Filipe Donghia Badaró
Tomás Serignolli D'Agostino

Controle de Rotação de um Motor de Combustão Interna

Texto apresentado à Universidade de
São Paulo para a graduação no curso
de Engenharia Mecatrônica

Área de Concentração:
Engenharia Mecâtrônica

Orientador:
Ettore Apolônio de Barros

São Paulo
2002

Resumo

Este trabalho teve como objetivos a seleção, adaptação, modelagem matemática, e aplicação de uma lei de controle de rotação a um MCI. Também foi assunto deste trabalho a interligação do MCI com um computador e o desenvolvimento do software necessário para operar este MCI remotamente através deste mesmo computador.

Primeiramente foi selecionado o motor a ser utilizado, e neste foram realizadas adaptações para a inclusão de atuadores e sensores. Foi construída uma interface eletrônica para interligar este sistema a um PC, via porta serial, de modo a permitir o controle remoto do sistema. Também foi desenvolvido o software necessário para esta interação.

A modelagem do sistema foi baseada em ensaios realizados no motor, e com base neste modelo foi projetado um controlador PI para reduzir o tempo de resposta do sistema.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

MCI	Motor de Combustão Interna
LQG	Controle Linear Quadrático e Gaussiano
GPC	Controle Preditivo Generalizado
P	Controle Proporcional
PD	Controle Proporcional e Derivativo
PI	Controle Proporcional e Integrativo
PID	Controle Proporcional, Integrativo e Derivativo
CI	Circuito Integrado
UART	Universal Asynchronous Receiver-Transmitter
TTL	Transistor-Transistor Logic

1. Introdução

Este projeto é parte de um projeto maior que visa a construção de um barco rápido de transporte de passageiros, para tanto, dois modelos de cascos foram produzidos. Um menor com dois metros de comprimento e um maior de quatro metros. Para o primeiro, uma motorização elétrica convencional foi aplicada com sucesso, enquanto que para o último, o cálculo da potência necessária para impulsioná-lo previa uma quantidade de baterias que ia além da capacidade de transporte do barco, tornando inviável a sua implantação. Surge dai a necessidade de procurar uma motorização alternativa que forneça uma melhor relação peso/potência para impulsionar o barco.

Os critérios de seleção, a economia de peso e espaço, conduzem à escolha de um motor de combustão interna (MCI), pois o cálculo da potência exigida para impulsionar o modelo, demonstra a necessidade de uma grande potência e, conclui-se, que a melhor alternativa será a utilização de um (MCI) devido à sua alta relação peso/potência.

Apesar de atender aos requisitos de economia de peso e espaço, o MCI apresenta grande complexidade na modelagem da dinâmica do seu funcionamento. Os textos que abordam a modelagem de um MCI com o intuito de controle são poucos e de difícil compreensão, pois abordam uma grande quantidade de assuntos relacionados ao modelamento. Como foge do escopo deste trabalho criar uma nova metodologia de modelagem de motores, o modelo será baseado em outros apresentados pela literatura [1, 13 e 8, 9, 10] contando com as modificações necessárias para a aplicação no motor em questão.

O assunto abordado por este trabalho de formatura compreende a seleção, a modelagem de um MCI, fornecidos pela literatura, e a aplicação de uma lei de controle da rotação neste mesmo motor. É também assunto deste trabalho a adaptação de um motor convencional, carburado, de forma que ele possa ser operado sem a intervenção humana, no que diz respeito ao controle da rotação. Outro ponto é a validação experimental do motor, teste de bancada, para efeito de comparação com os resultados obtidos pela modelagem teórico da dinâmica do motor.

Por fim, serão apresentados, também, todos os mecanismos de controle do MCI, como sensores, atuadores e um computador industrial responsável pelo comando automático da rotação do motor.

2. Histórico

2.1. Panorama Geral

Logo após o surgimento do Motor de Combustão Interna (MCI), em meados do século XIX, uma das principais preocupações de seus estudiosos era a obtenção de uma mistura de fluidos que propiciasse uma combustão completa e regular. Esta linha de pesquisa ocupou todo o final do século XIX e o início do XX, e contou com a participação de diversos pioneiros do estudo de MCIs, dentre os quais destacamos Nicholas Otto, Rudolph Diesel e Daimler.

Com o início da década de 50, surgem dos primeiros trabalhos estudando a aplicação de técnicas de controle ótimo para se maximizar o torque disponível no eixo. Destaca-se nesta época o trabalho de Draper & Li (DRAPER & LI, 1951, apud STIVENDER, 1978), que trouxe diversos resultados valiosos, dentre eles a indicação do inter-relacionamento existente entre o controle de ignição e o sistema de dosagem da mistura.

Esta linha de pesquisa se consolidou durante as décadas de 60 e 70, se tornando uma das formas mais eficientes de se alcançar os requisitos desejados de desempenho, como dirigibilidade, partida a frio, maximização do torque, minimização do consumo de combustível e minimização da emissão de gases poluentes. De fato, estes estudos levaram a um dos conceitos atuais de controle de MCIs, o conceito de controle total do motor, aonde por meio de eletrônica embarcada pode-se ajustar continuamente ajustar o ponto de operação do motor.

Este sistema de gerenciamento eletrônico solucionou o problema do controle de desempenho para um MCI operando em regime permanente, mas ainda podiam ser notados desvios durante os regimes transitórios. E é este problema que os estudos mais recentes tentam solucionar, com a aplicação de técnicas de controle tais como o controle LQG (Linear Quadrático e Gaussiano), a Modelagem Média com uso de Observadores de Estado, o GPC (Controle Preditivo Generalizado), e mais recentemente as Redes Neuronais e o Controle Adaptativo Não-Linear, com Estimador de parâmetros em tempo real.

2.2. Histórico de Modelagem

As primeiras tentativas de modelagem de um MCI foram realizadas durante os anos 60, como em YU (1962) apud WU (1980) e SCHWEITZER (1966) apud KIENCKE (1988), produzindo uma grande diversidade de modelos, cada um focado em um fenômeno diferente dentre os muitos que envolvem o funcionamento de um MCI.

Estes primeiros modelos, associados a observações e medições experimentais, tiveram grande importância na compreensão dos fenômenos químicos, fluidos e termodinâmicos que estão presentes em um MCI, tais como a indução da mistura, homogeneização, combustão, formação e exaustão dos gases de escape. Um trabalho característico desta filosofia pode ser visto em BENSON et al. (1975), ou em livros texto recentes sobre motores, tais como HEYWOOD (1988). No entanto, esta linha de modelagem não era apropriada à aplicação de técnicas de controle, devido à presença de equações matemáticas complexas, que sugeriam o uso de métodos numéricos de solução.

Os primeiros modelos objetivando a aplicação das leis de controle foram desenvolvidos durante a década de 70. A abordagem adotada por esses primeiros trabalhos visava caracterizar o MCI operando em regime permanente, através de modelos "caixa-preta", nos quais determinavam-se empiricamente os coeficientes de uma curva de operação a partir de entradas e saídas conhecidas. Uma revisão destes modelos pode ser encontrada em POWELL (1987).

Esta abordagem possuía dois grandes problemas: o número de entradas e saídas envolvidas e grande tempo gasto em bancada. Estes problemas se tornaram particularmente críticos a partir da 70, aonde se começa a dar maior preocupação os regimes transitórios, principalmente devido à necessidade de maior controle das emissões, decorrente de mudanças em regulamentações ambientais.

Por estes motivos, inicia-se neste momento uma linha de modelagem baseada na descrição dos fenômenos físicos presentes em um MCI (modelos fenomenológicos), abordagem presente até os dias de hoje. É interessante mencionar que mesmo estes modelos ainda possuem um grau de empirismo, utilizando-se de dados empíricos obtidos em laboratório.

2.3. Histórico de Controle

Paralelamente ao desenvolvimento dos modelos para os motores, foram sendo aplicadas diversas técnicas de controle aos MCI, sendo sua evolução geralmente determinada pelos requisitos de processamento computacional requeridos e por melhorias tecnológicas. Em geral, consideram-se três fases para a aplicação de controle em motores:

- Primeira Geração: Aplicações de controle clássico (PI, PID);
- Segunda Geração: Controle moderno, com formulação em espaço de estado, e uso de algoritmos avançados de controle (controle preditivo, controle não-linear, redes neuronais);
- Terceira Geração: Aplicação das técnicas da segunda geração, com a adição de sensores especializados. Ainda tem aplicação bastante restrita, geralmente em laboratórios.

3. Descrição dos Equipamentos

3.1. Motor de Combustão Interna

3.1.1. Seleção do Motor

Para a seleção do motor mais adequado ao projeto do casco foram efetuados os seguintes cálculos:

Admitindo o número de Froude com sendo igual a 1,32, pela tabela abaixo, obtida pelo ensaio de rebocamento em um tanque de provas, temos:

$EHP = 3{,}83KW$, onde EHP é a potencia requerida para rebocar o modelo a uma determinada velocidade. Este valor é obtido da tabela abaixo (transcrita somente com os valores de interesse de projeto):

Fn	EHP (Casco)	EHP (Apêndices)	EHP (Ar)	EHP (Total)
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1,32	3,15 (KW)	6,75e-1 (KW)	4,20e-3 (KW)	3,83 (KW)
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Tabela 3.1 – Valores para EHP

$V = Fn\sqrt{gL}$, onde V é a velocidade do modelo corrigida, Fn é o número de froude, g é a aceleração da gravidade e L é o comprimento do casco.

$$V = 1{,}32\sqrt{9{,}81 \times 4}$$

$$V = 8{,}27m/s$$

$EHP = R_T V$, onde R_T é a resistência ao movimento à qual o modelo está sujeito ao ser movimentado a velocidade V, corrigida pelo número de Froude.

$$R_T = \frac{3{,}83e3}{8{,}27}$$

$$R_T = 4{,}63e2N$$

$T = \frac{R_T}{(1-t)}$, onde T é a força exigida para empurrar o modelo a velocidade desejada e t é o fator de redução de empuxo estimado como 0,2.

$$T = \frac{4{,}63e2}{(1-0{,}2)}$$

$$T = 578{,}99N$$

Agora admitindo que o casco possua dois propulsores devemos dividir T por dois, então:

$$T_{eixo} = 289{,}49N$$

$K_{Treq} = \frac{T_{eixo}}{\rho D^2 V^2 (1-\omega)^2} J^2$, onde K_{Treq} é o coeficiente de empuxo requerido do propulsor, ρ é a densidade do liquido no qual o modelo esta flutuando, D é o diâmetro do hélice e ω é um fator de perdas.

$$K_{Treq} = \frac{289{,}49}{1000 \times 0{,}15^2 \times 8{,}27^2 \times 1} J^2$$

$$K_{Treq} = 0{,}19J^2$$

Com esta ultima equação pode-se plotar os valores de K_{Treq} no gráfico de ensaio de água aberta do propulsor e verifica-se o ponto de encontro com K_T, onde K_T é o coeficiente de empuxo fornecido pelo hélice. Do ponto de encontro em K_T estima-se o valor de K_Q e J e fazemos:

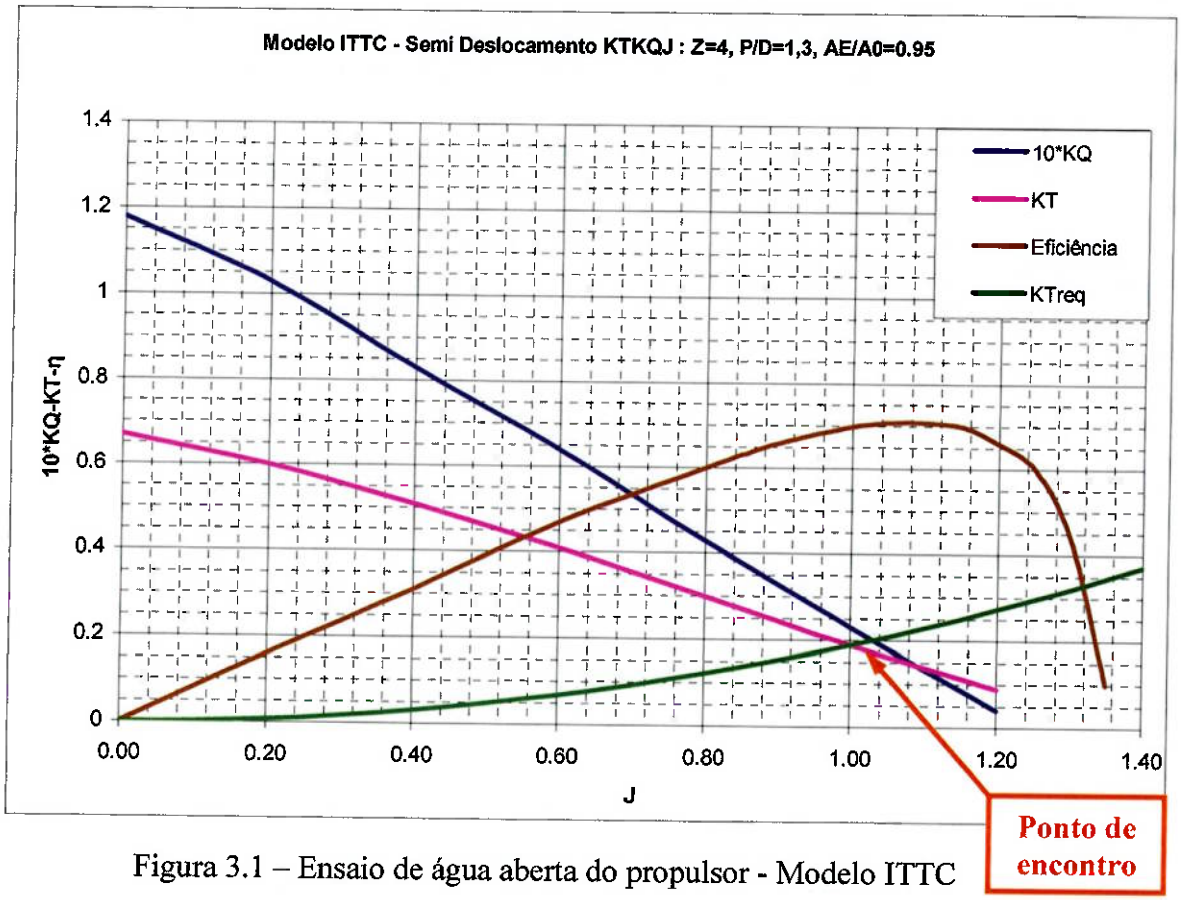


Figura 3.1 – Ensaio de água aberta do propulsor - Modelo ITTC

$K_{Q_0} = \frac{Q}{\rho n^2 D^5}$, onde Q é o torque exigido pelo motor para cada eixo e n é a rotação do eixo.

$$J = \frac{V}{nD}$$

$$n = \frac{8{,}27}{1 \times 0{,}15}$$

$$n = 55{,}1 rps$$

$$Q = 0{,}044 \times 1000 \times 55{,}1^2 \times 0{,}15^5$$

$$Q = 10{,}14 Nm$$

Admitindo 30% de margem de segurança e multiplicando por dois que é o número de eixos para obter o valor total temos:

$Q = 26{,}37 Nm$ a $3306 RPM$

Para selecionar um motor adequado basta comparar estes valores com os dados fornecidos pelos fabricantes de motores.

O projeto não impõe nenhuma restrição quanto à forma geradora de energia do motor podendo ele ser elétrico ou de combustão interna. Entretanto, outros fatores, como economia de peso e espaço, são fundamentais. Portanto, um motor de combustão interna se mostra muito mais vantajoso, pois consegue unir as características de alta potência e dimensões reduzidas. Um conjunto motor elétrico e suas baterias, capaz de gerar o torque necessário calculado anteriormente seria um equipamento demasiado grande e pesado por isso a sua utilização foi descartada.

Por fim, apesar da maior dificuldade de modelagem e controle, optou-se por um motor a combustão interna para a implementação do projeto devido as sua relação peso potência elevada. Neste projeto foi escolhido um motor da marca Honda, modelo GX 390, que cumpre com as especificações exigidas.

3.1.2. Características do Motor

O motor a ser selecionado para o projeto deve cumprir com as exigências de peso e potência requeridos pelo projeto. Como não havia um valor específico para o limite do peso do motor, apenas desejava-se economizar o máximo em peso, optou-se por um motor a combustão interna. O modelo escolhido está descrito abaixo pelas suas curvas características e pelos desenhos.

Variante	QAE2
Modelo	K1
Alerta de Óleo	•
Eixo Horizontal	•
Diâmetro x Curso	3 31/64 x 1 dia. tapped 3/8 24 UNF
Rotação máxima sem carga	3900
Ignição por Correia	•
Ignição Elétrica 12V	•
Ignição Transistorizada	•
Filtro de Ar	DE
Tanque de combustível (L)	6,5
Peso (Kg)	31

Tabela 3.2 – Características do MCI

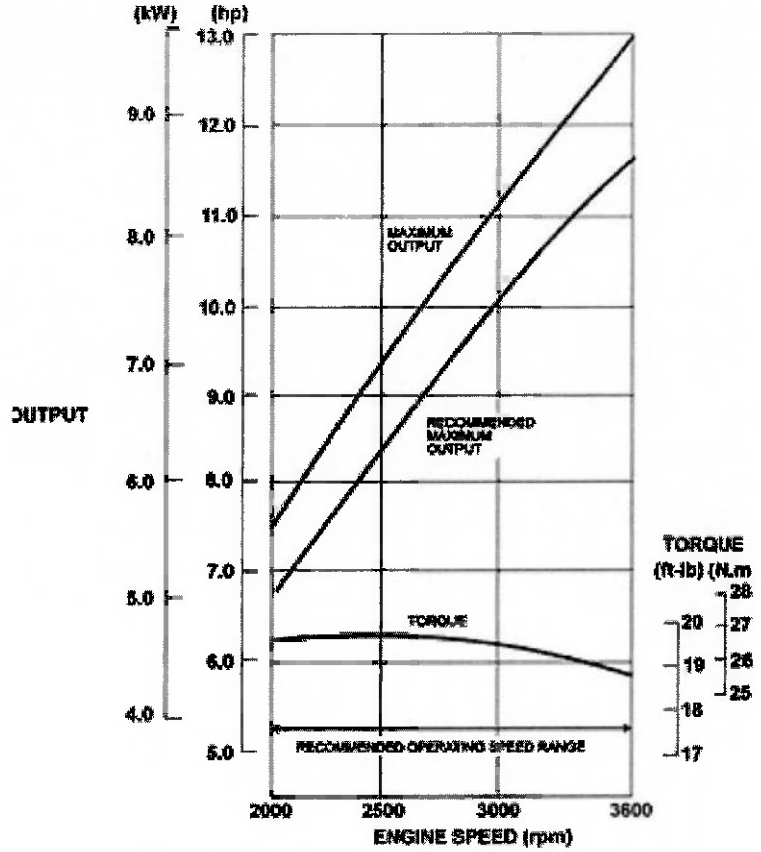


Figura 3.2 – Curva característica do MCI

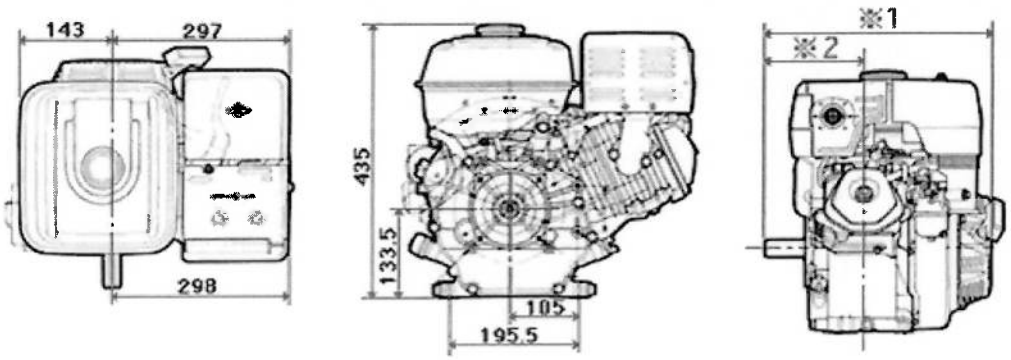


Figura 3.3 – Dimensões do MCI

3.1.3. Adaptações no Motor

O intuito do projeto é controlar a rotação do motor, durante as manobras do modelo, rejeitando as perturbações do meio ambiente. Como foi empregado um motor comercial no projeto, suas características originais não permitiam o controle adequado da rotação. Portanto, algumas modificações foram necessárias para que ele

pudesse ser empregado no projeto. Elas serão descritas em itens detalhados e ilustradas por fotografias, são elas.

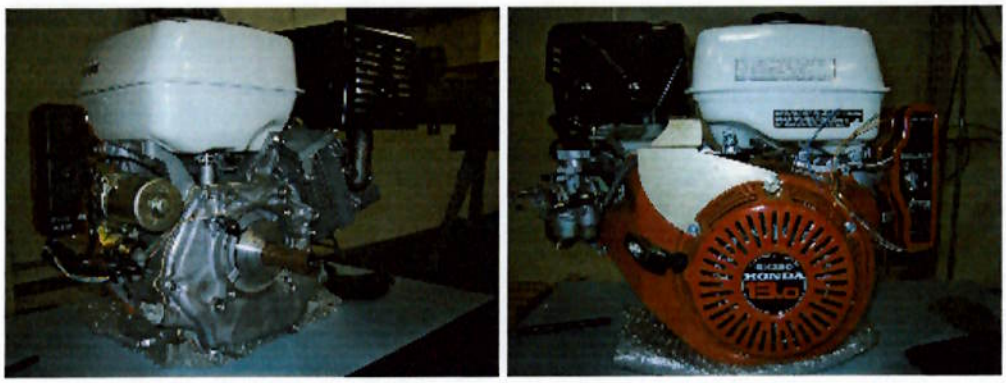


Figura 3.4 – Adaptações no motor

3.1.3.1. Carburador

O controle da velocidade se dá pela regulagem da passagem de ar pelo carburador do motor. Este, por sua vez, é controlado pela abertura, ou fechamento, da válvula borboleta, que é a responsável pelo fluxo de ar através do carburador. É interessante do ponto de vista do projeto que esta regulagem da válvula borboleta seja feita remotamente por um motor, que deverá acionar a válvula e, desta forma, promover a sua abertura, ou fechamento. Para que seja possível o controle da abertura da válvula borboleta por meio de um motor, no nosso caso um motor de passos, o eixo da válvula teve de ser modificado, seu sistema original de controle por atuadores manuais foram removidos e prolongou-se o eixo para que uma polia dentada fosse adaptada a ele. Desta forma, o motor aciona a válvula através de uma correia que liga o motor ao eixo da válvula.

Figura 3.5 – Modificações no carburador

3.1.3.2. Bloco do Motor

Outra característica importante do projeto é a incorporação dos atuadores e sensores no motor, ou seja, o motor de passos, responsável pelo controle da válvula borboleta, deve ser corretamente afixado no motor de forma que ele possa realizar suas funções com precisão. E, também, o sensor que vai determinar a velocidade do motor deve ser afixado no motor. Assim, o bloco do motor sofreu algumas modificações, como a inclusão de um suporte para o motor e a instalação de um parafuso de suporte para o sensor de velocidade.

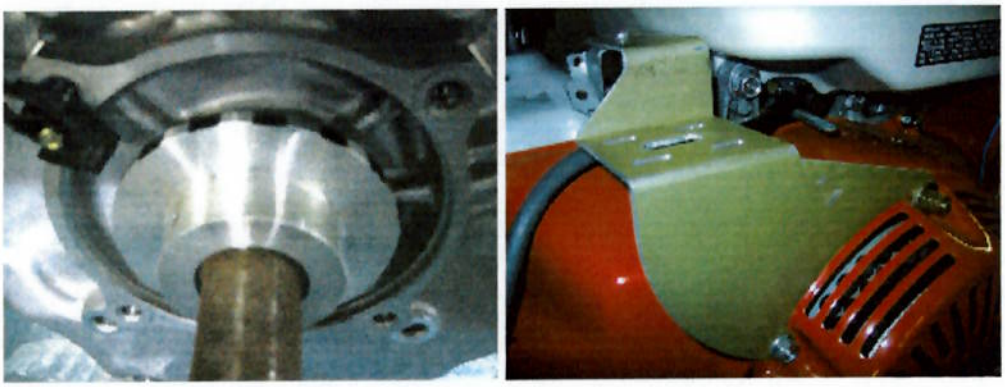


Figura 3.6 – Modificações no bloco do motor

3.1.3.3. Eixo de Saída

O sensor de velocidade, sozinho, não consegue identificar a rotação do motor. Ele só consegue diferenciar uma região reflexiva de uma não reflexiva. Para que a velocidade do motor seja calculada, o eixo de saída recebeu um volante dividido em 32 partes, pintadas de preto alternadamente, que será utilizado pelo sensor de velocidade para perceber a rotação do motor e, assim, gerar os pulsos para que a velocidade seja calculada posteriormente.

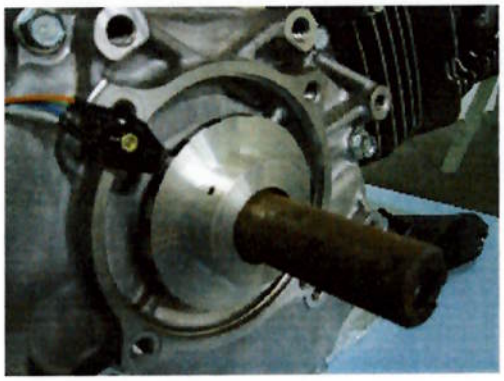


Figura 3.7 – Posicionamento do sensor de velocidade

3.2. Sensores

Neste projeto, a parte de sensoriamento será bastante simples, pois apenas um único sensor será utilizado, o de rotação. Ele e composto por um foto transistor e um emissor de infravermelho acoplado. Ele é capaz de perceber a reflexão do feixe de infravermelho numa peça polida, para cada vez que a parte polida passa em frente do

sensor ele emite um pulso. Desta forma, para cada volta, é possível atribuir um número de pulsos correspondentes e assim calcular a rotação do motor em tempo real. Ele será então instalado perto do eixo de saída do motor de forma que fique corretamente instalado. O sensor em questão pode ser visualizado na figura a seguir.

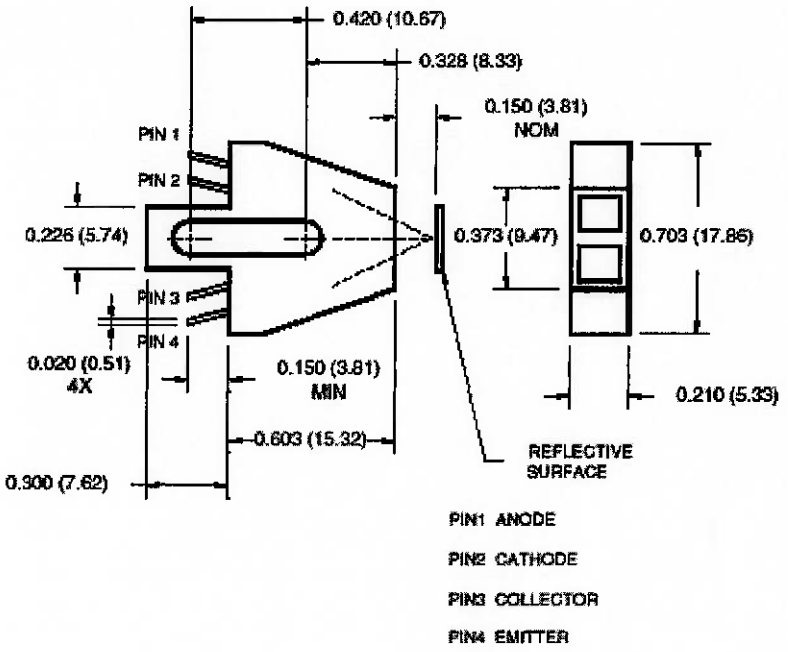


Figura 3.8 – Dimensões do sensor de velocidade

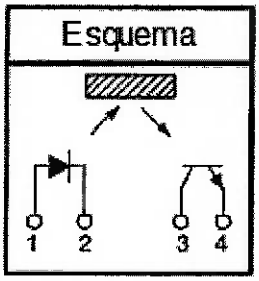


Figura 3.9 – Princípio de funcionamento do sensor de velocidade

A posição da borboleta do carburador será definida pela contagem de pulsos de um motor de passo, assim faz-se desnecessária a instalação de um sensor de posição na borboleta do carburador. Para tanto é necessário admitir que não há perda de passos no movimento do motor. Esta hipótese é bastante razoável, pois quase não há esforços para o deslocamento angular da válvula borboleta. É importante lembramos de realizar um fechamento completo da borboleta durante os

procedimentos de inicialização do sistema, passa possibilitar que abertura possa seja avaliada corretamente.

3.2.1. Circuito Comparador de Voltagem

Nas primeiras tentativas de se ligar o sensor ao circuito projetado, notamos que a resposta lida no circuito era incoerente com a medição esperada do sensor. Verificamos que isto ocorria devido ao grande consumo de corrente no CI TLL ao qual o sensor era ligado diretamente, o que provocava uma queda de tensão, "camuflando" a voltagem lida pelo sensor.

Para corrigir esta situação foi adicionado à saída do sensor de rotação um circuito comparador de voltagem baseado em um amp-op (LM339), conforme a figura a seguir:

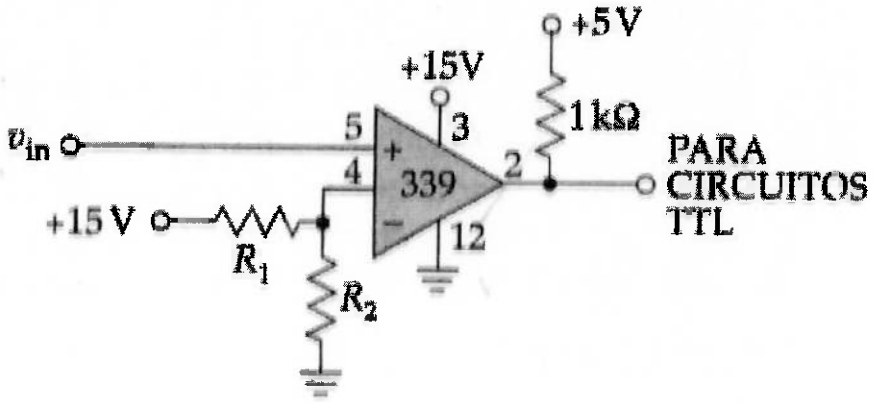


Figura 3.10 – Circuito comparador de voltagem

3.3. Atuadores

O sistema responsável pela atuação do controle sobre o sistema é composto de dois equipamentos: um motor de passo, ligado diretamente sobre a borboleta do MCI, que é o atuador propriamente dito; e o driver, responsável pela interface entre o motor de passo e o controlador. Abaixo serão descritos ambos os componentes em maior detalhe.

3.3.1. Motor de Passo

O motor de passo a ser utilizado é um modelo HT17-068, fabricado pela Applied Motion Products, Inc. A tabela abaixo, fornecida pelo fabricante, informa suas principais características operacionais:

Part #	HT17-068
Motor Lenght (inches)	1.3
Holding Torque (oz-in)	22.2
Leads	8
Step Angle	1.8
Volts	4.0
Amps	.95
Ohms	4.2
mH	2.8
Rotor Inertia (oz- in²/G-CM²)	.190/35.0
Motor Weight (Lbs.)	.44

Tabela 3.3 – Características do motor de passo

A figura abaixo mostra as principais dimensões do motor, e suas características de fixação.

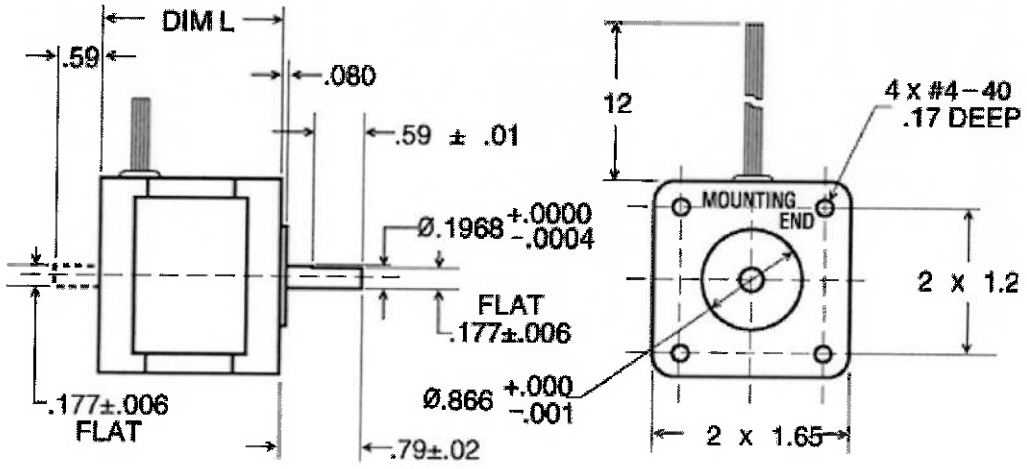


Figura 3.11 – Dimensões do motor de passo

3.3.2. Driver

O driver a ser utilizado em conjunto com o motor de passo acima é um modelo 3540m, também fabricado pela Applied Motion Products, Inc. Este driver possui as seguintes características operacionais:

- Resoluções para Micropasso: 1/2 passo (400 passos/volta), 1/5 passo (1,000 passos/volta), 1/10 passo (2,000 passos/volta), 1/64 passo (12,000 passos/volta);
- Dois Amplificadores tipo MOSFET
- Proteção Térmica
- Entradas Isoladas Oticamente
- Dimensões: 1.5 x 3.0 x 4.0 polegadas

Abaixo mostramos o diagrama de blocos deste driver:

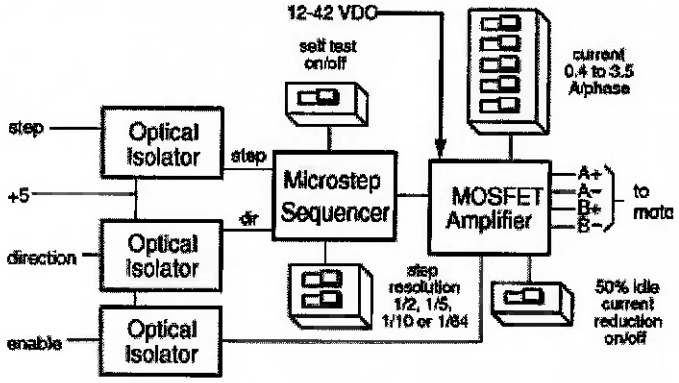


Figura 3.12 – Diagrama de blocos do driver

3.4. Sistema de Controle

O sistema de controle do MCI será implementado via um software desenvolvido nas linguagens C e C++. O software será executado em um computador embarcado que segue o padrão PC-104, utilizando como sistema operacional QNX. Ambos serão descritos abaixo em mais detalhe.

3.4.1. PC-104

A sigla PC-104 refere-se a um padrão desenvolvido por um consórcio de cerca de 100 empresas para computadores embarcados baseados no padrão Intel X86. As principais características deste padrão são:

- Dimensões reduzidas (90mmx96mm);

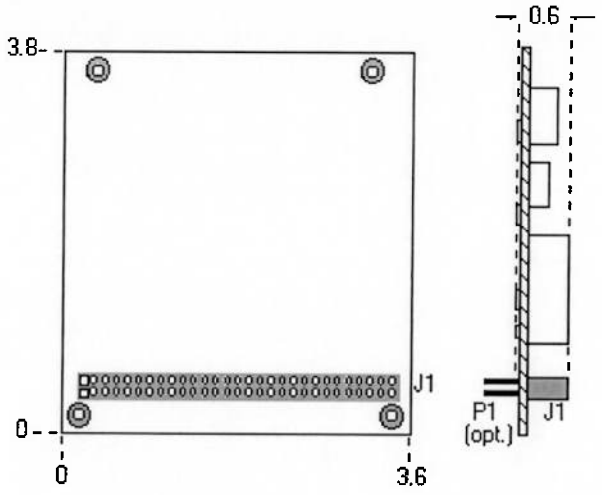


Figura 3.13 – Dimensões para um módulo PC-104

- Baixo consumo de energia (tipicamente 1-2W por módulo);
- Modularização, através de um barramento que permite o "empilhamento" de módulos, como ilustra a figura abaixo:

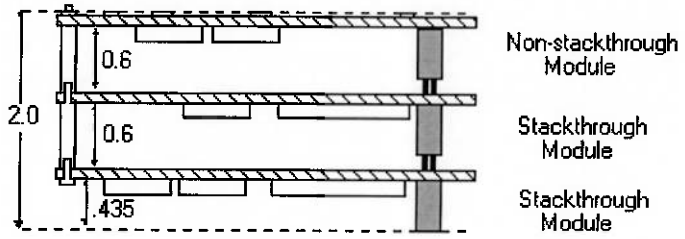


Figura 3.14 – Empilhamento de módulos PC-104

O PC-104 utilizado neste projeto é composto pelos seguintes módulos, todos fabricados pela Real Time Devices USA, Inc.:

- Módulo de CPU, com processador de 233 MHz da Cyrix;
- Módulo de Rede;
- Módulo de Hard Disk;
- Módulo de IO, com entradas e saídas digitais e analógicas;
- Módulo de Vídeo SVGA.

3.4.2. QNX RTOS

O QNX RTOS (RealTime Operating System) é um sistema operacional desenvolvido pela QNX Software Systems para uso em aplicações de tempo real. Sua principal característica é a presença de um kernel reduzido, também chamado de

Microkernel, que é responsável por somente três funções básicas do sistema: criação de processos, gerenciamento de memória e controle de timers. Todas as outras funcionalidades e subsistemas do QNX são módulos externos, "opcionais", que podem ser facilmente adicionados e removidos. Isso confere ao QNX uma grande modularidade e capacidade de customização, permitindo seu uso desde aplicações embarcadas até sistemas envolvendo múltiplos servidores. Adicionalmente, o QNX segue o padrão POSIX, sendo certificado na especificação POSIX.1 e incluindo diversas funcionalidades POSIX.1b.

O compilador a ser utilizado é o Watcom C/C++ Compiler, que acompanha a distribuição do sistema.

3.4.3. Interface de Comunicação PC-104/Sistema

3.4.3.1. Análise dos Sinais

A primeira consideração a ser realizada para o projeto de nossa interface de comunicação é a estudarmos os sinais que precisam ser transportados entre o sistema e o controlador. Para este projeto, temos:

- Um único sinal entrando no controlador, vindo do sensor;
- Três sinais saindo do controlador para o driver, "Enable", "Direction" e "Step".

Note que, neste projeto, todos os sinais envolvidos já estão na forma digital, dispensando conversões, o que simplificará bastante o projeto da interface de comunicação. A freqüência máxima dos sinais está em torno de 1600Hz, para o sinal de entrada, quanto estivermos com a rotação do eixo em aproximadamente 3000rpm.

3.4.3.2. Seleção da Porta

As portas comumente utilizadas para a aquisição de dados em um PC são a porta de comunicação serial e a porta de comunicação paralela. Ambos os tipos de porta trabalham com 8 bits, suficiente para lidar com os sinais existentes neste projeto. Consideramos inicialmente o uso de uma porta paralela, visto que com a

porta serial haveria a necessidade da conversão dos dados do protocolo RS-232 para uma forma mais utilizável pelo circuito lógico.

Infelizmente, o PC/104 utilizado para este projeto não possuía nenhuma porta paralela disponível para nosso uso, e tivemos que optar pelo uso de uma porta de comunicação serial. Para o uso desta porta, teremos que adicionar um circuito conversor RS-232/paralelo, construído com base em uma classe de CIs conhecida como UARTs (Universal Asynchronous Receiver-Transmitter).

Com isso, o circuito de interface será dividido em dois blocos, um conversor RS-232/paralelo e um circuito lógico para o tratamento dos dados. Note que os circuitos serão projetados já considerando que as entradas e saídas estão adequadas ao padrão TTL.

3.4.3.3. Circuito Conversor Serial/Paralelo

Para conectarmos uma porta serial a um conjunto de 8 bits é necessário um circuito de interface baseado em um UART. Este circuito é bastante conhecido, podendo ser facilmente encontrado em bibliografia especializada. Apresentamos abaixo um modelo simplificado deste:

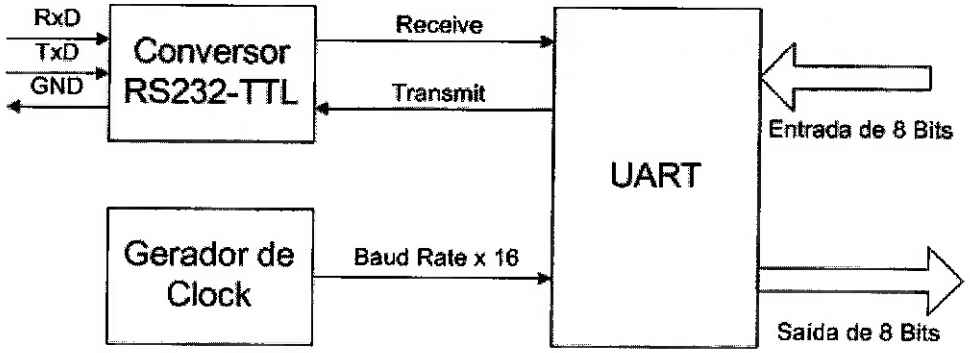


Figura 3.15 – Modelo simplificado do conversor serial/paralelo

O UART é o componente principal do circuito, responsável pela conversão entre o protocolo RS-232 e as estradas e saídas paralelas. Além disso, são necessários um conversor, para adequar as voltagens de saída do PC para o padrão TTL, e um sinal de clock para o controle da transmissão de dados.

Para este projeto, utilizaremos o UART 6402, da Intersil, um conversor MAX232 da Texas Instruments e um circuito gerador de clock baseado no timer

NE555, também da Texas. Abaixo são descritos com mais detalhes estes componentes:

- UART 6402: Este modelo de UART possui uma característica bastante distinta de outros CIs desta mesma família, e que facilita sua aplicação neste tipo de circuito: Possui vias separadas para a entrada e saída de dados, dispensando uma lógica externa de controle para a multiplexação. A ilustração abaixo mostra o diagrama funcional deste CI:

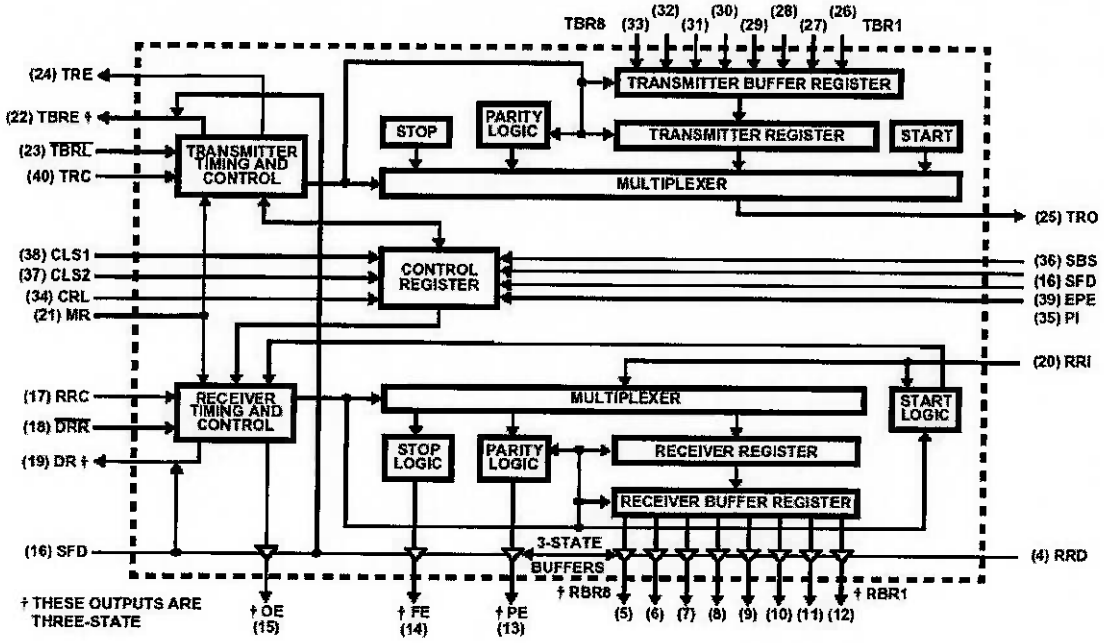


Figura 3.16 – Diagrama funcional do UART 6402

Alguns pontos de interesse neste diagrama são:

- Pinos 5-12 e 26-33: são, respectivamente, as saídas e entradas do lado paralelo do circuito;
- Pinos 25 e 20: representam a saída e entrada no extremo serial do circuito;
- Pino 23: controla a carga de dados no buffer de transmissão e o envio destes para a serial;
- Pinos 35-39: controlam o formato da transmissão serial (7 ou 8 bits, número de bits de parada, presença de bit de paridade);
- Pinos 17 e 40: o clock aplicado a estes pinos define a velocidade (baud rate) da transmissão dos dados.

Para este projeto, vamos trabalhar com um baud rate de 9600bps, 8 bits de dados, 1 bit de parada, sem checagem de paridade.

- Conversor MAX232: Este componente tem como função converter os níveis de voltagem utilizados pela serial do PC (tipicamente -8,5V Low e +8,5V High), para o formato TTL. De acordo com o datasheet, o MAX232 deve ser montado na seguinte configuração:

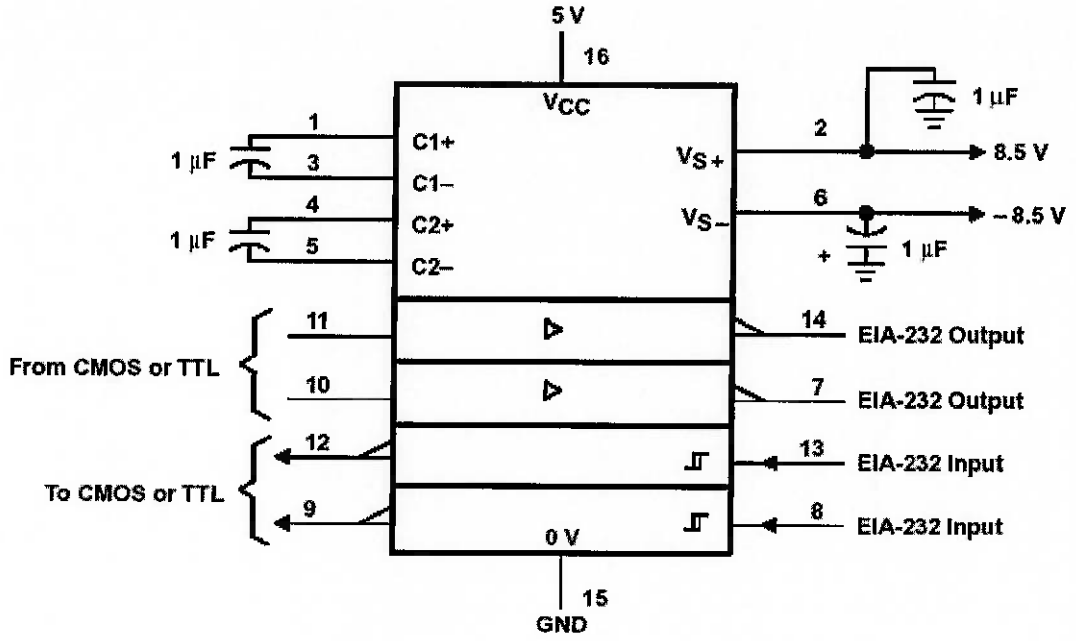


Figura 3.17 – Diagrama de montagem do MAX232

- Timer NE555: O timer NE555 é um componente eletrônico amplamente utilizado, que pode ter função de gerador de clock caso esteja em configuração astável, como mostrado na figura:

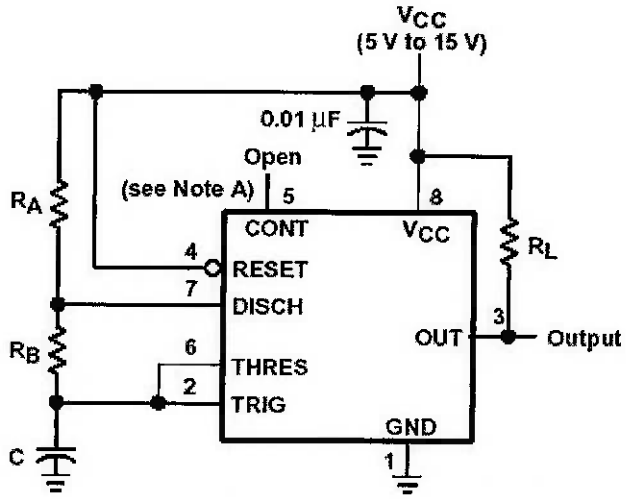


Figura 3.18 – Configuração astável do NE555

A figura abaixo mostra as formas de onda no NE555 durante sua operação:

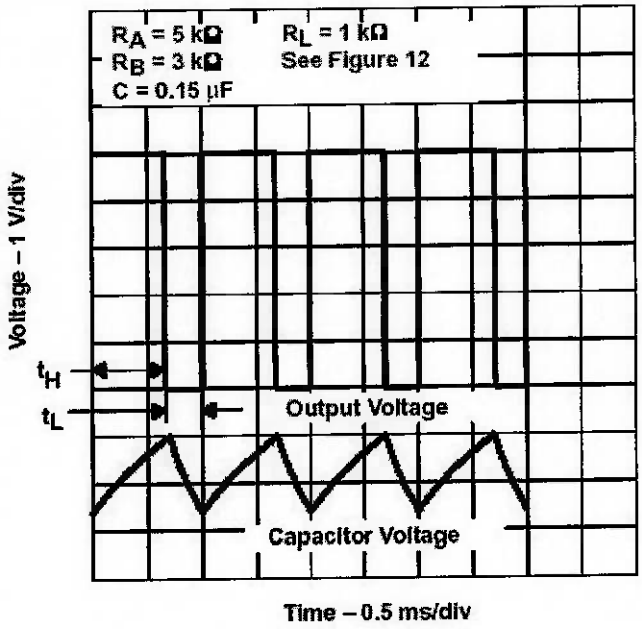


Figura 3.19 – Formas de onda no NE555

Os tempos do sinal podem ser calculados pela seguintes expressões:

$t_H = 0{,}693(R_A + R_B)C$

$t_L = 0{,}693(R_B C)$

Para o funcionamento apropriado do UART, a freqüência do sinal de clock aplicado deve ser igual a 16 vezes o baud rate desejado, no nosso caso 153600Hz, e não pode possuir erro superior a 3%. Neste projeto, serão adotados os seguintes valores para os resistores e capacitores: $R_A = 100k\Omega$, $R_B = 100k\Omega$, C=20pF. Será montado juntamente com R_A um resistor variável de $100k\Omega$, para permitir o ajuste fino da freqüência do sinal de saída.

As duas fotos a seguir mostram o protótipo desta interface que foi montado e está passando por um processo de testes e validação, e um circuito com chaves e LEDs que está sendo usado neste processo:

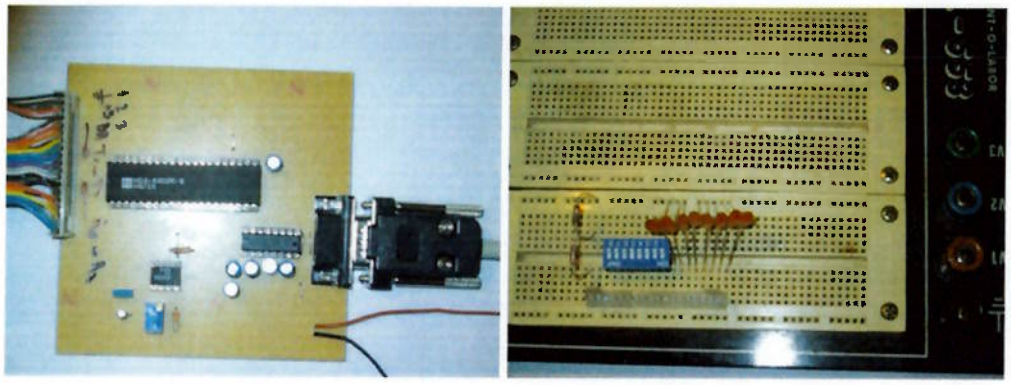


Figura 3.20 – Protótipo e circuito de testes

A figura a seguir mostra o diagrama final para este circuito de conversão:

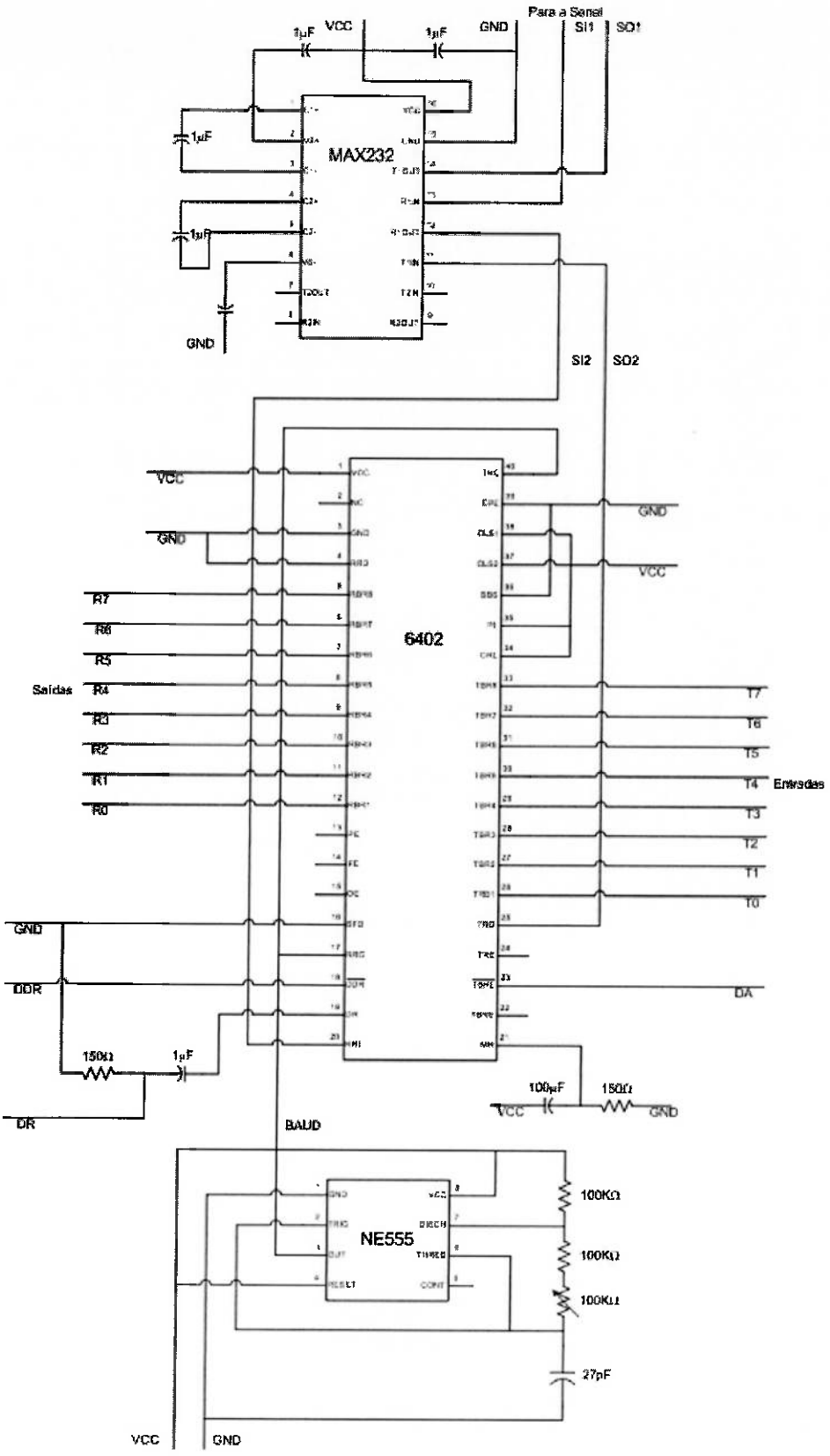


Figura 3.21 – Circuito de conversão

3.4.3.4. Circuito Lógico PC-104/Sensor

Foram desenvolvidos três modelos para esta ligação. O primeiro, mais simples, consiste na ligação direta do sinal recebido do sensor a uma das vias de entrada do PC-104:

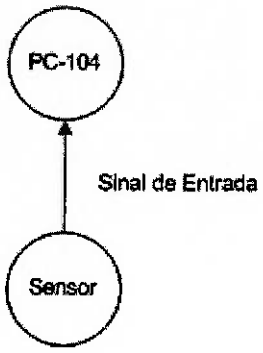


Figura 3.22 – Primeiro modelo para interface com o sensor

Apesar da grande simplicidade, este modelo é bastante viável. Sua grande vantagem é a velocidade, visto que o número de componentes (e conseqüentemente o atraso do sinal) é minimizado. Além disso, caso a porta seja configurada para trabalhar por meio de interrupções, o software será notificado toda vez que um dado é recebido, permitindo ao reduzir aos acessos à porta ao mínimo necessário.

O maior problema deste modelo é que o tratamento dos dados deverá ser realizado inteiramente por software, e um erro na entrada de dados pode ser de difícil detecção e correção.

Nosso segundo modelo é um versão simplificada da eletrônica de um encoder incremental. Utiliza-se um único contador para controlar a posição atual do eixo em relação a uma posição de referência, que nosso caso pode ser arbitrária, visto que nosso interesse é cálculo de velocidades, ou seja, precisamos apenas conhecer o deslocamento relativo do eixo. O diagrama abaixo ilustra este modelo:

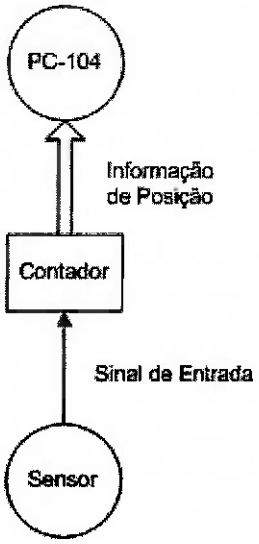


Figura 3.23 – Segundo modelo para interface com o sensor

O cálculo da velocidade pode ser feito através de duas leituras de posição, e do cálculo de tempo decorrido entre elas, que pode ser feito através dos próprios PC.

Note que este modelo é potencialmente mais impreciso que o primeiro apresentado. No primeiro caso, devido ao uso de interrupções, a marcação de posição é sempre feita na borda de subida do sinal do sensor, ou seja, na transição entre duas divisões. Já neste caso nós não temos condições de definir em que parte da divisão estará o sensor quando é feita a leitura.

Um terceiro modelo discutido baseia-se no cálculo do período por meio de hardware. Isto pode ser feito por meio de comparação com um segundo sinal de clock, de período conhecido, conforme ilustrado abaixo:

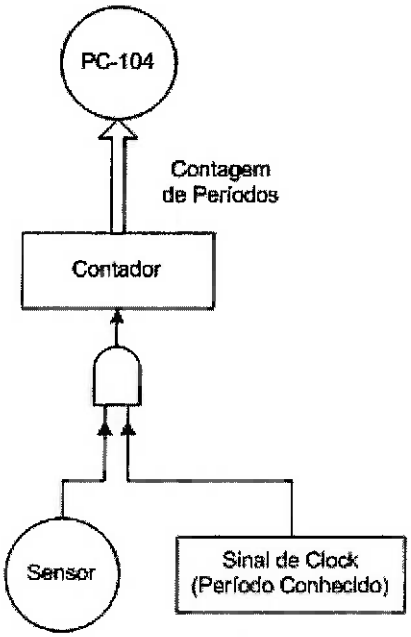


Figura 3.24 – Terceiro modelo para interface com o sensor

A contagem de pulsos armazenada no contador após um período do sinal de entrada, multiplicada pelo período do sinal de clock nos fornece o tempo total que o sinal permanece em nível alto. Um circuito análogo pode ser montado para o cálculo do tempo em nível baixo, obtendo com isso o período total do sinal.

Note que este circuito possui uma grande limitação: O tempo que o sinal permanece em um dos estados, dividido pelo período do clock, deve ser menor que a capacidade de armazenamento do contador. Ou seja, existe uma faixa de períodos que pode ser medida, limitada por baixo pelo clock e por cima pela capacidade do contador.

Esta limitação pode ser minimizada utilizando-se um contador com maior número de bits, mas devido à capacidade limitada de transmissão da porta serial (8 bits por ciclo), seria necessário multiplexar a entrada de dados, aumentando a complexidade de controle assim como o tempo necessário para a leitura dos dados.

Analisando os três modelos propostos e as condições do nosso sistema, acreditamos que o segundo modelo é mais apropriado, dada sua simplicidade de implementação e o fácil tratamento dos dados obtido.

O diagrama abaixo ilustra o circuito proposto:

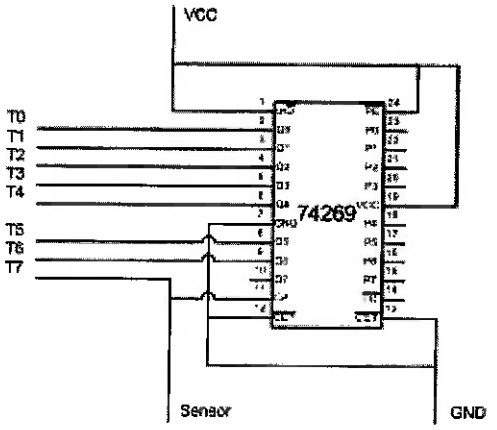


Figura 3.25 – Circuito lógico PC-104/sensor

3.4.3.5. Circuito Lógico PC-104/Driver

Na ligação do PC/104 com o driver, devem ser tomados alguns cuidados com relação aos sinais de "Enable" e "Direction". De acordo com o manual do driver,

estes dados devem ser enviados antes do sinal de "Step", e devem ser mantidos durante todo o período. Isto será implementado com o uso de elementos de memória e um sinal de controle para controlar a carga destes dados. O circuito está ilustrado abaixo.

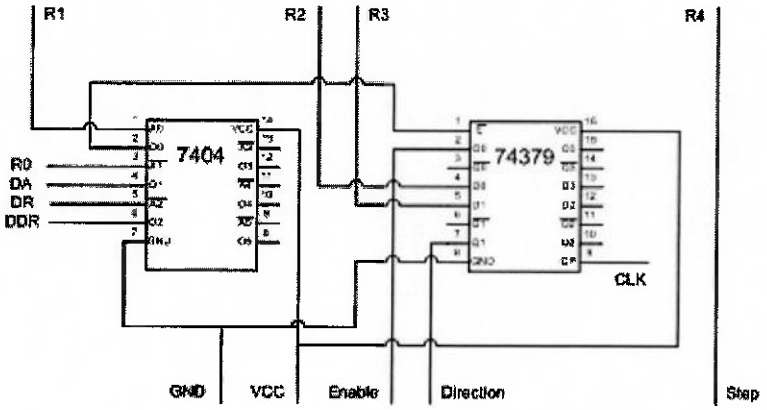


Figura 3.26 – Circuito lógico PC-104/driver

É interessante comentar que, tanto para o tratamento do sinal do sensor quanto para o controle das saídas para o driver é possível o uso de um circuito baseado em um microprocessador. Acreditamos que os modelos propostos são adequados à aplicação, porém poderemos recair nesta alternativa caso sejam encontrados obstáculos não previstos na implementação dos modelos atuais.

3.4.3.6. Tabela de Bits

A tabela abaixo resume o significado de cada bit de uma informação de entrada e saída do UART:

Bit	Função
R0	Comanda envio de dados do sensor para o PC
R1	Comanda carga dos dados de "Enable" e "Direction" no registrador
R2	Valor para "Enable" – Usar junto com R1
R3	Valor para "Direction" – Usar junto com R1
R4	Sinal de "Step"
R5	Não utilizado
R6	Não utilizado
R7	Não utilizado
T0	Posição do Eixo – Bit menos Significativo
T1	Posição do Eixo
T2	Posição do Eixo
T3	Posição do Eixo

T4	Posição do Eixo – Bit mais Significativo
T5	Número de Voltas – Bit menos Significativo
T6	Número de Voltas – Bit mais Significativo
T7	Sinal Original do Sensor

Tabela 3.4 – Bits utilizados na comunicação com o sistema

3.5. Fontes de Alimentação

Para que os circuitos funcionem perfeitamente precisamos garantir um suprimento de energia elétrica constante e de boa qualidade. Isto será realizado pela bateria do MCI e pelos circuitos eletrônicos descritos a seguir.

3.5.1. Fonte para os Circuitos Lógicos

Os circuitos lógicos utilizam fontes de alimentação elétrica de 5V e, portanto, os 12V da bateria devem ser convertidos para a voltagem correta, aceita pelos CI's. Pode-se realizar esta tarefa com um CI regulador de tensão (UA7805), que é amplamente difundido no mercado e de baixo custo. O CI empregado e seu esquema de ligação encontram-se abaixo descritos.

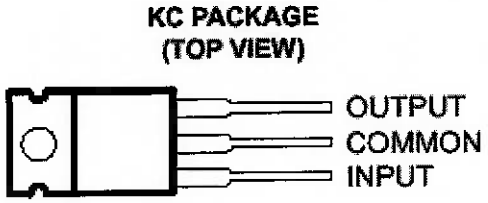


Figura 3.27 – Configuração do CI UA7805

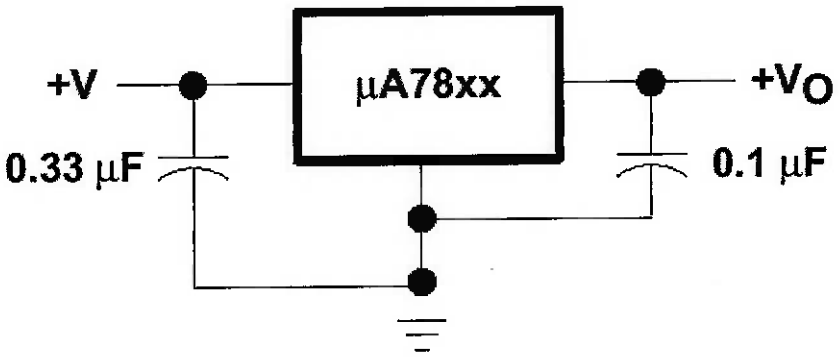


Figura 3.28 – Circuito para alimentação dos CIs lógicos

3.5.2. Fonte para o Driver do Motor de Passo

Para o fornecimento de energia elétrica para o driver do motor de passo os 12V fornecidos pela bateria do motor não são mais suficientes. Portanto, precisa-se elevar a tensão até o nível desejado (indicado pelo fabricante do driver como sendo pelo menos 5 vezes maior que a tensão de placa do motor de passos) que é de pelo menos 20V.

Um estágio amplificador de corrente pode ser construído com um circuito regulador de tensão chaveado elevador de tensão (step up). Este circuito também pode ser encontrado na forma de um CI pronto para o uso com a adição de alguns componentes. A forma do CI bem como o CI e os componentes auxiliares estão conectados, podem ser visualizados abaixo.

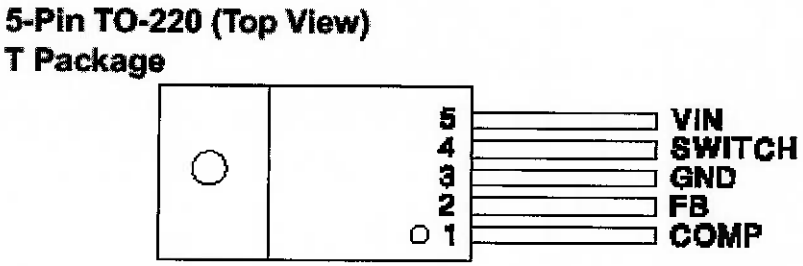


Figura 3.29 – Configuração do CI UC2577

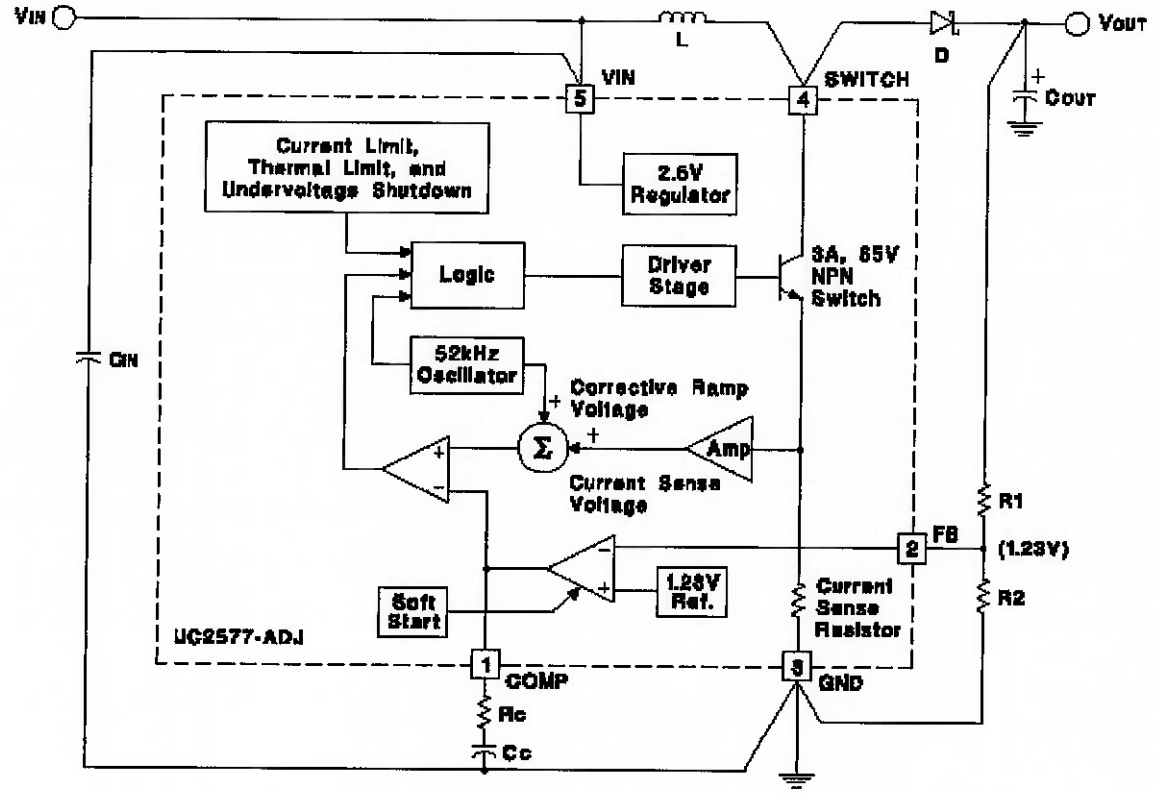


Figura 3.30 – Circuito de alimentação do driver

Com este circuito a tensão foi elevada primeiramente para 30V e, depois, um estágio regulador semelhante ao do item anterior, mas agora com tensão regulada em 24V, foi implementado com o CI UA7824. Os componentes auxiliares empregados no circuito foram escolhidos de acordo com o processo indicado pelo datasheet do CI, que pode ser encontrado, juntamente com todos os datasheets dos componentes utilizados neste projeto, em um apêndice no final do projeto. No caso dos capacitores, os valores encontrados não são valores comerciais exatos, e, portanto, o valor indicado é apenas uma referência.

R1 = 200KΩ C_{OUT} ≥342μF

R2 = 10KΩ, variável C_C ≥546μF

L = 330μH C_{IN} = 0,1μF

A figura abaixo mostra o protótipo montado para o circuito de alimentação do sistema:

Figura 3.31 – Protótipo do circuito de alimentação

4. Modelagem do Sistema

Um Motor de combustão interna é uma máquina altamente complexa e de caráter não linear. Para a modelagem de um MCI freqüentemente lineariza-se o modelo em torno de um ponto de operação, técnica que é bastante pertinente ao caso estudado, pois se pretende controlar o motor a uma única condição de operação, em regime permanente.

Assim sendo, desenvolvem-se neste trabalho três abordagens diferentes, cuja finalidade é gerar um modelo matemático que represente a dinâmica de um MCI, para que se possa observar como um modelo matemático é desenvolvido na prática. O enfoque de cada abordagem pesquisada difere no que cada um dos autores pretende estudar.

Para Lopes [1], o interesse são as respostas transitórias a fim de se verificar as emissões de poluentes durante as trocas de marcha, Moskwa [8] desenvolve um modelo cuja finalidade é estudar o MCI em regime permanente com o intuito de controle ou verificação de parâmetros. Por fim, uma terceira abordagem muito mais simplificada, descrita em Ogata [16], é introduzida para efeito de geração de modelo matemático, baseada nas respostas transientes do MCI utilizado no trabalho e ensaiado em um laboratório, este modelo pode ser utilizado como planta para planejamento do controlador do sistema ou para conferir resultados obtidos por outros modelos matemáticos.

Nos dois primeiros casos, os modelos podem ser adaptados a vários MCI's diferentes, desde que sejam observados as características individuais de cada um deles. Na terceira abordagem o modelo gerado diz respeito ao MCI de estudo no presente trabalho, não podendo ser estendida a outros motores uma vez que é um modelo gerado a partir das respostas obtidas em ensaios experimentais de um MCI específico.

Neste projeto será apresentado a teoria que Lopes [1] e Moskwa[8] desenvolvem em seus projetos na geração de um modelo matemático de um MCI, para que se possa entender como é a metodologia de desenvolvimento de um modelo matemático para um sistema tão complexo e de caráter não linear como um MCI. O modelo obtido pela análise de transiente será utilizado para as simulações finais do

equipamento completo, barco, hélices, motor, etc. Mas, também, pode ser utilizado com o intuito de conferir as respostas fornecidas pelos modelos matemáticos desenvolvidos por Lopes [1] e Moskwa [8], por exemplo.

4.1. Hipóteses Simplificadoras Adotadas

Nos trabalhos dos autores pesquisados, algumas hipóteses simplificadoras adotadas são as mesmas. Portanto, antes de apresentarmos a abordagem que cada um dos autores desenvolve em separado, descreveremos as hipóteses simplificadoras em um único parágrafo, a fim de simplificarmos o entendimento.

À montante da tomada de ar, admite-se pressão atmosférica (P_0) e temperatura ambiente (T_0). A entrada de combustível ocorre ao nível da borboleta onde existe uma seção convergente divergente responsável pela homogeneização da mistura. A válvula borboleta atua como uma perda de carga considerando-se escoamento compressível unidimensional e isentrópico, basicamente constituído por vazão de ar (a vazão de combustível é desprezada).

Na fase seguinte o modelo descrito pela literatura [1] não considera a formação de condensado, pois o motor é movido a gás natural. No caso de um motor movido à gasolina existe a formação de segunda fase no escoamento e, portanto, devem ser feitas as devidas correções. Entretanto, algumas inserções podem ainda ser feitas a respeito do escoamento no coletor de admissão, como considerar o escoamento unidimensional com distribuições de pressão e temperatura uniformes ao longo de seu comprimento. A partir da admissão no cilindro e da seqüência dos quatro tempos do ciclo do MCI há a geração de torque no eixo e então, após a exaustão do cilindro, a mistura flui através do coletor de exaustão até ser liberada na atmosfera. Variações na qualidade do combustível também são desprezadas bem como a dinâmica das bobinas de ignição.

No caso estudado considera-se um motor já pré-aquecido, operando em carga parcial, ou plena, sem a presença de turbo compressor, resfriador intermediário (intercooler) ou recirculação de gases de escape. Não é simulado também nenhum tipo de controlador auxiliar, como de detonação, por exemplo. A figura a seguir apresenta o modelo do coletor de admissão com as principais grandezas envolvidas.

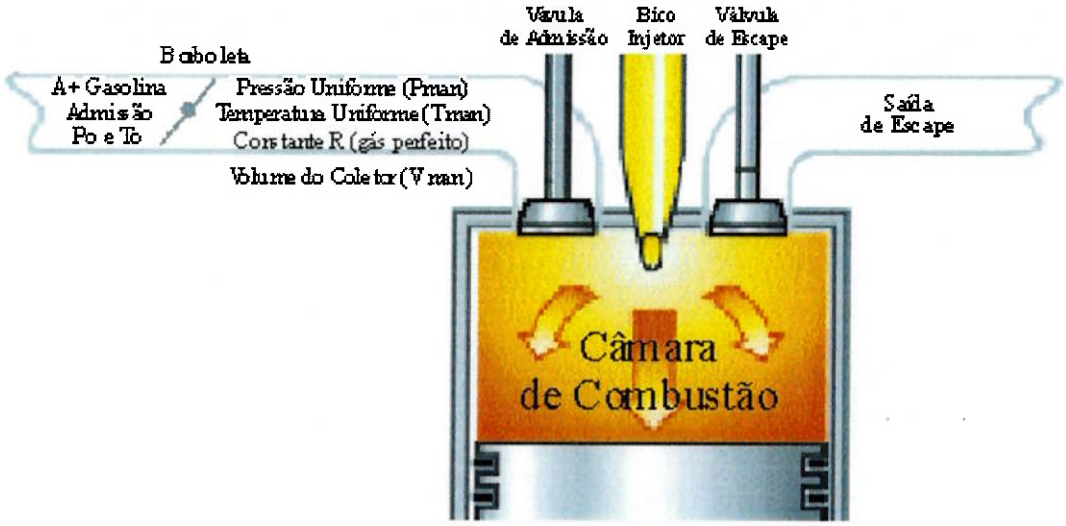


Figura 4.1 - Modelo do coletor de admissão

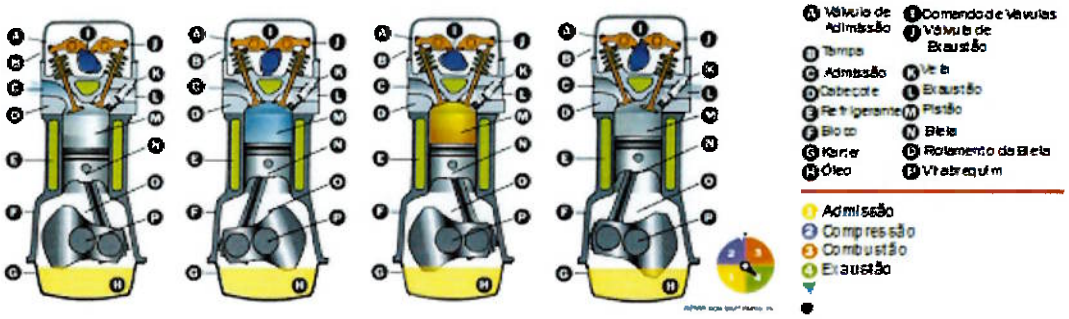


Figura 4.2 - Ciclo quatro tempos

É importante lembrar que, para o modelo gerado a partir das respostas transientes, não são consideradas nenhuma simplificação no que diz respeito ao funcionamento do MCI. No entanto, existe uma simplificação que é a da aproximação da resposta do modelo por uma função de primeiro grau.

4.2. Escoamento de Ar Através da Válvula Borboleta

Neste caso, vemos que ambos os autores pesquisaram a mesma literatura [3]. Portanto, apresentaremos aqui a teoria utilizada em ambas às modelagens no que diz respeito ao escoamento do ar através da válvula borboleta.

As hipóteses principais consideradas neste modelo são de escoamento compressível unidimensional e isentrópico, em condições de temperatura e pressão atmosféricas a montante da válvula, desprezando-se outras perdas de carga, como,

por exemplo, no filtro de ar. A pressão a jusante da válvula é reduzida devido à perda de carga compressível que ela impõe ao escoamento de fluido desconsiderando qualquer tipo de dissipação térmica. O escoamento do ar é então modelado apenas pelo movimento axial em relação ao duto de acordo com a área de passagem.

A área da seção transversal pode ser calcula conforme a figura mostra:

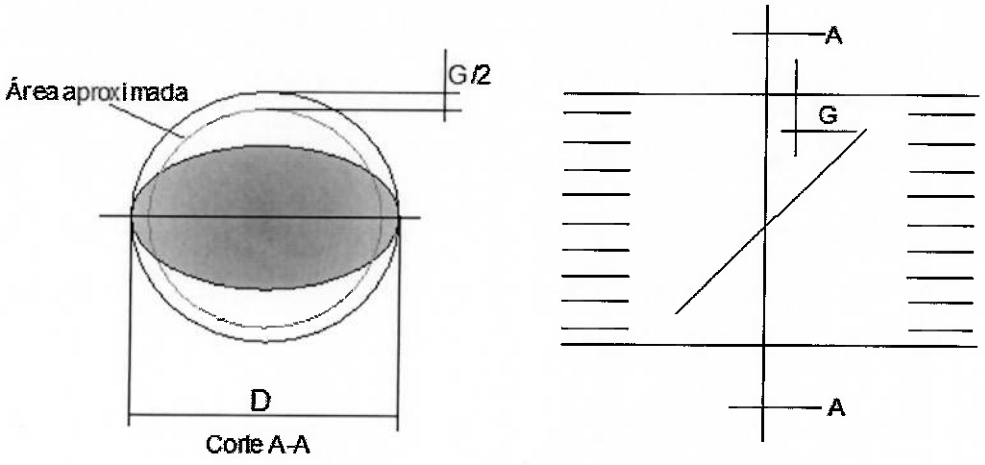


Figura 4.3 – Seção transversal da válvula borboleta

onde $A(\alpha) = \frac{\pi D^2}{16}(3 - 2\cos(\alpha) - \cos^2(\alpha))$

A equação anterior prevê duas hipóteses fundamentais:

- A influência desprezível do diâmetro do eixo de articulação da borboleta (que se torna considerável apenas em ângulos de abertura próximos de 90°);
- Consideração da existência do ângulo de batente (posição fechada de 0°).

O escoamento do ar neste caso aumenta conforme a relação de pressões (jusante/montante, isto é, $\frac{P}{P_0}$) diminui, até um valor limite a partir do qual não há mais crescimento da vazão de ar. As expressões do escoamento compressível podem ser visualizas abaixo, e foram retiradas de Heywood [3]. Assume-se neste caso que o escoamento sobre a válvula é basicamente constituído por ar (ainda que gasolina também seja injetada sobre a válvula) com k=1,4.

Se: $\frac{P}{P_0} \triangleright \left(\frac{2}{k+1}\right)^{\frac{k}{k-1}}$,

$$\dot{m}_{ai} = C_d A(\alpha) \frac{P_0}{\sqrt{RT_0}} \left(\frac{P}{P_0}\right)^{\frac{1}{k}} \left(\frac{2}{k+1}\right)^{\frac{k+1}{k-1}} \left\{ \frac{2k}{k-1} \left[1 - \left(\frac{P}{P_0}\right)^{\frac{k-1}{k}} \right] \right\}^{\frac{1}{2}}$$

Se: $\frac{P}{P_0} \leq \left(\frac{2}{k+1}\right)^{\frac{k}{k-1}}$,

$$\dot{m}_{ai} = C_d A(\alpha) \frac{P_0}{\sqrt{RT_0}} k^{\frac{1}{k}} \left(\frac{2}{k+1}\right)^{\frac{k+1}{2(k-1)}}$$

- $P_0 = 10^5$ Pa, pressão atmosférica;
- $T_0 = 300$K, temperatura ambiente,
- R = 287Nm/(KgK), constante termodinâmica relativa do ar.

O coeficiente C_d relaciona-se com o a relação entre o escoamento através da válvula e sua geometria real., através da hipótese de escoamento unidimensional. Na verdade o valor de C_d é variável tanto com o ângulo de abertura da válvula quanto com a relação de pressões P/P_0, e é tipicamente empírico. Vamos adotar aqui neste trabalho o valor de $C_d = 0,8$, constante.

Outro fator importante é a desconsideração dos efeitos de turbulência que permite que o escoamento logo à jusante da válvula seja considerado exatamente como o escoamento no início do coletor de admissão. De modo que a pressão logo a jusante da válvula corresponda à pressão ao longo de todo o coletor de admissão.

4.3. Modelo Dinâmico de Simulação do Motor

4.3.1. Abordagem de Lopes

4.3.1.1. Introdução

O modelo apresentado a seguir tem o intuito de simular um ambiente dinâmico de um MCI quatro tempos com ignição eletrônica. Ele foi construído a partir da literatura [1] e contém as características convenientes para o estudo em questão.

O modelo, não linear, foi dividido em outros subsistemas que descrevem os vários fenômenos que ocorrem durante o funcionamento de um MCI. São eles:

- Subsistema do coletor de admissão, representado pelas dinâmicas do ar e combustível no coletor;
- Subsistema de combustão é um modelo de geração de torque a partir da vazão de mistura na câmara de combustão e dá superfície de ignição, estaticamente otimizada;
- Subsistema da dinâmica rotacional, que considera carga, atrito e inércia total sobre o eixo para determinar sua velocidade angular.

Para cada um desses subsistemas, os parâmetros de interesse cujos valores numéricos dependem de determinação empírica, foram definidos com base em valores médios razoáveis extraídos principalmente de Heywood [3].

A literatura [1] apresenta dois modelos, um do tipo SISO que não é de interesse, pois não apresenta como variáveis de entrada e saída àquelas desejadas no trabalho em questão (posição da borboleta do carburador e rotação). E um modelo MIMO que é o de interesse, pois considera como entrada a vazão em massa de combustível que pode ser determinada a partir vazão de ar no carburador que, por sua vez, depende da abertura da válvula borboleta e também da rotação do motor. É importante citar que o modelo apresentado a seguir ainda não apresenta todas as características do MCI de estudo neste trabalho, como, por exemplo, o efeito do filme de combustível formado no coletor de admissão. Porém, elas serão comentadas em momento oportuno.

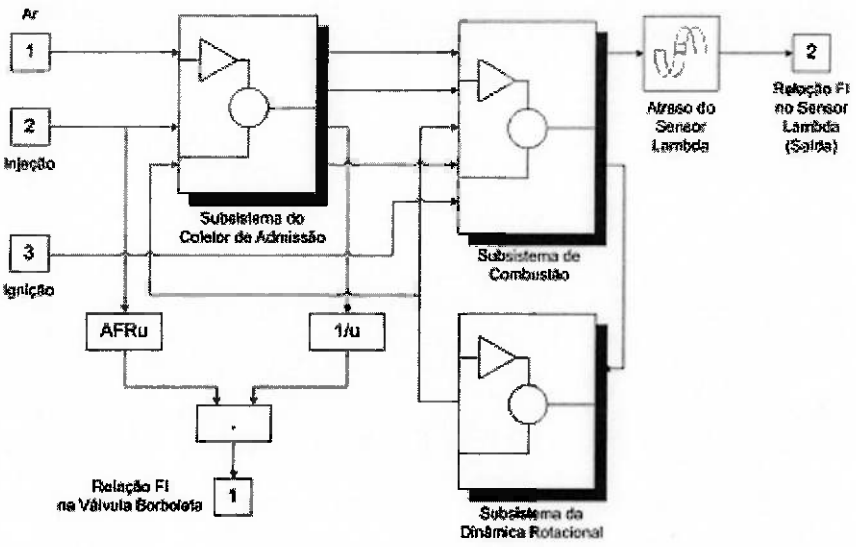


Figura 4.4 – Diagrama de blocos para o modelo dinâmico

4.3.1.2. Submodelo do Coletor de Admissão

O submodelo pode ser caracterizado pela descrição de três escoamentos:

- O escoamento do ar através da válvula borboleta, dadas as condições de pressão e temperatura a montante e a jusante, e a posição angular da válvula;
- O escoamento de ar através do coletor de admissão;
- O escoamento de gasolina através do coletor de admissão

Neste modelo, a mistura ar-combustível não possui características próprias, pois os dois escoamentos são tratados separadamente.

O diagrama de blocos deste submodelo pode ser visto na figura seguinte:

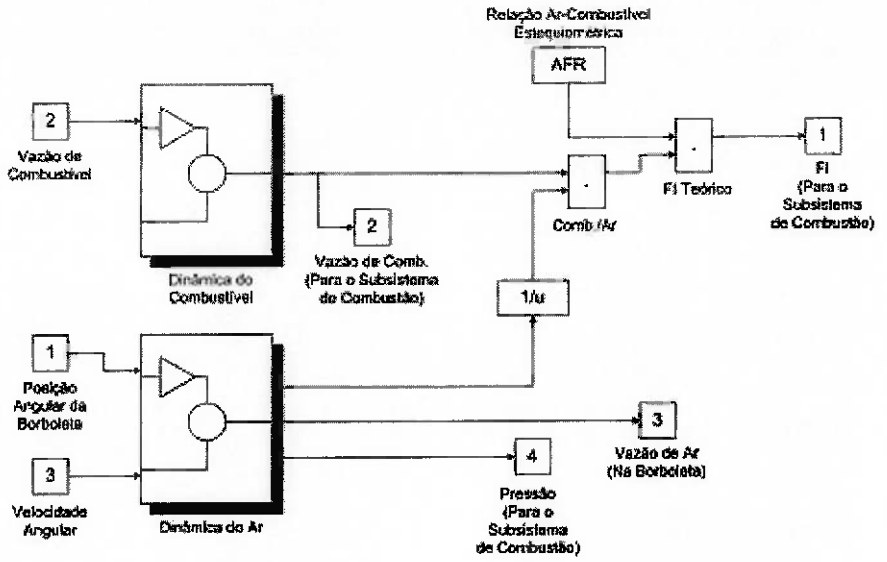


Figura 4.5 – Diagrama de blocos para o submodelo do coletor de admissão

4.3.1.3. Escoamento de Ar Através do Coletor de Admissão

Neste modelo o ar se comporta como gás perfeito, com a dinâmica da pressão associada às vazões de demanda pelo cilindro e de admissão pela válvula borboleta. No modelo apresentado na literatura o motor continha seis pistões, no nosso caso é preciso considerar apenas um.

A vazão de ar em massa admitida ($\dot{m}_{ai}$) é dada pelas expressões vistas anteriormente, e a vazão absorvida pelo cilindros ($\dot{m}_{ao}$) é função da própria pressão no coletor, além da rotação no eixo do motor (n).

A equação que representa a vazão média absorvida do coletor de admissão pelos cilindros, para um caso geral é:

$$\dot{m}_{ao} = \frac{N_{cil} n}{4\pi} \frac{PV_{cil}}{RT} \eta_v$$

onde:

- N_{cil} é o número de cilindros do motor;
- N é a rotação em rad/s;
- P é a pressão no coletor em Pa;
- V_{cil} é o volume útil de combustão em cada cilindro, $3{,}9 \times 10^{-4}$ m^3;

- η_v é o rendimento volumétrico do motor, função da pressão no coletor e rotação, admitido aqui como constante e igual a 0,9;
- T é a temperatura no coletor de admissão, considerada constante e igual a 304K

Se $\dot{m}_a = \dot{m}_{ai} - m_{ao}$, pode-se escrever;

$\dot{P} = m_a \frac{RT}{V}$, onde $\dot{P}$ é a pressão no coletor de admissão.

O modelo do escoamento, tanto através da válvula borboleta como através do coletor de admissão, esta representado na figura a seguir:

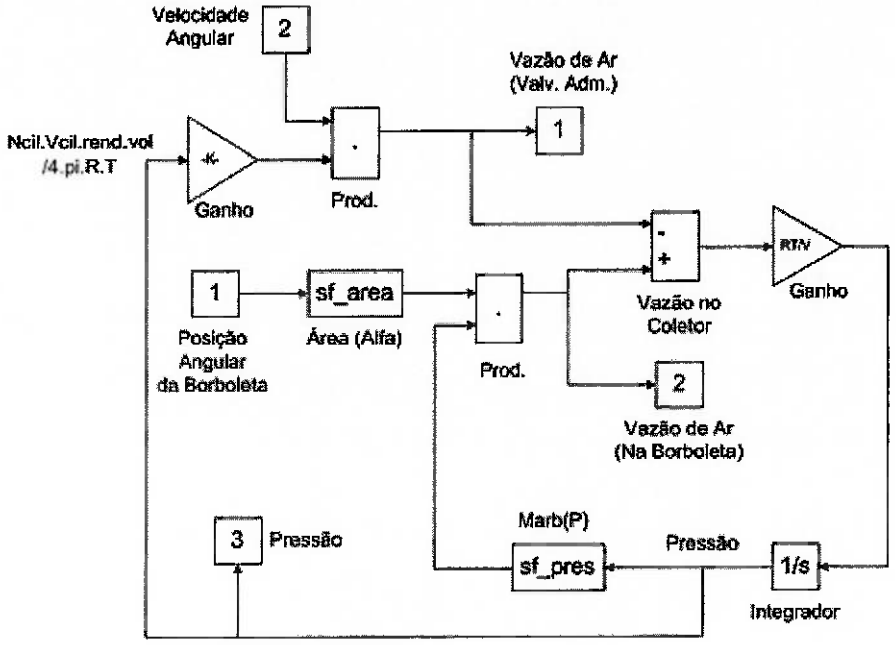


Figura 4.6 – Diagrama de blocos para o modelo do escoamento de ar

4.3.1.4. Escoamento de Combustível Através do Coletor de Admissão

Neste modelo as hipóteses simplificadoras adotadas na literatura [Lopes], não são mais válidas pois, no caso analisado, o motor funciona com gasolina, e na literatura está descrito um motor a gás. Portanto, é necessário considerar uma segunda fase no escoamento do combustível no coletor de admissão.

O processo de injeção de combustíveis líquidos tem um agravante que o de combustíveis gasosos não tem, e que deve ser considerado, que é a formação de um

escoamento bifásico durante o escoamento do combustível da válvula do carburador, ou bico da injeção eletrônica, até a câmara de combustão.

Este tipo de escoamento pode ser representado por um atraso no transporte de combustível que é diferente para combustíveis líquidos e gasosos. Ele depende basicamente da temperatura do coletor de admissão, fluxo de massa, velocidade do motor, etc [8]. Estes parâmetros são de difícil determinação e, portanto, podem ser obtidos empiricamente a partir de testes em motores [9,10]. O diagrama de funcionamento do sistema pode ser observado na figura abaixo.

Uma possível solução para o problema é apresentada por Moskwa [8]e será descrita na modelagem proposta pelo mesmo no capítulo seguintes. Por enquanto, modelo sugerido representa de forma satisfatória o fenômeno dos atrasos.

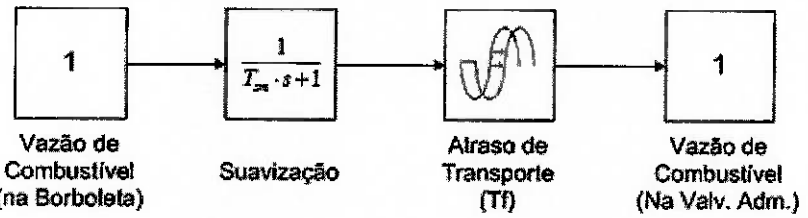


Figura 4.7 – Diagrama de blocos para o modelo do escoamento de combustível

4.3.1.5. Submodelo de Combustão

O subsistema de combustão é o responsável pela geração de torque indicado, ou seja, o torque efetivamente produzido pela queima do combustível.

Ele pode ser analisado como sendo função da vazão de combustível injetado, da energia específica do combustível (no caso do poder calorífico inferior da gasolina) e da própria rotação do eixo do virabrequim.

Para efeito de cálculo associa-se à produção de torque uma função de eficiência indicada, que representa a proporção da energia total inicialmente disponível no combustível que o processo de combustão efetivamente converte em torque. Na literatura, o valor da eficiência indicada depende basicamente do avanço da ignição, da rotação, da pressão no coletor e da relação λ no cilindro [Hendriks & Sorenson].

A superfície de eficiência a ser apresentada é a mesma que foi utilizada pelo autor [1], entretanto ela serve como ilustração para o problema, ou seja, o modelo de funcionamento do sistema não varia em relação ao apresentado na literatura, e pode ser visualizado na figura a seguir, junto com a superfície de controle:

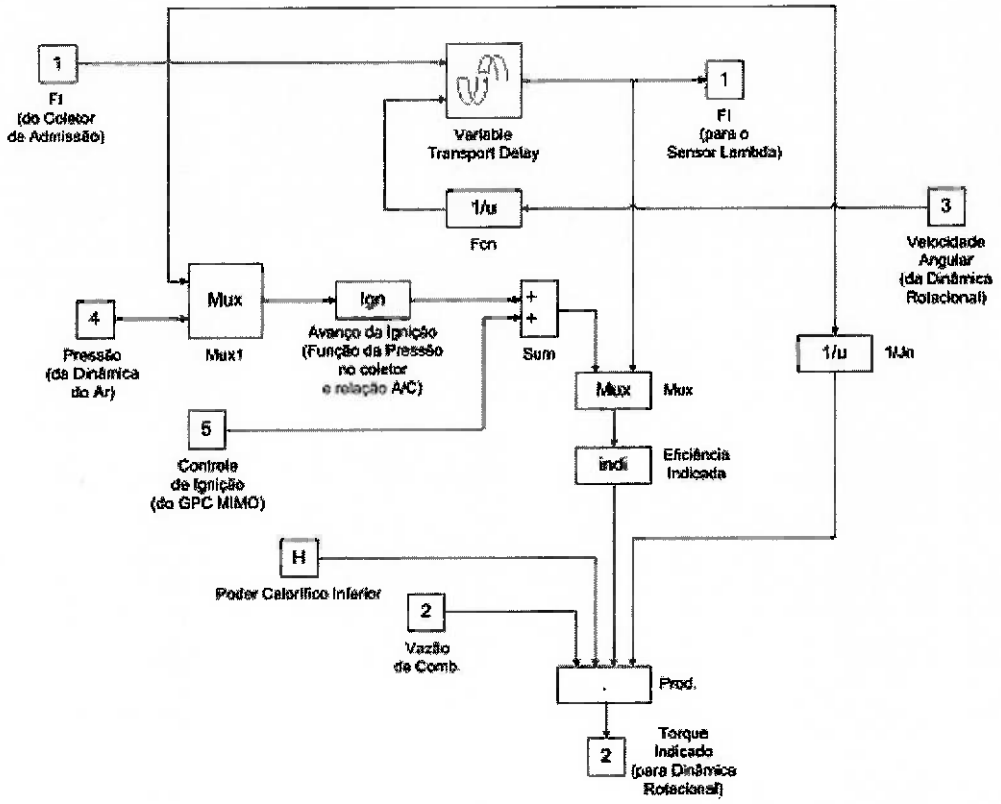


Figura 4.8 – Diagrama de blocos para o submodelo de combustão

4.3.1.6. Submodelo da Dinâmica Rotacional

Neste modelo, o torque indicado pelo submodelo de combustão equilibra ou não os torques resistentes representados pela carga e torque de atrito do motor, produzindo, ou não, aceleração.

A carga representa o esforço externo aplicado ao motor qualquer que seja ele, e o atrito representa os esforços internos ao motor que surgem do seu funcionamento, ambos os esforços são representados por uma função dependente basicamente da rotação. Os gráfico que ilustram esta parte da modelagem também foram retirados da literatura e não representam o MCI de estudo em valores absolutos, mas de um ponto

de vista qualitativo são iguais. O modelo da dinâmica rotacional pode ser visualizado na figura a seguir.

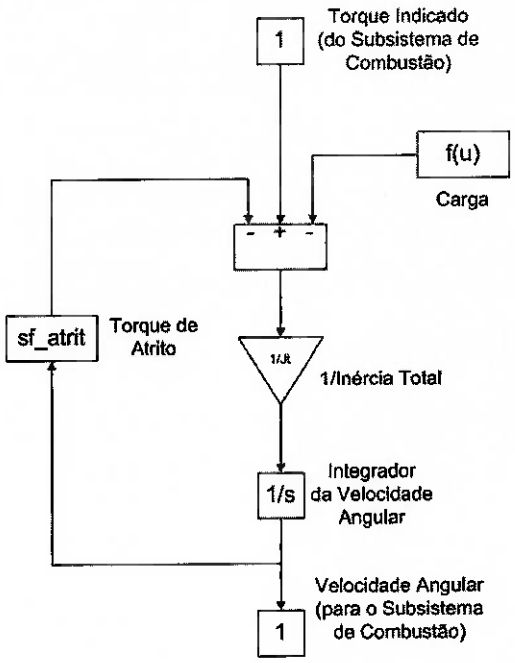


Figura 4.9 – Diagrama de blocos para o modelo da dinâmica rotacional

4.3.1.7. Conclusões

Apesar do modelo ter sido baseado apenas na literatura e não possuir validação experimental, principalmente quanto aos valores médios e as simplificações adotados. Pode-se dizer que este modelo representa o ambiente dinâmico relevante de um MCI quatro tempos à gasolina.

As principais deficiências dizem respeito às hipóteses de independência de algumas variáveis como apresentado anteriormente. Assim, as constantes de tempo do escoamento de gás são função da rotação e pressão no coletor, o coeficiente de descarga relativo à válvula borboleta depende da posição angular da mesma bem como da pressão no coletor, a eficiência indicada depende da rotação e relação λ, etc.

Entretanto, o modelo é bastante simples e global pois considera apenas valores médios e representa de forma bastante clara o funcionamento de um MCI.

4.3.2. Abordagem de Moskwa

4.3.2.1. Introdução

Nesta abordagem proposta por Moskwa apresenta-se um modelo apenas por suas equações diferenciais, diferentemente do apresentado na seção anterior que é apresentado pelo seu diagrama de blocos. Entretanto ambos os modelos representam o ambiente de um MCI sem perda de generalidade.

4.3.2.2. Escoamento de Ar no Coletor de Admissão

Neste caso o modelo do coletor de admissão é um modelo do tipo "lumped parameter". Ele é regido por uma função de eficiência volumétrica baseada nas condições de operação do coletor.

Este modelo assume as seguintes condições para sua validação:

- Os constituintes do escoamento obedecem a Lei dos Gases Perfeitos e a Lei de Dalton de misturas não reagentes
- Distribuição de temperatura uniforme
- Mistura completa do ar com combustível

Ainda, como fator de simplificação, o efeito do escoamento do combustível é desprezado com relação à dinâmica do escoamento do ar. Neste caso a equação da conservação da massa assume a seguinte forma:

$$\dot{m}_{ao} = \dot{m}_{ai} + \frac{P_a V}{RT^2}\dot{T} - \frac{V}{RT}\dot{P}_a$$

Esta equação pode ser rearranjada,

$$\frac{d}{dt}P_a = \left[\frac{\dot{P}}{P} - \frac{RT}{PV}\dot{m}_{ai}\right]P_a + \frac{RT}{V}\dot{m}_{ai}$$

onde,

- P_a = Pressão parcial do ar dentro do coletor de admissão em Pa
- P = Pressão do coletor de admissão em Pa
- R = Constante do gases ideais em kJ/kgK
- T = Temperatura em K
- V = Volume em m^3
- m_{ai} = Massa de ar que entra no coletor de admissão em g/s

Como o estado do coletor de admissão é regido agora pela pressão parcial do coletor de admissão esta pressão deve ser medida, ou estimada. Neste último caso, pode-se utilizar uma modelo de velocidade-densidade, que rege o escoamento total para fora do coletor de admissão e é dado pela seguinte equação:

$$m_o = \kappa m \omega_e \eta_v = \frac{\kappa_1 V}{R} \omega_e \eta_v \frac{P}{T}$$

Juntando-se a esta equação a equação da conservação de massa, obtém-se,

$$\frac{d}{dt} P = \left[\frac{\dot{T}}{T} - \kappa \omega_e \eta_v \right] P + \frac{RT}{V} \dot{m}_{ai}$$

onde,

- κ = Constante que depende de número de cilindros, volume dos cilindros, volume do coletor de admissão
- ω_e = Rotação do MCI em rad/s
- η_v = Eficiência volumétrica

Com isso, podemos agora estimar a massa de ar que escoa para fora do coletor de admissão da seguinte forma:

$$m_{ao} = \frac{\kappa V}{R} \omega_e \eta_v \frac{P}{T}$$

onde, m_{ao} = Massa de ar que escoa para fora do coletor de admissão em g/s

4.3.2.3. Escoamento de Combustível no Coletor de Admissão

Os processos que envolvem a determinação da quantidade de combustível no coletor de admissão são complexos, portanto, este modelo leva em consideração apenas os efeitos dinâmicos que são relevantes do ponto de vista do controle do motor.

Este é um modelo de escoamento de combustível desde o cálculo da demanda de combustível, até o fechamento da válvula de admissão. Uma vez que se trata de um modelo contínuo, muitos dos parâmetros de atraso são calculados com base em valores médios.

Em primeiro lugar considere-se o caso de não haver separação de fases no escoamento dentro do coletor de admissão. O primeiro atraso, seria o do transporte do combustível do carburador até a válvula de admissão, no nosso caso:

$$\Delta t_a = \frac{2\pi}{\omega_e}$$

Temos ainda o atraso do momento em que o combustível é injetado até o fechamento da válvula de admissão,

$$\Delta t_b = \frac{\phi_{ivc} - \phi_{inj}}{\omega_e}$$

onde,

- Φ_{IVC} = Ângulo do virabrequim no momento em que a válvula de admissão se fecha em graus
- Φ_{inj} = Ângulo do virabrequim no momento em que começa a injeção de combustível em graus

Se a injeção de combustível continuar ocorrendo depois que a válvula de admissão estiver fechada, então o restante de combustível será atrasado de duas revoluções do virabrequim, para um motor quatro tempos, ou:

$$\Delta t_1 = \Delta t_a + \Delta t_b$$

$$\Delta t_2 = \frac{4\pi}{\omega_e}$$

Então, para um motor com injeção contínua temos:

$$\int_{t_{inj}}^{t_{IVC}} \dot{m}_f \, dt = \dot{m}_{fIVC} \Delta t_2$$

onde,

- t_{IVC} = Momento em que a válvula de admissão se fecha em s
- t_{inj} = Momento em que o combustível começa a ser injetado em s
- m_f = Massa de combustível que escoa pelo injetor em g/s
- m_{fIVC} = Massa de combustível que entra no motor até o fechamento da válvula de admissão

Na verdade, o processo de injeção de combustível é mais complicado pois existe a separação de fase no escoamento do combustível no coletor de admissão, especialmente no caso da injeção de combustível com a válvula de admissão fechada,

como acontece nos motores com carburadores. Isto resulta num atraso ainda maior na admissão do combustível.

4.3.2.4. Produção de Torque

Apresentaremos aqui um modelo quase-estático baseado no modelo proposto por Dobner. Como se trata de um modelo quase estático, ele não leva em conta nenhum efeito dinâmico, exceto aqueles que tratam do atraso no escoamento do combustível. O torque máximo atingindo, no caso de uma mistura estequiométrica, é função da massa de ar que entra no cilindro. A função que estima o torque máximo é baseada nas experiências de Chang [14] e é dada por:

$$TF = \frac{3Q_{hv}}{2\pi\left(\frac{A}{F}\right)_{est}(AFI)_{est}}\eta_f\Big|_{est} = 1439\eta_f\Big|_{est}$$

$$100\eta_f\Big|_{est} = -157{,}5 + 0{,}926\eta_{CR} + 0{,}854\eta_{MAP} + 0{,}647\eta_N + 0{,}647\eta_{B/L} + 1{,}037\eta_{Vd} + 1{,}016\eta_\phi$$

$$\eta_N = 44{,}6 - 15{,}1N^{-0{,}088}$$

$$\eta_{CR} = 0{,}729\eta_{id} - 0{,}226r_c$$

$$\eta_{id} = [1 - r_c^{(k-1)}]100$$

$$\eta_\phi = 37{,}396$$

$$\eta_{MAP} = 42{,}1 - 4{,}36\left(\frac{P}{101{,}3}\right)^{-0{,}258}$$

$$\eta_{B/L} = -108{,}6 + 145{,}7\left(\frac{B}{L}\right)^{-0{,}020}$$

$$\eta_{Vd} = 45{,}7 - 13{,}8Vd^{-0{,}083}$$

onde,

- TF = Torque produzido pela queima do combustível em Nm/g
- $\eta \mid_{est}$ = Eficiência da mistura estequiométrica

Parâmetros específicos para cada MCI devem ser substituídos nas equações acima de forma que este modelo pode ser utilizado no MCI de interesse.

Para estimarmos o torque no eixo precisamos descontar as perdas por atrito nas partes internas do MCI. Uma estimativa para este valor pode ser obtida por:

$$T_{efetivo} = \left\{ \left[\frac{4\pi}{\#cil} \dot{m}_{ao} TF\ AFI \right]_{(t-\Delta t_3)} SI \right\}_{(t-\Delta t_4)} - T_{atrito} =$$

$$= \frac{4\pi}{\#cil \omega_e|_{(t-\Delta t_4)}} TF|_{(t-\Delta t_3-\Delta t_4)} AFI|_{(t-\Delta t_3-\Delta t_4)} SI|_{(t-\Delta t_3-\Delta t_4)} - T_{atrito}$$

onde,

- $T_{efetivo}$ = Torque efetivo no eixo do motor em Nm
- T_{atrito} = Torque de atrito devido ao funcionamento do MCI em Nm

E o torque consumido pelo atrito pode ser obtido esperimentalmente medindo-se com um dinamômetro o esforço necessário para girar o motor a uma determinada velocidade.

4.3.2.5. Dinâmica Rotacional

A dinâmica da inércia do MCI é modelada, neste caso, como sendo um valor constante, por motivos de simplicidade. Portanto, um virabrequim rígido é modelado utilizando-se da Segunda Lei de Newton.

$$\omega_e = \frac{1}{J_{ef}} \int_{t_0}^{t_1} T_{ap}\, dt + \omega_{e0}$$

onde,

- J_{ef} = Momento de inércia do sistema em kgm^2
- T_{ap} = Torque resistivo aplicado no eixo do motor em Nm

O torque aplicado inclui todas as cargas às quais o MCI possa estar sujeito.

4.3.2.6. Validação

Este modelo desenvolvido pode ser utilizado de três formas diferentes.

- Como modelo de simulação para controle ou verificação da resposta do sistema
- Para comparar os valores obtidos com as saídas do modelo e do MCI real, ou identificação de parâmetros
- Como parte de um algoritmo de controle num veículo qualquer

Estes modelos variam nas entradas que estão disponíveis como, por exemplo, pressão no coletor de admissão, velocidade do motor, etc. E quais as saídas desejadas.

$$\frac{d}{dt}\omega_d = \frac{T_{eixo} - T_d}{J_d}$$

$$\frac{d}{dt}\omega_e = \frac{T_{efetivo} - T_{eixo}}{J_e}$$

$$\frac{d}{dt}T_{eixo} = c(\dot{\omega}_e - \dot{\omega}_d) + K(\dot{\omega}_e - \dot{\omega}_d)$$

onde, ω_d = Rotação do dinamômetro.em rad/s

4.3.2.7. Conclusões

Este modelo é flexível o bastante para representar uma grande variedade de MCI's desde que corretamente adaptado aos parâmetros de cada MCI. No entanto ele não é capaz de prever os pulsos de torque produzidos durante o processo de combustão de cada cilindro individualmente, ele é capaz de prever com precisão considerável o torque médio produzido pelo motor. Erros típicos são da ordem de 3 a 5 por cento [8]).

4.3.3. Abordagem de Análise de Transiente

4.3.3.1. Introdução

Nesta abordagem gera-se um modelo a partir das características dinâmicas do MCI escolhido para o projeto, ou seja, ensaia-se o MCI em um laboratório e coletam-se os dados de interesse. Posteriormente, através de um gráfico gerado pelos dados coletados será analisado para obtenção dos parâmetros de uma função de primeiro grau extraída dos dados fornecidos pelo gráfico.

4.3.3.2. Modelagem de um Sistema de Primeira Ordem

Primeiro considerem um modelo matemático de primeira ordem como o da equação abaixo:

$\frac{C(s)}{R(s)} = \frac{K}{Ts+1}$, onde *C(s)* é a saída da função de transferência, em RPS; *R(s)* é a entrada do sistema, em passos do motor; K é o ganho do sistema, adimensional e T é a constante de tempo do sistema em segundos.

Esta equação pode representar um sistema RC, um sistema térmico, ou qualquer outro sistema que tenha uma resposta dinâmica semelhante. No nosso caso, as respostas obtidas no ensaio podem ser perfeitamente encaixadas nesta classe. Veja gráfico abaixo e compare com o gráfico genérico de um sistema de primeira ordem do próximo tópico.

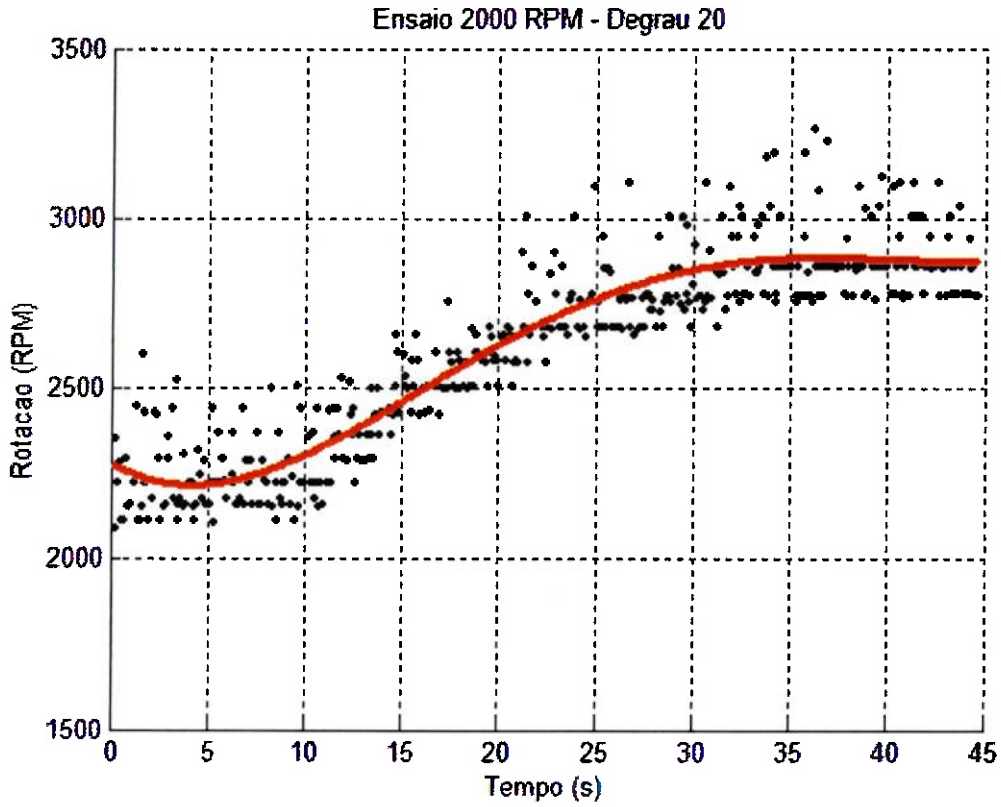


Figura 4.10 – Ensaio degrau do motor a 2000 RPM

Em seguida será analisado o comportamento do sistema dado uma determinada entrada que pode ser um degrau, um pulso ou uma rampa. No presente trabalho limitaremos o estudo do modelo à influência de um degrau no comportamento sistema. Este degrau será representado por uma abertura pré-determinada na borboleta do carburador.

4.3.3.3. Resposta a um Degrau em Sistemas de Primeira Ordem

Uma vez que a transformada de Laplace de um degrau unitário é dada por: 1/s, substituímos *R(s)* = 1/s na equação do sistema de primeira ordem obtemos:

$$C(s) = \frac{K}{Ts+1}\frac{1}{s}$$

Expandindo em frações parciais:

$$C(s) = \frac{1}{s} - \frac{KT}{Ts+1} = \frac{1}{s} - \frac{K}{s+(1/T)}$$

Fazendo a transformada inversa

$c(t) = K(1-e^{-t/T})$, para $t \geq 0$

A equação acima demonstra que a saída *c(t)* é zero e finalmente ela assume o valor de K.Uma característica importante dessa resposta exponencial é o fato de que para *t=T* o valor de *c(t)* é 0,632K, ou seja, a resposta alcançou 62,3% do seu valor total. Pode-se facilmente conferir este resultado substituindo-se t por T:

$c(t) = K(1-e^{-1}) = 0{,}632\,\mathrm{K}$

Note que quanto menor a constante de tempo T, mais rápida é a resposta do sistema. Outra característica importante do sistema é que em t=0 a tangente da curva é K/T. O diagrama do sistema pode ser observado na figura abaixo:

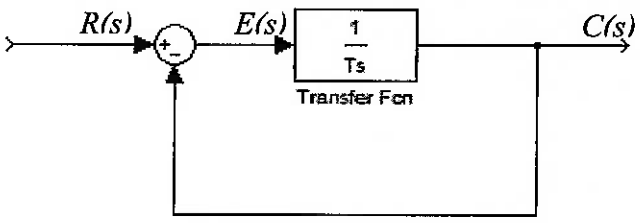


Figura 4.11 – Diagrama de blocos do sistema

O comportamento genérico de um sistema de primeira ordem pode ser observado na figura abaixo:

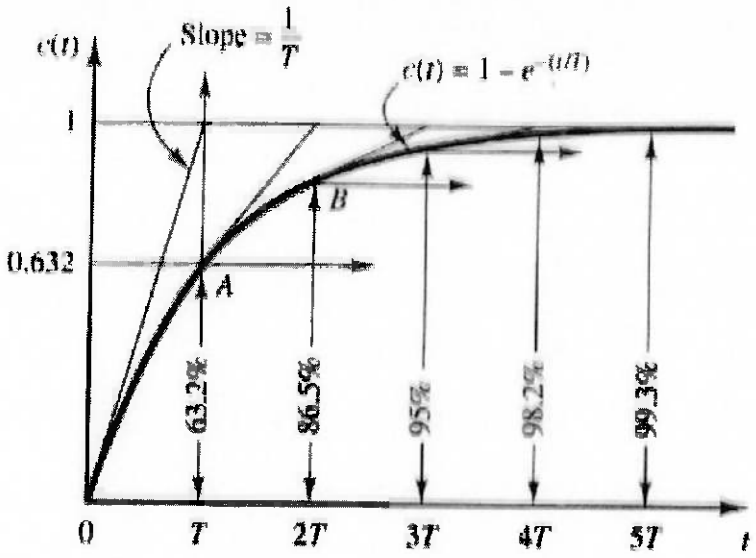


Figura 4.12 – Modelo de primeira ordem

Após a análise dos gráficos dos ensaios do motor, podemos gerar um modelo matemático a partir da equação geral de um sistema de primeira ordem apresentada neste capítulo.

4.3.3.4. Conclusões

O modelo obtido por esta técnica de modelagem é de muito mais fácil obtenção, porém, muito mais simples e não extensível a outros MCI's. Entretanto, não precisamos de análises complicadas como as necessárias nos modelos anteriores, como, por exemplo, o mapa de avanço de ignição ou o rendimento da mistura de ar-combustível. Neste caso só é necessário relacionar a abertura da válvula borboleta com a variação da rotação através do tempo. Isto pode ser obtido coletando-se o valor da rotação ao longo do tempo e colocando-se os valores obtidos num gráfico cuja abscissa tem os valores do tempo decorrido para cada dado amostrado.

5. Software

5.1. Introdução

O software para interação e controle do sistema está sendo desenvolvido para o sistema operacional QNX, utilizando a linguagem C++ com conceitos de orientação a objetos para facilitar o desenvolvimento e a posterior integração com os outros sistemas que compõem o controle automático do barco.

Nesta seção, comentaremos brevemente sobre o software desenvolvido neste trabalho. Uma documentação mais completa e uma listagem de código poderão ser encontradas nos apêndices.

5.2. Levantamento de Funções

Antes de iniciarmos a discussão do software desenvolvido, vamos comentar sobre as principais funcionalidades do sistema operacional que foram utilizadas por nós.

5.2.1. Acesso à Interface Serial

O acesso às portas de comunicação no QNX foi realizado por meio da metodologia padrão para sistemas POSIX, utilizando arquivos de dispositivo existentes no diretório /dev. Como exemplo, para enviar dados para a porta serial podemos utilizar uma sintaxe similar a:

```
fd = open ("/dev/ser1", flags)
write (fd, data, nbytes)
close (fd)
```

Analogamente, foram utilizadas as estruturas e funções padrão do POSIX (termios) para a configuração da porta serial.

O uso de interrupções é no QNX 4 é feito através das funções qnx_hint_attach e qnx_hint_detach, que tem como finalidade associar e desassociar

os Interrupt Requests (IRQs) do hardware com funções escritas para o tratamento destes eventos, chamadas de interrupt handlers. O exemplo abaixo ilustra como o uso destas funções:

```
int interruptID
pid_t intHandler (void)
{
     ...
}

int main (void)
{
     ...
     interruptID = qnx_hint_attach  (4,
                                     &intHandler,
                                     FP_SEG
                                (&interruptID);
     ...
     qnx_hint_dettach (interruptID);
     ...
}
```

5.2.2. Medição Precisa de Tempo Decorrido

Para obtermos uma medição de tempo com precisão na ordem de milissegundos é recomendado pela própria QNX a utilização de timers de hardware, como o controlador de interrupções 8254 ou o contador de ciclos presente em processadores classe Pentium ou superior.

Neste projeto, optamos pela utilização deste contador de ciclos do processador, por se tratar de uma medida de alta resolução e acesso extremamente rápido. Ao realizarmos uma leitura da porta serial, utilizamos o contador para gerar um "timestamp" para ser anexado a esta medida. Sendo o clock do processador conhecido, é possível utilizarmos estes "timestamps" para o cálculo do tempo decorrido entre cada medição.

As funções abaixo, retiradas de um código-fonte exemplo fornecido pela própria QNX são utilizadas neste processo:

```
void read64(int64 *ptr);
```

```
#pragma aux read64 = "db 0fh,31h" "mov [ebx],eax" "mov
[ebx+4],edx" \
        parm nomemory [ebx] modify exact nomemory [eax
edx];

unsigned int read32(void);
#pragma aux read32 = "db 0fh,31h" \
        parm nomemory [] modify exact nomemory [eax edx];
```

5.2.3. Timers

A criação de timers é realizada através da função timer_create:

```
timer_t timer_create(clockid_t clock_id,
                     struct sigevent *event)
```

O primeiro parâmetro é o identificador do relógio interno a ser utilizado pelo timer (em geral, utiliza-se CLOCK_REALTIME), enquanto que o segundo é uma estrutura que especifica que evento será realizado quando o timer disparar. No caso do QNX, podem ser utilizados sinais padrão de UNIX, ou o envio de uma mensagem "proxy" via o mecanismo de comunicação entre processos nativo do sistema operacional

Uma vez criado o timer, os parâmetros de disparo podem ser configurados por meio da função timer_settime:

```
int timer_settime (timer_t timerid,
                   int flags,
                   struct itimerspec *value
                   struct itimerspec *oldvalue)
struct itimerspec{
   struct timespec it_value,
                   it_interval;
}
```

O parâmetro flags especifica se o timer será absoluto ou relativo, enquanto que a estrutura *value define os parâmetros de disparo: it_value é o tempo de espera até o primeiro disparo, e it_interval especifica o tempo para disparos posteriores. Ambos estes parâmetros são do tipo timespec, uma estrutura padrão de sistemas

POSIX, que contém dois parâmetros: tv_sec, tempo em segundos, e tv_nsec, tempo em nanosegundos. O exemplo abaixo define uma estrutura itimerspec para um timer que dispara com após um segundo, e após o primeiro disparo a cada 0.5 s:

```
it_value_tv_sec = 1;
it_value_tv_nsec = 0;
it_interval_tv_sec = 0;
it_interval_tv_sec = 500000000;
```

No software desenvolvido por nós, foi utilizado preferencialmente o mecanismo de mensagens do QNX para a notificação de eventos.

5.3. Software para Interação com o Sistema

O software para interação com o sistema é composto de um total de quatro classes escritas em C++. As classes se dividem em dois grupos:

- Classes "Baixo Nível": Classes que interagem diretamente com o hardware do PC, acessando a porta serial ou o contador de ciclos do processador;
- Classes "Alto Nível": Classes que representam elementos do sistema físico, como os sensores e motor de passo. Utilizam as classes de baixo nível para a interação com o hardware, e encapsulam a lógica de tratamento de dados enviados e recebidos do sistema.

5.3.1. Classes "Baixo Nível"

Duas das classes desenvolvidas se enquadram nesta categoria:

- Serial: Encapsula as funções de configuração, envio e recebimento de dados da porta serial. As funções principais desta classe são o construtor, que inicializa e configura a porta, e as funções de envio e recebimento de dados, write_data e read_data.
- Ticker: Encapsula o acesso ao contador de ciclos do processador. Suas funções principais são get_tickcount, que retorna um "timestamp" a ser anexado a um valor medido, e get_clockspeed, que informa a freqüência de

clock do processador, necessária para o cálculo do tempo decorrido entre dois timestamps.

5.3.2. Classes "Alto Nível"

As duas classes remanescentes se encontram nesta categoria:

- Motor: Representa o motor de passo utilizado para o controle de abertura da borboleta do MCI. As funções mais interessantes são open_steps e close_steps, que permitem a abertura e fechamento da borboleta, e as funções get_position e get_angle, que informam a abertura atual da borboleta em passos ou graus;
- Sensor: Representa o sensor de velocidade fixo no MCI. Sua função principal é get_frequency, que retorna o freqüência de rotação atual do motor em RPM.

5.4. Controlador

Infelizmente, devido às dificuldades de modelagem do motor e à complexidade dos algoritmos de controle estudados inicialmente não foi possível o desenvolvimento de um software de controle para QNX no período deste projeto.

5.5. Softwares Adicionais

5.5.1. Software para Testes da Interface

Em adição ao software para QNX, foi escrito um programa para uso em testes e validação da interface de comunicação para Windows. Este software foi escrito em Visual Basic 6, utilizando o componente mscomm32.ocx para acesso à porta.

A figura abaixo mostra a interface deste programa:

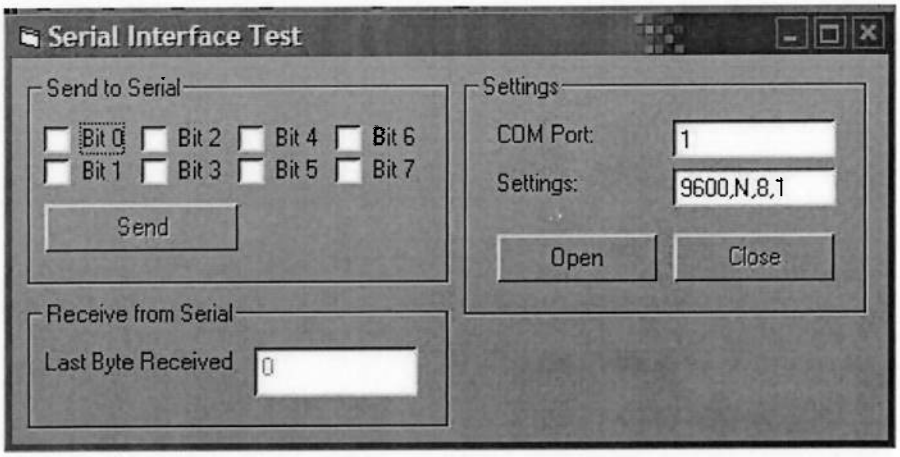


Figura 5.1 – Interface do software de testes

Por ser um programa de testes, suas funções são bastante restritas, limitando-se ao envio e recebimento de bytes individuais pela porta serial.

5.5.2. Software para Ensaios

Para a aplicação da metodologia escolhida de modelagem, foi desenvolvido um software para a realização de ensaios "degrau" do MCI. O software foi escrito para o QNX, utilizando as classes descritas no item 5.2, e possuía as seguintes funções:

- Regulagem de abertura da borboleta;
- Realização de medição individual de rotação;
- Busca automática de um valor de rotação;
- Realização de ensaio degrau.

Foi escrita uma classe adicional para este software, denominada Logger. Esta classe atua como "buffer" para os resultados obtidos no ensaio, e gera ao final do ensaio um arquivo CSV (valores separados por vírgula) que pode ser lido pelo Excel ou MatLab para a confecção de gráficos. Abaixo temos uma imagem mostrando o menu principal deste programa:

```
Programa de Controle do Motor - Versao 1.0
Filipe Badaro and Tomas D'Agostino

Menu Principal
Controle do Motor:
1) Aumentar abertura n passos
2) Diminuir abertura n passos
Posicao Atual: 10 passos ~   3.60 graus
Controle do Sensor:
3) Mudar parametros de leitura
4) Realizar Leitura
Ensaios:
5) Buscar Rotacao
6) Realizar Ensaio Degrau
Outros:
7) Sair
Parametros do Sensor: Samples = 2, Min Divisions = 4, Thres = 20
Parametros de Busca de Rotacao: Tol = 5 MinFreq = 3
Ultima Leitura do Sensor: 0 RPM 0.000 s
Ultima Busca de Rotacao: 0 RPM 0 passos

Opcao: _
```

Figura 5.2 – Interface do software de ensaios

6. Ensaios

6.1. Introdução

Para a obtenção de um modelo matemático baseado em respostas dinâmicas, primeiro se faz necessário ensaiar o MCI em um laboratório. Neste capítulo trataremos deste assunto, bem como o desenvolvimento dos modelos gerados para cada um dos testes realizados.

Os testes foram realizados em um laboratório, com o MCI acoplado a um freio elétrico. Este freio nada mais é do que um gerador de corrente contínua (dínamo) que tem o campo magnético da armadura excitado por uma corrente elétrica que varia de acordo com o torque resistivo desejado. As fotos abaixo mostram a montagem para realização deste ensaio:

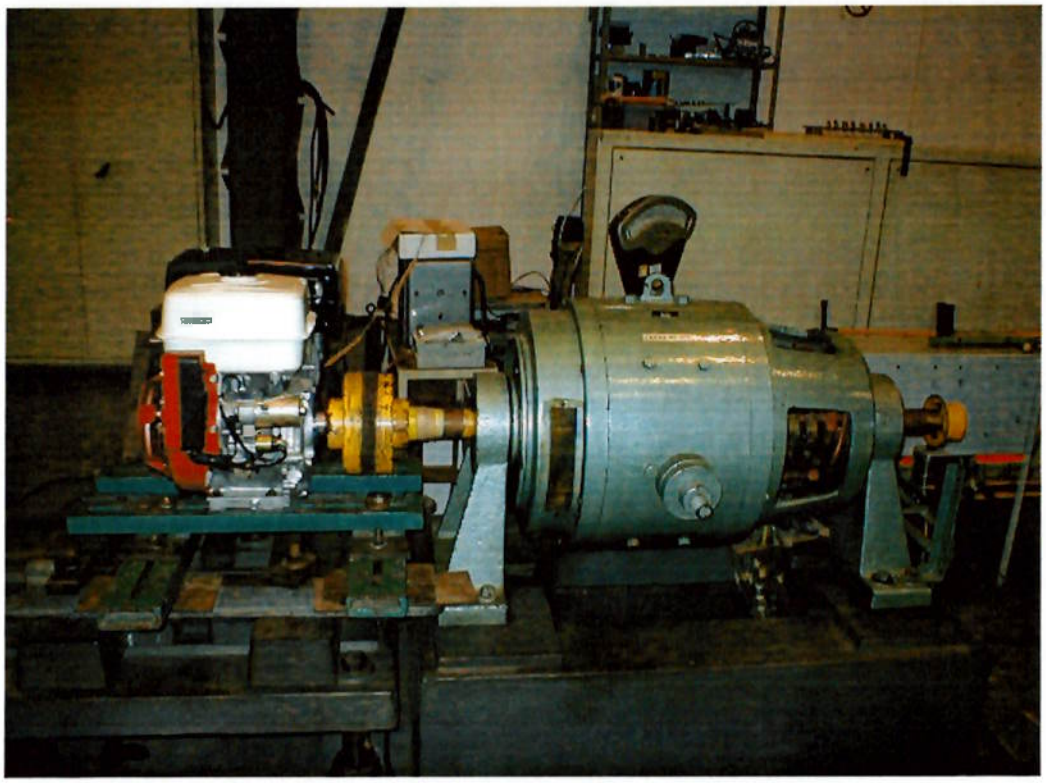


Figura 6.1 – Montagem para ensaio

Figura 6.2 – Visão geral do ensaio

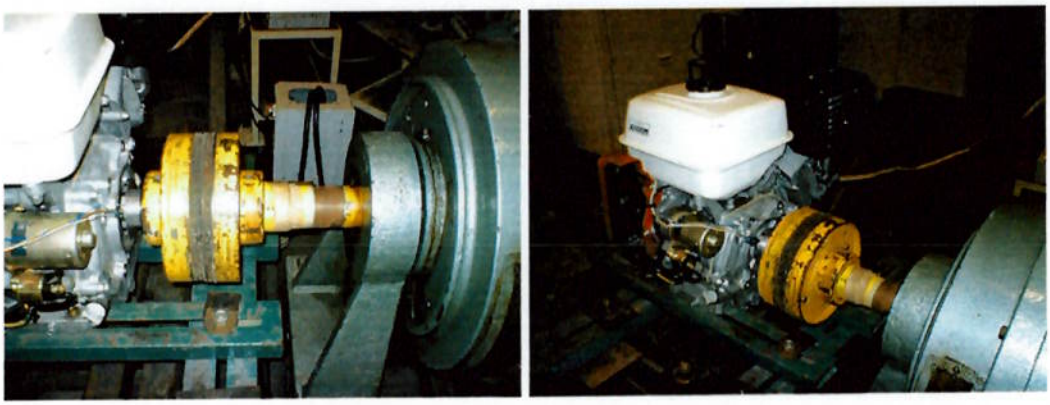


Figura 6.3 – Acoplamento entre motor e freio

Neste trabalho somente foram realizados testes com o freio sem resistência nenhuma, a não ser a própria inércia do sistema e as perdas envolvidas no acoplamento MCI/freio, visto que infelizmente o MCI não ficou pronto a tempo de permitir a realização de testes mais completos.

6.2. Resultados dos Ensaios

Os resultados dos ensaios serão expostos na forma de gráficos, com todos os parâmetros explicitados em suas legendas. Em seguida, serão apresentados os modelos obtidos para cada ensaio diferente.

Os ensaios consistem em estabilizar o MCI em uma determinada rotação inicial e, após 10 segundos do início da coleta de dados, dar um degrau na entrada do sistema (válvula borboleta). Para cada rotação foram escolhidos degraus diferentes para gerar diferentes situações e estudar o MCI em várias condições de operação.

Os gráficos estão agrupados por rotação, ou seja, para cada rotação inicial diferente existe um conjunto de gráficos, um para cada degrau escolhido e um último que junta os gráficos anteriores em um só para efeito de comparação.

Os modelos obtidos para cada ensaio serão colocados ao final da apresentação dos conjuntos de gráficos relativos às diferentes condições iniciais com o intuito de facilitar o entendimento.

6.2.1. Gráficos e Modelos

6.2.1.1. Ensaio a 2000 RPM

Neste ensaio partiu-se de uma rotação inicial de 2000 RPM e foram dados dois degrais diferentes na válvula borboleta, um de 10 passos e outro de 20 passos. Cada passo corresponde a uma abertura de 0,36° na válvula borboleta, ou 0,00628 rad.

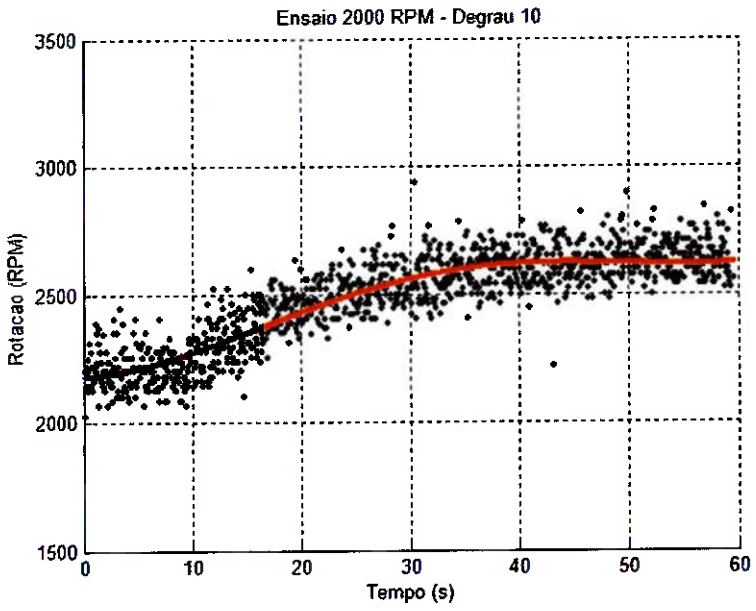


Figura 6.4 – Ensaio a 2000 RPM – Degrau 10

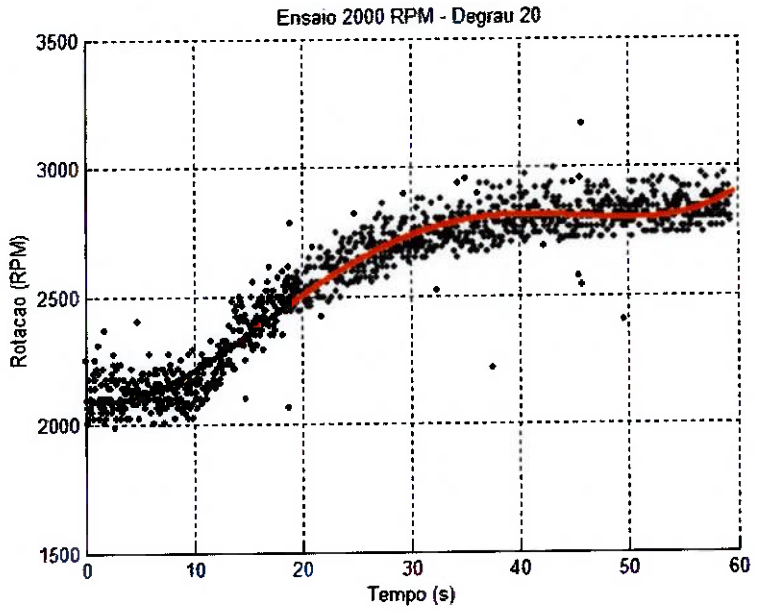


Figura 6.5 – Ensaio a 2000 RPM – Degrau 20

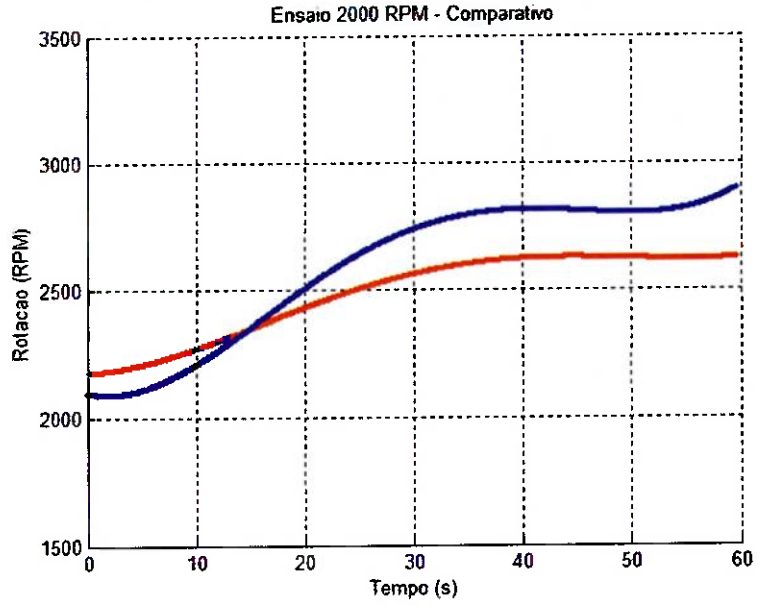


Figura 6.6 – Ensaio a 2000 RPM – Comparativo

Os modelos obtidos para cada gráfico foram calculados segundo o desenvolvimento explicitado em capítulo anterior, capítulo 4 - Modelagem do Sistema, e podem ser visualizados na forma das equações, a seguir:

$$\frac{C(s)}{R(s)} = \frac{0{,}67}{10s+1}$$, para o degrau de 10 passos

$$\frac{C(s)}{R(s)} = \frac{0{,}58}{10s+1}$$, para o degrau de 20 passos

O ganho é dado em RPS/passo em ambos os casos.

6.2.1.1.1. Comentários

É possível observar que apesar das condições de ensaio diferentes o sistema possui o mesmo tempo de acomodação, cerca de 30 segundos e a mesma constante de tempo de 10 segundos. O mesmo vale para os ganhos, que nos dois casos são muito similares.

6.2.1.2. Ensaio a 2500 RPM

Neste ensaio partiu-se de uma rotação inicial de 2500 RPM e novamente foram dados dois degrais diferentes na válvula borboleta, um de 10 passos e outro de 20 passos.

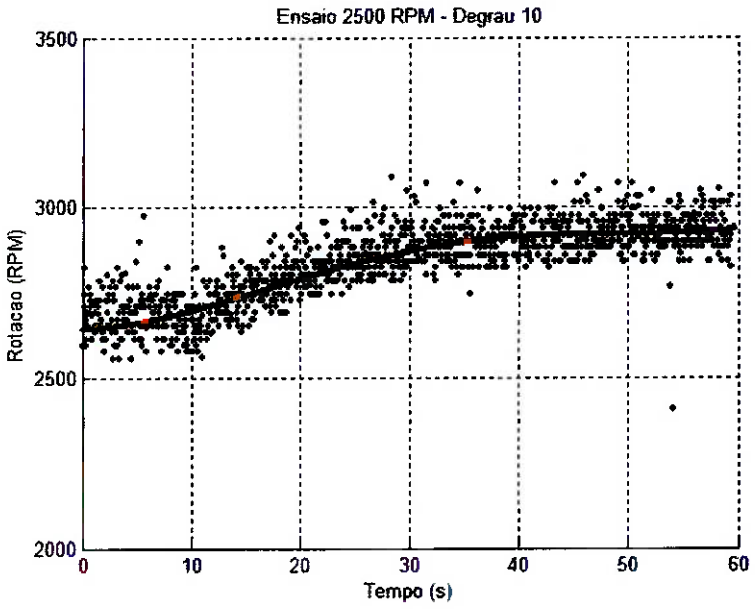


Figura 6.7 – Ensaio a 2500 RPM – Degrau 10

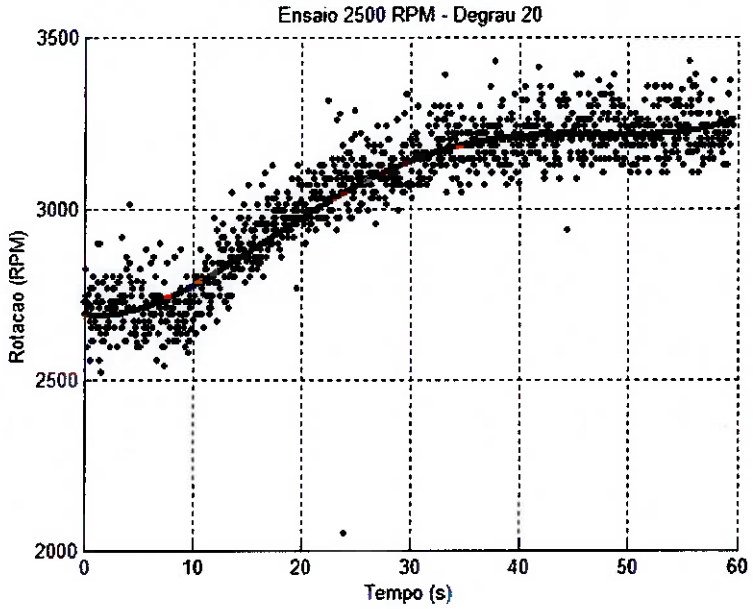


Figura 6.8 – Ensaio a 2500 RPM – Degrau 20

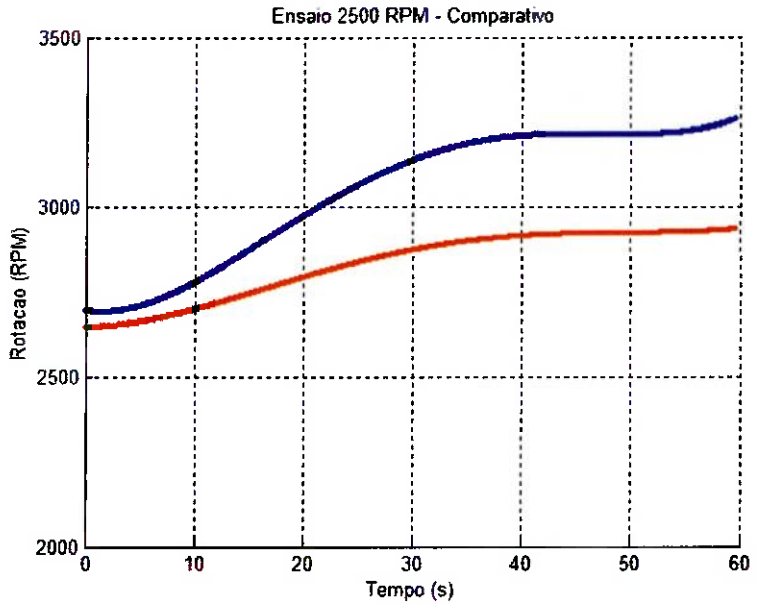


Figura 6.9 – Ensaio a 2500 RPM – Comparativo

Os modelos obtidos para cada gráfico neste caso utilizaram-se da mesma metodologia apresentada no capítulo 4, e podem ser visualizados na forma das equações, a seguir:

$$\frac{C(s)}{R(s)} = \frac{0{,}5}{10s+1}$$, para o degrau de 10 passos

$$\frac{C(s)}{R(s)} = \frac{0{,}45}{10s+1}$$, para o degrau de 20 passos

O ganho é dado em RPS/passo em ambos os casos.

6.2.1.2.1. Comentários

Observa-se que para esta faixa de rotação os modelos permanecem bastante semelhantes ao caso anterior, demonstrando a validade do modelo para uma faixa de rotações maior do que simplesmente aquelas em torno do ponto de equilíbrio. Também se observa que, apesar das condições de ensaio diferentes, o sistema possui o mesmo tempo de acomodação, cerca de 30 segundos. Entretanto, da mesma forma que no modelo anterior, para o degrau de 20 passos o ganho do sistema é ligeiramente menor do que o para o de 10 passos.

6.2.1.3. Ensaio a 3000 RPM

Neste último caso o MCI foi submetido a um único degrau de 10 passos, pois a rotação máxima do equipamento de laboratório estava perigosamente perto do seu limite de operação.

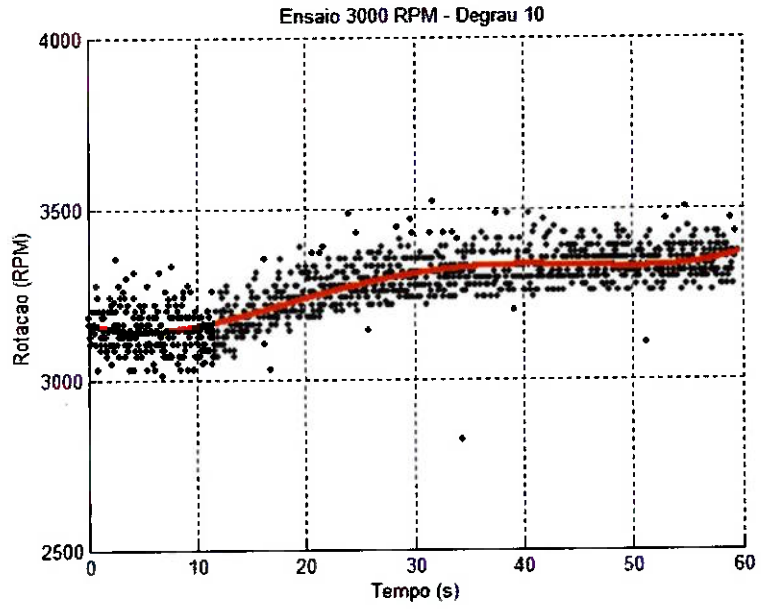


Figura 6.10 – Ensaio a 3000 RPM – Degrau 10

O modelo obtido neste último caso pode ser visualizado na forma da equação a seguir, gerada a partir da mesma metodologia aplicada no dois casos anteriores:

$\frac{C(s)}{R(s)} = \frac{0,33}{10s+1}$, para o degrau de 10 passos

O ganho novamente é dado em RPS/passo.

6.2.1.3.1. Comentários

Observa-se que para esta faixa de rotação o modelo também tem a mesma constante de tempo dos modelos anteriores. Este resultado é interessante, pois demonstra que, para uma ampla gama de pontos de operação, o MCI responde no mesmo período de tempo. Entretanto, pode-se notar que o tempo de acomodação não diminuiu com relação ao ensaio anterior que era de aproximadamente 30 segundos.

6.3. Conclusões

Concluímos que os resultados são satisfatórios, e complementam por meio de experimentação prática a teoria estudada neste trabalho com experimentação prática. Os gráficos obtidos para cada um dos ensaios demonstram com clareza o comportamento do sistema, apesar da dispersão dos pontos gerados pelo sensor de rotações, permitindo o desenvolvimento de um modelo consistente e simples o bastante para ser analisado ou simulado sob outras condições de operação.

Os modelos aqui obtidos serão estudados e incorporados ao modelo completo do sistema (casco e propulsores) para análise do seu comportamento, e projeto de um controlador de velocidade. Note que estes modelos consistem em uma primeira aproximação do funcionamento do sistema, e poderão ser ajustados durante as simulações para reproduzir com maior fidelidade o comportamento do sistema.

7. Simulações

7.1. Introdução

Neste capítulo descreveremos como foi realizada a modelagem do barco, seguido por simulações mostrando o comportamento do sistema barco-motor em malha aberta, finalizando com a adição de um controlador PID ao sistema.

7.2. Modelagem Matemática do Sistema Casco-Hélice

7.2.1. Equações de Movimento

Para a obtenção das equações de movimento, partiremos da aplicação da segunda lei de Newton aos macro-aspectos dos sistemas propulsor-casco, motor-propulsor e casco-hélice.

7.2.1.1. Interação Propulsor-Casco

As principais forças que atuam no casco do navio são:

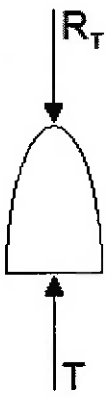


Figura 7.1 – Forças no casco do navio

Aplicando a segunda lei de Newton obtemos:

$$\dot{V} = \frac{1}{M}[T \cdot (1-t) - R_T]$$

Aonde,

V = Velocidade do casco

M = Massa total deslocada pelo navio

T = Empuxo produzido pelo hélice (função da rotação do hélice)

R_T = Resistência ao avanço do casco (função da velocidade do casco)

t = Coeficiente de redução da empuxo

n = Rotação do hélice

7.2.1.2. Interação Motor-Propulsor

Os principais esforços atuantes no eixo e hélice do navio são:

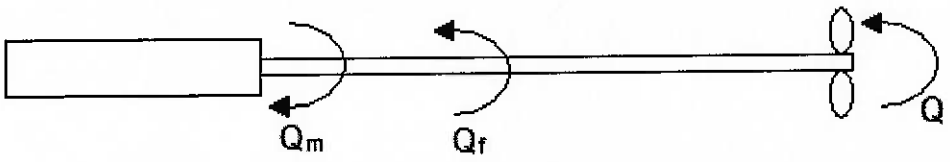


Figura 7.2 – Forças no eixo do navio

Aplicando novamente a segunda lei de Newton, obtemos:

$$\dot{n} = \frac{1}{2\pi J} \cdot (Q_m - Q_f - Q)$$

7.2.1.3. Casco

O modelo matemático do casco é representado pela curva que relaciona a resistência total (R_T) com a velocidade to barco (V). Esta curva é obtida através de um ensaio de reboque de modelos, em um tanque de prova. Este ensaio foi realizado em nosso modelo de barco pelo IPT, e foi obtida a seguinte tabela de resultados:

Fn	EHP (kW)
0,39	$1,49*10^{-1}$
0,46	$3,19*10^{-1}$
0,54	$5,36*10^{-1}$
0,58	$6,10*10^{-1}$
0,62	$6,77*10^{-1}$
0,66	$8,05*10^{-1}$
0,70	$9,57*10^{-1}$
0,77	1,17
0,85	1,42
0,93	1,71
1,01	2,04
1,08	2,41
1,16	2,84
1,24	3,31
1,32	3,83
1,39	4,42
1,47	5,07

Tabela 7.1 – Ensaio de reboque

A partir destes dados foi gerado o seguinte gráfico:

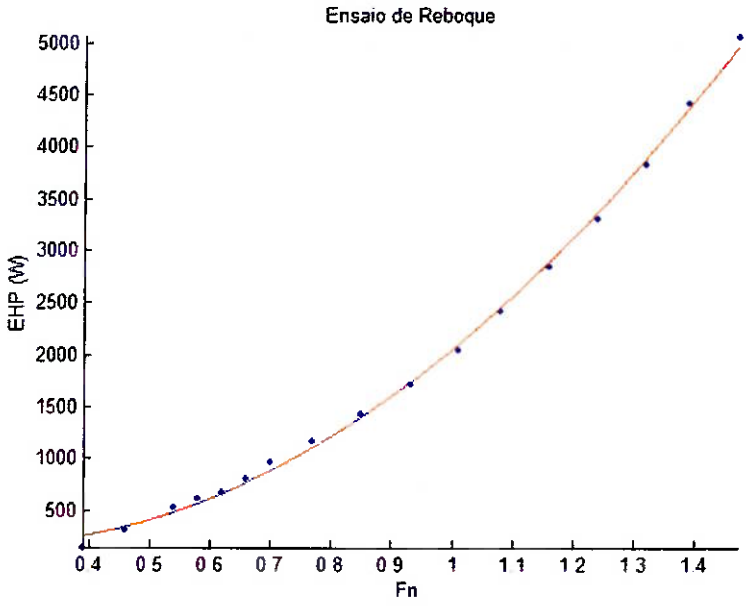


Figura 7.3 – Ensaio de reboque

Para permitir a utilização do gráfico acima no modelo, foi realizada uma interpolação polinomial de ordem 2, utilizando o método dos mínimos quadrados. O polinômio resultante desta interpolação foi:

$$EHP = 1000 \cdot (3.0478 \cdot Fn^2 - 1.2999 \cdot Fn + 0.2992)$$

Conforme mencionado no capítulo 3, EPH é a potencia requerida para rebocar o modelo a uma determinada velocidade e Fn é o número de froude. A relação entre estes parâmetros com a Resistência Total e a Velocidade é dada pelas seguintes equações:

$$V = Fn\sqrt{gL}$$

$$EHP = R_T V$$

Onde g é a aceleração da gravidade e L é o comprimento do barco.

7.2.1.4. Hélice

Adotamos para a modelagem do propulsor as curvas características de coeficientes de empuxo (K_T) e torque (K_Q) em função do coeficiente de avanço (J), mostradas na figura 3.1 deste trabalho. Estas curvas foram aproximadas por polinômios de primeiro grau utilizando-se o método dos mínimos quadrados, obtendo-se:

$$K_T = -0{,}4946 \cdot J + 0{,}6925$$

$$Kq = (0{,}1) \cdot (-0{,}9161 \cdot J + 1{,}3568)$$

7.2.1.5. Interação Casco-Hélice

Os coeficientes utilizados para definir a interação casco-hélice são:

- t = Coeficiente de redução de empuxo
- err = Eficiência relativa rotativa

O coeficiente de redução de empuxo é um artifício utilizado para considerar o aumento de resistência ao avanço provocado pelo posicionamento do hélice na popa. Já a eficiência rotativa relativa é utilizada para corrigir o torque obtido no ensaio de água aberta, obtendo-se efetivamente o torque requisitado pelo propulsor do navio. Com estes dois coeficientes, a expressão para o empuxo necessário e torque no eixo tornam-se:

$$T = K_T \cdot \rho \cdot n^2 \cdot D^4 \cdot (1 - t)$$

$$Q = K_Q \cdot \rho \cdot n^2 \cdot D^5 / err$$

Onde D é o diâmetro do hélice e r a densidade do líquido aonde o navio se encontra.

7.2.2. Diagrama de Blocos do Sistema Casco-Hélice

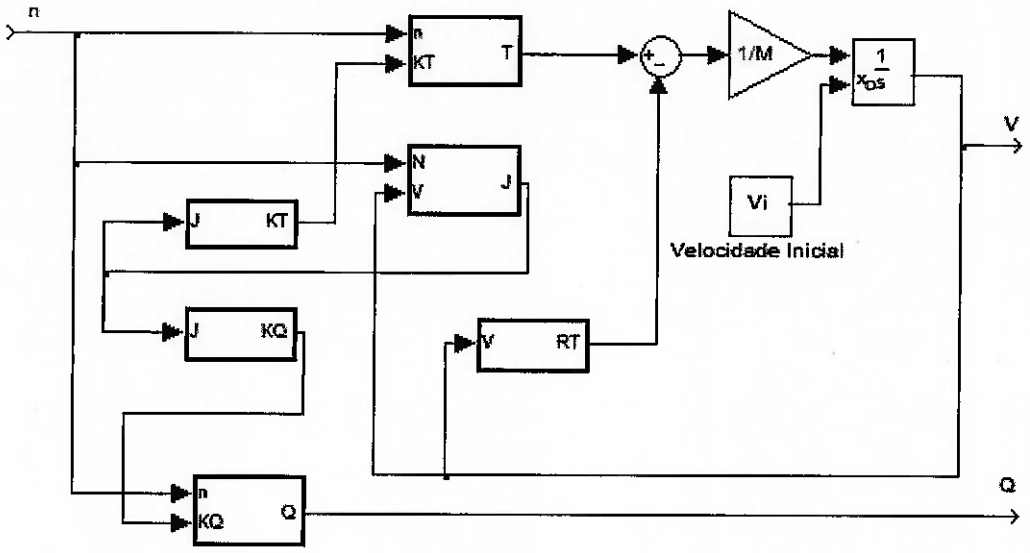


Figura 7.4 – Diagrama de blocos do modelo casco-hélice

7.3. Sistema completo: Casco, Hélice e Motor

Para a simulação completa do sistema foram utilizados os modelos matemáticos desenvolvidos no capítulo 6. Os modelos Apresentados tiveram pequenas modificações no ganho para que o comportamento observado no ensaio fosse representado com maior fidelidade possível na simulação.

Para cada situação ensaiada será apresentada a simulação da influência da rotação na velocidade desenvolvida pelo barco. Os gráficos conterão as saídas do sistema que são rotação do motor, a velocidade do barco e o torque exigido pelo motor para impulsionar o barco.

7.3.1. Gráficos e Modelos Corrigidos

Os gráficos gerados pelos modelos matemáticos serão agrupados segundo a rotação inicial do ensaio e no final de cada grupo será apresentado o modelo do motor corrigido.

O modelo utilizado na simulação do sistema completo pode ser visualizado na figura a seguir.

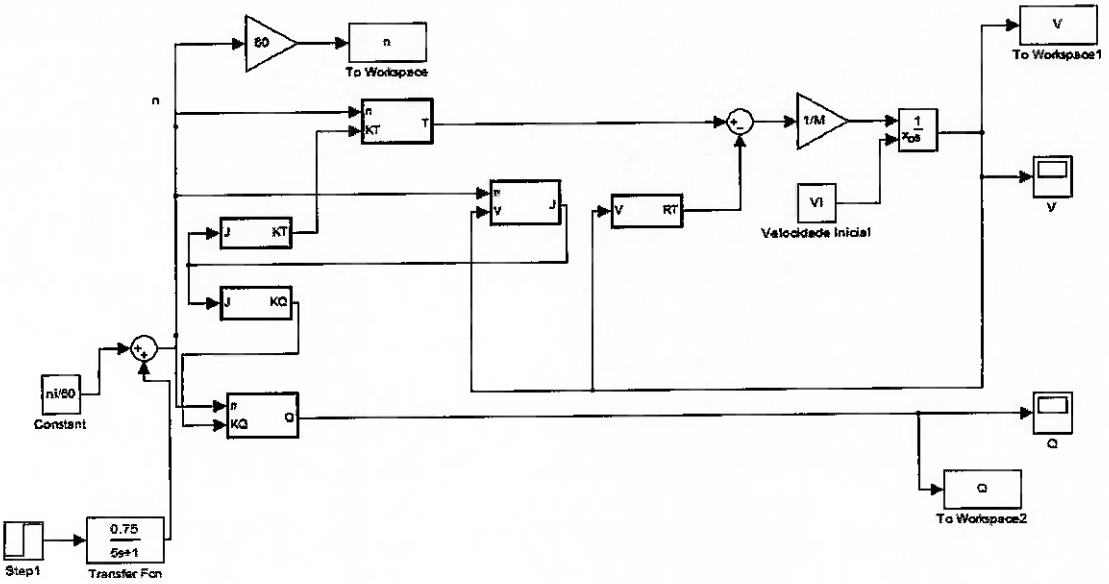


Figura 7.5 – Diagrama de blocos para o modelo do sistema completo

A planta apresentada foi modificada na parte que corresponde ao degrau de entrada e modelo do motor de acordo com as condições de cada simulação. Entretanto, as características gerais são absolutamente iguais em todas elas.

7.3.1.1. Simulação a 2000 RPM

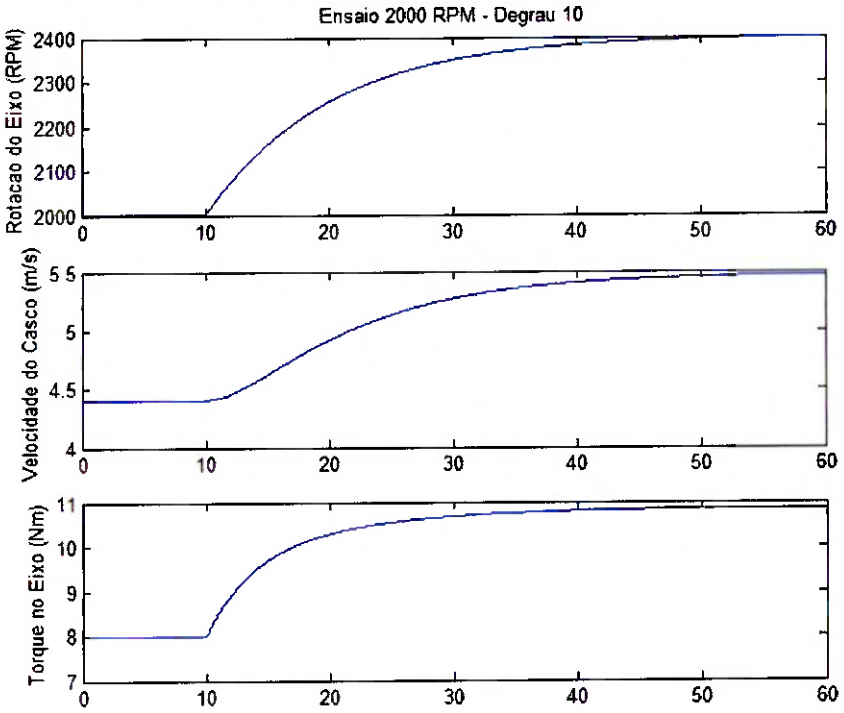


Figura 7.6 – Simulação 2000 RPM - Degrau 10

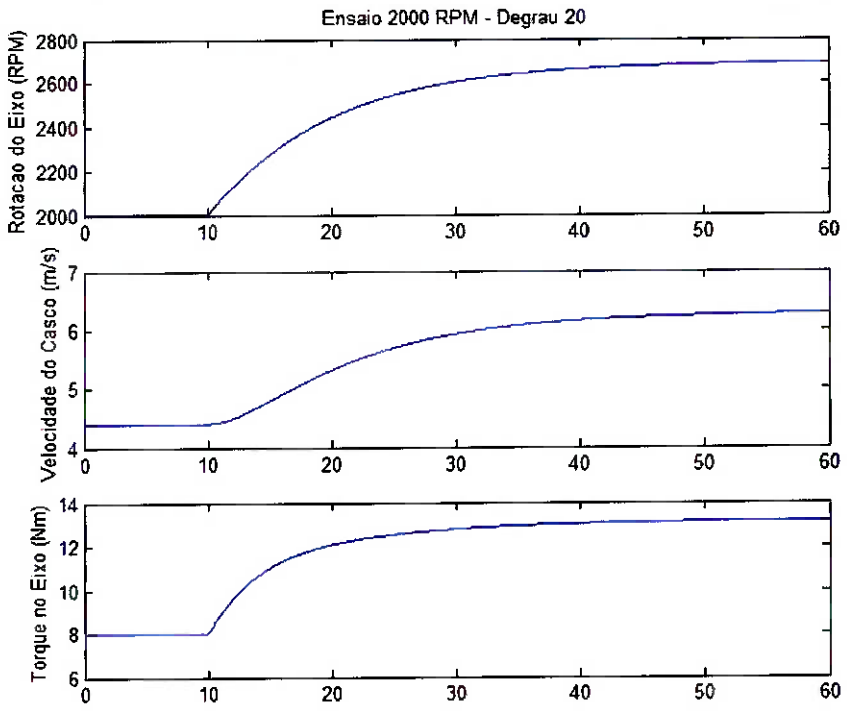


Figura 7.7 – Simulação 2000 RPM - Degrau 20

7.3.1.1.2. Modelos Corrigidos

Os modelos corrigidos podem ser observados a seguir:

$\frac{C(s)}{R(s)} = \frac{0,67}{10s+1}$, para o degrau de 10 passos

$\frac{C(s)}{R(s)} = \frac{0,58}{10s+1}$, para o degrau de 20 passos

7.3.1.2. Simulação a 2500 RPM

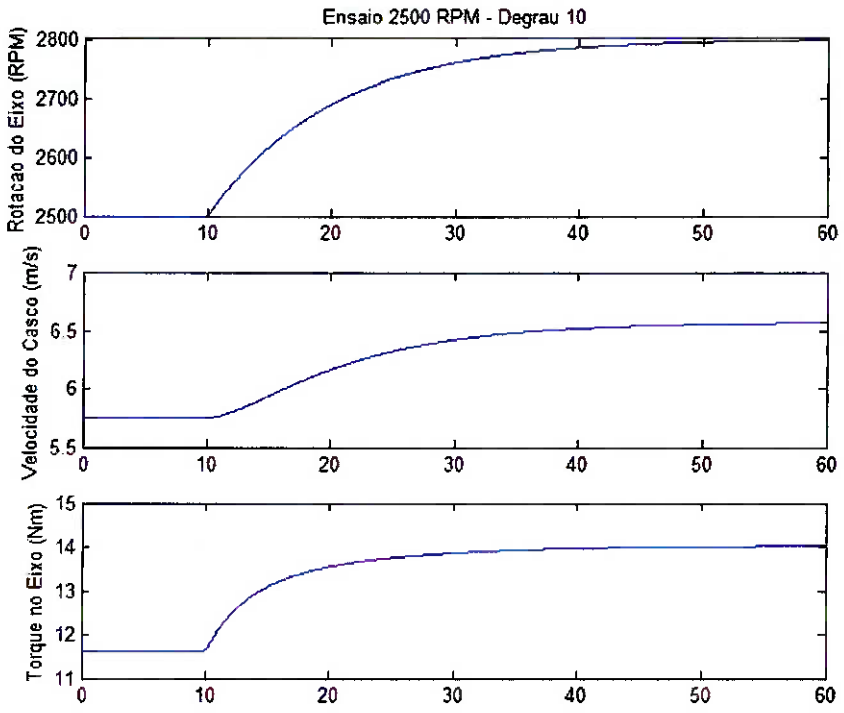


Figura 7.8 – Simulação 2500 RPM - Degrau 10

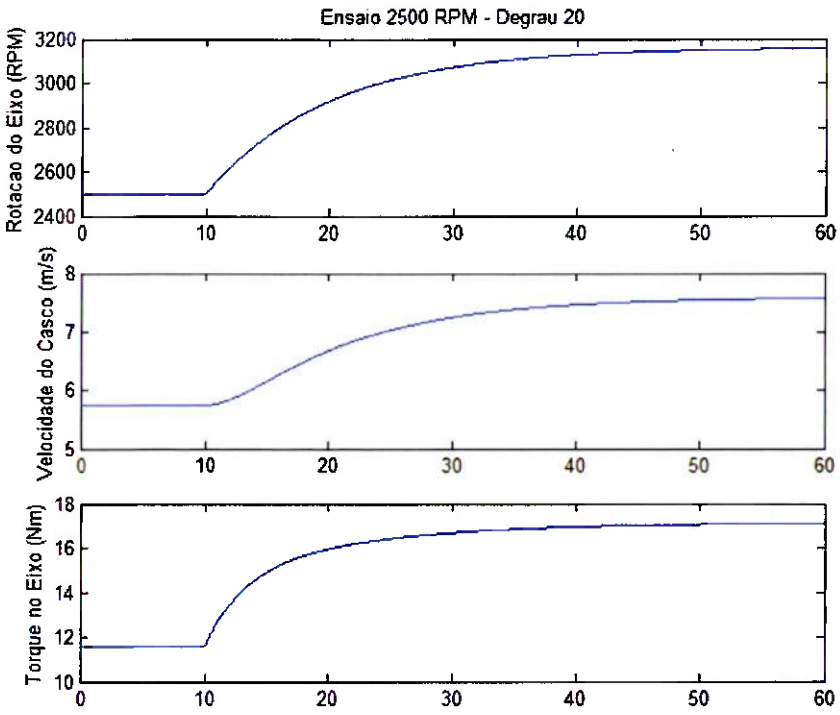


Figura 7.9 – Simulação 2500 RPM - Degrau 20

7.3.1.2.1. Modelos Corrigidos

Os modelos corrigidos para 2500 RPM podem ser visualizados a seguir:

$\frac{C(s)}{R(s)} = \frac{0,5}{10s+1}$, para o degrau de 10 passos

$\frac{C(s)}{R(s)} = \frac{0,55}{10s+1}$, para o degrau de 20 passos

7.3.1.3. Simulação a 3000 RPM

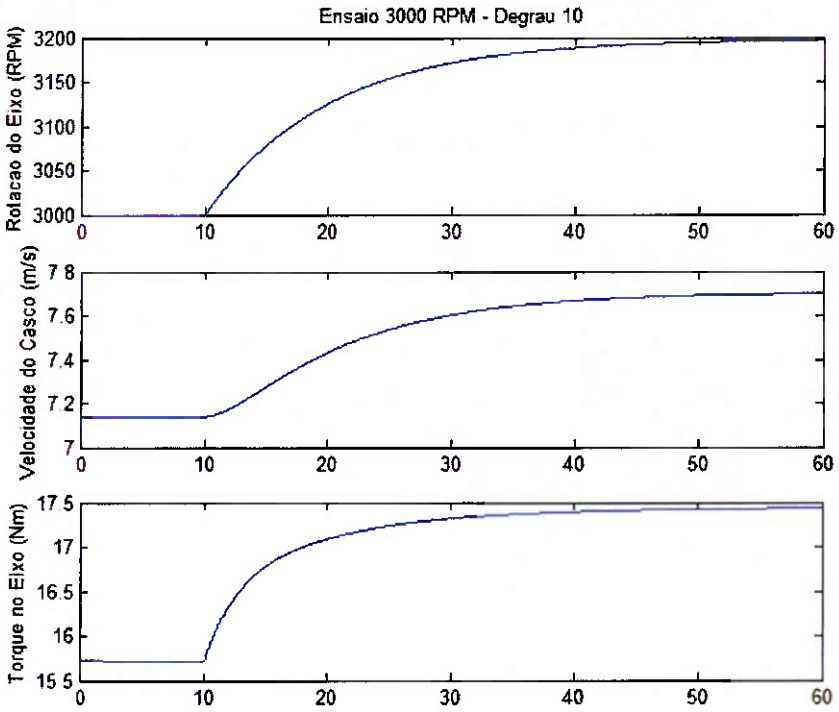


Figura 7.10 – Simulação 3000 RPM - Degrau 10

7.3.1.3.1. Modelo Corrigido

Neste último caso apenas uma simulação foi realizada e o modelo correspondente pode ser visto a seguir:

$$\frac{C(s)}{R(s)} = \frac{0{,}34}{10s+1}$$, para o degrau de 10 passos

7.3.2. Conclusões

Os gráficos dos modelos simulados refletem o efeito de uma variação de rotação no motor, a velocidade do barco acompanha a velocidade do motor com um atraso devido à inércia do sistema.

Os modelos corrigidos se comportaram como esperado e correspondem com os resultados obtidos durante a fase de ensaios.

7.4. Projeto do Controlador

Foi projetado um controlador com o intuito de reduzir o tempo de resposta do sistema a uma variação de abertura da borboleta. Esta simulação foi realizada apenas com dados gerados a partir de simulações anteriores e, portanto, não tem confirmação experimental. Entretanto é válida como experiência de projeto.

7.4.1. Controlador PID

A título de ilustração será apresentado o projeto de um PID para uma única condição de operação. Entretanto, o conceito pode ser entendido para todas a condições ensaiadas no presente trabalho.

7.4.1.1. Metodologia de Projeto – Alocação de Pólos

Um dos métodos possíveis para o projeto de um controlador PI em um sistema de primeira ordem é o da alocação de pólos. Suponhamos que a função de transferência do processo seja dada por:

$$G_w(s) = \frac{K_w}{Ts+1}$$

E o controlador seja:

$$H(s) = K\left(1+\frac{1}{T_i s}\right)$$

A equação característica para a malha fechada em estudo é dada por:

$$1 + G_w(s)H(s) = 0$$

Substituindo os valores de $G_w(s)$ e H(s), obtemos a seguinte expressão:

$$s^2 + s\frac{1+K_w K}{T} + \frac{K_p K}{TT_i} = 0$$

Sabemos que a equação característica de uma função de segunda ordem, dada em função da freqüência natural e do amortecimento é igual a:

$$s^2 + 2\xi\omega_n s + \omega_n{}^2 = 0$$

Comparando esta equação com a anterior, podemos isolar os valores de T_i e K em função de ω_n e ω, obtendo:

$$K = \frac{2\xi\omega_n T - 1}{K_p}$$

$$T_i = \frac{2\xi\omega_n T - 1}{\omega_n{}^2 T}$$

7.4.1.1.2. Controladores Estudados

Para descobrir que controladores podem ser utilizados, vamos verificar quais satisfazem o requisito de erro em regime estacionário nulo. O erro em regime estacionário é definido como sendo:

$$\lim_{s\to 0} sE(s)$$

Para podermos calcular este erro, precisamos antes equacionar o erro E(s). Observando a malha de controle, temos que:

$$E(s) = R(s) - Y(s)$$

Mas sabemos que a função de transferência da malha fechada é:

$$\frac{Y(s)}{R(s)} = \frac{G(s)H(s)}{1+G(s)H(s)} \Rightarrow Y(s) = R(s)\frac{G(s)H(s)}{1+G(s)H(s)}$$

Substituindo esta expressão acima e desenvolvendo, obtemos o seguinte resultado:

$$E(s) = R(s)\frac{1}{1+G(s)H(s)}$$

Para podermos calcular o erro em regime estacionário, vamos adotar os seguintes valores para G(s) e R(s):

- $G(s) = \frac{K_w}{Ts+1}$ (Sistema de Primeira Ordem)
- $R(s) = \frac{1}{s}$ (Entrada Degrau)

7.4.1.1.2.1. Controlador P

Neste caso, temos a seguinte função de transferência para o controlador:

$H(s) = K$

Substituindo este valor na expressão de E(s), podemos calcular o erro em regime:

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+\frac{KK_w}{Ts+1}}\right) = \lim_{s\to 0}\left(\frac{1}{1+\frac{KK_w}{Ts+1}}\right) = \frac{1}{1+KK_w}$$

Pois $Ts+1 \to 1$. Notamos a presença de erro em regime estacionário neste controlador.

7.4.1.1.2.2. Controlador PI

Analogamente ao procedimento acima:

$$H(s) = K\left(1+\frac{1}{T_i s}\right)$$

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+K\left(1+\frac{1}{T_i s}\right)\frac{K_w}{Ts+1}}\right) = \lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+KK_w\frac{T_i s+1}{TT_i s^2+T_i s}}\right) = 0$$

Pois $\frac{T_i s+1}{TT_i s^2+T_i s} \to \infty$. Não existe erro em regime estacionário neste controlador.

7.4.1.1.2.3. Controlador PD

Analogamente ao procedimento para controlador P:

$H(s) = K(1+T_d s)$

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+K(1+T_d s)\frac{K_w}{Ts+1}}\right) = \frac{1}{1+KK_w}$$

Pois $Ts+1 \to 1$ e $1+T_i s \to 1$. Notamos a presença de erro em regime estacionário neste controlador.

7.4.1.1.2.4. Controlador PID

Analogamente ao procedimento para controlador P:

$$H(s) = K\left(1+\frac{1}{T_i s}+T_d s\right)$$

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+K\left(1+\frac{1}{T_i s}+T_d s\right)\frac{K_w}{Ts+1}}\right) = 0$$

$$\lim_{s\to 0} s\frac{1}{s}\left(\frac{1}{1+KK_w\frac{T_i T_d s^2+T_i s+1}{TT_i s^2+T_i s}}\right) = 0$$

Pois $\frac{T_i T_d s^2+T_i s+1}{TT_i s^2+T_i s} \to \infty$. Não existe erro em regime estacionário neste controlador.

7.4.1.1.3. Conclusões

Verificamos que somente os controladores PI e PID podem ser utilizados, visto que somente estes satisfazem o critério de desempenho de não possuir erro de regime estacionário.

7.4.1.2. Projeto do Controlador PI

Para a escolha do controlador tentaremos encontrar um PI que satisfaça as especificações das características transitórias para degrau unitário. Caso não consigamos, tentaremos utilizar um controlador PID. Iremos realizar este projeto somente para o modelo obtido com a condição de 2500 RPM e degrau 10:

$$\frac{C(s)}{R(s)} = \frac{0,5}{10s+1}$$

Mas o mesmo procedimento pode ser adotado para os outros modelos. Inicialmente, realizamos uma simulação degrau unitário para o sistema motor, hélice e casco, a partir de uma condição estável em 2500RPM, mostrada abaixo:

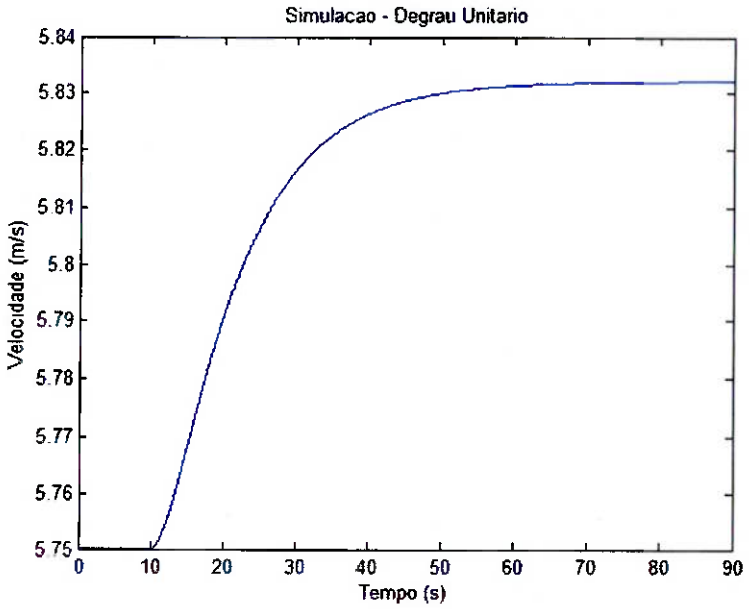


Figura 7.11 – Simulação do Sistema - Degrau unitário

Com base nesta simulação, obtivemos os parâmetros T=12,5 e K_w=0,05 para a planta. Para a construção do PI lançamos mão do método de alocação de pólos, descrito anteriormente neste trabalho. Deste método tiramos duas fórmulas importantes:

$$K = \frac{2\xi\omega_n T - 1}{K_w} \quad \text{(I)}$$

$$T_i = \frac{2\xi\omega_n T - 1}{\omega_n T} \quad \text{(II)}$$

Para o cálculo dos parâmetros K e T_i precisamos conhecer os valores de ω_n e ξ. Sabemos que:

$$M_p = \exp\left(\frac{-\pi\xi}{\sqrt{1-\xi^2}}\right) \quad \text{(III)}$$

$$t_s = \frac{4}{\xi\omega_n} \quad \text{(IV)}$$

Nossos requisitos de controle são

- $t_s < 5{,}0$ segundos
- $t_r < 20{,}0$ segundos
- $M_p < 5\%$

Escolhendo $M_p = 5\%$ obtivemos, a partir da equação III, $\xi = 0{,}48$. Da equação IV, obtemos $\omega_n = 1{,}67$ rad/s . Substituindo estes valores em I e II obtemos K=381 e T_i=6,61. Partindo destes valores iniciais, realizamos simulações experimentais de modo a escolher valores que produzissem resultados apropriados para nossas necessidades. Os valores escolhidos, após este procedimento foram K=200 e T_i=2.

7.4.1.2.1. Diagrama de Blocos incluindo o Controlador

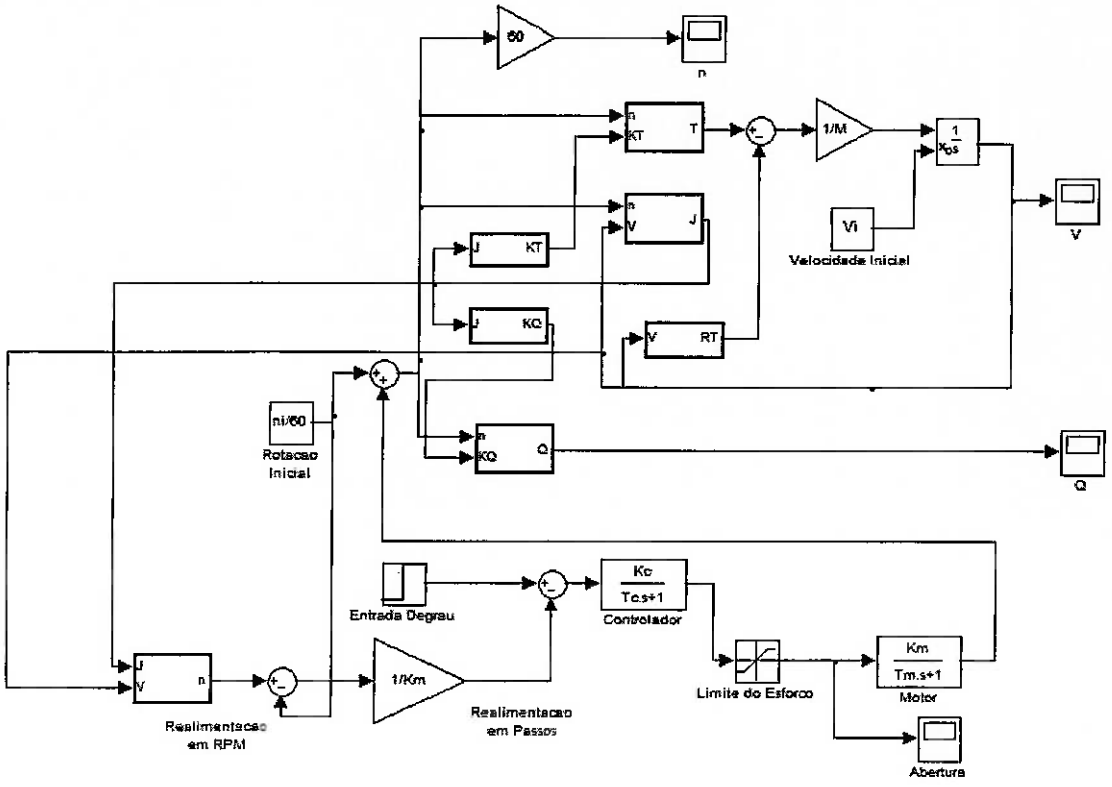


Figura 7.12 – Diagrama de blocos incluindo o controlador

7.2.1.2.2. Resultados

Os resultados do sistema simulado estão dispostos a seguir na forma de gráfico.

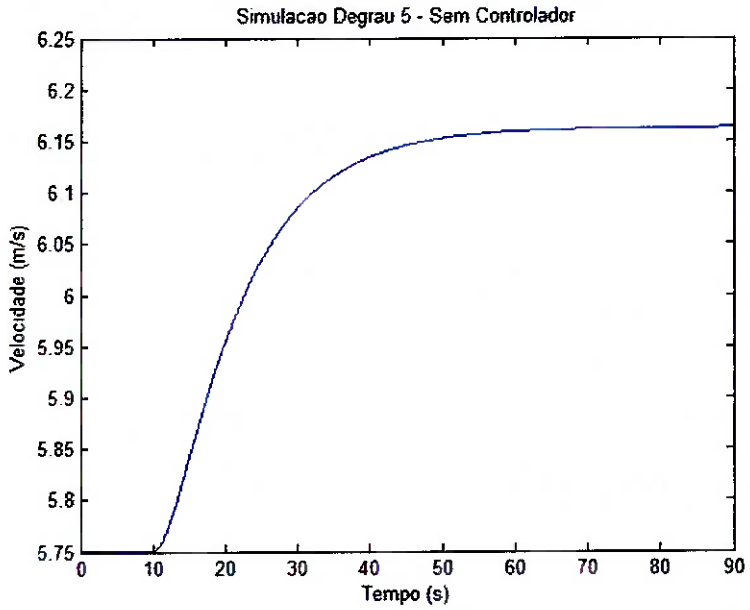


Figura 7.13 – Velocidade do barco – Sem controlador

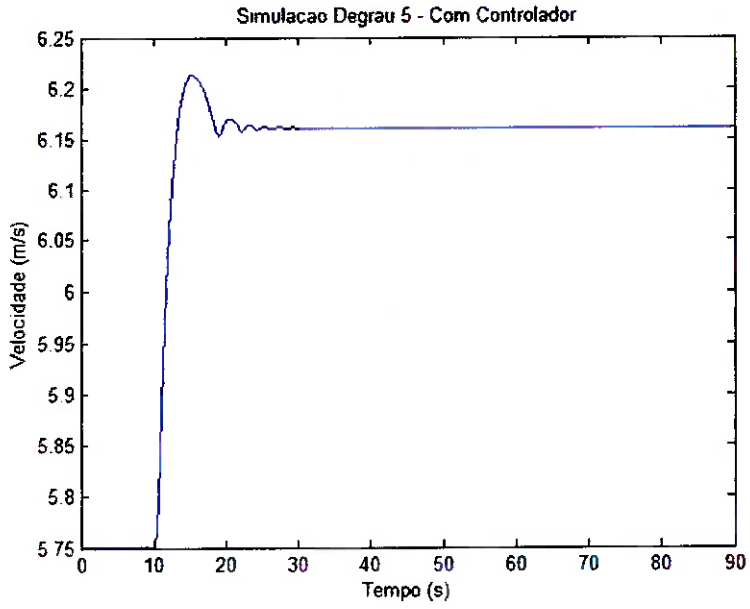


Figura 7.14 – Velocidade do Barco – Com controlador

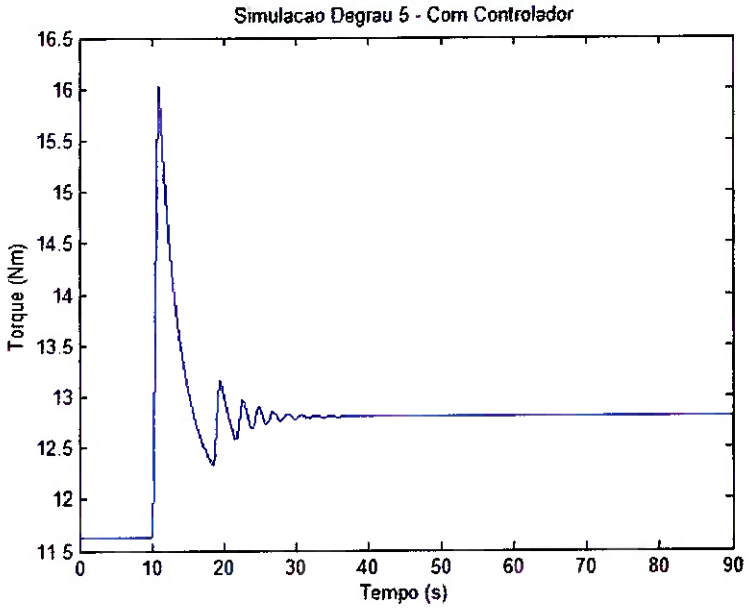


Figura 7.15 – Torque Requisitado – Com controlador

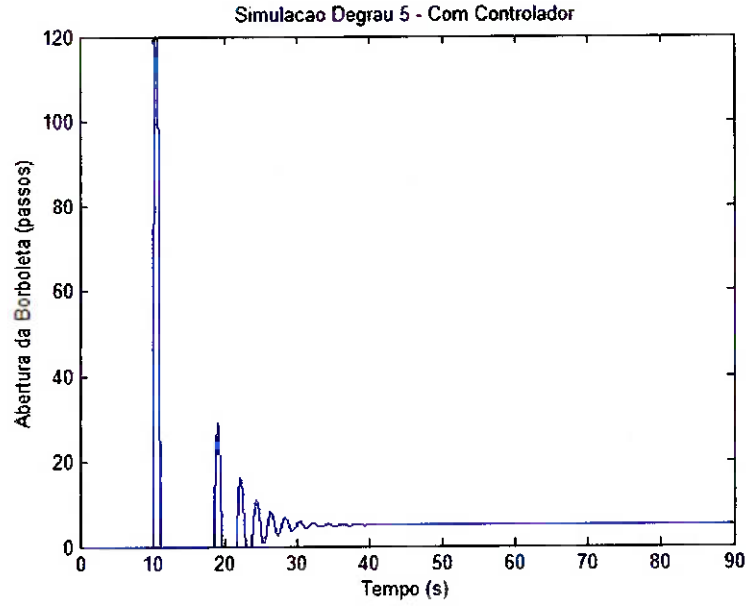


Figura 7.16 – Abertura da Borboleta – Com controlador

8. Conclusões

As metas deste trabalho compreendiam a seleção, adaptação, modelagem matemática, e aplicação de uma lei de controle de rotação a um MCI. Também foi assunto deste trabalho a interligação do MCI com um computador e o desenvolvimento do software necessário para operar este MCI remotamente através deste mesmo computador.

Durante a seleção do motor foram estudados diversos modelos, de um pequeno motor de aeromodelismo até motores de alta potência, como o que foi escolhido. Estávamos cientes que teríamos dificuldades extras devido à complexidade em se modelar um MCI, mas não nos deixamos desencorajar por este fato.

As adaptações realizadas no motor para permitir o seu controle remoto mostraram-se bastante complexas, com uma grande quantidade de alternativas a serem estudadas, e que consumiram grande parte do tempo que tínhamos disponível para este projeto. No entanto, deve-se ressaltar que esta foi uma das etapas de melhores resultados do projeto, e que permitiu a realização com sucesso dos ensaios do motor.

A confecção dos circuitos de comunicação, movimentação do motor de passo, fonte de alimentação e de leitura de rotação também não foi tarefa simples. Algumas das maiores dificuldades do projeto apareceram nesta etapa, em particular na identificação de erros na montagem do circuito e na interligação deste com o do sensor de rotação. No entanto, esta etapa também foi completada com sucesso.

Por fim, um terceiro grande sucesso foi obtido no projeto de um software de controle remoto do sistema, o qual apesar de ainda não estar completamente testado foi utilizado com sucesso nos ensaios do motor.

Tivemos sucesso apenas parcial em algumas outras atividades às quais nos dedicamos, tais como o estudo da modelagem matemática proposta por Lopes e Moskwa, que apesar de fornecerem modelos sofisticados e completos para um MCI, são de difícil compreensão e possuem alta complexidade de simulação e implementação.

Um ponto particularmente negativo foi o sensor de movimento adotado. Apesar de termos conseguirmos utilizá-lo nos ensaios, a qualidade dos dados fornecidos estava muito aquém do esperado, o que e impossibilitou a realização de ensaios mais detalhados do motor.

Finalizando, apesar de todos os problemas que tiveram de ser superados, e daqueles que ainda não estão completamente resolvidos, o trabalho produziu resultados significativos, e sem dúvida será de grande valia caso exista o interesse em dar continuidade aos estudos aqui iniciados. Várias etapas dos já projeto já estão completas, e aquelas que ainda necessitam de aperfeiçoamento já estão iniciadas e com um caminho claro para seu prosseguimento.

9. Bibliografia

01 - Lopes, J A; Um controlador preditivo generalizado (GPC) aplicado ao problema de controle da relação ar-combustível em motores ciclo Otto, operando com gás natural, com vistas em redução de emissões. São Paulo. 1996. Escola Politécnica da Universidade de São Paulo.

02 - Krutov, V I; Automatic Control of Internal Combustion Engines. Moscou. 1987. Mir Publishers.

03 - Heywood, J B; Internal Combustion Engines Fundamentals.1988. McGraw-Hill.

04 - Chow, A; Wyszynski, M L. Thermodynamic modeling of complete engine systems-a review. Birmingham. 1998. University of Birmingham.

05 - Hendricks, E; Sorenson, S C. SI engine controls and mean value engine modeling. SAE paper 910258. 1991.

06 - PC-104 Specification, Version 2.4, August 2001, Copyright 1992-2001, PC-104 Embebbed Consortium

07 - PC-104-Plus Specification, Version 1.2, August 2001, Copyright 1992-2001, PC-104 Embebbed Consortium

08 – Moskwa, J J; Hendrick, J K. Modeling and Validation of Automotive Engines for Control Algoithm Development. June 1992. Transactions of the ASME, vol. 114.

09 – Moskwa, J J. Automotive Engine Modeling for Real Time Control. Department of Mechanical Engeneering, MIT. Ph.D. thesis, 1998.

10 – Moskwa, J J; Hendrick, J K. Dynamic Fuel Parameter Estimation in Automotive Engines. Transportation Systems-1990. AMD vol. 108. 1990. ASME Winter Annual Meeting, Dallas, TX, November, 1990.

11 – Krten, Robert; Getting Started with QNX 4 – A Guide for Realtime Programmers. PARSE Software Devices

12 – Gajsk, Daniel D.; Principles of Digital Design. 1997. Prentice Hall

13 – Jones, L.; Chin, A.; Electronic Instruments and Measurements. 1991. Prentice Hall.

14 – Chang R T; "A modeling Study of the Influence of Spark Ignition Engine Design Parameters on Engine Thermal Efficiency and Performance"; MIT Department of Mechanical Engeneering, Sc.M. thesis, 1988.

15 – Sweet, Michael R.; Serial Programming Guide for POSIX Operating Systems.

16 – Ogata, K. Modern Control Engeneering. 3rd Edition. Prentice-Hall. 1997.

Apêndice A – Documentação das Classes

A.1. Classe Serial

A.1.1. Construtor

Chamada: Serial(char portstring[]).

Parâmetros:

- char portstring[]: Caminho do arquivo de dispositivo da porta serial, por exemplo "/dev/ser1".

Retorno: Nenhum.

Descrição: Obtém um file descriptor para a porta serial via comando "open", armazena a configuração atual da porta na variável "m_oldconfig", e configura a porta serial da forma apropriada para a comunicação com o sistema: 9600 bps, 8 bits de dados, sem paridade, 1 bit de parada, entrada e saída "raw", sem controle de fluxo, DTS e RTS em estado "low".

Notas: Nenhuma.

Melhorias Propostas: Verificar o retorno da função "open" e indicar possíveis erros em uma variável de status.

A.1.2. Destrutor

Chamada: ~Serial().

Parâmetros: Nenhum.

Retorno: Nenhum.

Descrição: Restaura a configuração original da porta serial e fecha o file descriptor via comando "close".

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.1.3. Write_Data

Chamada: write_data(char data[], int numbytes)

Parâmetros:

- char data[]: Buffer com os dados a serem enviado para a porta serial;
- int numbytes: Número de bytes a serem enviados para a porta.

Retorno: Nenhum.

Descrição: Envia o número de bytes requisitados para a porta serial.

Notas: Nenhuma.

Melhorias Propostas: Comparar o retorno da função "write" com o número de bytes requisitados para envio e indicar possíveis erros em uma variável de status.

A.1.4. Read_Data

Chamada: read_data(char data[], int numbytes)

Parâmetros:

- char data[]: Buffer para armazenamento dos dados recebidos;
- int numbytes: Número de bytes a serem lidos da porta.

Retorno: Nenhum.

Descrição: Lê o número de bytes requisitados da porta serial.

Notas: Nenhuma.

Melhorias Propostas: Comparar o retorno da função "read" com o número de bytes requisitados para leitura e indicar possíveis erros em uma variável de status.

A.1.5. Flush_Data

Chamada: flush_data(int option)

Parâmetros:

- int option = Buffers a serem limpos; 0 = Entrada, 1 = Saída, 2 = Ambos.

Retorno: Nenhum.

Descrição: Limpa os buffers indicados da porta serial.

Notas: Cuidado com o uso desta função, principalmente ao lidar com o buffer de saída. Um uso em hora imprópria pode ocasionar perda nos dados a serem enviados para o motor de passo.

Melhorias Propostas: Nenhuma.

A.1.6. Set_Config

Chamada: set_config(serialconfig config)

Parâmetros:

- serialconfig config = Estrutura contendo a configuração desejada para a porta serial.

Retorno: Nenhum.

Descrição: Configura a porta serial com os parâmetros informados, usando termios e as funções set_dtr e set_rst.

Notas: Nenhuma.

Melhorias Propostas: Verificar o retorno da função "tcsetattr", e indicar possíveis erros em uma variável de status.

A.1.7. Get_Config

Chamada: set_config(serialconfig config)

Parâmetros: Nenhum.

Retorno: struct serialconfig.

Descrição: Retorna uma estrutura do tipo serialconfig preenchida com a configuração atual da porta serial, utilizando termios e as funções get_dtr e get_rts.

Notas: Nenhuma.

Melhorias Propostas: Verificar o retorno da função "tcgetattr", e indicar possíveis erros em uma variável de status.

A.1.8. Set_Dtr

Chamada: set_dtr(int state)

Parâmetros:

- int state = Nível desejado para o sinal DTR (0 ou 1).

Retorno: Nenhum.

Descrição: Modifica o nível do sinal DTR.

Notas: Nenhuma.

Melhorias Propostas: Verificar a possibilidade de realizar este ajuste via Termios.

A.1.9. Set_Rts

Chamada: set_rts(int state)

Parâmetros:

- int state = Nível desejado para o sinal RTS (0 ou 1).

Retorno: Nenhum.

Descrição: Modifica o nível do sinal RTS.

Notas: Nenhuma.

Melhorias Propostas: Verificar a possibilidade de realizar este ajuste via Termios.

A.1.10. Get_Dtr

Chamada: get_dtr()

Parâmetros: Nenhum.

Retorno: int.

Descrição: Retorna o nível atual do sinal DTR.

Notas: Não implementado no momento.

Melhorias Propostas: Verificar a possibilidade de realizar este ajuste via Termios, caso contrário implementar via qnx_ioctl.

A.1.11. Get_Rts

Chamada: get_rts()

Parâmetros: Nenhum.

Retorno: int.

Descrição: Retorna o nível atual do sinal DTR.

Notas: Não implementado no momento.

Melhorias Propostas: Verificar a possibilidade de realizar este ajuste via Termios, caso contrário implementar via qnx_ioctl.

A.2. Classe Ticker

A.2.1. Construtor

Chamada: Ticker().
Parâmetros: Nenhum.
Retorno: Nenhum.
Descrição: Verifica o clock do sistema.
Notas: Sempre criar o objeto Ticker no começo do programa, de modo a minimizar a interferência de outros processos no cálculo do clock.
Melhorias Propostas: Nenhuma.

A.2.2. Destrutor

Chamada: ~Ticker().
Parâmetros: Nenhum.
Retorno: Nenhum.
Descrição: Não realiza nenhuma ação.
Notas: Nenhuma.
Melhorias Propostas: Nenhuma.

A.2.3. Get_Tickcount

Chamada: get_tickcount()
Parâmetros: Nenhum.
Retorno: unsigned int.
Descrição: Retorna o valor atual do contador de ciclos do processador.
Notas: Nenhuma.

Melhorias Propostas: Estudar a possibilidade de ler este valor diretamente do sistema operacional via estrutura "osdata".

A.2.4. Get_Elapsed_Time

Chamada: get_elapsed_time(unsigned int start, unsigned int end) Parâmetros: Nenhum.

Parâmetros:

- unsigned int start = Timestamp da posição inicial.
- unsigned int end = Timestamp da posição final.

Retorno: unsigned int.

Descrição: Retorna o tempo em segundos decorrido entre start e end.

Notas: O tempo decorrido máximo que pode ser calculado é igual ao limite do contador de ciclos (2^32-1), dividido pelo clock do computador.

Melhorias Propostas: Nenhuma.

A.2.5. Bench_Clockspeed

Chamada: bench_clockpeed()

Parâmetros: Nenhum.

Retorno: unsigned int.

Descrição: Calcula o clock do processador.

Notas: Este processo demora 5 segundos.

Melhorias Propostas: Estudar a possibilidade de ler este valor diretamente do sistema operacional via estrutura "osdata".

A.3. Classe Motor

A.3.1. Construtor

Chamada: Motor(Serial *pserial, int initposition, int maxposition);

Parâmetros:

- Serial *pserial: Objeto Serial relacionado com a porta aonde a interface do motor está conectada;
- int initposition: Abertura inicial do motor de passo, em passos;
- int maxposition: Abertura maxima permitida para o motor de passo, em passos.

Retorno: Nenhum.

Descrição: Posiciona o motor na posição inicial requisitada. Armazena o valor desta posição em m_initposition, e armazena o valor da posição máxima permitida em m_maxposition.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.3.2. Destrutor

Chamada: ~Motor().

Parâmetros: Nenhum.

Retorno: Nenhum.

Descrição: Não realiza nenhuma ação.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.3.3. Open_Steps

Chamada: open_steps(int steps)

Parâmetros:

- int steps: Abertura desejada, em passos;

Retorno: unsigned int.

Descrição: Ajusta a direção do motor de passo e aumenta a abertura da borboleta do motor.

Notas: Nenhuma.

Melhorias Propostas: Adicionar um valor de retorno de modo a informar quando foi não foi possível o aumento de abertura por termos atingido o limite máximo de passos.

A.3.4. Close_Steps

Chamada: close_steps(int steps)

Parâmetros:

- int steps: Fechamento desejado, em passos;

Retorno: unsigned int.

Descrição: Ajusta a direção do motor de passo e reduz a abertura da borboleta do motor.

Notas: Nenhuma.

Melhorias Propostas: Adicionar um valor de retorno de modo a informar quando foi não foi possível a redução de abertura por termos atingido o fechamento total da borboleta.

A.3.5. Get_Position

Chamada: get_position()

Parâmetros: Nenhum.

Retorno: int.

Descrição: Retorna a abertura atual do motor de passos, em passos.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.3.6. Get_Angle

Chamada: get_angle()

Parâmetros: Nenhum.

Retorno: double.

Descrição: Retorna a abertura atual do motor de passos, em graus.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.3.7. Initialize

Chamada: initialize()

Parâmetros: Nenhum.

Retorno: Nenhum.

Descrição: Coloca o motor na posição inicial indicada por m_initposition.

Notas: Este processo demora aproximadamente 3 segundos.

Melhorias Propostas: Estudar se existe a necessidade de tornar este método público. Se sim, modificá-lo de forma a garantir que a valor da variável m_position seja também reinicializado.

A.3.8. Set_Control

Chamada: set_control(int direction, int enable)

Parâmetros:

- int direction: Estado desejado para o sinal direction do motor de passo;
- int enable: Estado desejado para o sinal enable do motor de passo.

Retorno: Nenhum.

Descrição: Muda .

Notas: Função de "baixo nível" de controle do motor. Utiliza as seguintes constantes:

```
static const int Motor::controlsignal = 0x02;
static const int Motor::enablesignal = 0x04;
static const int Motor::directionsignal = 0x08;
```

Estas constantes estão relacionadas à montagem do circuito de interface do motor de passo. O bit 2 da UART (0x02) está ligado ao sinal de enable do dois flip-flops que armazenam os valores dos dois sinais de controle do motor, enable (que está ligada no bit 3, 0x04) e direction (que está ligado no bit 4, 0x08). Para reajustar

o valor destes sinais é necessário enviar o bit de controle juntamente com os bits dos sinais que deseja-se colocar em "high".
Melhorias Propostas: Adicionar variáveis com os valores atuais de enable e direction, e somente enviar mudanças via porta serial quando for realmente necessário.

A.3.9. Move_Steps

Chamada: move_steps(int steps)
Parâmetros:

- int steps: Número de passos a mover o motor de passos.

Retorno: Nenhum.
Descrição: Monta um array contendo a sequência de passos requisitada e envia para a porta serial.
Notas: Função de "baixo nível" de controle do motor. Utiliza a seguintes constante:

static const int Motor::stepsignal = 0x10;

Esta constante também está relacionada à montagem do circuito de interface do motor de passo, aonde o bit 5 da UART (0x10) está ligado diretamente ao sinal "step" do motor de passo.

É interessante notar que sempre que é enviado um sinal para a porta serial envia-se primeiro o byte desejado seguido de um byte "0". Isto é necessário devido a um defeito no circuito protótipo, aonde não existe a ligação entre os pinos 18 e 19 (DR e DDR). Sem esta ligação, o UART não retorna para a posição "0" automaticamente.

Melhorias Propostas: Nenhuma.

A.4. Classe Sensor

A.4.1. Construtor

Chamada: Sensor(Serial *pserial, Ticker *pticker);

Parâmetros:

- Serial *pserial: Objeto Serial relacionado com a porta aonde a interface do sensor está conectada;
- Ticker *pticker: Objeto Ticker a ser utilizado para cálculo do tempo;

Retorno: Nenhum.

Descrição: Preenche os valores de m_ticker e m_serial com os valores recebidos na chamada.

Notas: Nenhuma.

A.4.2. Destrutor

Chamada: ~Sensor().

Parâmetros: Nenhum.

Retorno: Nenhum.

Descrição: Não realiza nenhuma ação.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.4.3. Get_Frequency

Chamada: get_frequency(measure start, measure end)

Parâmetros:

- measure start: Leitura do sensor na posição inicial;
- measure end: Leitura do sensor na posição final.

Retorno: int.

Descrição: Retorna o valor atual da freqüência de rotação do eixo, em RPM.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.4.4. Get_Measure

Chamada: get_measure()

Parâmetros: Nenhum.

Retorno: struct measure.

Descrição: Lê um valor do sensor, e quebra ele em "número de voltas" e "posição". Anexa a esta leitura um timestamp.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

A.4.5. Get_Elapsed_Divisions

Chamada: get_elapsed_divisions(measure start, measure end);

Parâmetros:

- start: Posição inicial do sensor
- end: Posição final do sensor

Retorno: double.

Descrição: Retorna o número de divisões decorrido entre as duas medições.

Notas: Nenhuma.

Melhorias Propostas: Nenhuma.

Apêndice B – Listagem dos Programas

B.1. Sistema.hpp

```
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<termios.h>
#include<string.h>
#include<sys/qioctl.h>

#include "structs.hpp"
#include "logger.hpp"
#include "serial.hpp"
#include "ticker.hpp"
#include "motor.hpp"
#include "sensor.hpp"
```

B.2. Structs.hpp

```
typedef struct{
   long l,h;
} int64;

typedef struct
{
  tcflag_t cflag, oflag, iflag, lflag, qflag;
  speed_t ispeed, ospeed;
  int dtr, rts;
} serialconfig;

typedef struct{
  int laps;
  int position;
  unsigned int timestamp;
} measure;

typedef struct{
  int freq;
  double time;
} logdata;
```

B.3. Serial.hpp

```
class Serial
{
  //
  // --------------- Begin Public Members ---------------
  //
  public:
```

```
//
// Serial() -> Construtor, abre e configura porta,
//             inicializa m_fd e m_oldconfig
//
Serial(char portstring[]);

//
// ~Serial() -> Destrutor, fecha a porta e restaura a
//              configuracao original
//
~Serial();

//
// write_data -> Escreve bytes na serial
//
void write_data(char data[], int numbytes);

//
// read_data -> Le bytes da serial
//
void read_data(char data[], int numbytes);

//
// flush_data -> Limpa os buffers da serial
//
void flush_data(int option);

//
// --------------- Begin Private Members ---------------
//
private:

//
// m_fd -> Armazena file descriptor da serial
//
int m_fd;

//
// m_oldconfig -> Armazena configuracao original da porta
//
serialconfig m_oldconfig;

//
// set_config = Configura a porta de modo apropriado ao
//              nosso sistema
//
void set_config(serialconfig config);

//
// set_dtr -> Ajusta o status do sinal DTR
//
void set_dtr(int state);

//
// set_rts -> Ajusta o status do sinal RTS
//
void set_rts(int state);

//
```

```
  // get_config = Armazena a configuracao atual da porta
  //              em m_oldconfig
  //
  serialconfig get_config();

  //
  // get_dtr = Le o status do sinal DTR
  //
  int get_dtr();      // Nao Implementado

  //
  // get_rts = Le o status do sinal RTS
  //
  int get_rts();      // Nao Implementado
};
```

B.4. Serial.cpp

```
#include "sistema.hpp"

Serial::Serial(char portstring[])
{
  serialconfig config;

  // Abre a porta em modeo read/write, sem controle de terminal
  m_fd=open(portstring, O_RDWR | O_NOCTTY);

  // Armazena a configuracao anterior da porta
  m_oldconfig = get_config();

  // Copia a configuracao atual
  config.ispeed = m_oldconfig.ispeed;
  config.ospeed = m_oldconfig.ospeed;
  config.cflag = m_oldconfig.cflag;
  config.oflag = m_oldconfig.oflag;
  config.iflag = m_oldconfig.iflag;
  config.lflag = m_oldconfig.lflag;
  config.qflag = m_oldconfig.qflag;
  config.dtr = m_oldconfig.dtr;
  config.rts = m_oldconfig.rts;

  // Configura a porta
  config.ispeed=B9600;
  config.ospeed=B9600;

  // Paridade = none
  config.cflag &= ~PARENB;

  // Stop bits = 1
  config.cflag &= ~CSTOPB;

  // Data size = 8 bits
  config.cflag &=~CSIZE;
  config.cflag |= CS8;

  // Output = raw
```

```
  config.oflag &= ~OPOST;

  // Input = raw
  config.lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

  // Desprotege hardware flow control
  config.qflag &= ~TC_PROTECT_HFLOW;

  // Desprotege software flow control
  config.qflag &= ~TC_PROTECT_SFLOW;

  // Desliga hardware flow control
  config.cflag &= ~(IHFLOW | OHFLOW);

  // Desliga software flow control
  config.iflag &= ~(IXON | IXOFF);

  // Desliga RTS
  config.rts=0;

  // Desliga DTR
  config.dtr=0;

  // Configura a porta com os parametros selecionados
  set_config(config);

  // Limpa as entradas e saidas da serial
  flush_data(2);
}

Serial::~Serial()
{
  // Restaura a configuracao antiga
  set_config(m_oldconfig);

  // Fecha a porta
  close(m_fd);
}

void Serial::write_data(char data[], int numbytes)
{
  // Escreve os bytes na porta
  write(m_fd,data,numbytes);
}

void Serial::read_data(char data[], int numbytes)
{
  // Le os bytes da porta
  read(m_fd,data,numbytes);
}

void Serial::flush_data(int option)
{
  // Option:
  // 0 - Input
  // 1 - Output
  // 2 - Input and Output

  tcflush(m_fd,option);
```

```
}

void Serial::set_config(serialconfig config)
{
  struct termios options;

  // Le a estrutura com opcoes de comunicacao
  tcgetattr(m_fd,&options);

  // Configura o baud rate
  cfsetispeed(&options,config.ispeed);
  cfsetospeed(&options,config.ospeed);

  // Le as flags da configuracao nova
  options.c_cflag = config.cflag;
  options.c_oflag = config.oflag;
  options.c_iflag = config.iflag;
  options.c_lflag = config.lflag;
  options.c_qflag = config.qflag;

  // Aplica as modificacoes na configuracao
  tcsetattr(m_fd,TCSANOW,&options);

  // Configura sinal RTS
  set_rts(config.rts);

  // Configura sinal DTR
  set_dtr(config.dtr);
}

serialconfig Serial::get_config()
{
  struct termios options;
  serialconfig config;

  // Le a estrutura com opcoes de comunicacao
  tcgetattr(m_fd,&options);

  // Le o baud rate
  config.ispeed = cfgetispeed(&options);
  config.ospeed = cfgetospeed(&options);

  // Le as flags da configuracao
  config.cflag = options.c_cflag;
  config.oflag = options.c_oflag;
  config.iflag = options.c_iflag;
  config.lflag = options.c_lflag;
  config.qflag = options.c_qflag;

  // Le o sinal RTS
  config.rts = get_rts();

  // Le o sinal DTR
  config.dtr = get_dtr();

  return(config);
}

void Serial::set_dtr(int state)
```

```
{
unsigned long s[2];

// Ajusta os parametros de acordo com a opcao selecionada
if(state == 1) s[0] = 1L;
else s[0] = 0L;
s[1] = 1L;

qnx_ioctl(m_fd, QCTL_DEV_CTL, &s[0], 8, NULL,0);
}

void Serial::set_rts(int state)
{
unsigned long s[2];

// Ajusta os parametros de acordo com a opcao selecionada
if(state == 1) s[0] = 2L;
else s[0] = 0L;
s[1]=2L;

qnx_ioctl(m_fd, QCTL_DEV_CTL, &s[0], 8, NULL,0);
}

int Serial::get_dtr()
{
  // Nao Implementada
  return(0);
}

int Serial::get_rts()
{
  // Nao Implementada
  return(0);
}
```

B.5. Ticker.hpp

```
class Ticker
{
  //
  // ---------------- Begin Public Members ---------------
  //
  public:

  //
  // Ticker() -> Construtor, inicializa m_clockspeed
  //
  Ticker();

  //
  // ~Ticker() -> Destrutor
  //
  ~Ticker();

  //
  // get_tickcount() = Retorna o valor dos 32 bits mais
significativos
```

```
    //                    do contador de ciclos
    //
    unsigned int get_tickcount();

    //
    // get_elapsed_time() -> Retorna o tempo em segundos entre start e
end
    //
    double get_elapsed_time(unsigned int start, unsigned int end);

    //
    // --------------- Begin Private Members ---------------
    //
    private:

    //
    // m_clockspeed -> Armazena clock do processador em hertz
    //
    unsigned int m_clockspeed;

    //
    // numtestes -> Numero de testes a serem realizados de clock
    //
    static const int numtestes;

    //
    // maxuint -> Valor maximo de um unsigned int
    //
    static const unsigned int maxuint;

    //
    // Funcoes de acesso ao contador de ciclos do processador
    // Fonte: Exemplo "Micro_Time" disponivel no FTP da QNX
    //
    void rdtsc64(int64 *ptr);
    unsigned int rdtsc32(void);

    //
    // bench_clockspeed() = Retorna o numero de ciclos por segundo do
processador
    //
    unsigned int bench_clockspeed();
};
```

B.6. Ticker.cpp

```
#include "sistema.hpp"

const int Ticker::numtestes = 5;
const unsigned int Ticker::maxuint = 4294967295;

// Workaround para o Pragma não estar sendo reconhecido dentro de
classe
void read64(int64 *ptr);
#pragma aux read64 = "db 0fh,31h" "mov [ebx],eax" "mov [ebx+4],edx"
\
        parm nomemory [ebx] modify exact nomemory [eax edx];
```

```
unsigned int read32(void);
#pragma aux read32 = "db 0fh,31h" \
        parm nomemory [] modify exact nomemory [eax edx];

Ticker::Ticker()
{
  // Verifica o clock da máquina
  m_clockspeed = bench_clockspeed();
}

Ticker::~Ticker()
{
}

void Ticker::rdtsc64(int64 *ptr)
{
  read64(ptr);
}

unsigned int Ticker::rdtsc32(void)
{
  return(read32());
}

unsigned int Ticker::get_tickcount()
{
  int64 time64;
  unsigned int time;

  rdtsc64(&time64);
  time = rdtsc32();
  return(time);
}

unsigned int Ticker::bench_clockspeed()
{
  int i;
  unsigned int start, end, speed[numtestes], media;

  for(i=0;i<numtestes;i++)
  {
    // Conta o número de ciclos em um segundo
    start=get_tickcount();
    sleep(1);
    end=get_tickcount();

    // Calcula o tempo decorrido
    if(end>start)
      speed[i]=(end-start);
    else
      speed[i]=(end+(maxuint-start));
  }

  // Faz a media dos valores, descartando o primeiro e último
  media=0;
  for(i=1;i<numtestes-1;i++) media+=speed[i]/(numtestes-2);

  return(media);
```

```
}

double Ticker::get_elapsed_time(unsigned int start, unsigned int end)
{
  double elapsed;

  // Calcula o tempo decorrido em segundos
  if(end>=start)
     elapsed=((double)(end-start))/((double)m_clockspeed);
  else
     elapsed=((double)(end+(maxuint-start)))/((double)m_clockspeed);

  return(elapsed);
}
```

B.7. Motor.hpp

```
class Motor
{

  //
  // ---------------- Begin Public Members ---------------
  //
  public:

  //
  // Motor() -> Construtor, ajusta a posicao inicial do
  //            motor
  //
  Motor(Serial *pserial, int initposition, int maxposition);

  //
  // ~Serial() -> Destrutor
  //
  ~Motor();

  //
  // open_steps() -> Move o motor de modo a abrir a borboleta
  //                 o numero indicado de passos
  //
  void open_steps(int steps);

  //
  // close_steps() -> Move o motor de modo a fechar a borboleta
  //                  o numero indicado de passos
  //
  void close_steps(int steps);

  //
  // get_position() -> Le a abertura do motor em passos
  //
  int get_position();

  //
  // get_angle() -> Le a posicao do motor em graus
  //
  double get_angle();
```

```
  //
  // ---------------- Begin Private Members ----------------
  //
  private:

  //
  // m_position -> Posicao atual do motor
  //
  int m_position;

  //
  // m_initposition -> Abertura inicial do motor
  //
  int m_initposition;

  //
  // m_maxposition -> Abertura maxima do motor
  //
  int m_maxposition;

  //
  // Controles "low level" para controle do motor
  //
  static const int controlsignal,
                   enablesignal,
                   directionsignal,
                   stepsignal;

  //
  // Constantes "high level"  para controle do motor
  //
  static const int closedirection,
                   opendirection,
                   stepsperlap;

  //
  // m_serial -> Objeto para acesso a porta serial
  //
  Serial *m_serial;

  //
  // initialize() -> Executa a inicializacao do Motor, posicionando
ele
  //                 na posicao inicial indicada
  //
  void initialize();

  //
  // move_steps() -> Move o motor o numero indicado de passos
  //
  void move_steps(int steps);

  //
  // set_control() -> Ajusta os sinais de enable e direction
  //
  void set_control(int direction, int enable);

};
```

B.8. Motor.cpp

```
#include "sistema.hpp"

static const int Motor::controlsignal = 0x02;
static const int Motor::enablesignal = 0x04;
static const int Motor::directionsignal = 0x08;
static const int Motor::stepsignal = 0x10;

static const int Motor::closedirection = 1;
static const int Motor::opendirection = 0;
static const int Motor::stepsperlap = 1000;

Motor::Motor(Serial *pserial, int initposition, int maxposition)
{
  char data, *pdata;
  pdata = &data;

  m_initposition = initposition;
  m_maxposition = maxposition;
  m_position = 0;
  m_serial = pserial;

  // Inicializacao -> Limpa a saida da serial
  *pdata = 0;
  m_serial->write_data(pdata,1);

  // Posiciona o motor na abertura inicial
  initialize();
}

Motor::~Motor()
{
}

void Motor::initialize()
{
  // Inicializacao -> Fechar borboleta girando 360 graus
  set_control(closedirection,1);
  move_steps(stepsperlap);

  // Dar tempo do motor se posicionar
  sleep(3);

  // Inicializacao -> Ir ate a abertura inicial
  open_steps(m_initposition);
}

void Motor::open_steps(int steps)
{
  if(m_position+steps<=m_maxposition)
  {
    set_control(opendirection,1);
    move_steps(steps);
    m_position += steps;
  }
```

```
}

void Motor::close_steps(int steps)
{
  if(m_position-steps>=0)
  {
    set_control(closedirection,1);
    move_steps(steps);
    m_position -= steps;
  }
}

int Motor::get_position()
{
  return(m_position);
}

double Motor::get_angle()
{
  return((double)(360*m_position)/(double)stepsperlap);
}

void Motor::set_control(int direction, int enable)
{
   char data, *pdata;
   pdata = &data;

   // Envia o sinal de controle desejado
   *pdata = controlsignal;
   if(direction == 1) *pdata|=directionsignal;
   if(enable == 1) *pdata|=enablesignal;
   m_serial->write_data(pdata,1);

   // Limpa a saida
   *pdata = 0;
   m_serial->write_data(pdata,1);
}

void Motor::move_steps(int steps)
{
  char data[5000];
  int i,n;

  // Envia diversos comandos de passo/limpa
  for(i=0;i<steps;i++)
  {
    // Anda um passo
    data[2*i] = stepsignal;

    // Limpa a saida
    data[2*i+1] = 0;
  }

  m_serial->write_data(data,2*steps);
}
```

B.9. Sensor.hpp

```
class Sensor
{
  //
  // ---------------- Begin Public Members ---------------
  //
  public:

  //
  // Sensor() -> Construtor
  //
  Sensor(Serial *pserial, Ticker *pticker);

  //
  // ~Sensor() -> Destrutor
  //
  ~Sensor();

  //
  // get_frequency() -> Retorna a frequencia em RPM do motor
  //
  int get_frequency(measure start, measure end);

  //
  // get_measure() -> Le o status atual do sensor
  //
  measure get_measure();

  //
  // ---------------- Begin Private Members ---------------
  //
  private:

  //
  // maxlaps -> Numero maximo de voltas do "encoder"
  //
  static const int maxlaps;

  //
  // maxdivisions -> Numero de divisoes por volta do "encoder"
  //
  static const int maxdivisions;

  //
  // datasignal -> Sinal enviado para a requisicao de dados da
serial
  //
  static const int datasignal;

  //
  // Mascaras para interpretacao do sinal da serial
  //
  static const int lapmask,
                   lapdivisor,
                   posmask,
                   colormask;

  //
```

```
  // m_serial -> Objeto para acesso a serial
  //
  Serial *m_serial;

  //
  // m_ticker -> Objeto para acesso ao contador de ciclos
  //
  Ticker *m_ticker;

  //
  // get_elapsed_divisions() -> Retorna o numero de divisoes entre
start e end
  //
  int get_elapsed_divisions(measure start, measure end);
};
```

B.10. Sensor.cpp

```
#include "sistema.hpp"

static const int Sensor::maxlaps = 4;
static const int Sensor::maxdivisions = 64;
static const int Sensor::datasignal = 0x01;
static const int Sensor::lapmask = 0x60;
static const int Sensor::lapdivisor = 32;
static const int Sensor::posmask = 0x1F;
static const int Sensor::colormask = 0x80;

Sensor::Sensor(Serial *pserial, Ticker *pticker)
{
  char request, *prequest;
  prequest = &request;

  m_ticker = pticker;
  m_serial = pserial;

  // Limpa a saida da serial
  *prequest = 0;
  m_serial->write_data(&request,1);
}

Sensor::~Sensor()
{
}

int Sensor::get_frequency(measure start, measure end)
{
   double elapsed,divisions,freq;

   elapsed=m_ticker->get_elapsed_time(start.timestamp,end.timestamp);
   divisions=(double)get_elapsed_divisions(start,end);

   freq=60*(divisions/maxdivisions)/elapsed;

   return((int)freq);
}
```

```
measure Sensor::get_measure()
{
  measure data;
  char request, position, *prequest, *pposition;

  prequest = &request;
  pposition = &position;

  // Limpa os buffers de entrada, por seguranca
  m_serial->flush_data(0);

  // Le o timestamp para o dado atual
  data.timestamp = m_ticker->get_tickcount();

  // Envia a requisicao de dados
  *prequest = datasignal;
  m_serial->write_data(&request,1);
  *prequest = 0;
  m_serial->write_data(&request,1);

  // Lê um byte de dados
  m_serial->read_data(&position,1);

  // Extrai o numero de voltas
  data.laps=(lapmask & *pposition)/lapdivisor;

  // Extrai a posicao atual
  data.position=(posmask & *pposition);

  // Corrige a posicao de modo a considerar o bit menos
  // significativo, que nao e lido pelo encoder
  data.position=2*data.position;
  if((colormask & *pposition)==0) data.position++;

  return(data);
}

int Sensor::get_elapsed_divisions(measure start, measure end)
{
  int divisions, laps;

  // Checa quantas voltas passaram
  if(end.laps>=start.laps)
    laps = end.laps - start.laps;
  else
    laps = end.laps + (maxlaps-start.laps);

  // Verifica se a ultima volta esta completa, caso contrario
desconta ela
  if(start.position>end.position) laps--;

  // Checa quantas divisoes foram decorridas
  if(end.position>=start.position)
    divisions = end.position - start.position;
  else
    divisions = end.position + (maxdivisions - start.position);

  // Adiciona o efeito das voltas
  divisions += laps * maxdivisions;
```

```
  return(divisions);
}
```

B.11. Logger.hpp

```
class Logger
{
  //
  // ---------------- Begin Public Members ---------------
  //
  public:

  //
  // Logger() -> Construtor, inicializa o buffer
  //
  Logger(char *filename, int maxrecords);

  //
  // ~Logger() -> Salva os dados em arquivo
  //
  ~Logger();

  //
  // add_data = Adiciona um valor ao buffer
  //
  void add_data(int freq, double time);

  //
  // ---------------- Begin Private Members ---------------
  //
  private:

  //
  // m_filename -> Armazena o nome do arquivo
  //
  char *m_filename;

  //
  // m_itemcount -> Numero de items armazenados
  //
  int m_itemcount;

  //
  // m_buffer -> Buffer aonde serao armazenados os dados
  //
  logdata *m_buffer;

  //
  // dtoa() -> Helper, converte double para string
  //
  char* dtoa(double number, char* buffer, char* buffer2);

};
```

B.12. Logger.cpp

```
#include "sistema.hpp"

Logger::Logger(char *filename, int maxrecords)
{
  m_itemcount=0;

  // Aloca o buffer para armazenamento em memoria dos dados
  m_buffer = (logdata*)malloc(maxrecords*sizeof(logdata));

  // Copia o filename para a variavel local
  m_filename = (char*)malloc((strlen(filename)+1)*sizeof(char));
  strcpy(m_filename,filename);
}

Logger::~Logger()
{
  int i, fd;
  char buffer[50], buffer2[50];

  // Abre o arquivo
  fd=open(m_filename,O_RDWR | O_CREAT);

  // Escreve um cabecalho no arquivo
  write(fd,"Rotacao (RPM);Tempo (s)",10);
  write(fd,"\n",1);

  // Escreve os dados do buffer no arquivo
  for(i=0;i<m_itemcount;i++)
  {

    // Rotacao
      itoa(m_buffer[i].freq,buffer,10);
      write(fd,buffer,strlen(buffer));
    write(fd,";",1);

      // Tempo
    dtoa(m_buffer[i].time,buffer,buffer2);
      write(fd,buffer,strlen(buffer));
    write(fd,"\n",1);
  }

  // Fecha o arquivo
  close(fd);

  free(m_buffer);
}

void Logger::add_data(int freq, double time)
{
  // Adiciona um registro novo no buffer
  m_buffer[m_itemcount].freq = freq;
  m_buffer[m_itemcount].time = time;

  // Incrementa o contador de registros
  m_itemcount++;
}
```

```
char* Logger::dtoa(double number, char *buffer, char *buffer2)
{
  int precomma, postcomma;

  // Quebra o numero em valores antes e depois da virgula
  precomma = (int)number;
  postcomma = (number-precomma)*100000000;

  // Monta o string com o numero
  itoa(precomma,buffer,10);
  itoa(postcomma,buffer2,10);
  strcat(buffer,",");
  strcat(buffer,buffer2);

  return(buffer);
}
```

B.13. Ensaio.cpp

```
#include "sistema.hpp"
#include <signal.h>
#include <time.h>
#include <sys/proxy.h>
#include <sys/kernel.h>
#include <stdio.h>

int main()
{
Serial *oSerial;
Ticker *oTicker;
Motor  *oMotor;
Sensor *oSensor;
Logger *oLog;
int initposition, maxposition, maxrecords, sampletime, steps,
    pidsender, messagedata, pidmotor, pidsensor, tidmotor,
      tidsensor, receivecount, lastsensor, opcao, fexit, freq;
double totaltime;
char filename[50];
sigevent eventmotor, eventsensor;
itimerspec timermotor, timersensor;
timespec readtime;
measure start, end;

//
// Parametros do sensor
//
sampletime=50;

//
// Parametros do motor
//
initposition=20;
maxposition=225;

//
// Parametros do logger
//
```

```
maxrecords = 1000;

//
// Flags de controle de fluxo do programa
//
fexit=0;

//
// Variaveis extras para exibicao no menu
//
lastsensor = 0;

oSerial = new Serial("/dev/ser1");
oTicker = new Ticker();
oMotor = new Motor(oSerial,initposition,maxposition);
oSensor = new Sensor(oSerial,oTicker);

while(fexit==0)
{
  printf("Programa de Ensaios do Motor - Versao 1.2\n");
  printf("Filipe Badaro and Tomas D'Agostino\n");
  printf("\nMenu Principal\n\n");
  printf("Controle do Motor:\n");
  printf("1) Aumentar abertura n passos\n");
  printf("2) Diminuir abertura n passos\n\n");
  printf("Controle do Sensor:\n");
  printf("3) Mudar periodo de amostragem\n");
  printf("4) Realizar leitura\n\n");
  printf("Ensaios:\n");
  printf("5) Realizar ensaio degrau\n\n");
  printf("Outros:\n");
  printf("6) Sair\n\n");
  printf("Posicao Atual do Motor de Passo: %d passos, %f
graus\n",oMotor->get_position(),oMotor->get_angle());
  printf("Ultima Leitura do Sensor: %d RPM\n",lastsensor);
  printf("Periodo de amostragem: %d ms\n\n",sampletime);
  printf("\n");
  printf("Opcao: ");
  scanf("%d",&opcao);

  switch(opcao)
  {
    case 1:
      printf("Digite o numero de passos: ");
      scanf("%d",&steps);
      oMotor->open_steps(steps);
      break;

    case 2:
      printf("Digite o numero de passos: ");
      scanf("%d",&steps);
      oMotor->close_steps(steps);
      break;

    case 3:
      printf("Digite o novo periodo de amostragem (ms): ");
      scanf("%d",&sampletime);
      break;
```

```
case 4:
  // Ajusta o periodo de amostragem
  readtime.tv_sec=0;
  readtime.tv_nsec=sampletime*1000000;

  // Le o valor inicial
  start = oSensor->get_measure();

    // Aguarda a passagem de um periodo de amostragem
  nanosleep(&readtime, NULL);

  // Le o valor final
  end = oSensor->get_measure();

  // Calcula a rotacao
  lastsensor=oSensor->get_frequency(start,end);

  break;

case 5:

  printf("Digite o degrau desejado: ");
  scanf("%d",&steps);
  printf("Digite o nome do arquivo: ");
  scanf("%s",filename);

  oLog = new Logger(filename,maxrecords);

  // Cria a mensagem proxy do motor
  messagedata = 1;
    pidmotor = qnx_proxy_attach(0,&messagedata,sizeof(int),10);

    // Configura o evento do motor
    eventmotor.sigev_signo = -pidmotor;

  // Cria o timer do motor
  tidmotor = timer_create(CLOCK_REALTIME,&eventmotor);

  // Configura o timer do motor para um disparo em 5s
  timermotor.it_value.tv_sec=10;
  timermotor.it_value.tv_nsec=0;
  timermotor.it_interval.tv_sec=0;
  timermotor.it_interval.tv_nsec=0;

  // Cria a mensagem proxy do sensor
  messagedata = 2;
    pidsensor = qnx_proxy_attach(0,&messagedata,sizeof(int),10);

    // Configura o evento do sensor
    eventsensor.sigev_signo = -pidsensor;

  // Cria o timer do sensor
  tidsensor = timer_create(CLOCK_REALTIME,&eventsensor);

  // Configura o timer do sensor para um disparo a cada
    // periodo de amostragem
  timersensor.it_value.tv_sec=0;
  timersensor.it_value.tv_nsec=sampletime*1000000;
  timersensor.it_interval.tv_sec=0;
```

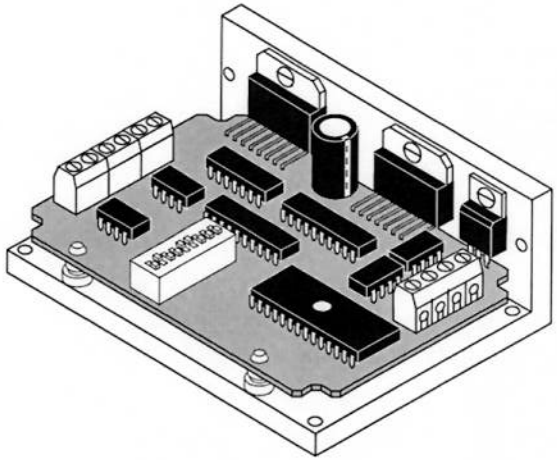
```
        timersensor.it_interval.tv_nsec=sampletime*1000000;

          // Ativa os timers
          timer_settime(tidmotor,0,&timermotor,NULL);
        timer_settime(tidsensor,0,&timersensor,NULL);

          receivecount = 0;
          totaltime=0;

          // Le a posicao inicial para medicao
        start = oSensor->get_measure();

          do{
          pidsender = Receive(0,&messagedata,sizeof(int));

            if(pidsender==pidmotor)
            {
            oMotor->open_steps(steps);
            }
            if(pidsender==pidsensor)
            {
                // Le a posicao final para medicao
            end = oSensor->get_measure();

                // Calcula a rotacao e o tempo decorrido
            freq=oSensor->get_frequency(start,end);
            totaltime+=oTicker-
>get_elapsed_time(start.timestamp,end.timestamp);

                // Posicao final = nova posicao inicial
                start.laps=end.laps;
            start.position=end.position;
            start.timestamp=end.timestamp;

              // Adiciona um registro no contador
              receivecount++;

            // Escreve no log
            oLog->add_data(freq,totaltime);
            }
        }while(receivecount<maxrecords);

        // Remove os timers
        timer_delete(tidmotor);
        timer_delete(tidsensor);

        delete(oLog);
        break;

      case 6:
        fexit=1;
        break;
    }
  }

  delete(oSerial);
  delete(oTicker);
  delete(oMotor);
  delete(oSensor);
```

```
return(0);
}
```

User's Manual

3540 M

Step Motor Driver



Applied Motion Products, Inc.
404 Westridge Drive • Watsonville, CA 95076
Tel (408) 761-6555 (800) 525-1609 Fax (408) 761-6544

Applied Motion Products
motors • drives • controls

Introduction

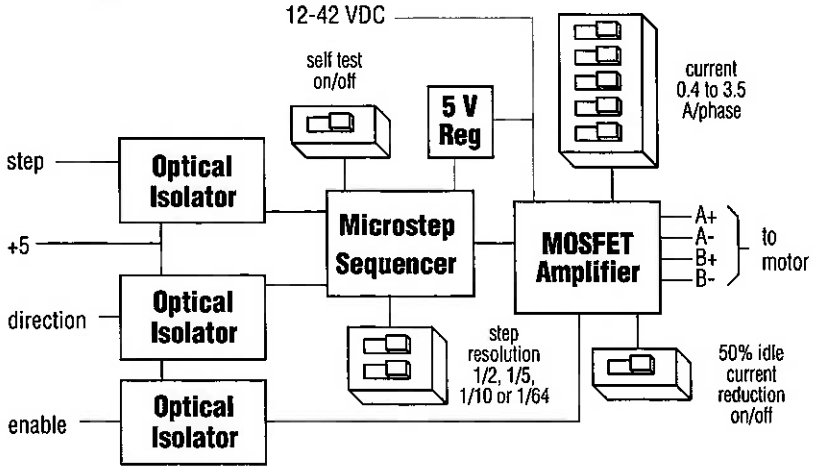
Thank you for selecting an Applied Motion Products motor control. We hope our dedication to performance, quality and economy will make your motion control project successful.

If there's anything we can do to improve our products or help you use them better, please call or fax. We'd like to hear from you. Our phone number is (800) 525-1609 or you can reach us by fax at (408) 761-6544.

Features

- Drives sizes 14 through 34 step motors
- Pulse width modulation, MOSFET 3 state switching amplifiers
- Phase current from 0.4 to 3.5 amps (switch selectable, 32 settings)
- Optically isolated step, direction and enable inputs
- Half, 1/5, 1/10, 1/64 step (switch selectable)
- Automatic 50% idle current reduction (can be switched off)

Block Diagram



Technical Specifications

Amplifiers Dual, bipolar MOSFET H-bridge, pulse width modulated three state switching at 20kHz. 12-42 VDC input. 0.4 - 3.5 amps/phase output current, switch selectable in 0.1 A increments. 122 watts maximum output power. Automatic idle current reduction (switch selectable), reduces current to 50% of setting after one second.

Inputs Step, direction and enable, optically isolated, 5V logic. 5mA/signal, sink requirement. Motor steps on rising edge of step input. 0.5 µsec minimum pulse width. 2 µsec minimum set up time for direction signal.

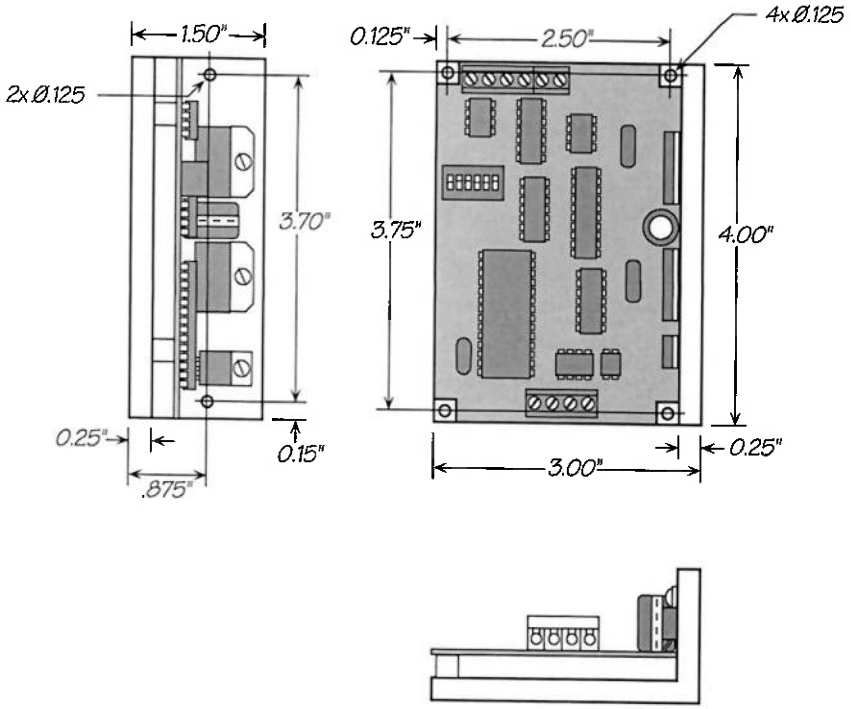
Physical Mounted on 1/4 inch thick black anodized aluminum heat transfer chassis. 1.5 x 3.0 x 4.0 inches overall. Power on red LED. See drawing on page 14 for more information. Maximum chassis temperature: 70° C.

Connectors European style screw terminal blocks. Max wire size: AWG 18.
Motor: 4 position (A+, A-, B+, B-)
Signal Input: 4 position (+5, STEP, DIR, EN)
DC Input: 2 position (V+, V-)

Self Test Switch selectable, rotates motor 1/2 revolution each direction at 100 steps/second, half step mode.

Microstepping Four switch selectable step resolutions. With 1.8¡ motor:
Half step (400 steps/rev)
1/5 step (1000 s/r)
1/10 step (2000 s/r)
1/64 step (12,800 s/r)
Other resolutions, up to 12,800, available to qualified OEMs upon request.

Mechanical Outline

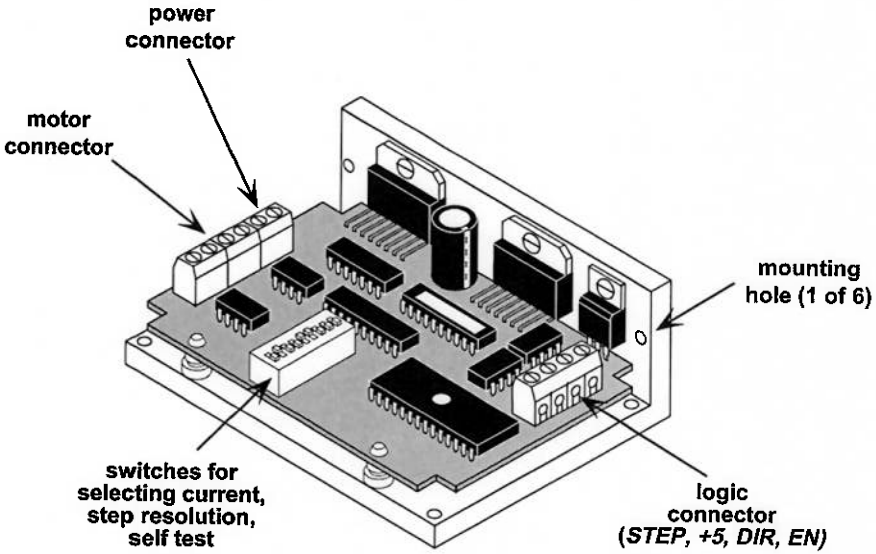


Getting Started

To use your Applied Motion Products motor control, you will need the following:

- a 12-42 volt DC power supply for the motor. Please read the section entitled *Choosing a Power Supply* for help in choosing the right power supply.
- +5 volts DC, 15mA to activate the optoisolation circuits (if you don't use 5 volt logic, see page 6.) This is provided by most indexers and PLCs.
- a source of step pulses capable of sinking at least 5 mA
- if your application calls for bidirectional rotation, you'll also need a direction signal, capable of sinking 5 mA
- a compatible step motor
- a small flat blade screwdriver for tightening the connectors

The sketch below shows where to find the important connection and adjustment points. Please examine it now.

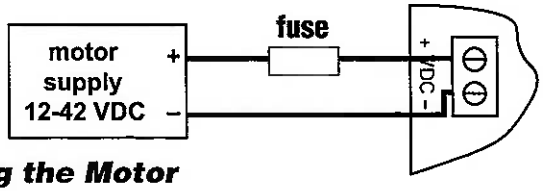


Connecting the Power Supply

If you need information about choosing a power supply, please read *Choosing a Power Supply* located on page 12 of this manual. The PS430 from Applied Motion Products is a good supply for this drive.

If your power supply does not have a fuse on the output or some kind of short circuit current limiting feature you need to put a 4 amp fast acting fuse between the drive and power supply. Install the fuse on the + power supply lead.

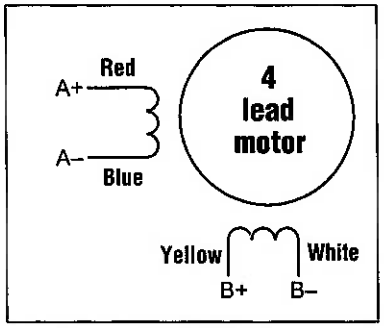
Connect the motor power supply + terminal to the driver terminal labeled "+ VDC". Connect power supply Ð to the drive terminal labeled "VDC Ð". Use no smaller than 20 gauge wire. **Be careful not to reverse the wires.** Reverse connection will destroy your driver, void your warranty and generally wreck your day.



Connecting the Motor

Warning: When connecting the motor to the driver, be sure that the motor power supply is off. Secure any unused motor leads so that they can't short out to anything. Never disconnect the motor while the drive is powered up. Never connect motor leads to ground or to a power supply!

You must now decide how to connect your motor to the drive.



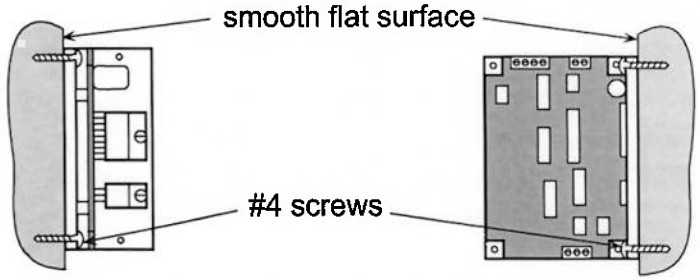
4 Leads

Four lead motors can only be connected one way. Please follow the sketch at the right.

Six lead motors can be connected in series or center tap. In series mode, motors produce more torque at low speeds, but cannot run as fast as in the center tap configuration. In series operation, the motor should be operated at 30% less than rated current to prevent overheating. Wiring diagrams for both connection methods are shown on the next page. NC means not connected to anything.

Mounting the Drive

You can mount your drive on the wide or the narrow side of the chassis. If you mount the drive on the wide side, use #4 screws through the four corner holes. For narrow side mounting applications, you can use #4 screws in the two side holes.



wide side mount narrow side mount

The amplifiers in the drive generate heat. Unless you are running at 1 amp or below, you may need a heat sink. To operate the drive continuously at maximum power you must properly mount it on a heat sinking surface with a thermal constant of no more than 4°C/watt. Applied Motion Products can provide a compatible heat sink. Often, the metal enclosure of your system will make an effective heat sink.

Never use your drive in a space where there is no air flow or where other devices cause the surrounding air to be more than 70 °C. Never put the drive where it can get wet or where metal particles can get on it.

Choosing a Power Supply

Voltage

Chopper drives work by switching the voltage to the motor terminals on and off while monitoring current to achieve a precise level of phase current. To do this efficiently and silently, youÕll want to have a power supply with a voltage rating at least five times that of the motor. Depending on how fast you want to run the motor, you may need even more voltage. More is better, the only upper limit being the maximum voltage rating of the drive itself: 42 volts (including ripple).

If you choose an unregulated power supply, do not exceed 30 volts DC. This is because unregulated supplies are rated at full load current. At lesser loads, like when the motor is not moving, the actual voltage can be up to 1.4 times the voltage list on the power supply label.

Current

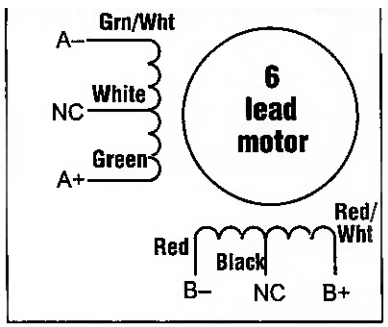
The maximum supply current you will need is the sum of the two phase currents. However, you will generally need a lot less than that, depending on the motor type, voltage, speed and load conditions. That's because the 3540 M uses switching amplifiers, converting a high voltage and low current into lower voltage and higher current. The more the power supply voltage exceeds the motor voltage, the less current youÕll need from the power supply.

We recommend the following selection procedure:

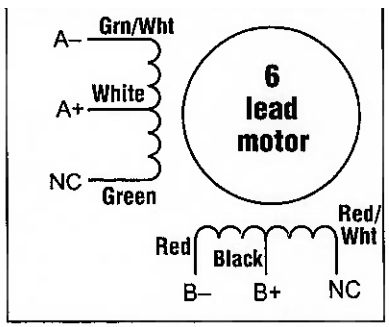
1. If you plan to use only a few drives, get a power supply with at least twice the rated phase current of the motor.
2. If you are designing for mass production and must minimize cost, get one power supply with more than twice the rated current of the motor. Install the motor in the application and monitor the current coming out of the power supply and into the drive at various motor loads. This will tell you how much current you really need so you can design in a lower cost power supply.

If you plan to use a regulated power supply you may encounter a problem with current foldback. When you first power up your drive, the full current of both motor phases will be drawn for a few milliseconds while the stator field is being established. After that the amplifiers start chopping and much less current is drawn from the power supply. If your power supply thinks this initial surge is a short circuit it may ÒfoldbackÓ to a lower voltage. With many foldback schemes the voltage returns to normal only after the first motor step and is fine thereafter. In that sense, unregulated power supplies are better. They are also less expensive.

The PS430 from Applied Motion Products is a good supply to use with the 3540 M.

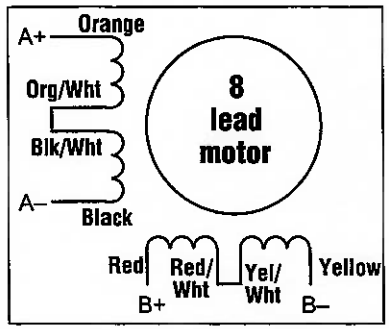


6 Leads Series Connected

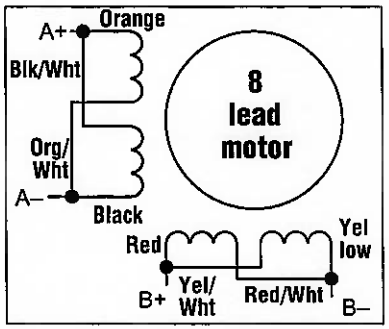


6 Leads Center Tap Connected

Eight lead motors can also be connected in two ways: series or parallel. As with six lead motors, series operation gives you more torque at low speeds and less torque at high speeds. In series operation, the motor should be operated at 30% less than the rated current to prevent over heating. The wiring diagrams for eight lead motors are shown below.



8 Leads Series Connected



8 Leads Parallel Connected

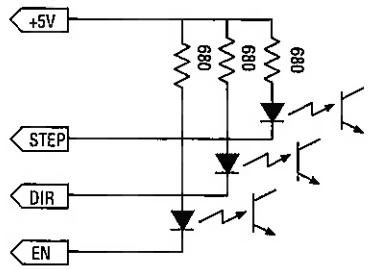
Connecting Logic

The 3540 M contains optical isolation circuitry to prevent the electrical noise inherent in switching amplifiers from interfering with your circuits. Optical isolation is accomplished by powering the motor driver from a different supply than your circuits. There is no electrical connection between the two: signal communication is achieved by infrared light. When your circuit is in the logic low state (near 0 volts), it is directing electrical current through an LED that is built into the drive. The LED, in turn, produces infrared light which turns on a phototransistor that is wired to the brains of the drive. When your circuit is in the logic high state, the LED and phototransistor turn off.

A schematic diagram of the input circuit is shown below.

You must supply 5 volts DC to supply current to the LEDs on the input side of the optoisolators. The maximum current draw is 15 mA total.

Your controlling logic must be capable of sinking at least 5 mA to control each drive input. Most CMOS and open collector TTL devices are directly compatible with this drive. Logic low, or 0, for a given input occurs when that input is pulled to less than 0.8 volts DC. In this state the LED is conducting current. Logic high, or 1, occurs when the input is greater than 4 volts or open.



Drive Input Circuit

STEP tells the driver when to move the motor one step. The drive steps on the falling edge of the pulse. The minimum pulse width is 0.5 microseconds.

DIRECTION signals which way the motor should turn. See the step table on page 8 for details. The *DIRECTION* signal should be changed at least 2 microseconds before a step pulse is sent. **If you change the state of the direction input and send a step pulse at the same instant the motor may take a step in the wrong direction.**

ENABLE allows the user to turn off the current to the motor by setting this signal to logic 0. The logic circuitry continues to operate, so the drive "remembers" the step position even when the amplifiers are disabled. However, the motor may move slightly when the current is removed depending on the exact motor and load characteristics. **If you have no need to disable the amplifiers, you don't need to connect anything to the *ENABLE* input.**

Using Logic Voltages other than 5 volts DC

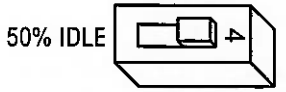
The 3540 M was designed to be used with 5 volt CMOS and TTL logic signals. To prevent interference between the drive and the controlling logic, the input signals are optically isolated. That means that your signals are powering LEDs within the drive's optocoupler circuits. The LEDs require at least 5 milliamps of current to turn on, but cannot stand more than 20 mA. Since the LEDs themselves only drop about two volts, current limiting resistors must be used on each logic input.

Idle Current Reduction

Your drive is equipped with a feature that automatically reduces the motor current by 50% anytime the motor is not moving. This reduces drive heating by about 50% and lowers motor heating by 75%. This feature can be disabled if desired so that full current is maintained at all times. This is useful when a high holding torque is required. To minimize motor and drive heating we highly recommend that you enable the idle current reduction feature unless your application strictly forbids it.

Idle current reduction is enabled by sliding switch #4 toward the *50% IDLE* label,as shown in the sketch below. Sliding the switch away from the *50% IDLE* label disables the reduction feature.

Idle Current Reduction Selected

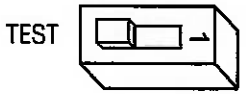


No Current Reduction

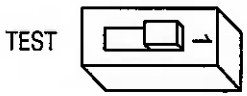
Self Test

The 3540 M includes a self test feature. This is used for trouble shooting. If you are unsure about the motor or signal connections to the drive, or if the 3540 M isn't responding to your step pulses, you can turn on the self test.

To activate the self test, slide switch #1 toward the *TEST* label. The drive will slowly rotate the motor, 1/2 revolution forward, then 1/2 rev backward. The pattern repeats until you slide the switch away from the *TEST* label. The 3540 M always uses half step mode during the self test, no matter how you set switches 2 and 3. The self test ignores the *STEP* and *DIRECTION* inputs while operating. The *ENABLE* input continues to function normally.



Self Test ON



Self Test OFF

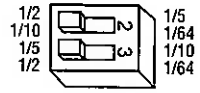
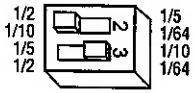
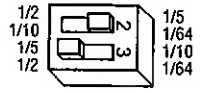
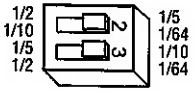
Microstepping

Most step motor drives offer a choice between full step and half step resolutions. In most full step drives, both motor phases are used all the time. Half stepping divides each step into two smaller steps by alternating between both phases on and one phase on. Microstepping drives like the 3540 M precisely control the amount of current in each phase at each step position as a means of electronically subdividing the steps even further. The 3540 M offers a choice of half step and 3 microstep resolutions. The highest setting divides each full step into 64 microsteps, providing 12,800 steps per revolution when using a 1.8° motor.

In addition to providing precise positioning and smooth motion, microstep drives can be used to provide motion in convenient units. When the drive is set to 2000 steps/rev (1/10 step) and used with a 5 pitch lead screw, you get .0001 inches/step.

Setting the step resolution is easy. Look at the dip switch on the 3540 M. Next to switches 2 and 3, there are labels on the printed circuit board. Each switch has two markings on each end. Switch 2 is marked 1/5, 1/10 at one end and 1/5, 1/64 at the other. Switch 3 is labeled 1/2, 1/5 and 1/10, 1/64. To set the drive for a resolution, push both switches toward the proper label. For example, if you want 1/10 step, push switch 2 toward the 1/10 label (to the left) and push switch 3 toward 1/10 (on the right).

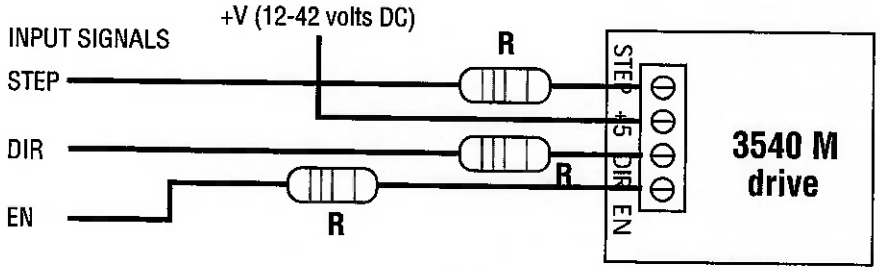
Please refer to the table below and set the switches for the resolution you want.

400 STEPS/REV (HALF) 	**2000 STEPS/REV (1/10)** 
1000 STEPS/REV (1/5) 	**12800 STEPS/REV (1/64)** 

We have included the proper resistor (680 ohms) within the drive for 5 volt operation. Therefore, if your logic voltage is 5 volts, you do not need to add resistors externally.

If your logic voltage is higher than five volts, you must add a resistor in series with each signal that you use (STEP, DIR and EN). The recommended wiring diagram is shown below. Table I lists the appropriate resistor value to use for a given power supply voltage. 1/4 watt or larger resistors should be used.

Please take care not to reverse the wiring, as damage to the LEDs will result rendering the drives inoperable. Check your wiring carefully before turning on the power supply!



Note: DIR signal is only required for bidirectional motion.
EN signal is only required to shut off motor current.
Both inputs can be left open if not needed.

Table I: External Dropping Resistors

Supply Voltage	R Ohms	Supply Voltage	R Ohms	Supply Voltage	R Ohms
12	1200	21	3000	30	4700
15	1800	24	3600	33	5100
18	2400	27	4200	35	5600

Step Table
(half stepping)

DIR=1 cw ↓

Step	A+	A-	B+	B-
0	open	open	+	–
1	+	–	+	–
2	+	–	open	open
3	+	–	–	+
4	open	open	–	+
5	–	+	–	+
6	–	+	open	open
7	–	+	+	–
8	open	open	+	–

↑ DIR=0 ccw

Step 0 is the Power Up State

Setting Phase Current

Before you turn on the power supply the first time, you need to set the driver for the proper motor phase current. The rated current is usually printed on the motor label. The 3540 M drive current is easy to set. If you wish, you can learn a simple formula for setting current and never need the manual again. Or you can skip to the table on the next page, find the current setting you want, and set the DIP switches according to the picture.

Current Setting Formula

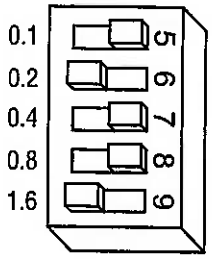
Locate the bank of tiny switches near the motor connector. Four of the switches have a value of current printed next to them, such as 0.4 and 0.8. Each switch controls the amount of current, in amperes (A), that its label indicates. There is always a base of current of 0.4 A. To add to that, slide the appropriate switches toward their labels on the PC board. You may need your small screwdriver for this.

Example

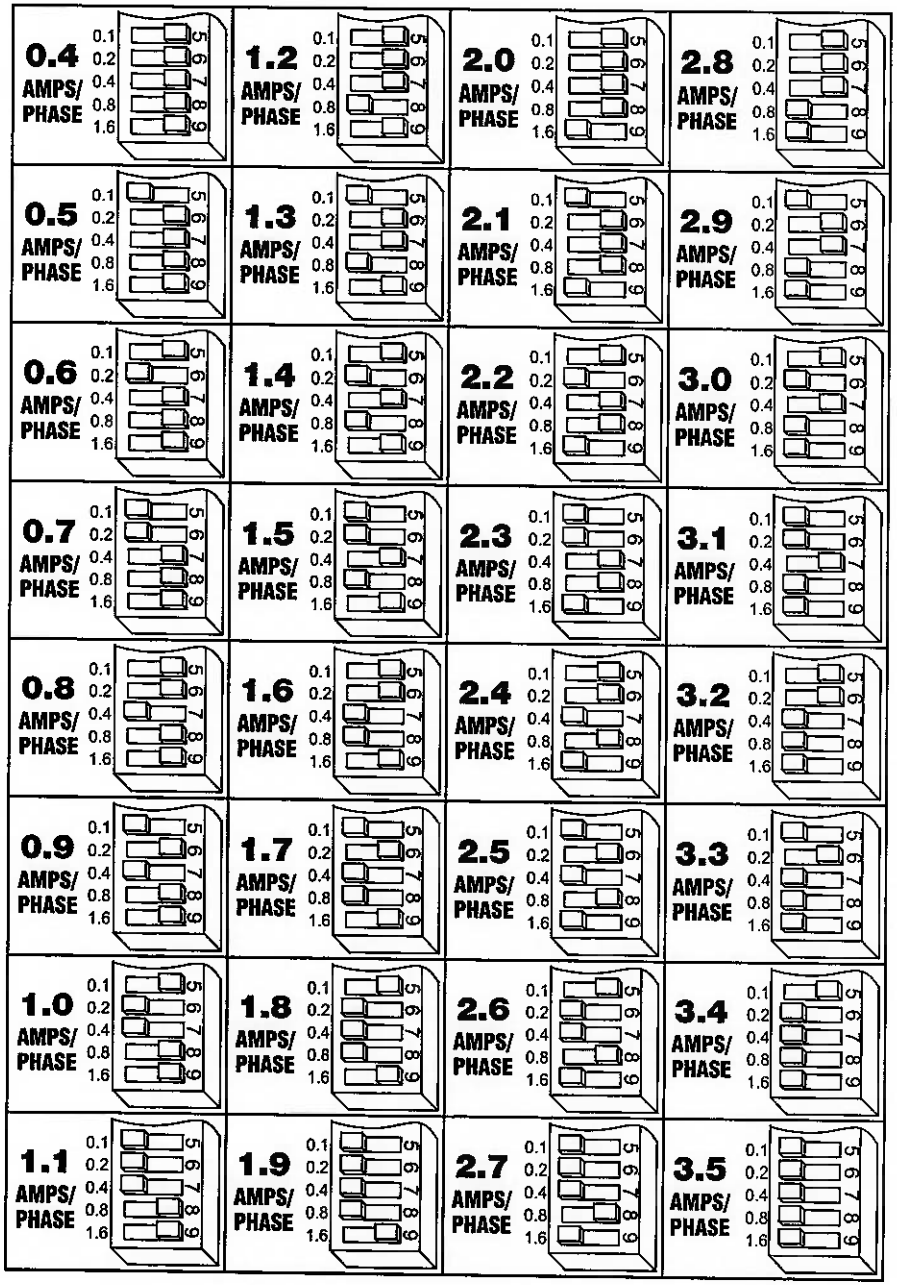
Suppose you want to set the driver for 2.2 amps per phase. You need the 0.4 A base current plus another 1.6 and 0.2 A.

2.2 = 0.4 + 1.6 + 0.2

Slide the 1.6 and 0.2 A switches toward the labels as shown in the figure.



Current Setting Table

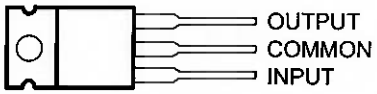


μA7800 SERIES
POSITIVE-VOLTAGE REGULATORS

SLVS056G – MAY 1976 – REVISED OCTOBER 2001

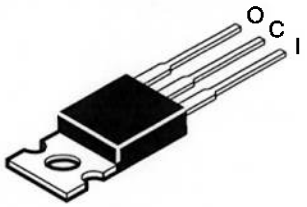
- **3-Terminal Regulators**
- **Output Current up to 1.5 A**
- **Internal Thermal-Overload Protection**
- **High Power-Dissipation Capability**
- **Internal Short-Circuit Current Limiting**
- **Output Transistor Safe-Area Compensation**
- **Direct Replacements for Fairchild μA7800 Series**

KC PACKAGE
(TOP VIEW)

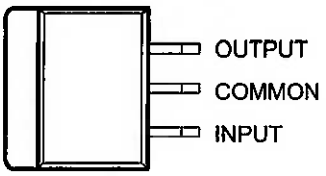


The COMMON terminal is in electrical contact with the mounting base.

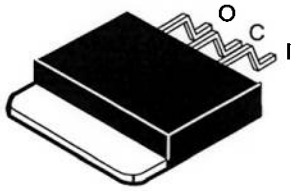
TO-220AB



KTE PACKAGE
(TOP VIEW)



The COMMON terminal is in electrical contact with the mounting base.



description

This series of fixed-voltage monolithic integrated-circuit voltage regulators is designed for a wide range of applications. These applications include on-card regulation for elimination of noise and distribution problems associated with single-point regulation. Each of these regulators can deliver up to 1.5 A of output current. The internal current-limiting and thermal-shutdown features of these regulators essentially make them immune to overload. In addition to use as fixed-voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents, and also can be used as the power-pass element in precision regulators.

The μA7800C series is characterized for operation over the virtual junction temperature range of 0°C to 125°C.

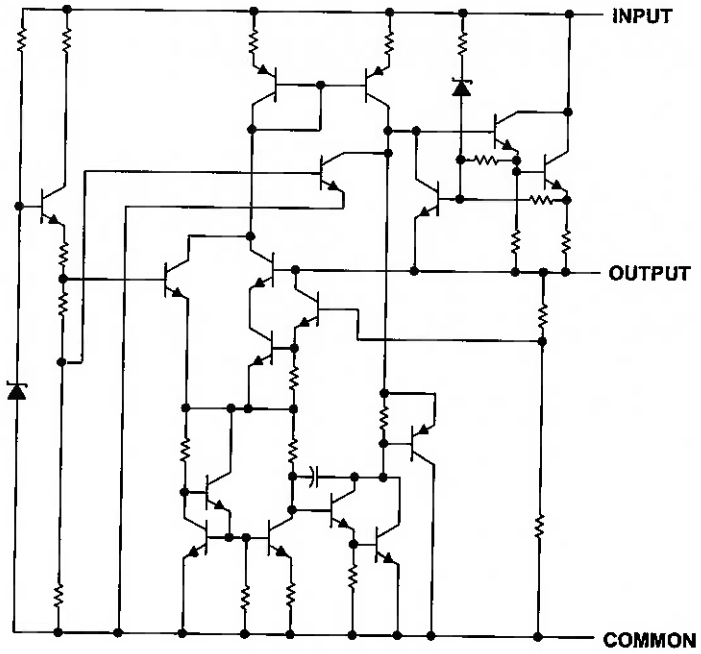
AVAILABLE OPTIONS

T_J	$V_{O(NOM)}$ (V)	PACKAGED DEVICES	
		PLASTIC FLANGE MOUNT (KC)	HEAT-SINK MOUNTED (KTE)
0°C to 125°C	5	μA7805CKC	μA7805CKTE
	8	μA7808CKC	μA7808CKTE
	10	μA7810CKC	μA7810CKTE
	12	μA7812CKC	μA7812CKTE
	15	μA7815CKC	μA7815CKTE
	24	μA7824CKC	μA7824CKTE

The KTE package is only available taped and reeled. Add the suffix R to the device type (e.g., μA7805CKTER).

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

schematic



absolute maximum ratings over virtual junction temperature range (unless otherwise noted)†

Input voltage, V_I: μA7824C ..... 40 V
All others ..... 35 V
Package thermal impedance, θ_{JA} (see Notes 1 and 2): KC package ..... 22°C/W
(see Notes 1 and 3): KTE package ..... 23°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C
Virtual junction temperature range, T_J ..... 0°C to 150°C
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) - T_A)/\theta_{JA}$. Selecting the maximum of 150°C can impact reliability.
2. The package thermal impedance is calculated in accordance with JESD 51-7.
3. The package thermal impedance is calculated in accordance with JESD 51-5.

recommended operating conditions

			MIN	MAX	UNIT
V_I	Input voltage	μA7805C	7	25	V
		μA7808C	10.5	25	
		μA7810C	12.5	28	
		μA7812C	14.5	30	
		μA7815C	17.5	30	
		μA7824C	27	38	
I_O	Output current			1.5	A
T_J	Operating virtual junction temperature	μA7800C series	0	125	°C

electrical characteristics at specified virtual junction temperature, V_I = 10 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7805C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le 15$ W, V_I = 7 V to 20 V	25°C	4.8	5	5.2	V
		0°C to 125°C	4.75		5.25	
Input voltage regulation	V_I = 7 V to 25 V	25°C		3	100	mV
	V_I = 8 V to 12 V			1	50	
Ripple rejection	V_I = 8 V to 18 V, f = 120 Hz	0°C to 125°C	62	78		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		15	100	mV
	I_O = 250 mA to 750 mA			5	50	
Output resistance	f = 1 kHz	0°C to 125°C		0.017		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1.1		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		40		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.2	8	mA
Bias current change	V_I = 7 V to 25 V	0°C to 125°C			1.3	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		750		mA
Peak output current		25°C		2.2		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

electrical characteristics at specified virtual junction temperature, V_I = 14 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7808C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le 15$ W, V_I = 10.5 V to 23 V	25°C	7.7	8	8.3	V
		0°C to 125°C	7.6		8.4	
Input voltage regulation	V_I = 10.5 V to 25 V	25°C		6	160	mV
	V_I = 11 V to 17 V			2	80	
Ripple rejection	V_I = 11.5 V to 21.5 V, f = 120 Hz	0°C to 125°C	55	72		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		12	160	mV
	I_O = 250 mA to 750 mA			4	80	
Output resistance	f = 1 kHz	0°C to 125°C		0.016		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–0.8		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		52		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.3	8	mA
Bias current change	V_I = 10.5 V to 25 V	0°C to 125°C			1	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		450		mA
Peak output current		25°C		2.2		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

electrical characteristics at specified virtual junction temperature, V_I = 17 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7810C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le$ 15 W, V_I = 12.5 V to 25 V	25°C	9.6	10	10.4	V
		0°C to 125°C	9.5	10	10.5	
Input voltage regulation	V_I = 12.5 V to 28 V	25°C		7	200	mV
	V_I = 14 V to 20 V			2	100	
Ripple rejection	V_I = 13 V to 23 V, f = 120 Hz	0°C to 125°C	55	71		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		12	200	mV
	I_O = 250 mA to 750 mA			4	100	
Output resistance	f = 1 kHz	0°C to 125°C		0.018		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		70		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.3	8	mA
Bias current change	V_I = 12.5 V to 28 V	0°C to 125°C			1	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		400		mA
Peak output current		25°C		2.2		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

electrical characteristics at specified virtual junction temperature, V_I = 19 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7812C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le$ 15 W, V_I = 14.5 V to 27 V	25°C	11.5	12	12.5	V
		0°C to 125°C	11.4		12.6	
Input voltage regulation	V_I = 14.5 V to 30 V	25°C		10	240	mV
	V_I = 16 V to 22 V			3	120	
Ripple rejection	V_I = 15 V to 25 V, f = 120 Hz	0°C to 125°C	55	71		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		12	240	mV
	I_O = 250 mA to 750 mA			4	120	
Output resistance	f = 1 kHz	0°C to 125°C		0.018		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		75		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.3	8	mA
Bias current change	V_I = 14.5 V to 30 V	0°C to 125°C			1	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		350		mA
Peak output current		25°C		2.2		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

electrical characteristics at specified virtual junction temperature, V_I = 23 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7815C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le$ 15 W, V_I = 17.5 V to 30 V	25°C	14.4	15	15.6	V
		0°C to 125°C	14.25		15.75	
Input voltage regulation	V_I = 17.5 V to 30 V	25°C		11	300	mV
	V_I = 20 V to 26 V			3	150	
Ripple rejection	V_I = 18.5 V to 28.5 V, f = 120 Hz	0°C to 125°C	54	70		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		12	300	mV
	I_O = 250 mA to 750 mA			4	150	
Output resistance	f = 1 kHz	0°C to 125°C		0.019		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		90		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.4	8	mA
Bias current change	V_I = 17.5 V to 30 V	0°C to 125°C			1	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		230		mA
Peak output current		25°C		2.1		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

electrical characteristics at specified virtual junction temperature, V_I = 33 V, I_O = 500 mA (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J†	μA7824C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le$ 15 W, V_I = 27 V to 38 V	25°C	23	24	25	V
		0°C to 125°C	22.8		25.2	
Input voltage regulation	V_I = 27 V to 38 V	25°C		18	480	mV
	V_I = 30 V to 36 V			6	240	
Ripple rejection	V_I = 28 V to 38 V, f = 120 Hz	0°C to 125°C	50	66		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		12	480	mV
	I_O = 250 mA to 750 mA			4	240	
Output resistance	f = 1 kHz	0°C to 125°C		0.028		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1.5		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		170		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.6	8	mA
Bias current change	V_I = 27 V to 38 V	0°C to 125°C			1	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		150		mA
Peak output current		25°C		2.1		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

APPLICATION INFORMATION

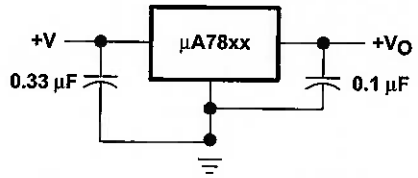


Figure 1. Fixed-Output Regulator

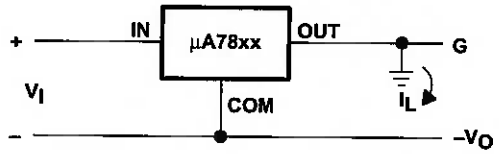
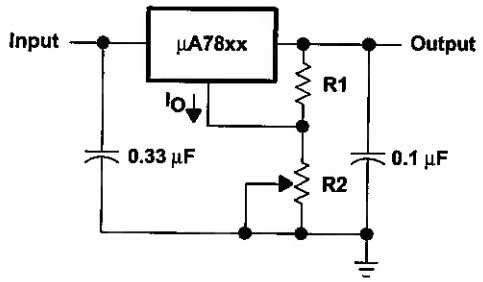


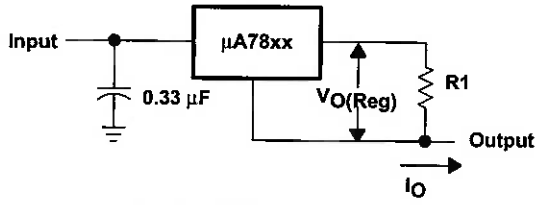
Figure 2. Positive Regulator in Negative Configuration (V_I Must Float)



NOTE A: The following formula is used when V_{XX} is the nominal output voltage (output to common) of the fixed regulator:

$$V_O = V_{xx} + \left(\frac{V_{xx}}{R1} + I_Q\right)R2$$

Figure 3. Adjustable-Output Regulator



$I_O = (V_O/R1) + I_O$ Bias Current

Figure 4. Current Regulator

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

APPLICATION INFORMATION

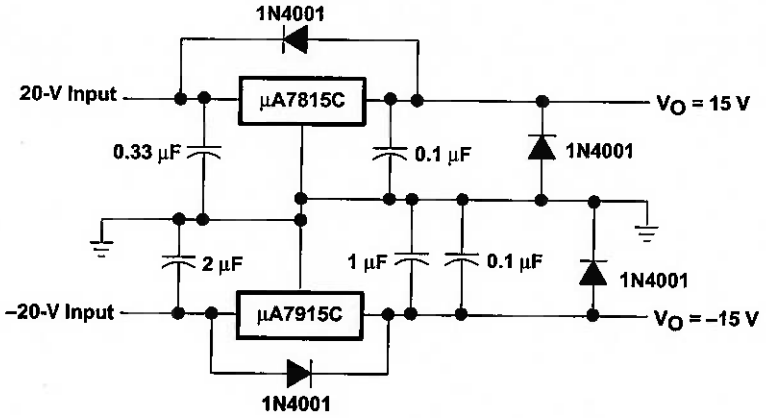


Figure 5. Regulated Dual Supply

operation with a load common to a voltage of opposite polarity

In many cases, a regulator powers a load that is not connected to ground but, instead, is connected to a voltage source of opposite polarity (e.g., operational amplifiers, level-shifting circuits, etc.). In these cases, a clamp diode should be connected to the regulator output as shown in Figure 6. This protects the regulator from output polarity reversals during startup and short-circuit operation.

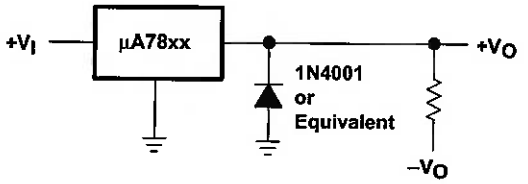


Figure 6. Output Polarity-Reversal-Protection Circuit

reverse-bias protection

Occasionally, the input voltage to the regulator can collapse faster than the output voltage. This can occur, for example, when the input supply is crowbarred during an output overvoltage condition. If the output voltage is greater than approximately 7 V, the emitter-base junction of the series-pass element (internal or external) could break down and be damaged. To prevent this, a diode shunt can be used as shown in Figure 7.

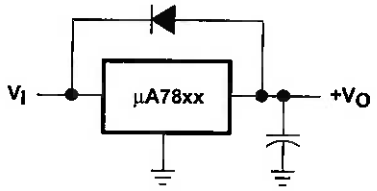


Figure 7. Reverse-Bias-Protection Circuit

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

UNITRODE

UC2577-ADJ

Simple Step-Up Voltage Regulator

FEATURES

- Requires Few External Components
- NPN Output Switches 3.0A, 65V(max)
- Extended Input Voltage Range: 3.0V to 40V
- Current Mode Operation for Improved Transient Response, Line Regulation, and Current Limiting
- Soft Start Function Provides Controlled Startup
- 52kHz Internal Oscillator
- Output Switch Protected by Current Limit, Undervoltage Lockout and Thermal Shutdown
- Improved Replacement for LM2577-ADJ Series

TYPICAL APPLICATIONS

- Simple Boost and Flyback Converters
- SEPIC Topology Permits Input Voltage to be Higher or Lower than Output Voltage
- Transformer Coupled Forward Regulators
- Multiple Output Designs

DESCRIPTION

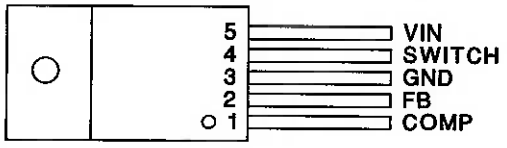
The UC2577-ADJ device provides all the active functions necessary to implement step-up (boost), flyback, and forward converter switching regulators. Requiring only a few components, these simple regulators efficiently provide up to 60V as a step-up regulator, and even higher voltages as a flyback or forward converter regulator.

The UC2577-ADJ features a wide input voltage range of 3.0V to 40V and an adjustable output voltage. An on-chip 3.0A NPN switch is included with undervoltage lockout, thermal protection circuitry, and current limiting, as well as soft start mode operation to reduce current during startup. Other features include a 52kHz fixed frequency on-chip oscillator with no external components and current mode control for better line and load regulation.

A standard series of inductors and capacitors are available from several manufacturers optimized for use with these regulators and are listed in this data sheet.

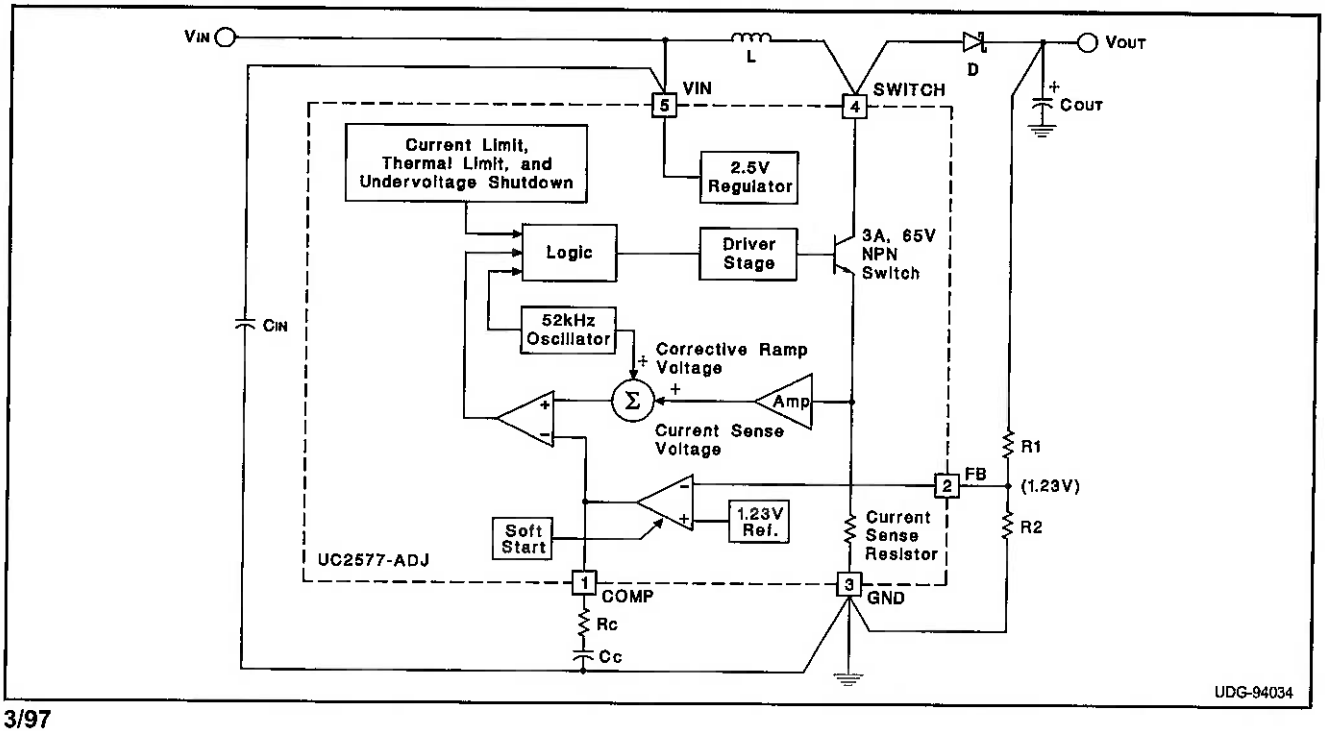
CONNECTION DIAGRAM

5-Pin TO-220 (Top View)
T Package



Also available in TO-263 Package (TD).

BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS (Note 1)

Supply Voltage 45V
Output Switch Voltage 65V
Output Switch Current (Note 2) 6.0A
Power Dissipation . . . . . Internally Limited
Storage Temperature Range . . . . . –65°C to +150°C
Lead Temperature (Soldering, 10 sec.) . . . . . 260°C
Maximum Junction Temperature . . . . . 150°C
Minimum ESD Rating (C = 100pF, R = 15kΩ) 2kV

RECOMMENDED OPERATING RANGE

Supply Voltage . . . . . 3.0V ≤ VIN ≤ 40V
Output Switch Voltage . . . . . 0V ≤ VSWITCH ≤ 60V
Output Switch Current . . . . . ISWITCH ≤ 3.0A
Junction Temperature Range . . . . . –40°C ≤ TJ ≤ +125°C

ELECTRICAL CHARACTERISTICS

Unless otherwise stated, these specifications apply for TA = –40°C to +125°C, VIN = 5V, VFB = VREF, ISWITCH = 0, and TA = TJ.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
System Parameters *Circuit Figure 1* (Note 3)					
Output Voltage	VIN = 5V to 10V, ILOAD = 100mA to 800mA	11.40	12.0	12.60	V
	TJ = 25°C	11.60		12.40	V
Line Regulation	VIN = 3.0V to 10V, ILOAD = 300mA		20	100	mV
	TJ = 25°C			50	mV
Load Regulation	VIN = 5V, ILOAD = 100mA to 800mA		20	100	mV
	TJ = 25°C			50	mV
Efficiency	VIN = 5V, ILOAD = 800mA		80		%
Device Parameters					
Input Supply Current	VFB = 1.5V (Switch Off)		7.5	14	mA
	TJ = 25°C			10	mA
	ISWITCH = 2.0A, VCOMP = 2.0V (Max Duty Cycle)		45	85	mA
	TJ = 25°C			70	mA
Input Supply UVLO	ISWITCH = 100mA		2.70	2.95	V
	TJ = 25°C			2.85	V
Oscillator Frequency	Measured at SWITCH Pin, ISWITCH = 100mA	42	52	62	kHz
	TJ = 25°C	48		56	kHz
Reference Voltage	Measured at FB Pin, VIN = 3.0V to 40V, VCOMP = 1.0V	1.206	1.230	1.254	V
	TJ = 25°C	1.214		1.246	V
Reference Voltage Line Regulation	VIN = 3.0V to 40V		0.5		mV
Error Amp Input Bias Current	VCOMP = 1.0V		100	800	nA
	TJ = 25°C			300	nA
Error Amp Transconductance	ICOMP = –30µA to +30µA, VCOMP = 1.0V	1600	3700	5800	µmho
	TJ = 25°C	2400		4800	µmho
Error Amp Voltage Gain	VCOMP = 0.8V to 1.6V, RCOMP = 1.0MW (Note 4)	250	800		V/V
	TJ = 25°C	500			V/V
Error Amplifier Output Swing	Upper Limit VFB = 1.0V	2.0	2.4		V
	TJ = 25°C	2.2			V
	Lower Limit VFB = 1.5V		0.3	0.55	V
	TJ = 25°C			0.40	V
Error Amp Output Current	VFB = 1.0V to 1.5V, VCOMP = 1.0V	±90	±200	±400	µA
	TJ = 25°C	±130		±300	µA
Soft Start Current	VFB = 1.0V, VCOMP = 0.5V	1.5	5.0	9.5	µA
	TJ = 25°C	2.5		7.5	µA
Maximum Duty Cycle	VCOMP = 1.5V, ISWITCH = 100mA	90	95		%
	TJ = 25°C	93			%

ELECTRICAL CHARACTERISTICS Unless otherwise stated, these specifications apply for T_A = –40°C to +125°C, VIN = 5V, V_{FB} = V_{REF}, I_{SWITCH} = 0, and T_A = T_J.

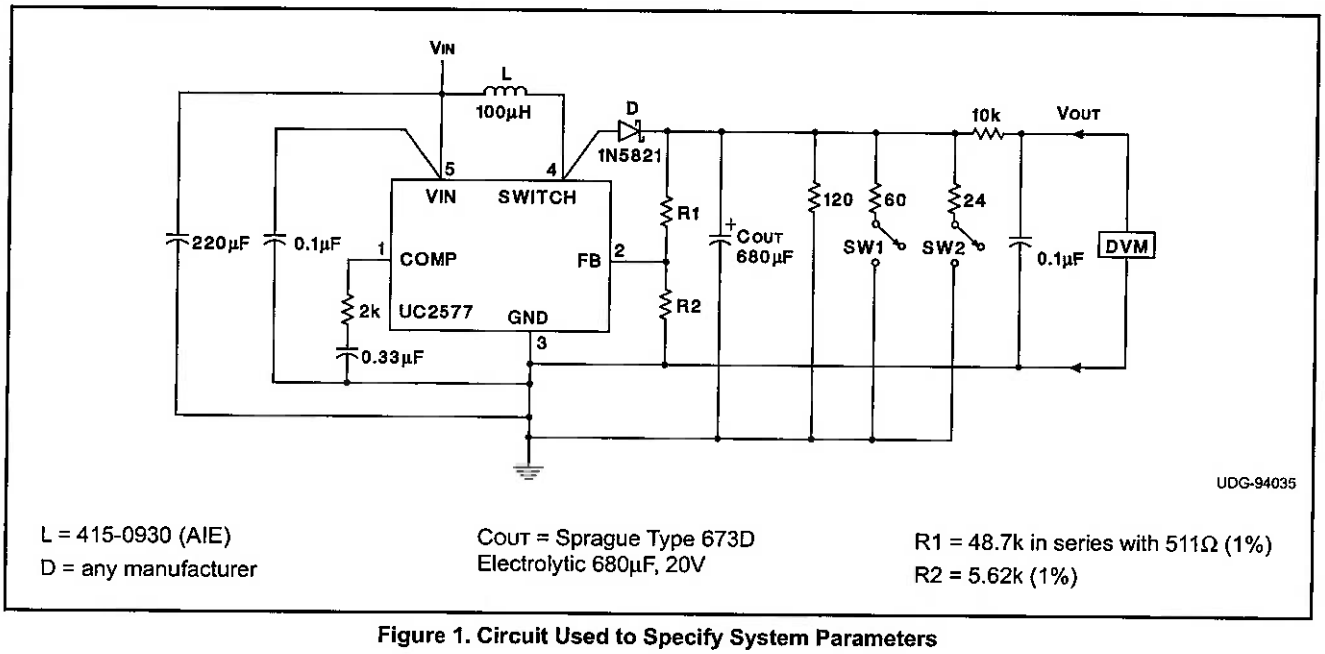
PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Device Parameters (cont.)					
Switch Transconductance			12.5		A/V
Switch Leakage Current	V_{SWITCH} = 65V, V_{FB} = 1.5V (Switch Off)		10	600	µA
	T_J = 25°C			300	µA
Switch Saturation Voltage	I_{SWITCH} = 2.0A, V_{COMP} = 2.0V (Max Duty Cycle)		0.5	0.9	V
	T_J = 25°C			0.7	V
NPN Switch Current Limit	V_{COMP} = 2.0V	3.0	4.3	6.0	A
Thermal Resistance	Junction to Ambient		65		°C/W
	Junction to Case		2		°C/W
COMP Pin Current	V_{COMP} = 0		25	50	µA
	T_J = 25°C			40	µA

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating ratings indicate conditions during which the device is intended to be functional, but device parameter specifications may not be guaranteed under these conditions. For guaranteed specifications and test conditions, see the Electrical Characteristics.

Note 2: Output current cannot be internally limited when the UC2577 is used as a step-up regulator. To prevent damage to the switch, its current must be externally limited to 6.0A. However, output current is internally limited when the UC2577 is used as a flyback or forward converter regulator.

Note 3. External components such as the diode, inductor, input and output capacitors can affect switching regulator performance. When the UC2577 is used as shown in the Test Circuit, system performance will be as specified by the system parameters.

Note 4: A 1.0MΩ resistor is connected to the compensation pin (which is the error amplifier's output) to ensure accuracy in measuring A_{VOL}. In actual applications, this pin's load resistance should be ≥ 10MΩ, resulting in A_{VOL} that is typically twice the guaranteed minimum limit.



L = 415-0930 (AIE)
D = any manufacturer

C_{OUT} = Sprague Type 673D Electrolytic 680µF, 20V

R1 = 48.7k in series with 511Ω (1%)
R2 = 5.62k (1%)

Figure 1. Circuit Used to Specify System Parameters

APPLICATIONS INFORMATION

Step-up (Boost) Regulator

The Block Diagram shows a step-up switching regulator utilizing the UC2577. The regulator produces an output voltage higher than the input voltage. The UC2577 turns its switch on and off at a fixed frequency of 52kHz, thus storing energy in the inductor (L). When the NPN switch is on, the inductor current is charged at a rate of VIN/L. When the switch is off, the voltage at the SWITCH terminal of the inductor rises above VIN, discharging the stored current through the output diode (D) into the output capacitor (C_{OUT}) at a rate of (V_{OUT} - VIN)/L. The energy stored in the inductor is thus transferred to the output.

The output voltage is controlled by the amount of energy transferred, which is controlled by modulating the peak inductor current. This modulation is accomplished by feeding a portion of the output voltage to an error amplifier which amplifies the difference between the feedback voltage and an internal 1.23V precision reference voltage. The output of the error amplifier is then compared to a voltage proportional to the switch current, or the inductor current, during the switch on time. A comparator terminates the switch on time when the two voltages are equal and thus controls the peak switch current to maintain a constant output voltage. Figure 2 shows voltage and current waveforms for the circuit. Formulas for calculation are shown in Figure 3.

STEP-UP REGULATOR DESIGN PROCEDURE

Refer to the Block Diagram

Given:

V_{INmin} = Minimum input supply voltage

V_{OUT} = Regulated output voltage

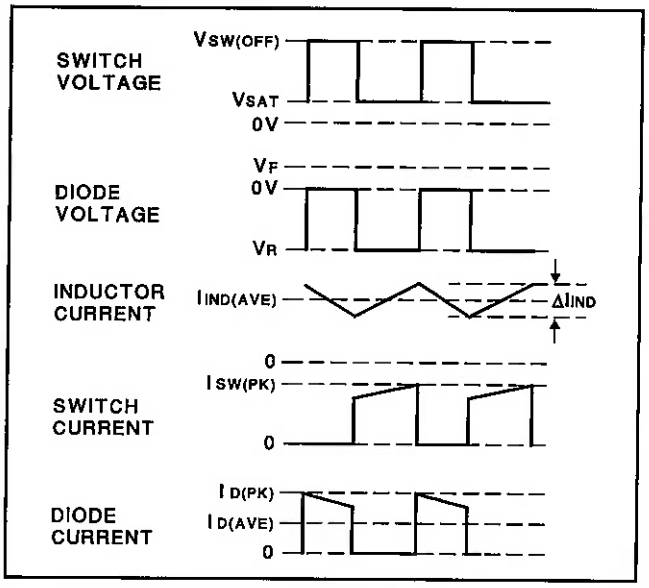


Figure 2. Step-up Regulator Waveforms

Duty Cycle	D	$\frac{V_{OUT} + V_F - V_{IN}}{V_{OUT} + V_F - V_{SAT}} \approx \frac{V_{OUT} - V_{IN}}{V_{OUT}}$
Avg. Inductor Current	$I_{IND(AVG)}$	$\frac{I_{LOAD}}{1 - D}$
Inductor Current Ripple	ΔI_{IND}	$\frac{V_{IN} - V_{SAT}}{L} \bullet \frac{D}{52{,}000}$
Peak Inductor Current	$I_{IND(PK)}$	$\frac{I_{LOAD}}{1-D} + \frac{\Delta I_{IND}}{2}$
Peak Switch Current	$I_{SW(PK)}$	$\frac{I_{LOAD}}{1-D} + \frac{\Delta I_{IND}}{2}$
Switch Voltage when Off	$V_{SW(OFF)}$	$V_{OUT} + V_F$
Diode Reverse Voltage	V_R	$V_{OUT} - V_{SAT}$
Avg. Diode Current	$I_{D(AVG)}$	I_{LOAD}
Peak Diode Current	$I_{D(PK)}$	$\frac{I_{LOAD}}{1-D} + \frac{\Delta I_{IND}}{2}$
Power Dissipation	P_D	$0.25\Omega \left(\frac{I_{LOAD}}{1-D}\right)^2 D + \frac{I_{LOAD} \bullet D \bullet V_{IN}}{50\,(1-D)}$

V_F = Forward Biased Diode Voltage, I_{LOAD} = Output Load

Figure 3. Step-up Regulator Formulas

First, determine if the UC2577 can provide these values of V_{OUT} and $I_{LOADmax}$ when operating with the minimum value of V_{IN}. The upper limits for V_{OUT} and $I_{LOADmax}$ are given by the following equations.

$$V_{OUT} \le 60V \text{ and}$$

$$V_{OUT} \le 10 \bullet V_{INmin}$$

$$I_{LOADmax} \le \frac{2.1A \bullet V_{INmin}}{V_{OUT}}$$

These limits must be greater than or equal to the values specified in this application.

1. Output Voltage Section

Resistors R1 and R2 are used to select the desired output voltage. These resistors form a voltage divider and present a portion of the output voltage to the error amplifier which compares it to an internal 1.23V reference. Select R1 and R2 such that:

$$\frac{R1}{R2} = \frac{V_{OUT}}{1.23V} - 1$$

APPLICATIONS INFORMATION (cont.)

2. Inductor Selection (L)

A. Preliminary Calculations

To select the inductor, the calculation of the following three parameters is necessary:

Dmax, the maximum switch duty cycle ($0 \le D \le 0.9$):

$$D_{max} = \frac{V_{OUT} + V_F - V_{INmin}}{V_{OUT} + V_F - 0.6V}$$

where typically $V_F = 0.5V$ for Schottky diodes and $V_F = 0.8V$ for fast recovery diodes.

E • T, the product of volts • time that charges the inductor:

$$E \bullet T = \frac{D_{max} \bullet (V_{INmin} - 0.6V)10^6}{52{,}000Hz} (V \bullet \mu s)$$

$I_{IND, DC}$, the average inductor current under full load:

$$I_{IND,\ DC} = \frac{1.05 \bullet I_{LOADmax}}{1 - D_{max}}$$

B. Identify Inductor Value:

1. From Figure 4, identify the inductor code for the region indicated by the intersection of E • T and $I_{IND, DC}$. This code gives the inductor value in microhenries. The L or H prefix signifies whether the inductor is rated for a maximum E • T of 90Vμs (L) or 250Vμs (H).

2. If D < 0.85, go to step C. If D ≥ 0.85, calculate the minimum inductance needed to ensure the switching regulator's stability:

If L_{min} is smaller than the inductor values found in step B1, go on to step C. Otherwise, the inductor value found in step B1 is too low; an appropriate inductor code should be obtained from the graph as follows:

1. Find the lowest value inductor that is greater than L_{min}.
2. Find where E • T intersects this inductor value to determine if it has an L or H prefix. If E • T intersects both the L and H regions, select the inductor with an H prefix.

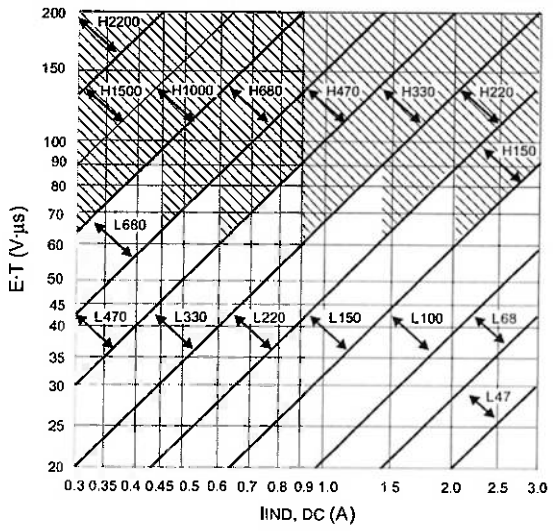
C. Inductor Selection

Select an inductor from the table of Figure 5 which cross references the inductor codes to the part numbers of the three different manufacturers. The inductors listed in this table have the following characteristics:

AIE (ferrite, pot-core inductors): Benefits of this type are low etectromagnetic interference (EMI), small physical size, and very low power dissipation (core loss).

Pulse (powdered iron, toroid core inductors): Benefits are low EMI and ability to withstand E • T and peak current above rated value better than ferrite cores.

Renco (ferrite, bobbin-core inductors): Benefits are low cost and best ability to withstand E • T and peak current above rated value. Be aware that these inductors generate more EMI than the other types, and this may interfere with signals sensitive to noise.



Note: This chart assumes that the inductor ripple current inductor is approximately 20% to 30% of the average inductor current (when the regulator is under full load). Greater ripple current causes higher peak switch currents and greater output ripple voltage. Lower ripple current is achieved with larger value inductors. The factor of 20% to 30% is chosen as a convenient balance between the two extremes.

Figure 4. Inductor Selection Graph

APPLICATIONS INFORMATION (cont.)

Inductor Code	Manufacturer's Part Number		
	AIE	Pulse	Renco
L47	415 - 0932	PE - 53112	RL2442
L68	415 - 0931	PE - 92114	RL2443
L100	415 - 0930	PE - 92108	RL2444
L150	415 - 0953	PE - 53113	RL1954
L220	415 - 0922	PE - 52626	RL1953
L330	415 - 0926	PE - 52627	RL1952
L470	415 - 0927	PE - 53114	RL1951
L680	415 - 0928	PE - 52629	RL1950
H150	415 - 0936	PE - 53115	RL2445
H220	430 - 0636	PE - 53116	RL2446
H330	430 - 0635	PE - 53117	RL2447
H470	430 - 0634	PE - 53118	RL1961
H680	415 - 0935	PE - 53119	RL1960
H1000	415 - 0934	PE - 53120	RL1959
H1500	415 - 0933	PE - 53121	RL1958
H2200	415 - 0945	PE - 53122	RL2448

AIE Magnetics, Div. Vernitron Corp., (813)347-2181
2801 72nd Street North, St. Petersburg, FL 33710
Pulse Engineering, (619)674-8100
12220 World Trade Drive, San Diego, CA 92128
Renco Electronics, Inc., (516)586-5566
60 Jeffryn Blvd. East, Deer Park, NY 11729

Figure 5. Table of Standardized Inductors and Manufacturer's Part Numbers

3. Compensation Network (Rc, Cc) and Output Capacitor (COUT) Selection

The compensation network consists of resistor RC and capacitor CC which form a simple pole-zero network and stabilize the regulator. The values of RC and CC depend upon the voltage gain of the regulator, $I_{LOADmax}$, the inductor L, and output capacitance COUT. A procedure to calculate and select the values for RC, CC, and COUT which ensures stability is described below. It should be noted, however, that this may not result in optimum compensation. To guarantee optimum compensation a standard procedure for testing loop stability is recommended, such as measuring VOUT transient responses to pulsing ILOAD.

A. Calculate the maximum value for RC.

$$Rc \le \frac{750 \bullet I_{LOADmax} \bullet V_{OUT}^2}{V_{INmin}^2}$$

Select a resistor less than or equal to this value, not to exceed 3kΩ.

B. Calculate the minimum value for COUT using the following two equations.

$$C_{OUT} \ge \frac{0.19 \bullet L \bullet Rc \bullet I_{LOADmax}}{V_{INmin} \bullet V_{OUT}} \quad \text{and}$$

$$C_{OUT} \ge \frac{V_{INmin} \bullet Rc \bullet (V_{INmin} + (3.74 \bullet 10^5 \bullet L))}{487{,}800 \bullet V_{OUT}^3}$$

The larger of these two values is the minimum value that ensures stability.

C. Calculate the minimum value of Cc.

$$Cc \ge \frac{58.5 \bullet V_{OUT}^2 \bullet C_{OUT}}{Rc^2 \bullet V_{INmin}}$$

The compensation capacitor is also used in the soft start function of the regulator. When the input voltage is applied to the part, the switch duty cycle is increased slowly at a rate defined by the compensation capacitor and the soft start current, thus eliminating high input currents. Without the soft start circuitry, the switch duty cycle would instantly rise to about 90% and draw large currents from the input supply. For proper soft starting, the value for Cc should be equal or greater than 0.22µF.

Figure 6 lists several types of aluminum electrolytic capacitors which could be used for the output filter. Use the following parameters to select the capacitor.

Working Voltage (WVDC): Choose a capacitor with a working voltage at least 20% higher than the regulator output voltage.

Ripple Current: This is the maximum RMS value of current that charges the capacitor during each switching cycle. For step-up and flyback regulators, the formula for ripple current is:

$$I_{RIPPLErms} = \frac{I_{LOADmax} \bullet D_{max}}{1 - D_{max}}$$

Choose a capacitor that is rated at least 50% higher than this value at 52kHz.

Equivalent Series Resistance (ESR): This is the primary cause of output ripple voltage, and it also affects the values of RC and CC needed to stabilize the regulator. As a result, the preceding calculations for CC and RC are only valid if the ESR does not exceed the maximum value specified by the following equations.

$$ESR \le \frac{0.01 \bullet 15V}{I_{RIPPLE(P-P)}} \quad \text{and} \quad \le \frac{8.7 \bullet 10^{-3} \bullet V_{IN}}{I_{LOADmax}} \quad \text{where}$$

$$I_{RIPPLE(P-P)} = \frac{1.15 \bullet I_{LOADmax}}{1 - D_{max}}$$

Select a capacitor with an ESR, at 52kHz, that is less than or equal to the lower value calculated. Most electrolytic capacitors specify ESR at 120kHz which is 15% to 30% higher than at 52kHz. Also, note that ESR increases by a factor of 2 when operating at –20°C.

In general, low values of ESR are achieved by using large value capacitors (C ≥ 470µF), and capacitors with high WVDC, or by paralleling smaller value capacitors.

APPLICATIONS INFORMATION (cont.)

4. Input Capacitor Selection (C_{IN})

To reduce noise on the supply voltage caused by the switching action of a step-up regulator (ripple current noise), VIN should be bypassed to ground. A good quality 0.1µF capacitor with low ESR should provide sufficient decoupling. If the UC2577 is located far from the supply source filter capacitors, an additional electrolytic (47µF, for example) is required.

Nichicon - Types PF, PX, or PZ
927 East StateParkway, Schaumburg, IL 60173
(708)843-7500

United Chemi-CON - Types LX, SXF, or SXJ
9801 West Higgens, Rosemont, IL 60018
(708)696-2000

Figure 6. Aluminum Electrolytic Capacitors Recommended for Switching Regulators

5. Output Diode Selection (D)

In the step-up regulator, the switching diode must withstand a reverse voltage and be able to conduct the peak output current of the UC2577. Therefore a suitable diode must have a minimum reverse breakdown voltage greater than the circuit output voltage, and should also be rated for average and peak current greater than $I_{LOADmax}$ and I_{Dpk}. Because of their low forward voltage drop (and thus higher regulator efficiencies), Schottky barrier diodes are often used in switching regulators. Refer to Figure 7 for recommended part numbers and voltage ratings of 1A and 3A diodes.

V_{OUTmax}	Schottky		Fast Recovery	
	1A	3A	1A	3A
20V	1N5817 MBR120P	1N5820 MBR320P		
30V	1N5818 MBR130P 11DQ03	1N5821 MBR330P 31DQ03		
40V	1N5819 MBR140P 11DQ04	1N5822 MBR340P 31DQ04		
50V	MBR150 11DQ05	MBR350 31DQ05	1N4933 MUR105	
100V			1N4934 MUR110 10DL1	MR851 30DL1 MR831

MBRxxx and MURxxx are manufactured by Motorola.
1DDxxx, 11Cxx and 31Dxx are manufactured by International Rectifier

Figure 7. Diode Selection Chart

ORDERING INFORMATION

Unitrode Type Number	
UC2577T-ADJ	5 Pin TO-220 Plastic Package
UC2577TD-ADJ	5 Pin TO-263 Plastic Package

UNITRODE CORPORATION
7 CONTINENTAL BLVD. • MERRIMACK, NH 03054
TEL. (603) 424-2410 • FAX (603) 424-3460

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 acknowledgement, 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.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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.

HARRIS Semiconductor

CDP6402, CDP6402C

CMOS Universal Asynchronous Receiver/Transmitter (UART)

March 1997

Features

- Low Power CMOS Circuitry. . . . . 7.5mW (Typ) at 3.2MHz (Max Freq.) at V_{DD} = 5V
- Baud Rate
  - DC to 200K Bits/s (Max) at. . . . . 5V, 85°C
  - DC to 400K Bits/s (Max) at. . . . . 10V, 85°C
- 4V to 10.5 Operation
- Automatic Data Formatting and Status Generation
- Fully Programmable with Externally Selectable Word Length (5 - 8 Bits), Parity Inhibit, Even/Odd Parity, and 1, 1-1/2, or 2 Stop Bits
- Operating Temperature Range
  - CDP6402D, CD . . . . . -55°C to +125°C
  - CDP6402E, CE . . . . . -40°C to +85°C
- Replaces Industry Type IM6402 and Compatible with HD6402

Ordering Information

PACK-AGE	TEMP. RANGE	5V/200K BAUD	10V/400K BAUD	PKG. NO.
PDIP	-40°C to +85°C	CDP6402CE	CDP6402E	E40.6
Burn-In		CDP6402CEX	-	
SBDIP	-40°C to +85°C	CDP6402CD	CDP6402D	D40.6
Burn-In		CDP6402CDX	CDP6402DX	

Description

The CDP6402 and CDP6402C are silicon gate CMOS Universal Asynchronous Receiver/Transmitter (UART) circuits for interfacing computers or microprocessors to asynchronous serial data channels. They are designed to provide the necessary formatting and control for interfacing between serial and parallel data channels. The receiver converts serial start, data, parity, and stop bits to parallel data verifying proper code transmission, parity and stop bits. The transmitter converts parallel data into serial form and automatically adds start parity and stop bits.

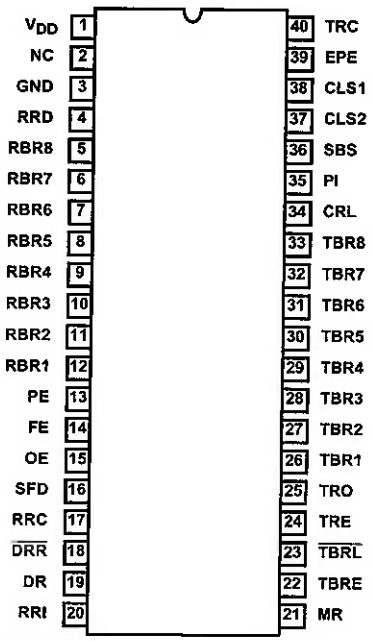
The data word can be 5, 6, 7 or 8 bits in length. Parity may be odd, even or inhibited. Stop bits can be 1, 1-1/2, or 2 (when transmitting 5-bit code).

The CDP6402 and CDP6402C can be used in a wide range of applications including modems, printers, peripherals, video terminals, remote data acquisition systems, and serial data links for distributed processing systems.

The CDP6402 and CDP6402C are functionally identical. They differ in that the CDP6402 has a recommended operating voltage range of 4V to 10.5V, and the CDP6402C has a recommended operating voltage range of 4V to 6.5V.

Pinout

CDP6402, CDP6402C (PDIP, SBDIP)
TOP VIEW



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper IC Handling Procedures.

File Number **1328.2**

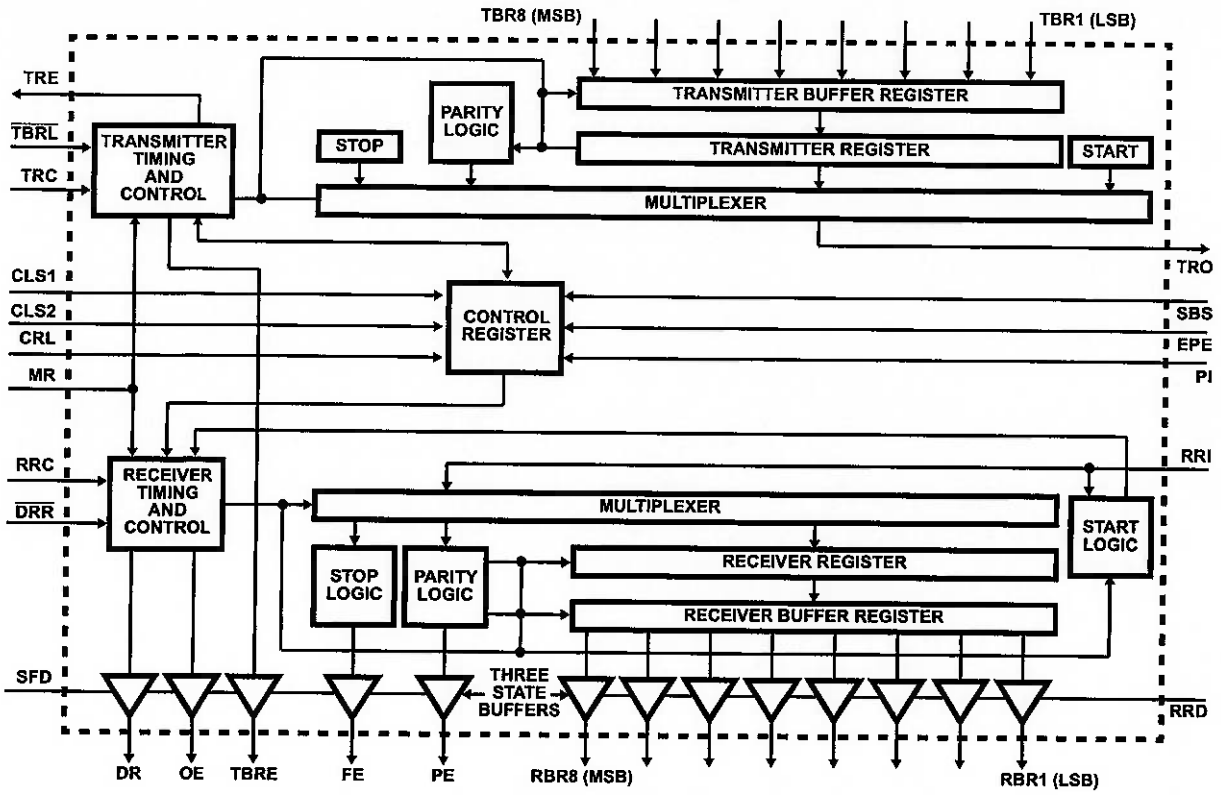


FIGURE 1. FUNCTIONAL BLOCK DIAGRAM

Absolute Maximum Ratings

DC Supply-Voltage Range, (V_{DD})
- CDP6402 . . . . . -0.5 to +11V
- CDP6402C . . . . . -0.5 to +7V

Input Voltage Range, All Inputs . . . . . -0.5 to V_{DD} +0.5V
DC Input Current, Any One Input. . . . . ±100µA
Device Dissipation Per Output Transistor
- For T_A = Full Package-Temperature Range (All Package Types) 100mW

Operating-Temperature Range (T_A)
- Package Type D (SBDIP) . . . . . -55°C to +125°C
- Package Type E (PDIP) . . . . . -40°C to +85°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
PDIP Package	50	N/A
SBDIP Package	55	15

Maximum Junction Temperature
- Plastic Package . . . . . +150°C
- Ceramic Package . . . . . +175°C

Maximum Storage Temperature Range (T_{STG}) . . . -65°C to +150°C
Maximum Lead Temperature (Soldering 10s):
- At Distance 1/16 ±1/32 inch (1.59 ±0.79mm) . . . . . +265°C

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. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

Operating Conditions At T_A = Full Package-Temperature Range. For maximum reliability, operating conditions should be selected so that operation is always within the following ranges:

PARAMETER	LIMITS				UNITS
	CDP6402		CDP6402C		
	MIN	MAX	MIN	MAX	
DC Operating Voltage Range	4	10.5	4	6.5	V
Input Voltage Range	V_{SS}	V_{DD}	V_{SS}	V_{DD}	V

Static Electrical Specifications at T_A = -40°C to +85°C, V_{DD} ±10%, Except as noted

PARAMETER		CONDITIONS			LIMITS						UNITS
					CDP6402			CDP6402C			
		V_O (V)	V_{IN} (V)	V_{DD} (V)	MIN	(NOTE 1) TYP	MAX	MIN	(NOTE 1) TYP	MAX	
Quiescent Device Current	I_{DD}	-	0, 5	5	-	0.01	50	-	0.02	200	µA
		-	0,10	10	-	1	200	-	-	-	µA
Output Low Drive (Sink) Current	I_{OL}	0.4	0,5	5	2	4	-	1.2	2.4	-	mA
		0.5	0,10	10	5	7	-	-	-	-	mA
Output High Drive (Source) Current	I_{OH}	4.6	0, 5	5	-0.55	-1.1	-	-0.55	-1.1	-	mA
		9.5	0,10	10	-1.3	-2.6	-	-	-	-	mA
Output Voltage Low-Level (Note 2)	V_{OL}	-	0, 5	5	-	0	0.1	-	0	0.1	V
		-	0, 10	10	-	0	0.1	-	-	-	V
Output Voltage High Level (Note 2)	V_{OH}	-	0, 5	5	4.9	5	-	4.9	5	-	V
		-	0, 10	10	9.9	10	-	-	-	-	V
Input Low Voltage	V_{IL}	0.5, 4.5	-	5	-	-	0.8	-	-	0.8	V
		0.5, 9.5	-	10	-	-	0.2 V_{DD}	-	-	-	V

Static Electrical Specifications at T_A = -40°C to +85°C, V_{DD} ±10%, Except as noted **(Continued)**

PARAMETER		CONDITIONS			LIMITS						UNITS
					CDP6402			CDP6402C			
		V_O (V)	V_{IN} (V)	V_{DD} (V)	MIN	(NOTE 1) TYP	MAX	MIN	(NOTE 1) TYP	MAX	
Input High Voltage	V_{IH}	0.5, 4.5	-	5	V_{DD}-2	-	-	V_{DD}-2	-	-	V
		0.5, 9.5	-	10	7	-	-	-	-	-	V
Input Leakage Current	I_{IN}	Any Input	0,5	5	-	$\pm10^{-4}$	±1	-	-	±1	µA
			0,10	10	-	$\pm10^{-4}$	±2	-	-	-	µA
Three-State Output Leakage Current	I_{OUT}	0, 5	0, 5	5	-	$\pm10^{-4}$	±1	-	$\pm10^{-4}$	±1	µA
		0, 10	0,10	10	-	$\pm10^{-4}$	±10	-	-	-	µA
Operating Current (Note 3)	I_{DD1}	-	0, 5	5	-	1.5	–	-	1.5	-	mA
		-	0,10	10	-	10	–	-	-	-	mA
Input Capacitance	C_{IN}	-	-	-	-	5	7.5	-	5	7.5	pF
Output Capacitance	C_{OUT}	-	-	-	-	10	15	-	10	15	pF

NOTES:

1. Typical values are for T_A = 25°C and nominal V_{DD}
2. $I_{OL} = I_{OH}$ = 1µA.
3. Operating current is measured at 200kHz or V_{DD} = 5V and 400kHz for V_{DD} = 10V, with open outputs (worst-case frequencies for CDP1802A system operating at maximum speed of 3.2MHz).

Description of Operation

Initialization and Controls

A positive pulse on the MASTER RESET (MR) input resets the control, status, and receiver buffer registers, and sets the serial output (TRO) High. Timing is generated from the clock inputs RRC and TRC at a frequency equal to 16 times the serial data bit rate. The RRC and TRC inputs may be driven by a common clock, or may be driven independently by two different clocks. The CONTROL REGISTER LOAD (CRL) input is strobed to load control bits for PARITY INHIBIT (PI), EVEN PARITY ENABLE (EPE), STOP BIT SELECTS (SBS), and CHARACTER LENGTH SELECTS (CLS1 and CLS2). These inputs may be hand wired to V_{SS} or V_{DD} with CRL to V_{DD}. When the initialization is completed, the UART is ready for receiver and/or transmitter operations.

Transmitter Operation

The transmitter section accepts parallel data, formats it, and transmits it in serial form (Figure 2) on the TRO terminal.

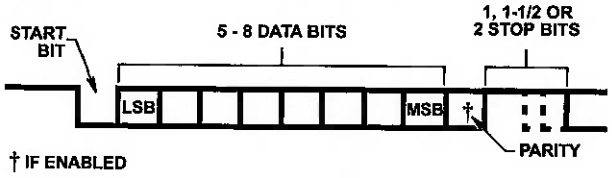


FIGURE 2. SERIAL DATA FORMAT

Transmitter timing is shown in Figure 3. (A) Data is loaded into the transmitter buffer register from the inputs TBR1 through TBR8 by a logic low on the $\overline{TBRL}$ input. Valid data must be present at least t_{DT} prior to, and t_{TD} following, the rising edge of $\overline{TBRL}$. If words less than 8-bits are used, only the least significant bits are used. The character is right justified into the least significant bit, TBR1. (B) The rising edge of $\overline{TBRL}$ clears TBRE. 1/2 to 11/2 cycles later, depending on when the $\overline{TBRL}$ pulse occurs with respect to TRC, data is transferred to the transmitter register and TRE is cleared. TBRE is set to a logic High one cycle after that.

Output data is clocked by TRC. The clock rate is 16 times the data rate. (C) A second pulse on $\overline{TBRL}$ loads data into the transmitter buffer register. Data transfer to the transmitter register is delayed until transmission of the current character is complete. (D) Data is automatically transferred to the transmitter register and transmission of that character begins.

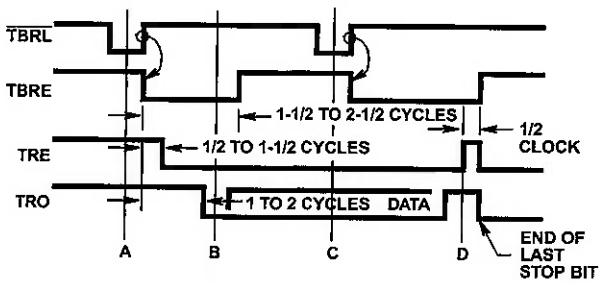


FIGURE 3. TRANSMITTER TIMING WAVEFORMS

Receiver Operation

Data is received in serial form at the RRI input. When no data is being received, RRI input must remain high. The data is clocked through the RRC. The clock rate is 16 times the data rate. Receiver timing is shown in Figure 4.

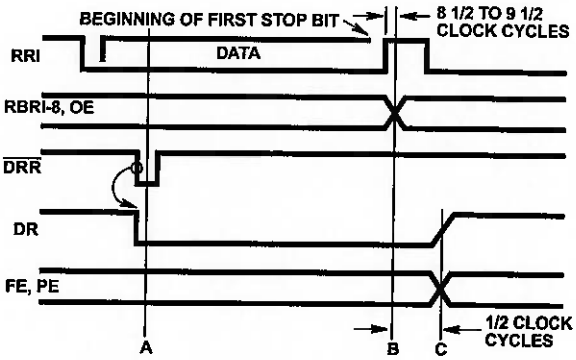


FIGURE 4. RECEIVER TIMING WAVEFORMS

(A) A low level on $\overline{DRR}$ clears the DR line. (B) During the first stop bit data is transferred from the receiver register to the RB Register. If the word is less than 8 bits, the unused most significant bits will be a logic low. The output character is right justified to the least significant bit RBR1. A logic high on OE indicates overruns. An overrun occurs when DR has not been cleared before the present character was transferred to the RBR. (C) 1/2 clock cycle later DR is set to a logic high and FE is evaluated. A logic high on FE indicates an invalid stop bit was received. A logic high on PE indicates a parity error.

Start Bit Detection

The receiver uses a 16X clock for timing (Figure 5). The start bit could have occurred as much as one clock cycle before it was detected, as indicated by the shaded portion. The center of the start bit is defined as clock count 7 1/2. If the receiver clock is a symmetrical square wave, the center of the start bit will be located within ±1/2 clock cycle ±1/32 bit or ±3.125%. The receiver begins searching for the next start bit at 9 clocks into the first stop bit.

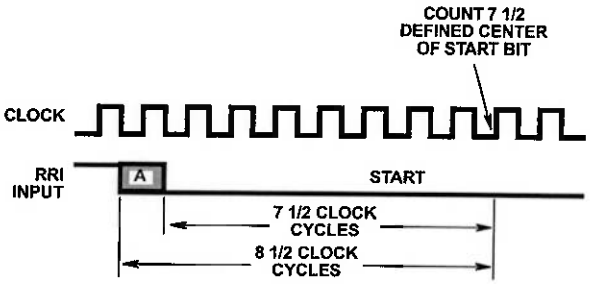


FIGURE 5. START BIT TIMING WAVEFORMS

TABLE 1. CONTROL WORD FUNCTION

CONTROL WORD					DATA BITS	PARITY BIT	STOP BIT (S)
CLS2	CLS1	PI	EPE	SBS			
L	L	L	L	L	5	ODD	1
L	L	L	L	H	5	ODD	1.5
L	L	L	H	L	5	EVEN	1
L	L	L	H	H	5	EVEN	1.5
L	L	H	X	L	5	DISABLED	1
L	L	H	X	H	5	DISABLED	1.5
L	H	L	L	L	6	ODD	1
L	H	L	L	H	6	ODD	2
L	H	L	H	L	6	EVEN	1
L	H	L	H	H	6	EVEN	2
L	H	H	X	L	6	DISABLED	1
L	H	H	X	H	6	DISABLED	2
H	L	L	L	L	7	ODD	1
H	L	L	L	H	7	ODD	2
H	L	L	H	L	7	EVEN	1
H	L	L	H	H	7	EVEN	2
H	L	H	X	L	7	DISABLED	1
H	L	H	X	H	7	DISABLED	2
H	H	L	L	L	8	ODD	1
H	H	L	L	H	8	ODD	2
H	H	L	H	L	8	EVEN	1
H	H	L	H	H	8	EVEN	2
H	H	H	X	L	8	DISABLED	1
H	H	H	X	H	8	DISABLED	2

NOTE: X = Don't Care

TABLE 2. FUNCTION PIN DEFINITION

PIN	SYMBOL	DESCRIPTION
1	V_{DD}	Positive Power Supply
2	N/C	No Connection
3	GND	Ground (V_{SS})
4	RRD	A high level on RECEIVER REGISTER DISABLE forces the receiver holding register ouputs RBR1-RBR8 to a high impedance state.
5	RBR8	The contents of the RECEIVER BUFFER REGISTER appear on these three-state outputs. Word formats less than 8 characters are right justified to RBR1.
6	RBR7	See Pin 5 - RBR8
7	RBR6	
8	RBR5	
9	RBR4	
10	RBR3	
11	RBR2	
12	RBR1	
13	PE	A high level on PARITY ERROR indicates that the received parity does not match parity programmed by control bits. The output is active until parity matches on a succeeding character. When parity is inhibited, this output is low.
14	FE	A high level on FRAMING ERROR indicates the first stop bit was invalid. FE will stay active until the next valid character's stop bit is received.
15	OE	A high level on OVERRUN ERROR indicates the data received flag was not cleared before the last character was transferred to the receiver buffer register. The Error is reset at the next character's stop bit if DRR has been performed (i.e., DRR; active low).
16	SFD	A high level on STATUS FLAGS DISABLE forces the outputs PE, FE, OE, DR, TBRE to a high impedance state.
17	RRC	The RECEIVER REGISTER CLOCK is 16X the receiver data rate.
18	$\overline{DRR}$	A low level on DATA RECEIVED RESET clears the data received output (DR), to a low level.
19	DR	A high level on DATA RECEIVED indicates a character has been received and transferred to the receiver buffer register.
20	RRI	Serial data on RECEIVER REGISTER INPUT is clocked into the receiver register.
21	MR	A high level on MASTER RESET (MR) clears PE, FE, OE and DR, and sets TRE, TBRE, and TRO. TRE is actually set on the first rising edge of TRC after MR goes high. MR should be strobed after power-up.
22	TBRE	A high level on TRANSMITTER BUFFER REGISTER EMPTY indicates the transmitter buffer register has transferred its data to the transmitter register and is ready for new data.
23	$\overline{TBRL}$	A low level on TRANSMITTER BUFFER REGISTER LOAD transfers data from inputs TBR1-TBR8 into the transmitter buffer register. A low to high transition on $\overline{TBRL}$ requests data transfer to the transmitter register. If the transmitter register is busy, transfer is automatically delayed so that the two characters are transmitted end to end.
24	TRE	A high level on TRANSMITTER REGISTER EMPTY indicates completed transmission of a character including stop bits.

TABLE 2. FUNCTION PIN DEFINITION (Continued)

PIN	SYMBOL	DESCRIPTION
25	TRO	Character data, start data and stop bits appear serially at the TRANSMITTER REGISTER OUTPUT.
26	TBR1	Character data is loaded into the TRANSMITTER BUFFER REGISTER via inputs TBR1-TBR8. For character formats less than 8 bits, the TBR8, 7, and 6 Inputs are ignored corresponding to the programmed word length.
27	TBR2	See Pin 26 - TBR1
28	TBR3	
29	TBR4	
30	TBR5	
31	TBR6	
32	TBR7	
33	TBR8	
34	CRL	A high level on CONTROL REGISTER LOAD loads the control register.
35	PI†	A high level on PARITY INHIBIT inhibits parity generation, parity checking and forces PE output low.
36	SBS†	A high level on STOP BIT SELECT selects 1.5 stop bits for a 5 character format and 2 stop bits for other lengths.
37	CLS2†	These inputs program the CHARACTER LENGTH SELECTED. (CLS1 low CLS2 low 5 bits) (CLS1 high CLS2 low 6 bits) (CLS1 low CLS2 high 7 bits) (CLS1 high CLS2 high 8 bits).
38	CLS1†	See Pin 37 - CLS2
39	EPE†	When PI is low, a high level on EVEN PARITY ENABLE generates and checks even parity. A low level selects odd parity.
40	TRC	The TRANSMITTER REGISTER CLOCK is 16X the transmit data rate.

† See Table 1 (Control Word Function)

Dynamic Electrical Specifications at T_A = -40°C to +85°C, V_{DD} ±5%, t_R, t_F = 20ns, V_{IH} = 0.7 V_{DD}, V_{IL} = 0.3 V_{DD}, C_L = 100pF

(NOTE 1) PARAMETER		V_{DD} (V)	LIMITS				UNITS
			CDP6402		CDP6402C		
			(NOTE 2) TYP	(NOTE 3) MAX	(NOTE 2) TYP	(NOTE 3) MAX	
SYSTEM TIMING (See Figure 6)							
Minimum Pulse Width CRL	t_{CRL}	5	50	150	50	150	ns
		10	40	100	-	-	ns
Minimum Setup Time Control Word to CRL	t_{CWC}	5	20	50	20	50	ns
		10	0	40	-	-	ns
Minimum Hold Time Control Word after CRL	t_{CCW}	5	40	60	40	60	ns
		10	20	30	-	-	ns
Propagation Delay Time SFD High to SOD	t_{SFDH}	5	130	200	130	200	ns
		10	100	150	-	-	ns
SFD Low to SOD	t_{SFDL}	5	130	200	130	200	ns
		10	40	60	-	-	ns
RRD High to Receiver Register High Impedance	t_{RRDH}	5	80	150	80	150	ns
		10	40	70	-	-	ns
RRD Low to Receiver Register Active	t_{RRDL}	5	80	150	80	150	ns
		10	40	70	-	-	ns
Minimum Pulse Width MR		5	200	400	200	400	ns
		10	100	200	-	-	ns

NOTES:

1. All measurements are made at the 50% point of the transition except three-state measurements.
2. Typical values for T_A = 25°C and nominal V_{DD}.
3. Maximum limits of minimum characteristics are the values above which all devices function.

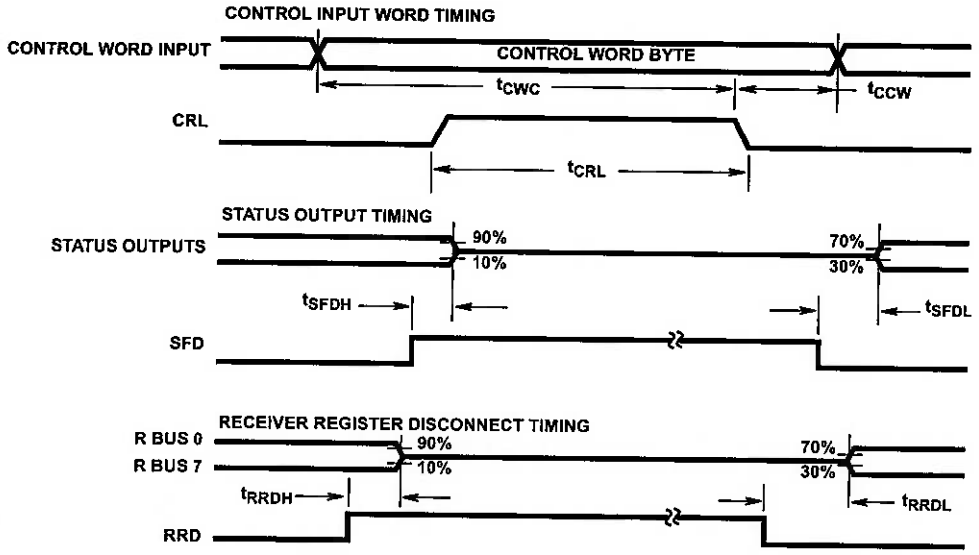


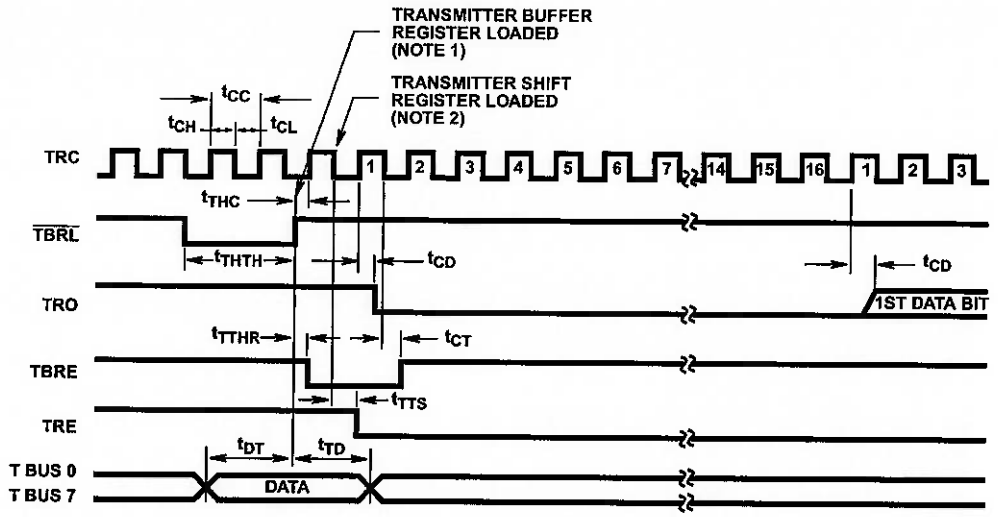
FIGURE 6. SYSTEM TIMING WAVEFORMS

Dynamic Electrical Specifications at T_A = -40°C to +85°C, V_{DD} ±5%, t_R, t_F = 20ns, V_{IH} = 0.7 V_{DD}, V_{IL} = 0.3 V_{DD}, C_L = 100pF

(NOTE 1) PARAMETER		V_{DD} (V)	LIMITS				UNITS
			CDP6402		CDP6402C		
			(NOTE 2) TYP	(NOTE 3) MAX	(NOTE 2) TYP	(NOTE 3) MAX	
TRANSMITTER TIMING (See Figure 7)							
Minimum Clock Period (TRC)	t_{CC}	5	250	310	250	310	ns
		10	125	155	-	-	ns
Minimum Pulse Width Clock Low Level	t_{CL}	5	100	125	100	125	ns
		10	75	100	-	-	ns
Clock High Level	t_{CH}	5	100	125	100	125	ns
		10	75	100	-	-	ns
$\overline{TBRL}$	t_{THTH}	5	80	200	80	200	ns
		10	40	100	-	-	ns
Minimum Setup Time $\overline{TBRL}$ to Clock	t_{THC}	5	175	275	175	275	ns
		10	90	150	-	-	ns
Data to $\overline{TBRL}$ ↗	t_{DT}	5	20	50	20	50	ns
		10	0	40	-	-	ns
Minimum Hold-Time Data after $\overline{TBRL}$ ↗	t_{TD}	5	40	60	40	60	ns
		10	20	30	-	-	ns
Propagation Delay Time Clock to Data Start Bit	t_{CD}	5	300	450	300	450	ns
		10	150	225	-	-	ns
Clock to TBRE	t_{CT}	5	330	400	330	400	ns
		10	100	150	-	-	ns
$\overline{TBRL}$ to TBRE	t_{TTHR}	5	200	300	200	300	ns
		10	100	150	-	-	ns
Clock to TRE	t_{TTS}	5	330	400	330	400	ns
		10	100	150	-	-	ns

NOTES:

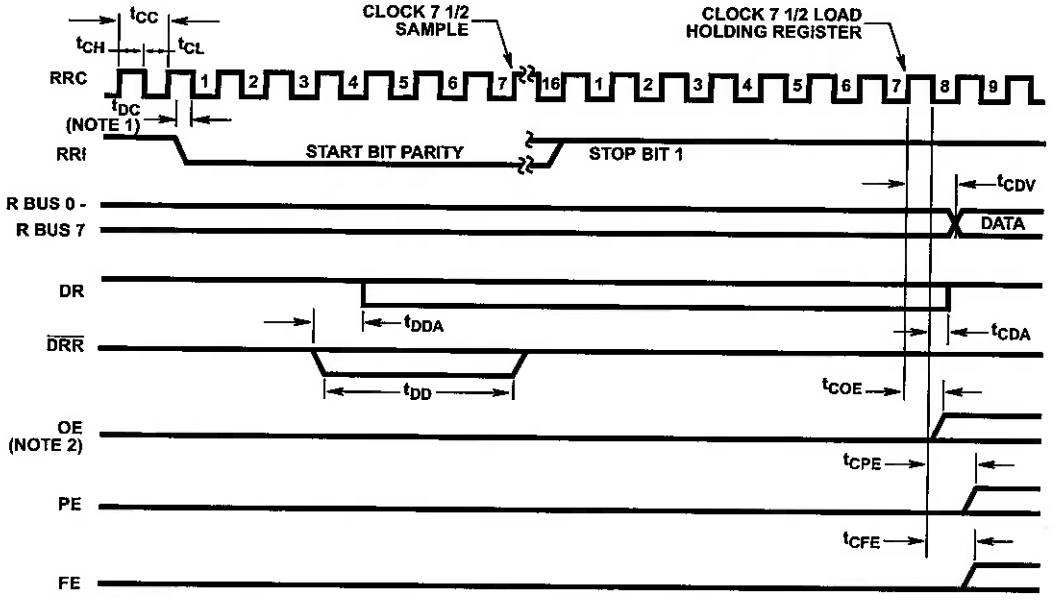
1. All measurements are made at the 50% point of the transition except three-state measurements.
2. Typical values for T_A = 25°C and nominal V_{DD}.
3. Maximum limits of minimum characteristics are the values above which all devices function.



NOTES:

1. The holding register is loaded on the trailing edge of $\overline{TBRL}$.
2. The transmitter shift register, if empty, is loaded on the first high-to-low transition of the clock which occurs at least 1/2 clock period + t_{THC} after the trailing edge of $\overline{TBRL}$ and transmission of a start bit occurs 1/2 clock period + t_{CD} later.

FIGURE 7. TRANSMITTER TIMING WAVEFORMS



NOTES:

1. If a start bit occurs at a time less than t_{DC} before a high-to-low transition of the clock, the start bit may not be recognized until the next high-to-low transition of the clock. The start bit may be completely asynchronous with the clock.
2. If a pending DA has not been cleared by a read of the receiver holding register by the time a new word is loaded into the receiver holding register, the OE signal will come true.

FIGURE 8. RECEIVER TIMING WAVEFORMS

Dynamic Electrical Specifications at T_A = -40°C to +85°C, V_{DD} ±5%, t_R, t_F = 20ns, V_{IH} = 0.7 V_{DD}, V_{IL} = 0.3 V_{DD}, C_L = 100pF

(NOTE 1) PARAMETERS		V_{DD} (V)	LIMITS				UNITS
			CDP6402		CDP6402C		
			(NOTE 2) TYP	(NOTE 3) MAX	(NOTE 2) TYP	(NOTE 3) MAX	
RECEIVER TIMING (See Figure 8)							
Minimum Clock Period (RRC)	t_{CC}	5	250	310	250	310	ns
		10	125	155	-	-	ns
Minimum Pulse Width Clock Low Level	t_{CL}	5	100	125	100	125	ns
		10	75	100	-	-	ns
Clock High Level	t_{CH}	5	100	125	100	125	ns
		10	75	100	-	-	ns
$\overline{\text{Data Received}}$ Reset	t_{DD}	5	50	75	50	75	ns
		10	25	40	-	-	ns
Minimum Setup Time Data Start Bit to Clock	t_{DC}	5	100	150	100	150	ns
		10	50	75	-	-	ns
Propagation Delay Time $\overline{\text{Data Received}}$ Reset to Data Received	t_{DDA}	5	150	250	150	250	ns
		10	75	125	-	-	ns
Clock to Data Valid	t_{CDV}	5	275	400	275	400	ns
		10	110	175	-	-	ns
Clock to DR	t_{CDA}	5	275	400	275	400	ns
		10	110	175	-	-	ns
Clock to Overrun Error	t_{COE}	5	275	400	275	400	ns
		10	100	150	-	-	ns
Clock to Parity Error	t_{CPE}	5	240	375	240	375	ns
		10	120	175	-	-	ns
Clock to Framing Error	t_{CFE}	5	200	300	200	300	ns
		10	100	150	-	-	ns

NOTES:

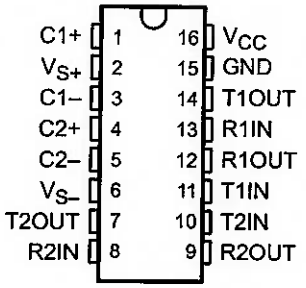
1. All measurements are made at the 50% point of the transition except three-state measurements.
2. Typical values for T_A = 25°C and nominal V_{DD}.
3. Maximum limits of minimum characteristics are the values above which all devices function.

MAX232, MAX232I
DUAL EIA-232 DRIVER/RECEIVER

SLLS047H – FEBRUARY 1989 – REVISED FEBRUARY 2002

- Operates With Single 5-V Power Supply
- LinBiCMOS™ Process Technology
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Designed to be Interchangeable With Maxim MAX232
- ESD Protection Exceeds JESD 22
  – 2000-V Human-Body Model (A114-A)
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers
- Package Options Include Plastic Small-Outline (D, DW, NS) Packages and Standard Plastic (N) DIPs

D, DW, N, OR NS PACKAGE
(TOP VIEW)



description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

The MAX232 is characterized for operation from 0°C to 70°C. The MAX232I is characterized for operation from –40°C to 85°C.

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES		
	SMALL OUTLINE (D, NS)	SMALL OUTLINE (DW)	PLASTIC DIP (N)
0°C to 70°C	MAX232D MAX232NS	MAX232DW	MAX232N
–40°C to 85°C	MAX232ID	MAX232IDW	MAX232IN

The D and DW packages are available taped and reeled by adding an R to the part number (i.e., MAX232DR). The NS package is only available taped and reeled.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) ..... −0.3 V to 6 V
Positive output supply voltage range, V_{S+} ..... V_{CC} −0.3 V to 15 V
Negative output supply voltage range, V_{S-} ..... −0.3 V to −15 V
Input voltage range, V_I: Driver ..... −0.3 V to V_{CC} + 0.3 V
Receiver ..... ±30 V
Output voltage range, V_O: T1OUT, T2OUT ..... V_{S-} −0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT ..... −0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT ..... Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 73°C/W
DW package ..... 57°C/W
N package ..... 67°C/W
NS package ..... 64°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C
Storage temperature range, T_{stg} ..... −65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	−40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (unless otherwise noted)

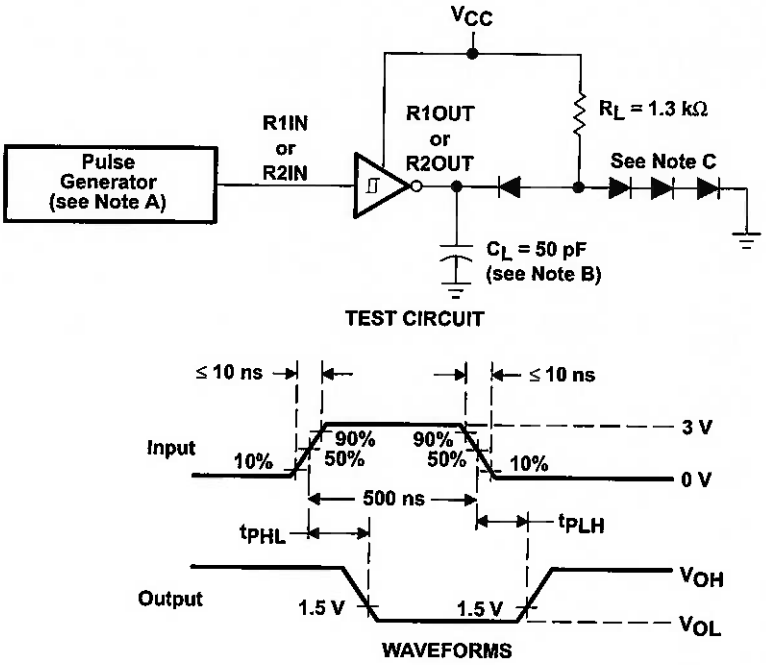
PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
		R1OUT, R2OUT	I_{OH} = –1 mA		3.5			
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
		R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	
VIT+	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
VIT–	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
Vhys	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
ri	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ
ro	Output resistance	T1OUT, T2OUT	V_{S+} = V_{S-} = 0,	V_O = ± 2 V	300			Ω
IOS§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
IIS	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA
ICC	Supply current		V_{CC} = 5.5 V, T_A = 25°C	All outputs open,		8	10	mA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	See Figure 1		500		ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	See Figure 1		500		ns
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(tr)	Driver transition region slew rate	See Figure 3		3		V/µs

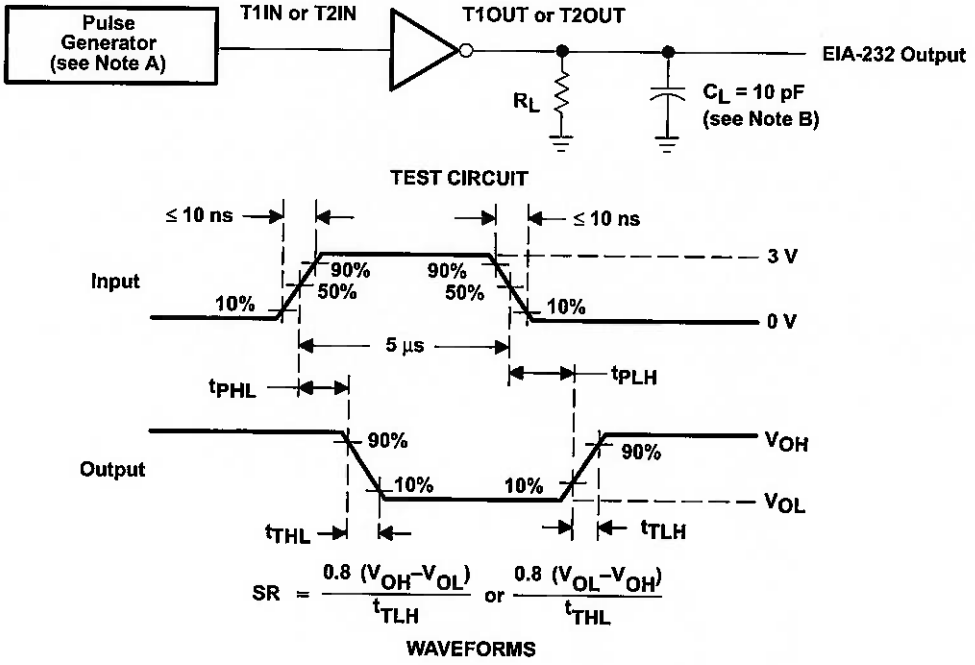
PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

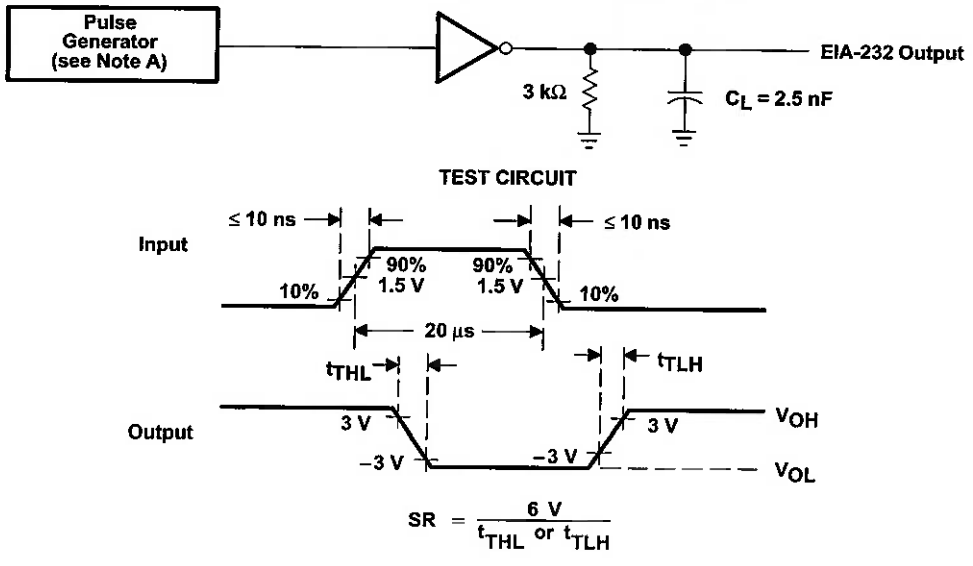
Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.

Figure 2. Driver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements (5-µs input)



NOTE A: The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.

Figure 3. Test Circuit and Waveforms for t_{THL} and t_{TLH} Measurements (20-µs input)

APPLICATION INFORMATION

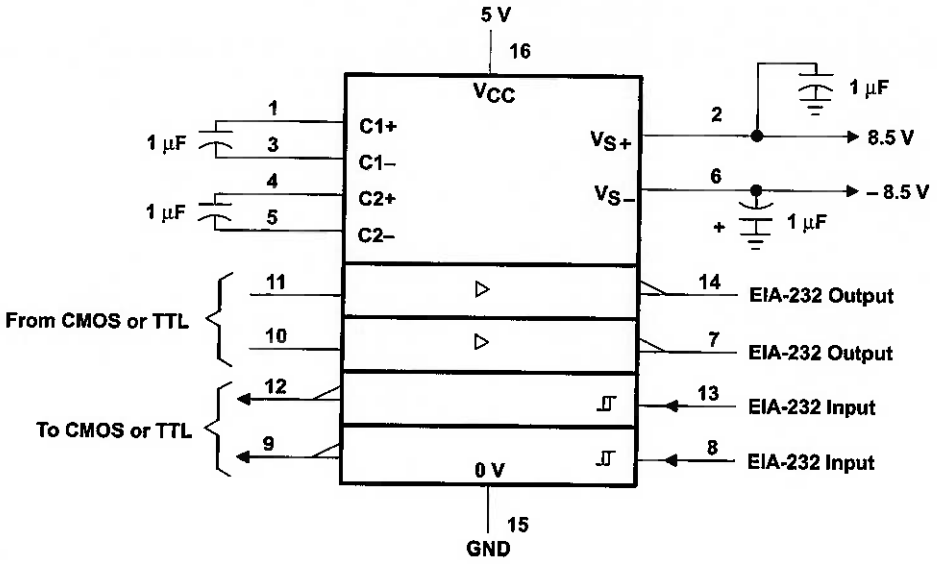


Figure 4. Typical Operating Circuit

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

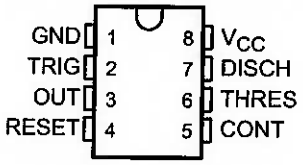
Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

NE555, SA555, SE555
PRECISION TIMERS

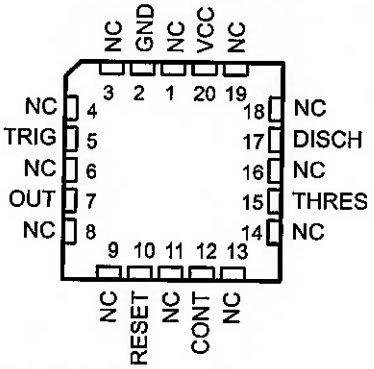
SLFS022C – SEPTEMBER 1973 – REVISED FEBRUARY 2002

- **Timing From Microseconds to Hours**
- **Astable or Monostable Operation**
- **Adjustable Duty Cycle**
- **TTL-Compatible Output Can Sink or Source up to 200 mA**
- **Designed To Be Interchangeable With Signetics NE555, SA555, and SE555**

NE555 . . . D, P, PS, OR PW PACKAGE
SA555 . . . D OR P PACKAGE
SE555 . . . D, JG, OR P PACKAGE
(TOP VIEW)



SE555 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

description

These devices are precision timing circuits capable of producing accurate time delays or oscillation. In the time-delay or monostable mode of operation, the timed interval is controlled by a single external resistor and capacitor network. In the astable mode of operation, the frequency and duty cycle can be controlled independently with two external resistors and a single external capacitor.

The threshold and trigger levels normally are two-thirds and one-third, respectively, of V_{CC}. These levels can be altered by use of the control-voltage terminal. When the trigger input falls below the trigger level, the flip-flop is set and the output goes high. If the trigger input is above the trigger level and the threshold input is above the threshold level, the flip-flop is reset and the output is low. The reset (RESET) input can override all other inputs and can be used to initiate a new timing cycle. When RESET goes low, the flip-flop is reset and the output goes low. When the output is low, a low-impedance path is provided between discharge (DISCH) and ground.

The output circuit is capable of sinking or sourcing current up to 200 mA. Operation is specified for supplies of 5 V to 15 V. With a 5-V supply, output levels are compatible with TTL inputs.

The NE555 is characterized for operation from 0°C to 70°C. The SA555 is characterized for operation from –40°C to 85°C. The SE555 is characterized for operation over the full military range of –55°C to 125°C.

AVAILABLE OPTIONS

T_A	PACKAGE					
	V_{THRES} MAX V_{CC} = 15 V	SMALL OUTLINE (D, PS)	CHIP CARRIER (FK)	CERAMIC DIP (JG)	PLASTIC DIP (P)	PLASTIC THIN SHRINK SMALL OUTLINE (PW)
0°C to 70°C	11.2 V	NE555D NE555PS	—	—	NE555P	NE555PW
–40°C to 85°C	11.2 V	SA555D	—	—	SA555P	—
–55°C to 125°C	10.6 V	SE555D	SE555FK	SE555JG	SE555P	—

The D package is available taped and reeled. Add the suffix R to the device type (e.g., NE555DR). The PS and PW packages are only available taped and reeled.

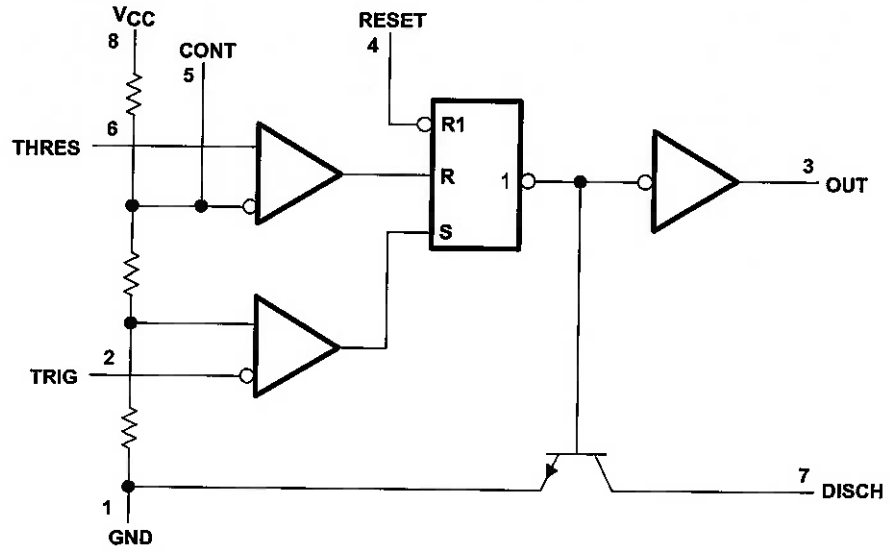
TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

FUNCTION TABLE

RESET	TRIGGER VOLTAGE†	THRESHOLD VOLTAGE†	OUTPUT	DISCHARGE SWITCH
Low	Irrelevant	Irrelevant	Low	On
High	<1/3 V_{DD}	Irrelevant	High	Off
High	>1/3 V_{DD}	>2/3 V_{DD}	Low	On
High	>1/3 V_{DD}	<2/3 V_{DD}	As previously established	

† Voltage levels shown are nominal.

functional block diagram



Pin numbers shown are for the D, JG, P, PS, and PW packages.
NOTE A: RESET can override TRIG, which can override THRES.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) ..... 18 V
Input voltage (CONT, RESET, THRES, and TRIG) ..... V_{CC}
Output current ..... ±225 mA
Continuous total dissipation ..... See Dissipation Rating Table
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 97°C/W
P package ..... 85°C/W
PS package ..... 95°C/W
PW package ..... 149°C/W
Case temperature for 60 seconds: FK package ..... 260°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds: D, P, PS, or PW package ..... 260°C
Lead temperature 1,6 mm (1/16 inch) from case for 60 seconds: JG package ..... 300°C
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to GND.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

DISSIPATION RATING TABLE

PACKAGE	$T_A \le 25°C$ POWER RATING	DERATING FACTOR ABOVE $T_A = 25°C$	$T_A = 70°C$ POWER RATING	$T_A = 85°C$ POWER RATING	$T_A = 125°C$ POWER RATING
FK	1375 mW	11.0 mW/°C	880 mW	715 mW	275 mW
JG (SE555)	1050 mW	8.4 mW/°C	672 mW	546 mW	210 mW

recommended operating conditions

			MIN	MAX	UNIT
V_{CC}	Supply voltage	SA555, NE555	4.5	16	V
		SE555	4.5	18	
V_I	Input voltage (CONT, RESET, THRES, and TRIG)			V_{CC}	V
I_O	Output current			±200	mA
T_A	Operating free-air temperature	NE555	0	70	°C
		SA555	–40	85	
		SE555	–55	125	

electrical characteristics, V_{CC} = 5 V to 15 V, T_A = 25°C (unless otherwise noted)

PARAMETER	TEST CONDITIONS		SE555			NE555 SA555			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
THRES voltage level	V_{CC} = 15 V		9.4	10	10.6	8.8	10	11.2	V
	V_{CC} = 5 V		2.7	3.3	4	2.4	3.3	4.2	
THRES current (see Note 3)				30	250		30	250	nA
TRIG voltage level	V_{CC} = 15 V		4.8	5	5.2	4.5	5	5.6	V
		T_A = –55°C to 125°C	3		6				
	V_{CC} = 5 V		1.45	1.67	1.9	1.1	1.67	2.2	
		T_A = –55°C to 125°C			1.9				
TRIG current	TRIG at 0 V			0.5	0.9		0.5	2	µA
RESET voltage level			0.3	0.7	1	0.3	0.7	1	V
	T_A = –55°C to 125°C				1.1				
RESET current	RESET at V_{CC}			0.1	0.4		0.1	0.4	mA
	RESET at 0 V			–0.4	–1		–0.4	–1.5	
DISCH switch off-state current				20	100		20	100	nA
CONT voltage (open circuit)	V_{CC} = 15 V		9.6	10	10.4	9	10	11	V
		T_A = –55°C to 125°C	9.6		10.4				
	V_{CC} = 5 V		2.9	3.3	3.8	2.6	3.3	4	
		T_A = –55°C to 125°C	2.9		3.8				
Low-level output voltage	V_{CC} = 15 V, I_{OL} = 10 mA			0.1	0.15		0.1	0.25	V
		T_A = –55°C to 125°C			0.2				
	V_{CC} = 15 V, I_{OL} = 50 mA			0.4	0.5		0.4	0.75	
		T_A = –55°C to 125°C			1				
	V_{CC} = 15 V, I_{OL} = 100 mA			2	2.2		2	2.5	
		T_A = –55°C to 125°C			2.7				
	V_{CC} = 15 V,	I_{OL} = 200 mA		2.5			2.5		
	V_{CC} = 5 V, I_{OL} = 3.5 mA	T_A = –55°C to 125°C			0.35				
	V_{CC} = 5 V, I_{OL} = 5 mA			0.1	0.2		0.1	0.35	
		T_A = –55°C to 125°C			0.8				
	V_{CC} = 5 V,	I_{OL} = 8 mA		0.15	0.25		0.15	0.4	
High-level output voltage	V_{CC} = 15 V, I_{OH} = –100 mA		13	13.3		12.75	13.3		V
		T_A = –55°C to 125°C	12						
	V_{CC} = 15 V,	I_{OH} = –200 mA		12.5			12.5		
	V_{CC} = 5 V, I_{OH} = –100 mA		3	3.3		2.75	3.3		
		T_A = –55°C to 125°C	2						
Supply current	Output low, No load	V_{CC} = 15 V		10	12		10	15	mA
		V_{CC} = 5 V		3	5		3	6	
	Output high, No load	V_{CC} = 15 V		9	10		9	13	
		V_{CC} = 5 V		2	4		2	5	

NOTE 3: This parameter influences the maximum value of the timing resistors R_A and R_B in the circuit of Figure 12. For example, when V_{CC} = 5 V, the maximum value is R = R_A + R_B ≈ 3.4 MΩ, and for V_{CC} = 15 V, the maximum value is 10 MΩ.

operating characteristics, V_{CC} = 5 V and 15 V

PARAMETER		TEST CONDITIONS†	SE555			NE555 SA555			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
Initial error of timing interval‡	Each timer, monostable§	T_A = 25°C		0.5%	1.5%*		1%	3%	
	Each timer, astable¶			1.5%			2.25%		
Temperature coefficient of timing interval	Each timer, monostable§	T_A = MIN to MAX		30	100*		50		ppm/°C
	Each timer, astable¶			90			150		
Supply-voltage sensitivity of timing interval	Each timer, monostable§	T_A = 25°C		0.05	0.2*		0.1	0.5	%/V
	Each timer, astable¶			0.15			0.3		
Output-pulse rise time		C_L = 15 pF, T_A = 25°C		100	200*		100	300	ns
Output-pulse fall time		C_L = 15 pF, T_A = 25°C		100	200*		100	300	ns

* On products compliant to MIL-PRF-38535, this parameter is not production tested.
† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ Timing interval error is defined as the difference between the measured value and the average value of a random sample from each process run.
§ Values specified are for a device in a monostable circuit similar to Figure 9, with the following component values: R_A = 2 kΩ to 100 kΩ, C = 0.1 μF.
¶ Values specified are for a device in an astable circuit similar to Figure 12, with the following component values: R_A = 1 kΩ to 100 kΩ, C = 0.1 μF.

TYPICAL CHARACTERISTICS†

LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

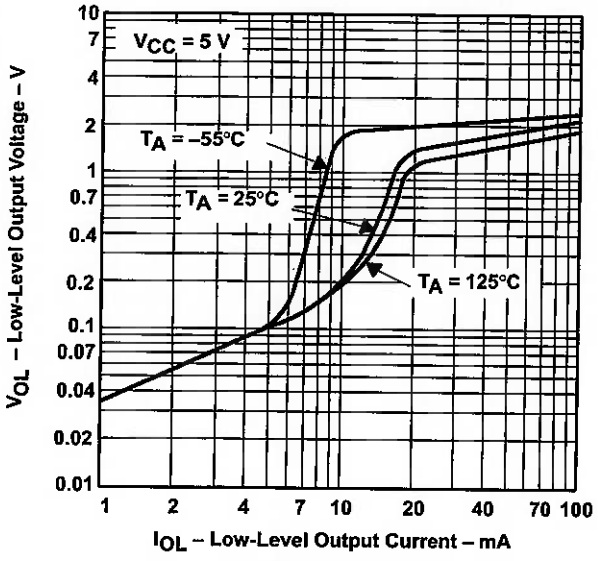


Figure 1

LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

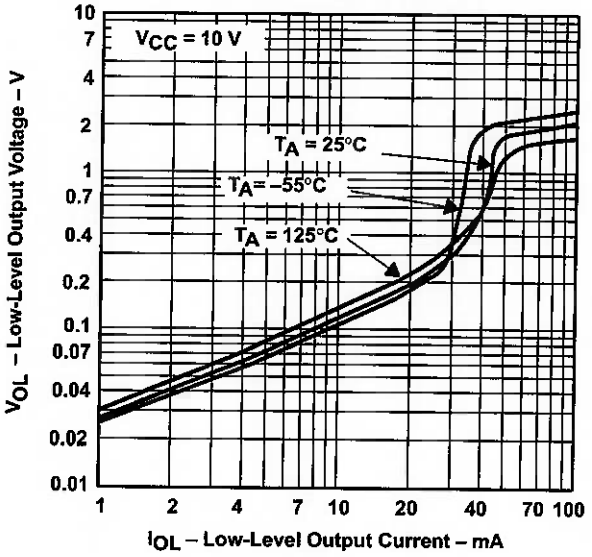


Figure 2

LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

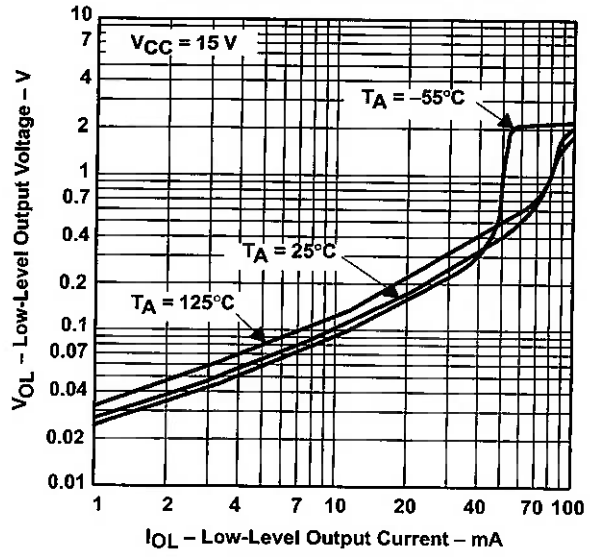


Figure 3

DROP BETWEEN SUPPLY VOLTAGE AND OUTPUT
vs
HIGH-LEVEL OUTPUT CURRENT

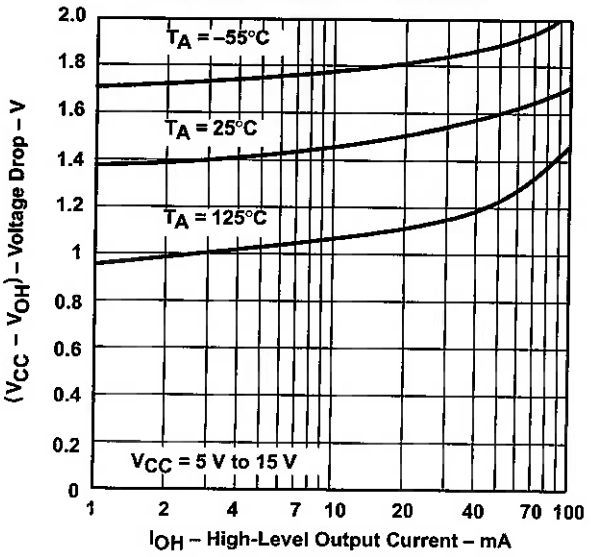


Figure 4

†Data for temperatures below 0°C and above 70°C are applicable for SE555 circuits only.

TYPICAL CHARACTERISTICS†

SUPPLY CURRENT
vs
SUPPLY VOLTAGE

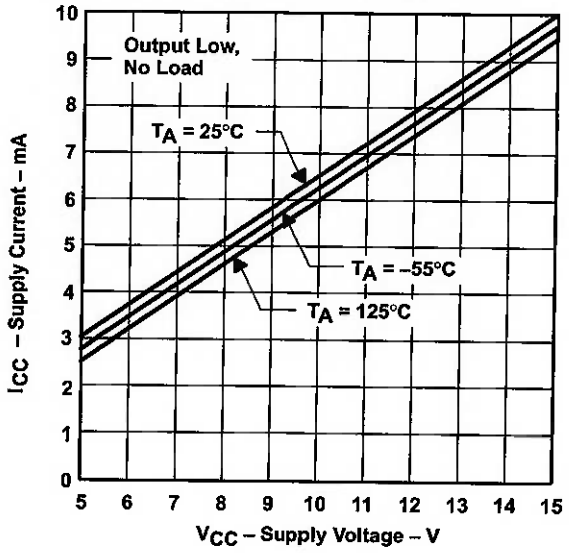


Figure 5

NORMALIZED OUTPUT PULSE DURATION
(MONOSTABLE OPERATION)
vs
SUPPLY VOLTAGE

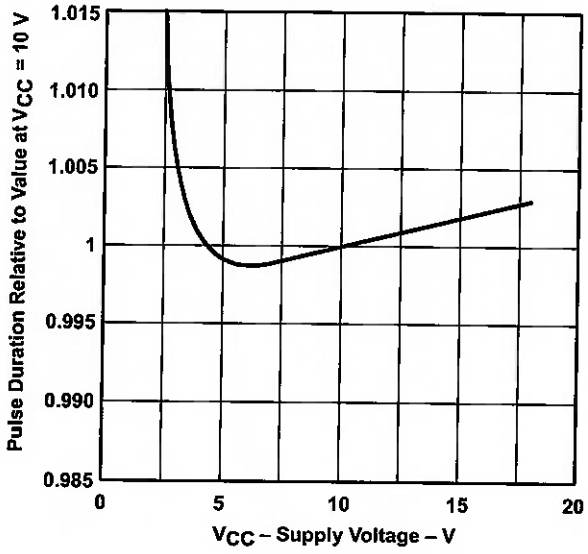


Figure 6

NORMALIZED OUTPUT PULSE DURATION
(MONOSTABLE OPERATION)
vs
FREE-AIR TEMPERATURE

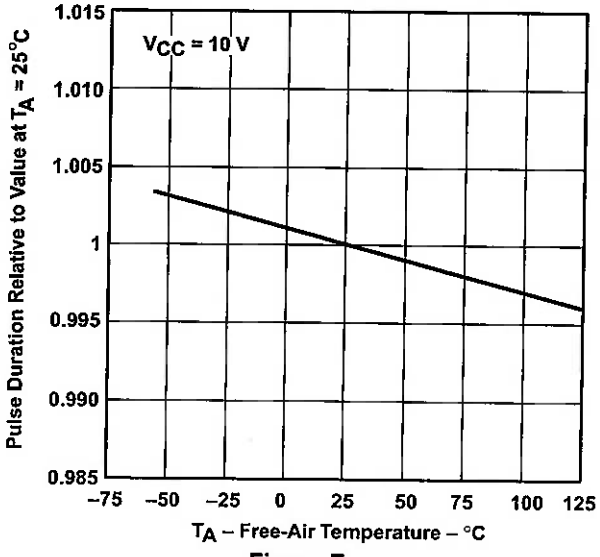


Figure 7

PROPAGATION DELAY TIME
vs
LOWEST VOLTAGE LEVEL
OF TRIGGER PULSE

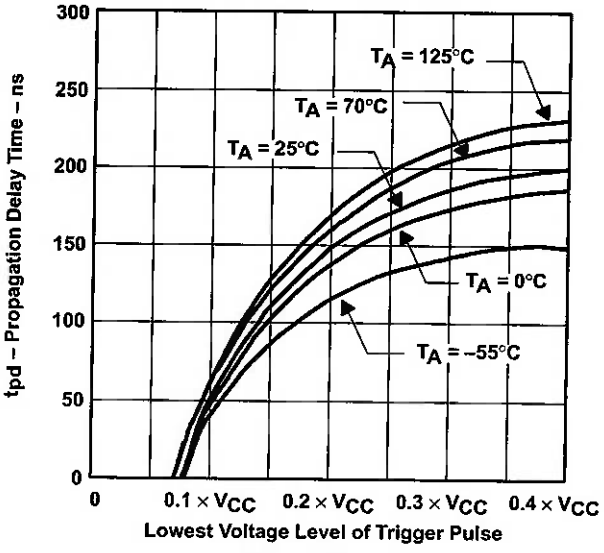


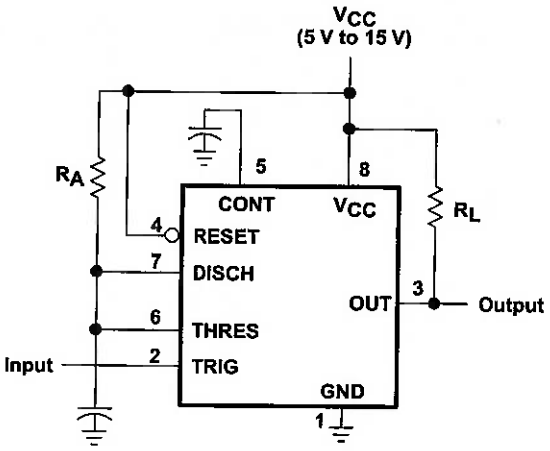
Figure 8

†Data for temperatures below 0°C and above 70°C are applicable for SE555 series circuits only.

APPLICATION INFORMATION

monostable operation

For monostable operation, any of these timers can be connected as shown in Figure 9. If the output is low, application of a negative-going pulse to the trigger (TRIG) sets the flip-flop ($\overline{Q}$ goes low), drives the output high, and turns off Q1. Capacitor C then is charged through R_A until the voltage across the capacitor reaches the threshold voltage of the threshold (THRES) input. If TRIG has returned to a high level, the output of the threshold comparator resets the flip-flop ($\overline{Q}$ goes high), drives the output low, and discharges C through Q1.



Pin numbers shown are for the D, JG, P, PS, and PW packages.

Figure 9. Circuit for Monostable Operation

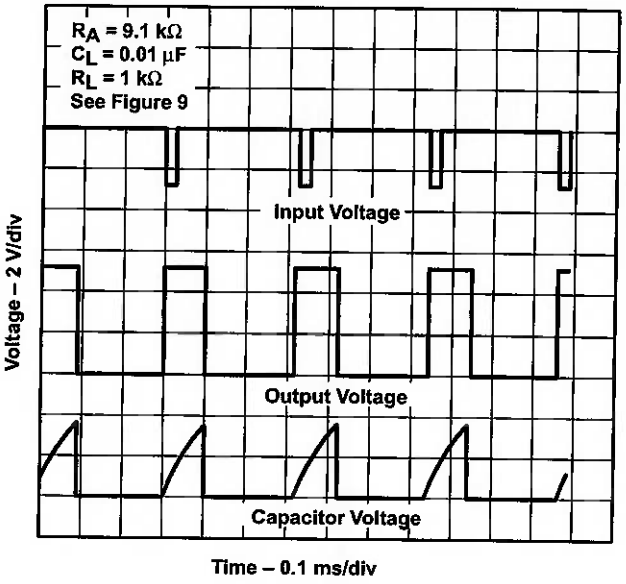


Figure 10. Typical Monostable Waveforms

Monostable operation is initiated when TRIG voltage falls below the trigger threshold. Once initiated, the sequence ends only if TRIG is high at the end of the timing interval. Because of the threshold level and saturation voltage of Q1, the output pulse duration is approximately $t_w = 1.1R_AC$. Figure 11 is a plot of the time constant for various values of R_A and C. The threshold levels and charge rates both are directly proportional to the supply voltage, V_{CC}. The timing interval is, therefore, independent of the supply voltage, so long as the supply voltage is constant during the time interval.

Applying a negative-going trigger pulse simultaneously to RESET and TRIG during the timing interval discharges C and reinitiates the cycle, commencing on the positive edge of the reset pulse. The output is held low as long as the reset pulse is low. To prevent false triggering, when RESET is not used, it should be connected to V_{CC}.

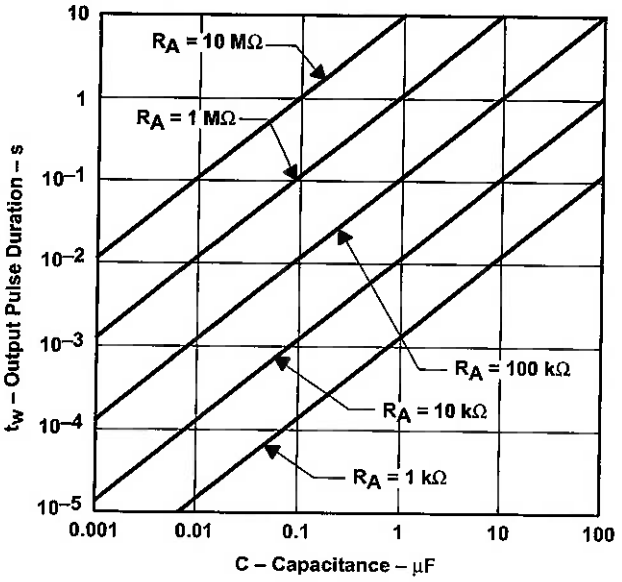


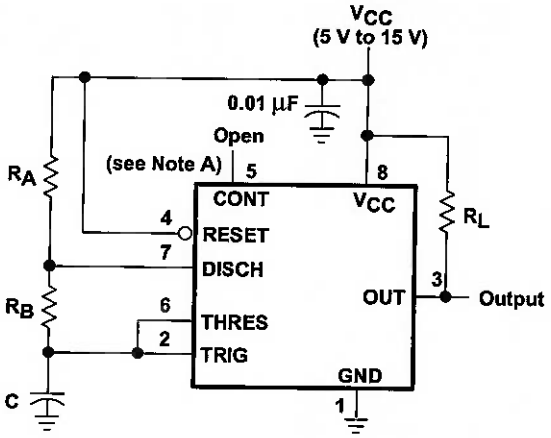
Figure 11. Output Pulse Duration vs Capacitance

APPLICATION INFORMATION

astable operation

As shown in Figure 12, adding a second resistor, R_B, to the circuit of Figure 9 and connecting the trigger input to the threshold input causes the timer to self-trigger and run as a multivibrator. The capacitor C charges through R_A and R_B and then discharges through R_B only. Therefore, the duty cycle is controlled by the values of R_A and R_B.

This astable connection results in capacitor C charging and discharging between the threshold-voltage level ($\approx 0.67 \times V_{CC}$) and the trigger-voltage level ($\approx 0.33 \times V_{CC}$). As in the monostable circuit, charge and discharge times (and, therefore, the frequency and duty cycle) are independent of the supply voltage.



Pin numbers shown are for the D, JG, P, PS, and PW packages.
NOTE A: Decoupling CONT voltage to ground with a capacitor can improve operation. This should be evaluated for individual applications.

Figure 12. Circuit for Astable Operation

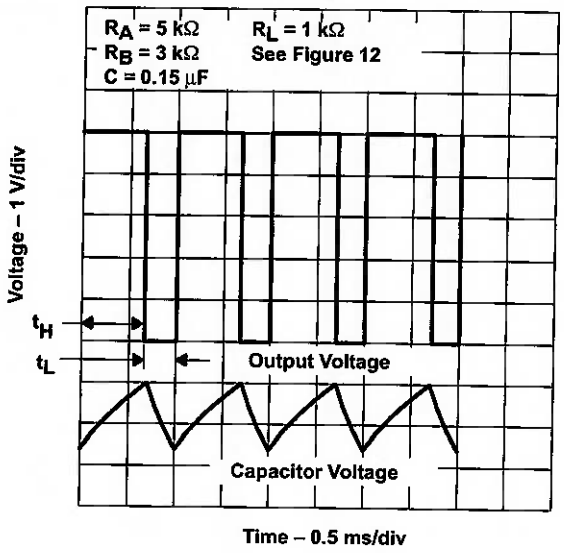


Figure 13. Typical Astable Waveforms

APPLICATION INFORMATION

astable operation (continued)

Figure 13 shows typical waveforms generated during astable operation. The output high-level duration t_H and low-level duration t_L can be calculated as follows:

$$t_H = 0.693\ (R_A + R_B)\ C$$

$$t_L = 0.693\ (R_B)\ C$$

Other useful relationships are shown below.

$$\text{period} = t_H + t_L = 0.693\ (R_A + 2R_B)\ C$$

$$\text{frequency} \approx \frac{1.44}{(R_A + 2R_B)\ C}$$

$$\text{Output driver duty cycle} = \frac{t_L}{t_H + t_L} = \frac{R_B}{R_A + 2R_B}$$

$$\text{Output waveform duty cycle} = \frac{t_H}{t_H + t_L} = 1 - \frac{R_B}{R_A + 2R_B}$$

$$\text{Low-to-high ratio} = \frac{t_L}{t_H} = \frac{R_B}{R_A + R_B}$$

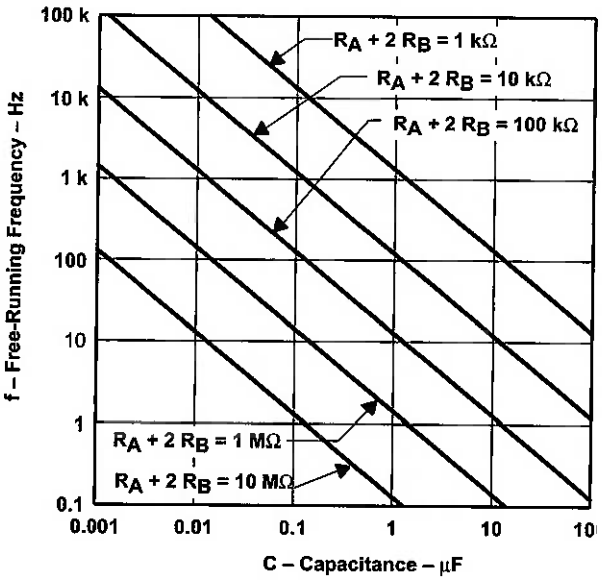


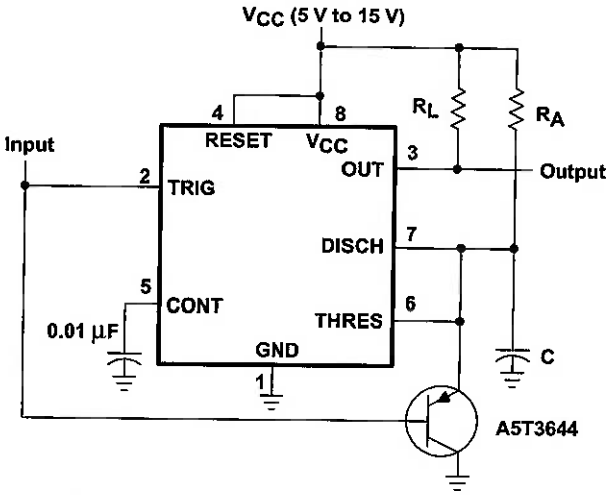
Figure 14. Free-Running Frequency

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

APPLICATION INFORMATION

missing-pulse detector

The circuit shown in Figure 15 can be used to detect a missing pulse or abnormally long spacing between consecutive pulses in a train of pulses. The timing interval of the monostable circuit is retriggered continuously by the input pulse train as long as the pulse spacing is less than the timing interval. A longer pulse spacing, missing pulse, or terminated pulse train permits the timing interval to be completed, thereby generating an output pulse as shown in Figure 16.



Pin numbers shown are shown for the D, JG, P, PS, and PW packages.

Figure 15. Circuit for Missing-Pulse Detector

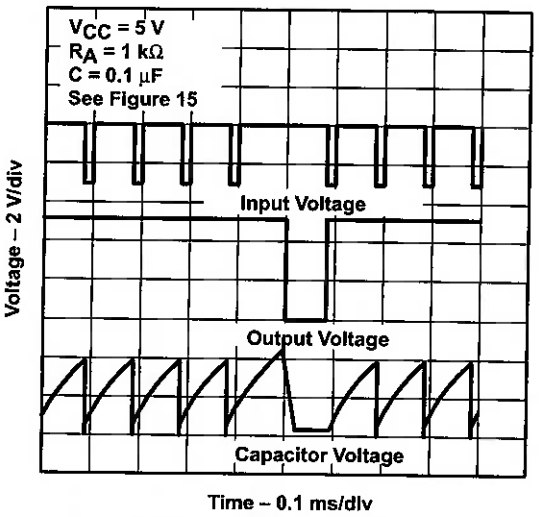


Figure 16. Completed-Timing Waveforms for Missing-Pulse Detector

APPLICATION INFORMATION

frequency divider

By adjusting the length of the timing cycle, the basic circuit of Figure 9 can be made to operate as a frequency divider. Figure 17 shows a divide-by-three circuit that makes use of the fact that retriggering cannot occur during the timing cycle.

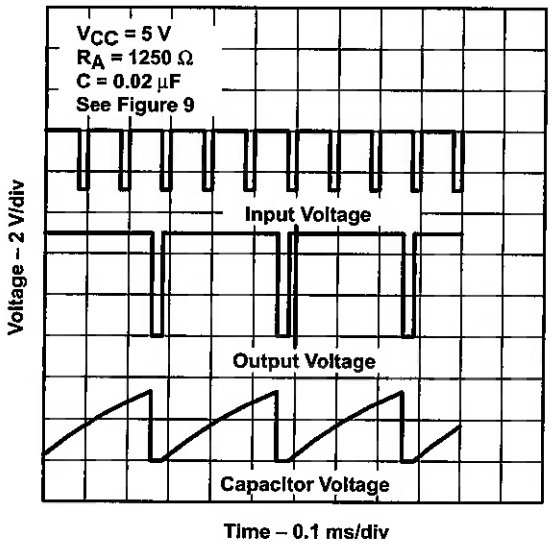
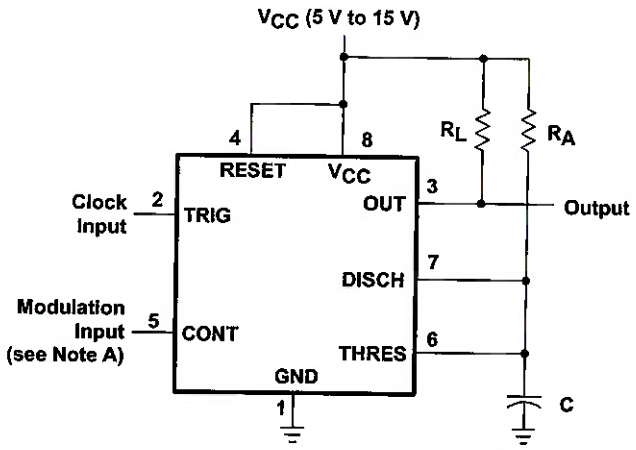


Figure 17. Divide-by-Three Circuit Waveforms

pulse-width modulation

The operation of the timer can be modified by modulating the internal threshold and trigger voltages, which is accomplished by applying an external voltage (or current) to CONT. Figure 18 shows a circuit for pulse-width modulation. A continuous input pulse train triggers the monostable circuit, and a control signal modulates the threshold voltage. Figure 19 shows the resulting output pulse-width modulation. While a sine-wave modulation signal is illustrated, any wave shape could be used.

APPLICATION INFORMATION



Pin numbers shown are for the D, JG, P, PS, and PW packages.

NOTE A: The modulating signal can be direct or capacitively coupled to CONT. For direct coupling, the effects of modulation source voltage and impedance on the bias of the timer should be considered.

Figure 18. Circuit for Pulse-Width Modulation

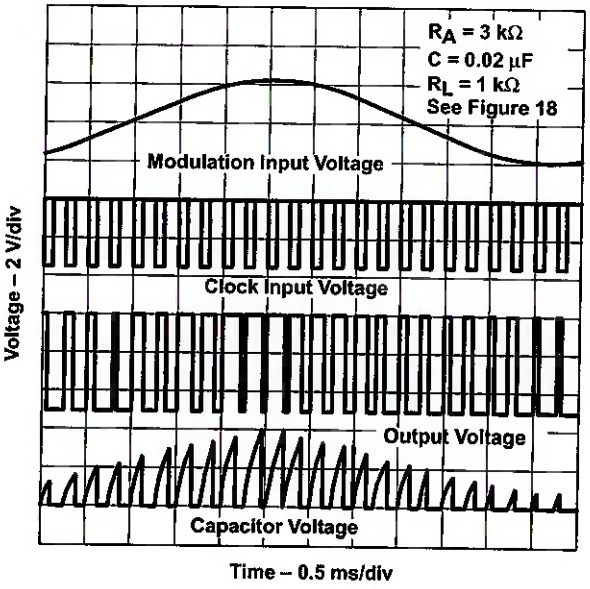
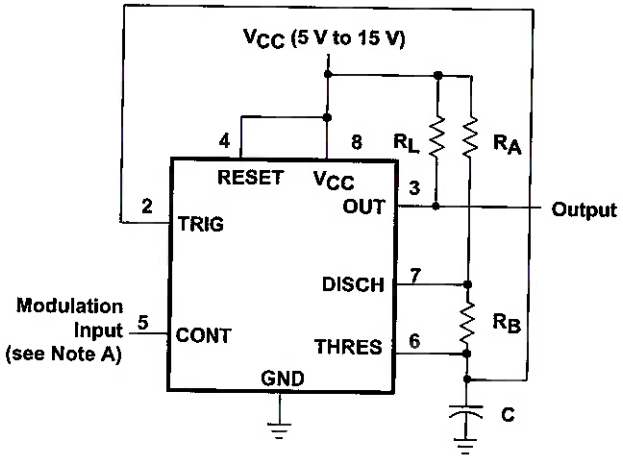


Figure 19. Pulse-Width-Modulation Waveforms

pulse-position modulation

As shown in Figure 20, any of these timers can be used as a pulse-position modulator. This application modulates the threshold voltage and, thereby, the time delay, of a free-running oscillator. Figure 21 shows a triangular-wave modulation signal for such a circuit; however, any wave shape could be used.



Pin numbers shown are for the D, JG, P, PS, and PW packages.

NOTE A: The modulating signal can be direct or capacitively coupled to CONT. For direct coupling, the effects of modulation source voltage and impedance on the bias of the timer should be considered.

Figure 20. Circuit for Pulse-Position Modulation

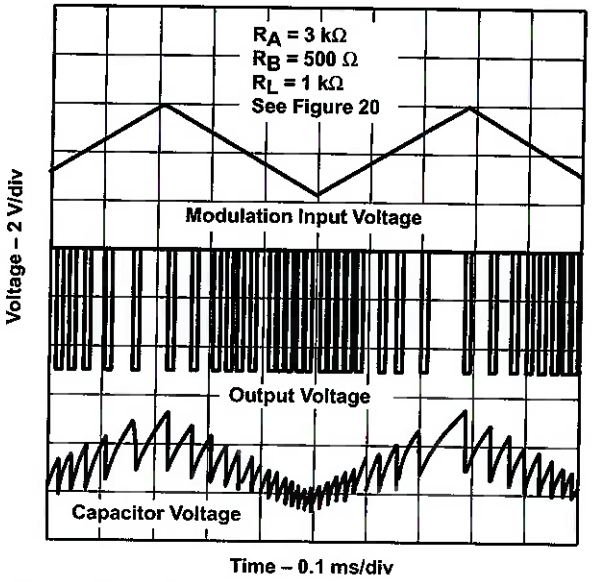
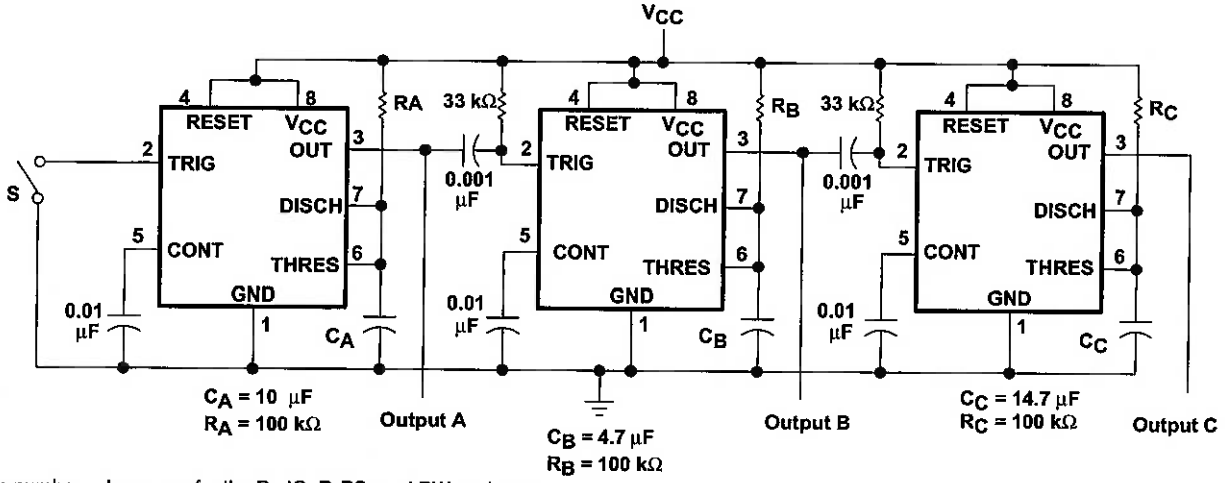


Figure 21. Pulse-Position-Modulation Waveforms

APPLICATION INFORMATION

sequential timer

Many applications, such as computers, require signals for initializing conditions during start-up. Other applications, such as test equipment, require activation of test signals in sequence. These timing circuits can be connected to provide such sequential control. The timers can be used in various combinations of astable or monostable circuit connections, with or without modulation, for extremely flexible waveform control. Figure 22 shows a sequencer circuit with possible applications in many systems, and Figure 23 shows the output waveforms.



Pin numbers shown are for the D, JG, P, PS, and PW packages.
NOTE A: S closes momentarily at t = 0.

Figure 22. Sequential Timer Circuit

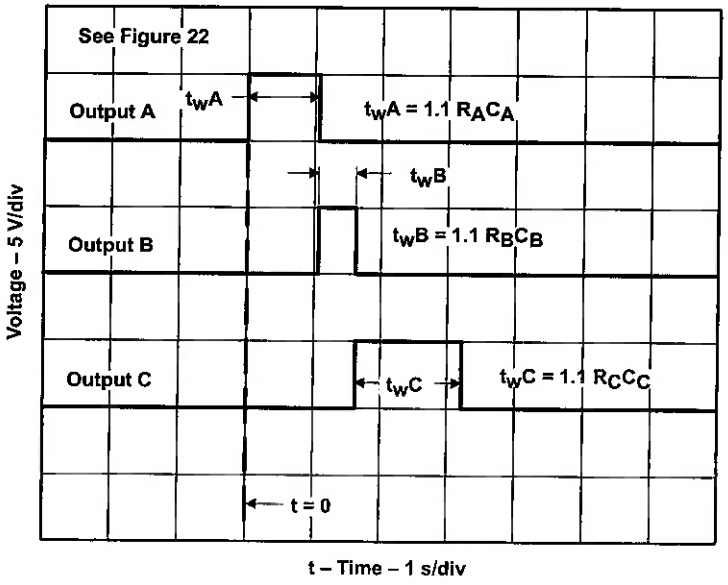


Figure 23. Sequential Timer Waveforms

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

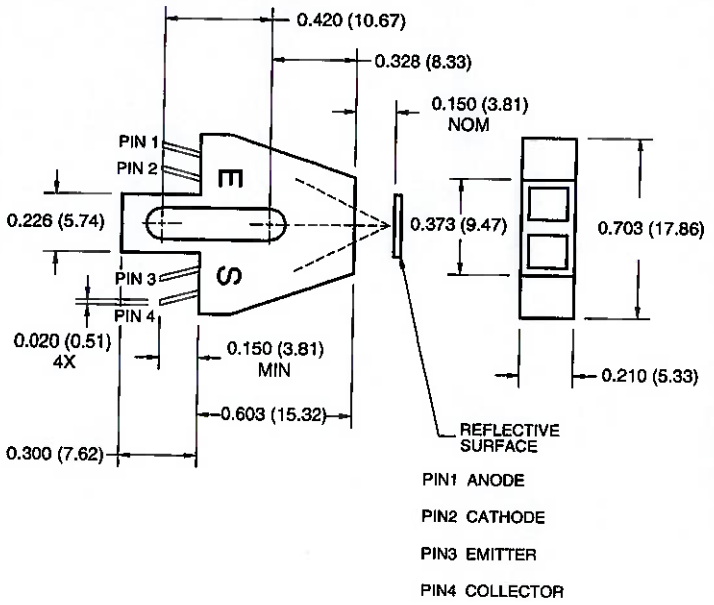
Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

FAIRCHILD
SEMICONDUCTOR®

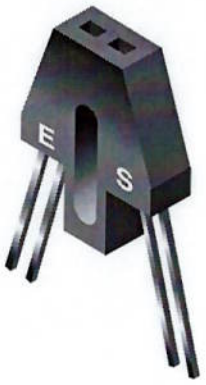
PHOTOTRANSISTOR REFLECTIVE OBJECT SENSOR

QRB1113 QRB1114

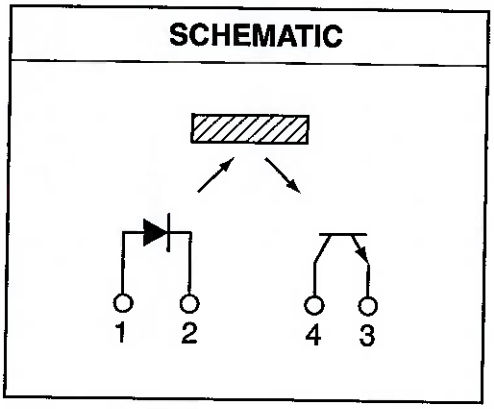
PACKAGE DIMENSIONS



NOTES:
1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.



SCHEMATIC



DESCRIPTION

The QRB1113/1114 consists of an infrared emitting diode and an NPN silicon phototransistor mounted side by side on a converging optical axis in a black plastic housing. The phototransistor responds to radiation from the emitting diode only when a reflective object passes within its field of view. The area of the optimum response approximates a circle .200" in diameter.

FEATURES

- No contact surface sensing
- Phototransistor output
- Focused for sensing specular reflection
- Daylight filter on photosensor
- Dust cover

3/5/02 DS300350

FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR REFLECTIVE OBJECT SENSOR

QRB1113 QRB1114

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)			
Parameter	**Symbol**	**Rating**	**Units**
Operating Temperature	T_{OPR}	-40 to +85	°C
Storage Temperature	T_{STG}	-40 to +85	°C
Soldering Temperature (Iron)(2,3,4)	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)(2,3)	T_{SOL-F}	260 for 10 sec	°C
EMITTER			
Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	5	V
Power Dissipation(1)	P_D	100	mW
SENSOR			
Collector-Emitter Voltage	V_{CEO}	30	V
Emitter-Collector Voltage	V_{ECO}	4.5	V
Collector Current		20	mA
Power Dissipation(1)	P_D	100	mW

NOTES
1. Derate power dissipation linearly 1.67 mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron 1/16" (1.6mm) minimum from housing.
5. D is the distance from the assembly face to the reflective surface.
6. Measured using an Eastman Kodak neutral test card with 90% diffused reflecting surface.
7. Cross talk is the photo current measured with current to the input diode and no reflecting surface.

ELECTRICAL/OPTICAL CHARACTERISTICS (T_A = 25°C)						
Parameter	**Test Conditions**	**Symbol**	**Min.**	**Typ.**	**Max.**	**Units**
EMITTER						
Forward Voltage	I_F = 40 mA	V_F	—	—	1.7	V
Reverse Current	V_R = 5.0 V	I_R	—	—	100	µA
Peak Emission Wavelength	I_F = 20 mA	λ_{PE}	—	940	—	nm
SENSOR						
Collector-Emitter Breakdown Voltage	I_C = 1 mA	BV_{CEO}	30	—	—	V
Emitter-Collector Breakdown Voltage	I_E = 0.1 mA	BV_{ECO}	5	—	—	V
Collector-Emitter Dark Current	V_{CE} = 10 V, I_F = 0 mA	I_{CEO}	—	—	100	nA
COUPLED						
On-state Collector Current						
QRB1113	I_F = 40 mA, V_{CE} = 5 V D = .150"(5,6)	$I_{C(ON)}$	0.20	—	—	mA
QRB1114			0.60	—		
Collector-Emitter Saturation Voltage	I_F = 20 mA, I_C = 0.5 mA	$V_{CE\ (SAT)}$	—	—	0.4	V
Rise Time	V_{CE} = 5 V, R_L = 100 V $I_{C(ON)}$ = 5 mA	t_r	—	8	—	µs
Fall Time		t_f	—	8	—	
Cross Talk	I_F = 40 mA, V_{CE} = 5 V(7)	I_{CX}	—	—	1.00	µA

FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR REFLECTIVE OBJECT SENSOR

QRB1113 QRB1114

TYPICAL PERFORMANCE CURVES

Fig. 1 Forward Voltage vs. Forward Current

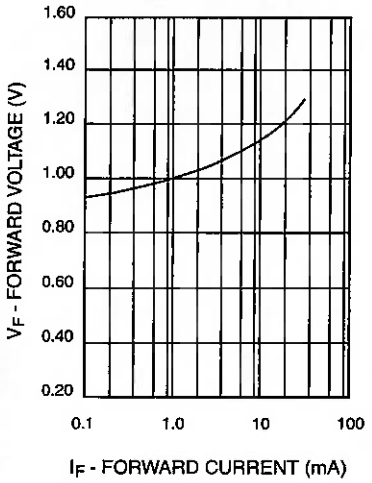


Fig. 2 Normalized Collector Current vs. Forward Current

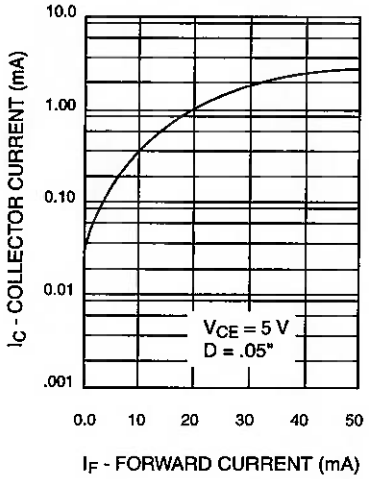


Fig. 3 Normalized Collector Current vs. Temperature

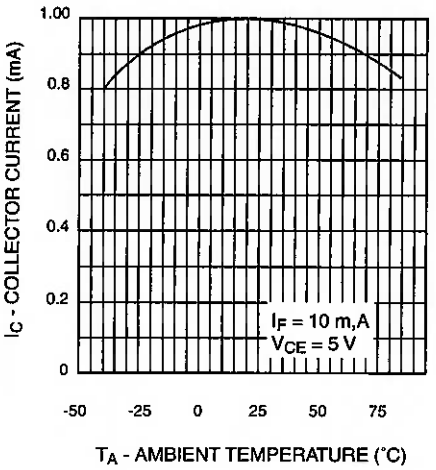


Fig. 4 Normalized Collector Dark Current vs. Temperature

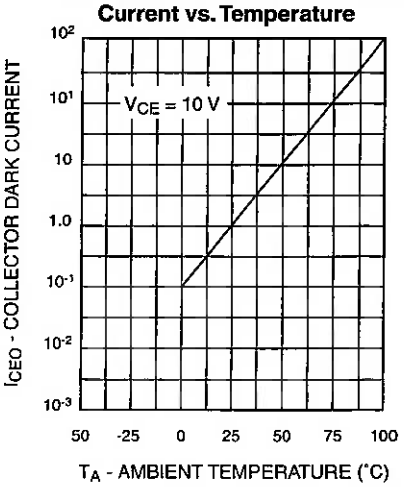
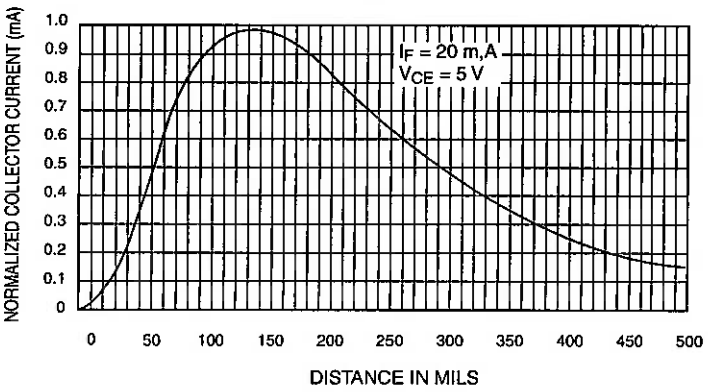


Fig. 5 Normalized Collector Current vs. Distance



FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR REFLECTIVE OBJECT SENSOR

QRB1113 QRB1114

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.