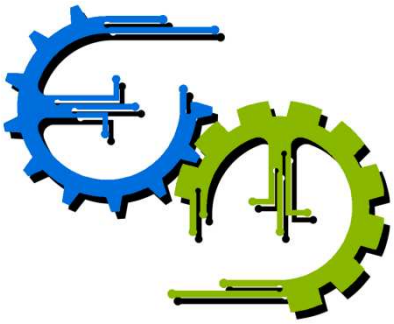
TRABALHO DE GRADUAÇÃO

MODELAGEM E SIMULAÇÃO DE PARQUE FERROVIÁRIO

Por,
Ivan Caetano Leão de Souza
Yuri Vicente

Brasília, julho de 2014



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

MODELAGEM E SIMULAÇÃO DE PARQUE FERROVIÁRIO

POR,

Ivan Caetano Leão de Souza
Yuri Vicente

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Gerson Henrique Pfitscher, UnB/ ENE
(Orientador) _____

Prof. Walter de Britto Vidal Filho, UnB/ ENM _____

Prof. Guilherme Caribé Carvalho, UnB/ ENM _____

Brasília, Julho de 2014

FICHA CATALOGRÁFICA

IVAN, CAETANO LEÃO DE SOUZA; YURI, VICENTE.

Modelagem e Simulação de Parque Ferroviário,

[Distrito Federal] 2014.

x, 102p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2013). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Tempo Real 2.Simulação
3.Automação 4.Maquete Ferroviária

I. Mecatrônica/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SOUZA, I. C. L.; VICENTE, Y., (2014). Modelagem e Simulação de Parque Ferroviário. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 01, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 102p.

CESSÃO DE DIREITOS

AUTORES: Ivan Caetano Leão de Souza, Yuri Vicente

TÍTULO DO TRABALHO DE GRADUAÇÃO: Modelagem e Simulação de Parque Ferroviário.

GRAU: Engenheiro ANO: 2014

Ivan Caetano Leão de Souza
SHCES 1209, Ap. 201 Bl. C – Cruzeiro
70.658-293 – Brasília – DF – Brasil

Yuri Vicente
Qd 207, Lt 4, Ap. 701D – Águas Claras
71.926-250 – Brasília – DF – Brasil

AGRADECIMENTOS

Agradeço primeiramente a meus pais, Eleuza e Sebastião, pelo suporte que me deram a vida toda, bem como toda minha família.
Agradeço aos amigos pelo apoio e um abraço especial ao Claudio Roberto que me ajudou bastante.
Por fim, agradeço a Deus pela oportunidade.

Ivan Caetano Leão de Souza

Agradeço a Deus pelo dom da vida e por todos os momentos que me carregou em seus braços.

A Universidade de Brasília e a Universidade Federal de Santa Catarina, por abrirem meus olhos para um mundo de infinitas possibilidades.
Ao Professor e Orientador Gerson, por ter aceitado a tarefa de nos conduzir com seus conhecimentos e experiências.
Ao meu Pai Edison pelo amor e exemplo de homem íntegro, em todos os momentos.
A minha Mãe Rosimari pelo amor incondicional e sua força.
Aos meus tios Nunu e Cléia por sempre pensarem e se preocuparem comigo.
A minhas irmãs Dalci e Mari, pela amizade e doçura mesmo na distância.
A minha esposa Alessiana que com tanto amor e carinho me tornou uma pessoa melhor. Feliz coincidência esse nosso destino.
Aos meus amigos de Brasília e Florianópolis, do Banco do Brasil, dos tempos da escola ou aonde eu tive a felicidade de conhecê-los.
E a todos aqueles que desta caminhada fazem ou fizeram parte.

Yuri Vicente

RESUMO

Este projeto consiste na idealização de um sistema automático e seguro que permita o escalonamento de trens sobre uma malha viária simplificada. A estrutura do sistema consiste de uma aplicação em tempo real voltada para alto desempenho e confiabilidade uma vez tratar-se de uma aplicação concorrente. O sistema viário é constituído por quatro trens, sendo três de uso geral e o quarto um trem de manutenção que efetua a limpeza das vias.

Palavras Chave: trem, simulação, modelagem, maquete.

ABSTRACT

This project involves the idealization of an automated system that allows the secure and scheduling trains on a simplified highway system. The system structure consists of a real time application oriented for high performance and reliability once that it was a concurrent application. The train system consists of four trains, three for general use and the fourth that a train maintenance performs the cleaning of the rails.

Keywords: train, simulation, modeling, model.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

B	Bytes	[GB]
v	Diferença de potencial elétrico	[v]
H	Frequência	[Ghz]
s	Tempo	[s]
V	Velocidade	[km/h]
v	Velocidade	[m/s]

Siglas

ADA	*Linguagem de Programação ADA*
ANSI	*American National Standards Institute*
ARM	*ADA Reference Manual*
C	Linguagem de Programação C
CI	Circuito Integrado
CN36	*Centronics 36 pins*
CPU	*Central Processing Unit*
DB35	*D-subminiature 25 pins*
DSL	*Domain Specific Language*
ECP	*Enhanced Capabilities Port*
EMF	*Eclipse Modeling Framework*
EPP	*Enhanced Parallel Port*
FIFO	*First In First Out*
GCC	*GNU Compiler Collection*
GNAT	*GNU NYU Ada Translator*
GNU	Sistema Operacional Unix
GPS	*GNAT Programming Studio*
ISO	*International Organization for Standardization*
SO	Sistema Operacional
SSP	*Standard Parallel Port*
UML	*Unified Modeling Language*

1 INTRODUÇÃO

1.1 O PROBLEMA DO ESCALONAMENTO EM TEMPO REAL

Sistemas de tempo real são extremamente comuns, pois são constituídos de qualquer aplicação que interage diretamente com usuários. Exemplos usuais desse tipo de sistema são os sistemas de ticket de estacionamento rotativo ou de aquecimento central. Em ambos os casos, existe uma especificação relativamente rígida quanto a execução das tarefas em um determinado tempo. Entretanto, *TimeSys Corporation* (2002) indica muitas outras aplicações sofisticadas em tempo real que podem ser elencadas, a exemplo:

- Controle de tráfego aéreo;
- Controle de processos em plantas industriais, laboratórios e usinas de geração de energia;
- Atividades remotas, como exploração espacial e submarina ou atividades em áreas de risco ou contaminadas;
- Monitoração e operações remotas de pacientes hospitalares;
- Aplicações em realidade virtual como jogos.

Conforme Nissanke (1997), podemos distinguir um sistema comum de um sistema em tempo real com base em seu resultado. Um sistema comum tem como premissa estar logicamente correto, ou seja, o resultado final deve estar funcionalmente coerente com a proposta do sistema, exemplificando que entradas corretas produzem saídas corretas. Para um sistema em tempo real, a noção de logicamente correto não é suficiente para assegurar a sua funcionalidade; além de estarem logicamente corretos, estes sistemas possuem ainda uma demanda muito grande por um estreito requisito de tempo. Assim, obter os resultados necessários e corretos no tempo correto é o que define um sistema em tempo real.

Nesta última tipologia de sistema, o não cumprimento do quesito temporal, chamado de *deadline*, em uma ou mais especificações, pode gerar severas consequências como, por exemplo, a falha do sistema.

Como método de diferenciação para as diferentes exigências de um sistema, Nissanke (1997) efetua a divisão em Sistema Real do tipo *Soft*, *Firm* e *Hard*, conforme segue:

- O Sistema Real tipo *Soft* é aquele no qual a degradação do desempenho em relação ao tempo, como a perda de um ponto de controle, não representa a falha do sistema, e sim uma perda na qualidade do mesmo. O quesito temporal neste caso não é crítico para o resultado, mas sim um fator

fortemente desejado. Como exemplo, podemos indicar um *software* para edição de textos ou de controle de iluminação pública.

- O Sistema do tipo *Firm* é caracterizado por um sistema que comporta um certo nível de degradação, ou seja, a perda de alguns *deadlines* não acarretam necessariamente no colapso do sistema, porém a continuidade destas perdas podem provocar catástrofes e a falha do sistema. Como exemplo, podemos indicar um sistema controlador de trafego aéreo.

- O Sistema do tipo *Hard* é aquele no qual a perda de um *deadline* é extremamente comprometedora, não sendo aceitável em nenhum nível, no qual os resultados são catastróficos e provocam a completa falha do sistema. Nestes sistemas, o quesito temporal é extremamente relevante e não pode ser dissociado da atividade fim. Como exemplo, temos de controle de freios e muitos dos sistemas espaciais.

Um outro aspecto importante para sistemas em tempo real, que é abordado por Buttazzo (2011), trata da questão dos eventos. Um evento é qualquer acontecimento que provoque uma mudança ou alteração no fluxo de controle, e que termina por demandar uma readequação por parte do sistema. Podemos exemplificar uma mudança de parâmetro ou interrupção por parte de um sensor ou de um acionamento direto por operador.

Os eventos podem ser do tipo síncrono (quando seu acontecimento pode ser previsto pelo sistema) ou assíncrono (quando podem ocorrer aleatoriamente); podem também ser divididos entre periódicos (que ocorrem em intervalos regulares), aperiódicos (que não possuem regularidade) e esporádicos (que ocorrem de forma esparsa, com raridade).

O projeto e implementação de um sistema em tempo real requer grande atenção nos quesitos de *hardware* e *software* para a aplicação em desenvolvimento. Neste aspecto, Buttazzo (2011) apresenta a forte ligação entre o escalonamento de tarefas e o desempenho resultante.

A otimização dos recursos disponíveis está ligada principalmente a capacidade de processamento computacional e de escalonamento e alocação de tarefas no grupamento finito de processadores. Assim, uma das bases para sistemas de tempo real consiste na utilização racional e eficiente dos limitados recursos computacionais.

De uma forma generalizada, uma tarefa pode estar em três estados distintos: executando, quando está em operação na CPU; pronta, quando está aguardando para ser executada e bloqueada, quando está à espera de um evento para dar continuidade a sua execução.

Como os recursos computacionais são escassos e geralmente compostos por poucas CPU, a maioria das tarefas está ou bloqueada ou pronta para execução. Segundo Buttazzo (2011), os sistemas escalonadores utilizam-se de diversos critérios, como:

- Taxa de utilização da CPU, a fração de tempo na qual ela está sendo utilizada;

- *Throughput*, a quantidade de processos executados por unidade de tempo;
- *Turnaround*, tempo entre a chegada de um processo para execução e o término da execução;
- Tempo de resposta, tempo entre a chegada e início da execução de um processo;
- Tempo de espera, soma de todos os períodos em que o processo aguardou para ser executado.

Com base nestes critérios, um sistema de política de escalonamento para aplicações em tempo real deve observar a maximização da taxa de utilização da CPU e da produtividade (*throughput*), com o maior número de tarefas executadas por unidade de tempo, e a minimização do tempo de resposta e de espera, para garantir que todas as tarefas, em especial as críticas, sejam executadas adequadamente, respeitando seus respectivos *deadlines*.

Os maiores problemas verificados no sistema escalonador estão relacionados aos eventos de natureza assíncrona e aperiódica pois, para processos desta categoria, não é possível realizar previsões, fazendo-se necessária a utilização intensa dos recursos computacionais, o que pode comprometer outras tarefas importantes inerentes ao sistema.

1.2 OBJETIVOS

Alguns casos recentes de falha em sistemas de tempo real em trens, como o ocorrido na Espanha em 2013, conforme relatado pelo jornal espanhol El País (2013), que suscita falhas nos sistemas de controle automático, o qual supostamente não diminuiu a velocidade da composição em uma curva de cerca de 190 para 80 km/h, e a necessidade de melhoras de segurança em tais sistemas. E também os relatos de Feng e Keping (2008), no qual existe uma grande atenção no desenvolvimento atual de sistemas avançados para fornecimento de informações em tempo real de posição e velocidade de trens. Assim propomos a utilização de uma malha viária como ambiente adequado para a realização de estudos e análises de desenvolvimento de um sistema em tempo real.

Ainda segundo Feng e Keping (2008) em seu artigo, além que questões de segurança e confiabilidade um bom sistema permite ainda uma redução no consumo de energia e permite um aumento na capacidade operacional.

Historicamente temos verificado muitos esforços para a modelagem e criação de sistema robustos para malhas ferroviárias. Seria o caso de Szpigel (1973), que propôs um sistema de escalonamento de trens baseado na combinação de um programa integrador e do sistema *branch and bound* para minimizar tempos em transito e avaliar pontos de gargalo. Posteriormente Kraft (1987) apresentou uma abordagem não linear com o objetivo de resolver os conflitos entre trens e minimizar atrasos. Carey e Lockwood (1995) desenvolveram uma abordagem de decomposição para resolver o

agendamento de trens e caminhos em uma rede ferroviária com uma única via e faixas nos dois sentidos.

Conforme apontado por Dorfman e Medanic (2004), os métodos até então estipulados tinham como objetivo obter a solução ótima, seja por métodos de programação inteira ou técnicas não lineares. E estes apresentaram duas problemáticas básicas:

- O cálculo para cada trem e desvio ainda é excessivamente longo para uma grande malha viária;
- No caso de atraso ou falha de um trem é necessário o recalculo e reprogramação de todo o sistema.

1.2.1 ESPECIFICAÇÃO DO OBJETIVO

Tendo em vista os conhecimentos obtidos durante o curso de Engenharia Mecatrônica acerca de sistemas de tempo real o seguinte trabalho tem como metas gerais os seguintes itens:

- Projetar e implementar técnicas de escalonamento em tempo real;
- Construir um laboratório na forma de uma linha ferroviária para realizar testes com os algoritmos.

E como objetivos específicos:

- Efetuar um estudo detalhado dos recursos necessários como: ferramenta de modelagem, linguagem de programação, recursos gráficos e demais técnicas para o desenvolvimento da simulação;
- Confecção de um modelo para a implementação da malha viária;
- Criação de uma simulação computadorizada e gráfica para a malha viária, seguida da análise de desempenho da mesma;
- Implementação de um circuito eletrônico em uma maquete viária simples para validação da simulação.

1.3 PROPOSTA

O sistema viário proposto para o projeto é uma idealização do tipo maquete, ou seja, possui como embasamento estrutural no dimensionamento e concepção a abrangência reduzida a uma aplicação local. Tal simplificação é realizada devido a possibilidade assim de uma análise mais genérica e irrestrita de fatores podendo assim ser composto uma diversidade de funcionalidades.

O sistema é limitado, em sua constituição, a quatro trens, sendo três de uso geral e o quarto um trem de manutenção que efetua a limpeza das vias. Esta limitação é realizada para permitir uma maior liberdade de atuação dentro do espaço geográfico de uma maquete ferroviária.

1.3.1 O CIRCUITO

Pode-se verificar, na figura (1.1), o trajeto completo no qual será implementado e desenvolvido o projeto. Nesta figura, é possível verificar os pontos de partida inicial dos trens e todas as intersecções de pistas.

O Parque Ferroviário, no estilo maquete, é constituído por três circuitos ovais principais e uma área de garagem, com pontos de estacionamento para cada trem. A concepção das linhas é oval, em vez de pistas lineares, como é esperado para uma modelagem real de malha viária.

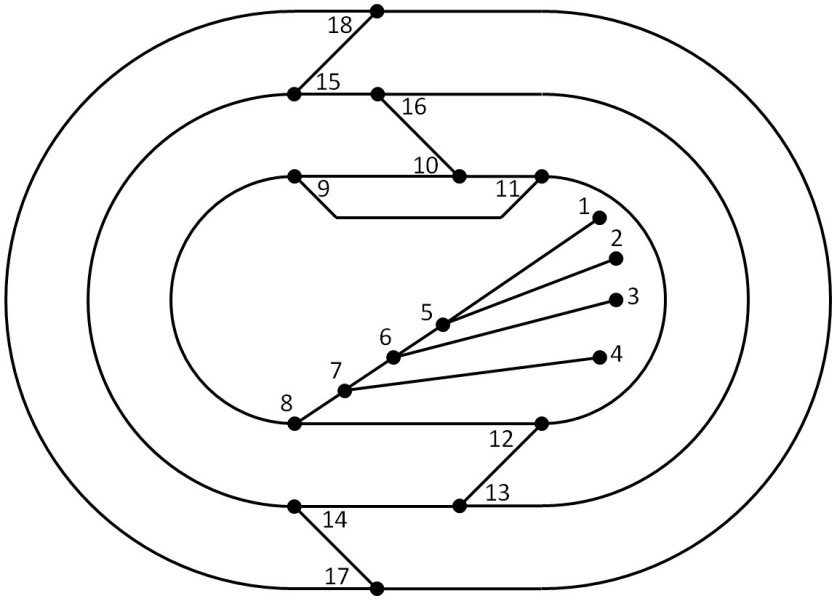


Figura 1.1. Trajeto proposto com pontos de início e intersecção.

Entretanto, é possível verificar, na figura (1.2), o primeiro mapa da linha de metrô subterrâneo de Londres, UK. Citado em Barman (1979), o mapa foi confeccionado em 1908 por autor desconhecido e apresenta a malha viária da época, na qual verifica-se a presença de circuitos circulares e lineares intercalados, o que justifica a escolha de circuitos iniciais ovalados para este trabalho por seu contexto histórico de iniciação de novas metodologias de transporte.

Figura 1.2. Metrô subterrâneo de Londres (1908).

1.3.2 OS DESVIOS

Na figura (1.3), é possível verificar os fluxos que podem ser efetuados em cada passagem de nível e os sentidos possíveis para cada trecho. Além dos desvios, estão caracterizados ainda os quatro pontos correspondentes às posições de garagem de cada trem.

Neste circuito, garante-se uma liberdade de trânsito bidirecional nos desvios, onde existe uma multiplicidade de rotas de seis movimentos por desvio, aumentando a complexidade do projeto, abrangendo, porém, maior gama de aplicabilidades.

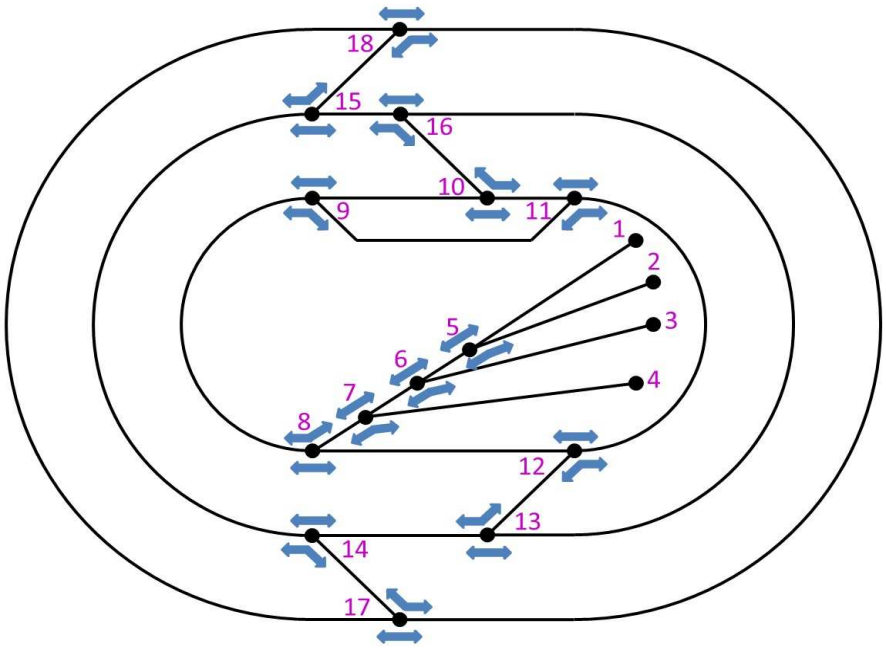


Figura 1.3. Trajeto com fluxos de movimento dos desvios.

1.3.3 OS TRAJETOS

A confecção de uma proposta final no formato de maquete utilizou como base o produto de mercado da Empresa Frateschi Ltda., a qual está no mercado de ferromodelismo há cerca de 40 anos e disponibiliza em seu catálogo os aspectos técnicos e referenciais de estrutura para a elaboração da composição, ou seja, dimensionamento e conexão de trilhos e desvios. Com base nestas informações, foram compostos os circuitos e seus respectivos trajetos, observados na figura (1.4).

Observa-se ainda, na figura (1.4), um detalhamento da malha. Em rosa (A), tem-se o trem 1 e seu respectivo trecho oval, bem como sua posição de garagem e início. Em azul (B), tem-se o trem 2 e seu respectivo trecho oval, bem como sua posição de garagem e início. Em verde (C), o trem 3 e seu respectivo trecho oval, bem como sua posição de garagem e início. Finalmente, em alaranjado (D), observa-se a posição de saída do trem de manutenção.

Em amarelo estão os trechos compartilhados, representando, principalmente, os desvios que caracterizam os trechos em concorrência pelos trens durante seus deslocamentos.

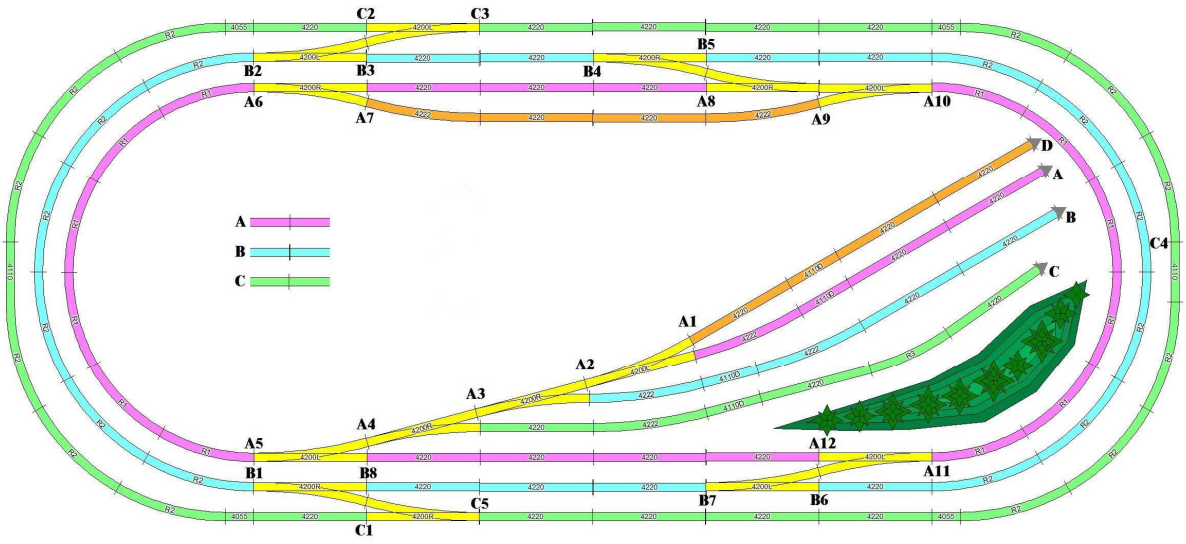


Figura 1.4. Trajeto com trechos e trilhos.

1.3.4 RESTRIÇÕES DO SISTEMA E PRINCÍPIOS DE MODELAGEM

Para uma correta abordagem de um sistema ferroviário, alguns aspectos devem ser elencados com o objetivo de firmar as configurações básicas e os principais condicionantes do sistema. Pode-se verificar em Martim (1999) alguns critérios funcionais que permitem efetuar esta qualificação simples e efetiva para modelagem.

Martim (1999) define as configurações em três elementos básicos, que são: o projeto civil, de sinalização e de operação. O projeto civil estaria direcionado à análise dos efeitos sensíveis ao corpo

utilizador do sistema, sejam pessoas ou cargas, e define assim quesitos como velocidade máxima e frenagem, resultando em elementos como o perfil de velocidade; avalia também questões referentes às demandas e utilidade do sistema. O projeto de sinalização é referente à infraestrutura das vias e aspectos de coordenação dos trens na mesma; deste projeto, obtemos regras de exceção, tal qual distâncias mínimas seguras e também de capacidade do sistema em seu todo. O projeto de operação seria o responsável final por quesitos como velocidade, distância mínima entre trens, compartilhamento de trechos, desvios e seus subsequentes conflitos.

Na estratégia simulacional, o objetivo foi focado nas regras operacionais do sistema, e assim obtive-se as características abaixo descriminadas, como os fatores mais relevantes para o sucesso do sistema.

- Plano de Alocação de Trens: projeto de linhas de trens com priorizações e tempo de trajeto.
- Velocidades: duas velocidades para trens (avanço e parado).
- Tempo de Manutenção de Trilho: máximo tempo de utilização do trilho antes de efetuar limpeza do trecho.
- Tempo de Segurança: intervalo mínimo de bloqueio após a saída de um trem, antes da entrada de outro trem em um desvio ou trecho.
- Sem Colisões: dois trens no sistema não podem colidir.
    - Dois trens não podem entrar em um mesmo trecho com sentidos opostos.
    - Dois trens não podem entrar no mesmo desvio.
- Sem Descarrilamento: desvios devem estar corretamente posicionados para a passagens de trem.
    - Dois trens não podem entrar no mesmo desvio.
    - Um trem não pode entrar em um desvio não pronto.
    - Um desvio não pode chavear na presença de um trem.

2 MODELAGEM

2.1 MODELAGEM

Conforme Pidd (2004), a modelagem de sistemas é uma tarefa que envolve a utilização de modelos e estudo interdisciplinar para a concepção e construção de aplicação em TI. O processo de modelagem utiliza-se de técnicas e princípios que permitem uma análise dos requisitos e funções a serem implementados.

As primeiras funções de modelagem para sistema iniciaram na década de 50 e tem como um dos primeiros pioneiros Young e Kent (1958), que afirmaram a necessidade de implementação de um sistema que permitisse uma estruturação de um problema de processamento de dados sem uma vinculação direta com o mecanismo que a irá implementar.

Para Pinkava (1988) o termo modelagem de sistema pode ser referenciado a vários significados distintos:

- Uso do modelo para conceituar e construir sistemas;
- Estudo interdisciplinar da utilização destes modelos;
- Modelagem, análise e esforços de projeto;
- Modelagem e Simulação de sistemas, tais como a dinâmica do sistema;
- Sistemas de linguagem de modelagem específica

De acordo com o escopo e escala de modelagem Pinkava (1988) diferencia a modelagem em quatro formatos: Modelagem Funcional, figura (2.1), que consiste em uma representação baseada nas funções e suas ações, atividades, processos e operações; Modelagem Estrutural, focada na arquitetura que será utilizada como suporte a estrutura do sistema; Modelagem ao Processo do Negócio, é uma modelagem orientada a plataforma de negócio com foco na qualidade e eficiência do negócio; Modelagem Empresarial, consiste em uma modelagem das estruturas empresariais com a finalidade de suportar as atividades de uma organização.

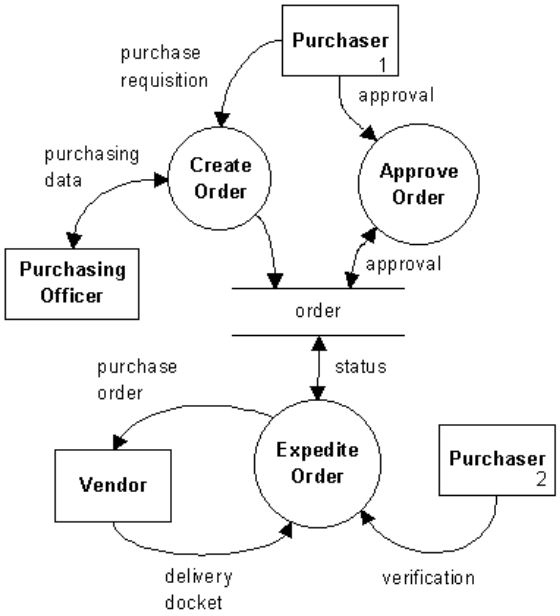


Figura 2.1. Exemplo de modelo funcional, ordem de compra genérica.

Os modelos para desenvolvimento de um *software* são abordados por Mazzola e Farines (2004), e seriam a representação abstrata do processo de desenvolvimento e a condução de suas etapas e relacionamentos. O autor aborda quatro modelos conhecidos para o desenvolvimento:

- Modelo Queda d'Água, figura (2.2), sendo o modelo mais simples com sequência linear no qual é possível identificar com precisão o início e fim de cada etapa. Neste modelo é possível avanços e retornos nas etapas do trabalho de acordo com a necessidade de ajustes ou reimplementação.

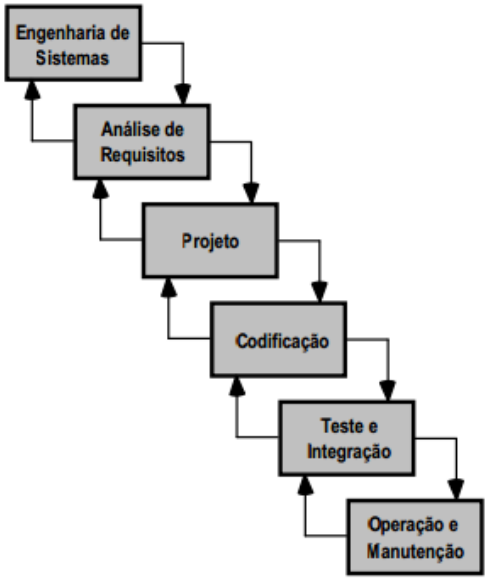


Figura 2.2. Representação do modelo Queda d'Água.

- Prototipação, figura (2.3), neste modelo é gerado um protótipo inicial com o intuito de garantir uma análise e avaliação do resultado esperado antes de iniciar o projeto definitivo.

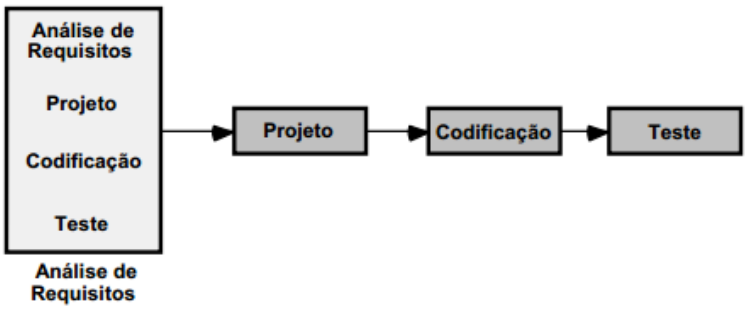


Figura 2.3. Representação do modelo de Prototipação.

- Desenvolvimento Iterativo, figura (2.4), neste modelo o desenvolvimento é efetuado de forma incremental, e cada novo passo permite a obtenção de um sistema mais completo e robusto. Ainda existe uma garantia de funcionalidade de cada versão produzida.

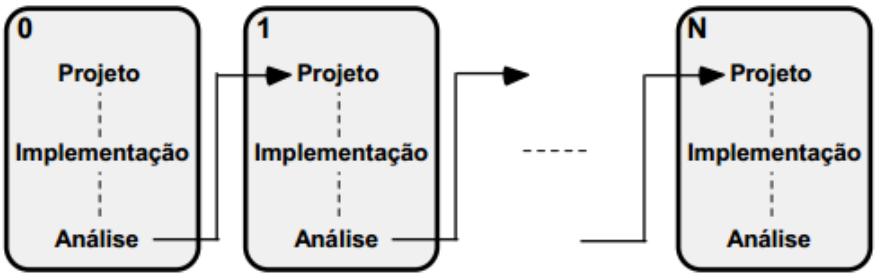


Figura 2.4. Representação do modelo de Desenvolvimento Iterativo.

- Modelo Espiral, figura (2.5), consiste em um sistema cíclico de desenvolvimento de protótipos até a obtenção de um resultado final, entretanto cada ciclo requer uma nova identificação dos objetivos e análise dos riscos envolvidos. Ao final de cada etapa construtiva é executada uma revisão que consiste da avaliação e implementação de um planejamento para as próximas etapas.

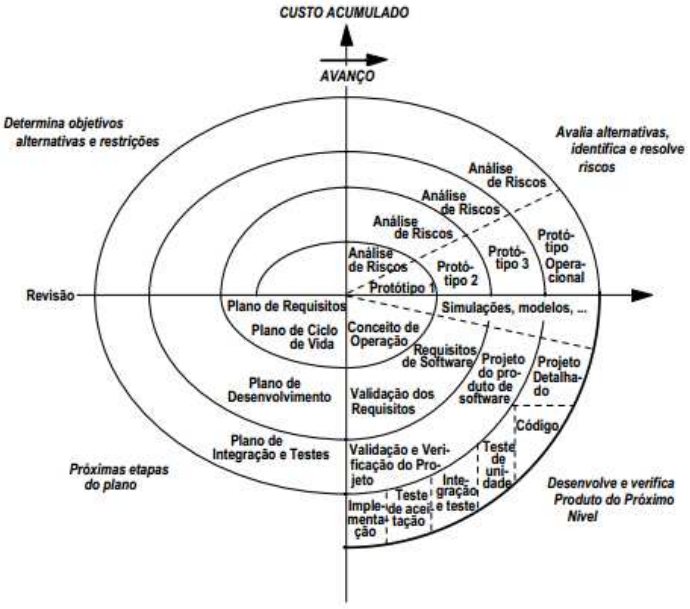


Figura 2.5. Representação do modelo Espiral.

2.2 A FERRAMENTA UML

A *Unified Modeling Language* é uma linguagem de modelagem não proprietária largamente utilizada para o desenvolvimento de softwares, conforme Booch, Rumbaugh e Jacobon (2006), trata-se de uma linguagem de terceira geração orientada a objeto. O UML em si não é caracterizado por ser uma metodologia para o desenvolvimento, e sim uma ferramenta que permite melhor visualização e desenho das funções e processos do sistema. De forma geral o UML permite aos desenvolvedores a visão dos trabalhos sobre uma notação gráfica e uma estratégia semântica unificada.

Para Booch, Rumbaugh e Jacobon (2006) os objetivos fundamentais do UML são: especificação, documentação e maior visualização lógica para o desenvolvimento completo de um sistema de informação. Sendo um padronizador dos métodos de modelagem.

Conforme Larman (2007) a ferramenta UML foi desenvolvida em meados da década de 90 e possuía como objetivo ser um direcionador para o desenvolvimento de programas com o nova padrão de orientação a objeto desenvolvido na década anterior. Seus idealizadores são Grady Booch, Ivar Jacobson e James Rumbaugh enquanto funcionários da Empresa Rational Software. Em 2000 a linguagem foi aceita pela ISO e largamente divulgada. Desde 2005 a versão em vigor é a 2.x.

O UML através de seus diagramas gráficos, descrito por Bezerra (2007) podem representar duas diferentes visões do modelo, sendo elas:

- Estática, ou estrutural, enfatiza a estrutura do sistemas através de seus objetos, atributos, operações e relacionamentos.

- Dinâmica, ou comportamental, enfatiza o dinamismo do sistema, com a determinação das colaborações e estados dos componentes.

Destas visões de modelagem Bezerra (2007) diferencia as estruturas de diagramas em duas categorias distintas, sendo elas:

- Diagrama de Estrutura, representam os componentes presentes no sistema e são utilizados para documentar a arquitetura de sistemas de software.
- Diagrama de Comportamento, representam as ações que devem ocorrer na modelagem, descrevendo assim as funcionalidades do sistema de software.

Medeiros (2004) nos chama atenção a importância do Diagrame de Classe, que seria um subtipo dos Diagramas de Estrutura e são amplamente utilizados por desenvolvedores de software orientados a objeto. Este diagrama, exemplo na figura (2.6), é a principal ferramenta para modelagem orientada a objeto e tem função tanto para modelagem sistemática da aplicação quanto para detalhamento de atributos. Este diagrama representa para os programadores os dois principais itens, as classes e as interações entre os objetos a serem programados. Outro detalhe é constituído da facilidade de geração de códigos em linguagens de programação, a exemplo C, Java, etc, diretamente a partir dos modelos de diagrama de classe, o que permite grande avanço na tarefa de desenvolvimento de códigos.

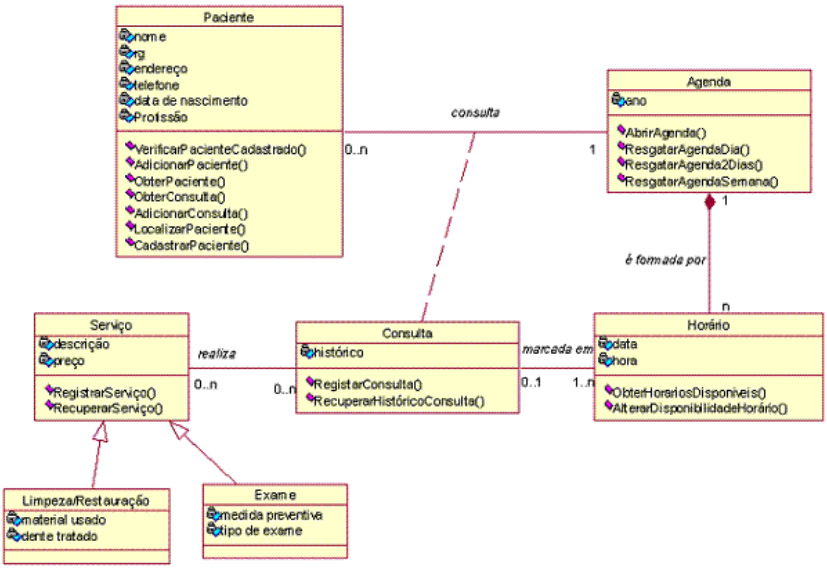


Figura 2.6. Representação do Diagrama de Classe.

2.3 MODELAGEM UML

A proposta para a confecção de um modelo, que será utilizado como direcionador na criação do código, trata-se da confecção de um diagrama de classe UML que representa as principais funcionalidades do sistemas, bem como dos atributos e comunicação do mesmo.

A concepção original trata-se da estruturação em módulos, nas quais cada um destes representaria uma funcionalidade no sistema. Tal método se faz interessante pois permite a geração de um modelo com múltiplas características distintas e que pode assim ser implementado em partes, ou ainda apenas parcialmente implementado.

Uma vez que o planejamento para a construção do código é feita na estrutura de desenvolvimento iterativo, ou seja, o programa será gerado gradualmente a medida que são desenvolvidas as habilidades com a ferramenta e linguagem a ser utilizada. A proposta para o modelo é gerar uma estrutura a qual pretende-se atingir, entretanto sem caracterizar a obrigatoriedade da confecção de todas as partes.

Assim o objetivo principal na geração do diagrama de classes é a criação de uma representação que permita a confecção interativa do código porém sem tornar-se um limitar de funcionalidades ou uma estrutura única e fixada como meta final de projeto.

2.3.1 FERRAMENTA DE MODELAGEM

Para o processo de escolha de uma ferramenta adequada para o desenvolvimento de uma modelo foi elaborado, conforme Tabela (2.1), uma relação de possíveis aplicação que possuíssem uma interface gráfica objetiva e fácil de manipular e principalmente que seja compatível com a geração de códigos direta para a linguagem de programação ADA.

Com base no estudo de viabilidade para estas escolhas optamos por utilizar o *Papyrus*, por caracterizar um software livre que opera sobre a Plataforma Eclipse e que possui uma excelente interface gráfica para o desenvolvimento do Diagrama de Classe UML.

Tabela 2.1. Alguns *softwares* de desenvolvimento de UML compatíveis com ADA.

Software	Licença	SO	ADA	Observações
ArgoUML	Livre	Java-Eclipse (Unix/Win/Mac)	Sim	Não está completamente implementado.
Artisan Studio (Atego)	Comercial	Windows	Sim	Recomendado pelos desenvolvedores ADA.
Dia	Livre	C (Unix/Win/Mac)	Sim	Não está totalmente desenvolvido, release atual 0.97.
Enterprise Architect	Comercial	Windows	Sim	Licença Comercial e implementação para Windows.
Papyrus	Livre	Java-Eclipse (Unix/Win/Mac)	Sim	Apenas UML2. Excelente interface Gráfica.
Rational Rhapsody	Comercial	Unix/Win/Mac	Sim	Licença Comercial, desenvolvido pela IBM.
Umbrello UML Modeller	Livre	Unix	Sim	Software livre muito usado por desenvolvedores em Linux.
Visual Paradigm for UML	Comercial	Java	Sim	Licença Comercial.

O *Papyrus*, segundo Eclipse (2014) seria "... uma ferramenta integrada para utilização pelo usuário integrada para editar qualquer tipo de EMF, *Eclipse Modeling Framework*, e, particularmente, apoiar UML e linguagens de modelagem relacionados..." ainda de Eclipse (2014) "*Papyrus* também oferece um suporte muito avançada de perfis UML que permite aos usuários definir editores para DSLs baseadas no padrão UML 2. A principal característica da *Papyrus* em relação a este último ponto é um conjunto de mecanismos de personalização muito poderoso que podem ser aproveitados para criar perspectivas *Papyrus* definidos pelo usuário e dar-lhe o mesmo olhar e sentir-se como um editor de DSL "puro"."

Assim o *Papyrus* é uma ferramenta que opera dentro do sistema Eclipse de desenvolvimento e que permite grande customização do ambiente de desenvolvimento. Sua efetiva interface gráfica e sua boa relação com a Linguagem ADA o capacitam como um excelente recurso para o desenvolvimento de modelos.

2.3.2 O MODELO

O modelo desenvolvido no projeto, e que servirá como padrão para a construção do código, conforme figura (2.7), foi desenvolvido com foco em áreas de agrupamento, ou seja, foi efetuado o subdivisão em módulos para uma melhor capacidade de análise e utilização.

Dentro de cada módulo temos a presença de algumas classes primordiais com a função de direcionar o desenvolvimento da aplicação e simplificar a leitura do sistema. Estas foram criadas com a ideia básica a qual se objetiva a aplicação.

Também é apresentado uma sistemática mínima de atributos em cada classe e da representação da comunicação entre as mesmas e consecutivamente entre os módulos.

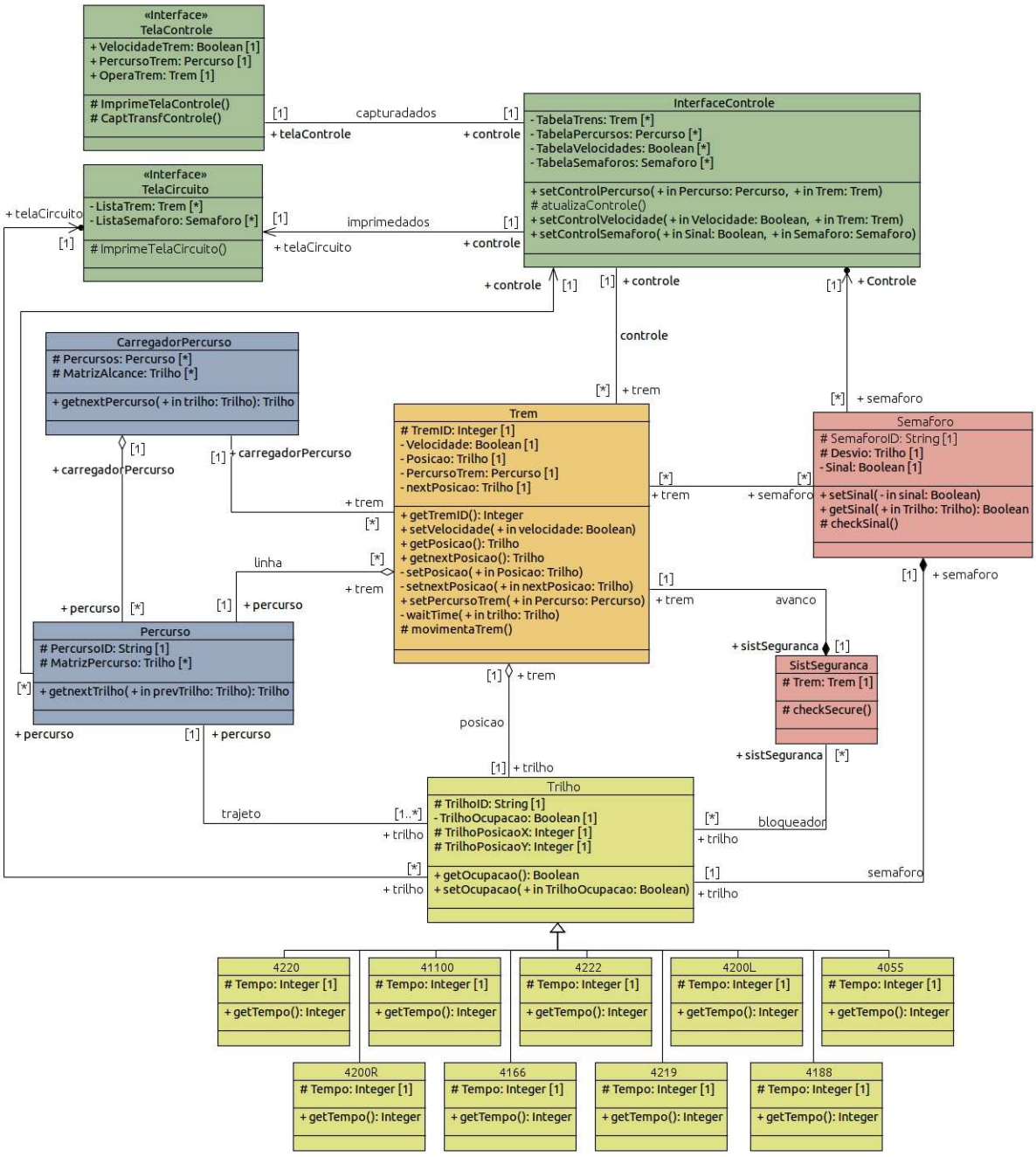


Figura 2.7. Modelo UML: Diagrama de Classes

Os módulos desenvolvidos são representados por cores na figura (2.7), conforme:

- Amarelo: Modelo da parte física do circuito, representa diretamente a estrutura física composta por trilhos e desvios. Representado pela classe trilho para a modelagem de cada trilho do circuito proposto no projeto. Esta classe é composta ainda por um dos trilhos propostos na figura (1.4) e especificado pela Frateschi (2012) em seu catálogo de produtos.

- Azul: Modelo dos percursos especificados na figura (1.4), a princípio três circuitos para os trens e um para o trem de manutenção. Na classe Percurso está especificado a rota a ser executada por cada trem que a utilize. A classe CarregadorPercurso é responsável por localizar o trem e levá-lo até seu percurso definido, está classe é ideal para a adição de funções inteligentes (algarismos de menor caminho, rede neural, árvore de decisão, dentre outros).

- Alaranjado: Modelo do trem propriamente dito, com sua posição, percurso, velocidade e funcionalidades. A classe Trem é determinada como a classe central do projeto, ou seja, onde são armazenadas as principais informações sobre o sistema e de onde temos os principais parâmetros para os demais módulos.

- Vermelho: Modelagem dos requisitos de segurança do projeto, contendo a classe SistSegurança responsável por impedir falhas de segurança e choques entre trens e também a classe Semaforo que representa os desvios sua sistemática de semáforo, com sinais de avanço ou pausa para os trens.

- Verde: Modelo da parte gráfica do sistema e de interface com o operador. Contém uma classe chamada InterfaceControle que armazena as informações para interação com o operador, a classe TelaCircuito que gera uma tela com o posicionamento dos trens e estados dos desvios e a classe TelaControle que fornece informações de velocidade e percurso dos trens bem como a possibilidade de manipulação dos mesmos via entrada de dados.

As classes, atributos e comunicação determinadas no modelo não correspondem a determinação do projeto final em si, porém são utilizadas como suporte para a construção estruturada do código com o intuito de simplificar o processo construtivo com a subdivisão de áreas de interesse.

3 PROGRAMAÇÃO E SIMULAÇÃO

3.1 PROGRAMAÇÃO E SIMULAÇÃO

O processo de desenvolvimento de um programa normalmente é vinculado ao uso de ferramentas e ambientes que permitam o acompanhamento pelo desenvolvedor, iniciando na etapa de codificação, passando pela geração do código e testes do código executável, conforme Sommerville (2007).

Para Sommerville (2007), as principais etapas para a geração de um programa, área está conhecida por Engenharia de Software, são:

- Geração do código fonte, nesta etapa é realizado a codificação do programa, ou seja a escrita do mesmo. Para esta etapa é utilizada uma linguagem de programação e a utilização de editores, os arquivos resultantes deste processo são denominados código-fonte.
- Tradução do código fonte, uma vez que o código-fonte não é executado diretamente pelos processadores faz-se necessário a tradução para o código máquina do processador. Esta tradução atualmente é gerada de forma bastante automática pelos Montadores, para linguagens de baixo nível, e Compiladores, para linguagens de alto nível, e tem como resultado o código-objeto. Nesta etapa além da geração final de um código objeto possui ainda as fases intermediárias de análise léxica, sintática e semântica, responsáveis pela verificação de erros ou irregularidades no código-fonte e, por vezes, em compiladores mais modernos a melhoria do código.
- Editores de ligação, efetuam a combinação entre as diversas partes de código-objeto que possam terem sido geradas para o programa e também a adição de códigos oriundos de bibliotecas, incorporando assim todas as partes referenciadas no código principal. Nesta etapa é gerado o programa executável.
- Operações de depuração, os depuradores são programas de auxílio aos programadores com a finalidade de eliminar, ou reduzir, erros e desperdícios no programa. Normalmente os depuradores executam o programa através de uma interface apropriada que permite uma verificação passo a passo, com a possibilidade de verificação dos estados e por vezes a alteração direta de dados das memórias para teste.
- Carregamento do programa, esta operação é efetuada por um carregador, normalmente pertencente ao sistema operacional que efetua o carregamento do programa no processador para a execução.

Dentre alguns paradigmas de programação apresentados por Wazlawick (2004), pode-se destacar a questão da orientação a objeto, onde em contraste a programação linear temos a existência de uma malha de objetos que interagem e apresentam estados distintos. A abordagem orientada a objeto requer a identificação das entidades do problema, estas entidades são comumente transcritas de forma gráfica, a exemplo do diagrama de classes figura (2.6), para uma definição hierárquica e de relacionamento.

Neste contexto, Wazlawick (2004) determina o termo objeto para descrever uma estrutura bem definida que contém todas as informações necessárias sobre a entidade. Para o autor os objetos determinam uma sistemática completamente diferente para a concepção, programação, análise e manutenção da aplicação. Este evidencia a grande disseminação no mundo profissional desta linguagem, impulsionadas principalmente pelo argumento de ganho na produtividade pelo grau de modularização naturalmente obtido na orientação a objeto e sua capacidade de reutilização de código.

3.2 LINGUAGEM ADA

O ADA trata-se de uma linguagem de alto nível, orientada a objeto que oferece suporte para simultaneidade, multitarefas e trocas de mensagens entre objetos e processos, conforme Burns e Wellings (2007). A linguagem teve seu desenvolvimento direcionado para sistemas embarcados e com requisitos de tempo real.

Seu desenvolvimento, conforme Ada (2012), partiu de outras linguagens, como o Pascal e o C++, teve origem através de um contrato entre o Departamento de Defesa dos Estados Unidos e a companhia francesa *BULL Informatique* entre os anos de 1977 e 1983. O desenvolvedor chefe da linguagem foi o cientista da computação Jean David Ichbiah (1940 – 2007) participante do programa de pesquisa da divisão *CII Honeywell Bull*.

Segundo Ada (2012) a origem do nome ADA provém de Augusta Ada King, Condessa de Lovelace (1815 – 1852), figura (3.1), comumente conhecida por Ada Lovelace. Ada foi uma matemática inglesa famosa por criar o primeiro código de computação em 1842, em suas notas ele descreve uma máquina analítica para calcular Números de Bernoulli, ainda que seu código não tenha sido testado a época.

Figura 3.1. Ada King, Condessa de Lovelace, 1840.

Durante o início de seu desenvolvimento, a linguagem atraiu forte atenção da comunidade de programadores, sua primeira versão operacional foi criada em 1983 3 chamada de Ada 83, nas décadas seguintes a linguagem adquiriu aumento de performance e foi seguida pelos lançamentos Ada 95, Ada 2005 e Ada 2012, referenciado em Ada (2012). A linguagem adquiriu o padrão ANSI em 1983 e sem grandes alterações o padrão ISO em 1987. Ada é a única linguagem orientada a objeto, concorrente e de tempo real no padrão ISO.

Burns e Wellings (2007) ressaltam que, devido a seu extenso suporte e recursos a aplicações críticas, a linguagem passou a ser utilizada não apenas a aplicações militares mas também para projetos comerciais onde falhas podem apresentar consequências catastróficas, a exemplos aplicações em aviação, controle de tráfego aéreo e transporte ferroviário e sistemas espaciais como foguetes e satélites comerciais.

Por Feldman (2012) tem-se como exemplos de utilização atuais temos: o sistema *fly-by-wire* do Boeing 777, no sistema de tráfego aéreo canadense, no sistema de sinalização do sistema ferroviário de alta velocidade francês, e nos trens suburbanos de metrô em Paris, Londres, Hong Kong e Nova York.

Para Burns e Wellings (2007), o sucesso da linguagem Ada é atribuído principalmente aos seguintes fatores:

- Bibliotecas hierárquicas e outras implementações de apoio ao desenvolvimento de software em larga escala;
- Eficiência e robustez no tempo necessário para compilação;

- Implementações para orientação a objeto seguras;
- Apoio em nível de linguagem para programação concorrente;
- Uma abordagem coerente para o desenvolvimento de sistemas em tempo real;
- Implementações de alta performance;
- Mecanismos individuais bem definidos, em particular o subconjunto SPARK para verificações formais do código.

A sintaxe, conforme Burns e Wellings (2007), tende a minimizar as escolhas para efetuar as operações básicas, dando preferência a palavras-chave em inglês ao invés da utilização de símbolos, por exemplo and ao invés de &&, entretanto, mantém o uso de símbolos aritméticos como “+”, “-”, “*” e “/”. Os blocos de código são delimitados por palavras como "declare", "begin", e "end", onde normalmente “end” é seguindo pela descrição do bloco, por exemplo “if … end if”, o que garante em casos de condicionais minimização de erros por finalização de instrução. O ponto e vírgula “;” é responsável pelo término de instruções, entretanto um único ponto e vírgula, sem uma declaração explícita de término não é permitida. Exemplificação na figura (3.2).

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
   Put_Line ("Hello, world!");
end Hello;
```

Figura 3.2. Exemplo de código em Ada, Olá Mundo!

Ada é projetado para o desenvolvimento de sistemas de software de grande tamanho, assim pacotes Ada podem ser compilados separadamente. Pacote específicos do ADA, como por exemplo os de interface também podem ser compilados separadamente, sem a implementação completa para verificar sua consistência. Isso torna possível a detecção de falhas ainda mais cedo, ainda durante a fase de projeto, antes da implementação completa.

Como a concorrência é parte da especificação da linguagem, o compilador pode em muitos casos detectar possíveis *deadlocks*. Este também verifica se há erros ortográficos, visibilidade dos pacotes, declarações redundantes, dentre outros, e pode fornecer avisos e sugestões úteis sobre correções dos erros e melhorias. Ada também suporta verificações em tempo de execução, como por exemplo para proteção contra acesso à memória não alocada, erros de estouro de buffer, violações de alcance, passos excessivos em laços, erros de acesso à matriz e outros erros detectáveis.

O gerenciamento de memória dinâmica do Ada é caracterizado como de alto nível e segurança. Não existem ponteiros genéricos ou sem tipo, não sendo assim possível a declaração de ponteiros indiretamente. Em vez disso, toda a alocação ou desalocação de memória dinâmica deve ocorrer por meio de tipos de acesso explicitamente declarados. Além disso a linguagem fornece verificações de acessibilidade, tanto durante a compilação quanto na execução, o que garante que um apontamento não permaneça ativo além da duração do tipo de objeto a qual faz apontamento.

Em contrapartida, a maioria dos padrões ISO a linguagem Ada, mais especificamente em se manual de referência, ARM, é de conteúdo livre. Assim o ARM é referência comum a todos os programadores, bem como diversas outras documentações sobre o desenvolvimento da linguagem e seus métodos de uso.

3.3 SIMULAÇÃO ADA

A proposta para a construção de uma aplicação foi realizada com base no desenvolvimento do tipo iterativo, descrito na seção 2.1. Desta forma as atividades foram construídas em versão, com o lançamento de sucessivas versões até obter o resultado final.

Neste aspecto construtivo os módulos idealizados no modelo não são diretamente mapeados para o código, isto porque o código inicia em uma versão básica e progride a até a versão completa. Entretanto os módulos são desenvolvidos, nem sempre integralmente, no sistema, com a exceção do módulo amarelo referente a parte física do circuito que foi fortemente simplificado no código pois durante o desenvolvimento foi verificado que a quantidade de detalhamento deste agrupamento não era necessária para a obtenção de um bom resultado.

Inicialmente, os trabalhos de codificação tiveram como base a aquisição de conhecimento e das ferramentas relacionadas a linguagem ADA. Nas etapas seguintes foram desenvolvidas as principais funcionalidades do sistema até atingir o principal fator de inteligência do mesmo, que consiste na estratégia de melhor percurso a ser executada pelos trens.

O projeto é constituído de quatro trens deslocando-se simultaneamente, de forma quase aleatória, pelos circuitos viários. Assim cada trem possui uma trajetória distinta que, certamente devido a estrutura do setor de garagem, possuem diversas intersecções de caminho. Neste cenário o desafio é encontrar uma solução que permita aos trens efetuarem uma análise das condições gerais dos circuitos e definirem uma estratégia, preferencialmente ótima, que o transporte eficientemente de um ponto do circuito a outro ponto qualquer de interesse com o deslocamento mínimo e seguro do trem.

A proposta desenvolvida foi a utilização de grafos e a geração de uma função com base em algoritmos de menor caminho, com esta solução o objetivo é que os trens redefinam sua melhor rota com base nas características do sistema no qual estão inseridos.

3.3.1 GRAFOS E ALGORITMOS DE MENOR CAMINHO

Conforme Sedgewick (2002), os grafos são oriundos de um ramo da matemática que estuda relações entre objetos de um dado conjunto. Estas estruturas são do tipo G (V,A), no qual V é o conjunto de objetos chamados de vértices e A é o conjunto de pares não ordenados de V chamado arestas.

De acordo com a funcionalidade as arestas podem ou não ter direção, e ainda podem ou não possuir arestas ligando um vértice a si próprio. Os vértices e arestas podem ainda ter um peso numérico associado. Para o caso de arestas que tenham uma direção associada teremos um grafo orientado.

Estruturas representadas por grafos podem ser encontradas em diversas aplicações e muitos dos problemas práticos de interesse podem ser formulados com um sistema composto de grafos. No caso específico deste projeto foi desenvolvido um grafo de arestas direcionais e com pesos em cada conexão.

Com base na estrutura do circuito, dimensionou-se os parâmetros do circuito de forma que as arestas representam o posicionamento dos semáforos no sistema e os pesos são descritos pelo tamanho dos trilhos, ou mais precisamente para o caso da simulação do número de pixels na imagem correspondente ao trilho. Para as arestas não comunicantes, que são os semáforos que não possuem trilhos de ligação entre si, adiciona-se um peso infinito, ou de elevado valor, para corresponder a impossibilidade de utilização da rota.

Em um sistema de grafos a análise de um mecanismo para obter o melhor caminho entre dois vértices partiu do estudo de desempenho de três modelos amplamente difundidos e conhecidos na sociedade acadêmica, sendo eles: algoritmo de *Dijkstra*, de *Bellman-Ford* e de *Floyd-Warshall*.

Assim foram efetuados testes padronizados sobre cada método com o intuito de obter o mais rápido e eficiente para um sistema com as proporções deste projeto.

3.3.1.1 ALGORITMO DE DIJKSTRA

O algoritmo de *Dijkstra* foi idealizado pelo cientista da computação holandês Edsger Dijkstra em 1956, conforme Sedgewick (2002), o propósito deste algoritmo é o problema do caminho mais curto num grafo, com a única restrição de não aceitar arestas de peso negativo.

A base para este algoritmo é a busca e mapeamento em todo o grafo pela soluções mais eficiente de menor caminho, no caso esta função é dita "estratégia gulosa", tomando a decisão que parece ótima no momento. Esta proposta é bastante conveniente uma vez que para um menor caminho entre dois vértices, todo subconjunto é um menor caminho entre dois vértices intermediários pertencentes ao caminho original, desta forma é possível construir os melhores caminhos dos vértices alcançáveis pelo vértice inicial e assim determinar todos os melhores caminhos intermediários.

O algoritmo considera um conjunto de menores caminhos, iniciado com um vértice inicial. A cada passo do algoritmo este busca nos vértices conectados por uma aresta aquele com menor distância relativa a ao vértice inicial, então repete os passos até que todos os vértices alcançáveis estejam mapeados. Escolhendo por fim o de menor custo final.

Na figura (3.3) pode ser verificado o pseudocódigo para o algoritmo de *Dijkstra*.

```
para todo v ∈V[G]
d[v]← ∞
π[v] ← -1
d[s] ← 0
Q ← V[G]
enquanto Q ≠ ø
        u ← extrair-mín(Q)
        para cada v adjacente a u
        se d[v] > d[u] + w(u, v)
                então d[v] ← d[u] + w(u, v)
                π[v] ← u
                Q ← Q ∪{v}
```

Figura 3.3. Pseudocódigo do Algoritmo de *Dijkstra*

3.3.1.2 ALGORITMO DE BELLMAN-FORD

O Algoritmo de *Bellman-Ford* foi idealizado por Richard Bellman e Lester Ford Jr. em 1958, conforme Sedgewick (2002). O propósito deste algoritmo é o problema do caminho mais curto num grafo, não possuindo a restrição de não aceitar arestas de peso negativo. O Algoritmo de *Dijkstra* tende a resolver os problemas em menor tempo, entretanto exige que todas as arestas tenham pesos positivos. Desta forma este algoritmo é normalmente usado apenas quando existem arestas de peso negativo.

O algoritmo consiste em iniciar com as estimativas dos pesos dos caminhos mais curtos com infinito e os predecessores nulo. Após isto é efetuado um procedimento similar ao *Dijkstra* de verificação das arestas de todos os vértices. E finalmente é efetuado uma verificação da existência de ciclos com peso negativo.

Na figura (3.4) pode ser verificado o pseudocódigo para o algoritmo de *Bellman-Ford*.

```
para todo v ∈V [G]
d[v] ← ∞
d[s] ← 0
para i ← 1 até |V [G]|
        relaxação ← Falso
para cada (u, v) ∈ E[G]
        se d[v] > d[u] + w(u, v)
                d[v] ← d[u] + w(u, v)
                relaxação ← Verdadeiro
se relaxação = Falso
        Fim
para cada (u, v) ∈ E[G]
        se d[v] > d[u] + w(u, v)
                resultado Falso
resultado Verdadeiro
```

Figura 3.4. Pseudocódigo do Algoritmo de *Bellman-Ford*.

3.3.1.3 ALGORITMO DE FLOYD-WARSHALL

O Algoritmo de *Floyd-Warshall* foi idealizado por Robert Floyd e Stephen Warshall em 1962, conforme Sedgewick (2002). O propósito deste algoritmo é o problema do caminho mais curto num grafo. Este método tem como objetivo uma computação dinâmica que parte de uma solução ótima previamente obtida e desenvolve a solução ótima global.

O algoritmo explora um relacionamento entre um caminho do vértice inicial ao final e os caminhos mais curtos entre os vértices intermediários do conjunto. O relacionamento depende dos caminhos intermediários participarem ou não do caminho inicialmente proposto.

Na figura (3.5) pode ser verificado o pseudocódigo para o algoritmo de *Floyd-Warshall*.

```
para dist[i,j]
 para i de 1 a n
        para j de 1 a n
                se dist[i,j] < Infinito
                        pred[i,j] := i
para k de 1 a n
        para i de 1 a n
                para j de 1 a n
                        se dist[i,j] > dist[i,k] + dist[k,j]
                                dist[i,j] = dist[i,k] + dist[k,j]
                                pred[i,j] = pred[k,j]
resultado dist
```

Figura 3.5. Pseudocódigo do Algoritmo de *Floyd-Warshall*.

3.3.1.4 COMPLEXIDADES

Para classificar problemas computacionais de acordo com sua dificuldade inerente e efetuar um relacionamento entre essas classes podemos utilizar uma comparação entre suas complexidades. Em geral, a eficiência ou complexidade de um algoritmo é uma função do tamanho do problema, do número de passos necessário para sua execução e da necessidade de memória do sistema para a execução do algoritmo, segundo Sipser (2005).

Tomando como base o número de operações necessárias para a execução do algoritmo a tabela (3.1) traz os graus de complexidade teóricos dos três algoritmos de interesse.

Tabela 3.1. Complexidade em algoritmos de menor caminho.

Algoritmo	Complexidade
Dijkstra	$O(\|E\|+\|V\|\log\|V\|)$
Bellman-Ford	$O(\|v\times a\|)$
Floyd-Warshall	$O(\|V^3\|)$

A figura (3.6) demonstra o gráfico relacional destas complexidades com a quantidade de iterações em relação a ordem de complexidade do grafo de até 45, e assim podemos esperar, ao menos de forma teórica, que o algoritmo de *Dijkstra* seja o mais eficiente para o caso deste projeto.

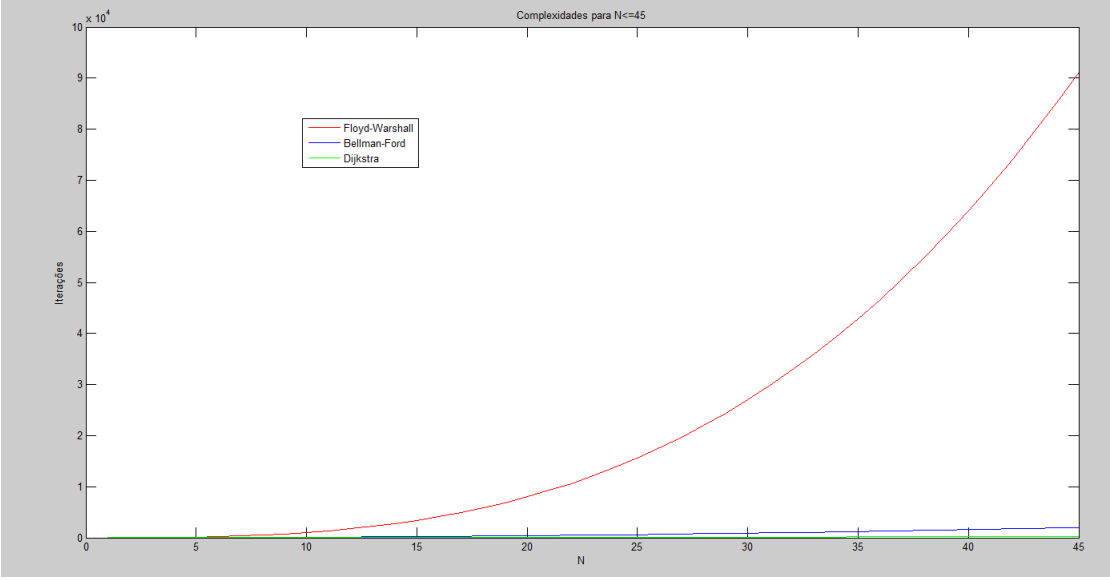


Figura 3.6. Gráfico da complexidade de algoritmos de menor caminho.

3.3.1.5 TESTES DE DESEMPENHO

Como um método prático para análise de desempenho entre os algoritmos realizou-se um procedimento padronizado. Este procedimento consistiu na execução de dez mil testes para cada algoritmo de cálculo de menor caminho com um gráfico unidirecional de 45 arestas, nestes os tempos foram computados e os resultados podem ser verificados na figura (3.7).

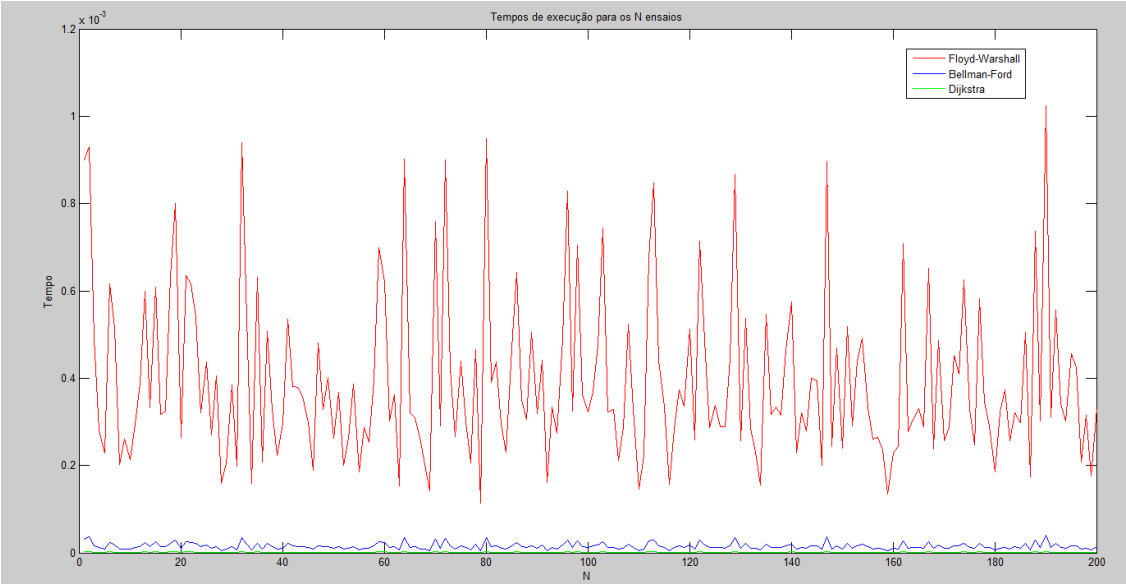


Figura 3.7. Gráfico de desempenho algoritmos de menor caminho.

Os testes foram realizados em computador de mercado, conforme especificações na tabela (3.2).

Tabela 3.2. Principais componentes computador de teste.

Componente	Especificação
Fabricante	Positivo
Modelo	POS-PIH67CH
Tipo	x64
Processador	Pentium G870 3,1 Ghz Dual Core
Memória RAM	8 GB
Sistema Operacional	Win 7
Disco Rígido	ST500DM002 500GB
Memória Cache	3MB L3

Assim concluímos, com base nos resultados da figura (3.7) e da compilação dos mesmos na tabela (3.3), que o algoritmo de *Dijkstra* é a melhor opção a ser utilizada para o desenvolvimento do sistema de menor caminho na malha viária de trens proposta para o projeto. Tal fato pode ser verificado teoricamente na seção 3.1.2.1.4.

Tabela 3.3. Resultado da análise de tempo de execução de algoritmos de menor caminho.

Algoritmo	Média tempo (s)	Desvio Padrão
Dijkstra	1,60E-06	1,19E-05
Bellman-Ford	1,87E-05	1,49E-04
Floyd-Warshall	4,87 E-04	3,62E-03

3.3.2 FERRAMENTA DE DESENVOLVIMENTO

Para a criação dos códigos em ADA foi utilizado o aplicativo GPS que, conforme AdaCore (2014) consiste na aplicação recomendada pelos desenvolvedores do ADA como ferramenta de edição. O GPS trata-se de um estúdio de programação para o GNAT. O GPS possui uma excelente interface gráfica para a construções de códigos e a exemplo das demais aplicativos baseados no GCC, em Linux, possui elevado poder de compilação de códigos.

O GNAT é uma biblioteca livre e de elevada qualidade com capacidade completa para efetuar a compilação de todas as versões do ADA. Atualmente o GNAT está integrado no GCC, e assim, participa da abrangente coleção de linguagens de programação livres para os sistemas GNU, a base do Linux.

Para uma construção efetiva e principalmente para pleno utilização das funcionalidades da linguagem ADA, em especial para as associadas a construções gráficas e de tempo real, fez-se necessário a utilização da biblioteca *GtkAda*.

Verificado em AdaCore (2014) a biblioteca *GtkAda* é utilizada pelo AdaCore como um kit de ferramentas gráficas para a implementação de suas ferramentas. Assim a biblioteca pode ser utilizada a partir de uma aplicação Ada, sem a necessidade de ter de criar ou recompilar qualquer código. Com esta abordagem, o trabalho de concepção da interface gráfica pode ser realizado de forma independente ao da programação da aplicação.

Além disso, a tecnologia *GtkAda* confia no nível mais baixo em Win32 ou X11 primitivas, de acordo com a plataforma, para desenhar os seus *widgets*, garantindo uma execução nativa de elevada eficiência. Ela possui também uma conexão direta com a plataforma que permite optar por utilizar na aplicação uma formatação que reflete a aparência da plataforma nativa.

3.3.3 O PROGRAMA

A criação do código para a simulação do projeto se deu em versões. Foram desenvolvidas em quatro etapas as funcionalidades previstas para o sistema, segue o resumo de cada uma:

- Versão 1, com as funcionalidades básicas da linguagem ADA, para testes de utilização de semáforos e interface gráfica;
- Versão 2, apresenta um princípio do circuito, iniciado alguns semáforos com os quatro trens e desenvolvimento da função de menor caminho;
- Versão 3, desenvolvido o circuito completo proposto para o projeto ainda com sentido único de deslocamento para os trens;
- Versão 4, adicionado o deslocamento em ambos os sentidos dos trens e efetuado ajustes para evitar a ocorrência de *deadlocks*.

3.3.3.1 VERSÃO 1

A primeira versão desenvolvida trata se de um circuito simples com o objetivo de testar as funcionalidades da Linguagem Ada, suas capacidades e ganhos por tratar-se de uma linguagem orientada a sistemas em tempo real.

Nesta etapa, foi gerado um circuito simples, conforme figura (3.8), que consiste de dois trens em dois circuitos deslocando-se em sentidos opostos, com o trecho intermediário compartilhado por ambos.

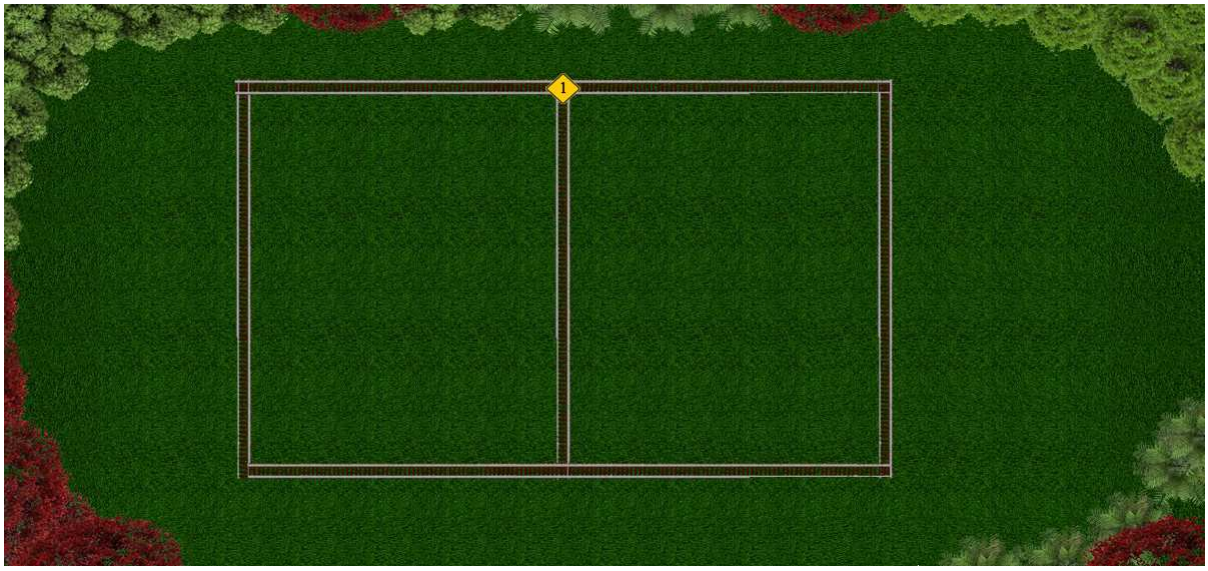


Figura 3.8. Circuito simulado na versão 1 do código.

Esta implementação possibilitou verificar a estrutura de classes e *treads* para a criação dos trens e do circuito. Foi possível efetuar testes primários com o uso do semáforo como solução para o escalonamento.

Na linguagem ADA, o semáforo é um tipo abstrato de dado, que possui como função o controle de acesso a recursos compartilhados. Possui como parâmetro para a função os comandos LOCK e UNLOCK que bloqueiam a utilização do recurso pelo demais processos, os demais pedidos de bloqueios, a partir do primeiro, iniciam uma fila de espera para a utilização de acordo com a ordem de prioridade.

A estrutura do semáforo opera naturalmente na linguagem, com a concepção dos semáforos cada trem ao utilizar o trecho compartilhado efetua o bloqueio do mesmo, liberando após a utilização. O semáforo gera de forma automática uma fila de pedidos de uso do recurso, apesar de não ser constatado a geração de filas nesta etapa por termos apenas dois trens. Uma vez que o primeiro trem libera o semáforo, trecho, o segundo trem pode utilizá-lo normalmente.

Observa-se ainda na linguagem ADA que o percurso bloqueado pode ser tratado como uma variável compartilhada, aonde o processo que efetuou o bloqueio, ou seja o que está acessando a variável, não pode mais ser interrompido, nem mesmo por um processo de maior prioridade. Entretanto devemos destacar que a fila formada por múltiplos pedidos de acesso a um semáforo leva em conta as prioridades e assim o trem, representado pelo processo, com maior prioridade será o próximo a utilizar o semáforo.

Nas figuras (3.9) e (3.10) são apresentados dois pseudocódigos simples que demonstram o funcionamento do semáforo e do trem nesta fase do projeto.

```
inicializar Semáforo P
repita continuamente
        se LOCK então
                se P > 0 então
                        bloqueado
                        P ← P + 1
                        adiciona processo na fila
                senão
                        liberado
                        P ← P + 1
        se UNLOCK então
                P ← P - 1
                        se processo na fila então
                                libera próximo processo
```

Figura 3.9. Pseudocódigo para o semáforo na versão 1 do código.

```
inicializar Trem
repita continuamente
        anda
        se semáforo então
                LOCK
                repita
                        se bloqueado então
                                para
                até que liberado
                repita
                        anda
                até que fim semáforo
                UNLOCK
```

Figura 3.10. Pseudocódigo para o trem na versão 1 do código.

Em termos práticos, seria possível adicionar quantos trens fossem de interesse no circuito, desde que se tenha o cuidado de adicionar mais semáforos no circuito com o intuito de evitar colisões. Este circuito simplificado não possui o risco de gerar *deadlocks* em sua implementação.

3.3.3.2 VERSÃO 2

A segunda versão desenvolvida trata se da ampliação do número de trens para quatro, o total final, e a utilização parcial do circuito proposto para o projeto, com a área de garagens e o primeiro circuito, conforme figura (3.11). Nesta versão os trens se deslocam em sentido único no trajeto, todos com a mesma orientação.

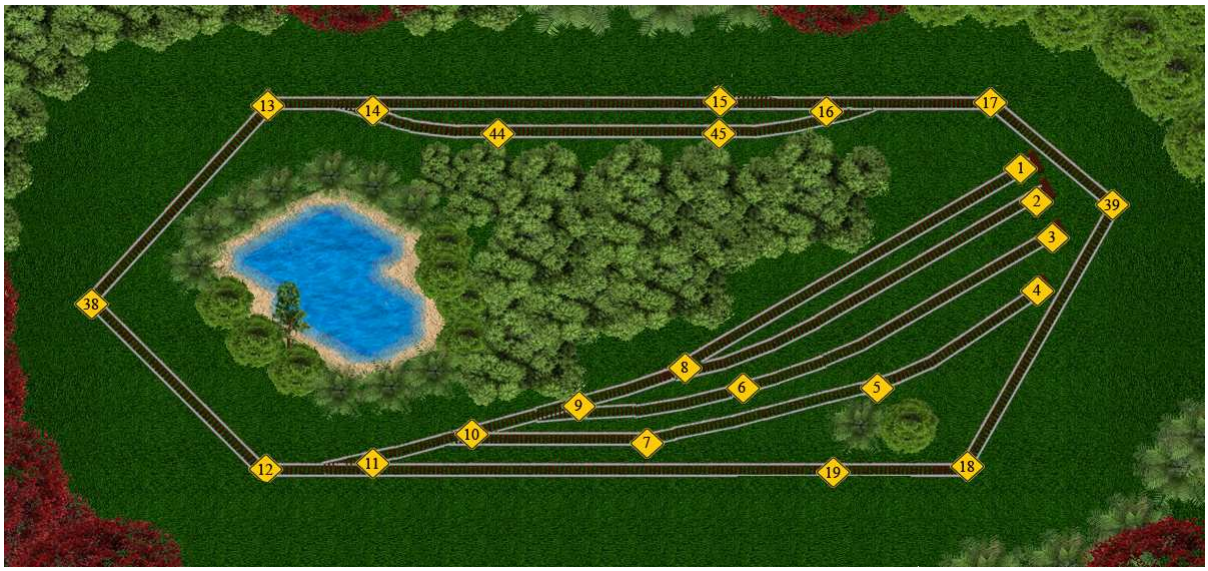


Figura 3.11. Circuito simulado na versão 2 do código.

Nesta etapa, inicia-se a utilização de filas nos semáforos, propriamente dito, uma vez que tratamos agora de vários trens concorrendo pelos mesmos trechos e desvios. A cada desvio temos a implementação de um semáforo com a capacidade de bloquear o trecho subsequente.

As filas, decorrentes da linguagem ADA ocorrem naturalmente no sistema, ou seja, cada objeto semáforo instanciado tem plena capacidade para a geração e controle de sua fila de acordo os pedidos e prioridades dos trens.

Neste ponto, tem-se o desenvolvimento da função menorcaminho(), que consiste em uma função que desenvolve o menor caminho entre a posição atual e a posição futura, de forma linear entre cada semáforo, desvio. No aspecto gráfico, considera-se cada trecho de trilho como uma linha reta entre dois pontos, e o trem deslocando-se linearmente entre ambos.

Uma vez que para o desenvolvimento da função menorcaminho() foi utilizado o algoritmo de *Dijkstra*, tem-se a montagem de um grafo que consiste de um *array* 23x23 representando os 23 semáforos. Os pesos do grafo são as distancias lineares entre os semáforos, número inteiro, que representam quantitativamente a distância de deslocamento entre cada semáforo. Assim, por exemplo, pode-se considerar como infinito a distância entre dois semáforos que não se comunicam, ou seja, que não possuem um trecho os ligando.

Na versão 2, o sistema é mantido unidirecional, ou seja, todos os trens deslocam-se no mesmo sentido, o horário, desta forma não existe a ocorrência de *deadlocks* que incapacitem a operação pois os semáforos executam o controle das filas adequadamente.

Os trens são assim alocados no ponto cartesiano e ficam travados em sua posição atual até a demanda por uma posição futura. Ao iniciar o processo de deslocamento, o mesmo se desloca em uma função reta até o próximo ponto de acordo com um atraso entre cada nova posição que tem como funcionalidade representar a velocidade de deslocamento.

O processo de chaveamento de semáforos se dá de forma a garantir segurança de colisões nos semáforos. Um trem ao entrar em um semáforo efetua o bloqueio deste e do próximo semáforo, garantindo que o trecho entre semáforos esteja totalmente livre de outros trens. Ao alcançar o semáforo seguinte o trem efetuar a liberação do semáforo anterior, com o objetivo de liberar o recurso, o trecho anterior, para que outros trens o utilizem.

O primeiro teste efetivo foi realizado sobre esta implementação, foi efetuado uma simulação gráfica de cerca de 10 minutos com rotas aleatórias para os trens e não foram identificados erros ou falhas no sistema de controle.

Nesta simulação, todos os trens partiram da área de garagem, semáforos 1, 2, 3 e 4, e iniciaram concorrência nos semáforos 8, 9, 10 e 11, e então entram no circuito principal permanecendo no mesmo até o fim da simulação. Até esta versão não temos implementado um retorno para a área de garagem pois o movimento de cada trem está mantido em sentido único, sem possibilidade de retorno.

Não houve alteração em relação ao pseudocódigo do semáforo. Entretanto, uma nova implementação foi desenvolvida para o pseudocódigo do trem, que pode ser observada na figura (3.12).

```
inicializar Trem P
repita continuamente
        ponto final = novo trajeto()
        repita
                se P != ponto final então
                        P ← menorcaminho()
                        anda
                se semáforo então
                        LOCK
                        repita
                                se bloqueado então
                                        para
                        até que liberado
                        repita
                                anda
                        até que fim semáforo
                        UNLOCK
        até que ponto final
```

Figura 3.12. Pseudocódigo para o trem na versão 2 do código.

3.3.3.3 VERSÃO 3

A terceira versão desenvolvida trata da ampliação do circuito proposto no projeto, com a área de garagens e os três circuitos, conforme figura (3.13). Nesta versão, os trens se deslocam em sentido único no trajeto, todos com a mesma orientação.

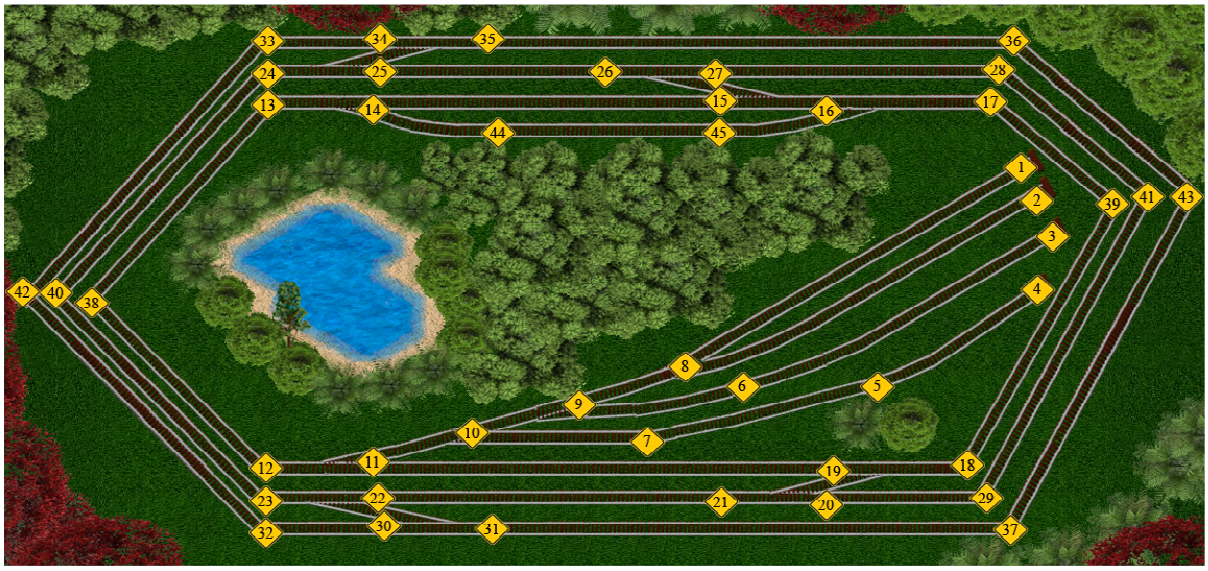


Figura 3.13. Circuito simulado na versão 3 do código.

São aqui desenvolvidos todos os semáforos do sistema e todos os trajetos para os trens. Permanece aqui a utilização do algoritmo de menor caminho e sistema de bloqueio vistos na versão 2.

Nesta versão a quantidade semáforos é aumentada para 45, condizendo assim com o circuito completo a ser desenvolvido. Para este novo espaçamento foi efetuado a ampliação do grafo, que agora contém um *array* de 45x45, entretanto não ocasionou maiores dificuldades de processamento da função de busca por menor caminho e assim mostrou-se eficaz para esta atividade.

Seguindo ainda o estipulado na versão 2 o sistema é mantido unidirecional, ou seja, todos os trens deslocam-se no mesmo sentido, o horário, desta forma não existe a ocorrência de *deadlocks* que incapacitem a operação pois os semáforos executam o controle das filas adequadamente.

Foi realizado novamente um teste sobre esta implementação, foi efetuado uma simulação gráfica de cerca de 10 minutos com rotas aleatórias para os trens e não foram identificados erros ou falhas no sistema de controle.

Aqui tem-se o primeiro teste completo do circuito onde os trens saem da área de garagem e se deslocam pelo circuito povoando cada trajeto. Aqui pode ser observada uma grande quantidade de bloqueio e filas em semáforos, geradas principalmente pela presença do trem de limpeza que possui trajetos aleatórios no sistema.

Os trens após saírem da área da garagem e atingirem os circuitos concorrem livremente pelos desvios e deslocam-se aleatoriamente. Até esta versão não temos implementado um retorno para a área de garagem pois o movimento de cada trem está mantido em sentido único, sem possibilidade de retorno.

3.3.3.4 VERSÃO 4

A quarta versão desenvolvida trata da ampliação do sentido de deslocamento dos trens, nesta versão os trens se deslocam em ambos os sentidos nos trechos, ou seja, não estão mais restritos ao movimento em apenas um sentido. Com esta implementação os trens podem então andar para frente e para trás, e assim foi verificado a ocorrência de *deadlocks* no sistema.

Não temos mais adição de implementação de circuitos e assim a figura (3.13) mantém a representatividade gráfica da estrutura final desenvolvida.

O grafo foi novamente modificado, Anexo I, com a finalidade de permitir o movimento em ambos os sentidos, ou seja, foram implementados os arcos bidirecionais. Temos aqui a criação de pesos entre o sentido inverso do semáforo, ou seja, o peso para retornar não é mais infinito e assim o deslocamento nesta via está liberado.

Esta nova implementação gerou um problema até então não verificado no sistema, a ocorrência de *deadlocks*, isto porque agora temos trens movimentando-se em ambos os sentidos o que permite o que gera o travamento de um semáforo por dois trens em sentidos opostos, criando assim uma impossibilidade de deslocamento.

Ao executarmos o teste na simulação gráficas verificamos o *deadlock* gerado no sistema rapidamente assim que os trens iniciam a operar nos circuitos.

Como primeiro passo para a resolução dos *deadlocks* foi implementado uma nova sistemática que consiste no bloqueio de três semáforos a frente, de acordo com trajeto gerado pela função menorcaminho() para obtenção do menor caminho. O bloqueio dos três próximos trechos não garantiu a estabilidade do sistema no quesito de concorrência entre trens de sentidos opostos.

Uma segunda abordagem para a resolução dos *deadlocks* foi implementar o circuito do meio com sentido fixo, ou seja, neste circuito todos os trens movimentam-se apenas no sentido horário. Este procedimento tinha como objetivo minimizar ao máximo a ocorrência de *deadlocks* quando os trens entrassem no circuito do meio, entretanto esta operação também não garantiu a estabilidade do sistema.

A terceira abordagem e definitiva foi a definição de rota específica para cada trem antes da saída da garagem, assim cada trem efetua o cálculo de melhor caminho para sua rota através da função menorcaminho() e então a executa.

Esta abordagem, em somatória com a abordagem de bloqueio de três semáforos, resultou em uma solução eficiente para o sistema. Foi realizado novamente um teste sobre esta implementação, com

uma simulação gráfica de cerca de 10 minutos, com rotas pré definidas na inicialização para os trens e não foram identificados erros ou falhas no sistema de controle.

O trem de manutenção parte do semáforo 4 e foi pré-definido para passar por todos os trilhos, o mesmo possui prioridade menor aos demais trens quando em fila nos semáforos. Este também possui um trajeto fixo e assim não executa o processamento de menos caminho e não parte imediatamente junto com os demais trens para evitar concorrências iniciais.

Estão apresentamos nas figuras (3.14) e (3.15) os códigos simplificados em ADA que demonstram o funcionamento do semáforo e do trem nesta fase final do projeto.

```
task type t_semaphore is
    entry LOCK;
    entry UNLOCK;
end t_semaphore;
task body t_semaphore is
    NB:natural; --numero de acessos
    begin
        loop
            accept LOCK; --cria fila a medida que recebe lock
            accept UNLOCK;
        end loop;
end t_semaphore;
```

Figura 3.14. Código ADA para o semáforo na versão 4.

```
task type t_trem is
    entry INIT(NB_ACCES:natural);
    pragma priority(System.Default_Priority+1); --define a prioridade
end t_trem;
task body t_trem is
    NB:natural; --ID
    prox_a:integer; --ID
    begin
        accept INIT(NB_ACCES:natural)do
            NB:=NB_ACCES;
            Put_Line("Inicio Trem A"& Integer'Image(NB));
        end INIT;
        while S_A/=T_A loop --executa função menor caminho
        prox_a:=menor(matriz, S_A, T_A, n); --chamada da função menor caminho
        Put_Line ("Menor Caminho: " & integer'Image(prox_a));
        Put_line("Trem A" & Integer'Image(NB) & " quer travar " & Integer'Image
(prox_a));
        trava(prox_a); --trava semáforo
        Put_line("Trem A" & Integer'Image(NB) & " travou " & Integer'Image
(prox_a));
        destrava(S_A); --destrava semáforo
         Put_line("Trem A" & Integer'Image(NB) & " destravou " & Integer'Image
(S_A));
        S_A:=prox_a;
    end loop;
end t_trem;
```

Figura 3.15. Código ADA para o trem na versão 4.

4 IMPLEMENTAÇÃO

4.1 PROTÓTIPO

Um item possivelmente presente no desenvolvimento de projetos, principalmente para a realização de testes ou de planejamentos consiste na construção de um protótipo, conforme Lobach (2001). O protótipo é um modelo físico, gráfico ou computacional para a representação conceitual de um produto de engenharia.

Segundo Baxter (2000), não existe um consenso sobre a constituição de um protótipo, e principalmente quanto ao uso em sinônimo da palavra modelo. Entretanto este classifica os protótipos em cinco categorias:

- Protótipo de Prova de Conceito, utilizado para efetuar testes de algum aspecto específico de um sistema sem necessariamente uma representação visual ou dos materiais empregados no produto. São utilizados principalmente para provar uma abordagem do projeto, como por exemplo análises mecânicas, sensores, e arquitetura da construção;
- Protótipo de Estudo de Forma, possui função de permitir aos desenvolvedores uma relação de exploração com o produto, permitindo uma verificação sensorial sem uma implementação final do mesmo. São bastante utilizado para tomadas de decisão quanto a ergonomia, dimensões e volume;
- Protótipo de Experimentação do Usuário, permite realizar uma avaliação de desempenho do produto através de uma avaliação junto aos usuários, consistindo da manipulação e utilização do protótipo. Assim é possível efetuar uma avaliação inicial de reação e desempenho diretamente ao público de destino;
- Protótipo Visual, possui função de fornecer uma visualização do produto, referente a sua aparência, cor e textura. São bastante utilizados principalmente para avaliações de publicidade e impacto de mercado;
- Protótipo Funcional, utilizado na maior medida prática como uma tentativa de simular o desenho final, a estrutura, os materiais e funcionalidade do produto. O protótipo funcional pode ser uma redução em tamanho ou em funcionalidade para a execução de uma análise final do projeto.

Ainda conforme Baxter (2000), segundo a própria definição dos protótipos estes representam um compromisso do projeto final de produção. Assim devido a possíveis diferenças de materiais, processos e fidelidade de criação é provável que um protótipo pode apresentar falhas para um desempenho funcional em níveis aceitáveis, uma vez não tratar-se do produto final. Uma outra abordagem é a utilização de protótipos iniciais, que implementam partes do projeto completo, permitindo uma análise de possíveis problemas e a minimização de riscos antes de iniciar a construção do projeto completo.

A implementação proposta neste projeto consiste da criação de um protótipo do tipo funcional que permita efetuar uma avaliação das principais atribuições do projeto. Com a elaboração de um protótipo em forma de maquete para a ratificação das funcionalidades do programa desenvolvido, garantindo assim a aplicação prática dos estudos e elementos desenvolvidos no capítulo 3.

4.2 MAQUETE EM FERRORAMA

A maquete utilizada para a implementação dos circuitos trata-se de componentes comerciais da Empresa Frateschi Ltda., a qual está no mercado de ferromodelismo a cerca de 40 anos. A montagem do circuito seguiu as técnicas indicadas no Manual de Construção de Maquetes Ferroviárias, Frateschi (2012), que indica as melhores práticas para o desenvolvimento desta atividade.

4.2.1 SISTEMA ELÉTRICO

O sistema elétrico da maquete para o deslocamento dos trens é baseado na energização dos trilhos, ou seja, para o movimento dos trens faz-se necessário alimentar os trilhos com 16v e potência suficiente para o sistema de tração.

As linhas são energizadas por trechos onde a alguns intervalos de trilhos existe uma chave de energia para o sistema. Os trechos são unidos por talas de junção que efetuam a continuidade do sistema elétrico para cada trecho.

4.2.2 SISTEMA DE CONTROLE

O ferromodelismo da Frateschi possui duas estruturas básicas de controle, o controle do deslocamento do trem e de posição dos desvios.

Para o controle de deslocamento do trem é necessário a energização dos trilhos, desta forma, no padrão do fabricante, a energização do trilho garante o imediato acionamento dos trens e seu movimento, até que o trilho seja desligado. Para a energização dos trilhos é utilizado um interruptor de duas posições, ligado/desligado.

No caso do acionamento da posição dos desvios é executado um pulso de 16v sobre um solenoide instalado no desvio que chaveia a posição do mesmo. Este acionamento é realizado através de um interruptor elétrico de três posição, ligado/lado1/lado2.

4.3 CIRCUITOS DE INTERFACE

Para a execução do controle computacional dos trens e desvios foi necessário o desenvolvimento de uma interface de ligação eletrônica entre o computador e os dispositivos de controle da maquete.

A opção empregada nesta etapa do projeto foi a utilização da porta paralela de um computador comercial que aciona um circuito eletrônico de controle de baixa potência.

Especificamente para os desvios fez-se necessário a elaboração de um circuito de potência, tendo em vista a necessidade de elevada corrente nestes componentes.

4.3.1 PORTA PARALELA

Conforme Axelson (1996) “A porta paralela é uma interface de comunicação entre um computador e um periférico.”. Esta porta foi desenvolvida pela IBM quando da criação de seu primeiro computador pessoal, a proposta seria efetuar a conexão entre essa porta a uma impressora, entretanto, é verificado uma grande diversidade de periféricos que efetuam utilização desta conexão para o envio e recebimento de dados no computador. Como exemplos podemos citar, câmeras de vídeo, scanners, unidades de disco.

Para a execução de comunicação em paralelo, grupos de bits são conduzidos simultaneamente através de diversas linhas transmissoras dos sinais, segundo Gadre (2000). Nesta estratégia, como diversos bits são transmitidos simultaneamente, a taxa de transferência de dados é elevada, alto throughput. No caso da porta desenvolvida pela IBM temos a transmissão de oito bits, um byte, a cada ciclo de transmissão da porta.

Para transmissões unidirecionais a porta paralela SPP (*Standard Parallel Port*) pode obter taxas de transmissão de dados na ordem de 150KB/s. Comunica-se com a CPU através de um BUS de dados de 8 bits.

Na transmissão bidirecional é utilizado a porta avançada EPP (*Enhanced Parallel Port*) obtendo taxa de transferência de até 2 MB/s. Comunica-se com a CPU através de um BUS de dados de 32 bits.

A porta avançada ECP (*Enhanced Capabilities Port*) possui as mesmas especificações que a EPP, entretanto utiliza DMA (acesso direto à memória), dispensando a necessidade do uso do processador para a transferência de dados. Possui ainda um buffer FIFO de 16 bytes.

Os sistemas operacionais de mercado tendem a nomear as portas paralelas, chamando-as de LPT1, LPT2, LPT3, e assim consecutivamente. Entretanto a porta física padrão é normalmente a LPT1, e seus endereços são: 378h, para enviar um byte de dados pela Porta, 378+1h, para receber um valor através da Porta e, 378+2h, para enviar dados. Nos casos de disponibilidade da LPT2, e seus endereços são: 278h, 278+1h e 278+2h, com as mesmas funções dos endereços da porta LPT1, respectivamente.

O conector mais utilizado, e normalmente instalado fisicamente nas placas-mãe dos microcomputadores é o DB25. Através deste o cabo paralelo se conecta ao computador para poder enviar e receber dados. No DB25, por especificação, o pino está em nível lógico 0 quando a tensão elétrica no mesmo está entre 0 e 0,4v e em nível lógico 1 quando a tensão elétrica no mesmo está acima de 3,1 e até 5v. É possível verificar na figura (4.1) o conector DB25 e seu esquema de ligação.

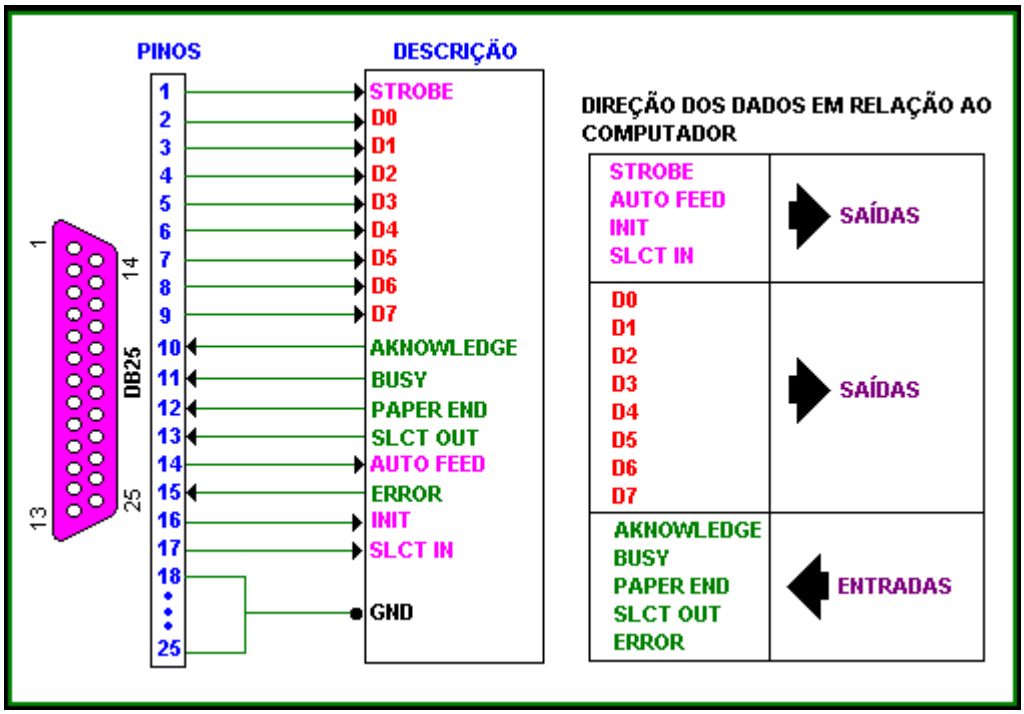


Figura 4.1. Esquema da Porta Paralela do conector DB25.

Um segundo conector bastante difundido trata-se do *Centronics* 36 pinos (CN36), trata-se de um conector bastante difundido devido a sua utilização como parte do cabo da impressora, sendo através deste cabo que a impressora é conectada ao computador. Devido à popularidade deste conector o

mesmo é muito utilizado para aplicações e projetos com outros periféricos ou de uso individual e particular. Na figura (4.2) podemos verificar o esquema simplificado do conector CN36.

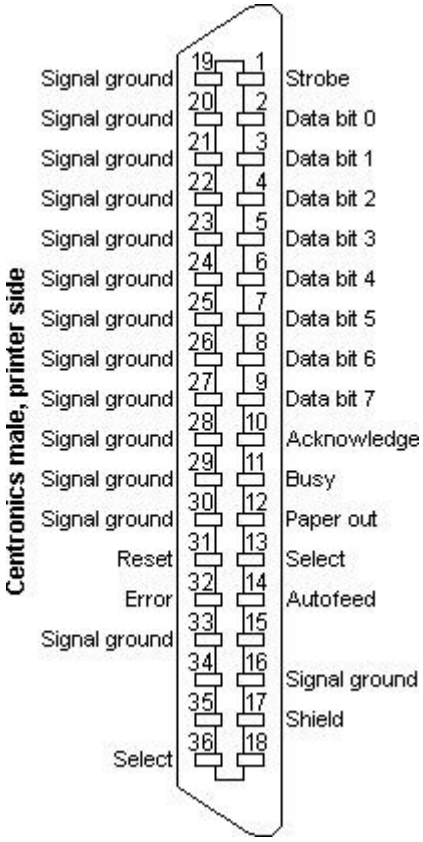


Figura 4.2. Esquema da Porta Paralela do conector CN36.

Para o projeto foi utilizado um cabo com conector CN36, devido a facilidade em encontrar cabos do gênero. Desta forma efetuamos uma subdivisão entre as pinagens do conector dividindo entre 8 bits de dados entre os pinos 2 a 9, representados por Data Bit 0-7, e a sequência de 3 bits de seleção, os pinos 1, 14 e 36, representados respectivamente por *Strobe*, *Autofeed* e *Select*, conforme verificado na figura (4.2).

Desta forma, foi possível efetuar uma subdivisão lógica nas transferências de dados para interface, na qual temos um campo para dados, onde existe a transferência da informação, de 8 bits, e o campo de controle do circuito digital, com 3 bits.

4.3.2 CIRCUITO DIGITAL

De acordo com a especificação da maquete, faz-se necessário o acionamento de quatro trens e quatorze desvios através da porta paralela do computador. Uma vez que em projeto cada trem possui três possibilidades de acionamento, frente, trás e parado é necessário a utilização de dois sinais por trem, para os desvios temos também três possibilidades de acionamento, direita, esquerda e indefinido, e desta forma também é necessário a utilização de dois sinais de controle.

O estado indefinido para o desvio refere-se ao estado sem acionamento, no qual é mantido o estado anterior, pois sem a atuação de um dos estados a chave do desvio permanece em posição fixa. O componente de desvio da Frateschi por tratar-se de um solenoide com elevado consumo de potência é recomendado a ser ativado apenas através de um pulso, o qual é suficiente para ativar o estado desejado.

Para o projeto foram desenvolvidos dois protótipos eletrônicos, o protótipo 1 tratou das configurações lógicas do sistema porém mostrou-se ineficiente no quesito de potência, e assim foi necessária sua reconfiguração para o protótipo 2, que apresenta configurações lógicas similares ao protótipo 1 porem com a implementação da interface de potência necessária.

4.3.2.1 PROTÓTIPO 1

O Protótipo 1 foi desenvolvido principalmente para a obtenção de conhecimento quanto a utilização da porta paralela como fonte de comunicação entre um periférico e o computador.

A proposta para este circuito foi obter os dados da porta paralela, na subdivisão de 8 bits para dados e 3 bits para controle, além de manter de forma persiste na saída os valores, de forma que os mesmos não se alterem ao cessar do envio de sinal diretamente da porta paralela.

Visto que temos uma quantidade limitada de dados de saída da porta e a necessidade de controlar 8 bits para os trens e 28 bits para os desvios fez-se necessário o carregamento por etapas dos dados para as saídas, e assim forma definidas cinco passos de carregamento dos estados dos elementos da maquete.

Foram utilizados cinco circuitos integrados 74LS374, que tratam-se de *flip-flops* tipo D que travam o circuito com a informação de entrada de acordo com um sinal de controle. Cada um destes CI's tem a capacidade de trabalhar com 8 bits e assim utilizamos o primeiro para os trens e os demais para os desvios. Todos os CI's são alimentados em suas entradas com um barramento com os 8 bits

provenientes do grupamento de dados da porta paralela e assim a transferência da informação e sua fixação na saída do circuito ocorre pela ativação do correto circuito de interesse.

A seleção do circuito é efetuada através da utilização do circuito integrado 74LS138, que trata-se de um Demultiplexador 3x8 que obtém os 3 bits de seleção e controle oriundos da porta paralela para efetuar a ativação do correto circuito 74LS374. Para este CI foi gerado a tabela da verdade (4.1) de acionamento.

Tabela 4.1. Tabela de acionamento Protótipo 1.

Bits de Seleção (74LS138)	Circuito Acionado (74LS374)	Resultado nos Estados
0 0 0	X	Não efetua atualização de estados
0 0 1	1	Atualização do estado dos 4 trens
0 1 0	2	Atualização do estado de 4 desvios
0 1 1	3	Atualização do estado de 4 desvios
1 0 0	4	Atualização do estado de 4 desvios
1 0 1	5	Atualização do estado de 2 desvios
1 1 1	X	Não efetua atualização de estados

Podemos observar na figura (4.3) a estrutura construtiva deste circuito. As saídas referenciadas por s1, s2, s3, s4, s5, s6, s7 e s8, em cada componente 74LS374 representam os acionamentos dos estados nos conjuntos de trens e desvios.

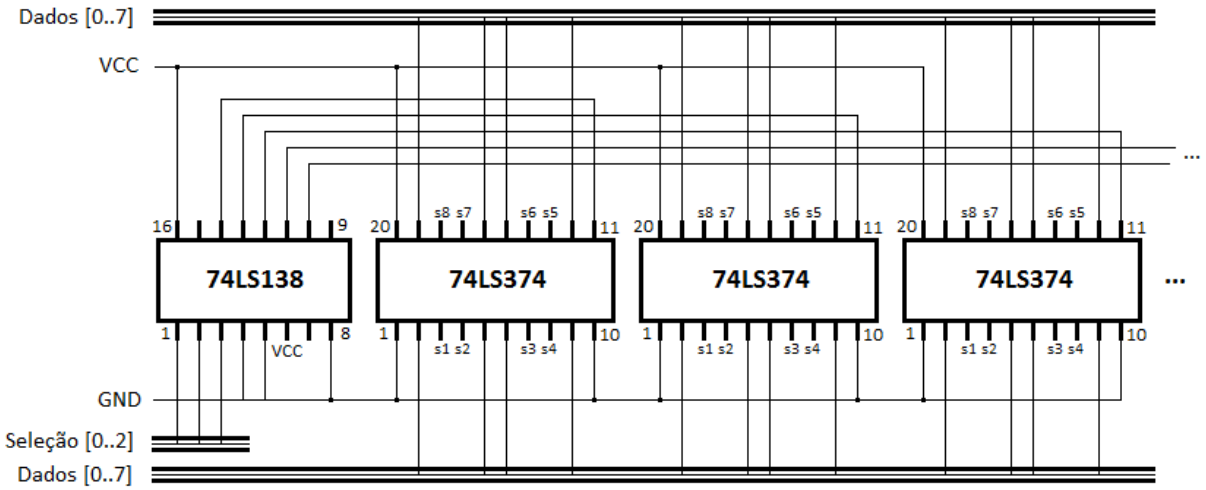


Figura 4.3. Circuito Parcial Eletrônico Protótipo 1.

Ainda que logicamente correto e eficiente, o Protótipo 1 demonstrou algumas incompatibilidades de potência com os componentes da maquete, uma vez que o mesmo fornecia as saídas apenas como um sinal de controle lógico, o que se mostrou ineficiente para o acionamento principalmente para os desvios.

4.3.2.2 PROTÓTIPO 2

O Protótipo 2 tem como função principal resolver a questão do acionamento com potência dos desvios, além de efetuar a manipulação lógica dos sinais de controle da porta paralela.

A proposta neste circuito permaneceu a de obter os dados da porta paralela, na subdivisão de 8 bits para dados e 3 bits para controle. E efetuar o acionamento final dos quatro trens e dos quatorze desvios propostos para o projeto.

Através de uma análise das especificações dos componentes foi concluído que a melhor operação para os desvios seria através de um pulso de corrente de cerca de um décimo de segundo, o que resultaria no correto posicionamento da chave sem gastos excessivos de potência. Outra verificação efetuada trata-se do ativamento do sistema de rádio dos trens através de um sinal lógico em zero para ativo e de alta impedância para desativado.

Para o acionamento dos trens foi utilizado o circuito integrado 74LS374, o qual possui os 8 bits de saída necessários para o acionamento de quatro trens e a manutenção das saídas mesmo com a retirada do sinal de entrada. Para gerar o sinal lógico 0, baixa impedância, para o sistema de rádio dos trens foi utilizado quatro circuitos integrados CD4051, que trata-se de uma chave lógica com função de multiplexador/demultiplexador e que permite efetuar conversões lógicas através de sua porta comum.

Cada trem foi instalado em um CD4051 para garantir a trava lógica na qual fica impossível acionar um trem em ambas as direções simultaneamente, e possui como saída o sinal lógico 0. A ativação do 74LS374 é efetuada diretamente através de um dos bits de seleção, conforme tabela (4.2).

Tabela 4.2. Tabela de acionamento Protótipo 2.

Bits de Seleção [0..2]	Bits de Dados [0..7]	Circuito Acionado	Resultado nos Estados
0 1 1	X	X	Não efetua atualização de estados
1 1 1	[0..7]	Trem	Atualização do estado dos 4 trens
0 0 1	[0..3]	Desvio 1	Atualização do estado de 4 desvios
0 0 1	[4..7]	Desvio 2	Atualização do estado de 4 desvios
0 1 0	[0..3]	Desvio 3	Atualização do estado de 4 desvios
0 1 0	[4..7]	Desvio 4	Atualização do estado de 2 desvios

Os desvios são ativados também com a utilização de quatro CI's CD4051 operando com saída lógica de 5v para o sistema de potência dos desvios. A construção permite a ativação de dois desvios por vez, com um dos bits de seleção ativando dois dos CD4051 e o outro os restantes, neste modelo os dados são divididos entre os componentes e assim durante a ativação de uma dupla quatro bits são fornecidos a cada a cada componente CD4051.

Com a saída de 5V sinalizando a ativação do desvio construiu-se um sistema de potência para cada um dos desvios, com base no transistor TIP110, que permite correntes de até 4 amperes e um alto ganho, na ordem de 1000, pois possui em sua constituição uma configuração do tipo *Darlington* de alto ganho.

Podemos observar na figura (4.4) e (4.5) a estrutura construtiva deste circuito, e na figura (4.6) a estrutura de potência para o acionamento dos desvios.

As saídas referenciadas na figura (4.4) por s1 e s2 em cada componente CD4051B representam os acionamentos dos estados dos conjuntos de trens através do rádio transmissor.

As saídas referenciadas na figura (4.5) por s1, s2, s3, s4, s5, s6, s7 e s8, em cada componente CD4051B representam os acionamentos dos estados nos conjuntos de desvios. Estes compões aos pares os comandos 1 e 2 representados na figura (4.6) de acionamento do circuito de potência dos desvios.

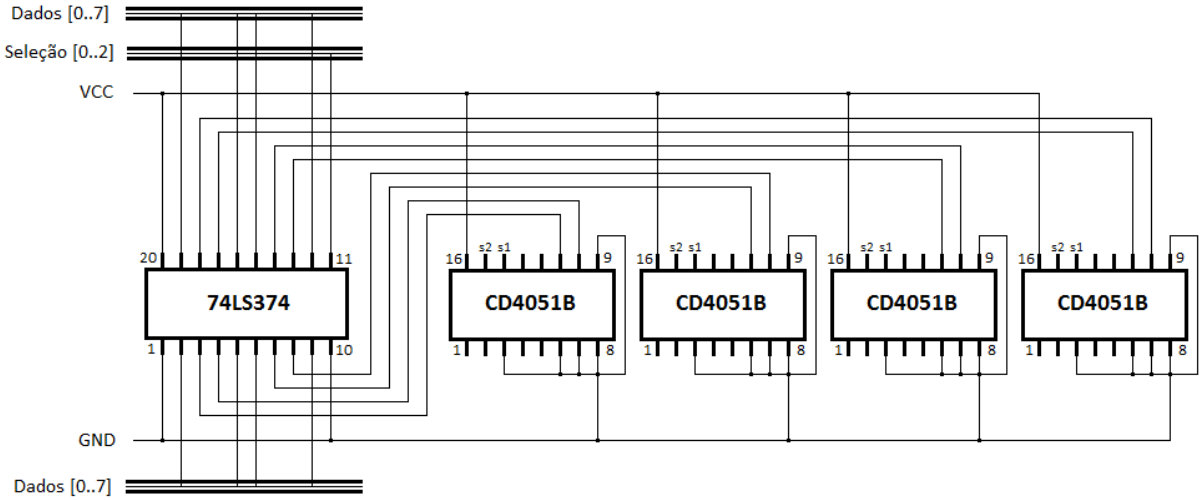


Figura 4.4. Circuito Eletrônico dos Trens, protótipo 2.

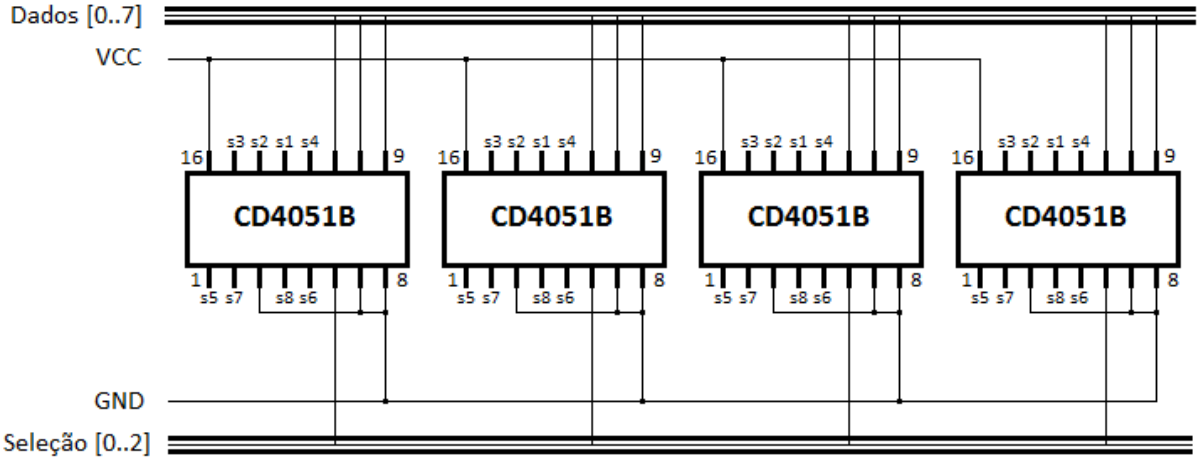


Figura 4.5. Circuito Eletrônico dos Desvios, protótipo 2.

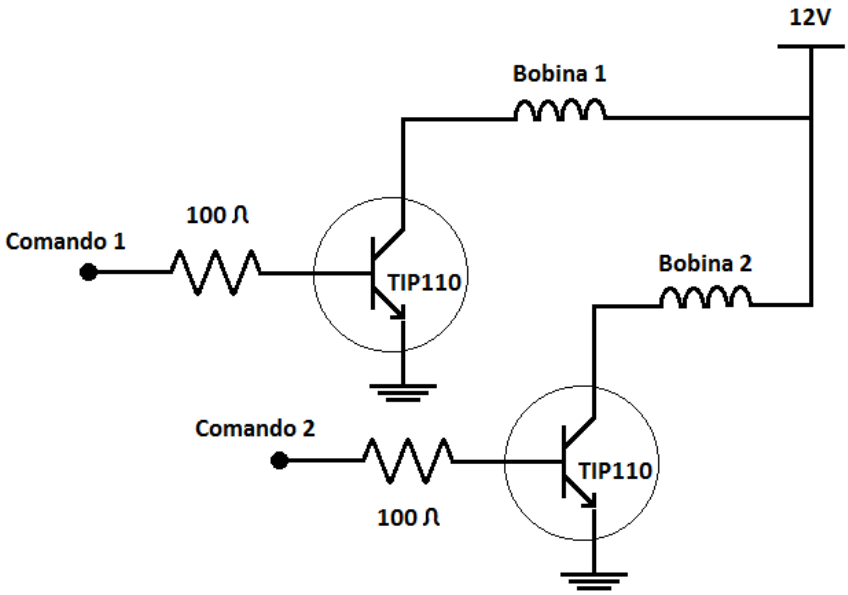


Figura 4.6. Circuito Eletrônico de Potência para os Desvios, protótipo 2.

4.4 CÓDIGO INTERMEDIÁRIO

Devido a necessidade de operação da Porta Paralela para o controle dos componentes da maquete foi criado um código específico para os acionamentos e interrupções em linguagem C. Tal construção em C e não em ADA é realizada uma vez que tal linguagem não comporta o acionamento direto ao registro de controle da porta.

O código em C gerado foi então adicionado ao código ADA completo no formato composto de biblioteca.

4.4.1 ACESSO A PORTA PARALELA

Para a utilização dos registradores da porta paralela, 378h e 37Ah, faz-se necessário a inclusão da biblioteca específica de acesso as entradas e saídas de hardware, de acordo com o sistema operacional.

Assim são adicionados ao código as função ioperm() que marcam o nível de permissão administrador aos registros. Para ativá-los é utilizado a função outb() que envia um byte de dado ao registrador da porta paralela, que é então transmitida pelo sistema operacional ao hardware de saída.

A figura (4.7) apresenta um código simplificado de acesso a porta paralela onde todos os bits de dados e de seleção são ativados.

```
#include <stdio.h>
#include <stdbool.h>
#include <sys/io.h>
int main(void)
{
        ioperm(888,3,1); //atribui permissão de acesso a memória
        ioperm(890,3,1); //atribui permissão de acesso a memória
        outb(255,888); //ativas todos os bits de dados
        outb(4,890); //ativas todos os bits de seleção
        return 0;
}
```

Figura 4.7. Código simplificado de acionamento da porta paralela.

4.4.1 CÓDIGO C

O código C desenvolvido recebe um vetor de booleanos que indicam as condições de acionamento da maquete. São assim utilizados 8 bits para os quatro trens, com dois estados de acionamento em cada um, e 28 bits para o acionamento dos quatorze desvios, utilizando também dois estados para acionamento.

Desta forma a função desenvolvida recebe o vetor booleano e efetua os acionamentos, primeiro ativando os estados dos trens e em seguida efetuando o acionamento dos desvios a cada dois, devido a necessidade de gerar apenas um pulso nos desvios.

Por questões de segurança, ainda que em redundância com o circuito eletrônico desenvolvido, conforme verificado no item 4.3.2.2, foi efetuado uma exclusão lógica para cada elemento, de forma que não seja possível acionar os dois estados de um trem ou desvio.

4.4.1 ADAPTAÇÃO NO CÓDIGO PRINCIPAL ADA

No código principal ADA, é efetuado uma chamada da função intermediária diretamente em C. Neste caso o nome do arquivo e código executável em C são utilizados como argumento para o processo de compilador do ADA.

O código e o executável gerados em C são assim interpretados pelo ADA como um item de biblioteca ativado diretamente pela chamada de função. Como exemplo temos o comando: pragma Import (C, Funcao, "parametros");, que efetua a chamada de um função genérica “Funcao” em C e ainda repassa alguns parâmetros para a função.

4.5 CONSTRUÇÃO DO SISTEMA

Para efetuar testes e verificações do projeto foi construído, nesta fase, a maquete da figura (4.8). A qual permite efetuar os principais testes de acionamento dos elementos do circuito, o trem e os desvios, e também efetuar testes de funcionalidade prática com o código implementado em ADA e sua iteração com o código intermediário.

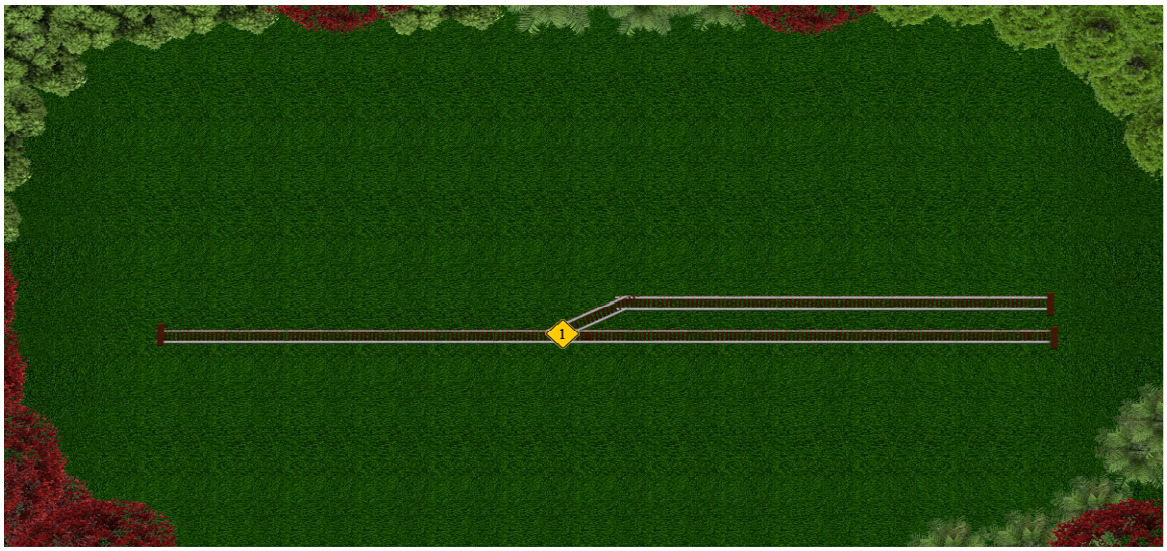


Figura 4.8. Maquete implementada para testes

Verifica-se, na figura (4.9), uma representação da comunicação entre os sistemas do projeto.

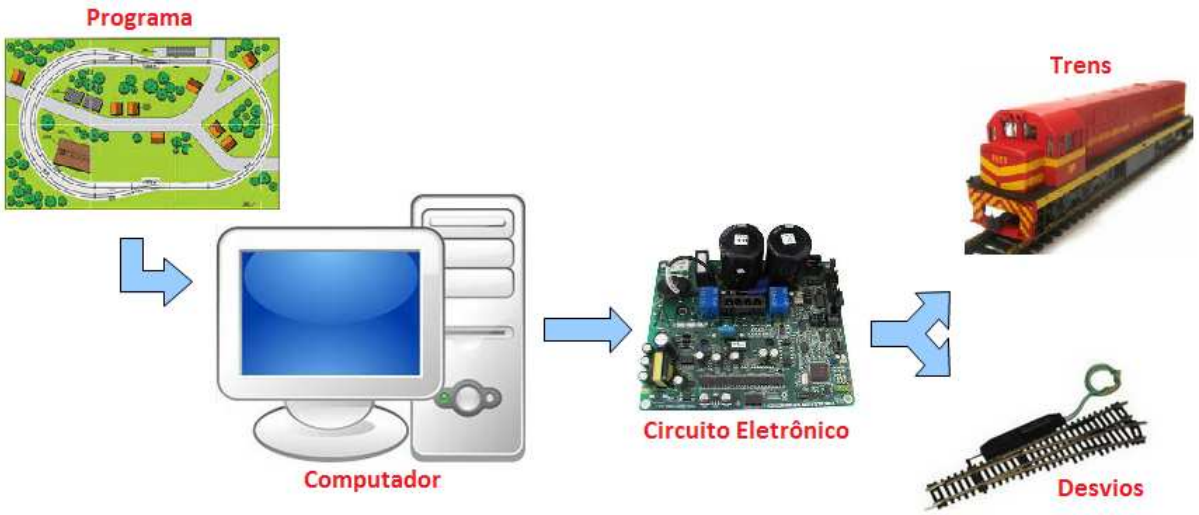


Figura 4.9. Representação dos Sistemas do Projeto.

A montagem completa do circuito foi realizada conforme as figuras (4.10) e (4.11), e representou a integração entre computador, circuito de interface e maquete.

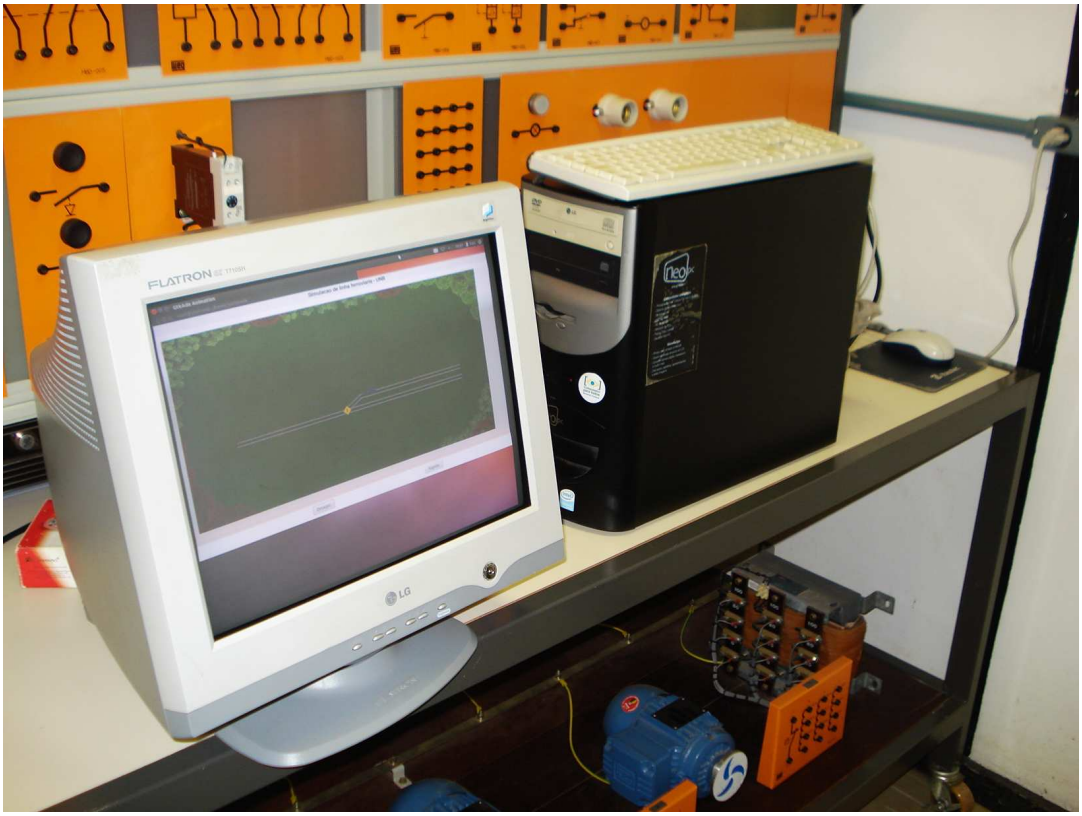


Figura 4.10. Foto 1 do Protótipo.

Figura 4.11. Foto 2 do protótipo.

4.5.1 ADAPTAÇÃO DO TREM

Uma vez que os trens miniaturizados da Frateschi operam através do acionamento energético dos trilhos, ou seja, a alimentação dos trilhos impulsionam o trem para a frente ou para trás de acordo com a polaridade empregada, foram necessárias adaptações para que o mesmo possa ser controlada através de um sistema de rádio.

A adaptação realizada foi a inserção de um circuito em ponte H, capaz de efetuar a inversão de tensão no motor do trem sem que se faça necessário a troca de polaridade no circuito de trilhos. Para esta operação foi efetuado a utilização do circuito integrado L298, que consiste de um circuito para pontes H que opera com potência suficiente para o acionamento do trem.

O circuito do trem pode ser verificado na figura (4.12) e possui acionamento direto do receptor de rádio.

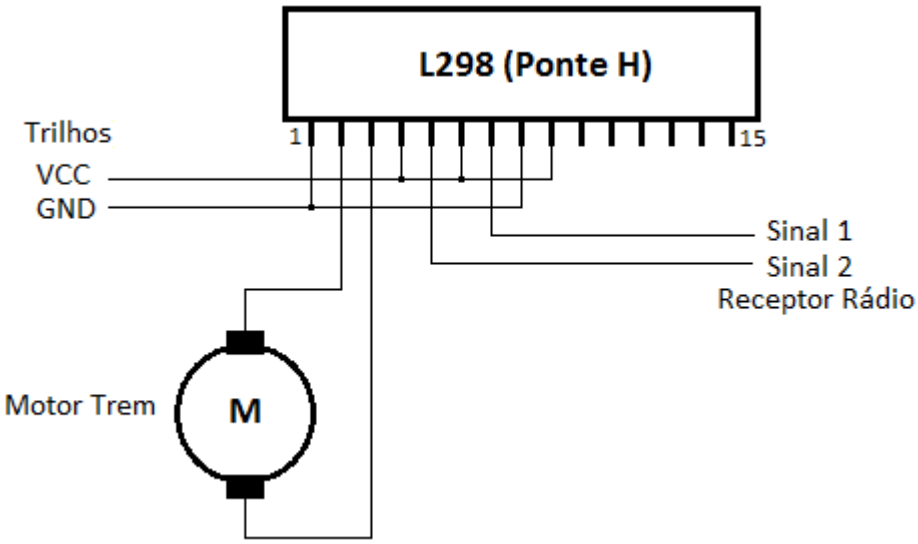


Figura 4.12. Circuito eletrônico do trem.

4.5.2 TESTES REALIZADOS

Inicialmente, foram efetuados testes individuais dos acionamentos para a garantia das funcionalidades básicas.

Desta forma foi iniciado o teste acionando o desvio individualmente e validando seu funcionamento e sua ativação.

O segundo teste efetuado foi o do acionamento do trem através do sistema de rádio, com controle direto através do computador. Neste passo, foram efetuados deslocamentos menores do trem para a frente e para trás. Nesta fase foi obtida por meio de inferência a velocidade de operação do trem, após algumas análises de seu deslocamento em um trecho de um metro, sendo assim verificado que o trem opera a uma velocidade média de 0,083 m/s.

A seguir, foi realizado um acionamento em vazio, sem desvio ou trem, do programa em ADA, para garantir o correto acionamento do trem e do desvio pelo código de controle desenvolvido. Também a velocidade inferida através dos testes do trem foi utilizada para o ajuste das velocidades da simulação, tanto gráfica quanto de ativação dos componentes da maquete.

Por fim, como teste final, foram realizados ajustes finos em relação ao deslocamento do trem em relação aos comandos computacionais da simulação para que o trem represente em paridade a maquete com o gráfico de acompanhamento do código.

5 CONCLUSÃO

5.1 RESULTADOS DO PROJETO

Os aspectos que envolvem o desenvolvimento de um sistema em tempo real caracterizam um grande desafio para a construção de um produto de engenharia que tenha em seu fim um sistema físico, como o caso da maquete ferroviária objeto deste projeto, acionado e controlado por um sistema computacional.

Desta forma a proposta de iniciar uma modelagem UML do sistema após as delimitações do escopo e dos objetivos resultaram em uma simplificação do trabalho uma vez que permitiram efetuar uma avaliação das partes conjuntas da aplicação antes do início da criação do programa. Ainda que nem todas as funcionalidades propostas no modelo tenham sido construídas no código, pelo simples fato de perderem sua relevância, a proporção maior dos modelos criados foram utilizadas na construção final.

Um fator que é considerado muito importante na construção do código do projeto e de sua interface gráfica está relacionado com a linguagem de programação utilizada. Sem dúvida a utilização da Linguagem ADA, que é predominantemente direcionada para o desenvolvimento de sistemas em tempo real foi determinante para o cumprimento de requisitos e serviu intensamente como canal de simplificação, mesmo no simples processo do pensar em tempo real quanto na construção de elementos, como os semáforos.

Foi atingida a obtenção satisfatória de uma aplicação capaz de efetuar o escalonamento dos recursos em uma planta ferroviária, permitindo segurança e algoritmos de otimização de percurso, que demonstram de forma direta as peculiaridades e potenciais de um sistema em tempo real.

A obtenção de um protótipo funcional, que permite a experimentação e o contato, dos operadores junto ao sistema, através de uma rede de circuitos eletrônicos interligados a maquete foi de significante importância. Implementar a integração computacional com a maquete traz luz a realidade que pode ser obtida através de uma intervenção metodológica sobre um sistema, e teve por fim o contato com os fatores ambientais e restritivos que não pode ser facilmente verificada em simulações computacionais.

5.2 DIFICULDADES ENCONTRADAS

O estudo e a utilização da linguagem de programação ADA, que não é largamente utilizada como o C, Java, Delphi e outras, demonstrou ser um grande fator de dificuldade no projeto. Apesar da linguagem apresentar pontos decisivos ao trabalhar com tempo real a busca por exemplos e bibliotecas pré-estabelecidas, principalmente no quesito gráfico, demonstrou-se complexa e de difícil referenciação. Um ponto que pode ser verificado nesta dificuldade foi a necessidade de introduzir um código em C dentro do código ADA com a finalidade de operar os endereços de memória da porta paralela.

Um segundo desafio foi a avaliação e desenvolvimento de sistemas inteligentes que permitissem aos trens maior capacidade de ação dentro da malha viária. A exemplo temos o estudo e implementação do algoritmo para a obtenção do melhor caminho, responsável por uma melhora significativa na capacidade de independência dos trens no circuito.

A implementação dos circuitos eletrônicos e da maquete, principalmente a montagem da integração entre computador e ambiente físico apresentou uma série de dificuldades, entre elas podemos citar a necessidade de aquisição de diversos componentes, tanto material quanto ferramental, para a construção dos sistemas, dificuldades em efetuar adaptação a produtos pré-definidos de mercado, como o caso da intervenção no acionamento e alimentação dos trens, e também a perda de componentes na fase de testes, como o caso da queima de alguns desvios da maquete durante a construção dos elementos de potência do sistema.

5.3 TRABALHOS FUTUROS

Com o término do protótipo da malha ferroviária, pode ser verificado uma série de melhorias e implementações funcionais para o projeto, bem como a construção plena da maquete em seu traçado proposto nas definições dos objetivos.

Dentre os principais fatores que podem ser melhorados estão a implementação de uma interface com usuários que permita uma maior intervenção do operador, como por exemplo efetuar comandos diretos de parada ou mudança nas trajetórias estipuladas dos trens.

Outro fator que deve ser considerado nos projetos futuros é a implantação de melhorias nos sistemas de segurança, tanto em nível de software quanto de hardware para maior proteção dos componentes da maquete, como questões relativas a sobrecarga de tensão nos circuitos de rádio e aquecimento das bobinas dos desvios e motores dos trens.

Como proposta futura tem-se também a adição de uma série de sensores ao longo das vias para maior controle de posição dos trens e dos estados dos desvios, minimizando principalmente reforços de atuação, como a necessidade de ativação contínua do desvio para garantir seu correto posicionamento ante a passagem de um trem.

Por fim tem-se a proposta futura de substituição do atual sistema de comunicação do computador, que é efetuado pela porta paralela, por um que permita um incremento de funcionalidades e segurança, como por exemplo o emprego de um micro controlador para realizar a interface entre computador e maquete.

5.4 CONSIDERAÇÕES FINAIS

A ordem como foi visualizado e concebido o Projeto Trem certamente gerou conhecimentos valiosos por parte de todos os envolvidos no mesmo. A implementação de um projeto, que envolve um estudo apurado e com metodologias tem por finalidade uma extrapolação maior da nossa capacidade de compreender e intervir no mundo.

As muitas dificuldades encontradas no desenvolvimento de um produto de engenharia, como a planta ferroviária, nos leva a um novo nível de discussão e de nossa capacidade de arquitetar e realizar solução efetivas e eficientes. Um projeto tão abrangente como este nos permitiu e estimulou a utilização dos conhecimentos obtidos ao longo do curso de Engenharia Mecatrônica.

O pensamento do construtor, que possui a capacidade de ir um passo adiante da realidade em que habita, aqui representado na figura do engenheiro, é o grande feito e resultado, não apenas deste projeto, mas dos esforços e anos de estudo dedicados a prática das ciências.

REFERENCIAS BIBLIOGRAFICAS

ADA. Ada 2012 Language Reference Manual. Ada-auth.org. 2014.

AXELSON, J. **Parallel port complete: programming, interfacing & using the PC's parallel printer port**. Lakeview Research. 1996. p. 343.

BAXTER, M. **Projeto de produto: guia prático para o Design de novos produtos**. 2. ed. São Paulo. Edgard Blucher, 2000.

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro. Elsevier. 2007. p. 286.

BOOCH, G.; RUMBAUGH J.; JACOBSON I. **Unified Modeling Language User Guide, The**. 1 ed. Addison Wesley. 1998.

BOOCH, G.; RUMBAUGH, J; JACOBSON, I. **UML: guia do usuário**. 2. ed. Rio de Janeiro. Elsevier. 2006. p 474.

BURNS, A.; WELLINGS, A. **Concurrent and Real Time Programming in Ada**. 1 ed. Cambridge University Press: Cambridge, 2007.

BUTTAZZO, G. C. **Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications**. 3 ed. Springer. 2011.

DORFMAN, M. J.; MEDANIC, J. **Scheduling trains on a railway network using a discrete event model of railway traffi**c. Transportation Research Part B 38 (1). 2004. p. 81–98.

ECLIPSE. Disponível em: <http://www.eclipse.org/papyrus/>. Acessado em 15 abr. 2014.

EL PAÍS. Disponível em: <http://politica.elpais.com/politica/2013/07/25/actualidad/1374713876_139202.html>. Acesso em: 23 mar. 2014.

FELDMAN, M. Who's using Ada?. SIGAda Education Working Group. 2012.

GADRE, D. V. **Programming the Parallel Port: Interfacing the PC for Data Acquisition and Process Control**. R&D Books. 2000.

Industria Reunidas Frateschi Ltda. **Manual de Iniciação, Construção de Maquetes Ferroviárias**. 2012. Disponível em: < http://www.frateschi.com.br/site/?page=publicacoes >. Acesso em: 26/03/2013.

LARMAN, C. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientado a objetos e ao desenvolvimento iterativo**. 3. ed. Porto Alegre, RS. Bookman. 2007. p 695.

LOBACH, B. **Design Industrial: bases para a configuração dos produtos industriais**. São Paulo. Edgard Blucher. 2001. p 206.

MARTIN, P. **Train Performance and Simulation**. Winter Simulation Conference. 1999; p. 1287-94.

MAZZOLA, V. B.; FARINES, J. M. **Metodologias de Concepção de Software e de Sistemas**. Apostila de curso de Especialização. UFSC. 2004.

MEDEIROS, E. **Desenvolvendo software com UML 2.0**. São Paulo. Pearson Makron Books. 2004. p. 264.

NISSANKE, N. **Realtime Systems**. Pretince Hall. 1997.

PIDD, M. **Systems Modelling: Theory and Practice**. John Wiley & Sons. 2004.

PINKAVA, V. **Introduction to Logic for Systems Modelling**. Taylor & Francis. 1988.

SEDGEWICK, R. **Algorithms in C**. 3 ed. Addison-Wesley/Longman. 2002.

SIPSER, M. Introduction to the Theory of Computation. 2 ed. Brooks/Cole Pub Co. 2005.

SOMMERVILLE, I. **Engenharia de software**. 8 ed. São Paulo. Pearson Wesley. 2007. p. 552.

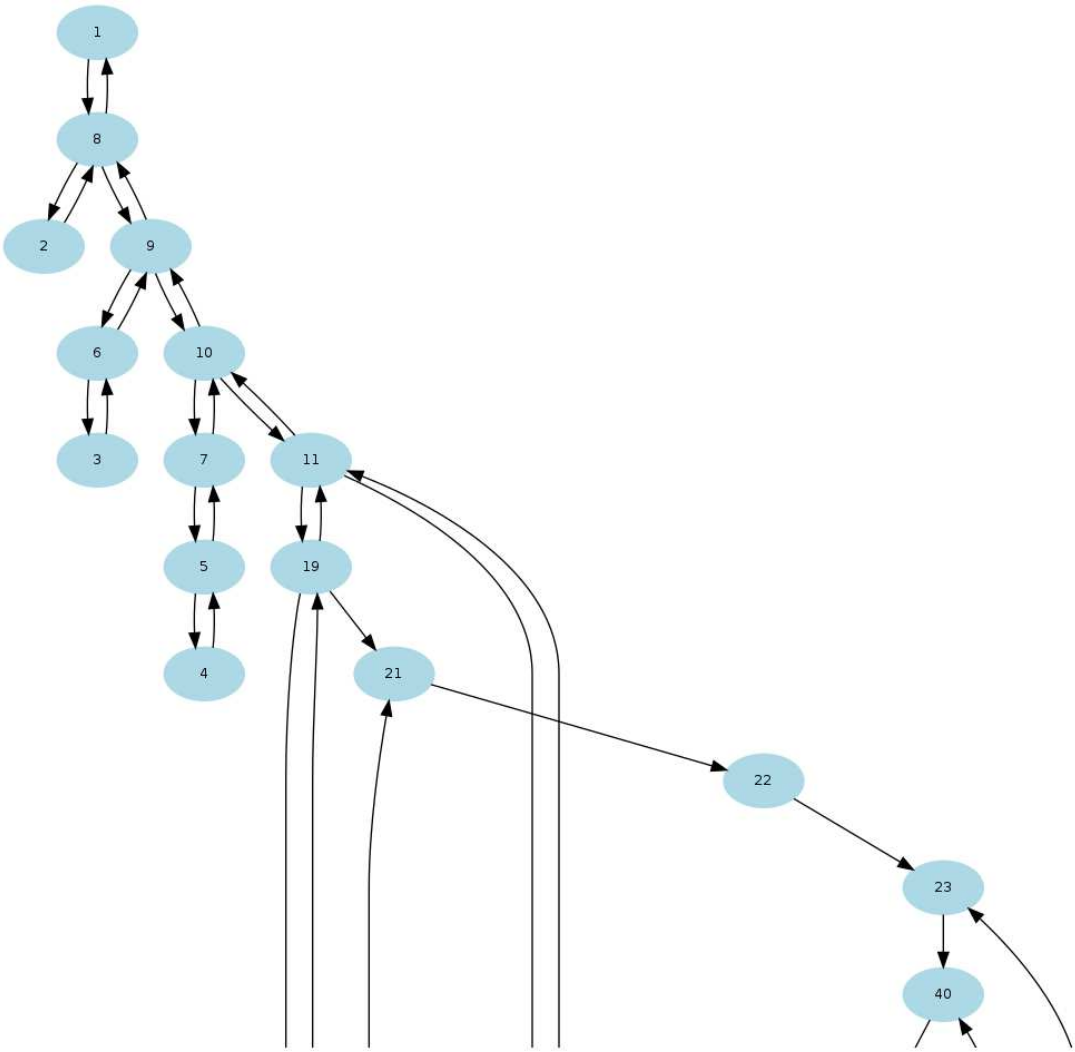
TimeSys Corporation. **The Concise Handbook Of Real-Time Systems**. V1.3. 2002.

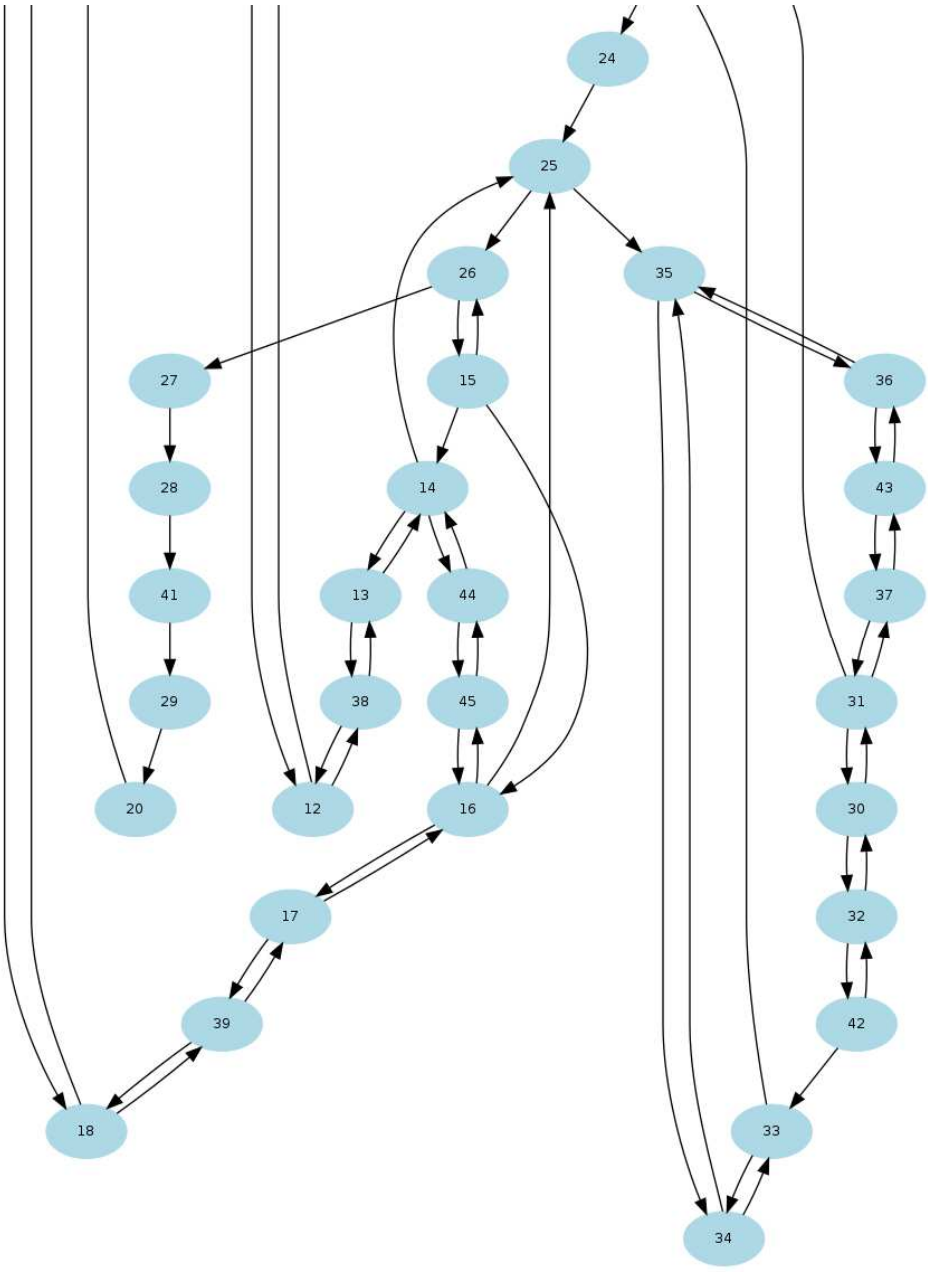
YOUNG, J. W.; KENT, H. K. **Abstract Formulation of Data Processing Problems**. In: Journal of Industrial Engineering. Nov-Dec 1958. 9(6), p. 471-479.

WAZLAWICK, R. S. **Análise e Projeto de Sistemas de Informação Orientado a Objetos**. Ed. Campus. 2004.

ANEXOS

ANEXO I: Grafo bidirecional da versão 4 do código




24
25
26
35
27
15
36
28
14
43
41
13
44
37
29
38
45
31
20
12
16
30
17
32
39
42
18
33
34

FAIRCHILD SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS138 • DM74LS139 Decoder/Demultiplexer

General Description

These Schottky-clamped circuits are designed to be used in high-performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible.

The DM74LS138 decodes one-of-eight lines, based upon the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented with no external inverters, and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

The DM74LS139 comprises two separate two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

All of these decoders/demultiplexers feature fully buffered inputs, presenting only one normalized load to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design.

Features

- Designed specifically for high speed:
  - Memory decoders
  - Data transmission systems
- DM74LS138 3-to-8-line decoders incorporates 3 enable inputs to simplify cascading and/or data reception
- DM74LS139 contains two fully independent 2-to-4-line decoders/demultiplexers
- Schottky clamped for high performance
- Typical propagation delay (3 levels of logic)
  - DM74LS138 21 ns
  - DM74LS139 21 ns
- Typical power dissipation
  - DM74LS138 32 mW
  - DM74LS139 34 mW

Ordering Code:

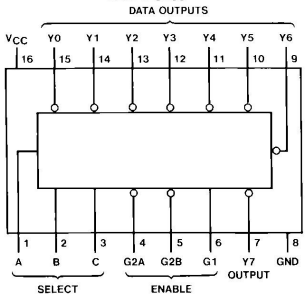
Order Number	Package Number	Package Description
DM74LS138M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS138SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS138N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
DM74LS139M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS139SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS139N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

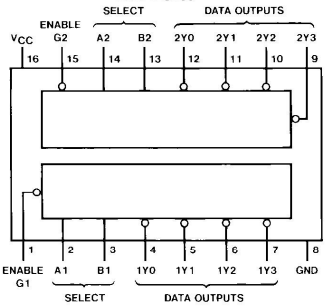
 DS006391

Connection Diagrams

DM74LS138



DM74LS139



Function Tables

DM74LS138

Inputs					Outputs						
Enable		Select									
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

DM74LS139

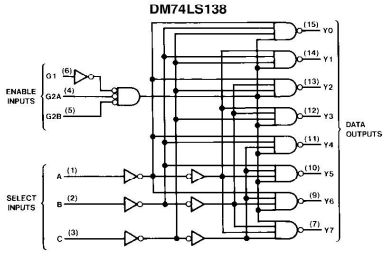
Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Level
L = LOW Level
X = Don't Care

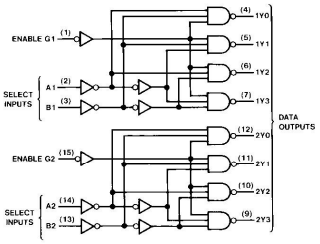
Note 1: G2 = G2A + G2B

Logic Diagrams

DM74LS138



DM74LS139



DM74LS139 Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

DM74LS139 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 6)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 4 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 7)	−20		−100	mA
I_{CC}	Supply Current	V_{CC} = Max (Note 8)		6.8	11	mA

Note 6: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 7: Not more than one output should be shorted at a time, and the duration should not exceed one second.

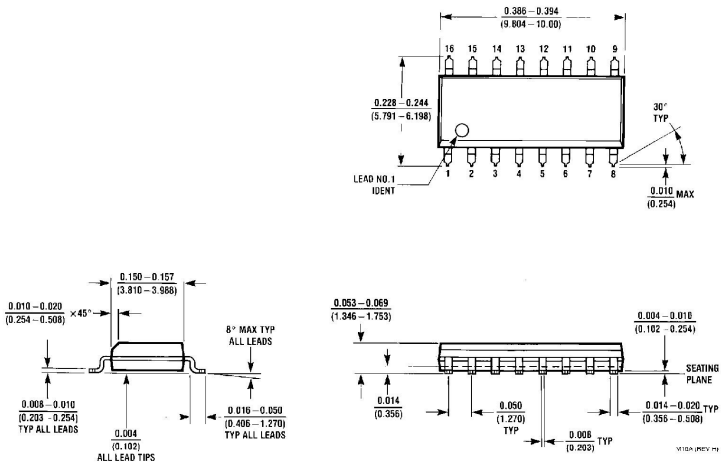
Note 8: I_{CC} is measured with all outputs enabled and OPEN.

DM74LS139 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

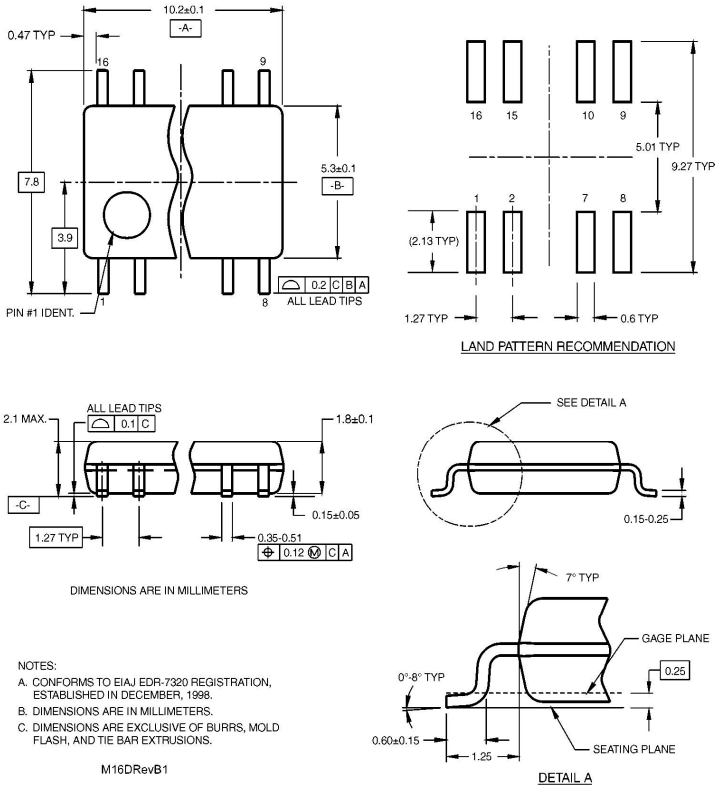
Symbol	Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
			C_L = 15 pF		C_L = 50 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output		24		40	ns

Physical Dimensions inches (millimeters) unless otherwise noted



16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
Package Number M16A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



NOTES:

A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.

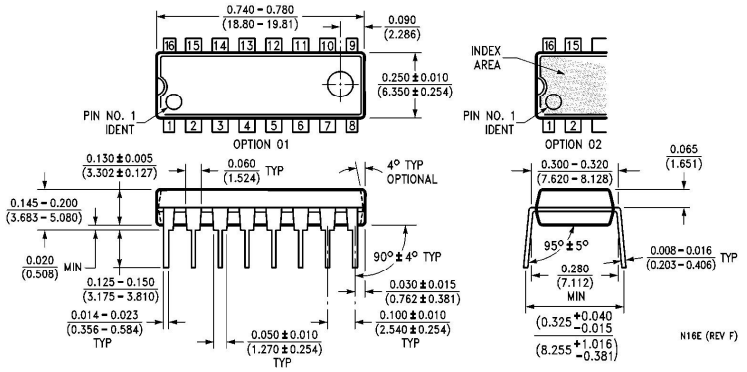
B. DIMENSIONS ARE IN MILLIMETERS.

C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

M16DRevB1

16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Number M16D

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
Package Number N16E

FAIRCHILD SEMICONDUCTOR™

April 1986
Revised March 2000

DM74LS373 • DM74LS374
3-STATE Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

General Description

These 8-bit registers feature totem-pole 3-STATE outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance state and increased high-logic level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the DM74LS373 are transparent D-type latches meaning that while the enable (G) is HIGH the Q outputs will follow the data (D) inputs. When the enable is taken LOW the output will be latched at the level of the data that was set up.

The eight flip-flops of the DM74LS374 are edge-triggered D-type flip flops. On the positive transition of the clock, the Q outputs will be set to the logic states that were set up at the D inputs.

A buffered output control input can be used to place the eight outputs in either a normal logic state (HIGH or LOW logic levels) or a high-impedance state. In the high-impedance state the outputs neither load nor drive the bus lines significantly.

The output control does not affect the internal operation of the latches or flip-flops. That is, the old data can be retained or new data can be entered even while the outputs are OFF.

Features

- Choice of 8 latches or 8 D-type flip-flops in a single package
- 3-STATE bus-driving outputs
- Full parallel-access for loading
- Buffered control inputs
- P-N-P inputs reduce D-C loading on data lines

Ordering Code:

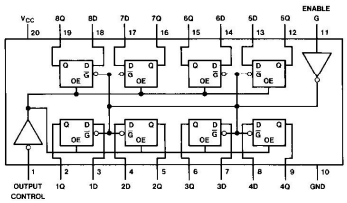
Order Number	Package Number	Package Description
DM74LS373WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS373SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS373N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
DM74LS374WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS374SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
IDM29901NC	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

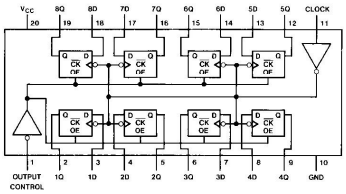
 DS006431

Connection Diagrams

DM74LS373



DM74LS374



Function Tables

DM74LS373

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

DM74LS374

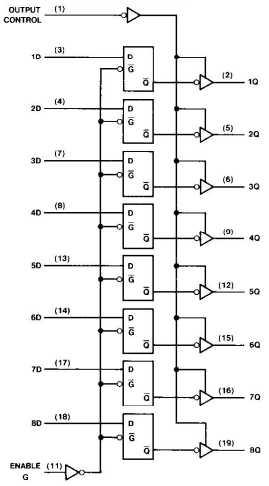
Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

H = HIGH Level (Steady State) L = LOW Level (Steady State) X = Don't Care Z = High Impedance State

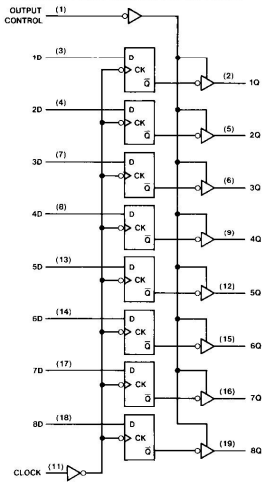
↑ = Transition from LOW-to-HIGH level Q_0 = The level of the output before steady-state input conditions were established.

Logic Diagrams

DM74LS373
Transparent Latches



DM74LS374
Positive-Edge-Triggered Flip-Flops



Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Storage Temperature Range	−65°C to +150°C
Operating Free Air Temperature Range	0°C to +70°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

DM74LS373 Recommended Operating Conditions

Symbol	Parameter		Min	Nom	Max	Units
V_{CC}	Supply Voltage		4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage		2			V
V_{IL}	LOW Level Input Voltage				0.8	V
I_{OH}	HIGH Level Output Current				−2.6	mA
I_{OL}	LOW Level Output Current				24	mA
t_W	Pulse Width	Enable HIGH	15			ns
	(Note 3)	Enable LOW	15			
t_{SU}	Data Setup Time (Note 2) (Note 3)		5↓			ns
t_H	Data Hold Time (Note 2) (Note 3)		20↓			ns
T_A	Free Air Operating Temperature		0		70	°C

Note 2: The symbol (↓) indicates the falling edge of the clock pulse is used for reference.

Note 3: $T_A = 25°C$ and $V_{CC} = 5V$.

DM74LS373 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 4)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.4	3.1		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 12 mA, V_{CC} = Min			0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	μA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OZH}	Off-State Output Current with HIGH Level Output Voltage Applied	V_{CC} = Max, V_O = 2.7V, V_{IH} = Min, V_{IL} = Max			20	μA
I_{OZL}	Off-State Output Current with LOW Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max			−20	μA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 5)	−50		−225	mA
I_{CC}	Supply Current	V_{CC} = Max, OC = 4.5V, D_n, Enable = GND		24	40	mA

Note 4: All typicals are at $V_{CC} = 5V$, $T_A = 25°C$.

Note 5: Not more than one output should be shorted at a time, and the duration should not exceed one second.

DM74LS373 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	From (Input) To (Output)	R_L = 667Ω				Units
			C_L = 45 pF		C_L = 150 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Data to Q		18		26	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Data to Q		18		27	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Q		30		38	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Q		30		36	ns
t_{PZH}	Output Enable Time to HIGH Level Output	Output Control to Any Q		28		36	ns
t_{PZL}	Output Enable Time to LOW Level Output	Output Control to Any Q		36		50	ns
t_{PHZ}	Output Disable Time from HIGH Level Output (Note 6)	Output Control to Any Q		20			ns
t_{PLZ}	Output Disable Time from LOW Level Output (Note 6)	Output Control to Any Q		25			ns

Note 6: C_L = 5 pF.

DM74LS374 Recommended Operating Conditions

Symbol	Parameter		Min	Nom	Max	Units
V_{CC}	Supply Voltage		4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage		2			V
V_{IL}	LOW Level Input Voltage				0.8	V
I_{OH}	HIGH Level Output Current				−2.6	mA
I_{OL}	LOW Level Output Current				24	mA
t_W	Pulse Width (Note 8)	Clock HIGH	15			ns
		Clock LOW	15			
t_{SU}	Data Setup Time (Note 7) (Note 8)		20↑			ns
t_H	Data Hold Time (Note 7) (Note 8)		1↑			ns
T_A	Free Air Operating Temperature		0		70	°C

Note 7: The symbol (↑) indicates the rising edge of the clock pulse is used for reference.

Note 8: T_A = 25°C and V_{CC} = 5V.

DM74LS374 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 9)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.4	3.1		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 12 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	μA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OZH}	Off-State Output Current with HIGH Level Output Voltage Applied	V_{CC} = Max, V_O = 2.7V, V_{IH} = Min, V_{IL} = Max			20	μA
I_{OZL}	Off-State Output Current with LOW Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max			−20	μA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 10)	−50		−225	mA
I_{CC}	Supply Current	V_{CC} = Max, D_n = GND, OC = 4.5V		27	45	mA

Note 9: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 10: Not more than one output should be shorted at a time, and the duration should not exceed one second.

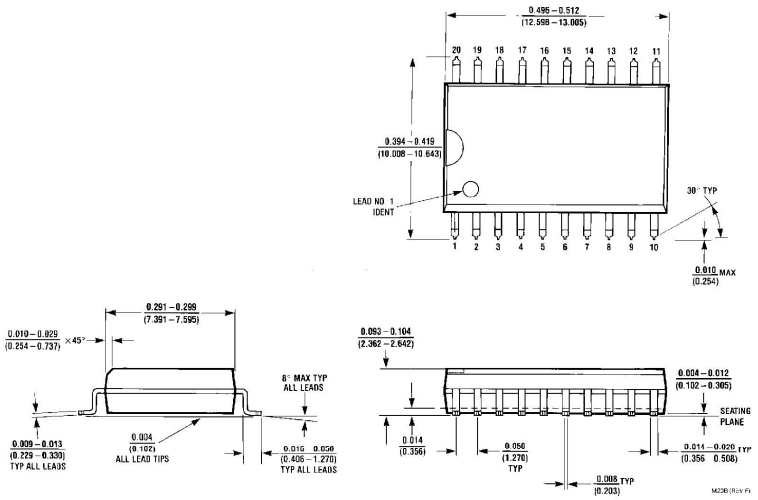
DM74LS374 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	R_L = 667Ω				Units
		C_L = 45 pF		C_L = 150 pF		
		Min	Max	Min	Max	
f_{MAX}	Maximum Clock Frequency	35		20		MHz
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output		28		32	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output		28		38	ns
t_{PZH}	Output Enable Time to HIGH Level Output		28		44	ns
t_{PZL}	Output Enable Time to LOW Level Output		28		44	ns
t_{PHZ}	Output Disable Time from HIGH Level Output (Note 11)		20			ns
t_{PLZ}	Output Disable Time from LOW Level Output (Note 11)		25			ns

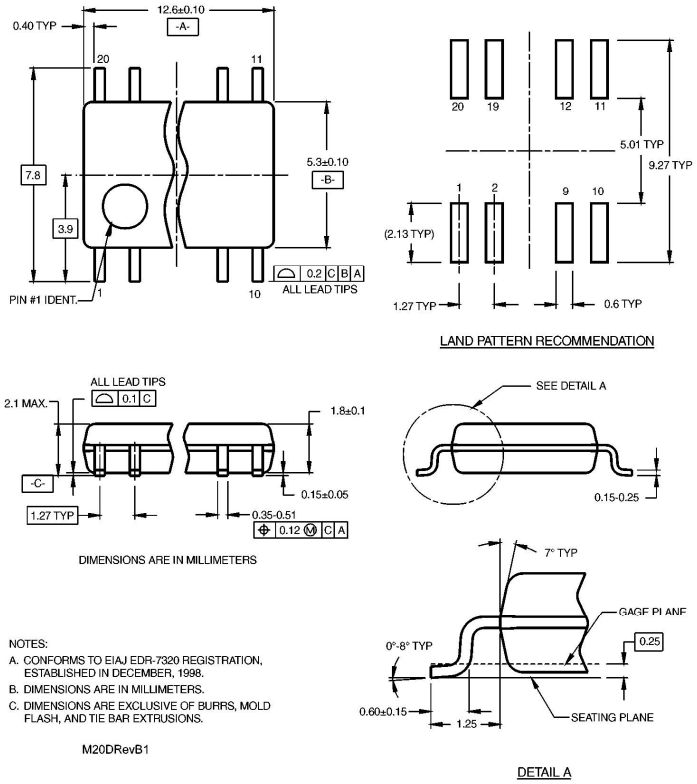
Note 11: C_L = 5 pF.

Physical Dimensions inches (millimeters) unless otherwise noted



20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
Package Number M20B

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



NOTES:

A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.

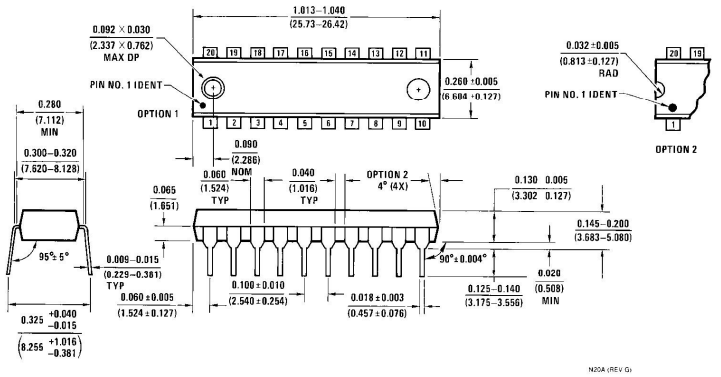
B. DIMENSIONS ARE IN MILLIMETERS.

C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

M20DRevB1

20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Number M20D

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
Package Number N20A

TEXAS INSTRUMENTS

CD4051B, CD4052B, CD4053B

Data sheet acquired from Harris Semiconductor
SCHS047D

August 1998 - Revised March 2000

CMOS Analog Multiplexers/Demultiplexers with Logic Level Conversion

The CD4051B, CD4052B, and CD4053B analog multiplexers are digitally-controlled analog switches having low ON impedance and very low OFF leakage current. Control of analog signals up to $20V_{P-P}$ can be achieved by digital signal amplitudes of 4.5V to 20V (if V_{DD}-V_{SS} = 3V, a V_{DD}-V_{EE} of up to 13V can be controlled; for V_{DD}-V_{EE} level differences above 13V, a V_{DD}-V_{SS} of at least 4.5V is required). For example, if V_{DD} = +4.5V, V_{SS} = 0V, and V_{EE} = -13.5V, analog signals from -13.5V to +4.5V can be controlled by digital inputs of 0V to 5V. These multiplexer circuits dissipate extremely low quiescent power over the full V_{DD}-V_{SS} and V_{DD}-V_{EE} supply-voltage ranges, independent of the logic state of the control signals. When a logic "1" is present at the inhibit input terminal, all channels are off.

The CD4051B is a single 8-Channel multiplexer having three binary control inputs, A, B, and C, and an inhibit input. The three binary signals select 1 of 8 channels to be turned on, and connect one of the 8 inputs to the output.

The CD4052B is a differential 4-Channel multiplexer having two binary control inputs, A and B, and an inhibit input. The two binary input signals select 1 of 4 pairs of channels to be turned on and connect the analog inputs to the outputs.

The CD4053B is a triple 2-Channel multiplexer having three separate digital control inputs, A, B, and C, and an inhibit input. Each control input selects one of a pair of channels which are connected in a single-pole, double-throw configuration.

When these devices are used as demultiplexers, the "CHANNEL IN/OUT" terminals are the outputs and the "COMMON OUT/IN" terminals are the inputs.

Ordering Information

PART NUMBER	TEMP. RANGE (^oC)	PACKAGE
CD4051BF, CD4052BF, CD4053BF	-55 to 125	16 Ld CERAMIC DIP
CD4051BE, CD4052BE, CD4053BE	-55 to 125	16 Ld PDIP
CD4051BM, CD4051BNS	-55 to 125	16 Ld SOIC
CD4051BPW, CD4052BPW, CD4053BPW	-55 to 125	16 Ld TSSOP

Features

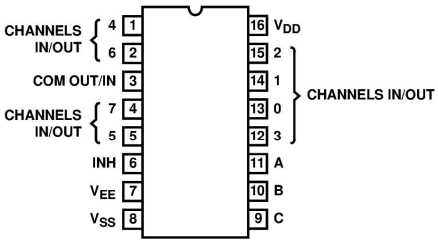
- Wide Range of Digital and Analog Signal Levels
  - Digital . . . . . 3V to 20V
  - Analog. . . . . $\leq 20V_{P-P}$
- Low ON Resistance, 125Ω (Typ) Over $15V_{P-P}$ Signal Input Range for V_{DD}-V_{EE} = 18V
- High OFF Resistance, Channel Leakage of ±100pA (Typ) at V_{DD}-V_{EE} = 18V
- Logic-Level Conversion for Digital Addressing Signals of 3V to 20V (V_{DD}-V_{SS} = 3V to 20V) to Switch Analog Signals to $20V_{P-P}$ (V_{DD}-V_{EE} = 20V)
- Matched Switch Characteristics, r_{ON} = 5Ω (Typ) for V_{DD}-V_{EE} = 15V
- Very Low Quiescent Power Dissipation Under All Digital-Control Input and Supply Conditions, 0.2μW (Typ) at V_{DD}-V_{SS} = V_{DD}-V_{EE} = 10V
- Binary Address Decoding on Chip
- 5V, 10V and 15V Parametric Ratings
- 10% Tested for Quiescent Current at 20V
- Maximum Input Current of 1μA at 18V Over Full Package Temperature Range, 100nA at 18V and 25^oC
- Break-Before-Make Switching Eliminates Channel Overlap

Applications

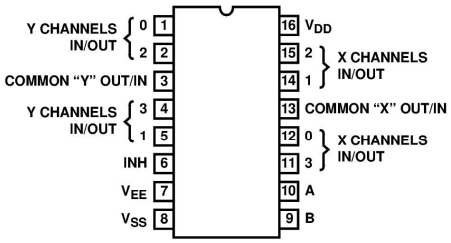
- Analog and Digital Multiplexing and Demultiplexing
- A/D and D/A Conversion
- Signal Gating

Pinouts

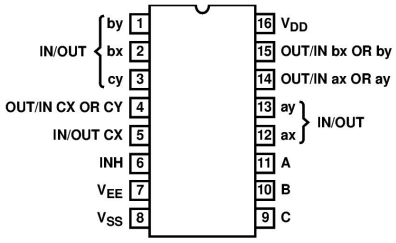
CD4051B (PDIP, CDIP, SOIC, TSSOP)
TOP VIEW



CD4052B (PDIP, CDIP, TSSOP)
TOP VIEW

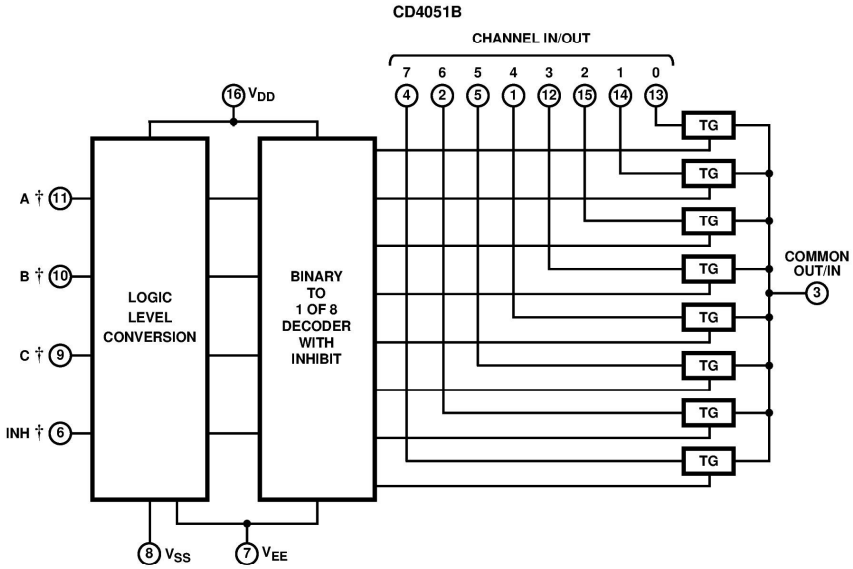


CD4053B (PDIP, CDIP, TSSOP)
TOP VIEW



Functional Block Diagrams

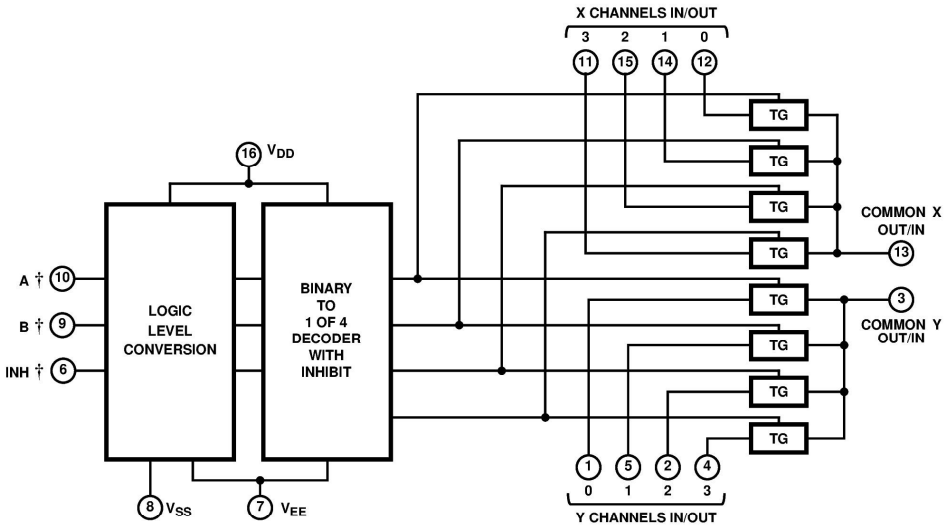
CD4051B



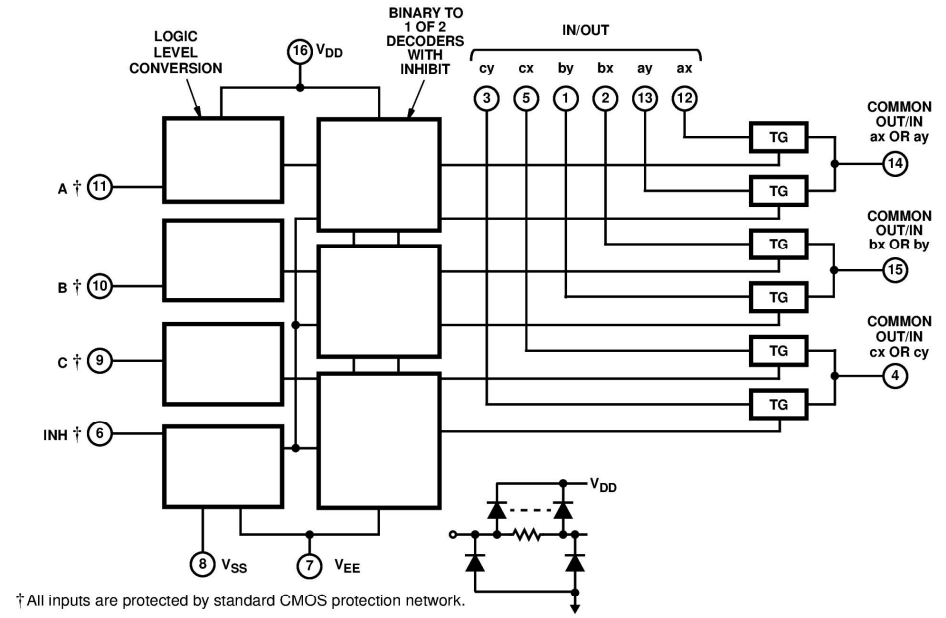
†All inputs are protected by standard CMOS protection network.

Functional Block Diagrams (Continued)

CD4052B



CD4053B



† All inputs are protected by standard CMOS protection network.

TRUTH TABLES

INPUT STATES				
INHIBIT	**C**	**B**	**A**	**"ON" CHANNEL(S)**
CD4051B				
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	None
CD4052B				
INHIBIT		**B**	**A**	
0		0	0	0x, 0y
0		0	1	1x, 1y
0		1	0	2x, 2y
0		1	1	3x, 3y
1		X	X	None
CD4053B				
INHIBIT		**A OR B OR C**		
0		0		ax or bx or cx
0		1		ay or by or cy
1		X		None

X = Don't Care

Absolute Maximum Ratings

Supply Voltage (V+ to V-)
Voltages Referenced to V_{SS} Terminal . . . . . -0.5V to 20V
DC Input Voltage Range . . . . . -0.5V to V_{DD} +0.5V
DC Input Current, Any One Input. . . . . ±10mA

Operating Conditions

Temperature Range . . . . . -55°C to 125°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
E Package	67	N/A
F Package	115	45
D Package	73	N/A
NS Package	64	N/A
PW Package	108	N/A

Maximum Junction Temperature (Ceramic Package) . . . . . 175°C
Maximum Junction Temperature (Plastic Package) . . . . . 150°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 265°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. The package thermal impedance is calculated in accordance with JESD51.

Electrical Specifications

Common Conditions Here: If Whole Table is For the Full Temp. Range, V_{SUPPLY} = ±5V, A_V = +1, R_L = 100Ω, Unless Otherwise Specified (Note 3)

PARAMETER	CONDITIONS				LIMITS AT INDICATED TEMPERATURES (°C)							UNITS
									25			
	V_{IS} (V)	V_{EE} (V)	V_{SS} (V)	V_{DD} (V)	-55	-40	85	125	MIN	TYP	MAX	
SIGNAL INPUTS (V_{IS}) AND OUTPUTS (V_{OS})												
Quiescent Device Current, I_{DD} Max	-	-	-	5	5	5	150	150	-	0.04	5	µA
	-	-	-	10	10	10	300	300	-	0.04	10	µA
	-	-	-	15	20	20	600	600	-	0.04	20	µA
	-	-	-	20	100	100	3000	3000	-	0.08	100	µA
Drain to Source ON Resistance r_{ON} Max $0 \le V_{IS} \le V_{DD}$	-	0	0	5	800	850	1200	1300	-	470	1050	Ω
	-	0	0	10	310	330	520	550	-	180	400	Ω
	-	0	0	15	200	210	300	320	-	125	240	Ω
Change in ON Resistance (Between Any Two Channels), Δr_{ON}	-	0	0	5	-	-	-	-	-	15	-	Ω
	-	0	0	10	-	-	-	-	-	10	-	Ω
	-	0	0	15	-	-	-	-	-	5	-	Ω
OFF Channel Leakage Current: Any Channel OFF (Max) or ALL Channels OFF (Common OUT/IN) (Max)	-	0	0	18	±100 (Note 2)		±1000 (Note 2)		-	±0.01	±100 (Note 2)	nA
Capacitance:	-	-5	5-	5								
Input, C_{IS}					-	-	-	-	-	5	-	pF
Output, C_{OS}												
CD4051					-	-	-	-	-	30	-	pF
CD4052					-	-	-	-	-	18	-	pF
CD4053					-	-	-	-	-	9	-	pF
Feedthrough C_{IOS}					-	-	-	-	-	0.2	-	pF
Propagation Delay Time (Signal Input to Output	V_{DD}	R_L = 200kΩ, C_L = 50pF, t_r, t_f = 20ns		5	-	-	-	-	-	30	60	ns
				10	-	-	-	-	-	15	30	ns
				15	-	-	-	-	-	10	20	ns

Electrical Specifications Common Conditions Here: If Whole Table is For the Full Temp. Range, V_{SUPPLY} = ±5V, A_V = +1, R_L = 100Ω, Unless Otherwise Specified **(Continued)** (Note 3)

PARAMETER	CONDITIONS				LIMITS AT INDICATED TEMPERATURES (°C)							UNITS
									25			
	V_{IS} (V)	V_{EE} (V)	V_{SS} (V)	V_{DD} (V)	-55	-40	85	125	MIN	TYP	MAX	
CONTROL (ADDRESS OR INHIBIT), V_C												
Input Low Voltage, V_{IL}, Max	V_{IL} = V_{DD} through 1kΩ; V_{IH} = V_{DD} through 1kΩ	V_{EE} = V_{SS}, R_L = 1kΩ to V_{SS}, I_{IS} < 2μA on All OFF Channels		5	1.5	1.5	1.5	1.5	-	-	1.5	V
				10	3	3	3	3	-	-	3	V
				15	4	4	4	4	-	-	4	V
Input High Voltage, V_{IH}, Min				5	3.5	3.5	3.5	3.5	3.5	-	-	V
				10	7	7	7	7	7	-	-	V
				15	11	11	11	11	11	-	-	V
Input Current, I_{IN} (Max)	V_{IN} = 0, 18			18	±0.1	±0.1	±1	±1	-	$\pm 10^{-5}$	±0.1	μA
Propagation Delay Time: Address-to-Signal OUT (Channels ON or OFF) See Figures 10, 11, 14	t_r, t_f = 20ns, C_L = 50pF, R_L = 10kΩ	0	0	5	-	-	-	-	-	450	720	ns
		0	0	10	-	-	-	-	-	160	320	ns
		0	0	15	-	-	-	-	-	120	240	ns
		-5	0	5	-	-	-	-	-	225	450	ns
Propagation Delay Time: Inhibit-to-Signal OUT (Channel Turning ON) See Figure 11	t_r, t_f = 20ns, C_L = 50pF, R_L = 1kΩ	0	0	5	-	-	-	-	-	400	720	ns
		0	0	10	-	-	-	-	-	160	320	ns
		0	0	15	-	-	-	-	-	120	240	ns
		-10	0	5	-	-	-	-	-	200	400	ns
Propagation Delay Time: Inhibit-to-Signal OUT (Channel Turning OFF) See Figure 15	t_r, t_f = 20ns, C_L = 50pF, R_L = 10kΩ	0	0	5	-	-	-	-	-	200	450	ns
		0	0	10	-	-	-	-	-	90	210	ns
		0	0	15	-	-	-	-	-	70	160	ns
		-10	0	5	-	-	-	-	-	130	300	ns
Input Capacitance, C_{IN} (Any Address or Inhibit Input)					-	-	-	-	-	5	7.5	pF

NOTE:

2. Determined by minimum feasible leakage measurement for automatic testing.

Electrical Specifications

PARAMETER	TEST CONDITIONS					LIMITS	UNITS
	V_{IS} (V)	V_{DD} (V)	R_L (kΩ)			TYP	
Cutoff (-3dB) Frequency Channel ON (Sine Wave Input)	5 (Note 3)	10	1	V_{OS} at Common OUT/IN	CD4053	30	MHz
	V_{EE} = V_{SS}, $20\text{Log}\frac{V_{OS}}{V_{IS}} = -3\text{dB}$				CD4052	25	MHz
					CD4051	20	MHz
				V_{OS} at Any Channel		60	MHz

Electrical Specifications

PARAMETER	TEST CONDITIONS					LIMITS	UNITS
	V_{IS} (V)	V_{DD} (V)	R_L (kΩ)			TYP	
Total Harmonic Distortion, THD	2 (Note 3)	5	10			0.3	%
	3 (Note 3)	10				0.2	%
	5 (Note 3)	15				0.12	%
	$V_{EE} = V_{SS}$, f_{IS} = 1kHz Sine Wave						%
-40dB Feedthrough Frequency (All Channels OFF)	5 (Note 3)	10	1	V_{OS} at Common OUT/IN	CD4053	8	MHz
	$V_{EE} = V_{SS}$, $20\text{Log}\frac{V_{OS}}{V_{IS}} = -40\text{dB}$				CD4052	10	MHz
					CD4051	12	MHz
				V_{OS} at Any Channel		8	MHz
-40dB Signal Crosstalk Frequency	5 (Note 3)	10	1	Between Any 2 Channels		3	MHz
	$V_{EE} = V_{SS}$, $20\text{Log}\frac{V_{OS}}{V_{IS}} = -40\text{dB}$			Between Sections, CD4052 Only	Measured on Common	6	MHz
					Measured on Any Channel	10	MHz
				Between Any Two Sections, CD4053 Only	In Pin 2, Out Pin 14	2.5	MHz
					In Pin 15, Out Pin 14	6	MHz
Address-or-Inhibit-to-Signal Crosstalk	-	10	10 (Note 4)			65	mV_{PEAK}
	V_{EE} = 0, V_{SS} = 0, t_r, t_f = 20ns, V_{CC} = V_{DD} - V_{SS} (Square Wave)					65	mV_{PEAK}

NOTES:

3. Peak-to-Peak voltage symmetrical about $\frac{V_{DD} - V_{EE}}{2}$
4. Both ends of channel.

Typical Performance Curves

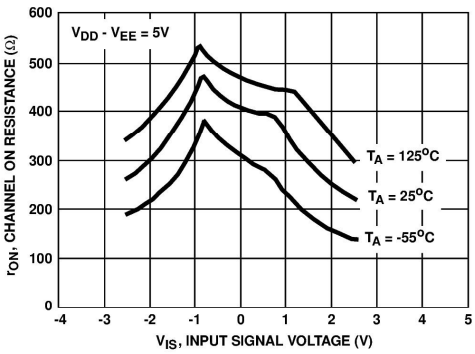


FIGURE 1. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

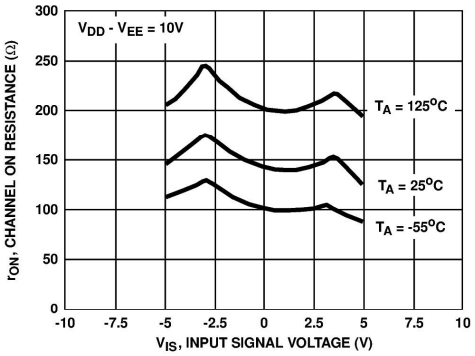


FIGURE 2. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

Typical Performance Curves (Continued)

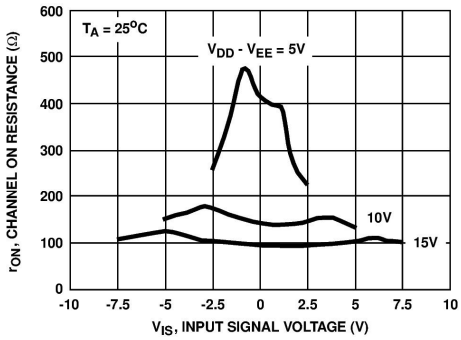


FIGURE 3. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

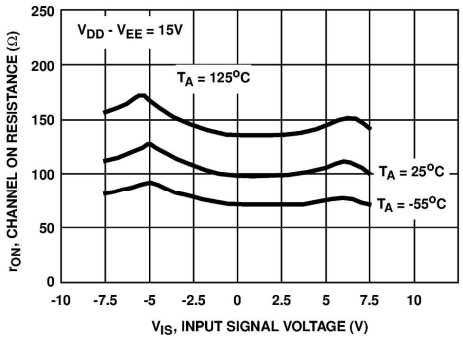


FIGURE 4. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

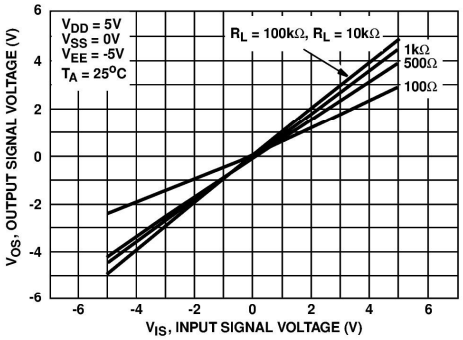


FIGURE 5. ON CHARACTERISTICS FOR 1 OF 8 CHANNELS (CD4051B)

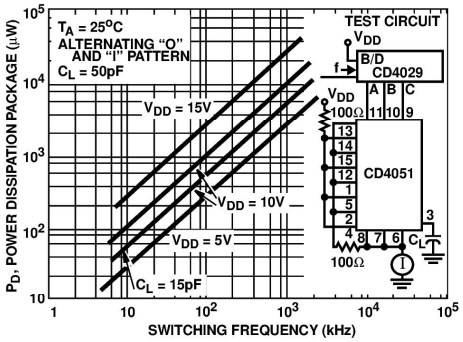


FIGURE 6. DYNAMIC POWER DISSIPATION vs SWITCHING FREQUENCY (CD4051B)

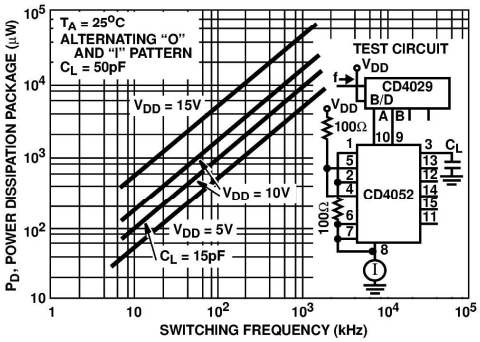


FIGURE 7. DYNAMIC POWER DISSIPATION vs SWITCHING FREQUENCY (CD4052B)

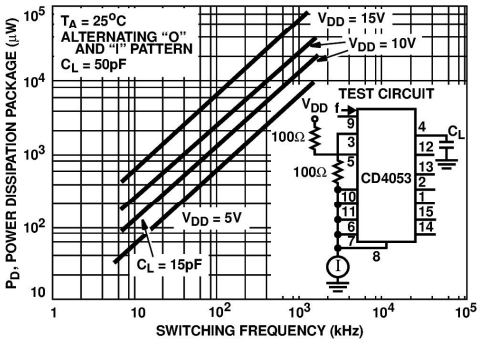
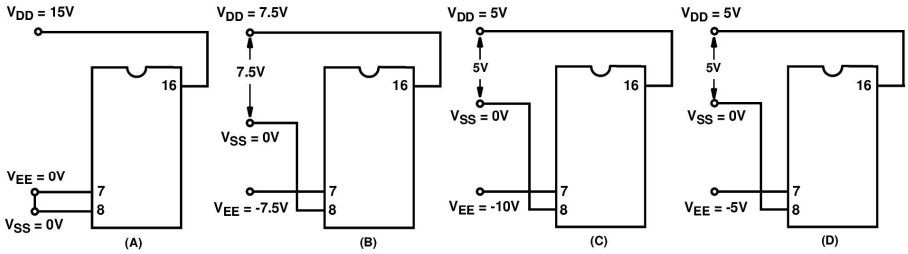


FIGURE 8. DYNAMIC POWER DISSIPATION vs SWITCHING FREQUENCY (CD4053B)

Test Circuits and Waveforms



NOTE: The ADDRESS (digital-control inputs) and INHIBIT logic levels are: "0" = V_{SS} and "1" = V_{DD}. The analog signal (through the TG) may swing from V_{EE} to V_{DD}.

FIGURE 9. TYPICAL BIAS VOLTAGES

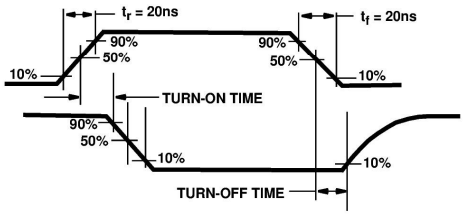


FIGURE 10. WAVEFORMS, CHANCEL BEING TURNED ON (R_L = 1kΩ)

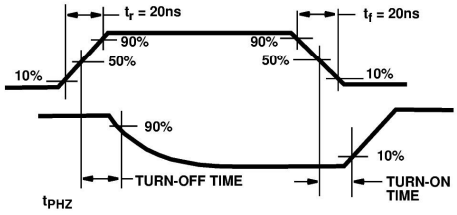


FIGURE 11. WAVEFORMS, CHANNEL BEING TURNED OFF (R_L = 1kΩ)

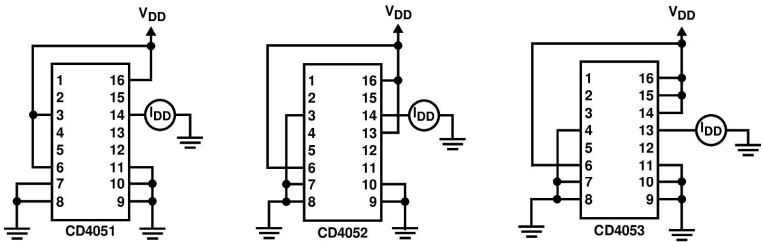


FIGURE 12. OFF CHANNEL LEAKAGE CURRENT - ANY CHANNEL OFF

Test Circuits and Waveforms (Continued)

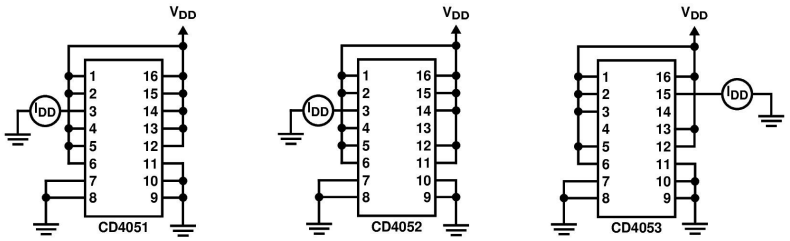


FIGURE 13. OFF CHANNEL LEAKAGE CURRENT - ALL CHANNELS OFF

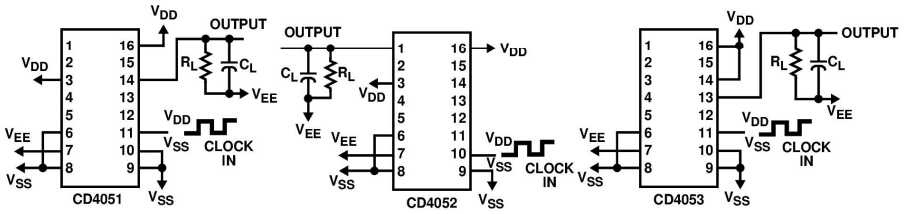


FIGURE 14. PROPAGATION DELAY - ADDRESS INPUT TO SIGNAL OUTPUT

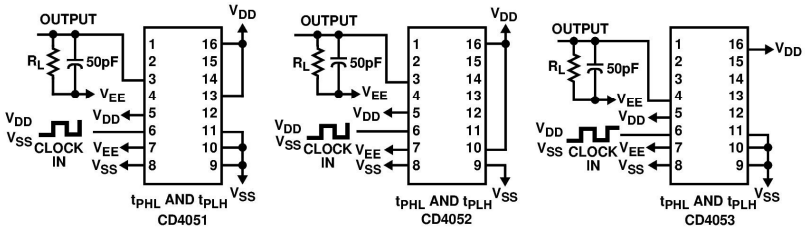


FIGURE 15. PROPAGATION DELAY - INHIBIT INPUT TO SIGNAL OUTPUT

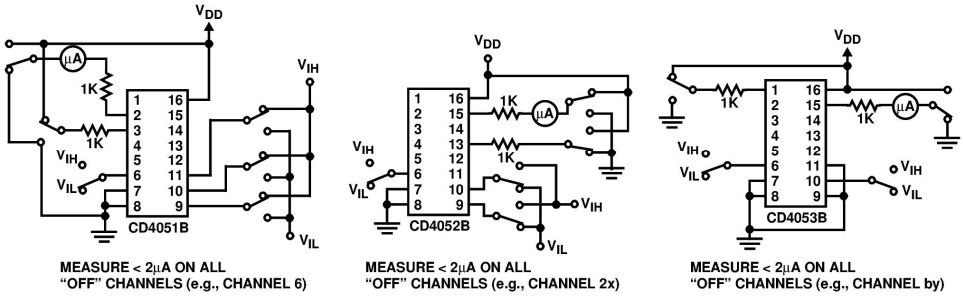


FIGURE 16. INPUT VOLTAGE TEST CIRCUITS (NOISE IMMUNITY)

Test Circuits and Waveforms (Continued)

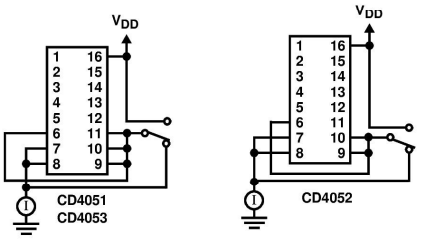


FIGURE 17. QUIESCENT DEVICE CURRENT

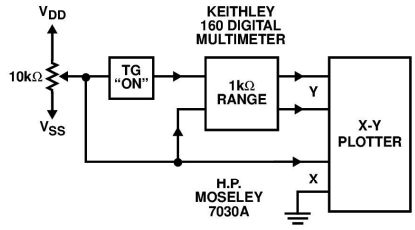


FIGURE 18. CHANNEL ON RESISTANCE MEASUREMENT CIRCUIT

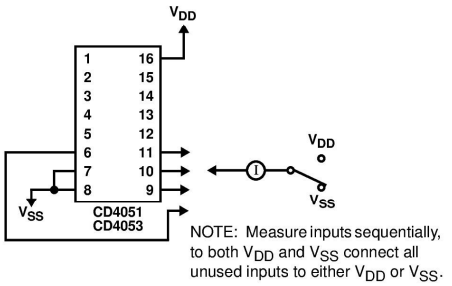


FIGURE 19. INPUT CURRENT

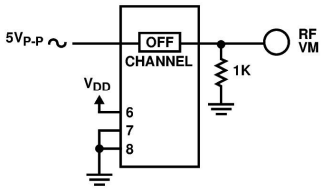


FIGURE 20. FEEDTHROUGH (ALL TYPES)

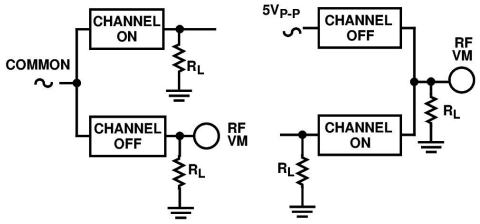


FIGURE 21. CROSSTALK BETWEEN ANY TWO CHANNELS (ALL TYPES)

FIGURE 22. CROSSTALK BETWEEN DUALS OR TRIPLETS (CD4052B, CD4053B)

Test Circuits and Waveforms (Continued)

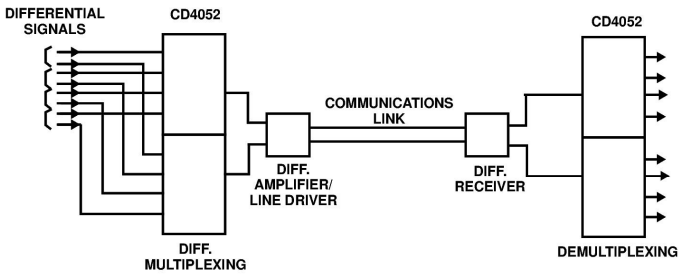


FIGURE 23. TYPICAL TIME-DIVISION APPLICATION OF THE CD4052B

Special Considerations

In applications where separate power sources are used to drive V_{DD} and the signal inputs, the V_{DD} current capability should exceed V_{DD}/R_L (R_L = effective external load). This provision avoids permanent current flow or clamp action on the V_{DD} supply when power is applied or removed from the CD4051B, CD4052B or CD4053B.

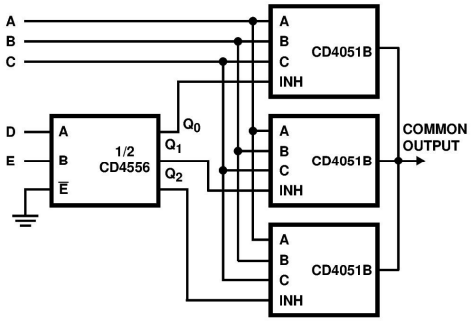


FIGURE 24. 24-TO-1 MUX ADDRESSING

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

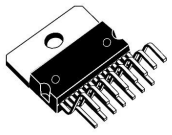
In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

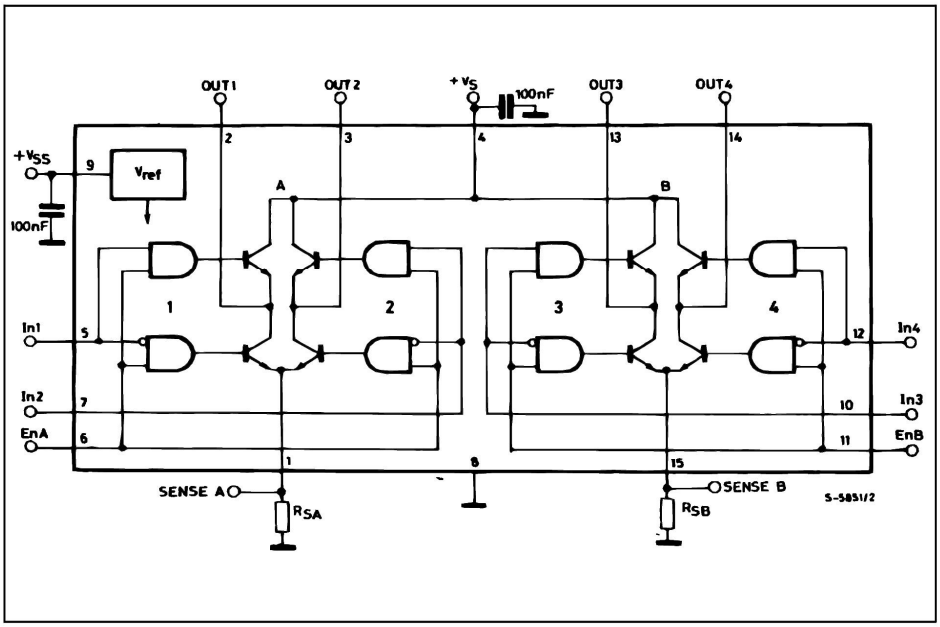
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

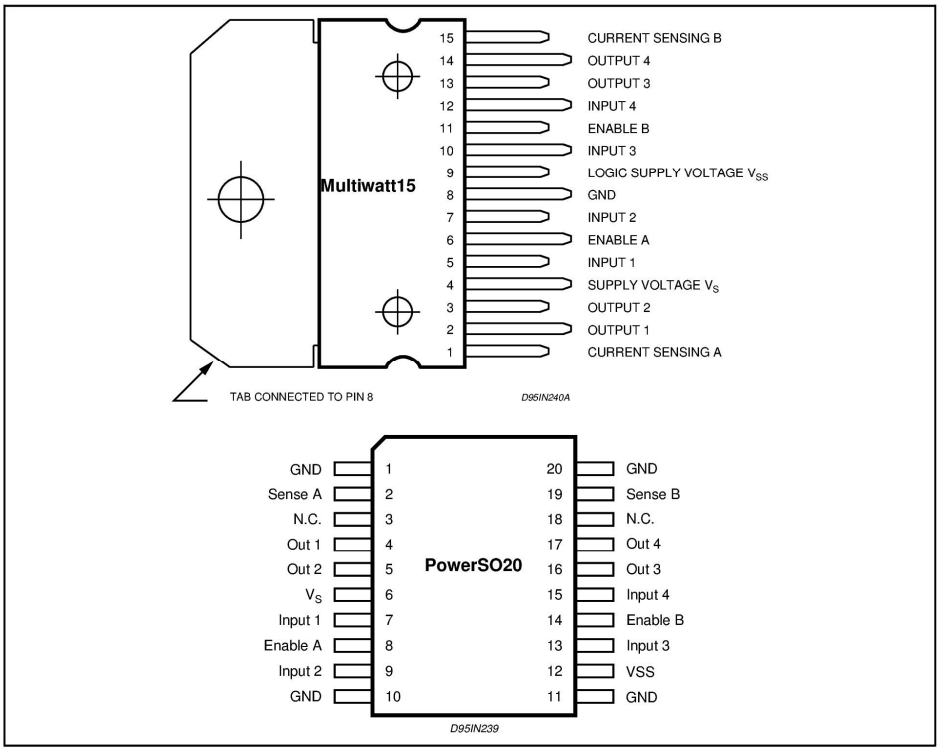
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

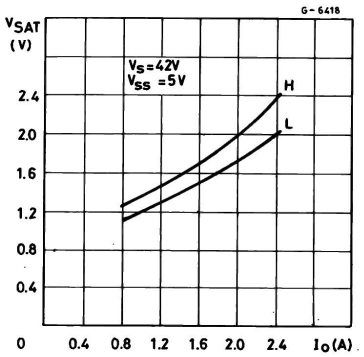
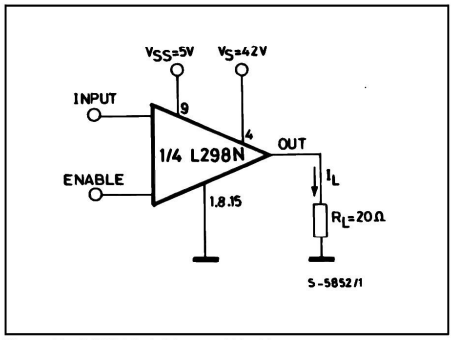


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

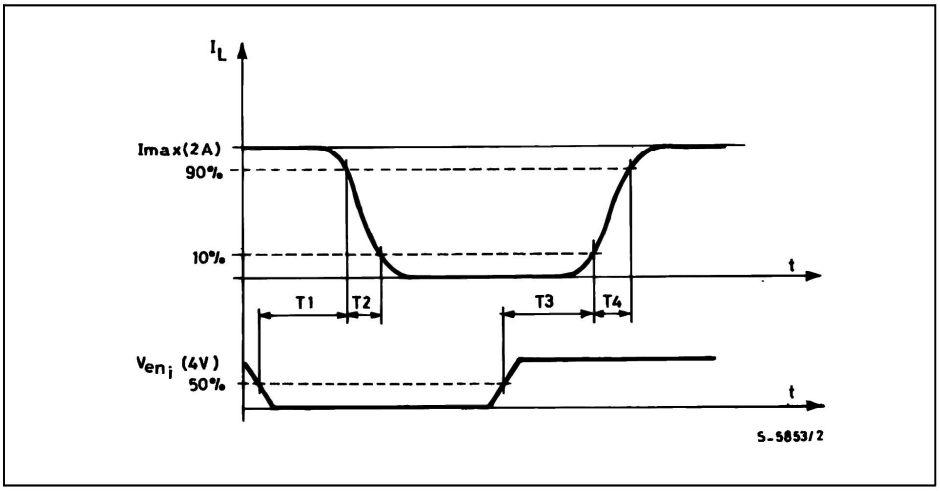
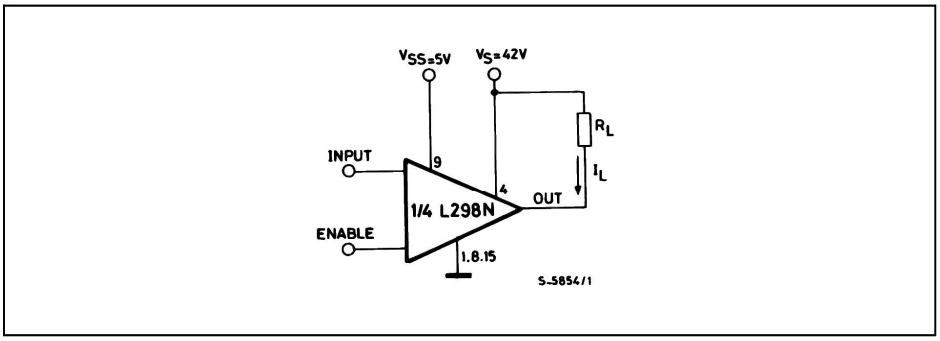


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

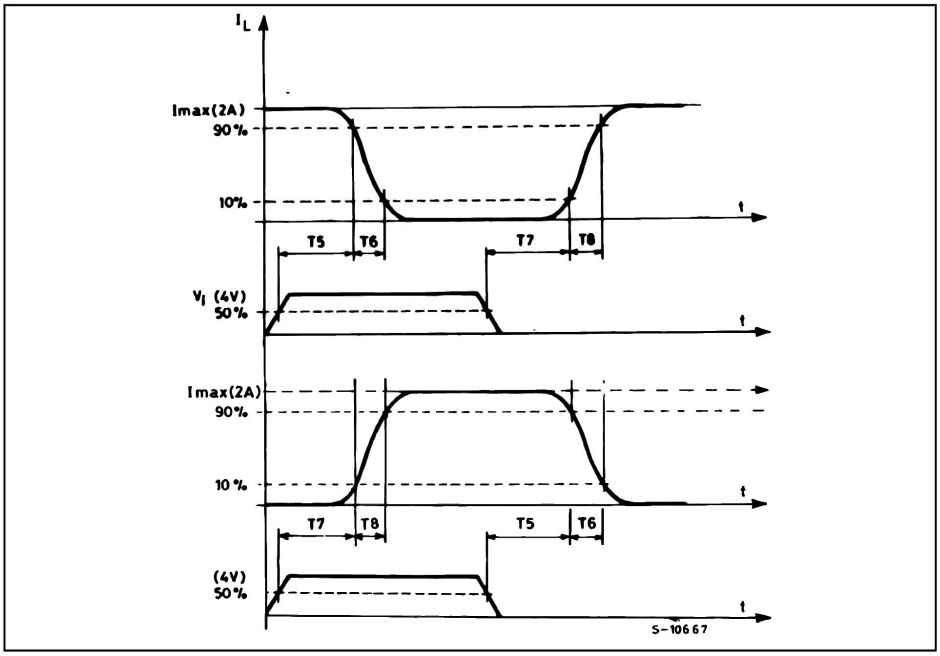
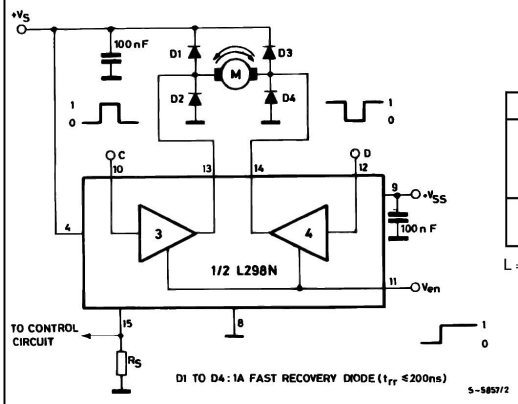


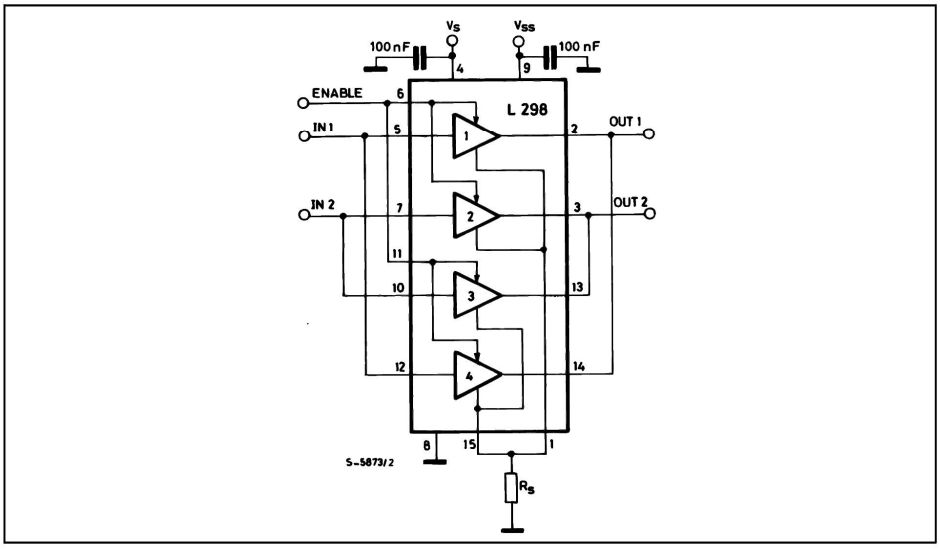
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements (trr ≤ 200 nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

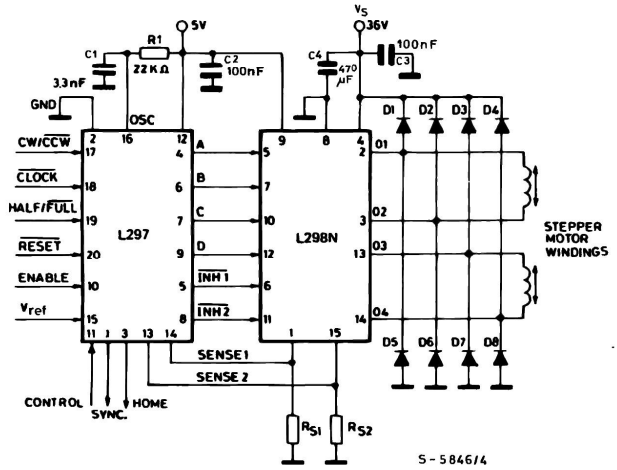
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A; trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

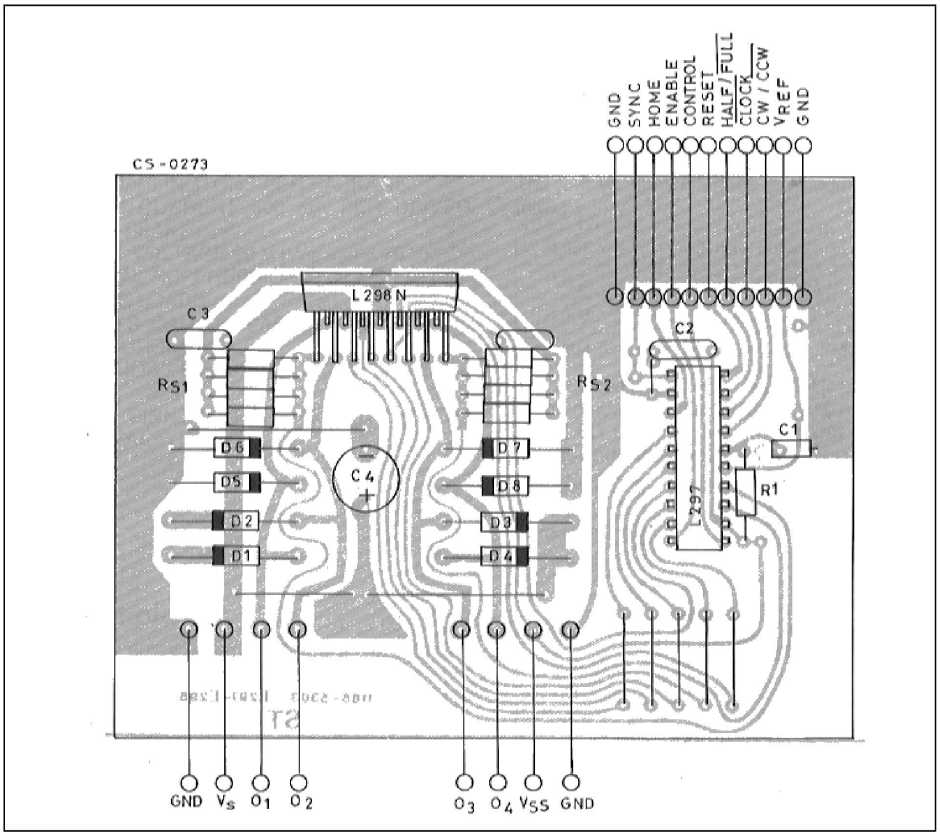
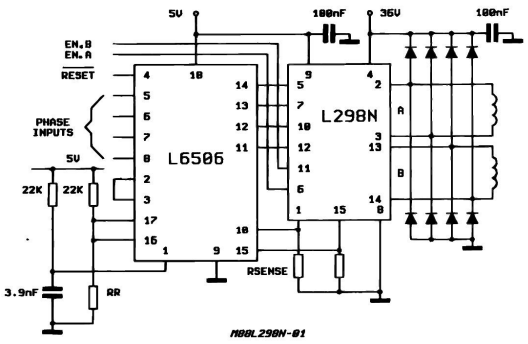


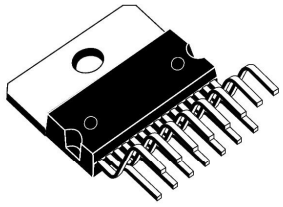
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



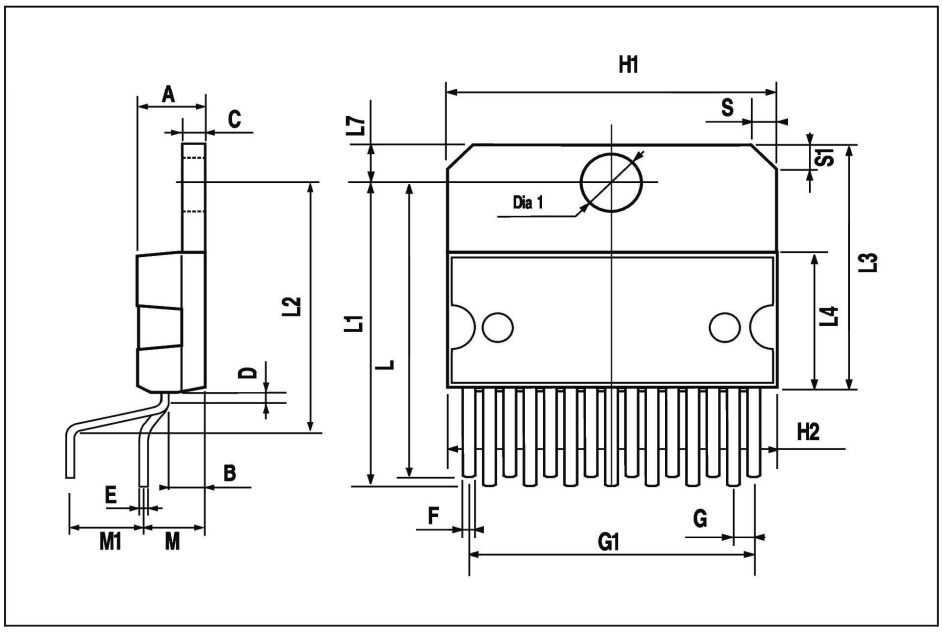
R_R and R_{sense} depend from the load current

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

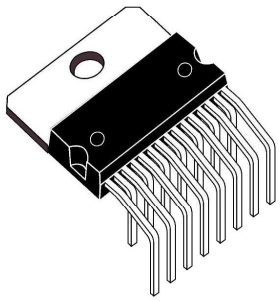


Multiwatt15 V

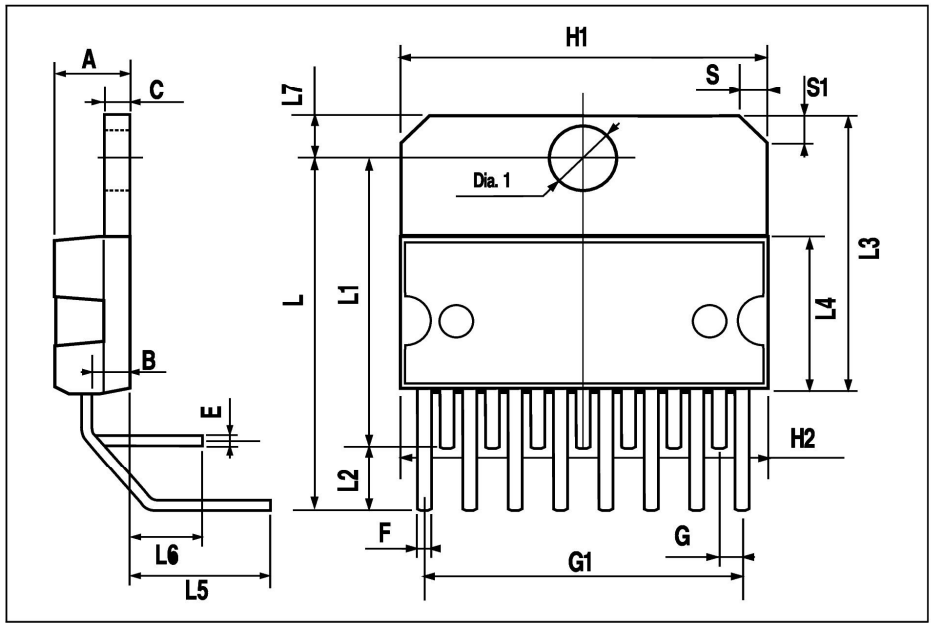


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



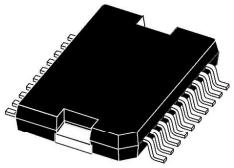
Multiwatt15 H



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10° (max.)					
S	8° (max.)					
T		10			0.394	

(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20