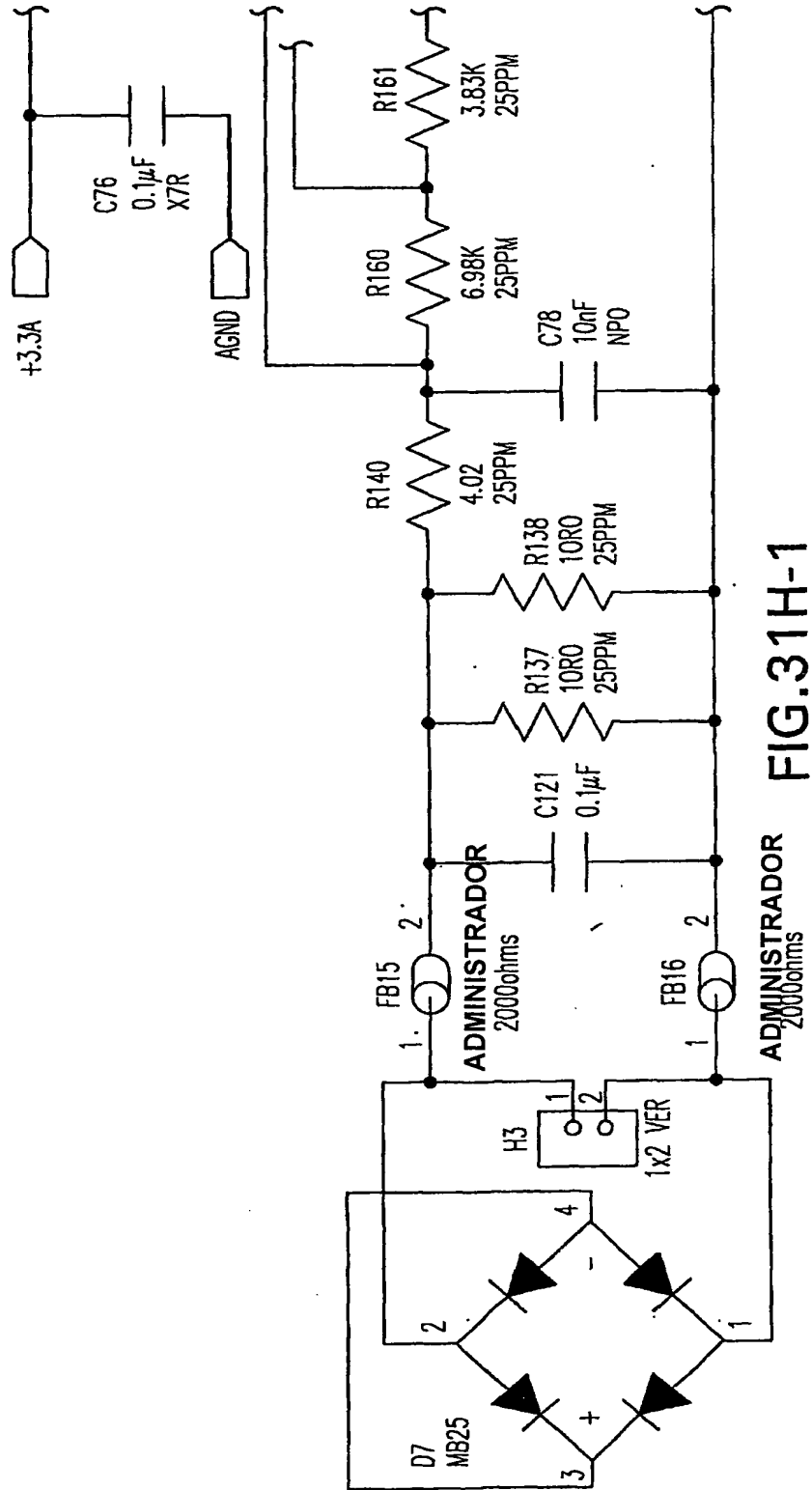
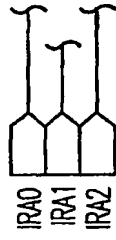


FIG.31H-1

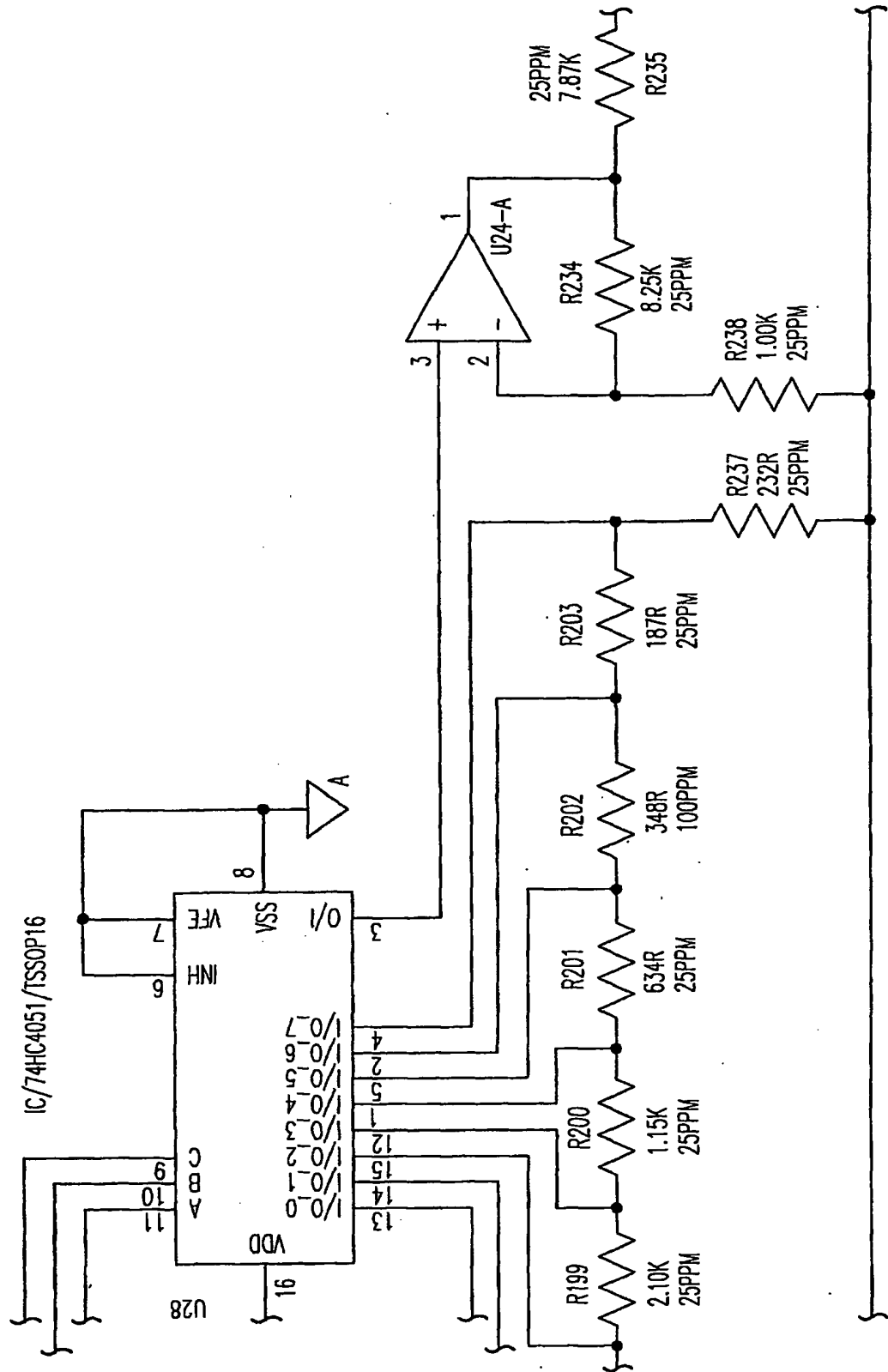


FIG. 31H-2

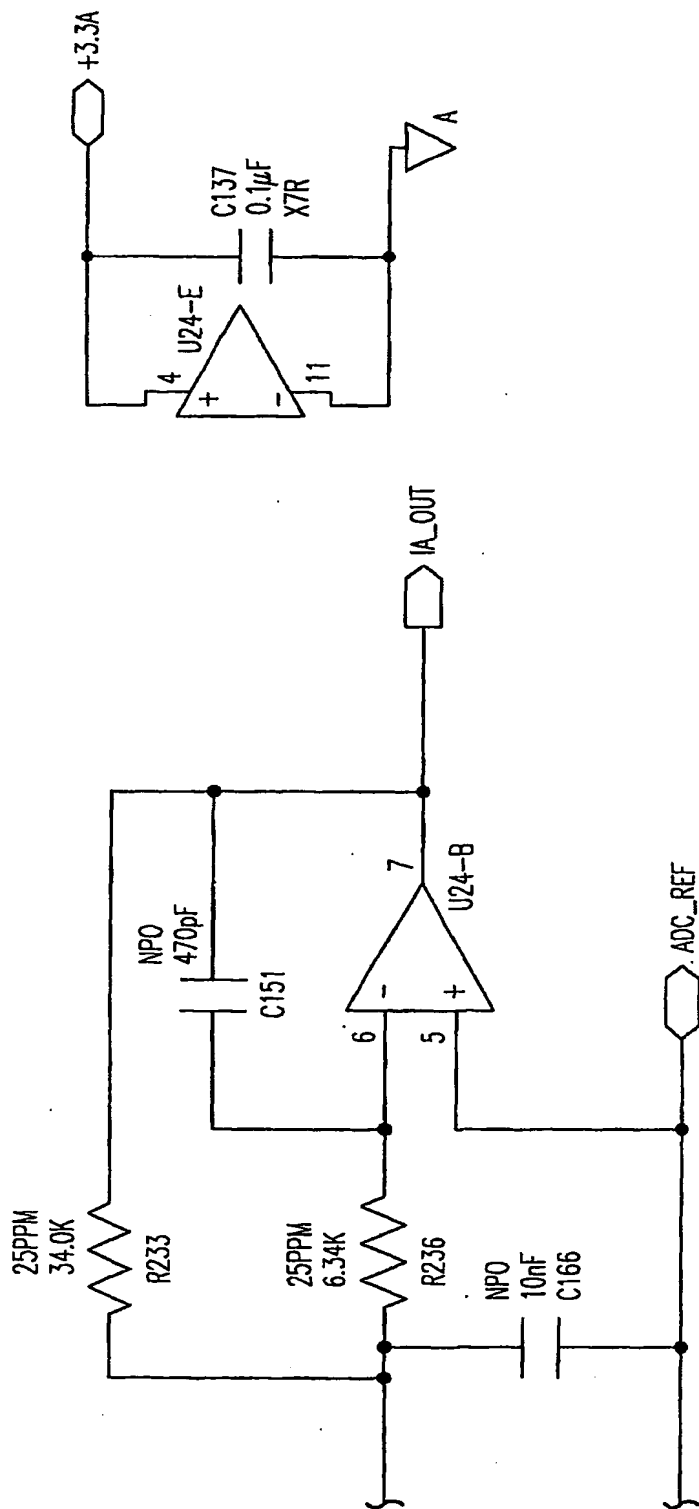


FIG. 31H-3

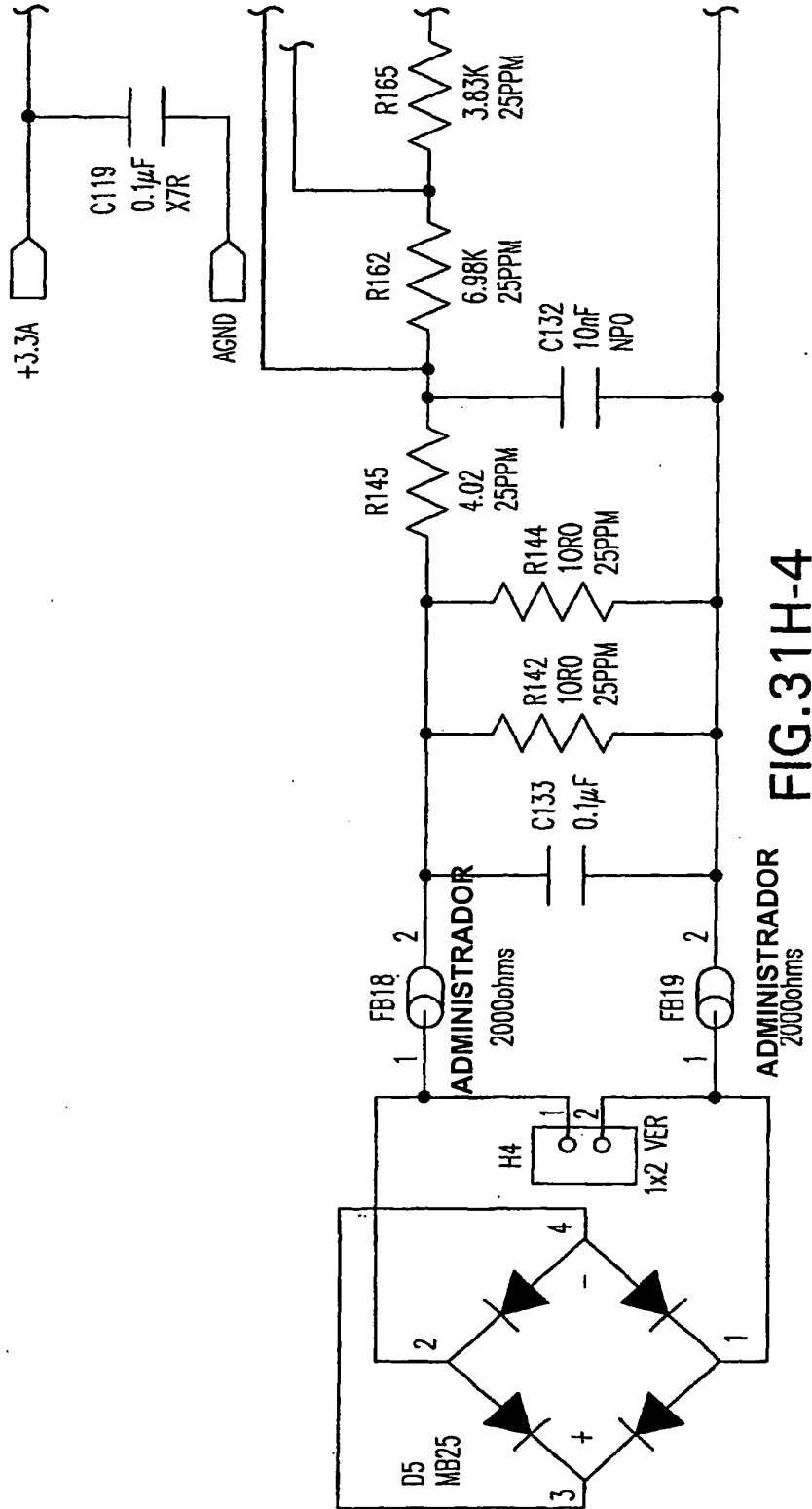
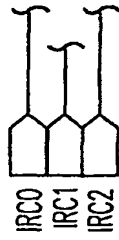


FIG. 31H-4

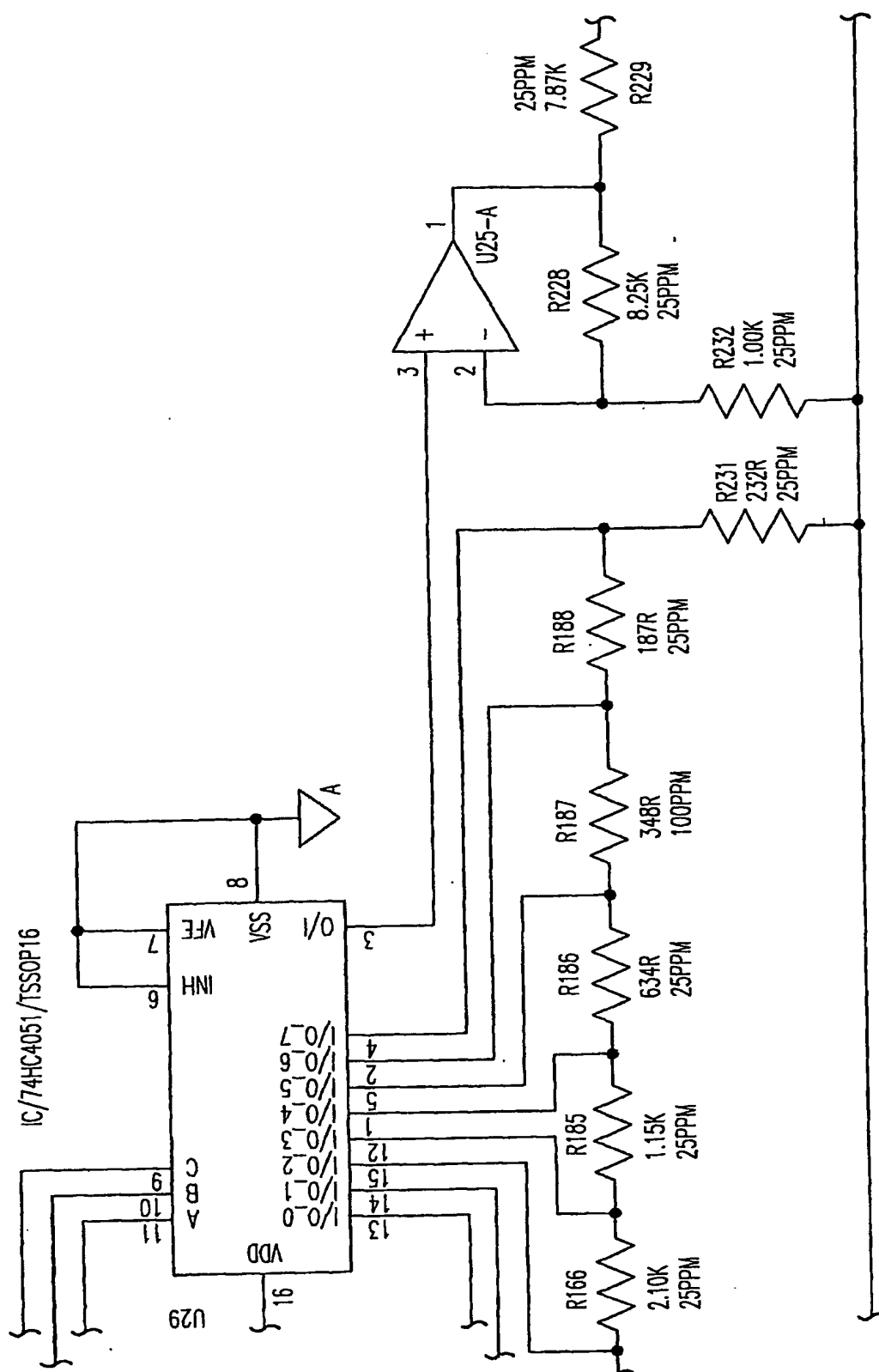


FIG. 31H-5

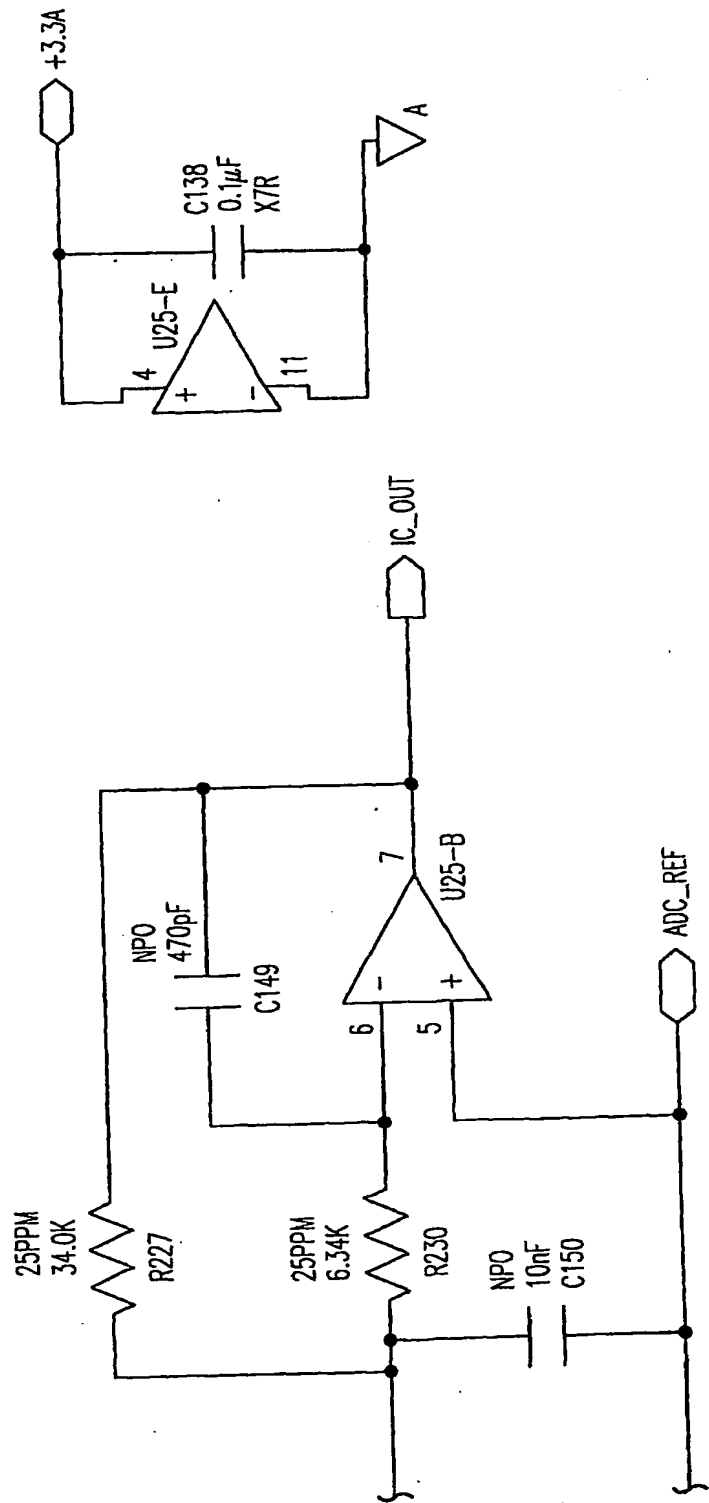


FIG. 31H-6

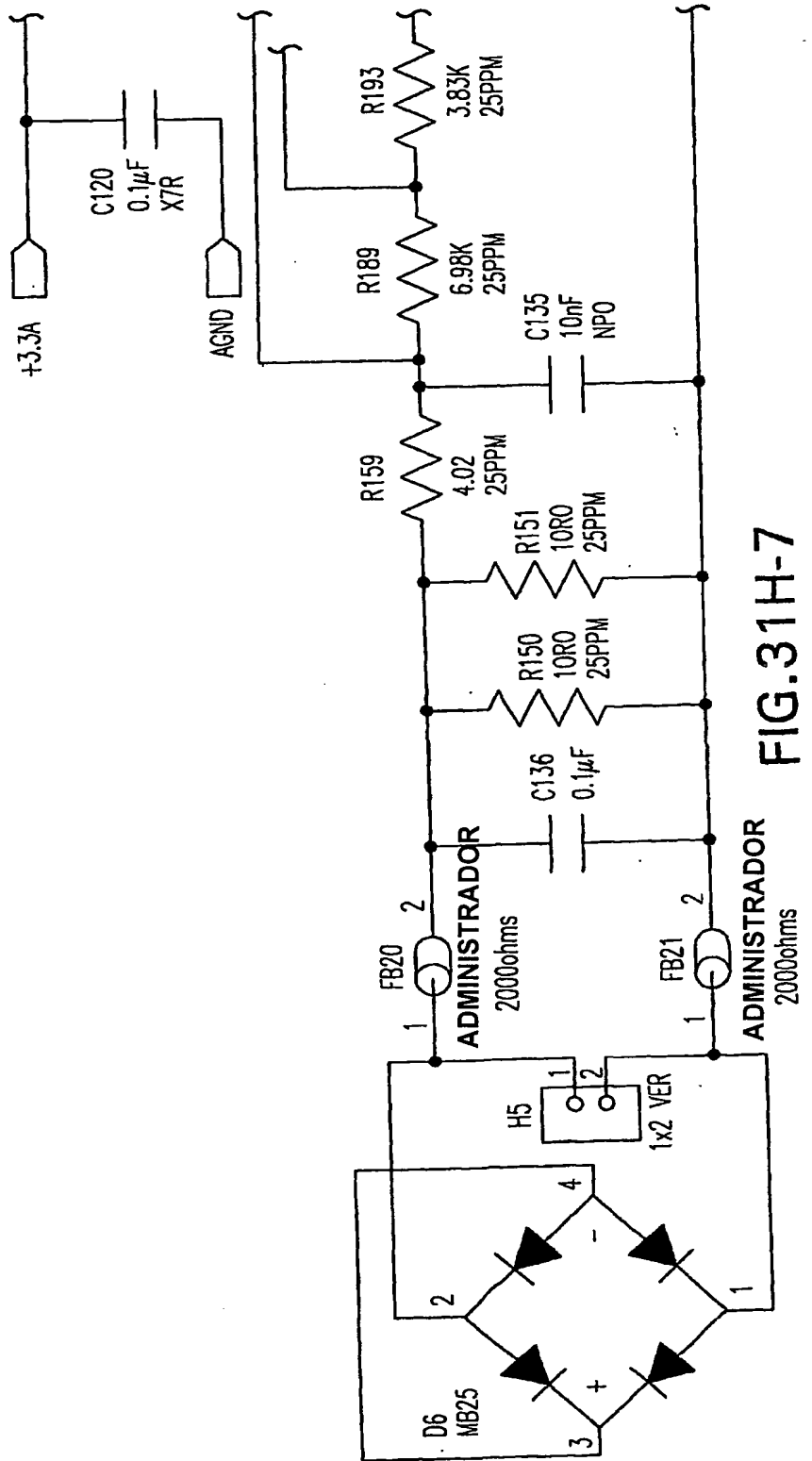
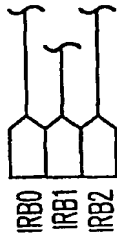


FIG. 31H-7

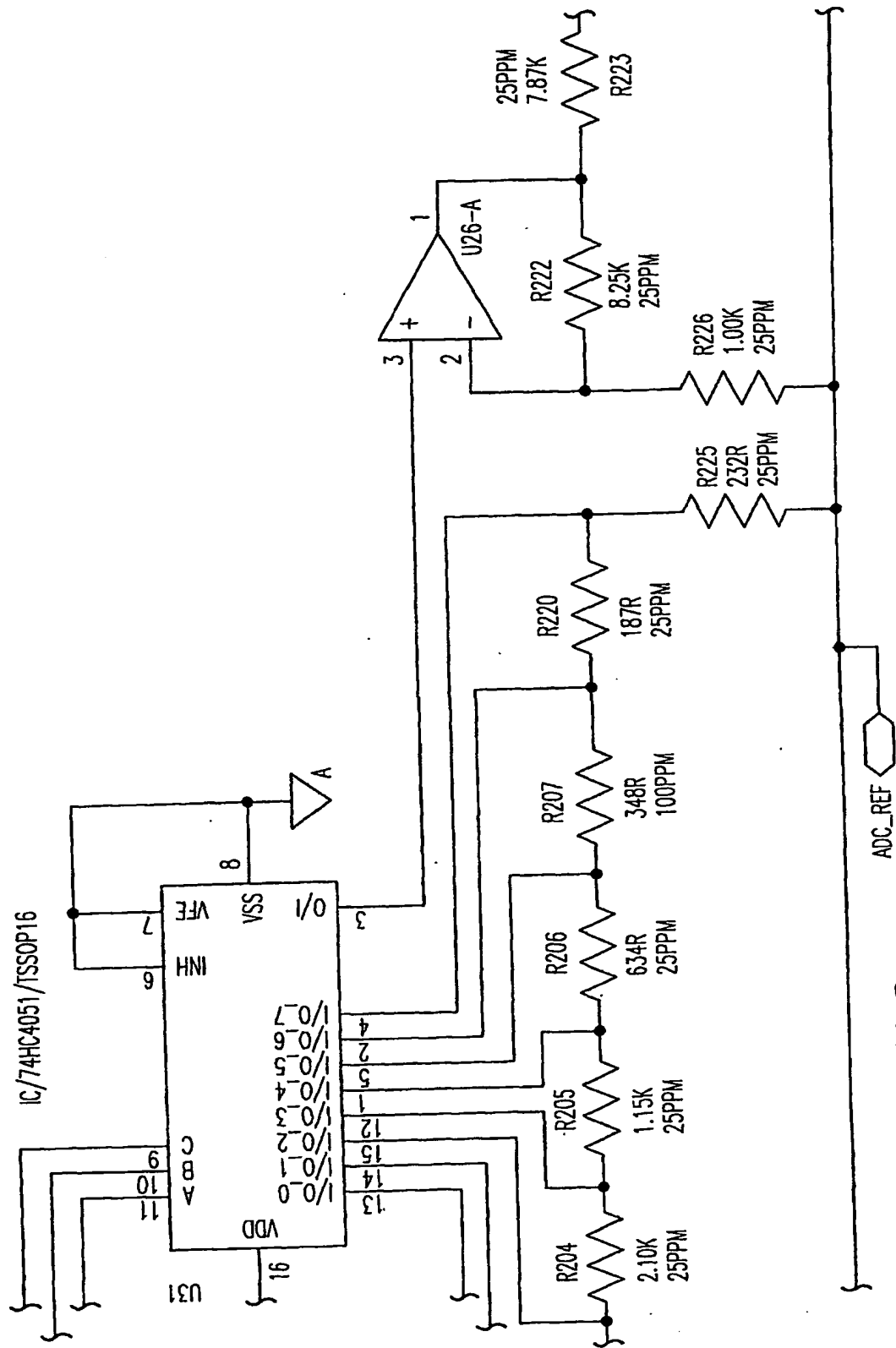


FIG.31H-8

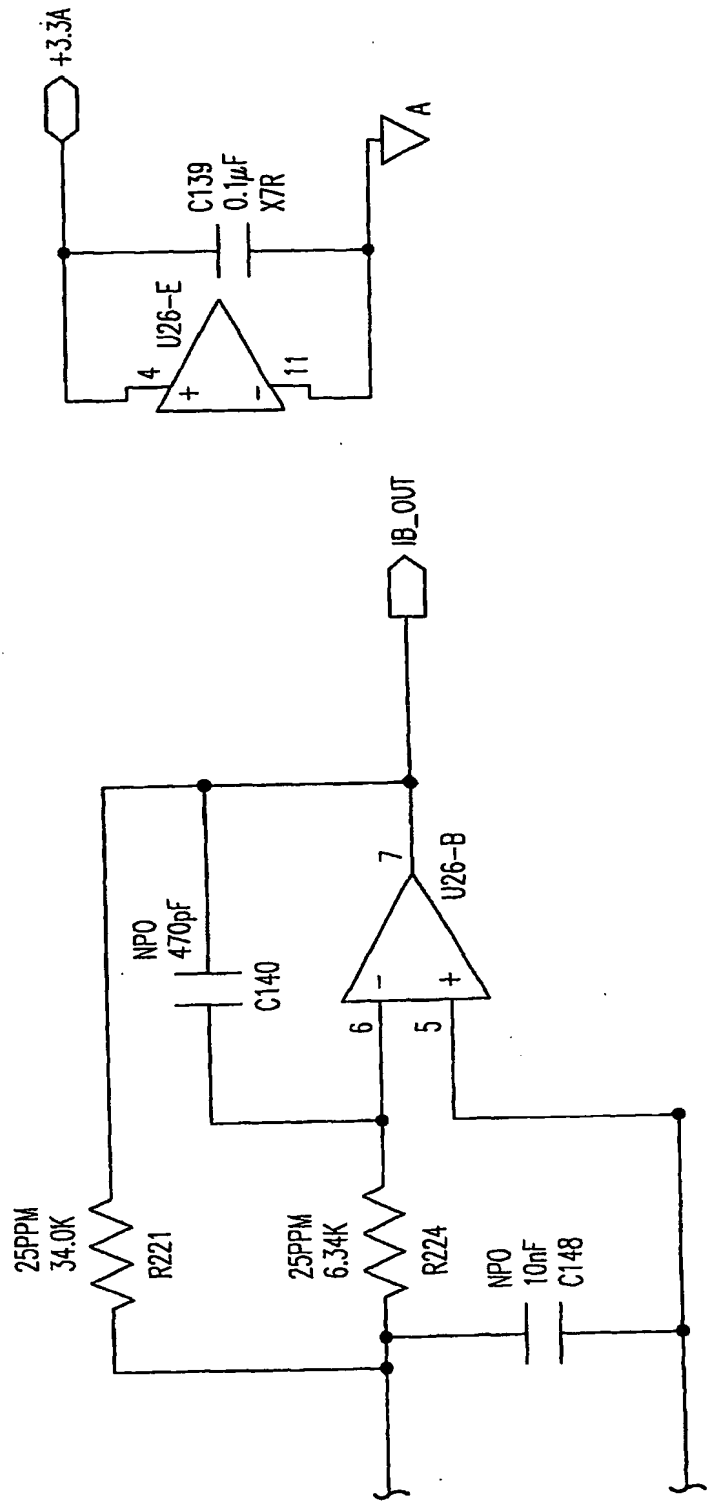


FIG. 31H-9

MEDIDOR V SCH202-3

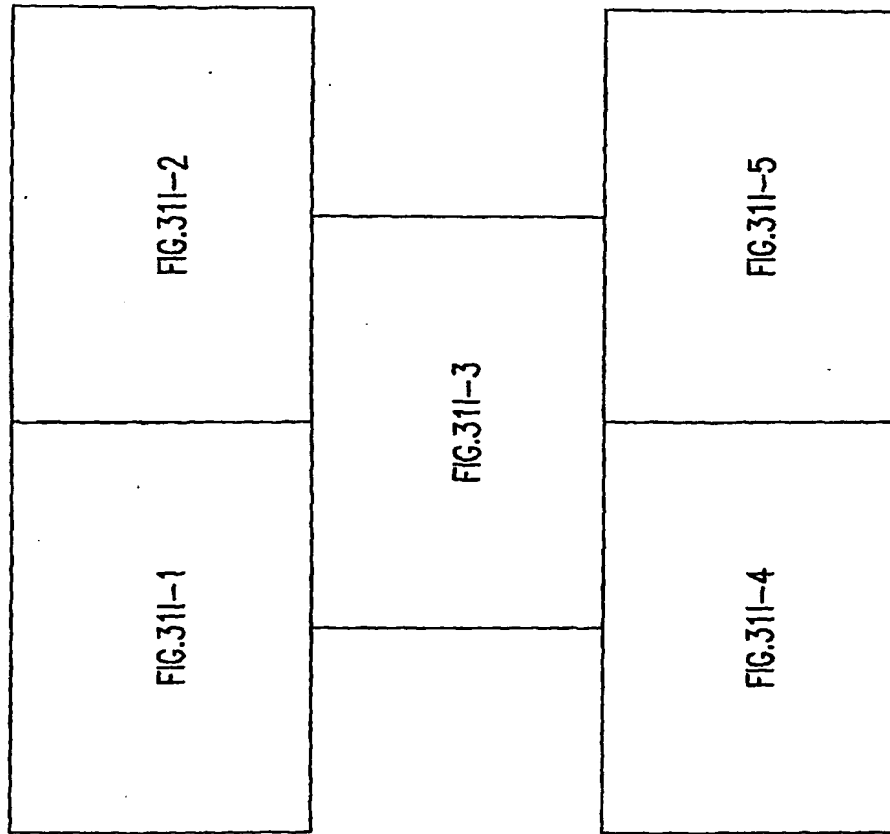


FIG.31I

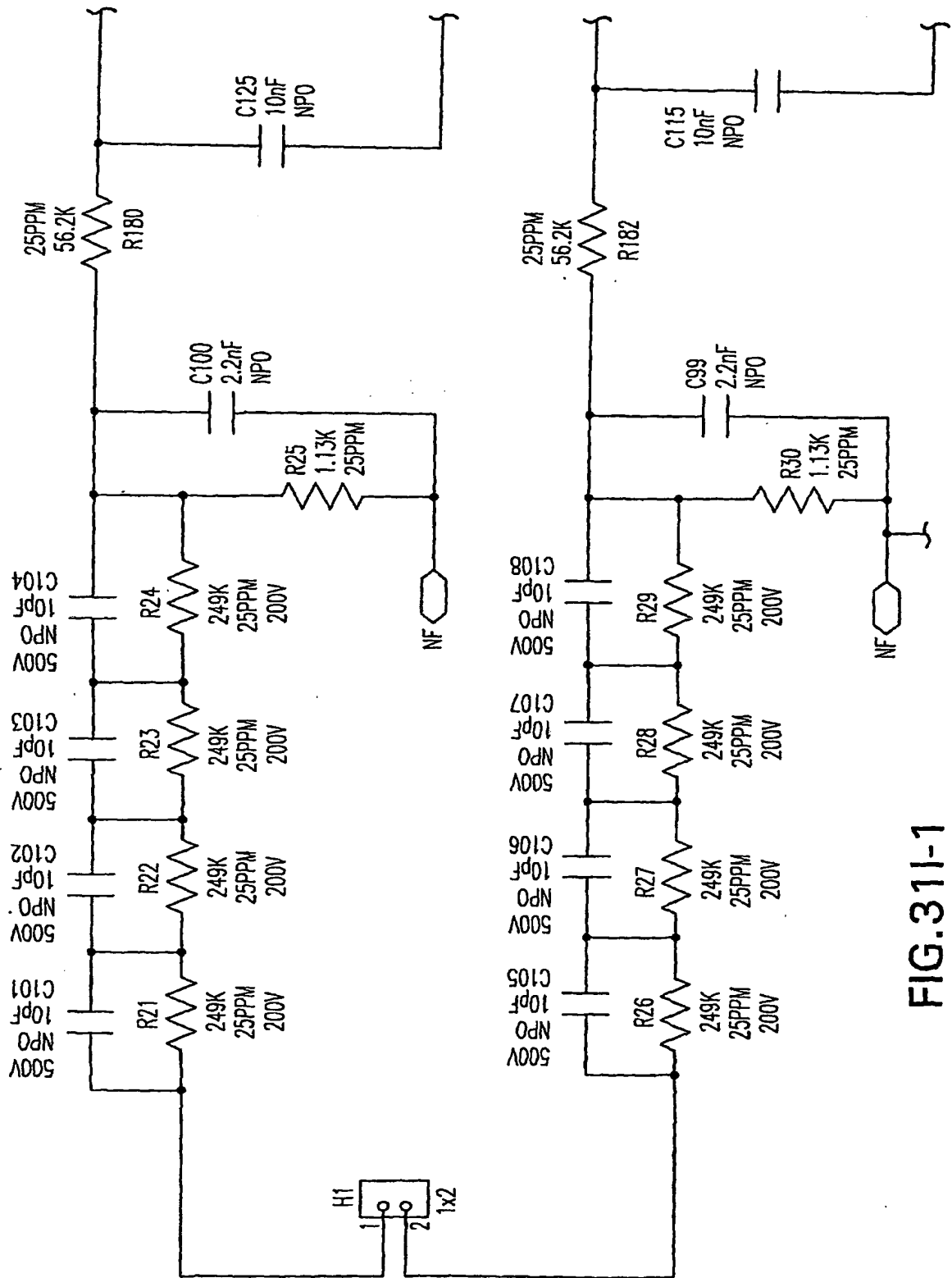


FIG.311-1

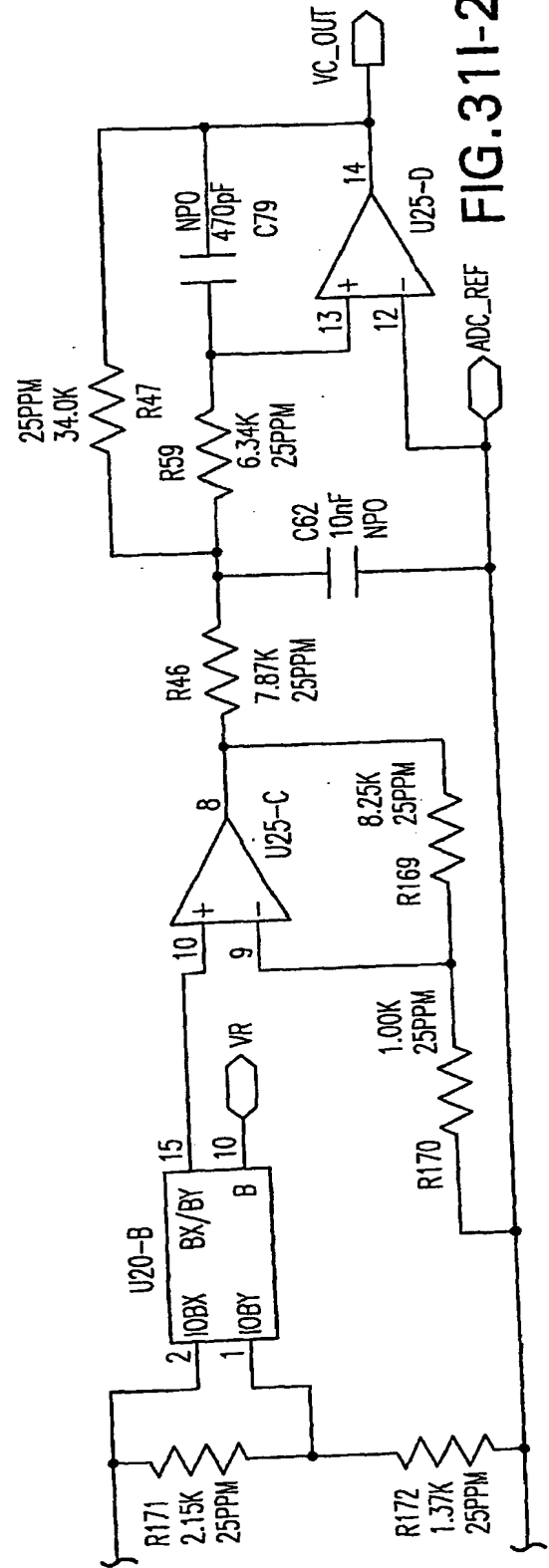
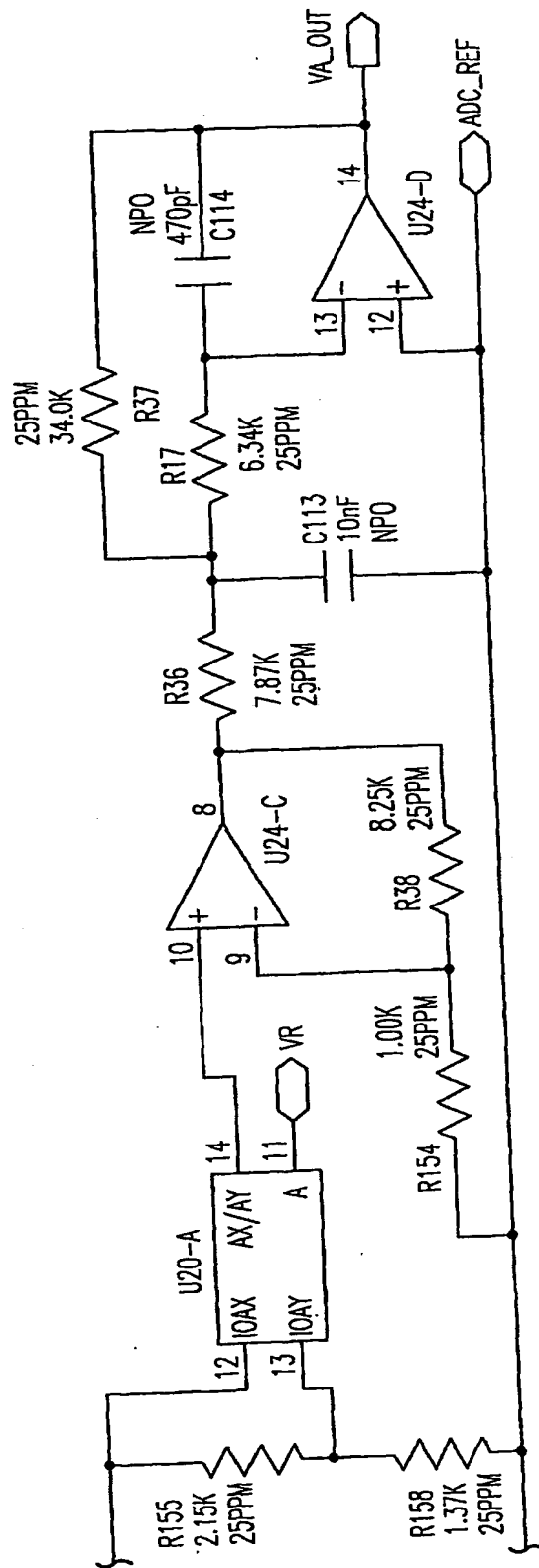


FIG. 31-2

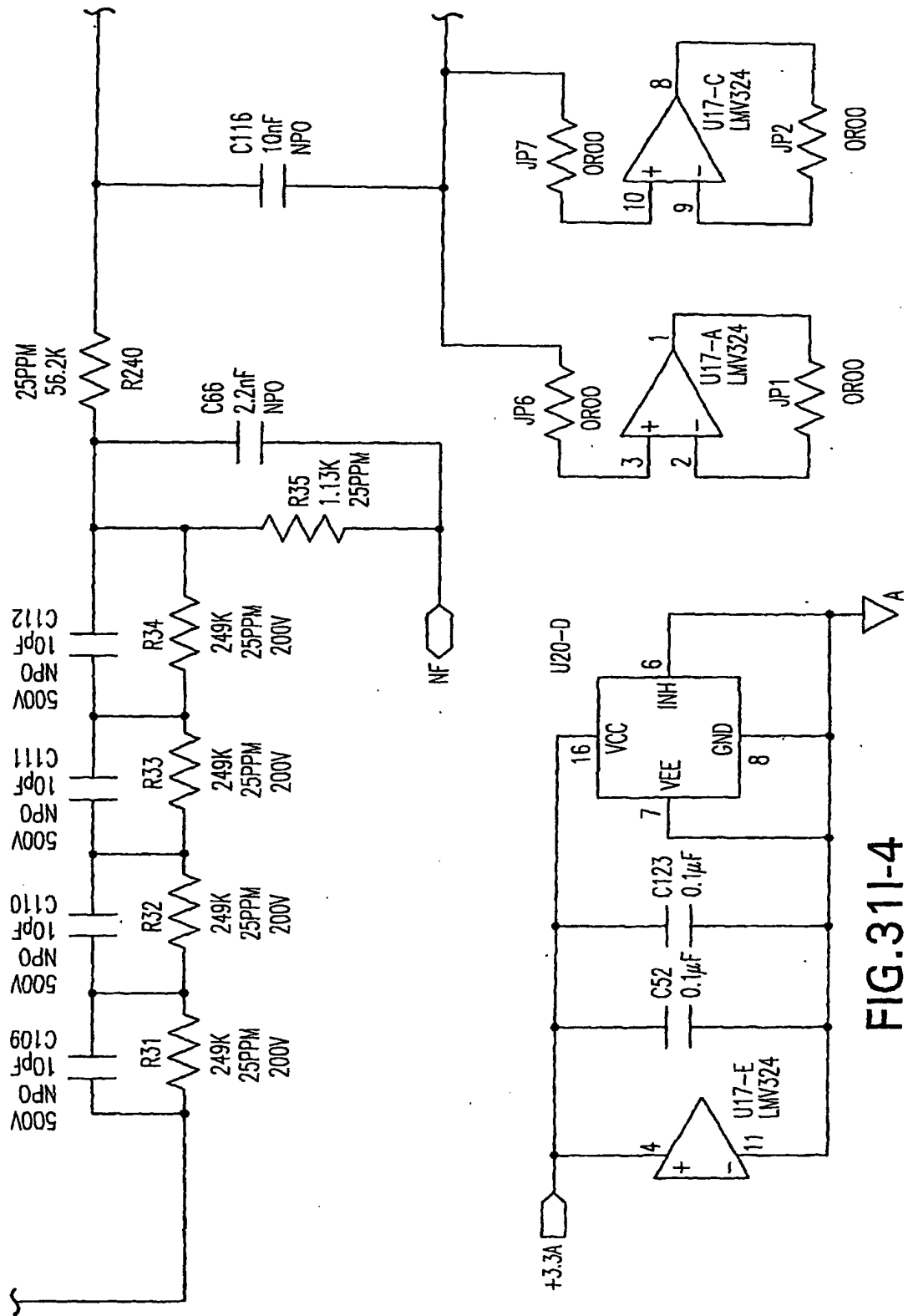


FIG. 311-4

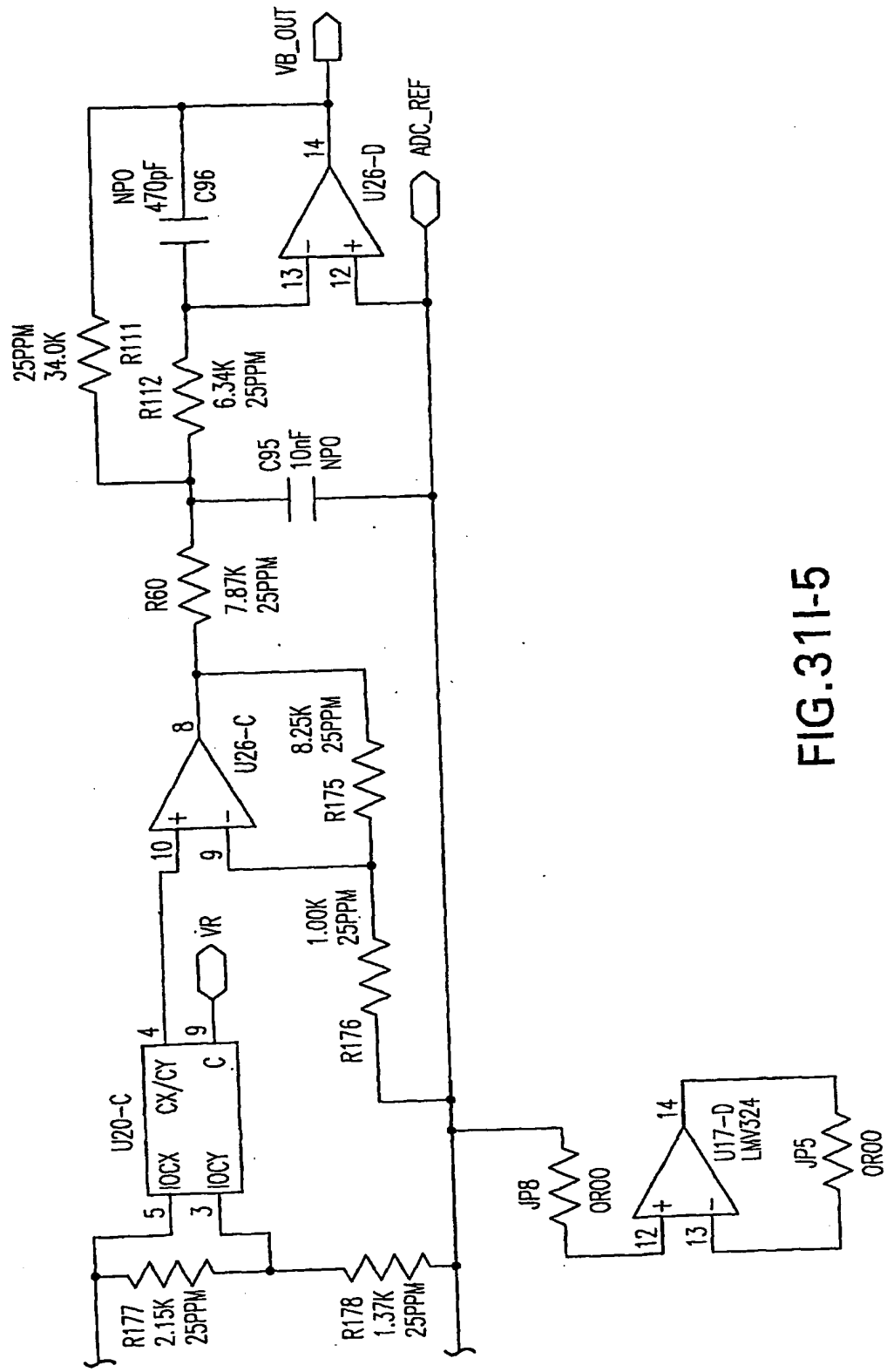


FIG.311-5

PLC RCV PCB202-3

FIG.31J-1	FIG.31J-2	FIG.31J-3
FIG.31J-4	FIG.31J-5	FIG.31J-6

FIG.31J

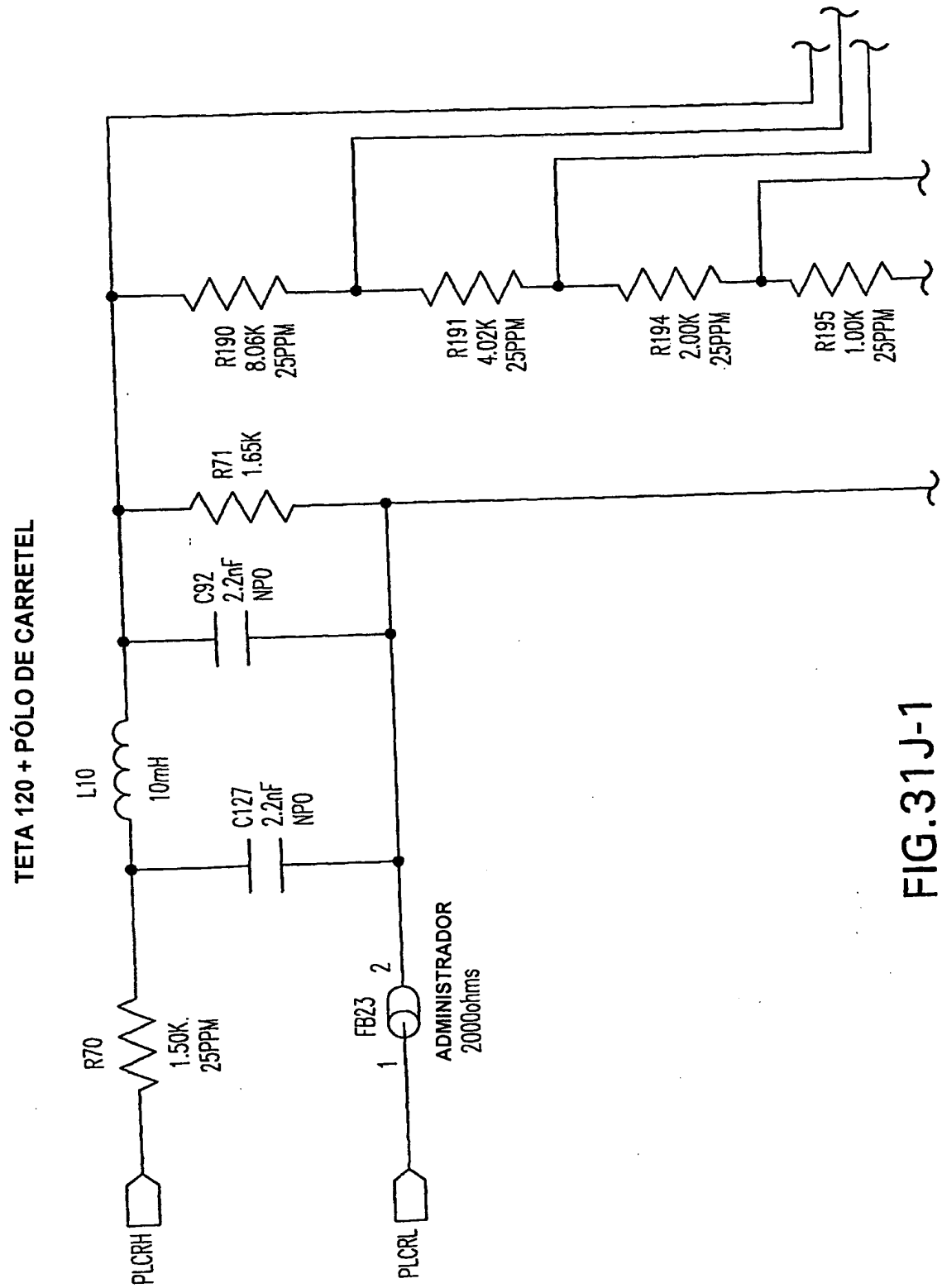
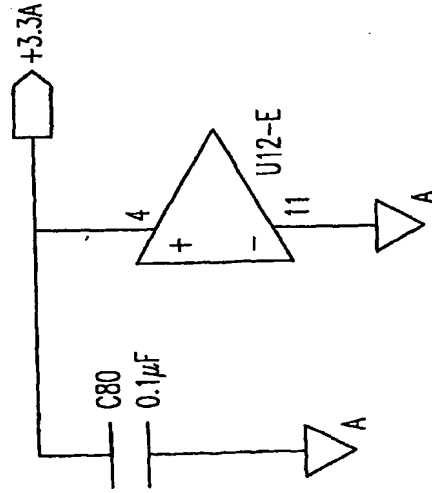
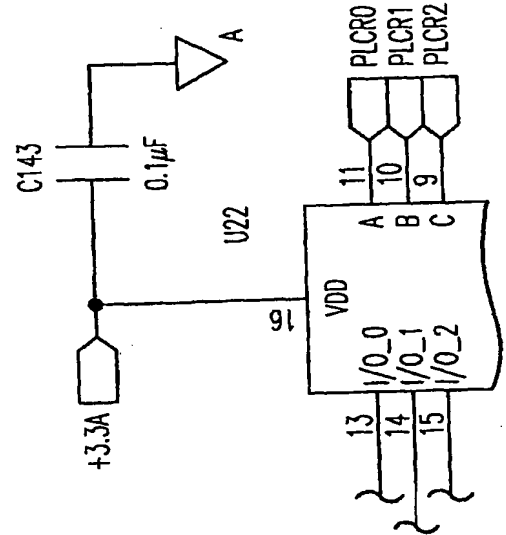


FIG.31J-1



IC/74HC4051/TSSQP16



160 TETA

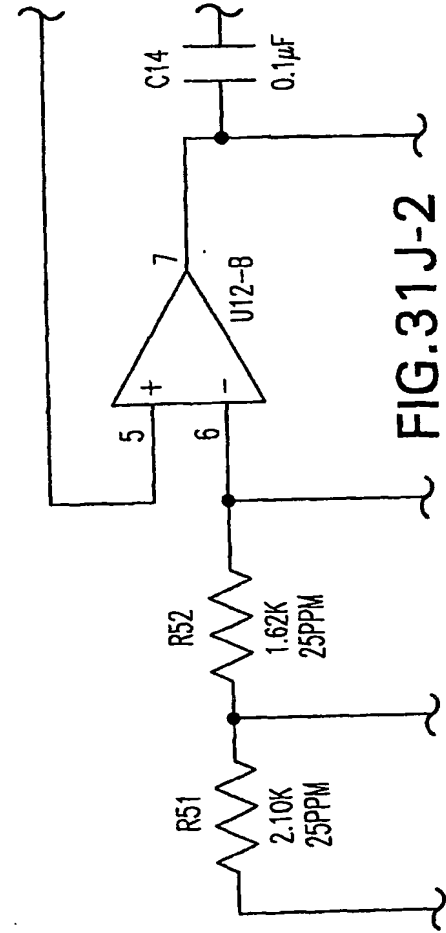
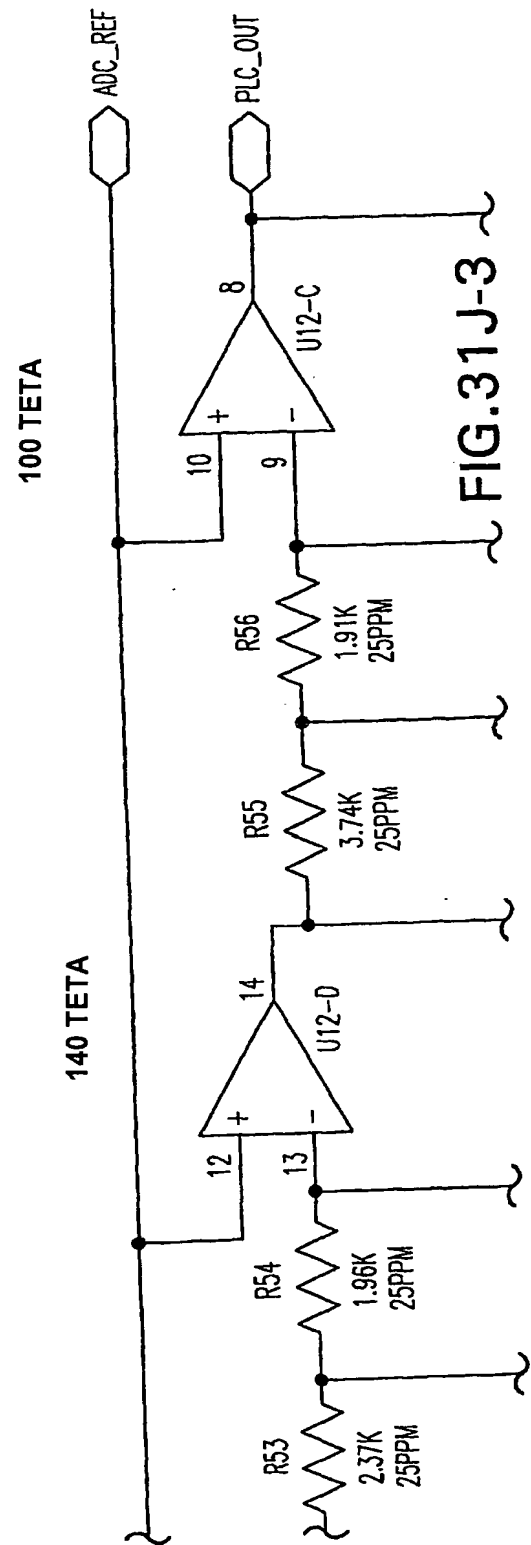


FIG.31J-2



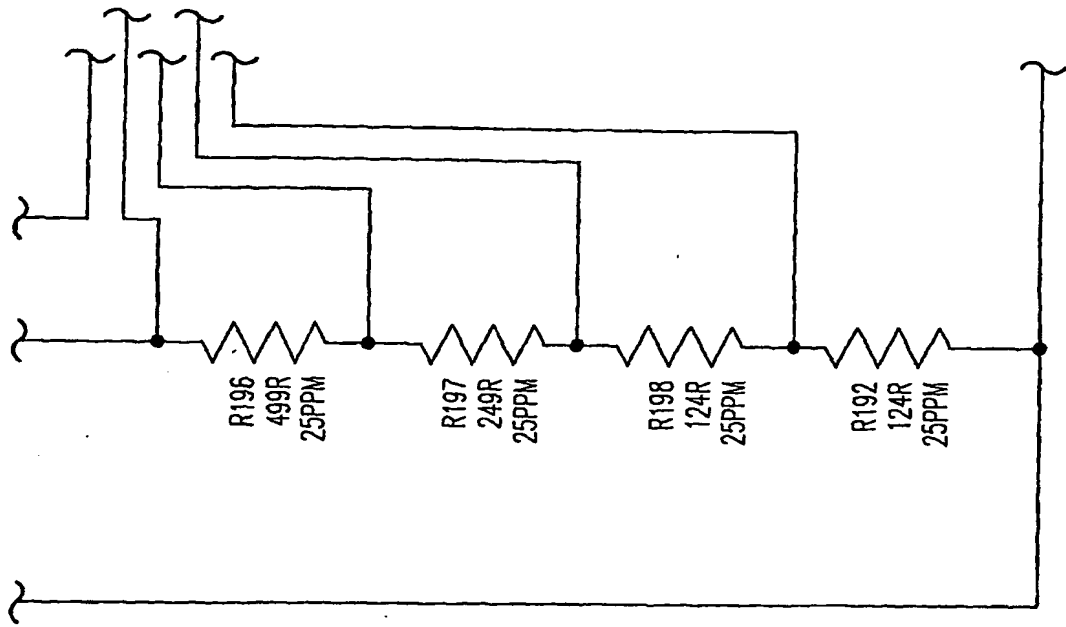


FIG.31J-4

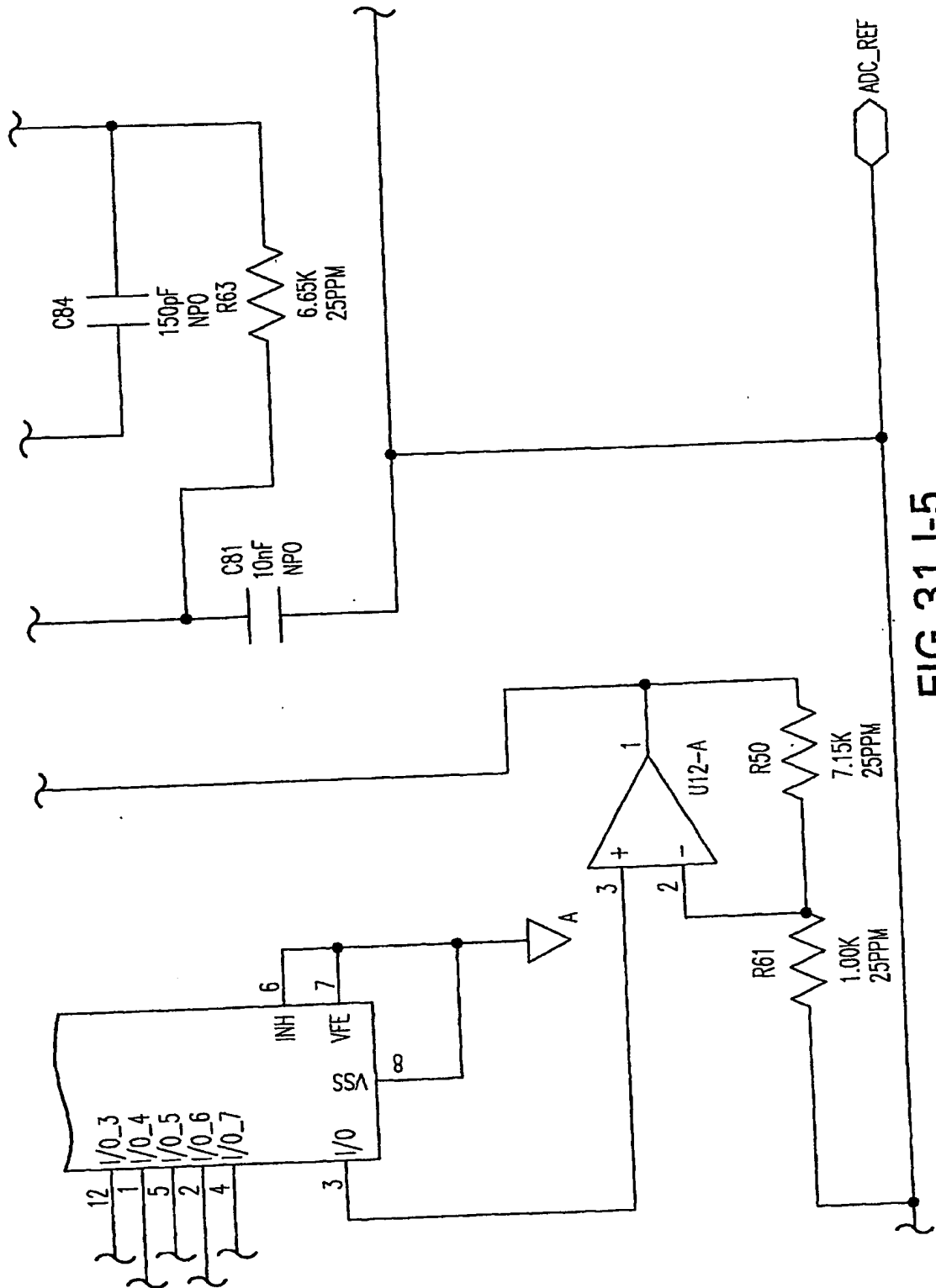


FIG.31J-5

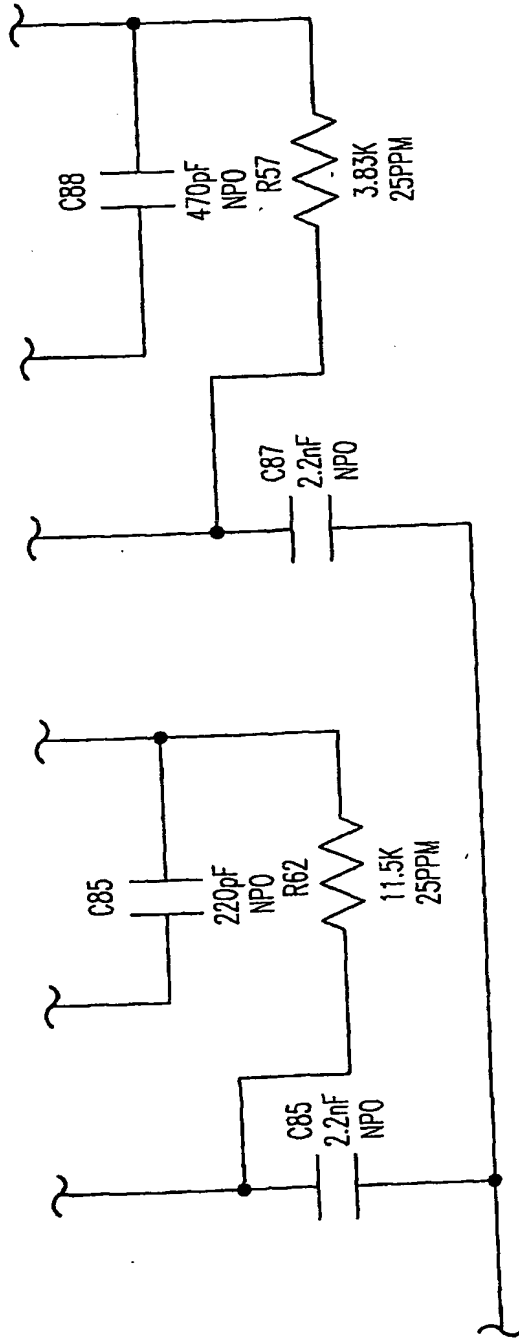


FIG.31J-6

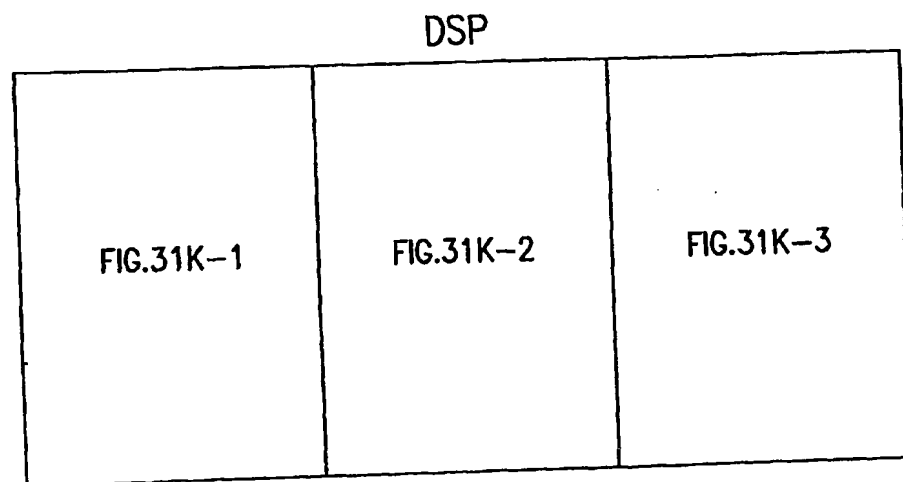


FIG.31K

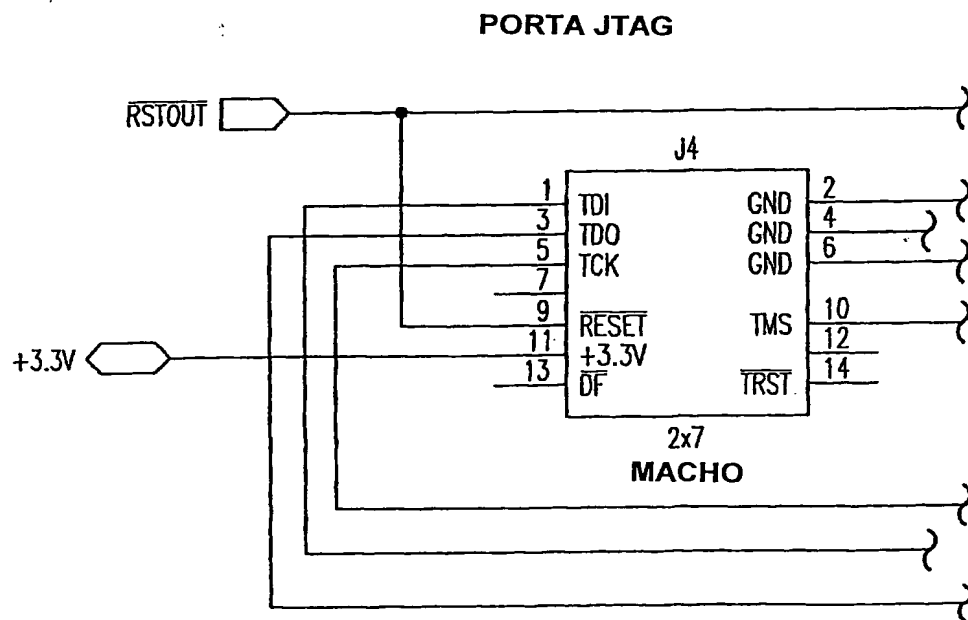
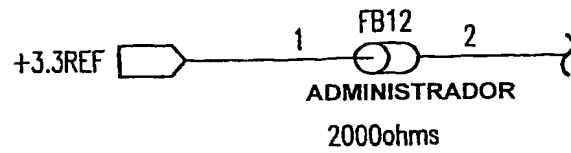


FIG.31K-1

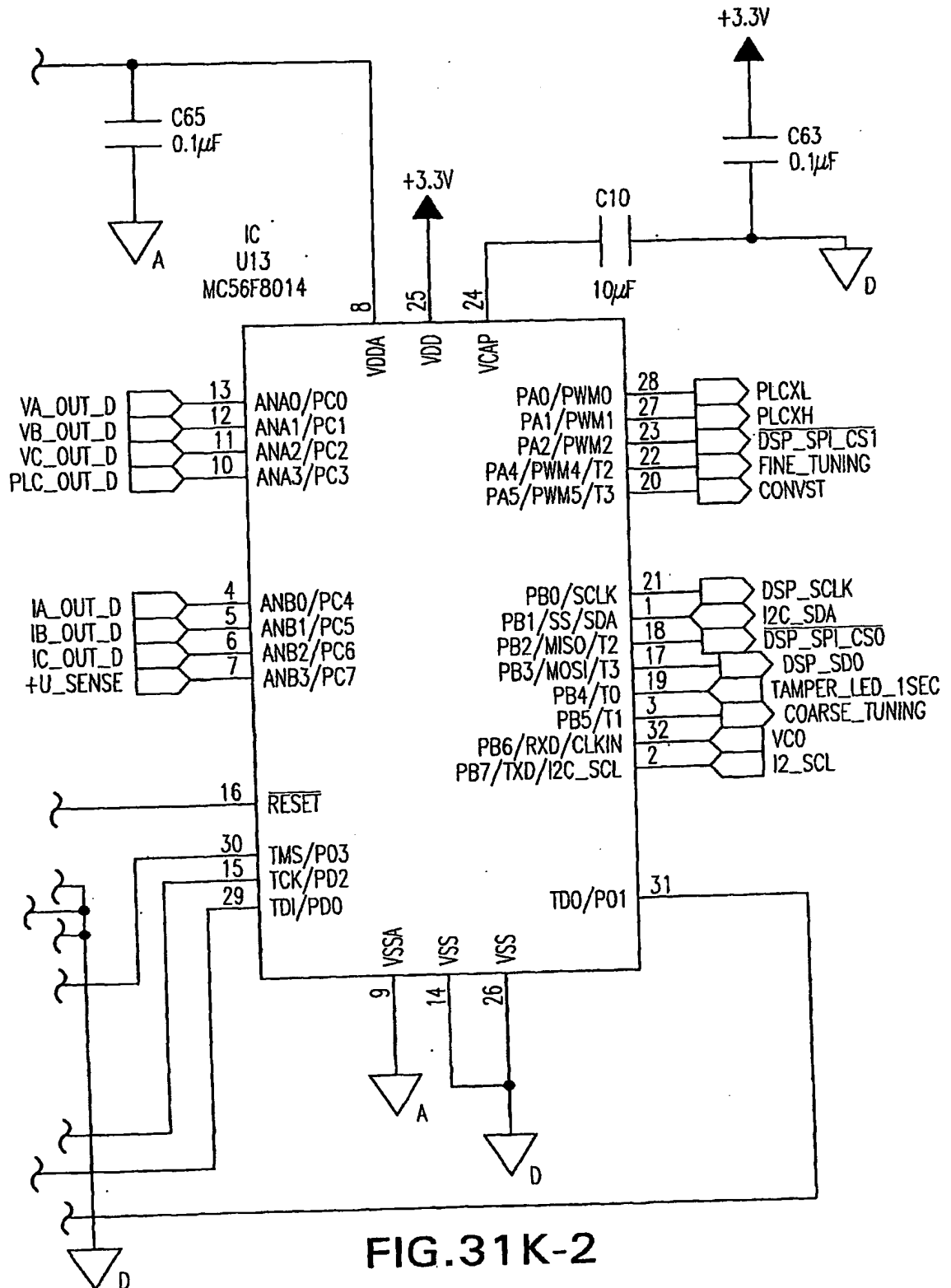


FIG.31K-2

CONECTADO A ADC DE DSP PARA SENTIR
NÍVEIS DE FONTE DE ENERGIA DE ENTRADA

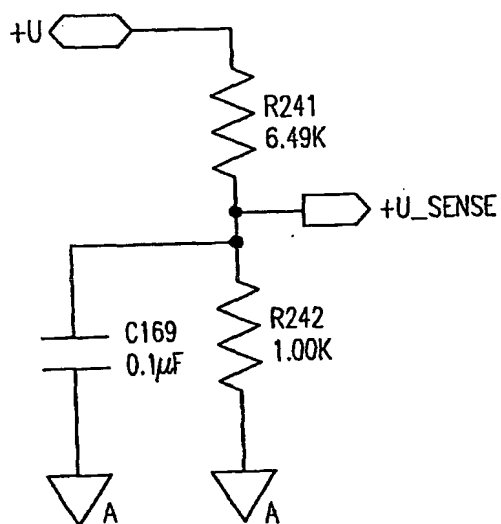
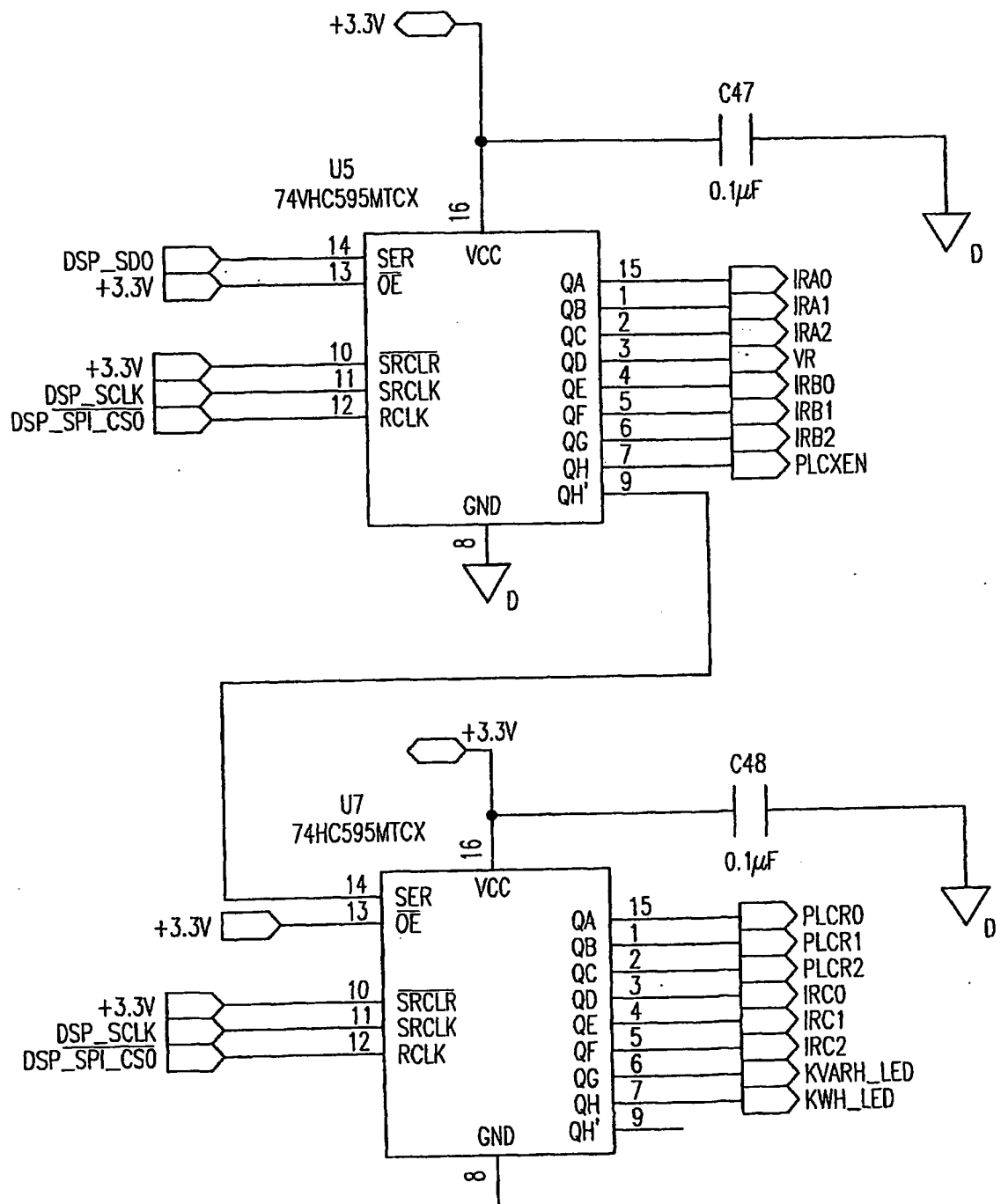


FIG.31K-3



IO EXP

FIG.31L

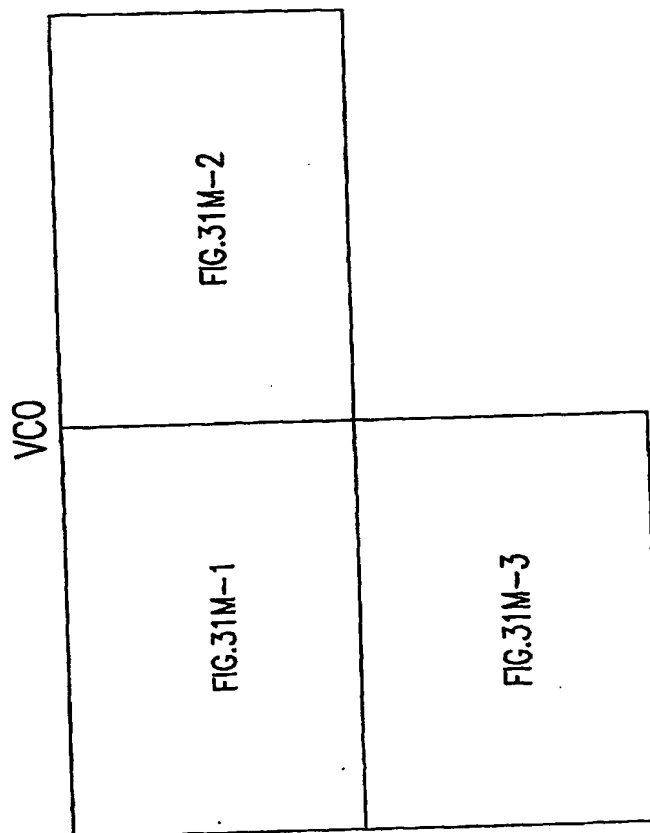


FIG.31M

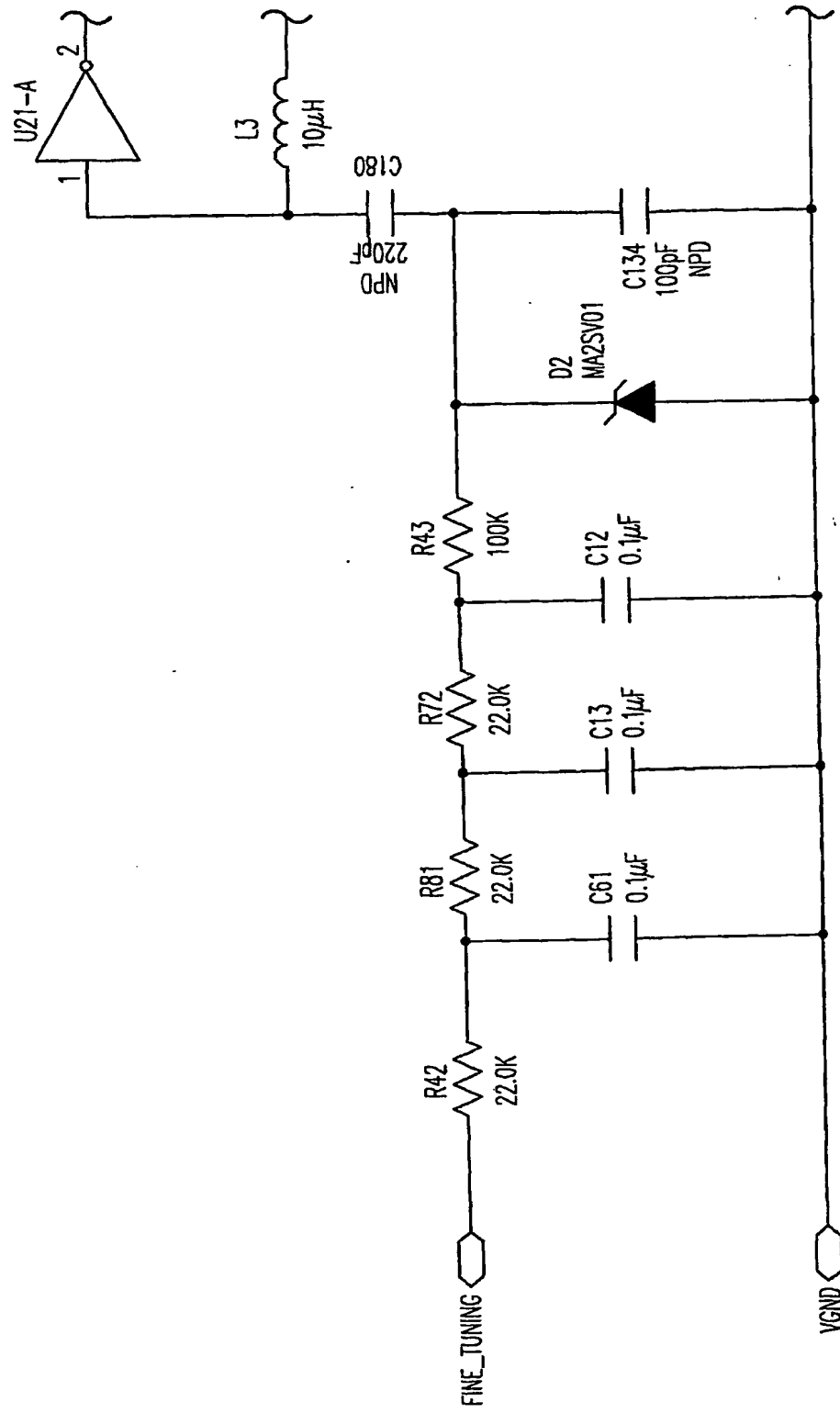


FIG. 31M-1

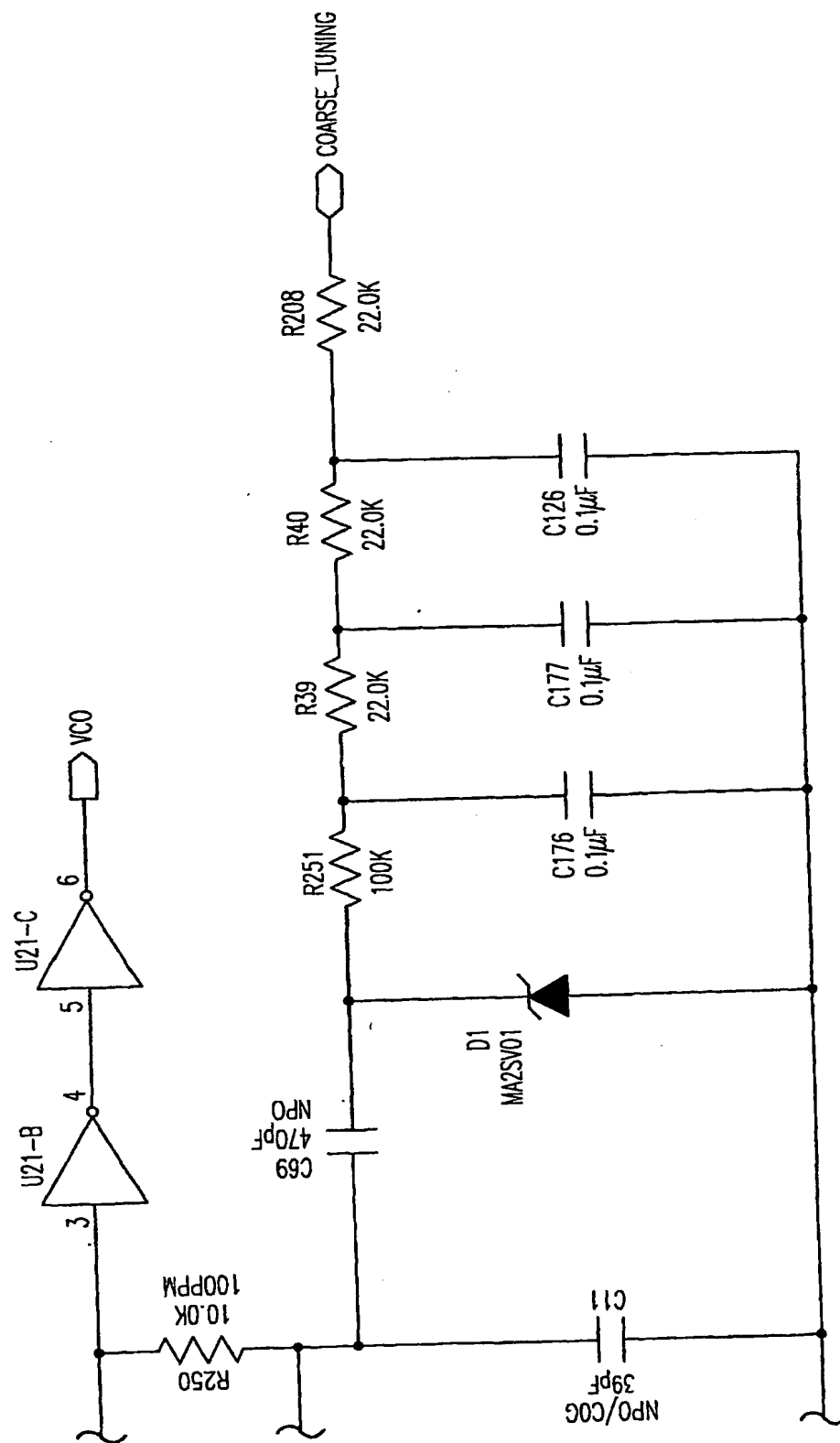


FIG. 31M-2

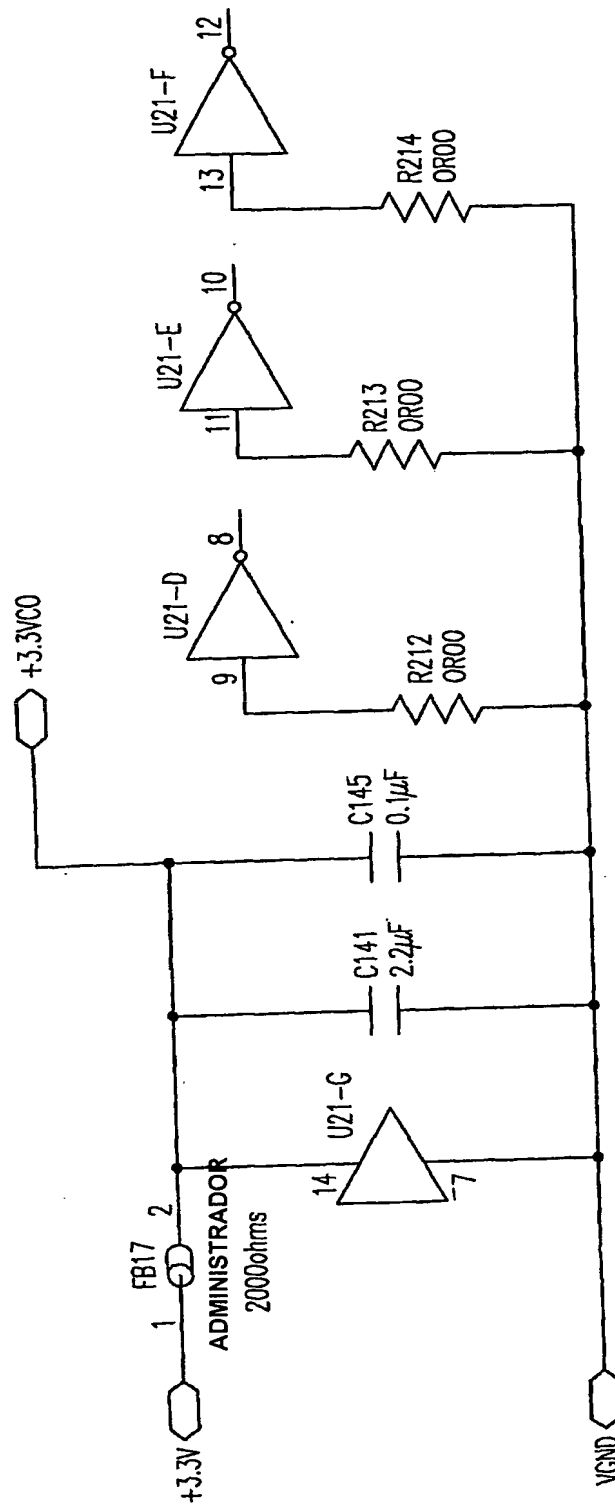


FIG. 31M-3

PLC SCH202-3

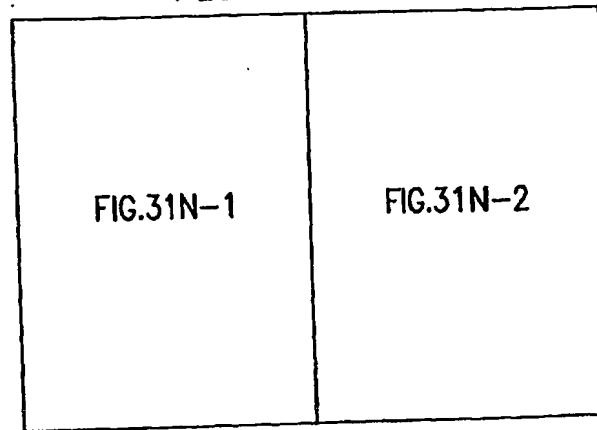


FIG.31N

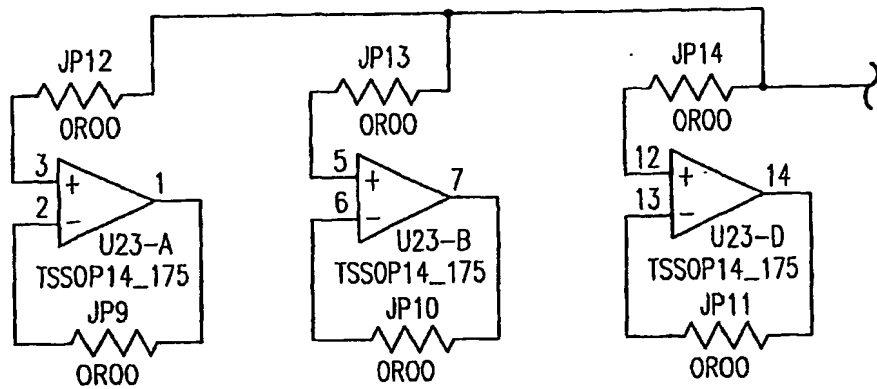
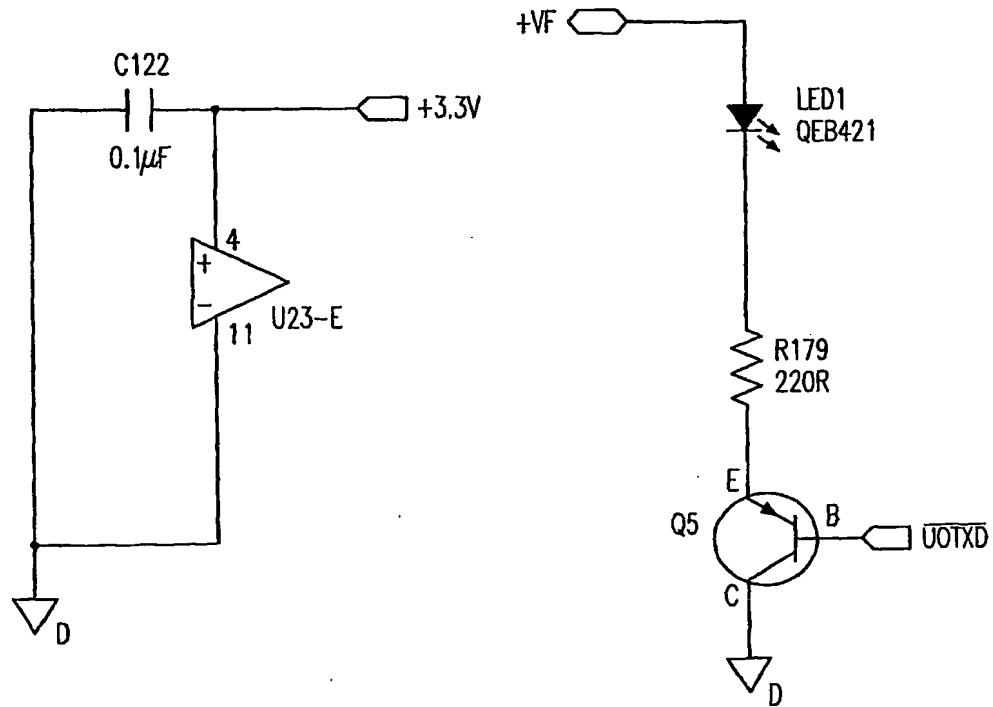


FIG.31N-1

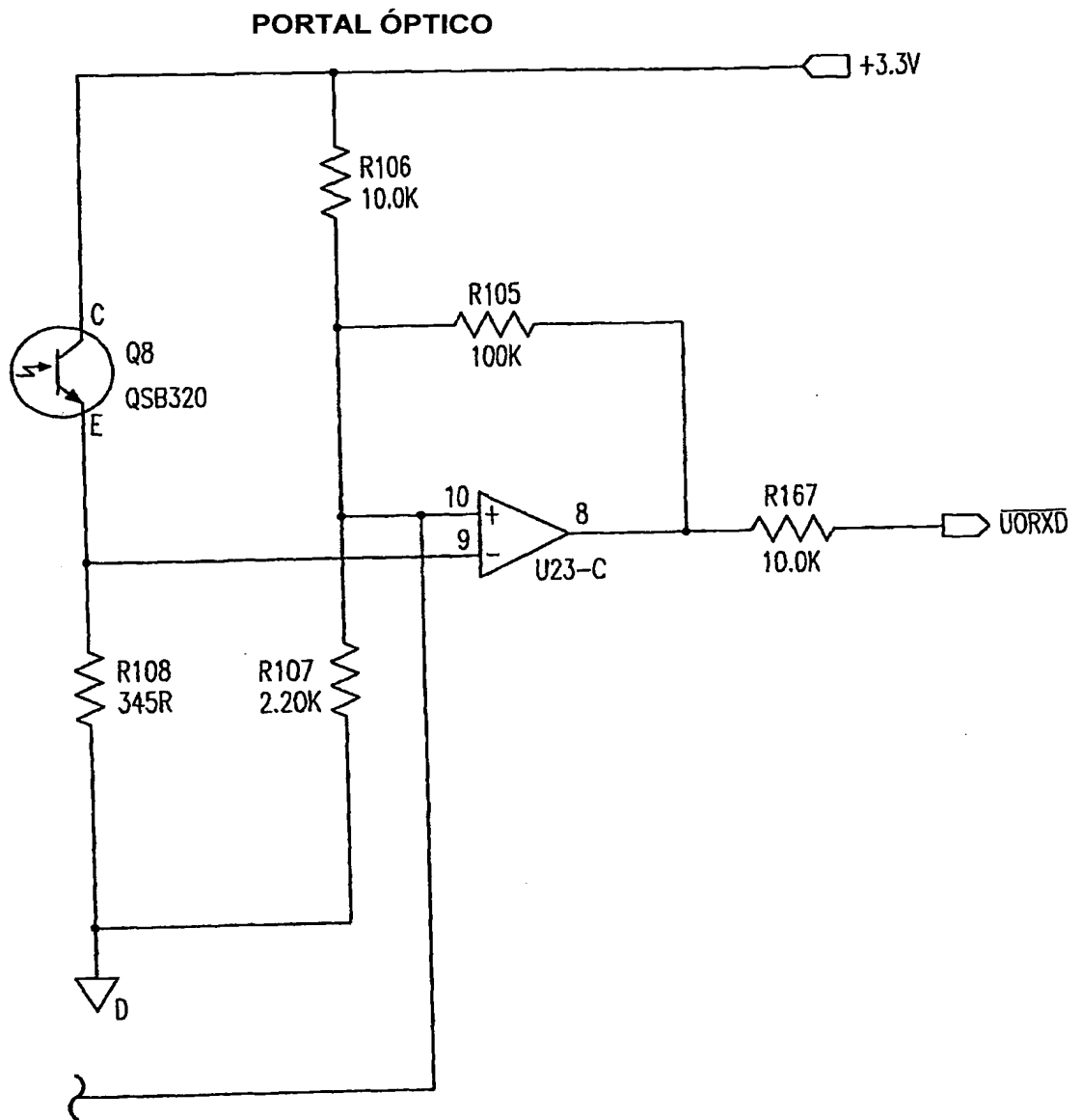


FIG.31N-2

I/O SERIAL SCH202-3

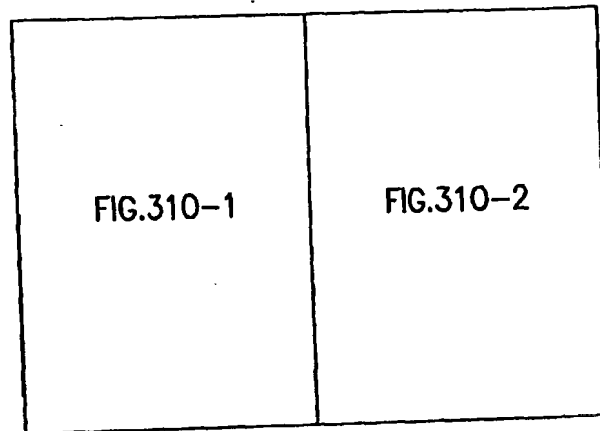
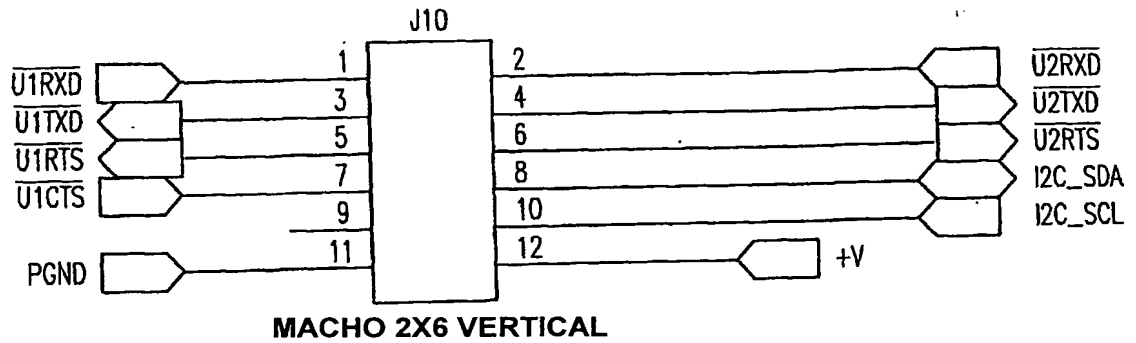


FIG.310

COLETOR PARA PLACA DE IO



COLETOR PARA PLACA DE POTÊNCIA

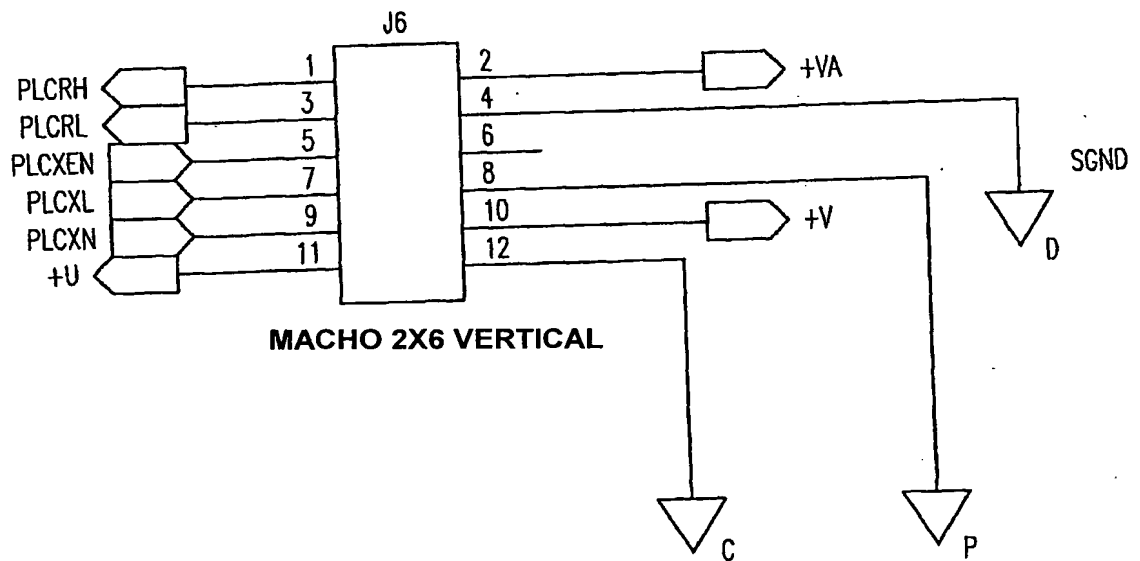
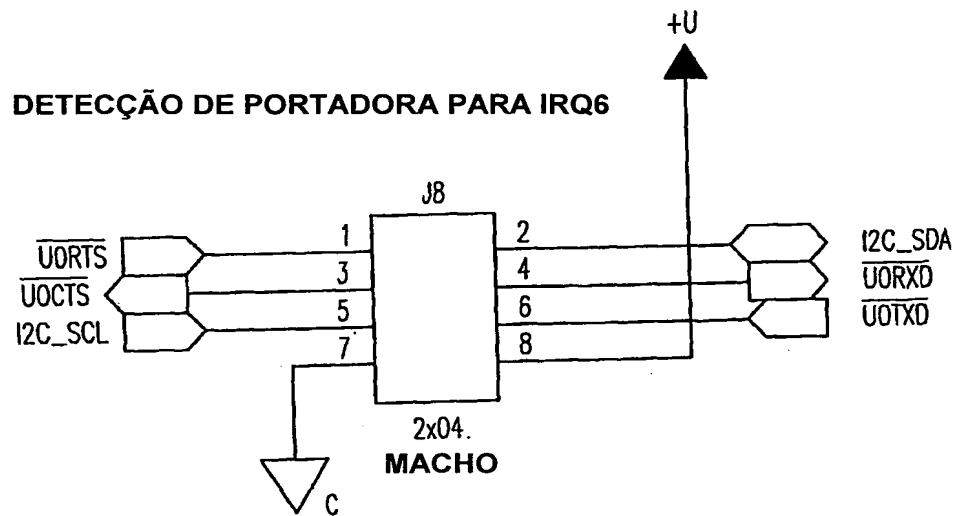


FIG.310-1

COLETOR PARA PLACA DE MODEM



COLETOR PARA PLACA DE MOSTRADOR

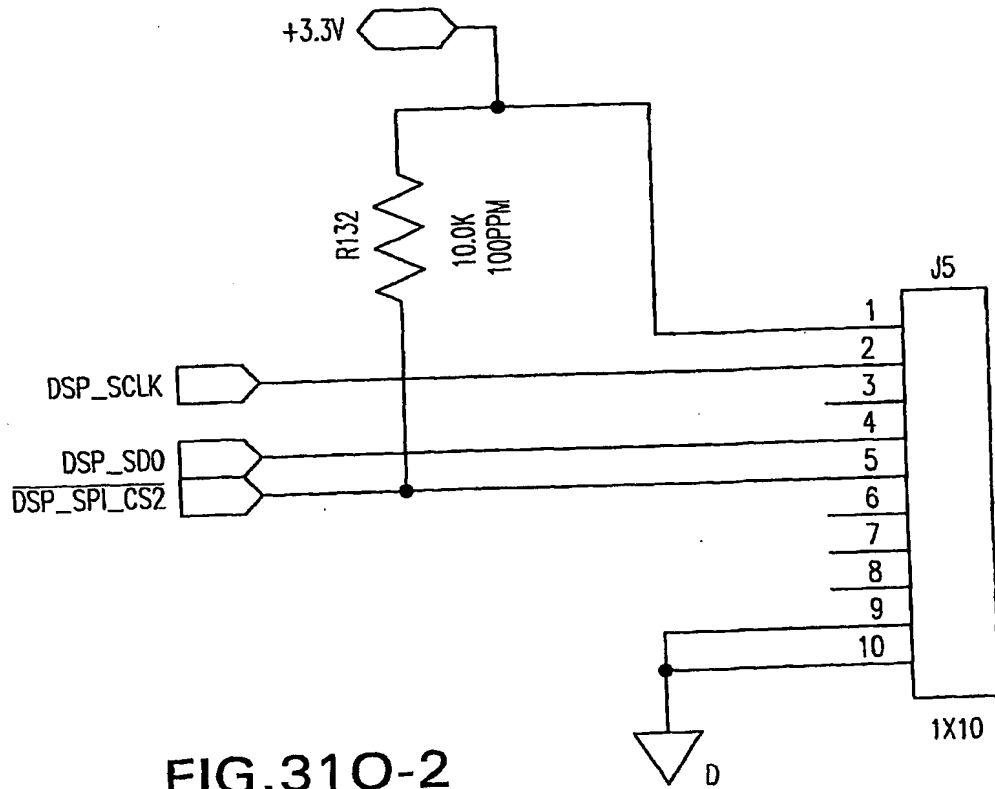
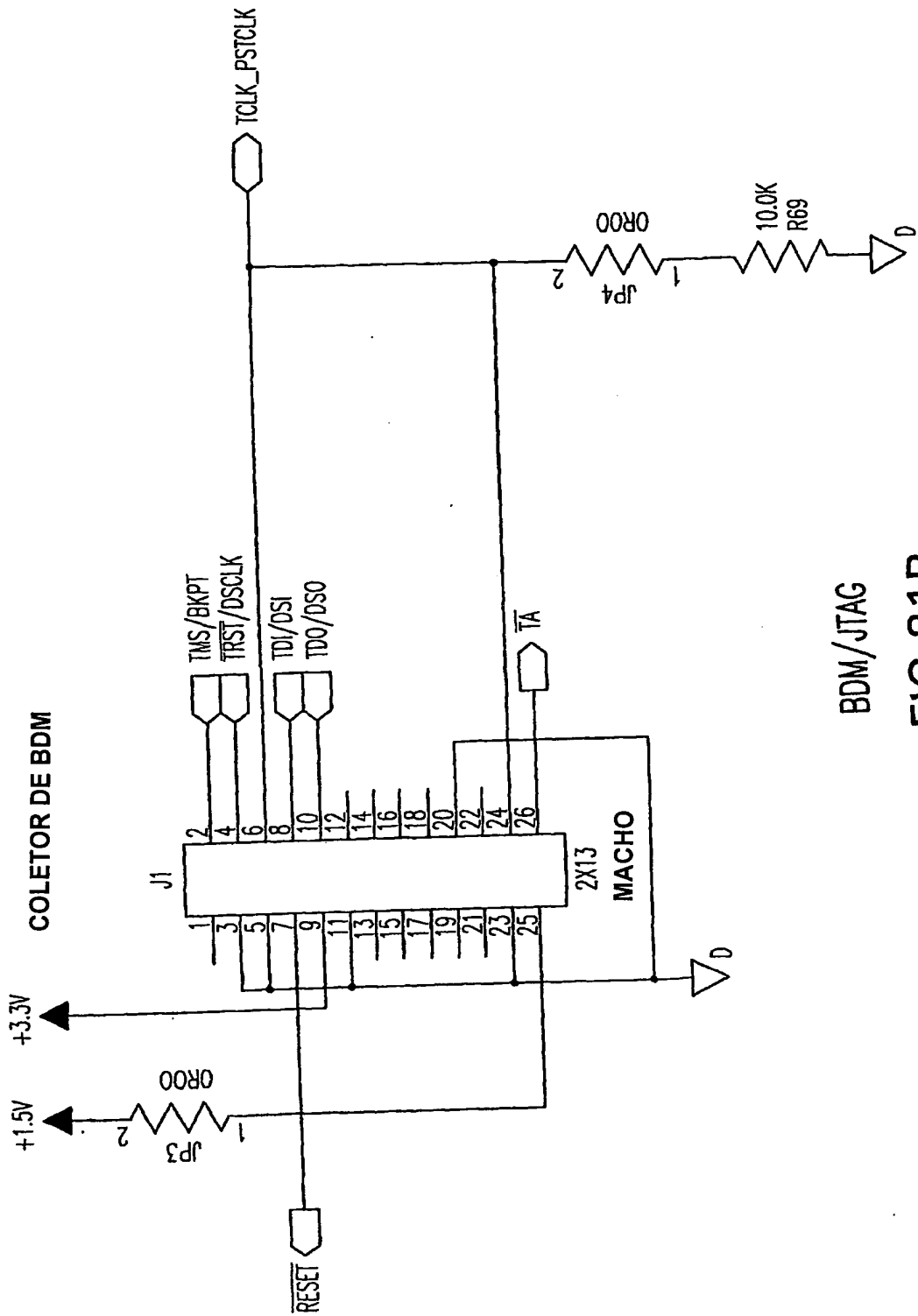


FIG.310-2



BDM/JTAG

FIG.31P

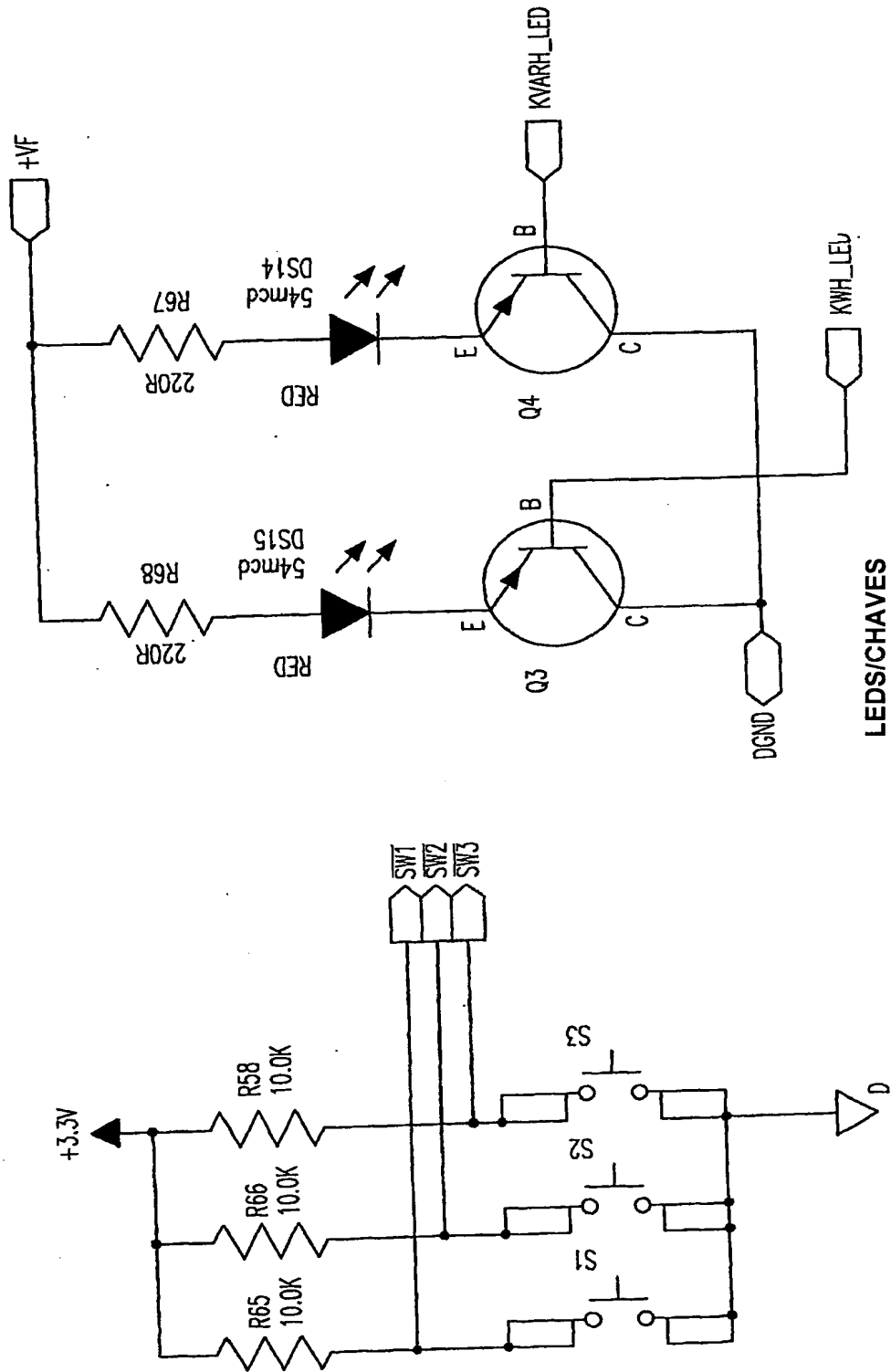


FIG.310

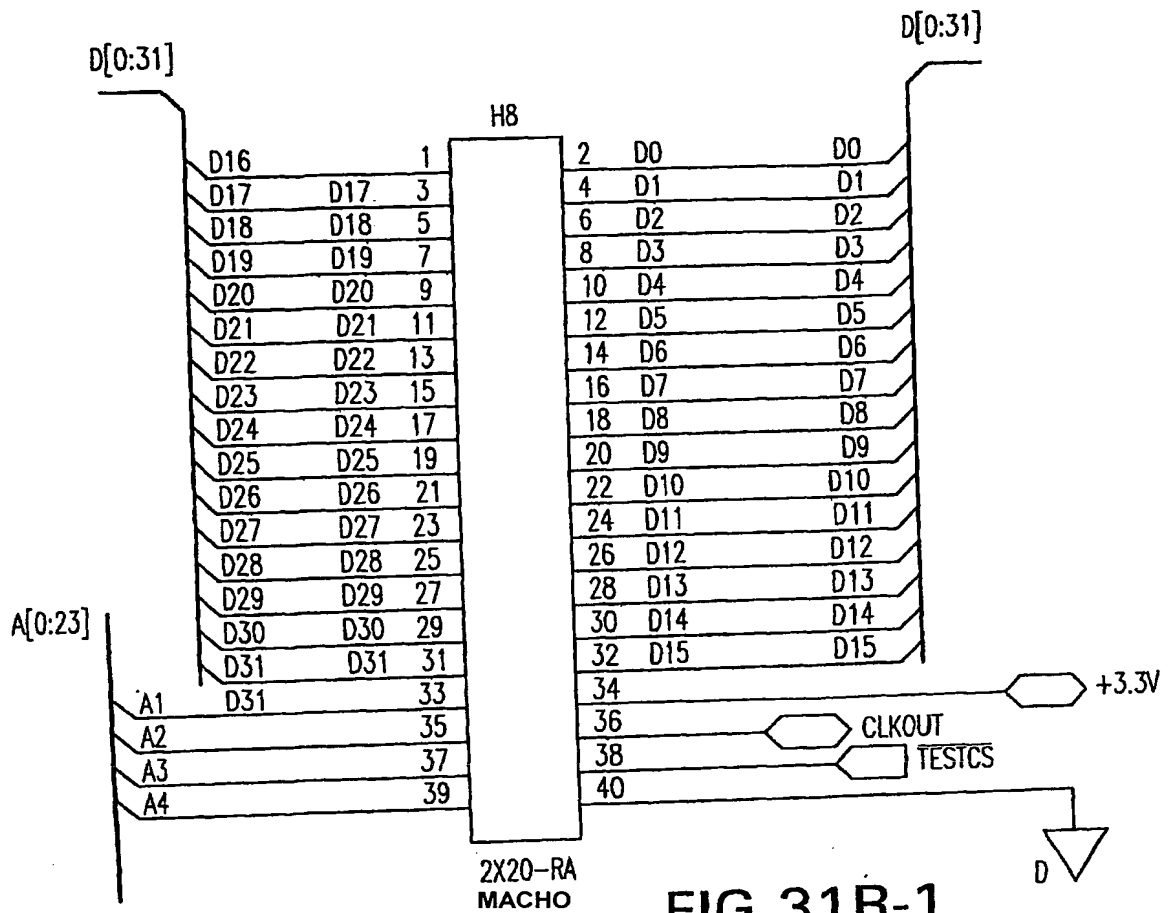


FIG.31R-1

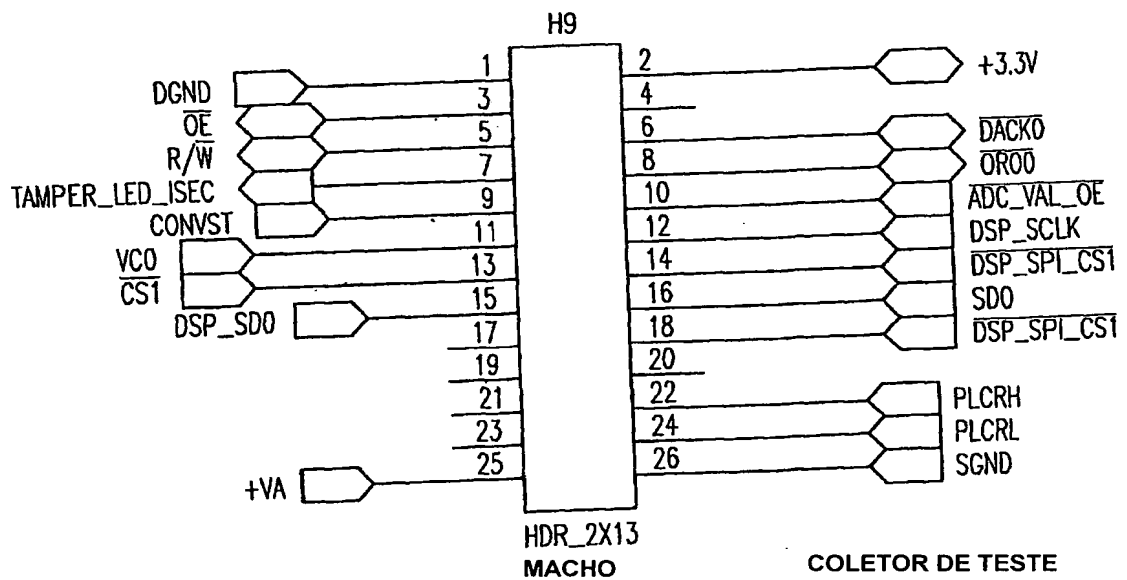
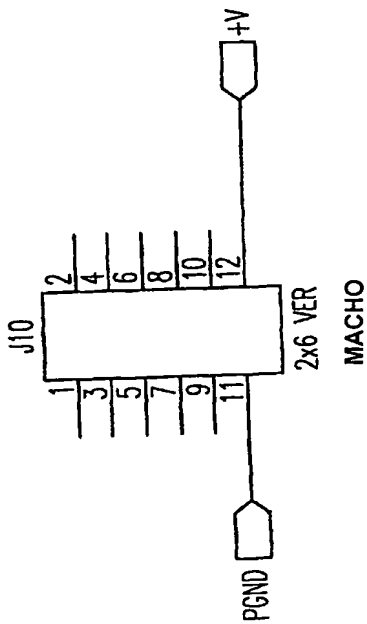


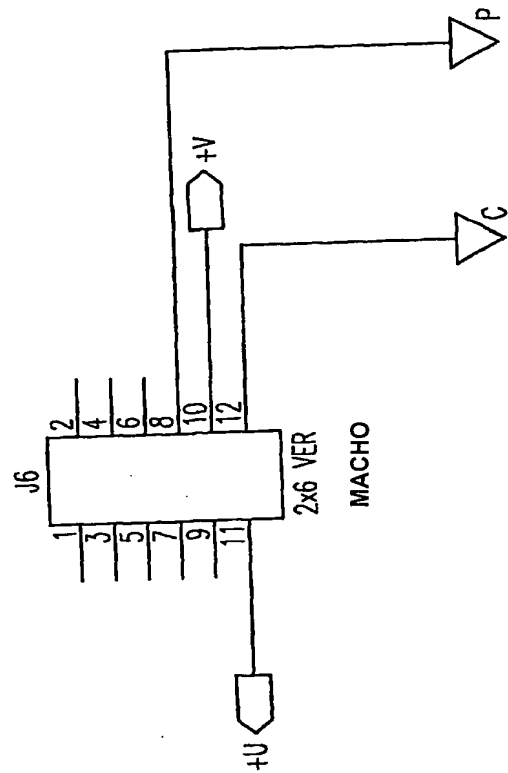
FIG.31R-2

COLETOR DE TESTE

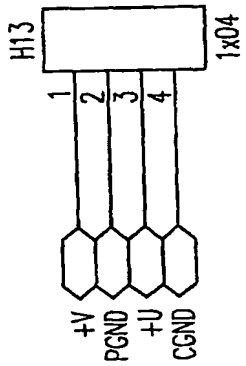
COLETOR PARA PLACA DE IO



COLETOR PARA PLACA DE POTÊNCIA



COLETOR DE POTÊNCIA



I/O SERIAL
FIG.32

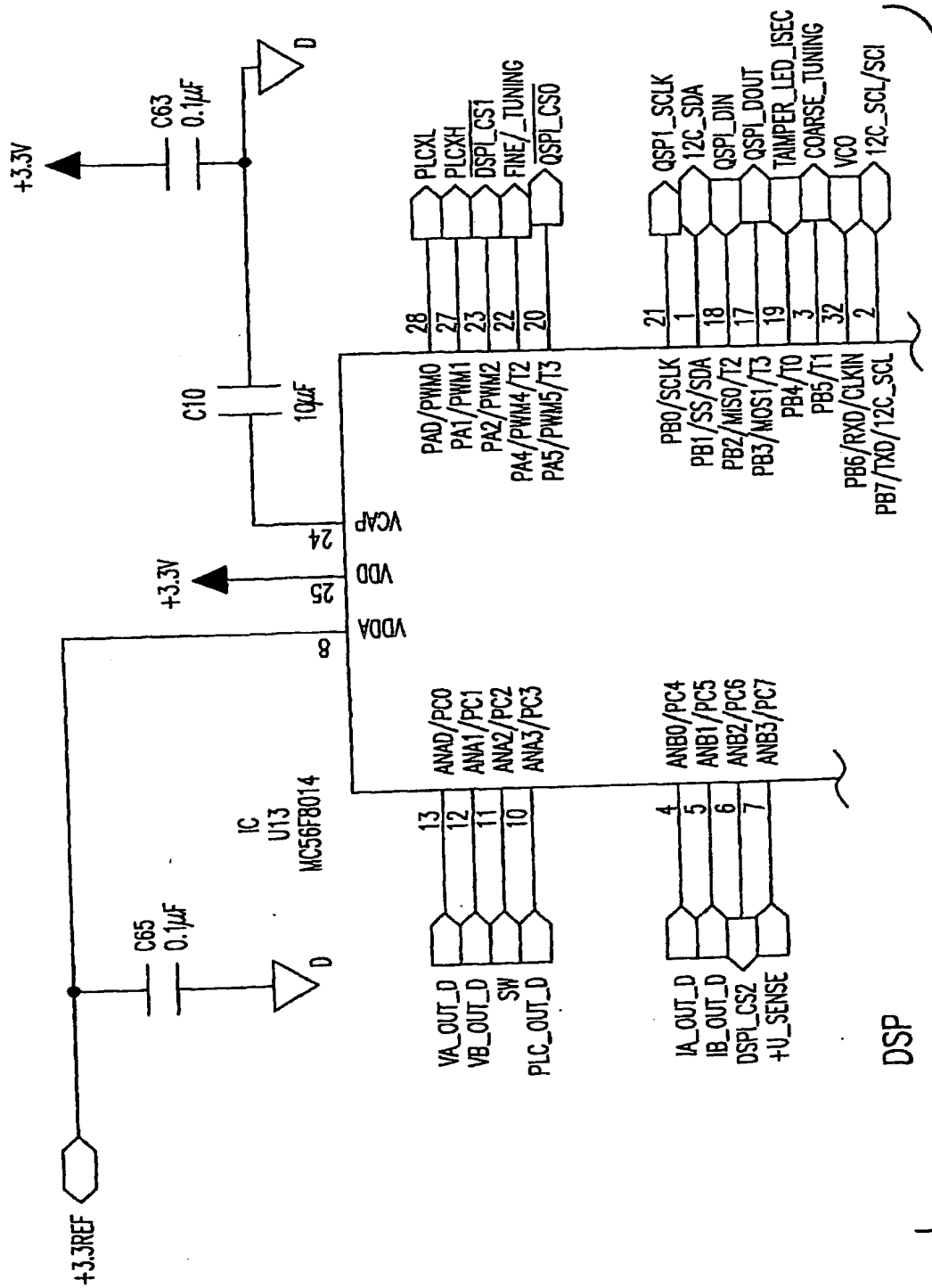
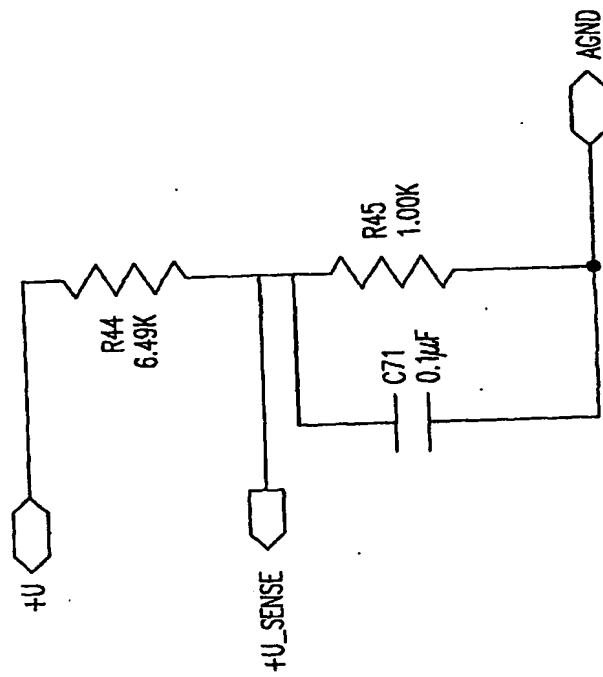


FIG. 33A-1

Cont.
Fig. 33A-2



DSP

FIG.33A-3

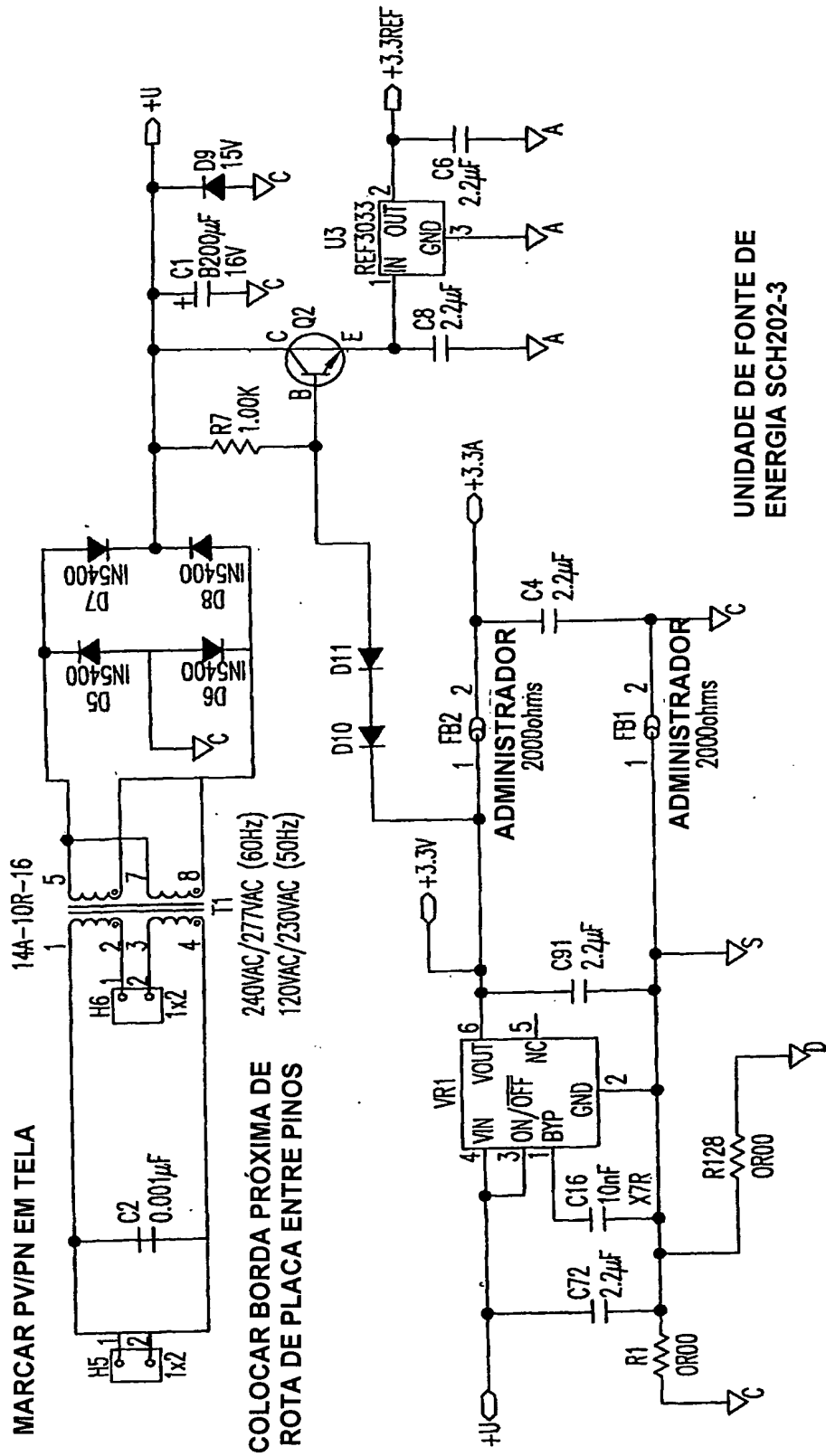
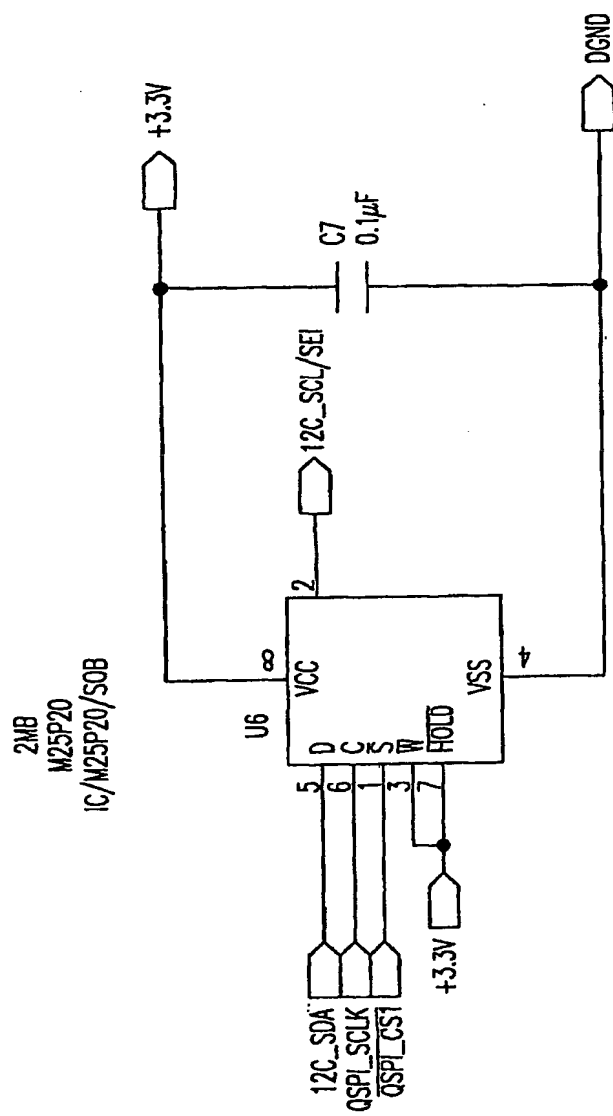


FIG. 33B



MEMÓRIA FLASH
FIG.33C

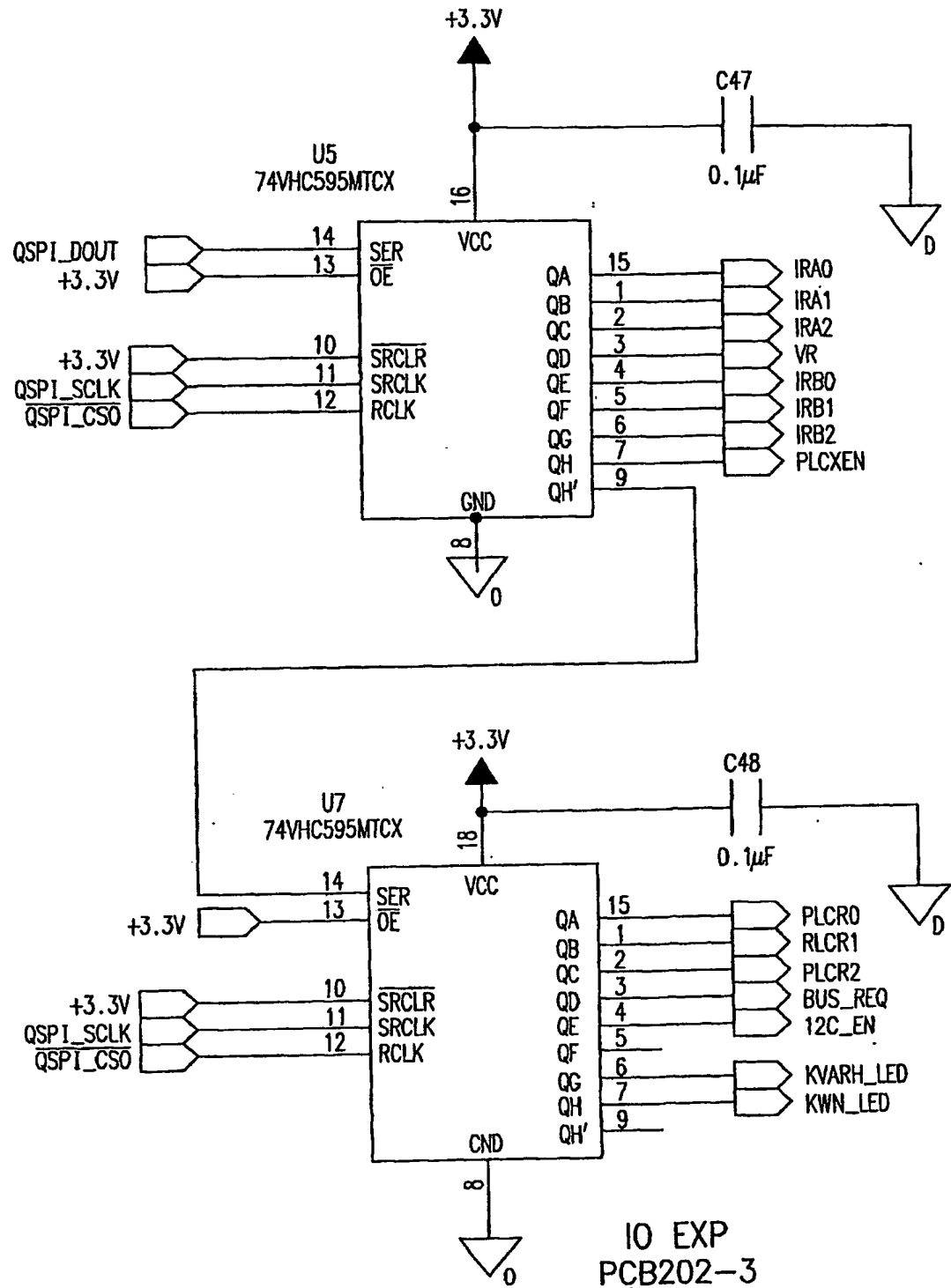


FIG.33D

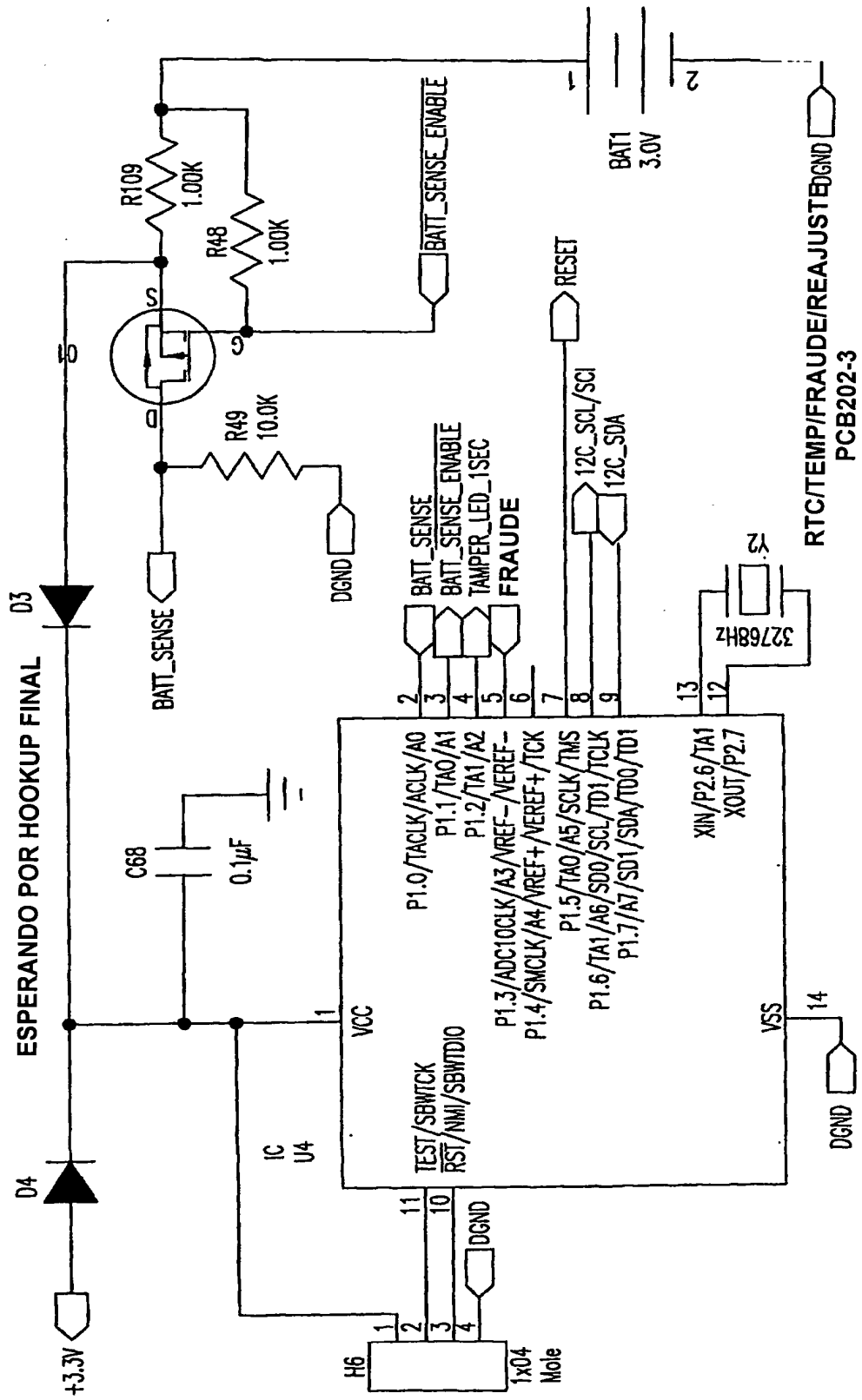


FIG.33E-1

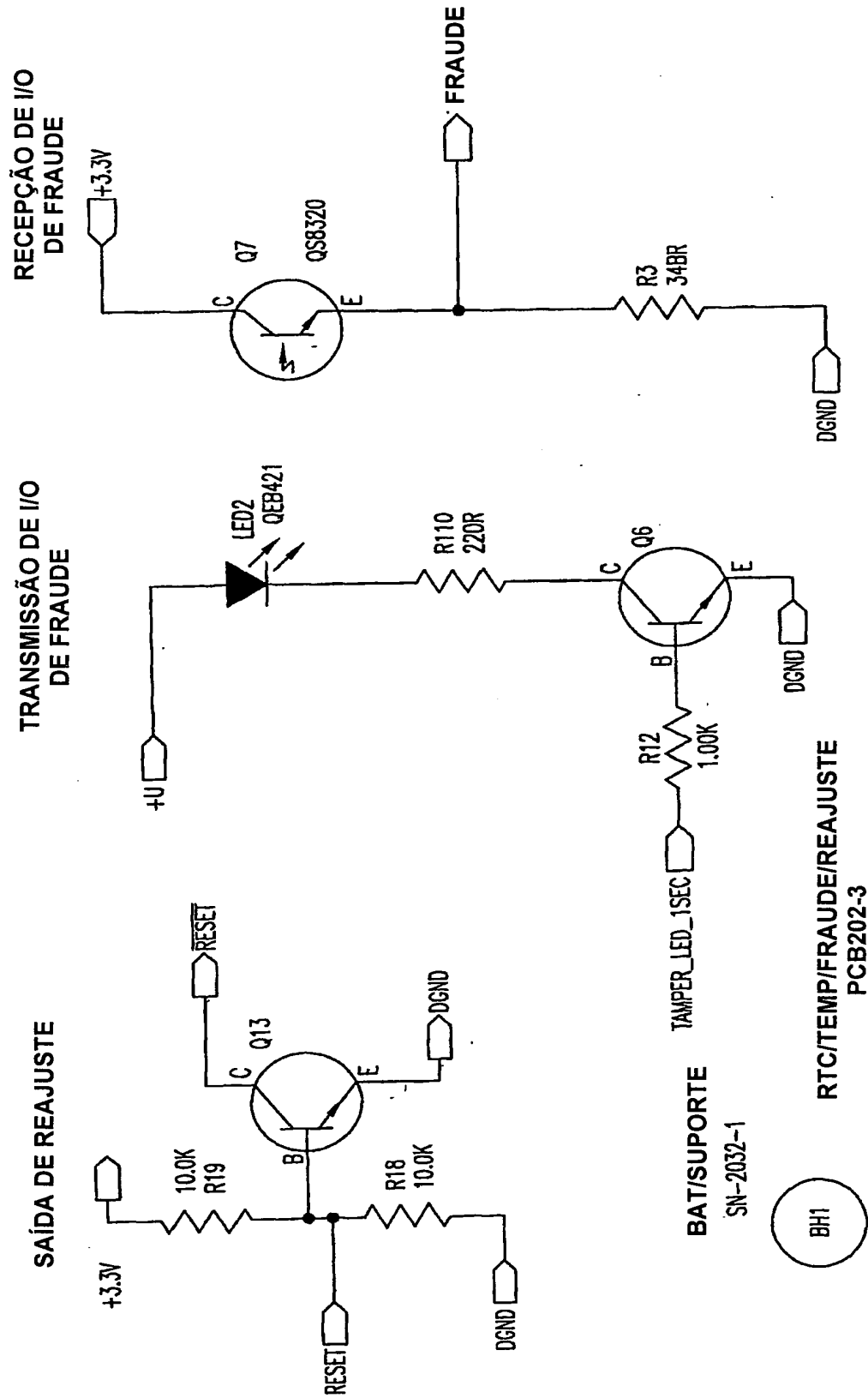


FIG. 33E-2

**8 AMPLIFICADORES DIFERENCIAIS
DESLOCAMENTOS 1,0 - 2,0 (1,0 VPP A
REDOR DE 1,5 REF)
PARA 5VPP AO REDOR DE 2,5 (MSV)**

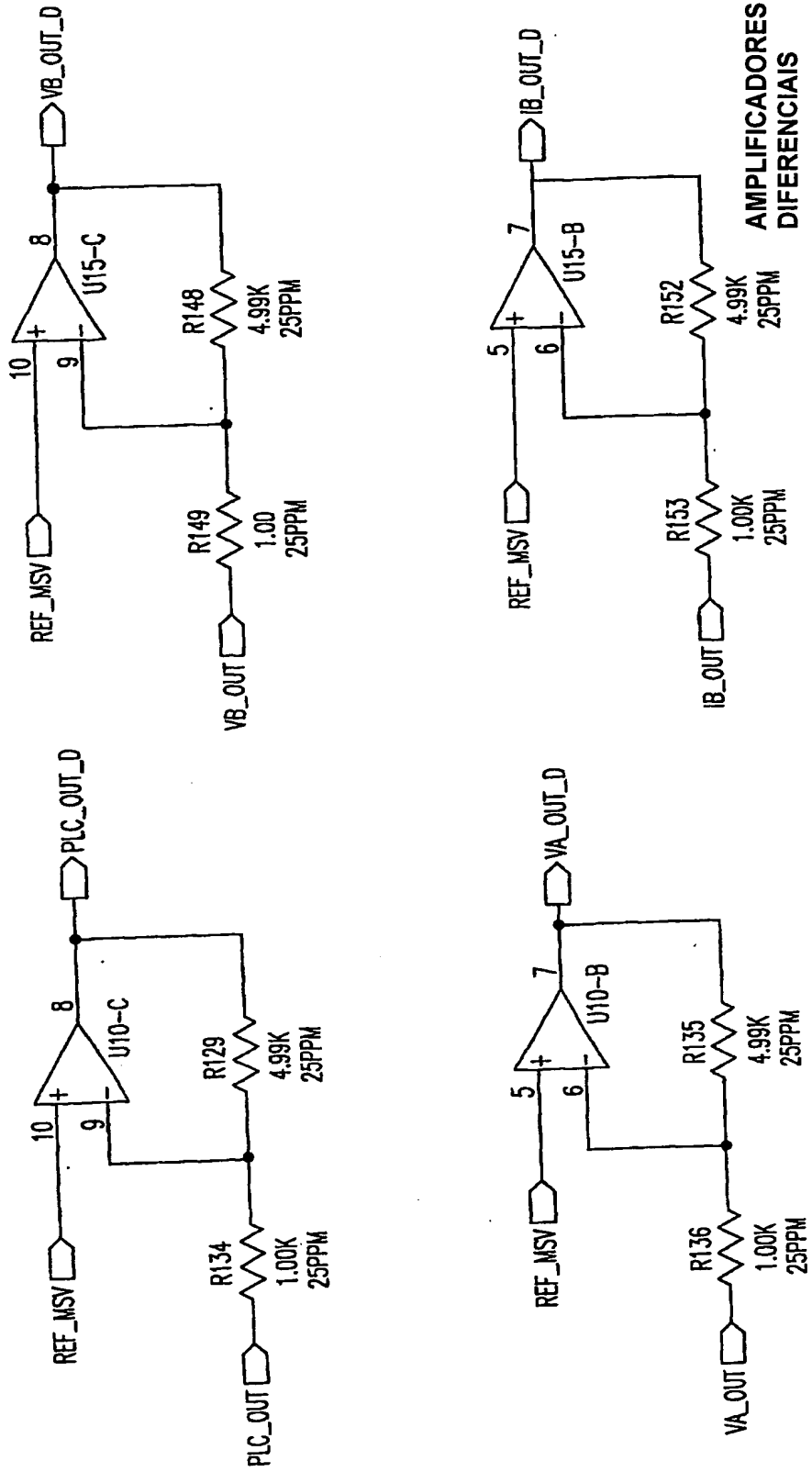


FIG.33F-1

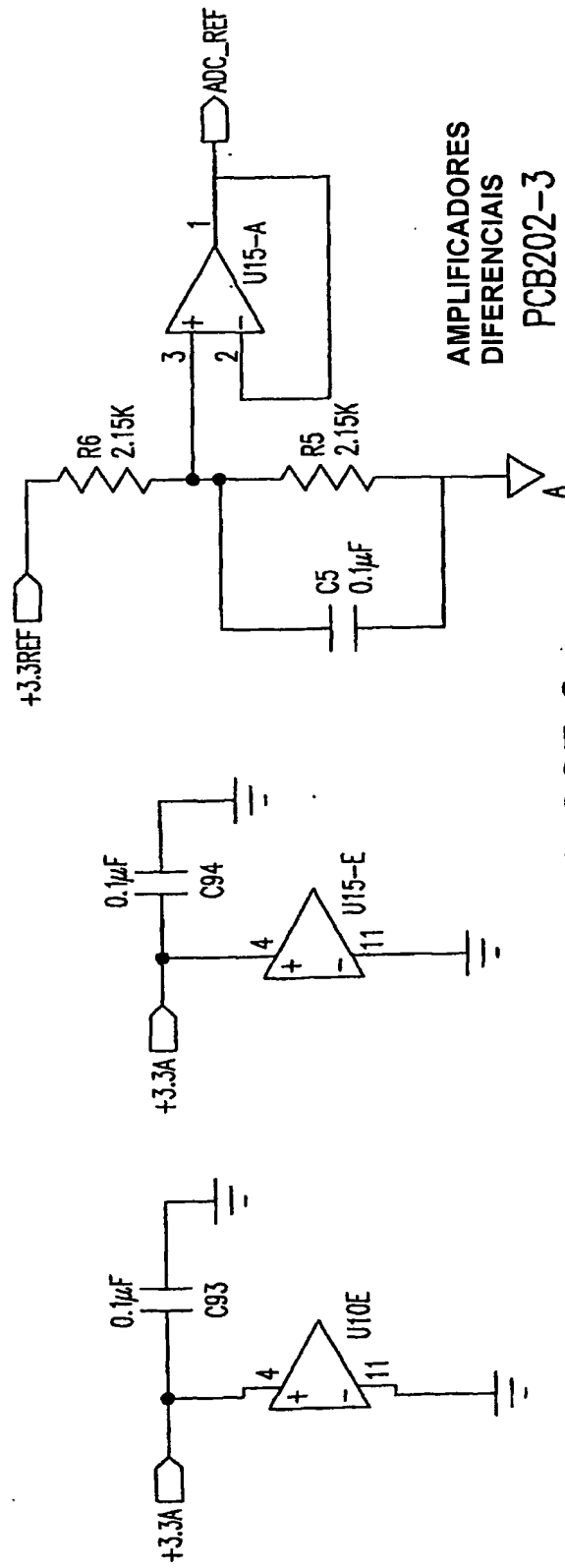
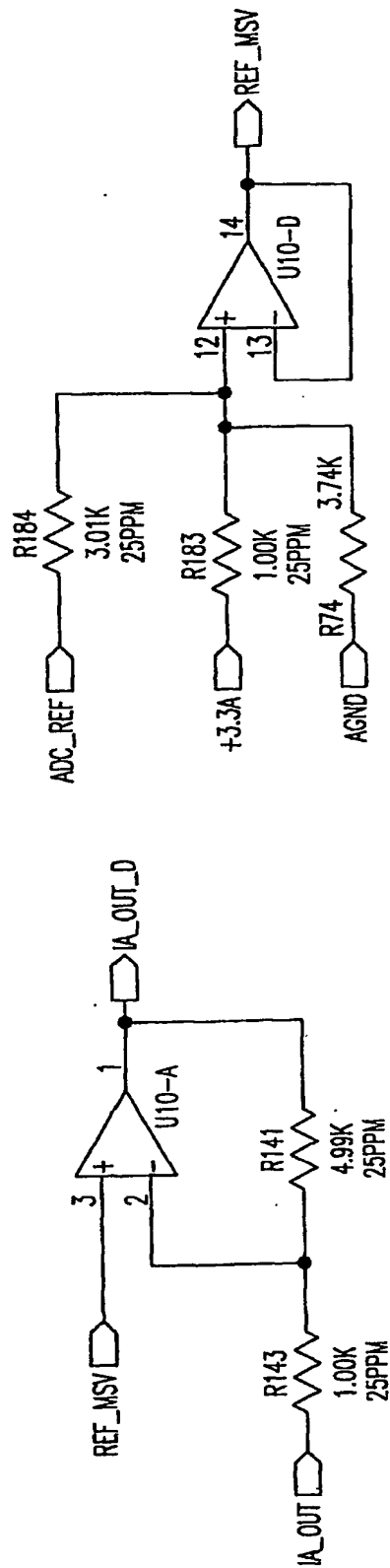


FIG.33F-2

Cont.
Fig. 33G-2

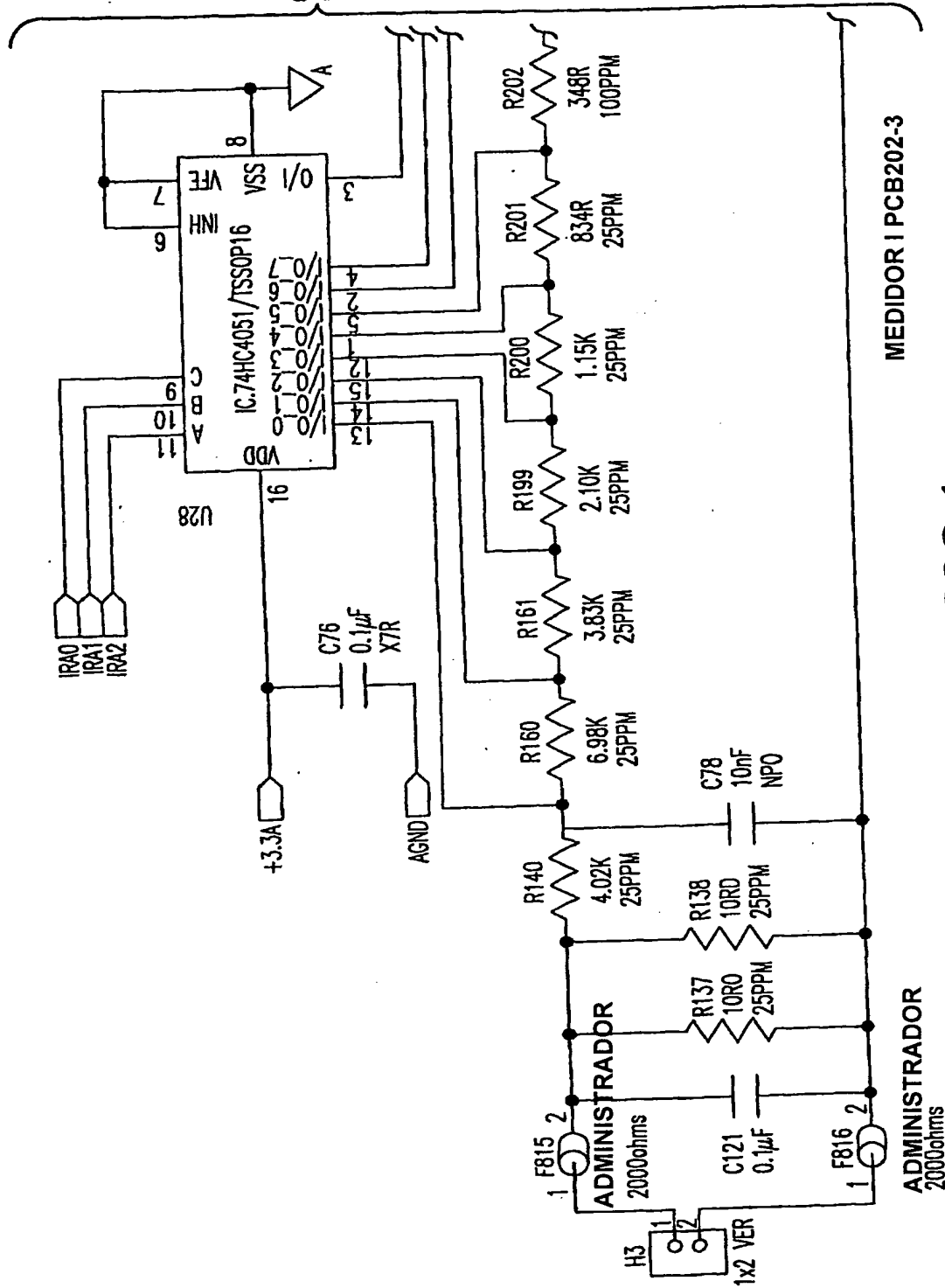


FIG. 33G-1

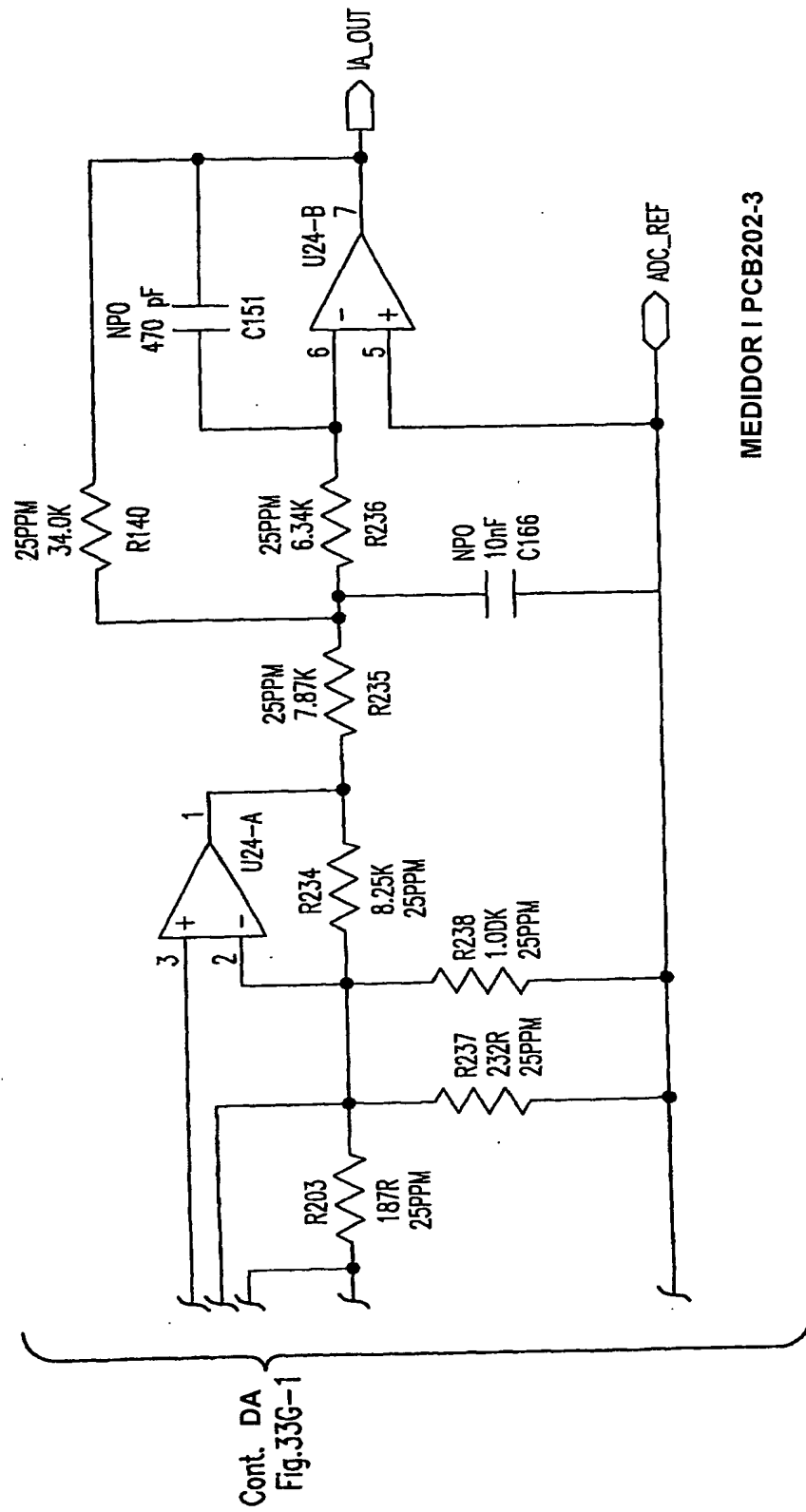
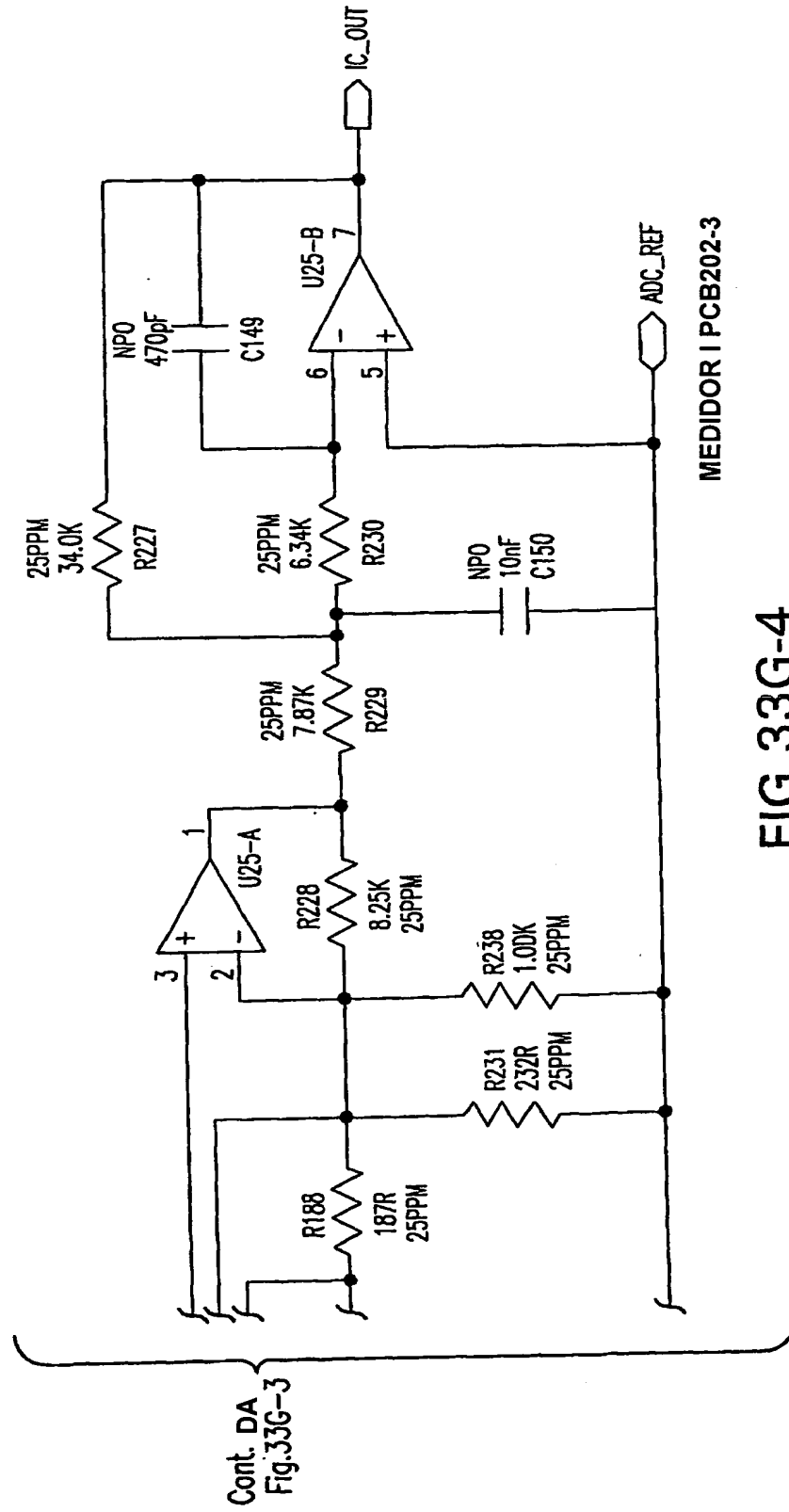
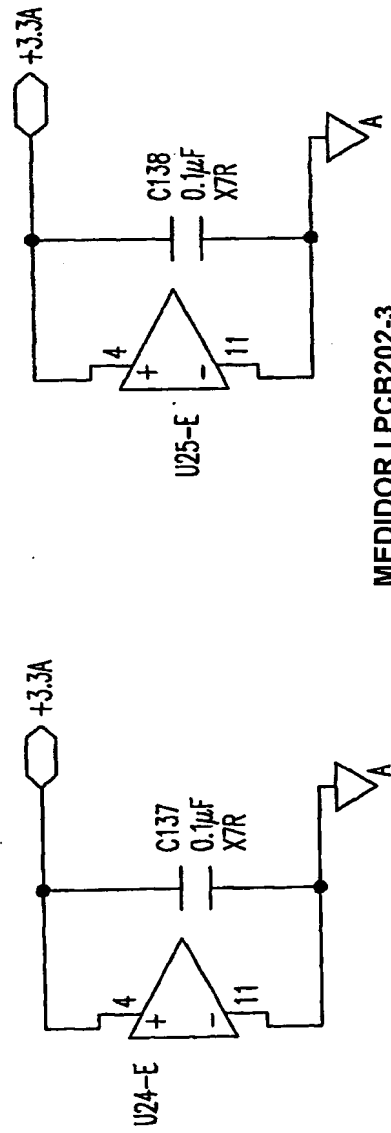


FIG.33G-2

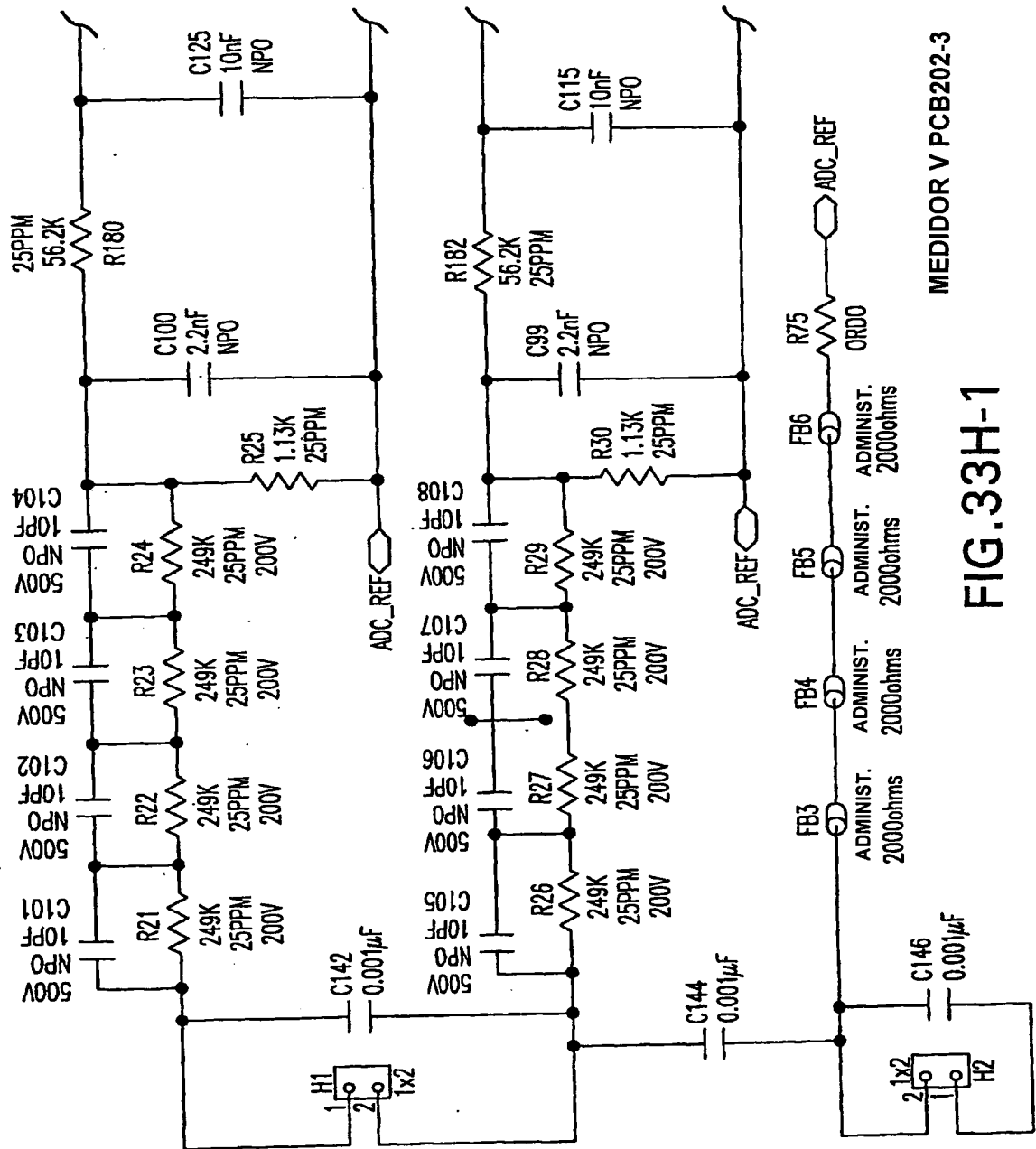


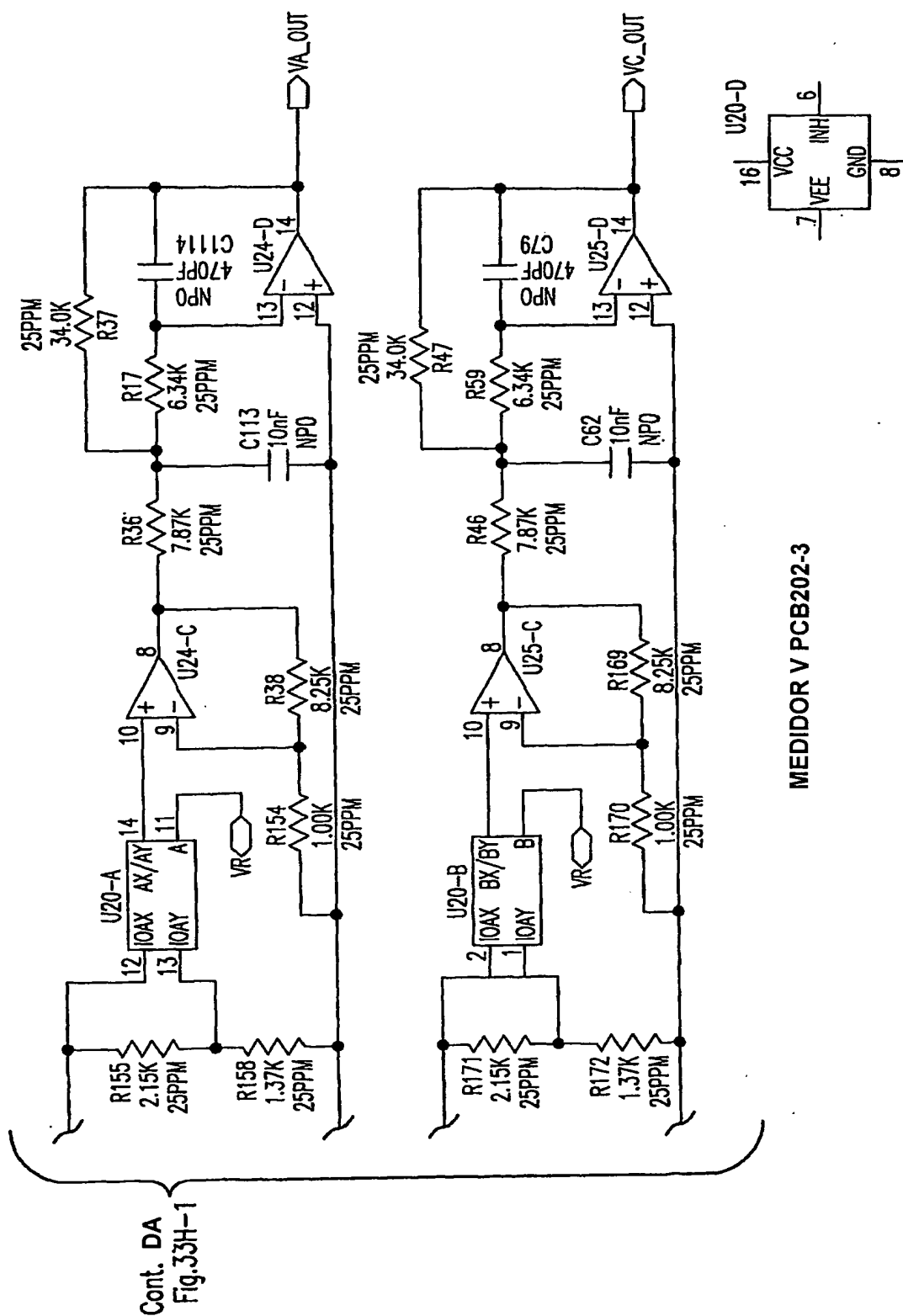


MEDIDOR I PCB202-3

FIG.33G-5

Cont. DA
Fig.33H-2





MEDIDOR V PCB202-3

FIG. 33H-2

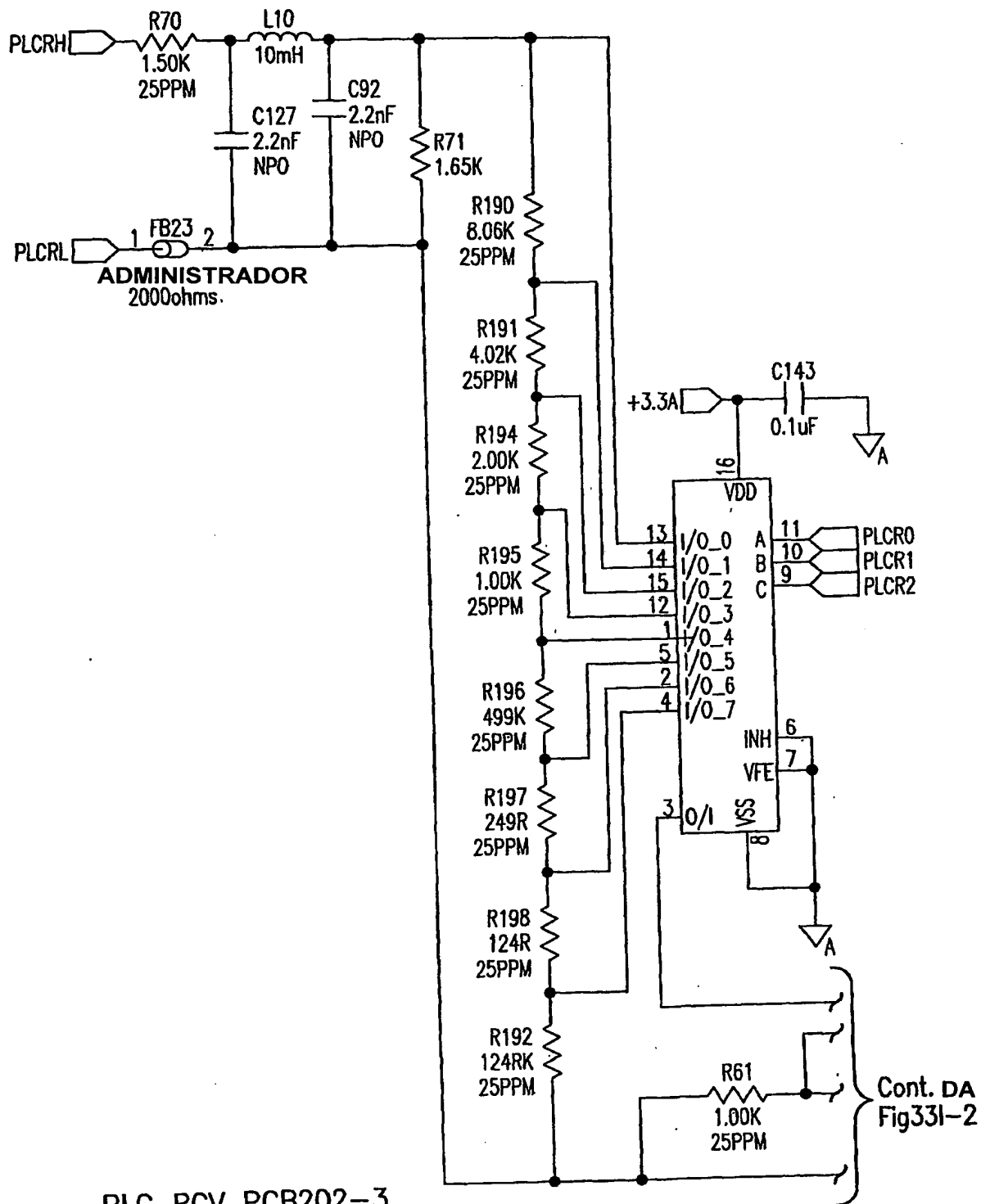
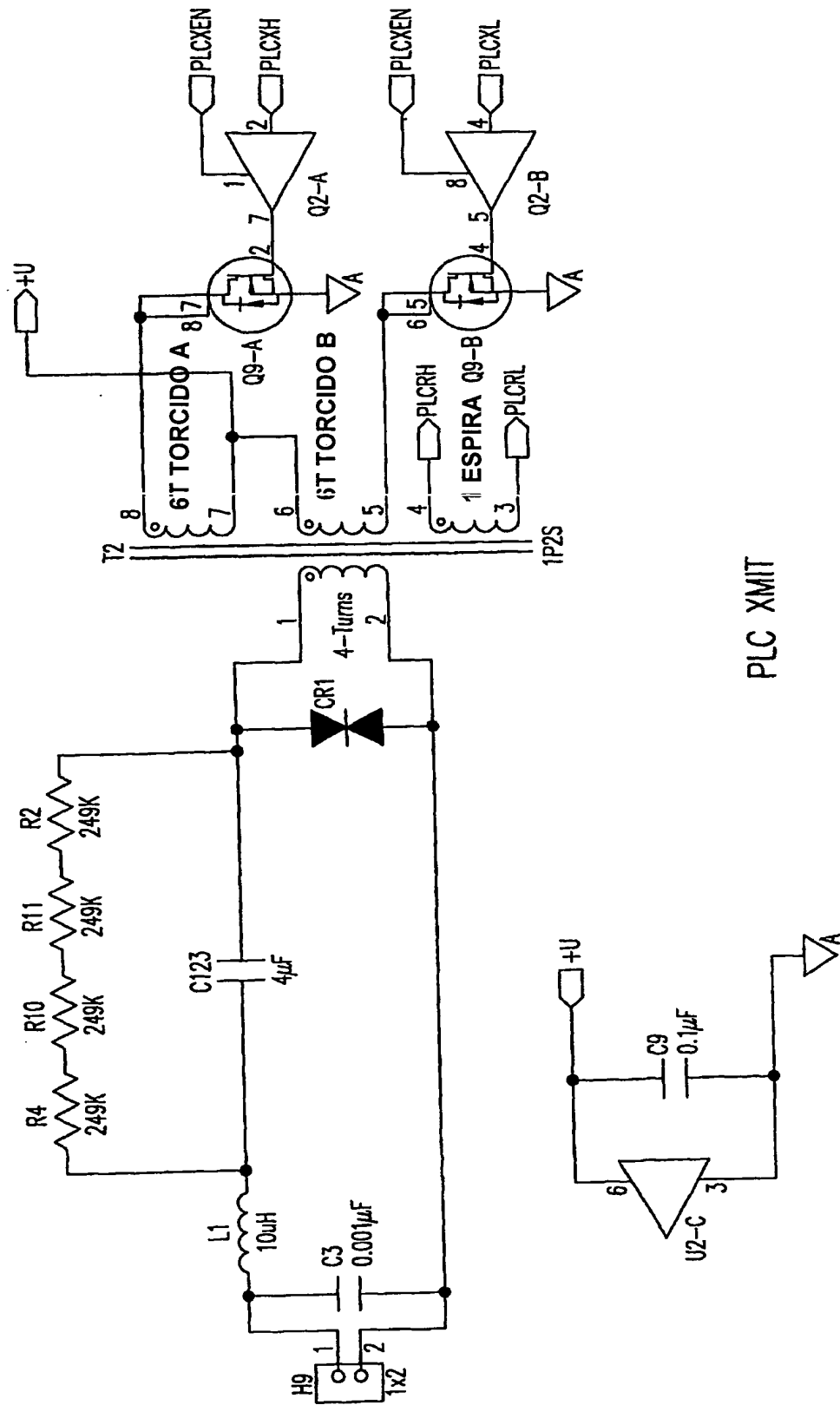


FIG.33I-1

Cont. DA
Fig33I-2



PLC XMIT

FIG.33J

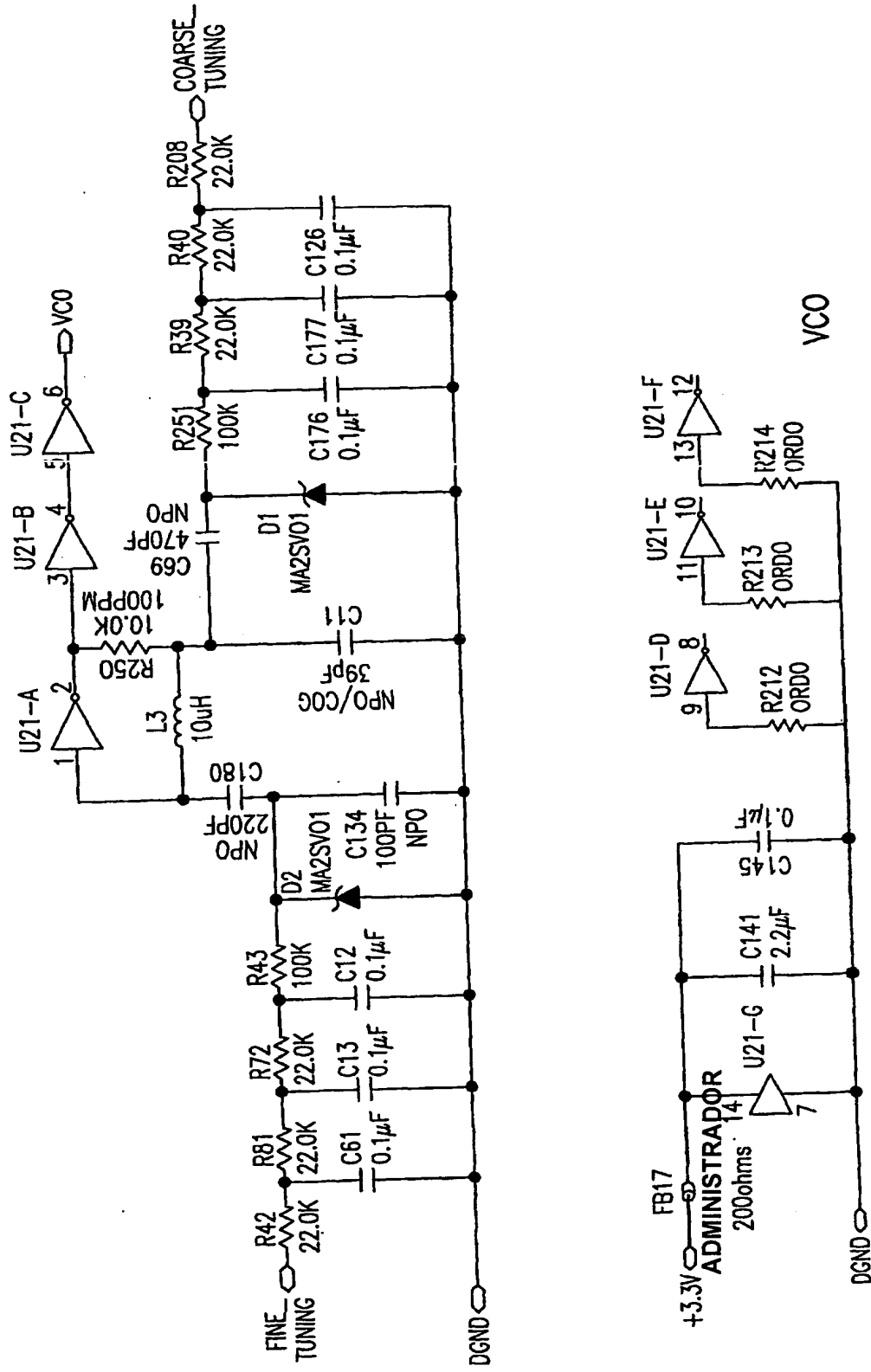


FIG. 33K

COLETOR PARA PLACA DE MOSTRADOR

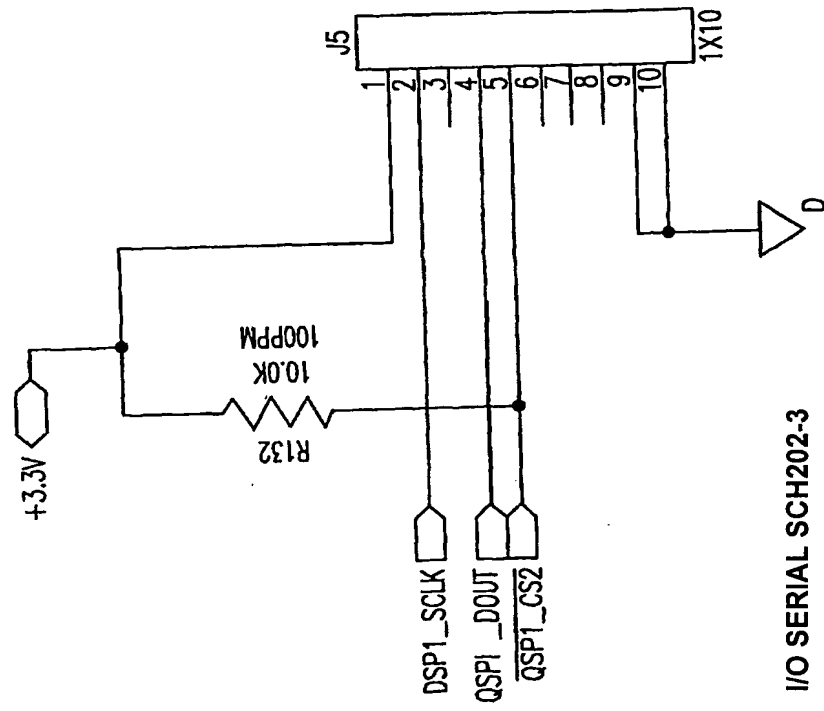


FIG. 33M

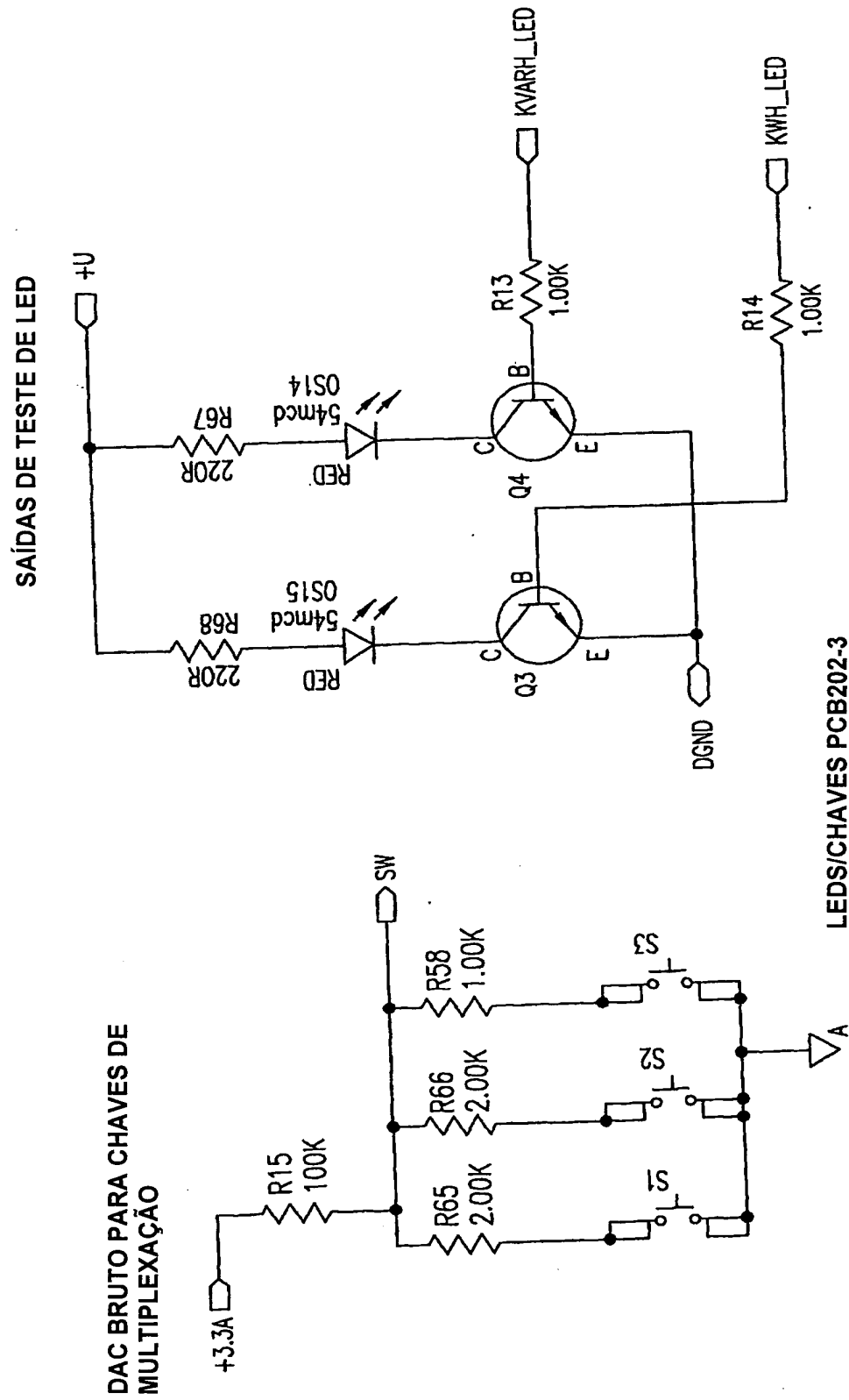
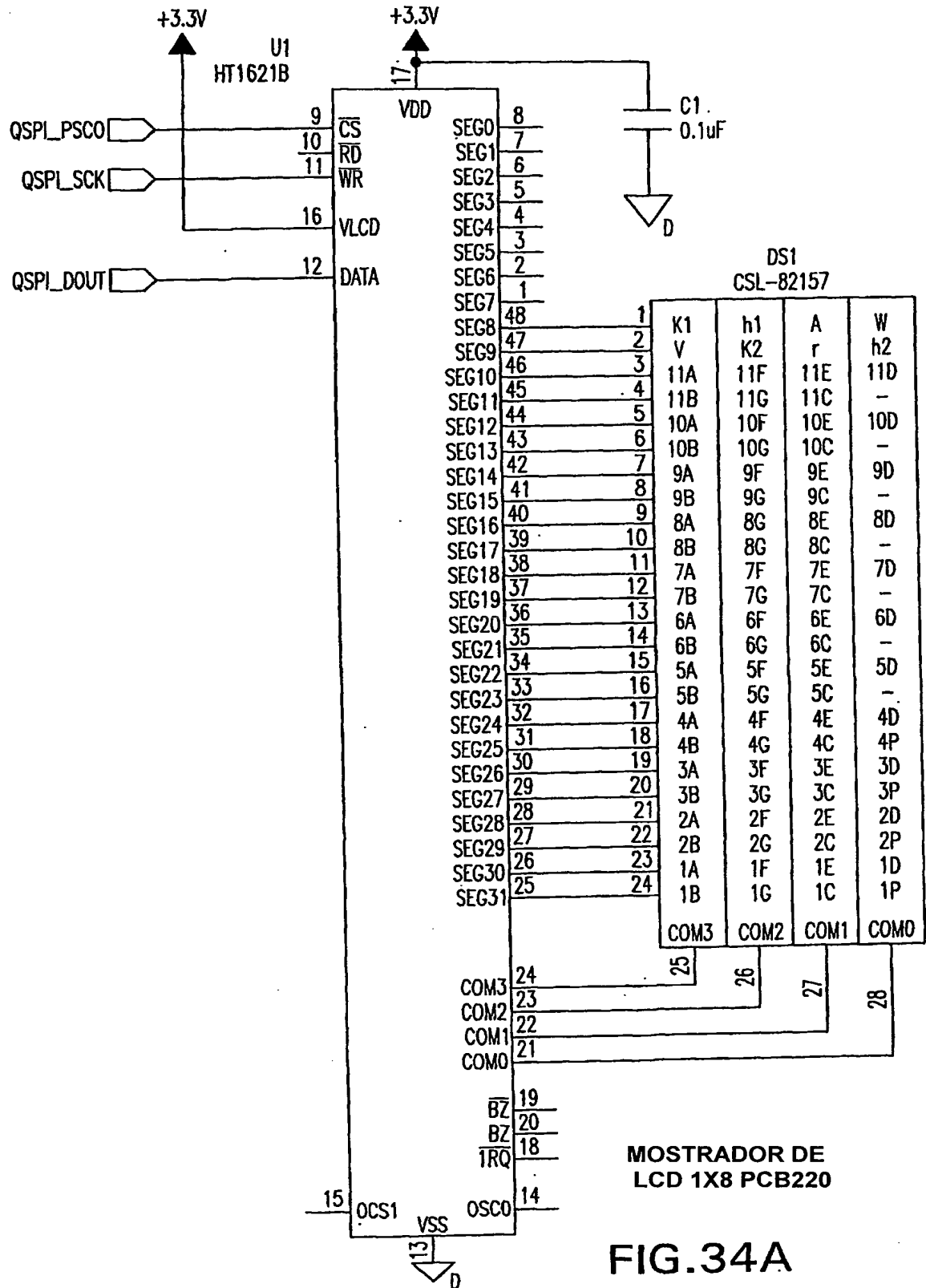


FIG.33N



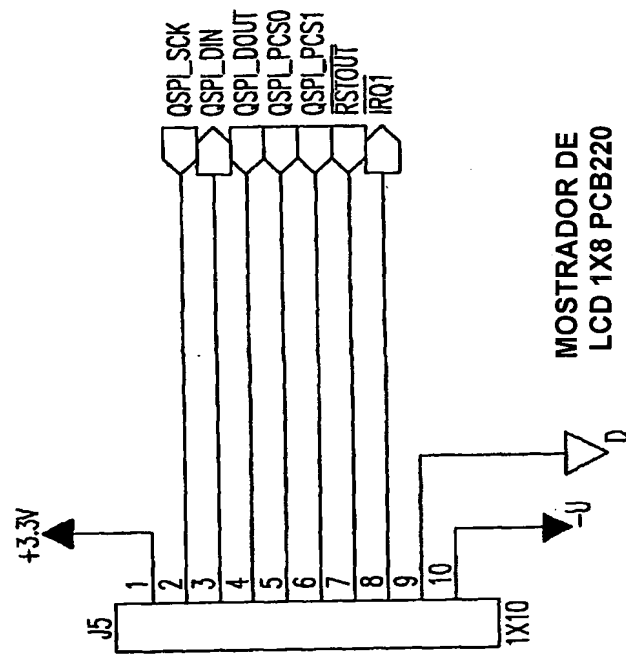


FIG.34B

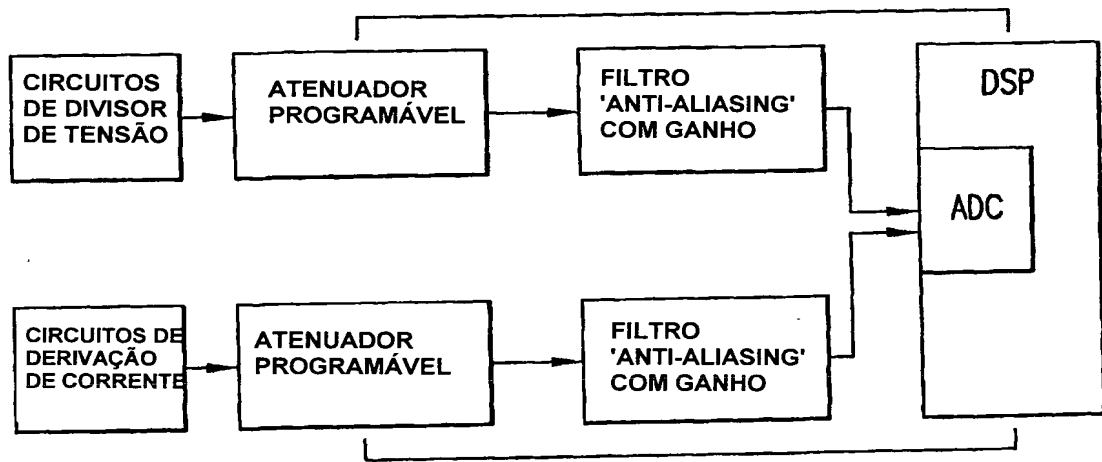


FIG.35

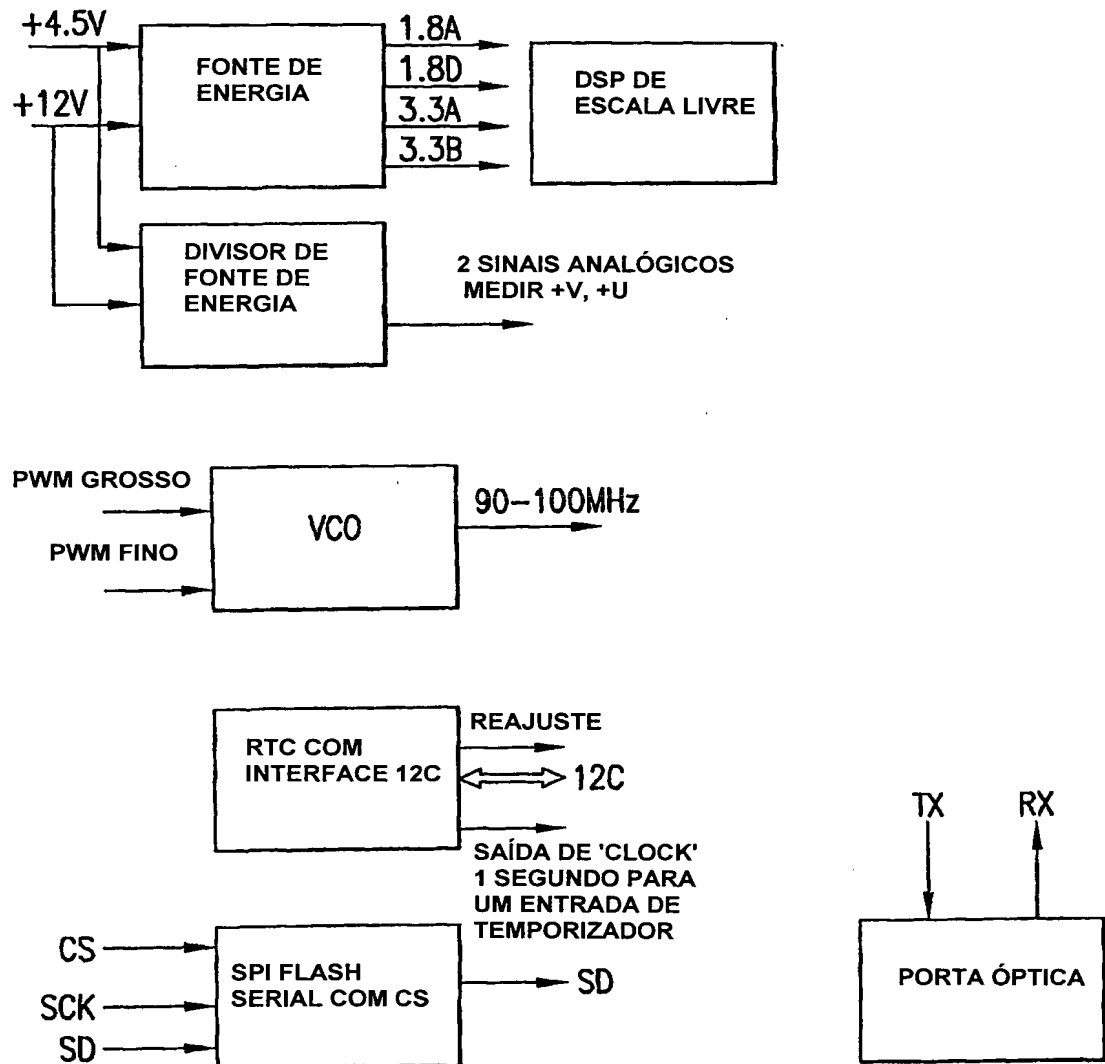
DIAGRAMA DE BLOCO DE HARDWARE DE ID

FIG.36

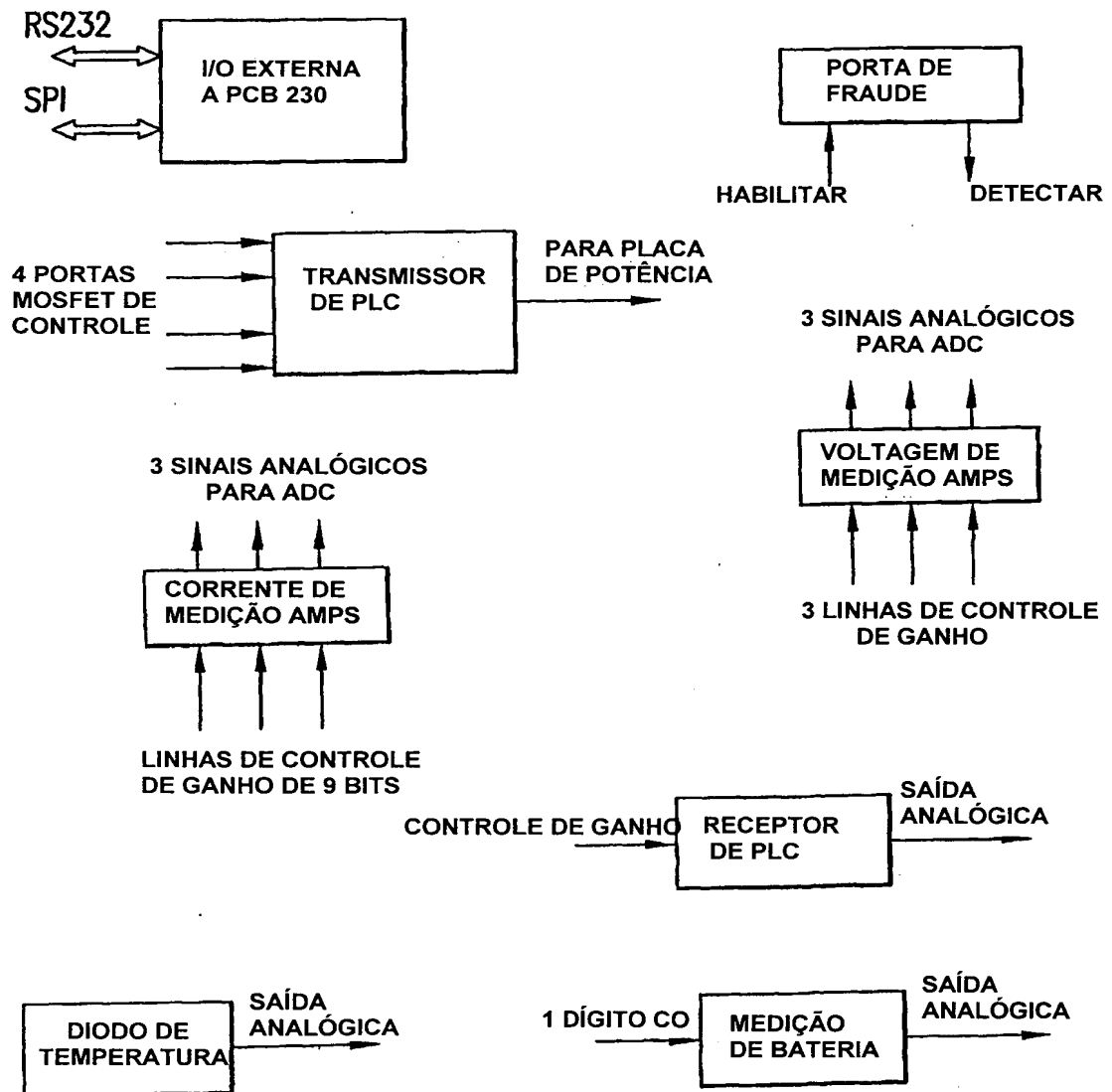
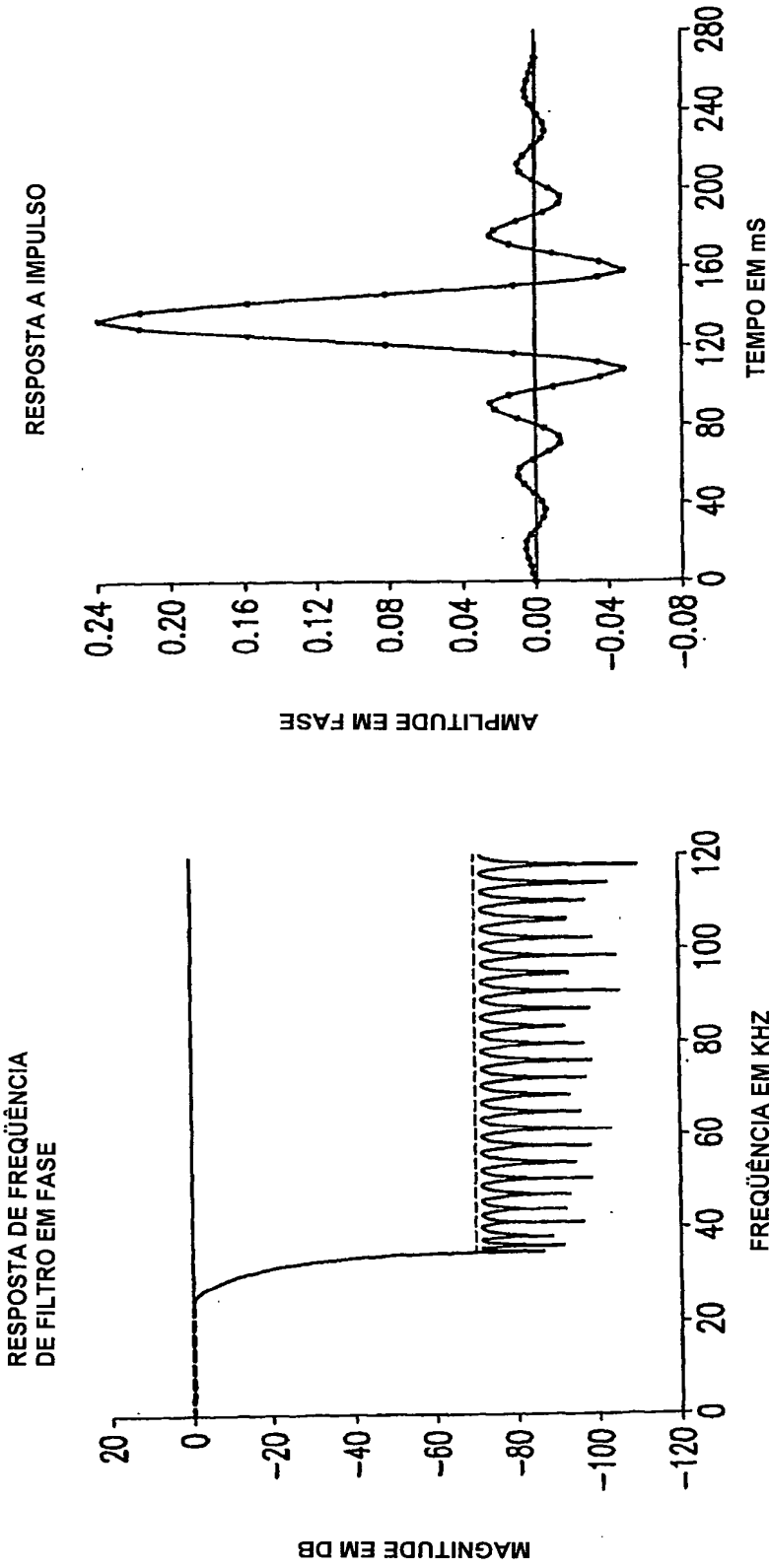


FIG.37



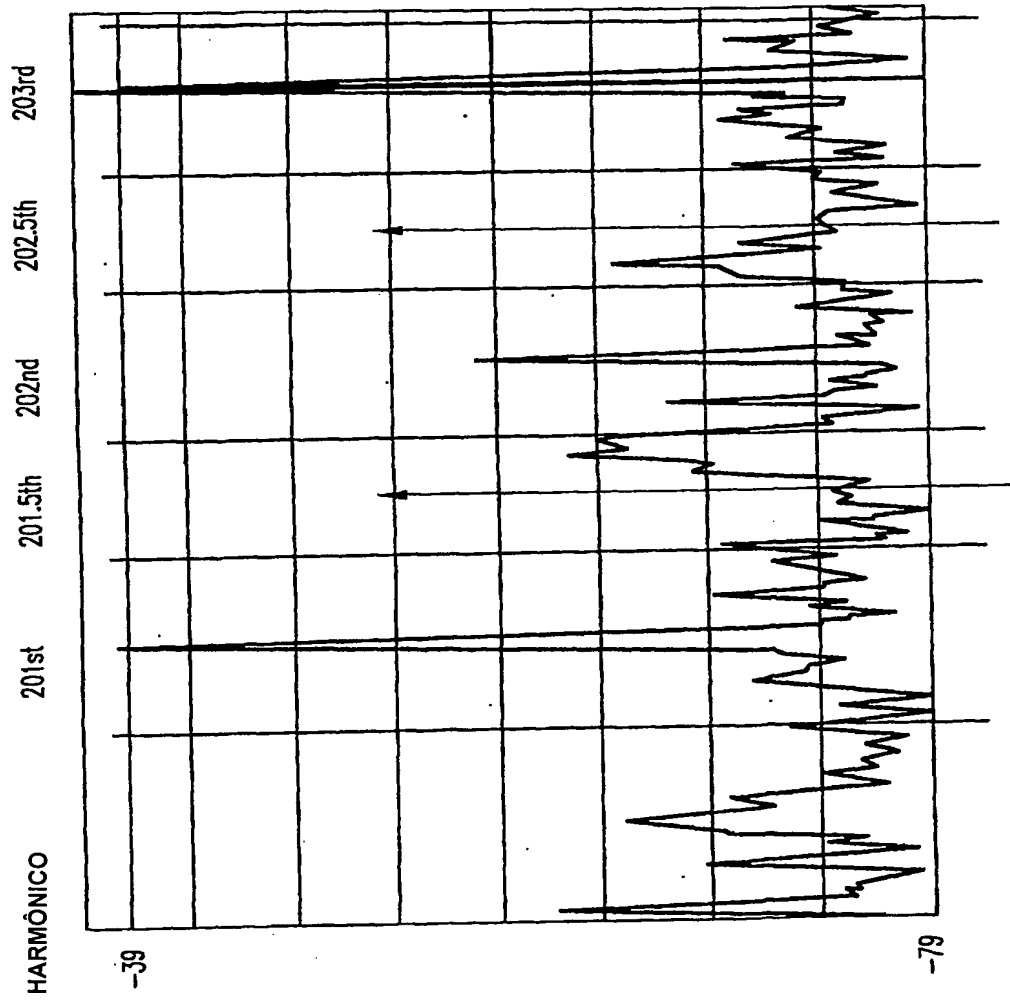


FIG. 39

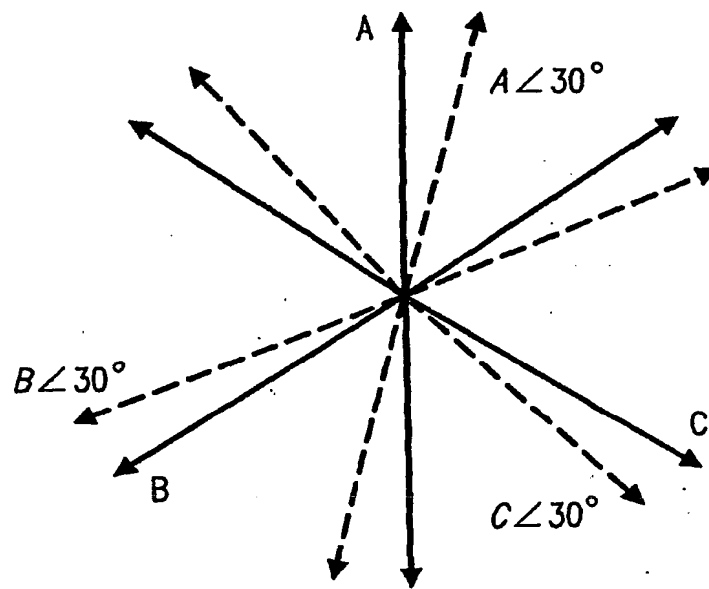


FIG.40

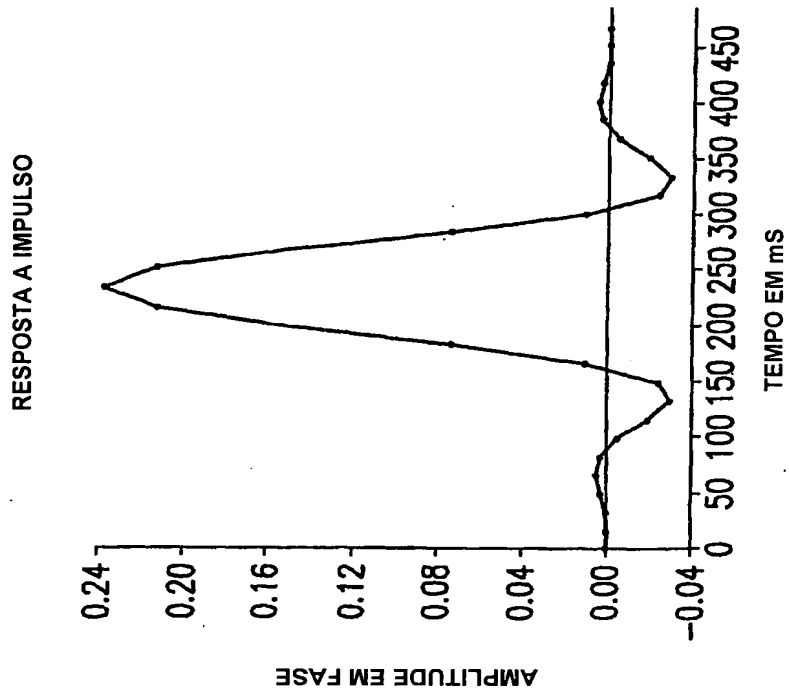


FIG.41B

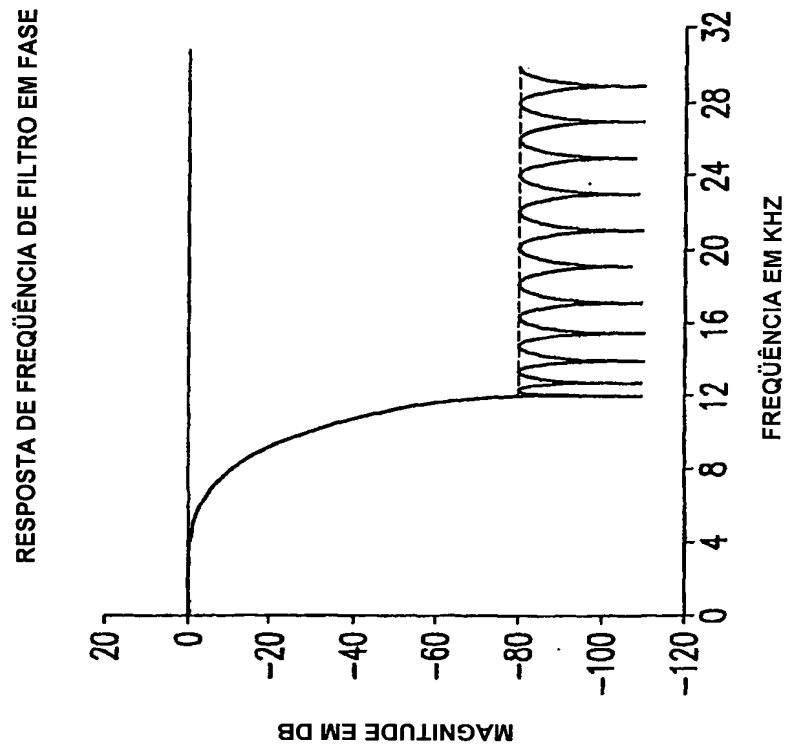


FIG.41A

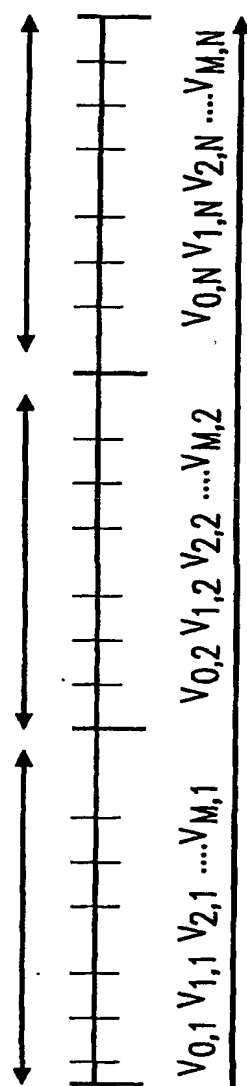


FIG. 42

RESUMO“APARELHOS PARA MEDIÇÃO DE MULTI-CANAL DE ELETRICIDADE”

Em um aspecto, a invenção inclui um dispositivo para medir
5 uso de eletricidade, incluindo: meio para desconexão remota por comunicação
de linha de potência; meio para detecção de roubo de eletricidade; meio para
detecção de fraude; e meio para detecção de tensão inversa. Em outro aspecto,
a invenção inclui um aparelho para medição de multi-canal de eletricidade,
incluindo: (a) uma cabeça de medidor operável para medir uso de eletricidade
10 para uma pluralidade de linhas de consumidor de eletricidade; (b) um
transponder operável para transmitir dados recebidos da cabeça de medidor
por comunicação de linha de potência para um computador localizado
remotamente, e para transmitir dados recebidos por comunicação de linha de
potência do computador localizado remotamente para a cabeça de medidor; e
15 (c) um módulo de controle de carga operável para atuar conexão e desconexão
de cada um de uma pluralidade de relés, cada relé da pluralidade de relés
correspondendo a uma da pluralidade de linhas de consumidor de eletricidade.