

Università degli studi di Roma “La Sapienza”



Dipartimento di Informatica e Sistemistica

Tesi di Laurea
in
Ingegneria Informatica

Progetto e Realizzazione di una Base Robotica Mobile

Relatore:
Prof. Daniele Nardi
Università di Roma
“La Sapienza”

Laureando:
Flavio Cappelli
mat. 09060065

Anno Accademico
1999/2000

Indice

Capitolo 1**LA ROBOTICA MOBILE****1.1*****Introduzione*****1.2*****Strutture mobili***

1.2.1

Differential drive

1.2.2

Ackerman steering

1.2.3

Tricycle drive

1.2.4

Tank treads

1.2.5

Synchro drive

1.2.6

Omnidirectional drive

1.3***Attuatori***

1.3.1

Motori in corrente continua

1.3.2

Motori senza spazzole (brushless)

1.3.3

Motori passo-passo (stepper)

1.4***Sensori***

1.4.1

Encoder ottici

1.4.2

Accelerometri e giroscopi

1.4.3

Sensori per telemetria

1.4.4

GPS e radiofari

1.4.5

Telecamere

1.4.6

Altri tipi di sensori

Capitolo 2**MODELLISTICA DEL ROBOT****2.1*****Modello dinamico degli attuatori e del carico***

2.1.1

Il servomotore

2.1.2

La trasmissione

2.1.3

La riduzione

2.1.4

Il carico

2.2***Cinematica del robot***

2.2.1

Cinematica diretta

2.2.2

Cinematica inversa

2.2.3

Odometria ed ambiguità di posizionamento

Capitolo 3**AZIONAMENTO E CONTROLLO****3.1*****Controllo del Robot***

3.1.1

Controllo dei servomotori

3.1.2

Considerazioni sul controllo PID

3.1.3

Controllore PID a tempo discreto

3.2***Amplificatori di potenza***

3.2.1

Amplificatori lineari

3.2.2

Amplificatori a commutazione

3.2.3

Ponte H a BJT e a MOSFET

Capitolo 4**PROGETTO E REALIZZAZIONE****4.1*****Descrizione generale*****4.2*****Scelta dei motori e degli accumulatori*****4.3*****Realizzazione dell'hardware***

4.3.1

Scheda di controllo

4.3.2

Scheda di potenza

4.4***Realizzazione del driver in Realtime Linux***

4.4.1

Perché occorre un sistema realtime

4.4.2

Il sistema operativo RTLinux

4.4.3

Il driver del dispositivo

4.5***Conclusioni*****APPENDICI****A1*****Schemi elettrici*****A2*****Circuiti stampati*****A3*****Funzioni logiche (GAL)*****A4*****Codice C del driver*****A5*****Datasheets*****A6*****La RoboCup e la squadra ART*****A7*****Richiami di fisica*****BIBLIOGRAFIA**

Capitolo 1

LA ROBOTICA MOBILE

1.1 Introduzione

Un robot è un sistema artificiale capace di apportare modifiche all'ambiente in cui opera, espliando azioni che sono condizionate da un insieme di regole di comportamento e da un insieme di informazioni acquisite dal robot sul suo stato e su quello dell'ambiente.

La capacità di agire sull'ambiente è offerta da un *sistema meccanico*, dotato in generale di organi di locomozione per muoversi nell'ambiente e di organi di manipolazione per intervenire sugli oggetti presenti nello stesso.

La capacità di percezione è affidata ad un *sistema sensoriale*, in grado di acquisire informazioni sullo stato interno del sistema meccanico e sullo stato esterno dell'ambiente.

La funzione di connettere in maniera intelligente percezione e azione è affidata ad un *sistema di calcolo*, in grado di eseguire programmi cognitivi con alto livello di astrazione. Questi ultimi comprendono algoritmi di analisi delle percezioni, di pianificazione del moto, di apprendimento e di ragionamento.

Un robot è quindi in grado di "interagire" con l'ambiente, mediante elaborazioni intelligenti delle informazioni ed esecuzioni di determinate operazioni motorie. Evolvendosi in questa direzione, la *robotica* va sempre più ad identificarsi come la scienza che studia la “*connessione intelligente fra percezione e azione*”, definizione che meglio di altre mette a fuoco la natura di questa disciplina.

Una branca significativa della robotica è *la robotica mobile*. Un robot mobile è un automa svincolato dalla postazione fissa, in grado cioè di muoversi liberamente e di interagire con un ambiente più o meno strutturato. Tralasciando quelli che sono i compiti che l'automa può svolgere nel suo cammino, la robotica mobile focalizza i suoi studi per rendere la macchina in grado di eseguire le seguenti operazioni [Leonard e Durrant-Whyte, 1991]:

- localizzare la propria posizione nell'ambiente;
- determinare la posizione successiva da raggiungere;
- stabilire qual è il percorso da seguire per arrivare in tale posizione.

Ovviamente lo studio di un robot mobile è strutturato su più livelli, dalla creazione di modelli del mondo reale (dove il veicolo dovrà muoversi) all'implementazione di algoritmi per la pianificazione delle traiettorie e la navigazione, fino allo studio dei sensori e degli attuatori che permettono la locomozione, e all'integrazione di tutte le parti. È pertanto evidente il carattere interdisciplinare della robotica, che si articola nelle aree culturali della meccanica, dell'elettronica, dell'automatica e dell'informatica.

In questo lavoro di tesi ci siamo occupati del livello più basso, cioè dello studio degli attuatori e dei sensori, della realizzazione di una struttura hardware e software per il movimento del veicolo e per la comunicazione con le architetture di livello più elevato.

1.2 Strutture mobili

La struttura dei robot dipende fortemente dallo scopo per cui sono progettati. La modalità di spostamento di un robot in ambiente terrestre non fa eccezioni e può essere di due tipi.

La prima è quella adottata dai mezzi dotati di *gambe* e consiste nel continuo spostamento del peso tra i diversi punti d'appoggio (le gambe), in modo da consentire l'avanzamento dell'arto libero dal peso nella direzione desiderata. Esempi di robot con gambe sono gli AIBO della Sony e il più recente robot umanoide P3 della Honda.

La seconda modalità è la più semplice e la più utilizzata, cioè quella su *ruote*: il veicolo poggia il suo peso su tutte le ruote (anche se in proporzioni diverse) e l'azione di movimento è ottenuta orientando in modo opportuno una o più ruote ed imprimendo la rotazione desiderata ad almeno una di esse. Poiché prevediamo che il nostro veicolo dovrà muoversi su superfici piane adottiamo la soluzione su ruote.

Ovviamente vi sono diverse tipologie di veicoli su ruote: la scelta della struttura meccanica, più adatta all'applicazione che si vuole realizzare, comporta un compromesso tra diversi fattori, sia tecnici sia organizzativi. Per quanto riguarda i primi è usuale domandarsi:

- con quali ambienti il robot deve confrontarsi;
- quali sono le possibilità di movimento richieste;
- come il controllo viene condizionato dalla meccanica.

Gli aspetti organizzativi riguardano invece il tempo e il denaro che si vogliono investire nella realizzazione del sistema di movimentazione nel suo complesso.

Uno dei problemi associati al movimento è la determinazione del cammino percorso dal robot. È essenziale infatti che alla macchina sia costantemente nota la sua posizione rispetto ad un riferimento esterno presente nell'ambiente. Come vedremo nel paragrafo 1.4, vi sono vari metodi, ma il più semplice è quello di ricavare lo spazio percorso considerando le velocità istantanee delle ruote e gli angoli di sterzata. Questa misura, indicata con il termine di *odometria*, ha il vantaggio di non richiedere alcun oggetto o supporto esterno al robot. Lo schiacciamento delle ruote e l'eventuale slittamento rendono questo metodo poco accurato: gli errori si accumulano e la stima della posizione diventa inutilizzabile entro breve tempo. Questo rende necessario il ricorso periodico ad una procedura di individuazione di un riferimento esterno, per riportare l'errore di posizione a valori accettabili [Cox, 1991].

Vediamo ora una panoramica delle diverse tipologie di veicoli su ruote. La base meccanica più semplice ha tre gradi di libertà, la *posizione* (x,y) e l'*orientazione* θ . Robot più complessi, come quelli articolati, sono costituiti da più basi accoppiate e possiedono un numero maggiore di gradi di libertà. Ma per la maggior parte delle applicazioni e delle ricerche, una base sola è sufficiente. Anche restando in quest'ambito si possono avere vari metodi di locomozione [Borenstein *et al.*, 1996].

1.2.1 DIFFERENTIAL DRIVE

Questo sistema è composto da due ruote motrici indipendenti ed almeno una ruota libera di ruotare e sterzare, detta *castor* (figura 1.1).

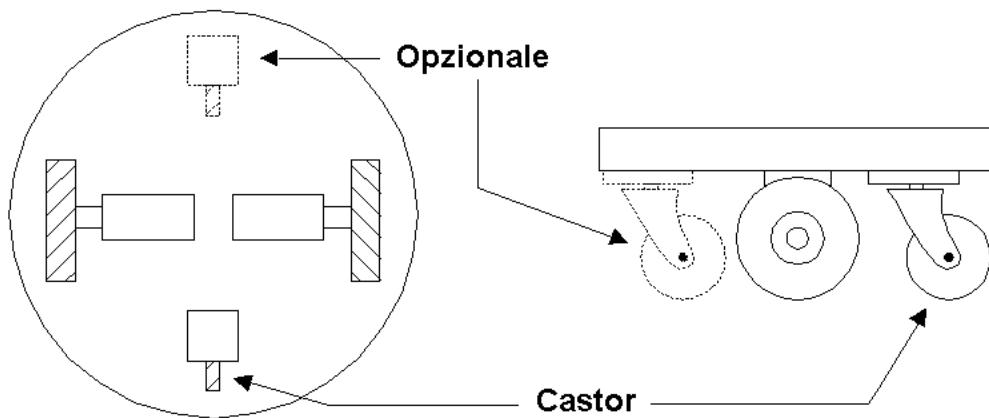


Figura 1.1 – Base mobile con configurazione “differential drive”.

La presenza di un secondo castor migliora la stabilità del sistema durante il moto, ma crea problemi se il terreno non è perfettamente piano: infatti in presenza di asperità una delle ruote motrici potrebbe non fare più presa sul terreno. Bisogna quindi utilizzare sistemi di sospensione che complicano il progetto meccanico. Il controllo di velocità sulle due ruote permette di ottenere la sterzata variando il loro rapporto; inoltre, maggiore è la definizione che si vuole ottenere per la sterzata, maggiore sarà la precisione richiesta ai controllori di velocità. Torneremo comunque su questo argomento, in quanto questa è la tipologia adottata per il nostro veicolo mobile (paragrafi 2.1, 2.2 e 3.1).

1.2.2 ACKERMAN STEERING

Usato nell'industria automobilistica, il sistema di sterzata “Ackerman” è progettato in modo da assicurare che, durante la sterzata, la ruota interna alla

curva sia ruotata di un angolo leggermente maggiore rispetto a quello della ruota esterna, evitando lo slittamento dei pneumatici (figura 1.2).

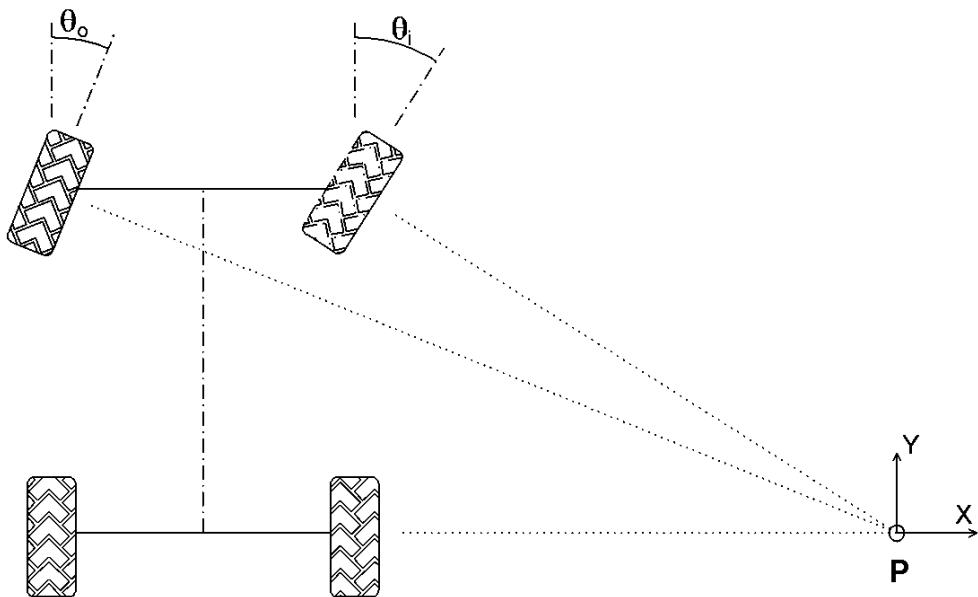


Figura 1.2 – Sistema “Ackerman”: gli assi si intersecano in un punto.

In pratica i due assi delle ruote anteriori vanno ad intersecarsi in un punto P , che giace sull’asse delle ruote posteriori. Tracciando delle circonferenze, passanti per le ruote e con centro in P , si comprende come i vettori velocità siano tutti tangenti a queste circonferenze. In tal modo si ottiene una odometria abbastanza accurata e contemporaneamente un sistema di trazione potente ed adatto a diversi tipi di terreno (purché sia dotato di sospensioni tali da garantire il continuo contatto delle ruote col terreno). Grazie alla presenza delle quattro ruote la stabilità in curva è molto buona. Il sistema “Ackerman” è dunque il più idoneo per veicoli autonomi da esterno. La trazione può essere anteriore, posteriore od integrale; in ogni caso la coppia, fornita dal motore, è trasmessa alle ruote motrici tramite differenziali meccanici.

1.2.3 TRICYCLE DRIVE

Questo sistema impiega una ruota anteriore attuata sia in rotazione sia in sterzata, e due ruote passive (figura 1.3).

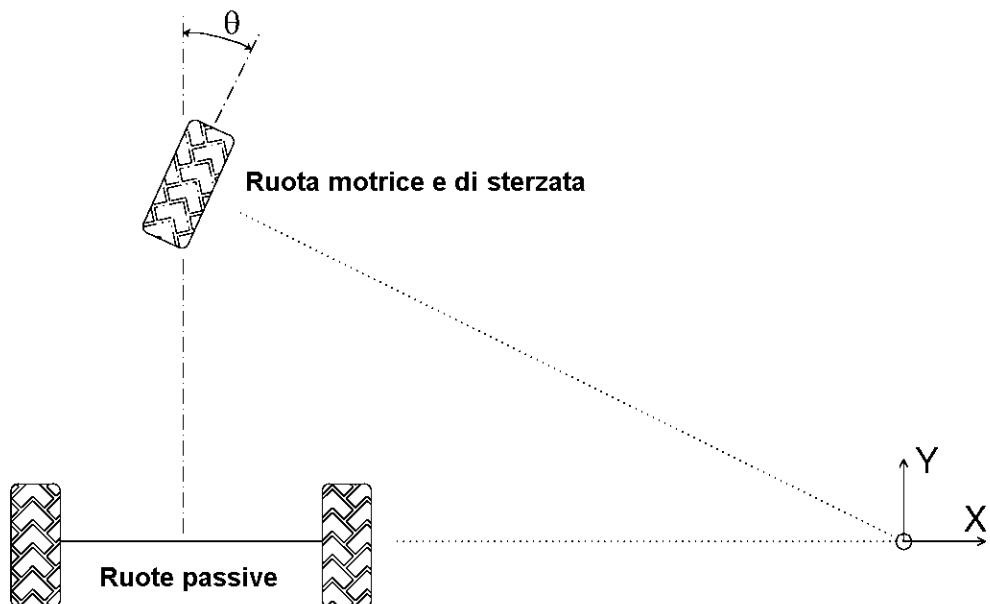


Figura 1.3 – Configurazione “tricycle drive”.

In alternativa la coppia motrice può essere fornita dal motore alle ruote posteriori tramite un differenziale meccanico e la ruota anteriore viene utilizzata solo per la sterzata. Questa configurazione è abbastanza comune data la sua semplicità, ma presenta minore stabilità in curva rispetto al sistema “Ackerman”. Inoltre se il veicolo sale su una rampa, il centro di gravità tende ad allontanarsi dalla ruota anteriore, con conseguente perdita di trazione. Un altro problema (in comune con la configurazione precedente) è che l'attuatore di sterzata deve esercitare una coppia elevata quando il veicolo è fermo e una coppia abbastanza piccola quando il veicolo è in moto. L'odometria può essere determinata leggendo le velocità istantanee delle due ruote posteriori (e si

ottiene un sistema odometrico equivalente al “differential drive” oppure leggendo velocità e posizione angolare della ruota anteriore.

1.2.4 TANK TREADS

Questa configurazione è una variante del sistema “differential drive”. I due cingoli sono attuati in maniera indipendente e la sterzata è ottenuta variando il rapporto delle velocità (figura 1.4).

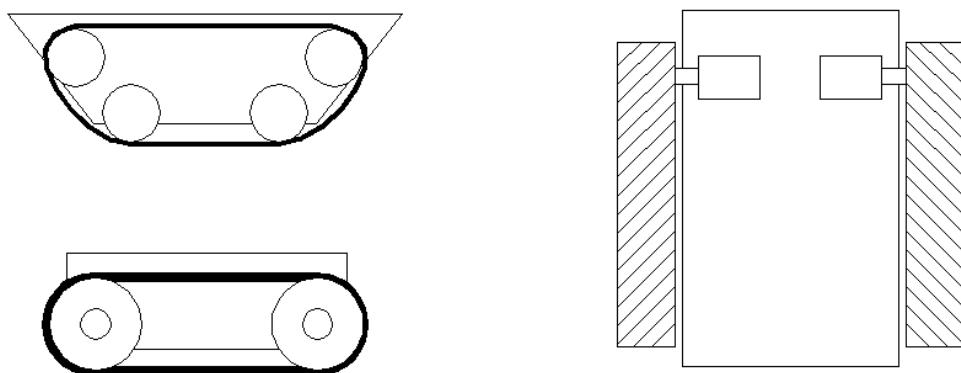


Figura 1.4 – Configurazioni “tank treads”.

La struttura è molto stabile anche in presenza di terreni piuttosto accidentati ed è alla base di vari veicoli militari. I molteplici punti di appoggio e il terreno non ben definito su cui si muove il sistema contribuiscono a rendere la sterzata assolutamente non accurata, con possibilità di sbandate laterali. Questo rende impossibile tenere traccia del movimento semplicemente considerando la velocità di rotazione dei cingoli.

1.2.5 SYNCHRO DRIVE

È un tipo particolare di base progettata per veicoli autonomi da interno. Questa configurazione è equipaggiata con tre o più ruote meccanicamente accoppiate,

in modo che tutte puntino nella stessa direzione e ruotino con la stessa velocità. Quando il veicolo deve cambiare direzione, le ruote sterzano tutte simultaneamente, grazie ad una sincronizzazione (meccanica od elettronica) che permette alle ruote di girare attorno al proprio asse verticale di uno stesso angolo. La sterzata si ottiene quindi senza ruotare la base. Questo riduce lo slittamento e determina una migliore accuratezza nell'odometria perché tutte le ruote generano sempre dei vettori di forza paralleli ed uguali. La sincronizzazione meccanica può essere ottenuta con cinghie, catene od ingranaggi (figura 1.5).

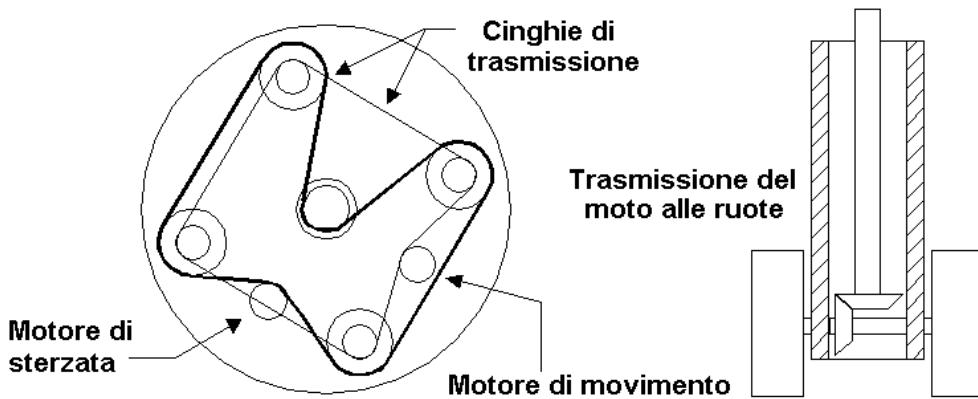


Figura 1.5 – Sistema “synchro drive” a quattro ruote.

A causa di giochi ed elasticità sempre presenti in un sistema meccanico, si ottiene un'accuratezza migliore con la sincronizzazione elettronica, la quale però richiede due motori per ogni ruota, uno per la rotazione della ruota ed uno per la sterzata. I veicoli dotati di questo sistema sono anche indicati con il nome di *veicoli omnidirezionali anolonomi*. Il termine *anolonomo* indica che il meccanismo non consente al veicolo di spostarsi in una qualsiasi direzione a partire da una configurazione iniziale arbitraria. Infatti nel sistema “synchro drive” anche se il veicolo è in grado di muoversi verso una qualsiasi direzione

senza ruotare la base è comunque necessaria una manovra preventiva di orientazione delle ruote.

1.2.6 OMNIDIRECTIONAL DRIVE

Questa configurazione permette un movimento laterale senza la necessità di orientare le ruote nella direzione di movimento. Essa si basa su tre o più ruote, ciascuna delle quali è in grado di scivolare in ogni direzione, grazie alla presenza di rulli sulla circonferenza esterna (figura 1.6a).

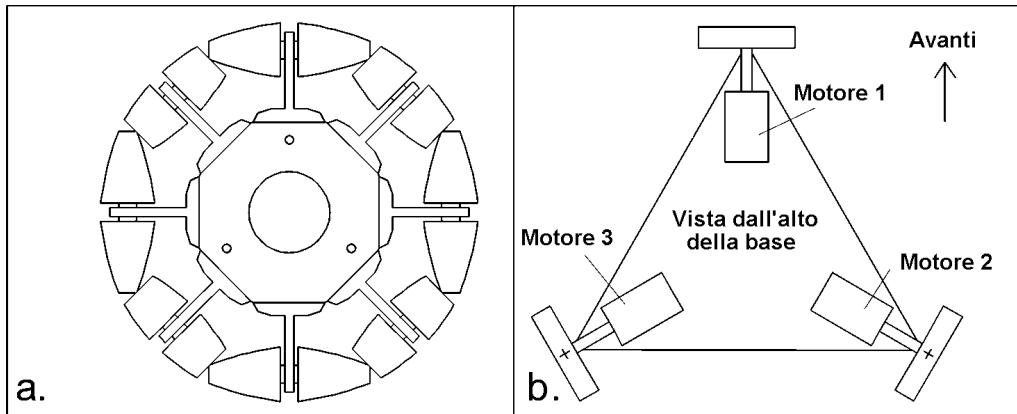


Figura 1.6 – a) Struttura meccanica delle ruote per il moto omnidirezionale - b) Esempio di configurazione “omnidirectional drive” a tre ruote.

Una coppia applicata all’asse di rotazione della ruota genera attrito con il terreno, mentre uno spostamento lungo l’asse genera attrito nullo, grazie alla presenza dei rulli. Componendo le forze generate sul terreno da tre o più ruote si può imprimere qualsiasi direzione di spostamento senza alcuna manovra preventiva di orientazione delle ruote (figura 1.6b). Comunque la delicata struttura delle ruote richiede un terreno estremamente piatto e non consente di caricare il robot con un peso eccessivo. Tali svantaggi rendono questo tipo di base poco utilizzata.

La seguente tabella riporta pro e contro delle varie tipologie di veicoli viste:

Struttura	Vantaggi	Svantaggi
Differential Drive	<ul style="list-style-type: none"> • Facile da programmare • Facile da costruire • Sterzata e movimento disaccoppiati 	<ul style="list-style-type: none"> • Non adatto a terreni sconnessi • Difficile da mantenere in linea retta • Può essere instabile
Ackerman Steering	<ul style="list-style-type: none"> • Può essere utilizzato anche per terreni un po' sconnessi • Miglior controllo direzionale 	<ul style="list-style-type: none"> • Sterzata accoppiata al movimento • Manovre più difficili • Programmazione più complicata
Tricycle Drive	<ul style="list-style-type: none"> • Compromesso tra stabilità e semplicità • Può sterzare con piccoli raggi 	<ul style="list-style-type: none"> • Minore stabilità che con 4 ruote • Sterzata accoppiata al movimento
Tank Treads	<ul style="list-style-type: none"> • Stabile su terreni molto sconnessi • Sterzata e movimento disaccoppiati 	<ul style="list-style-type: none"> • Possibilità di sbandamento • Elevato attrito e potenza • Sterzata non accurata
Synchro Drive	<ul style="list-style-type: none"> • Stabile • Facile da programmare • Sterzata e movimento disaccoppiati 	<ul style="list-style-type: none"> • Non adatto a terreni sconnessi • Meccanica complessa
Omni-directional Drive	<ul style="list-style-type: none"> • Stabile • Facile da programmare • Sterzata e movimento disaccoppiati 	<ul style="list-style-type: none"> • Non adatto a terreni sconnessi • Ruote complesse da realizzare • Non sopporta carichi pesanti

1.3 Attuatori

Gli *attuatori* di movimento possono classificarsi o in base all'azione o in base al tipo di conversione energetica effettuata. Così avremo attuatori lineari o angolari, per quanto riguarda l'azione e attuatori elettrici, idraulici o pneumatici per l'energia che li alimenta [Sciavicco e Siciliano, 1995]. Nei robot mobili, specialmente se di piccole dimensioni, risulta più pratico l'uso dell'energia elettrica, quindi considereremo solo attuatori di questo tipo, i *servomotori elettrici*.

Esistono svariati tipi di motori elettrici, che differiscono per principio di funzionamento e tecnologia: una prima modalità di classificazione fa riferimento al principio di funzionamento per quanto riguarda la generazione della coppia [De Carli, 1998/a]. Alcuni tipi di motore utilizzano attrazione e repulsione sviluppate da magneti permanenti e/o da elettromagneti, questi ultimi realizzati tramite bobine solitamente avvolte attorno a nuclei di materiale ferroso. In essi la coppia motrice prende origine dalla interazione del flusso al traferro con la corrente che fluisce negli avvolgimenti di statore o di rotore. In altri tipi di motore, la coppia motrice prende origine dalla variazione che presenta la riluttanza magnetica fra gli avvolgimenti. Seguendo tale criterio si perviene alla seguente suddivisione:

a) **Motori a flusso impresso:**

- *Motori a corrente continua* ad eccitazione indipendente;
- *Motori a corrente continua* a magneti permanenti;
- *Motori brushless* a campo trapezoidale;
- *Motori brushless* a campo sinusoidale.

b) Motori a flusso indotto:

- *Motori asincroni monofase;*
- *Motori asincroni trifase;*
- *Motori universali.*

c) Motori a riluttanza:

- *Motori passo-passo;*
- *Motori sincroni a riluttanza;*
- *Motori commutati a riluttanza;*
- *Motori ad isteresi.*

Un'altra suddivisione può essere effettuata in base all'andamento della tensione e della corrente di alimentazione necessaria per produrre il movimento. Seguendo questo criterio i motori elettrici si possono suddividere in tre grandi categorie:

- a) **Motori in corrente continua (CC);**
- b) **Motori in corrente alternata (CA), non necessariamente sinusoidale;**
- c) **Motori alimentati da convertitore di tipo statico.**

I motori in corrente alternata possono, a loro volta, essere ulteriormente suddivisi in motori *sincroni* e motori *asincroni* (detti anche *ad induzione*). I motori alimentati con forme d'onda di tipo dedicato vengono classificati in base al loro principio di funzionamento. I motori passo-passo si potrebbero inserire tra i motori CA sincroni, ma data la particolare forma della sollecitazione, necessariamente digitale, possono essere considerati una categoria a sé stante.

In relazione al comportamento dinamico desiderato nella generazione della coppia motrice e al relativo campo di escursione, è possibile effettuare l'alimentazione secondo modalità differenti. Al fine di rendere più semplice l'inserimento del motore controllato nel sistema di movimentazione, la modalità di alimentazione del motore viene collegata al valore di una variabile di comando, opportunamente definita. Quest'ultima è scelta in modo da stabilire in genere un legame lineare con la velocità di rotazione o con la coppia motrice. Solo in alcuni tipi di motore la posizione angolare dell'asse del motore è direttamente collegata alla variabile di comando.

I motori a corrente continua e quelli sincroni vengono ulteriormente suddivisi a seconda che il flusso sia ottenuto da un opportuno circuito di eccitazione oppure sia impresso per mezzo di magneti permanenti. Si hanno così motori in corrente continua a magneti permanenti e motori sincroni senza circuito di eccitazione e relativo collettore. Per tale motivo sono comunemente indicati come *motori brushless*. Nei motori a flusso impresso e a flusso indotto la rotazione è sempre di tipo continuo, mentre in alcuni motori a riluttanza il moto è di tipo incrementale.

Per ogni tipologia di motore esistono tecniche di pilotaggio e di controllo della velocità e/o della coppia: si va dal semplice controllo della tensione e della corrente nei motori CC, all'utilizzo di inverter per i motori CA, alla commutazione retroazionata delle diverse fasi nei motori "brushless", fino ai circuiti digitali per le complesse sequenze di pilotaggio dei motori passo-passo. A questo proposito va detto che alle tecniche analogiche classiche si sono affiancate metodologie di controllo digitale sempre più sofisticate che permettono di ottenere soluzioni efficienti e a basso costo anche per i motori

tradizionalmente "analogici", come i motori ad induzione e i motori commutati a riluttanza.

Vediamo ora in maggior dettaglio i principali motori utilizzati nel campo della robotica.

1.3.1 MOTORI IN CORRENTE CONTINUA

Nei motori in corrente continua un campo magnetico statico viene creato nello *stator*, cioè sulla parte fissa del motore, per mezzo di magneti permanenti, o tramite un avvolgimento che prende il nome di *circuito di eccitazione*. Un altro campo magnetico è generato dal *circuito di armatura*, un avvolgimento posto sul *rotore*, che è la parte mobile del motore. L'interazione dei due campi magnetici dà luogo ad una coppia che produce la rotazione del motore, poiché i poli del campo magnetico generato dal rotore cercheranno di allinearsi a quelli del campo magnetico fisso, generato dallo stator [Pisano, 2001].

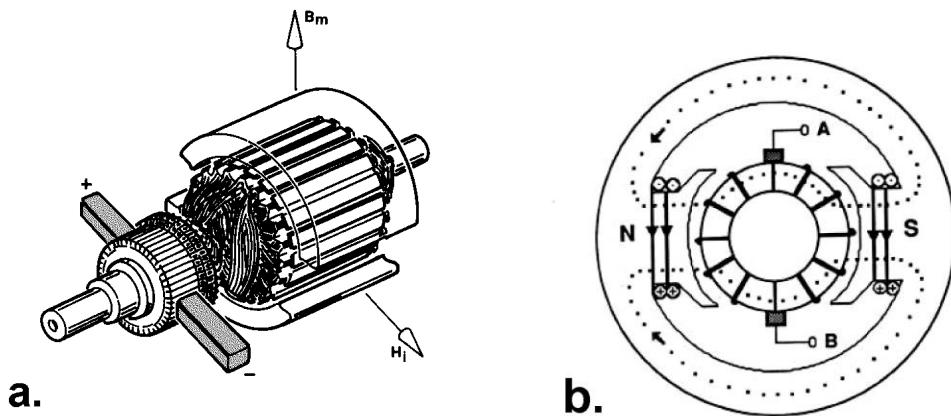


Figura 1.7 – a) Struttura di un motore in corrente continua; b) Sezione trasversale.

In figura 1.7b si può notare l'anello interno di rotore, su cui è avvolto il circuito di armatura e l'anello esterno dello stator sulle cui *espansioni polari* è avvolto

il circuito di eccitazione. Quest'ultimo viene attraversato dalla *corrente di eccitazione*, che produce un campo magnetico le cui linee di flusso tendono a concentrarsi sui percorsi in ferro, seguendo l'andamento qualitativo mostrato in figura. Il circuito di armatura, alimentato dalla *tensione di armatura*, è invece costituito da una bobina avvolta su un anello di materiale ferromagnetico, in modo da formare nel complesso un avvolgimento chiuso. Ogni spira del circuito di armatura presenta un tratto in cui il conduttore è scoperto, in modo che sia possibile stabilire un contatto tra ciascuna delle due *spazzole di grafite*, poste nei punti A e B (fissi nello spazio) e le spire che, per via della rotazione del rotore, transitano nella posizione corrispondente a questi due punti di contatto. Poiché l'alimentazione viene fornita attraverso le spazzole, il circuito di armatura viene ad essere rappresentato, in ogni istante, come il parallelo di due avvolgimenti (*vie interne*) che, seguendo due percorsi differenti, uniscono i punti A e B (figura 1.8).

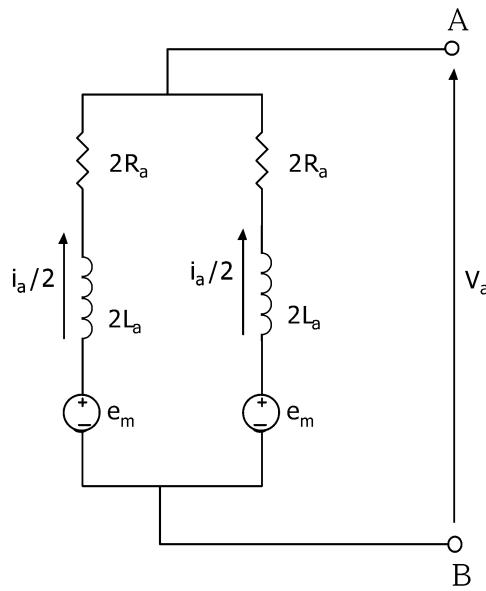


Figura 1.8 – Le “vie interne” del circuito di armatura.

Ovviamente, mentre il rotore ruota, tali punti di contatto del circuito di armatura con le spazzole si spostano lungo l'avvolgimento di armatura (che è chiuso su se stesso), per cui il circuito di armatura è sempre "diviso" in due vie interne simmetriche, che presenteranno pertanto i medesimi parametri elettrici e saranno, di conseguenza, attraversate dalla stessa corrente $i_a/2$.

Le matasse del circuito di armatura sono connesse tra loro in serie, ma hanno anche un punto di contatto con una *lamella* di rame. L'insieme di queste lamelle, adiacenti tra loro ma elettricamente isolate, forma il *collettore* su cui "strisciano" le spazzole durante il moto. Scopo del collettore è quello di far scorrere la corrente attraverso avvolgimenti successivi, in modo da mantenere il campo magnetico del rotore sempre perpendicolare a quello dello statore, indipendentemente dalla velocità. Infatti, in assenza di questo "commutatore meccanico" la rotazione tenderebbe a ridurre l'angolo esistente tra il campo magnetico dello statore e quello generato dal sistema di correnti rotoriche e la coppia si annullerebbe dopo aver compiuto un arco di 90° elettrici.

Andiamo ora ad individuare la serie dei fenomeni che determinano il funzionamento del motore. Siano R_e ed L_e rispettivamente la resistenza e l'induttanza del circuito di eccitazione, che soddisferà l'equazione di equilibrio elettrico:

$$v_e(t) = R_e i_e(t) + L_e \frac{di_e(t)}{dt}$$

dove $v_e(t)$ rappresenta la differenza di potenziale ai capi del circuito. Tale circuito sarà attraversato dalla *corrente di eccitazione* $i_e(t)$ che produce un campo magnetico le cui linee di flusso seguono il percorso evidenziato in figura 1.7b. Se N_e è il numero delle spire del circuito di eccitazione, ed \mathfrak{N} è la

riluttanza del circuito magnetico percorso dalle linee di flusso, in assenza di saturazione magnetica ($i_e(t) < I_{eMax}$) l'ampiezza del flusso generato sarà pari a:

$$\Phi(t) = \frac{N_e i_e(t)}{\mathfrak{R}} \quad (\text{Legge di Hopkinson})$$

Ricordiamo che, per effetto della tensione di armatura $v_a(t)$, il circuito di armatura è attraversato da una corrente $i_a(t)$. Poiché il flusso $\Phi(t)$ si concatena con le spire del circuito di armatura, queste subiscono una *coppia motrice* $T_m(t)$ di natura elettromagnetica, la cui risultante complessiva sull'intero avvolgimento è proporzionale al prodotto tra $\Phi(t)$ e $i_a(t)$, ed è espressa come segue:

$$T_m(t) = K^I \Phi(t) i_a(t) = K^{II} i_e(t) i_a(t)$$

Nel caso di motori a magneti permanenti, il circuito di eccitazione non esiste: il flusso Φ è generato dai magneti ed è costante. In tal caso la coppia è data da:

$$T_m(t) = K^{III} i_a(t)$$

K^I , K^{II} e K^{III} sono parametri costanti che dipendono dalle caratteristiche costruttive del motore. L'effetto della coppia motrice $T_m(t)$ è quello di porre in movimento il rotore, in quanto il circuito di eccitazione è rigidamente connesso allo statore, che ne impedisce il movimento. Con il rotore in movimento, il circuito di armatura si trova ad essere in moto in un campo magnetico e questo produce una *forza elettromotrice* che agisce sul circuito di armatura, data da:

$$e_m(t) = K^{IV} i_e(t) \omega(t)$$

oppure nel caso di motori a magneti permanenti:

$$e_m(t) = K^V \omega(t)$$

ove $\omega(t)$ è la velocità istantanea di rotazione del rotore e K^I , K^V sono opportune costanti. Si noti come la forza elettromotrice $e_m(t)$ agisca solo se il rotore è in movimento e sia nulla a rotore fermo ($\omega(t)=0$). Poiché il verso di $e_m(t)$ è tale da opporsi alla tensione di armatura $v_a(t)$, essa viene chiamata *forza contro-elettromotrice*.

In base a quanto detto, possiamo esprimere l'equazione di equilibrio elettrico del circuito di armatura nella forma seguente, essendo R_a ed L_a la resistenza e l'induttanza di tale circuito:

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_m(t)$$

Per la definizione del modello matematico complessivo, resta da considerare la condizione di equilibrio meccanico tra le coppie agenti sull'albero di rotore. Vedremo le equazioni che regolano tale equilibrio nel paragrafo 2.1.

La semplicità dei circuiti di pilotaggio, la possibilità di invertire la direzione di rotazione semplicemente invertendo la polarità e il basso costo hanno fatto del motore a collettore, alimentato in corrente continua, una delle categorie più diffuse nelle applicazioni. Esso è largamente utilizzato, specialmente per azionamenti che devono operare a velocità variabile, per la relativa semplicità di controllo. Purtroppo, un grosso problema è rappresentato dalle spazzole, per via dell'usura, della generazione di scintille e di disturbi elettromagnetici.

Inoltre, esso è più costoso rispetto ad altri tipi di servomotori elettrici (es. i motori asincroni), richiede una maggiore manutenzione ed è più ingombrante. Infine la struttura stessa del motore impone dei limiti alla velocità di rotazione e alla massima temperatura operativa. I motori CC sono comunque utilizzati in svariati settori e un nuovo impulso alla loro diffusione deriva dai recenti progressi nella realizzazione dei materiali magnetici, che rende disponibili a costo contenuto motori dalle elevate prestazioni dinamiche. Inoltre, da alcuni anni, hanno raggiunto una discreta diffusione sul mercato i motori a commutazione elettronica (motori “brushless”), che non presentano contatti strisciati (spazzole) e sono più robusti rispetto ai motori a collettore.

1.3.3 MOTORI SENZA SPAZZOLE (BRUSHLESS)

Da un certo punto di vista, se si assume come sistema di riferimento fisso il rotore, si può osservare come un motore CC non sia che un motore sincrono a magneti permanenti, azionato a corrente alternata a frequenza variabile, in modo tale che l'angolo di fase tra il campo rotorico e quello statorico sia costante. In altri termini, il collettore ha funzione di invertitore meccanico, con frequenza e fase vincolate alla rotazione del motore. La realizzazione fisica di questa funzione costituisce la principale limitazione del motore CC: infatti il collettore deve essere posto in rotazione, rendendo così necessario far ruotare l'intera parte avvolta del motore. Questi inconvenienti possono essere superati sostituendo il commutatore meccanico con un *invertitore elettronico*.

Nella sua forma più semplice, il motore CC a commutazione elettronica (figura 1.9) è costituito da un induttore a magneti permanenti (1), un indotto recante un avvolgimento polifase (solitamente trifase (2)) ed un commutatore elettronico [Venturini, 1985]. A sua volta questo è formato da un sensore di

posizione, solidale con l'asse del motore (3), e da una matrice di interruttori elettronici, in grado di alimentare ogni avvolgimento di statore (4).

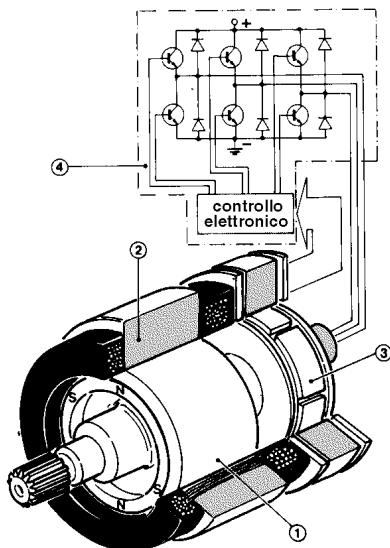


Figura 1.9 – Struttura di un motore “brushless”.

Il funzionamento è concettualmente assai simile a quello di un motore CC: il controllo elettronico, tramite il *sensore di posizione*, identifica il sistema di avvolgimenti perpendicolare al campo magnetico del rotore e lo alimenta. A seguito della rotazione, il modulo elettronico commuta la corrente nelle varie fasi, in modo tale che il campo risultante delle correnti di indotto sia mantenuto sempre perpendicolare a quello dei magneti.

Poiché l'inverter elettronico, al contrario del suo analogo meccanico, non è fisicamente solidale con la parte rotante, non è più necessario avvolgere il rotore; questo incorpora i magneti mentre gli *avvolgimenti di armatura* vengono trasferiti sullo statore.

Le caratteristiche esterne del sistema motore + elettronica possono essere ricavate dallo schema di principio riportato nella figura 1.10a, rappresentativo del tipo di motore più comune, con avvolgimento trifase collegato a stella e forza elettromotrice (F.E.M.) ad “onda trapezoidale”. La forma d’onda delle tre fasi del motore in rotazione è rappresentata nella figura 1.10b, mentre la commutazione delle fasi è riportata nella figura 1.10c. Come risulta dalla figura 1.10d, se si trascurano l’induttanza degli avvolgimenti e le cadute di tensione nell’elettronica, la F.E.M. risultante del sistema motore + elettronica è una tensione continua, composta dai contributi sincroni delle F.E.M. delle singole fasi e quindi proporzionale alla velocità.

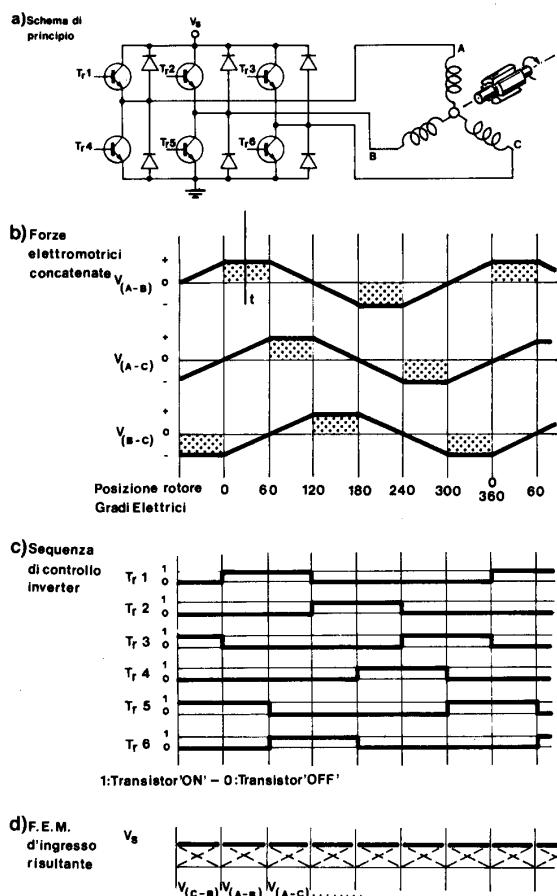


Figura 1.10 – Forme d’onda tipiche di un sistema “brushless”.

Si osservi che prestazioni simili a quelle di un servomotore CC possono essere ottenute anche con un sistema motore asincrono + inverter, ove l'inverter venga opportunamente controllato (*controllo vettoriale*). Gli azionamenti di questo tipo sono talvolta denominati “AC brushless”. Per evitare ogni ambiguità, in questo contesto definiamo azionamenti “brushless” esclusivamente gli azionamenti costituiti da un motore sincrono ed un inverter elettronico, la cui fase sia rigidamente vincolata alla posizione del rotore.

Da un punto di vista applicativo entrambi i sistemi superano il limite di commutazione del motore CC; il motore a induzione (asincrono), tuttavia, è caratterizzato dalla generazione di calore nel rotore, ed è quindi capace di coppie continuative, accelerazioni e potenze rese per unità di volume inferiori a quelle dei motori “brushless”. L'azionamento elettronico, inoltre, deve fornire energia di magnetizzazione del motore, oltre a quella attiva, e va pertanto dimensionato per potenze maggiori. Il controllo vettoriale, infine, impone l'uso di un inverter sinusoidale, di maggiore complessità.

L'assenza di spazzole nei motori “brushless” comporta una serie di vantaggi: innanzitutto si aumenta l'efficienza eliminando una fonte di attrito; in secondo luogo viene rimossa una sorgente di interferenza elettromagnetica e di possibili guasti, per usura dei carboncini di grafite. L'assenza di componenti meccanici nella commutazione permette poi di raggiungere velocità di rotazione più elevate, limitate essenzialmente dalla qualità dei cuscinetti. Inoltre il fatto di avere gli avvolgimenti solo sullo statore permette una migliore dissipazione del calore che, a parità di temperatura operativa, si traduce in dimensioni più compatte per i motori. Infine grazie al ridotto momento di inerzia e all'alta coppia di spunto, si possono ottenere bande passanti considerevolmente superiori a quelle ottenibili con i motori CC tradizionali.

Dal punto di vista dell'uniformità della rotazione e della coppia resa, esiste tuttavia un'importante differenza pratica: un tipico servomotore CC è realizzato con un indotto avente moltissime fasi (anche fino a 100 ed oltre) collegate tra loro ad anello; tale molteplicità di fasi garantisce una bassa ondulazione della forza elettromotrice risultante e quindi della coppia motrice. Teoricamente, analoga tecnica potrebbe essere impiegata con la commutazione elettronica; in pratica, per contenere le dimensioni ed il costo dell'inverter, i motori "brushless" vengono solitamente realizzati con avvolgimento trifase. Questo dà luogo a una ondulazione intrinseca della coppia. Ad alta velocità, a questo fenomeno si aggiunge la discontinuità nella coppia generata, imputabile al tempo finito necessario alla commutazione della corrente da una fase del motore alla successiva: a causa dell'induttanza propria del motore, man mano che questo accelera, la forma d'onda di corrente si trasforma progressivamente da una forma d'onda rettangolare ad una forma d'onda trapezoidale, con conseguente perdita di coppia durante i transitori di commutazione. Tutti gli azionamenti "brushless" dovranno pertanto utilizzare qualche metodo atto a stabilizzare la coppia e a garantire l'uniforme rotazione del motore.

1.3.5 MOTORI PASSO-PASSO (STEPPER)

I motori passo-passo sono motori che, a differenza di tutti gli altri, hanno come scopo quello di mantenere fermo l'albero in una posizione di equilibrio e solo indirettamente permettono la rotazione: se alimentati si limitano a bloccare l'albero in una ben precisa posizione angolare. La rotazione dell'albero è ottenuta inviando una serie di impulsi di corrente, secondo un opportuno ordine: in questo modo la posizione di equilibrio dell'albero cambia per scatti successivi. È possibile far ruotare l'albero nella posizione e alla velocità voluta semplicemente contando gli impulsi ed impostando la frequenza di invio, dato che le posizioni di equilibrio dell'albero sono note con estrema precisione.

Generalmente, tutti gli avvolgimenti del motore sono parte dello statore e il rotore è un magnete permanente oppure, nel caso dei motori a riluttanza variabile, un cilindro in ferro dolce dentato. L'intera commutazione deve essere gestita esternamente dal *controller* del motore; i controller sono progettati in modo tale che il motore possa essere mantenuto in una qualsiasi posizione fissa, ma anche ruotato nell'una o nell'altra direzione [Jones, 1995].

Per alcune applicazioni, è possibile scegliere se utilizzare servomotori o motori passo-passo. Entrambi i tipi di motore offrono le stesse possibilità di effettuare posizionamenti precisi, ma sono caratterizzati da numerose differenze. I servomotori richiedono sistemi di controllo retroazionati. In genere ciò comporta la presenza di un sensore per fornire il feedback sulla posizione del rotore e di circuiti per indurre attraverso il motore una corrente che è funzione della differenza tra la posizione desiderata e la posizione effettiva.

La scelta tra motori passo-passo e servomotori deve partire da diverse considerazioni che dipendono dal tipo di applicazione. Per esempio, la ripetibilità del posizionamento possibile con un motore passo-passo dipende dalla geometria del rotore del motore, mentre nel caso del servomotore la ripetibilità dipende generalmente dalla stabilità del sensore di posizione e da altri componenti analogici o digitali nel circuito di retroazione.

I motori passo-passo possono essere utilizzati nei sistemi di controllo semplici ad anello aperto; sono infatti generalmente adeguati per sistemi che operano a basse accelerazioni con carichi statici. In alcuni casi, però, può essere essenziale il controllo ad anello chiuso per le alte accelerazioni, in particolare in presenza di carichi variabili. Se un motore passo-passo in un sistema di controllo ad anello aperto viene sottoposto a una coppia troppo elevata, la posizione del

rotore va persa ed è necessario re-inizializzare il sistema. I servomotori, invece, non sono soggetti a questo problema.

I motori passo-passo hanno diverse risoluzioni angolari, che variano da 90 fino a 0,72 gradi per passo. Con un controller appropriato, la maggior parte dei motori possono funzionare a mezzi passi; alcuni controller sono in grado di gestire anche frazioni di passo inferiori (*micropassi*).

Dal punto di vista applicativo i motori passo-passo si possono suddividere in tre gruppi: *motori a riluttanza variabile*, *motori unipolari* e *motori bipolari*.

MOTORI A RILUTTANZA VARIABILE

Un motore passo-passo a riluttanza variabile è dotato di tre avvolgimenti, solitamente collegati come mostrato nella figura 1.11, vale a dire con un terminale comune a tutti gli avvolgimenti.

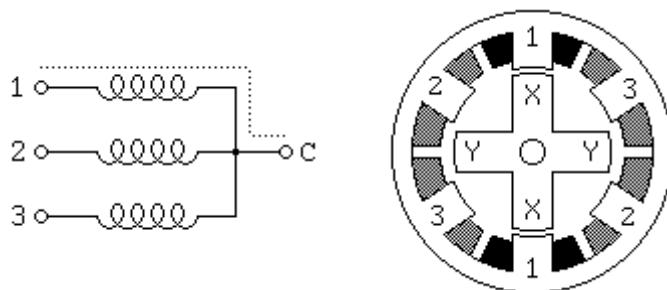


Figura 1.11 – Struttura di un motore passo-passo a riluttanza variabile.

Il terminale comune in genere viene collegato al positivo dell'alimentazione e gli avvolgimenti vengono alimentati in sequenza. La sezione trasversale riportata si riferisce a un motore a riluttanza variabile con 30 gradi per passo.

Il rotore di questo motore è dotato di 4 denti mentre lo statore ha 6 poli, e ciascun avvolgimento è disposto su poli opposti. Se viene alimentato l'avvolgimento numero 1, i denti del rotore contrassegnati da X vengono attratti verso i poli di questo avvolgimento. Se la corrente che attraversa l'avvolgimento 1 viene interrotta e viene attivato l'avvolgimento 2, il rotore ruota di 30 gradi in senso orario, in modo che i poli contrassegnati da Y si allineano con i poli contrassegnati da 2 e così via. Per ruotare in modo continuo il motore, occorre alimentare i 3 avvolgimenti nella sequenza opportuna. Assumendo una logica positiva, dove 1 rappresenta l'attivazione della corrente attraverso un avvolgimento del motore, la sequenza di controllo sotto riportata farà ruotare il motore in senso orario di 24 passi (4 rivoluzioni):

Avvolgimento 1	1001001001001001001001001
Avvolgimento 2	0100100100100100100100100
Avvolgimento 3	0010010010010010010010010

tempo -->

La geometria del motore illustrata nella figura 1.11, caratterizzata da 30 gradi per passo, utilizza il minor numero possibile di denti del rotore e di poli dello statore per garantire prestazioni soddisfacenti. Utilizzando un numero superiore di poli del motore e denti del rotore è possibile costruire motori con angolo di passo inferiore.

MOTORI UNIPOLARI

I motori passo-passo unipolari possono essere motori a magnete permanente o motori ibridi ed hanno una configurazione analoga a quella riportata nella figura 1.12, con un collegamento di centro su ciascuno dei due avvolgimenti. Dal motore escono pertanto 5 o 6 fili. I collegamenti di centro sono generalmente connessi al polo positivo dell'alimentazione mentre le due

estremità di ciascun avvolgimento sono alternativamente collegate a terra per invertire la direzione del campo generato dall'avvolgimento.

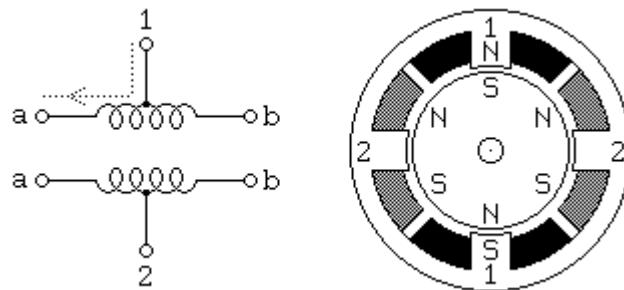


Figura 1.12 – Struttura di un motore passo-passo unipolare.

La sezione trasversale del motore riportata in figura si riferisce a un motore con 30 gradi per passo. L'avvolgimento numero 1 del motore è disposto tra la parte superiore e la parte inferiore del polo dello statore, mentre l'avvolgimento numero 2 è disposto tra i poli di sinistra e di destra del motore. Il rotore è un magnete permanente con 6 poli, 3 sud e 3 nord, collocati intorno alla sua circonferenza. Per risoluzioni angolari superiori, il rotore deve avere proporzionalmente più poli. Il motore con 30 gradi per passo illustrato nella figura è una delle configurazioni di motore a magnete permanente più comuni, anche se è facile trovare motori con 15 e 7,5 gradi per passo. Vengono prodotti anche motori a magnete permanente con risoluzioni dell'ordine di 1,8 gradi per passo. I motori ibridi, invece, hanno generalmente una risoluzione di 3,6 e 1,8 gradi per passo, ma possono raggiungere anche 0,72 gradi per passo.

Come illustrato nella figura 1.12, la corrente che fluisce tra il collegamento di centro dell'avvolgimento 1 e il terminale fa sì che il polo superiore dello statore sia polo nord e quello inferiore polo sud. Questo attrae il rotore nella posizione

mostrata. Se l'alimentazione viene rimossa all'avvolgimento 1 e attivata all'avvolgimento 2, il rotore girerà di 30 gradi, cioè di un passo.

Per ruotare in modo continuo questo motore, occorre alimentare i due avvolgimenti in sequenza. Assumendo al solito una logica positiva, le due sequenze di controllo sotto riportate faranno ruotare entrambi il motore in senso orario di 24 passi (4 rivoluzioni):

Sequenza A (funzionamento a passo intero):

Avvolgimento 1a	1000100010001000100010001
Avvolgimento 1b	0010001000100010001000100
Avvolgimento 2a	0100010001000100010001000
Avvolgimento 2b	0001000100010001000100010
tempo -->	

Sequenza B (funzionamento a passo intero):

Avvolgimento 1a	1100110011001100110011001
Avvolgimento 1b	0011001100110011001100110
Avvolgimento 2a	0110011001100110011001100
Avvolgimento 2b	1001100110011001100110011
tempo -->	

Si osservi che le due metà di ciascun avvolgimento non vengono mai alimentate simultaneamente. Entrambe le sequenze precedentemente riportate faranno ruotare il motore di un passo alla volta. La sequenza A alimenta un solo avvolgimento per volta, quindi richiede meno potenza; la sequenza B, invece, fornisce potenza a due avvolgimenti per volta ma produce una coppia superiore di circa 1,4 volte rispetto a quella generata dalla sequenza A, mentre utilizza il doppio della potenza.

Combinando le due sequenze precedenti è possibile ottenere un funzionamento a mezzi passi:

Sequenza C (funzionamento a mezzo passo):

Avvolgimento 1a	11000001110000011100000111
Avvolgimento 1b	00011100000111000001110000
Avvolgimento 2a	01110000011100000111000001
Avvolgimento 2b	00000111000001110000011100

tempo -->

I motori unipolari sono piuttosto facili da pilotare, almeno nei casi in cui vengano richieste basse prestazioni, in quanto è sufficiente usare quattro interruttori elettronici, per alimentare le quattro fasi secondo la sequenza scelta.

MOTORI BIPOLARI

I motori bipolari sono costruiti con lo stesso criterio dei motori unipolari, ma i due avvolgimenti non hanno un collegamento centrale (figura 1.13). Quindi il motore in se stesso è più semplice, mentre il circuito di pilotaggio risulta più complesso poiché deve poter invertire la polarità di ciascuna coppia di poli. Un circuito di azionamento di questo tipo (*H-bridge*) verrà utilizzato anche per il controllo dei motori in corrente continua e sarà descritto nel paragrafo 3.2.

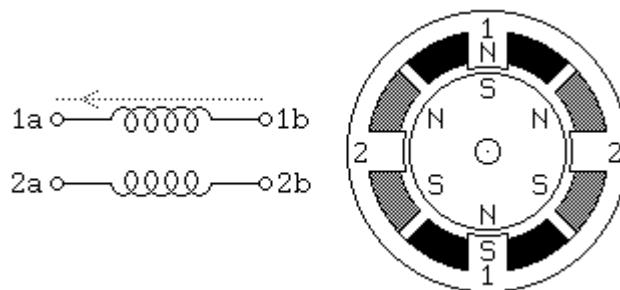


Figura 1.13 – Struttura di un motore passo-passo bipolare.

Il vantaggio deriva dal fatto che, essendo le fasi due anziché quattro, a parità di potenza del motore, il peso e la dimensione sono minori in quanto è necessario usare una minor quantità di rame. Inoltre, usando appositi schemi, è possibile ottenere circuiti di pilotaggio più efficienti in termini di consumo e velocità. Anche per i motori bipolari è possibile il funzionamento a passi interi, mezzi passi e micropassi.

Per riepilogare, i motori passo-passo offrono i seguenti vantaggi:

- Permettono di realizzare azionamenti di precisione in catena aperta;
- Offrono elevata robustezza meccanica ed elettrica;
- Consentono di realizzare piccole rotazioni dell'albero senza riduttori meccanici.

Naturalmente presentano anche degli svantaggi:

- Richiedono sempre circuiti elettronici per il pilotaggio;
- Hanno un funzionamento a scatti, soprattutto ai bassi regimi;
- Mostrano un rendimento energetico basso e, in genere, piccola potenza;
- Possiedono costo elevato, relativamente ad altri tipi di motore.

1.4 Sensori

Il più importante problema che un robot mobile deve affrontare è la determinazione della sua posizione nell'ambiente circostante, in altre parole poter dare una risposta alla domanda “dove sono ?”.

Il risultato che emerge dalla varia letteratura esistente sul problema della determinazione della posizione dei robot mobili, è che ad oggi non esiste una soluzione ottimale al problema. Le numerose soluzioni parziali possono essere suddivise in due categorie generali: stima della posizione *relativa* e stima della posizione *assoluta* [Borenstein *et al.*, 1996].

In assenza di un metodo unico e generalmente valido, gli sviluppatori di robot mobili solitamente combinano due metodi, uno per ciascuna categoria. Le due categorie a loro volta possono essere ulteriormente suddivise in sottogruppi:

a) **Stima della posizione relativa:** si appoggia al moto o a forze indotte dallo stesso robot, senza riferimenti all’ambiente esterno. I sensori che danno una stima relativa della posizione sono chiamati sensori *propriocettivi*, perché forniscono al sistema di controllo una percezione sullo stato interno del robot [Sciavicco e Siciliano, 1995]. Attraverso semplici calcoli matematici si ricava la posizione corrente a partire dalla conoscenza della precedente posizione e dalle informazioni sulla traiettoria e la velocità acquisite dai sensori. Il calcolo della posizione relativa può basarsi su due metodi:

- **Odometria:** questo metodo si serve di encoder (dischi ottici calettati sugli assi dei motori o sullo sterzo) per la misurazione della rotazione delle ruote e/o dell’orientazione della sterzata. Il vantaggio dell’odometria è che si tratta di un metodo completamente autonomo ed è sempre in grado di fornire una stima della posizione del veicolo.

Lo svantaggio del metodo odometrico è che l'errore di posizione, (sempre presente a causa di schiacciamenti e slittamenti delle ruote) aumenta senza limite se non si ricorre periodicamente all'uso di un riferimento indipendente per determinare la reale posizione del robot [Cox, 1991]. La maggior parte dei robot mobili basano su encoder la loro strategia di navigazione ed annullano periodicamente gli errori accumulati sfruttando metodi di posizionamento assoluto.

- **Navigazione inerziale:** il principio base della navigazione inerziale consiste nel calcolare, mediante accelerometri, l'accelerazione del robot lungo ognuno dei tre assi direzionali e nella conseguente determinazione della velocità e della posizione mediante l'integrazione dei dati acquisiti. Una piattaforma di sensori, stabilizzata con giroscopi ottici o meccanici, viene utilizzata per mantenere fisse le orientazioni dei tre assi [Dunlap e Shufeldt, 1972]. Anche i sistemi di navigazione inerziale sono completamente autonomi, ma l'implementazione di tale metodo non è immediata a causa della necessità di utilizzare dispositivi di alta qualità: poiché la determinazione della velocità e della posizione richiede calcoli di integrazione di primo e di secondo grado, ogni piccola deriva introdotta da accelerometri e giroscopi comporta la crescita, senza limiti, degli errori nelle misure integrate. Quindi anche la navigazione inerziale non è consigliabile per periodi di tempo lunghi (a meno di utilizzare apparati ad elevata precisione come quelli presenti sugli aerei, che hanno però costi proibitivi). Recentemente i prezzi dei giroscopi a fibre ottiche (noti anche con il nome inglese di *laser gyros*), molto accurati, hanno subito un calo notevole, ed oggi rappresentano una soluzione interessante per la navigazione dei robot mobili.

b) **Stima della posizione assoluta:** si appoggia a riferimenti presenti nell'ambiente esterno. I sensori che danno una stima assoluta della posizione, sono chiamati *sensori eterocettivi*, perché forniscono al sistema di controllo una percezione sullo stato dell'ambiente in cui il robot si muove. Ci sono diversi metodi per calcolare la posizione in modo assoluto:

- **Guide attive:** questo metodo calcola la posizione assoluta del robot a partire dalla misurazione dell'incidenza di tre o più *beacon* (fasci luminosi o microonde) attivamente trasmessi. I trasmettitori devono essere collocati nell'ambiente, in luoghi noti al robot. Le guide attive sono state utilizzate per molti secoli come efficaci ed accurati strumenti per la navigazione: sono guide attive sia le stelle che i fari. La moderna tecnologia ha aumentato le capacità di questi sistemi grazie all'utilizzo di trasmettitori laser, ad ultrasuoni ed a radiofrequenze. Si è giunti così ai radiofari e ultimamente al sistema GPS (Global Positioning System).
- **Riconoscimento di landmark:** la parola “landmark” indica quegli elementi distintivi dell'ambiente, che un robot può riconoscere grazie ai suoi sensori. I landmark devono avere posizioni fisse e ben note ed essere facilmente individuabili; una volta riconosciuti sono usati per la navigazione. Per semplificare il problema della ricerca dei landmark spesso si assume che la posizione e l'orientazione del robot siano noti con una certa approssimazione, così che i landmark debbano essere cercati in un'area limitata. Per tale motivo una buona precisione del sistema odometrico è un requisito fondamentale per l'individuazione dei landmark. I landmark possono essere naturali o artificiali: il vantaggio di utilizzare landmark artificiali è che questi vengono

progettati specificatamente per la navigazione, in modo tale da garantire la rilevabilità ottimale anche in condizioni ambientali avverse. Viceversa i landmark naturali sono quegli elementi o caratteristiche che già si trovano nell'ambiente e che hanno una loro funzione, oltre quella di consentire la navigazione del robot. Il metodo di rilevazione tramite landmark ha come vantaggio il fatto che gli errori di determinazione della posizione sono limitati; inoltre, alcune informazioni aggiuntive (per esempio la distanza) possono essere ottenute misurando le proprietà geometriche dei landmark, ma questo approccio richiede una notevole potenza di calcolo e non è molto preciso. Il problema principale nella navigazione tramite landmark è nel rilevamento dei landmark stessi. In generale la scelta del sensore ricade sui sensori di visione: le telecamere. La scelta delle caratteristiche da riconoscere è importante perché determina la complessità delle attività di descrizione, di individuazione e di “matching”. Un tipo di navigazione con landmark largamente utilizzato nelle industrie è la navigazione lungo linee, che sfrutta un landmark continuo che deve essere seguito dal robot.

- **Corrispondenza di modelli:** questo metodo prevede che le informazioni acquisite dai sensori, incorporati nel robot, vengano utilizzate per costruire una mappa dell'ambiente; questa mappa viene poi confrontata con un modello in memoria e se viene trovata corrispondenza tra le due, il robot è in grado di calcolare la propria posizione ed orientazione. La mappa di riferimento memorizzata, che si sfrutta per il confronto, potrebbe essere stata costruita sia da un programmatore, sia attraverso un'attività di esplorazione preventiva dell'ambiente: in quest'ultimo caso, la determinazione della posizione

spesso comporta il miglioramento della mappa globale sulla base delle nuove osservazioni sensoriali e l'integrazione di mappe locali nella mappa globale, fino a coprire aree prima inesplorate. Le mappe utilizzate nella navigazione possono essere di due tipi: *geometriche* e *topologiche*. Le mappe geometriche rappresentano il mondo in un sistema di coordinate globali, mentre le mappe topologiche rappresentano il mondo con nodi e archi.

Vediamo ora i sensori più comunemente utilizzati nella robotica mobile per la determinazione della posizione. Tratteremo in maggior dettaglio gli encoder, utilizzati, tra l'altro, per il controllo di velocità dei motori, e passeremo in rapida rassegna gli altri tipi di sensori.

1.4.1 ENCODER OTTICI

L'encoder è un dispositivo elettromeccanico che fornisce una informazione digitale relativa alla rotazione dell'asse su cui è montato [ELCIS, 1991]. Un raggio di luce focalizzato, puntato su un fotorilevatore, viene periodicamente interrotto da un pattern codificato opaco/trasparente disposto su un disco intermedio, che ruota applicato all'asse. Il disco ruotante può essere realizzato in vetro, metallo o plastica. Grazie alla tecnologia costruttiva e all'uscita digitale l'encoder offre una buona immunità al rumore.

Esistono essenzialmente due tipi di encoder ottici: quelli *incrementali* e quelli *assoluti*. Gli encoder incrementali misurano, in pratica, la velocità di rotazione e possono inferire la posizione relativa, mentre i modelli assoluti misurano direttamente la posizione angolare e possono inferire la velocità. Se non sono richieste informazioni non volatili sulla posizione, gli encoder incrementali

forniscono una risoluzione equivalente a costi notevolmente inferiori rispetto agli encoder ottici assoluti.

ENCODER OTTICI INCREMENTALI

Il tipo più semplice di encoder incrementale è l'*encoder tachimetrico* a canale singolo, un dispositivo che produce un determinato numero di impulsi per ciascuna rivoluzione dell'albero. Aumentando il numero di impulsi aumenta la risoluzione, ma anche il costo del dispositivo (che è comunque basso, se paragonato ad altri sistemi di misura). L'encoder è anche adatto come sensore di feedback di velocità nei sistemi di controllo a bassa e media velocità, ma ha problemi di rumore e stabilità a velocità estremamente basse, a causa degli errori di quantizzazione [Nickson, 1985]. Oltre ai problemi di instabilità alle basse velocità, l'encoder tachimetrico a canale singolo non è in grado neanche di rilevare la direzione di rotazione e quindi non può essere utilizzato come sensore di posizione.

Questi problemi vengono superati negli *encoder incrementali a quadratura di fase*, ove con l'aggiunta di un secondo fotorivelatore, leggermente spostato rispetto al primo, si ottengono due treni di impulsi sfasati di 90° (figura 1.14).

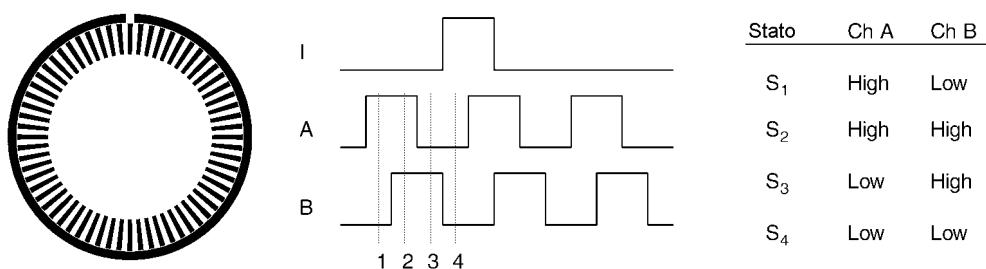


Figura 1.14 – La relazione di fase tra i canali A e B può essere usata per determinare la direzione di rotazione e per aumentare la risoluzione dell'encoder.

Questa tecnica consente ai componenti elettronici di decodifica di determinare quale canale stia anticipando l'altro e quindi permette di accettare la direzione di rotazione, con il beneficio aggiunto di una risoluzione migliore.

La natura incrementale dei segnali di uscita implica che qualsiasi determinazione della posizione angolare possa solo essere relativa rispetto a qualche riferimento specifico, e non assoluta. Esistono diversi modi per definire tale riferimento e la maggior parte degli encoder incorporano come terzo canale una speciale *uscita indice* che segnala una posizione di zero. Le posizioni intermedie dell'albero vengono quindi determinate conteggiando il numero di impulsi e tenendo conto del verso di rotazione. Uno svantaggio di questo approccio è che tutte le informazioni sulla posizione relativa si perdono in caso di interruzione dell'alimentazione.

Un crescente numero di componenti elettronici a basso costo ha contribuito a rendere l'encoder incrementale a quadratura di fase il sensore di punta nella realizzazione di robot e di altri azionamenti elettrici. Svariati produttori offrono motoriduttori con l'encoder incrementale già montato all'interno, con risoluzioni comprese tra 96 e 4096 impulsi ed uscita TTL.

ENCODER OTTICI ASSOLUTI

Gli encoder assoluti sono utilizzati per applicazioni in cui non può essere assolutamente tollerata la mancanza di informazione sulla posizione, in seguito ad una interruzione dell'alimentazione. Invece di due treni di impulsi, gli encoder assoluti forniscono un valore binario che identifica univocamente la posizione (quantizzata) dell'asse. Questo viene ottenuto utilizzando un array di fotosensori allineati su più tracce (figura 1.15). Una regola generale è che ogni traccia aggiuntiva raddoppia la risoluzione ma quadruplica i costi

[Agent, 1991]. Inoltre, il fatto di utilizzare una traccia dedicata per ogni bit di risoluzione comporta una dimensione maggiore per il disco e un corrispondente diminuzione di tolleranza agli urti e alle vibrazioni.

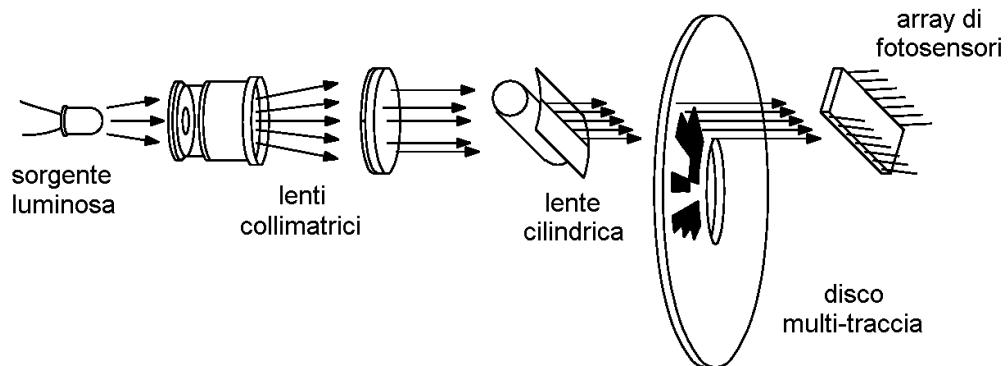


Figura 1.15 – Schema di funzionamento di un encoder assoluto.

Il sistema di codifica dei bit di uscita è solitamente il *codice GRAY*, che è caratterizzato dal fatto che solo un bit alla volta cambia, un decisivo vantaggio nell'eliminazione di ambiguità causate da tolleranze nei componenti meccanici ed elettronici (figura 1.16a).

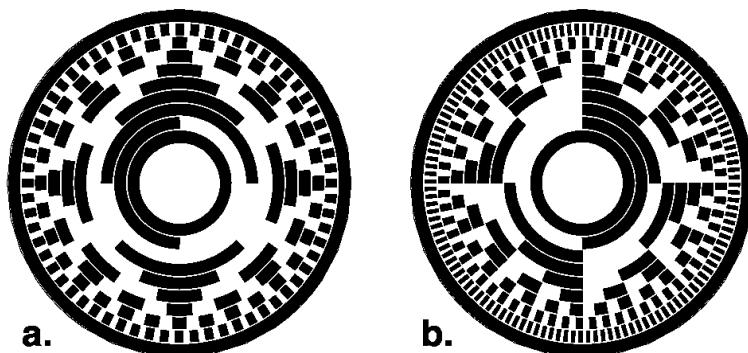


Figura 1.16 – a) Codifica GRAY; b) Codifica binaria.

Infatti il codice binario può comportare cambiamenti su più di un bit: ad esempio passando dalla posizione 0 alla posizione 255 otto bit subiscono una

transizione di stato (figura 1.16b). Poiché non c'è alcuna garanzia che la transizione degli otto bit avvenga esattamente nello stesso istante, una considerevole ambiguità può esistere durante le transizioni di stato con uno schema di codifica di questo tipo [Avolio, 1993].

Gli encoder assoluti vengono utilizzati nella robotica mobile per la misura della posizione dello sterzo, in quanto non richiedono manovre per conoscere la posizione iniziale. Viceversa gli encoder incrementali sono più utilizzati per misurare la velocità delle ruote e determinare lo spazio percorso durante il cammino in brevi intervalli di tempo.

1.4.2 ACCELEROMETRI E GIROSCOPI

Elemento essenziale per la navigazione inerziale, l'accelerometro è essenzialmente un trasduttore di accelerazioni, fornisce cioè in uscita un segnale elettrico proporzionale alle accelerazioni che la struttura subisce secondo un ben determinato asse. Il principio di funzionamento dell'accelerometro è molto semplice: una massa, vincolata elasticamente tramite una molla al contenitore dell'accelerometro, è sottoposta all'azione dell'accelerazione lineare che si desidera misurare: per la terza legge della dinamica ($F = ma$) l'accelerazione produce una forza, la quale agendo sulla massa la sposta fino a determinare una nuova posizione di equilibrio (infatti sulla massa agisce anche la forza elastica della molla). Un trasduttore di posizione fornisce un segnale proporzionale allo spostamento della massa e quindi all'accelerazione.

L'*accelerometro meccanico* (quello descritto) non è in grado di fornire una banda passante estesa; per ottenere caratteristiche dinamiche migliori, alcuni costruttori realizzano *accelerometri ad equilibrio di forza*, ove al posto della

molla, vi è un elettromagnete controllato da un circuito integrato inserito nel contenitore dell'accelerometro.

I giroscopi, anch'essi elementi essenziali per la navigazione inerziale, possono essere classificati in due categorie: *giroscopi meccanici* e *giroscopi ottici*. Il giroscopio meccanico, sensore di rotazione ben noto ed affidabile, si basa sulle proprietà inerziali di un rotore che gira rapidamente.

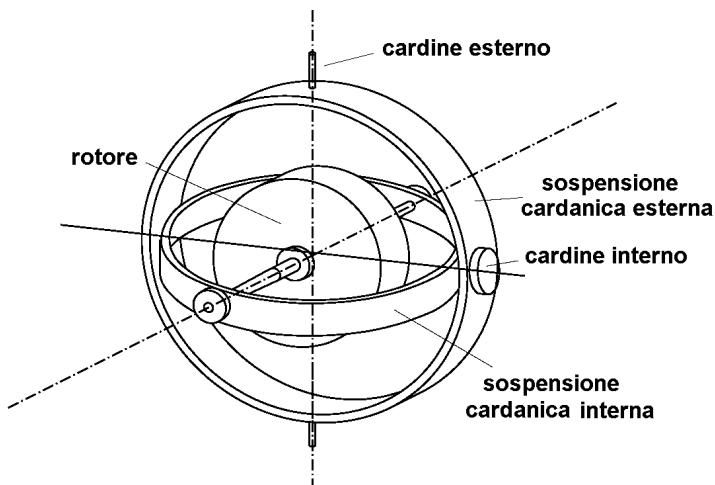


Figura 1.17 – Giroscopio a due assi

La figura 1.17 illustra la struttura di un giroscopio meccanico, il vettore momento della quantità di moto diretto nella direzione dell'asse di rotazione, deve rimanere invariato nel tempo nel sistema di riferimento inerziale adottato [Sette, 1991]. La proprietà di orientazione del giroscopio è sfruttata in navigazione perché fornisce una direzione nota e prescelta indipendentemente dai movimenti del veicolo: al variare della posizione del veicolo muta la posizione del solido in rotazione rispetto al sostegno, ma la direzione dell'asse di rotazione resta invariata nel sistema inerziale di riferimento.

I giroscopi ottici, non avendo parti in movimento, sono virtualmente liberi da manutenzione ed insensibili alla forza di gravità [Borenstein *et al.*, 1996]. Il principio si basa su due raggi laser che viaggiano in direzioni opposte lungo un percorso circolare chiuso. Le forme delle interferenze costruttive e distruttive formate dalla divisione e dalla fusione dei due raggi possono essere usati per determinare la velocità e la direzione di rotazione del dispositivo stesso.

1.4.3 SENSORI PER TELEMETRIA

La maggior parte dei sensori utilizzati per costruire una mappa dell'ambiente coinvolge sempre qualche tipo di misura delle distanze in gioco. Esistono due fondamentali approcci per effettuare tale tipo di misure:

1. Sensori basati sulla misura del tempo di volo (TOF) di un impulso di energia emessa verso un oggetto riflettente e che torna indietro verso il ricevitore.
2. Tecniche basate sulla misura dello spostamento di fase che implicano la continua trasmissione di una forma d'onda, in opposizione ai brevi impulsi utilizzati dai sistemi TOF.

SENSORI TOF

La maggior parte dei sensori in uso oggi per la misurazione di distanze è di questo tipo; essi sono tipicamente dotati di sorgenti di ultrasuoni, onde radio e raggi laser. La distanza è determinata moltiplicando la velocità con cui si muove l'onda di energia per il tempo di andata e ritorno (e dimezzando il risultato poiché il segnale di ritorno, captato dal ricevitore, percorre lo stesso tragitto del segnale emesso). La distanza assoluta di un punto è disponibile

direttamente senza la necessità di analisi complicate [Borenstein *et al.*, 1996]. Vi sono comunque potenziali sorgenti di errore:

- variazioni della velocità di propagazione. Riguardano in particolare i sistemi acustici che risentono delle variazioni di temperatura ed umidità;
- incertezze nella determinazione dell'esatto momento di arrivo dell'impulso riflesso. Sono causate dalle variazioni di intensità del segnale di ritorno a loro volta dovute alle diverse proprietà di riflessione dei materiali osservati;
- imprecisioni nei circuiti per il calcolo del ‘tempo di volo’ dell’onda. Riguardano principalmente i sensori basati su raggi laser ed onde radio, più veloci delle onde acustiche, e che richiedono sistemi di controllo più sofisticati e costosi;
- interazioni tra l’onda incidente e la superficie osservata. Il segnale captato dal ricevitore rappresenta solo una piccola parte del segnale trasmesso. La restante energia viene riflessa in tutte le direzioni e può essere assorbita o passare attraverso l’oggetto osservato a seconda delle caratteristiche della superficie e dell’angolo di incidenza del raggio. Se l’angolo con cui viene trasmessa l’onda supera un certo valore critico, l’energia riflessa non sarà più captata dal ricevitore;
- interazioni tra più sensori dello stesso tipo. Negli ambienti con ostacoli, le onde sonore, emesse da un sensore, possono essere riflesse da più oggetti e ricevute anche dagli altri sensori. Questo fenomeno è conosciuto come *crosstalk*.

I sensori TOF ad ultrasuoni sono oggi i più utilizzati per robot mobili impiegati in ambienti chiusi, principalmente per il loro basso costo. Più precisi, ma anche più costosi sono i sensori TOF basati su laser.

SENSORI PER LA MISURA DELLO SPOSTAMENTO DI FASE

Questi tipi di sensori trasmettono con continuità una forma d'onda contrariamente ai sistemi TOF che utilizzano brevi impulsi. Solo una piccola parte dell'onda di energia laser, a radiofrequenza o acustica, diretta verso l'oggetto, torna indietro con un percorso rettilineo. Tale porzione di energia è comparata con un segnale di riferimento ottenuto dal segnale di partenza: dalla differenza di fase dei due segnali si ricava la distanza dell'oggetto. Rispetto ai sensori TOF questi sistemi hanno il vantaggio di poter misurare anche la direzione e la velocità di un oggetto oltre che la sua distanza.

1.4.4 GPS E RADIOFARI

L'approccio più promettente per la navigazione in ambienti esterni è il *Sistema di Posizionamento Globale (GPS)*. Esso si basa sull'utilizzo di una costellazione di 24 satelliti in orbita intorno alla Terra ad un'altezza di circa 11.000 miglia nautiche e con un periodo di rotazione di 12 ore. Vi sono sei piani inclinati di 55 gradi rispetto all'equatore, e su ciascuno di essi sono posti quattro satelliti. L'assoluta posizione tridimensionale di ogni ricevitore GPS è determinata considerando il tempo di volo di segnali radio codificati e trasmessi dai satelliti. Conoscendo la distanza esatta tra il ricevitore di terra e tre satelliti, si possono calcolare altitudine, latitudine e longitudine del ricevitore [Borenstein *et al.*, 1996].

Un approccio meno recente è basato su *radiofari* fissi a terra. Tali sistemi possono essere di due tipi:

1. sistemi attivi di tipo radar, posti sul veicolo, che misurano i ritardi di propagazione del segnale nel percorso di andata e ritorno verso un certo numero di *transponder* fissi;
2. sistemi passivi che paragonano le differenze di fase dei segnali captati e che sono stati emessi simultaneamente da trasmettitori in posizione nota.

A causa delle interferenze che si possono avere tra sensori, i sistemi passivi sono generalmente da preferire quando un eccessivo numero di veicoli si trova ad operare nella stessa zona.

1.4.5 TELECAMERE

I *sensori di visione* (o telecamere) sono la principale fonte di informazione sull'ambiente utilizzata dai robot. Se da un lato tali sensori costituiscono una straordinaria fonte di informazione, dall'altro sono sicuramente i dispositivi più complessi che si possono utilizzare.

Una telecamera è caratterizzata dalla risoluzione che supporta (numero di linee dell'immagine) e dallo standard del segnale output prodotto [Grasso *et al.*, 1982/b]: NTSC (standard americano), PAL (standard europeo) o SECAM (standard francese). Il segnale standard prodotto dalla telecamera contiene *frames* (fotogrammi) di informazioni video e a sua volta ogni frame può essere suddiviso in *campi* (linee). Il segnale elettrico video è delimitato da impulsi di sincronismo, che permettono all'hardware di digitalizzazione (*frame grabber*) di identificare ogni linea ed ogni frame contenuti nel segnale analogico. Esistono anche telecamere, create appositamente per l'automazione, che hanno una uscita digitale.

Il processo di digitalizzazione del segnale analogico (conversione A/D) comprende un campionamento del segnale ad una frequenza fissa e la conversione della tensione in un valore che viene memorizzato e che rappresenta la luminosità dell'immagine nel punto campionato. Ne viene fuori un'immagine in cui ciascun *pixel* è rappresentato con un certo numero di bit:

- le immagini binarie contengono solo punti rappresentati con due valori: 0 o 1. Questo tipo di immagini è usato in applicazioni in cui l'unica informazione richiesta, la forma o il bordo dell'oggetto, è facilmente rilevabile grazie a particolari condizioni di illuminazione;
- le immagini a livelli di grigio, dette anche *monocromatiche*, contengono solo l'informazione relativa alla luminosità e nessuna informazione relativa al colore. Tipicamente un'immagine a livelli di grigio viene rappresentata usando 8 bit/pixel, il che permette di avere 256 livelli che forniscono una risoluzione maggiore di quella raggiungibile dal nostro sistema visivo (che ne distingue 64);
- le immagini a colori, possono utilizzare diversi sistemi di codifica dei colori (RGB, CMYK, YUV, ecc.). Una descrizione di questi sistemi esula dallo scopo del presente lavoro: una consistente documentazione è fornita in [3W, 1]. In genere per ogni punto vengono utilizzati da 16 a 32 bit, ed i colori vengono memorizzati imitando la capacità di percezione di questi da parte dell'occhio umano.

1.4.6 ALTRI TIPI DI SENSORI

Per quanto riguarda gli altri tipi di sensori si possono nominare, tra quelli a “basso livello”, i *bumpers* (sensori di contatto), i sensori ad *infrarossi* (utilizzati

in genere per scoprire la presenza di ostacoli nelle immediate vicinanze), i *sensori termici*, i sensori di olfatto (*sniffers*), ed i sensori di luminosità.

Vi sono poi *sensori doppler* che sono utilizzati per stabilire la velocità relativa del veicolo mobile, rispetto ad un riferimento esterno. Essi sono basati sul principio dello spostamento in frequenza osservato quando le onde di energia, emesse da uno strumento di bordo, vengono riflesse da una superficie che viene vista come mobile relativamente all'emettitore [Borenstein *et al.*, 1996]. I sistemi marittimi utilizzano l'energia acustica riflessa dal fondo dell'oceano (*sonar doppler*), mentre i sistemi in volo quella delle microonde riflesse dalla superficie terrestre (*radar doppler*).

Un'altra categoria di sensori è quella dei *sensori di forza*. Essi vengono in genere utilizzati per determinare quando un robot è in contatto con un altro oggetto e dove tale oggetto si trova relativamente al robot stesso.

Infine ci sono i *sensori geomagnetici* (le bussole), sensibili al campo magnetico terrestre: essi possono essere utilizzati, nei robot mobili, per il calcolo della direzione di navigazione. Si noti, però, che l'uso per applicazioni al chiuso di tali strumenti è notevolmente limitato dal fatto che il campo magnetico terrestre viene spesso distorto in prossimità di linee elettriche e di strutture metalliche.

Capitolo 2

MODELLISTICA DEL ROBOT

2.1 Modello dinamico degli attuatori e del carico

Abbiamo visto che il primo passo nella realizzazione di un robot mobile è lo studio degli attuatori e delle possibilità di movimento. Anche l'analisi degli organi di trasmissione e del carico, che ciascun motore deve movimentare, è essenziale per una corretta ed efficiente progettazione degli algoritmi di controllo. La struttura mobile scelta, la configurazione “differential drive”, è essenzialmente ottenuta accoppiando due sistemi di movimentazione identici (uno per ogni ruota); dal punto di vista della sola trasmissione del moto ciascun sistema di movimentazione è caratterizzato da quattro elementi (figura 2.1):

- **il servomotore;**
- **l'inerzia e l'attrito di trasmissione;**
- **la riduzione;**
- **l'inerzia e l'attrito del carico.**

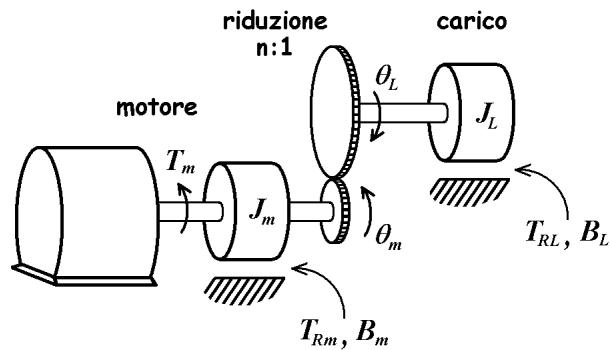


Figura 2.1 – Schema della trasmissione del moto.

Il motore esercita una coppia (T_m) sull'albero di trasmissione che è dotato di una certa inerzia (J_m); una riduzione ($n:1$) è utilizzata per aumentare la coppia e

movimentare il carico, anch'esso dotato di inerzia (J_L). Nella situazione più generale sono inoltre presenti l'attrito secco (T_{Rm} , T_{RL}) e quello viscoso (B_m , B_L), sia all'albero motore sia al carico [De Luca, 1995].

2.1.1 Il servomotore

In questo paragrafo specializziamo l'analisi ai servomotori utilizzati per movimentare il robot: trattasi di due motori in corrente continua a magneti permanenti, prodotti dalla casa americana PITTMAN (www.pittmannet.com). I vantaggi, nell'utilizzare questi motori, rispetto ai motori passo-passo (che sappiamo essere più facili da controllare) sono l'elevata potenza in rapporto a volume e peso, e l'elevata coppia di spunto. Abbiamo scartato l'utilizzo di motori "brushless", perché 2/3 volte più costosi rispetto ai motori CC.

IL MODELLO DI BASE DEL MOTORE

Il motore considerato è un servomotore in corrente continua a magneti permanenti, con spazzole e collettore, *controllato in tensione d'armatura* [Isidori, 1992]. Il circuito di eccitazione non esiste (il flusso è generato dai magneti permanenti), mentre il circuito di armatura può essere schematizzato come in figura 2.2:

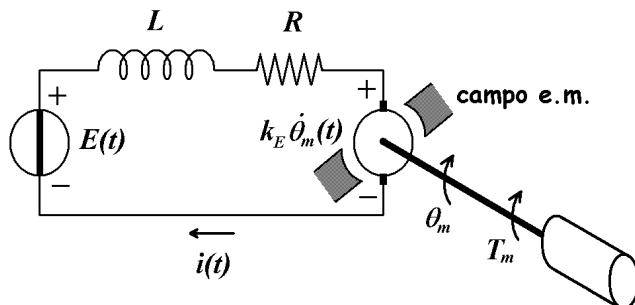


Figura 2.2 – Il modello di base del motore CC a magneti permanenti.

Nella figura vengono anche rappresentati l'angolo istantaneo di rotazione dell'asse motore e la coppia generata dall'interazione dei due campi magnetici (quello statorico e quello rotorico). Il significato dei simboli utilizzati è il seguente:

$E(t)$	tensione istantanea d'armatura;
$i(t)$	corrente istantanea d'armatura;
R	resistenza d'armatura vista dalle spazzole;
L	induttanza d'armatura vista dalle spazzole;
$\theta_m(t)$	posizione angolare istantanea dell'asse;
$T_m(t)$	coppia istantanea prodotta.

Ricordiamo che la *forza controelettromotrice* $k_E \dot{\theta}_m(t)$ è una tensione indotta, prodotta dal moto relativo tra il campo generato dal magnete permanente e le spire dell'avvolgimento del rotore ed è proporzionale alla velocità di rotazione del motore. Una semplice analisi del circuito d'armatura porta all'equazione:

$$E(t) = L \frac{di(t)}{dt} + Ri(t) + k_E \dot{\theta}_m(t)$$

mentre la coppia prodotta è data da:

$$T_m(t) = k_T i(t)$$

La *costante di coppia* k_T è una misura della coppia fornita dal motore per unità di corrente. Infatti, in un motore in corrente continua a magneti permanenti, la coppia è una funzione lineare della corrente del motore (vedere il paragrafo 1.3.1). Applicando la trasformata di Laplace si ottiene:

$$T_m(s) = \frac{k_T}{Ls + R} [E(s) - k_E s \theta_m(s)]$$

In molte applicazioni, ove la costante di tempo elettrica del motore (L/R) è significativamente minore della costante di tempo meccanica del sistema, il termine $Ldi(t)/dt$ può essere trascurato e si ottengono le equazioni:

$$E(t) \cong R \cdot i(t) + k_E \dot{\theta}_m(t) \quad T_m(s) \cong \frac{k_T}{R} [E(s) - k_E s \theta_m(s)]$$

Quando il motore è fermo, applicando una tensione E costante, si provoca negli avvolgimenti del rotore una forte corrente, limitata solo dall'impedenza del circuito (essendo la forza controelettromotrice inizialmente nulla). Tale corrente è data da:

$$I_s = \frac{E}{R}$$

Questo picco di corrente genera una corrispondente coppia di picco, che pone immediatamente in rotazione il rotore. La forza controelettromotrice inizia allora ad aumentare, limitando la corrente negli avvolgimenti. A regime la velocità di rotazione sarà quella in cui la forza controelettromotrice generata limiterà la corrente a quel valore necessario a produrre una coppia eguale a quella di carico. In breve:

$$\frac{E - k_E \dot{\theta}_m}{R} = \frac{T_m}{k_T} = I$$

Osservare che se il rotore viene bloccato la corrente torna al valore I_s che prende pertanto il nome di *corrente di stallo*, mentre la corrispondente coppia

T_S viene ovviamente chiamata *coppia di stallo*. La persistenza dello stallo può danneggiare gli avvolgimenti del rotore; per evitare questa possibilità è indispensabile un circuito in grado di limitare la corrente. Come vedremo (paragrafo 2.1.4), questo circuito introdurrà nel modello del sistema una *non linearità*, corrispondente ad una saturazione della corrente.

Vediamo ora un paio di osservazioni:

- La forza controelettromotrice $k_E \dot{\theta}_m(t)$ è una forza stabilizzante, poiché si oppone a variazioni della $\dot{\theta}_m(t)$: se un disturbo provoca un rallentamento nell'asse di rotazione, la forza controelettromotrice diminuisce, causando un aumento della corrente nel circuito d'armatura e quindi una maggiore coppia all'asse motore, che tenderà ad aumentare la velocità angolare (finché non si stabilisce un nuovo equilibrio).
- Per il bilanciamento della potenza, le costanti k_E e k_T devono essere numericamente uguali (vedere il paragrafo successivo), ma in pratica differiscono di poco.

POTENZA, COPPIA, VELOCITÀ E RENDIMENTO

Ricordiamo che il prodotto tra la tensione applicata $E(t)$ e la corrente $i(t)$ rappresenta la *potenza elettrica* P_e d'ingresso, mentre il prodotto tra la coppia $T_m(t)$ e la velocità angolare $\dot{\theta}_m(t)$ rappresenta la *potenza meccanica* P_m d'uscita. A causa degli attriti e dell'energia elettrica dissipata, la potenza meccanica in uscita dal motore è solo una frazione della potenza elettrica fornita al sistema [3W, 2]. Si può quindi definire un *rendimento del motore* dato da:

$$\eta = \frac{P_m}{P_e}$$

Se il motore è ben realizzato le perdite per attrito sono minime rispetto all'energia elettrica dissipata per effetto Joule; si può quindi scrivere:

$$P_m \cong P_e - I^2 R$$

Considerando una situazione di equilibrio, si può mostrare l'egualanza numerica delle costanti k_T e k_E . Infatti, considerando che la potenza meccanica è data dal prodotto tra la coppia e la velocità angolare dell'asse del motore, la precedente equazione diventa:

$$T_m \dot{\theta}_m = I \cdot E - I^2 R$$

Poiché $T_m = k_T I$ e la situazione è di equilibrio, vale l'espressione $E = I \cdot R + k_E \dot{\theta}_m$ e si ottiene:

$$k_T I \dot{\theta}_m = I^2 R + I k_E \dot{\theta}_m - I^2 R$$

cioè:

$$k_T = k_E$$

In pratica questi due valori si differenziano sempre per qualche punto percentuale. Ad ogni modo, per la successiva analisi, possiamo trascurare tale differenza e porre $k_T = k_E = k$. A questo punto si può facilmente mostrare che, a parità di tensione applicata E , quando la velocità angolare sale, la coppia

diminuisce e viceversa. Infatti, la coppia vale $T_m = k \cdot I$, mentre la tensione applicata è espressa da:

$$E = \frac{T_m}{k} R + k \dot{\theta}_m$$

Ricavando $\dot{\theta}_m$ si ottiene l'equazione successiva, che per $E = \text{cost}$ rappresenta una retta con pendenza negativa:

$$\dot{\theta}_m = -\left(\frac{R}{k^2}\right)T_m + \left(\frac{E}{k}\right)$$

I costruttori di motori solitamente riportano le caratteristiche dei loro prodotti sia in forma tabellare, che in forma grafica. Di particolare rilevanza è il *grafico coppia-velocità* (figura 2.3) che, per E costante, riporta in ascissa la coppia e in ordinata la velocità angolare, assieme a corrente, potenza ed efficienza del motore:

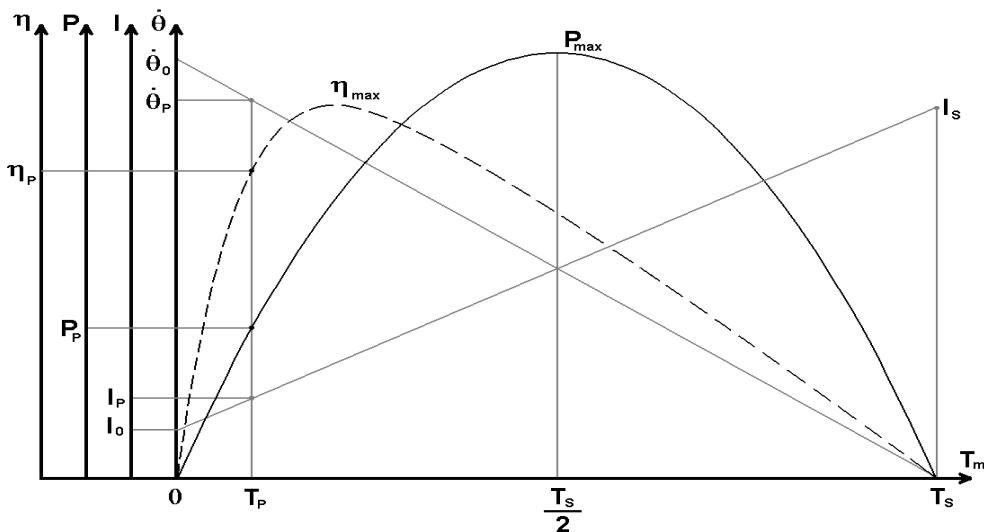


Figura 2.3 – Grafico coppia-velocità.

In questo grafico si possono osservare diverse curve; in particolare:

- Una curva $\dot{\theta}_m = \dot{\theta}(T_m)$, lineare a pendenza negativa, che intercetta gli assi in due punti dipendenti dalla tensione applicata:

$$T_m = 0 \Rightarrow \dot{\theta}_0 = \frac{E}{k} = \dot{\theta}_{m(\max)} \quad \text{Velocità senza carico}$$

$$\dot{\theta}_m = 0 \Rightarrow T_s = \frac{kE}{R} = T_{m(\max)} \quad \text{Coppia di stallo}$$

- Una parabola che rappresenta l'andamento della potenza meccanica fornita dal motore, al variare della coppia:

$$P_m = T_m \cdot \dot{\theta}_m = -\frac{R}{k^2} T_m^2 + \frac{E}{k} T_m$$

Per $T_m = 0$ e per $T_m = T_s$ si ha $P_m = 0$. Derivando la potenza meccanica rispetto alla coppia ed uguagliando a zero l'espressione ottenuta, si ricava facilmente che la potenza massima è fornita in corrispondenza della metà della coppia di stallo. In corrispondenza di tale valore, la velocità angolare del motore vale:

$$\dot{\theta}_m \left(\frac{T_s}{2} \right) = -\frac{R}{k^2} \left(\frac{kE}{2R} \right) + \frac{E}{k} = \frac{E}{2k} = \frac{\dot{\theta}_{m(\max)}}{2}$$

Per cui:

$$P_{\max} = \frac{1}{4} T_{m(\max)} \dot{\theta}_{m(\max)}$$

- Una curva $I = I(T_m) = T_m / k$, anch'essa lineare e a pendenza positiva. Per questa curva dovrebbe essere $I_0 = I(0) = 0$, ma in pratica esiste sempre una piccola corrente I_0 (corrente senza carico) necessaria per vincere gli attriti.
- Una curva $\eta = \eta(T_m)$, nella quale si può notare che la massima efficienza non è ottenuta in corrispondenza della potenza massima, bensì in un altro punto a più bassa coppia e maggiore velocità.

Ci domandiamo infine che cosa accade se il motore viene alimentato con un voltaggio più basso $E' < E$. In questo caso diminuiscono sia la coppia T_s , sia la velocità angolare $\dot{\theta}_0$, sia la corrente I_s , mentre la corrente I_0 resta invariata.

Ciò è mostrato dalla figura seguente:

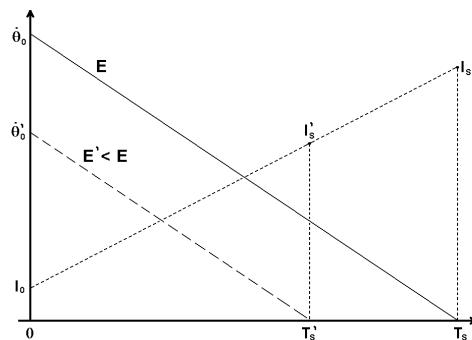


Figura 2.4 – Spostamento del punto di lavoro al variare della tensione d'armatura.

CONSIDERAZIONI TERMICHE

Nel motore le perdite di potenza sono dissipate come calore; ciò causa un innalzamento della temperatura d'armatura:

$$\Delta T = TPR \cdot P_{LOSS}$$

ove TPR è l'*impedenza termica* del motore (espressa in $^{\circ}\text{C}/\text{watt}$) cioè una misura dell'innalzamento della temperatura degli avvolgimenti, relativamente alla temperatura ambiente, per unità di potenza dissipata, mentre P_{LOSS} rappresenta la quantità di potenza dissipata e vale:

$$P_{LOSS} = I^2 R + T_R \dot{\theta}_m$$

essendo T_R la coppia persa per attrito. Solitamente i costruttori forniscono il coefficiente TPR con il motore in una situazione di stallo, in aria e senza dissipatori aggiuntivi. Questo corrisponde al peggior caso possibile, in quanto, la rotazione causa un flusso d'aria all'interno del motore che migliora il trasferimento di calore, causando una diminuzione dell'impedenza termica.

Purtroppo la resistenza d'armatura, la costante di coppia e il *gradiente di tensione motore* (k_E) sono funzioni della temperatura: un innalzamento di temperatura provoca una variazione in questi parametri, in modo tale da degradare le prestazioni del motore ed aumentare le perdite di potenza. Occorre quindi prestare attenzione perché, anche in presenza di un accettabile innalzamento di temperatura, il mantenimento di una coppia costante in uscita può portare il motore ad una “fuga termica”, con conseguente fusione degli avvolgimenti.

MISURA DEI PARAMETRI DEL MOTORE

A volte i datasheets dei motori sono incompleti o non sono disponibili (nel caso di motore acquistato presso un rivenditore surplus). In tal caso occorre procedere ad una serie di misure per identificare i parametri essenziali [3W, 2]:

- *Resistenza elettrica*: non può essere misurata con un ohmetro perché la scarsa corrente che questo strumento mette a disposizione non è in grado di penetrare totalmente il sottile film che ricopre la superficie di commutazione. Occorre invece bloccare l'asse del motore, applicare una tensione in grado di far scorrere una corrente di diverse decine di milliampercere, misurare la resistenza come rapporto $R = E / I$, ripetere il procedimento per diverse posizioni angolari dell'asse e mediare i risultati.
- *Induttanza*: deve essere misurata ad 1 KHz e mediata su diverse posizioni angolari dell'asse.
- *Gradiente di tensione motore*: questo parametro può essere misurato ai terminali del motore usato come dinamo, ponendo in rotazione l'asse a velocità costante e nota.
- *Costante di coppia*: si può determinare misurando la corrente e la coppia in due differenti punti di lavoro e calcolando

$$k_T = \frac{T_2 - T_1}{I_2 - I_1}$$

- *Attrito secco*: si può stabilire usando un generatore di corrente costante, misurando la corrente richiesta per stabilire la rotazione continua e moltiplicando il risultato per la costante di coppia.
- *Attrito dinamico*: è in genere composto da due componenti, una costante e una funzione della velocità angolare. In prima approssimazione si può

calcolare moltiplicando la corrente senza carico e la costante di coppia alla data velocità.

- *Coppia di stallo:* occorre effettuare varie misure di velocità angolare per vari valori di coppia compresi tra 0 e $\frac{3}{4} T_S$. Il punto di intersezione tra la retta che meglio approssima questi punti e l'asse di coppia determina il valore della coppia di stallo. In realtà, l'effettiva coppia di stallo può risultare un po' inferiore di quella calcolata, comunque la regione di stallo non è una zona consigliata per il buon funzionamento del motore.

2.1.2 La trasmissione

Per caratterizzare il trasferimento della coppia prodotta dal motore fino agli ingranaggi della riduzione occorre considerare le eventuali coppie resistenti di attrito secco e di attrito viscoso presenti nella trasmissione del moto. Applicando l'equilibrio meccanico dinamico, che tiene conto anche delle forze inerziali, si ottiene (omettendo il tempo t per una maggiore chiarezza tipografica):

$$T_m = J_m \ddot{\theta}_m + B_m \dot{\theta}_m + T_{Rm} sgn(\dot{\theta}_m) + T_L$$

ove:

- $T_m(t)$ è la coppia prodotta dal motore;
- J_m è il momento d'inerzia totale del rotore, dell'asse e degli ingranaggi;
- B_m è il coefficiente di attrito viscoso totale del rotore, dell'asse e degli ingranaggi;
- T_{Rm} è l'attrito secco totale del rotore, dell'asse e degli ingranaggi;
- $T_L(t)$ è la coppia di carico riportata all'asse del motore (tramite la riduzione).

Il termine di attrito secco, il cui valore assoluto è costante, si oppone sempre alla direzione del moto e introduce quindi un'altra *non linearità* nel sistema (la funzione $sgn(x)$ restituisce infatti $-1, 0, +1$ a seconda che il valore di x sia negativo, nullo o positivo). Ciò andrà considerato in modo opportuno nel modello del sistema.

2.1.3 La riduzione

Quando un attuatore non è in grado di fornire una buona coppia motrice a basse velocità, si utilizza un sistema di riduzione per aumentare la coppia motrice al carico [De Luca, 1995]; la riduzione moltiplica per n la coppia e riduce di n la velocità angolare (ove $n > 1$ è il *fattore di riduzione*):

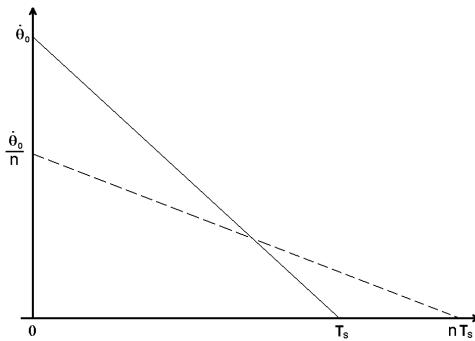


Figura 2.5 - Coppia e velocità angolare all'ingresso e all'uscita della riduzione.

Purtroppo, a causa dell'attrito tra ingranaggi, una riduzione presenta sempre un'efficienza minore di 1. Con riduzioni a planetario particolarmente buone si possono raggiungere valori del 90% di efficienza, mentre si può scendere anche sotto il 50% nel caso di riduzioni scadenti. Con riferimento alla figura 2.6, sia:

- n:1** il *rappporto di riduzione* ($n > 1$): ogni n giri dell'albero motore il carico compie un giro. Vale quindi l'equazione: $2\pi r_L = n \cdot 2\pi r_m$

ζ l'efficienza della riduzione definita come: $\zeta = \frac{P_o}{P_i}$

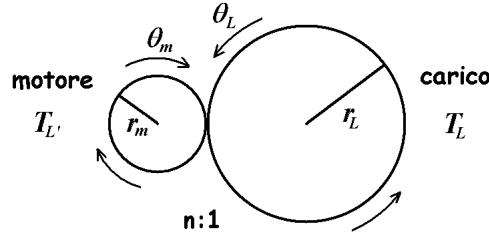


Figura 2.6 – Schema di principio di una riduzione.

Considerando la stessa distanza lineare percorsa dai denti degli ingranaggi, si ha:

$$\theta_m \cdot r_m = \theta_L \cdot r_L = \theta_L \cdot n \cdot r_m \Rightarrow \theta_m = n \cdot \theta_L$$

cioè

$$\dot{\theta}_m = n \cdot \dot{\theta}_L \Rightarrow \dot{\theta}_L = \frac{1}{n} \dot{\theta}_m$$

Si osservi che θ_L e θ_m sono legate mediante una semplice costante perché abbiamo implicitamente ipotizzato di avere a che fare con una trasmissione rigida: se essa fosse stata elastica, la presenza di una possibile torsione avrebbe comportato una relazione differenziale tra le due grandezze θ_L e θ_m .

Dunque la velocità di rotazione al carico è effettivamente ridotta di un fattore n .

Per quanto riguarda la coppia, considerando il bilanciamento delle potenze di uscita e di ingresso alla riduzione, si ha:

$$T_L \cdot \dot{\theta}_L = P_o = \zeta \cdot P_i = \zeta \cdot T_{L'} \cdot \dot{\theta}_m$$

da cui si ottiene:

$$T_L = \zeta \cdot \frac{\dot{\theta}_m}{\dot{\theta}_L} \cdot T_{L'} \quad \Rightarrow \quad T_L = n \cdot \zeta \cdot T_{L'}$$

La coppia fornita al carico viene quindi amplificata di un fattore $n \cdot \zeta$. Si comprende come sia di fondamentale importanza l'efficienza della riduzione. Sostituendo $T_{L'}$ nell'equazione dell'equilibrio ricavato nella sezione precedente, si ottiene:

$$T_m = J_m \ddot{\theta}_m + B_m \dot{\theta}_m + T_{Rm} sgn(\dot{\theta}_m) + \frac{T_L}{n \cdot \zeta}$$

2.1.4 Il carico

Siamo ora in grado di determinare la condizione di equilibrio meccanico con tutte le coppie agenti sull'albero di rotore. Considereremo inizialmente un solo motore con carico generico rotativo e successivamente specializzeremo l'analisi al caso del robot, con struttura mobile di tipo “differential drive”.

CARICO GENERICO IN ROTAZIONE

L'equazione di equilibrio al carico è espressa da:

$$T_L = J_L \ddot{\theta}_L + B_L \dot{\theta}_L + T_{RL} sgn(\dot{\theta}_L) + T_D$$

ove:

- $\mathbf{T}_L(t)$ è la coppia utile;
- \mathbf{J}_L è il momento d'inerzia del carico;
- \mathbf{B}_L è il coefficiente di attrito viscoso del carico;
- \mathbf{T}_{RL} è l'attrito secco del carico;
- $\mathbf{T}_D(t)$ è l'eventuale coppia di disturbo.

Esprimendo θ_L in funzione di θ_m si ottiene:

$$T_L = \frac{J_L}{n} \ddot{\theta}_m + \frac{B_L}{n} \dot{\theta}_m + T_{RL} \operatorname{sgn}(\dot{\theta}_m) + T_D$$

Sostituendo T_L nell'equazione che esprime l'equilibrio dinamico sull'asse del motore e raccogliendo i termini si ottiene:

$$T_m = \left(J_m + \frac{J_L}{n^2 \zeta} \right) \ddot{\theta}_m + \left(B_m + \frac{B_L}{n^2 \zeta} \right) \dot{\theta}_m + \left(T_{Rm} + \frac{T_{RL}}{n \cdot \zeta} \right) \operatorname{sgn}(\dot{\theta}_m) + \frac{T_D}{n \cdot \zeta}$$

Come si può osservare dall'equazione, le coppie agenti al carico sono fortemente ridotte all'asse motore e diventano comparabili con i parametri caratteristici del motore (in modo duale l'inerzia, l'attrito viscoso e l'attrito secco del motore e della trasmissione sono riportati amplificati al carico e pertanto non possono essere trascurati). Indicando con:

$$J_{eff} = J_m + \frac{J_L}{n^2 \zeta} \quad B_{eff} = B_m + \frac{B_L}{n^2 \zeta} \quad T_{Reff} = T_{Rm} + \frac{T_{RL}}{n \cdot \zeta}$$

si ottiene:

$$T_m = J_{eff} \ddot{\theta}_m + B_{eff} \dot{\theta}_m + T_{Reff} \operatorname{sgn}(\dot{\theta}_m) + \frac{T_D}{n \cdot \zeta}$$

Posto

$$T' = T_m - T_{Reff} \operatorname{sgn}(\dot{\theta}_m) - \frac{T_D}{n \cdot \zeta}$$

si può osservare che

$$T'(s) = (J_{\text{eff}} s^2 + B_{\text{eff}} s) \theta_m(s) = (J_{\text{eff}} s + B_{\text{eff}}) \dot{\theta}_m(s)$$

da cui:

$$\frac{\dot{\theta}_m(s)}{T'(s)} = \frac{1}{J_{\text{eff}} s + B_{\text{eff}}}$$

Considerando anche le equazioni del motore si ottiene il seguente modello del sistema (figura 2.7), il cui controllo è detto *in tensione di armatura*, essendo la variabile di ingresso la tensione $E(t)$ fornita al motore. La variabile di uscita $\theta_L(t)$ è la posizione dell'asse di carico; vi è inoltre una generica coppia di disturbo $T_D(t)$ (positiva se resistente), riportata all'asse del motore tramite la riduzione ed utilizzabile per verificare la robustezza degli algoritmi di controllo in presenza di disturbi agenti sul sistema:

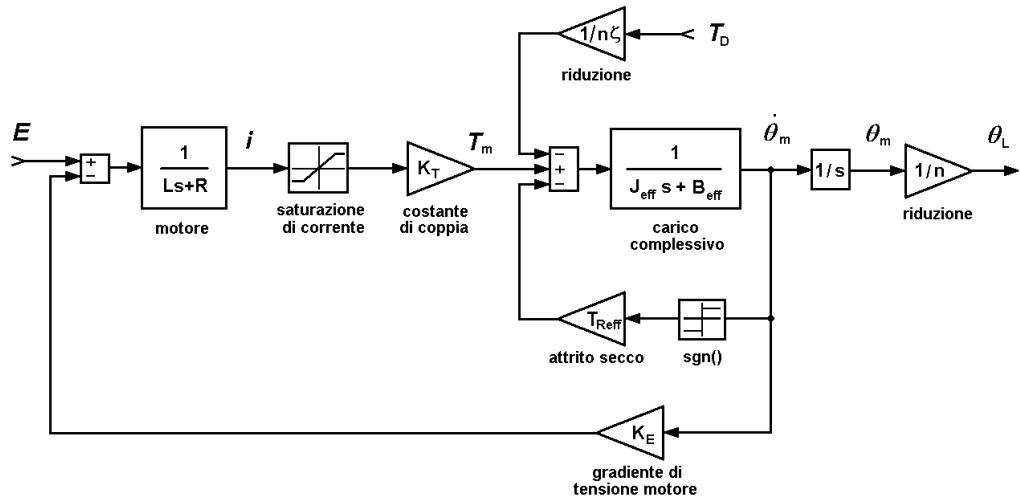


Figura 2.7 – Modello del sistema di movimentazione con controllo in tensione d’armatura.

Si tratta dunque di un sistema del 3° ordine, con un polo nell'origine (per θ_m ma non per $\dot{\theta}_m$) e due costanti di tempo:

- L/R costante di tempo elettrica,
- J_{eff}/B_{eff} costante di tempo meccanica.

In molti sistemi elettromeccanici il circuito elettrico del motore va a regime ben più rapidamente della parte meccanica: in tal caso vale la condizione $L/R \ll J_{eff}/B_{eff}$ e la quantità $I/(Ls+R)$ diventa approssimabile con $1/R$. Si ottiene così un sistema del 2° ordine con una sola costante di tempo su cui risulta più facile la progettazione del sistema di controllo.

Un altro possibile modello del sistema, cui corrisponde un *controllo in corrente di armatura*, è il seguente:

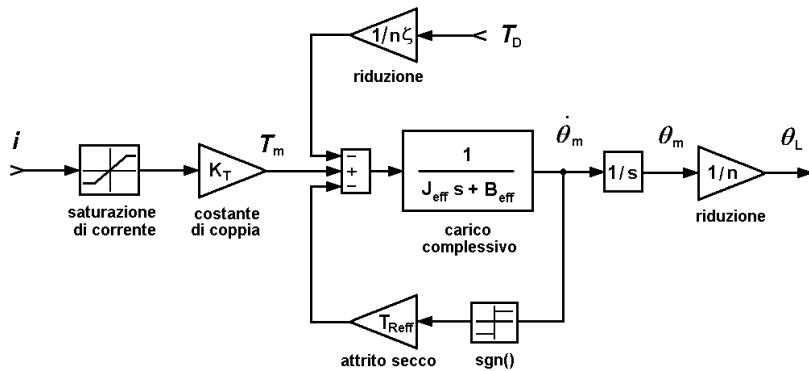


Figura 2.8 - Modello del sistema di movimentazione con controllo in corrente d'armatura.

In questo caso la variabile di ingresso è la $i(t)$, fornita al motore tramite un generatore di corrente. La variabile di uscita $\theta_L(t)$ è sempre la posizione dell'asse di carico. Il modello che ne deriva è del 2° ordine, con un polo nell'origine e una costante di tempo (J_{eff}/B_{eff}).

BASE MOBILE “DIFFERENTIAL DRIVE”

Immaginiamo che la base sia in movimento rettilineo a velocità \vec{v} , come mostrato in figura:

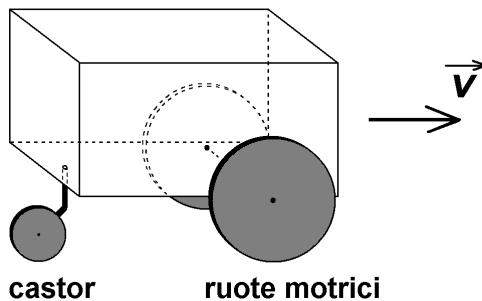


Figura 2.9 – Base mobile in movimento rettilineo.

Ricordiamo che questa struttura mobile è dotata di due ruote motrici anteriori, motorizzate in modo indipendente e di un castor (ruota passiva basculante) posteriore. Per descrivere il moto di questo sistema, dobbiamo utilizzare le equazioni cardinali della meccanica:

$$\begin{cases} \sum_i \vec{F}_i^{(e)} = m \frac{d\vec{v}}{dt} \\ \sum_i \vec{M}_{ij}^{(e)} = I_j \frac{d\vec{\omega}_j}{dt} \end{cases}$$

ove $j = 1, 2$ e 3 specializza l'equazione a ciascuna delle tre ruote. In questa analisi siamo più interessati alla caratterizzazione delle prestazioni dinamiche (cioè velocità e accelerazione), piuttosto che alla traiettoria (vedremo la cinematica nel paragrafo 2.2). Introduciamo quindi un'ipotesi di moto rettilineo: sotto questa ipotesi, la velocità lineare del centro di massa \vec{v} coincide con la velocità lineare dei baricentri delle ruote e le due ruote motrici condividono la stessa velocità angolare $\vec{\omega}$. Inoltre il castor risulta allineato alla direzione di moto. Questo permette di ridurre il numero di equazioni e di semplificare l'analisi. Ma prima definiamo i simboli utilizzati nel seguito:

A_{LM}	attrito di rotolamento tra ruota motrice e pavimento;
A_{LP}	attrito di rotolamento tra castor e pavimento;
m_{LM}	massa della ruota motrice;
m_{LP}	massa del castor;
m_{LR}	massa del robot (ruote escluse);
$a(t)$	accelerazione lineare del robot;
I_{LM}	momento d'inerzia della ruota motrice;
I_{LP}	momento d'inerzia del castor;
$\ddot{\theta}_{LM}(t)$	accelerazione angolare della ruota motrice;
$\ddot{\theta}_{LP}(t)$	accelerazione angolare del castor.
r_{LM}	Raggio della ruota motrice
r_{LP}	Raggio del castor.
$T_L(t)$	Coppia utile applicata ad una ruota motrice.

Si consideri quindi il seguente schema, che tiene conto di tutte le forze ed i momenti agenti sulle varie parti del robot (in grigio sono indicate le *forze interne*):

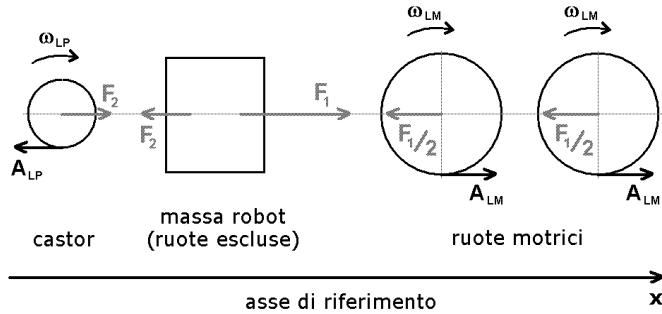


Figura 2.10 – Rappresentazione di forze e momenti agenti sulle parti del robot.

Trascurando l'attrito dell'aria, si può riscrivere il sistema di equazioni cardinali nel modo seguente:

$$\begin{cases} 2A_{LM} - A_{LP} = (2m_{LM} + m_{LR} + m_{LP})a \\ I_{LM}\ddot{\theta}_{LM} = T_L - A_{LM}r_{LM} \\ I_{LP}\ddot{\theta}_{LP} = A_{LP}r_{LP} \end{cases}$$

Per un moto di rotolamento vale l'espressione $v = \dot{\theta}_{LM} \cdot r_{LM}$ perciò si ha:

$$a = \ddot{\theta}_{LM}r_{LM} = \ddot{\theta}_{LP}r_{LP}$$

Esplicitando gli attriti dalle ultime due equazioni del sistema si ottiene:

$$\begin{cases} A_{LM} = \frac{T_L - I_{LM}\ddot{\theta}_{LM}}{r_{LM}} \\ A_{LP} = \frac{I_{LP}\ddot{\theta}_{LP}}{r_{LP}} = \frac{I_{LP}}{r_{LP}} \frac{r_{LM}}{r_{LP}} \ddot{\theta}_{LM} \end{cases}$$

Sostituendo A_{LM} , A_{LP} e l'accelerazione a nella prima equazione cardinale si ha:

$$2\left(\frac{T_L - I_{LM}\ddot{\theta}_{LM}}{r_{LM}}\right) - \frac{I_{LP}}{r_{LP}}\frac{r_{LM}}{r_{LP}}\ddot{\theta}_{LM} = (2m_{LM} + m_{LR} + m_{LP})\ddot{\theta}_{LM}r_{LM}$$

Ricavando T_L si può porre:

$$T_L = J_{LMeq} \cdot \ddot{\theta}_{LM}$$

ove

$$J_{LMeq} = \frac{r_{LM}^2}{2} \left(\frac{2I_{LM}}{r_{LM}^2} + \frac{I_{LP}}{r_{LP}^2} + 2m_{LM} + m_{LR} + m_{LP} \right)$$

può essere considerato il momento d'inerzia equivalente visto dall'asse di una ruota motrice. Riportando la coppia utile all'asse del motore si ottiene:

$$T_L = \frac{J_{LMeq}}{n} \dot{\theta}_m$$

Sostituendo T_L nell'equazione dell'equilibrio dinamico presente sull'asse del motore e raccogliendo i termini si ha:

$$T_m = \left(J_m + \frac{J_{LMeq}}{n^2 \zeta} \right) \ddot{\theta}_m + B_m \dot{\theta}_m + T_{Rm} \operatorname{sgn}(\dot{\theta}_m)$$

pertanto i parametri del modello di simulazione sono:

$$J_{eff} = J_m + \frac{J_{LMeq}}{n^2 \zeta} \quad B_{eff} = B_m \quad T_{Reff} = T_{Rm} \quad T_D = 0$$

Per ottenere lo spazio percorso dal centro di massa del robot, basta ricordare che il moto è di rotolamento ($v = \dot{\theta}_{LM} \cdot r_{LM}$), perciò si ha:

$$s = \dot{\theta}_{LM} \cdot r_{LM}$$

Si può quindi ottenere lo spazio percorso dal robot semplicemente aggiungendo un blocco di guadagno r_{LM} all'uscita del modello precedentemente riportato in figura 2.7:

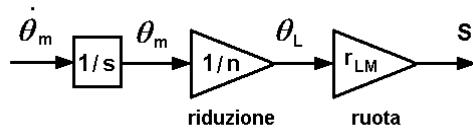


Figura 2.11 – Il guadagno R_{LM} permette di ottenere lo spazio percorso dal robot.

Infine, per garantire che il moto sia effettivamente di rotolamento, occorre evitare la possibilità che le ruote motrici slittino per l'eccessiva coppia. Tale slittamento è evitato sicuramente se l'attrito di rotolamento si mantiene minore del prodotto tra coefficiente di attrito statico e reazione vincolare del pavimento nel punto di contatto con la ruota. Deve valere perciò la condizione:

$$A_{LM} = \frac{T_L - I_{LM} \ddot{\theta}_{LM}}{r_{LM}} = \frac{(J_{LMeq} - I_{LM}) \ddot{\theta}_{LM}}{r_{LM}} \leq \mu_{S(LM)} \left(m_{LM} + \frac{1}{3} m_{LR} \right) g$$

essendo $\mu_{S(LM)}$ il coefficiente di attrito statico e g la forza di gravità. Nella formula precedente si è ipotizzato implicitamente che su ogni ruota (le due

motrici e il castor), gravi un terzo della massa del robot (ruote escluse). La condizione precedente riportata all'asse del motore diventa:

$$\ddot{\theta}_m \leq \frac{n \cdot r_{LM} \cdot \mu_{S(LM)} \cdot g}{J_{LMeq} - I_{LM}} \left(m_{LM} + \frac{1}{3} m_{LR} \right)$$

Poiché il controllo prevede la lettura degli encoder dei motori, risulta abbastanza facile implementare via software la condizione precedente. In alternativa è possibile, dopo l'esame dei grafici di simulazione, stabilire una limitazione hardware di corrente adatta a garantire la condizione di rotolamento.

2.2 Cinematica del robot

La cinematica esprime le relazioni che legano il movimento del veicolo ad un sistema di coordinate solidale con l'ambiente in cui si muove il robot. Poiché nella realtà il moto è soggetto ad una serie di disturbi, per poter determinare le leggi cinematiche è necessario porre alcune ipotesi fondamentali:

1. il veicolo è un corpo rigido: gli unici movimenti possibili sono le rotazioni delle ruote intorno agli assi dei motori e la rotazione orizzontale e verticale del castor;
2. le forze generate dall'attrito del castor con il terreno sono trascurabili, in modo da non provocare alterazioni nel moto generato dalle ruote motrici;
3. gli assi di rotazione delle ruote motrici sono perfettamente paralleli al terreno;
4. per le ruote è sempre verificata la condizione di perfetto rotolamento;
5. le ruote sono ideali, ossia indeformabili e di spessore nullo;
6. la superficie da percorrere è perfettamente piana e orizzontale;
7. non ci sono forze esterne agenti sul veicolo.

2.2.1 Cinematica diretta

La figura seguente illustra le variabili di ingresso e di uscita del modello:

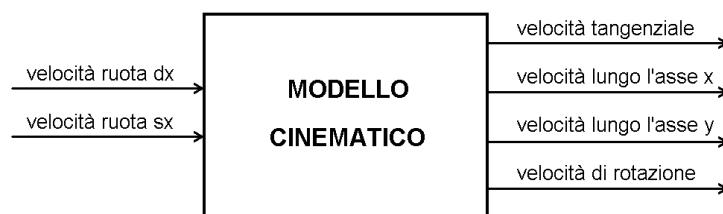


Figura 2.12 – Variabili di ingresso/uscita della cinematica diretta.

Come variabili di ingresso prendiamo le velocità di rotazione delle due ruote posteriori, in quanto esse sono le due ruote attuate e sensorizzate.

Approssimando l'ambiente a un piano cartesiano, la posizione del veicolo sarà identificata dalle coordinate (x_p, y_p) di un suo punto P generico (questo è possibile in quanto la struttura è rigida) e dall'angolo φ fra l'asse di avanzamento del veicolo e uno degli assi cartesiani. Per semplificare scegliamo come punto P quello mediano sull'asse delle ruote motrici e come asse di riferimento l'asse delle ascisse. Riassumendo abbiamo:

- x_p ascissa del punto P nel piano;
- y_p ordinata del punto P nel piano;
- φ orientazione del veicolo nel piano.

Per ricavare queste variabili di interesse, dovremo passare attraverso l'operazione di integrazione. Infatti, come vedremo, è possibile legare soltanto le grandezze derivate, in quanto la traiettoria percorsa dal veicolo dipende dalla velocità delle sue ruote [De Luca, 2000]. Si consideri la seguente figura:

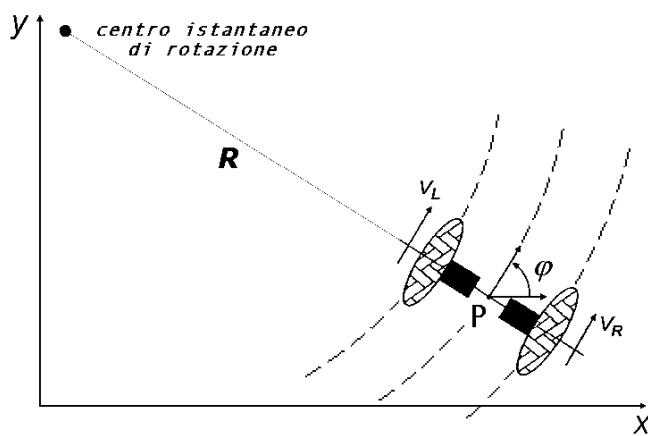


Figura 2.13 – Traiettorie percorse dal veicolo in moto.

In ogni istante le ruote motrici, essendo soggette ai vincoli di perfetto rotolamento ed essendo allineate sul medesimo asse, non possono che percorrere un tratto infinitesimo di due circonferenze concentriche; lo stesso farà il punto P. Siano allora:

- $v_L(t)$ la velocità lineare della ruota sinistra;
- $v_R(t)$ la velocità lineare della ruota destra;
- $\dot{\theta}_L(t)$ la velocità angolare della ruota sinistra;
- $\dot{\theta}_R(t)$ la velocità angolare della ruota destra;
- r_M il raggio delle due ruote motrici;
- d la distanza tra le due ruote motrici;
- $R(t)$ il raggio della circonferenza istantanea passante per P;
- $\Omega(t)$ la velocità angolare del punto P sulla circonferenza istantanea;
- $v_P(t)$ la velocità tangenziale del punto P sulla circonferenza istantanea.

Possiamo scrivere (omettendo il tempo t per maggiore chiarezza tipografica):

$$v_L = \dot{\theta}_L \cdot r_M = \Omega(R - d/2)$$

$$v_R = \dot{\theta}_R \cdot r_M = \Omega(R + d/2)$$

Da queste possiamo ricavare:

$$R = \frac{d}{2} \frac{\dot{\theta}_R + \dot{\theta}_L}{\dot{\theta}_R - \dot{\theta}_L}$$

$$\Omega = r_M \frac{\dot{\theta}_R - \dot{\theta}_L}{d}$$

Si osservi che i segni saranno positivi se la ruota interna alla curva è la sinistra, negativi altrimenti. Ciò non limita l'utilizzo del modello, in quanto il segno negativo del raggio equivale al centro istantaneo di rotazione posto sulla destra del veicolo e quindi ad una velocità angolare negativa (rotazione in senso orario). Si noti inoltre che, se le velocità angolari delle due ruote sono uguali, R risulta infinito e Ω nullo: il moto istantaneo che ne deriva è rettilineo.

La velocità tangenziale del punto P sarà data da:

$$v_p = \Omega \cdot R = r_M \frac{\dot{\theta}_R + \dot{\theta}_L}{2}$$

mentre le velocità lungo gli assi x , y e la velocità angolare sono:

$$\begin{aligned} \dot{\phi} &= \Omega = r_M \frac{\dot{\theta}_R - \dot{\theta}_L}{d} \\ \dot{x}_p &= v_{px} = v_p \cos(\varphi) = r_M \frac{\dot{\theta}_R + \dot{\theta}_L}{2} \cos(\varphi) \\ \dot{y}_p &= v_{py} = v_p \sin(\varphi) = r_M \frac{\dot{\theta}_R + \dot{\theta}_L}{2} \sin(\varphi) \end{aligned}$$

Le componenti x_p , y_p , φ si ottengono integrando le espressioni precedenti:

$$\begin{aligned} \varphi(t) &= \varphi_0 + \frac{r_M}{d} \int_0^t [\dot{\theta}_R(\tau) - \dot{\theta}_L(\tau)] d\tau \\ x_p(t) &= x_{p0} + \frac{r_M}{2} \int_0^t [\dot{\theta}_R(\tau) + \dot{\theta}_L(\tau)] \cos[\varphi(\tau)] d\tau \end{aligned}$$

$$y_P(t) = y_{P0} + \frac{r_M}{2} \int_0^t [\dot{\theta}_R(\tau) + \dot{\theta}_L(\tau)] \sin[\phi(\tau)] d\tau$$

2.2.2 Cinematica inversa

Se consideriamo il moto planare del veicolo come composto da due componenti, una di traslazione pura e una di rotazione, possiamo considerare la cinematica inversa di ognuna e poi unirle per ottenere quella complessiva. La componente di traslazione è:

$$v_P = \frac{r_M}{2} (\dot{\theta}_R + \dot{\theta}_L) = \pm \sqrt{\dot{x}_P^2 + \dot{y}_P^2}$$

il cui segno indica se il veicolo si sta spostando in avanti o a marcia indietro. Se si esclude questa seconda ipotesi, v_P può assumersi sempre positivo.

La componente rotazionale è:

$$\dot{\phi} = \frac{r_M}{d} (\dot{\theta}_R - \dot{\theta}_L)$$

Ricavando $\dot{\theta}_L$ e $\dot{\theta}_R$ dalle precedenti equazioni, si ottiene il modello cinematico inverso:

$$\dot{\theta}_R = \frac{v_P + \frac{d}{2} \dot{\phi}}{r} = \frac{\sqrt{\dot{x}_P^2 + \dot{y}_P^2} + \frac{d}{2} \dot{\phi}}{r}$$

$$\dot{\theta}_L = \frac{v_P - \frac{d}{2} \dot{\phi}}{r} = \frac{\sqrt{\dot{x}_P^2 + \dot{y}_P^2} - \frac{d}{2} \dot{\phi}}{r}$$

2.2.3 Odometria ed ambiguità di posizionamento

Nella configurazione “differential drive” l’odometria si serve degli encoder incrementali, calettati sugli assi dei motori, per la stima della posizione e l’orientazione del veicolo [Borenstein *et al.*, 1996]. Ricordiamo che è più corretto parlare di stima piuttosto che di misurazione perché l’integrazione nel tempo di informazioni incrementali sul movimento genera un inevitabile accumulo di errori, dovuti al fatto che la maggior parte delle ipotesi fatte nella determinazione del modello cinematico non sono soddisfatte. In particolare:

1. le ruote non sono ideali, esse subiscono una deformazione che dipende dalla loro costruzione, dal peso del robot e dalla rigidità del terreno; spesso tale deformazione è diversa da ruota a ruota;
2. per le ruote non sempre è verificata la condizione di perfetto rotolamento: possono verificarsi slittamenti, ad esempio dovuti a terreno scivoloso o forti accelerazioni;
3. lo spessore delle ruote motrici non è mai nullo e possono nascere forze di attrito laterali che ne alterano il moto;
4. non sempre gli assi delle ruote motrici sono perfettamente allineati e paralleli al terreno;
5. le forze generate dall’attrito del castor con il terreno molto spesso non sono trascurabili e dipendono dalla posizione assunta dal castor durante il moto;
6. la superficie da percorrere non è mai perfettamente piana e orizzontale; a volte possono esserci oggetti imprevisti;
7. possono esserci forze esterne che agiscono sul robot (ad esempio il vento, se il veicolo opera all’esterno di edifici).

Inoltre:

8. la risoluzione degli encoder e il periodo di campionamento finito condizionano la velocità di rotazione ottenibile dai motori: questo comporta una discretizzazione dei raggi di curvatura;
9. gli errori di orientazione causano errori di posizione che aumentano con la distanza percorsa dal robot.

Dunque gli errori odometrici dipendono in parte dalle caratteristiche costruttive del veicolo. In particolare si può considerare che:

- veicoli con una piccola distanza tra le ruote sono soggetti ad errori di orientazione più grandi rispetto ai veicoli che hanno distanze maggiori;
- castor sottoposti ad una significativa porzione del peso totale sono possibili fonti di slittamento quando invertono la direzione;
- le ruote usate per l'odometria dovrebbero essere sottili e non comprimibili: la ruota ideale dovrebbe essere di alluminio con un sottile strato di gomma per migliorare la trazione, ma spesso ciò non accade perché le ruote odometriche sono di solito ruote motrici che richiedono, quindi, una larga superficie di contatto con il terreno;
- sia gli encoder, sia i controllori di velocità dovrebbero avere elevata risoluzione, compatibile con la massima precisione ottenibile dalla struttura meccanica adottata.

Nonostante queste limitazioni, l'odometria è sicuramente il metodo di stima più utilizzato per la navigazione dei robot mobili, in quanto garantisce una buona precisione nelle misure a breve tempo, è poco costosa e consente alte frequenze di campionamento. Essa può essere usata per fornire una valutazione corretta della posizione per un tempo sufficiente a garantire, alle tecniche basate su landmark e “matching” di mappe, l'elaborazione dei loro dati sensoriali; in

alcuni casi l'odometria è l'unica fonte di informazione disponibile per la navigazione, soprattutto quando il robot si muove in ambienti del tutto sconosciuti o quando accade che altri tipi di sensori non riescano a fornire dati utilizzabili.

EQUAZIONI PER L'ODOMETRIA

Assumiamo che all'istante i l'encoder della ruota sinistra e quello della ruota destra mostrino un incremento nel conteggio degli impulsi rispettivamente pari a N_L^i e N_R^i . Sia inoltre:

$$K_e = \frac{2\pi \cdot r_M}{n \cdot C_e}$$

ove:

- K_e è il fattore numerico che converte il numero di impulsi acquisiti dagli encoder nelle distanze curvilinee percorse dalle rispettive ruote;
- r_M è il raggio nominale delle ruote motrici;
- C_e è la risoluzione degli encoder (in impulsi/giro);
- n è il rapporto di riduzione tra l'asse dei motori (a cui gli encoder sono collegati) e l'asse delle ruote motrici.

Ora possiamo calcolare la distanza incrementale percorsa dalle ruote durante l' i -esimo intervallo di campionamento [Klarer, 1988]:

$$\Delta S_L^i = K_e \cdot N_L^i$$

$$\Delta S_R^i = K_e \cdot N_R^i$$

e la distanza incrementale percorsa dal punto P (vedere paragrafo 2.2.1):

$$\Delta S_P^i = \frac{\Delta S_R^i + \Delta S_L^i}{2}$$

La variazione incrementale dell'orientazione è data da:

$$\Delta \theta^i = \frac{\Delta S_R^i - \Delta S_L^i}{d}$$

ove d è la distanza tra le ruote, idealmente misurata tra i due punti di contatto delle ruote con il terreno. Dunque la nuova orientazione del veicolo è:

$$\theta^i = \theta^{i-1} + \Delta \theta^i$$

e la posizione relativa del punto P:

$$x_P^i = x_P^{i-1} + \Delta S_P^i \cos(\theta^i)$$

$$y_P^i = y_P^{i-1} + \Delta S_P^i \sin(\theta^i)$$

ELLISSI D'ERRORE

Le sorgenti di errore viste, possono essere suddivise in due categorie *errori sistematici* ed *errori non sistematici* [Borenstein *et al.*, 1996]. Appartengono alla prima categoria errori derivati da disallineamento delle ruote, diametri e distanza tra le ruote differenti dal valore nominale, risoluzione e periodo di campionamento finiti per gli encoder. Mentre alla seconda categoria appartengono errori dovuti a passaggi su terreno accidentato o oggetti imprevisti, contatto non perfetto col terreno, slittamenti delle ruote e forze impreviste (interne ed esterne).

Questa distinzione è utile allo scopo di ridurre gli errori. Ad esempio, gli errori sistematici sono particolarmente gravi poiché vanno costantemente accumulandosi e, su superfici abbastanza regolari, pesano molto più di quelli non sistematici. Tuttavia, su superfici irregolari, gli errori del secondo tipo risultano essere dominanti e difficilmente possono essere previsti, soprattutto in ambienti poco noti.

Molti ricercatori studiano algoritmi che stimano l'incertezza della posizione di un robot [Borenstein e Feng, 1995]. Grazie a questo approccio, ogni posizione calcolata dal robot è contornata da un'*ellisse di errore* che indica una regione di incertezza per la posizione attuale del robot (figura 2.14):

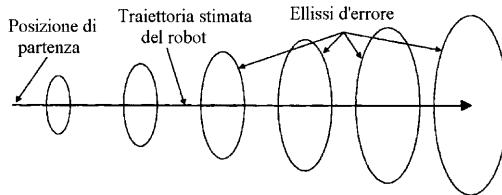


Figura 2.14 – Crescita dell'incertezza sulla posizione calcolata tramite l'odometria.

Tipicamente queste ellissi crescono con la distanza percorsa finché un metodo di posizionamento assoluto riduce l'incertezza e quindi azzera la dimensione dell'ellisse d'errore. Queste tecniche di stima dell'errore devono fare affidamento sui parametri derivanti dall'analisi delle prestazioni del veicolo. Chiaramente questi parametri considerano solo gli errori sistematici perché la grandezza degli errori non sistematici non può essere predetta.

Capitolo 3

AZIONAMENTO E CONTROLLO

3.1 Controllo del robot

Un robot è, in questo contesto, un sistema complesso che deve essere in grado di acquisire dati da sensori, muoversi nell'ambiente, eventualmente manipolare oggetti. Una realizzazione efficiente di queste funzioni può essere ottenuta con un'*architettura funzionale* concepita come sovrapposizione di vari *livelli di attività* organizzati secondo una struttura gerarchica (figura 3.1):

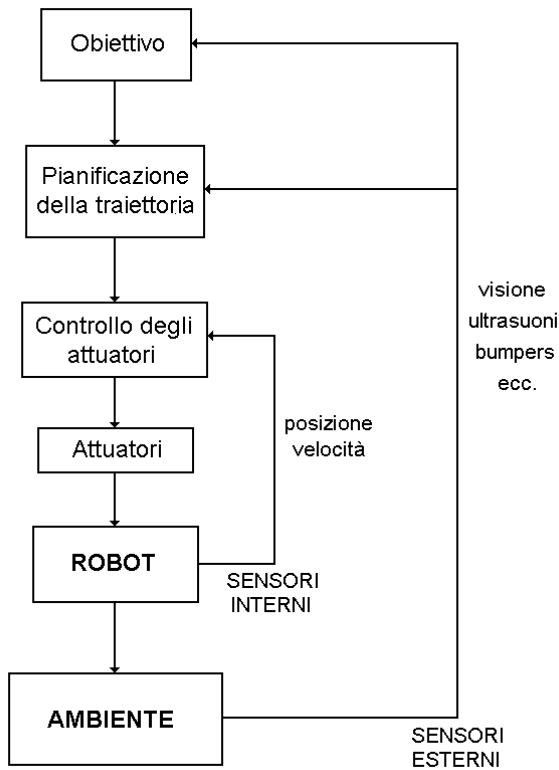


Figura 3.1 – Struttura funzionale del controllo di un robot.

I livelli inferiori della struttura sono orientati alla realizzazione del movimento a livello fisico, laddove i livelli superiori sono dedicati alla pianificazione delle azioni a livello logico. I livelli sono attraversati da flussi di informazione: quelli

orientati verso livelli superiori riguardano misure e risultati di azioni, quelli orientati verso livelli inferiori comportano la trasmissione di direttive di azione [Sciavicco e Siciliano, 1995].

Nel caso di un veicolo mobile possiamo suddividere il controllo in:

- **controllo di basso livello:** è il controllo degli attuatori di movimento; in genere si basa sul *modello dinamico* del robot e il suo scopo è quello di stabilizzare la velocità impostata sulle ruote soddisfacendo determinate specifiche statiche e dinamiche;
- **controllo di alto livello:** è il controllo della posizione e dell'orientazione del robot; si basa esclusivamente sul *modello cinematico*, in quanto riceve in ingresso una posizione e un'orientazione desiderate, e fornisce in uscita la velocità che deve essere impostata sulle due ruote (che costituisce l'ingresso per il controllo di basso livello);
- **pianificazione della traiettoria:** fornisce al controllo di alto livello un “set point” di posizioni e orientazioni del robot al fine di raggiungere un obiettivo immediato;
- **programmazione degli obiettivi:** determina una sequenza di obiettivi che il robot deve raggiungere per poter eseguire il proprio compito.

Dunque cosa vuol dire controllare un robot mobile ? La risposta dipende dal livello considerato; si possono dare diversi livelli di definizione:

- completare con successo un obiettivo;
- eseguire correttamente una traiettoria di moto;
- ridurre a zero l'errore di velocità.

Indipendentemente dal livello considerato, il controllo deve comunque presentare una *robustezza di comportamento in condizioni perturbate*: esso dovrà agire per ovviare, nel modo più idoneo, alla perturbazione verificatasi. Si ha quindi un concetto generalizzato di *controreazione*, che comporta un'adozione dello schema ad *anello chiuso* (e quindi un'introduzione di *feedback*), ad ogni livello della struttura gerarchica. Anche al livello superiore, quello più intelligente, vi sarà un feedback sensoriale (ad esempio tramite visione) ed una adattabilità alle variazioni dell'ambiente.

In questo lavoro ci siamo occupati della realizzazione della base mobile e del controllo dei servomotori elettrici, quindi del controllo di basso livello. Abbiamo realizzato una struttura hardware (paragrafo 4.3) ed un driver software in Linux (paragrafo 4.4) in grado di fornire una serie di primitive al controllo di alto livello. Esaminiamo dunque alcune modalità di controllo (di basso livello) solitamente impiegate nel controllo di velocità di motori elettrici.

3.1.1 Controllo dei servomotori

Il primo passo nella realizzazione di un sistema di controllo è quello di individuare, sul sistema da controllare, delle grandezze di ingresso modificabili in maniera indipendente e delle grandezze di uscita di cui si vuole indurre un opportuno comportamento.

In un motore CC a magneti permanenti le grandezze su cui si può andare ad agire sono la tensione e la corrente del circuito di armatura. Ma tensione e corrente di uno stesso avvolgimento sono correlate: il fatto che le grandezze di ingresso debbano essere indipendenti tra loro determina, in questo caso, la possibilità di utilizzare una sola grandezza, scelta fra le due disponibili. Poiché il controllo di traiettoria di un veicolo mobile in configurazione

“differential drive” si basa sul controllo di velocità delle due ruote, adottiamo come grandezza “di forzamento” la tensione applicata al circuito di armatura. Come noto, l’alimentazione in tensione comporta, all’interno del motore, un effetto intrinseco parzialmente stabilizzante, grazie alla presenza della tensione controelettromotrice che si oppone a variazioni della velocità angolare [Isidori, 1992].

Le variabili di uscita del sistema, quelle sottoposte a controllo, sono dunque le velocità angolari delle due ruote motrici. Ogni coppia resistente è vista come un disturbo su cui si deve operare la reiezione. Durante il funzionamento si deve esercitare comunque un monitoraggio su tensioni e correnti, affinché non superino i valori ammissibili, per evitare danni ai motori e per non mandare in saturazione i circuiti magnetici, eventualità che invaliderebbe il modello matematico che è stato ricavato al paragrafo 2.1. Tale monitoraggio verrà realizzato parte in hardware, parte in software (paragrafi 3.2, 4.3 e 4.4). Inoltre poiché il sistema di movimentazione è formato da due sottosistemi identici, sarà sufficiente analizzare le modalità per il controllo della velocità di un solo motore.

La struttura di principio del controllo a ciclo chiuso è mostrata nella figura 3.2. In ogni sistema di controllo a *controreazione* vi è sempre una grandezza di riferimento che rappresenta il *valore desiderato* che si vuole far assumere alla grandezza da controllare (nel nostro caso la velocità della ruota considerata). Questo valore desiderato viene confrontato con il *valore misurato* della grandezza controllata; l’errore risultante dal confronto viene posto in ingresso al controllore, il quale mediante un’opportuna legge di controllo, impone un’azione al sistema (tramite la grandezza di forzamento).



Figura 3.2 – Schema a blocchi di un sistema di controllo a contoreazione.

L'obiettivo del controllo di basso livello è quindi determinare la legge di controllo ed implementarla via hardware o via software. Esiste una vasta letteratura sui metodi di sintesi di controllori ad anello chiuso: stessi sistemi possono essere controllati in modi più o meno sofisticati, in funzione delle specifiche richieste. Tra queste dobbiamo citare [Isidori, 1992]:

- **la stabilità:** è una condizione indispensabile per il buon funzionamento del sistema controllato;
- **la reiezione ai disturbi:** è necessaria per far fronte a “piccoli” disturbi imprevisti;
- **il comportamento a regime permanente:** normalmente l’errore a regime dovrà essere nullo, ma vi sono situazioni in cui un piccolo errore può essere accettabile;
- **la robustezza:** riguarda l’insensibilità del sistema controllato a variazioni parametriche di una certa entità, significative incertezze di modellizzazione e “grandi” disturbi;
- **le prestazioni dinamiche:** riguardano il comportamento del sistema controllato durante i transitori.

Si osservi che l'azione di controllo si basa sempre su un modello del sistema: l'affidabilità del controllo dipende anche da quanto è accurato il modello. Perciò schemi di controllo più sofisticati impongono una modellizzazione più accurata e complessa del processo da controllare [De Carli, 1998/a].

Esistono anche altre metodiche, quale la *logica fuzzy*, con cui è possibile realizzare sistemi di controllo senza possedere un modello matematico del sistema da controllare, ma partendo da un insieme di regole ottenute dalla osservazione del comportamento del sistema. Queste metodiche tendono ad imitare il comportamento di un operatore umano esperto [De Carli, 1998/b].

Nel nostro caso il modello dinamico complessivo del sistema è noto (paragrafo 2.1); minime variazioni dei parametri possono essere considerate come "piccoli" disturbi. Possiamo quindi adottare alcuni schemi classici di controllo, le cui prestazioni sono caratterizzabili prendendo in esame l'andamento della risposta a gradino [Isidori, 1992]:

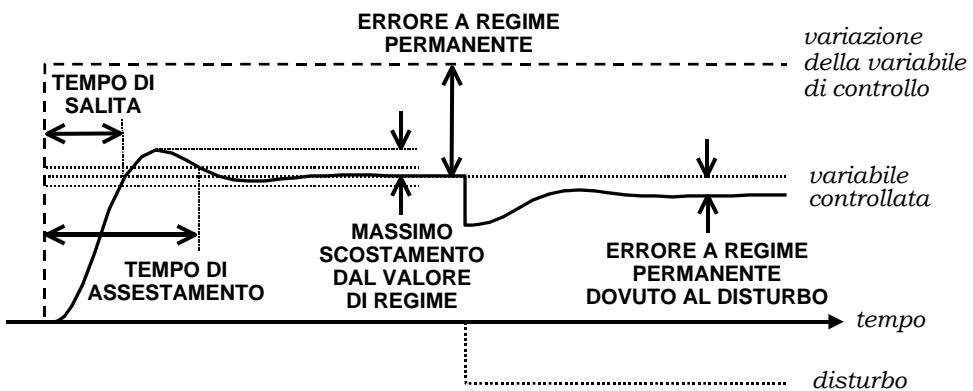


Figura 3.3 – Parametri caratteristici della risposta a gradino.

Oltre i parametri illustrati prenderemo in considerazione la *sovraelongazione* (*overshoot*), definita come il rapporto tra il massimo scostamento in eccesso

della risposta dal valore di regime e il valore di regime stesso. Vediamo ora alcune semplici metodologie di controllo.

CONTROLLO PROPORZIONALE (P)

È l'azione di controllo più semplice: la grandezza di forzamento è ottenuta moltiplicando l'errore per una costante di guadagno K_p :

$$u(s) = K_p [\dot{\theta}_D(s) - \dot{\theta}_M(s)]$$

ove $\dot{\theta}_D(s)$ è la velocità angolare desiderata e $\dot{\theta}_M(s)$ quella misurata (entrambe espresse nel dominio di Laplace). Idealmente, più è elevato il valore del guadagno K_p più velocemente l'errore viene corretto; sfortunatamente questa semplicità presenta due inconvenienti:

- 1) all'aumentare di K_p il sistema meccanico risponde con un'oscillazione smorzata sempre più ampia, senza che migliori il tempo di assestamento;
- 2) l'errore a regime permanente non può essere completamente eliminato, in quanto l'azione di controllo esiste proprio in virtù di questo errore, non essendoci fattori di integrazione nel sistema [Isidori, 1992].

La figura 3.4 mostra l'azione proporzionale su un “modello didattico” di motore con controllo in tensione di armatura. Per piccoli guadagni ($K_p=10$) il motore raggiunge la velocità di regime abbastanza lentamente e l'errore a regime permanente è elevato. Incrementando il guadagno ($K_p=30$) diminuiscono sia il tempo di salita, sia l'errore. Per valori più elevati ($K_p=100$) il motore parte sempre più velocemente, ma si determina anche un'oscillazione attorno al valore di regime. Aumentando ancora il guadagno, si determina una maggiore sovraelongazione, senza apprezzabili riduzioni del tempo di assestamento.

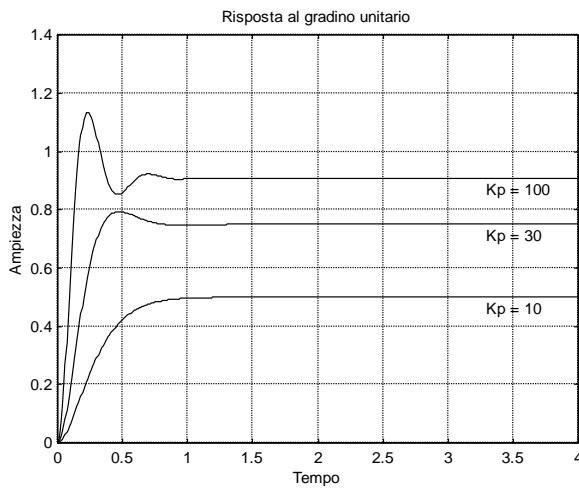


Figura 3.4 – Risposta del sistema con azione di controllo proporzionale.

CONTROLLO PROPORZIONALE-DERIVATIVO (PD)

Un'azione derivativa nel controllore riduce la sovraelongazione e il tempo di assestamento, mentre non comporta effetti apprezzabili sul tempo di salita e sull'errore a regime permanente. La legge di controllo è espressa da:

$$u(s) = [K_p + K_d \cdot s] \cdot [\dot{\theta}_D(s) - \dot{\theta}_M(s)]$$

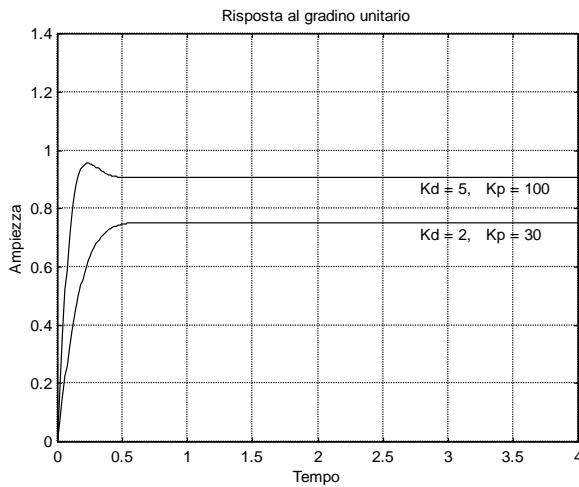


Figura 3.5 – Risposta del sistema con azione di controllo proporzionale-derivativa.

La figura 3.5 mostra l'effetto derivativo (si confronti con la figura 3.4). Il comportamento transitorio è migliorato ma, come abbiamo già detto, l'errore a regime permanente è rimasto invariato.

CONTROLLO PROPORZIONALE-INTEGRALE (PI)

Per eliminare l'errore a regime permanente occorre un'azione integrale, in grado di mantenere l'effetto dell'azione di controllo anche quando l'errore tende a zero. La legge di controllo diventa quindi:

$$u(s) = [K_p + \frac{K_I}{s}] \cdot [\dot{\theta}_D(s) - \dot{\theta}_M(s)]$$

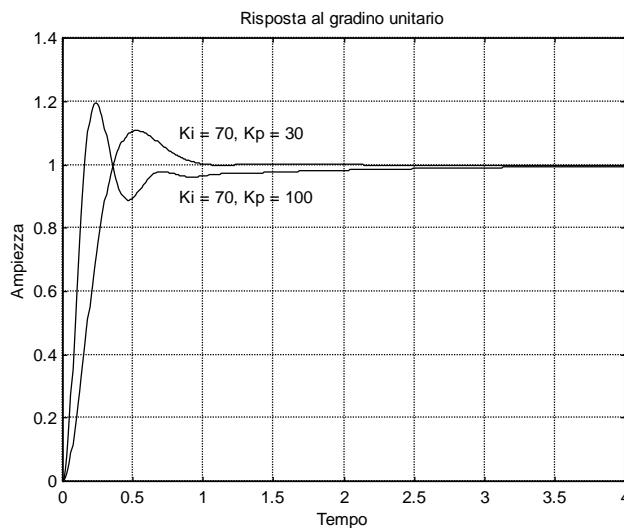


Figura 3.6 - Risposta del sistema con azione di controllo proporzionale-integrale.

Come si può osservare dalla figura 3.6, l'errore a regime è stato eliminato, ma il sistema presenta ancora una discreta sovraelongazione. Per ridurla si può diminuire il guadagno K_p , ma questo comporta un aumento del tempo di salita. Infatti, diminuendo K_p , diventa più sensibile l'azione dell'integratore, il quale comportandosi anche come filtro “passa basso” riduce le componenti di alta

frequenza e rallenta la dinamica. Si noti inoltre che l'azione proporzionale non può essere eliminata: infatti diminuendo troppo K_p la presenza di un'azione integrale fortemente dominante introduce instabilità nel sistema [Isidori, 1992].

CONTROLLO PROPORZIONALE-INTEGRALE-DERIVATIVO (PID)

Combinando l'effetto delle tre azioni, quella proporzionale, necessaria per migliorare il tempo di salita, quella derivativa, utile per diminuire la sovraelongazione, e quella integrale, indispensabile per eliminare l'errore a regime permanente, si possono ottimizzare globalmente le prestazioni statiche e dinamiche del sistema. Si ottiene il *controllore PID*, la cui legge di controllo è espressa da:

$$u(s) = [K_p + \frac{K_I}{s} + K_d \cdot s] \cdot [\dot{\theta}_D(s) - \dot{\theta}_M(s)]$$

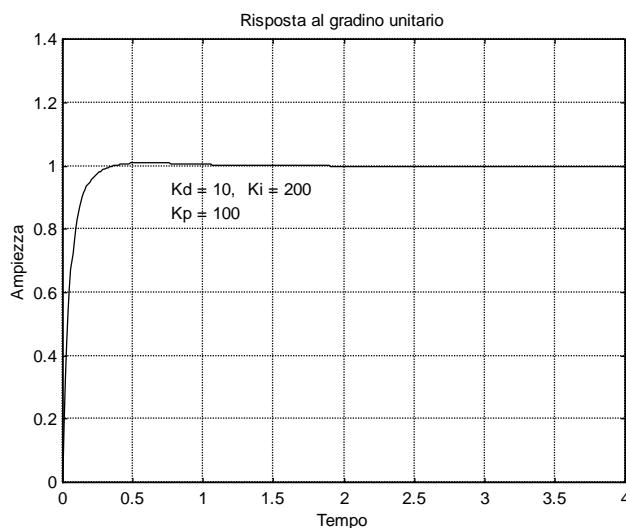


Figura 3.7 - Risposta del sistema con azione di controllo PID.

Come visibile nella figura 3.7 l'effetto è notevole: il sistema non presenta più l'errore a regime permanente, la sovraelongazione è quasi scomparsa, il tempo di salita e quello di assestamento sono diminuiti.

Il controllo PID è quindi il migliore tra quelli visti ed è molto utilizzato nella pratica; comunque, non sempre sono necessarie tutte e tre le azioni, a volte basta soltanto un controllore PI oppure PD. Un buon criterio di scelta è che la legge di controllo sia la più semplice possibile, compatibilmente con le specifiche richieste.

3.1.2 Considerazioni sul controllo PID

Una metodologia di tipo PID permette di ottenere un buon comportamento per il sistema controllato, specialmente quando al sistema di movimentazione non è richiesta una dinamica rapida [De Carli, 1998/a]. La struttura di controllo è semplice da implementare, sia tramite hardware analogico [SGS, 1995/a], sia mediante algoritmi software (paragrafo 4.4), e facile da mettere a punto. Tuttavia, nel controllo di un sistema reale, lo schema classico del PID presenta alcuni difetti:

- 1) l'azione integrale dovrebbe essere attivata solo quando il sistema si è portato vicino all'equilibrio, cioè quando il valore dell'errore diventa talmente piccolo da rendere inefficace l'azione proporzionale; intervenire con l'azione integrale quando l'errore in uscita è molto grande equivale a favorire il raggiungimento della condizione di saturazione dell'attuatore;
- 2) né l'azione integrale, né l'uscita complessiva del controllore possono assumere un valore arbitrariamente grande per via dei limiti fisici dei componenti utilizzati: occorre prevedere un effetto di saturazione;
- 3) l'azione derivativa illustrata (derivata matematica) non è fisicamente realizzabile; in realtà, occorre un'azione derivativa solo entro la banda di frequenza che interessa la dinamica del sistema, poiché nella realtà i

segnali sono sempre affetti da rumore in alta frequenza, che verrebbe amplificato se sottoposto ad una derivata matematica.

Queste considerazioni portano ad introdurre nella struttura classica del PID alcune modifiche. Lo schema funzionale del controllore diventa:

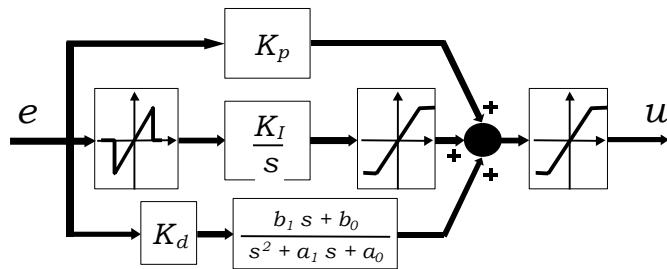


Figura 3.8 – Versione “modificata” del controllore PID.

ove e rappresenta l'errore ed u la grandezza di forzamento. In questo schema si notano le non-linearità sull'azione integrale e sull'uscita (la prima non-linearietà blocca l'integrazione per valori dell'errore superiori ad una certa soglia, le altre due rappresentano la saturazione a cui sono soggetti i componenti) e l'azione derivativa “in banda”, ottenuta con un filtro del secondo ordine.

Un altro problema di fondamentale importanza è costituito dall'effetto *windup* (colpo di vento). Supponiamo che l'attuatore venga portato in saturazione e che non sia quindi in grado di rispondere adeguatamente all'azione di forzamento. L'errore persiste più a lungo del normale e viene continuamente accumulato dall'integratore, che dà quindi luogo ad un'azione integrale eccessiva. Tale azione non ha effetto finché l'attuatore resta in saturazione ma, non appena esso torna in zona lineare, l'azione accumulata si scarica sull'attuatore, forzandolo ad un valore maggiore di quello che si avrebbe se il sistema fosse rimasto in zona lineare. Perciò, se l'uscita ha nel frattempo raggiunto il valore di regime,

l’azione accumulatasi forza l’uscita oltre tale valore: l’effetto finale è un eccessivo carattere oscillatorio nella risposta del sistema. La situazione può essere tale che il sistema potrebbe non raggiungere mai la condizione di equilibrio, ma oscillare permanentemente attorno ad essa.

Per evitare questo fenomeno occorre una azione *anti-windup*, che blocchi l’integrazione quando l’attuatore è in saturazione. Ci sono moltissimi schemi anti-windup: alcuni effettuano una compensazione dell’effetto windup tramite un modello matematico dell’attuatore, altri bloccano semplicemente l’azione integrale quando l’errore supera una certa soglia [De Carli, 1998/c]. Quindi la versione “modificata” del controllore PID presenta anche un positivo effetto anti-windup.

Ultimo problema da affrontare è la messa a punto dei parametri del PID, che solitamente risulta un processo facile, ma lungo e noioso. Per la sintesi dei guadagni K_I , K_p e K_d vi sono sostanzialmente due strade:

- metodi sistematici;
- metodi empirici.

I metodi sistematici prevedono un’analisi accurata del sistema e una sintesi dei guadagni tramite una procedura di calcolo per l’assegnazione degli autovalori del sistema a ciclo chiuso, in modo da ottenere un sistema stabile che soddisfi le specifiche preassegnate. I metodi empirici si basano invece su procedure di test eseguite sul sistema da controllare: dalla risposta del sistema si estraggono determinati parametri, dai quali si ricavano i coefficienti di guadagno del PID [De Carli, 1998/c].

METODO DI ZIEGLER-NICHOLS (CLOSED LOOP)

Questo metodo fornisce una procedura generale di messa a punto dei parametri dei vari controllori visti (P, PI, PD e PID). Le fasi da seguire sono:

- 1) si realizza il sistema a ciclo chiuso, collegando il controllore PID (od altro controllore) all'attuatore e quest'ultimo al sistema da controllare;
- 2) si pone $K_I = K_d = 0$, in modo tale che sia dominante la sola azione proporzionale, e si aumenta a piccoli passi il guadagno K_p attendendo ogni volta la condizione di regime permanente;
- 3) si continua ad aumentare K_p fino a che il sistema arriva al limite della stabilità, presentando un'oscillazione permanente per la presenza della saturazione nell'attuatore (*situazione di ciclo limite*);
- 4) si considera il valore di K_p raggiunto (*guadagno critico* K_c) e il periodo dell'oscillazione (*oscillazione critica* T_c);
- 5) usando questi due valori si ricavano i parametri K_p , T_I e T_d dalla tabella seguente, sulla base del controllore desiderato:

Controllore	K_p	T_I	T_d
P	$0.5 K_c$		
PI	$0.45 K_c$	$0.8 T_c$	
PD	$0.5 K_c$		$0.3 T_c$
PID	$0.55 K_c$	$0.5 T_c$	$0.12 T_c$

- 6) si ottengono gli altri due parametri del PID dalle formule:

$$K_I = \frac{K_p}{T_I} \quad K_d = K_p \cdot T_d$$

(nel caso di controllore a tempo discreto i parametri dovranno essere modificati per tenere conto del tempo di campionamento: vedere il paragrafo 3.1.3).

- 7) si controllano i parametri ottenuti sollecitando il sistema a ciclo chiuso con un ingresso a gradino: si deve ottenere una sovraelongazione abbastanza bassa (dell'ordine del 10-15%) e un tempo di risposta il migliore possibile; qualora non si desideri alcuna sovraelongazione è sufficiente mantenere i valori dell'azione integrale e derivativa ed abbassare l'azione proporzionale fino ad ottenere una risposta che non superi mai il valore di regime permanente.

Questo approccio è semplice, ma presenta un difetto: non sempre è possibile portare un sistema in oscillazione di ciclo limite, alcuni sistemi infatti potrebbero subire danni; altre volte la risposta del sistema è talmente lenta che occorrerebbe attendere molto tempo tra una prova e l'altra e questo rende il metodo inutilizzabile.

In questi casi esistono altre metodologie, sempre di tipo empirico, per estrarre i parametri del controllore dalla risposta a gradino del sistema a ciclo aperto. Per maggiori dettagli si veda [De Carli, 1998/c].

3.1.3 Controllore PID a tempo discreto

Consideriamo la legge di controllo PID nel dominio del tempo:

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Per realizzare il controllore digitale dobbiamo considerare che la precedente espressione verrà calcolata solo negli istanti kT_c , ove T_c è il periodo di campionamento. Approssimiamo l'integrale con una sommatoria trapezoidale e la derivata con una differenza tra il valore nell'istante di campionamento considerato e il valore nell'istante di campionamento precedente:

$$\left[\int_0^t e(\tau) d\tau \right]_{t=kT_c} \approx T_c \sum_{h=1}^k \frac{e(hT_c) + e((h-1)T_c)}{2}$$

$$\left[\frac{de(t)}{dt} \right]_{t=kT_c} \approx \frac{e(kT_c) - e((k-1)T_c)}{T_c}$$

Sostituendo nell'espressione del PID si ottiene:

$$u(kT_c) = K_p e(kT_c) + K_I T_c \sum_{h=1}^k \frac{e(hT_c) + e((h-1)T_c)}{2} + K_d \frac{e(kT_c) - e((k-1)T_c)}{T_c}$$

Se consideriamo la sequenza dei valori campionati anziché i valori negli istanti di campionamento, la precedente si può riscrivere nel modo seguente [Oppenheim *et al.*, 1998]:

$$u[k] = K_p e[k] + K_I T_c \sum_{h=1}^k \frac{e[h] + e[h-1]}{2} + K_d \frac{e[k] - e[k-1]}{T_c}$$

ove $e[0]=0$. Questa espressione permette già di ottenere un algoritmo numerico che implementa il controllore PID. Ma andiamo avanti e consideriamo il valore della legge di controllo calcolata in $k-1$:

$$u[k-1] = K_p e[k-1] + K_I T_c \sum_{h=0}^{k-1} \frac{e[h] + e[h-1]}{2} + K_d \frac{e[k-1] - e[k-2]}{T_c}$$

ove $e[-1]=e[0]=0$. Sottraendo membro a membro quest'ultima espressione dalla legge di controllo calcolata in k si ottiene:

$$\begin{aligned} u[k] &= u[k-1] + K_p (e[k] - e[k-1]) + K_I T_c \frac{e[k] + e[k-1]}{2} + \\ &+ K_d \frac{e[k] - 2e[k-1] + e[k-2]}{T_c} \end{aligned}$$

Questa equazione alle differenze permette di ottenere l'azione del controllo all'istante kT_c come “correzione” dell'azione avvenuta nell'istante di campionamento precedente. Essa non utilizza una accumulazione dell'errore (azione integrale) e non è quindi soggetta al problema del windup.

Dovremo comunque considerare un effetto di saturazione dovuto alla rappresentazione numerica adottata, in quanto l'algoritmo sarà realizzato in virgola fissa, cioè con aritmetica intera.

Anche nei controllori digitali si presenta il problema della rumorosità della derivata, poiché i dati sono sempre affetti da rumore introdotto dai sensori di misura (nel nostro caso tale rumore è principalmente dovuto alle tolleranze dei componenti meccanici presenti negli encoder). Inoltre a causa della forte quantizzazione dei valori numerici, determinata dall'impiego di aritmetica intera, la derivata calcolata come differenza di due termini comporta un sensibile rumore aggiuntivo, specialmente quando i dati variano lentamente.

Un modo pratico per rendere meno sensibile la derivata al rumore è quello di far seguire all'operazione di derivazione un operatore di media su N valori [MOTOROLA, 1996]. Comunque questo filtraggio introduce un ritardo di fase proporzionale ad N che contrasta con l'utile anticipo di fase generato dall'operazione di differenziazione. Per bilanciare questi due vincoli, si pone $N = 2$: questo comporta comunque una risoluzione doppia nei valori utilizzati per calcolare l'operazione di derivazione.

Dunque un'approssimazione meno rumorosa della derivata è data da:

$$\left[\frac{de(t)}{dt} \right]_{t=kT_c} \approx \frac{1}{2} \left(\frac{e[k] - e[k-1]}{T_c} + \frac{e[k-1] - e[k-2]}{T_c} \right) = \frac{e[k] - e[k-2]}{2T_c}$$

Ripetendo i passaggi visti prima, si ottiene la nuova equazione alle differenze:

$$\begin{aligned} u[k] = & u[k-1] + K_p (e[k] - e[k-1]) + K_I T_c \frac{e[k] + e[k-1]}{2} + \\ & + K_d \frac{e[k] - e[k-1] - e[k-2] + e[k-3]}{2T_c} \end{aligned}$$

Questa è la legge di controllo che andremo ad implementare nel nostro controllore. Un'altra possibile raffinatezza è quella di calcolare la derivata non sull'errore, ma sulla grandezza controllata: in tal caso la legge di controllo esibisce un comportamento migliore quando in ingresso al sistema vi sono brusche variazioni della grandezza di riferimento [De Carli, 1998/c].

SCELTA DELL'INTERVALLO DI CAMPIONAMENTO

Un aspetto importante del controllo digitale è la determinazione del periodo di campionamento. Per un controllore che tenta di emulare un sistema a tempo continuo la scelta è semplice: scegliere T_c come il più piccolo possibile. Infatti, a causa delle approssimazioni effettuate nel ricavare l'equazione alle differenze, intervalli di campionamento più piccoli causano una minore distorsione delle proprietà del controllore a tempo continuo.

Comunque un campionamento troppo veloce comporta uno spreco di risorse hardware e una maggiore introduzione di rumori di alta frequenza nell'anello di controllo, a causa dell'operazione di derivazione. D'altro canto se il tempo di campionamento è troppo basso si può verificare il fenomeno dell'*aliasing*, anch'esso dannoso [Oppenheim *et al.*, 1998].

Dunque il tempo di campionamento deve riflettere le proprietà dinamiche del processo: una pratica comune è quella di considerare l'intervallo di campionamento minore di $1/(30*B_p)$, dove B_p è la banda passante a ciclo chiuso del sistema a tempo continuo.

3.2 Amplificatori di potenza

La grandezza elettrica in uscita dal sistema di controllo (valore di tensione per un controllo in tensione d'armatura, valore di corrente per un controllo in corrente d'armatura) non può essere applicata direttamente al motore, ma richiede un *amplificatore di potenza* [Sciavicco e Siciliano, 1995]. Questo dispositivo ha la funzione di modulare, sotto l'azione di un opportuno segnale di controllo, il flusso di potenza che riceve dalla sorgente di alimentazione e di inviarlo al servomotore, per realizzare il movimento desiderato. La potenza assorbita viene in parte ceduta all'attuatore e in parte convertita in calore per effetti dissipativi: se indichiamo con P_a la potenza elettrica media fornita dalla sorgente di alimentazione e con P_e quella fornita al motore, possiamo definire il rendimento dell'amplificatore come:

$$\eta_{AMP} = \frac{P_e}{P_a}$$

Per un amplificatore di potenza, oltre al rendimento si dovrebbero considerare la linearità della risposta e la banda passante. Queste due ultime caratteristiche assumono un'importanza fondamentale solo per controlli di posizione accurati ed azionamenti che devono presentare una dinamica rapida [De Carli, 1998/a]. Nella realizzazione di un robot mobile essi assumono scarsa importanza, mentre risulta fondamentale il rendimento dell'amplificatore, poiché la sorgente di alimentazione di un veicolo elettrico è costituita da accumulatori: è di vitale importanza ridurre il più possibile l'energia dissipata per effetto Joule, ed è proprio nei sottosistemi di maggior potenza (cioè nel controllo dei motori) che è possibile intervenire in tal senso.

Vediamo dunque i principali tipi di amplificatori di potenza.

3.2.1 Amplificatori lineari

L'amplificatore di potenza può essere realizzato con tecniche di tipo lineare (*classe A, B, AB*) o a commutazione (*classe D*). Nel primo caso i transistor di potenza (*BJT* o *MOSFET*) lavorano in zona lineare [Millman e Halkias, 1978]. Questo causa una discreta perdita di potenza per effetto Joule e quindi uno scarso rendimento dell'amplificatore (dell'ordine del 25-30%) che rende poco pratico questo tipo di implementazione con correnti elevate. I vantaggi risiedono nell'assenza di emissione di onde elettromagnetiche di disturbo (*EMI*) che affliggono invece i sistemi a commutazione. Inoltre il circuito può essere molto semplice, se si utilizzano dispositivi integrati operazionali di potenza [SGS, 1995/a]. Il controllo può essere in corrente oppure in tensione: nel primo caso (figura 3.9) l'amplificatore permette un controllo diretto della coppia del motore, poiché la grandezza di forzamento imposta all'attuatore è la corrente (paragrafo 2.1). Questo schema viene spesso adottato nel controllo di coppia o di posizione.

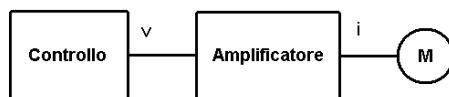


Figura 3.9 – Amplificatore lineare con controllo in corrente d'armatura.

L'altra possibilità è quella che prevede il controllo indiretto della velocità del motore variando la tensione applicata al circuito di armatura (figura 3.10). Esso risulta maggiormente adottato laddove sia necessario un controllo di velocità, per gli effetti intrinseci stabilizzanti di questo schema di controllo [Isidori, 1992].

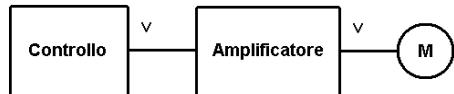


Figura 3.10 – Amplificatore lineare con controllo in tensione d’armatura.

3.2.2 Amplificatori a commutazione

Quando occorrono correnti significative si preferisce utilizzare una struttura a commutazione, unita ad una metodologia di tipo *PWM* (*Pulse Width Modulation*). In pratica, un’onda rettangolare di frequenza costante modulata in durata permette di variare la grandezza di forzamento, mantenendo al minimo le perdite. In questo caso, infatti, i dispositivi di potenza lavorano come interruttori, cioè sempre all’interdizione o alla saturazione. La potenza dissipata è minima: all’interdizione la corrente è zero, mentre alla saturazione la tensione ai capi di ciascun interruttore è molto piccola, tra 0.2 e 0.5 volt per i transistor BJT, 0.1 volt o meno per i MOSFET. Il calo di rendimento è dato esclusivamente dalla fase di transizione tra i due stati, che non si riesce a rendere istantanea a causa delle capacità parassite presenti nei transistor [Marcellini, 1985].

Negli ultimi anni il progresso tecnologico ha prodotto dispositivi a stato solido con tensioni di saturazione sempre più piccole e frequenze di commutazione sempre più elevate. Questo sta rendendo gli amplificatori in classe D molto diffusi, anche in applicazioni tipicamente di predominio degli amplificatori lineari, quali ad esempio sistemi audio di alta qualità [3W, 3].

Si osservi che l’utilizzo di una metodica PWM è possibile solo in presenza di carichi con capacità di integrazione: ad esempio in tutti i sistemi che presentano un effetto induttivo. Questo perché l’amplificatore manda in uscita un segnale a frequenza costante (tra i 20 KHz ed 1 MHz a seconda dell’applicazione) in cui

viene modulato il ciclo utile (*duty cycle*), ossia il rapporto tra la durata del periodo di saturazione dei transistori e il periodo del segnale. Questo segnale deve essere poi integrato dal carico: l'integrazione filtra le componenti di alta frequenza e lascia un valore medio continuo proporzionale al duty cycle dell'onda PWM:

$$V_M = \frac{t_{on}}{T} V_{MAX} = \delta V_{MAX}$$

ove δ rappresenta il duty cycle precedentemente introdotto (figura 3.11):

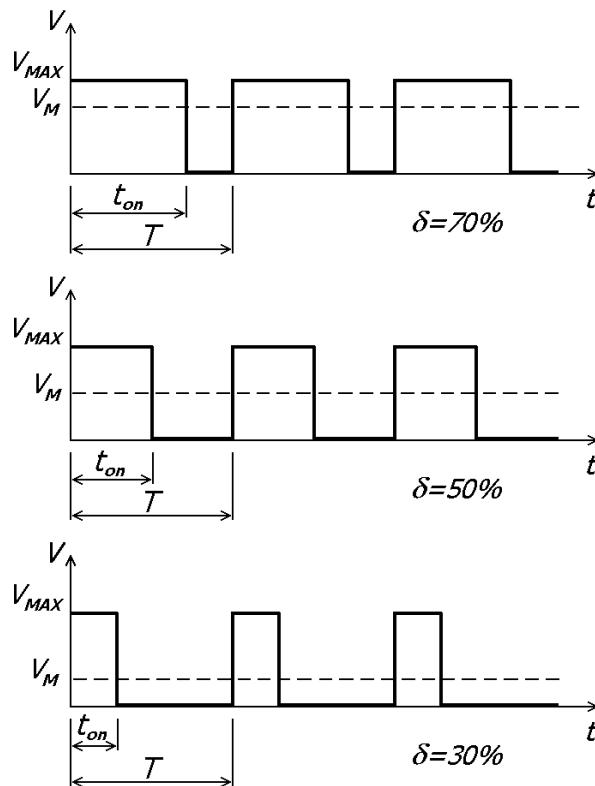


Figura 3.11 – Segnali PWM con diverso duty cycle.

Nel caso del motore questo “filtraggio” è dovuto sia alla presenza dell’induttanza nel circuito d’armatura, sia al fatto che la costante di tempo

meccanica è nettamente superiore al periodo del segnale applicato. La frequenza di commutazione del segnale PWM deve essere infatti scelta opportunamente in fase di progettazione, non solo per garantire la condizione precedente, ma anche per non rendere troppo elevate le perdite nei circuiti magnetici del motore (che aumentano all'aumentare della frequenza).

Qualora si voglia far ruotare il motore nelle due direzioni sono possibili due alternative per il pilotaggio PWM:

- **PWM a semionda.** In questa modalità l'amplificatore fornisce al carico un segnale PWM o tutto positivo o tutto negativo: una tensione di controllo nulla determina un'assenza del segnale PWM. Il circuito di controllo, tipicamente realizzato con componenti digitali, fornisce all'amplificatore il valore assoluto della tensione e un segnale di direzione (figura 3.12).

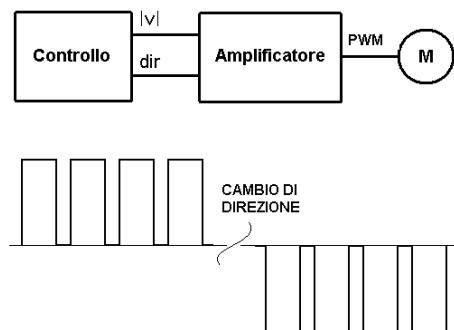


Figura 3.12 – Azionamento PWM a semionda.

- **PWM ad onda intera.** Qui l'amplificatore fornisce al carico un segnale PWM positivo e negativo: una tensione di controllo nulla determina un segnale PWM con un duty cycle del 50% (il cui valore medio è zero). Può essere facilmente realizzata sia con componenti analogici che digitali.

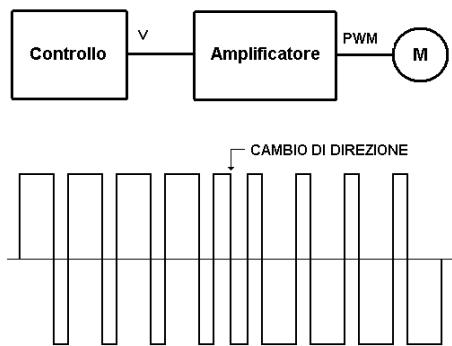


Figura 3.13 – Azionamento PWM ad onda intera.

Poiché nell’azionamento PWM digitale ad onda intera si utilizza l’aritmetica in complemento a due, la risoluzione risulta dimezzata rispetto a quella ottenuta con l’altra modalità, perché il bit più significativo del valore di tensione è utilizzato per stabilire la direzione. Questo tipo di azionamento è comunque preferito quando si vuole una risposta più pronta dal motore, essendo questo sempre percorso da una corrente, anche quando la velocità impostata è zero.

La topologia di base per l’amplificatore a commutazione è, a livello di principio, indipendente dalla classe di dispositivi di potenza utilizzati (BJT, MOSFET, IGBT): nel caso di moto bidirezionale, si utilizza una particolare struttura indicata come “*Ponte H*” (in inglese *H-Bridge*), costituita da quattro interruttori a stato solido idealmente disposti sui tratti verticali della lettera H, mentre il motore si trova sul tratto orizzontale [3W, 4]:

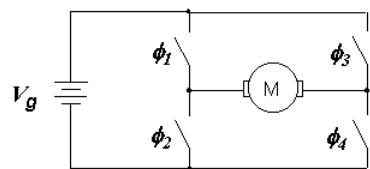


Figura 3.14 – Schema di principio del “Ponte H”.

Questa configurazione permette l'inversione di polarità utilizzando una singola alimentazione: chiudendo opportunamente gli *switch elettronici* si può imporre un determinato verso alla corrente nel circuito di armatura del motore:

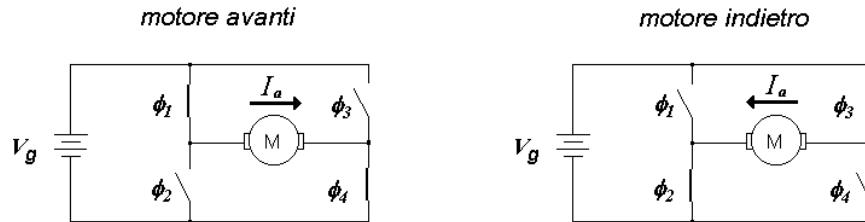


Figura 3.15 – La chiusura degli switch determina il verso della corrente nel motore.

Queste sono solo due delle molteplici possibilità offerte da questa struttura. Come vedremo tra breve, altre semplici configurazioni permettono la *rotazione libera* del motore e la sua *frenatura veloce*.

Sebbene il principio di funzionamento del “Ponte H” sia semplice, varie precauzioni devono essere prese. Innanzitutto gli istanti di apertura e chiusura degli switch elettronici devono essere ben determinati, al fine di evitare la conduzione contemporanea dei due transistor presenti nello stesso ramo, in quanto ciò causerebbe una corrente elevata tra alimentazione e massa che distruggerebbe il dispositivo [3W, 4]. Per evitare questo problema i circuiti logici attendono un certo tempo tra lo spegnimento di un transistor e l'accensione di un altro (*dead time*).

Inoltre è necessario imporre una limitazione della corrente, al fine di prevenire sovraccarichi, specialmente in fase di frenatura del motore ove, a causa dei

carichi inerziali, l'energia posseduta dal sistema meccanico viene ceduta al sistema elettrico (vedere i paragrafi successivi).

Infine occorre prevedere l'emissione di forti disturbi elettromagnetici associati alla commutazione: più tale commutazione è rapida, maggiori saranno le frequenze emesse. Poiché sono in gioco anche forti correnti, l'emissione può influenzare facilmente circuiti vicini. Vedremo nel paragrafo 4.3 un insieme di metodiche atte a prevenire e ridurre il più possibile queste interferenze [Tonielli, 2000].

3.2.3 Ponte H a BJT e a MOSFET

Per poter impiegare correttamente un “Ponte H” occorre comprendere i fenomeni transitori che avvengono in seguito all’apertura e alla chiusura degli switch elettronici. A tal scopo consideriamo un ponte a BJT, il cui schema è mostrato nella figura successiva:

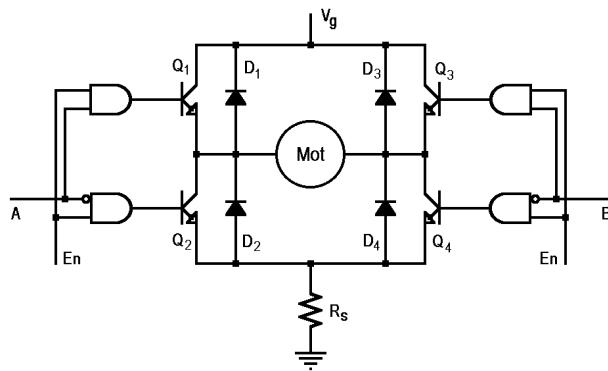


Figura 3.16 – Ponte H a transistor bipolar.

Le quattro porte AND schematizzano la sezione logica del ponte; i transistor, opportunamente pilotati, agiscono da interruttori, mentre quattro diodi veloci (detti *diodi di ricircolo* o *flyback diode*) sono utilizzati per recuperare l’energia

del carico induttivo. Il resistore R_S (*sense resistor*) viene usato per il monitoraggio della corrente [SGS, 1995/b].

Si supponga che il ponte sia inizialmente disabilitato (tutti i transistor all'interdizione), condizione ottenuta ponendo $En = L$, e che il motore sia fermo con la sua induttanza senza energia immagazzinata. In queste condizioni nessuna corrente circola nel motore.

Abilitando il ponte ($En = H$) ed imponendo i livelli logici $A = H$ e $B = L$, i transistor Q_1 e Q_4 vengono portati alla saturazione mentre Q_2 e Q_3 vengono mantenuti nello stato di interdizione. La corrente inizia a fluire dal generatore di tensione V_g , attraversando il transistor Q_1 , il motore, il transistor Q_4 ed il resistore R_S , per giungere infine a massa. La figura seguente riepiloga la situazione:

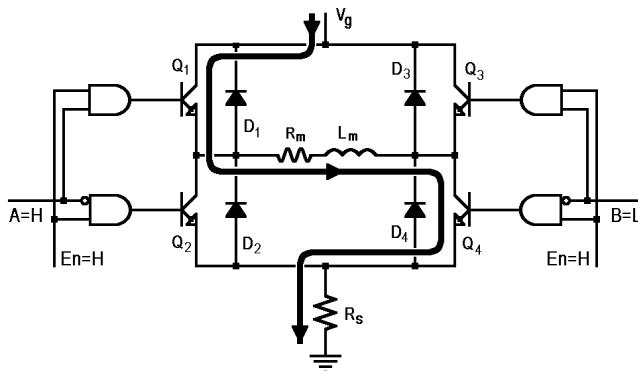


Figura 3.17 – La corrente inizia a fluire nel motore, al momento fermo.

Il valore della corrente sale rapidamente (in funzione del valore di V_g e della costante di tempo del circuito) e può essere monitorato tramite la tensione presente ai capi di R_S . Solitamente questo resistore ha valori compresi tra 0.01 e 0.5Ω (con bassa tolleranza) ed è del tipo “**non a filo**”, per evitare fenomeni indutti che falserebbero la misura della corrente.

Quando il motore inizia a ruotare, in serie all'induttanza L_m e alla resistenza dell'avvolgimento R_m appare una forza controelettromotrice $V_{bemf}(t) = k_E \dot{\theta}(t)$, che a regime tenderà a ridurre il valore della corrente fino al portarlo al valore minimo necessario per vincere le coppie resistenti. A causa degli attriti si avrà sempre $|V_{bemf}(t)| < V_g$.

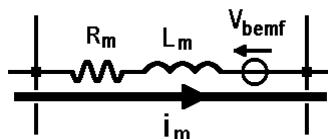


Figura 3.18 – La rotazione del motore provoca la comparsa di una forza controelettromotrice nel circuito di armatura.

È importante osservare che la variazione della corrente avviene seguendo l'evoluzione del transitorio elettrico, mentre la variazione della forza controelettromotrice avviene in funzione del transitorio meccanico del sistema, solitamente molto più lento del precedente (i due transitori sono confrontabili solo negli azionamenti a dinamica rapida, che presentano un accoppiamento diretto tra rotore e carico [De Carli, 1998/a]).

Consideriamo, come esempio, un certo istante in cui il motore sia in rotazione ad una velocità angolare pari a circa la metà di quella di regime: in tal caso, ipotizzando che il transitorio elettrico sia terminato, nel motore circolerà una corrente costante, pari a circa la metà di quella di stallo. In questa situazione, variando opportunamente i livelli logici applicati al ponte si può:

- invertire la polarità ai terminali del motore;
- disabilitare il ponte per ridurre la corrente;
- cortocircuitare il motore.

INVERSIONE DELLA POLARITÀ

Si è visto che la struttura “H-Bridge” permette di scambiare facilmente la polarità della tensione presente sui terminali del motore, invertendo la diagonale attiva: se la logica mantiene il ponte abilitato ($En = H$) e impone i livelli logici $A = L$ e $B = H$, i transistor Q_1 e Q_4 vengono portati all’interdizione, mentre Q_2 e Q_3 passano in saturazione.

A causa delle proprietà dei circuiti induttivi, la corrente che scorre nel motore non potrà variare istantaneamente, per cui essa continuerà a fluire all’interno dell’induttanza, nello stesso verso, per un certo transitorio seguendo però un diverso percorso nel ponte: partendo da massa e attraversando il resistore R_S , il diodo D_2 , il motore e il diodo D_3 , la corrente arriverà finalmente a V_g (che in questo caso non dovrà fornire, quanto piuttosto assorbire l’energia immagazzinata nell’induttanza). La situazione è illustrata dalla figura 3.19.

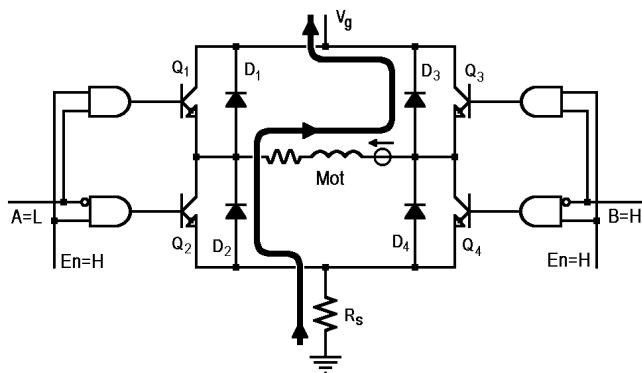


Figura 3.19 – Fase di scarica dell’induttanza.

La corrente tenderà però a diminuire rapidamente poiché ai capi dell’induttanza risulta applicata una tensione la cui polarità è tale da opporsi al flusso di corrente. Si osservi che, in questa fase, ai capi di R_S si ottiene una tensione negativa.

Una volta raggiunto il valore zero (ammesso che la diagonale attiva non cambi ancora) la corrente inizierà a scorrere in senso inverso: partendo da V_g , essa attraverserà il transistor Q_3 , il motore, il transistor Q_2 e il resistore R_S per giungere finalmente a massa. La figura 3.20 riepiloga la situazione:

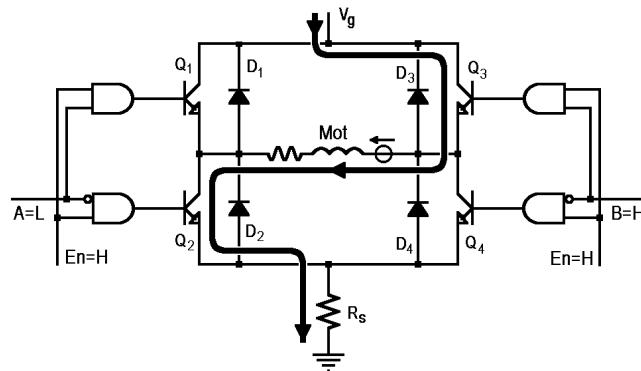


Figura 3.20 – La corrente inizia a fluire in senso opposto.

È importante osservare che la forza controelettromotrice V_{bemf} continua a presentarsi con la stessa polarità di prima, anche quando la corrente ha cambiato verso, poiché il sistema meccanico reagisce lentamente: il motore inizia a rallentare (per via del nuovo verso di corrente che comporta un coppia contraria al moto), ma impiegherà del tempo per invertire il senso di rotazione. In tutta la fase di frenatura del motore, la forza controelettromotrice è diretta nello stesso verso della tensione del generatore V_g ; essa contribuisce quindi a far circolare una corrente maggiore di quella di stallo (al limite una corrente quasi doppia!). È in questa situazione che si può verificare un fenomeno di overcurrent, pericoloso per il circuito ed il motore. Occorre tenere presente che la durata di questa fase dipende dal momento d'inerzia del carico e da eventuali fenomeni di attrito. La soluzione da adottare per eliminare il pericolo di overcurrent consiste nel disabilitare (per brevi intervalli di tempo) il ponte quando la corrente supera un valore prestabilito, mediante un circuito

comparatore. Si osservi che, in questa fase, ai capi di R_S si ottiene una tensione positiva.

Un altro fatto da considerare è che il generatore deve avere capacità di assorbire la corrente (sink), oltre che fornirla, altrimenti questa, fluendo in senso inverso, caricherà troppo i condensatori dell'alimentatore portando la tensione a livelli non tollerabili dai dispositivi elettronici (condizione di *overvoltage*). Per prevenire tale fenomeno è opportuno dotare V_g di una capacità adeguata [SGS, 1995/b].

Un pilotaggio PWM ad onda intera è ottenuto con una continua inversione di polarità: la tensione applicata al motore, alternativamente positiva e negativa, risulta pari a $\pm(V_g - 2V_{cesat})$, ove V_{cesat} è la tensione tra collettore ed emettitore nei transistor in saturazione. La frequenza di inversione viene scelta abbastanza alta (in genere tra i 20 e i 50 KHz, per evitare vibrazioni nella banda acustica) e comunque più elevata dell'inverso della costante di tempo del circuito. A causa di questa condizione la corrente non può cambiare verso nell'induttanza: per un po' il suo valore diminuisce, poi la successiva inversione del ponte determina un nuovo aumento del valore della corrente. Il risultato è che la corrente oscilla attorno ad un valore medio: il valore di corrente che si avrebbe in presenza di pilotaggio lineare con una tensione di armatura pari a δV_g (essendo δ il duty cycle del segnale). Quindi, ai fini del controllo, un driver PWM può essere visto come un driver lineare.

DISABILITAZIONE DEL PONTE

Quando supera i valori ammessi, la corrente deve essere ridotta rapidamente per evitare possibili danni ai dispositivi (motore e circuito di potenza); in tal caso la logica può agire disabilitando completamente il ponte, cioè ponendo $En = L$:

tutti i transistor vengono portati all'interdizione e la corrente continua a fluire nello stesso verso all'interno dell'induttanza, fino a completa scarica dell'energia immagazzinata (figura 3.19). Infatti, anche in questo caso all'induttanza L_m viene applicata una tensione la cui polarità è tale da opporsi all'attuale flusso di corrente, che tenderà quindi a diminuire.

Normalmente quando la corrente scende ad un valore sicuro il ponte viene nuovamente abilitato dalla logica di controllo. La persistenza della disabilitazione comporta l'annullamento totale della corrente: una volta raggiunto il valore zero la corrente cesserà (anche in presenza della forza controelettromotrice V_{bemf}) poiché tutti i diodi verranno a trovarsi polarizzati inversamente. Non si possono quindi verificare fenomeni di overcurrent. Però, anche in questa situazione, valgono le stesse considerazioni fatte circa le capacità di assorbimento (sink) dell'alimentatore.

Se il ponte resta disabilitato ed il carico è di tipo frizionale, il motore si arresterà rapidamente, viceversa con carico prevalentemente inerziale il motore resterà a lungo nello stato di moto (*free running*).

MOTORE IN CORTOCIRCUITO

Se il circuito logico varia i livelli in ingresso al ponte, ponendo $A = B = L$ e mantenendo il ponte abilitato ($En = H$), i transistor Q_2 e Q_4 vengono portati in saturazione, mentre Q_1 e Q_3 passano all'interdizione: il motore viene praticamente cortocircuitato e la corrente (che non può variare istantaneamente nell'induttanza del motore) continua a fluire nello stesso verso attraversando il transistor Q_4 , il diodo D_2 e il motore, come mostrato in figura 3.21:

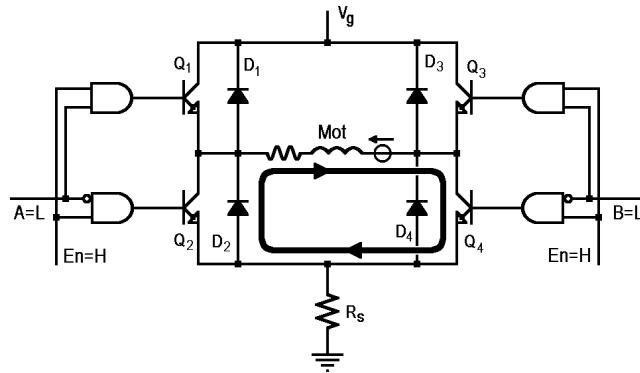


Figura 3.21 – Motore in cortocircuito.

Anche in questo caso all’induttanza L_m viene applicata una tensione la cui polarità è tale da opporsi al flusso di corrente, che tenderà quindi a diminuire e poi a cambiare segno, crescendo in senso opposto, essendo V_{bemf} l’unica forza elettromotrice presente. Questo causerà una coppia contraria al moto, che persistrà finché esiste la tensione V_{bemf} (cioè finché la velocità angolare del motore non arriva a zero), causando l’arresto veloce del motore (*fast brake*).

Quando il ponte si trova nella condizione della figura 3.21, la tensione ai capi del motore è quasi nulla (essendo praticamente cortocircuitato): alternando velocemente questa condizione a quella illustrata nella figura 3.17 (o a quella illustrata nella figura 3.20), si ottiene un pilotaggio PWM a semionda.

Una importante osservazione riguarda il fatto che durante la fase illustrata dalla figura 3.21 la corrente non è monitorabile perché non attraversa il resistore R_s . Quest’inconveniente può essere facilmente risolto modificando il circuito, come indicato nella figura 3.22: scollegando i diodi inferiori dal resistore R_s e collegandoli a massa si costringe la corrente di scarica a transitare su R_s generando una tensione positiva ai suoi capi.

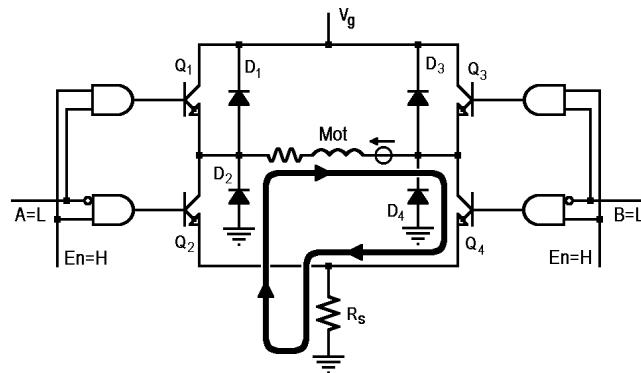


Figura 3.22 – Passaggio della corrente dopo la modifica al circuito.

In commercio sono disponibili molti “Ponti H” integrati, ma la maggior parte di essi lavora con basse correnti (in genere non superiori ai 4 ampere). Quando si richiedono correnti maggiori il circuito è realizzato a componenti discreti, oppure in configurazione ibrida. In tutti i casi gli interruttori elettronici possono essere BJT o MOSFET.

I MOSFET presentano alcuni vantaggi: sono più veloci, hanno una tensione di saturazione più bassa, permettono correnti più elevate a parità di potenza dissipata, alcuni non necessitano dei diodi di ricircolo perché già incorporati nel substrato. Ma il loro impiego comporta una maggiore difficoltà nella progettazione circuitale, per il diverso tipo di polarizzazione richiesta [Millman e Halkias, 1978].

Per questo sono nati dei circuiti integrati “pilota” che mettono a disposizione una struttura a “Ponte H” in cui mancano solo i MOSFET “finali” (utilizzeremo un integrato di questo tipo per il nostro circuito di potenza, vedere il paragrafo 4.3). Essi si occupano di tutti gli aspetti riguardanti la polarizzazione dei MOSFET, in modo tale che il progettista possa concentrarsi meglio su altri aspetti del circuito, quale la riduzione di emissioni elettromagnetiche. Infatti, a

causa della maggiore velocità di commutazione, la quantità di disturbi generata risulta di gran lunga superiore che con i BJT.

L'ultima considerazione riguarda la presenza dei diodi flyback nel substrato dei MOSFET: questo fatto rende impossibile realizzare la configurazione di figura 3.22, poiché i diodi non sono "accessibili". È quindi necessario un diverso approccio per la misura della corrente: a questo scopo alcuni dispositivi integrati forniscono un circuito "di sensing" interno; per quelli che non ne dispongono e per le soluzioni a componenti discreti è possibile utilizzare un circuito operazionale che, eseguendo la differenza tra le correnti nei due rami del ponte (misurate utilizzando due resistori separati), ricostruisce l'effettivo valore di corrente (figura 3.23). Sono possibili anche altre soluzioni, come descritto in [SGS, 1995/c].

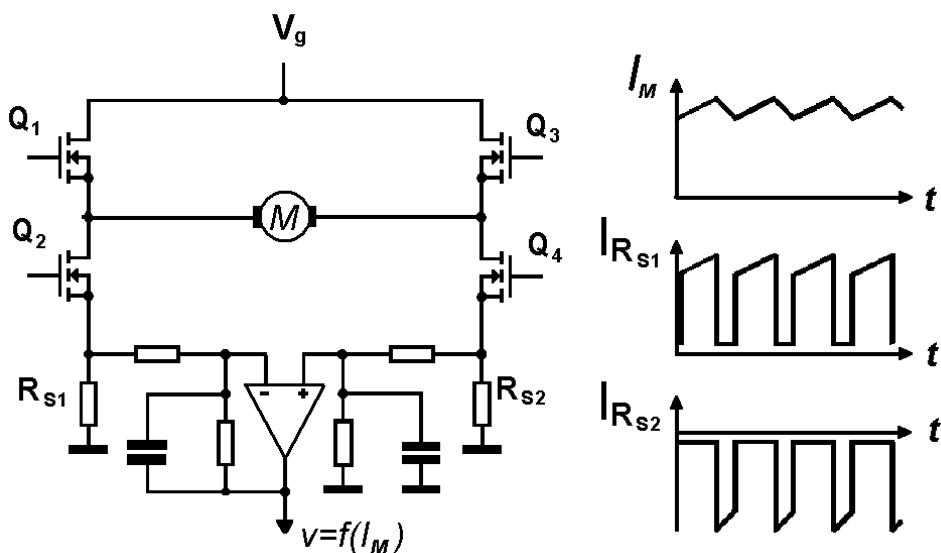


Figura 3.23 – Ricostruzione della corrente in un "Ponte H" a MOSFET.

Capitolo 4

PROGETTO E REALIZZAZIONE

4.1 Descrizione generale

Il robot mobile da noi realizzato è stato sviluppato nell'ambito delle necessità della squadra ART (Azzurra Robot Team), il team italiano di robot calciatori che partecipa alle competizioni internazionali della RoboCup (appendice A6).

ART è formata da robot calciatori con diverse caratteristiche hardware e software, ma completamente autonomi, cioè con tutti i sensori e le capacità di elaborazione a bordo. La piattaforma maggiormente utilizzata è costituita da:

- una base mobile PIONEER I o II, prodotta dalla ActivMedia Robotics, dotata di motori a 12V e sonar;
- un computer di bordo basato su una architettura convenzionale, con componenti di normale reperibilità (mother board ASUS, processore AMD, 64 Mbyte di RAM, HD 20GB);
- un alimentatore switching e una batteria a 12V;
- un sistema di visione formato da una telecamera a colori e una scheda di acquisizione (frame grabber);
- un collegamento di rete wireless a 2Mbit (wavelan).

Il sistema operativo utilizzato nei PC presenti a bordo dei robot è Linux, completo di driver e librerie per il frame grabber e la scheda wavelan, nonché programmi vari per il supporto e la programmazione.

Questa architettura nasce con l'obiettivo di realizzare un robot specializzato, in grado di competere nelle manifestazioni calcistiche organizzate dalla RoboCup, coordinando il proprio comportamento con i compagni di squadra.

Per il gioco sono stati aggiunti alla base mobile alcuni dispositivi di calcio (basati su elettrovalvole ed aria compressa) ed altri dispositivi per acquisire informazioni sull'ambiente di gioco.

Studiando le prestazioni di questa piattaforma si è visto, dopo un'attenta analisi delle caratteristiche dinamiche, che i suoi motori erano poco sfruttati. Si è quindi partiti con l'idea di riprogettare l'elettronica di movimentazione e la parte software del controllo al fine di migliorare le prestazioni delle basi mobili preesistenti. Il risultato è stato la realizzazione di un sistema hardware/software a basso costo per il controllo di due motori CC a magneti permanenti. Questo è servito anche come banco di prova, con l'intento di realizzare in futuro piattaforme più complesse.

Successivamente nel team è nata la necessità di disporre di un nuovo portiere. Sono stati così acquistati i motori sulla base delle prestazioni desiderate (paragrafo 4.1) e l'elettronica realizzata è stata nel frattempo “dirottata” per quest’altro scopo. Il robot è tuttora in costruzione: al momento esiste la sola base mobile, ma a breve sarà dotata di dispositivi di visione binoculare (stereovisione) e di dispositivi di calcio ad aria compressa.

L’obiettivo è quello di renderla operante per le prossime competizioni che si terranno a Seattle, negli Stati Uniti, i primi di agosto di quest’anno.

Tale base mobile è fondamentalmente costituita da un veicolo in configurazione “differential drive” (paragrafo 1.2). La base è rettangolare e le due ruote motrici sono poste al centro del veicolo; sui due lati vi sono due rotelle a sfera che funzionano da castor, evitando però i problemi legati ai movimenti di quest’ultimo.

La necessità di bilanciare il peso e rendere simmetrico il movimento (un robot portiere deve infatti muoversi parallelamente alla porta) ci ha costretto ad utilizzare un sistema (artigianale) di sospensioni per assicurare la necessaria continuità di contatto delle ruote motrici con il terreno.

Per quanto riguarda l'hardware progettato questo è stato tranquillamente in grado di soddisfare la rapidità nelle azioni di gioco che si richiedono ad un robot portiere. Tale hardware era stato sviluppato con un'ottica di impiego più generale, cioè non solo nel campo della robotica mobile, ma nella movimentazione in genere. La figura seguente mostra la struttura di principio dell'elettronica realizzata:

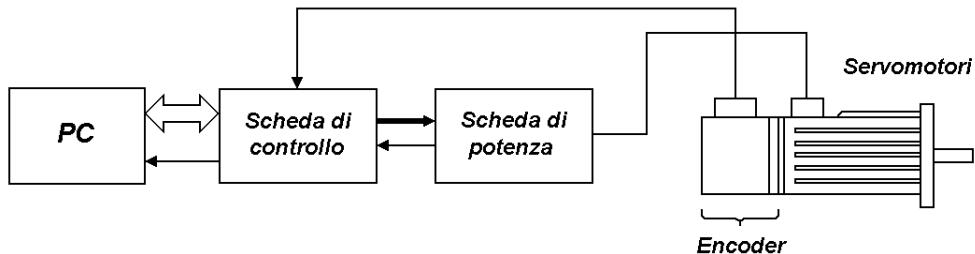


Figura 4.1 – Struttura di principio dell'elettronica realizzata.

Come in ogni robot della squadra, anche nella nostra base mobile è stato inserito un sistema di elaborazione basato su un PC standard. Parte del sistema di gestione dei motori è costituito da una scheda ISA di controllo, che si va ad innestare in uno slot del PC (paragrafo 4.3.1). Questa scheda ha il compito di ricevere e decodificare gli impulsi provenienti dagli encoder (paragrafo 1.4.1), e generare i segnali PWM (paragrafo 3.2) sulla base del valore della grandezza di forzamento determinata dal ciclo di controllo (paragrafo 3.1.3).

L'altra parte del sistema hardware realizzato è costituito da una scheda di potenza, incassata sotto ai motori (paragrafo 4.3.2). Questa ha il compito di traslare in potenza i segnali PWM provenienti dalla scheda ISA di controllo e di segnalare alla stessa eventuali “fault” dei motori, come le situazioni di “overcurrent”.

La chiusura dell’anello di controllo è affidata ad un processo realtime (paragrafo 4.4). La legge di controllo è basata su una metodologia PID, in cui la grandezza di forzamento è legata all’errore di velocità, tramite un fattore proporzionale, integrale e derivativo (paragrafo 3.1.3).

Per ottenere il rispetto dei vincoli temporali necessari al corretto svolgimento dell’anello di controllo abbiamo sostituito Linux, con la sua versione realtime RTLinux (paragrafo 4.4.2). Questo ci ha consentito di utilizzare il PC sia per operazioni di basso livello che per compiti di maggiore intelligenza.

La sezione di alimentazione di tutto il robot è stata realizzata con due batterie da 12V-7Ah, collegate in serie. Il PC viene alimentato mediante un convertitore CC/CC da 200W con ingresso a 24V ed uscita ATX (3.3V, 5V, -5V, 12V, -12V). La scheda di potenza può essere alimentata con tensioni comprese tra i 12V e i 24V: si utilizza quindi la tensione direttamente fornita dal pacco batterie. I motori sono a 24V. In realtà potrebbero essere impiegati anche motori a più bassa tensione perché la scheda di potenza dispone di un sistema di limitazione delle correnti massime circolanti nei motori.

Nel prossimo paragrafo vedremo i criteri che hanno guidato la scelta dei motori e degli accumulatori.

4.2 Scelta dei motori e degli accumulatori

Per acquistare il motore più adatto all'applicazione che si intende realizzare occorre stimare le caratteristiche di coppia e di velocità angolare necessarie per movimentare il carico desiderato e scegliere dai cataloghi il motore che meglio approssima queste caratteristiche.

Questo potrebbe essere fatto simulando il sistema mediante il modello ricavato al paragrafo 2.1. Tuttavia, prima della costruzione completa del robot, conoscere alla perfezione tutti i parametri potrebbe non essere possibile; si hanno infatti numerosi fattori di incertezza, quali gli attriti reali, il peso del veicolo (che potrebbe variare spesso, specialmente se il robot è un oggetto di studio e sperimentazione), l'eventuale carico trasportato, ecc. Conviene utilizzare un metodo approssimato e sovradimensionare la potenza dei motori.

Ricordiamo che un veicolo si muove grazie all'attrito secco presente tra le ruote e il terreno: il robot si muove nella direzione dell'unica forza esterna esercitata, cioè della forza d'attrito (appendice A7). Se trascuriamo i momenti d'inerzia e gli attriti delle parti in rotazione, ipotizziamo che il veicolo si muova su terreno pianeggiante e consideriamo che le ruote non devono subire slittamento, ma muoversi con un moto di rotolamento puro, la forza totale applicata alle ruote tramite le coppie fornite dai motori deve soddisfare la relazione:

$$F_{APP} = \frac{T}{r} \leq \mu_s Mg$$

ove T è la coppia motrice (complessiva) fornita dai motori alle ruote, M è la massa del robot, r il raggio delle ruote, g l'accelerazione di gravità e μ_s il

coefficiente di attrito statico tra le ruote e la superficie su cui il veicolo si muove. In linea di principio qualsiasi coppia applicata alle ruote dovrebbe generare il movimento; in realtà è necessario superare l'attrito interno dei supporti, della riduzione e della trasmissione. Dalla precedente si ottiene:

$$T_{Max} = \mu_s Mgr$$

Questo approccio fornisce quindi la massima coppia applicabile: aumentando la coppia oltre tale valore si potrebbe ottenere lo slittamento delle ruote non appena il veicolo comincia a muoversi. Questo è vero in teoria: nella realtà l'attrito secco dei meccanismi e l'attrito di rotolamento dissipano parte della potenza di uscita del motore prima che le ruote comincino a girare. Vi potrebbero essere poi altri fattori quali piccoli ostacoli che rendono il terreno non perfettamente pianeggiante. Infine occorre ricordare che l'efficienza massima di un motore si esplica ad un punto più basso, circa 1/3 della coppia massima (paragrafo 2.1.1).

Tutte queste considerazioni ci portano a sovradimensionare i motori, anche in considerazione del fatto che è sempre possibile rallentarli via software. Se il motore è sottodimensionato, il robot potrebbe muoversi molto lentamente o non muoversi affatto. Pertanto stimiamo una coppia massima desiderata per ciascun motore pari a:

$$T_{des} = \frac{3}{2} \mu_s Mgr$$

Questa coppia si intende applicata all'asse delle ruote. Nella scelta del motore dovremo considerare il rapporto di riduzione e l'efficienza della stessa (paragrafo 2.1).

Per quanto riguarda la velocità angolare massima, questa dipenderà dalla velocità lineare che si vuole imprimere al robot a regime e dal diametro delle ruote, secondo la relazione:

$$\omega_{Max} = \frac{v_{Max}}{r}$$

Anche questa andrà riportata all'asse motore tramite il rapporto di riduzione.

Per il nostro robot abbiamo scelto due servomotori PITTMAN GM9236S021 sulla base dei seguenti parametri:

- raggio delle ruote: *7.5 cm*;
- massa totale del robot: *20 Kg*;
- coefficiente di attrito statico: *0.5*;
- velocità massima: *1.5 m/s*.

Il coefficiente di attrito statico μ_s è stato stimato utilizzando un dinamometro ed una delle ruote, bloccata e caricata con metà peso.

Le principali caratteristiche dei motori sono:

- tensione massima di alimentazione: *24V*;
- fattore di riduzione: *19.7:1*, efficienza: *73%*;
- coppia continua all'uscita della riduzione: *1.081 Nm*,
- coppia massima (di stallo) all'uscita della riduzione: *6.074 Nm*;
- velocità angolare massima all'uscita della riduzione: *24.7 rad/s*;
- encoder incrementale calettato sull'asse del motore con risoluzione di 500 impulsi/giro, alimentazione a 5V ed uscita TTL.

Per le altre caratteristiche si veda il datasheet riportato in appendice A5 oppure il catalogo della PITTMAN [3W,5]. I due motori sono illustrati nella figura seguente:

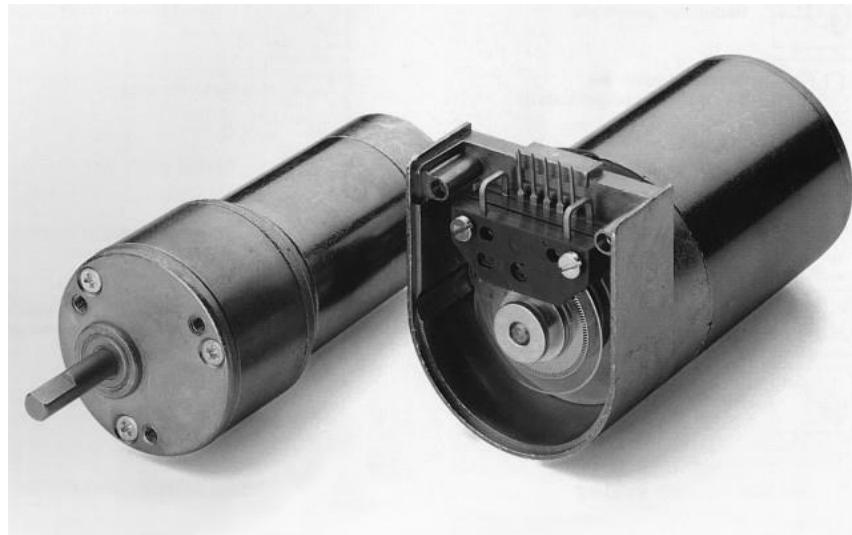


Figura 4.2 – Motori PITTMAN.

Per il dimensionamento degli accumulatori possiamo considerare che l'autonomia del robot dipende dal pacco batterie; se le batterie hanno complessivamente E joule di energia il tempo di vita sarà:

$$t = \frac{E}{P_{TOT}}$$

ove P_{TOT} è la potenza totale assorbita dai motori, dall'elettronica, dai sensori, ecc. Normalmente per una batteria è disponibile la carica elettrica espressa in Ah (ampere/ora). Si ha:

$$Ah = \frac{joule}{volt \cdot 3600} = \frac{t \cdot P_{TOT}}{volt \cdot 3600}$$

Ora dobbiamo stimare l'energia totale assorbita. Per quanto riguarda i motori possiamo considerare il lavoro meccanico che si vuole far eseguire al veicolo:

$$L = F_{APP} \cdot d = M \cdot a_c \cdot d$$

essendo d la distanza percorsa ed a_c l'accelerazione impressa al robot. L'energia elettrica sarà legata a quella meccanica attraverso il rendimento del motore (paragrafo 2.1). Ipotizzando:

- un'autonomia di 1000 m;
- un'accelerazione di 1m/s²;
- una massa totale pari a 20 Kg;
- un rendimento medio dei motori pari al 30%;
- un consumo per l'elettronica ed i sensori pari a 25W;
- un consumo per il PC a bordo del robot pari a 100W;
- un rendimento per l'alimentatore del PC pari all'85%;
- una durata delle batterie di 30 minuti;

la carica elettrica richiesta al pacco batterie da 24V sarà quindi:

$$Q = \frac{t \cdot \left(\frac{P_{PC}}{\eta_{ALIM}} + P_{E\&S} \right) + \frac{M \cdot a_c \cdot d}{\eta_{MOT}}}{V \cdot 3600} = \frac{30 \cdot 60 \cdot \left(\frac{100}{0.85} + 25 \right) + \frac{20 \cdot 1 \cdot 1000}{0.3}}{24 \cdot 3600} = 3.7 Ah$$

4.3 Realizzazione dell'hardware

Abbiamo visto brevemente che la parte hardware del sistema realizzato per controllo dei motori è costituito da due schede. In particolare abbiamo:

- Una scheda di controllo su bus ISA che integra la logica per la generazione dei due segnali PWM e la lettura degli encoder. Per il primo scopo si sono resi necessari 2 contatori a 16 bit più un generatore di clock a 10 MHz, per il secondo scopo 4 contatori a 16 bit più della logica d'interfaccia agli encoder. Infatti, questi ultimi riportano due segnali TTL in quadratura (cioè sfasati di 90 gradi, vedere paragrafo 1.4.1), che vengono trasformati in impulsi UP/DOWN secondo la direzione di rotazione del motore (paragrafo 4.3.1). La trasformazione è effettuata con una funzione logica implementata tramite GAL (chip programmabile) in modo da ridurre il numero di componenti sulla scheda [LATTICE, 1998]. Un'altra GAL fornisce l'interfaccia al bus di sistema. I 6 contatori a 16 bit sono integrati in due chip 82C54. Infine, alcuni buffer a trigger di schmitt vengono usati per ripulire dal rumore i segnali provenienti dagli encoder e fornire i segnali di controllo alla scheda di potenza.
- Una scheda di potenza la cui funzione è di fornire ai motori la necessaria corrente (fino ad 8.5 ampere per motore). Il cuore della scheda è costituito da due chip HIP4081A e otto MOSFET IRF520. La scheda di potenza isola i segnali provenienti dalla scheda ISA tramite alcuni optoisolatori "veloci". Altri chip assicurano in caso di guasto un'immediata disabilitazione dei motori (trattandosi di un sistema fisico in movimento occorre considerare le eventuali conseguenze di un guasto). Infine un'accurata progettazione del PCB riduce le emissioni elettromagnetiche (sempre presenti in un sistema a

commutazione). Una sostituzione dei MOSFET con tipi più potenti (IRF530 od altri) permette facilmente di superare i 20 ampere per motore.

La figura seguente rappresenta uno schema a blocchi completo dell'hardware realizzato:

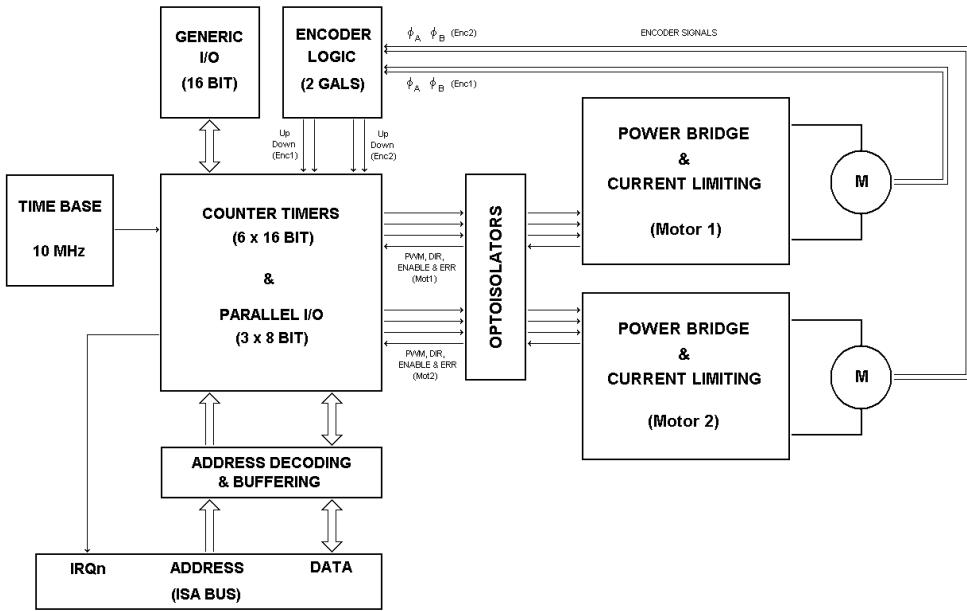


Figura 4.3 – Schema a blocchi della parte hardware di controllo dei motori.

Vediamo ora in dettaglio ciascuna delle due schede.

4.3.1 Scheda di controllo

La scheda di controllo realizzata presenta le seguenti caratteristiche:

- interfaccia con il bus ISA; spazio di indirizzamento di 16 byte, selezionabile tra 280h e 39Fh tramite jumper; interrupt selezionabile tra IRQ10, IRQ11, IRQ15;

- frequenza delle interruzioni (*servo update time*) pari a 1.6ms, 800 μ s o 400 μ s, selezionabile tramite jumper;
- generazione di due segnali PWM a semionda o ad onda intera (la modalità è stabilita via software dal driver); risoluzione e frequenza del PWM, selezionabili tramite jumper tra le seguenti: 10 bit e 10 KHz; 9 bit e 20 KHz; 8 bit e 40 KHz;
- lettura di due encoder con interfaccia TTL; filtraggio antirumore dei segnali; decodifica con moltiplica 1X/4X e cambio di direzione, entrambi selezionabili tramite jumper;
- 16 bit di I/O per usi generali.

Lo schema elettrico completo della scheda è riportato in appendice A1. Possiamo suddividere questo circuito in cinque parti:

- interfaccia con il bus ISA;
- base dei tempi e generazione degli interrupt;
- generazione dei segnali PWM;
- lettura di posizione degli encoder;
- gestione dell'I/O.

Prima di vedere in dettaglio il funzionamento di ciascuna parte esaminiamo il componente che risulta il cuore del nostro circuito, il counter timer 82C54.

IL COUNTER TIMER 82C54

L'82C54 è un componente CMOS progettato dalla Intel, originariamente prodotto in tecnologia NMOS (le due versioni hanno la stessa piedinatura, tutti gli ingressi ed uscite compatibili TTL e forniscono le stesse funzionalità) ed è stato pensato come una soluzione “one-chip” a molti problemi di conteggio e

temporizzazione [INTEL, 1994]. L'82C54 è semplice da utilizzare e da interfacciare, ed è dotato di tre contatori a 16 bit indipendenti che possiedono sei modi di funzionamento programmabili. Il datasheet in appendice A5 dà tutti i dettagli su questo componente estremamente flessibile. La figura seguente ne mostra la struttura interna:

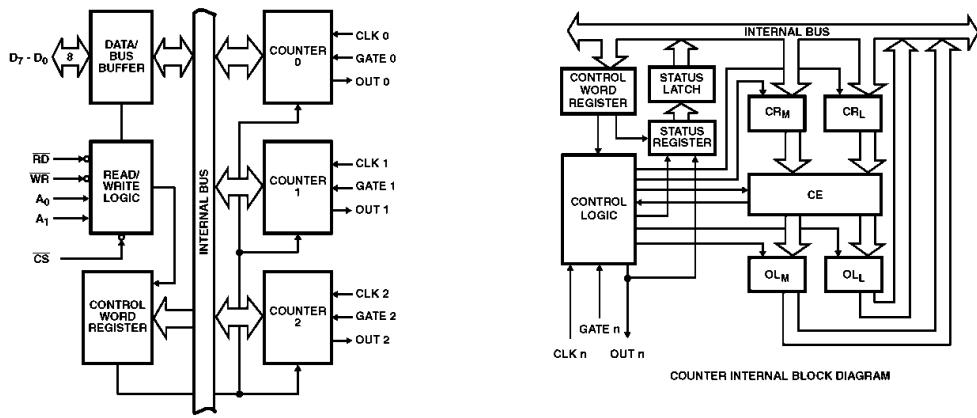


Figura 4.4 – Struttura funzionale dell'82C54.

Come mostrato dalla figura, ciascun contatore possiede un'uscita (OUT), un ingresso di abilitazione (GATE) ed un ingresso di clock (CLK). Il componente si interfaccia direttamente con i principali microprocessori: i segnali -RD (*read*) e -WR (*write*) indicano se l'operazione è di lettura o di scrittura, il pin -CS (*chip select*) deve essere fornito dalla logica di decodifica degli indirizzi per attivare il chip. Gli indirizzi A0 e A1 vengono utilizzati per selezionare i registri interni; l'82C54 occupa uno spazio di indirizzamento di 4 byte.

Come abbiamo già accennato, il modo di funzionamento di ciascun contatore è programmabile e può essere scelto tra sei modi:

- Modo 0: “interrupt on terminal count”;
- Modo 1: “hardware retrigable one-shot”;
- Modo 2: “rate generator”;
- Modo 3: “square wave mode”;
- Modo 4: “software triggered mode”;
- Modo 5: “hardware triggered strobe”.

Nella nostra scheda ISA abbiamo impiegato due integrati 82C54 (ciascuno associato ad un motore), utilizzando il counter 0 per generare il segnale PWM e i counter 2 e 3 per leggere l'encoder corrispondente. Il counter 0 viene fatto funzionare in modo 1 mentre i counter 2 e 3 in modo 4. Vediamo quindi queste due modalità di funzionamento (per gli altri modi e per maggiori dettagli si veda il datasheet in appendice A5).

MODO 1 (hardware retrigable one-shot)

La figura a fianco mostra il funzionamento dei counter in modo 1: dopo aver ricevuto la modalità di funzionamento e il valore del conteggio il contatore si pone “in attesa”. Un fronte positivo sul gate determina, sul successivo impulso di clock, il caricamento del valore di conteggio e un valore basso per OUT; da quel momento ogni altro impulso di clock determina un decremento del valore presente nel contatore. Quando questo arriva a zero OUT torna alto e il conteggio prosegue (il contatore riparte da FFFFh), finché un nuovo fronte positivo sul GATE attiva un nuovo caricamento del valore.

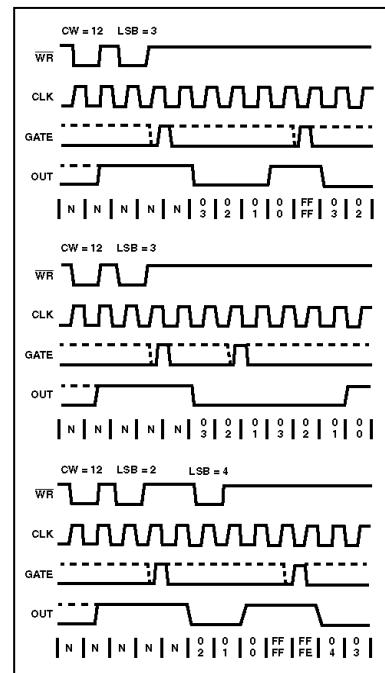


Figura 4.5 – Modo 1

MODO 4 (software triggered mode)

La figura a fianco mostra il funzionamento dei counter in modo 4: dopo aver ricevuto la modalità di funzionamento e il valore del conteggio il contatore viene caricato (ma non decrementato) sul primo impulso di clock, mentre gli impulsi successivi determinano un decremento del contatore. OUT va basso (solo per la durata di un impulso di clock) quando il conteggio arriva a zero. Un valore basso sul GATE disabilita il decremento del contatore. L'intera sequenza di conteggio è attivata via software dalla scrittura del valore da conteggiare.

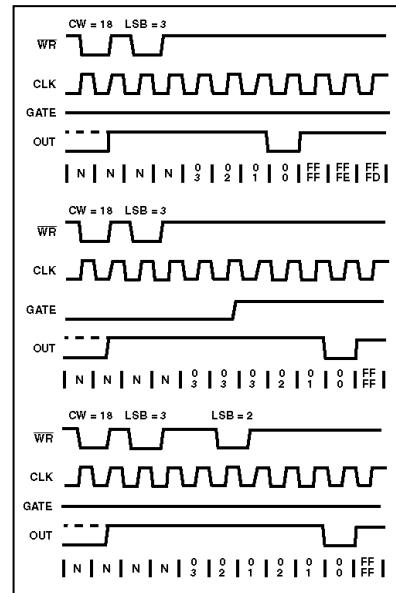


Figura 4.6 – Modo 4

L'82C54 ha l'utile possibilità di congelare simultaneamente il valore di tutti contatori senza disturbare il conteggio. Questo viene effettuato tramite un comando particolare (*read back*) inviato al *Control Register*: il valore di ogni contatore viene depositato in un registro a 16 bit (composto dai due registri OL_M , OL_L visibili nella figura 4.4), e lì mantenuto fino alla lettura. Useremo questa possibilità per determinare in modo sicuro la posizione simultanea degli encoder (vedere più avanti).

INTERFACCIA CON IL BUS ISA

Il bus ISA (*Industry Standard Architecture*) ha origine nel lontano 1980 presso un laboratorio dell'IBM in Florida. Il primo personal computer, introdotto dall'IBM sul mercato nel 1981 includeva un sottoinsieme ad 8 bit del bus ISA. Nel 1984, l'IBM introdusse il PC-AT il quale possedeva la piena

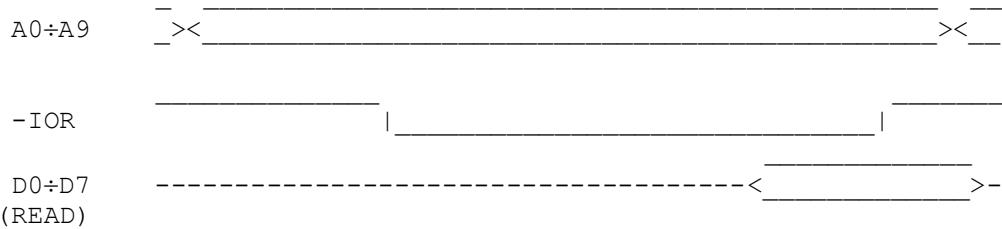
implementazione del bus ISA a 16 bit, all'epoca indicato come "AT bus". Quando altre aziende produttrici di computer cominciarono a produrre cloni del PC IBM, venne coniato il termine ISA, come sinonimo di AT-bus: questo nome infatti non poteva essere utilizzato in quanto coperto da marchio registrato.

Sul bus ISA sono presenti indirizzi, dati e segnali di controllo. Per quel che riguarda la nostra scheda ci interessano particolarmente i seguenti segnali [Shanley e Anderson, 1995]:

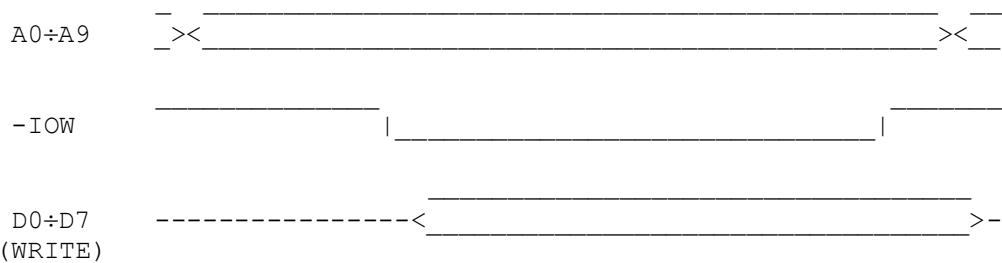
- **A0 ÷ A23** (*bit di indirizzo*): sono utilizzati per indirizzare la memoria e l'I/O. Per ragioni storiche, la maggior parte delle schede in commercio utilizza per l'I/O solo A0 ÷ A9. Questi bit restano validi durante tutto il ciclo di lettura o di scrittura e sono normalmente controllati dalla CPU o dal controllore DMA (*direct memory access*) presente sulla scheda madre del PC, ma possono anche essere gestiti da una scheda ISA che abbia capacità di controllo del bus (*bus-master*).
- **AEN** (*address enable*): questo segnale è attivo quando il controllore DMA di sistema ha il controllo del bus: tale segnale deve perciò essere incluso nella logica di decodifica della scheda ISA per evitare accessi indesiderati durante i cicli DMA.
- **D0 ÷ D15** (*bit di dati*): costituiscono il bus dati. Per i dispositivi ad 8 bit sono utilizzati solo D0 ÷ D7. Alcuni dispositivi a 16 bit richiedono due cicli di lettura/scrittura ad 8 bit anziché uno unico a 16 bit.
- **-IOR** (*I/O read*): è controllato dal proprietario del bus ed istruisce il dispositivo di I/O selezionato di porre il dato sul bus. È attivo basso.

- **-IOW (I/O write)**: è controllato dal proprietario del bus ed istruisce il dispositivo di I/O selezionato di prelevare il dato dal bus. È attivo basso.
- **IRQ3 ÷ IRQ7, IRQ9 ÷ IRQ12, IRQ14, IRQ15 (interrupt requests)**: sono utilizzati per segnalare alla CPU che una scheda ISA richiede attenzione. Un interrupt è generato quando una linea IRQ è portata dallo stato basso a quello alto. La linea deve essere mantenuta alta finché il microprocessore non abbia riconosciuto l'interruzione ed avviato la routine di gestione delle interruzioni.
- **RESET DRV (reset)**: è mantenuto alto per inizializzare la logica di sistema.

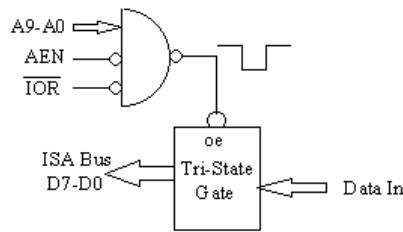
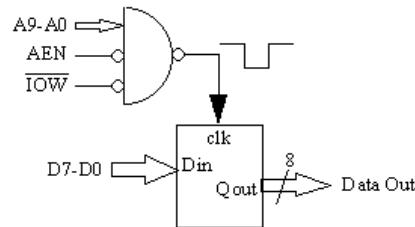
Il diagramma seguente mostra le temporizzazioni di una lettura di I/O ad 8 bit:



Queste sono invece le temporizzazioni di una scrittura di I/O ad 8 bit:



Le due figure seguenti mostrano invece due semplici schemi per la lettura e la scrittura su porte di I/O ad 8 bit:

**Figura 4.7 – Semplice schema per la lettura di I/O.****Figura 4.8 – Semplice schema per la scrittura su I/O.**

Nelle due figure si nota la decodifica degli indirizzi, schematizzata con una freccia e l'uso del segnale AEN.

Nella nostra scheda la decodifica degli indirizzi è fatta dalla GAL U2 (la funzione logica è fornita in appendice A3): l'indirizzo di base della scheda è impostabile tramite il gruppo di jumper JP2 sui seguenti valori: 0280h, 0290h, 0320h, 0330h, 0340h, 0350h, 0380h, 0390h. Nella scelta di questi indirizzi si è tenuto conto dello spazio normalmente riservato ai componenti presenti sulla motherboard del PC (quali porte seriali e parallele, interfaccia HD, ecc) e quello riservato alle schede più comuni (VGA, schede di rete, ecc). I bit di selezione dell'indirizzo di base vengono confrontati dalla GAL U2 con l'indirizzo fornito dal bus sui bit A2 ÷ A9 (figura 4.9). La scheda occupa uno spazio di I/O pari ad 16 indirizzi contigui che sono assegnati come mostra la seguente tabella (con

offset si è indicato il valore da sommare all'indirizzo base per avere l'effettivo indirizzo del dispositivo):

Offset	Dispositivo Selezionato	Parte del dispositivo
00h (0)	82C55	Porta PA
01h (1)	82C55	Porta PB
02h (2)	82C55	Porta PC
03h (3)	82C55	Control Register
04h (4)	82C54 #1	Counter 0
05h (5)	82C54 #1	Counter 1
06h (6)	82C54 #1	Counter 2
07h (7)	82C54 #1	Control Register
08h (8)	82C54 #2	Counter 0
09h (9)	82C54 #2	Counter 1
0Ah (10)	82C54 #2	Counter 2
0Bh (11)	82C54 #2	Control Register
0Ch (12)	82C54 #1 & #2 (sola scrittura)	Counter 0
0Dh (13)	82C54 #1 & #2 (sola scrittura)	Counter 1
0Eh (14)	82C54 #1 & #2 (sola scrittura)	Counter 2
0Fh (15)	82C54 #1 & #2 (sola scrittura)	Control Register

L'integrato 82C55 (vedere più avanti la parte sull'I/O) viene abilitato dalla GAL U2 ponendo basso il segnale -CS0 (figura 4.9), mentre i due 82C54 vengono abilitati ponendo bassi -CS1 e -CS2. Si noti che l'indirizzamento simultaneo dei due chip 82C54 permette di inizializzare contemporaneamente i contatori, risparmiando preziosi cicli di I/O (sul bus ISA l'I/O è lento) e, cosa più importante, permette attraverso il comando “read-back” di congelare simultaneamente i valori dei 4 contatori che vengono utilizzati per la lettura

della posizione degli encoder. Questo elimina la possibilità di errori dovuti a transizioni che si potrebbero presentare tra una lettura e l'altra. Infatti, una volta congelati, i valori possono essere letti “con comodità” dall'anello di controllo, indirizzando i singoli contatori. Per prevenire errori di programmazione che potrebbero danneggiare l'hardware, l'accesso simultaneo ai due timer 82C54 è reso possibile solo in scrittura (la lettura provocherebbe infatti conflitti sul bus). Questo è imposto dalla GAL U2, che tiene conto dello stato del segnale -IOR, fornito dal bus ISA. Infine il buffer driver 74LS245 isola il bus dati del PC ed amplifica i segnali dei chip 82C54 / 82C55, che essendo CMOS (anche se con livelli di tensione TTL) non sono in grado di fornire le stesse correnti della loro controparte NMOS.

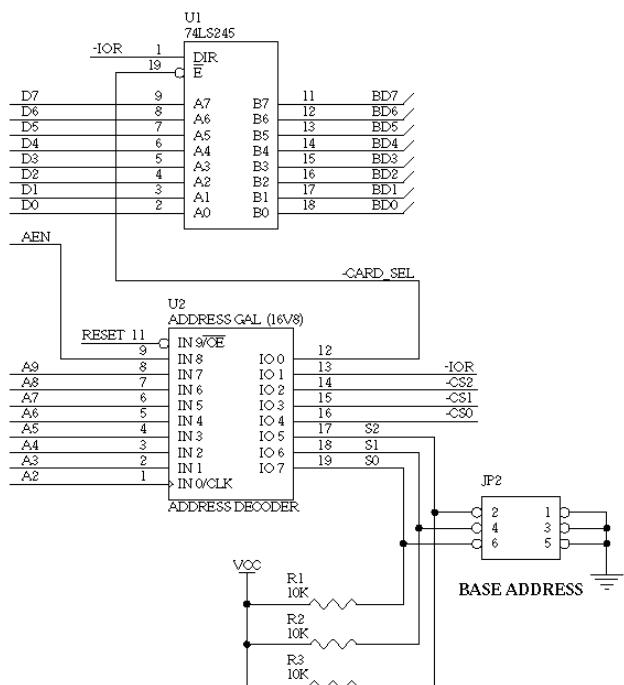


Figura 4.9 – Interfaccia con il bus ISA.

GESTIONE DELL'I/O

Si basa sull'82C55, dispositivo di interfaccia parallela, che permette di avere a disposizione 24 linee di I/O, divise su 3 porte [INTEL, 1995]. Le tre porte, chiamate PA, PB, e PC hanno ciascuna 8 bit. Vi sono tre modalità di funzionamento: modo 0, modo 1 e modo 2; le tre porte non sono tra loro equivalenti e quindi non tutte possono funzionare nelle 3 modalità (per i dettagli si rimanda al datasheet presente in appendice A5). Noi abbiamo utilizzato il modo 0, a disposizione di tutte le porte: in questa modalità ciascuna porta funziona come normale *latch*, se programmata come output, o nel ruolo di *buffer 3-state*, se programmata come input; fa eccezione la porta C che può essere suddivisa in due gruppi di quattro bit (PC0 ÷ PC3 e PC4 ÷ PC7), ciascuno configurabile indipendentemente dall'altro come input o come output. L'82C55 occupa 4 indirizzi di I/O, tre per le porte PA, PB e PC ed il quarto per il *Control Register* del componente.

Come mostra la figura 4.10 (o lo schema elettrico in appendice) la porta PA e la porta PB sono a disposizione del programmatore e possono essere usate sia come input che come output (la modalità è stabilita durante l'inizializzazione del driver). La porta PC è invece utilizzata dalla scheda: i quattro bit meno significativi sono configurati come output ed utilizzati per abilitare/disabilitare i motori e gli interrupt (PC0, PC1), e per stabilire la direzione di rotazione dei motori nella modalità PWM a semionda (PC2, PC3). I quattro bit più significativi sono invece configurati come input: due di loro (PC4, PC5) sono utilizzati per rilevare le condizioni di overcurrent dei motori, gli altri due (PC6, PC7) sono inutilizzati.

Si osservi che gli interrupt vengono abilitati quando almeno un motore risulta abilitato. Al reset l'82C55 si pone con tutte le porte in input e i pin PC0 e PC1

vengono mantenuti alti da R16 e R17, disabilitando gli interrupt: è infatti di fondamentale importanza che non vengano generati interrupt nella fase di inizializzazione del PC.

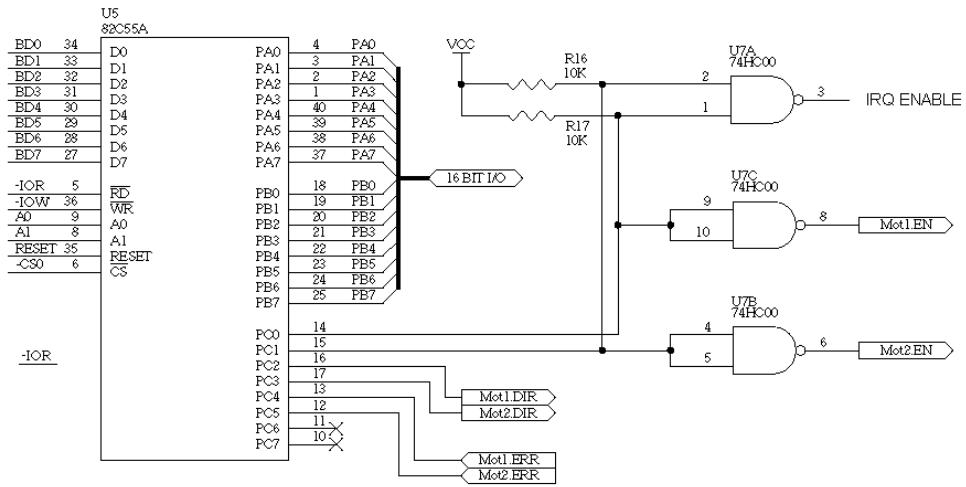
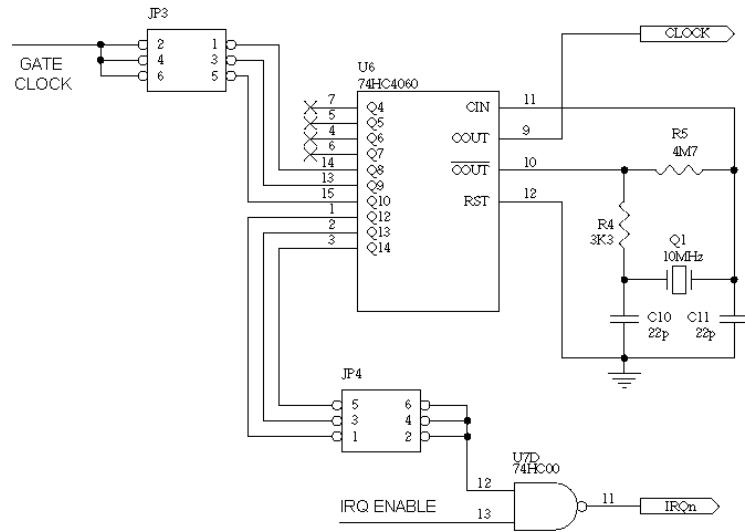


Figura 4.10 – Gestione dell'I/O

BASE DEI TEMPI E GENERAZIONE DEGLI INTERRUPT

La base dei tempi, mostrata in figura 4.11, provvede a fornire i segnali di clock necessari ai vari dispositivi per le temporizzazioni e per la generazione degli interrupt. L'integrato 74HC4060, un oscillatore/divisore CMOS, mette a disposizione sui suoi piedini un'onda quadra di frequenza pari a quella di oscillazione del quarzo divisa per una potenza di due (ad esempio la frequenza dell'onda quadra in uscita da Q4 è $10E6/2^4$, quella in uscita da Q14 è $10E6/2^{14}$, ecc). Tre di queste uscite (Q12, Q13, Q14) sono utilizzate per generare gli interrupt: il periodo di interruzione che è anche l'intervallo di aggiornamento dell'anello di controllo (*servo update time*), può essere scelto tramite JP4 tra i seguenti: $(10^7/2^{12})^{-1} \approx 410\text{ ns}$, $(10^7/2^{13})^{-1} \approx 820\text{ ns}$ e $(10^7/2^{14})^{-1} \approx 1.64\text{ ms}$.

**Figura 4.11 – Base dei tempi.**

Abbiamo detto che gli interrupt sul bus ISA devono restare attivi finché non vengono riconosciuti; poiché il segnale utilizzato per gli interrupt è un'onda quadra, non vi sono problemi di riconoscimento: il segnale resta alto quanto basta affinché un processo realtime sia attivato per servire le interruzioni. Vedremo infatti nel paragrafo 4.4 che, grazie all'impiego di un sistema operativo realtime, la latenza massima non supera i 25 microsecondi.

Altre tre uscite del 74HC4060 (Q8, Q9, Q10) vengono utilizzate per pilotare i GATE del counter 0 nei due 82C54. Come vedremo tra poco, questo permette di generare facilmente i segnali PWM. Per ora si tenga presente che il segnale inviato ai due gate è un'onda quadra di frequenza scelta tramite JP3 tra: $10^7/2^8 \approx 39 \text{ KHz}$, $10^7/2^9 \approx 19.5 \text{ KHz}$ e $10^7/2^{10} \approx 9.8 \text{ KHz}$.

Generazione dei segnali PWM

Per la generazione dei segnali PWM si utilizza il counter 0 di ciascun 82C54: la figura 4.12 mostra il circuito di generazione del PWM del secondo motore

(l'altro è analogo). Il counter 0 viene fatto lavorare nella modalità 1; in tale modalità ogni nuovo conteggio è avviato dal fronte di salita del GATE. Ma il GATE è pilotato da un'onda quadra la cui frequenza è un sottomultiplo della frequenza del clock, applicato al pin CLK del contatore. Se il valore caricato nel contatore è inferiore al numero di impulsi di clock che cadono in un periodo del GATE, sul pin OUT si ottiene un'onda rettangolare il cui duty cycle riflette il valore caricato nel contatore.

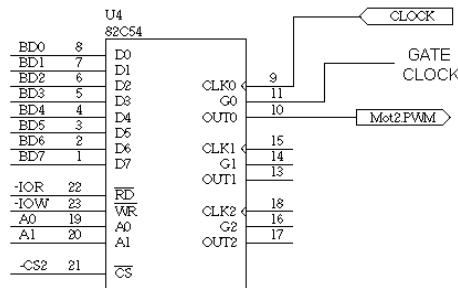


Figura 4.12 – Circuito di generazione dell'onda PWM.

La figura seguente mostra che cosa accade con un'ipotetica onda quadra applicata al GATE di frequenza pari a 1/16 della frequenza del clock applicato al pin CLK, e un valore di conteggio pari a 4.

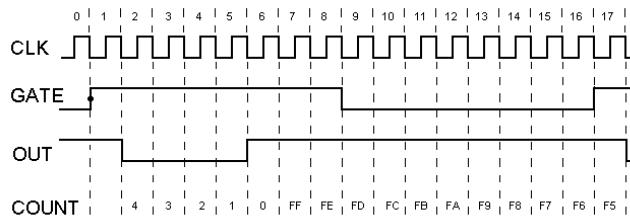


Figura 4.13 – Temporizzazioni nella generazione dell'onda PWM

Come si può osservare il pin OUT va basso il ciclo successivo a quello in cui il GATE va alto e rimane basso finché il contatore decrementandosi va a zero. Variando il valore memorizzato nel contatore si può variare il duty cycle

dell'onda in uscita dal pin OUT, con una risoluzione di 4 bit (16 possibili duty cycle). In particolare un valore memorizzato pari a 0 mantiene OUT sempre alto, mentre un valore memorizzato pari a 15 (o un valore maggiore) mantiene OUT sempre basso, perché il conteggio viene fatto ripartire dal fronte positivo del GATE prima che il precedente sia terminato.

Nella nostra scheda il periodo del GATE è 2^8 , 2^9 o 2^{10} volte il periodo del clock (secondo la scelta operata tramite il jumper JP3), quindi otteniamo un'onda PWM con frequenza prossima a 40, 20 o 10 KHz, e con una risoluzione pari rispettivamente ad 8, 9 o 10 bit.

LETTURA DI POSIZIONE DEGLI ENCODER

I segnali (Phase A, Phase B) provenienti da ciascun encoder incrementale, sono filtrati mediante un filtro resistivo-capacitivo ed un trigger di Schmidt: la figura 4.14 mostra il front-end per l'encoder del primo motore (l'altro è analogo).

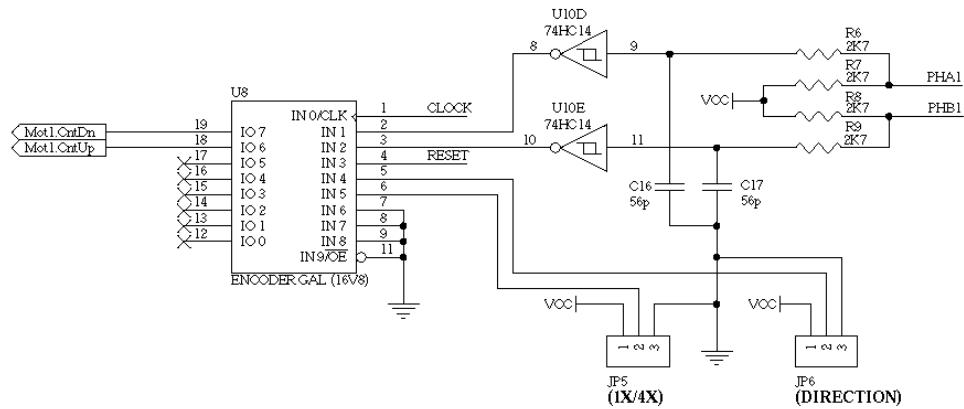


Figura 4.14 – Circuito di filtraggio e trasformazione dei segnali di un encoder.

I segnali in quadratura di fase, ripuliti dall'eventuale rumore, sono processati da una GAL (U8 per l'encoder del motore 1, U9 per l'encoder del motore 2).

Questa GAL, la cui funzione logica è riportata in appendice A3, ha il ruolo di convertire gli impulsi in quadratura, provenienti da un encoder, in due treni di impulsi *CntUp* e *CntDn* (figura 4.15). Mediante un jumper (JP5) la GAL permette di migliorare la risoluzione dell'encoder moltiplicando per 4 il numero di impulsi (sfruttando opportunamente i fronti di salita e di discesa dei segnali in quadratura); è possibile anche un'inversione "virtuale" del senso di rotazione (jumper JP6).

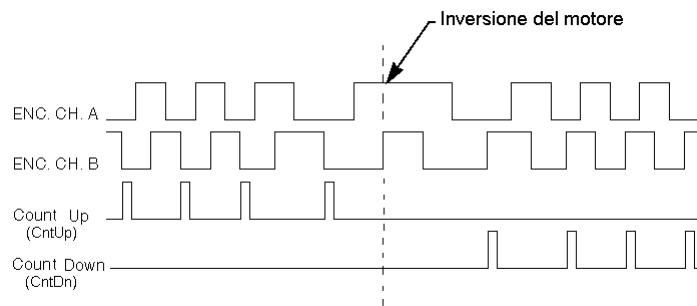


Figura 4.15 – Conversione logica dei segnali di un encoder.

La lettura effettiva di posizione è realizzata utilizzando i contatori 2 e 3 presenti nell'82C54 (figura 4.16): i segnali *CntUp* e *CntDn* sono inviati agli ingressi CLK dei due counter, utilizzati in modo 4, mentre il GATE è tenuto a +5V affinché il conteggio sia sempre abilitato. In pratica i counter 2 e 3 vengono utilizzati come normali contatori a decremento. Il driver tiene traccia dell'incremento di posizione dell'encoder facendo la differenza tra i questi due contatori; questa operazione è effettuata ogni volta che viene invocata la routine di gestione delle interruzioni. Poiché entrambi i contatori sono a 16 bit, non vi sono rischi di perdere impulsi a meno che la frequenza dei segnali provenienti dagli encoder non sia più grande di 32767 volte la frequenza di aggiornamento dell'anello di controllo (cioè quella di generazione degli interrupt). Inoltre non occorre preoccuparsi di dover resettare i contatori ad ogni ciclo: poiché si considera la differenza tra i contatori l'aritmetica in complemento a due tiene

automaticamente conto di eventuali overflow dei valori a 16 bit (*wrap around*). Per maggiori dettagli sulla lettura dei contatori si veda il codice riportato in appendice A4.

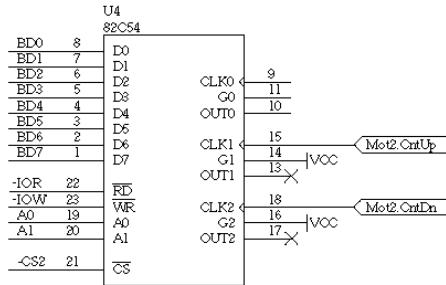


Figura 4.16 – Circuito per la lettura della posizione incrementale di un encoder.

4.3.2 Scheda di potenza

La scheda di potenza realizzata presenta le seguenti caratteristiche:

- optoisolamento dei segnali di controllo;
- unica tensione di alimentazione: $12 \div 24V$;
- stadio di potenza per due motori, corrente massima di 8.5 A per motore, limitazione della corrente nei motori: $100 \text{ mA} \div 8.5 \text{ A}$, regolabile tramite trimmer multigiri;
- modalità PWM a semionda o ad onda intera, selezionabile tramite jumper; risparmio energetico nel PWM a semionda (disabilitazione automatica del motore se il duty cycle del segnale PWM persiste a lungo al valore di 0%);
- disabilitazione automatica dei motori in assenza dei segnali PWM (per prevenire guasti della scheda di controllo e falsi contatti dei connettori);
- bassa emissione elettromagnetica, circuito *antispike* verso l'alimentazione;
- led di segnalazione “motor enabled” e “overcurrent”;
- dissipatore isolato;

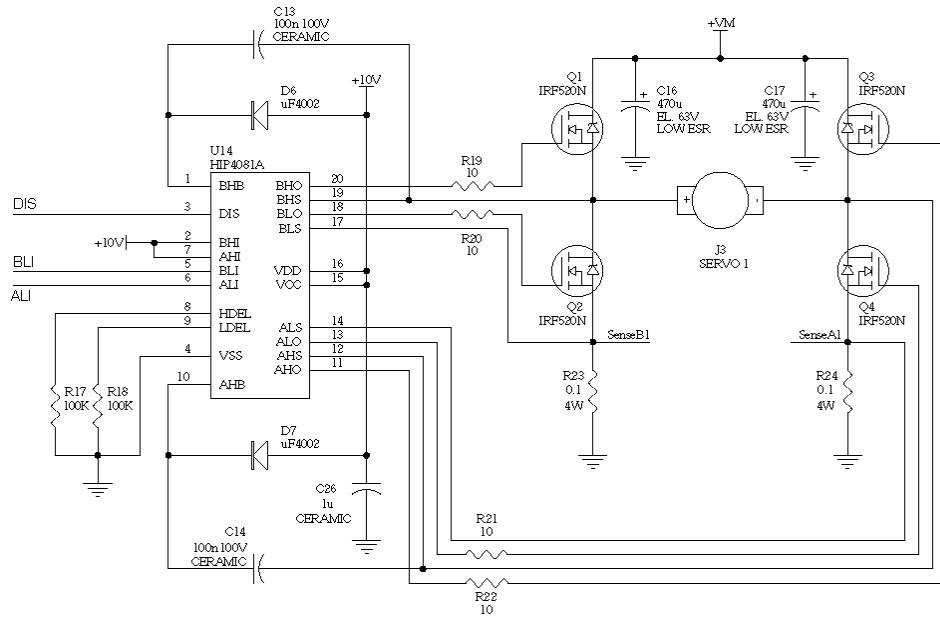
- facilmente modificabile per correnti maggiori (devono essere sostituiti i MOSFET, i condensatori di *bootstrap* e i *sense resistor*).

Lo schema elettrico completo della scheda è riportato in appendice A1. Possiamo suddividere questo circuito in cinque parti:

- ponte di potenza;
- circuito di protezione e limitazione della corrente;
- circuito di selezione della modalità del PWM (semonda, onda intera);
- sezione di optoisolamento;
- sezione di alimentazione.

PONTE DI POTENZA

I due ponti di potenza per il pilotaggio dei motori sono basati su una configurazione ibrida formata essenzialmente dal circuito integrato HIP4081A e quattro MOSFET di potenza IRF520 [SGS, 1997]. La figura 4.17 mostra il ponte di potenza relativo al motore 1 (l'altro è analogo). L'HIP4081A è un driver per “Ponti H” a MOSFET (a canale N), che supporta frequenze di commutazione fino a 1 MHz e tensioni di alimentazione fino ad 80V [INTERSIL, 1996]. Esso è in grado di gestire tutte le possibili configurazioni del ponte, tranne quelle che ne determinerebbero la sua distruzione (vedere paragrafo 3.2.3). L'integrato si fa carico di risolvere tutti i problemi di polarizzazione dei MOSFET: un problema di progettazione dei “Ponti H” a MOSFET è infatti legato alla difficoltà pratica di pilotare i due MOSFET superiori: i loro *drain* non sono connessi a massa, ma ad una tensione variabile. Come conseguenza si ha che l'ampiezza degli impulsi di *gate* non deve essere riferita a massa, ma al *drain*, e quindi deve risultare superiore alla tensione di alimentazione del motore [Millman e Halkias, 1978].

**Figura 4.17 – Ponte H di potenza a MOSFET.**

L'HIP4081A ha 5 ingressi: ALI, BLI, AHI e BHI, che controllano la configurazione del ponte, e DIS che permette di disabilitare il ponte indipendentemente dallo stato degli altri 4 ingressi. La tabella seguente illustra il funzionamento logico dell'integrato (X ha il significato di “indifferent”):

INPUT			OUTPUT	
ALI, BLI	AHI, BHI	DIS	ALO, BLO	AHO, BHO
X	X	1	0	0
1	X	0	1	0
0	1	0	0	1
0	0	0	0	0
X	X	X	0	0

Tabella 4.1 – Tabella logica di verità dell'integrato HIP4081A.

Come illustrato dalla tabella, gli ingressi ALI e BLI “dominano” gli ingressi AHI e BHI; per questo è possibile mantenere sempre alti AHI e BHI per semplificare la logica di controllo.

Il dead time (paragrafo 3.2.3) è regolabile tramite i resistori collegati ai piedini HDEL e LDEL dell'integrato HIP4081A.

CIRCUITO DI PROTEZIONE E LIMITAZIONE DELLA CORRENTE

Nel paragrafo 3.2.3 abbiamo visto che secondo “lo stato” del motore, durante la commutazione del ponte, si possono generare correnti molto elevate e sovratensioni, tali da mettere in pericolo sia il motore che l'elettronica di comando. Uno degli accorgimenti è quello di dotare la tensione di alimentazione del ponte di capacità adeguate, anti-induttive ed in grado di assorbire forti picchi di tensione della durata di pochi nanosecondi (LOW ESR). L'altro accorgimento è quello di dotare il sistema di un circuito di monitoraggio per la disabilitazione del ponte quando la corrente supera il valore massimo consentito.

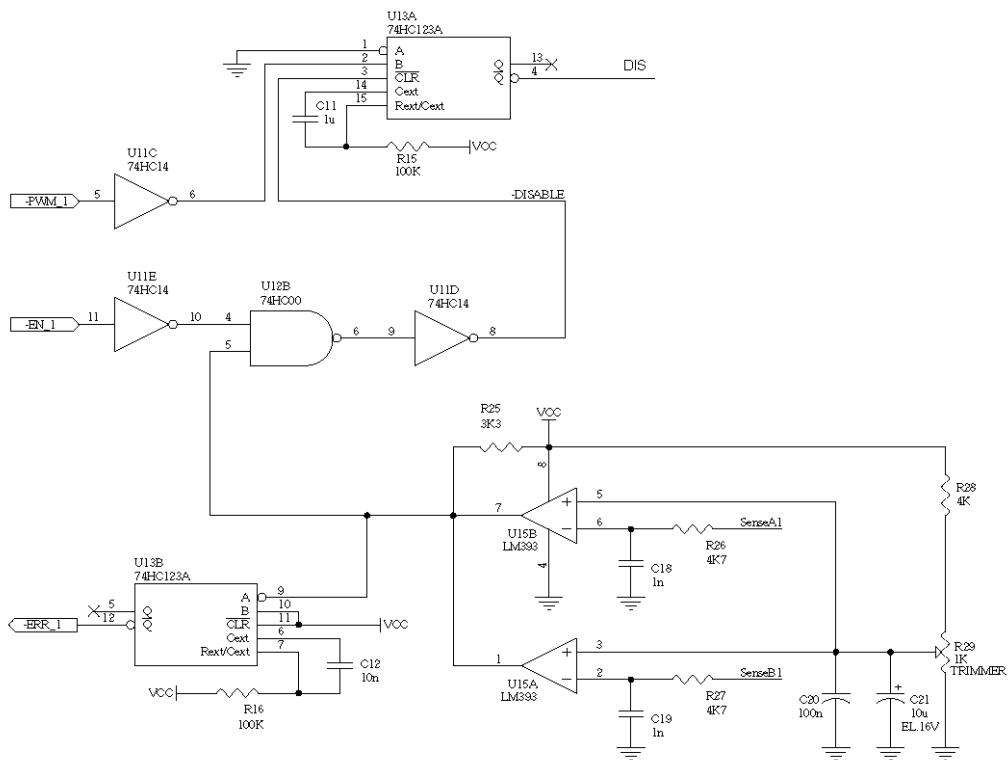


Figura 4.18 – Circuito di protezione e limitazione della corrente.

Il circuito di protezione da noi realizzato (mostrato in figura 4.18) provvede alla disabilitazione del ponte non solo quando si verifica uno stato di overcurrent, ma anche quando il segnale PWM non giunge più alla scheda di potenza (ad esempio per guasto della scheda di controllo o per falsi contatti sui connettori). Il funzionamento è il seguente: normalmente $DIS = \bar{Q} = 1$, poiché il monostabile U13A è a riposo. Quando arriva il segnale PWM, il monostabile viene commutato e DIS portato basso: il ponte viene quindi abilitato; un'eventuale assenza dell'onda PWM riporta, dopo un ritardo di pochi millisecondi, di nuovo il monostabile nello stato di riposo, disabilitando il ponte. In presenza di segnale PWM regolare, il ponte può essere disabilitato per uno o più periodi dell'onda PWM, quando il pin \bar{CLR} di U13A è portato basso; questo può accadere sia in presenza di disabilitazione da parte della scheda di controllo (ingresso -EN), sia in presenza di una condizione di overcurrent. Infatti le correnti sui due rami del ponte vengono rilevate attraverso i resistori R23 ed R24 (figura 4.17), filtrate per eliminare picchi spuri di corrente e confrontate con il valore impostato sul trimmer R29 (figura 4.18): un eventuale superamento del valore impostato determina un livello basso all'uscita di U15A/U15B (utilizzati come comparatori) e quindi una condizione $\bar{CLR} = 0$, che causa la disabilitazione del ponte.

CIRCUITO DI SELEZIONE DELLA MODALITÀ DEL PWM

La modalità PWM desiderata (a semionda o ad onda intera) è impostata tramite i jumper JP1 e JP2 (per il primo motore). In particolare si ha:

Modalità	JP1	JP2
PWM ad onda intera	1-2	1-2
PWM a semionda	2-3	2-3

La figura seguente illustra il circuito di selezione:

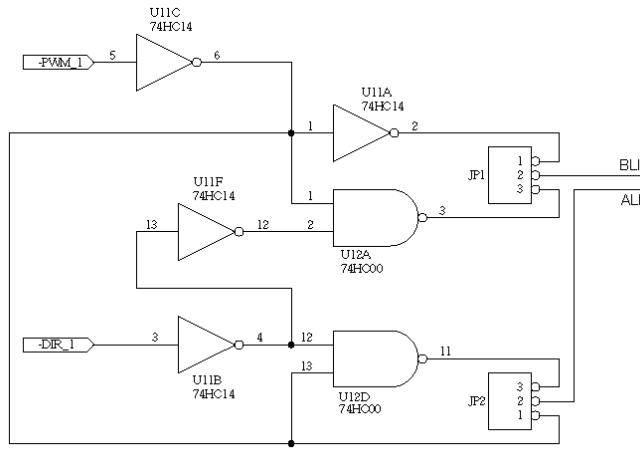


Figura 4.19 – Circuito di selezione della modalità PWM (a semionda /ad onda intera).

SEZIONE DI OPTOISOLAMENTO

Tra la scheda di controllo e quella di potenza non vi è accoppiamento diretto: tutti i segnali sono optoisolati (vedere lo schema in appendice A1). Per i segnali PWM abbiamo utilizzato optoisolatori veloci con uscita TTL, che garantiscono tempi di risposta di pochi nanosecondi e non introducono distorsioni nella forma d'onda; per gli altri segnali (più lenti) si sono utilizzati optoisolatori normali.

SEZIONE DI ALIMENTAZIONE

Si basa su due regolatori integrati (vedere lo schema in appendice A1): il 7805, che dà un'uscita di 5V e il BA10T che dà un'uscita a 10V, entrambe stabilizzate. I due regolatori possono essere alimentati con tensioni comprese tra i 12V e i 24V: non è richiesta quindi un'alimentazione separata per la logica in quanto si può utilizzare la tensione di alimentazione dei motori.

MINIMIZZAZIONE DEI DISTURBI

Grande attenzione è stata posta per ridurre il più possibile i disturbi generati dalla commutazione dei due ponti. I MOSFET hanno infatti tempi di transizione molto rapidi: i fronti di commutazione risultano ripidi e generano una notevole emissione di disturbi in alta frequenza (EMI). I criteri che ci hanno guidato in questa fase sono:

- uno studio attento del datasheet dell'HIP4081A e delle note applicative [INTERSIL, 1997]; queste forniscono consigli su come disporre le piste che collegano l'integrato ai MOSFET per ridurre gli accoppiamenti elettromagnetici;
- l'impiego di capacità di bypass tra la tensione di alimentazione del ponte e la massa, in ragione di almeno $100\mu F$ per ogni ampere di corrente: questo per assorbire adeguatamente la corrente di ricircolo durante i transitori di commutazione;
- l'impiego di capacità di bypass adeguate su ogni integrato utilizzato;
- disposizione delle piste scelta in modo da minimizzare la lunghezza di quelle percorse da forti correnti;
- impiego di un piano di massa ed immersione di alcune “piste critiche” in tale piano;
- riduzione dei disturbi verso l'alimentazione realizzata mediante un varistore (per la scarica dei picchi di tensione) ed un avvolgimento bilanciato (*balun*) di filtraggio.

4.4 Realizzazione del driver in Realtime Linux

La gestione di un dispositivo hardware presenta sempre la necessità di scrivere codice che interagisca con il dispositivo utilizzando le istruzioni privilegiate di input ed output (I/O) quali *inb()*, *outb()* e simili. Normalmente ai programmi in modalità utente non è consentito accedere all'hardware, per evitare danni al sistema: l'uso di una qualsiasi operazione di I/O viene intercettata da Linux e bloccata. Per abilitare l'I/O in modalità utente (in esecuzione come *root*) il programma deve eseguire una chiamata alla funzione *iopl()*, oppure *ioperm()* per le porte al di sotto dell'indirizzo *0x3FF* [Wall *et al.*, 2000].

Ma l'input e l'output di dati non sono le due uniche interazioni possibili con l'hardware: quando un dispositivo necessita di attenzione da parte del relativo driver, avvia una richiesta di interruzione. Il processore risponde all'interrupt salvando il suo stato e saltando alla routine di gestione delle interruzioni in precedenza notificata al sistema dal driver. Sfortunatamente la gestione degli interrupt non è possibile in modalità utente, ed occorre operare in modalità kernel.

In Linux un driver che opera a livello del kernel può essere compilato e linkato direttamente nel kernel. Ma grazie all'esistenza degli LKM (*Loadable Kernel Modules*) un driver può essere scritto sotto forma di modulo e caricato (o rimosso) senza la necessità di eseguire di nuovo la compilazione del kernel, consentendo la configurazione dinamica del sistema, mentre questo è in esecuzione.

La maggior parte dei driver che operano a livello kernel (compreso il nostro) implementano dispositivi a carattere: gli applicativi comunicano con il

dispositivo hardware leggendo e scrivendo *stream* di caratteri su file speciali (*device*) creati nella directory */dev* usando il comando *mknod*. Normalmente questi stream comprendono comandi e dati, codificati in maniera opportuna. Nel nostro driver, per facilitare l’interfacciamento con il software di più alto livello, abbiamo fornito un “wrapper”, cioè un modulo oggetto che si preoccupa di convertire chiamate a funzioni in opportuni flussi di dati inviati verso il device utilizzato (paragrafo 4.4.4).

La scrittura di un driver che opera a livello del kernel non è un’operazione banale: oltre a dover comprendere un complesso meccanismo basato su un sistema di code di messaggi, vi è la necessità di dover fare attenzione anche ai più piccoli errori. Infatti ogni errore di programmazione può portare al blocco del sistema: il debugging risulta difficile poiché, in modalità kernel, non è possibile seguire passo passo l’esecuzione del programma [Wall *et al.*, 2000].

Nel nostro caso si pone una condizione aggiuntiva: la necessità del sistema operativo di rispondere agli interrupt in tempi estremamente brevi (vedere paragrafo 4.4.3). Quando si devono gestire dispositivi elettronici con stringenti vincoli temporali, e privi di intelligenza “locale”, non è possibile utilizzare i classici sistemi operativi, a causa della loro inefficienza nel rispondere prontamente agli interrupt e agli eventi periodici. Siamo quindi dovuti ricorrere alla versione modificata di Linux, che conferisce al sistema operativo capacità hard-realtime.

Prima di illustrare il driver realizzato, vediamo meglio cosa è un sistema realtime e perché è necessaria questa capacità aggiuntiva.

4.4.1 Perché occorre un sistema realtime

La progettazione di un sistema operativo parte dal presupposto di ottimizzare il comportamento medio dell'elaboratore. Se questo non crea problemi utilizzando un editor di testo, ne può presentare qualcuno quando occorre rispondere velocemente ad un evento periodico, quale la visualizzazione di frames video. Il player video dovrebbe quasi sempre stare al passo del frame rate: la perdita di un frame ogni tanto non viene notata, mentre due pause di mezzo secondo ogni minuto provocano un effetto sgradevole. Un sistema operativo in grado di garantire il più delle volte il rispetto di determinate temporizzazioni è chiamato sistema *soft-realtime*.

Se un player video può permettersi di perdere qualche frame senza grossi danni, vi sono situazioni in cui i limiti di tempo non possono essere assolutamente superati, ad esempio nel controllo dei motori in un razzo: la sequenza di spegnimento deve essere eseguita esattamente secondo le specifiche, altrimenti il motore rischia di esplodere. In questo caso il sistema di controllo deve essere in grado di garantire sempre certi tempi di risposta massimi. Un sistema di questo tipo è indicato con il termine *hard-realtime* [Stankovic e Ramamritham, 1988].

Fortunatamente le principali applicazioni realtime non hanno malfunzionamenti così catastrofici; ciò nonostante il rispetto delle temporizzazioni deve essere garantito, poiché in genere, il venire meno alle specifiche di temporizzazione del dispositivo, porta quasi sempre alla perdita di dati. È importante osservare che i sistemi operativi più diffusi sono in grado di dare garanzie solo su un ritardo medio e questo non va bene nei sistemi realtime: infatti, un ritardo medio di risposta, ad esempio di 5 millisecondi, non esclude l'eventualità di un

ritardo occasionale di 100 millisecondi nelle circostanze più sfavorevoli, cioè quando il sistema è sovraccarico per l'esecuzione di numerosi compiti.

In alcuni casi l'hardware può compensare la mancanza di supporto realtime: ad esempio in una scheda di acquisizione si può introdurre della RAM buffer, in cui l'hardware pone i campioni acquisiti: se il buffer è abbastanza grande, allora non si perde alcun dato se il buffer viene letto entro un tempo adeguato. Però in questo approccio vi sono dei difetti: il primo è che se il sistema è sovraccarico, il buffer potrebbe comunque andare in overflow; un secondo problema è che un hardware incaricato di far fronte alle incertezze del lato software diventa necessariamente più complesso. Infine, nel caso si debba controllare un dispositivo in base all'input acquisito, un ritardo di 10 millisecondi potrebbe essere insostenibile: infatti, il buffer hardware può causare instabilità dal momento che introduce un ritardo nell'anello di controllo. Quindi, in determinate situazioni, o si sposta parte del controllo sul dispositivo utilizzando chip dedicati, microcontrollori o DSP (che richiedono sempre costosi tool di sviluppo e lunghi tempi di debugging), o si ricorre all'impiego di un sistema operativo realtime.

In commercio sono presenti varie soluzioni hard-realtime, come ad esempio il QNX, un sistema POSIX completamente scalabile (<http://www.qnx.com>). Anche nell'ambito del software “open source” vi sono alcune soluzioni, che vanno da semplici kernel, sviluppati all'interno di università, fino a sistemi più complessi quali RTLinux (<http://www.rtlinux.org>). RTLinux e' sviluppato per fornire un supporto a processi hard-realtime, sfruttando Linux per le normali applicazioni non realtime [Barabanov e Yodaiken, 1997]. Questo consente di avere un sistema hard-realtime, mantenendo tutte le potenzialità di Linux, quindi software di rete, interfaccia grafica, applicazioni scientifiche, ecc.

Introdurre un supporto realtime in un sistema operativo, mantenendo veloci le normali applicazioni, è un problema complesso. La maggior parte delle soluzioni utilizzabili per accelerare le applicazioni non realtime introduce ritardi non predibili che sono fatali per il supporto realtime. L'esempio ovvio è il *paging* della memoria virtuale: poniamo che il 99.99% delle richieste di accesso alla memoria di un programma sia soddisfatto dalla cache, e che un accesso alla cache richieda 1 millisecondo (mentre in caso contrario il dato non è disponibile prima di 500 millisecondi); in tal caso, su un periodo sufficientemente lungo, il tempo medio di accesso sarà di 1.05 millisecondi. Ma d'altra parte un processo realtime che non trova nella cache le sue prime 10 pagine impiegherà 5000 millisecondi per fare ciò che diversamente avrebbe richiesto 10 millisecondi, ed è questo caso peggiore quello che va preso in considerazione.

A causa di questi problemi in passato i sistemi hard-realtime erano prodotti di artigianato manuale, "sottosistemi" basati solo su un ciclo, in cui venivano eseguiti una serie di semplici compiti. Con l'aumento della complessità delle applicazioni realtime si è reso necessario un ambiente di esecuzione migliore. Se si vuole che il programma di controllo realtime per l'autodiagnosi del motore di un aereo sia in grado di gestire centinaia di sensori, mostrare i dati in forma grafica, collegarsi con un database e una rete, magari fornire anche un'interfaccia web, allora è fuori discussione progettare un sistema basato su un semplice ciclo. Il problema è come conservare predicitività e velocità pur lavorando in un ambiente molto sofisticato.

Una soluzione è quella di aggiungere il supporto realtime a un kernel che ne sia originariamente sprovvisto, ma questo non è semplice quanto potrebbe sembrare. Il problema maggiore è che i sistemi operativi generici sono progettati col seguente criterio: *offrire buone prestazioni nella maggior parte*

delle circostanze [3W, 6]. A esempio, studiando il sorgente di Linux con sufficiente attenzione si possono notare molte contraddizioni rispetto alle esigenze di un supporto realtime:

- Linux utilizza una sincronizzazione a “grana grossa” nella gestione di alcune delle sue strutture dati fondamentali; ciò implica che ci siano lunghi intervalli di tempo in cui un processo ha l'accesso esclusivo ad alcuni dati. Questo ritarderà l'esecuzione di un processo realtime che richieda accesso agli stessi dati. D'altra parte una sincronizzazione "a grana fine" porterebbe il sistema a sprecare molto tempo nel bloccare e sbloccare senza necessità le strutture dati, rallentando così tutti i processi.
- Linux concede prima o poi una *time-slice* anche al processo meno importante e con priorità più bassa, anche in caso ci sia un processo più importante pronto ad andare in esecuzione; questo per dar modo a qualsiasi processo in background di essere eseguito. Ma in un sistema realtime un processo a priorità maggiore non dovrebbe mai dover attendere per via di un altro con priorità minore.
- Linux riordina ed eventualmente unisce le richieste di accesso all'hardware da parte dei vari processi al fine di rendere più efficiente l'uso dell'hardware: le richieste di leggere dati dal disco di un processo a priorità minore possono avere la precedenza rispetto a quelle di un processo a priorità maggiore, con lo scopo di minimizzare le attese generali dovute agli spostamenti della testina.
- Linux fa attendere processi ad alta priorità finché processi a priorità minore non abbiano rilasciato le risorse necessarie. Se ad esempio un programma

ha allocato l'ultimo buffer di rete e il controllore di un robot deve allo stesso tempo inviare il comando "stop", il controllore dovrà attendere finché il processo meno importante non abbia rilasciato la risorsa che gli serve.

Dal momento che le finalità dei sistemi operativi generici e di quelli realtime sono differenti, non è sorprendente che meccanismi intelligenti per gli uni risultino controproducenti per gli altri.

4.4.2 Il sistema operativo RTLinux

RTLinux permette di eseguire processi hard-realtime, sulla stessa macchina su cui gira una versione standard di Linux, indipendentemente da quanto stia facendo Linux [Barabanov e Yodaiken, 1997]. Nel caso peggiore, con RTLinux, il tempo trascorso tra la rilevazione di un interrupt hardware da parte del processore e l'esecuzione del gestore di interrupt è inferiore ai 25 microsecondi (su un PC classe 486 !), mentre un processo periodico di RTLinux è eseguito sullo stesso hardware con una precisione di ± 20 microsecondi. Questi tempi hanno un limite nell'hardware e con hardware migliore anche RTLinux diventa più preciso. Come termine di paragone, Linux standard impiega fino a 30 millisecondi (NB: 30000 microsecondi) ad avviare un gestore di interrupt e può facilmente rimanere indietro di 20 millisecondi (20000 microsecondi) per un processo periodico [3W, 7].

RTLinux non tenta di implementare il supporto dei processi realtime all'interno del kernel di Linux, piuttosto scinde il sistema operativo generico da quello realtime (figura 4.20): in pratica Linux diventa il processo in background di un piccolo kernel realtime, e viene eseguito solo quando quest'ultimo non ha processi realtime da eseguire. E poiché ogni comando che Linux invia al

gestore hardware degli interrupt viene intercettato dal kernel realtime, a Linux non è mai consentito disabilitare veramente gli interrupt hardware, né introdurre ritardi nella loro gestione. Quando arriva una richiesta di interrupt dall'hardware, questa viene intercettata dal sistema realtime, che decide cosa farne: se dispone di un processo realtime specifico per la gestione di tale interrupt, lo attiva; altrimenti la richiesta è marcata come "in attesa" e il controllo viene passato a Linux. Se Linux aveva lasciato attivi gli interrupt le richieste in attesa vengono passate al sistema operativo e i suoi meccanismi standard di gestione vi operano normalmente.

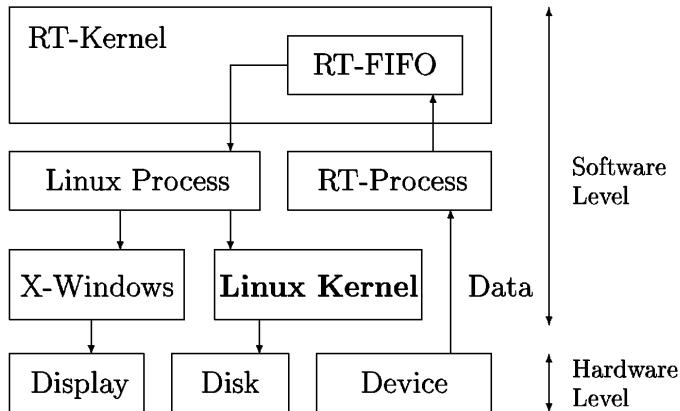


Figura 4.20 – Struttura di RTLinux.

Quindi non importa cosa faccia Linux, se sta girando in *kernel mode* o sta eseguendo un normale processo utente, se sta lavorando con gli interrupt disabilitati o attivi: il sistema operativo realtime sottostante è sempre e comunque in grado di rispondere opportunamente agli interrupt.

RTLinux scinde dunque i meccanismi del kernel generico da quelli del kernel realtime, in modo che entrambi possano essere ottimizzati secondo le rispettive

finalità, e facendo sì che il kernel realtime possa essere mantenuto semplice e compatto [3W, 6].

RTLinux è progettato in modo che il suo kernel RT non debba mai essere lasciato in attesa della liberazione di risorse da parte di Linux. Il kernel RT non richiede memoria, né condivide strutture dati, se non in situazioni strettamente controllate. I canali di comunicazione usati per trasferire dati tra i processi realtime e quelli eseguiti da Linux sono di tipo “non-blocking” dal lato RT: non esiste alcuna situazione in cui un processo realtime debba attendere per trasmettere o ricevere dati. Per la comunicazione RTLinux fornisce sia *shared memory* sia particolari *device* (*RealTime-FIFO*) tramite apposite entry nella directory */dev*. Dal punto di vista del programmatore, una FIFO appare come un normale dispositivo a carattere, accessibile mediante le classiche funzioni *read()*, *write()*, *open()* e *ioctl()*. La memoria condivisa è accessibile tramite la funzione POSIX *mmap()*.

Una delle direttive principali nella progettazione di RTLinux è demandare quanto più possibile a Linux e il restante al kernel RT: Linux ha il compito di gestire le procedure di inizializzazione del sistema (perché non esiste alcuna esigenza particolare di realtime in fase di avvio), nonché ogni risorsa non condivisibile (perché qualsiasi flusso di esecuzione che possa essere bloccato in mancanza della particolare risorsa necessaria non è sicuramente realtime). Per esempio un programma realtime non può assolutamente effettuare le chiamate *malloc()* o *kmalloc()* o qualsiasi altra funzione di allocazione dinamica della memoria: un processo realtime dispone solo di un’allocazione statica.

Per finire, RTLinux si affida al meccanismo di caricamento dei moduli kernel di Linux per installare i processi nel sistema realtime: caricare un modulo

realtime non è un'operazione realtime e può quindi essere lasciata senza problemi a Linux. Il compito del kernel RT è solo quello di fornire ai programmi realtime accesso diretto al nudo hardware, in modo che possano avere una latenza minima e il massimo potere computazionale. Tutto il resto è solo d'intralcio.

4.4.3 Il driver del dispositivo

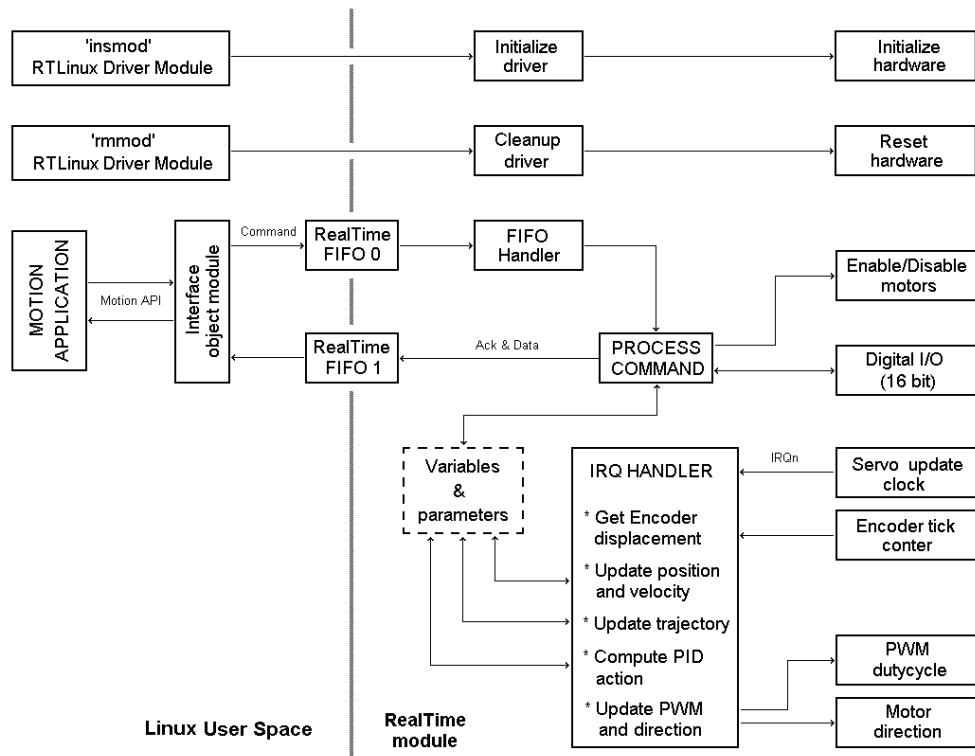
La necessità di avere RTLinux come sistema operativo deriva dalla decisione di non utilizzare un chip dedicato al controllo dei motori, ma di demandare ad un processo realtime la gestione dell'anello di controllo. L'impiego di un processo realtime è essenziale poiché, nella nostra applicazione, i ritardi che possono subire i normali gestori di interrupt di Linux non sono accettabili (paragrafo 4.4.1). Infatti, in presenza di ritardi non predicibili, sarebbe impossibile determinare con precisione la reale velocità di rotazione del motore considerato, essendo questa calcolata come differenza delle posizioni assunte dal relativo encoder tra due interrupt successivi. Inoltre, a causa della necessità di aggiornare regolarmente la variabile di forzamento, la presenza di questi ritardi introducerebbe instabilità nell'anello di controllo. Poiché nel nostro circuito il periodo di tempo che intercorre tra due interrupt è di 1.6 millisecondi, il kernel “standard” di Linux non è in grado di garantire la corretta gestione delle interruzioni.

Questo ci ha “costretti” a ricorrere all’impiego della versione realtime di Linux, ma questa scelta si è dimostrata vincente: non solo ci ha consentito di realizzare un dispositivo semplice ed efficiente, che non richiede l’uso di microcontrollori (nei quali la cui la messa a punto di programmi complessi necessita di costosi emulatori), ma ha anche facilitato l’acquisizione e l’analisi dei dati per il debugging, grazie alla grande mole di software disponibile in Linux. Inoltre

l'aver realizzato la parte di controllo come processo software ci consente una flessibilità enorme: volendo è possibile cambiare facilmente strategia di controllo (ad esempio utilizzare la logica fuzzy, magari sfruttando codice già disponibile per Linux), oppure è possibile realizzare facilmente un sistema adattativo che varia la strategia di controllo in funzione delle condizioni operative. Solo la fantasia pone limiti: la difficoltà risiede nel riscrivere una funzione software.

Si noti che l'impiego di RTLinux non determina praticamente alcun calo delle prestazioni del sistema per i normali processi Linux. Il tempo impiegato dalla CPU per la gestione del controllo di basso livello risulta assolutamente trascurabile, essendo di circa 20÷25 microsecondi ogni interrupt, cioè l'1.5% di tempo di CPU: gran parte di questo tempo (circa 16 microsecondi) è dovuto alla lentezza del bus ISA nelle operazioni di I/O (ogni istruzione di I/O impiega circa 1 microsecondo); una efficienza maggiore si otterrebbe progettando una scheda PCI, ma questa realizzazione non è alla portata di un hobbista, non solo per le alte frequenze in gioco, ma soprattutto per la difficoltà di reperire sul mercato hobbistico dei chip di interfaccia (il bus PCI è infatti notevolmente più complesso del bus ISA, sia a livello hardware, sia a livello software, dato che qualsiasi scheda PCI deve avere funzionalità plug-and-play).

Il nostro driver si appoggia dunque al kernel realtime, ed essendo costituito da processi realtime ha un funzionamento leggermente differente dai normali driver in Linux. Per la sua realizzazione abbiamo fatto riferimento alla documentazione disponibile per RTLinux (www.rtlinux.org) e a quella per la progettazione di moduli [Pomerantz, 1999]. Infatti il driver è costituito essenzialmente da un modulo, anche se per comodità d'uso l'intero codice è stato suddiviso in due parti, come mostrato dalla figura seguente:

**Figura 4. 21 - Schema a blocchi del driver.**

In particolare abbiamo:

- **un modulo realtime**, che si fa carico di leggere la posizione degli encoder e variare le grandezze di controllo applicate alla scheda di potenza (il duty cycle del PWM e la direzione del motore). Questo modulo utilizza due FIFO realtime, per dialogare con i processi Linux. Ad ogni comando inviato sulla prima FIFO, viene invocato un gestore realtime nel modulo, il quale, se possibile, esegue il comando immediatamente, oppure lo schedula sul prossimo interrupt aggiornando opportunamente alcune variabili (vedere più avanti). La seconda FIFO viene utilizzata per passare i dati dal modulo realtime all'applicazione Linux (occorrono infatti due FIFO per uno scambio bidirezionale);

- **un file oggetto**, che, linkato ad una applicazione Linux, fornisce una serie di comandi (sotto forma di funzioni) al controllo di alto livello del robot. Questi comandi sono del tipo "dammi la posizione corrente", "riportami la velocità del motore 1", "stabilisci la velocità massima e l'accelerazione", "abilita il motore 2", ecc. Le funzioni dialogano con il modulo realtime attraverso le due FIFO descritte: il loro scopo è quello di fornire una semplice interfaccia software (*API*) alle applicazioni, rendendo completamente trasparente la gestione del driver RTLinux al software di controllo di livello superiore (vedere paragrafo 3.1).

Sia l'hardware realizzato, sia il driver software sono stati pensati non solo per la robotica mobile, ma per l'utilizzo negli azionamenti in genere. Per questo l'interfaccia software fornisce la possibilità di gestire oltre alla velocità, anche la posizione dell'asse dei motori. Comunque questa possibilità è di scarsa utilità nel controllo di un robot mobile, per l'impossibilità di definire precisamente la traiettoria. Infatti nel controllo di posizione, la velocità viene impostata automaticamente dal driver mediante un generatore di profili trapezoidali, che determina un'accelerazione costante fino al raggiungimento della velocità massima ammissibile e poi una decelerazione costante fino al raggiungimento di una velocità nulla nella posizione desiderata.

La parte più importante del driver è quella costituita dal modulo realtime. In esso sono presenti due processi, che comunicano tra loro attraverso alcune variabili condivise, allocate staticamente nello spazio del kernel RT:

- **un “FIFO-Handler”**: è un processo realtime, che viene attivato non appena una applicazione Linux invia un comando al driver, cioè non appena nella FIFO associata all'handler (FIFO0 nello schema) vengono scritti dei

caratteri tramite l'API di interfaccia. Il processo realtime invoca un *gestore dei comandi* che controlla la validità del dato e lo processa: se il comando è eseguibile immediatamente senza disturbare l'anello di controllo dei motori, esso viene eseguito (per esempio la lettura della posizione corrente); qualora il comando non sia eseguibile immediatamente (come l'impostazione di una nuova velocità “desiderata”), esso viene schedulato sull'interrupt successivo. Il gestore dei comandi invia poi all'applicazione che ha invocato il comando un segnale di *ack*, (acknowledgment) per evidenziare che il comando è stato riconosciuto e che è (o sarà) processato al più presto. Alcuni comandi che possono essere eseguiti immediatamente riportano dei dati all'applicazione. Come per l'*ack* questo avviene attraverso la seconda FIFO (indicata nello schema come FIFO1). Questo processo realtime può essere interrotto dal gestore di interrupt (descritto di seguito), ma non dai processi Linux.

- **Un “IRQ Handler”:** è il processo realtime più importante, quello che ha il compito di servire gli interrupt provenienti dalla nostra scheda ISA e gestire l'anello di controllo dei motori: come tale esso non deve subire interruzioni da altri processi (ne realtime, né Linux). I compiti del gestore delle interruzioni consistono nelle seguenti operazioni:
 1. verificare che non vi sia uno stato di overcurrent che perduri più di quanto specificato dall'applicazione: se questo accade, entrambi i motori vengono disabilitati;
 2. vedere se vi sono richieste in sospeso da parte del gestore dei comandi; se esiste un comando pendente questo viene eseguito: ciò potrebbe condizionare le operazioni successive;

3. leggere gli encoder mediante i contatori associati (paragrafo 4.3.1);
4. aggiornare la posizione corrente e calcolare la velocità di rotazione dell'asse dei motori;
5. aggiornare la posizione “impostata” (nel controllo di posizione) o la velocità “impostata” (nel controllo di velocità) tramite il generatore di profili trapezoidali;
6. calcolare il valore della grandezza di forzamento tramite una legge PID (paragrafo 3.1.2);
7. aggiornare i contatori preposti alla generazione dei segnali PWM.

Per quanto riguarda gli algoritmi utilizzati, i due più importanti risiedono nel gestore delle interruzioni e riguardano:

- **la legge di controllo PID:** riflette l'equazione alle differenze data nel paragrafo 3.1.3, banalmente codificabile;
- **il generatore di profili trapezoidali di velocità e posizione:** riportiamo nella figura 4.22 il diagramma di flusso relativo alla generazione del profilo nel controllo di velocità (quello che maggiormente interessa per il controllo di basso livello del robot); per il profilo relativo al controllo di posizione (decisamente più complesso) si rimanda a [MICROCHIP, 1999] e al codice in appendice.

Un'ultima nota riguarda i calcoli: abbiamo utilizzato l'aritmetica intera ed impiegato alcuni “shift” per ottenere una parte decimale in virgola fissa. Questo è stato necessario perchè l'uso delle operazioni in virgola mobile, nello spazio del kernel realtime, non è normalmente consentito, a meno di utilizzare funzioni che salvano e ripristinano in continuazione lo stato della FPU (*Floating Point*

Unit) interna al processore. Ma nel nostro caso ciò avrebbe comportato un maggiore “overhead” computazionale, senza peraltro apportare significativi vantaggi.

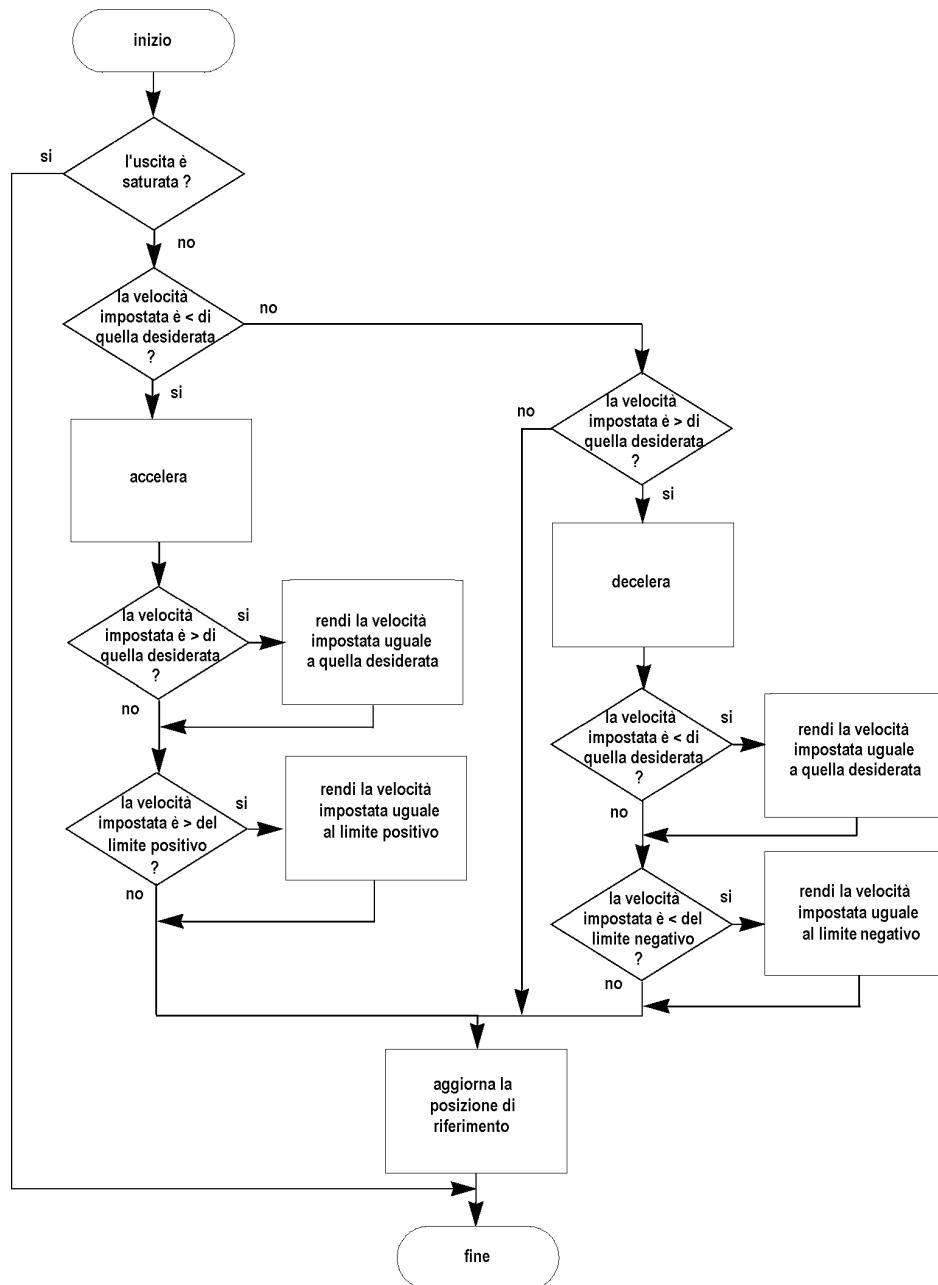


Figura 4.22 – Generazione del profilo di velocità.

4.5 Conclusioni

In questo lavoro abbiamo prima esaminato una serie di questioni teoriche.

Nel primo capitolo abbiamo passato in rassegna le tipologie di veicoli mobili, gli attuatori per il movimento, i sensori che normalmente accompagnano un robot mobile. Nel secondo capitolo abbiamo analizzato la dinamica e la cinematica del robot. Nel terzo, infine, ci siamo occupati del controllo e degli azionamenti di potenza, con particolare enfasi sui sistemi a commutazione.

Partendo dalle analisi svolte nei capitoli 1, 2 e 3, siamo passati alla descrizione della fase realizzativa del robot mobile nel capitolo 4.

La prima scelta riguardava la tipologia del veicolo e, come spiegato al capitolo 1, abbiamo optato per un veicolo su ruote in configurazione “differential drive” (paragrafo 1.2.1), cioè dotato di due ruote motrici attuate in modo autonomo.

Abbiamo poi scelto la classe di motori, constatando che una coppia di servomotori in CC a magneti permanenti avrebbe reso il giusto rapporto prestazioni/prezzo.

La struttura meccanica del veicolo si è andata pian piano delineando nell’ambito delle necessità della squadra ART (Azzurra Robot Team), il team italiano di robot calciatori che partecipa alle competizioni internazionali della RoboCup.

La RoboCup (Robot World-Cup Soccer) è un'iniziativa per promuovere la ricerca sull'intelligenza artificiale e sulla robotica al fine di valutarne applicazioni ed architetture in uno scenario comune, il gioco del calcio (appendice A6).

Come in ogni robot della squadra, anche nel nostro veicolo è stato inserito un sistema di elaborazione basato su un PC standard (paragrafo 4.1). La parte di movimentazione è stata invece realizzata in modo differente: anziché utilizzare una base commerciale della ActivMedia Robotics, condivisa dalla maggior parte degli altri robot (Pioneer I e II), abbiamo optato per una base autocostruita, nell'ottica di realizzare in futuro basi più complesse con maggiori gradi di libertà.

Abbiamo quindi progettato un sistema di movimentazione basato su una scheda ISA di controllo, una scheda di potenza e un driver in RTLinux (paragrafi 4.2 e 4.3).

Abbiamo scelto il bus ISA, per la maggiore facilità di costruzione in proprio, senza dover ricorrere a costosi prototipi prodotti in fabbrica. Se, come sembra dalle sperimentazioni finora effettuate, i risultati si dimostreranno validi, in futuro si riprogetterà l'elettronica adattandola allo standard PC/104 ed utilizzando componenti SMD.

La scheda di potenza è stata progettata in modo ibrido: parte con circuiti integrati e parte a componenti discreti. La scheda è infatti in grado di operare con correnti fino ad 8.5 ampere per motore (25 ampere sostituendo i MOSFET con tipi più potenti). Grande impegno è stato posto per ridurre il più possibile i disturbi elettromagnetici generati dal pilotaggio PWM.

La scheda ISA ha un ruolo semplice: leggere gli encoder e generare i segnali PWM; tutto questo è stato realizzato tramite una serie di contatori a 16 bit. La gestione dell'anello di controllo è infatti demandata completamente ad una routine software, implementata come processo realtime.

Al fine di assicurare il rispetto dei vincoli temporali, richiesti dall'anello di controllo, si è reso necessario l'utilizzo di RTLinux come sistema operativo. Abbiamo visto al paragrafo 4.4 che questo rende possibile utilizzare lo stesso PC sia per compiti di basso livello che per compiti di più alto livello, essendo il tempo di CPU dedicato al controllo dei motori assolutamente trascurabile.

In sintesi l'aver implementato l'anello di controllo come processo realtime in RTLinux, ci ha permesso di mantenere semplice l'hardware e ci ha semplificato il debugging del driver, permettendoci di ottenere buoni risultati in tempi abbastanza brevi.

Il robot non è ancora pronto per giocare. È privo, infatti, dei sensori adeguati: un altro gruppo di studio sta sviluppando un sistema di visione binoculare. Si spera che il robot possa giocare nell'imminente campionato, che si terrà a Seattle, negli Stati Uniti, nei primi di agosto di quest'anno.

Per terminare riportiamo alcuni grafici di simulazione di movimento. La simulazione considera il modello dinamico ricavato al paragrafo 2.1, con i parametri ottenuti al paragrafo 4.2. Viene applicata ai motori una tensione di 24V a gradino nell'istante $t = 0$, con corrente limitata 3.5 ampere, valore che determina una coppia massima prossima, ma inferiore, a quella di slittamento. I grafici evidenziano la posizione e la velocità lineare del robot in funzione del tempo.

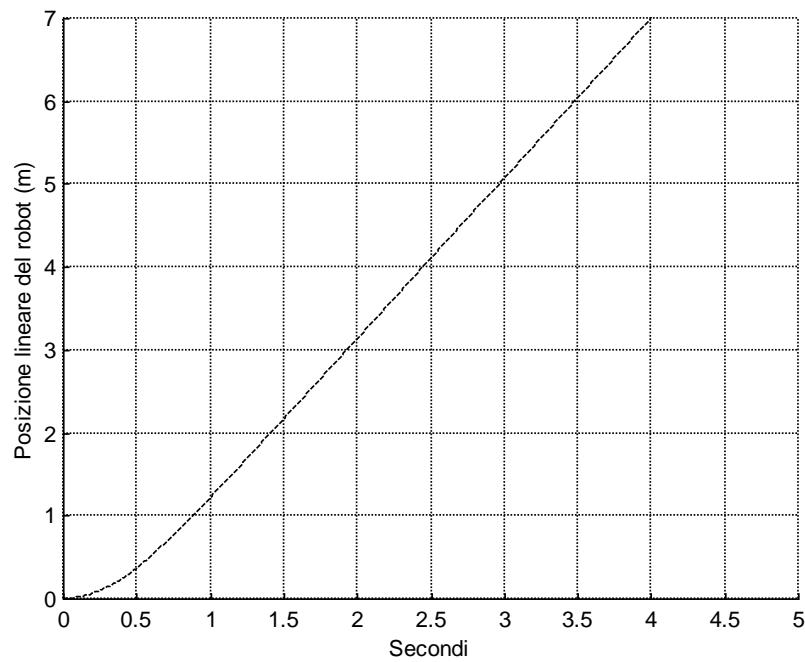


Figura 4.23 – Simulazione dello spazio percorso dal robot.

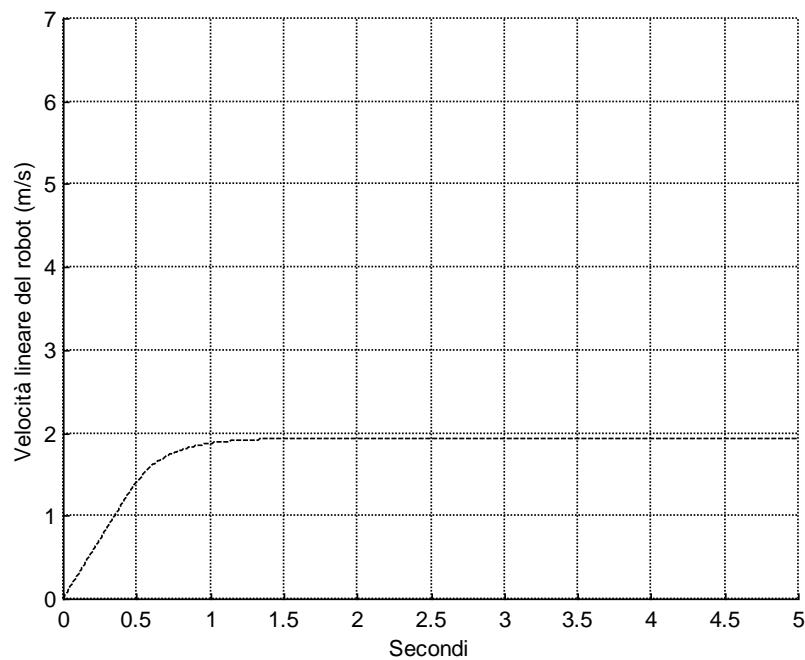


Figura 4.24 – Simulazione della velocità del robot.

Appendici

Appendice A1

SCHEMI ELETTRICI

Di seguito riportiamo gli schemi elettrici dei circuiti realizzati, con le relative liste dei materiali.

Nell'ordine abbiamo:

1. **Scheda ISA**
2. **Scheda di potenza**

ISA DC Motor Control Board – Rev 1.1

(C) 2000, 2001 by Flavio Cappelli

Bill of Material

Qty	Part Type	Designator
<hr/>		
8	2.7K 1/4W	R6 R7 R8 R9 R10 R11 R12 R13
1	3.3K 1/4W	R4
5	10K 1/4W	R1 R2 R3 R16 R17
2	4.7K 1/4W	R14 R15
1	4.7M 1/4W	R5
2	22pF	C10 C11
4	56pF	C16 C17 C18 C19
11	100nF	C3 C4 C5 C6 C7 C8 C9 C12 C13 C14 C15
2	4.7uF EL. 16V	C1 C2
1	Crystal 10MHz	Q1
1	74HC00	U7
2	74HC14	U10 U11
1	74HC4060	U6
1	74LS245	U1
2	82C54	U3 U4
1	82C55A	U5
1	GAL16V8 (ADDR)	U2
2	GAL16V8 (ENCOD)	U8 U9
1	DB9	J1

1	DB15	J2
1	HEADER 10X2	J3
1	HEADER 5X2	J4
4	JUMPER 3X2	JP1 JP2 JP3 JP4
4	JUMPER 3x1	JP5 JP6 JP7 JP8

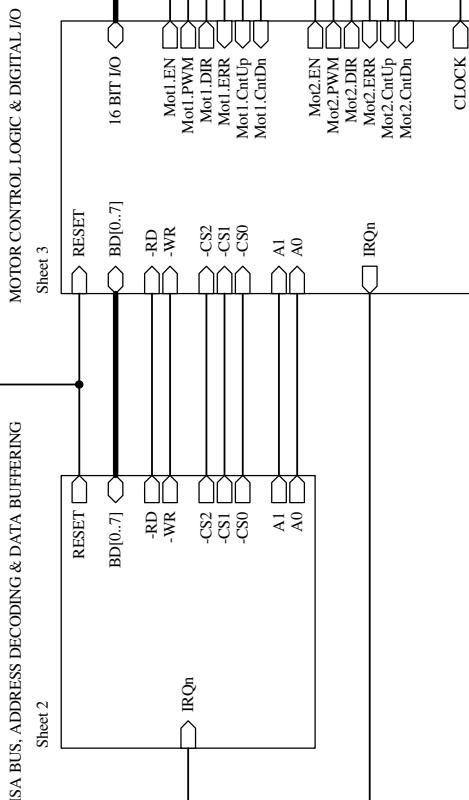
ATTENZIONE

Prima di procedere al collaudo e quindi all'inserzione in uno slot del PC, controllare che non vi siano cortocircuiti tra le piste, in particolare tra VCC (pin B3 dello slot ISA) e GND (pin B1 dello slot ISA). Un cortocircuito tra questi due terminali potrebbe provocare la DISTRUZIONE DELLA SCHEDA MADRE DEL PC a causa delle alte correnti fornite dall'alimentatore.

ISA DC Motor Control Board

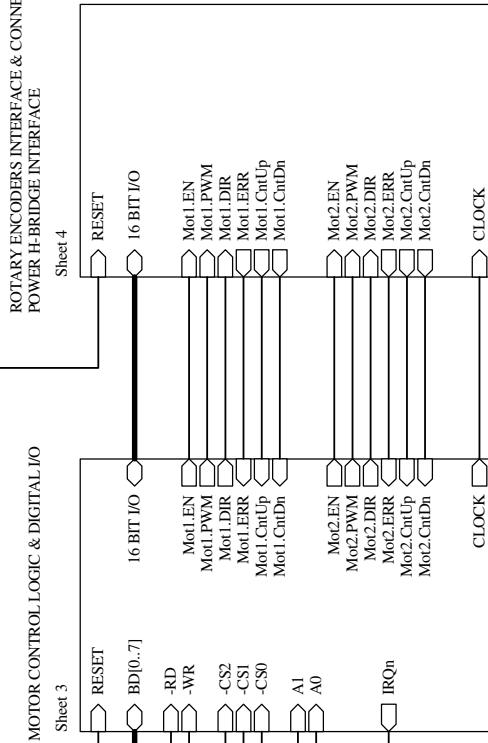
ISA BUS, ADDRESS DECODING & DATA BUFFERING

Sheet 2



ROTARY ENCODERS INTERFACE & CONNECTORS,
POWER H-BRIDGE INTERFACE

Sheet 4



(C) by Flavio Cappelli

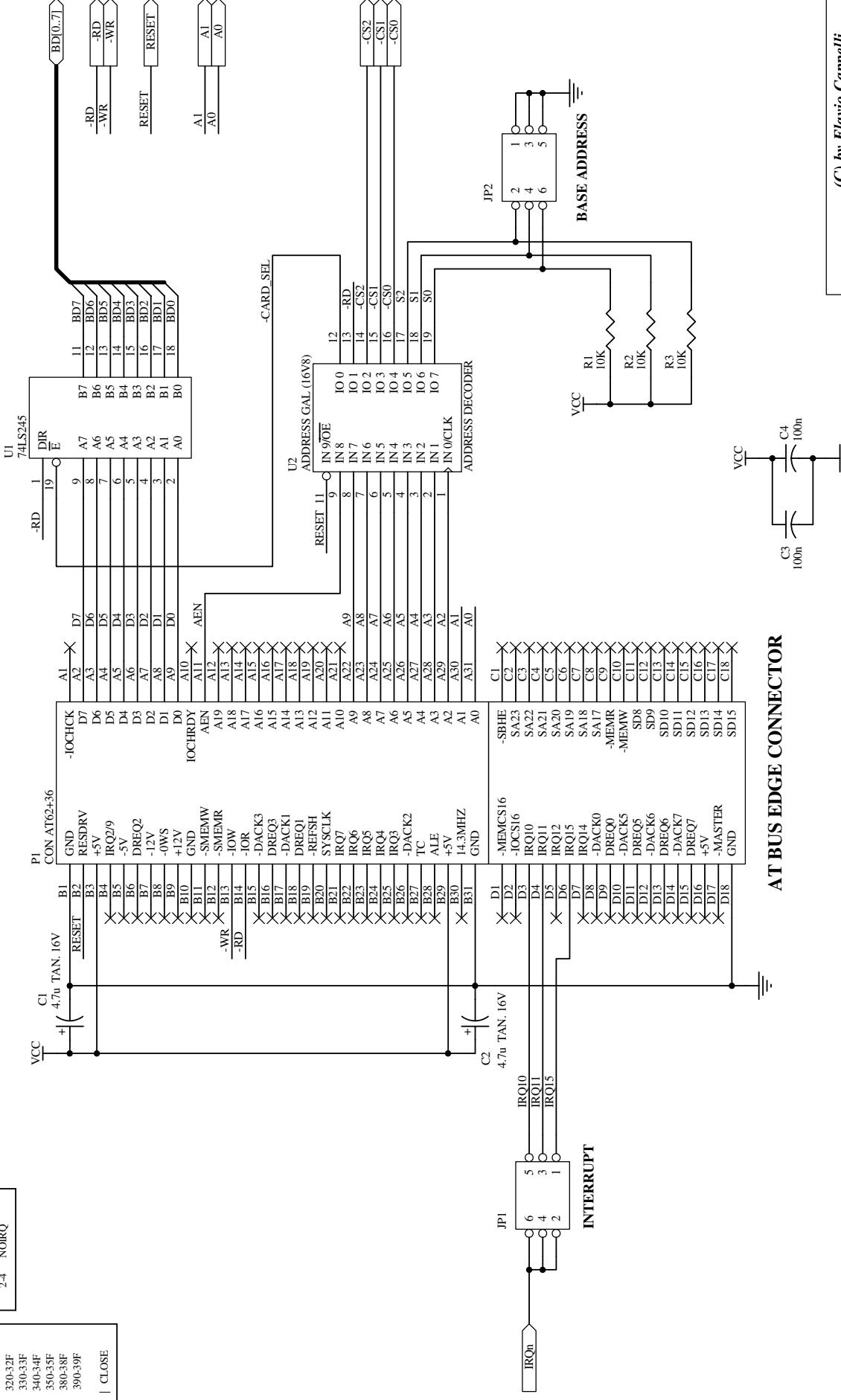
Title ISA DC Motor Control Board

Size: A4 Number: None Revision: 1.1

Date: 21-Mar-2001 Time: 10:51:09 Sheet 1 of 4

BASE ADDRESS (JP1)		INTERRUPT (JP1)	
1.2	3-4 5-6	RANGE	1-2 IRQ 15 3-4 IRQ 11 5-6 IRQ 10 2-4 NORQ

ISA BUS, ADDRESS DECODING & DATA BUFFERING



(C) by Flavio Cappelli

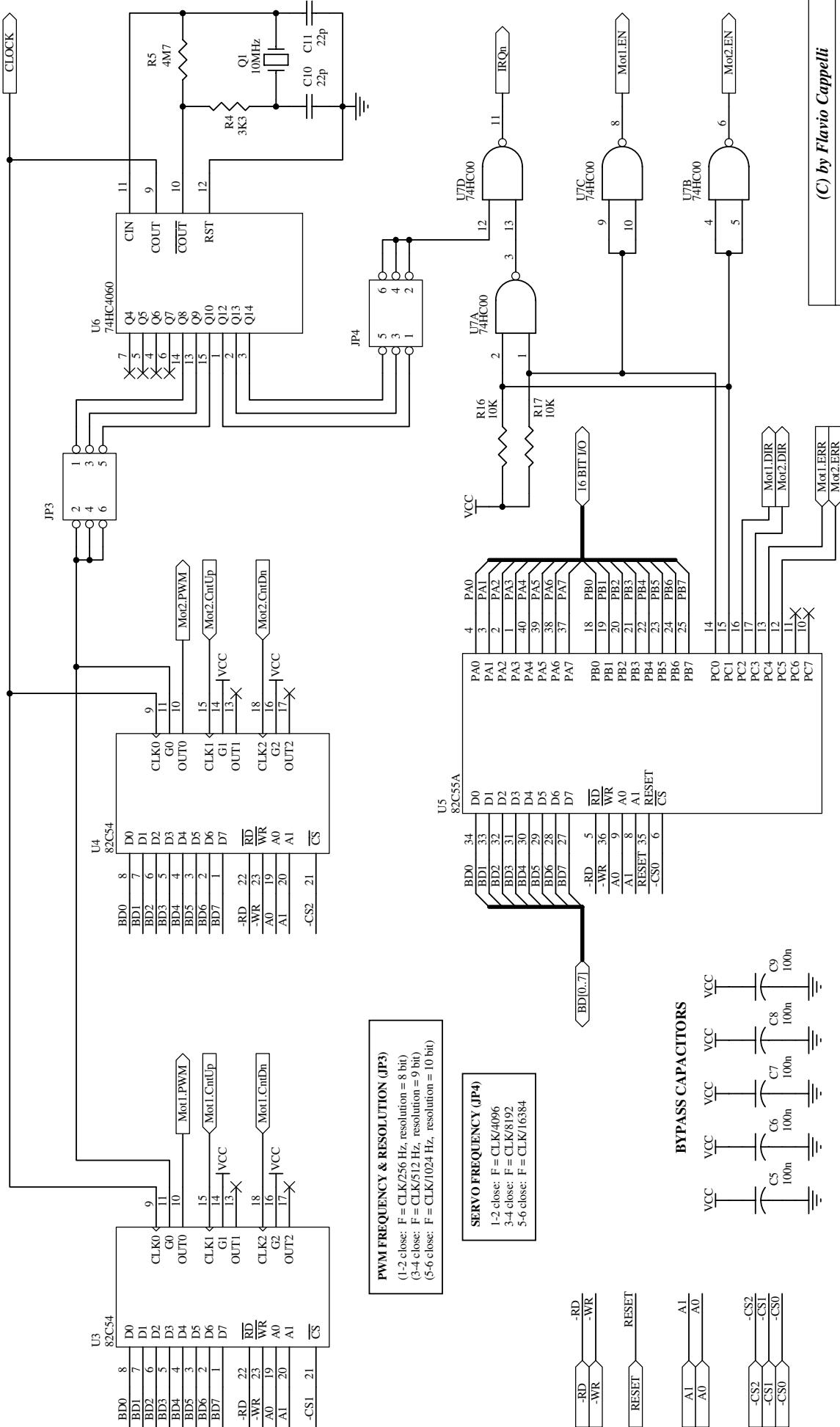
Title ISA DC Motor Control Board

Revision: 1.1

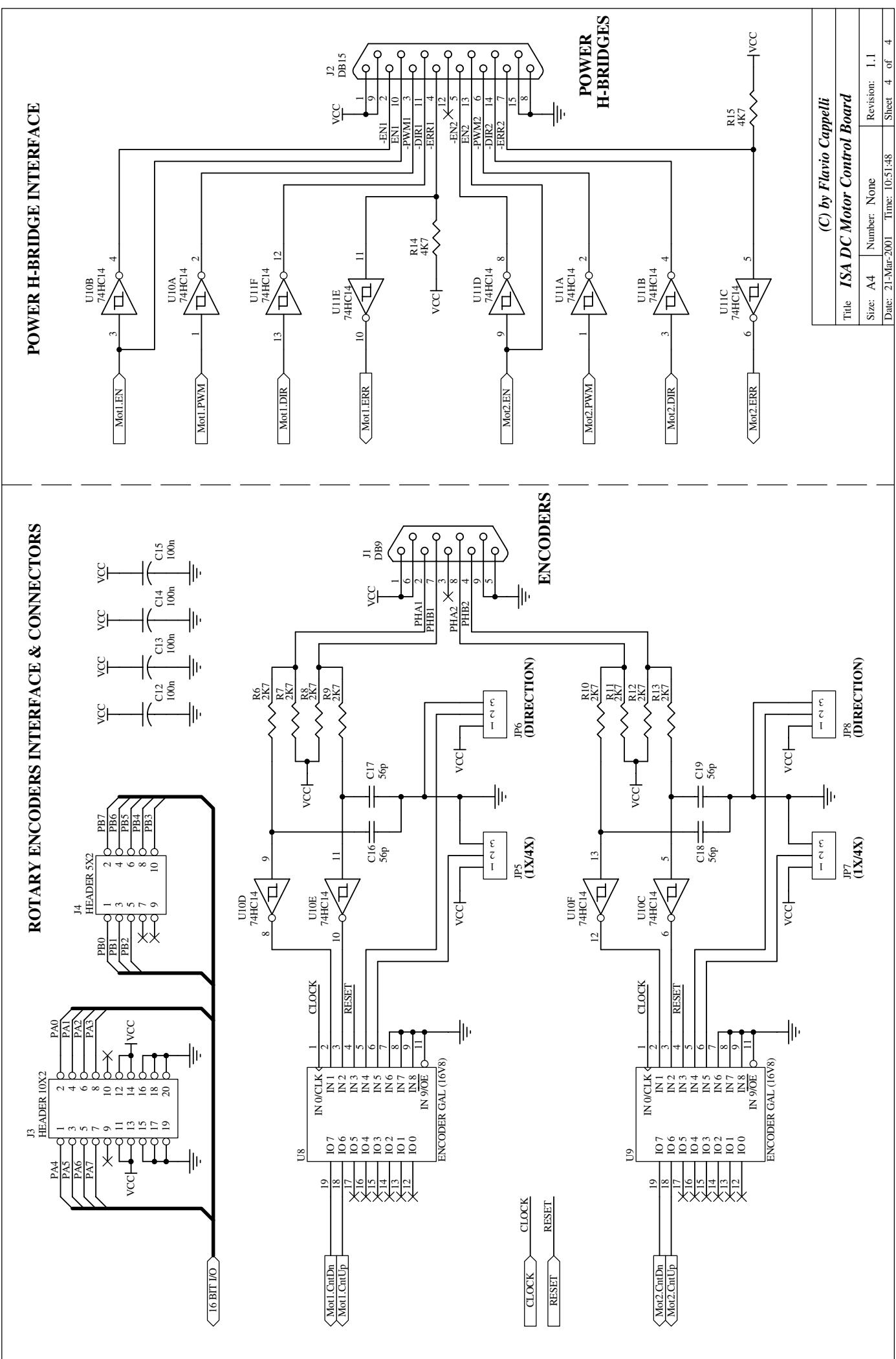
Size: A4 Number: None

Date: 21-Mar-2001 Time: 10:51:23 Sheet 2 of 4

MOTOR CONTROL LOGIC & DIGITAL I/O



PWM FREQUENCY & RESOLUTION (JP3)



H-Bridge DC Motor Driver – Rev 1.1

(C) 2000, 2001 by Flavio Cappelli

Bill of Material

Qty	Part Type	Designator
<hr/>		
4	0.1 ohm 4W	R23 R24 R38 R39
8	10 ohm 1/4W	R19 R20 R21 R22 R34 R35
		R36 R37
2	220 ohm 1/4W	R10 R14
2	330 ohm 1/4W	R7 R11
6	470 ohm 1/4W	R1 R2 R3 R4 R5 R6
2	1K 1/4W	R8 R12
2	1K Precision Trimmer	R29 R44
2	3.3K 1/4W	R25 R40
2	4K 1/4W	R28 R43
6	4.7K 1/4W	R9 R13 R26 R27 R41 R42
8	100K 1/4W	R15 R16 R17 R18 R30 R31
		R32 R33
4	1nF	C18 C19 C34 C35
2	10nF	C12 C28
15	100nF	C2 C6 C8 C9 C10 C20 C22 C23 C24 C25 C36 C38 C39 C40 C41
4	100nF CER. 100V	C13 C14 C29 C30
4	1uF CERAMIC	C11 C26 C27 C42
3	10uF EL. 16V	C1 C21 C37

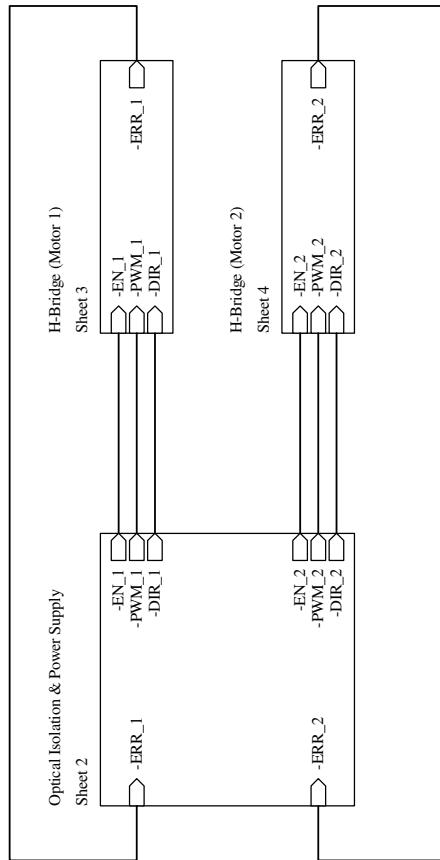
1	22uF LOW ESR 25V	C5
1	47uF EL. 16V	C7
1	220uF EL. 63V	C3
1	330nF	C4
4	470uF LOW ESR 63V	C16 C17 C32 C33
1	1N4007	D1
2	RED LED	D3 D5
2	YELLOW LED	D2 D4
4	uF4002	D6 D7 D8 D9
8	IRF520N	Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8
1	BA10T	U1
1	7805	U2
6	TLP521-1	U3 U5 U6 U7 U9 U10
2	6N137	U4 U8
2	74HC00	U12 U17
2	74HC14	U11 U16
2	74HC123A	U13 U18
2	HIP4081A	U14 U19
2	LM393	U15 U20
1	S10K20 (VARISTOR)	V1
1	DB15	J1
1	4 PIN CONNECTOR	J2
2	2 PIN CONNECTOR	J3 J4
4	JUMPER 3x1	JP1 JP2 JP3 JP4
1	BALUN (*)	T1

(*) Induttanza bilanciata (a 4 terminali) ottenuta avvolgendo 10 spire doppie di filo di rame smaltato (diametro 1.6 mm) su nucleo Mn-Zn ad elevata permeabilità ($AL = 6110 \text{ nH}$, diametro interno 13.7 mm). Tale nucleo è reperibile da RS COMPONENTS, codice 232-9561. I due avvolgimenti inseriti nei conduttori di andata e di ritorno generano due flussi magnetici opposti che presentano bassa impedenza alla corrente di carico e elevata impedenza ai segnali spuri di rumore.

ATTENZIONE

Gli otto MOSFET devono essere dotati di dissipatore adeguato. E' possibile utilizzare un unico dissipatore (va bene ad esempio quello di un processore Pentium II) isolandolo dal corpo dei MOSFET mediante piastrine di mica e pasta al silicone.

H-Bridge DC Motor Driver

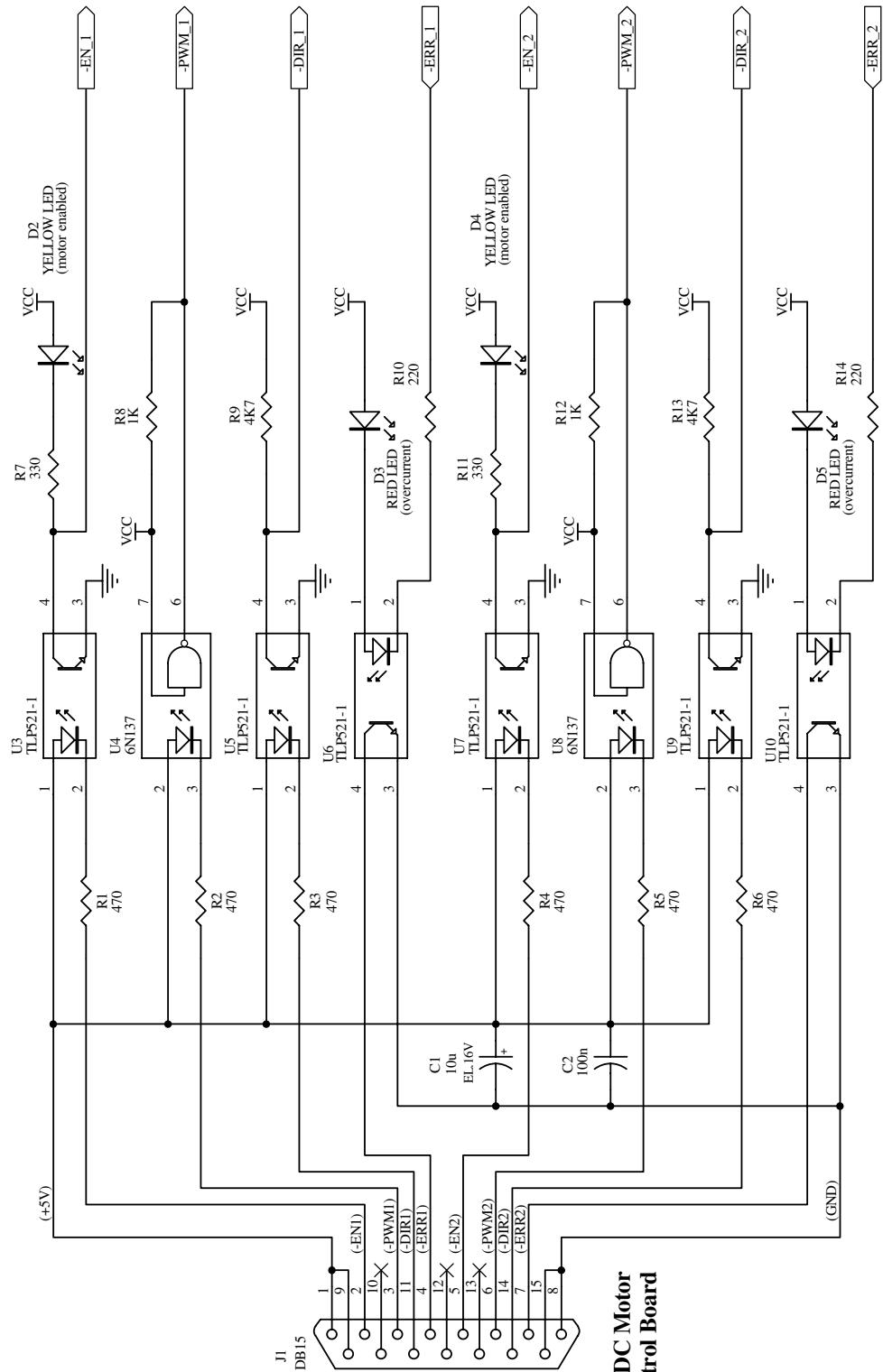


(C) by Flavio Cappelli

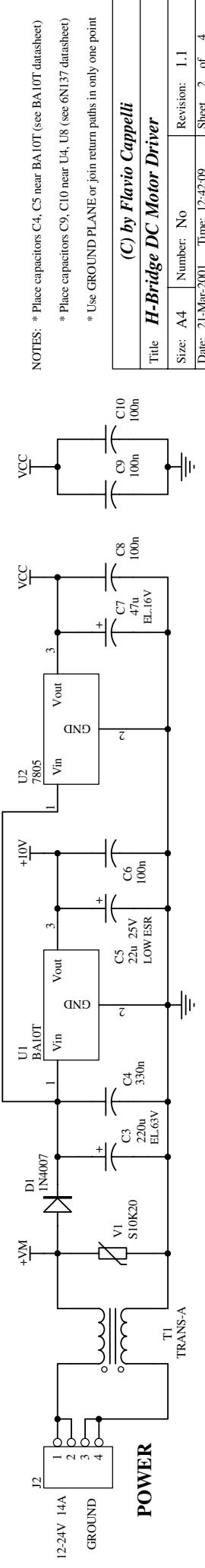
(C) by Flavio Cappel
dae DC Motor Driver

Title ***n-brige DC motor Driver***
 Size: A4 Number: No Revision: 1.1
 Date: 21-Mar-2001 Time: 12:41:57 Sheet 1 of 4

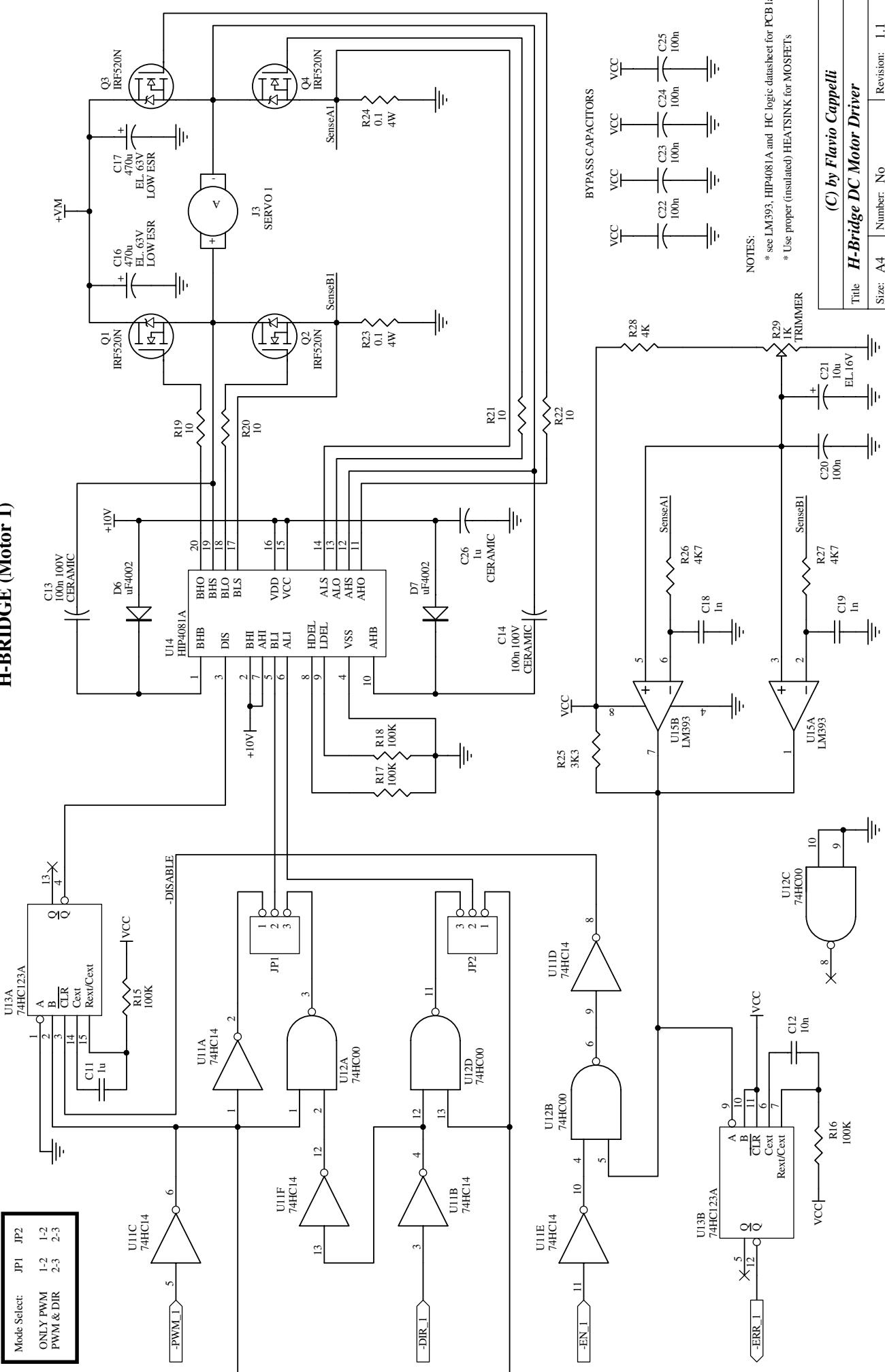
OPTICAL ISOLATION & POWER SUPPLY



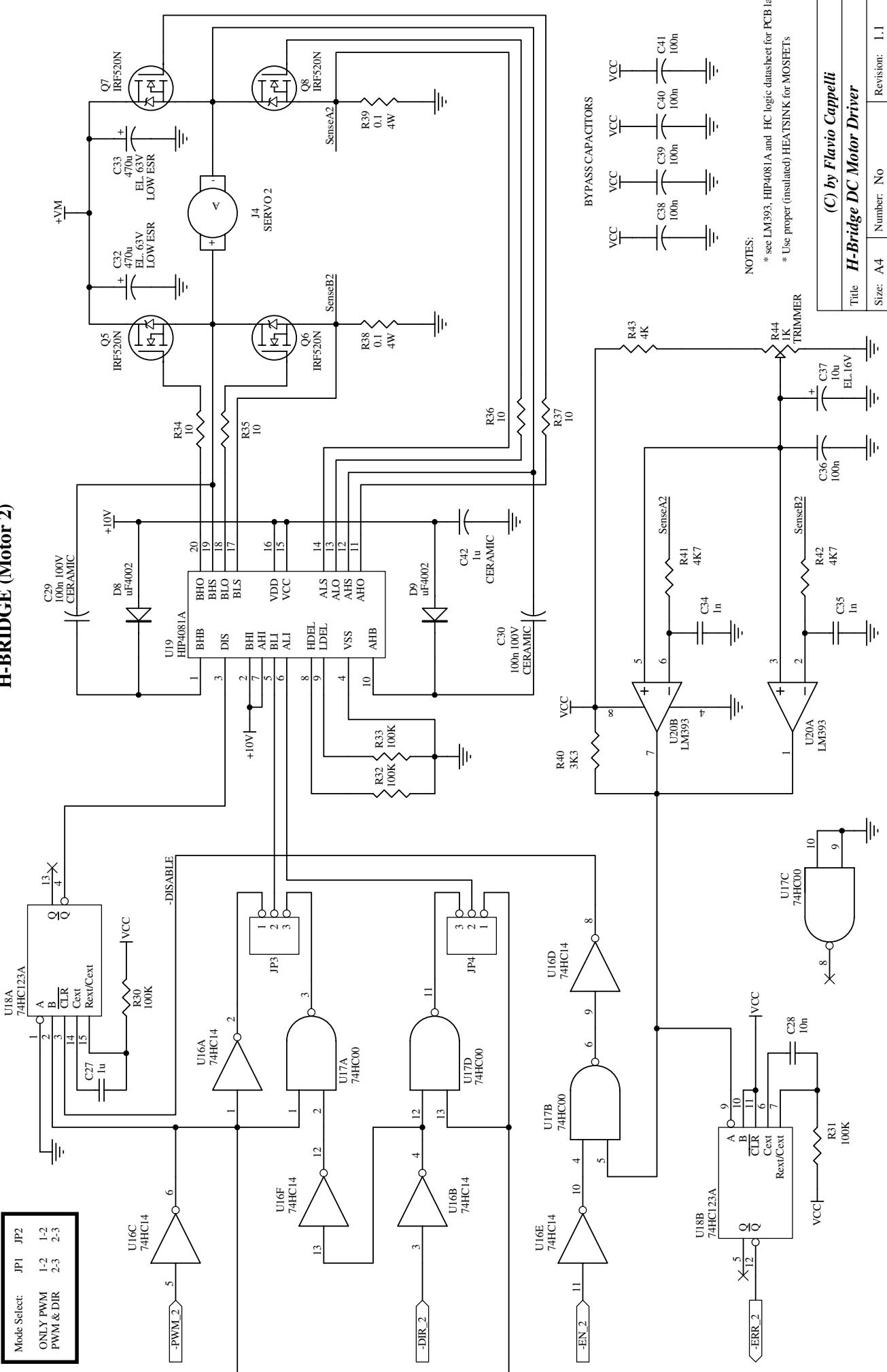
NISA DC Motor
Control Board



H-BRIDGE (Motor 1)



H-BRIDGE (Motor 2)



Appendice A2

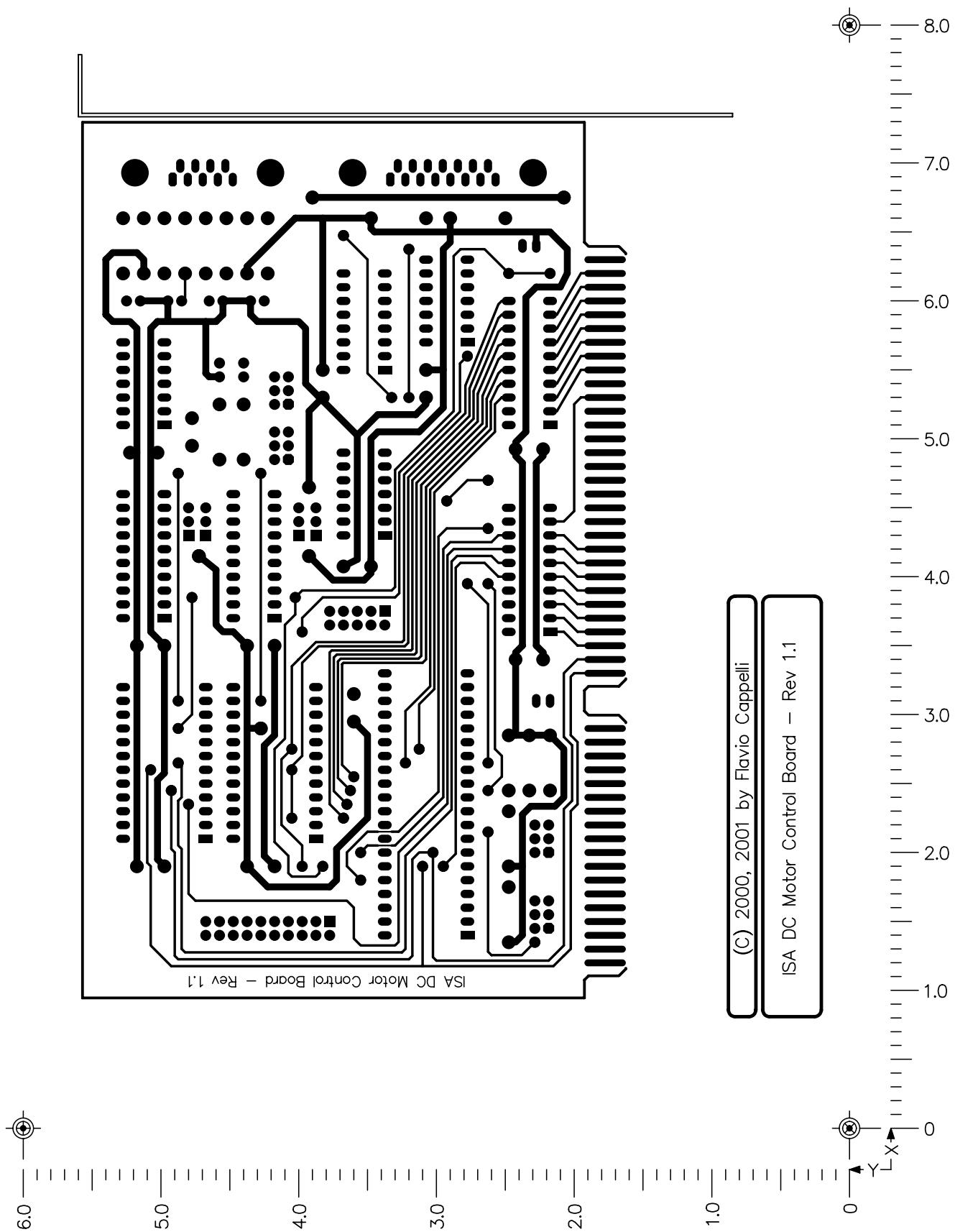
CIRCUITI STAMPATI

Di seguito riportiamo tutta la documentazione per la produzione dei circuiti stampati (PCB).

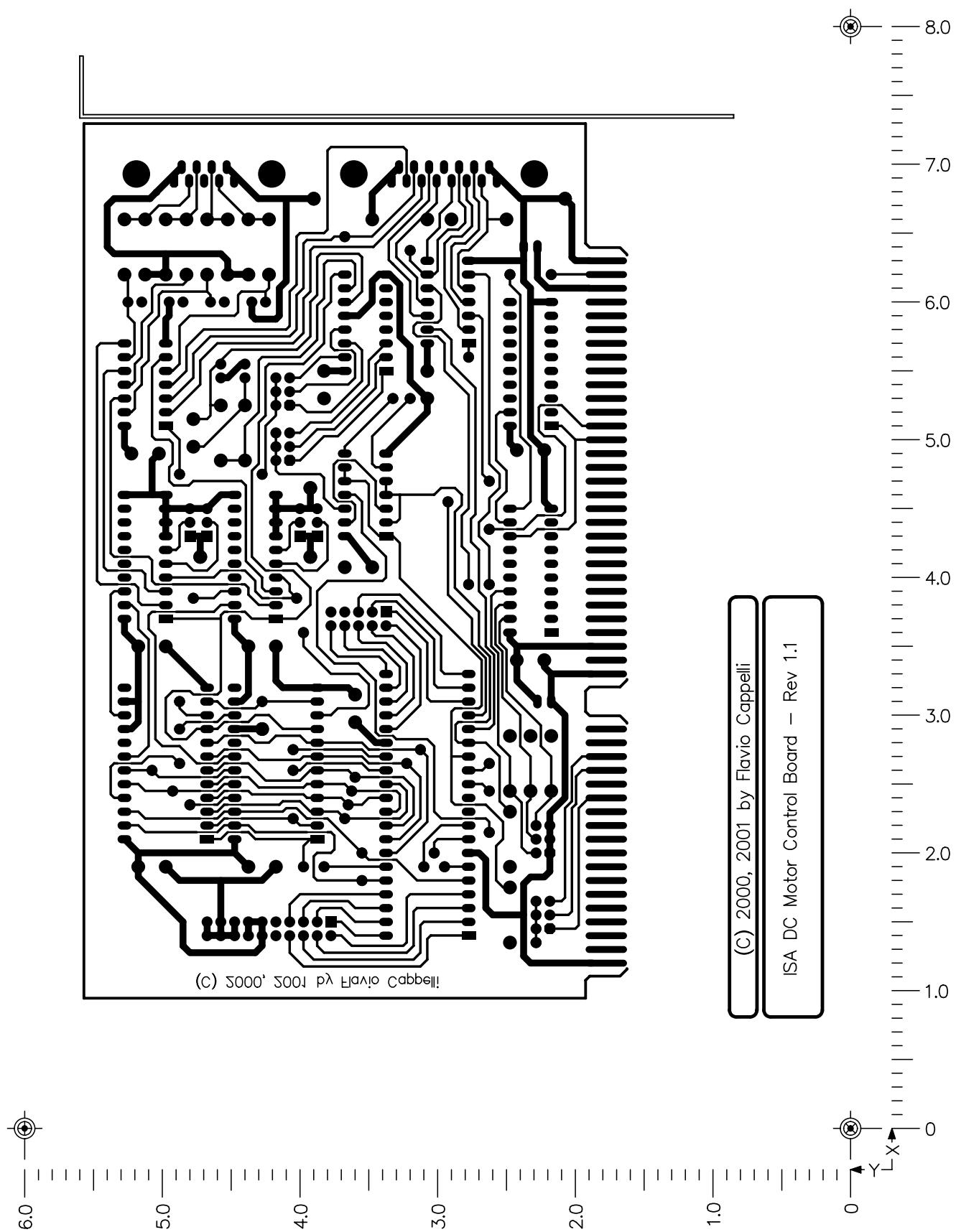
Nell'ordine abbiamo:

- 1. Scheda ISA**
- 2. Scheda di potenza**

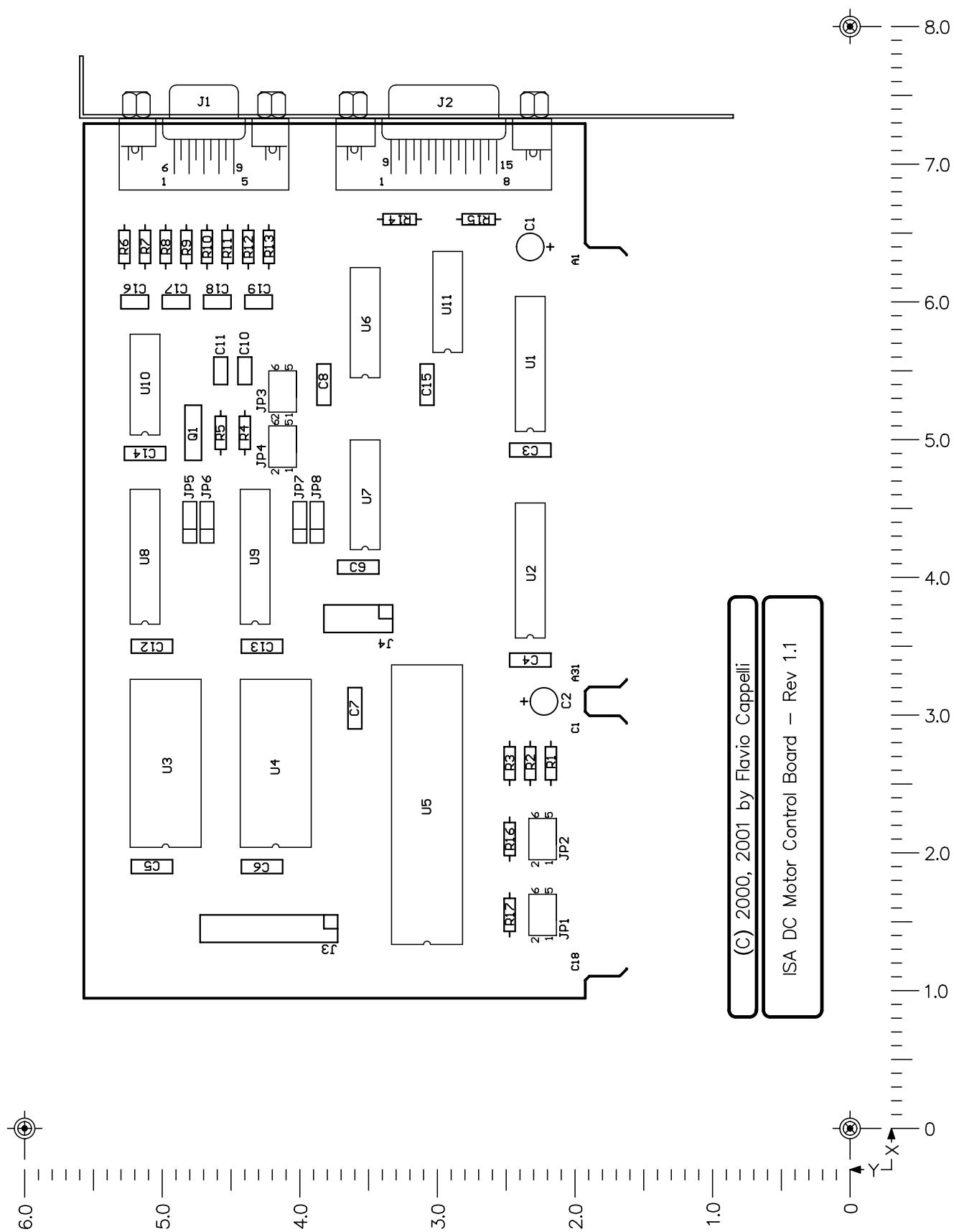
Scheda ISA, TOP LAYER
(vista lato componenti)



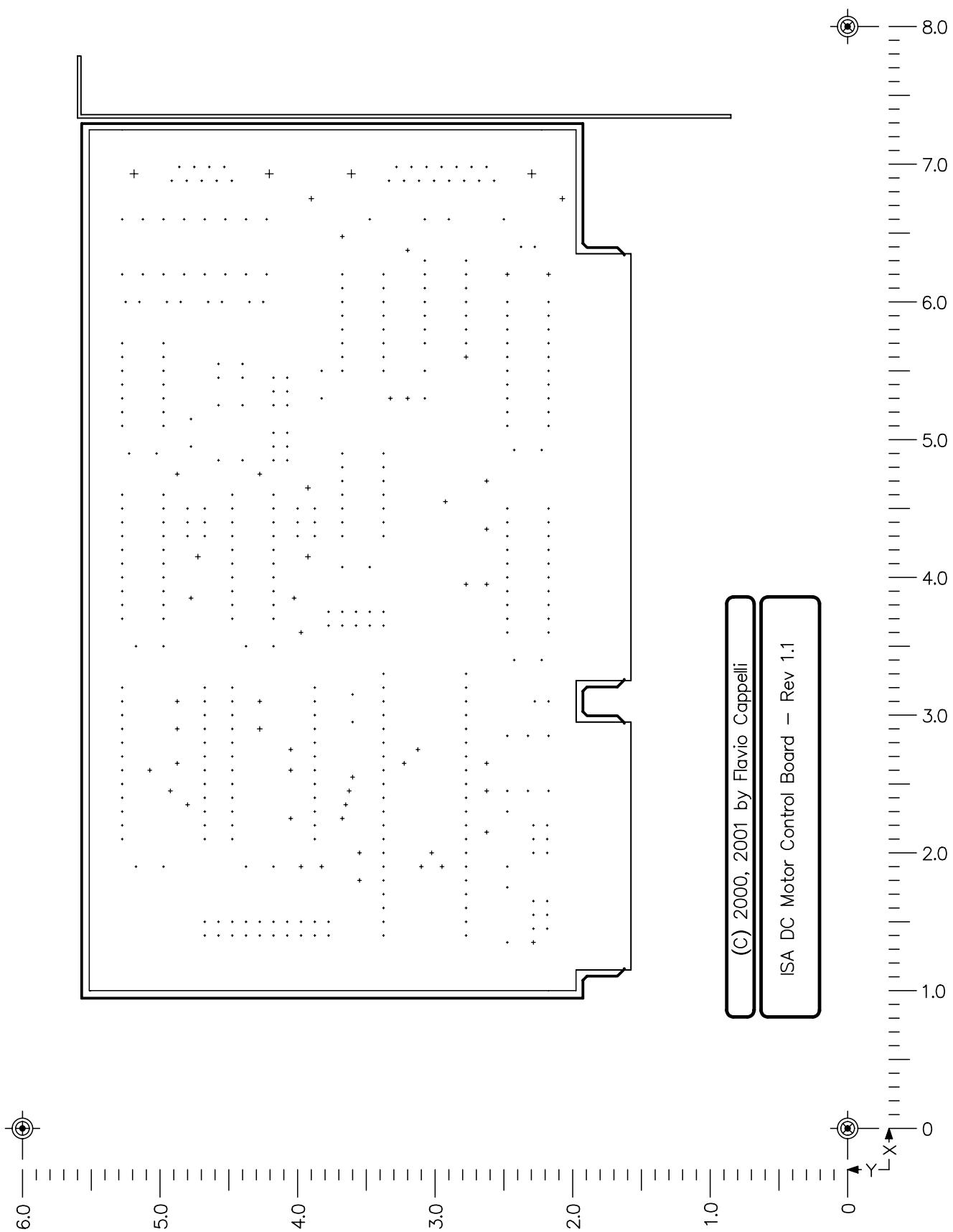
**Scheda ISA, BOTTOM LAYER
(vista lato componenti)**



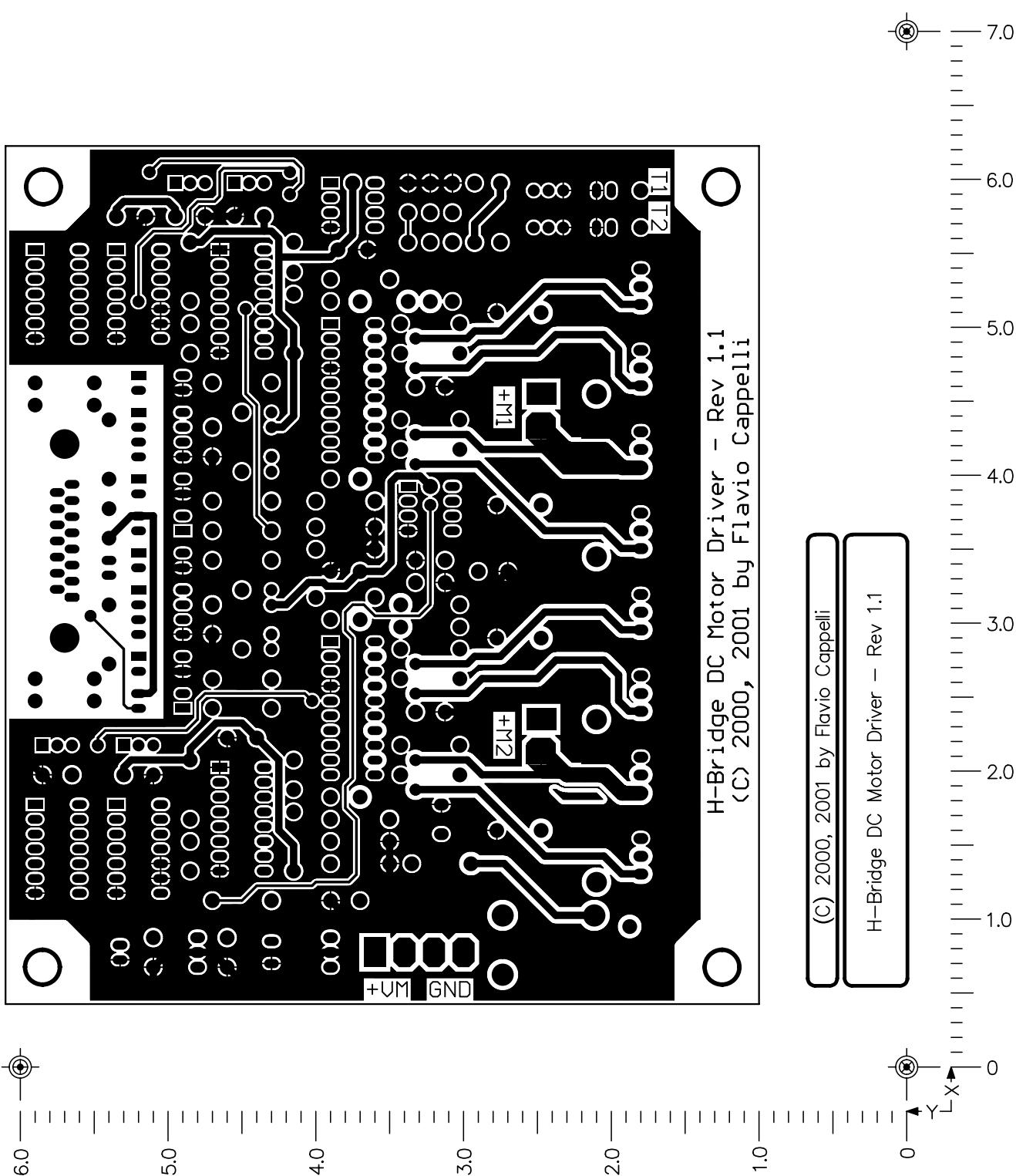
Scheda ISA, SILKSCREEN OVERLAY



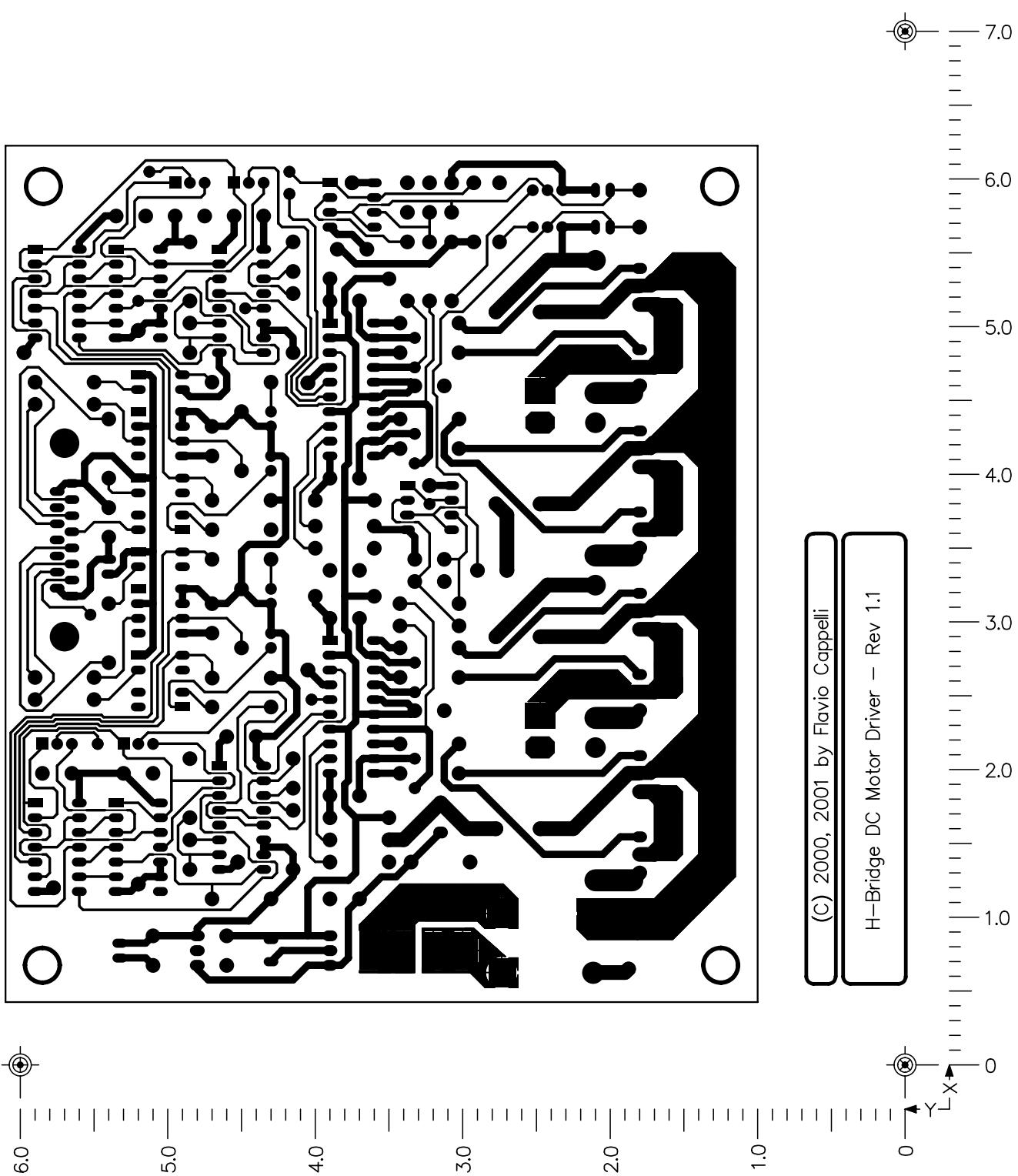
Scheda ISA, DRILL GUIDE



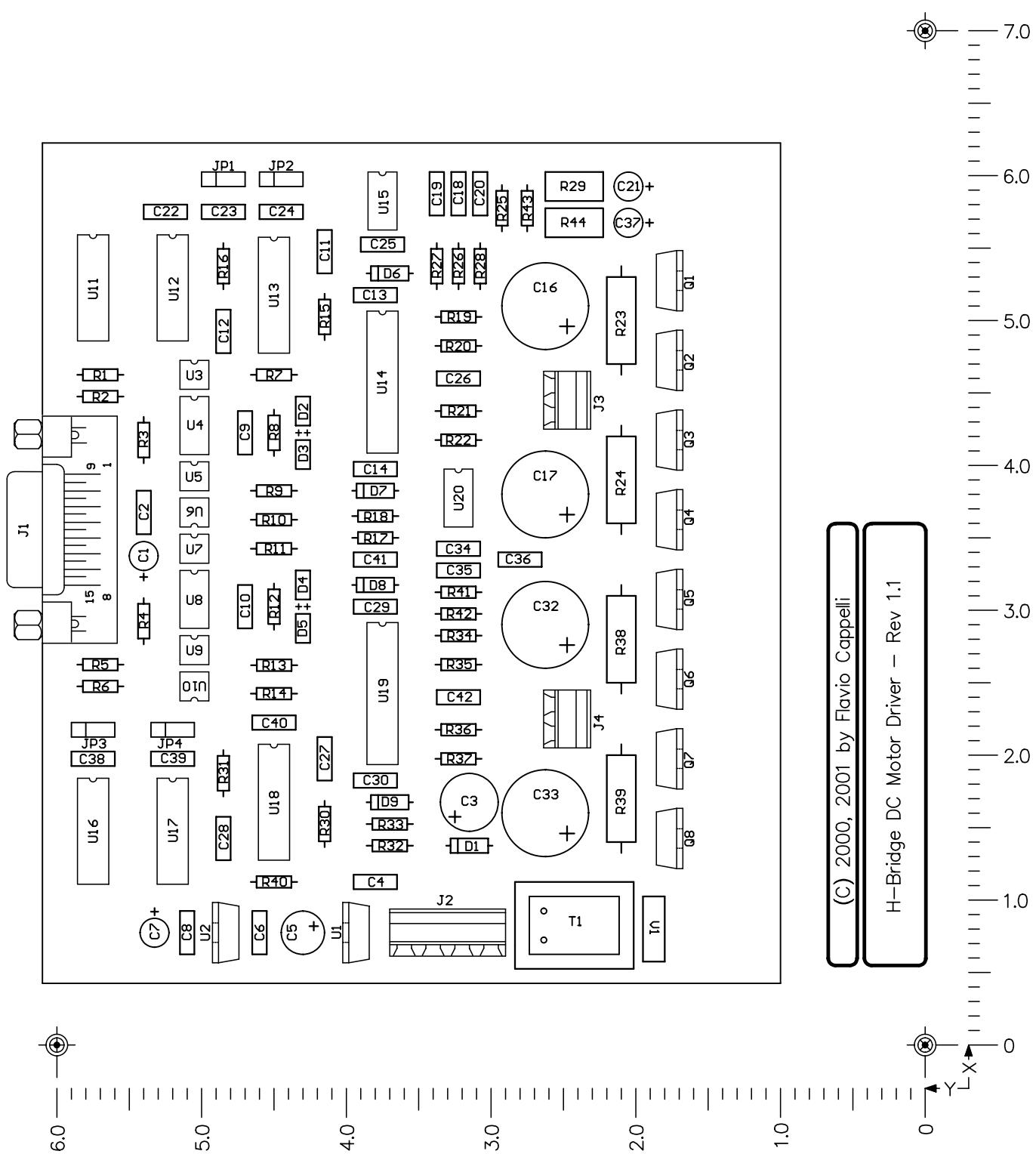
**Scheda di Potenza, TOP LAYER
(vista lato componenti)**



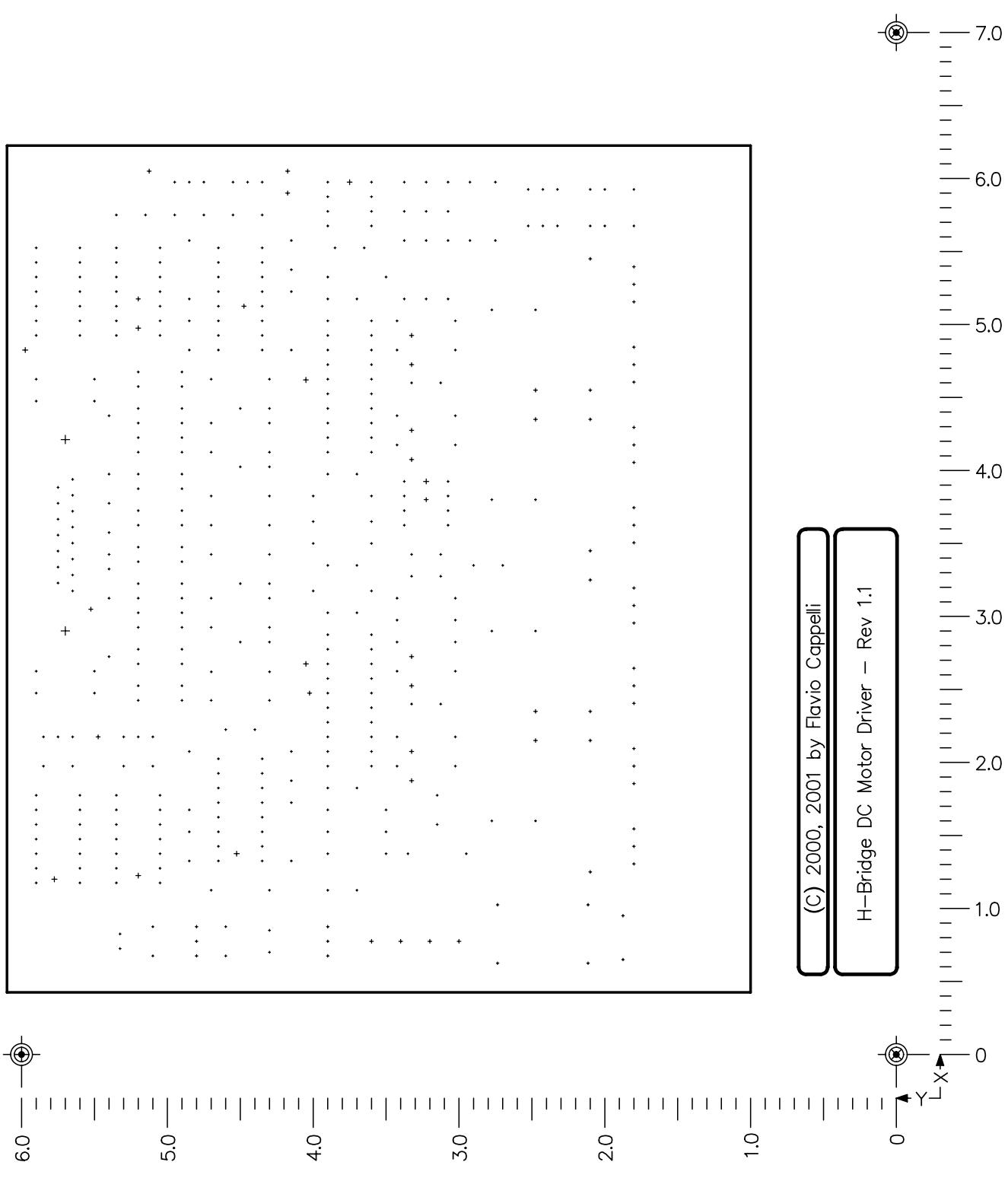
Scheda di Potenza, BOTTOM LAYER
(vista lato componenti)



Scheda di Potenza, SILKSCREEN OVERLAY



Scheda di Potenza, DRILL GUIDE



Appendice A3

FUNZIONI LOGICHE (GAL)

Di seguito riportiamo tutta la documentazione per la programmazione dei dispositivi GAL16V8. Le funzioni logiche sono espresse in CUPL, linguaggio diffuso nell'ambito delle logiche programmabili [3W, 8]. Vengono anche forniti i files JEDEC corrispondenti.

Nell'ordine abbiamo:

- 1. Decodifica degli indirizzi del bus ISA**
- 2. Trasformazione dei segnali degli encoders.**

Progetto e realizzazione di una base robotica mobile

```
/*
 *          ISA ADDRESS DECODER GAL.
 */
/*
 *      Version 1.0 (C) 2000 by Flavio Cappelli
 *          GAL16V8, CUPL Logic Compiler
 */
/*
 * Gal Mode: SIMPLE (in this mode, pin 15 and 16 cannot be used as input)
 */
/*
 * Pins:
 * Outputs.....: !CARD_SEL, !CS0,!CS1,!CS2
 * Inputs.....: AEN, A9,A8,A7,A6,A5,A4,A3,A2, !RD, S2,S1,S0, RESET
 * Power Supply Pins: VCC, GND
 */
/*
 * This device decodes the PC adress bus and generates enable signals for TIMERS
 * and PPI.The base address is set by comparing A9 - A4 of the ISA bus against 3
 * bits derived from jumpers JP2 as the following table:
 */
/*
 *          1-2   3-4   5-6       RANGE
 */
-----
/*
 *      .     .     .       280-28F
 */
/*
 *      .     .     x       290-29F
 */
/*
 *      .     x     .       320-32F
 */
/*
 *      .     x     x       330-33F
 */
/*
 *      x     .     .       340-34F
 */
/*
 *      x     .     x       350-35F
 */
/*
 *      x     x     .       380-38F
 */
/*
 *      x     x     x       390-39F
 */
/*
 * When a selected range is decoded and adress enable (AEN) is low the CARD_SEL
 * signal is generated. Address bits A3 & A2 are used to select devices as:
 */
/*
 *      CARD_SEL & A3 = 0 & A2 = 0 => CS0 (PPI 82C55)
 */
/*
 *      CARD_SEL & A3 = 0 & A2 = 1 => CS1 (TIMER 82C54 #1)
 */
/*
 *      CARD_SEL & A3 = 1 & A2 = 0 => CS2 (TIMER 82C54 #2)
 */
/*
 *      CARD_SEL & A3 = 1 & A2 = 1 => CS1 + CS2 (both Timers)
 */
/*
 * TO AVOID BUS CONFLICT LAST MODE IS AVAILABLE ONLY TO WRITE REGISTERS.
 */
*****
```

NAME AddrGal;
DATE 03-Feb-2000;
PARTNO ;
REVISION 1.0;
DESIGNER Flavio Cappelli;
COMPANY ;
ASSEMBLY ISA DC Motor Control Board;
LOCATION U2;
DEVICE g16v8c;

```
/*
 * Input Pin Declarations
 */
*****
```

PIN [1..8]	= [A2..9];	/* System Address A9-A2 */
PIN 9	= AEN;	/* 0/1 = CPU/DMA Address */
PIN 11	= RESET;	/* 0/1 = Normal Op/Reset */
PIN 13	= !RD;	/* I/O Read Strobe */
PIN [17..19]	= [S2..0];	/* Jumper Address RANGE */
		/* PIN 10 (GND) to GND */
		/* PIN 20 (VCC) to +5V */

```
/*
 * Output Pin Declarations
 */
*****
```

PIN 12	= !CARD_SEL;	/* CARD Select */
PIN 14	= !CS2;	/* 82C54 #2 CHIP Select */
PIN 15	= !CS1;	/* 82C54 #1 CHIP Select */
PIN 16	= !CS0;	/* 82C55 CHIP Select */

Progetto e realizzazione di una base robotica mobile

```
/*
 * Intermediate Variable Definitions
 */
/*
 * CUPL Logic Equations
 */
/*
 * CARD_SEL = !AEN & ((Switches:0 & CardAddr:'h'28X) #  
 *                      (Switches:1 & CardAddr:'h'29X) #  
 *                      (Switches:2 & CardAddr:'h'32X) #  
 *                      (Switches:3 & CardAddr:'h'33X) #  
 *                      (Switches:4 & CardAddr:'h'34X) #  
 *                      (Switches:5 & CardAddr:'h'35X) #  
 *                      (Switches:6 & CardAddr:'h'38X) #  
 *                      (Switches:7 & CardAddr:'h'39X)) & !RESET;  
  
CS0      = CARD_SEL & !A3 & !A2;          /* A3 A2 | Chip Select */  
CS1      = CARD_SEL & A2 & !(A3 & RD);    /* 0 0 | CS0 */  
CS2      = CARD_SEL & A3 & !(A2 & RD);    /* 0 1 | CS1 */  
                  /* 1 0 | CS2 */  
                  /* 1 1 | CS1+CS2 */  

*/
```

Progetto e realizzazione di una base robotica mobile

```
ADVANCED PLD      4.0  Serial# MW-67999999
Device           g16v8cas  Library DLIB-h-36-1
Created          dom mar 11 12.00.22 2001
Name             AddrGal
Partno
Revision        1.0
Date            03-Feb-2000
Designer        Flavio Cappelli
Company
Assembly        ISA DC Motor Control Board
Location        U2
*QP20
*QF2194
*QV108
*G0
*F0
*L00768 10101111111111111111111101111
*L01024 1101111111111111111111110111101111
*L01056 1001111111111111111111111111111101111
*L01280 0111111111111111111111111111111101111
*L01312 0110111111111111111111111111111101111
*L01792 11111010101010100111101101111010
*L01824 111110101101010100111101101111010
*L01856 1111101001011010101110111101111010
*L01888 1111010101011010101110111101111010
*L01920 11111010101001011011011101111010
*L01952 1111101011010010110110111101111010
*L01984 111110101001100101110111101111010
*L02016 111101011001100101110111101111010
*L02112 0000000011100010111111111111111111
*L02144 111111111111111111111111111111111111
*L02176 111111111111111111111111111111111111
*C2DB4
```

Progetto e realizzazione di una base robotica mobile

```
/*
 *          QUADRATURE ENCODER GAL.
 */
/*
 *      Version 1.1 (C) 2000 by Flavio Cappelli
 *      GAL16V8, CUPL Logic Compiler
 */
/*
 * Gal Mode: REGISTER.
 */
/*
 * Pins:
 * Outputs.....: CntUp, CntDn
 * Inputs.....: CLK (Reg.Mode), PHIA, PHIB, RESET, DIR, 4X, !OE (Reg.Mode)
 * Auxiliar Outputs.: P1Q, P2Q, P1D, P2D, UP, COUNT
 * Power Supply Pins: VCC, GND
 */
/*
 * NAME          QuadGal;
 * DATE          11-Jan-2000;
 * PARTNO        ;
 * REVISION      1.1;
 * DESIGNER      Flavio Cappelli;
 * COMPANY        ;
 * ASSEMBLY      ISA DC Motor Control Board;
 * LOCATION       U8, U9;
 * DEVICE         g16v8c;

/*
 * Input Pin Declarations
 */
/*
 * PIN 1 = CLK;           /* Register Mode: PIN 1 (CLK) to CLOCK */
 * PIN 2 = PHIA;          /* Encoder Phase A (0) */
 * PIN 3 = PHIB;          /* Encoder Phase B (PI/2) */
 * PIN 4 = RESET;         /* 0/1 for Normal/Reset operation */
 * PIN 5 = DIR;           /* 0/1 for Normal/Reverse direction */
 * PIN 6 = 4X;            /* 0/1 for 1X/4X decoding */
 * PIN 11 = !OE;          /* Register Mode: PIN 11 (!OE) to GND */
 *                         /* Power Supply: PIN 10 (GND) to GND */
 *                         /* Power Supply: PIN 20 (VCC) to +5V */
 */
/*
 * Output Pin Declarations
 */
/*
 * PIN 12 = P1Q;
 * PIN 13 = P2Q;
 * PIN 14 = P1D;
 * PIN 15 = P2D;
 * PIN 16 = UP;
 * PIN 17 = COUNT;
 * PIN 18 = CntUp;        /* UP COUNT Encoder Tick */
 * PIN 19 = CntDn;        /* DOWN COUNT Encoder Tick */
 */
/*
 * CUPL Logic Equations
 */
/*
 * P1Q.d      = !RESET & PHIA & !DIR
 *               # !RESET & PHIB &  DIR;
 * P2Q.d      = !RESET & PHIA &  DIR
 *               # !RESET & PHIB & !DIR;
 *
 * P1D.d      = P1Q & !RESET;
 * P2D.d      = P2Q & !RESET;
 */
```

Progetto e realizzazione di una base robotica mobile

```
!UP.d      =  P2Q  &  P1Q  &  P2D
#  P2Q  &  !P1Q  &  !P1D
#  !P2Q  &  P1Q  &  P1D
#  !P2Q  &  !P1Q  &  !P2D
#  RESET;

!COUNT.d = !RESET  &  P1Q  &  P1D  &  !P2Q  &  P2D  &  4X
#  !RESET  &  !P1Q  &  !P1D  &  P2Q  &  !P2D
#  !RESET  &  !P1Q  &  P1D  &  !P2Q  &  !P2D  &  4X
#  !RESET  &  P1Q  &  !P1D  &  P2Q  &  P2D  &  4X
#  !RESET  &  P1Q  &  P1D  &  P2Q  &  !P2D  &  4X
#  !RESET  &  !P1Q  &  !P1D  &  !P2Q  &  P2D
#  !RESET  &  !P1Q  &  P1D  &  P2Q  &  P2D  &  4X
#  !RESET  &  P1Q  &  !P1D  &  !P2Q  &  !P2D  &  4X;

CntUp.d   =  !COUNT  &  UP;
CntDn.d   =  !COUNT  &  !UP;

/********************************************/
```

Progetto e realizzazione di una base robotica mobile

```
ADVANCED PLD      4.0  Serial# MW-67999999
Device           g16v8cms  Library DLIB-h-36-1
Created          dom mar 11 12.10.05 2001
Name             QuadGal
Partno
Revision        1.1
Date            11-Jan-2000
Designer        Flavio Cappelli
Company
Assembly        ISA DC Motor Control Board
Location        U8, U9
*QP20
*QF2194
*QV1056
*G0
*F0
*L00000 111111111101110111111111111111
*L00256 111111111101101111111111111111
*L00512 1111111110111111010111011101101
*L00544 1111111110111111111011101101110
*L00576 1111111110111111011011011101110
*L00608 1111111110111111010111011011101
*L00640 1111111110111111011011011101101
*L00672 1111111110111111110111011101110
*L00704 1111111110111111010111011101110
*L00736 11111111101111110110111011101101
*L00768 1111111111111111111111110111111101101
*L00800 111111111111111111111111011011110
*L00832 1111111111111111111111110111101101
*L00864 1111111111111111111111110111111101110
*L00896 111111111011111111111111111111111111
*L01024 111111111011111111111111111111111111
*L01280 111111111011111111111111111111111111
*L01536 011111111011011111111111111111111111
*L01568 11110111101110111111111111111111111
*L01792 01111111101110111111111111111111111
*L01824 1111011110110111111111111111111111
*L02048 11001111000000000000000000000000000000
*L02112 000000000000000011111111111111111111
*L02144 111111111111111111111111111111111111
*L02176 111111111111111111111111111111111111
*C521C
```

Appendice A4

CODICE DEL DRIVER

Di seguito riportiamo i listati in C per la realizzazione del driver in RTLinux (versione 2.2 o successiva). E' disponibile un pacchetto software completo, comprendente oltre al codice del driver alcuni programmi di test.

Nell'ordine abbiamo:

1. Modulo realtime

- dcmod.h
- dcmod.c

2. Front-end Linux

- dcdrv.h
- dcdrv.c
- mrobot.h
- mrobot.c

```

/*
** DCMMOD.H - Include file for DCMMOD.C, DCMDRV.C
**
** (C) by Flavio Cappelli
*/
#ifndef _FC_DCMMOD_H
#define _FC_DCMMOD_H

/* Define RTFIFOs used for this driver.*/

#define CMD_FIFO_ID          0
#define ARG_FIFO_ID           1
#define MON_FIFO_ID           2

/* Define data size of FIFO used to monitor motion.*/

#define MON_FIFO_LEN          16384

/* Define values returned from module driver.*/

#define CMDERR                0xEE

/* Define commands that user side of the driver sends to RTLinux module. */
/*
 *      Command          data out        data in
 */
/* -----
/* SETUP_DRIVER_COMMAND    2 bytes       ack
/* CLEANUP_DRIVER_COMMAND  0 bytes       ack
/*
/* ENABLE_MOTOR_COMMAND    1 byte        ack
/* DISABLE_MOTOR_COMMAND   1 byte        ack
/*
/* SET_PORT_VALUE_COMMAND  2 bytes       ack
/* GET_PORT_VALUE_COMMAND  1 byte       ack + 1 byte
/*
/* SET_DUTYCYCLE_COMMAND  2 shorts     ack
/*
/* SET_VELOCITY_COMMAND    2 shorts     ack
/* GET_VELOCITY_COMMAND    0 bytes      ack + 2 shorts
/*
/* RESET_ORIGIN_COMMAND    1 byte        ack
/* SET_POSITION_COMMAND    2 longs      ack
/* GET_POSITION_COMMAND    0 bytes      ack + 2 longs
/*
/* SET_PID_GAINS_COMMAND   1 byte + 3 shorts ack
/* SET_ACC_AND_VLIM_COMMAND 1 byte + 2 shorts ack
/*
/* SET_OVERCTIME_COMMAND   1 byte + 1 short ack
/* GET_OVERCSTATUS_COMMAND  0 bytes      ack + 1 byte
/* RESET_OVERCSTATUS_COMMAND 0 bytes      ack
/*
/* SET_ERR_AND_SC_COMMAND  0 bytes      NONE
/*
/* ack = same sent command.

#define SETUP_DRIVER_COMMAND 0x11
#define CLEANUP_DRIVER_COMMAND 0x12
#define ENABLE_MOTOR_COMMAND 0x21
#define DISABLE_MOTOR_COMMAND 0x22
#define SET_PORT_VALUE_COMMAND 0x31
#define GET_PORT_VALUE_COMMAND 0x32
#define SET_DUTYCYCLE_COMMAND 0x41
#define SET_VELOCITY_COMMAND 0x42
#define GET_VELOCITY_COMMAND 0x43
#define RESET_ORIGIN_COMMAND 0x44
#define SET_POSITION_COMMAND 0x45
#define GET_POSITION_COMMAND 0x46
#define SET_PID_GAINS_COMMAND 0x47
#define SET_ACC_AND_VLIM_COMMAND 0x48
#define SET_OVERCTIME_COMMAND 0x51
#define GET_OVERCSTATUS_COMMAND 0x52
#define RESET_OVERCSTATUS_COMMAND 0x53
#define SET_ERR_AND_SC_COMMAND 0x7F

/* Interface with the realtime side of the driver. */

struct data_B { char arg; };
struct data_S { short arg; };
struct data_BB { char arg1, arg2; };

```

```
struct data_LL { long arg1, arg2; };
struct data_SS { short arg1, arg2; };
struct data_BS { char arg1; short arg2; };
struct data_SSS { short arg1, arg2, arg3; };
struct data_BSS { char arg1; short arg2, arg3; };
struct data_BSSS { char arg1; short arg2, arg3, arg4; };

struct command { unsigned char cmd; };
struct command_B { unsigned char cmd; struct data_B val; };
struct command_S { unsigned char cmd; struct data_S val; };
struct command_BB { unsigned char cmd; struct data_BB val; };
struct command_LL { unsigned char cmd; struct data_LL val; };
struct command_SS { unsigned char cmd; struct data_SS val; };
struct command_BS { unsigned char cmd; struct data_BS val; };
struct command_BSS { unsigned char cmd; struct data_BSS val; };
struct command_BSSS { unsigned char cmd; struct data_BSSS val; };

#endif /* _FC_DCMMOD_H */
```

```

/*
** DCMMOD.C - DC Motor Control Driver (RTLinux module).
**
** (C) by Flavio Cappelli
**
**
** THIS FILE MUST BE COMPILED WITH OPTIMIZATION -O2 or GREATER.
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
*/
#ifndef __RTL__
#error This file must be compiled under REALTIME LINUX
#endif

#include <rtl.h>
#include <rtl_sync.h>
#include <rtl_fifo.h>
#include <asm/io.h>
#include <limits.h>
#include "dcmod.h"
#include "dcmdrv.h"

/* I/O Macros.*/
#define inp_byte(port)           inb(port)
#define out_byte(port,data)      outb((data),(port))

/* 82C55 PPI Addresses.*/
#define _82C55_PortA            (io + 0x00)
#define _82C55_PortB            (io + 0x01)
#define _82C55_PortC            (io + 0x02)
#define _82C55_CtrlREG          (io + 0x03)

/* 82C54 TIMER 1 Addresses.*/
#define _82C54_T1_Counter0       (io + 0x04)
#define _82C54_T1_Counter1       (io + 0x05)
#define _82C54_T1_Counter2       (io + 0x06)
#define _82C54_T1_CtrlREG        (io + 0x07)

/* 8254 TIMER 2 Addresses.*/
#define _82C54_T2_Counter0       (io + 0x08)
#define _82C54_T2_Counter1       (io + 0x09)
#define _82C54_T2_Counter2       (io + 0x0A)
#define _82C54_T2_CtrlREG        (io + 0x0B)

/* Both Timers Addresses (Write only).*/
#define _82C54_BOTH_Counter0     (io + 0x0C)
#define _82C54_BOTH_Counter1     (io + 0x0D)
#define _82C54_BOTH_Counter2     (io + 0x0E)
#define _82C54_BOTH_CtrlREG      (io + 0x0F)

/* Some useful definitions.*/
#define SENDBACK_CMD             rtf_put(ARG_FIFO_ID, &(inbuf->cmd), 1)
#define SENDBACK_DATA             if (inbuf->cmd != CMDERR) rtf_put(ARG_FIFO_ID, &outbuf, sizeof(outbuf))
#define GET_CMDARGS(cmdtype)      if (index == 1) {to_get = sizeof(struct cmdtype) - 1; continue;}
#define CALL_FUNCTION(func, argtype) ##func((struct argtype *)inpbuff)
#define DISABLE_RTL_INTERRUPT    rtl_no_interrupts(IEN)
#define RESTORE_RTL_INTERRUPT    rtl_restore_interrupts(IEN)
#define DISABLE_BOARD_IRQ         rtl_hard_disable_irq(irq)
#define ENABLE_BOARD_IRQ          rtl_hard_enable_irq(irq)
#define UWORLD                     unsigned short

/* Motion modes: */
#define _MANUAL_MODE              0
#define _VELOCITY_MODE             1
#define _POSITION_MODE              2

/* Phases in trajectory generation.*/
#define _NO_PHASE                 0
#define _VEL_CHANGE                1      /* Velocity mode: acceleration or deceleration.*/
#define _POS_GOTO_0V                2      /* Position mode: unconditionally go to 0 velocity.*/
#define _POS_PHASE_1                  3      /* Position mode: first phase of trajectory generation.*/
#define _POS_PHASE_2                  4      /* Position mode: second phase of trajectory generation.*/

/* PID and trajectory generation default parameters.*/
#define _KP_PID                    2000    /* 0, +32767 */
#define _KI_PID                    10      /* 0, +32767 */

```

```

#define _KD_PID          0      /* 0, +32767 */
#define _VLIMIT          200    /* 0, +32767 */
#define _ACCEL           20     /* 0, +32767 */

/* PID integral action: max position error that generates integral action.*/
#define _MAX_INT_ERR     4096   /* (N. of encoder tick) */

/* Max position error in trajectory generation that can be safely corrected by PID */
/* without abrupt motion; greater errors schedule a new move and a new trajectory.*/
#define _MAX_POS_ERR     32     /* (N. of encoder tick) */

/* Define fractional bit of position, acceleration and velocity.*/
#define _FRACT_POS        4      /* To hold fractional encoder counts.*/
#define _FRACT_VEL        12     /* To smooth position increment.*/
#define _FRACT_ACC        4      /* To smooth velocity increment.*/

/* Version string.*/
#define _DCMMOD_STRVER_ "v1.0.2 21/03/2001"

/* Shared variables: module parameters.*/
int io          = -1;           /* Board I/O address (see dcmdrv.h).*/
int irq         = -1;           /* Hardware interrupt (see dcmdrv.h).*/
int pwm_res    = -1;           /* PWM resolution (see dcmdrv.h).*/

MODULE_PARM(io,      "i");
MODULE_PARM(irq,     "i");
MODULE_PARM(pwm_res, "i");

/* Non-shared variables.*/
static rtl_irqstate_t IEN;
static UBYTE PPI_Ctrl = 0x9A;
static UBYTE PPI_PortC = 0x03;
static UBYTE is_drv_init = 0;
static UBYTE pwm_mode;
static UBYTE phase[2];
static UBYTE reset[2];
static UBYTE neg_move[2];
static UBYTE is_saturated[2];
static UWORLD enc_lastUp[2], enc_lastDn[2];
static UWORLD ovc_count[2], ovc_max[2], ovc_stat;
static long cmd_pos[2], gen_pos[2], cur_pos[2];
static long cmd_vel[2], gen_vel[2];
static long cmd_PWM[2], cur_PWM[2];
static long vlimit[2], accel[2];
static long e1[2], integral[2];
static long phase1_dist[2];
/*
static long flat_count[2];
static short Kp[2], Ki[2], Kd[2];
static short cur_vel[2];
static int motion_mode;
*/
/* Irq state (realtime linux flag). */
/* PPI 82C55 Control Register value. */
/* PPI 82C55 PORT C (Low Nibble) initialization value. */
/* Flag: 1 = driver and board initialized properly. */
/* PWM mode (_PWM_AND_DIR / _PWM_ONLY). */
/* Flag: Phases in trajectory generation. */
/* Flag: 1 = schedule reset of encoder increment. */
/* Flag: 1 = commanded position < current position. */
/* Flag: 1 = PID saturate PWM dutycycles. */
/* Encoders Up and Down ticks: last read. */
/* Counters, max values and status for overcurrent. */
/* Commanded, generated and current position (shifted). */
/* Commanded and generated velocity (shifted). */
/* Commanded and current motors PWM dutycycles. */
/* Velocity limit (shifted), acceleration (shifted). */
/* PID last error and integrated value. */
/* Total distance for first half of trajectory. */

/* Counters for flat portion of generated trajectory. */
/* PID proportional, integral and differential gain. */
/* Current velocity (not shifted). */
/* Motion Mode: MANUAL, VELOCITY or POSITION. */

/***** INTERRUPT HANDLER FUNCTIONS *****/
static void check_overcurrent(void)
{
    /* Get signal from PPI 8255.*/
    int i = inp_byte(_82C55_PortC) >> 4;

    /* Update or reset counters.*/
    if (i & _Motor1)  ++ovc_count[0];           /* Overcurrent.*/
    else ovc_count[0] = 0;                         /* No overcurrent.*/
    if (i & _Motor2)  ++ovc_count[1];
    else ovc_count[1] = 0;

    /* Check max cycles to consider a valid overcurrent condition.*/
    if (ovc_max[0] > 0 && ovc_count[0] > ovc_max[0]) /* First test if overcurrent checking is enabled.*/
        ovc_stat |= _Motor1;
    if (ovc_max[1] > 0 && ovc_count[1] > ovc_max[1])
        ovc_stat |= _Motor2;

    /* Disable motors if overcurrent occurs.*/
    if (ovc_stat)
        out_byte(_82C55_PortC, PPI_PortC |= 0x03);
}

static void get_encoder_count(UWORD *enc_CntUp, UWORLD *enc_CntDn)
{
    UWORLD lsb, msb;                           /* DO NOT CHANGE TYPE.*/
    /* Latch counters 1 and 2 on both timers (read back).*/
}

```

```

out_byte(_82C54_BOTH_CtrlREG, 0xDC);
out_byte(0x80, 255);                                /* Wait about 1 us: DO NOT REMOVE THIS.*/

/* Read counter 1, #1 8254.*/
lsb = inp_byte(_82C54_T1_Counter1);
msb = inp_byte(_82C54_T1_Counter1);
enc_CntUp[0] = (msb << 8) + lsb;
                                         /* Read least significant byte of counter.*/
                                         /* Read most significant byte of counter.*/
                                         /* Pack to 16 bit.*/

/* Read counter 2, #1 8254.*/
lsb = inp_byte(_82C54_T1_Counter2);
msb = inp_byte(_82C54_T1_Counter2);
enc_CntDn[0] = (msb << 8) + lsb;

/* Read counter 1, #2 8254.*/
lsb = inp_byte(_82C54_T2_Counter1);
msb = inp_byte(_82C54_T2_Counter1);
enc_CntUp[1] = (msb << 8) + lsb;

/* Read counter 2, #2 8254.*/
lsb = inp_byte(_82C54_T2_Counter2);
msb = inp_byte(_82C54_T2_Counter2);
enc_CntDn[1] = (msb << 8) + lsb;
}

static void update_position_and_velocity(void)
{
    UWORD enc_CntUp[2], enc_CntDn[2];

    /* Get the encoders increments from last call of this procedure using the */
    /* difference of the counters. This avoids clearing of timers at each sample */
    /* period, which involves a potential loss of counts. Using two's complement */
    /* arithmetic we magically take into consideration timers overflow.          */
    get_encoder_count(enc_CntUp, enc_CntDn);
    cur_vel[0] = (short)((enc_lastUp[0] - enc_CntUp[0]) - (enc_lastDn[0] - enc_CntDn[0]));
    cur_vel[1] = (short)((enc_lastUp[1] - enc_CntUp[1]) - (enc_lastDn[1] - enc_CntDn[1]));

    /* Save counters (used on next call to this procedure).*/
    enc_lastUp[0] = enc_CntUp[0]; enc_lastDn[0] = enc_CntDn[0];
    enc_lastUp[1] = enc_CntUp[1]; enc_lastDn[1] = enc_CntDn[1];

    /* Reset last encoder increment if scheduled.*/
    if (reset[0]) {
        reset[0] = 0; cur_vel[0] = 0;
    }
    if (reset[1]) {
        reset[1] = 0; cur_vel[1] = 0;
    }

    /* Update the positions (shifted bits hold fractional encoder counts). */
    cur_pos[0] += (long)(cur_vel[0]) << _FRACT_POS;
    cur_pos[1] += (long)(cur_vel[1]) << _FRACT_POS;
}

static long PID_action(const int m)
{
    long p_e, ypid;

    /* Get position error (discard bits of position that hold fractional encoder counts).*/
    p_e = (gen_pos[m] - cur_pos[m]) >> _FRACT_POS;

    /* Limit error to 16-bit signed integer.*/
    if (p_e > +32767) p_e = +32767;
    if (p_e < -32767) p_e = -32767;

    /* Anti-Windup: bypass integration if error is too big or PWM was saturated.*/
    if (p_e < -_MAX_INT_ERR || p_e > _MAX_INT_ERR)
        integral[m] = 0;                                /* Reset integration.*/
    else if (!is_saturated[m])
        integral[m] += p_e;

    /* Calculate sum of proportional, integral and differential term of PID.*/
    ypid = p_e * Kp[m] + integral[m] * Ki[m] + (p_e - e1[m]) * (Kd[m] << 8);

    /* Save current error.*/
    e1[m] = p_e;

    /* Return PWM dutycycle (shifting improves precision of fixed point math).*/
    return (ypid >> 12);
}

```

```

/* For the trajectory generation I have taken some ideas from application notes AN532 and AN718 */
/* on MICROCHIP(R) site (http://www.microchip.com). However my procedure allows commanding of new */
/* positions also while move is already in progress (microchip version does not allow that). */
/* Furthermore, you can change motion mode at any time without abrupt motion. */
static void update_trajectory(const int m)
{
    switch(motion_mode)
    {
        case _VELOCITY_MODE:
            phase[m] = _NO_PHASE;
            if(gen_vel[m] < cmd_vel[m])
            {
                phase[m] = _VEL_CHANGE;
                gen_vel[m] += accel[m];
                if(gen_vel[m] > cmd_vel[m])
                    gen_vel[m] = cmd_vel[m];
            }
            else if(gen_vel[m] > cmd_vel[m])
            {
                phase[m] = _VEL_CHANGE;
                gen_vel[m] -= accel[m];
                if(gen_vel[m] < cmd_vel[m])
                    gen_vel[m] = cmd_vel[m];
            }
            gen_pos[m] += (gen_vel[m] >> (_FRACT_VEL - _FRACT_POS)); /* Update position.*/
            if (gen_pos[m] > (LONG_MAX/2)) { /* Avoid wrapping around of position when the */
                gen_pos[m] -= (LONG_MAX/2); /* same direction is maintained for long time.*/
                cur_pos[m] -= (LONG_MAX/2);
            }
            if (gen_pos[m] < -(LONG_MAX/2)) {
                gen_pos[m] += (LONG_MAX/2);
                cur_pos[m] += (LONG_MAX/2);
            }
            break;

        case _POSITION_MODE:
            switch(phase[m])
            {
                case _POS_PHASE_1:
                    if (neg_move[m])
                    {
                        if(gen_vel[m] > -vlimit[m]) {
                            gen_vel[m] -= accel[m];
                            if (gen_vel[m] < -vlimit[m])
                                gen_vel[m] = -vlimit[m];
                        }
                        else ++flat_count[m];
                        phase1_dist[m] += gen_vel[m] >> (_FRACT_VEL - _FRACT_POS); /* Update remaining linear */
                        /* distance for phase 2. */
                    }
                    else
                    {
                        if(gen_vel[m] < vlimit[m]) {
                            gen_vel[m] += accel[m];
                            if (gen_vel[m] > vlimit[m])
                                gen_vel[m] = vlimit[m];
                        }
                        else ++flat_count[m];
                        phase1_dist[m] -= gen_vel[m] >> (_FRACT_VEL - _FRACT_POS); /* Update remaining linear */
                        /* distance for phase 2. */
                    }
                    gen_pos[m] += gen_vel[m] >> (_FRACT_VEL - _FRACT_POS); /* Update position.*/
                    if(phase1_dist[m] < 0) /* If phase1_dist has gone
negative, the first */
                        phase[m] = _POS_PHASE_2; /* half of the move has been completed. */
                    break;

                case _POS_PHASE_2:
                    if(flat_count[m] > 0)
                        --flat_count[m];
                    else {
                        if(gen_vel[m] < 0) {
                            gen_vel[m] += accel[m];
                            if (gen_vel[m] > 0)
                                gen_vel[m] = 0;
                        }
                        else if(gen_vel[m] > 0) {
                            gen_vel[m] -= accel[m];
                        }
                    }
            }
    }
}

```

```

        if (gen_vel[m] < 0)
            gen_vel[m] = 0;
    }

    if (gen_pos[m] != 0)
        gen_pos[m] += gen_vel[m] >> (_FRACT_VEL - _FRACT_POS);      /* Update position.*/
    else {
        /* Move is completed, check position error.*/
        if (gen_pos[m] > cmd_pos[m] + (_MAX_POS_ERR << _FRACT_POS)
            || gen_pos[m] < cmd_pos[m] - (_MAX_POS_ERR << _FRACT_POS) )
        {
            /* Schedule again move to the last commanded position to correct position error.*/
            phase1_dist[m] = cmd_pos[m] - gen_pos[m]; /* Evaluate the relative move distance.*/
            if(phase1_dist[m] < 0)                                /* If the relative move is
negative,   */
            {
                neg_move[m] = 1;                                /* set flag to indicate negative move */
                phase1_dist[m] = -phase1_dist[m];               /* and convert distance to a positive */
                                                       /* value. */
            }
            else neg_move[m] = 0;                            /* Clear the flag for a positive move.*/
            phase1_dist[m] /= 2;                            /* Now holds the total distance, so divide by 2.*/
            flat_count[m] = 0;                            /* Clear counter for flat part of trajectory.*/
            phase[m] = _POS_PHASE_1;                      /* Start phase 1 of trajectory generation.*/
        }
        else {
            /* Position error is small, PID will correct it, so set final position.*/
            gen_pos[m] = cmd_pos[m];
            phase[m] = _NO_PHASE;
        }
    }
    break;
}

case _POS_GOTO_0V:                                /* Unconditional move until zero velocity is reached.*/
{
    if(gen_vel[m] < 0) {                         /* Negative velocity */
        gen_vel[m] += accel[m];                   /* Decelerate.*/
        if (gen_vel[m] > 0)
            gen_vel[m] = 0;
    }
    else if(gen_vel[m] > 0) {                     /* Positive velocity ? */
        gen_vel[m] -= accel[m];                   /* Decelerate.*/
        if (gen_vel[m] < 0)
            gen_vel[m] = 0;
    }

    if (gen_vel[m] != 0)
        gen_pos[m] += gen_vel[m] >> (_FRACT_VEL - _FRACT_POS);      /* Update position.*/
    else
    {
        /* Position with 0 velocity is reached so schedule move to the last commanded position.*/
        phase1_dist[m] = cmd_pos[m] - gen_pos[m];      /* Evaluate the relative move distance.*/
        if(phase1_dist[m] < 0)                                /* If the relative move is
negative,   */
        {
            neg_move[m] = 1;                                /* set flag to indicate negative move */
            phase1_dist[m] = -phase1_dist[m];               /* and convert distance to a positive */
                                                       /* value. */
        }
        else neg_move[m] = 0;                            /* Clear the flag for a positive move.*/
        phase1_dist[m] /= 2;                            /* Now holds the total distance, so divide by 2.*/
        flat_count[m] = 0;                            /* Clear counter for flat part of trajectory.*/
        phase[m] = _POS_PHASE_1;                      /* Start phase 1 of trajectory generation.*/
    }
    break;
}
break;
}

static void set_pwm_and_dir(void)
{
    int m = 0, pwm_range = 1 << pwm_res, sign, dcycle[2];
    long tmp;

    do {
        /* Save sign of PWM and make it positive.*/
        tmp = cur_PWM[m]; sign = 1;
        if (tmp < 0) {sign = -1; tmp = -tmp;}

        /* Saturate PWM value.*/
        is_saturated[m] = 0;

```

```

if (tmp > 1023) {tmp = 1023; is_saturated[m] = 1;}

/* Scale to current dutycycle range.*/
tmp >>= 10 + pwm_mode - pwm_res;
if (pwm_mode == _PWM_ONLY)
    tmp = sign * tmp + pwm_range / 2;

/* Adjust value to drive H-Bridge hardware and save calculated dutycycle.*/
dcycle[m] = pwm_range - tmp;

/* Update motor direction.*/
if (sign == 1) PPI_PortC &= ~(0x04 << m);           /* Set normal direction.*/
else PPI_PortC |= (0x04 << m);                     /* Set reverse direction.*/
}
while (++m < 2);

/* Update counters 0 to generate new PWM dutycycle.*/
out_byte(_82C54_T1_Counter0, (UBYTE)(dcycle[0]));      /* Write LSB of counter (motor 1).*/
out_byte(_82C54_T1_Counter0, (UBYTE)(dcycle[0] >> 8)); /* Write MSB of counter (motor 1).*/
out_byte(_82C54_T2_Counter0, (UBYTE)(dcycle[1]));      /* Write LSB of counter (motor 2).*/
out_byte(_82C54_T2_Counter0, (UBYTE)(dcycle[1] >> 8)); /* Write MSB of counter (motor 2).*/

/* Update motor direction.*/
out_byte(_82C55_PortC, PPI_PortC);                    /* Update hardware port.*/
}

static void save_position_and_velocity(void)
{
    struct monitor_data data;

    /* Save current position and velocity. */
    data.pos1 = cur_pos[0] >> _FRACT_POS;                /* Return position (discard bits that hold */
    data.vel1 = cur_vel[0];                                /* fractional encoder counts) and velocity.*/
    data.pos2 = cur_pos[1] >> _FRACT_POS;
    data.vel2 = cur_vel[1];
    rtf_put(MON_FIFO_ID, &data, sizeof(data));

    /* Save generated position and velocity. */
    data.pos1 = gen_pos[0] >> _FRACT_POS;
    data.vel1 = gen_vel[0] >> _FRACT_VEL;
    data.pos2 = gen_pos[1] >> _FRACT_POS;
    data.vel2 = gen_vel[1] >> _FRACT_VEL;
    rtf_put(MON_FIFO_ID, &data, sizeof(data));
}

unsigned int interrupt_handler(unsigned int irq, struct pt_regs *regs)
{
    if((PPI_PortC & 0x03) != 0x03)                      /* Ignore false interrupts (motors disabled).*/
    {
        /* Check if an overcurrent condition has occurred.*/
        check_overcurrent();

        /* Get the distances travelled during the sample period.*/
        update_position_and_velocity();

        /* Save data for monitoring and tuning motion.*/
        save_position_and_velocity();

        /* PID controller update PWM values.*/
        if (motion_mode != _MANUAL_MODE) {
            update_trajectory(0);                         /* Get new generated position. */
            update_trajectory(1);
            cmd_PWM[0] = PID_action(0);                  /* Use PID to set new PWM values.*/
            cmd_PWM[1] = PID_action(1);
        }

        /* Update PWM dutycycle (if necessary).*/
        if (cur_PWM[0] != cmd_PWM[0] || cur_PWM[1] != cmd_PWM[1]) {
            cur_PWM[0] = cmd_PWM[0];
            cur_PWM[1] = cmd_PWM[1];
            set_pwm_and_dir();
        }
    }
    /* Handler has interrupt disabled, so reenable hard irq.*/
    ENABLE_BOARD_IRQ;
    return 0;
}

```

***** FIFO HANDLER FUNCTIONS *****

```

static void setup_driver_cmd(struct command_BB *inbuf)
{
    /* Check parameters.*/
    if ((inbuf -> val.arg1 & ~18) == 0 && (inbuf -> val.arg2 & ~1) == 0)
    {
        /* If driver has been initialized cleanup it first.*/
        if (is_drv_init)
        {
            DISABLE_RTL_INTERRUPT;                                /* Disable interrupts: PPI_PortC modified by ISR.*/
            out_byte(_82C55_CtrlREG, (PPI_Ctrl = 0x9A));          /* Port A & B input (side effect: enable motors).*/
            out_byte(_82C55_PortC, (PPI_PortC = 0x03));          /* Disable motors and board interrupts. */
            RESTORE_RTL_INTERRUPT;                               /* Restore interrupts to previous state.*/
            is_drv_init = 0;
        }

        /* Initialize variables.*/
        PPI_Ctrl = 0x88 + inbuf -> val.arg1;                  /* I/O direction: A & B specified, CH in, CL out.*/
        pwm_mode = inbuf -> val.arg2;                          /* Save PWM mode to calculate correct dutycycle. */
        motion_mode = _MANUAL_MODE;                            /* Set default motion mode.*/
    }

    /* Do PPI initialization and put high C0 and C1 to disable motors. NOTE: when setting the */
    /* port mode, the lines of ports go low (side effect), so putting high C0/C1 might generate */
    /* a false interrupt. But interrupt are ignored if motors are disabled (see irq handler). */
    out_byte(_82C55_CtrlREG, PPI_Ctrl);
    out_byte(_82C55_PortC, PPI_PortC);

    /* Do timers mode initialization (both).*/
    out_byte(_82C54_BOTH_CtrlREG, 0x32);                    /* Counter 0: mode 1.*/
    out_byte(_82C54_BOTH_CtrlREG, 0x78);                    /* Counter 1: mode 4.*/
    out_byte(_82C54_BOTH_CtrlREG, 0xB8);                    /* Counter 2: mode 4.*/
}

/* Set PID and motion parameters.*/
phase[1] = phase[0] = _NO_PHASE;
accel[1] = accel[0] = (_ACCEL << _FRACT_ACC);
vlimit[1] = vlimit[0] = (_VLIMIT << _FRACT_VEL);
Kp[1] = Kp [0] = _KP_PID; Ki[1] = Ki [0] = _KI_PID; Kd[1] = Kd [0] = _KD_PID;
e1[1] = e1[0] = integral[1] = integral[0] = 0;

/* Set position and velocity to 0.*/
cmd_pos[1] = cmd_pos[0] = gen_pos[1] = gen_pos[0] = cur_pos[1] = cur_pos[0] = 0;
cmd_vel[1] = cmd_vel[0] = gen_vel[1] = gen_vel[0] = cur_vel[1] = cur_vel[0] = 0;

/* Set PWM duty cycle to 0 (counters 0 initialization) */
cmd_PWM[1] = cmd_PWM[0] = cur_PWM[1] = cur_PWM[0] = 0;
set_pwm_and_dir();

/* Reset some other conditions.*/
ovc_count[1] = ovc_count[0] = ovc_max[1] = ovc_max[0] = ovc_stat = 0;
reset[1] = reset[0] = 0;

/* Performs encoder ticks counter initialization: each counter will be set on its value - 1 */
/* because the first CLK impulse loads the new value, but does not decrement the counter. */
get_encoder_count(enc_lastUp, enc_lastDn);
out_byte(_82C54_T1_Counter1, (UBYTE)((enc_lastUp[0] - 1)));           /* Write LSB of counter.*/
out_byte(_82C54_T1_Counter1, (UBYTE)((enc_lastUp[0] - 1) >> 8));   /* Write MSB of counter.*/
out_byte(_82C54_T1_Counter2, (UBYTE)((enc_lastDn[0] - 1)));
out_byte(_82C54_T2_Counter1, (UBYTE)((enc_lastUp[1] - 1)));
out_byte(_82C54_T2_Counter1, (UBYTE)((enc_lastUp[1] - 1) >> 8));
out_byte(_82C54_T2_Counter2, (UBYTE)((enc_lastDn[1] - 1)));
out_byte(_82C54_T2_Counter2, (UBYTE)((enc_lastDn[1] - 1) >> 8));

/* Inform other procedures that board is initialized properly.*/
is_drv_init = 1;
}

else inbuf -> cmd = CMDERR;                                         /* Set error.*/
SENDBACK_CMD;                                                       /* Send acknowledge command (if */
                                                               /* everything is ok) or error. */
}

static void cleanup_driver_cmd(struct command *inbuf)
{
    /* Check if driver has been initialized properly.*/
    if (is_drv_init)
    {
        /* Ensures only one cleanup.*/
        is_drv_init = 0;

        /* Restore default hardware conditions.*/
        DISABLE_RTL_INTERRUPT;                                /* Disable interrupts: PPI_PortC modified by ISR.*/
        out_byte(_82C55_CtrlREG, (PPI_Ctrl = 0x9A));          /* Port A & B input (side effect: enable motors).*/
        out_byte(_82C55_PortC, (PPI_PortC = 0x03));          /* Disable motors and board interrupts. */
        RESTORE_RTL_INTERRUPT;                               /* Restore interrupts to previous state.*/
    }
}

```

```

}

else inbuf -> cmd = CMDERR;
SENDBACK_CMD;
} /* Set error.*/
/* Send acknowledge command (if */
/* everything is ok) or error. */

static void enable_motor_cmd(struct command_B *inbuf)
{
    if (is_drv_init && (inbuf->val.arg & ~3) == 0)      /* Check driver initialization and data.*/
    {
        DISABLE_RTL_INTERRUPT;                            /* Disable interrupts: PPI_PortC modified by ISR.*/
        if (inbuf->val.arg & _Motor1)                   /* Schedule reset of last encoder increment.*/
            reset[0] = 1;
        if (inbuf->val.arg & _Motor2)                   /* Schedule reset of last encoder increment.*/
            reset[1] = 1;
        out_byte(_82C55_PortC, PPI_PortC &= ~inbuf->val.arg); /* Enable motor(s).*/
        RESTORE_RTL_INTERRUPT;                           /* Restore interrupts to previous state.*/
    }
    else inbuf -> cmd = CMDERR;                         /* Set error.*/
    SENDBACK_CMD;                                       /* Send acknowledge command (if */
    /* everything is ok) or error. */
}

static void disable_motor_cmd(struct command_B *inbuf)
{
    if (is_drv_init && (inbuf->val.arg & ~3) == 0)
    {
        DISABLE_RTL_INTERRUPT;
        out_byte(_82C55_PortC, PPI_PortC |= inbuf->val.arg); /* Disable motor(s).*/
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_dutycycle_cmd(struct command_SS *inbuf)
{
    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;
        motion_mode = _MANUAL_MODE;
        cmd_PWM[0] = inbuf -> val.arg1;
        cmd_PWM[1] = inbuf -> val.arg2;
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
}

static void get_velocity_cmd(struct command *inbuf)
{
    struct data_SS outbuf;

    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;
        outbuf.arg1 = cur_vel[0];                         /* Save current velocity. CUR_VEL is not used in */
        outbuf.arg2 = cur_vel[1];                         /* motion update functions, so it is not shifted.*/
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
    SENDBACK_DATA;                                     /* Send data to the user side of */
    /* driver (if no error occurs). */
}

static void get_position_cmd(struct command *inbuf)
{
    struct data_LL outbuf;

    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;
        outbuf.arg1 = cur_pos[0] >> _FRACT_POS;          /* Return position (discard bits that */
        outbuf.arg2 = cur_pos[1] >> _FRACT_POS;          /* hold fractional encoder counts). */
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
}

```

```

SENDBACK_CMD;
SENDBACK_DATA;
}

static void set_velocity_cmd(struct command_BSS *inbuf)
{
    int i = 0, fbits = inbuf->val.arg1;                      /* Get fractional bits of specified velocities.*/
    if (is_drv_init && fbits >= 0 && fbits <= 4)
    {
        DISABLE_RTL_INTERRUPT;
        if (motion_mode == _MANUAL_MODE)                         /* INITIALIZE VELOCITY KEEPING THE SAME OLD VELOCITY.*/
        {
            gen_vel[0] = (long)(cur_vel[0]) << _FRACT_VEL;
            gen_vel[1] = (long)(cur_vel[1]) << _FRACT_VEL;
            gen_pos[0] = cur_pos[0] + ((long)(cur_vel[0]) << _FRACT_POS);
            gen_pos[1] = cur_pos[1] + ((long)(cur_vel[1]) << _FRACT_POS);
            e1[1] = e1[0] = integral[1] = integral[0] = 0;          /* Reset PID variables.*/
        }

        motion_mode = _VELOCITY_MODE;
        cmd_vel[0] = (long)(inbuf -> val.arg2) << (_FRACT_VEL - fbits);      /* 32 bits are used to */
        cmd_vel[1] = (long)(inbuf -> val.arg3) << (_FRACT_VEL - fbits);      /* smooth motion update.*/
        do {
            if (vlimit[i] > 0 && accel[i] > 0)                /* Check if profile generator is enabled for motor.*/
            {
                if(cmd_vel[i] > vlimit[i])                         /* If so, don't exceed velocity limit parameter.*/
                    cmd_vel[i] = vlimit[i];
                else if(cmd_vel[i] < -vlimit[i])
                    cmd_vel[i] = -vlimit[i];
            }
            else gen_vel[i] = cmd_vel[i];                          /* If not, ignore profile generator */
            /* and force commanded velocity. */
        } while(++i<2);                                         /* Repeat for all motors.*/
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_position_cmd(struct command_LL *inbuf)
{
    int i = 0;

    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;
        if (motion_mode == _MANUAL_MODE)                         /* INITIALIZE VELOCITY KEEPING THE SAME OLD VELOCITY.*/
        {
            gen_vel[0] = (long)(cur_vel[0]) << _FRACT_VEL;
            gen_vel[1] = (long)(cur_vel[1]) << _FRACT_VEL;
            gen_pos[0] = cur_pos[0] + ((long)(cur_vel[0]) << _FRACT_POS);
            gen_pos[1] = cur_pos[1] + ((long)(cur_vel[1]) << _FRACT_POS);
            e1[1] = e1[0] = integral[1] = integral[0] = 0;          /* Reset PID variables.*/
        }

        motion_mode = _POSITION_MODE;
        cmd_pos[0] = (inbuf -> val.arg1) << _FRACT_POS; /* (some bits of position hold */
        cmd_pos[1] = (inbuf -> val.arg2) << _FRACT_POS; /* fractional encoder counts). */

        do {
            phase[i] = _NO_PHASE;                            /* By default ignore motion profile generator. */
            if (vlimit[i] > 0 && accel[i] > 0)             /* Check if profile generator is enabled for motor.*/
            {
                long p0, dis, n;

                if (gen_vel[i] >= 0) {                      /* If positive move is in progress, */
                    n = gen_vel[i]/accel[i];                  /* estimate distance from the nearest */
                    dis = (long)((long long)n * (n+1) * accel[i]) >> (_FRACT_VEL + 1 - _FRACT_POS));
                    p0 = gen_pos[i] - dis;                   /* virtual position with 0 velocity. */
                    if (cmd_pos[i] < gen_pos[i] + dis)       /* If commanded position is too near: stop first.*/
                        phase[i] = _POS_GOTO_0V;
                }
                else if (gen_vel[i] < 0) {                  /* If negative move is in progress, */
                    n = -gen_vel[i]/accel[i];                 /* estimate distance from the nearest */
                    dis = (long)((long long)n * (n+1) * accel[i]) >> (_FRACT_VEL + 1 - _FRACT_POS));
                    p0 = gen_pos[i] + dis;                   /* virtual position with 0 velocity. */
                    if (cmd_pos[i] > gen_pos[i] - dis)       /* If commanded position is too near: stop first.*/
                        phase[i] = _POS_GOTO_0V;
                }
            }
        }
    }
}

```

```

if (phase[i] != _POS_GOTO_0V)
{
    phase1_dist[i] = cmd_pos[i] - p0;           /* Evaluate the relative move distance.*/
    if(phase1_dist[i] < 0)                     /* If the relative move is negative, */
    {
        neg_move[i] = 1;                         /* set flag to indicate negative move.*/
        phase1_dist[i] = -phase1_dist[i];         /* and convert distance to a positive value.*/
    }
    else neg_move[i] = 0;                       /* Clear the flag for a positive move.*/
    phase1_dist[i] /= 2;                        /* Now holds the total distance, so divide by 2.*/
    phase1_dist[i] -= dis;                      /* Consider part of trajectory already travelled.*/
    flat_count[i] = 0;                          /* Clear counter for flat part of trajectory.*/
    phase[i] = _POS_PHASE_1;                    /* Start phase 1 of trajectory generation.*/
}
else gen_pos[i] = cmd_pos[i];                  /* If not, ignore profile generator */
while(++i<2);                                /* and force commanded position. */
/* Repeat for all motors.*/
RESTORE_RTL_INTERRUPT;
}
else inbuf->cmd = CMDERR;
SENDBACK_CMD;
}

static void reset_origin_cmd(struct command_B *inbuf)
{
    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;
        if (inbuf->val.arg & _Motor1) {           /* THIS RESET DOES NOT DISTURB MOTION !*/
            cmd_pos[0] -= cur_pos[0];
            gen_pos[0] -= cur_pos[0];
            cur_pos[0] = 0;
        }
        if (inbuf->val.arg & _Motor2) {
            cmd_pos[1] -= cur_pos[1];
            gen_pos[1] -= cur_pos[1];
            cur_pos[1] = 0;
        }
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf->cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_pid_gains_cmd(struct command_BSS *inbuf)
{
    short kp = inbuf->val.arg2, ki = inbuf->val.arg3, kd = inbuf->val.arg4;
    if (is_drv_init && kp >= 0 && ki >= 0 && kd >= 0)
    {
        DISABLE_RTL_INTERRUPT;
        if (inbuf->val.arg1 & _Motor1) {
            Kp[0] = kp; Ki[0] = ki; Kd[0] = kd;
        }
        if (inbuf->val.arg1 & _Motor2) {
            Kp[1] = kp; Ki[1] = ki; Kd[1] = kd;
        }
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf->cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_acc_and_vlim_cmd(struct command_BSS *inbuf)
{
    long vlim = (long)(inbuf->val.arg2) << _FRACT_VEL; /* Scaling to obtain fractional bits.*/
    long acc = (long)(inbuf->val.arg3) << _FRACT_ACC;

    if (is_drv_init && vlim >= 0 && acc >= 0)
    {
        DISABLE_RTL_INTERRUPT;
        if ((inbuf->val.arg1 & _Motor1) && phase[0] == _NO_PHASE) {
            vlimit[0] = vlim; accel[0] = acc;
        }
        if ((inbuf->val.arg1 & _Motor2) && phase[1] == _NO_PHASE) {
            vlimit[1] = vlim; accel[1] = acc;
        }
    }
}

```

```

        }
        RESTORE_RTL_INTERRUPT;
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_port_value_cmd(struct command_BB *inbuf)
{
    if (is_drv_init)                                /* Check driver initialization.*/
    {
        switch (inbuf->val.arg1)
        {
            case _PortA:   if (!(PPI_Ctrl & _PortA))           /* Check portA direction.*/
                            out_byte(_82C55_PortA, inbuf->val.arg2); /* Out data to port A.*/
                            break;
            case _PortB:   if (!(PPI_Ctrl & _PortB))           /* Check portB direction.*/
                            out_byte(_82C55_PortB, inbuf->val.arg2); /* Out data to port B.*/
                            break;
            default:      inbuf -> cmd = CMDERR;             /* Bad parameter.*/
        }
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
}

static void get_port_value_cmd(struct command_B *inbuf)
{
    struct data_B outbuf;
    outbuf.arg = 0;                                  /* Default return value.*/
    if (is_drv_init)
    {
        switch (inbuf->val.arg)
        {
            case _PortA:   if (PPI_Ctrl & _PortA)           /* Check portA direction.*/
                            outbuf.arg = inp_byte(_82C55_PortA); /* Get data from port A.*/
                            break;
            case _PortB:   if (PPI_Ctrl & _PortB)           /* Check portB direction.*/
                            outbuf.arg = inp_byte(_82C55_PortB); /* Get data from port B.*/
                            break;
            default:      inbuf -> cmd = CMDERR;             /* Bad parameter.*/
        }
    }
    else inbuf -> cmd = CMDERR;
    SENDBACK_CMD;
    SENDBACK_DATA;
}

static void get_overcstatus_cmd(struct command *inbuf)
{
    struct data_B outbuf;
    outbuf.arg = 0;
    if (is_drv_init)
        outbuf.arg = ovc_stat;
    else inbuf->cmd = CMDERR;
    SENDBACK_CMD;
    SENDBACK_DATA;
}

static void reset_overcstatus_cmd(struct command *inbuf)
{
    if (is_drv_init)                                /* Reset overcurrent status. */
        ovc_stat = 0;
    else inbuf->cmd = CMDERR;
    SENDBACK_CMD;
}

static void set_overctime_cmd(struct command_BS *inbuf)
{
    if (is_drv_init && inbuf->val.arg2 >= 0)
    {
        DISABLE_RTL_INTERRUPT;
        if (inbuf->val.arg1 & _Motor1)

```

```

    ovc_max[0] = inbuf->val.arg2;
    if (inbuf->val.arg1 & _Motor2)
        ovc_max[1] = inbuf->val.arg2;
    RESTORE_RTL_INTERRUPT;
}
else inbuf -> cmd = CMDERR;
SENDBACK_CMD;
}

static void set_err_and_sc_cmd(struct command *inbuf) /* Restore safe condition on error.*/
{
    if (is_drv_init)
    {
        DISABLE_RTL_INTERRUPT;                                /* Disable interrupts: PPI_PortC modified by ISR.*/
        out_byte(_82C55_CtrlREG, (PPI_Ctrl = 0x9A));      /* Port A & B input (side effect: enable motors).*/
        out_byte(_82C55_PortC, (PPI_PortC = 0x03));       /* Disable motors and board interrupts. */
        RESTORE_RTL_INTERRUPT;                            /* Restore interrupts to previous state.*/
    }
}                                                       /* NO acknowledge required.*/
}

static int fifo_handler(unsigned int fifo)           /* Handler invoked in rt-task */
{                                                     /* when CMD fifo is accessed. */
    static unsigned char inpbuff[16];
    static int index = 0, to_get = 1;
    int n;

    while((n = rtf_get(CMD_FIFO_ID, &inpbuff[index], to_get)) > 0)          /* Read bytes from command */
    {                                         /* fifo if available. */
        index += n;
        if ((to_get -= n) > 0) break;          /* Wait if command is incomplete.*/
        switch (*inpbuff)                      /* Process current command.*/
        {
            case SETUP_DRIVER_COMMAND:          GET_CMDARGS(command_BB);      /* Get command arguments (if any)*/
                CALL_FUNCTION(setup_driver_cmd, command_BB); /* Process command.*/
                break;
            case CLEANUP_DRIVER_COMMAND:        CALL_FUNCTION(cleanup_driver_cmd, command);
                break;
            case ENABLE_MOTOR_COMMAND:         GET_CMDARGS(command_B);
                CALL_FUNCTION(enable_motor_cmd, command_B);
                break;
            case DISABLE_MOTOR_COMMAND:        GET_CMDARGS(command_B);
                CALL_FUNCTION(disable_motor_cmd, command_B);
                break;
            case SET_PORT_VALUE_COMMAND:       GET_CMDARGS(command_BB);
                CALL_FUNCTION(set_port_value_cmd, command_BB);
                break;
            case GET_PORT_VALUE_COMMAND:       GET_CMDARGS(command_B);
                CALL_FUNCTION(get_port_value_cmd, command_B);
                break;
            case SET_DUTYCYCLE_COMMAND:        GET_CMDARGS(command_SS);
                CALL_FUNCTION(set_dutycycle_cmd, command_SS);
                break;
            case SET_VELOCITY_COMMAND:         GET_CMDARGS(command_BSS);
                CALL_FUNCTION(set_velocity_cmd, command_BSS);
                break;
            case GET_VELOCITY_COMMAND:         CALL_FUNCTION(get_velocity_cmd, command);
                break;
            case RESET_ORIGIN_COMMAND:        GET_CMDARGS(command_B);
                CALL_FUNCTION(reset_origin_cmd, command_B);
                break;
            case SET_POSITION_COMMAND:         GET_CMDARGS(command_LL);
                CALL_FUNCTION(set_position_cmd, command_LL);
                break;
            case GET_POSITION_COMMAND:         CALL_FUNCTION(get_position_cmd, command);
                break;
            case SET_PID_GAINS_COMMAND:        GET_CMDARGS(command_BSS);
                CALL_FUNCTION(set_pid_gains_cmd, command_BSS);
                break;
            case SET_ACC_AND_VLIM_COMMAND:     GET_CMDARGS(command_BSS);
                CALL_FUNCTION(set_acc_and_vlim_cmd, command_BSS);
                break;
            case SET_OVERCTIME_COMMAND:        GET_CMDARGS(command_BS);
                CALL_FUNCTION(set_overcetime_cmd, command_BS);
                break;
            case GET_OVERCSTATUS_COMMAND:      CALL_FUNCTION(get_overcstatus_cmd, command);
                break;
            case RESET_OVERCSTATUS_COMMAND:   CALL_FUNCTION(reset_overcstatus_cmd, command);
                break;
            case SET_ERR_AND_SC_COMMAND:      CALL_FUNCTION(set_err_and_sc_cmd, command);
                break;
        }
    }
}

```

```

        default:                      return -EINVAL;
    }
    index = 0; to_get = 1;           /* Set variables to get next command.*/
}
return 0;                         /* Normal return value.*/
}

/******************* MODULE INSERTION AND REMOVE *****/
int init_module(void)
{
    /* Send startup message to console (or to kernel).*/
    rtl_printf("DCMOD: " _DCMMOD_STRVER " Flavio Cappelli (flv.cpp@libero.it)\n");
    rtl_printf("Isa DC Motor Control Board");

    /* Check module parameters.*/
    if ((io != 0x280 && io != 0x290 && io != 0x320 && io != 0x330
        && io != 0x340 && io != 0x350 && io != 0x380 && io != 0x390)
        || (irq != 10 && irq != 11 && irq != 15) || (pwm_res != 8 && pwm_res != 9 && pwm_res != 10)) {
        rtl_printf(" error: bad module parameters\n");
        return 1;
    }

    /* Create realtime fifos.*/
    if (rtf_create(CMD_FIFO_ID, 32)) {
        rtl_printf(" error: cannot create realtime fifo /dev/rtf%d\n", CMD_FIFO_ID);
        return 1;
    }
    if (rtf_create(ARG_FIFO_ID, 32)) {
        rtl_printf(" error: cannot create realtime fifo /dev/rtf%d\n", ARG_FIFO_ID);
        rtf_destroy(CMD_FIFO_ID);
        return 1;
    }
    if (rtf_create(MON_FIFO_ID, MON_FIFO_LEN*sizeof(struct monitor_data))) {
        rtl_printf(" error: cannot create realtime fifo /dev/rtf%d\n", MON_FIFO_ID);
        rtf_destroy(ARG_FIFO_ID);
        rtf_destroy(CMD_FIFO_ID);
        return 1;
    }

    /* Install fifo handler.*/
    if (rtf_create_handler(CMD_FIFO_ID, fifo_handler)) {
        rtl_printf(" error: cannot install fifo handler\n");
        rtf_destroy(MON_FIFO_ID);
        rtf_destroy(ARG_FIFO_ID);
        rtf_destroy(CMD_FIFO_ID);
        return 1;
    }

    /* Force rtl interrupts disabled.*/
    DISABLE_RTL_INTERRUPT;

    /* Install interrupt handler.*/
    if (rtl_request_irq(irq, interrupt_handler)) {
        RESTORE_RTL_INTERRUPT;
        rtl_printf(" error: cannot install interrupt handler\n");
        rtf_destroy(MON_FIFO_ID);
        rtf_destroy(ARG_FIFO_ID);
        rtf_destroy(CMD_FIFO_ID);
        return 1;
    }

    /* Enable interrupts for the board.*/
    ENABLE_BOARD_IRQ;

    /* Enable rtl interrupt.*/
    RESTORE_RTL_INTERRUPT;

    /* Send string with module parameters.*/
    rtl_printf(" at 0x%x, irq %d, %d bit pwm resolution\n", io, irq, pwm_res);
    rtl_printf("Using rt-fifos /dev/rtf%d, /dev/rtf%d, /dev/rtf%d\n", CMD_FIFO_ID, ARG_FIFO_ID, MON_FIFO_ID);
    return 0;
}

void cleanup_module(void)
{
    /* Disable board irq so that no interrupt is generated when restoring ports.*/
    DISABLE_BOARD_IRQ;

    /* Restore default hardware conditions.*/
    out_byte(_82C55_CtrlREG, 0x9A);          /* Port A and B input (side effect: enable motors).*/
}

```

```
out_byte(_82C55_PortC, 0x03);           /* Disable motors.*/

/* Remove interrupt handler.*/
rtl_free_irq(irq);

/* Destroy realtime fifos.*/
rtf_destroy(MON_FIFO_ID);
rtf_destroy(ARG_FIFO_ID);
rtf_destroy(CMD_FIFO_ID);

/* Send cleanup message to console (or to kernel).*/
rtl_printf("Removing module DCMMOD.0\n");
}
```

```

/*
** ISA DC Motor Control Board - Rev 1.x
**
** C/C++ Driver Library - Rev 1.0.2
**
** (C) by Flavio Cappelli
**
**
** NOTE: All errors are reported through the variables 'DCMotor_Error' and
** 'DCMotor_ProcErr' (see below). The error numbers are:
**
**      0  Everything ok.
**      1  Driver not initialized.
**      2  Invalid or not allowed command.
**      3  Cannot open realtime fifos (on user side).
**      4  Cannot send command to module driver or get valid data from it.
** -1, -2,..., -5: First, second,... fifth parameter is invalid.
**
** If error occurred subsequent operations are ignored.
**
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
*/
#ifndef _FC_DCMDRV_H
#define _FC_DCMDRV_H

#ifdef __cplusplus
extern "C" {
#endif

#ifndef __linux
#error this driver is only for LINUX
#endif

/* PWM Modes: DO NOT CHANGE FOLLOWING VALUES.*/
#define _PWM_AND_DIR      0
#define _PWM_ONLY         1

/* Motor definitions: DO NOT CHANGE FOLLOWING VALUES.*/
#define _Motor1           1
#define _Motor2           2

/* Port definitions: DO NOT CHANGE FOLLOWING VALUES.*/
#define _PortA            16
#define _PortB            2
#define _A_INP_B_INP     18
#define _A_OUT_B_INP     2
#define _A_INP_B_OUT     16
#define _A_OUT_B_OUT     0

/* A useful definition.*/
typedef unsigned char UBYTE;

/* Structure used to get monitor data.*/
struct monitor_data { long pos1, pos2; short vel1, vel2; };

/* Error number and procedure that generates error (or NULL).*/
extern int DCMotor_Error;
extern char *DCMotor_ProcErr;

/* Set manual mode and specify PWM dutycycle (within range [-1024, 1024]).*/
void DCMotor_SetDutyCycle(const short dutycycle_m1, const short dutycycle_m2);

/* Set velocity mode and specify velocity (step/servo time); fbits (0 <= fbits <= 4) */
/* specify how many bits of set velocities are fractional bits (for very low speed). */
/* SET VELOCITIES MUST BE LESS OR EQUAL TO MAX VELOCITIES THAT MOTORS CAN RUN. */
void DCMotor_SetVelocity(const short velocity_m1, const short velocity_m2, const short fbits);

/* Get current velocity.*/
void DCMotor_GetVelocity(short * const velocity_m1, short * const velocity_m2);

/* Reset origin position on motor(s) (_Motor1, _Motor2 or both if ORed).*/
void DCMotor_ResetOrigin(const int motor);

/* Set position mode and specify position (step).*/
void DCMotor_SetPosition(const long position_m1, const long position_m2);

```

```

/* Get current position.*/
void DCMotor_GetPosition(long * const position_m1, long * const position_m2);

/* Change PID gains on motor(s).*/
void DCMotor_SetPIDGains(const int motor, const short kp, const short ki, const short kd);

/* Change velocity limit and acceleration. If 'vlim' or 'acc' is  */
/* zero, the profile generator is disabled for specified motor(s).*/
/* TAKE INTO CONSIDERATION THE DINAMIC OF MOTOR AND LOAD.          */
void DCMotor_SetMotionParams(const int motor, const short vlim, const short acc);

/* Set board port value (_PortA or _PortB).*/
void DCMotor_SetPortValue (const int board_port, const UBYTE value);

/* Get board port value (_PortA or _PortB).*/
UBYTE DCMotor_GetPortValue(const int board_port);

/* Disable motor(s) (_Motor1, _Motor2 or both if ORed).*/
void DCMotor_DisableMotor(const int motor);

/* Enable motor(s) (_Motor1, _Motor2 or both if ORed).*/
void DCMotor_EnableMotor(const int motor);

/* Cleanup routine. SHOULD BE CALLED BEFORE PROGRAM TERMINATION.*/
void DCMotor_CleanupBoard(void);

/* Initialization routine. Set port direction and PWM mode:
/*  port_dir = _A_INP_B_INP, _A_OUT_B_INP, _A_INP_B_OUT,_A_OUT_B_OUT      */
/*  pwm_mode = _PWM_AND_DIR, _PWM_ONLY                                     */
/*
/* Following parameters are specified at module insertion:
/*  io        = 0x280, 0x290, 0x320, 0x330, 0x340, 0x350, 0x380, 0x390      */
/*  irq       = 10, 11, 15                                                 */
/*  pwm_res   = 8, 9, 10 (pwm resolution bits)                            */
/*
void DCMotor_SetupBoard(const int port_dir, const int pwm_mode);

/* Get last recorded position and velocity profile (current and generated) from */
/* monitor FIFO (if pointers are not NULL). Return number of saved data point. */
int DCMotor_GetMonitorData(const int max_point, struct monitor_data *cur, struct monitor_data *gen);

/* Discard last recorded position and velocity profile.*/
void DCMotor_ResetMonitorData(void);

/* Set max overcurrent time. 'cycles' is the number of servo update intervals that */
/* driver has to consider for a valid overcurrent condition (0 disable overcurrent */
/* check).If a valid overcurrent situation occurs BOTH motors are disabled and an */
/* 'overcurrent' status is set for motor(s). */
void DCMotor_SetMaxOvercurrentTime(const int motor, const short cycles);

/* Reset overcurrent status (if any) to 'normal' mode.*/
void DCMotor_ResetOvercurrentStatus(void);

/* Get overcurrent status: return _Motor1, _Motor2 (or both).*/
int DCMotor_GetOvercurrentStatus(void);

#ifndef __cplusplus
#endif

#endif /* _FC_DCMDRV_H */

```

```

/*
** DCMDRV.C - DC Motor Control Driver (user side).
**
** (C) by Flavio Cappelli
**
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
*/
#include <fcntl.h>
#include <unistd.h>
#include "dcmod.h"
#include "dcmdrv.h"

/* Macro: send command to realtime side of the driver and check acknowledgement (same sent command).*/
#define CMD_WRITE          {char ack=0; if (write(cmdfifo,(&outbuf),sizeof(outbuf))!=sizeof(outbuf) || read(argfifo,(&ack),1)!=1 || ack!=outbuf.cmd) {set_error(4); return;}}
#define CMD_WRITE2         {char ack=0; if (write(cmdfifo,(&outbuf),sizeof(outbuf))!=sizeof(outbuf) || read(argfifo,(&ack),1)!=1 || ack!=outbuf.cmd) {set_error(4); return retval;}}

/* Macro: get data from realtime side of the driver.*/
#define ARG_READ           if (read(argfifo,(&inbuf),sizeof(inbuf))!=sizeof(inbuf)) {set_error(4); return;}
#define ARG_READ2          if (read(argfifo,(&inbuf),sizeof(inbuf))!=sizeof(inbuf)) {set_error(4); return retval;}

/* Macro: check if driver has been initialized properly and no error occurred.*/
#define CHECK_DRV_OK        if (!is_drv_init) set_error(1); if (DCMotor_Error != 0) return;
#define CHECK_DRV_OK2       if (!is_drv_init) set_error(1); if (DCMotor_Error != 0) return retval;

/* Non-shared variables.*/
static int is_drv_init = 0;                                /* Flag: 1 = driver initialized. */
static int cmdfifo, argfifo, monfifo;                      /* Descriptors for realtime fifos. */
static char *proc_name = NULL;                             /* Name of procedure being executed.*/

/* Shared variable.*/
char *DCMotor_ProcErr = NULL;                            /* Procedure that generates error.*/
int DCMotor_Error = 0;                                    /* Error: 0 = no error. */

/********************* NON-SHARED PROCEDURES *****************/
static void set_error(const int err)
{
    /* Ignore error if it is not the first.*/
    if (DCMotor_Error == 0)
    {
        /* If driver has been initialized restore safe conditions.*/
        if (is_drv_init)
        {
            struct command outbuf;

            /* Inform realtime side of driver that an error occurred.*/
            outbuf.cmd = SET_ERR_AND_SC_COMMAND;           /* Send command string to RTLinux module */
            write(cmdfifo,(&outbuf),sizeof(outbuf));      /* (do not use CMD_WRITE to avoid loop */
                                                       /* on fifo errors. Note that this command */
                                                       /* does not require 'ack'). */
            /* Update error variables.*/
            DCMotor_ProcErr = proc_name;
            DCMotor_Error = err;
        }
    }
}

static char *dev_rtf(unsigned int n)                         /* Return "/dev/rtfN" string (N < 99).*/
{
    static char buf[16] = "/dev/rtf";
    unsigned int u, d, i = 8;

    u = n % 10;
    d = (n / 10) % 10;
    if (d > 0)
        *(buf + i++) = d + '0';
    *(buf + i++) = u + '0';
    *(buf + i) = 0;
    return buf;
}

```

```

***** SHARED PROCEDURES *****
void DCMotor_SetupBoard(const int port_dir, const int pwm_mode)
{
    char dummy1, ack;
    struct monitor_data dummy2;
    struct command_BB outbuf;
    proc_name = "DCMotor_SetupBoard";                                /* Save name of procedure */
                                                                /* (for error reporting). */
                                                                /* Everything ok.*/
    DCMotor_Error = 0;
    DCMotor_ProcErr = NULL;
    if (is_drv_init) {                                              /* Ensure only one initialization.*/
        set_error(2); return;                                         /* Command not allowed.*/
    }

    /* Check parameters.*/
    if (port_dir != _A_INP_B_INP && port_dir != _A_OUT_B_INP
        && port_dir != _A_INP_B_OUT && port_dir != _A_OUT_B_OUT) {
        set_error(-1); return;                                       /* First parameter invalid.*/
    }
    if (pwm_mode != _PWM_AND_DIR && pwm_mode != _PWM_ONLY) {
        set_error(-2); return;                                       /* Second parameter invalid.*/
    }

    /* Open realtime fifos.*/
    if ((cmdfifo = open(dev_rtf(CMD_FIFO_ID), O_WRONLY)) < 0) {
        set_error(3); return;                                         /* Error: cannot open realtime fifo.*/
    }
    if ((argfifo = open(dev_rtf(ARG_FIFO_ID), O_RDONLY | O_NONBLOCK)) < 0) {
        close (cmdfifo);                                            /* Close realtime fifo on error.*/
        set_error(3); return;
    }
    if ((monfifo = open(dev_rtf(MON_FIFO_ID), O_RDONLY | O_NONBLOCK)) < 0) {
        close (argfifo); close (cmdfifo);
        set_error(3); return;
    }

    /* Flush realtime fifos (if they are not empty).*/
    while (read(argfifo, &dummy1, sizeof(dummy1)) > 0);
    while (read(monfifo, &dummy2, sizeof(dummy2)) > 0);

    /* Initialize board and realtime side of driver.*/
    outbuf.cmd = SETUP_DRIVER_COMMAND;                               /* Send command string to RTLinux module */
    outbuf.val.arg1= (char)port_dir;                                 /* (do not use CMD_WRITE to close fifos */
    outbuf.val.arg2= (char)pwm_mode;                                /* on error. Note that this command and */
    if (write(cmdfifo,(&outbuf),sizeof(outbuf))!=sizeof(outbuf) /* all next commands require 'ack'). */
        || read(argfifo,(&ack),1)!=1 || ack!=outbuf.cmd) {
        close (monfifo); close (argfifo); close (cmdfifo);          /* Close realtime fifos on error.*/
        set_error(4); return;
    }

    /* Inform other procedures that driver has been initialized properly.*/
    is_drv_init = 1;
}

void DCMotor_CleanupBoard(void)
{
    char ack;
    struct command outbuf;
    proc_name = "DCMotor_CleanupBoard";                                /* Save name of procedure.*/
    if (is_drv_init) {                                              /* Check if driver has been */
                                                                /* initialized properly. */
                                                                /* Send command string to RTLinux module.*/
        outbuf.cmd = CLEANUP_DRIVER_COMMAND;
        if (write(cmdfifo,(&outbuf),sizeof(outbuf))!=sizeof(outbuf)
            || read(argfifo,(&ack),1)!=1 || ack!=outbuf.cmd) {
            set_error(4);
        }
        close (monfifo);                                           /* Close realtime fifos.*/
        close (argfifo);
        close (cmdfifo);
        is_drv_init = 0;                                            /* Ensure only one cleanup.*/
    }
    else set_error(1);                                             /* Driver not initialized.*/
}

void DCMotor_EnableMotor(const int motor)
{
}

```

```

struct command_B outbuf;
proc_name = "DCMotor_EnableMotor";                                /* Save name of procedure.*/

CHECK_DRV_OK;
if (motor > 0 && motor <= 3)                                         /* Check parameter.*/
{
    outbuf.cmd = ENABLE_MOTOR_COMMAND;
    outbuf.val.arg = (char)(motor);                                     /* Send command string to RTLinux module.*/
    CMD_WRITE;
}
else set_error(-1);                                                 /* Invalid parameter.*/
}

void DCMotor_DisableMotor(const int motor)
{
    struct command_B outbuf;
    proc_name = "DCMotor_DisableMotor";

    CHECK_DRV_OK;
    if (motor > 0 && motor <= 3)
    {
        outbuf.cmd = DISABLE_MOTOR_COMMAND;
        outbuf.val.arg = (char)(motor);
        CMD_WRITE;
    }
    else set_error(-1);
}

void DCMotor_SetDutyCycle(const short dutycycle_m1, const short dutycycle_m2)
{
    struct command_SS outbuf;
    proc_name = "DCMotor_SetDutyCycle";

    CHECK_DRV_OK;
    outbuf.cmd = SET_DUTYCYCLE_COMMAND;
    outbuf.val.arg1 = dutycycle_m1;
    outbuf.val.arg2 = dutycycle_m2;
    CMD_WRITE;
}

void DCMotor_GetVelocity(short * const velocity_m1, short * const velocity_m2)
{
    struct command outbuf;
    struct data_SS inbuf;
    proc_name = "DCMotor_GetVelocity";

    CHECK_DRV_OK;
    outbuf.cmd = GET_VELOCITY_COMMAND;
    CMD_WRITE;

    ARG_READ;                                                       /* Get data from RTLinux module */
    *velocity_m1 = inbuf.arg1;                                     /* (velocities of encoders). */
    *velocity_m2 = inbuf.arg2;
}

void DCMotorGetPosition(long * const position_m1, long * const position_m2)
{
    struct command outbuf;
    struct data_LL inbuf;
    proc_name = "DCMotor_GetPosition";

    CHECK_DRV_OK;
    outbuf.cmd = GET_POSITION_COMMAND;
    CMD_WRITE;

    ARG_READ;                                                       /* Get data from RTLinux module */
    *position_m1 = inbuf.arg1;                                     /* (positions of encoders). */
    *position_m2 = inbuf.arg2;
}

void DCMotor_SetVelocity(const short velocity_m1, const short velocity_m2, const short fbits)
{
    struct command_BSS outbuf;
    proc_name = "DCMotor_SetVelocity";
}

```

```

CHECK_DRV_OK;
if (fbits >= 0 && fbits <= 4)
{
    outbuf.cmd = SET_VELOCITY_COMMAND;
    outbuf.val.arg1 = (char)fbits;
    outbuf.val.arg2 = velocity_m1;
    outbuf.val.arg3 = velocity_m2;
    CMD_WRITE;
}
else set_error(-3);
}

void DCMotor_SetPosition(const long position_m1, const long position_m2)
{
    struct command_LL outbuf;
    proc_name = "DCMotor_SetPosition";

    CHECK_DRV_OK;
    outbuf.cmd = SET_POSITION_COMMAND;
    outbuf.val.arg1 = position_m1;
    outbuf.val.arg2 = position_m2;
    CMD_WRITE;
}

void DCMotor_ResetOrigin(const int motor)
{
    struct command_B outbuf;
    proc_name = "DCMotor_ResetOrigin";

    CHECK_DRV_OK;
    if (motor > 0 && motor <= 3)
    {
        outbuf.cmd = RESET_ORIGIN_COMMAND;
        outbuf.val.arg = (char)(motor);
        CMD_WRITE;
    }
    else set_error(-1);
}

void DCMotor_SetPIDGains(const int motor, const short kp, const short ki, const short kd)
{
    struct command_BSS outbuf;
    proc_name = "DCMotor_SetPIDGains";

    CHECK_DRV_OK;
    if (motor > 0 && motor <= 3)
    {
        outbuf.cmd = SET_PID_GAINS_COMMAND;
        outbuf.val.arg1 = (char)(motor);
        outbuf.val.arg2 = kp >= 0 ? kp : -kp; /* 'kp', 'ki' and 'kd' have to be positive.*/
        outbuf.val.arg3 = ki >= 0 ? ki : -ki;
        outbuf.val.arg4 = kd >= 0 ? kd : -kd;
        CMD_WRITE;
    }
    else set_error(-1);
}

void DCMotor_SetMotionParams(const int motor, const short vlim, const short acc)
{
    struct command_BSS outbuf;
    proc_name = "DCMotor_SetMotionParams";

    CHECK_DRV_OK;
    if (motor > 0 && motor <= 3)
    {
        outbuf.cmd = SET_ACC_AND_VLIM_COMMAND;
        outbuf.val.arg1 = (char)(motor);
        outbuf.val.arg2 = vlim >= 0 ? vlim : -vlim; /* 'vlim' and 'acc' have to be positive.*/
        outbuf.val.arg3 = acc >= 0 ? acc : -acc;
        CMD_WRITE;
    }
    else set_error(-1);
}

void DCMotor_SetPortValue(const int board_port, const UBYTE value)

```

```

{
    struct command_BB outbuf;
    proc_name = "DCMotor_SetPortValue";

    CHECK_DRV_OK;
    if (board_port == _PortA || board_port == _PortB)
    {
        outbuf.cmd = SET_PORT_VALUE_COMMAND;
        outbuf.val.arg1 = (char)(board_port);
        outbuf.val.arg2 = (char)(value);
        CMD_WRITE;
    }
    else set_error(-1);
}

UBYTE DCMotor_GetPortValue(const int board_port)
{
    struct command_B outbuf;
    struct data_B inbuf;
    UBYTE retval = 0;
    proc_name = "DCMotor_GetPortValue";

    CHECK_DRV_OK2;
    if (board_port == _PortA || board_port == _PortB)
    {
        outbuf.cmd = GET_PORT_VALUE_COMMAND; /* Send command string to RTLinux module.*/
        outbuf.val.arg = (char)(board_port);
        CMD_WRITE2;

        ARG_READ2;
        retval = (UBYTE)inbuf.arg; /* Get data from RTLinux module */
        /* (one byte from input port). */
    }
    else set_error(-1); /* Invalid parameter.*/
    return retval;
}

int DCMotor_GetMonitorData(const int max_point, struct monitor_data *cur, struct monitor_data *gen)
{
    struct monitor_data point;
    int retval = 0;
    proc_name = "DCMotor_GetMonitorData";

    CHECK_DRV_OK2; /* Get data from monitor fifo.*/
    if (max_point >= 0)
    {
        while (retval < max_point && read(monfifo, &point, sizeof(point)) > 0)
        {
            ++retval;
            if (cur) *cur++ = point; /* Save current point.*/

            read(monfifo, &point, sizeof(point));
            if (gen) *gen++ = point; /* Save generated point.*/
        }
    }
    else set_error(-1);
    return retval;
}

void DCMotor_ResetMonitorData(void)
{
    struct monitor_data dummy;
    proc_name = "DCMotor_ResetMonitorData";

    CHECK_DRV_OK;
    while (read(monfifo, &dummy, sizeof(dummy)) > 0); /* Discard monitor data.*/
}

void DCMotor_SetMaxOvercurrentTime(const int motor, const short cycles)
{
    struct command_BS outbuf;
    proc_name = "DCMotor_SetMaxOvercurrentTime";

    CHECK_DRV_OK;
    if (motor > 0 && motor <= 3)
    {
        outbuf.cmd = SET_OVERCETIME_COMMAND;
        outbuf.val.arg1 = (char)(motor);
    }
}

```

```
    outbuf.val.arg2 = cycles >= 0 ? cycles : -cycles;
    CMD_WRITE;
}
else set_error(-1);
}

void DCMotor_ResetOvercurrentStatus(void)
{
    struct command outbuf;
    proc_name = "DCMotor_ResetOvercurrentStatus";

    CHECK_DRV_OK;
    outbuf.cmd = RESET_OVERCSTATUS_COMMAND;
    CMD_WRITE;
}

int DCMotor_GetOvercurrentStatus(void)
{
    struct command outbuf;
    struct data_B inbuf;
    int retval = 0;
    proc_name = "DCMotor_GetOvercurrentStatus";

    CHECK_DRV_OK2;
    outbuf.cmd = GET_OVERCSTATUS_COMMAND;
    CMD_WRITE2;

    ARG_READ2;
    retval = inbuf.arg;
    return retval;
}
```

```

/*
** Mobile Robot Control Library - v1.0
**
** (C) by Flavio Cappelli
**
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
*/
#ifndef _FC_MROBOT_H
#define _FC_MROBOT_H

#ifdef __cplusplus
extern "C" {
#endif

#include "../dcmdrv.h"

/* User parameters.*/
#define SERVO_TIME      0.0016384          /* Servo update time (sec).*/
#define ENCOD_RES       500                /* Encoder resolution. */
#define PULSE_MAGN      4                  /* Pulse magnify (1X / 4X) */
#define GEAR_RATIO      19.7               /* Gear ratio (value > 1). */
#define WHEEL_DIAM      150                /* Diameter of wheels (mm). */
#define WHEEL_BASE      300                /* Distance between wheels.*/

/* Set velocity mode and specify velocity of wheels (mm/s).*/
/* TAKE INTO CONSIDERATION THE LIMITS OF MOTOR AND LOAD. */
void MRobot_SetWheelVelocity(const short v_right, const short v_left);

/* Get current velocity.*/
void MRobot_GetWheelVelocity(short * const v_right, short * const v_left);

/* Set velocity mode and specify robot linear velocity (mm/s) and angular velocity (deg/s).*/
void MRobot_SetRobotVelocity(const short v_lin, const short v_ang);

/* Get linear and angular velocity of robot.*/
void MRobot_GetRobotVelocity(short * const v_lin, short * const v_ang);

/* Change velocity limit and acceleration. If 'vlim' or 'acc' is */
/* zero, the profile generator is disabled for specified motor(s).*/
/* TAKE INTO CONSIDERATION THE DINAMIC OF ROBOT. */
void MRobot_SetMotionParams(const short vlim, const short acc);

/* Change PID gains on motors.*/
void MRobot_SetPIDGains(const short kp, const short ki, const short kd);

/* Disable motors.*/
void MRobot_DisableMotors(void);

/* Enable motors.*/
void MRobot_EnableMotors(void);

#endif /* _FC_MROBOT_H */

```

```
/*
** MROBOT.C - Mobile Robot Control Library.
**
** (C) by Flavio Cappelli
**
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
*/
#include "mrobot.h"

#define M_PI           3.14159265358979323846
#define RAD2DEG        (180 / M_PI)
#define DEG2RAD        (M_PI / 180)

/* Define this to exchange right motor with left motor.*/
/* #define XCHG_MOT */

/* This are to set motor polarity.*/
#define _RP(x)          +(x)
#define _LP(x)          -(x)

/* Conversion factor that translate encoder pulses into linear displacement (mm).*/
#define P_FACT          ((M_PI * WHEEL_DIAM) / (GEAR_RATIO * ENCOD_RES * PULSE_MAGN))
#define V_FACT          (P_FACT / SERVO_TIME)

/********************* NON-SHARED PROCEDURES ********************/

static short round_s(double v)
{
    v += .5;
    if (v < -32767) v = -32767;
    if (v > 32767) v = 32767;
    return (short)v;
}

/********************* SHARED PROCEDURES ********************/

void MRobot_SetWheelVelocity(const short v_right, const short v_left)
{
    int fbits = 0;
    double vR, vL;

    /* Translate velocities and shift them to get max precision.*/
    vR = v_right / V_FACT;
    vL = v_left / V_FACT;
    while (vR > -16383 && vR < 16383 && vL > -16383 && vL < 16383 && fbits < 4)
    {
        vR *= 2; vL *= 2; ++fbits;
    }

    /* Set wheel velocities.*/
    #ifdef XCHG_MOT
        DCMotor_SetVelocity( _RP(round_s(vR)), _LP(round_s(vL)), fbits);
    #else
        DCMotor_SetVelocity( _LP(round_s(vL)), _RP(round_s(vR)), fbits);
    #endif
}

void MRobot_GetWheelVelocity(short * const v_right, short * const v_left)
{
    short vR, vL;

    /* Get wheel velocities.*/
    #ifdef XCHG_MOT
        DCMotor_GetVelocity(&vR, &vL);
    #else
        DCMotor_GetVelocity(&vL, &vR);
    #endif

    /* Translate velocities.*/
}
```

```

if (v_right)
    *v_right = _RP(round_s(vR * V_FACT));
if (v_left)
    *v_left = _LP(round_s(vL * V_FACT));
}

void MRobot_SetRobotVelocity(const short v_lin, const short v_ang)
{
    int fbits = 0;
    double vR, vL;

    /* Translate velocities and shift them to get max precision.*/
    vR = (2 * v_lin + WHEEL_BASE * DEG2RAD * v_ang) / (2 * V_FACT);
    vL = (2 * v_lin - WHEEL_BASE * DEG2RAD * v_ang) / (2 * V_FACT);
    while (vR > -16383 && vR < 16383 && vL > -16383 && vL < 16383 && fbits < 4)
    {
        vR *= 2; vL *= 2; ++fbits;
    }

    /* Set wheel velocities.*/
    #ifdef XCHG_MOT
        DCMotor_SetVelocity( _RP(round_s(vR)), _LP(round_s(vL)), fbits);
    #else
        DCMotor_SetVelocity( _LP(round_s(vL)), _RP(round_s(vR)), fbits);
    #endif
}

void MRobot_GetRobotVelocity(short * const v_lin, short * const v_ang)
{
    short vR, vL;

    /* Get wheel velocities.*/
    #ifdef XCHG_MOT
        DCMotor_GetVelocity(&vR, &vL);
    #else
        DCMotor_GetVelocity(&vL, &vR);
    #endif

    /* Translate velocities.*/
    if (v_lin)
        *v_lin = round_s(( _RP(vR) + _LP(vL)) * (V_FACT / 2));
    if (v_ang)
        *v_ang = round_s(( _RP(vR) - _LP(vL)) * (V_FACT * RAD2DEG / WHEEL_BASE));
}

void MRobot_SetMotionParams(const short vlim, const short acc)
{
    DCMotor_SetMotionParams(_Motor1 | _Motor2, round_s(vlim / V_FACT), round_s(acc / V_FACT));
}

void MRobot_SetPIDGains(const short kp, const short ki, const short kd)
{
    DCMotor_SetPIDGains(_Motor1 | _Motor2, kp, ki, kd);
}

void MRobot_DisableMotors(void)
{
    DCMotor_DisableMotor(_Motor1 | _Motor2);
}

void MRobot_EnableMotors(void)
{
    DCMotor_EnableMotor(_Motor1 | _Motor2);
}

```

Appendice A5

DATA SHEETS

Di seguito riportiamo la documentazione tecnica relativa ai motori, ai circuiti integrati ed ai componenti di potenza utilizzati nel progetto.

Nell'ordine abbiamo:

- 1. Motori GM9236**
- 2. 82C54**
- 3. 82C55A**
- 4. GAL16V8**
- 5. MM74HC00**
- 6. MM74HC14**
- 7. MM74HC123A**
- 8. DM74LS245**
- 9. MM74HC4060**
- 10. HIP4081A**
- 11. IRF520N**
- 12. LM393**
- 13. 6N137**
- 14. TLP521-1**
- 15. BA10T**
- 16. L7805**

BULLETIN LCG
Series GM8000,
GM9000, GM14900

LO-COG® DC Gearmotors



Pittman brand LO-COG® brush-commutated DC gearmotors offer smooth, quiet operation and long life. LO-COG gearmotors feature sintered steel spur gears and are available with several reduction ratios and torque ratings to provide an economical solution for a wide range of applications. Armatures are skewed to minimize magnetic cogging, even at low speeds, and windings are resin impregnated for greater reliability in incremental motion applications. An innovative cartridge brush assembly reduces audible and electrical noise and significantly improves brush life by maintaining optimum brush force throughout the life of the motor. For precision motor control, Hewlett-Packard® optical encoders are available in 2 or 3 channel versions with several CPR ranges to meet your position, velocity and direction feedback needs.

Construction

- 2 pole permanent magnet stators are constructed of ceramic magnets enclosed in heavy-gauge steel return rings
- Diamond turned commutators ensure maximum brush life
- Standard copper graphite brushes (Other brush materials available)
- Precision ground hardened stainless steel shafts
- Silicon-steel laminations
- Self-aligning, sintered bronze bearings

Options

- High-torque gears or high-torque wide-face gears
- Low noise, primary cluster gears
- Custom cables
- Multiple shaft configurations
- Shaft-mounted pulleys and gears
- Ball bearings
- Multiple windings
- Electromechanical brakes
- Integrated Hewlett-Packard® optical encoders
- Adaptors available for other encoders
- RFI suppression
- Dynamic armature balancing

Series GM8000

- 11 ratios from 6.3:1 to 1803:6:1
- Peak Torques to 100 oz-in standard
- 160 and 175 oz-in gearheads optional
- Available in 3 motor lengths
- Encoder resolutions from 96 to 1024

Series GM9000

- 12 ratios from 5.9:1 to 4732:5:1
- Peak Torques to 175 oz-in standard
- 300 and 500 oz-in gearheads optional
- Available in 6 motor lengths
- Encoder resolutions from 96 to 2048

Series GM14000

- 4 ratios from 5.9:1 to 218:4:1
- Peak Torques to 175 oz-in standard
- 300 and 500 oz-in gearheads optional
- Available in 7 motor lengths
- Encoder resolutions from 96 to 2048

PITTMAN®
Power Your Ideas™



Get same day shipment of sample motors for models listed in the Pittman Express Catalog (Bulletin PE).

Every Pittman motor is subjected to automated performance testing prior to shipment.

SERIES GM 8000

Gearmotor Data

Line No.	Parameter	Symbol	Units	Reduction Ratios											
				6.3:1	9.9:1	19.5:1	30.9:1	60.5:1	95.9:1	187.7:1	297.5:1	581.8:1	922.3:1	1803.6:1	
MECHANICAL SPECIFICATIONS															
1	Max. Load Standard Gears ¹	T _L	oz-in (N·m)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)
2	Max. Load Cut Steel Gears ¹	T _L	oz-in (N·m)	N/A (N/A)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)
3	Max. Load Wide Face Gears ¹	T _L	oz-in (N·m)	N/A (N/A)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)
4	Gearbox Shaft Rotation ²	—	—	CW	CCW	CCW	CW	CW	CCW	CW	CW	CCW	CW	CCW	CCW
5	Gearbox Efficiency	—	%	81	73	73	66	66	59	59	53	53	48	48	48
6	Gearbox Weight	W _G	oz (gm)	2.35 (66.6)	2.49 (70.6)	2.49 (70.6)	2.62 (74.3)	2.62 (74.3)	2.76 (78.2)	2.76 (78.2)	3.11 (88.2)	3.11 (88.2)	3.25 (92.1)	3.25 (92.1)	3.25 (92.1)
7	Gearbox Length	L ₂	in max (mm max)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	1.164 (29.6)	1.164 (29.6)	1.164 (29.6)	1.164 (29.6)	1.164 (29.6)
8	Length, GM82X2	L ₃	in max (mm max)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	3.173 (80.6)	3.173 (80.6)	3.173 (80.6)	3.173 (80.6)	3.173 (80.6)
9	Length, GM82X3	L ₃	in max (mm max)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.298 (83.8)	3.298 (83.8)	3.298 (83.8)	3.298 (83.8)	3.298 (83.8)
10	Length, GM82X4	L ₃	in max (mm max)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.548 (90.1)	3.548 (90.1)	3.548 (90.1)	3.548 (90.1)	3.548 (90.1)
11	Length, GM87X2	L ₃	in max (mm max)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	3.106 (78.9)	3.106 (78.9)	3.106 (78.9)	3.106 (78.9)	3.106 (78.9)
12	Length, GM87X3	L ₃	in max (mm max)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.231 (82.1)	3.231 (82.1)	3.231 (82.1)	3.231 (82.1)	3.231 (82.1)
13	Length, GM87X4	L ₃	in max (mm max)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.481 (88.4)	3.481 (88.4)	3.481 (88.4)	3.481 (88.4)	3.481 (88.4)
NO-LOAD SPEED															
14	GM8X12	S _{NL}	rpm (rad/s)	1227 (128)	774 (81.1)	396 (41.5)	250 (26.2)	128 (13.4)	80.5 (8.43)	41.2 (4.31)	26.0 (2.72)	13.3 (1.39)	8.38 (.877)	4.29 (.449)	
15	GM8X22	S _{NL}	rpm (rad/s)	1246 (130)	786 (82.3)	402 (42.1)	253 (26.5)	130 (13.6)	81.8 (8.57)	41.8 (4.38)	26.4 (2.76)	13.5 (1.41)	8.51 (.891)	4.35 (.456)	
16	GM8X13	S _{NL}	rpm (rad/s)	1308 (137)	825 (86.4)	422 (44.2)	266 (27.9)	136 (14.2)	85.8 (8.98)	43.9 (4.60)	27.7 (2.90)	14.2 (1.49)	8.93 (.935)	4.57 (.479)	
17	GM8X23	S _{NL}	rpm (rad/s)	1317 (138)	831 (87.0)	425 (44.5)	268 (28.1)	137 (14.3)	86.5 (9.06)	44.2 (4.63)	27.9 (2.92)	14.3 (1.50)	9.00 (.942)	4.60 (.482)	
18	GM8X14	S _{NL}	rpm (rad/s)	1690 (177)	1066 (112)	545 (57.1)	344 (36.0)	176 (18.4)	111 (11.6)	56.7 (5.94)	35.8 (3.75)	18.3 (1.92)	11.5 (1.20)	5.90 (.618)	
19	GM8X24	S _{NL}	rpm (rad/s)	1612 (169)	1017 (107)	520 (54.5)	328 (34.3)	168 (17.6)	106 (11.1)	54.1 (5.67)	34.1 (3.57)	17.5 (1.83)	11.0 (1.15)	5.63 (.590)	

¹Represents gearbox capability only. Continuous load torque capability will vary with gear ratio, motor selection, and operating conditions.

²Shaft rotation is designated while looking at output shaft with motor operating in a clockwise direction.

Motor Data

Line No.	Parameter	Symbol	Units	8X12	8X22	8X13	8X23	8X14	8X24
20	Continuous Torque (Max.) ³	T _C	oz-in (N·m)	1.3 (8.90 X 10 ⁻³)	1.6 (11.2 X 10 ⁻³)	1.5 (10.7 X 10 ⁻³)	2.0 (14.1 X 10 ⁻³)	2.1 (14.5 X 10 ⁻³)	2.6 (18.5 X 10 ⁻³)
21	Peak Torque (Stall)	T _{PK}	oz-in (N·m)	5.1 (35.7 X 10 ⁻³)	7.4 (52.0 X 10 ⁻³)	6.8 (47.7 X 10 ⁻³)	10.5 (74.2 X 10 ⁻³)	11.9 (84.0 X 10 ⁻³)	16.8 (118.6 X 10 ⁻³)
22	Motor Constant	K _M	oz-in/√W (N·m/√W)	0.93 (6.6 X 10 ⁻³)	1.12 (7.9 X 10 ⁻³)	1.05 (7.4 X 10 ⁻³)	1.30 (9.2 X 10 ⁻³)	1.22 (8.6 X 10 ⁻³)	1.49 (710.5 X 10 ⁻³)
23	No-Load Speed	S ₀	rpm (rad/s)	7729 (809)	7847 (822)	8238 (863)	8298 (869)	10648 (1115)	10158 (1064)
24	Friction Torque	T _F	oz-in (N·m)	0.35 (2.5 X 10 ⁻³)					
25	Rotor Inertia	J _M	oz-in-s ² (kg·m ²)	1.3 X 10 ⁻⁴ (9.18 X 10 ⁻⁷)	1.4 X 10 ⁻⁴ (9.89 X 10 ⁻⁷)	1.6 X 10 ⁻⁴ (1.13 X 10 ⁻⁶)	1.7 X 10 ⁻⁴ (1.20 X 10 ⁻⁶)	2.2 X 10 ⁻⁴ (1.55 X 10 ⁻⁶)	2.3 X 10 ⁻⁴ (1.62 X 10 ⁻⁶)

³Continuous torque specified at 25°C ambient temperature and without additional heat sink.



Motor Data, continued

Line No.	Parameter	Symbol	Units	8X12	8X22	8X13	8X23	8X14	8X24
26	Electrical Time Constant	τ_E	ms	0.50	0.52	0.48	0.55	0.54	0.54
27	Mechanical Time Constant	τ_M	ms	21.5	15.6	21.0	14.1	21.0	14.7
28	Viscous Damping— Infinite Source Impedance	D	oz-in/krpm (N·m/(rad/s))	0.0087 (129.02)	0.0153 (226.89)	0.0104 (154.23)	0.0176 (261.01)	0.0147 (218.00)	0.0202 (299.57)
29	Viscous Damping— Zero Source Impedance	K_D	oz-in/krpm (N·m/(rad/s))	0.64 (4.31 X 10 ⁻⁵)	0.92 (6.20 X 10 ⁻⁵)	0.81 (5.46 X 10 ⁻⁵)	1.25 (8.43 X 10 ⁻⁵)	1.10 (7.42 X 10 ⁻⁵)	1.63 (1.10 X 10 ⁻⁴)
30	Maximum Winding Temperature	θ_{MAX}	°F (°C)	311 (155)	311 (155)	311 (155)	311 (155)	311 (155)	311 (155)
31	Thermal Impedance	R_{TH}	°F/watt °C/watt	75.9 (24.4)	75.9 (24.4)	72.9 (22.7)	72.9 (22.7)	70.5 (21.4)	70.5 (21.4)
32	Thermal Time Constant	τ_{TH}	min	7.75	7.75	9.00	9.00	10.70	10.70
33	Motor Weight	W_M	oz (gm)	4.49 (127.3)	4.69 (133.0)	4.86 (137.8)	5.05 (143.2)	5.62 (159.3)	5.81 (164.7)

Model GM8XX2 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	GM8X12				GM8X22			
				12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
34	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
35	Torque Constant	K_T	oz·in/A (N·m/A)	1.94 (13.7 X 10 ⁻³)	3.06 (21.7 X 10 ⁻³)	3.87 (27.3 X 10 ⁻³)	4.89 (34.6 X 10 ⁻³)	1.94 (13.7 X 10 ⁻³)	3.07 (21.7 X 10 ⁻³)	3.88 (27.4 X 10 ⁻³)	4.88 (34.5 X 10 ⁻³)
36	Back-EMF Constant	K_E	V/krpm (V/rad/s)	1.43 (13.7 X 10 ⁻³)	2.27 (21.7 X 10 ⁻³)	2.86 (27.3 X 10 ⁻³)	3.62 (34.6 X 10 ⁻³)	1.43 (13.7 X 10 ⁻³)	2.27 (21.7 X 10 ⁻³)	2.87 (27.4 X 10 ⁻³)	3.61 (34.5 X 10 ⁻³)
37	Resistance	R_T	Ω	4.38	10.80	17.20	27.3	3.10	7.61	12.1	19.1
38	Inductance	L	mH	2.15	5.40	8.62	13.8	1.57	3.93	6.27	9.92
39	No-Load Current	I_{NL}	A	0.22	0.14	0.11	0.09	0.25	0.16	0.12	0.10
40	Peak Current (Stall) ⁴	I_p	A	2.74	1.76	1.40	1.11	3.88	2.51	1.99	1.59

Model GM8XX3 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	GM8X13				GM8X23			
				12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
41	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
42	Torque Constant	K_T	oz·in/A (N·m/A)	1.85 (13.1 X 10 ⁻³)	2.95 (20.8 X 10 ⁻³)	3.70 (26.1 X 10 ⁻³)	4.67 (33.0 X 10 ⁻³)	1.88 (13.3 X 10 ⁻³)	2.94 (20.8 X 10 ⁻³)	3.73 (26.4 X 10 ⁻³)	4.71 (33.3 X 10 ⁻³)
43	Back-EMF Constant	K_E	V/krpm (V/rad/s)	1.37 (13.1 X 10 ⁻³)	2.18 (20.8 X 10 ⁻³)	2.73 (26.1 X 10 ⁻³)	3.45 (33.0 X 10 ⁻³)	1.39 (13.3 X 10 ⁻³)	2.18 (20.8 X 10 ⁻³)	2.76 (26.4 X 10 ⁻³)	3.48 (33.3 X 10 ⁻³)
44	Resistance	R_T	Ω	3.20	7.94	12.5	19.8	2.17	5.20	8.24	13.1
45	Inductance	L	mH	1.48	3.78	5.93	9.46	1.17	2.85	4.57	7.29
46	No-Load Current	I_{NL}	A	0.24	0.15	0.12	0.09	0.27	0.17	0.13	0.11
47	Peak Current (Stall) ⁴	I_p	A	3.75	2.40	1.92	1.53	5.54	3.67	2.91	2.32

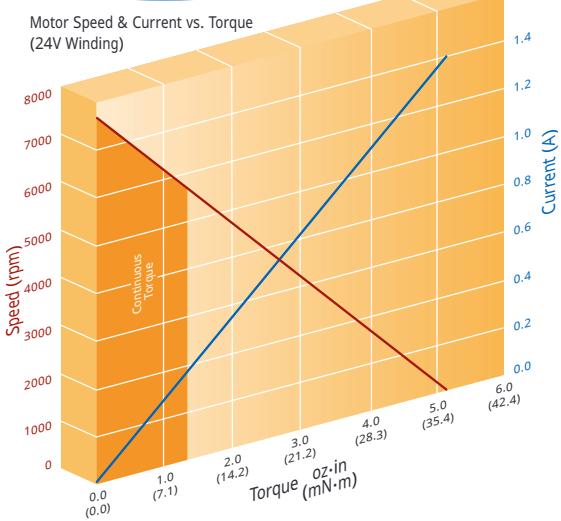
Model GM8XX4 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	GM8X14				GM8X24			
				12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
48	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
49	Torque Constant	K_T	oz·in/A (N·m/A)	1.46 (10.3 X 10 ⁻³)	2.31 (16.3 X 10 ⁻³)	2.92 (20.6 X 10 ⁻³)	3.69 (26.1 X 10 ⁻³)	1.54 (10.9 X 10 ⁻³)	2.47 (17.5 X 10 ⁻³)	3.09 (21.9 X 10 ⁻³)	3.86 (27.3 X 10 ⁻³)
50	Back-EMF Constant	K_E	V/krpm (V/rad/s)	1.08 (10.3 X 10 ⁻³)	1.71 (16.3 X 10 ⁻³)	2.16 (20.6 X 10 ⁻³)	2.73 (26.1 X 10 ⁻³)	1.14 (10.9 X 10 ⁻³)	1.83 (17.5 X 10 ⁻³)	2.29 (21.9 X 10 ⁻³)	2.86 (27.3 X 10 ⁻³)
51	Resistance	R_T	Ω	1.52	3.64	5.73	9.06	1.17	2.79	4.33	6.75
52	Inductance	L	mH	0.77	1.94	3.10	4.95	0.58	1.50	2.34	3.65
53	No-Load Current	I_{NL}	A	0.35	0.22	0.18	0.14	0.36	0.23	0.18	0.15
54	Peak Current (Stall) ⁴	I_p	A	7.90	5.24	4.19	3.34	10.3	6.85	5.54	4.49

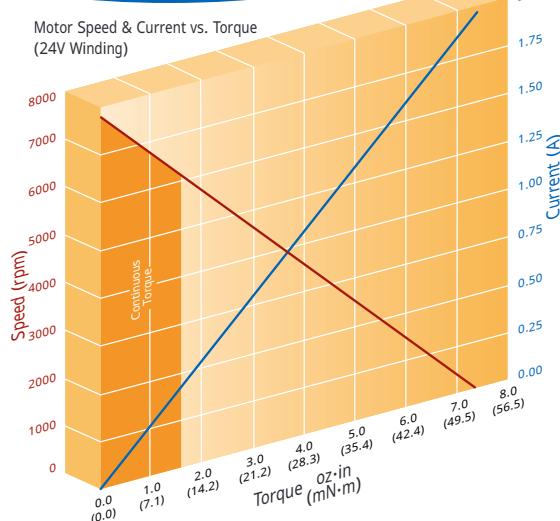
⁴Theoretical values supplied for reference only.

SERIES GM 8000

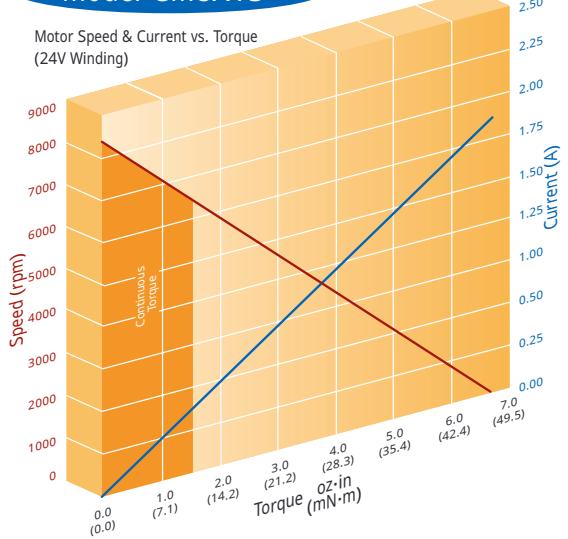
Model GM8X12



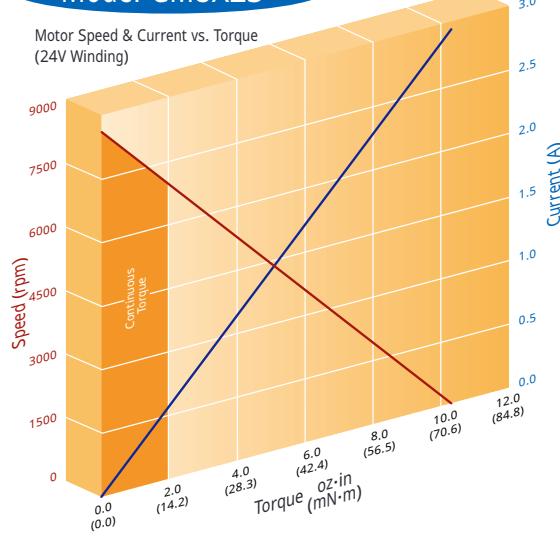
Model GM8X22



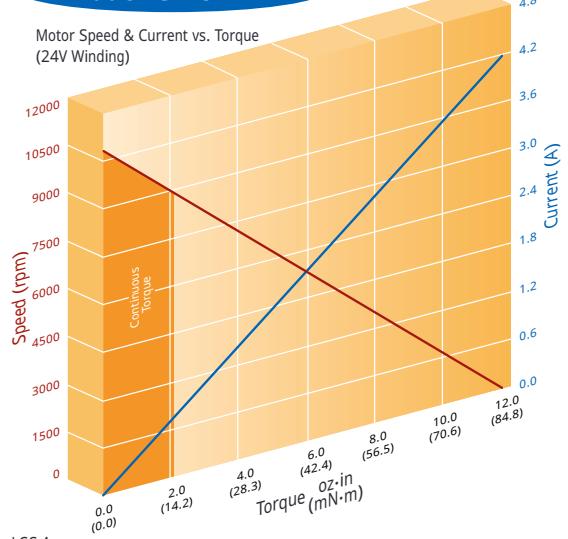
Model GM8X13



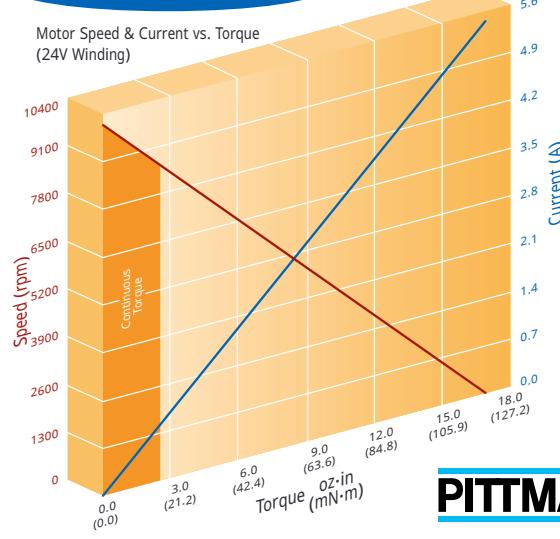
Model GM8X23



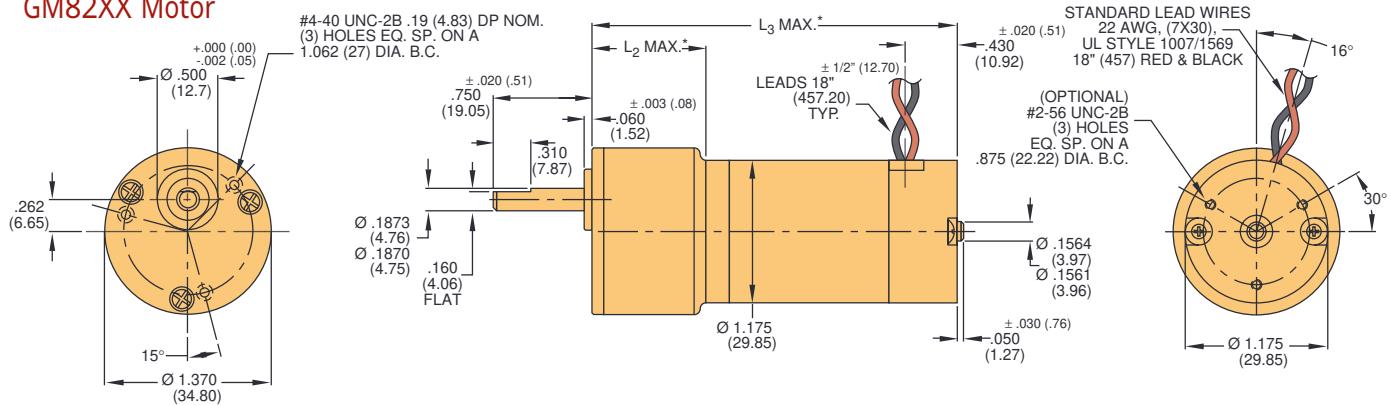
Model GM8X14



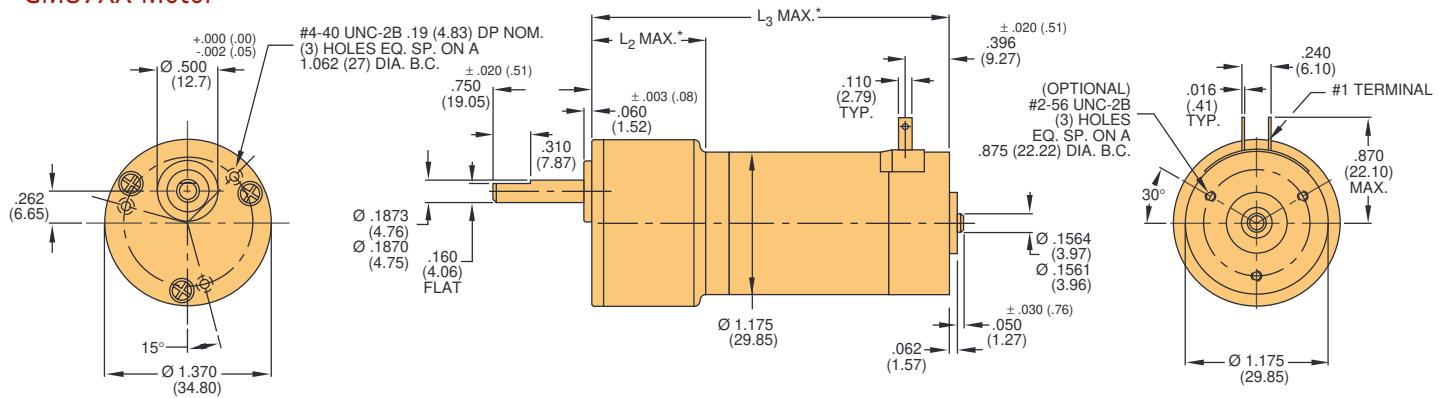
Model GM8X24



GM82XX Motor



GM87XX Motor



Notes:

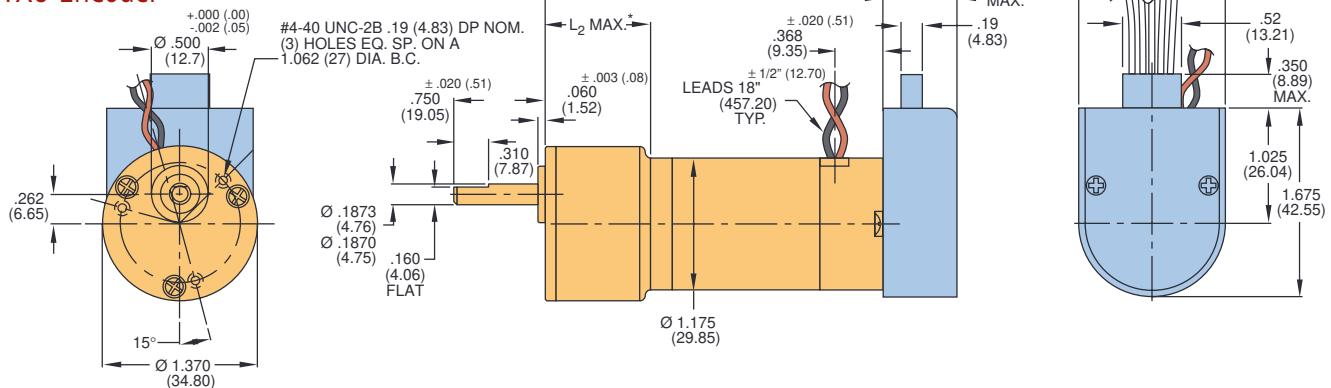
- NOTES:

 - Unless otherwise specified, all tolerances are to be $\pm .005$ (.01)
 - All measurements are in inches (mm)

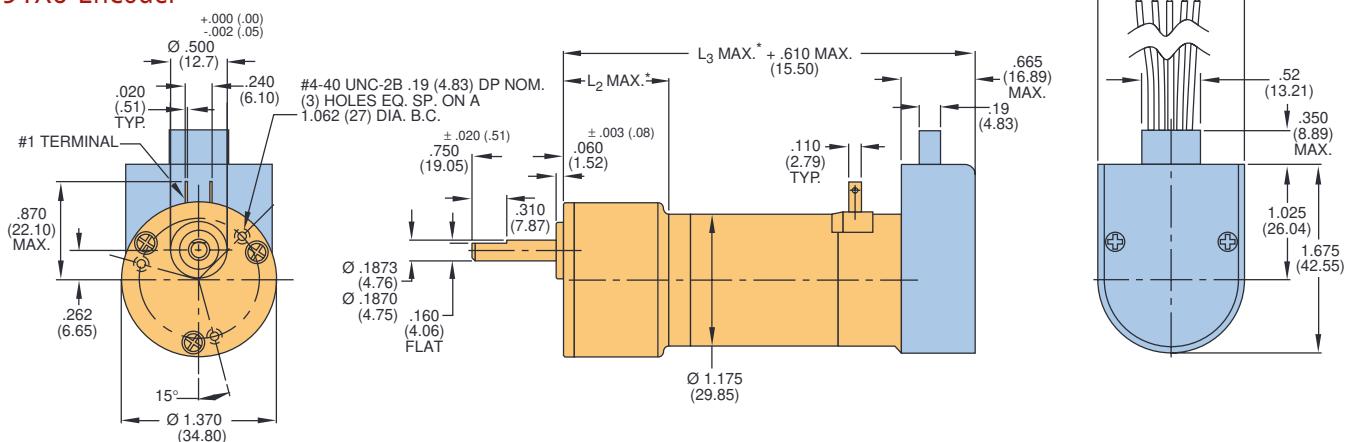
*See line numbers 7 through 13 in gearmotor data chart

SERIES GM8000

GM82XX Motor with 91X0 Encoder



GM87XX Motor with 91X0 Encoder



Encoder Connection Chart

Pin No.	Color	Connection
1	Black	Ground
2	Green	Index/NC
3	Yellow	Channel A
4	Red	Vcc
5	Blue	Channel B

Notes:

- Unless otherwise specified, all tolerances are to be ±.005 (.01)
- All measurements are in inches (mm)

*See line numbers 7 through 13 in gearmotor data chart

PITTMAN®

PITTMAN®

SERIES GM 9000

Gearmotor Data

Line No.	Parameter	Symbol	Units	Reduction Ratios												
				5.9:1	11.5:1	19.7:1	38.3:1	65.5:1	127.8:1	218.4:1	425.9:1	728.1:1	1419.8:1	2426.9:1	4732.5:1	
MECHANICAL SPECIFICATIONS (Standard and High-Torque Gears)																
1	Max. Load Standard Gears ¹	T _L	oz-in (N·m)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	
2	Max. Load High-Torque Gears ¹	T _L	oz-in (N·m)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	300 (2.12)	
3	Gearbox Shaft Rotation ²	—	—	CW	CW	CCW	CCW	CW	CW	CCW	CCW	CW	CW	CCW	CCW	
4	Gearbox Efficiency	—	%	81	81	73	73	66	66	59	59	53	53	48	48	
5	Gearbox Weight	W _G	oz (gm)	5.90 (167.3)	5.90 (167.3)	6.26 (177.5)	6.26 (177.5)	6.62 (187.7)	6.62 (187.7)	6.98 (197.9)	6.98 (197.9)	7.34 (208.1)	7.34 (208.1)	8.18 (231.9)	8.18 (231.9)	
6	Gearbox Length	L ₂	in max (mm max)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.373 (34.9)	1.528 (38.8)	1.528 (38.8)	
7	Length, GM92X2/GM94X2	L ₃	in max (mm max)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.101 (78.8)	3.256 (82.7)	3.256 (82.7)	
8	Length, GM92X3/GM94X3	L ₃	in max (mm max)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.476 (88.3)	3.631 (92.2)	3.631 (92.2)	
9	Length, GM92X4/GM94X4	L ₃	in max (mm max)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.676 (93.4)	3.831 (97.3)	3.831 (97.3)	
10	Length, GM9235/GM9435	L ₃	in max (mm max)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	3.976 (101.0)	4.131 (104.9)	4.131 (104.9)	
11	Length, GM9236/GM9436	L ₃	in max (mm max)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.326 (109.9)	4.481 (113.8)	4.481 (113.8)	4.481 (113.8)	
MECHANICAL SPECIFICATIONS (High-Torque Wide Face Gears)																
12	Max. Load ¹	T _L	oz-in (N·m)	500 (3.53)												
13	Gearbox Shaft Rotation ²	—	—	CCW CCW CW CW CCW CCW CW CW												
14	Gearbox Efficiency	—	%	73 73 66 66 59 59 53 53												
15	Gearbox Weight	W _G	oz (gm)	6.52 (184.8) 6.52 (184.8) 6.88 (195.0) 6.88 (195.0) 7.24 (205.3) 7.24 (205.3) 8.08 (229.1) 8.08 (229.1)												
16	Gearbox Length	L ₂	in max (mm max)	1.373 (34.9) 1.373 (34.9) 1.373 (34.9) 1.373 (34.9) 1.373 (34.9) 1.373 (34.9) 1.528 (38.8) 1.528 (38.8)												
17	Length, GM92X2/GM94X2	L ₃	in max (mm max)	3.101 (78.8) 3.101 (78.8) 3.101 (78.8) 3.101 (78.8) 3.101 (78.8) 3.101 (78.8) 3.256 (82.7) 3.256 (82.7)												
18	Length, GM92X3/GM94X3	L ₃	in max (mm max)	3.476 (88.3) 3.476 (88.3) 3.476 (88.3) 3.476 (88.3) 3.476 (88.3) 3.476 (88.3) 3.631 (92.2) 3.631 (92.2)												
19	Length, GM92X4/GM94X4	L ₃	in max (mm max)	3.676 (93.4) 3.676 (93.4) 3.676 (93.4) 3.676 (93.4) 3.676 (93.4) 3.676 (93.4) 3.831 (97.3) 3.831 (97.3)												
20	Length, GM9235/GM9435	L ₃	in max (mm max)	3.976 (101.1) 3.976 (101.1) 3.976 (101.1) 3.976 (101.1) 3.976 (101.1) 3.976 (101.1) 4.131 (104.9) 4.131 (104.9)												
21	Length, GM9236/GM9436	L ₃	in max (mm max)	4.326 (109.9) 4.326 (109.9) 4.326 (109.9) 4.326 (109.9) 4.326 (109.9) 4.326 (109.9) 4.481 (113.8) 4.481 (113.8)												
NO-LOAD SPEED (All Gears)																
22	GM9X12	S _{NL}	rpm (rad/s)	1399 (147)	717 (75.1)	420 (44.0)	215 (22.5)	126 (13.2)	64.6 (6.76)	37.8 (3.96)	19.4 (2.03)	11.3 (1.18)	5.8 (.607)	3.4 (.356)	1.7 (.178)	
23	GM9X32	S _{NL}	rpm (rad/s)	1189 (125)	610 (63.9)	357 (37.4)	183 (19.2)	107 (11.2)	54.9 (5.75)	32.1 (3.36)	16.5 (1.73)	9.6 (.513)	4.9 (.304)	2.9 (.157)		
24	GM9X13	S _{NL}	rpm (rad/s)	948 (99.3)	486 (50.9)	284 (29.7)	146 (15.3)	85.3 (8.93)	43.8 (4.59)	25.6 (2.68)	13.1 (1.37)	7.7 (.806)	3.9 (.408)	2.3 (.241)	1.2 (.126)	
25	GM9X33	S _{NL}	rpm (rad/s)	1016 (106)	521 (54.6)	305 (31.9)	156 (16.3)	91.5 (9.58)	46.9 (4.91)	27.4 (2.87)	14.1 (1.48)	8.2 (.859)	4.2 (.440)	2.5 (.262)	1.3 (.136)	
26	GM9X14	S _{NL}	rpm (rad/s)	1300 (136)	667 (69.8)	390 (40.8)	200 (20.9)	117 (12.3)	60.0 (6.28)	35.1 (3.68)	18.0 (1.88)	10.5 (1.10)	5.4 (.565)	3.2 (.335)	1.6 (.168)	
27	GM9X34	S _{NL}	rpm (rad/s)	1043 (109)	535 (56.0)	313 (32.8)	160 (16.8)	93.9 (9.83)	48.1 (5.04)	28.2 (2.95)	14.4 (1.51)	8.5 (.890)	4.3 (.450)	2.5 (.262)	1.3 (.136)	

¹Represents gearbox capability only. Continuous load torque capability will vary with gear ratio, motor selection, and operating conditions.

²Shaft rotation is designated while looking at output shaft with motor operating in a clockwise direction.

PITTMAN

Gearmotor Data, continued

Line No.	Parameter	Symbol	Units	Reduction Ratios											
				5.9:1	11.5:1	19.7:1	38.3:1	65.5:1	127.8:1	218.4:1	425.9:1	728.1:1	1419.8:1	2426.9:1	4732.5:1
NO-LOAD SPEED (All Gears), continued															
28	GM9X35	S _{NL}	rpm (rad/s)	1075 (112.5)	552 (57.8)	322 (33.7)	166 (17.4)	96.9 (10.1)	49.7 (5.2)	29.1 (3.0)	14.9 (1.6)	8.7 (.913)	4.4 (.468)	2.6 (.274)	1.3 (.140)
29	GM9X36	S _{NL}	rpm (rad/s)	834 (87.3)	427 (44.7)	250 (26.2)	128 (13.4)	75 (7.85)	38.5 (4.03)	22.5 (2.36)	11.5 (1.20)	6.8 (.712)	3.5 (.367)	2.0 (.209)	1.0 (.105)

Motor Data

Line No.	Parameter	Symbol	Units	GM9X12	GM9X32	GM9X13	GM9X33	GM9X14	GM9X34	GM9X35	GM9X36
30	Continuous Torque (Max.) ³	T _C	oz-in (N·m)	1.6 (11.2 X 10 ⁻³)	2.3 (16.2 X 10 ⁻³)	3.2 (22.6 X 10 ⁻³)	4.7 (33.2 X 10 ⁻³)	3.9 (27.8 X 10 ⁻³)	6.1 (43.1 X 10 ⁻³)	6.9 (48.7 X 10 ⁻³)	9.5 (67.1 X 10 ⁻³)
31	Peak Torque (Stall) ⁴	T _{PK}	oz-in (N·m)	8.4 (59.0 X 10 ⁻³)	13.8 (97.5 X 10 ⁻³)	15.6 (110.2 X 10 ⁻³)	31.6 (223.2 X 10 ⁻³)	23.9 (168.8 X 10 ⁻³)	41.3 (291.7 X 10 ⁻³)	49.4 (348.9 X 10 ⁻³)	61.8 (436.4 X 10 ⁻³)
32	Motor Constant	K _M	oz-in/√W (N·m/√W)	1.16 (8.2 X 10 ⁻³)	1.62 (11.4 X 10 ⁻³)	1.94 (13.7 X 10 ⁻³)	2.66 (18.8 X 10 ⁻³)	2.05 (14.5 X 10 ⁻³)	3.01 (21.3 X 10 ⁻³)	3.21 (22.7 X 10 ⁻³)	4.11 (29.0 X 10 ⁻³)
33	No-Load Speed	S ₀	rpm (rad/s)	8251 (864.1)	7015 (734.6)	5592 (585.6)	5993 (627.6)	7666 (802.8)	6151 (644.2)	6348 (664.8)	4916 (514.8)
34	Friction Torque	T _F	oz-in (N·m)	0.4 (2.8 X 10 ⁻³)	0.5 (3.5 X 10 ⁻³)	0.5 (3.5 X 10 ⁻³)	0.6 (4.2 X 10 ⁻³)	0.5 (3.5 X 10 ⁻³)	0.6 (4.2 X 10 ⁻³)	0.65 (4.6 X 10 ⁻³)	0.8 (5.6 X 10 ⁻³)
35	Rotor Inertia	J _M	oz-in·s ² (kg·m ²)	2.2 X 10 ⁻⁴ (1.55 X 10 ⁻⁶)	2.7 X 10 ⁻⁴ (1.91 X 10 ⁻⁶)	3.9 X 10 ⁻⁴ (2.75 X 10 ⁻⁶)	4.6 X 10 ⁻⁴ (3.25 X 10 ⁻⁶)	5.4 X 10 ⁻⁴ (3.81 X 10 ⁻⁶)	5.9 X 10 ⁻⁴ (4.17 X 10 ⁻⁶)	7.9 X 10 ⁻⁴ (5.58 X 10 ⁻⁶)	1.0 X 10 ⁻³ (7.06 X 10 ⁻⁶)
36	Electrical Time Constant	τ _E	ms	0.53	0.63	0.74	0.84	0.80	0.85	0.89	1.06
37	Mechanical Time Constant	τ _M	ms	22.8	14.4	14.7	9.29	18.1	9.25	10.9	8.5
38	Viscous Damping—Infinite Source Impedance	D	oz-in/krpm (N·m/rad/s)	0.0086 (5.79 X 10 ⁻⁷)	0.0272 (1.83 X 10 ⁻⁶)	0.0113 (7.62 X 10 ⁻⁷)	0.0335 (2.25 X 10 ⁻⁶)	0.0125 (8.43 X 10 ⁻⁷)	0.0387 (2.60 X 10 ⁻⁶)	0.0450 (3.03 X 10 ⁻⁶)	0.0525 (3.54 X 10 ⁻⁶)
39	Viscous Damping—Zero Source Impedance	K _D	oz-in/krpm (N·m/rad/s)	1.00 (6.74 X 10 ⁻⁵)	1.94 (1.31 X 10 ⁻⁴)	2.78 (1.87 X 10 ⁻⁴)	5.23 (3.52 X 10 ⁻⁴)	3.11 (2.09 X 10 ⁻⁴)	6.68 (4.50 X 10 ⁻⁴)	7.6 (5.12 X 10 ⁻⁴)	12.5 (8.42 X 10 ⁻⁴)
40	Maximum Winding Temp.	θ _{MAX}	°F (°C)	311 (155)							
41	Thermal Impedance	R _{TH}	°F/watt °C/watt	72.9 (22.7)	72.9 (22.7)	66.4 (19.1)	66.4 (19.1)	62.8 (17.1)	62.8 (17.1)	58.5 (14.7)	56.3 (13.5)
42	Thermal Time Constant	τ _{TH}	min	7.21	7.21	11.1	11.1	12.0	12.0	12.9	13.5
43	Motor Weight	W _M	oz (gm)	6.96 (197.3)	6.98 (197.9)	8.98 (254.6)	8.90 (252.3)	10.1 (286.3)	10.1 (286.3)	0.0 (TBD)	13.8 (391.2)

Model GM9XX2 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	9X12				9X32			
44	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
45	Torque Constant	K _T	oz-in/A (N·m/A)	1.86 (13.2 X 10 ⁻³)	2.95 (20.8 X 10 ⁻³)	3.72 (26.3 X 10 ⁻³)	4.68 (33.1 X 10 ⁻³)	2.20 (15.6 X 10 ⁻³)	3.50 (24.7 X 10 ⁻³)	4.40 (31.1 X 10 ⁻³)	5.53 (39.1 X 10 ⁻³)
46	Back-EMF Constant	K _E	V/krpm (V/rad/s)	1.38 (13.2 X 10 ⁻³)	2.18 (20.8 X 10 ⁻³)	2.75 (26.3 X 10 ⁻³)	3.46 (33.1 X 10 ⁻³)	1.63 (15.6 X 10 ⁻³)	2.59 (24.7 X 10 ⁻³)	3.25 (31.1 X 10 ⁻³)	4.09 (39.1 X 10 ⁻³)
47	Resistance	R _T	Ω	2.63	6.45	10.2	16.1	1.93	4.70	7.38	11.6
48	Inductance	L	mH	1.35	3.40	5.42	8.56	1.16	2.94	4.64	7.34
49	No-Load Current	I _{NL}	A	0.26	0.16	0.13	0.10	0.32	0.20	0.16	0.13
50	Peak Current (Stall)	I _P	A	4.56	2.96	2.35	1.88	6.22	4.06	3.25	2.60

³Continuous torque specified at 25°C ambient temperature and without additional heat sink.

SERIES GM9000

Model GM9XX3 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	9X13				9X33			
51	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
52	Torque Constant	K _T	oz·in/A (N·m/A)	2.80 (19.8 X 10 ⁻³)	4.47 (31.6 X 10 ⁻³)	5.60 (39.5 X 10 ⁻³)	7.07 (49.9 X 10 ⁻³)	2.67 (18.9 X 10 ⁻³)	4.20 (29.7 X 10 ⁻³)	5.28 (37.3 X 10 ⁻³)	6.68 (47.2 X 10 ⁻³)
53	Back-EMF Constant	K _E	V/krpm (V/rad/s)	2.07 (19.8 X 10 ⁻³)	3.31 (31.6 X 10 ⁻³)	4.14 (39.5 X 10 ⁻³)	5.23 (49.9 X 10 ⁻³)	1.98 (18.9 X 10 ⁻³)	3.10 (29.7 X 10 ⁻³)	3.90 (37.3 X 10 ⁻³)	4.94 (47.2 X 10 ⁻³)
54	Resistance	R _T	Ω	2.17	5.32	8.33	13.2	1.08	2.53	3.94	6.21
55	Inductance	L	mH	1.54	3.93	6.17	9.84	0.84	2.08	3.29	5.27
56	No-Load Current	I _{NL}	A	0.20	0.13	0.10	0.08	0.30	0.19	0.15	0.12
57	Peak Current (Stall)	I _P	A	5.54	3.59	2.88	2.30	11.1	7.55	6.09	4.88

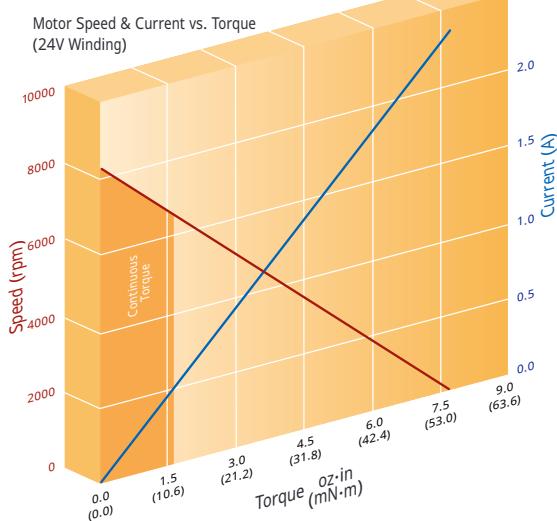
Model GM9XX4 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	9X14				9X34			
58	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
59	Torque Constant	K _T	oz·in/A (oz·in/A)	2.06 (14.5 X 10 ⁻³)	3.27 (23.1 X 10 ⁻³)	4.13 (29.2 X 10 ⁻³)	5.22 (36.9 X 10 ⁻³)	2.58 (18.2 X 10 ⁻³)	4.07 (28.7 X 10 ⁻³)	5.17 (36.5 X 10 ⁻³)	6.50 (45.9 X 10 ⁻³)
60	Back-EMF Constant	K _E	V/krpm (V/krpm)	1.53 (14.5 X 10 ⁻³)	2.42 (23.1 X 10 ⁻³)	3.05 (29.2 X 10 ⁻³)	3.86 (36.9 X 10 ⁻³)	1.91 (18.2 X 10 ⁻³)	3.01 (28.7 X 10 ⁻³)	3.82 (36.5 X 10 ⁻³)	4.81 (45.9 X 10 ⁻³)
61	Resistance	R _T	Ω	1.10	2.59	4.06	6.40	0.83	1.89	2.96	4.62
62	Inductance	L	mH	0.81	2.04	3.25	5.19	0.63	1.56	2.51	3.97
63	No-Load Current	I _{NL}	A	0.29	0.18	0.14	0.11	0.33	0.21	0.16	0.13
64	Peak Current (Stall)	I _P	A	10.9	7.36	5.91	4.73	14.5	10.1	8.11	6.55

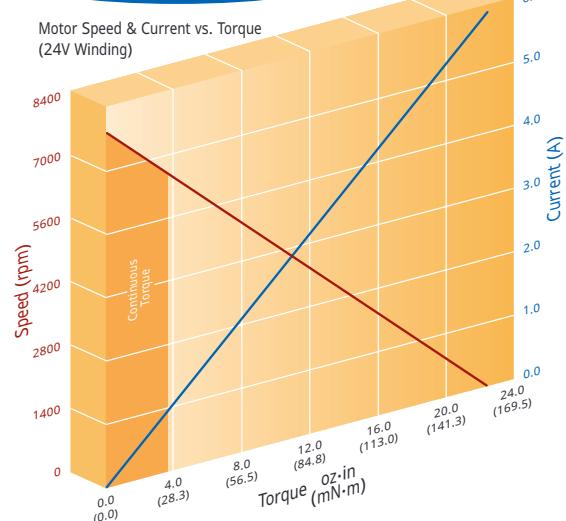
Model GM9X35/9X36 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	9X35				9X36			
65	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1	24.0	30.3
66	Torque Constant	K _T	oz·in/A (oz·in/A)	2.47 (17.4 X 10 ⁻³)	3.99 (28.2 X 10 ⁻³)	4.94 (34.9 X 10 ⁻³)	6.27 (44.3 X 10 ⁻³)	3.25 (23.0 X 10 ⁻³)	5.24 (37.0 X 10 ⁻³)	6.49 (45.8 X 10 ⁻³)	8.24 (58.2 X 10 ⁻³)
67	Back-EMF Constant	K _E	V/krpm (V/krpm)	1.83 (17.4 X 10 ⁻³)	2.95 (28.2 X 10 ⁻³)	3.65 (34.9 X 10 ⁻³)	3.65 (44.3 X 10 ⁻³)	2.4 (23.0 X 10 ⁻³)	3.88 (37.0 X 10 ⁻³)	4.8 (45.8 X 10 ⁻³)	6.09 (58.2 X 10 ⁻³)
68	Resistance	R _T	Ω	.68	1.56	2.37	3.72	0.71	1.64	2.49	3.91
69	Inductance	L	mH	.51	1.34	2.05	3.30	0.66	1.72	2.63	4.24
70	No-Load Current	I _{NL}	A	0.38	0.24	0.19	0.16	0.33	0.20	0.16	0.13
71	Peak Current (Stall)	I _P	A	17.6	12.2	10.1	8.14	16.9	11.7	9.64	7.74

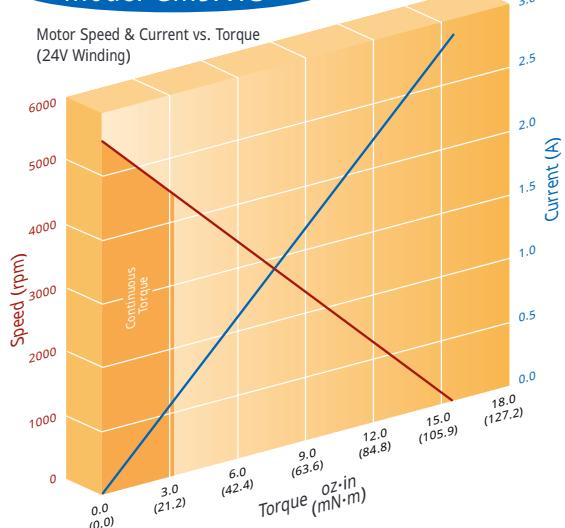
Model GM9X12



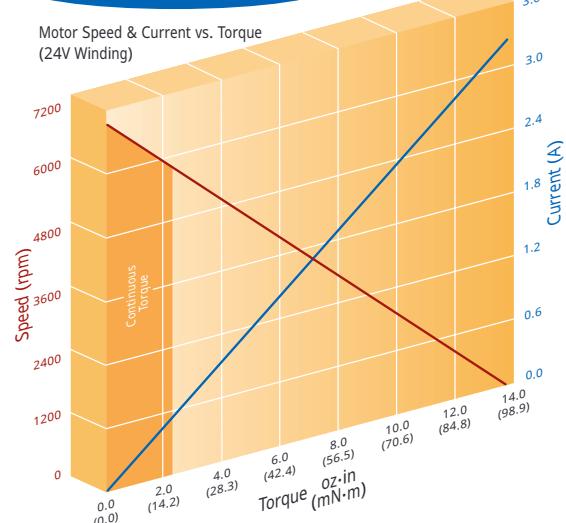
Model GM9X14



Model GM9X13

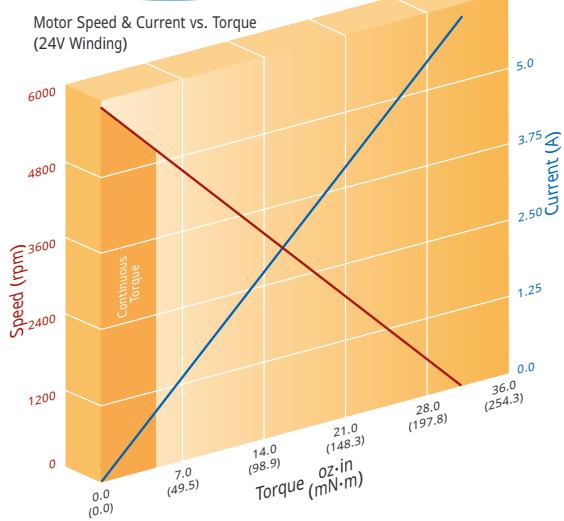


Model GM9X32

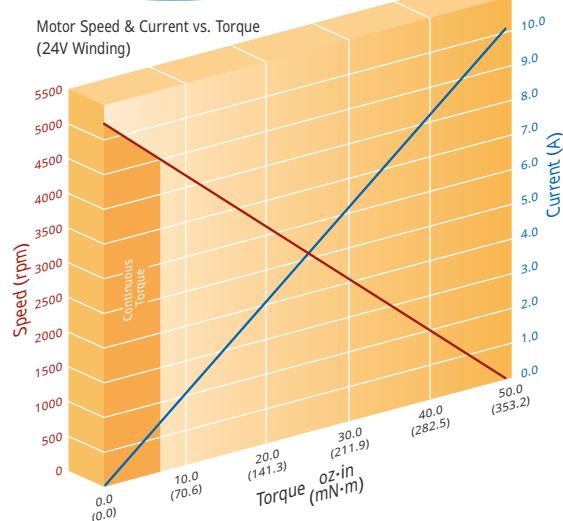


SERIES GM 9000

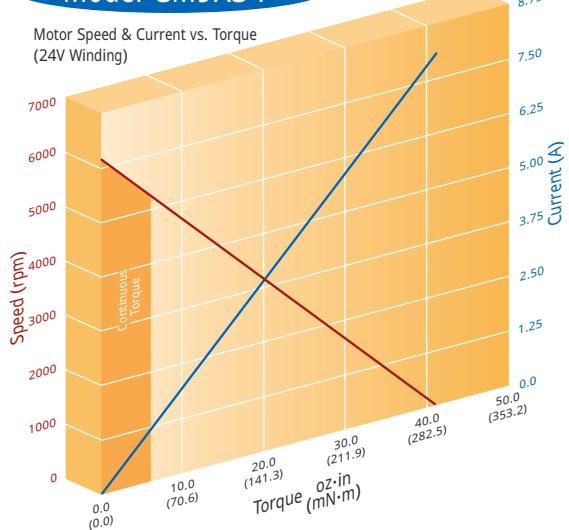
Model GM9X33



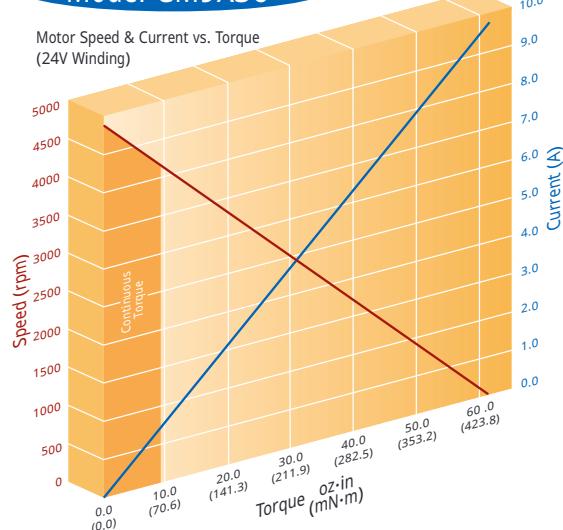
Model GM9X35



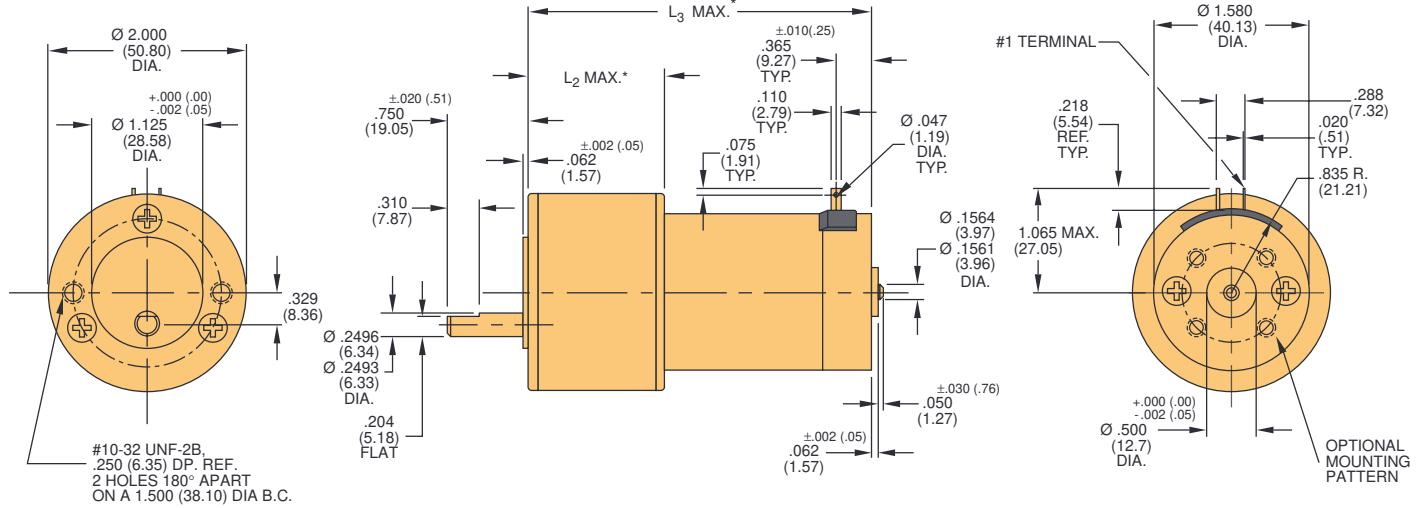
Model GM9X34



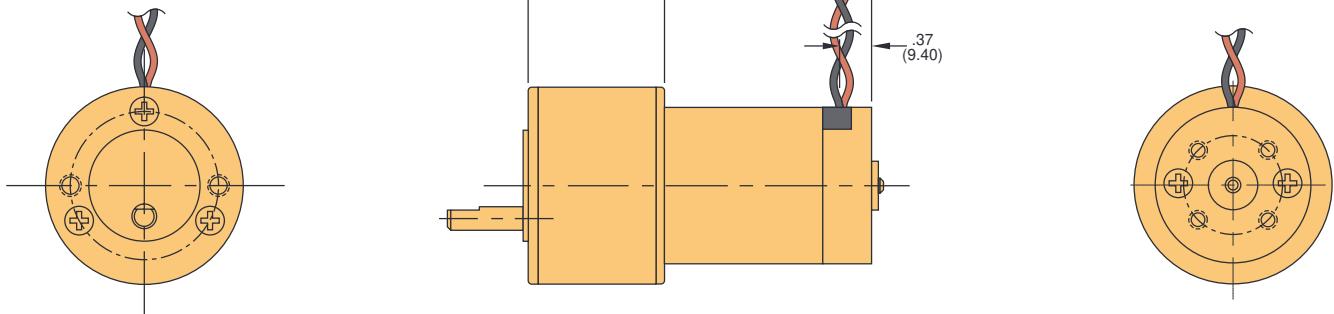
Model GM9X36



GM94XX Motor



GM92XX Motor



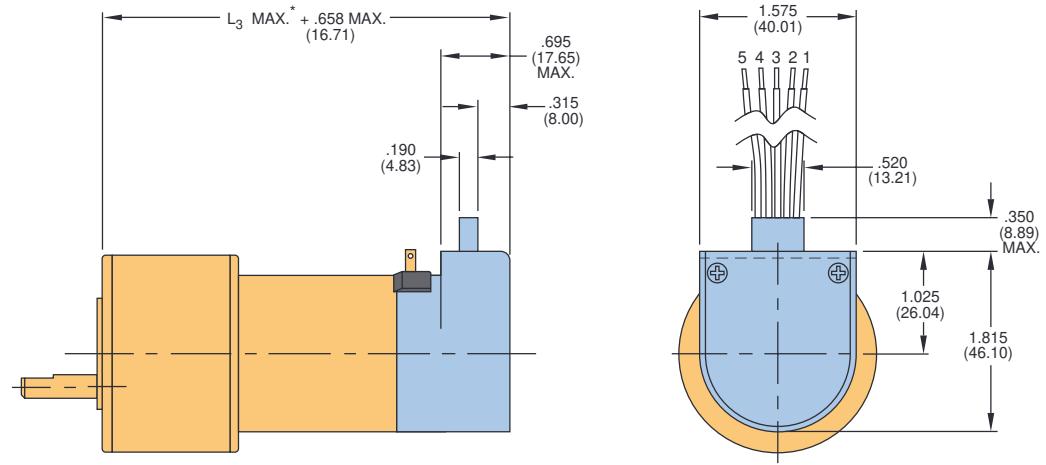
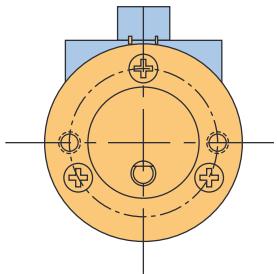
Notes:

- Unless otherwise specified, all tolerances are to be $\pm .005$ (.01)

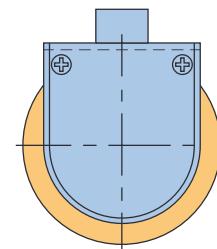
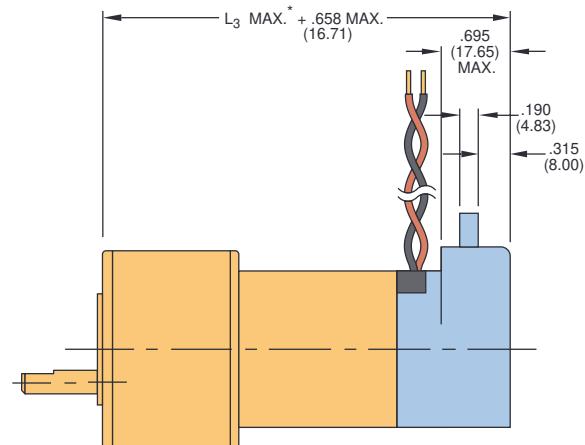
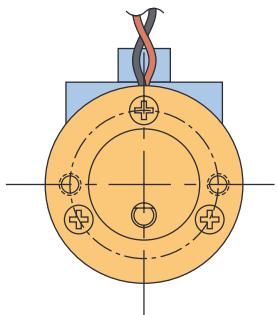
- All measurements are in inches (mm)

SERIES GM9000

**GM94XX Motor
with 91XX Encoder**



**GM92XX Motor
with 91XX Encoder**



Encoder Connection Chart

Pin No.	Color	Connection
1	Black	Ground
2	Green	Index/NC
3	Yellow	Channel A
4	Red	Vcc
5	Blue	Channel B

Notes:

- Unless otherwise specified, all tolerances are to be $\pm .005$ (.01)

- All measurements are in inches (mm)

*See line numbers 6 through 11 and 17 through 21

PITTMAN®

PITTMAN®

SERIES GM14900

Gearmotor Data

Line No.	Parameter	Symbol	Reduction Ratios			
			Units	5.9:1	19.7:1	65.5:1*
MECHANICAL SPECIFICATIONS (Standard and High-Torque Gears)						
1	Max. Load Standard Gears ¹	T _L	oz-in (N·m)	175 (1.24)	175 (1.24)	175 (1.24)
2	Max. Load High-Torque Gears ¹	T _L	oz-in (N·m)	N/A N/A	300 (2.12)	300 (2.12)
3	Gearbox Shaft Rotation ²	—	—	CW	CCW	CW
4	Gearbox Efficiency	—	%	81	73	66
5	Gearbox Weight	W _G	oz (gm)	5.90 (167.3)	6.26 (177.5)	6.62 (187.7)
6	Gearbox Length	L ₂	in max (mm max)	1.373 (34.87)	1.373 (34.87)	1.373 (34.87)
7	Length, GM14901	L ₃	in max (mm max)	4.322 (109.78)	4.322 (109.78)	4.322 (109.78)
8	Length, GM14902	L ₃	in max (mm max)	4.572 (116.13)	4.572 (116.13)	4.572 (116.13)
MECHANICAL SPECIFICATIONS (High-Torque, Wide-Face Gears)						
9	Max. Load ¹	T _L	oz-in (N·m)	N/A N/A	500 (3.53)	500 (3.53)
10	Gearbox Shaft Rotation	—	—	CW	CCW	CW
11	Gearbox Efficiency	—	%	81	73	66
12	Gearbox Weight	W _G	oz (gm)	N/A N/A	6.52 (184.8)	6.88 (195.0)
13	Gearbox Length	L ₂	in max (mm max)	N/A N/A	1.373 (34.87)	1.373 (34.87)
14	Length, GM14901	L ₃	in max (mm max)	N/A N/A	4.322 (109.78)	4.322 (109.78)
15	Length, GM14902	L ₃	in max (mm max)	N/A N/A	4.572 (116.13)	4.572 (116.13)
NO-LOAD SPEED (All Gears)						
16	GM14901	S _{NL}	rpm (rad/s)	713 (75)	214 (22)	64.2 (7)
17	GM14902	S _{NL}	rpm (rad/s)	690 (72)	207 (22)	62.1 (6)

¹Represents gearbox capability only. Continuous load torque capability will vary with gear ratio, motor selection, and operating conditions.

²Shaft rotation is designated while looking at output shaft with motor operating in a clockwise direction.

*Contact factory for higher ratios.

Motor Data

Line No.	Parameter	Symbol	Units	14X01	14X02
18	Continuous Torque ³	T _C	oz·in (N·m)	10.0 (70.6 X 10 ⁻³)	14.0 (98.9 X 10 ⁻³)
19	Peak Torque (Stall)	T _{PK}	oz·in (N·m)	62.8 (44)	107 (76)
20	Motor Constant	K _M	oz·in/√W (N·m/√W)	4.45 (31.4 X 10 ⁻³)	5.93 (41.9 X 10 ⁻³)
21	No-Load Speed	S ₀	rpm (rad/s)	4230 (443)	4087 (428)
22	Friction Torque	T _F	oz·in (N·m)	1.20 (8.5 X 10 ⁻³)	1.20 (8.5 X 10 ⁻³)
23	Rotor Inertia	J _M	oz·in·s ² (kg·m ²)	1.6 X 10 ⁻³ (1.13 X 10 ⁻⁵)	2.3 X 10 ⁻³ (1.62 X 10 ⁻⁵)
24	Electrical Time Constant	τ _E	ms	0.91	1.47
25	Mechanical Time Constant	τ _M	ms	11.4	9.26
26	Viscous Damping— Infinite Source Impedance	D	oz·in/krpm (N·m/(rad/s))	0.17 (1.15 X 10 ⁻⁵)	0.17 (1.15 X 10 ⁻⁵)
27	Viscous Damping— Zero Source Impedance	K _D	oz·in/krpm (N·m/(rad/s))	14.7 (9.91 X 10 ⁻⁴)	26.0 (1.75 X 10 ⁻³)
28	Maximum Winding Temperature	θ _{MAX}	°F (°C)	311 (155)	311 (155)
29	Thermal Impedance	R _{TH}	°F/watt °C/watt	49.8 (9.90)	48.2 (9.00)
30	Thermal Time Constant	τ _{TH}	min	22.0	24.0
31	Motor Weight	W _M	oz (gm)	20.8 (589.7)	26.0 (737.1)
32	Motor Length, 1410X, 1420X	L ₁	in max (mm max)	2.953 (75.01)	3.203 (81.36)

³Continuous torque specified at 25°C ambient temperature and without additional heat sink.

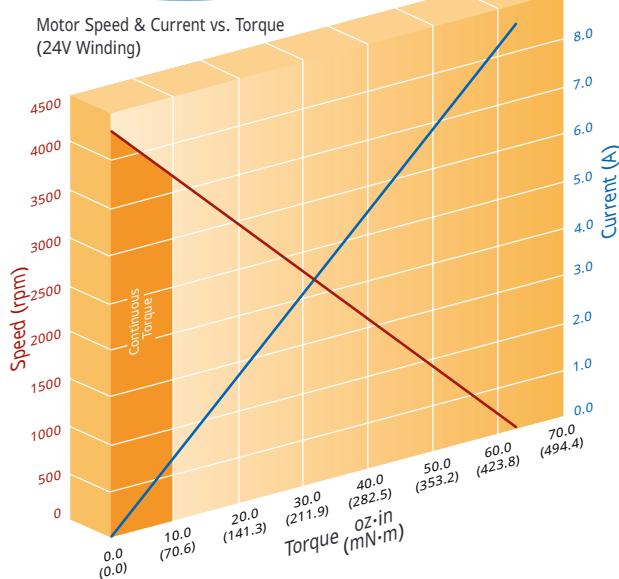
Model GM14X01/GM14X02 Winding Data (Other windings available upon request)

Line No.	Parameter	Symbol	Units	GM14X01			GM14X02		
				12.0	19.1	24.0	30.3	12.0	19.1
33	Reference Voltage	E	V	12.0	19.1	24.0	30.3	12.0	19.1
34	Torque Constant	K _T	oz·in/A (N·m/A)	3.72 (26.3 X 10 ⁻³)	5.89 (41.6 X 10 ⁻³)	7.44 (52.5 X 10 ⁻³)	9.46 (66.8 X 10 ⁻³)	3.90 (27.5 X 10 ⁻³)	6.16 (43.5 X 10 ⁻³)
35	Back-EMF Constant	K _E	V/krpm (V/rad/s)	2.75 (26.3 X 10 ⁻³)	4.36 (41.6 X 10 ⁻³)	5.50 (52.5 X 10 ⁻³)	6.99 (66.8 X 10 ⁻³)	2.88 (27.5 X 10 ⁻³)	4.55 (43.5 X 10 ⁻³)
36	Resistance	R _T	Ω	0.72	1.76	2.79	4.45	0.45	1.09
37	Inductance	L	mH	0.63	1.59	2.54	4.10	0.63	1.58
38	No-Load Current	I _{NL}	A	0.52	0.33	0.26	0.20	0.49	0.31
39	Peak Current (Stall) ⁴	I _P	A	16.7	10.8	8.60	6.80	26.4	17.5
								13.9	11.1

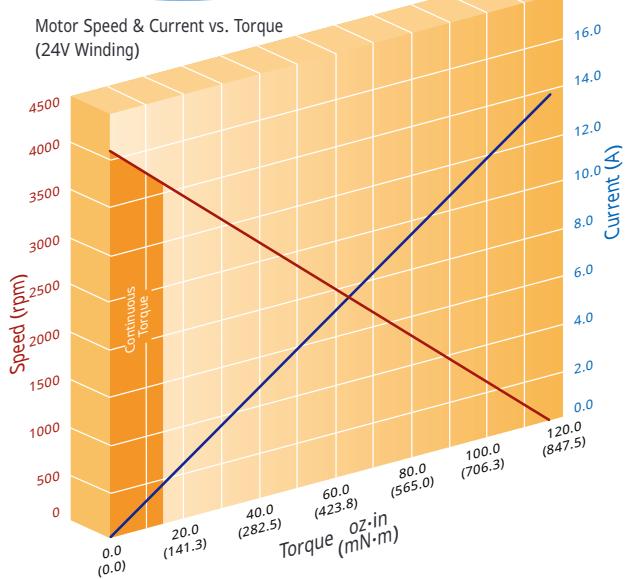
⁴Theoretical values supplied for reference only.

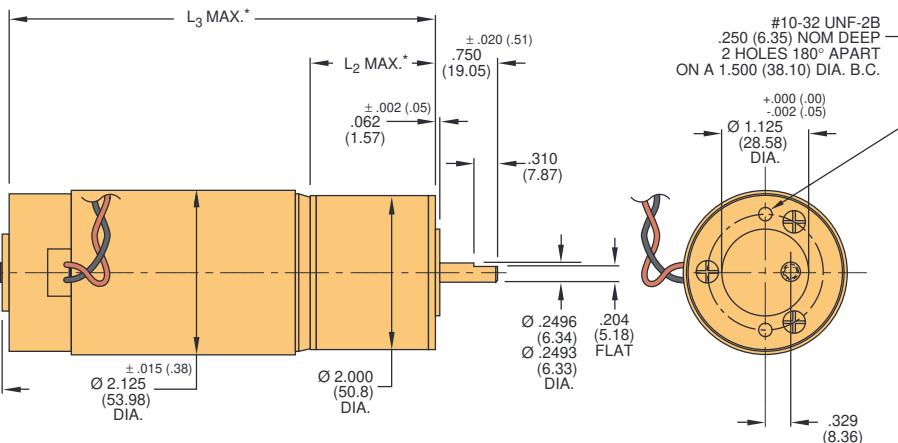
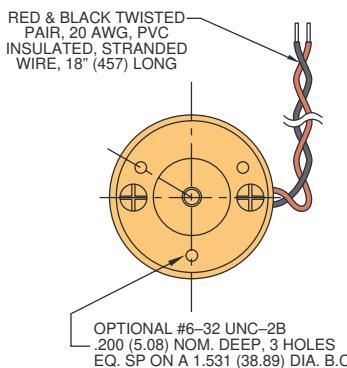
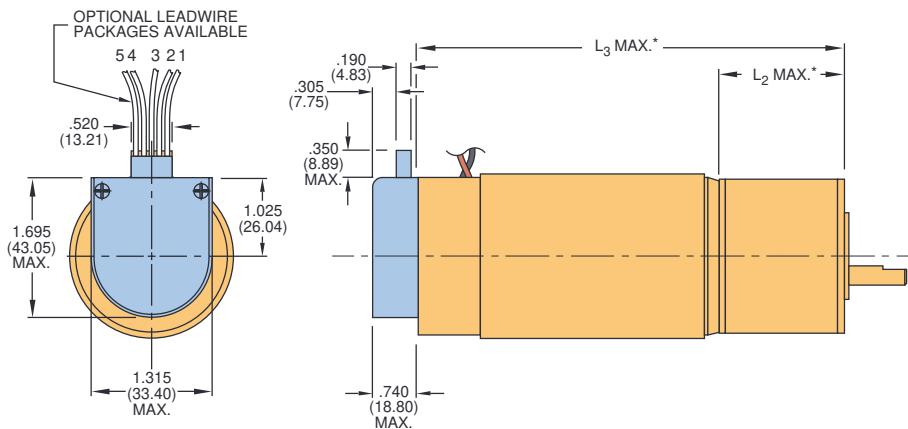
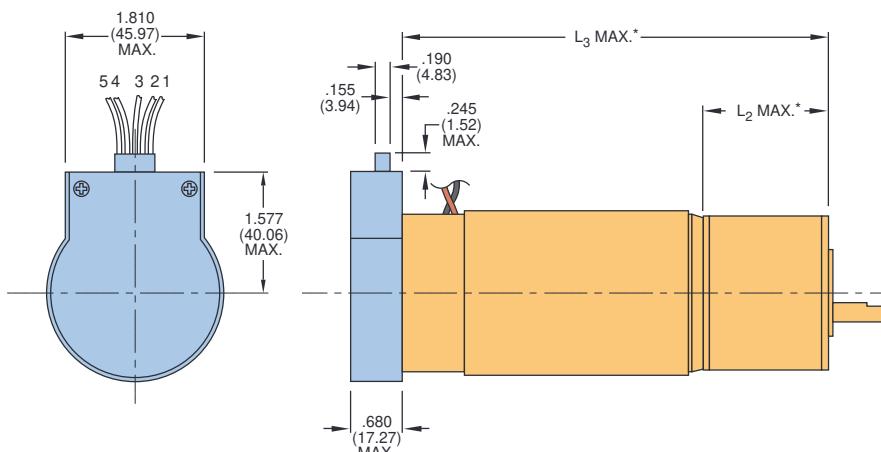
SERIES GM14900

Model GM14X01



Model GM14X02



GM149XX Motor**GM149XX Motor
with 91X0 Encoder****GM149XX Motor
with 90X0 Encoder****Encoder Connection Chart**

Pin No.	Color	Connection
1	Black	Ground
2	Green	Index/NC
3	Yellow	Channel A
4	Red	Vcc
5	Blue	Channel B

Notes:

- Unless otherwise specified, all tolerances are to be $\pm .005 (.01)$
- All measurements are in inches (mm)

*See line no. 6 through 8 and 13 through 15 in gearmotor data chart

PITTMAN®

343 GODSHALL DRIVE
HARLEYSVILLE, PENNSYLVANIA 19438 U.S.A.
PHONE: 215-256-6601 • FAX: 215-256-1338
TOLL FREE: 877-PITTMAN (USA Only)
E-MAIL: info@pittmannet.com
WEB SITE: www.pittmannet.com

Specifications subject to change without notice.



LCG-20

5/00

March 1997

CMOS Programmable Interval Timer

Features

- 8MHz to 12MHz Clock Input Frequency
- Compatible with NMOS 8254
- Enhanced Version of NMOS 8253
- Three Independent 16-Bit Counters
- Six Programmable Counter Modes
- Status Read Back Command
- Binary or BCD Counting
- Fully TTL Compatible
- Single 5V Power Supply
- Low Power
 - ICCSB 10 μ A
 - ICCOP 10mA at 8MHz
- Operating Temperature Ranges
 - C82C54 0°C to +70°C
 - I82C54 -40°C to +85°C
 - M82C54 -55°C to +125°C

Description

The Harris 82C54 is a high performance CMOS Programmable Interval Timer manufactured using an advanced 2 micron CMOS process.

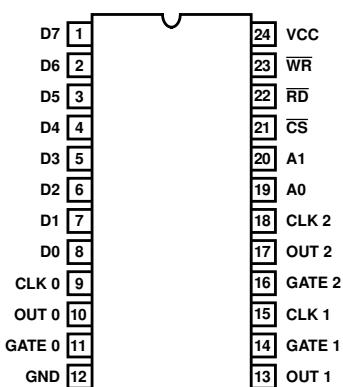
The 82C54 has three independently programmable and functional 16-bit counters, each capable of handling clock input frequencies of up to 8MHz (82C54) or 10MHz (82C54-10) or 12MHz (82C54-12).

The high speed and industry standard configuration of the 82C54 make it compatible with the Harris 80C86, 80C88, and 80C286 CMOS microprocessors along with many other industry standard processors. Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and many other applications. Static CMOS circuit design insures low power operation.

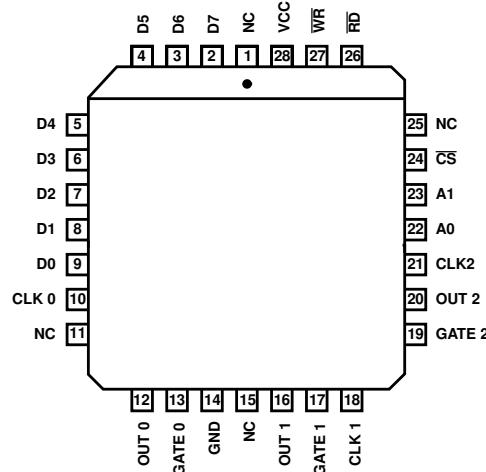
The Harris advanced CMOS process results in a significant reduction in power with performance equal to or greater than existing equivalent products.

Pinouts

82C54 (PDIP, CERDIP, SOIC)
TOP VIEW



82C54 (PLCC/CLCC)
TOP VIEW

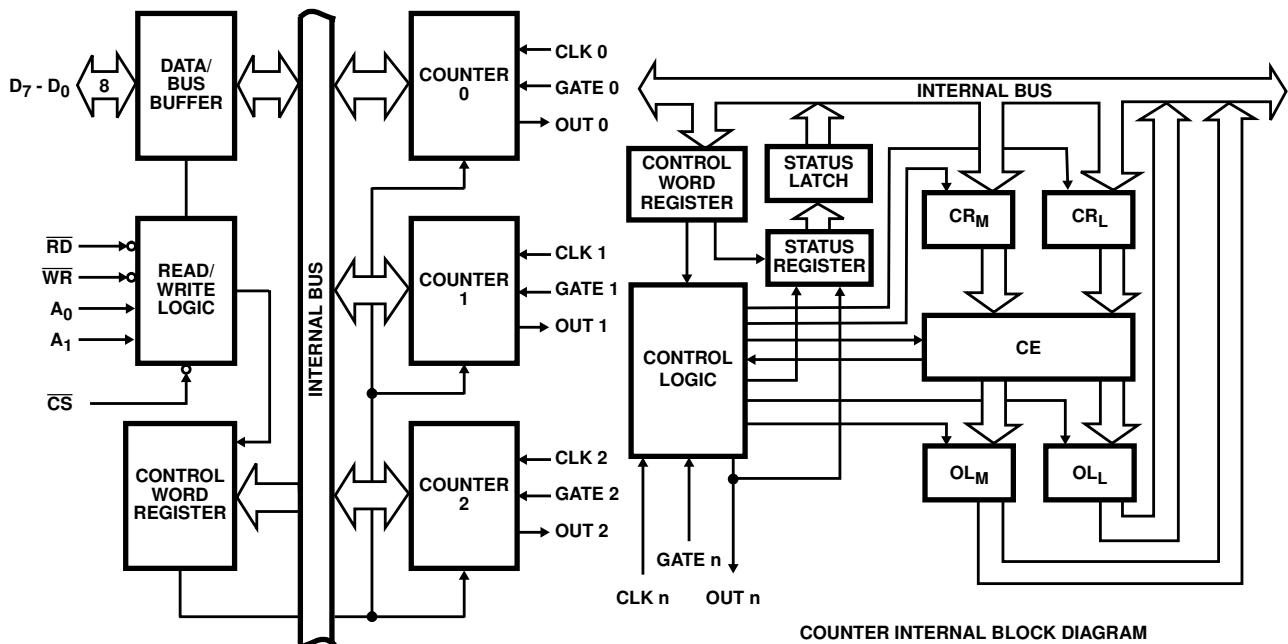


82C54

Ordering Information

PART NUMBERS			TEMPERATURE RANGE	PACKAGE	PKG. NO.
8MHz	10MHz	12MHz			
CP82C54	CP82C54-10	CP82C54-12	0°C to +70°C	24 Lead PDIP	E24.6
IP82C54	IP82C54-10	IP82C54-12	-40°C to +85°C	24 Lead PDIP	E24.6
CS82C54	CS82C54-10	CS82C54-12	0°C to +70°C	28 Lead PLCC	N28.45
IS82C54	IS82C54-10	IS82C54-12	-40°C to +85°C	28 Lead PLCC	N28.45
CD82C54	CD82C54-10	CD82C54-12	0°C to +70°C	24 Lead CERDIP	F24.6
ID82C54	ID82C54-10	ID82C54-12	-40°C to +85°C	24 Lead CERDIP	F24.6
MD82C54/B	MD82C54-10/B	MD82C54-12/B	-55°C to +125°C	24 Lead CERDIP	F24.6
MR82C54/B	MR82C54-10/B	MR82C54-12/B	-55°C to +125°C	28 Lead CLCC	J28.A
SMD # 8406501JA	-	8406502JA	-55°C to +125°C	24 Lead CERDIP	F24.6
SMD# 84065013A	-	84065023A	-55°C to +125°C	28 Lead CLCC	J28.A
CM82C54	CM82C54-10	CM82C54-12	0°C to +70°C	24 Lead SOIC	M24.3

Functional Diagram



Pin Description

SYMBOL	DIP PIN NUMBER	TYPE	DEFINITION
D7 - D0	1 - 8	I/O	DATA: Bi-directional three-state data bus lines, connected to system data bus.
CLK 0	9	I	CLOCK 0: Clock input of Counter 0.
OUT 0	10	O	OUT 0: Output of Counter 0.
GATE 0	11	I	GATE 0: Gate input of Counter 0.
GND	12		GROUND: Power supply connection.
OUT 1	13	O	OUT 1: Output of Counter 1.
GATE 1	14	I	GATE 1: Gate input of Counter 1.
CLK 1	15	I	CLOCK 1: Clock input of Counter 1.
GATE 2	16	I	GATE 2: Gate input of Counter 2.
OUT 2	17	O	OUT 2: Output of Counter 2.

Pin Description (Continued)

SYMBOL	DIP PIN NUMBER	TYPE	DEFINITION															
CLK 2	18	I	CLOCK 2: Clock input of Counter 2.															
A0, A1	19 - 20	I	ADDRESS: Select inputs for one of the three counters or Control Word Register for read/write operations. Normally connected to the system address bus.															
			<table border="1"> <thead> <tr> <th>A1</th> <th>A0</th> <th>SELECTS</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Counter 0</td> </tr> <tr> <td>0</td> <td>1</td> <td>Counter 1</td> </tr> <tr> <td>1</td> <td>0</td> <td>Counter 2</td> </tr> <tr> <td>1</td> <td>1</td> <td>Control Word Register</td> </tr> </tbody> </table>	A1	A0	SELECTS	0	0	Counter 0	0	1	Counter 1	1	0	Counter 2	1	1	Control Word Register
A1	A0	SELECTS																
0	0	Counter 0																
0	1	Counter 1																
1	0	Counter 2																
1	1	Control Word Register																
CS	21	I	CHIP SELECT: A low on this input enables the 82C54 to respond to RD and WR signals. RD and WR are ignored otherwise.															
RD	22	I	READ: This input is low during CPU read operations.															
WR	23	I	WRITE: This input is low during CPU write operations.															
VCC	24		VCC: The +5V power supply pin. A 0.1 μ F capacitor between pins VCC and GND is recommended for decoupling.															

Functional Description**General**

The 82C54 is a programmable interval timer/counter designed for use with microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other computer/timer functions common to microcomputers which can be implemented with the 82C54 are:

- Real time clock
- Event counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Data Bus Buffer

This three-state, bi-directional, 8-bit buffer is used to interface the 82C54 to the system bus (see Figure 1).

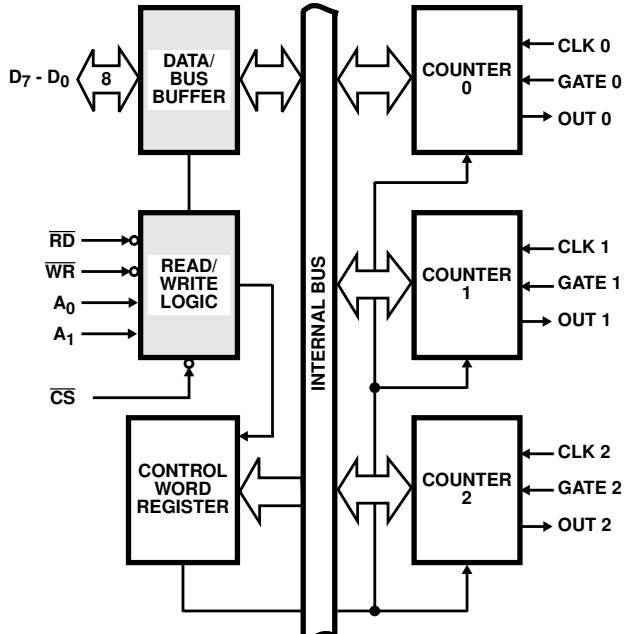


FIGURE 1. DATA BUS BUFFER AND READ/WRITE LOGIC FUNCTIONS

Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 82C54. A1 and A0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the RD input tells the 82C54 that the CPU is reading one of the counters. A "low" on the WR input tells the 82C54 that the CPU is writing either a Control Word or an initial count. Both RD and WR are qualified by CS; RD and WR are ignored unless the 82C54 has been selected by holding CS low.

Control Word Register

The Control Word Register (Figure 2) is selected by the Read/Write Logic when A1, A0 = 11. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the Counter operation.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

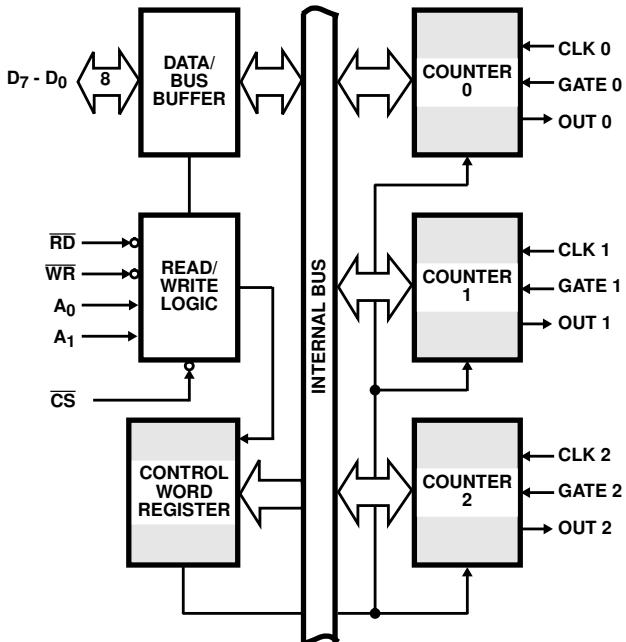


FIGURE 2. CONTROL WORD REGISTER AND COUNTER FUNCTIONS

Counter 0, Counter 1, Counter 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a signal counter is shown in Figure 3. The counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

The status register, shown in the figure, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labeled CE (for Counting Element). It is a 16-bit presetable synchronous down counter.

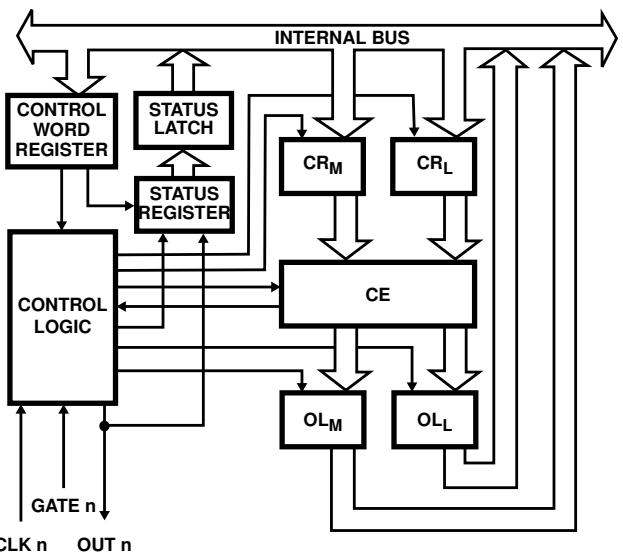


FIGURE 3. COUNTER INTERNAL BLOCK DIAGRAM

OLM and OLL are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L for "Most significant byte" and "Least significant byte", respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 82C54, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CRM and CRL (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CRM and CRL are cleared when the Counter is programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK n, GATE n, and OUT n are all connected to the outside world through the Control Logic.

82C54 System Interface

The 82C54 is treated by the system software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method or it can be connected to the output of a decoder.

Operational Description

General

After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 82C54

Counters are programmed by writing a Control Word and then an initial count.

All Control Words are written into the Control Word Register, which is selected when A1, A0 = 11. The Control Word specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A1, A0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

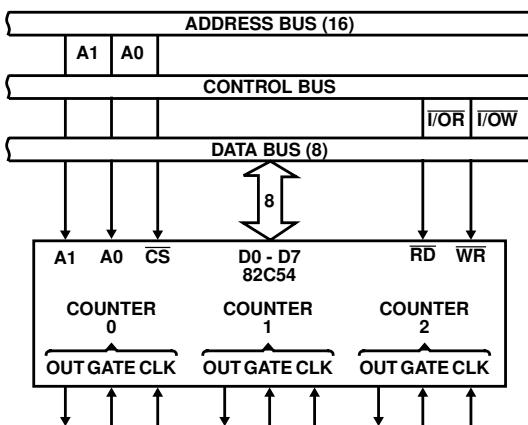


FIGURE 4. 82C54 SYSTEM INTERFACE

Write Operations

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:

1. For Each Counter, the Control Word must be written before the initial count is written.
2. The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A1, A0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

Control Word Format

A1, A0 = 11; $\overline{CS} = 0$; $\overline{RD} = 1$; $\overline{WR} = 0$

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC - Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

RW - Read/Write

RW1	RW0	
0	0	Counter Latch Command (See Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

M - Mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD - Binary Coded Decimal

0	Binary Counter 16-bit
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't Care bits (X) should be 0 to insure compatibility with future products.

Possible Programming Sequence

	A1	A0
Control Word - Counter 0	1	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0
Control Word - Counter 1	1	1
LSB of Count - Counter 1	0	1
MSB of Count - Counter 1	0	1
Control Word - Counter 2	1	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 2	1	0

Possible Programming Sequence

	A1	A0
Control Word - Counter 0	1	1
Control Word - Counter 1	1	1
Control Word - Counter 2	1	1
LSB of Count - Counter 2	1	0

Possible Programming Sequence (Continued)

	A1	A0
LSB of Count - Counter 1	0	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0
MSB of Count - Counter 1	0	1
MSB of Count - Counter 2	1	0

Possible Programming Sequence

	A1	A0
Control Word - Counter 2	1	1
Control Word - Counter 1	1	1
Control Word - Counter 0	1	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 2	1	0
LSB of Count - Counter 1	0	1
MSB of Count - Counter 1	0	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0

Possible Programming Sequence

	A1	A0
Control Word - Counter 1	1	1
Control Word - Counter 0	1	1
LSB of Count - Counter 1	0	1
Control Word - Counter 2	1	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 1	0	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 0	0	0
MSB of Count - Counter 2	1	0

NOTE: In all four examples, all counters are programmed to Read/Write two-byte counts. These are only four of many programming sequences.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies. A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 82C54.

There are three possible methods for reading the Counters. The first is through the Read-Back command, which is

explained later. The second is a simple read operation of the Counter, which is selected with the A1, A0 inputs. The only requirement is that the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in process of changing when it is read, giving an undefined result.

Counter Latch Command

The other method for reading the Counters involves a special software command called the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when A1, A0 = 11. Also, like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

A1, A0 = 11; CS = 0; RD = 1; WR = 0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	COUNTER
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5, D4 - 00 designates Counter Latch Command, X - Don't Care. NOTE: Don't Care bits (X) should be 0 to insure compatibility with future products.

The selected Counter's output latch (OL) latches the count when the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 82C54 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a counter is programmed to read or write two-byte counts, the following precaution applies: A program MUST NOT transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

Read-Back Command

The read-back command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 5. The command applies to the counters selected by setting their corresponding bits D3, D2, D1 = 1.

A0, A1 = 11; CS = 0; RD = 1; WR = 0

D7	D6	D5	D4	D3	D2	D1	D0
1	1	COUNT	STATUS	CNT 2	CNT 1	CNT 0	0

D5: 0 = Latch count of selected Counter (s)

D4: 0 = Latch status of selected Counter(s)

D3: 1 = Select Counter 2

D2: 1 = Select Counter 1

D1: 1 = Select Counter 0

D0: Reserved for future expansion; Must be 0

FIGURE 5. READ-BACK COMMAND FORMAT

The read-back command may be used to latch multiple counter output latches (OL) by setting the COUNT bit D5 = 0 and selecting the desired counter(s). This signal command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting STATUS bit D4 = 0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 6. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

D7	D6	D5	D4	D3	D2	D1	D0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D7: 1 = Out pin is 1

0 = Out pin is 0

D6: 1 = Null count

0 = Count available for reading

D5 - D0 = Counter programmed mode (See Control Word Formats)

FIGURE 6. STATUS BYTE

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the counter is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown below.

THIS ACTION:

CAUSES:

- A. Write to the control word register:(1) Null Count = 1
- B. Write to the count register (CR):(2) Null Count = 1
- C. New count is loaded into CE (CR - CE)..... Null Count = 0
- (1) Only the counter specified by the control word will have its null count set to 1. Null count bits of other counters are unaffected.
- (2) If the counter is programmed for two-byte counts (least significant byte then most significant byte) null count goes to 1 when the second byte is written.

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

COMMANDS								DESCRIPTION	RESULT
D7	D6	D5	D4	D3	D2	D1	D0		
1	1	0	0	0	0	1	0	Read-Back Count and Status of Counter 0	Count and Status Latched for Counter 0
1	1	1	0	0	1	0	0	Read-Back Status of Counter 1	Status Latched for Counter 1
1	1	1	0	1	1	0	0	Read-Back Status of Counters 2, 1	Status Latched for Counter 2, But Not Counter 1
1	1	0	1	1	0	0	0	Read-Back Count of Counter 2	Count Latched for Counter 2
1	1	0	0	0	1	0	0	Read-Back Count and Status of Counter 1	Count Latched for Counter 1, But Not Status
1	1	1	0	0	0	1	0	Read-Back Status of Counter 1	Command Ignored, Status Already Latched for Counter 1

FIGURE 7. READ-BACK COMMAND EXAMPLE

Both count and status of the selected counter(s) may be latched simultaneously by setting both COUNT and STATUS bits D5, D4 = 0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 7.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

CS	RD	WR	A1	A0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (Three-State)
1	X	X	X	X	No-Operation (Three-State)
0	1	1	X	X	No-Operation (Three-State)

FIGURE 8. READ/WRITE OPERATIONS SUMMARY

Mode Definitions

The following are defined for use in describing the operation of the 82C54.

CLK PULSE:

A rising edge, then a falling edge, in that order, of a Counter's CLK input.

TRIGGER:

A rising edge of a Counter's Gate input.

COUNTER LOADING:

The transfer of a count from the CR to the CE (See "Functional Description")

Mode 0: Interrupt on Terminal Count

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written to the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- (1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required).
- (2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again OUT does not go high until N + 1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the counter as this has already been done.

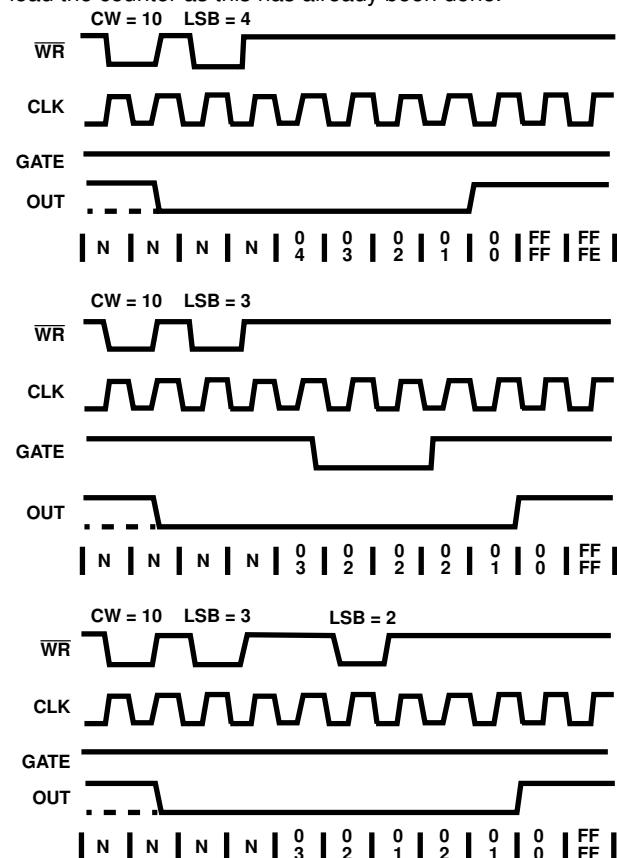


FIGURE 9. MODE 0

NOTES: The following conventions apply to all mode timing diagrams.

1. Counters are programmed for binary (not BCD) counting and for reading/writing least significant byte (LSB) only.
2. The counter is always selected (\bar{CS} always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, Hex is written to the counter.
4. LSB stands for Least significant "byte" of count.
5. Numbers below diagrams are count values. The lower number is the least significant byte. The upper number is the most significant byte. Since the counter is programmed to read/write LSB only, the most significant byte cannot be read.
6. N stands for an undefined count.
7. Vertical lines show transitions between count values.

Mode 1: Hardware Retriggerable One-Shot

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggerable. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

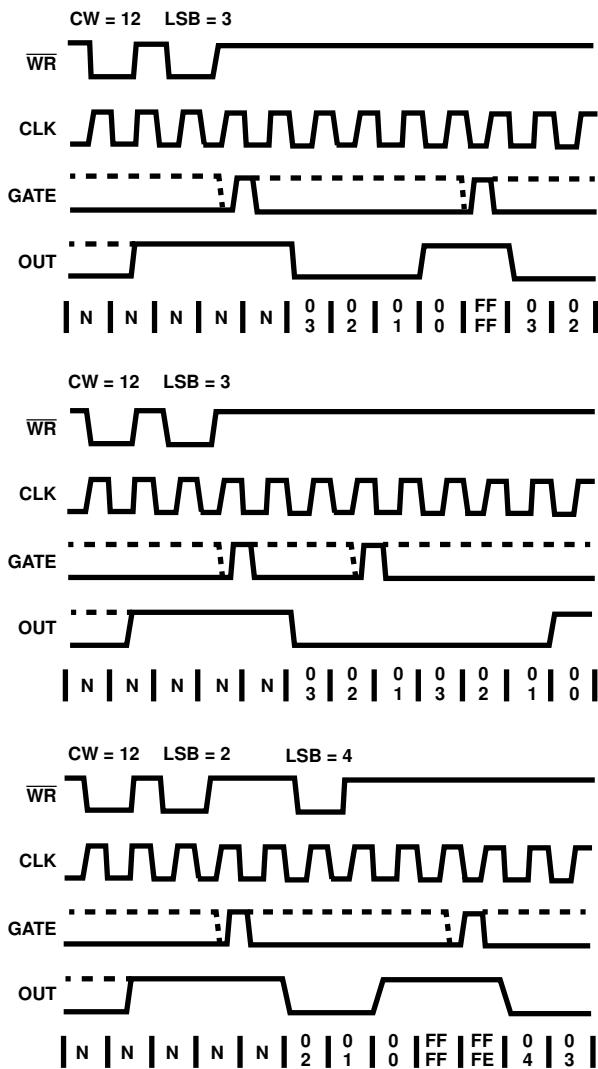


FIGURE 10. MODE 1

Mode 2: Rate Generator

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock Interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK pulses after the initial count is written. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the end of the current counting cycle.

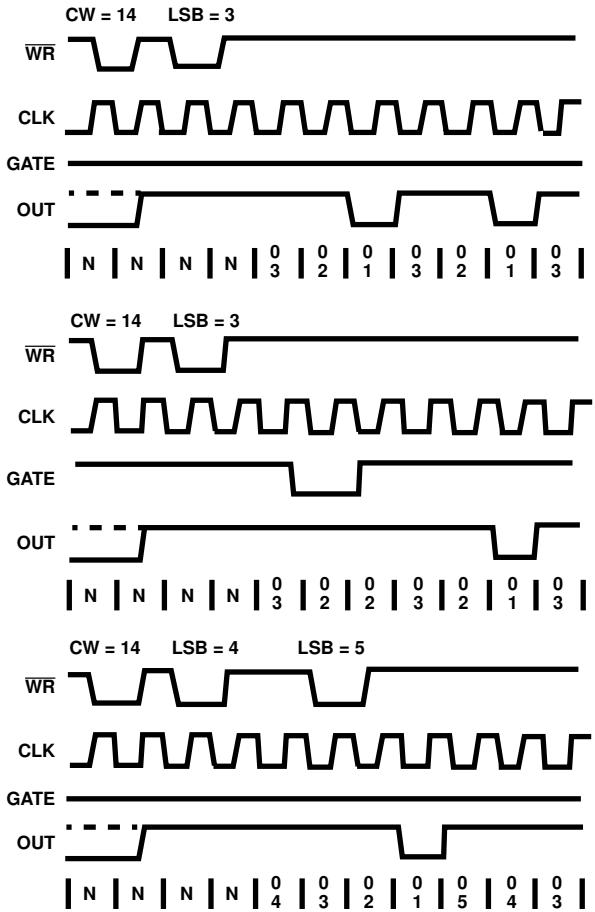


FIGURE 11. MODE 2

Mode 3: Square Wave Mode

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

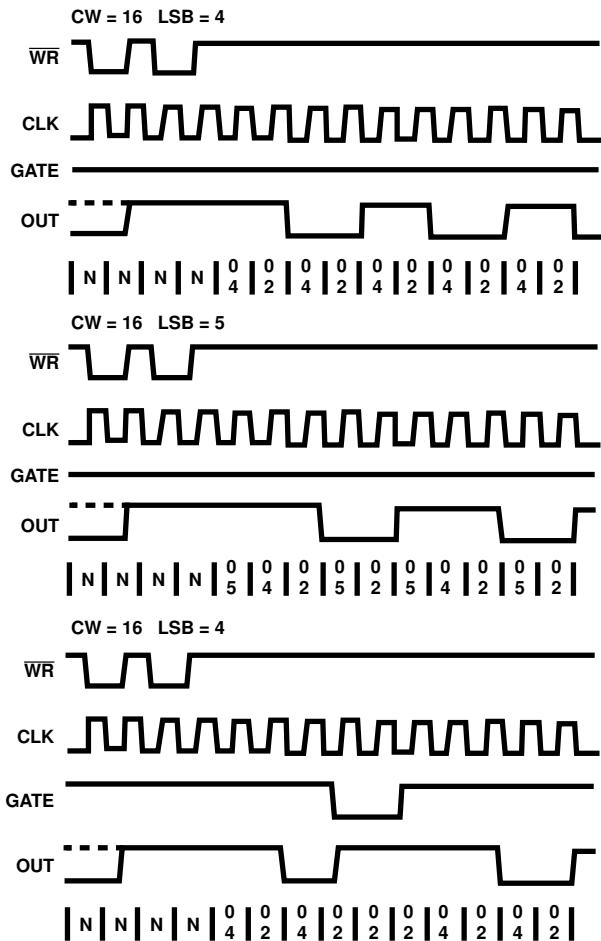


FIGURE 12. MODE 3

Mode 3 is Implemented as Follows:

EVEN COUNTS: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires, OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

ODD COUNTS: OUT is initially high. The initial count is loaded on one CLK pulse, decremented by one on the next CLK pulse, and then decremented by two on succeeding CLK pulses. When the count expires, OUT goes low and the Counter is reloaded with the initial count. The count is decremented by three on the next CLK pulse, and then by two on succeeding CLK pulses. When the count expires, OUT goes high again and the Counter is reloaded with the initial count. The above process is repeated indefinitely. So for odd counts, OUT will be high for $(N + 1)/2$ counts and low for $(N - 1)/2$ counts.

Mode 4: Software Triggered Mode

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse then go high again. The counting sequence is "Triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting.
GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until $N + 1$ CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- (1) Writing the first byte has no effect on counting.
 - (2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be “retriggered” by software. OUT strobes low N + 1 CLK pulses after the new count of N is written.

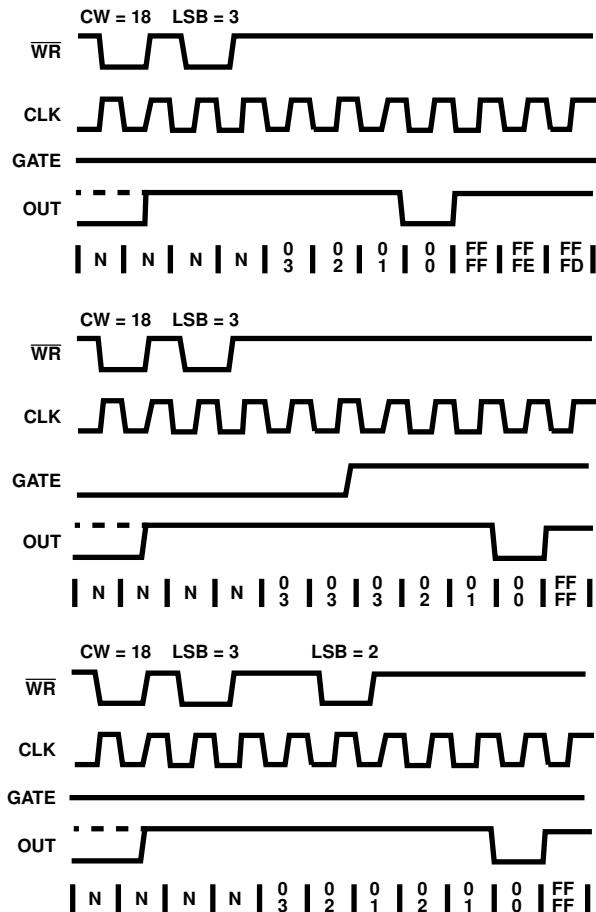


FIGURE 13. MODE 4

Mode 5: Hardware Triggered Strobe (Retriggerable)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is triggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with new count on the next CLK pulse and counting will continue from there.

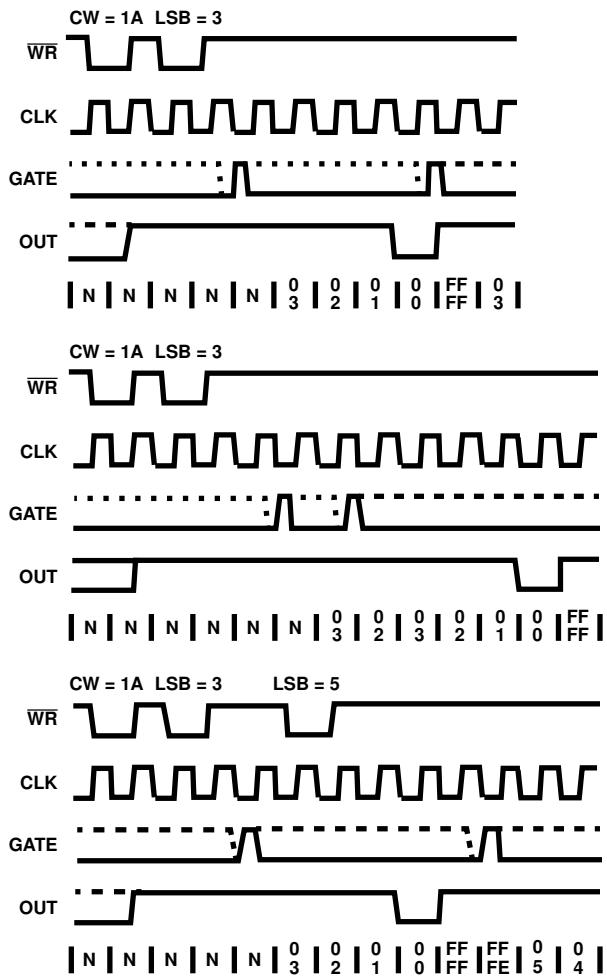


FIGURE 14. MODE 5

Operation Common to All Modes**Programming**

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

Gate

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3 and 4 the GATE input is level sensitive, and logic level is sampled on the rising edge of CLK. In modes 1, 2, 3 and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of Gate (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK. The flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs - a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive.

Counter

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

The counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter “wraps around” to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

SIGNAL STATUS MODES	LOW OR GOING LOW	RISING	HIGH
0	Disables Counting	-	Enables Counting
1	-	1) Initiates Counting 2) Resets output after next clock	-
2	1) Disables counting 2) Sets output immediately high	Initiates Counting	Enables Counting
3	1) Disables counting 2) Sets output immediately high	Initiates Counting	Enables Counting
4	1) Disables Counting	-	Enables Counting
5	-	Initiates Counting	-

FIGURE 15. GATE PIN OPERATIONS SUMMARY

MODE	MIN COUNT	MAX COUNT
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE: 0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

FIGURE 16. MINIMUM AND MAXIMUM INITIAL COUNTS

82C54

Absolute Maximum Ratings

Supply Voltage +8.0V
 Input, Output or I/O Voltage GND-0.5V to V_{CC} +0.5V
 ESD Classification Class 1

Operating Conditions

Operating Voltage Range +4.5V to +5.5V
 Operating Temperature Range
 C82C54, C82C54-10, -12 0°C to +70°C
 I82C54, I82C54-10, -12 -40°C to +85°C
 M82C54, M82C54-10, -12 -55°C to +125°C

Thermal Information

Thermal Resistance (Typical)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
CERDIP Package	55	12
CLCC Package	65	14
PDIP Package	60	N/A
PLCC Package	65	N/A
SOIC Package.....	75	N/A
Storage Temperature Range.....	-65°C to +150°C	
Maximum Junction Temperature Ceramic Package	+175°C	
Maximum Junction Temperature Plastic Package.....	+150°C	
Maximum Lead Temperature Package (Soldering 10s)	+300°C	
(PLCC and SOIC - Lean Tips Only)		

Die Characteristics

Gate Count 2250 Gates

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.

DC Electrical Specifications $V_{CC} = +5.0V \pm 10\%$, $T_A = 0^{\circ}C$ to $+70^{\circ}C$ (C82C54, C82C54-10, C82C54-12)

$T_A = -40^{\circ}C$ to $+85^{\circ}C$ (I82C54, I82C54-10, I82C54-12)

$T_A = -55^{\circ}C$ to $+125^{\circ}C$ (M82C54, M82C54-10, M82C54-12)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
VIH	Logical One Input Voltage	2.0	-	V	C82C54, I82C54
		2.2	-	V	M82C54
VIL	Logical Zero Input Voltage	-	0.8	V	
VOH	Output HIGH Voltage	3.0	-	V	IOH = -2.5mA
		$V_{CC} - 0.4$	-	V	IOH = -100μA
VOL	Output LOW Voltage	-	0.4	V	IOL = +2.5mA
II	Input Leakage Current	-1	+1	μA	VIN = GND or V_{CC} DIP Pins 9,11,14-16,18-23
IO	Output Leakage Current	-10	+10	μA	VOUT = GND or V_{CC} DIP Pins 1-8
ICCSB	Standby Power Supply Current	-	10	μA	$V_{CC} = 5.5V$, VIN = GND or V_{CC} , Outputs Open, Counters Programmed
ICCOP	Operating Power Supply Current	-	10	mA	$V_{CC} = 5.5V$, CLK0 = CLK1 = CLK2 = 8MHz, VIN = GND or V_{CC} , Outputs Open

Capacitance $T_A = +25^{\circ}C$; All Measurements Referenced to Device GND, Note 1

SYMBOL	PARAMETER	TYP	UNITS	TEST CONDITIONS
CIN	Input Capacitance	20	pF	FREQ = 1MHz
COUT	Output Capacitance	20	pF	FREQ = 1MHz
CI/O	I/O Capacitance	20	pF	FREQ = 1MHz

NOTE:

- Not tested, but characterized at initial design and at major process/design changes.

82C54

AC Electrical Specifications $V_{CC} = +5.0V \pm 10\%$, $T_A = 0^{\circ}C$ to $+70^{\circ}C$ (C82C54, C82C54-10, C82C54-12)

$T_A = -40^{\circ}C$ to $+85^{\circ}C$ (I82C54, I82C54-10, I82C54-12)

$T_A = -55^{\circ}C$ to $+125^{\circ}C$ (M82C54, M82C54-10, M82C54-12)

SYMBOL	PARAMETER	82C54		82C54-10		82C54-12		UNITS	TEST CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX		
READ CYCLE									
(1)	TAR	Address Stable Before \overline{RD}	30	-	25	-	25	-	ns
(2)	TSR	\overline{CS} Stable Before \overline{RD}	0	-	0	-	0	-	ns
(3)	TRA	Address Hold Time After \overline{RD}	0	-	0	-	0	-	ns
(4)	TRR	\overline{RD} Pulse Width	150	-	95	-	95	-	ns
(5)	TRD	Data Delay from \overline{RD}	-	120	-	85	-	85	ns
(6)	TAD	Data Delay from Address	-	210	-	185	-	185	ns
(7)	TDF	\overline{RD} to Data Floating	5	85	5	65	5	65	ns
(8)	TRV	Command Recovery Time	200	-	165	-	165	-	ns
WRITE CYCLE									
(9)	TAW	Address Stable Before \overline{WR}	0	-	0	-	0	-	ns
(10)	TSW	\overline{CS} Stable Before \overline{WR}	0	-	0	-	0	-	ns
(11)	TWA	Address Hold Time After \overline{WR}	0	-	0	-	0	-	ns
(12)	TWW	\overline{WR} Pulse Width	95	-	95	-	95	-	ns
(13)	TDW	Data Setup Time Before \overline{WR}	140	-	95	-	95	-	ns
(14)	TWD	Data Hold Time After \overline{WR}	25	-	0	-	0	-	ns
(15)	TRV	Command Recovery Time	200	-	165	-	165	-	ns
CLOCK AND GATE									
(16)	TCLK	Clock Period	125	DC	100	DC	80	DC	ns
(17)	TPWH	High Pulse Width	60	-	30	-	30	-	ns
(18)	TPWL	Low Pulse Width	60	-	40	-	30	-	ns
(19)	TR	Clock Rise Time	-	25	-	25	-	25	ns
(20)	TF	Clock Fall Time	-	25	-	25	-	25	ns
(21)	TGW	Gate Width High	50	-	50	-	50	-	ns
(22)	TGL	Gate Width Low	50	-	50	-	50	-	ns
(23)	TGS	Gate Setup Time to CLK	50	-	40	-	40	-	ns
(24)	TGH	Gate Hold Time After CLK	50	-	50	-	50	-	ns
(25)	TOD	Output Delay from CLK	-	150	-	100	-	100	ns
(26)	TODG	Output Delay from Gate	-	120	-	100	-	100	ns
(27)	TWO	OUT Delay from Mode Write	-	260	-	240	-	240	ns
(28)	TWC	CLK Delay for Loading	0	55	0	55	0	55	ns
(29)	TWG	Gate Delay for Sampling	-5	40	-5	40	-5	40	ns
(30)	TCL	CLK Setup for Count Latch	-40	40	-40	40	-40	40	ns

NOTE:

- Not tested, but characterized at initial design and at major process/design changes.

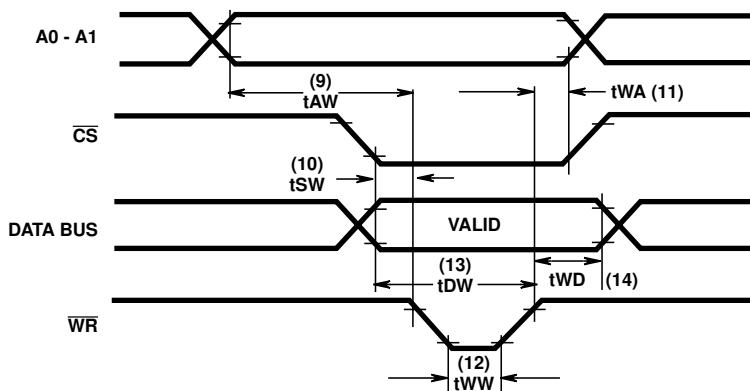
Timing Waveforms

FIGURE 17. WRITE

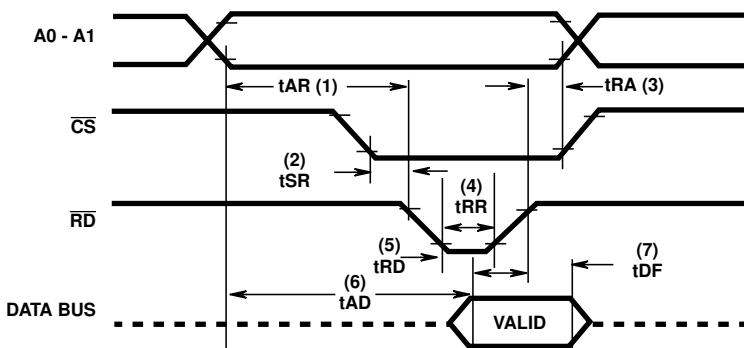


FIGURE 18. READ

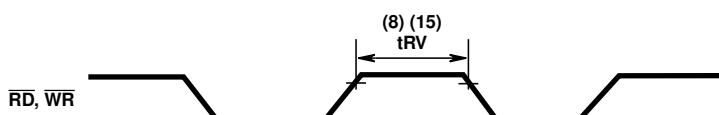


FIGURE 19. RECOVERY

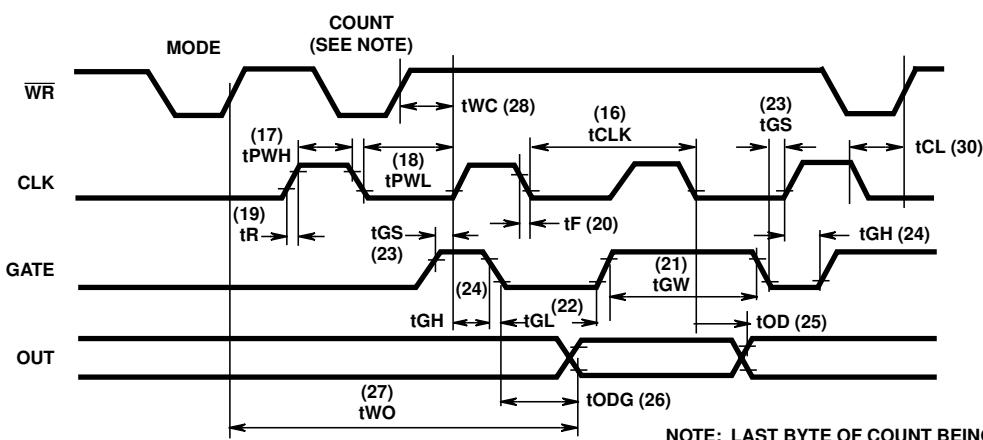
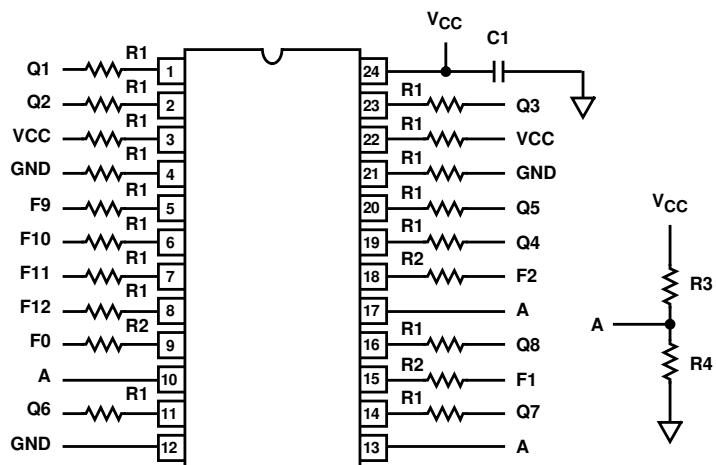


FIGURE 20. CLOCK AND GATE

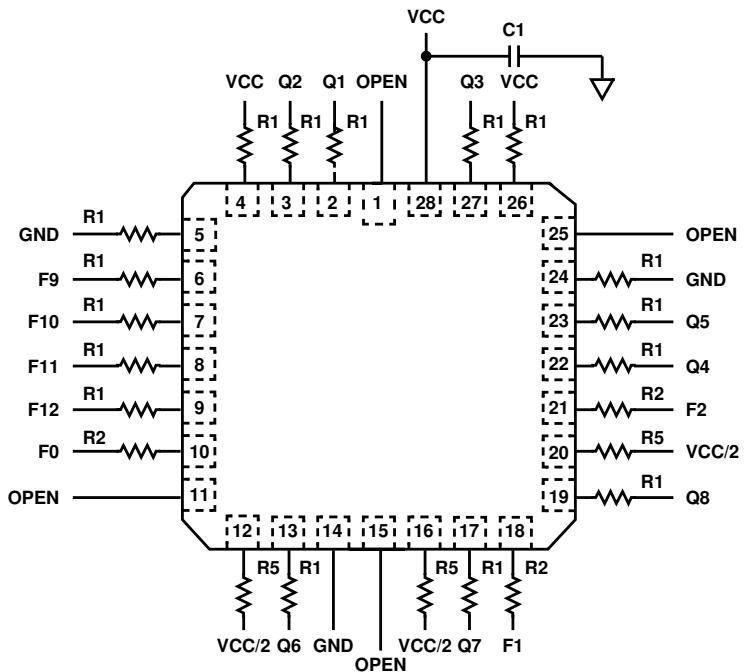
82C54

Burn-In Circuits

MD 82C54 CERDIP



MR 82C54 CLCC



NOTES:

1. $V_{CC} = 5.5V \pm 0.5V$
2. $GND = 0V$
3. $VIH = 4.5V \pm 10\%$
4. $VIL = -0.2V$ to $0.4V$
5. $R1 = 47k\Omega \pm 5\%$
6. $R2 = 1.0k\Omega \pm 5\%$
7. $R3 = 2.7k\Omega \pm 5\%$
8. $R4 = 1.8k\Omega \pm 5\%$
9. $R5 = 1.2k\Omega \pm 5\%$
10. $C1 = 0.01\mu F$ Min
11. $F0 = 100kHz \pm 10\%$
12. $F1 = F0/2, F2 = F1/2, \dots F12 = F11/2$

Die Characteristics**DIE DIMENSIONS:**

129mils x 155mils x 19mils
(3270 μ m x 3940 μ m x 483 μ m)

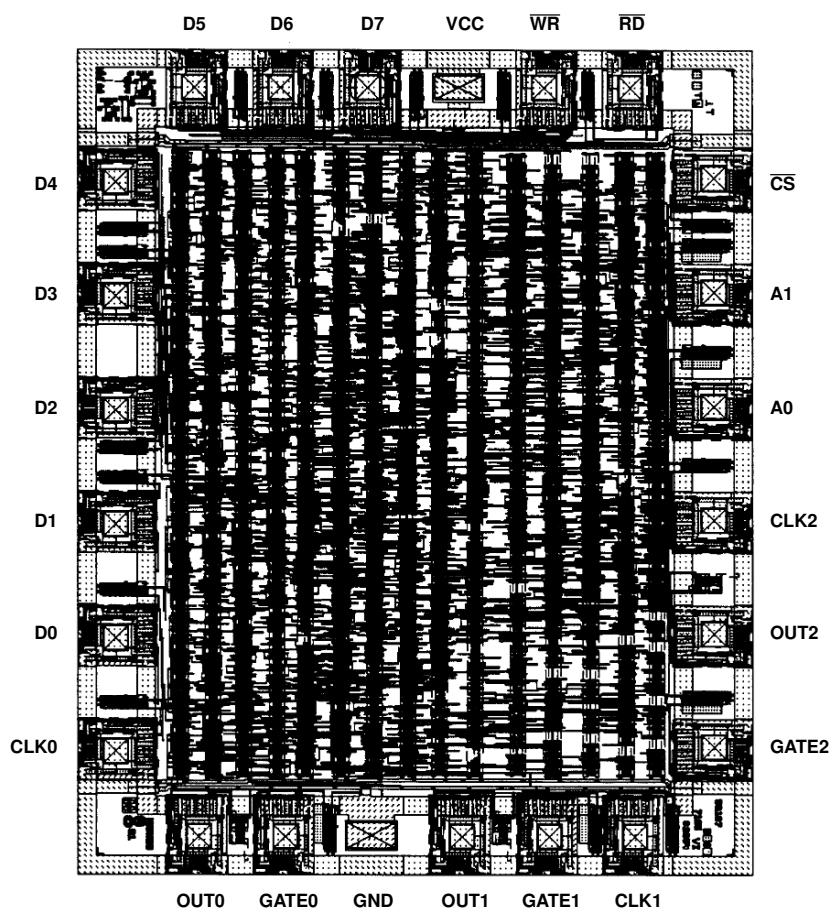
Thickness: Metal 1: 8k \AA \pm 0.75k \AA
Metal 2: 12k \AA \pm 1.0k \AA

METALLIZATION:

Type: Si-Al-Cu

GLASSIVATION:

Type: Nitrox
Thickness: 10k \AA \pm 3.0k \AA

Metallization Mask Layout



82C55A CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

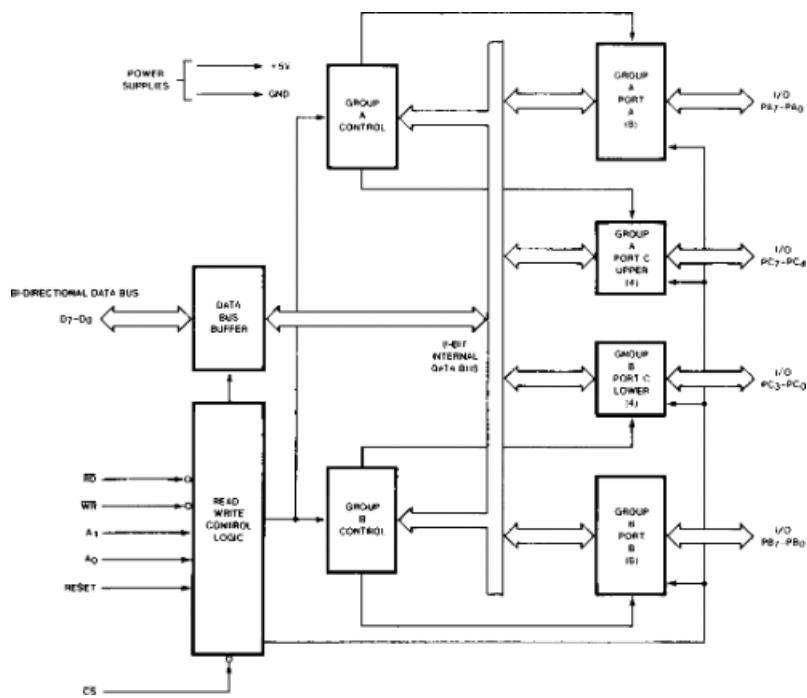


Figure 1. 82C55A Block Diagram

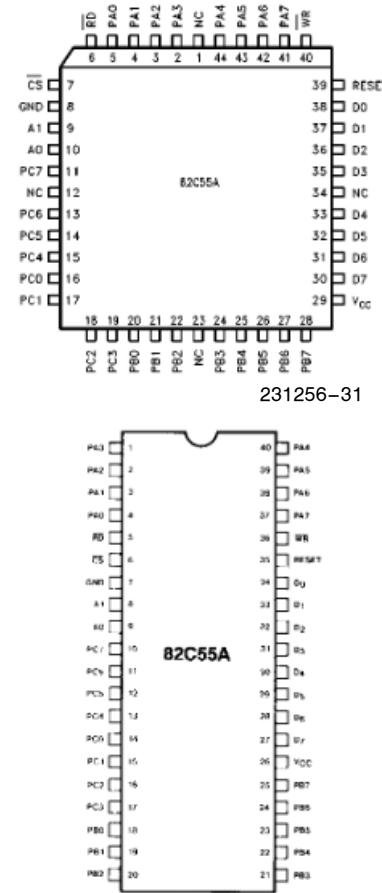


Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

Table 1. Pin Description

Symbol	Pin Number		Type	Name and Function																																		
	Dip	PLCC																																				
PA ₃₋₀	1–4	2–5	I/O	PORT A, PINS 0–3: Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.																																		
RD	5	6	I	READ CONTROL: This input is low during CPU read operations.																																		
CS	6	7	I	CHIP SELECT: A low on this input enables the 82C55A to respond to RD and WR signals. RD and WR are ignored otherwise.																																		
GND	7	8		System Ground																																		
A ₁₋₀	8–9	9–10	I	ADDRESS: These input signals, in conjunction RD and WR, control the selection of one of the three ports or the control word registers.																																		
				<table border="1"> <thead> <tr> <th>A₁</th> <th>A₀</th> <th>RD</th> <th>WR</th> <th>CS</th> <th>Input Operation (Read)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>Port A - Data Bus</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>Port B - Data Bus</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>Port C - Data Bus</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>Control Word - Data Bus</td> </tr> </tbody> </table>					A ₁	A ₀	RD	WR	CS	Input Operation (Read)	0	0	0	1	0	Port A - Data Bus	0	1	0	1	0	Port B - Data Bus	1	0	0	1	0	Port C - Data Bus	1	1	0	1	0	Control Word - Data Bus
A ₁	A ₀	RD	WR	CS	Input Operation (Read)																																	
0	0	0	1	0	Port A - Data Bus																																	
0	1	0	1	0	Port B - Data Bus																																	
1	0	0	1	0	Port C - Data Bus																																	
1	1	0	1	0	Control Word - Data Bus																																	
				<table border="1"> <thead> <tr> <th colspan="6">Output Operation (Write)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port A</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port B</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port C</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Control</td> </tr> </tbody> </table>					Output Operation (Write)						0	0	1	0	0	Data Bus - Port A	0	1	1	0	0	Data Bus - Port B	1	0	1	0	0	Data Bus - Port C	1	1	1	0	0	Data Bus - Control
Output Operation (Write)																																						
0	0	1	0	0	Data Bus - Port A																																	
0	1	1	0	0	Data Bus - Port B																																	
1	0	1	0	0	Data Bus - Port C																																	
1	1	1	0	0	Data Bus - Control																																	
				<table border="1"> <thead> <tr> <th colspan="6">Disable Function</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>1</td> <td>Data Bus - 3 - State</td> </tr> <tr> <td>X</td> <td>X</td> <td>1</td> <td>1</td> <td>0</td> <td>Data Bus - 3 - State</td> </tr> </tbody> </table>					Disable Function						X	X	X	X	1	Data Bus - 3 - State	X	X	1	1	0	Data Bus - 3 - State												
Disable Function																																						
X	X	X	X	1	Data Bus - 3 - State																																	
X	X	1	1	0	Data Bus - 3 - State																																	
PC ₇₋₄	10–13	11,13–15	I/O	PORT C, PINS 4–7: Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.																																		
PC ₀₋₃	14–17	16–19	I/O	PORT C, PINS 0–3: Lower nibble of Port C.																																		
PB ₀₋₇	18–25	20–22, 24–28	I/O	PORT B, PINS 0–7: An 8-bit data output latch/buffer and an 8-bit data input buffer.																																		
V _{CC}	26	29		SYSTEM POWER: + 5V Power Supply.																																		
D ₇₋₀	27–34	30–33, 35–38	I/O	DATA BUS: Bi-directional, tri-state data bus lines, connected to system data bus.																																		
RESET	35	39	I	RESET: A high on this input clears the control register and all ports are set to the input mode.																																		
WR	36	40	I	WRITE CONTROL: This input is low during CPU write operations.																																		
PA ₇₋₄	37–40	41–44	I/O	PORT A, PINS 4–7: Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.																																		
NC		1, 12, 23, 34		No Connect																																		

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7-C4)
Control Group B - Port B and Port C lower (C3-C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

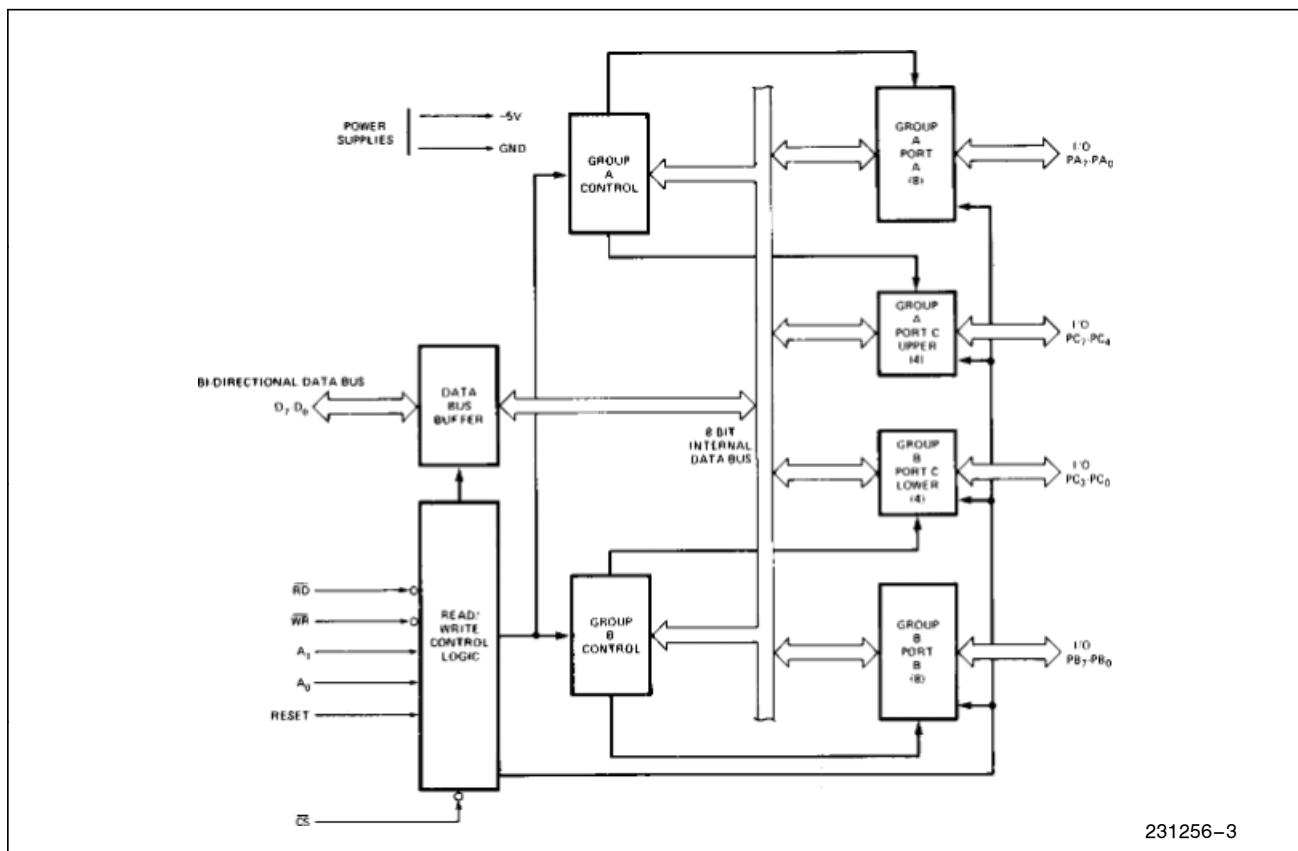


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

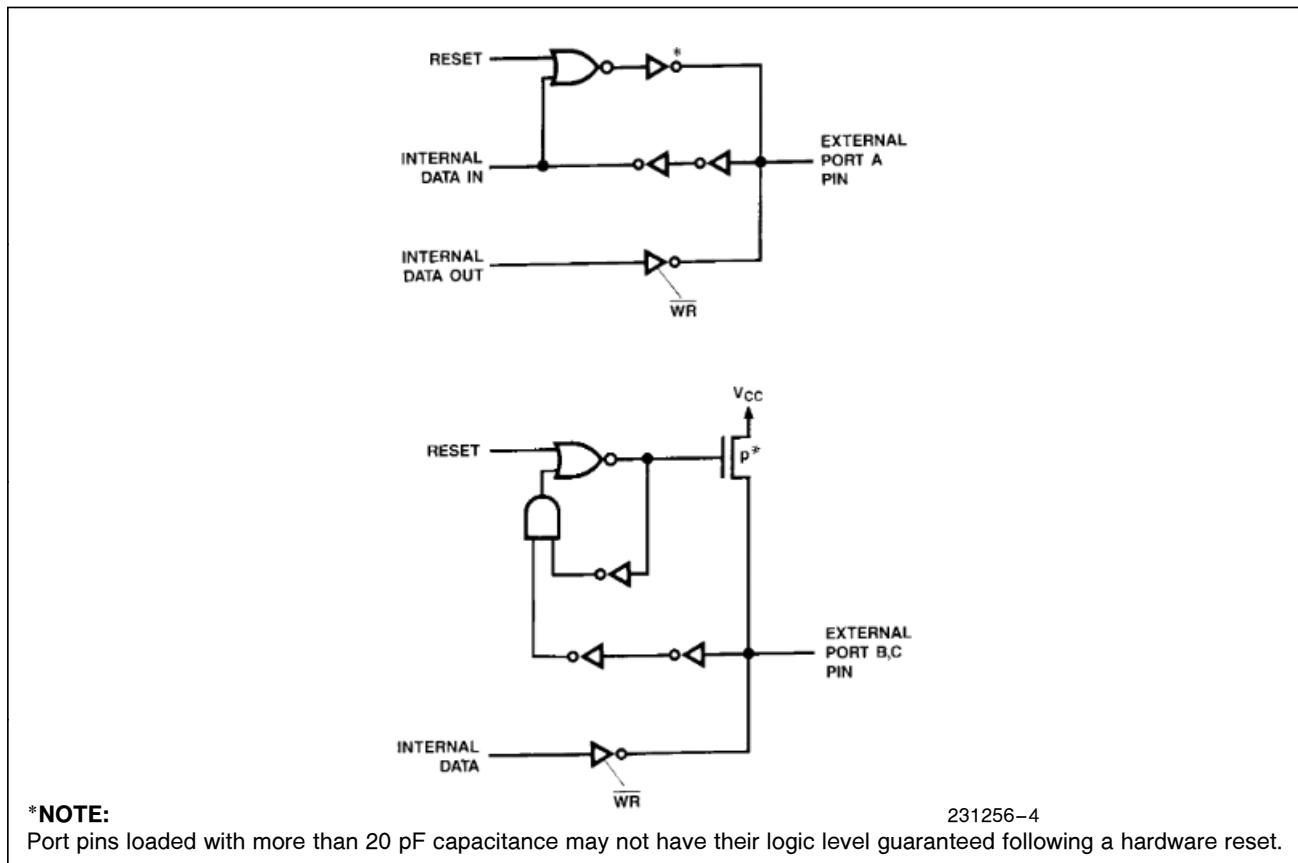


Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

- Mode 0 — Basic input/output
- Mode 1 — Strobed Input/output
- Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

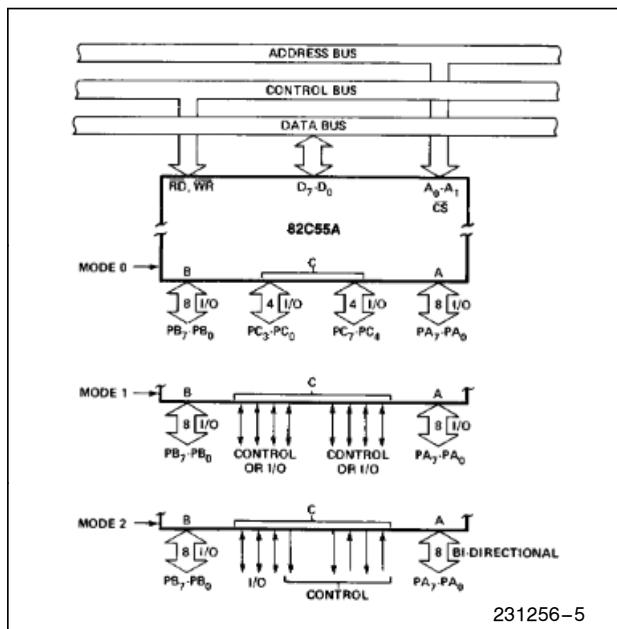


Figure 5. Basic Mode Definitions and Bus Interface

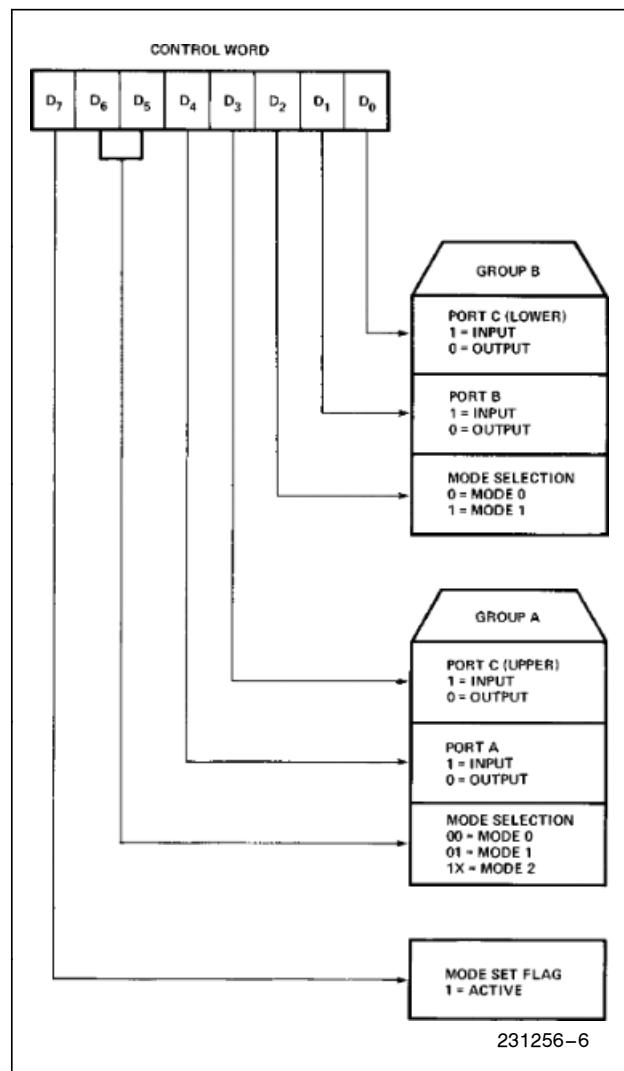


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

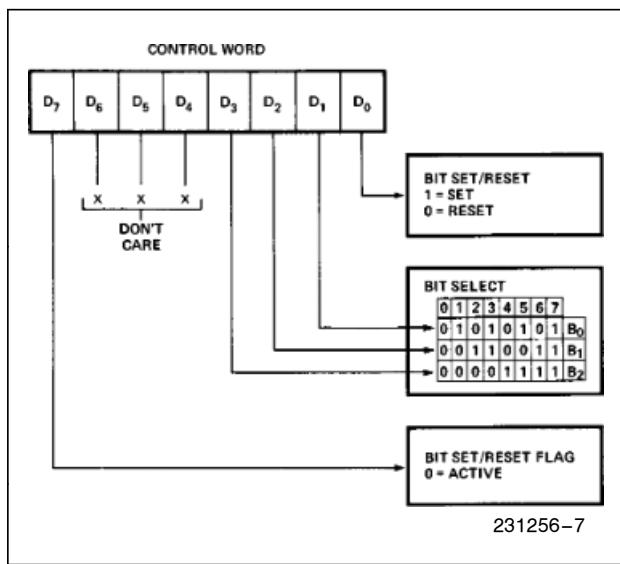


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:

All Mask flip-flops are automatically reset during mode selection and device Reset.

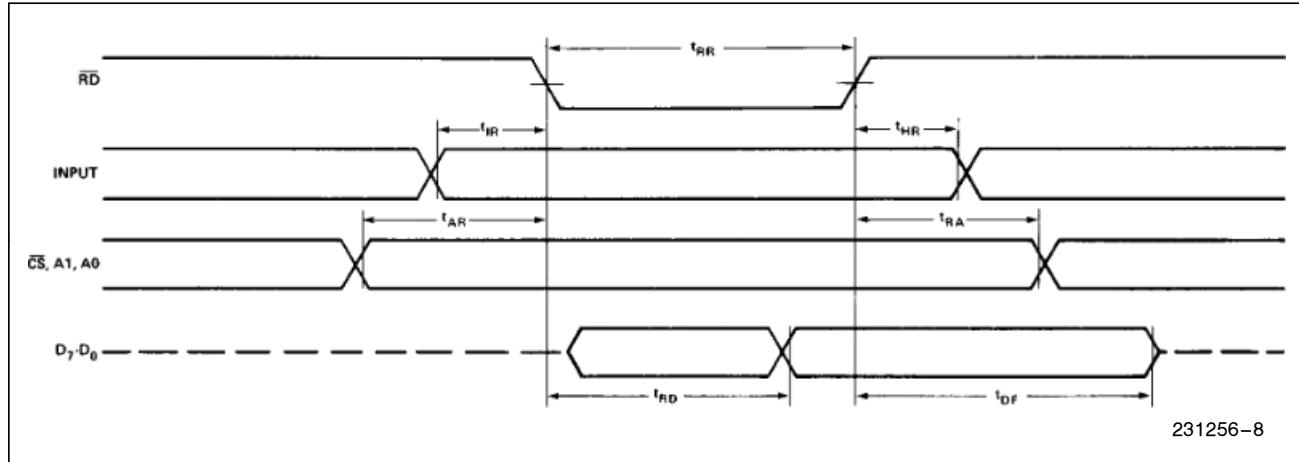
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

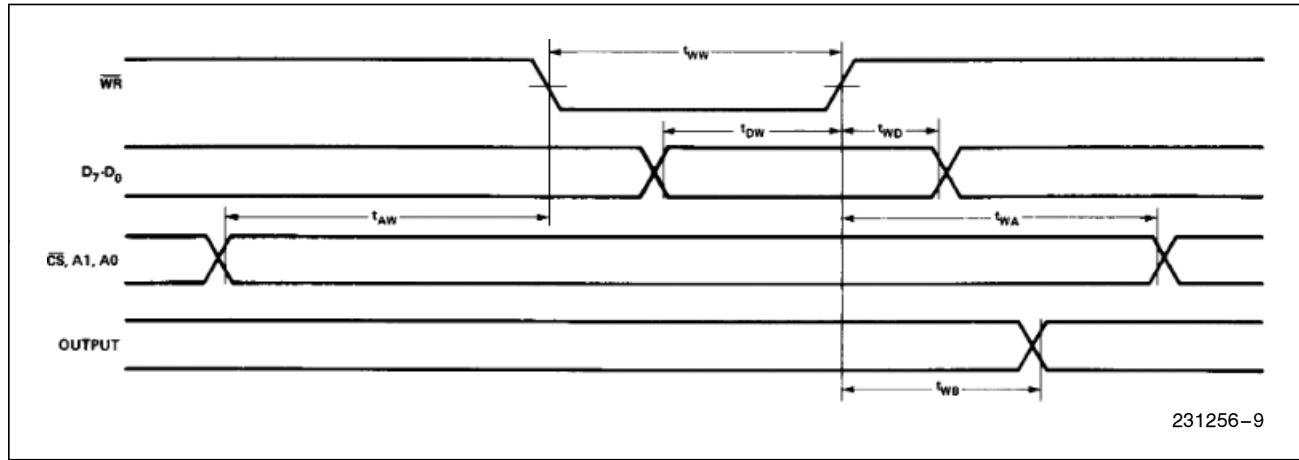
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)

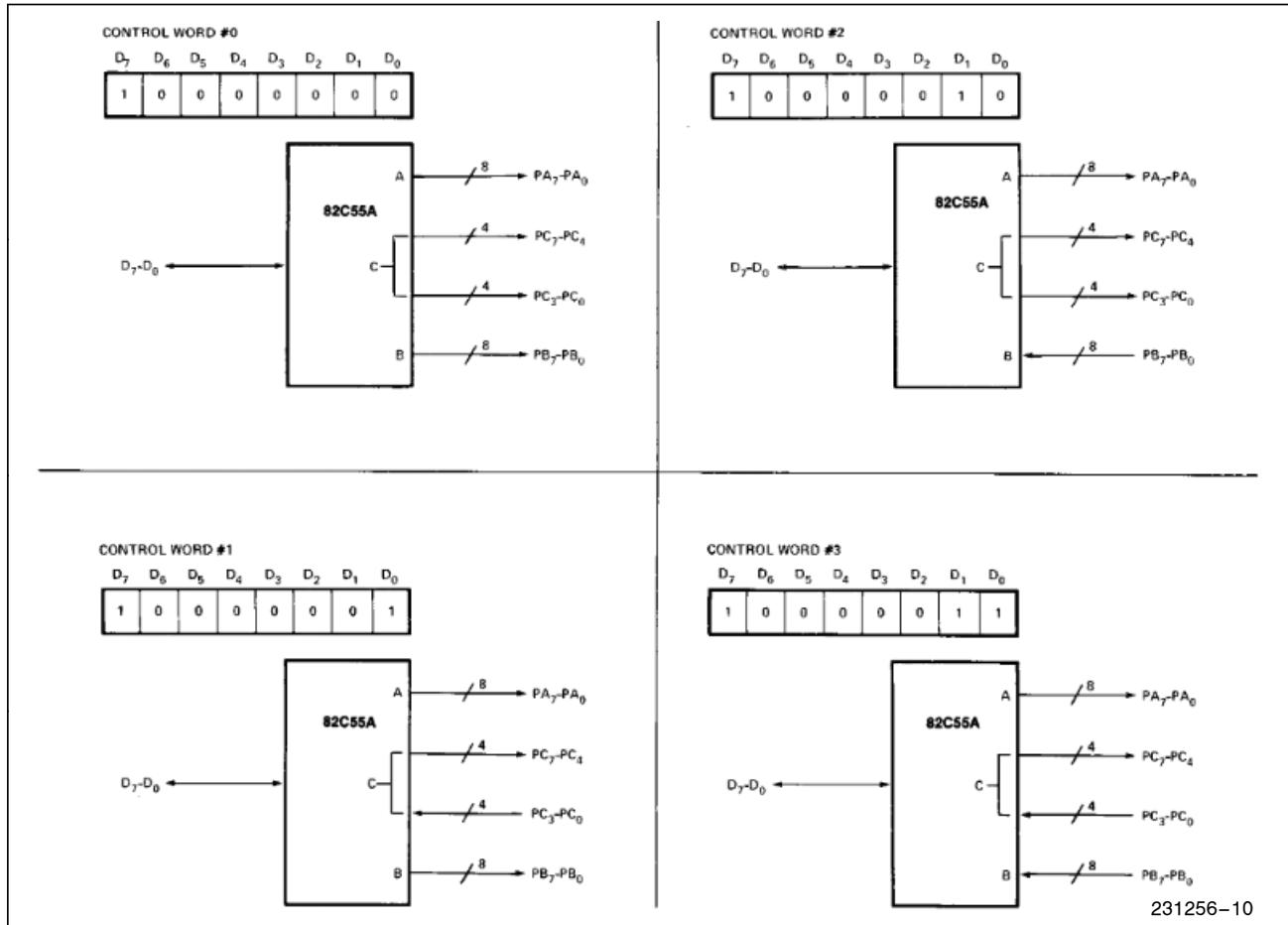


MODE 0 (BASIC OUTPUT)



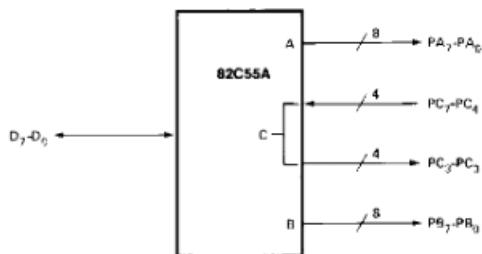
MODE 0 Port Definition

A		B		GROUP A						GROUP B	
D ₄	D ₃	D ₁	D ₀	PORT A		PORT C (UPPER)		#	PORT B	PORT C (LOWER)	
0	0	0	0	OUTPUT		OUTPUT		0	OUTPUT		OUTPUT
0	0	0	1	OUTPUT		OUTPUT		1	OUTPUT		INPUT
0	0	1	0	OUTPUT		OUTPUT		2	INPUT		OUTPUT
0	0	1	1	OUTPUT		OUTPUT		3	INPUT		INPUT
0	1	0	0	OUTPUT		INPUT		4	OUTPUT		OUTPUT
0	1	0	1	OUTPUT		INPUT		5	OUTPUT		INPUT
0	1	1	0	OUTPUT		INPUT		6	INPUT		OUTPUT
0	1	1	1	OUTPUT		INPUT		7	INPUT		INPUT
1	0	0	0	INPUT		OUTPUT		8	OUTPUT		OUTPUT
1	0	0	1	INPUT		OUTPUT		9	OUTPUT		INPUT
1	0	1	0	INPUT		OUTPUT		10	INPUT		OUTPUT
1	0	1	1	INPUT		OUTPUT		11	INPUT		INPUT
1	1	0	0	INPUT		INPUT		12	OUTPUT		OUTPUT
1	1	0	1	INPUT		INPUT		13	OUTPUT		INPUT
1	1	1	0	INPUT		INPUT		14	INPUT		OUTPUT
1	1	1	1	INPUT		INPUT		15	INPUT		INPUT

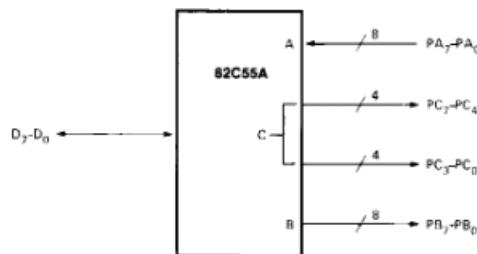
MODE 0 Configurations

MODE 0 Configurations (Continued)
CONTROL WORD #4

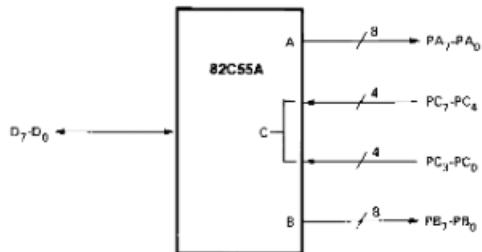
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	0	0


CONTROL WORD #6

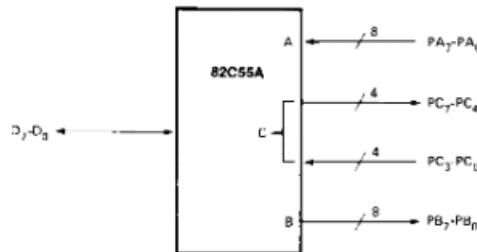
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	0	0


CONTROL WORD #5

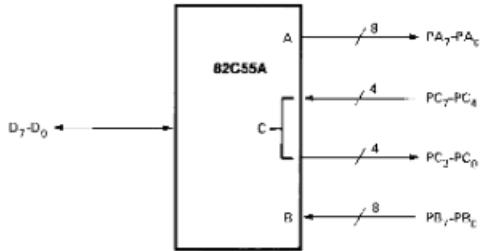
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	0	1


CONTROL WORD #9

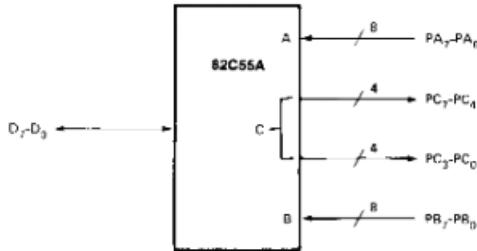
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	0	?


CONTROL WORD #6

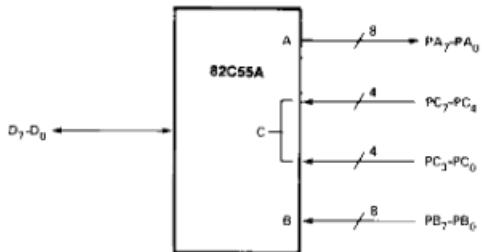
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	1	0


CONTROL WORD #10

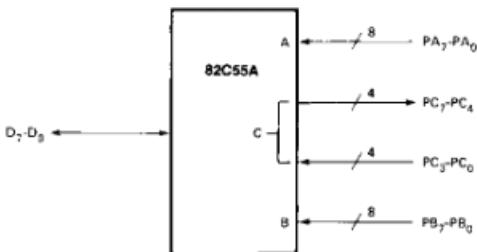
U ₇	U ₆	U ₅	U ₄	U ₃	U ₂	U ₁	U ₀
1	0	0	1	0	0	1	0


CONTROL WORD #7

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	1	1


CONTROL WORD #11

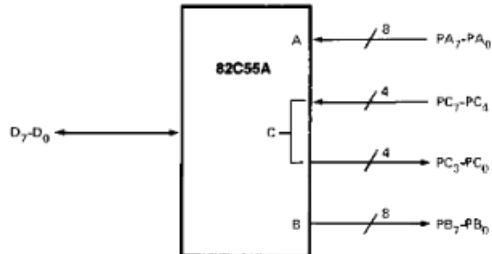
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	1	1



MODE 0 Configurations (Continued)

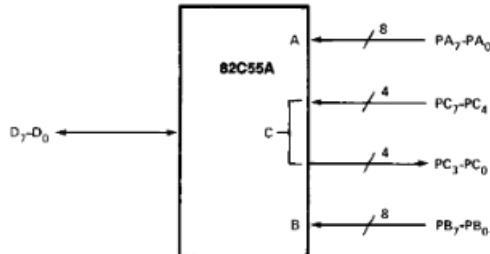
CONTROL WORD #12

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	1	0	0	0



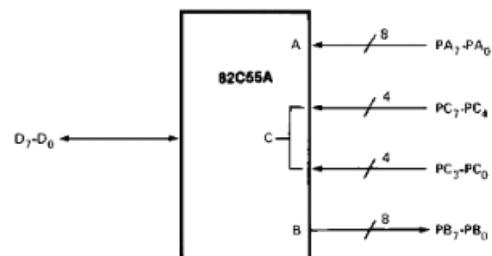
CONTROL WORD #14

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	1	0	1	0



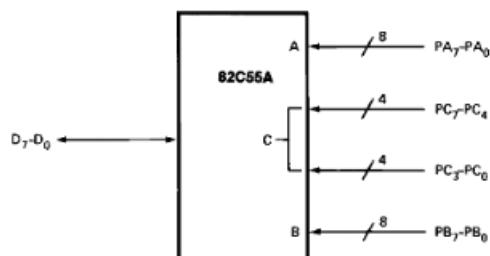
CONTROL WORD #13

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	1	0	0	1



CONTROL WORD #15

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	1	0	1	1



231256-12

Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

STB (Strobe Input). A “low” on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A “high” on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A “high” on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the \overline{STB} is a “one”, IBF is a “one” and INTE is a “one”. It is reset by the falling edge of \overline{RD} . This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC₄.

INTE B

Controlled by bit set/reset of PC₂.

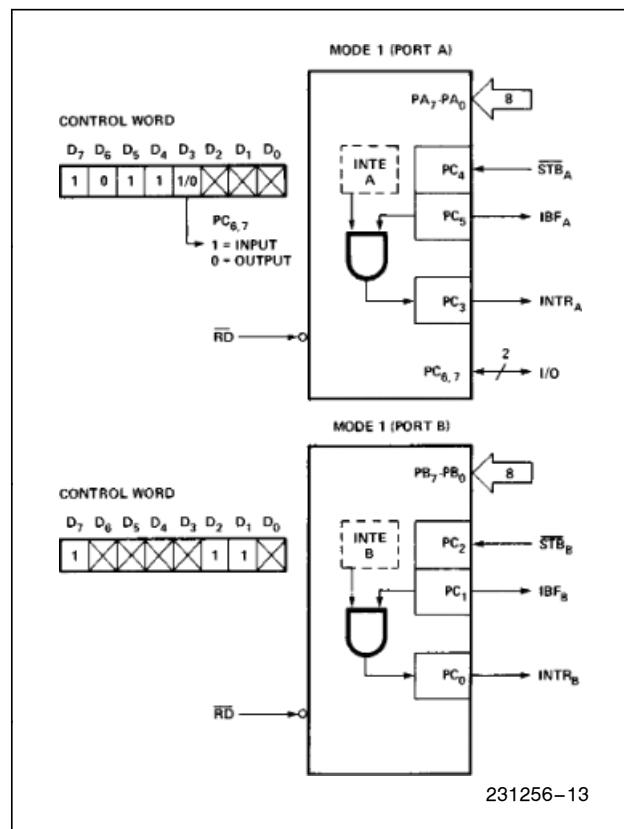


Figure 8. MODE 1 Input

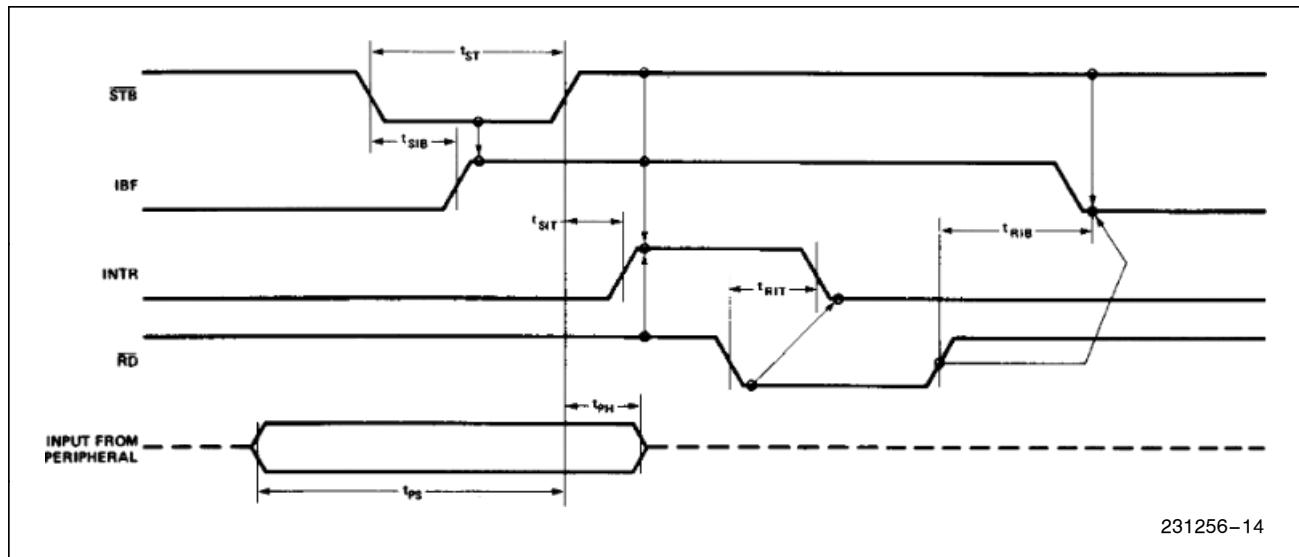


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

OBF (Output Buffer Full F/F). The \overline{OBF} output will go “low” to indicate that the CPU has written data out to the specified port. The \overline{OBF} F/F will be set by the rising edge of the WR input and reset by ACK Input being low.

ACK (Acknowledge Input). A “low” on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A “high” on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when ACK is a “one”, OBF is a “one” and INTA is a “one”. It is reset by the falling edge of WR.

INTA A

Controlled by bit set/reset of PC₆.

INTA B

Controlled by bit set/reset of PC₂.

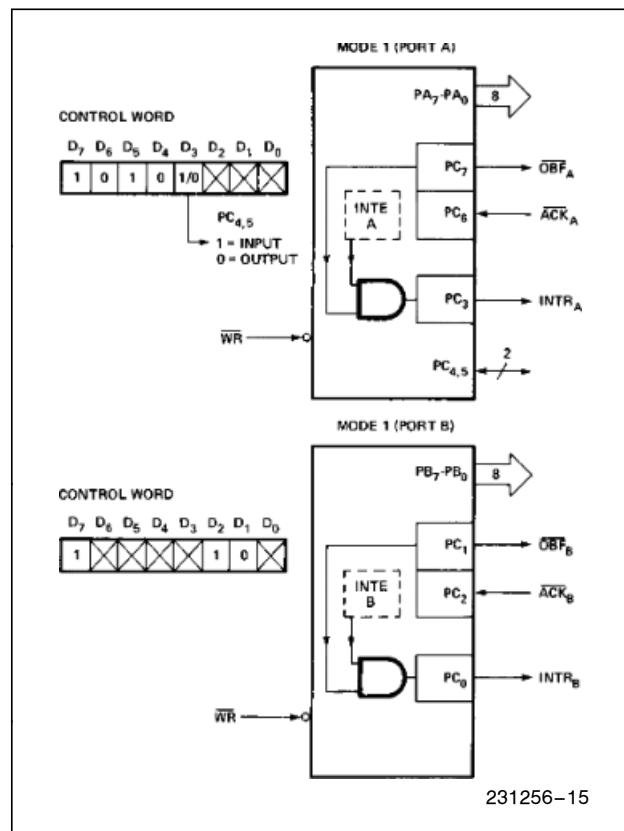


Figure 10. MODE 1 Output

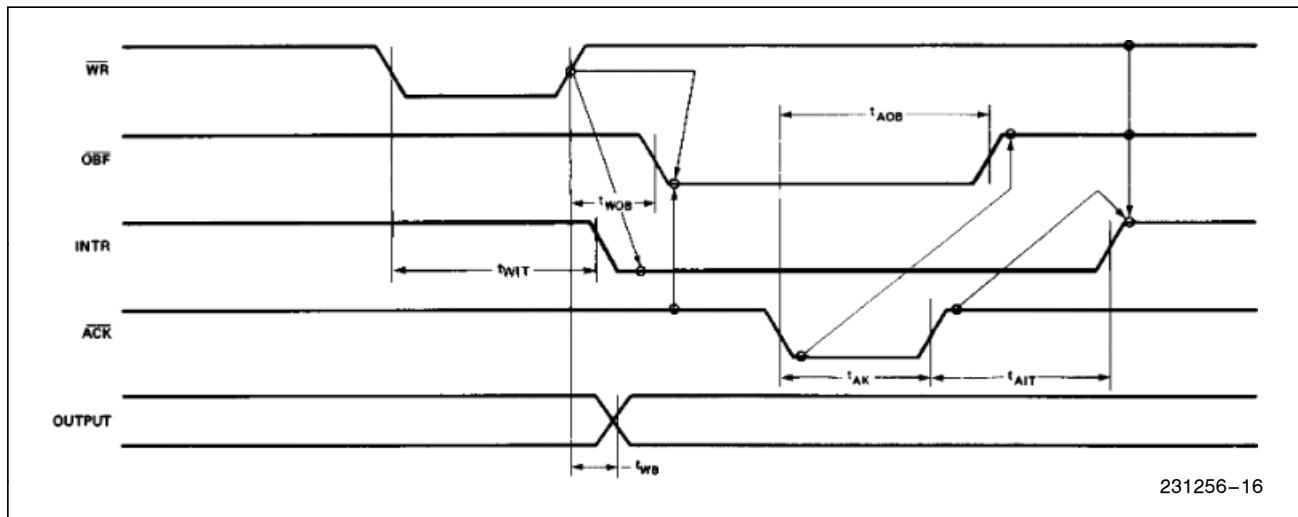


Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

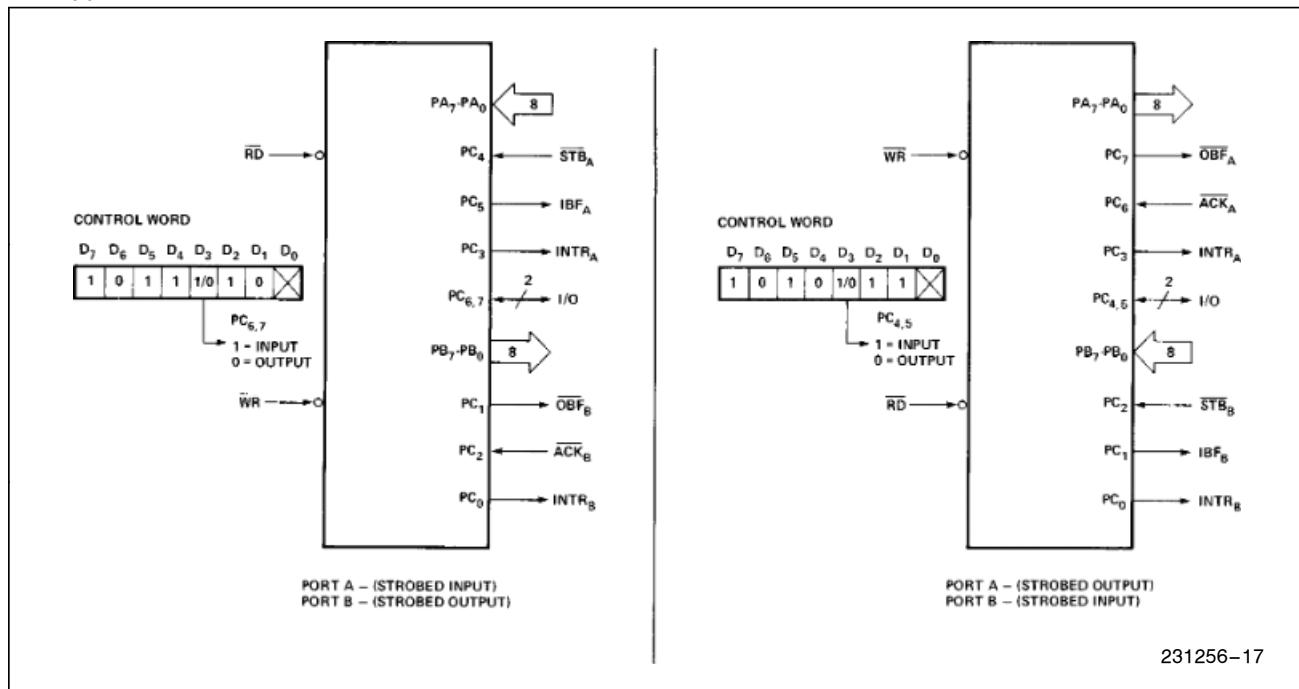


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). “Handshaking” signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

OBF (Output Buffer Full). The OBF output will go “low” to indicate that the CPU has written data out to port A.

ACK (Acknowledge). A “low” on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC₆.

Input Operations

STB (Strobe Input). A “low” on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A “high” on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC₄.

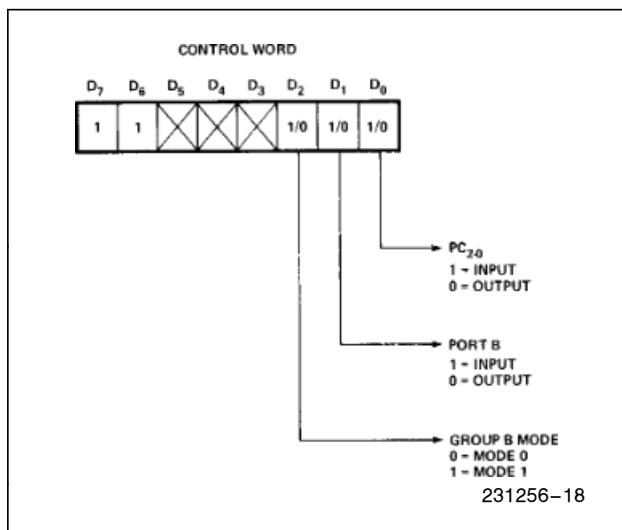


Figure 13. MODE Control Word

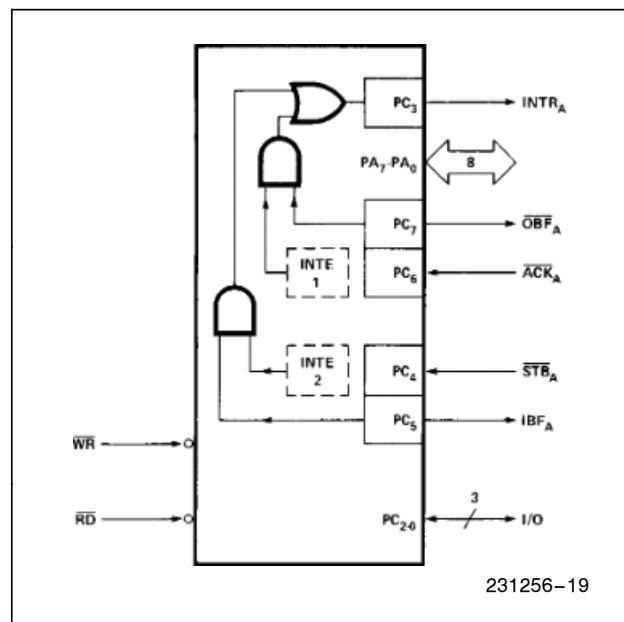


Figure 14. MODE 2

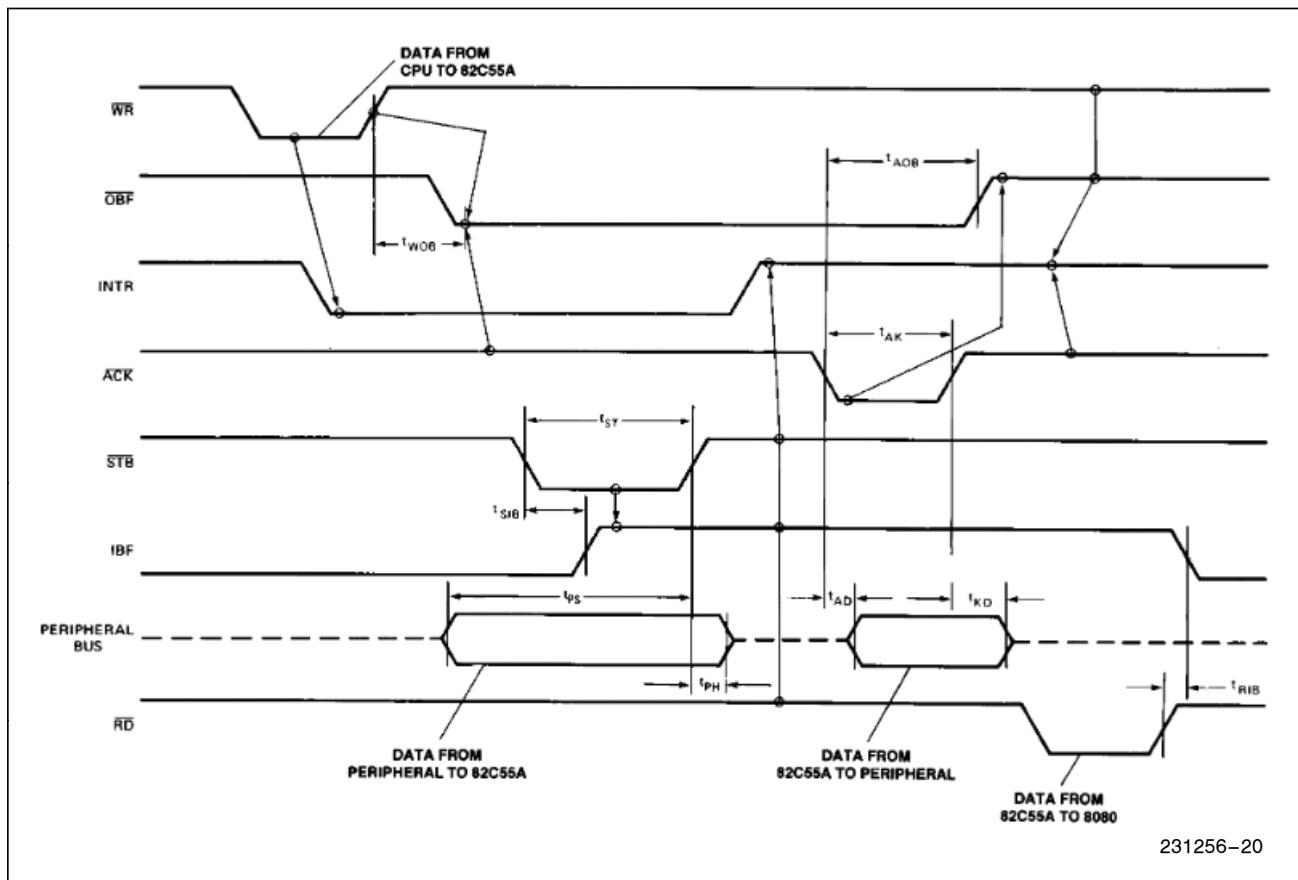


Figure 15. MODE 2 (Bidirectional)

NOTE:

Any sequence where WR occurs before ACK, and STB occurs before RD is permissible.
(INTR = IBF • MASK • STB • RD + OBF • MASK • ACK • WR)

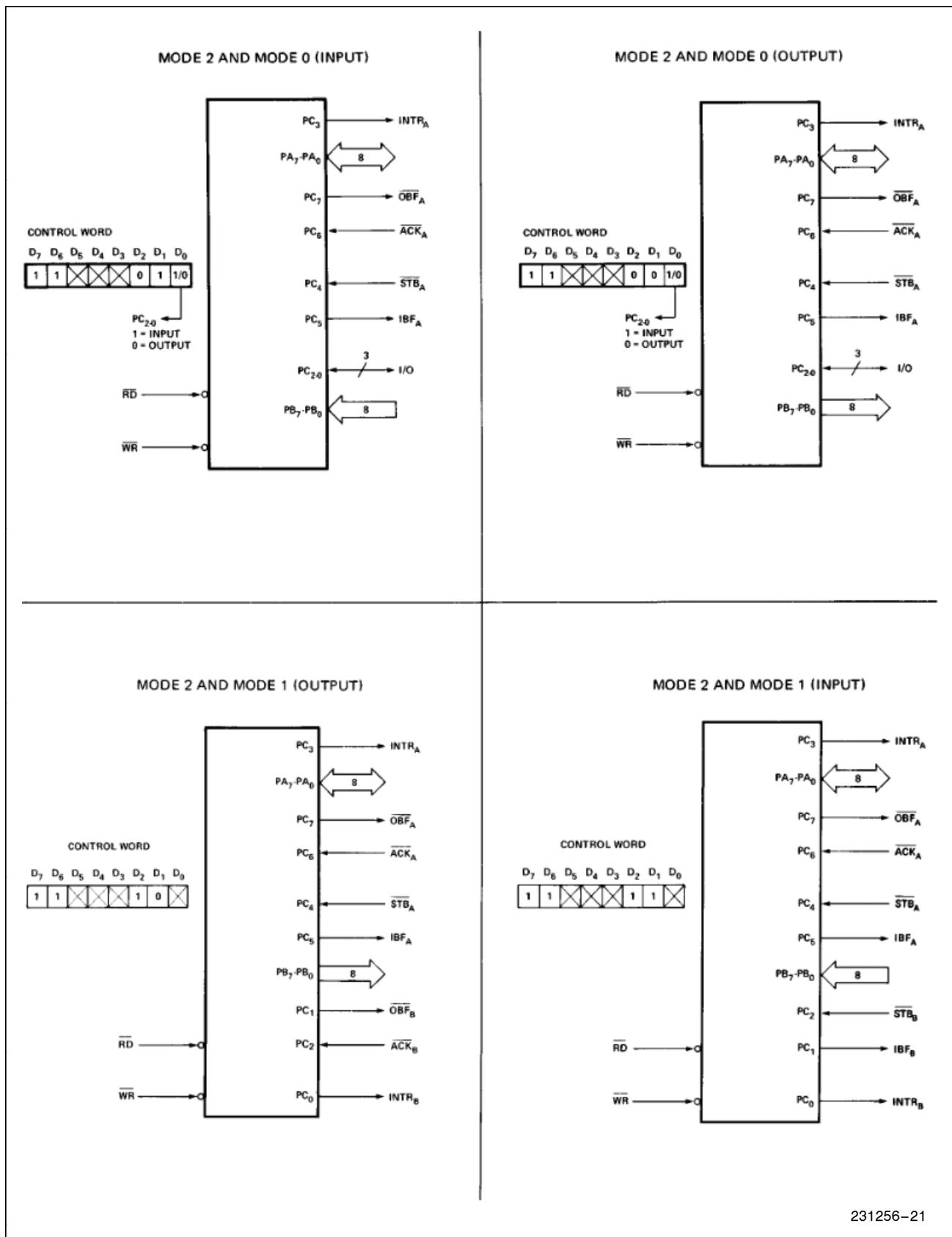


Figure 16. MODE 1/4 Combinations

231256-21

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA ₀	IN	OUT	IN	OUT	↔	
PA ₁	IN	OUT	IN	OUT	↔	
PA ₂	IN	OUT	IN	OUT	↔	
PA ₃	IN	OUT	IN	OUT	↔	
PA ₄	IN	OUT	IN	OUT	↔	
PA ₅	IN	OUT	IN	OUT	↔	
PA ₆	IN	OUT	IN	OUT	↔	
PA ₇	IN	OUT	IN	OUT	↔	
PB ₀	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB ₁	IN	OUT	IN	OUT	—	
PB ₂	IN	OUT	IN	OUT	—	
PB ₃	IN	OUT	IN	OUT	—	
PB ₄	IN	OUT	IN	OUT	—	
PB ₅	IN	OUT	IN	OUT	—	
PB ₆	IN	OUT	IN	OUT	—	
PB ₇	IN	OUT	IN	OUT	—	
PC ₀	IN	OUT	INTR _B	INTR _B	I/O	
PC ₁	IN	OUT	IBF _B	OBF _B	I/O	
PC ₂	IN	OUT	STB _B	ACK _B	I/O	
PC ₃	IN	OUT	INTR _A	INTR _A	INTRA	
PC ₄	IN	OUT	STB _A	I/O	STBA	
PC ₅	IN	OUT	IBF _A	I/O	IBFA	
PC ₆	IN	OUT	I/O	ACK _A	ACKA	
PC ₇	IN	OUT	I/O	OBF _A	OBFA	

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the ACK and STB lines, will be placed on the data bus. In place of the ACK and STB line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to

change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and OBF) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including ACK and STB lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the ACK and STB lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

INPUT CONFIGURATION								
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
I/O	I/O	IBFA	INTEA	INTR _A	INTEB	IBFB	INTR _B	
GROUP A					GROUP B			
OUTPUT CONFIGURATIONS								
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
OBFA	INTEA	I/O	I/O	INTR _A	INTEB	OBFB	INTR _B	
GROUP A					GROUP B			

Figure 17a. MODE 1 Status Word Format

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
OBF _A	INTE ₁	IBF _A	INTE ₂	INTR _A			
GROUP A				GROUP B			
(Defined By Mode 0 or Mode 1 Selection)							

Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	$\overline{\text{ACK}_B}$ (Output Mode 1) or $\overline{\text{STB}_B}$ (Input Mode 1)
INTE A2	PC4	$\overline{\text{STB}_A}$ (Input Mode 1 or Mode 2)
INTE A1	PC6	ACK_A (Output Mode 1 or Mode 2)

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to + 70°C
Storage Temperature	- 65°C to + 150°C
Supply Voltage	- 0.5 to + 8.0V
Operating Voltage	+ 4V to + 7V
Voltage on any Input	GND - 2V to + 6.5V
Voltage on any Output	GND - 0.5V to V _{CC} + 0.5V
Power Dissipation	1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, GND = 0V (T_A = - 40°C to + 85°C for Extended Temperture)

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	- 0.5	0.8	V	
V _{IH}	Input High Voltage	2.0	V _{CC}	V	
V _{OL}	Output Low Voltage		0.4	V	I _{OL} = 2.5 mA
V _{OH}	Output High Voltage	3.0 V _{CC} - 0.4		V	I _{OH} = - 2.5 mA I _{OH} = - 100 μA
I _{IL}	Input Leakage Current		± 1	μA	V _{IN} = V _{CC} to 0V (Note 1)
I _{OFL}	Output Float Leakage Current		± 10	μA	V _{IN} = V _{CC} to 0V (Note 2)
I _{DAR}	Darlington Drive Current	± 2.5	(Note 4)	mA	Ports A, B, C R _{ext} = 500Ω V _{ext} = 1.7V
I _{PHL}	Port Hold Low Leakage Current	+ 50	+ 300	μA	V _{OUT} = 1.0V Port A only
I _{PHH}	Port Hold High Leakage Current	- 50	- 300	μA	V _{OUT} = 3.0V Ports A, B, C
I _{PHLO}	Port Hold Low Overdrive Current	- 350		μA	V _{OUT} = 0.8V
I _{PHHO}	Port Hold High Overdrive Current	+ 350		μA	V _{OUT} = 3.0V
I _{CC}	V _{CC} Supply Current		10	mA	(Note 3)
I _{CCSB}	V _{CC} Supply Current-Standby		10	μA	V _{CC} = 5.5V V _{IN} = V _{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low CS = High Reset = Low Pure Inputs = Low/High

NOTES:

1. Pins A₁, A₀, CS, WR, RD, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

CAPACITANCE

$T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pins returned to GND $f_c = 1 \text{ MHz}^{(5)}$
$C_{I/O}$	I/O Capacitance		20	pF	

NOTE:

5. Sampled not 100% tested.

A.C. CHARACTERISTICS

$T_A = 0^\circ \text{ to } 70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$, $\text{GND} = 0\text{V}$

$T_A = -40^\circ\text{C} \text{ to } +85^\circ\text{C}$ for Extended Temperature

BUS PARAMETERS

READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{\text{RD}} \downarrow$	0		ns	
t_{RA}	Address Hold Time After $\overline{\text{RD}} \uparrow$	0		ns	
t_{RR}	$\overline{\text{RD}}$ Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{\text{RD}} \downarrow$		120	ns	
t_{DF}	$\overline{\text{RD}} \uparrow$ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{\text{RD}}/\overline{\text{WR}}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{\text{WR}} \downarrow$	0		ns	
t_{WA}	Address Hold Time After $\overline{\text{WR}} \uparrow$	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	$\overline{\text{WR}}$ Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{\text{WR}} \uparrow$	100		ns	
t_{WD}	Data Hold Time After $\overline{\text{WR}} \uparrow$	30		ns	Ports A & B
		30		ns	Port C

OTHER TIMINGS

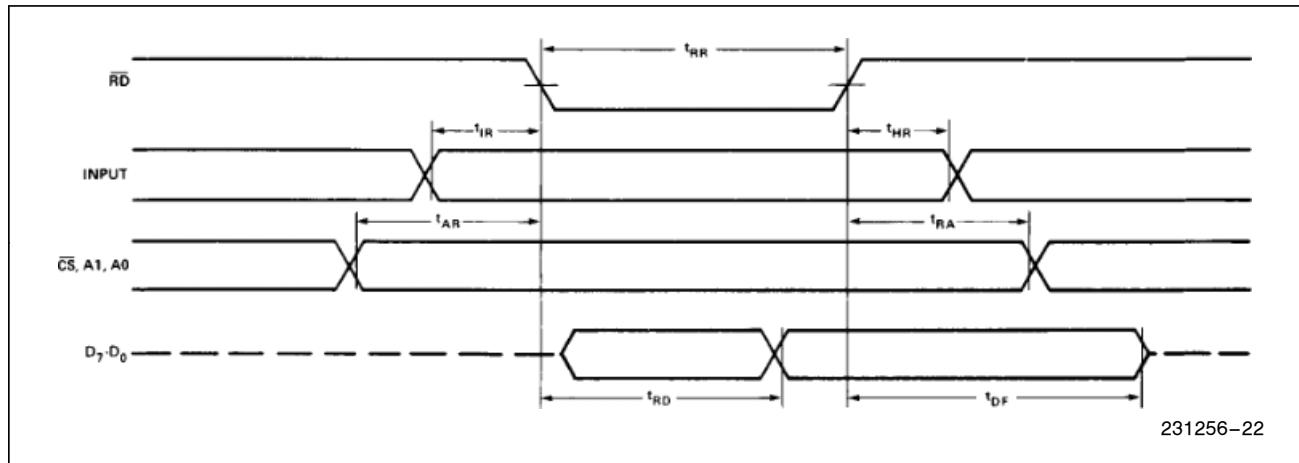
Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t_{WB}	$\overline{WR} = 1$ to Output		350	ns	
t_{IR}	Peripheral Data Before \overline{RD}	0		ns	
t_{HR}	Peripheral Data After \overline{RD}	0		ns	
t_{AK}	\overline{ACK} Pulse Width	200		ns	
t_{ST}	\overline{STB} Pulse Width	100		ns	
t_{PS}	Per. Data Before \overline{STB} High	20		ns	
t_{PH}	Per. Data After \overline{STB} High	50		ns	
t_{AD}	$\overline{ACK} = 0$ to Output		175	ns	
t_{KD}	$\overline{ACK} = 1$ to Output Float	20	250	ns	
t_{WOB}	$\overline{WR} = 1$ to $\overline{OBF} = 0$		150	ns	
t_{AOB}	$\overline{ACK} = 0$ to $\overline{OBF} = 1$		150	ns	
t_{SIB}	$\overline{STB} = 0$ to IBF = 1		150	ns	
t_{RIB}	$\overline{RD} = 1$ to IBF = 0		150	ns	
t_{RIT}	$\overline{RD} = 0$ to INTR = 0		200	ns	
t_{SIT}	$\overline{STB} = 1$ to INTR = 1		150	ns	
t_{AIT}	$\overline{ACK} = 1$ to INTR = 1		150	ns	
t_{WIT}	$\overline{WR} = 0$ to INTR = 0		200	ns	see note 1
t_{RES}	Reset Pulse Width	500		ns	see note 2

NOTE:

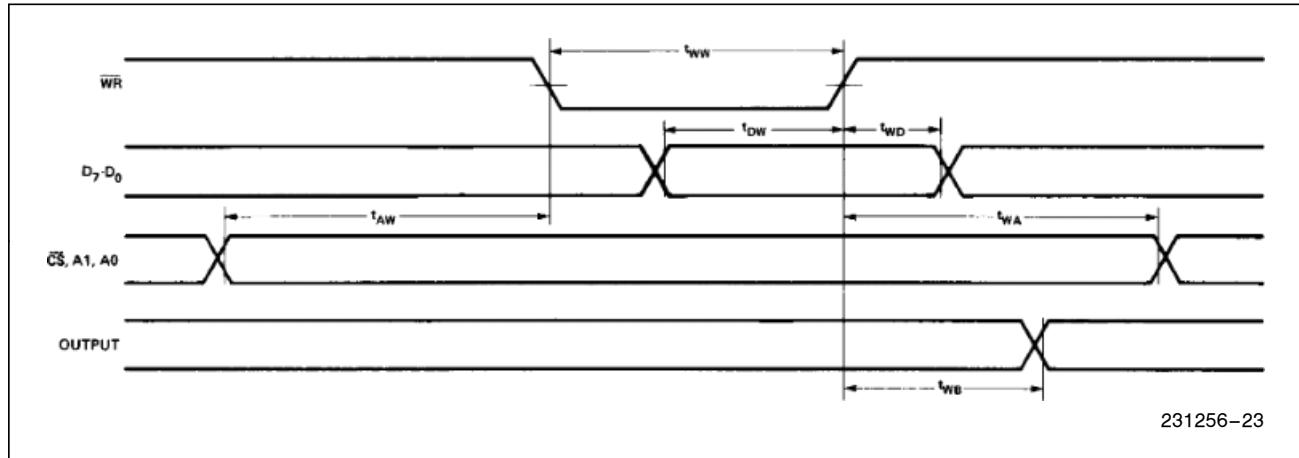
1. INTR ↑ may occur as early as $\overline{WR} \downarrow$.
2. Pulse width of initial Reset pulse after power on must be at least 50 μ Sec. Subsequent Reset pulses may be 500 ns minimum. The output Ports A, B, or C may glitch low during the reset pulse but all port pins will be held at a logic "one" level after the reset pulse.

WAVEFORMS

MODE 0 (BASIC INPUT)

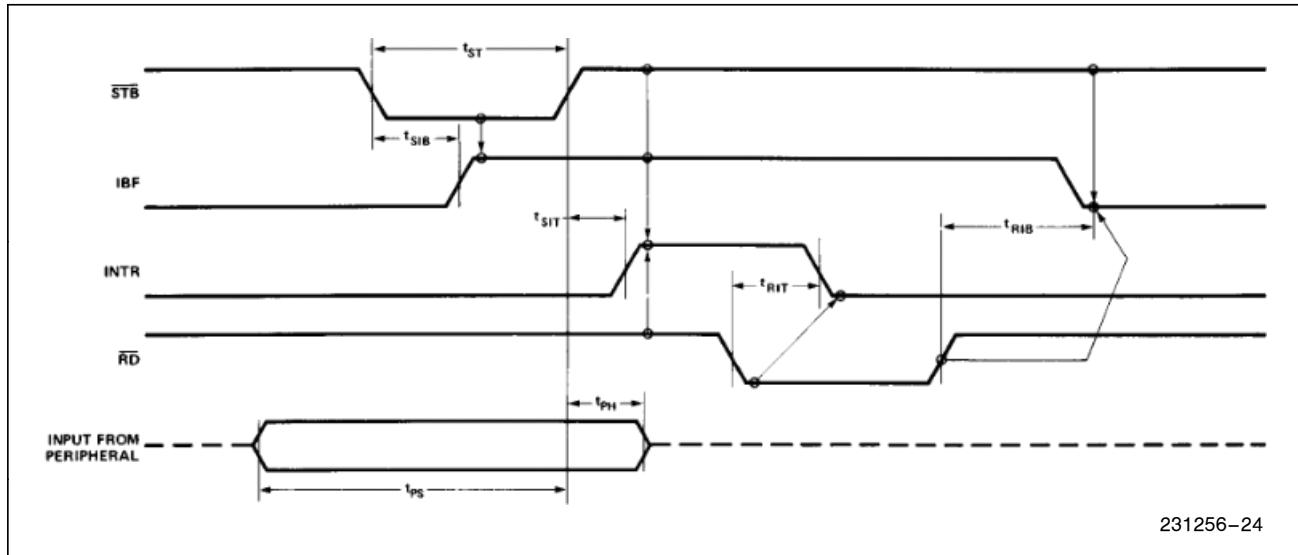


MODE 0 (BASIC OUTPUT)

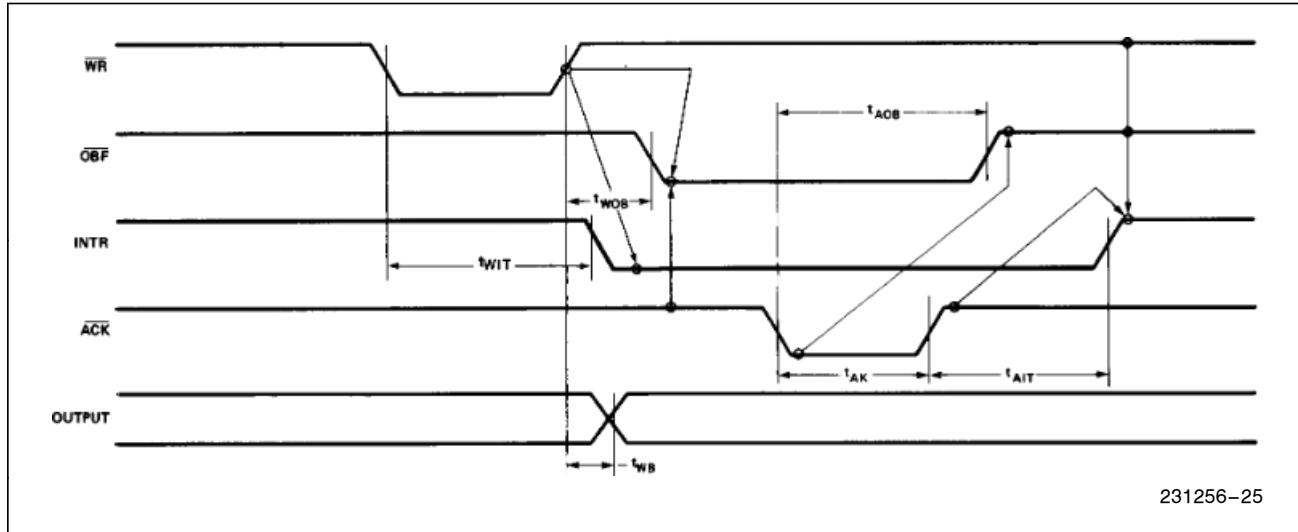


WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

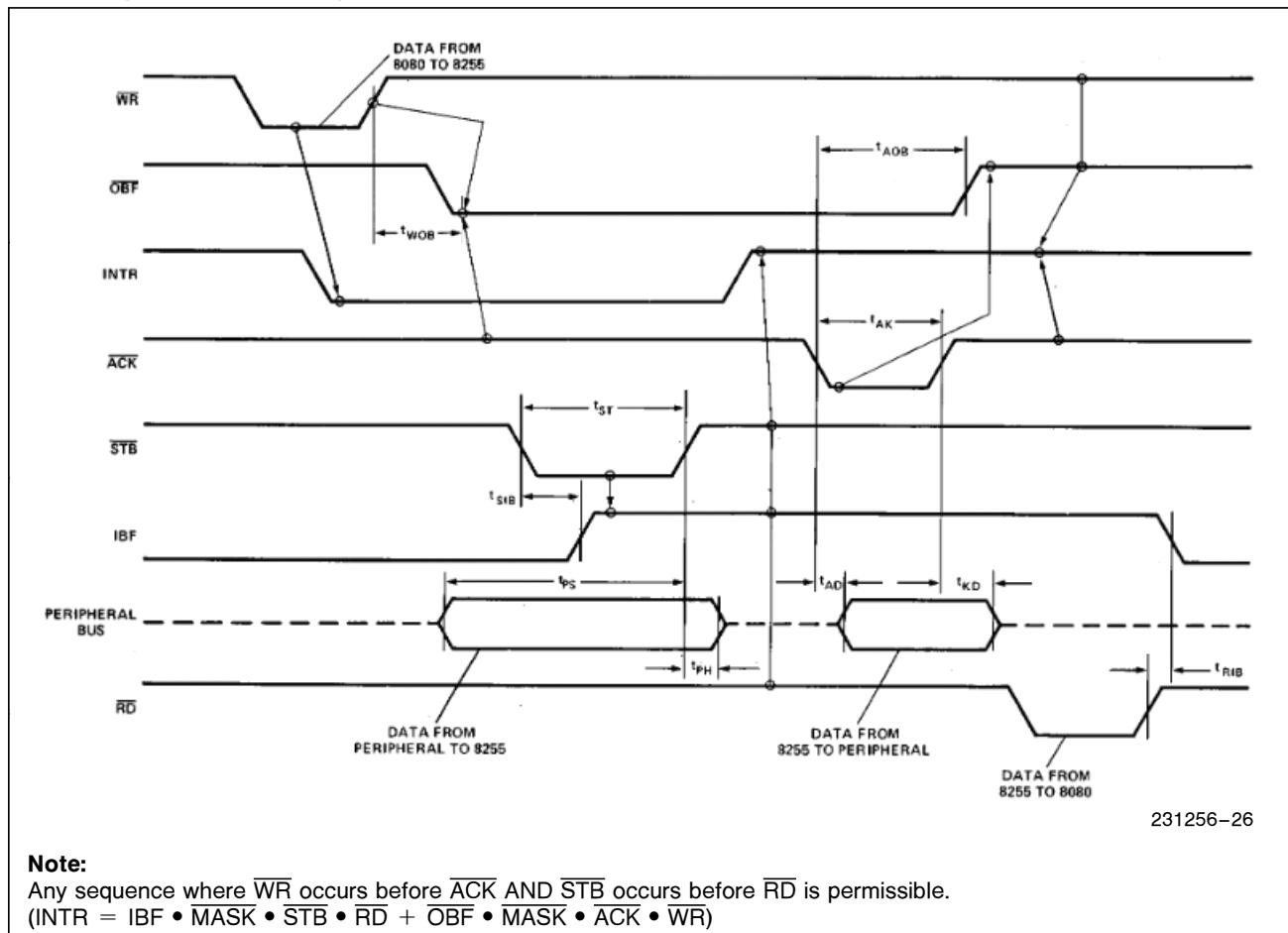


MODE 1 (STROBED OUTPUT)

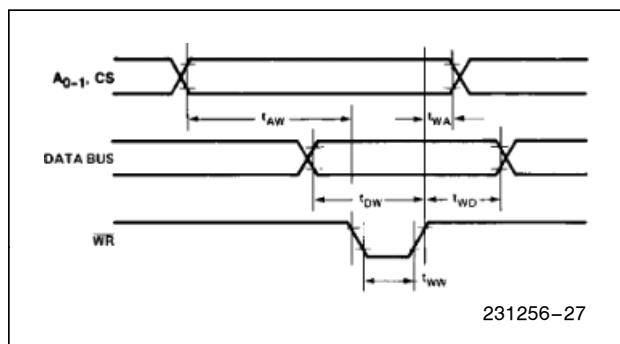


WAVEFORMS (Continued)

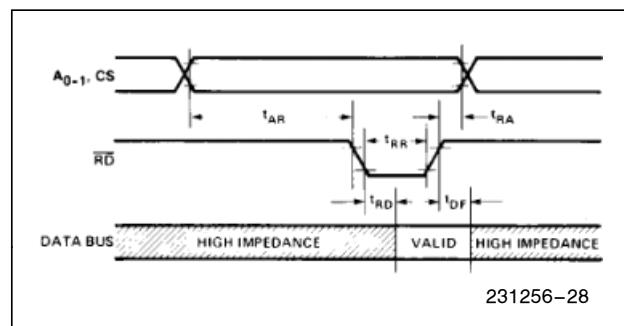
MODE 2 (BIDIRECTIONAL)



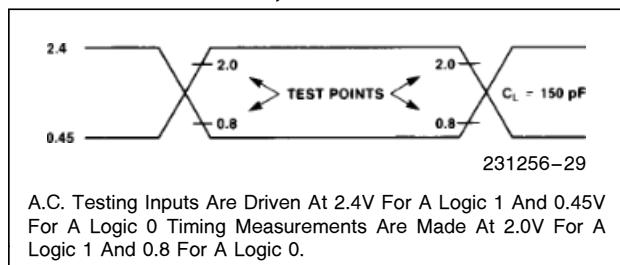
WRITE TIMING



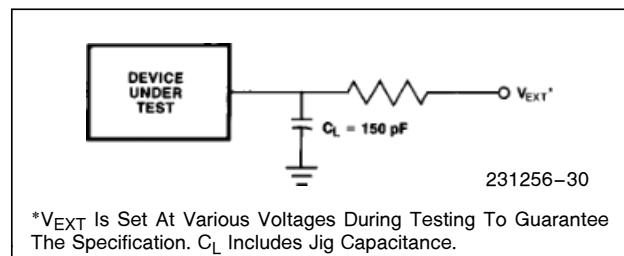
READ TIMING



A.C. TESTING INPUT, OUTPUT WAVEFORM



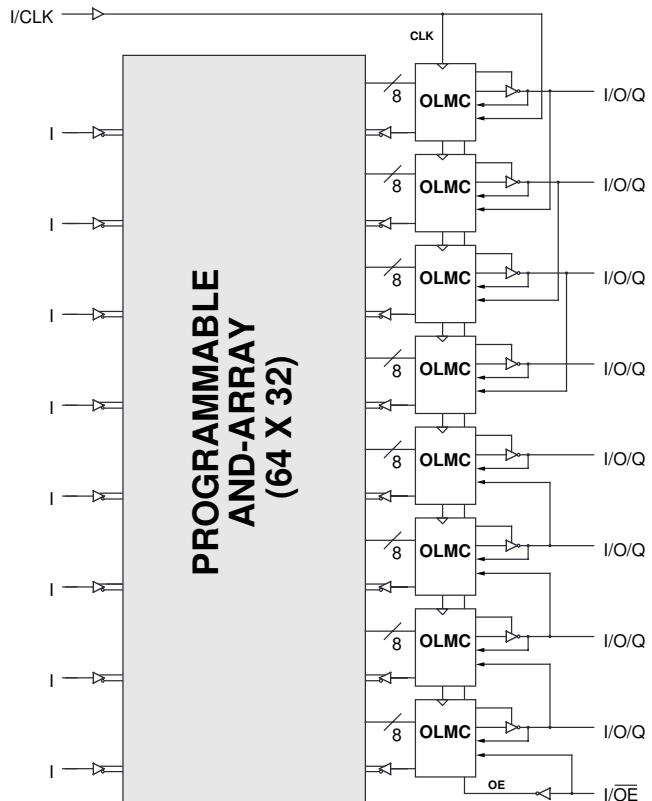
A.C. TESTING LOAD CIRCUIT



Features

- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
 - 3.5 ns Maximum Propagation Delay
 - Fmax = 250 MHz
 - 3.0 ns Maximum from Clock Input to Data Output
 - UltraMOS® Advanced CMOS Technology
- 50% to 75% REDUCTION IN POWER FROM BIPOLAR
 - 75mA Typ Icc on Low Power Device
 - 45mA Typ Icc on Quarter Power Device
- ACTIVE PULL-UPS ON ALL PINS
- E² CELL TECHNOLOGY
 - Reconfigurable Logic
 - Reprogrammable Cells
 - 100% Tested/100% Yields
 - High Speed Electrical Erasure (<100ms)
 - 20 Year Data Retention
- EIGHT OUTPUT LOGIC MACROCELLS
 - Maximum Flexibility for Complex Logic Designs
 - Programmable Output Polarity
 - Also Emulates 20-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
 - 100% Functional Testability
- APPLICATIONS INCLUDE:
 - DMA Control
 - State Machine Control
 - High Speed Graphics Processing
 - Standard Logic Speed Upgrade
- ELECTRONIC SIGNATURE FOR IDENTIFICATION

Functional Block Diagram



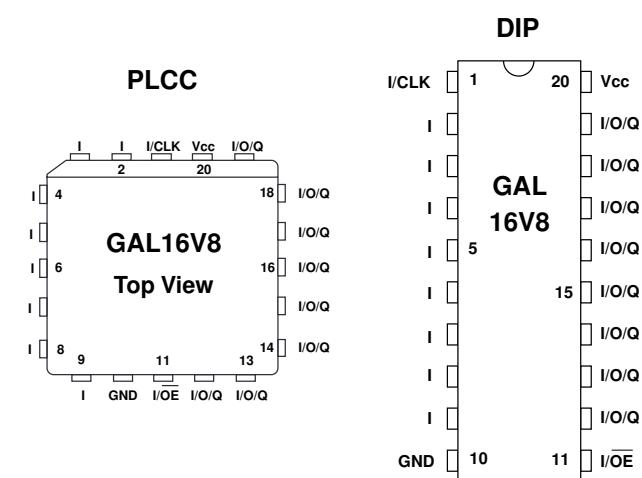
Description

The GAL16V8, at 3.5 ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL16V8 are the PAL architectures listed in the table of the macrocell description section. GAL16V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Pin Configuration





Specifications GAL16V8

GAL16V8 Ordering Information

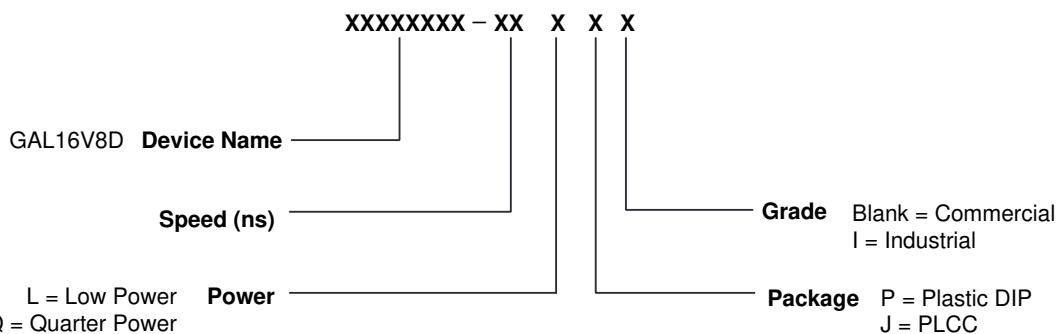
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
3.5	2.5	3.0	115	GAL16V8D-3LJ	20-Lead PLCC
5	3	4	115	GAL16V8D-5LJ	20-Lead PLCC
7.5	7	5	115	GAL16V8D-7LP	20-Pin Plastic DIP
			115	GAL16V8D-7LJ	20-Lead PLCC
10	10	7	55	GAL16V8D-10QP	20-Pin Plastic DIP
			55	GAL16V8D-10QJ	20-Lead PLCC
			115	GAL16V8D-10LP	20-Pin Plastic DIP
			115	GAL16V8D-10LJ	20-Lead PLCC
15	12	10	55	GAL16V8D-15QP	20-Pin Plastic DIP
			55	GAL16V8D-15QJ	20-Lead PLCC
			90	GAL16V8D-15LP	20-Pin Plastic DIP
			90	GAL16V8D-15LJ	20-Lead PLCC
25	15	12	55	GAL16V8D-25QP	20-Pin Plastic DIP
			55	GAL16V8D-25QJ	20-Lead PLCC
			90	GAL16V8D-25LP	20-Pin Plastic DIP
			90	GAL16V8D-25LJ	20-Lead PLCC

Industrial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
7.5	7	5	130	GAL16V8D-7LPI	20-Pin Plastic DIP
			130	GAL16V8D-7LJI	20-Lead PLCC
10	10	7	130	GAL16V8D-10LPI	20-Pin Plastic DIP
			130	GAL16V8D-10LJI	20-Lead PLCC
15	12	10	130	GAL16V8D-15LPI	20-Pin Plastic DIP
			130	GAL16V8D-15LJI	20-Lead PLCC
20	13	11	65	GAL16V8D-20QPI	20-Pin Plastic DIP
			65	GAL16V8D-20QJI	20-Lead PLCC
25	15	12	65	GAL16V8D-25QPI	20-Pin Plastic DIP
			65	GAL16V8D-25QJI	20-Lead PLCC
			130	GAL16V8D-25LPI	20-Pin Plastic DIP
			130	GAL16V8D-25LJI	20-Lead PLCC

Part Number Description



Output Logic Macrocell (OLMC)

The following discussion pertains to configuring the output logic macrocell. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

There are three global OLMC configuration modes possible: **simple**, **complex**, and **registered**. Details of each of these modes are illustrated in the following pages. Two global bits, SYN and AC0, control the mode configuration for all macrocells. The XOR bit of each macrocell controls the polarity of the output in any of the three modes, while the AC1 bit of each of the macrocells controls the input/output configuration. These two global and 16 individual architecture bits define all possible configurations in a GAL16V8. The information given on these architecture bits is only to give a better understanding of the device. Compiler software will transparently set these architecture bits from the pin definitions, so the user should not need to directly manipulate these architecture bits.

The following is a list of the PAL architectures that the GAL16V8 can emulate. It also shows the OLMC mode under which the GAL16V8 emulates the PAL architecture.

PAL Architectures Emulated by GAL16V8	GAL16V8 Global OLMC Mode
16R8	Registered
16R6	Registered
16R4	Registered
16RP8	Registered
16RP6	Registered
16RP4	Registered
16L8	Complex
16H8	Complex
16P8	Complex
10L8	Simple
12L6	Simple
14L4	Simple
16L2	Simple
10H8	Simple
12H6	Simple
14H4	Simple
16H2	Simple
10P8	Simple
12P6	Simple
14P4	Simple
16P2	Simple

Compiler Support for OLMC

Software compilers support the three different global OLMC modes as different device types. These device types are listed in the table below. Most compilers have the ability to automatically select the device type, generally based on the register usage and output enable (OE) usage. Register usage on the device forces the software to choose the registered mode. All combinatorial outputs with OE controlled by the product term will force the software to choose the complex mode. The software will choose the simple mode only when all outputs are dedicated combinatorial without OE control. The different device types listed in the table can be used to override the automatic device selection by the software. For further details, refer to the compiler software manuals.

When using compiler software to configure the device, the user must pay special attention to the following restrictions in each mode. In **registered mode** pin 1 and pin 11 are permanently configured

as clock and output enable, respectively. These pins cannot be configured as dedicated inputs in the registered mode.

In **complex mode** pin 1 and pin 11 become dedicated inputs and use the feedback paths of pin 19 and pin 12 respectively. Because of this feedback path usage, pin 19 and pin 12 do not have the feedback option in this mode.

In **simple mode** all feedback paths of the output pins are routed via the adjacent pins. In doing so, the two inner most pins (pins 15 and 16) will not have the feedback option as these pins are always configured as dedicated combinatorial output.

	Registered	Complex	Simple	Auto Mode Select
ABEL	P16V8R	P16V8C	P16V8AS	P16V8
CUPL	G16V8MS	G16V8MA	G16V8AS	G16V8
LOG/iC	GAL16V8_R	GAL16V8_C7	GAL16V8_C8	GAL16V8
OrCAD-PLD	"Registered" ¹	"Complex" ¹	"Simple" ¹	GAL16V8A
PLDesigner	P16V8R ²	P16V8C ²	P16V8C ²	P16V8A
TANGO-PLD	G16V8R	G16V8C	G16V8AS ³	G16V8

1) Used with **Configuration** keyword.

2) Prior to Version 2.0 support.

3) Supported on Version 1.20 or later.

Registered Mode

In the Registered mode, macrocells are configured as dedicated registered outputs or as I/O functions.

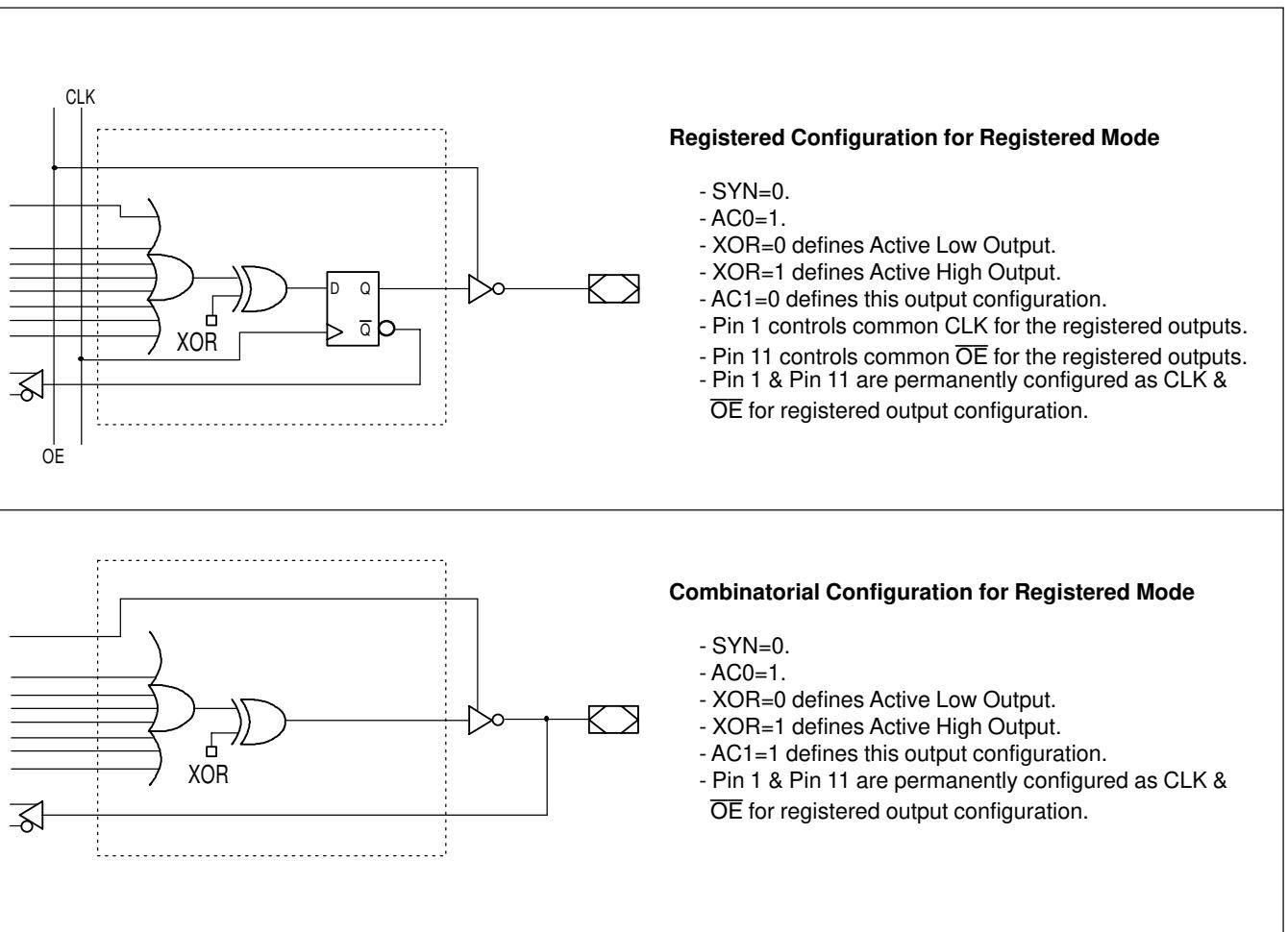
Architecture configurations available in this mode are similar to the common 16R8 and 16RP4 devices with various permutations of polarity, I/O and register placement.

All registered macrocells share common clock and output enable control pins. Any macrocell can be configured as registered or I/O. Up to eight registers or up to eight I/O's are possible in this mode.

Dedicated input or output functions can be implemented as subsets of the I/O function.

Registered outputs have eight product terms per output. I/O's have seven product terms per output.

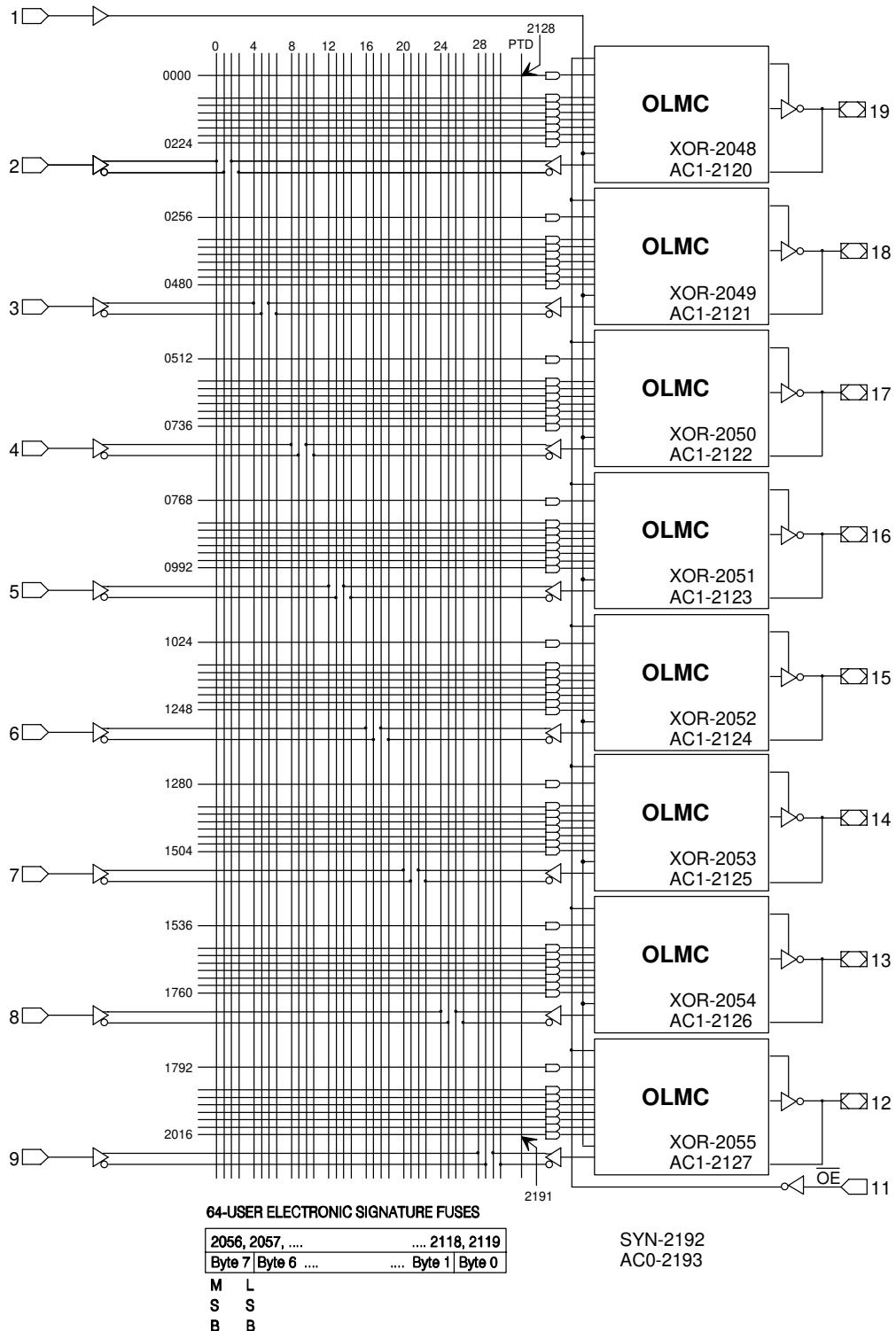
The JEDEC fuse numbers, including the User Electronic Signature (UES) fuses and the Product Term Disable (PTD) fuses, are shown on the logic diagram on the following page.



Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Registered Mode Logic Diagram

DIP & PLCC Package Pinouts



Complex Mode

In the Complex mode, macrocells are configured as output only or I/O functions.

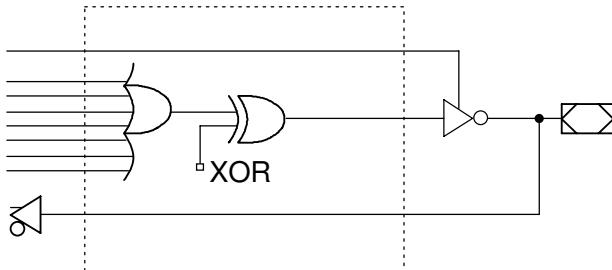
Architecture configurations available in this mode are similar to the common 16L8 and 16P8 devices with programmable polarity in each macrocell.

Up to six I/O's are possible in this mode. Dedicated inputs or outputs can be implemented as subsets of the I/O function. The two outer most macrocells (pins 12 & 19) do not have input capa-

bility. Designs requiring eight I/O's can be implemented in the Registered mode.

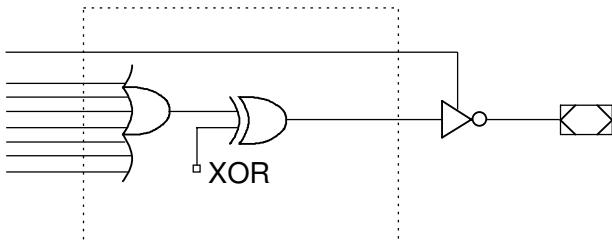
All macrocells have seven product terms per output. One product term is used for programmable output enable control. Pins 1 and 11 are always available as data inputs into the AND array.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram on the following page.



Combinatorial I/O Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 13 through Pin 18 are configured to this function.



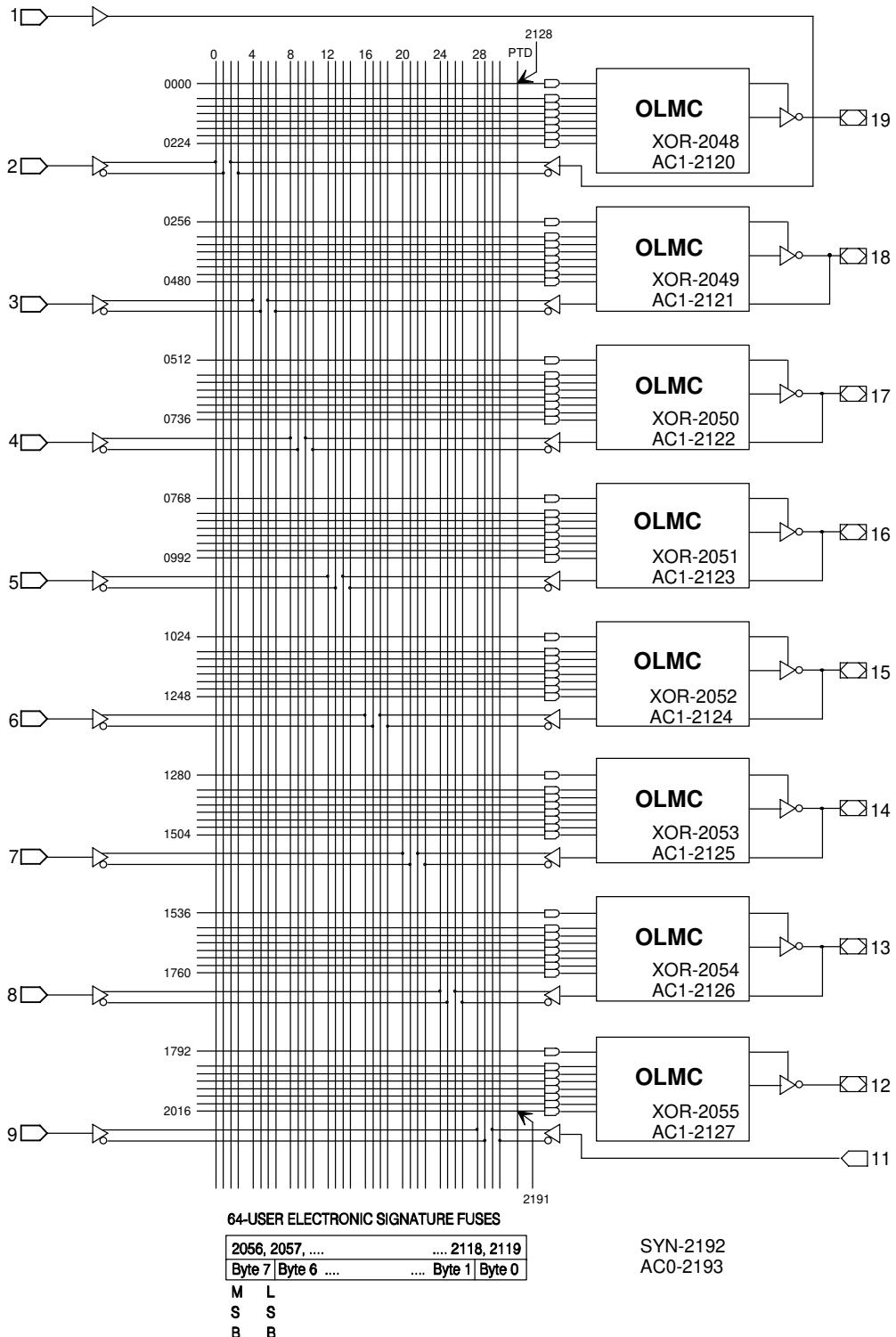
Combinatorial Output Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 12 and Pin 19 are configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Complex Mode Logic Diagram

DIP & PLCC Package Pinouts



Simple Mode

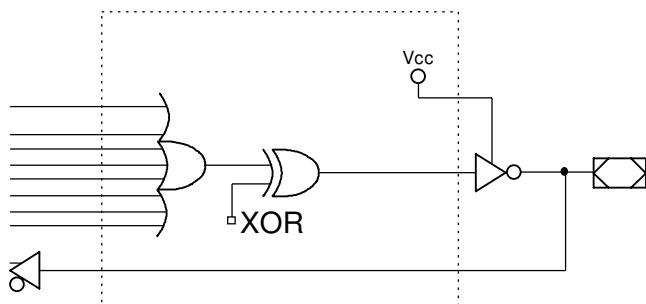
In the Simple mode, macrocells are configured as dedicated inputs or as dedicated, always active, combinatorial outputs.

Architecture configurations available in this mode are similar to the common 10L8 and 12P6 devices with many permutations of generic output polarity or input choices.

All outputs in the simple mode have a maximum of eight product terms that can control the logic. In addition, each output has programmable polarity.

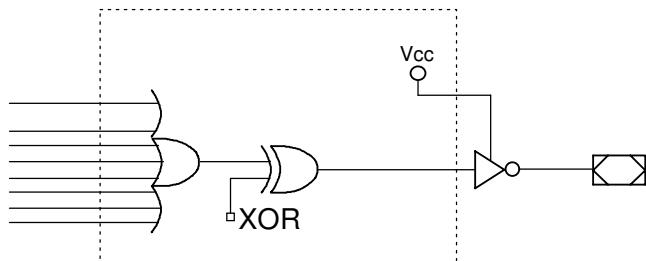
Pins 1 and 11 are always available as data inputs into the AND array. The center two macrocells (pins 15 & 16) cannot be used as input or I/O pins, and are only available as dedicated outputs.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram.



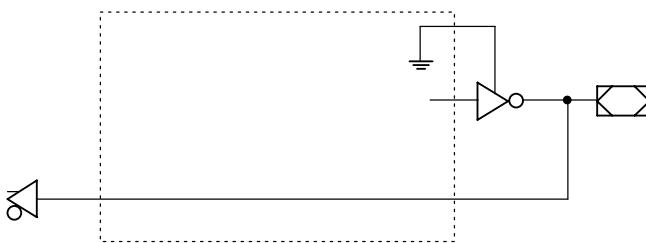
Combinatorial Output with Feedback Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.



Combinatorial Output Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- Pins 15 & 16 are permanently configured to this function.



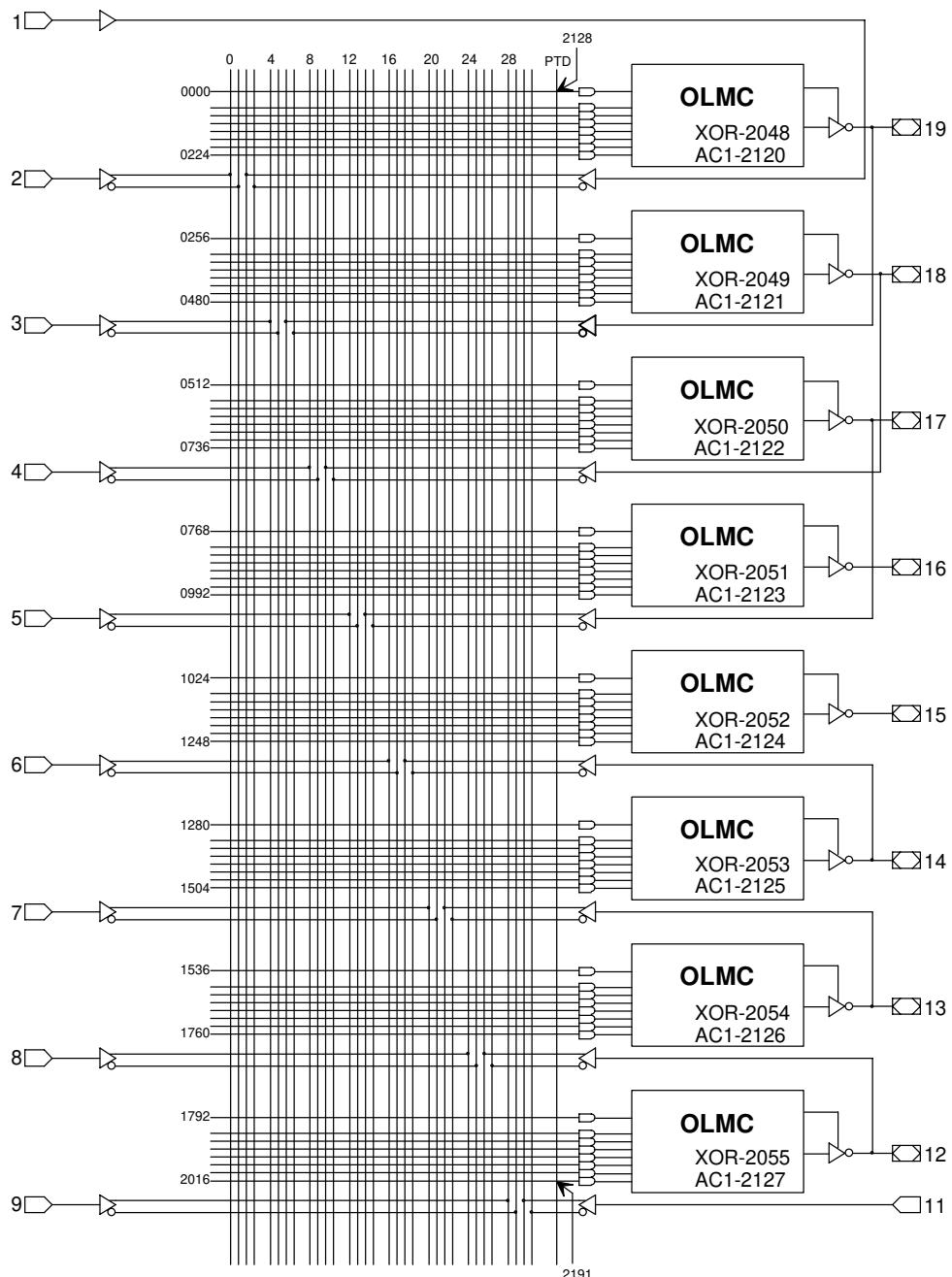
Dedicated Input Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Simple Mode Logic Diagram

DIP & PLCC Package Pinouts



64-USER ELECTRONIC SIGNATURE FUSES

2056, 2057, 2118, 2119
Byte 7 Byte 6 Byte 1 Byte 0

SYN-2192
AC0-2193

M L
S S
B B

Absolute Maximum Ratings⁽¹⁾

Supply voltage V_{CC}	-0.5 to +7V
Input voltage applied	-2.5 to V_{CC} +1.0V
Off-state output voltage applied	-2.5 to V_{CC} +1.0V
Storage Temperature	-65 to 150°C
Ambient Temperature with Power Applied	-55 to 125°C
1. Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied (while programming, follow the programming specifications).	

Recommended Operating Conditions

Commercial Devices:

Ambient Temperature (T_A)	0 to 75°C
Supply voltage (V_{CC}) with Respect to Ground	+4.75 to +5.25V

Industrial Devices:

Ambient Temperature (T_A)	-40 to 85°C
Supply voltage (V_{CC}) with Respect to Ground	+4.50 to +5.50V

DC Electrical Characteristics

Over Recommended Operating Conditions (Unless Otherwise Specified)

SYMBOL	PARAMETER	CONDITION	MIN.	TYP. ³	MAX.	UNITS
V_{IL}	Input Low Voltage		$V_{SS} - 0.5$	—	0.8	V
V_{IH}	Input High Voltage		2.0	—	$V_{CC} + 1$	V
I_{IL}^1	Input or I/O Low Leakage Current	$0V \leq V_{IN} \leq V_{IL}$ (MAX.)	—	—	-100	μA
I_{IH}	Input or I/O High Leakage Current	$3.5V \leq V_{IN} \leq V_{CC}$	—	—	10	μA
V_{OL}	Output Low Voltage	$I_{OL} = MAX.$ $V_{in} = V_{IL}$ or V_{IH}	—	—	0.5	V
V_{OH}	Output High Voltage	$I_{OH} = MAX.$ $V_{in} = V_{IL}$ or V_{IH}	2.4	—	—	V
I_{OL}	Low Level Output Current	L-3/-5 & -7 (Ind. PLCC)	—	—	16	mA
		L-7 (Except Ind. PLCC)/-10/-15/-25 Q-10/-15/-20/-25	—	—	24	mA
I_{OH}	High Level Output Current		—	—	-3.2	mA
I_{OS}^2	Output Short Circuit Current	$V_{CC} = 5V$ $V_{OUT} = 0.5V$ $T_A = 25^\circ C$	-30	—	-150	mA

COMMERCIAL

I_{CC}	Operating Power Supply Current	$V_{IL} = 0.5V$ $V_{IH} = 3.0V$ $f_{toggle} = 15MHz$ Outputs Open	L-3/-5/-7/-10	—	75	115	mA
			L-15/-25	—	75	90	mA
			Q-10/-15/-25	—	45	55	mA

INDUSTRIAL

I_{CC}	Operating Power Supply Current	$V_{IL} = 0.5V$ $V_{IH} = 3.0V$ $f_{toggle} = 15MHz$ Outputs Open	L-7/-10/-15/-25	—	75	130	mA
			Q-20/-25	—	45	65	mA

1) The leakage current is due to the internal pull-up resistor on all pins. See **Input Buffer** section for more information.

2) One output at a time for a maximum duration of one second. $V_{out} = 0.5V$ was selected to avoid test problems caused by tester ground degradation. Characterized but not 100% tested.

3) Typical values are at $V_{CC} = 5V$ and $T_A = 25^\circ C$

AC Switching Characteristics

Over Recommended Operating Conditions

PARAMETER	TEST COND ¹ .	DESCRIPTION	COM		COM		COM / IND		UNITS	
			-3		-5		-7			
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.		
t_{pd}	A	Input or I/O to Comb. Output	1	3.5	1	5	1	7.5	ns	
t_{co}	A	Clock to Output Delay	1	3	1	4	1	5	ns	
t_{cf}²	—	Clock to Feedback Delay	—	2.5	—	3	—	3	ns	
t_{su}	—	Setup Time, Input or Feedback before Clock↑	2.5	—	3	—	7	—	ns	
t_h	—	Hold Time, Input or Feedback after Clock↑	0	—	0	—	0	—	ns	
f_{max}³	A	Maximum Clock Frequency with External Feedback, 1/(t _{su} + t _{co})	182	—	142.8	—	83.3	—	MHz	
	A	Maximum Clock Frequency with Internal Feedback, 1/(t _{su} + t _{cf})	200	—	166	—	100	—	MHz	
	A	Maximum Clock Frequency with No Feedback	250	—	166	—	100	—	MHz	
t_{wh}	—	Clock Pulse Duration, High	2 ⁴	—	3 ⁴	—	5	—	ns	
t_{wl}	—	Clock Pulse Duration, Low	2 ⁴	—	3 ⁴	—	5	—	ns	
t_{en}	B	Input or I/O to Output Enabled	—	4.5	—	6	—	9	ns	
	B	OE to Output Enabled	—	4.5	—	6	—	6	ns	
t_{dis}	C	Input or I/O to Output Disabled	—	4.5	—	5	—	9	ns	
	C	OE to Output Disabled	—	4.5	—	5	—	6	ns	

 1) Refer to **Switching Test Conditions** section.

 2) Calculated from f_{max} with internal feedback. Refer to **f_{max} Descriptions** section.

 3) Refer to **f_{max} Descriptions** section. Characterized but not 100% tested.

4) Characterized but not 100% tested.

Capacitance (T_A = 25°C, f = 1.0 MHz)

SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C _I	Input Capacitance	8	pF	V _{CC} = 5.0V, V _I = 2.0V
C _{I/O}	I/O Capacitance	8	pF	V _{CC} = 5.0V, V _{I/O} = 2.0V

*Characterized but not 100% tested.

AC Switching Characteristics

Over Recommended Operating Conditions

PARAM.	TEST COND ¹ .	DESCRIPTION	COM / IND		COM / IND		IND		COM / IND		UNITS	
			-10		-15		-20		-25			
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.		
t_{pd}	A	Input or I/O to Comb. Output	3	10	3	15	3	20	3	25	ns	
t_{co}	A	Clock to Output Delay	2	7	2	10	2	11	2	12	ns	
t_{cf}²	—	Clock to Feedback Delay	—	6	—	8	—	9	—	10	ns	
t_{su}	—	Setup Time, Input or Fdbk before Clk↑	10	—	12	—	13	—	15	—	ns	
t_h	—	Hold Time, Input or Fdbk after Clk↑	0	—	0	—	0	—	0	—	ns	
f_{max}³	A	Maximum Clock Frequency with External Feedback, $1/(t_{su} + t_{co})$	58.8	—	45.5	—	41.6	—	37	—	MHz	
	A	Maximum Clock Frequency with Internal Feedback, $1/(t_{su} + t_{cf})$	62.5	—	50	—	45.4	—	40	—	MHz	
	A	Maximum Clock Frequency with No Feedback	62.5	—	62.5	—	50	—	41.6	—	MHz	
t_{wh}	—	Clock Pulse Duration, High	8	—	8	—	10	—	12	—	ns	
t_{wl}	—	Clock Pulse Duration, Low	8	—	8	—	10	—	12	—	ns	
t_{en}	B	Input or I/O to Output Enabled	—	10	—	15	—	18	—	20	ns	
	B	OE to Output Enabled	—	10	—	15	—	18	—	20	ns	
t_{dis}	C	Input or I/O to Output Disabled	—	10	—	15	—	18	—	20	ns	
	C	OE to Output Disabled	—	10	—	15	—	18	—	20	ns	

 1) Refer to **Switching Test Conditions** section.

 2) Calculated from fmax with internal feedback. Refer to **fmax Descriptions** section.

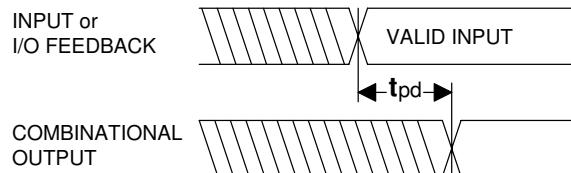
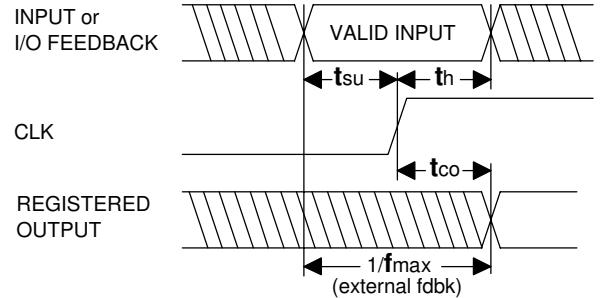
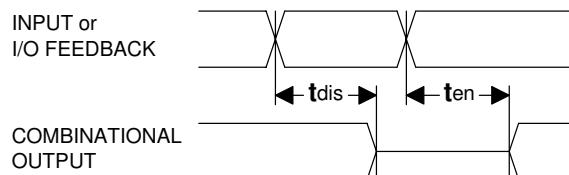
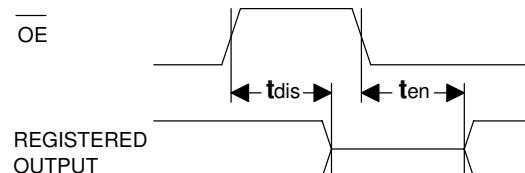
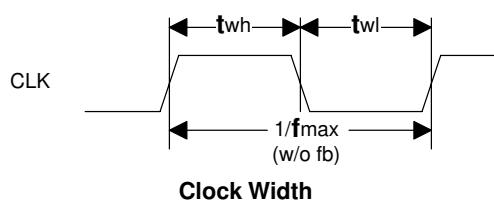
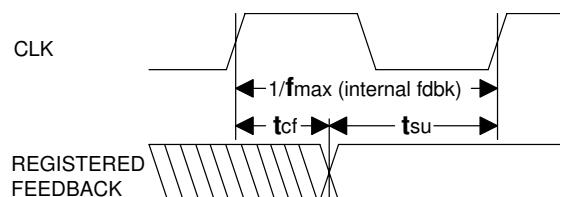
 3) Refer to **fmax Descriptions** section. Characterized but not 100% tested.

Capacitance ($T_A = 25^\circ\text{C}$, $f = 1.0 \text{ MHz}$)

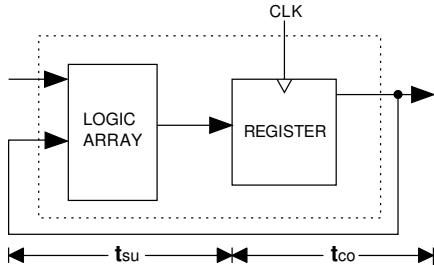
SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	8	pF	$V_{CC} = 5.0\text{V}$, $V_I = 2.0\text{V}$
$C_{I/O}$	I/O Capacitance	8	pF	$V_{CC} = 5.0\text{V}$, $V_{I/O} = 2.0\text{V}$

*Characterized but not 100% tested.

Switching Waveforms

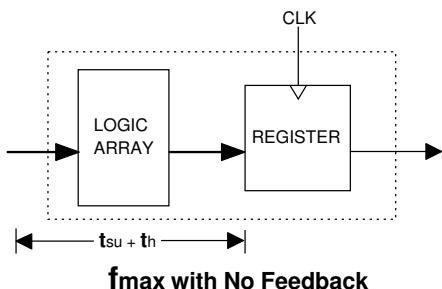
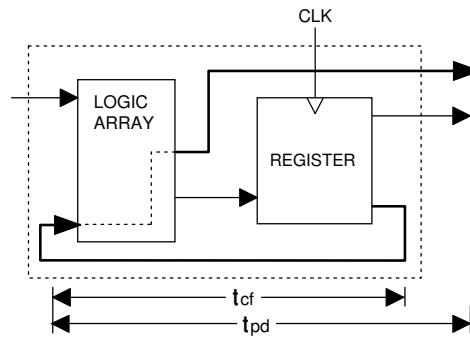

Combinatorial Output

Registered Output

Input or I/O to Output Enable/Disable

 \overline{OE} to Output Enable/Disable

Clock Width

 f_{max} with Feedback

fmax Descriptions



fmax with External Feedback $1/(t_{su}+t_{co})$

Note: fmax with external feedback is calculated from measured t_{su} and t_{co}.



fmax with No Feedback

Note: fmax with no feedback may be less than $1/(t_{wh} + t_{wl})$. This is to allow for a clock duty cycle of other than 50%.

fmax with Internal Feedback $1/(t_{su}+t_{cf})$

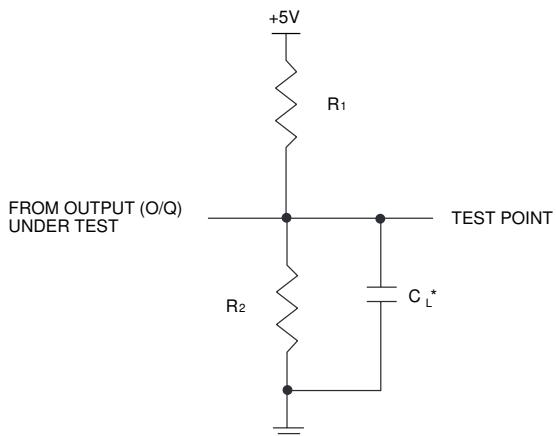
Note: t_{cf} is a calculated value, derived by subtracting t_{su} from the period of fmax w/internal feedback ($t_{cf} = 1/f_{max} - t_{su}$). The value of t_{cf} is used primarily when calculating the delay from clocking a register to a combinatorial output (through registered feedback), as shown above. For example, the timing from clock to a combinatorial output is equal to t_{cf} + t_{pd}.

Switching Test Conditions

Input Pulse Levels		GND to 3.0V
Input Rise and Fall Times	GAL16V8D-10 (and slower)	2 – 3ns 10% – 90%
	GAL16V8D-3/-5/-7	1.5ns 10% – 90%
Input Timing Reference Levels		1.5V
Output Timing Reference Levels		1.5V
Output Load		See figure at right

3-state levels are measured 0.5V from steady-state active level.

Table 2-0003/16V8



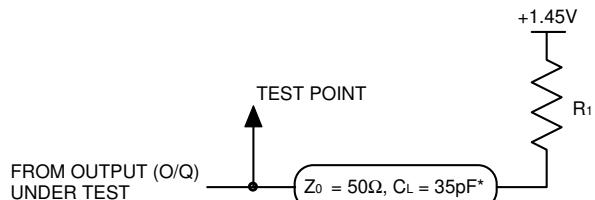
*C_L INCLUDES TEST FIXTURE AND PROBE CAPACITANCE

Test Condition	R ₁	R ₂	C _L
A	200Ω	390Ω	50pF
B	Active High	∞	390Ω
	Active Low	200Ω	390Ω
C	Active High	∞	5pF
	Active Low	200Ω	5pF

Switching Test Conditions (Continued)

GAL16V8D-3 Output Load Conditions (see figure at right)

Test Condition	R ₁	C _L
A	50Ω	35pF
B	High Z to Active High at 1.9V	50Ω
	High Z to Active Low at 1.0V	35pF
C	Active High to High Z at 1.9V	50Ω
	Active Low to High Z at 1.0V	35pF



*C_L includes test fixture and probe capacitance.

Electronic Signature

An electronic signature is provided in every GAL16V8 device. It contains 64 bits of reprogrammable memory that can contain user defined data. Some uses include user ID codes, revision numbers, or inventory control. The signature data is always available to the user independent of the state of the security cell.

NOTE: The electronic signature is included in checksum calculations. Changing the electronic signature will alter the checksum.

Security Cell

A security cell is provided in the GAL16V8 devices to prevent unauthorized copying of the array patterns. Once programmed, this cell prevents further read access to the functional bits in the device. This cell can only be erased by re-programming the device, so the original configuration can never be examined once this cell is programmed. The Electronic Signature is always available to the user, regardless of the state of this control cell.

Latch-Up Protection

GAL16V8 devices are designed with an on-board charge pump to negatively bias the substrate. The negative bias minimizes the potential of latch-up caused by negative input undershoots. Additionally, outputs are designed with n-channel pull-ups instead of the traditional p-channel pull-ups in order to eliminate latch-up due to output overshoots.

Device Programming

GAL devices are programmed using a Lattice Semiconductor approved Logic Programmer, available from a number of manufacturers. Complete programming of the device takes only a few seconds. Erasing of the device is transparent to the user, and is done automatically as part of the programming cycle.

Output Register Preload

When testing state machine designs, all possible states and state transitions must be verified in the design, not just those required in the normal machine operations. This is because, in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper treatment of these conditions, a way must be provided to break the feedback paths, and force any desired (i.e., illegal) state into the registers. Then the machine can be sequenced and the outputs tested for correct next state conditions.

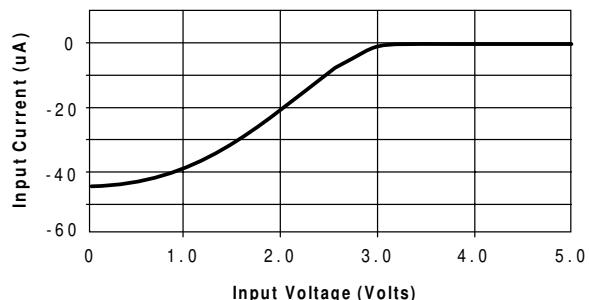
GAL16V8 devices include circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing text vectors perform output register preload automatically.

Input Buffers

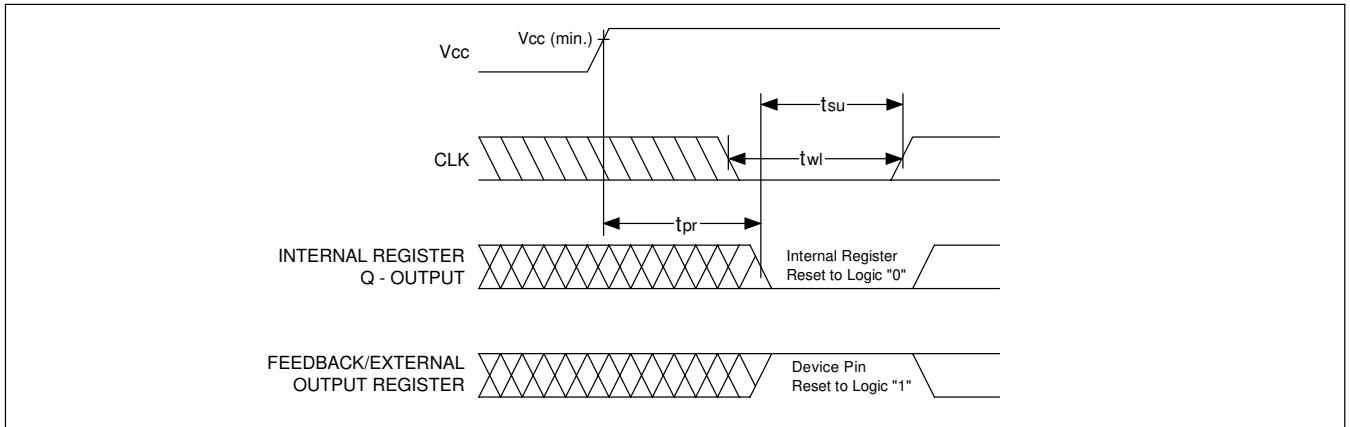
GAL16V8 devices are designed with TTL level compatible input buffers. These buffers have a characteristically high impedance, and present a much lighter load to the driving logic than bipolar TTL devices.

The GAL16V8 input and I/O pins have built-in active pull-ups. As a result, unused inputs and I/O's will float to a TTL "high" (logical "1"). Lattice Semiconductor recommends that all unused inputs and tri-stated I/O pins be connected to another active input, V_{CC}, or Ground. Doing this will tend to improve noise immunity and reduce I_{CC} for the device.

Typical Input Pull-up Characteristic



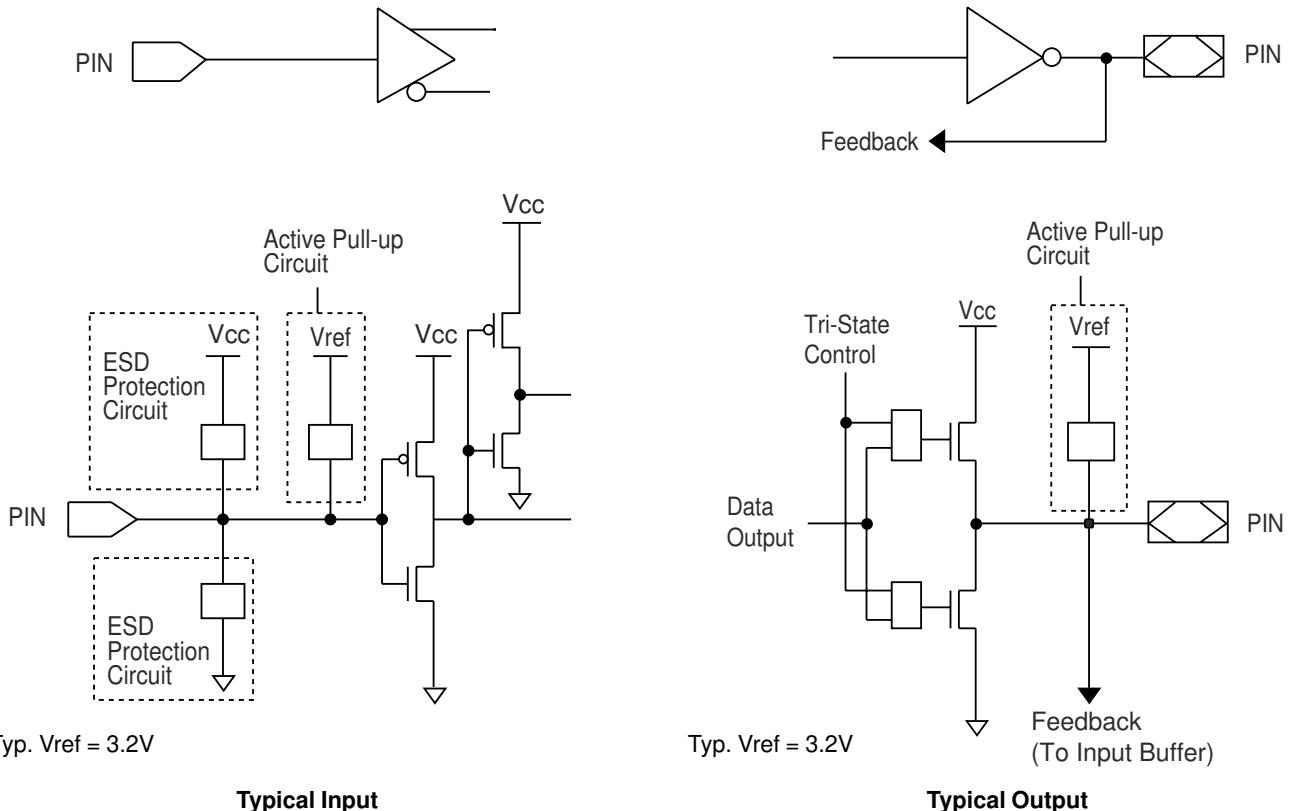
Power-Up Reset

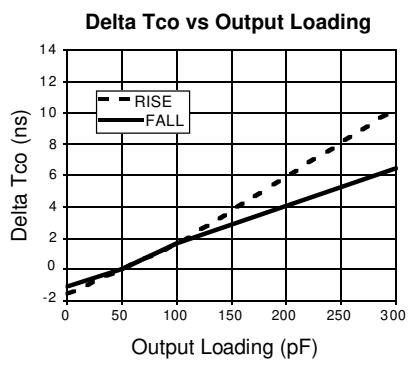
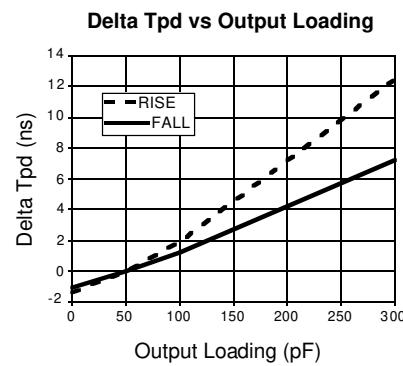
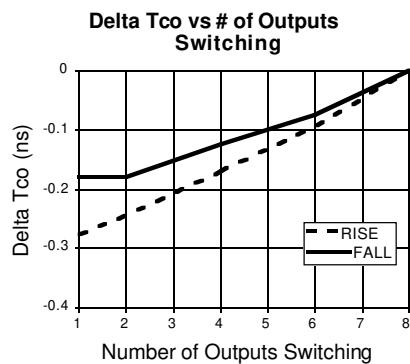
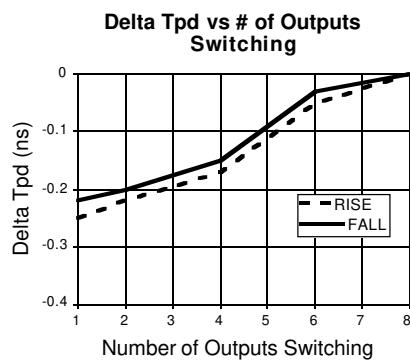
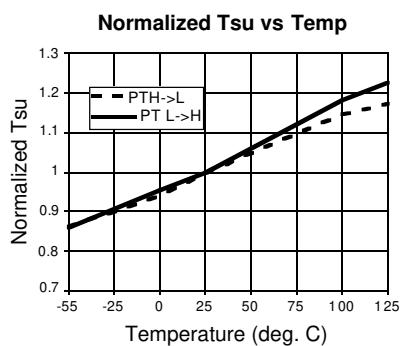
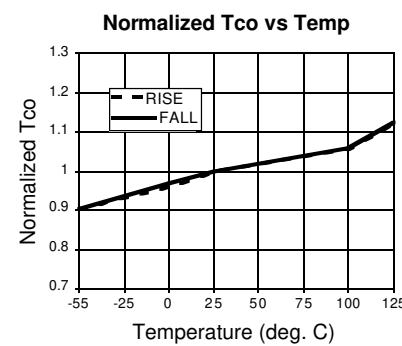
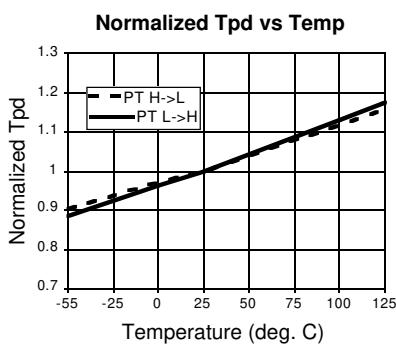
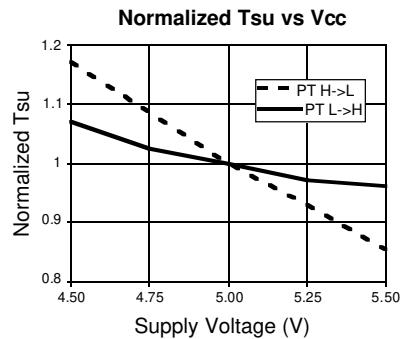
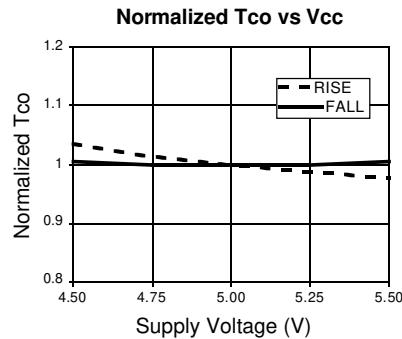
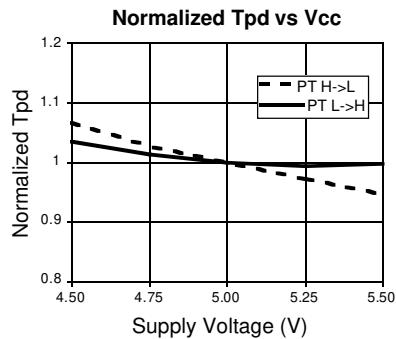


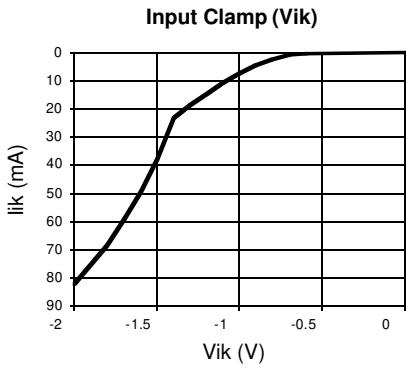
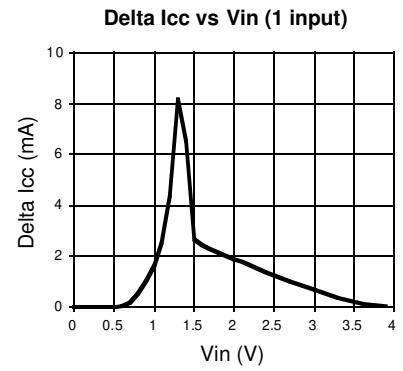
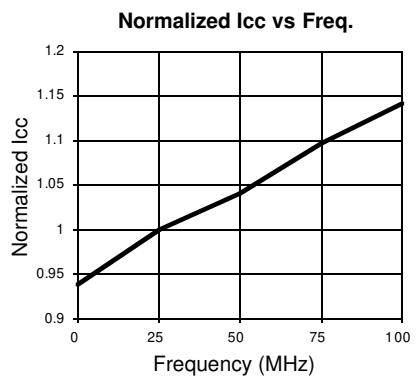
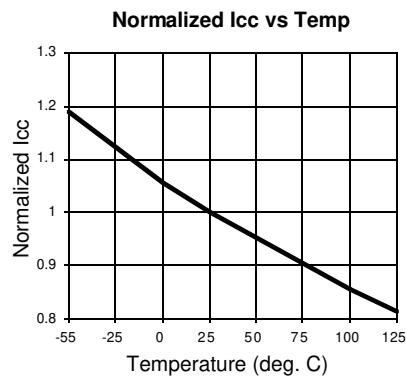
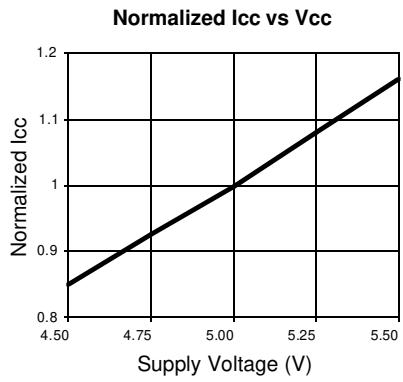
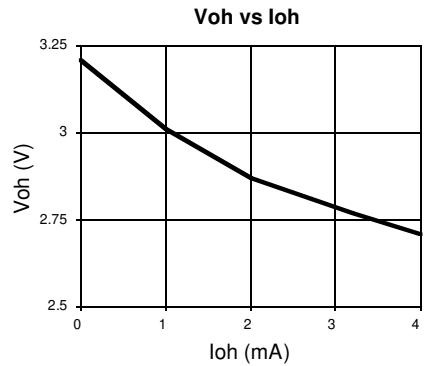
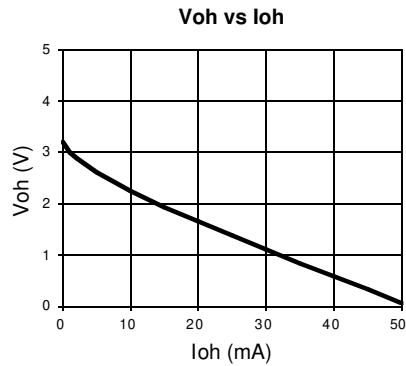
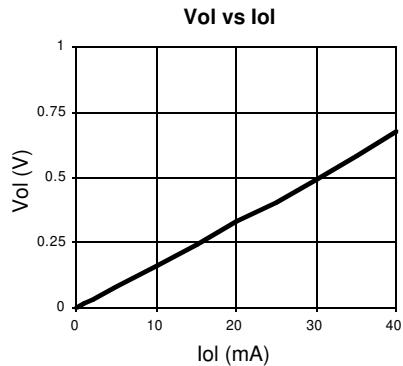
Circuitry within the GAL16V8 provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{pr} , 1 μ s MAX). As a result, the state on the registered output pins (if they are enabled) will always be high on power-up, regardless of the programmed polarity of the output pins. This feature can greatly simplify state machine design by providing a known state on power-up. Because of the asynchronous nature of system power-up, some

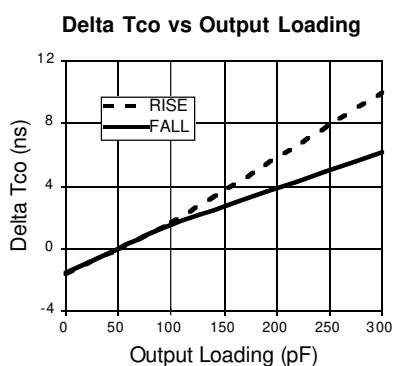
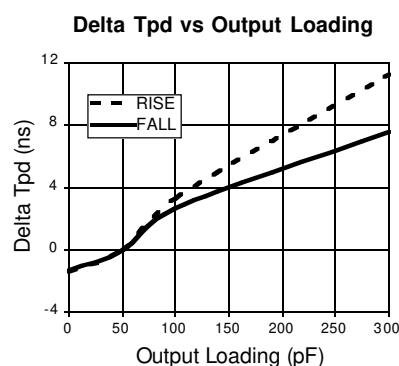
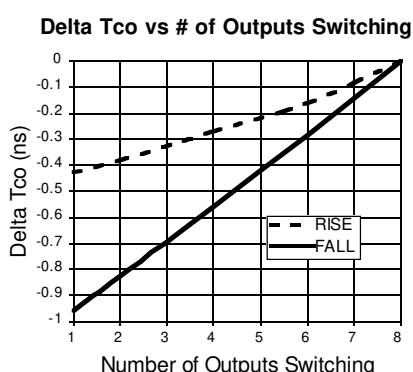
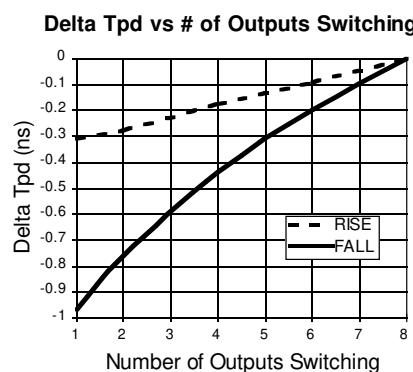
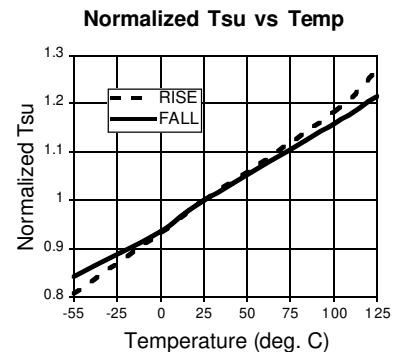
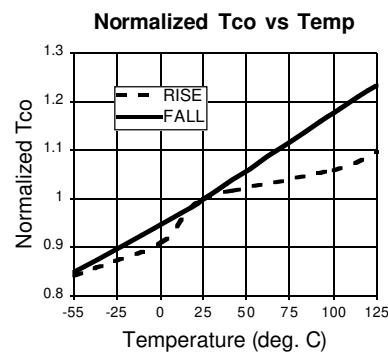
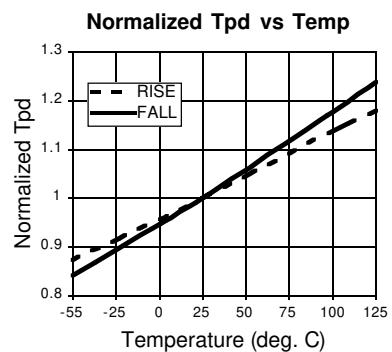
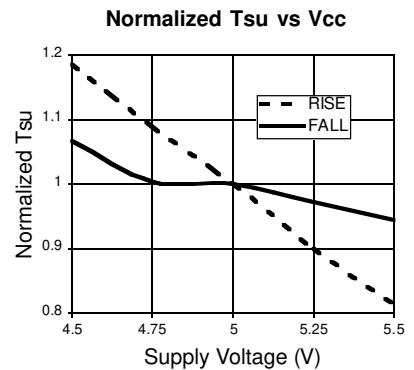
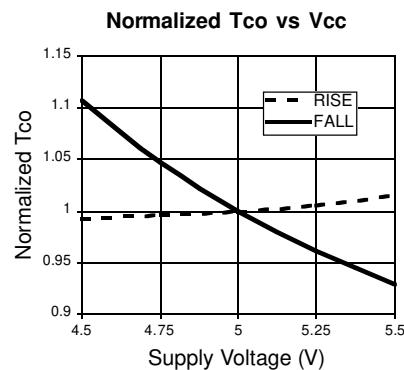
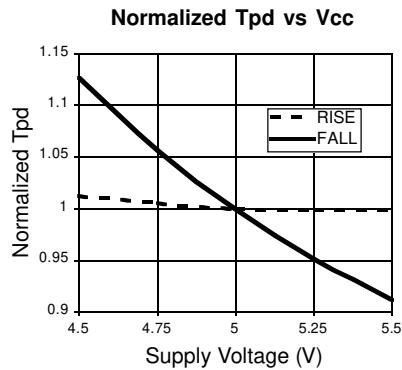
conditions must be met to provide a valid power-up reset of the device. First, the Vcc rise must be monotonic. Second, the clock input must be at static TTL level as shown in the diagram during power up. The registers will reset within a maximum of t_{pr} time. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met. The clock must also meet the minimum pulse width requirements.

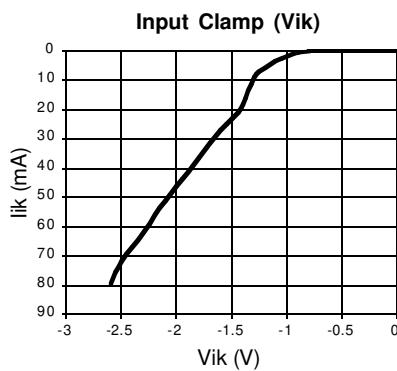
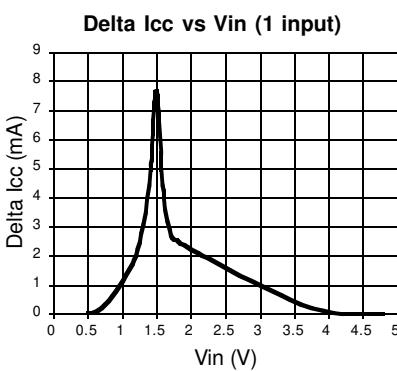
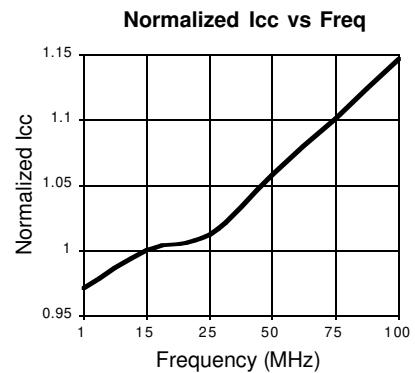
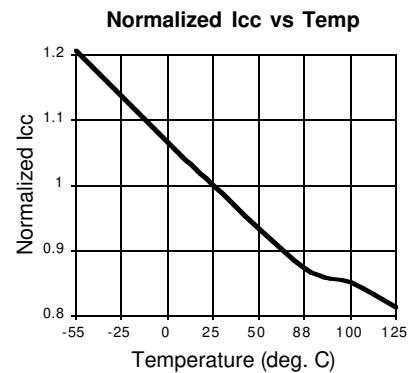
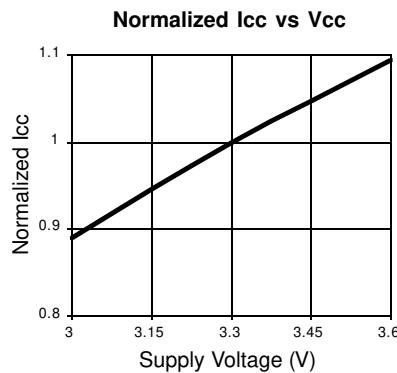
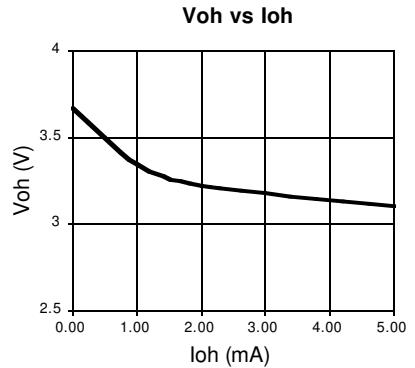
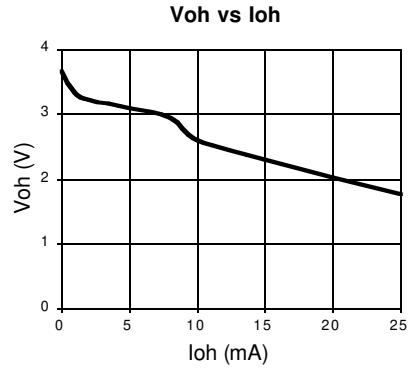
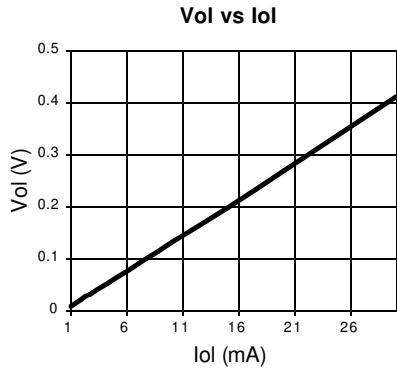
Input/Output Equivalent Schematics

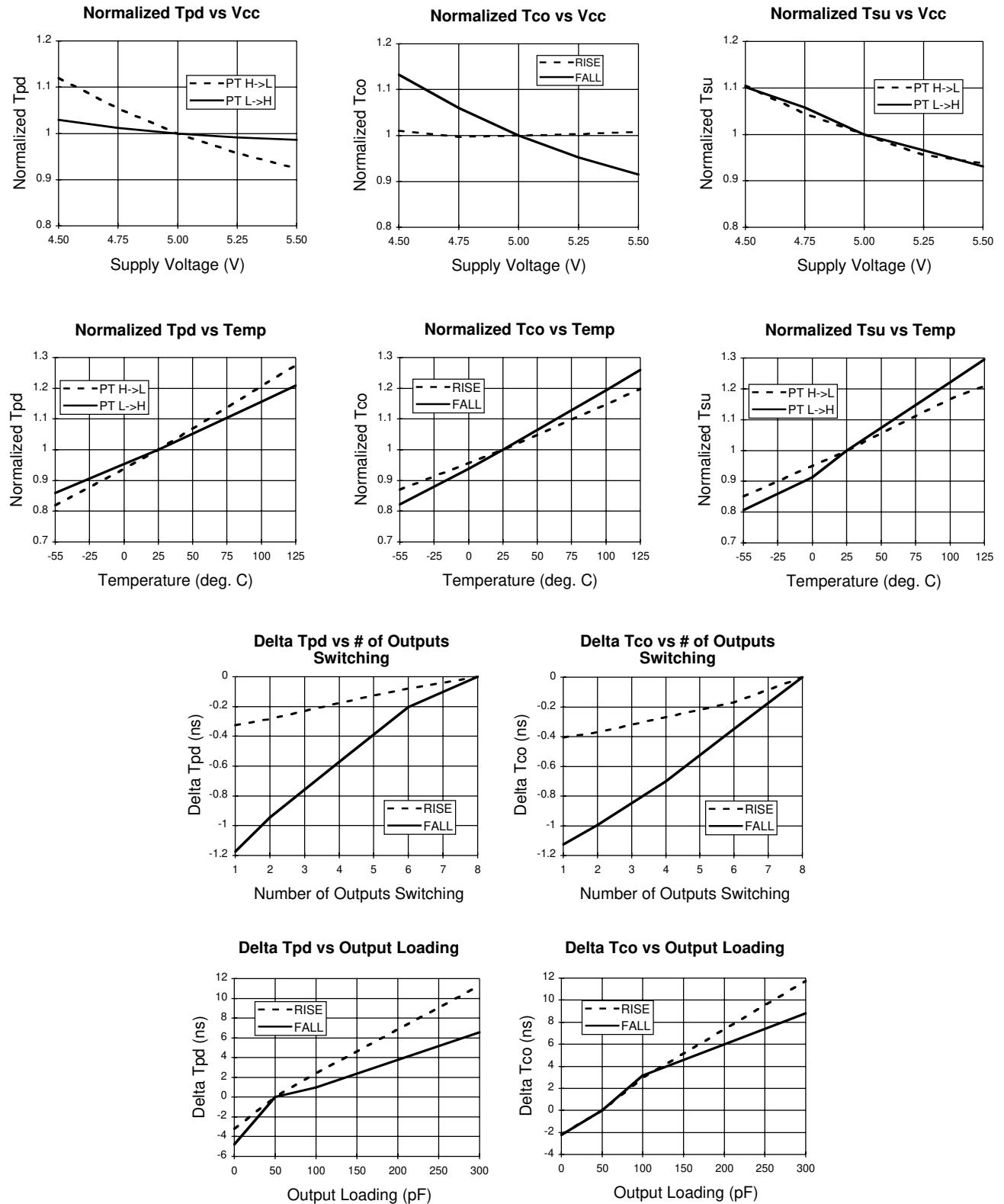


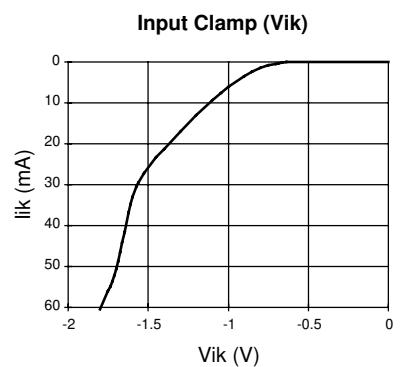
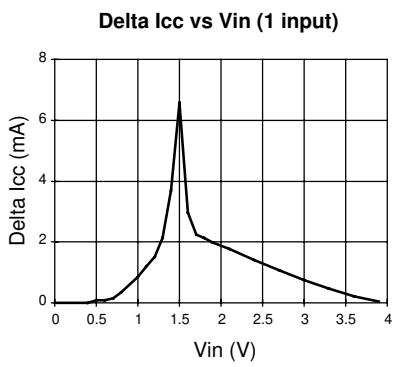
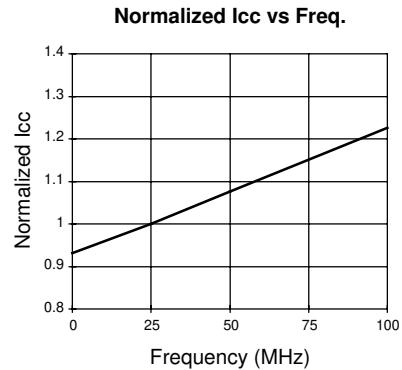
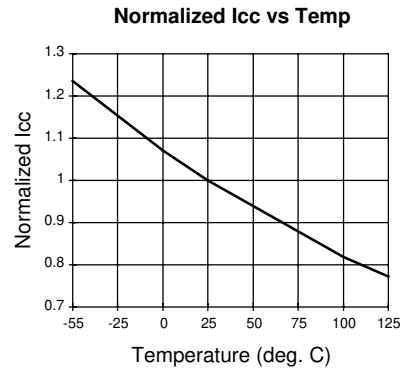
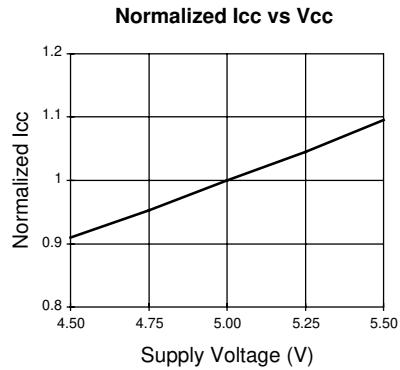
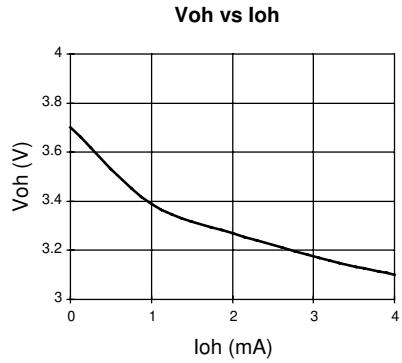
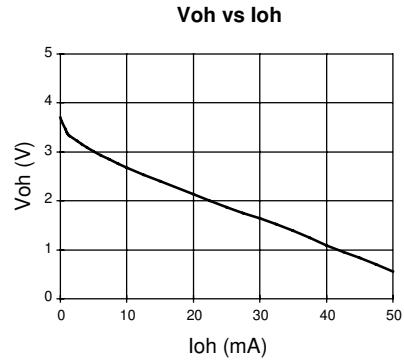
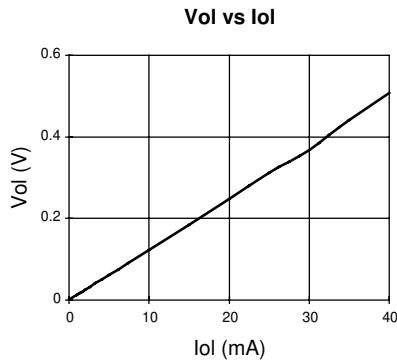
GAL16V8D-3-5/-7 (IND PLCC): Typical AC and DC Characteristic Diagrams


GAL16V8D-3-5/-7 (IND PLCC): Typical AC and DC Characteristic Diagrams


GAL16V8D-7 (Except IND PLCC)/-10L: Typical AC and DC Characteristic Diagrams


GAL16V8D-7 (Except IND PLCC)/-10L: Typical AC and DC Characteristic Diagrams


GAL16V8D-10Q (and Slower): Typical AC and DC Characteristic Diagrams


GAL16V8D-10Q (and Slower): Typical AC and DC Characteristic Diagrams




January 1988

MM54HC00/MM74HC00 Quad 2-Input NAND Gate

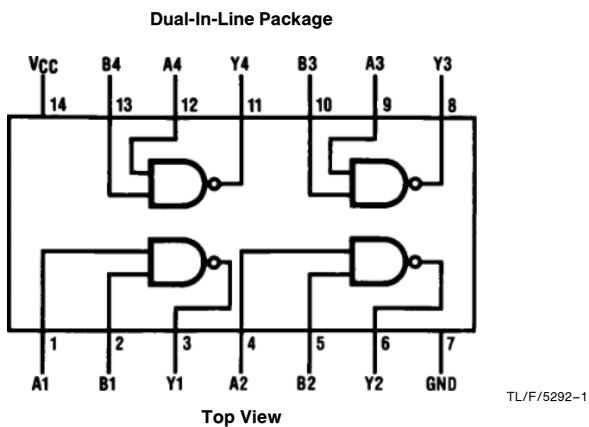
General Description

These NAND gates utilize advanced silicon-gate CMOS technology to achieve operating speeds similar to LS-TTL gates with the low power consumption of standard CMOS integrated circuits. All gates have buffered outputs. All devices have high noise immunity and the ability to drive 10 LS-TTL loads. The 54HC/74HC logic family is functionally as well as pin-out compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

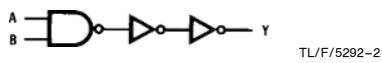
Features

- Typical propagation delay: 8 ns
- Wide power supply range: 2–6V
- Low quiescent current: 20 μ A maximum (74HC Series)
- Low input current: 1 μ A maximum
- Fanout of 10 LS-TTL loads

Connection and Logic Diagrams



Order Number MM54HC00 or MM74HC00



Absolute Maximum Ratings (Notes 1 & 2)		Operating Conditions							
If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.									
Supply Voltage (V_{CC})	-0.5 to +7.0V	Supply Voltage (V_{CC})	2	6	V				
DC Input Voltage (V_{IN})	-1.5 to V_{CC} + 1.5V	DC Input or Output Voltage (V_{IN} , V_{OUT})	0	V_{CC}	V				
DC Output Voltage (V_{OUT})	-0.5 to V_{CC} + 0.5V	Operating Temp. Range (T_A)							
Clamp Diode Current (I_{IK} , I_{OK})	± 20 mA	MM74HC	-40	+85	$^{\circ}$ C				
DC Output Current, per pin (I_{OUT})	± 25 mA	MM54HC	-55	+125	$^{\circ}$ C				
DC V_{CC} or GND Current, per pin (I_{CC})	± 50 mA	Input Rise or Fall Times							
Storage Temperature Range (T_{STG})	-65 $^{\circ}$ C to +150 $^{\circ}$ C	(t_r , t_f)	$V_{CC} = 2$ V	1000	ns				
Power Dissipation (P_D)			$V_{CC} = 4.5$ V	500	ns				
(Note 3) S.O. Package only	600 mW 500 mW		$V_{CC} = 6.0$ V	400	ns				
Lead Temperature (T_L) (Soldering 10 seconds)	260 $^{\circ}$ C								
DC Electrical Characteristics (Note 4)									
Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^{\circ}$ C		$74HC$	$54HC$	Units	
				Typ		$T_A = -40$ to 85° C	$T_A = -55$ to 125° C		
V_{IH}	Minimum High Level Input Voltage		2.0V 4.5V 6.0V	1.5 3.15 4.2	1.5 3.15 4.2	1.5 3.15 4.2	1.5 3.15 4.2	V	
	V_{IL}	Maximum Low Level Input Voltage**		2.0V 4.5V 6.0V	0.5 1.35 1.8	0.5 1.35 1.8	0.5 1.35 1.8	0.5 1.35 1.8	V
		V_{OH}	Minimum High Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20$ μ A	2.0V 4.5V 6.0V	2.0 4.5 6.0	1.9 4.4 5.9	1.9 4.4 5.9	1.9 4.4 5.9
V_{OL}			Maximum Low Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 4.0$ mA $ I_{OUT} \leq 5.2$ mA	4.5V 6.0V	4.2 5.7	3.98 5.48	3.84 5.34	3.7 5.2
	I_{IN}		Maximum Input Current	$V_{IN} = V_{CC}$ or GND	6.0V		± 0.1	± 1.0	± 1.0
		I_{CC}	Maximum Quiescent Supply Current	$V_{IN} = V_{CC}$ or GND $I_{OUT} = 0$ μ A	6.0V		2.0	20	40
<p>Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.</p> <p>Note 2: Unless otherwise specified all voltages are referenced to ground.</p> <p>Note 3: Power Dissipation temperature derating — plastic "N" package: -12 mW/$^{\circ}$C from 65$^{\circ}$C to 85$^{\circ}$C; ceramic "J" package: -12 mW/$^{\circ}$C from 100$^{\circ}$C to 125$^{\circ}$C.</p> <p>Note 4: For a power supply of 5V $\pm 10\%$ the worst case output voltages (V_{OH} and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at $V_{CC} = 5.5$V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.</p> <p>**V_{IL} limits are currently tested at 20% of V_{CC}. The above V_{IL} specification (30% of V_{CC}) will be implemented no later than Q1, CY'89.</p>									

AC Electrical Characteristics $V_{CC} = 5V$, $T_A = 25^\circ C$, $C_L = 15 \text{ pF}$, $t_r = t_f = 6 \text{ ns}$

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Propagation Delay		8	15	ns

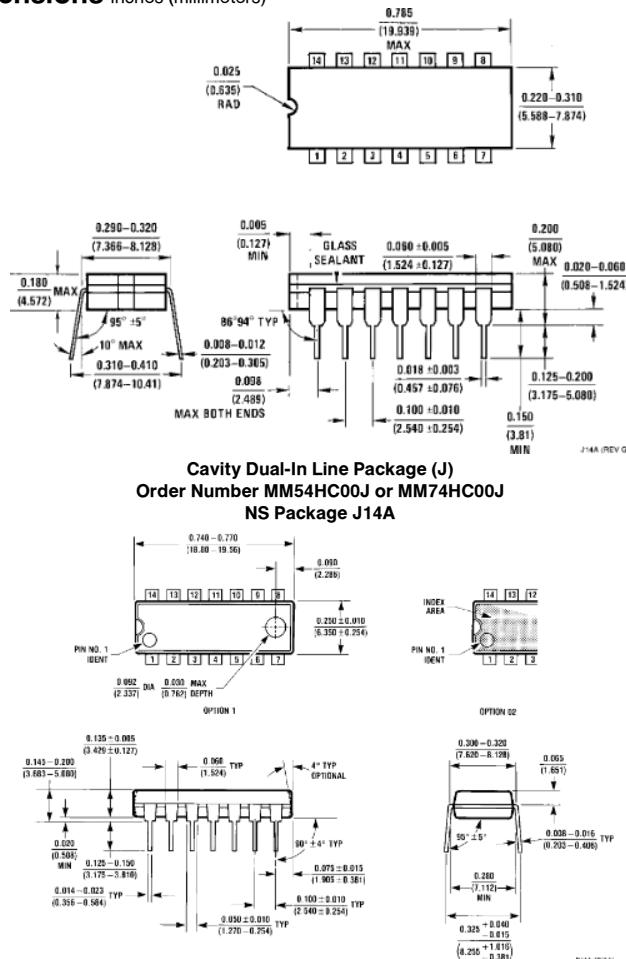
AC Electrical Characteristics $V_{CC} = 2.0V \text{ to } 6.0V$, $C_L = 50 \text{ pF}$, $t_r = t_f = 6 \text{ ns}$ (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^\circ C$		$74HC$	$54HC$	Units
				Typ	Guaranteed Limits			
t_{PHL}, t_{PLH}	Maximum Propagation Delay		2.0V 4.5V 6.0V	45 9 8	90 18 15	113 23 19	134 27 23	ns ns ns
t_{TLH}, t_{THL}	Maximum Output Rise and Fall Time		2.0V 4.5V 6.0V	30 8 7	75 15 13	95 19 16	110 22 19	ns ns ns
C_{PD}	Power Dissipation Capacitance (Note 5)	(per gate)		20				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

MM54HC00/MM74HC00 Quad 2-Input NAND Gate

Physical Dimensions inches (millimeters)



Cavity Dual-In Line Package (J)
Order Number MM54HC00J or MM74HC00J
NS Package J14A

NS Package J14A (REV G)

LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 to the user.
2. A critical component is 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.



National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe

Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.

13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.

Tel: 81-043-299-2309
Fax: 81-043-299-2408

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

MM54HC14/MM74HC14 Hex Inverting Schmitt Trigger

General Description

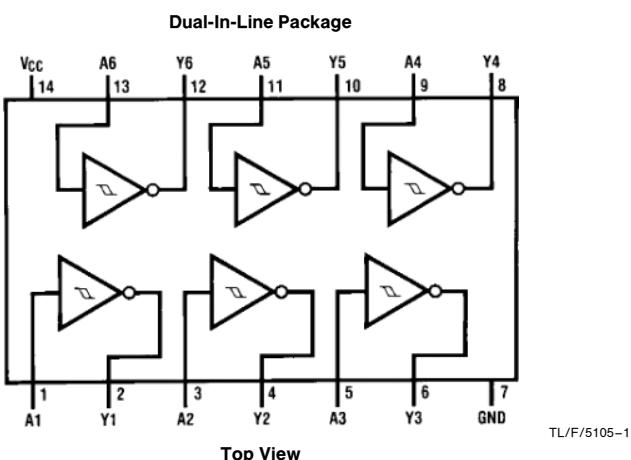
The MM54HC14/MM74HC14 utilizes advanced silicon-gate CMOS technology to achieve the low power dissipation and high noise immunity of standard CMOS, as well as the capability to drive 10 LS-TTL loads.

The 54HC/74HC logic family is functionally and pinout compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

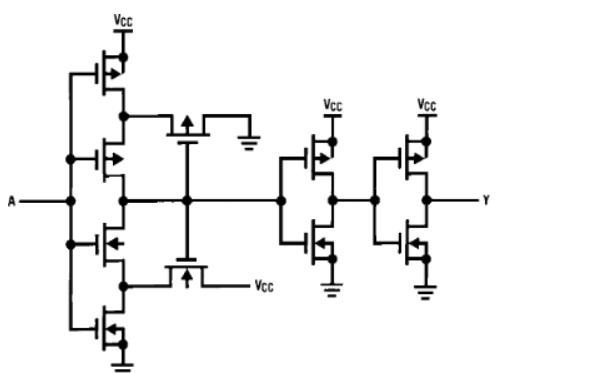
Features

- Typical propagation delay: 13 ns
- Wide power supply range: 2–6V
- Low quiescent current: 20 μ A maximum (74HC Series)
- Low input current: 1 μ A maximum
- Fanout of 10 LS-TTL loads
- Typical hysteresis voltage: 0.9V at V_{CC} = 4.5V

Connection and Schematic Diagrams



Order Number MM54HC14 or MM74HC14



Absolute Maximum Ratings (Notes 1 & 2)										
If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.				Power Dissipation (P_D)						
Supply Voltage (V_{CC})	−0.5 to +7.0V					600 mW				
DC Input Voltage (V_{IN})	−1.5 to V_{CC} + 1.5V					500 mW				
Symbol	Parameter	Conditions	V_{CC}	Min	Max	Units	Lead Temp. (T_L) (Soldering 10 seconds)			
DC Output Voltage (V_{OUT})				60°C						
Clamp Diode Current (I_{IK}, I_{OK})	± 20 mA									
DC Output Current, per pin (I_{OUT})	± 25 mA									
DC V_{CC} or GND Current, per pin (I_{CC})	± 50 mA									
Storage Temperature Range (T_{STG})	−65°C to +150°C									
Operating Conditions										
Supply Voltage (V_{CC})				2	6	V	Power Dissipation (P_D)			
DC Input or Output Voltage ((V_{IN}, V_{OUT}))	0 V_{CC}					V	600 mW 500 mW	S.O. Package only		
Operating Temp. Range (T_A)				260°C						
MM74HC				−40	+ 85	°C				
MM54HC				−55	+ 125	°C				
DC Electrical Characteristics (Note 4)										
Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^\circ\text{C}$		74HC	54HC	Units		
				Typ	Guaranteed Limits					
V_{T+}	Positive Going Threshold Voltage	Minimum	2.0V 4.5V 6.0V	1.2	1.0	1.0	1.0	V		
				2.7	2.0	2.0	2.0			
		Maximum		3.2	3.0	3.0	3.0			
				2.0V 4.5V 6.0V	1.2	1.5	1.5	V		
		Negative Going Threshold Voltage	2.0V 4.5V 6.0V	2.7	3.15	3.15	3.15			
				3.2	4.2	4.2	4.2			
V_{T-}	Hysteresis Voltage	Minimum		0.7	0.3	0.3	0.3	V		
				1.8	0.9	0.9	0.9			
		Maximum		2.2	1.2	1.2	1.2			
				2.0V 4.5V 6.0V	1.0	1.0	1.0	V		
V_H	Hysteresis Voltage	Minimum	2.0V 4.5V 6.0V	0.5	0.2	0.2	0.2			
				0.9	0.4	0.4	0.4			
		Maximum		1.0	0.5	0.5	0.5			
				2.0V 4.5V 6.0V	1.0	1.0	1.0	V		
V_{OH}	Minimum High Level Output Voltage	$V_{IN} = V_{IL}$ $ I_{OUT} = 20 \mu\text{A}$	2.0V 4.5V 6.0V	2.0	1.9	1.9	1.9	V		
				4.5	4.4	4.4	4.4			
		$V_{IN} = V_{IL}$ $ I_{OUT} = 4.0 \text{ mA}$ $ I_{OUT} = 5.2 \text{ mA}$		6.0	5.9	5.9	5.9			
				4.5V 6.0V	4.2 5.7	3.98 5.48	3.84 5.34	V		
V_{OL}	Maximum Low Level Output Voltage	$V_{IN} = V_{IH}$ $ I_{OUT} = 20 \mu\text{A}$	2.0V 4.5V 6.0V	0	0.1	0.1	0.1	V		
				0	0.1	0.1	0.1			
		$V_{IN} = V_{IH}$ $ I_{OUT} = 4.0 \text{ mA}$ $ I_{OUT} = 5.2 \text{ mA}$		0	0.1	0.1	0.1			
				4.5V 6.0V	0.2 0.2	0.26 0.26	0.33 0.33	V		
I_{IN}	Maximum Input Current	$V_{IN} = V_{CC}$ or GND	6.0V		± 0.1	± 1.0	± 1.0	μA		
I_{CC}	Maximum Quiescent Supply Current	$V_{IN} = V_{CC}$ or GND $ I_{OUT} = 0 \mu\text{A}$	6.0V		2.0	20	40	μA		

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ± 10% the worst case output voltages (V_{OH} and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at $V_{CC} = 5.5\text{V}$ and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN} , I_{CC} , and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

AC Electrical Characteristics $V_{CC}=5V$, $T_A=25^\circ C$, $C_L=15\text{ pF}$, $t_r=t_f=6\text{ ns}$

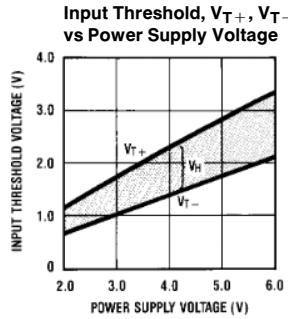
Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PHL}, t_{PLH}	Maximum Propagation Delay		12	22	ns

AC Electrical Characteristics $V_{CC}=2.0V$ to $6.0V$, $C_L=50\text{ pF}$, $t_r=t_f=6\text{ ns}$ (unless otherwise specified)

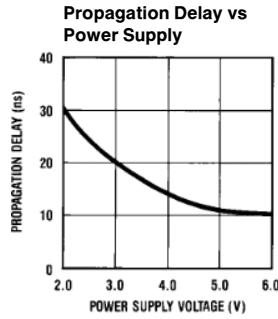
Symbol	Parameter	Conditions	V_{CC}	$T_A=25^\circ C$		$74HC$	$54HC$	Units
				Typ		$T_A=-40$ to $85^\circ C$	$T_A=-55$ to $125^\circ C$	
t_{PHL}, t_{PLH}	Maximum Propagation Delay		2.0V	60	125	156	188	ns
			4.5V	13	25	31	38	ns
			6.0V	11	21	26	32	ns
t_{TLH}, t_{THL}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
C_{PD}	Power Dissipation Capacitance (Note 5)	(per gate)		27				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Typical Performance Characteristics



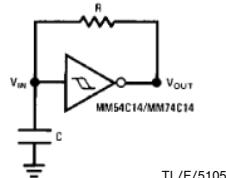
TL/F/5105-3



TL/F/5105-4

Typical Applications

Low Power Oscillator

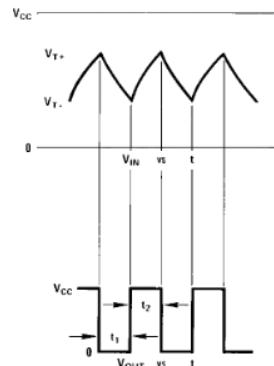


$$t_1 \approx RC \ln \frac{V_{T+}}{V_{T-}}$$

$$t_2 \approx RC \ln \frac{V_{CC} - V_{T-}}{V_{CC} - V_{T+}}$$

$$f \approx \frac{1}{RC \ln \frac{V_{T+}(V_{CC} - V_{T-})}{V_{T-}(V_{CC} - V_{T+})}}$$

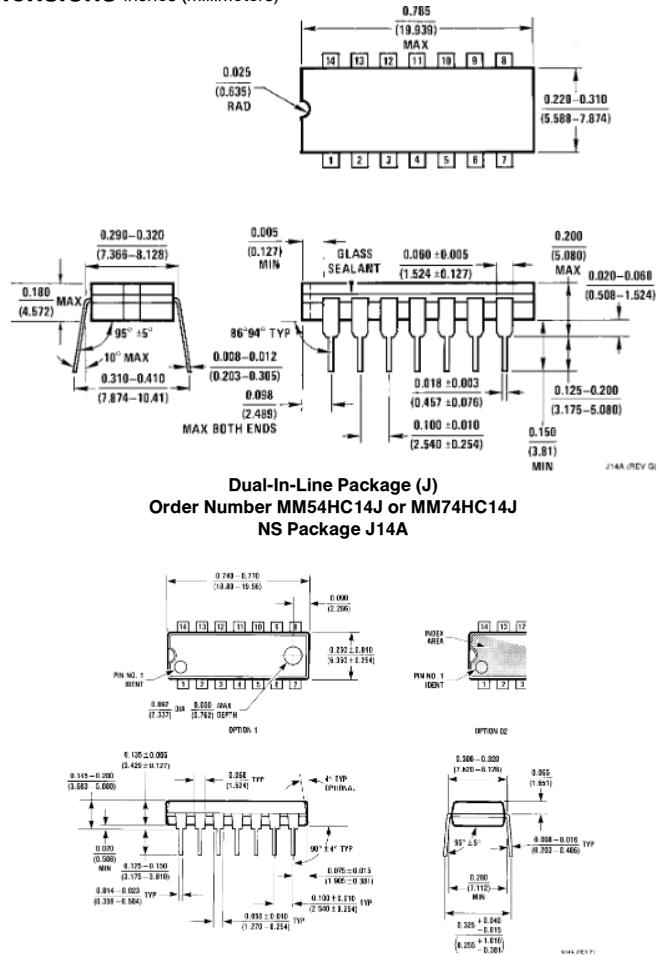
Note: The equations assume $t_1 + t_2 \gg t_{pd0} + t_{pd1}$



TL/F/5105-6

MM54HC14/MM74HC14 Hex Inverting Schmitt Trigger

Physical Dimensions inches (millimeters)



LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 to the user.
2. A critical component is 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.



National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

MM54HC123A/MM74HC123A Dual Retriggerable Monostable Multivibrator

General Description

The MM54/74HC123A high speed monostable multivibrators (one shots) utilize advanced silicon-gate CMOS technology. They feature speeds comparable to low power Schottky TTL circuitry while retaining the low power and high noise immunity characteristic of CMOS circuits.

Each multivibrator features both a negative, A, and a positive, B, transition triggered input, either of which can be used as an inhibit input. Also included is a clear input that when taken low resets the one shot. The 'HC123 can be triggered on the positive transition of the clear while A is held low and B is held high.

The 'HC123A is retriggerable. That is it may be triggered repeatedly while their outputs are generating a pulse and the pulse will be extended.

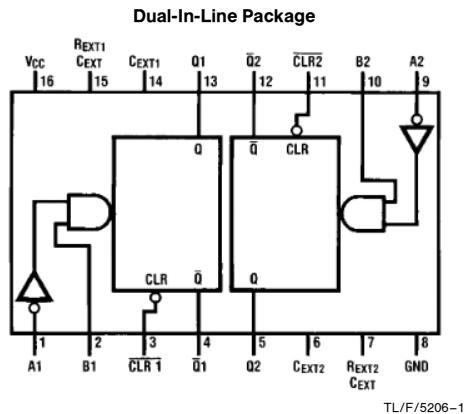
Pulse width stability over a wide range of temperature and supply is achieved using linear CMOS techniques. The out-

put pulse equation is simply: $PW = (R_{EXT}) (C_{EXT})$; where PW is in seconds, R is in ohms, and C is in farads. All inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

Features

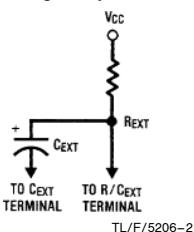
- Typical propagation delay: 25 ns
- Wide power supply range: 2V–6V
- Low quiescent current: 80 μ A maximum (74HC Series)
- Low input current: 1 μ A maximum
- Fanout of 10 LS-TTL loads
- Simple pulse width formula $T = RC$
- Wide pulse range: 400 ns to ∞ (typ)
- Part to part variation: $\pm 5\%$ (typ)
- Schmitt Trigger A & B inputs enable infinite signal input rise and fall times.

Connection Diagram



Top View
Order Number MM54HC123A or MM74HC123A

Timing Component



Note: Pin 6 and Pin 14 must be hard-wired to GND.

Truth Table

Inputs			Outputs	
Clear	A	B	Q	\bar{Q}
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	↑	↑
H	↓	H	↓	↓
↑	L	H	↑	↑

H = High Level
L = Low Level
↑ = Transition from Low to High
↓ = Transition from High to Low
↑ = One High Level Pulse
↓ = One Low Level Pulse
X = Irrelevant

Absolute Maximum Ratings (Notes 1 & 2)		Operating Conditions						
If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.								
Supply Voltage (V_{CC})	-0.5V to +7.0V	Supply Voltage (V_{CC})	2	6	V			
DC Input Voltage (V_{IN})	-1.5V to V_{CC} + 1.5V	DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V			
DC Output Voltage (V_{OUT})	-0.5V to V_{CC} + 0.5V	Operating Temp. Range (T_A)						
Clamp Diode Current (I_{IK}, I_{OK})	± 20 mA	MM74HC	-40	+85	$^{\circ}$ C			
DC Output Current, per pin (I_{OUT})	± 25 mA	MM54HC	-55	+125	$^{\circ}$ C			
DC V_{CC} or GND Current, per pin (I_{CC})	± 50 mA	Input Rise or Fall Times (Clear Input)						
Storage Temperature Range (T_{STG})	-65 $^{\circ}$ C to +150 $^{\circ}$ C	(t_r, t_f) $V_{CC} = 2.0V$		1000	ns			
Power Dissipation (P_D)		$V_{CC} = 4.5V$		500	ns			
(Note 3) S.O. Package only	600 mW 500 mW	$V_{CC} = 6.0V$		400	ns			
Lead Temperature (T_L) (Soldering 10 seconds)	260 $^{\circ}$ C							
DC Electrical Characteristics (Note 4)								
Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^{\circ}$ C		74HC	54HC	Units
				Typ		$T_A = -40$ to 85° C	$T_A = -55$ to 125° C	
V_{IH}	Minimum High Level Input Voltage		2.0V 4.5V 6.0V	1.5 3.15 4.2	1.5 3.15 4.2	1.5 3.15 4.2	1.5 3.15 4.2	V
V_{IL}	Maximum Low Level Input Voltage		2.0V 4.5V 6.0V	0.3 0.9 1.2	0.3 0.9 1.2	0.3 0.9 1.2	0.3 0.9 1.2	V
V_{OH}	Minimum High Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20 \mu A$	2.0V 4.5V 6.0V	2.0 4.5 6.0	1.9 4.4 5.9	1.9 4.4 5.9	1.9 4.4 5.9	V
		$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 4.0$ mA $ I_{OUT} \leq 5.2$ mA	4.5V 6.0V	4.2 5.7	3.98 5.48	3.84 5.34	3.7 5.2	V
V_{OL}	Maximum Low Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20 \mu A$	2.0V 4.5V 6.0V	0 0 0	0.1 0.1 0.1	0.1 0.1 0.1	0.1 0.1 0.1	V
		$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 4$ mA $ I_{OUT} \leq 5.2$ mA	4.5V 6.0V	0.2 0.2	0.26 0.26	0.33 0.33	0.4 0.4	V
I_{IN}	Maximum Input Current (Pins 7, 15)	$V_{IN} = V_{CC}$ or GND	6.0V		± 0.5	± 5.0	± 5.0	μA
I_{IN}	Maximum Input Current (all other pins)	$V_{IN} = V_{CC}$ or GND	6.0V		± 0.1	± 1.0	± 1.0	μA
I_{CC}	Maximum Quiescent Supply Current (standby)	$V_{IN} = V_{CC}$ or GND $I_{OUT} = 0 \mu A$	6.0V		8.0	80	160	μA
I_{CC}	Maximum Active Supply Current (per monostable)	$V_{IN} = V_{CC}$ or GND $R/C_{EXT} = 0.5V_{CC}$	2.0V 4.5V 6.0V	36 0.33 0.7	80 1.0 2.0	110 1.3 2.6	130 1.6 3.2	μA mA mA
<p>Note 1: Maximum Ratings are those values beyond which damage to the device may occur.</p> <p>Note 2: Unless otherwise specified all voltages are referenced to ground.</p> <p>Note 3: Power Dissipation Temperature Derating: Plastic "N" Package: -12mW/$^{\circ}$C from 65$^{\circ}$C to 85$^{\circ}$C Ceramic "J" Package: -12mW/$^{\circ}$C from 100$^{\circ}$C to 125$^{\circ}$C.</p> <p>Note 4: For a power supply of 5V $\pm 10\%$ the worst-case output voltages (V_{OH}, V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst-case V_{IH} and V_{IL} occur at $V_{CC} = 5.5$V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst-case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.</p>								

AC Electrical Characteristics $V_{CC} = 5V$, $T_A = 25^\circ C$, $C_L = 15 pF$, $t_r = t_f = 6 ns$

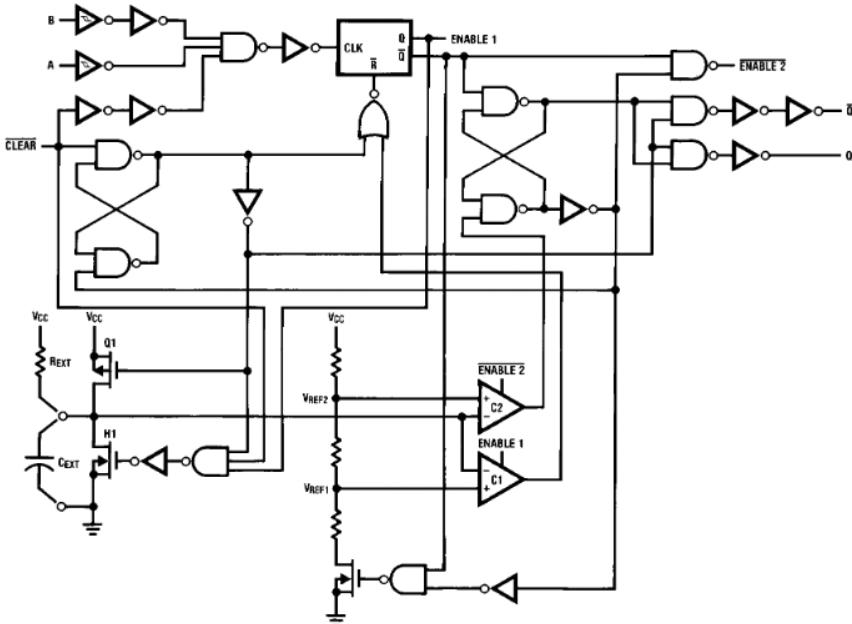
Symbol	Parameter	Conditions	Typ	Limit	Units
t_{PLH}	Maximum Trigger Propagation Delay A, B or Clear to Q		22	33	ns
t_{PHL}	Maximum Trigger Propagation Delay A, B or Clear to Q		25	42	ns
t_{PHL}	Maximum Propagation Delay, Clear to Q		20	27	ns
t_{PLH}	Maximum Propagation Delay, Clear to \bar{Q}		22	33	ns
t_W	Minimum Pulse Width, A, B or Clear		14	26	ns
t_{REM}	Minimum Clear Removal Time			0	ns
$t_{WQ(MIN)}$	Minimum Output Pulse Width	$C_{EXT} = 28 pF$ $R_{EXT} = 2 k\Omega$	400		ns
t_{WQ}	Output Pulse Width	$C_{EXT} = 1000 pF$ $R_{EXT} = 10 k\Omega$	10		μs

AC Electrical Characteristics $C_L = 50 pF$, $t_r = t_f = 6 ns$ (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^\circ C$		$74HC$	$54HC$	Units
				Typ	Guaranteed Limits			
t_{PLH}	Maximum Trigger Propagation Delay, A, B or Clear to Q		2.0V	77	169	194	210	ns
			4.5V	26	42	51	57	ns
			6.0V	21	32	39	44	ns
t_{PHL}	Maximum Trigger Propagation Delay, A, B or Clear to \bar{Q}		2.0V	88	197	229	250	ns
			4.5V	29	48	60	67	ns
			6.0V	24	38	46	51	ns
t_{PHL}	Maximum Propagation Delay Clear to Q		2.0V	54	114	132	143	ns
			4.5V	23	34	41	45	ns
			6.0V	19	28	33	36	ns
t_{PLH}	Maximum Propagation Delay Clear to \bar{Q}		2.0V	56	116	135	147	ns
			4.5V	25	36	42	46	ns
			6.0V	20	29	34	37	ns
t_W	Minimum Pulse Width A, B, Clear		2.0V	57	123	144	157	ns
			4.5V	17	30	37	42	ns
			6.0V	12	21	27	30	ns
t_{REM}	Minimum Clear Removal Time		2.0V		0	0	0	ns
			4.5V		0	0	0	ns
			6.0V		0	0	0	ns
t_{TLH}, t_{THL}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
$t_{WQ(MIN)}$	Minimum Output Pulse Width	$C_{EXT} = 28 pF$ $R_{EXT} = 2 k\Omega$ $R_{EXT} = 6 k\Omega (V_{CC} = 2V)$	2.0V	1.5				μs
			4.5V	450				ns
			6.0V	380				ns
t_{WQ}	Output Pulse Width	$C_{EXT} = 0.1 \mu F$	Min	5.0V	1	0.9	0.86	0.85
		$R_{EXT} = 10 k\Omega$	Max	5.0V	1	1.1	1.14	1.15
C_{IN}	Maximum Input Capacitance (Pins 7 & 15)			12	20	20	20	pF
C_{IN}	Maximum Input Capacitance (other inputs)			6	10	10	10	pF
C_{PD}	Power Dissipation Capacitance	(Note 5)		70				pF

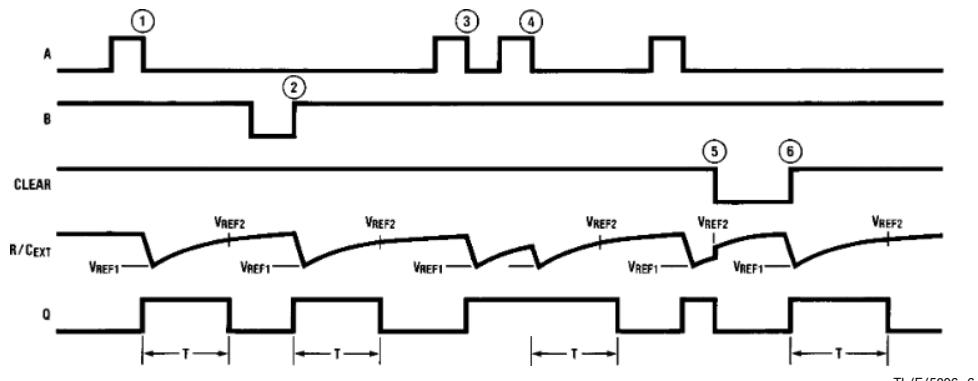
Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Logic Diagram



TL/F/5206-5

Theory of Operation



TL/F/5206-6

- ① POSITIVE EDGE TRIGGER (PULSE LENGTHENING)
- ② NEGATIVE EDGE TRIGGER
- ③ RESET PULSE SHORTENING
- ④ POSITIVE EDGE TRIGGER
- ⑤ CLEAR TRIGGER

FIGURE 1

TRIGGER OPERATION

As shown in Figure 1 and the logic diagram before an input trigger occurs, the one shot is in the quiescent state with the Q output low, and the timing capacitor C_{EXT} completely charged to V_{CC}. When the trigger input A goes from V_{CC} to GND (while inputs B and clear are held to V_{CC}) a valid trigger is recognized, which turns on comparator C1 and N-

channel transistor N1 ①. At the same time the output latch is set. With transistor N1 on, the capacitor C_{EXT} rapidly discharges toward GND until V_{REF1} is reached. At this point the output of comparator C1 changes state and transistor N1 turns off. Comparator C1 then turns off while at the same time comparator C2 turns on. With transistor N1 off, the capacitor C_{EXT} begins to charge through the timing re-

sistor, R_{EXT} , toward V_{CC} . When the voltage across C_{EXT} equals V_{REF2} , comparator C2 changes state causing the output latch to reset (Q goes low) while at the same time disabling comparator C2. This ends the timing cycle with the monostable in the quiescent state, waiting for the next trigger.

A valid trigger is also recognized when trigger input B goes from GND to V_{CC} (while input A is at GND and input clear is at V_{CC}). The 'HC123A can also be triggered when clear goes from GND to V_{CC} (while A is at GND and B is at V_{CC}).

It should be noted that in the quiescent state C_{EXT} is fully charged to V_{CC} causing the current through resistor R_{EXT} to be zero. Both comparators are "off" with the total device current due only to reverse junction leakages. An added feature of the 'HC123A is that the output latch is set via the input trigger without regard to the capacitor voltage. Thus, propagation delay from trigger to Q is independent of the value of C_{EXT} , R_{EXT} , or the duty cycle of the input waveform.

RETRIGGER OPERATION

The 'HC123A is retriggered if a valid trigger occurs ③ followed by another trigger ④ before the Q output has returned to the quiescent (zero) state. Any retrigger, after the timing node voltage at the R/ C_{EXT} pin has begun to rise from V_{REF1} , but has not yet reached V_{REF2} , will cause an increase in output pulse width T. When a valid retrigger is initiated ④, the voltage at the R/ C_{EXT} pin will again drop to V_{REF1} before progressing along the RC charging curve

toward V_{CC} . The Q output will remain high until time T, after the last valid retrigger.

Because the trigger-control circuit flip-flop resets shortly after C_X has discharged to the reference voltage of the lower reference circuit, the minimum retrigger time, t_{rr} is a function of internal propagation delays and the discharge time of C_X :

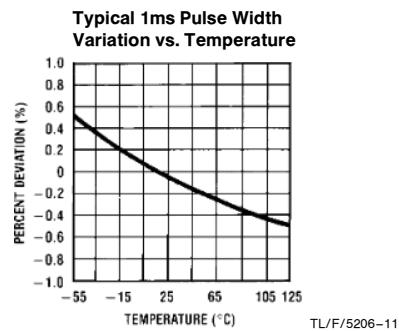
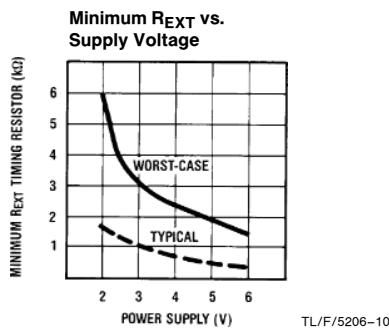
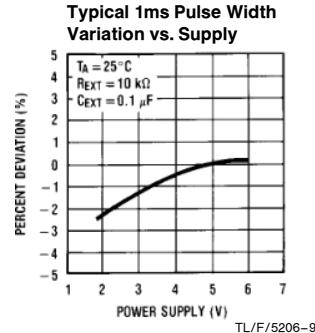
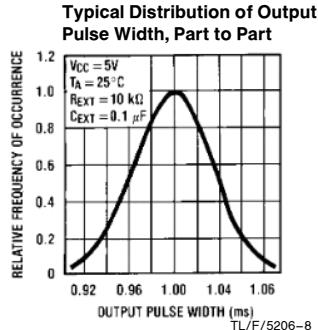
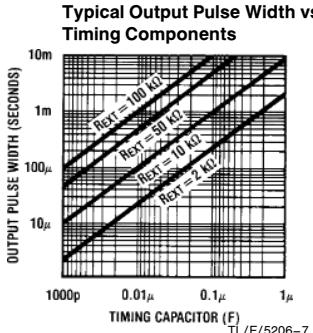
$$t_{rr} \approx 20 + \frac{187}{V_{CC} - 0.7} + \frac{565 + (0.256 V_{CC}) C_X}{[V_{CC} - 0.7]^2}$$

Another removal/retrigger time occurs when a short clear pulse is used. Upon receipt of a clear, the one shot must charge the capacitor up to the upper trip point before the one shot is ready to receive the next trigger. This time is dependent on the capacitor used and is approximately:

$$t_{rr} = 196 + \frac{640}{V_{CC} - 0.7} + \frac{522 + (0.3 V_{CC}) C_X}{(V_{CC} - 0.7)^2} \text{ ns}$$

RESET OPERATION

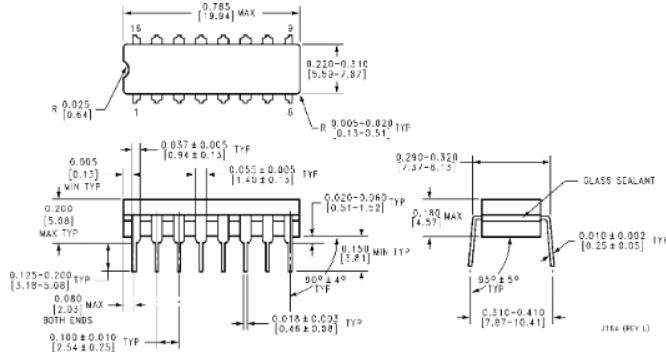
These one shots may be reset during the generation of the output pulse. In the reset mode of operation, an input pulse on clear sets the reset latch and causes the capacitor to be fast charged to V_{CC} by turning on transistor Q1 ⑤. When the voltage on the capacitor reaches V_{REF2} , the reset latch will clear and then be ready to accept another pulse. If the clear input is held low, any trigger inputs that occur will be inhibited and the Q and \bar{Q} outputs of the output latch will not change. Since the Q output is reset when an input low level is detected on the Clear input, the output pulse T can be made significantly shorter than the minimum pulse width specification.



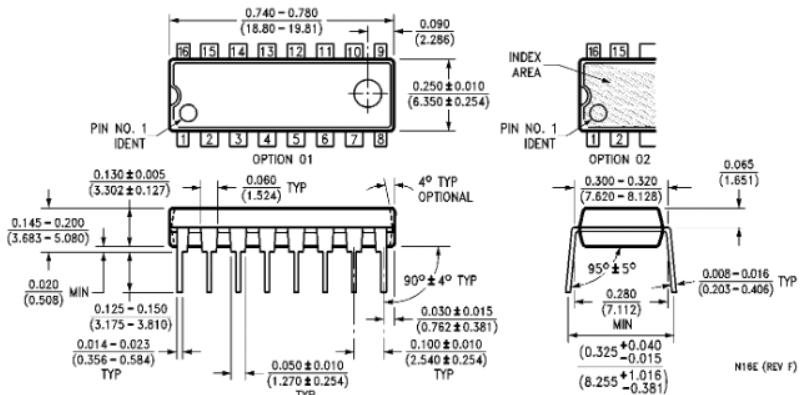
Note: R and C are not subjected to temperature. The C is polypropylene.

MM54HC123A/MM74HC123A Dual Retriggerable Monostable Multivibrator

Physical Dimensions inches (millimeters)



Dual-In-Line Package (J)
Order Number MM54HC123AJ or MM74HC123AJ
NS Package Number J16A



Dual-In-Line Package (N)
Order Number MM74HC123AN
NS Package Number N16E

LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 to the user.
2. A critical component is 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.



National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

54LS245/DM54LS245/DM74LS245 TRI-STATE® Octal Bus Transceiver

General Description

These octal bus transceivers are designed for asynchronous two-way communication between data buses. The control function implementation minimizes external timing requirements.

The device allows data transmission from the A bus to the B bus or from the B bus to the A bus depending upon the logic level at the direction control (DIR) input. The enable input (\bar{G}) can be used to disable the device so that the buses are effectively isolated.

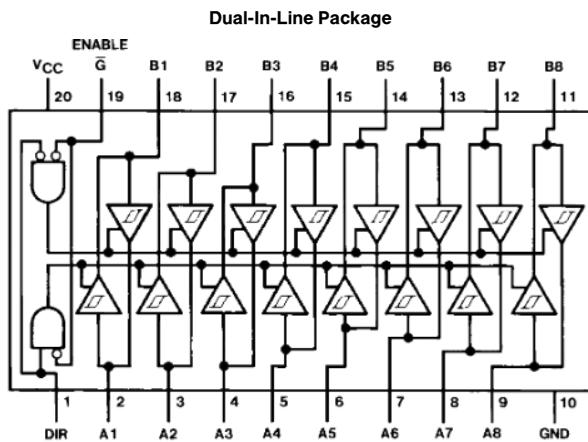
Features

- Bi-Directional bus transceiver in a high-density 20-pin package
- TRI-STATE outputs drive bus lines directly
- PNP inputs reduce DC loading on bus lines
- Hysteresis at bus inputs improve noise margins
- Typical propagation delay times, port-to-port 8 ns
- Typical enable/disable times 17 ns
- I_{OL} (sink current)

54LS	12 mA
74LS	24 mA
- I_{OH} (source current)

54LS	-12 mA
74LS	-15 mA
- Alternate Military/Aerospace device (54LS245) is available. Contact a National Semiconductor Sales Office/Distributor for specifications.

Connection Diagram



TL/F/6413-1

Order Number 54LS245MQB, 54LS245FMB, 54LS245LMQB,
DM54LS245J, DM54LS245W, DM74LS245WM or DM74LS245N
See NS Package Number E20A, J20A, M20B, N20A or W20A

Function Table

Enable \bar{G}	Direction Control DIR	Operation
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

H = High Level, L = Low Level, X = Irrelevant

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage DIR or \overline{G}	7V
A or B	5.5V
Operating Free Air Temperature Range DM54LS and 54LS	-55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	-65°C to +150°C

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

Recommended Operating Conditions

Symbol	Parameter	DM54LS245			DM74LS245			Units
		Min	Nom	Max	Min	Nom	Max	
V _{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V _{IH}	High Level Input Voltage	2			2			V
V _{IL}	Low Level Input Voltage			0.7			0.8	V
I _{OH}	High Level Output Current			-12			-15	mA
I _{OL}	Low Level Output Current			12			24	mA
T _A	Free Air Operating Temperature	-55		125	0		70	°C

Electrical Characteristics

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

Symbol	Parameter	Conditions			Min	Typ (Note 1)	Max	Units	
V _I	Input Clamp Voltage	V _{CC} = Min, I _I = -18 mA					-1.5	V	
HYS	Hysteresis (V _{T+} - V _{T-})	V _{CC} = Min			0.2	0.4		V	
V _{OH}	High Level Output Voltage	V _{CC} = Min, V _{IH} = Min V _{IL} = Max, I _{OH} = -1 mA	DM74		2.7			V	
		V _{CC} = Min, V _{IL} = Min V _{IL} = Max, I _{OH} = -3 mA	DM54/DM74		2.4	3.4			
		V _{CC} = Min, V _{IH} = Min V _{IL} = 0.5V, I _{OH} = Max	DM54/DM74		2				
V _{OL}	Low Level Output Voltage	V _{CC} = Min	I _{OL} = 12 mA	DM74			0.4	V	
		V _{IL} = Max	I _{OL} = Max	DM54			0.4		
		V _{IH} = Min		DM74			0.5		
I _{OZH}	Off-State Output Current, High Level Voltage Applied	V _{CC} = Max V _{IL} = Max V _{IH} = Min	V _O = 2.7V				20	μA	
I _{OZL}	Off-State Output Current, Low Level Voltage Applied		V _O = 0.4V				-200	μA	
I _I	Input Current at Maximum Input Voltage	V _{CC} = Max	A or B	V _I = 5.5V			0.1	mA	
			DIR or \overline{G}	V _I = 7V			0.1		
I _{IH}	High Level Input Current	V _{CC} = Max, V _I = 2.7V					20	μA	
I _{IL}	Low Level Input Current	V _{CC} = Max, V _I = 0.4V					-0.2	mA	
I _{OS}	Short Circuit Output Current	V _{CC} = Max (Note 2)			-40		-225	mA	
I _{CC}	Supply Current	Outputs High		V _{CC} = Max		48	70	mA	
		Outputs Low				62	90		
		Outputs at Hi-Z				64	95		

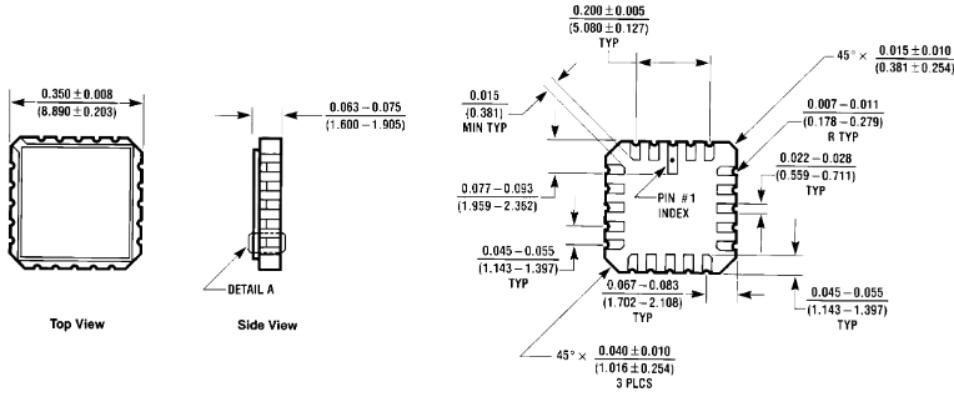
Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, not to exceed one second duration

Switching Characteristics $V_{CC} = 5V$, $T_A = 25^\circ C$ (See Section 1 for Test Waveforms and Output Load)

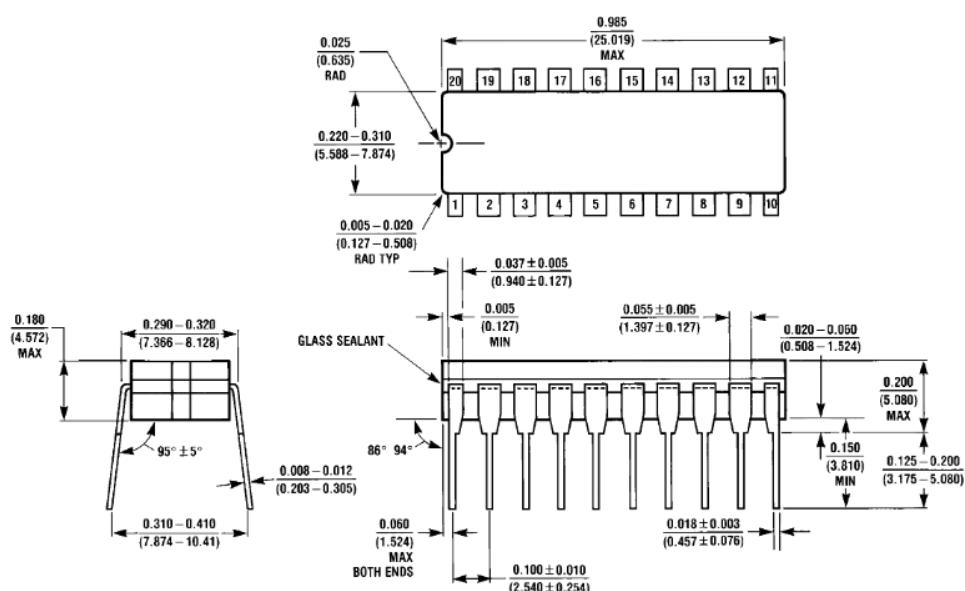
Symbol	Parameter	Conditions	DM54/74		Units	
			LS245			
			Min	Max		
t_{PLH}	Propagation Delay Time, Low-to-High-Level Output	$C_L = 45 \text{ pF}$ $R_L = 667\Omega$		12	ns	
t_{PHL}	Propagation Delay Time, High-to-Low-Level Output			12	ns	
t_{PZL}	Output Enable Time to Low Level			40	ns	
t_{PZH}	Output Enable Time to High Level			40	ns	
t_{PLZ}	Output Disable Time from Low Level	$C_L = 5 \text{ pF}$ $R_L = 667\Omega$		25	ns	
t_{PHZ}	Output Disable Time from High Level			25	ns	
t_{PLH}	Propagation Delay Time, Low-to-High-Level Output	$C_L = 150 \text{ pF}$ $R_L = 667\Omega$		16	ns	
t_{PHL}	Propagation Delay Time, High-to-Low-Level Output			17	ns	
t_{PZL}	Output Enable Time to Low Level			45	ns	
t_{PZH}	Output Enable Time to High Level			45	ns	

Physical Dimensions inches (millimeters)



E20A (REV D)

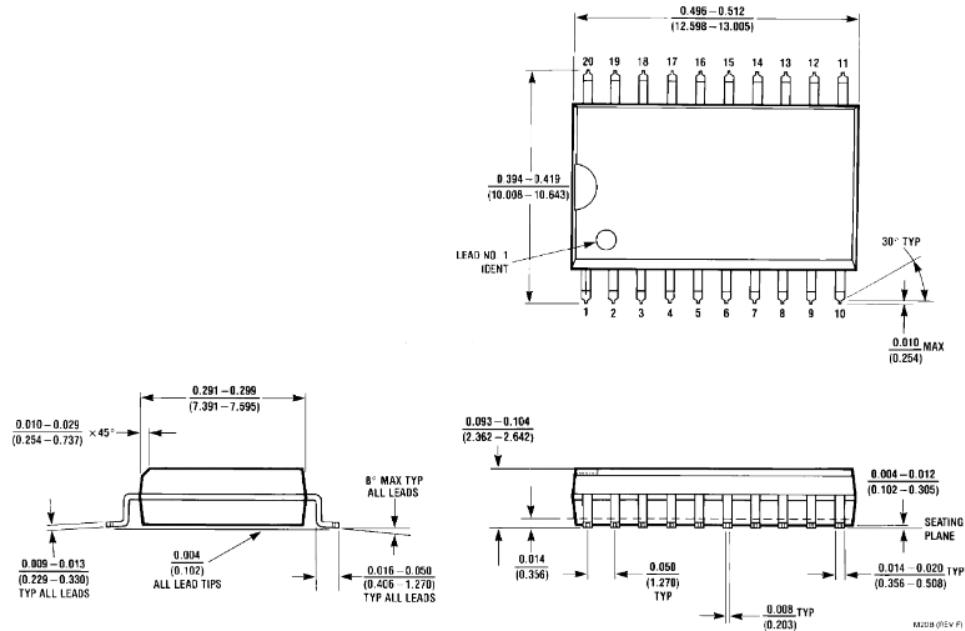
Ceramic Leadless Chip Carrier Package (E)
Order Number 54LS245LMQB
NS Package Number E20A



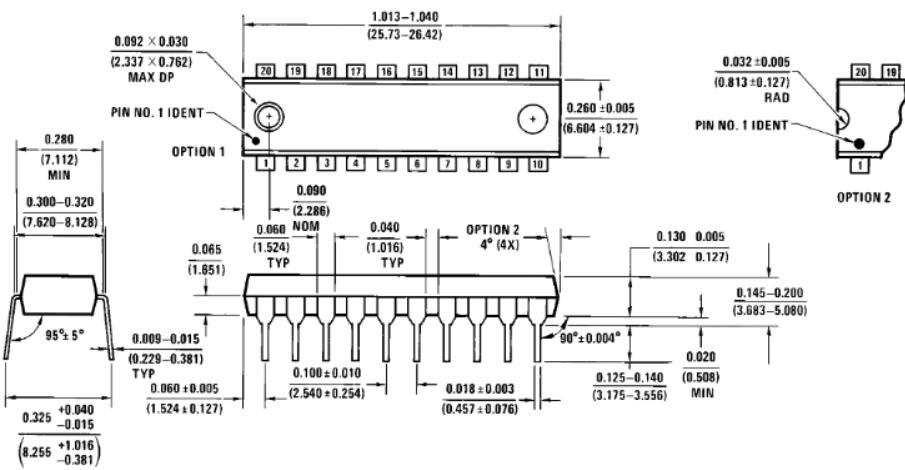
J20A (REV M)

20-Lead Ceramic Dual-In-Line Package (J)
Order Number 54LS245DMQB or DM54LS245J
NS Package Number J20A

Physical Dimensions inches (millimeters) (Continued)

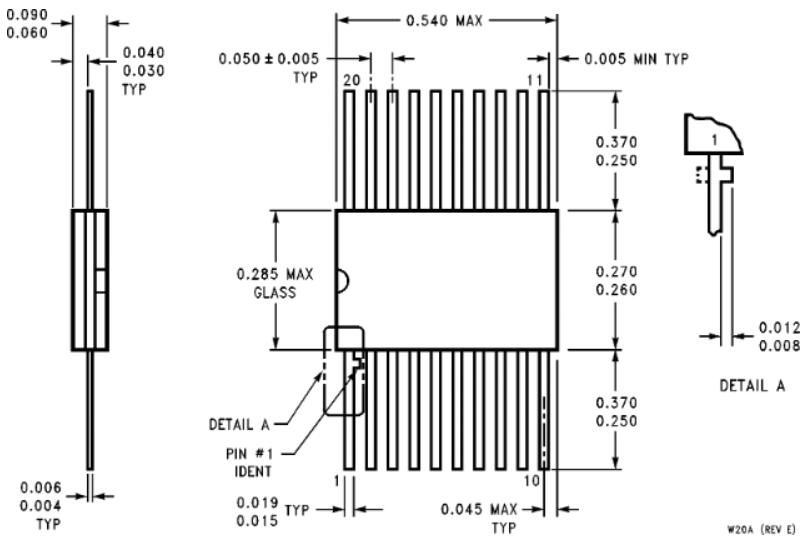


20-Lead Small Outline Molded Package (M)
Order Number DM74LS245WM
NS Package Number M20B



20-Lead Molded Dual-In-Line Package (N)
Order Number DM74LS245N
NS Package Number N20A

Physical Dimensions inches (millimeters) (Continued)



20-Lead Ceramic Flat Package (W)
Order Number 54LS245FMQB or DM54LS245W
NS Package Number W20A

W20A (REV E)

LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 to the user.
2. A critical component is 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.



National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-0718

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

MM54HC4060/MM74HC4060 14 Stage Binary Counter

General Description

The MM54HC4060/MM74HC4060 is a high speed binary ripple carry counter. These counters are implemented utilizing advanced silicon-gate CMOS technology to achieve speed performance similar to LS-TTL logic while retaining the low power and high noise immunity of CMOS.

The 'HC4060 is a 14-stage counter, which device increments on the falling edge (negative transition) of the input clock, and all their outputs are reset to a low level by applying a logical high on their reset input. The 'HC4060 also has two additional inputs to enable easy connection of either an RC or crystal oscillator.

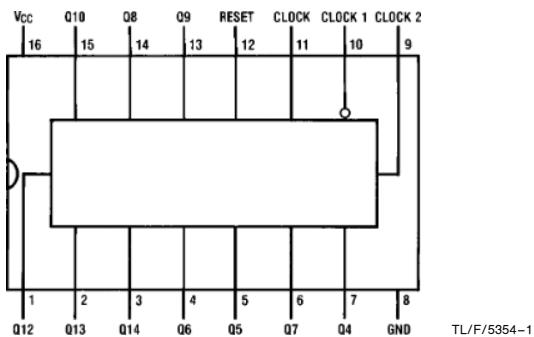
This device is pin equivalent to the CD4060. All inputs are protected from damage due to static discharge by protection diodes to V_{CC} and ground.

Features

- Typical propagation delay: 16 ns
- Wide operating voltage range: 2–6V
- Low input current: 1 μ A maximum
- Low quiescent current: 80 μ A maximum (74 Series)
- Output drive capability: 10 LS-TTL loads

Connection and Logic Diagrams

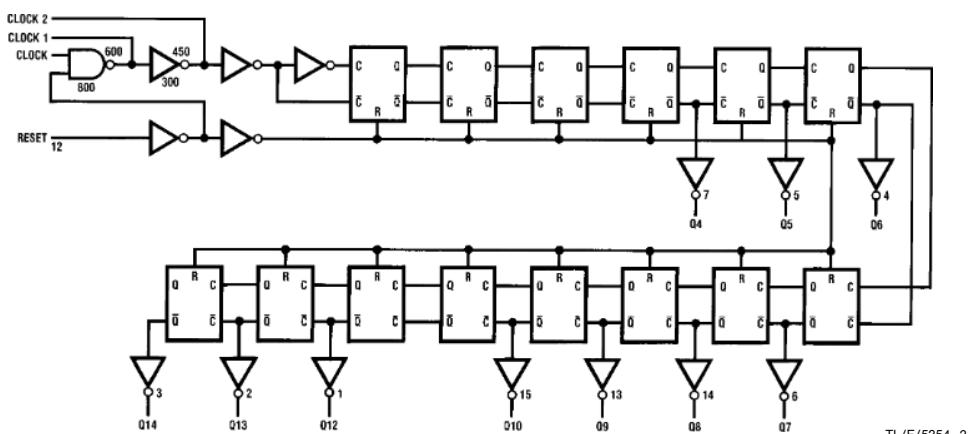
Dual-In-Line Package



TL/F/5354-1

Top View

Order Number MM54HC4060 or MM74HC4060



TL/F/5354-2

Absolute Maximum Ratings (Notes 1 & 2)							
Operating Conditions							
If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.				Min	Max	Units	
Supply Voltage (V_{CC})	-0.5 to + 7.0V	Supply Voltage (V_{CC})	2	6	V		
DC Input Voltage (V_{IN})	-1.5 to V_{CC} + 1.5V	DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V		
DC Output Voltage (V_{OUT})	-0.5 to V_{CC} + 0.5V	Operating Temp. Range (T_A)					
Clamp Diode Current (I_{CD})	± 20 mA	MM74HCT	-40	+ 85	°C		
DC Output Current, per pin (I_{OUT})	± 25 mA	MM54HCT	-55	+ 125	°C		
DC V_{CC} or GND Current, per pin (I_{CC})	± 50 mA	Input Rise or Fall Times (t_r, t_f)					
Storage Temperature Range (T_{STG})	-65°C to + 150°C	$V_{CC} = 2.0V$		1000	ns		
Power Dissipation (P_D)		$V_{CC} = 4.5V$		500	ns		
(Note 3)	600 mW	$V_{CC} = 6.0V$		400	ns		
S.O. Package only	500 mW						
Lead Temperature (T_L)							
(Soldering 10 seconds)	260°C						
DC Electrical Characteristics (Note 4)							
Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^\circ C$	$74HC$	$54HC$	Units
				Typ	Guaranteed Limits		
V_{IH}	Minimum High Level Voltage (Not Applicable to Pins 9 & 10)		2.0V 4.5V 6.0V	1.5 3.15 4.2	1.5 3.15 4.2	1.5 3.15 4.2	V
V_{IL}	Maximum Low Level Input Voltage ** (Not Applicable to Pins 9 & 10)		2.0V 4.5V 6.0V	0.5 1.35 1.8	0.5 1.35 1.8	0.5 1.35 1.8	V
V_{OH}	Minimum High Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20 \mu A$	2.0V 4.5V 6.0V	2.0 4.5 6.0	1.9 4.4 5.9	1.9 4.4 5.9	V
	Except Pins 9 & 10	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 4.0$ mA $ I_{OUT} \leq 5.2$ mA	4.5V 6.0V	4.2 5.7	3.98 5.48	3.84 5.34	V
	Pins 9 & 10	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} = 0.4$ mA $ I_{OUT} = 0.52$ mA			3.98 5.48	3.84 5.34	V
V_{OL}	Maximum Low Level Output Voltage	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20 \mu A$	2.0V 4.5V 6.0V	0 0 0	0.1 0.1 0.1	0.1 0.1 0.1	V
	Except Pins 9 & 10	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 4.0$ mA $ I_{OUT} \leq 5.2$ mA	4.5V 6.0V	0.2 0.2	0.26 0.26	0.33 0.33	V
	Pins 9 & 10	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} = 0.4$ mA $ I_{OUT} = 0.52$ mA			0.26 0.26	0.33 0.33	V
I_{IN}	Maximum Input Current	$V_{IN} = V_{CC}$ or GND	6.0V	± 0.1	± 1.0	± 1.0	μA
I_{CC}	Maximum Quiescent Supply Current	$V_{IN} = V_{CC}$ or GND	6.0V	8.0	80	160	μA
Note 1: Maximum Ratings are those values beyond which damage may occur.							
Note 2: Unless otherwise specified all voltages are referenced to ground.							
Note 3: Power Dissipation temperature derating: plastic "N" package: -12 mW/°C from 65°C to 85°C ceramic "J" package: -12 mW/°C from 100°C to 125°C							
Note 4: For a power supply of 5V $\pm 10\%$ the worst case output voltages (V_{OH} , and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at $V_{CC} = 5.5V$ and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN} , I_{CC} , and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.							
** V_{IL} limits are currently tested at 20% of V_{CC} . The above V_{IL} specification (30% of V_{CC}) will be implemented no later than Q1, CY'89.							

AC Electrical Characteristics

$V_{CC} = 5V$, $T_A = 25^\circ C$, $C_L = 15 \text{ pF}$, $t_r = t_f = 6 \text{ ns}$

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Clock Frequency			30	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay to Q_4	(Note 5)	40	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay to any Q		16	40	ns
t_{REM}	Minimum Reset Removal Time		10	20	ns
t_W	Minimum Pulse Width		10	16	ns

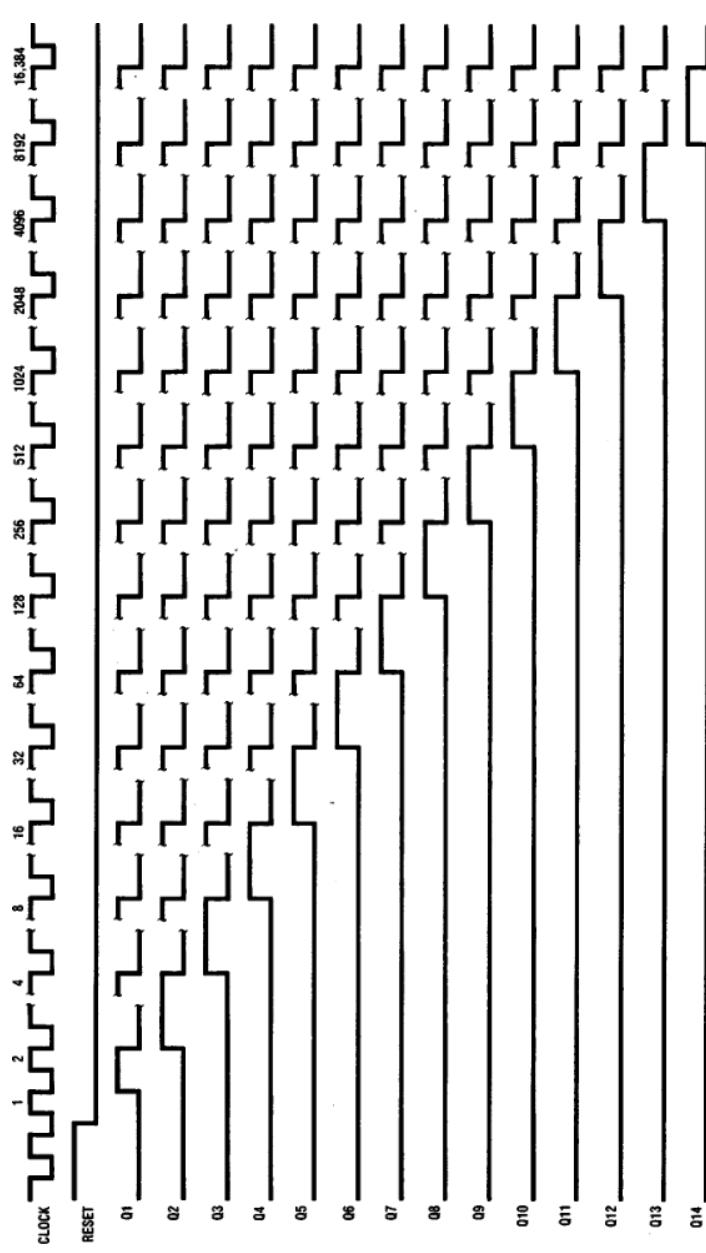
AC Electrical Characteristics $V_{CC} = 2.0V$ to $6.0V$, $C_L = 50 \text{ pF}$, $t_r = t_f = 6 \text{ ns}$ (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A = 25^\circ C$		$T_A = 74HC$	$T_A = 54HC$	Units
				Typ	Guaranteed Limits			
f_{MAX}	Maximum Operating Frequency		2.0V 4.5V 6.0V	6 30 35	5 24 28	4 20 24	MHz MHz MHz	
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock to Q_4		2.0V 4.5V 6.0V	120 42 35	380 76 65	475 95 81	171 114 97	ns ns ns
t_{PHL}	Maximum Propagation Delay Reset to any Q		2.0V 4.5V 6.0V	72 24 20	240 48 41	302 60 51	358 72 61	ns ns ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay Between Stages Q_n to Q_{n+1}		2.0V 4.5V 6.0V	125 25 21	125 31 26	156 38 31	188 38 31	ns ns ns
t_{REM}	Minimum Reset Removal Time		2.0V 4.5V 6.0V	100 20 17	125 25 21	125 30 25	150 30 25	ns ns ns
t_W	Minimum Pulse Width		2.0V 4.5V 6.0V	80 16 14	100 20 17	100 24 20	120 24 20	ns ns ns
t_r, t_f	Maximum Input Rise and Fall Time		2.0V 4.5V 6.0V	1000 500 400	1000 500 400	1000 500 400	1000 500 400	ns ns ns
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time		2.0V 4.5V 6.0V	30 10 9	75 15 13	95 19 16	110 22 19	ns ns ns
C_{PD}	Power Dissipation Capacitance (Note 6)	(per package)		55				pF
C_{IN}	Maximum Input Capacitance			5	10	10	10	pF

Note 5: Typical Propagation delay time to any output can be calculated using: $t_p = 17 + 12(N-1) \text{ ns}$; where N is the number of the output, Q_W , at $V_{CC} = 5V$.

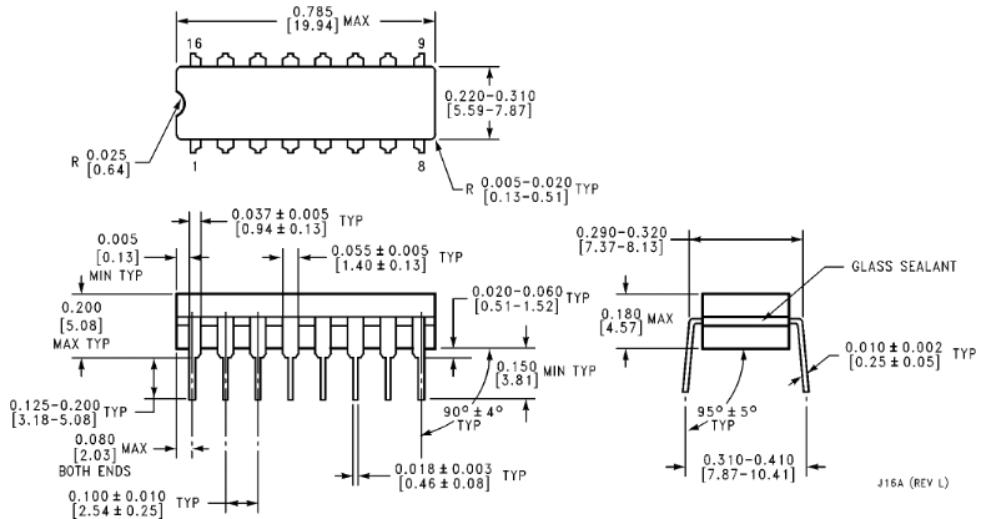
Note 6: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

Timing Diagram



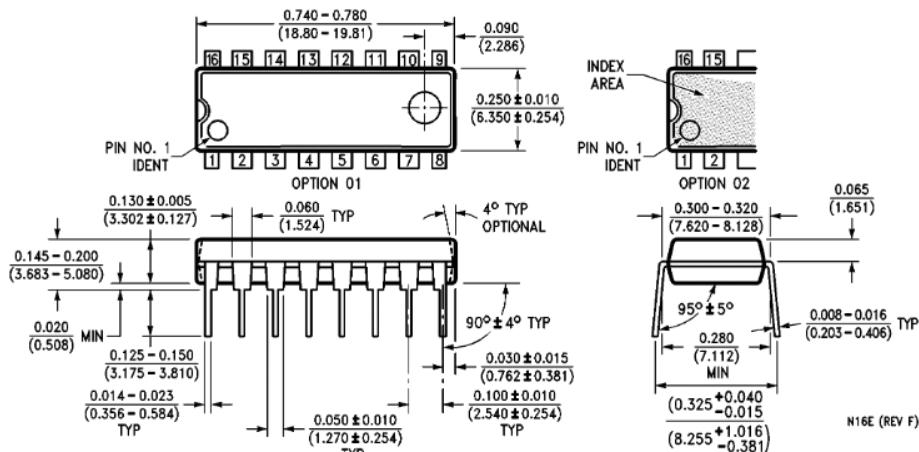
TL/F/5354-3

Physical Dimensions inches (millimeters)



Order Number MM54HC4060J
NS Package J16A

Physical Dimensions inches (millimeters) (Continued)



Order Number MM74HC4060J, N
NS Package N16E

LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 to the user.
2. A critical component is 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.

 National Semiconductor Corporation 1111 West Bardin Road Arlington, TX 76017 Tel: 1(800) 272-9959 Fax: 1(800) 737-7018	National Semiconductor Europe Fax: (+49) 0-180-530 85 86 Email: cnjwge@tevm2.nsc.com Deutsch Tel: (+49) 0-180-530 85 85 English Tel: (+49) 0-180-532 78 32 Français Tel: (+49) 0-180-532 93 58 Italiano Tel: (+49) 0-180-534 16 80	National Semiconductor Hong Kong Ltd. 13th Floor, Straight Block, Ocean Centre, 5 Canton Rd. Tsimshatsui, Kowloon Hong Kong Tel: (852) 2736-1600 Fax: (852) 2736-9960	National Semiconductor Japan Ltd. Tel: 81-043-299-2309 Fax: 81-043-299-2408
---	---	--	--

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

**80V/2.5A Peak, High Frequency
Full Bridge FET Driver**

November 1996

Features

- Independently Drives 4 N-Channel FET in Half Bridge or Full Bridge Configurations
- Bootstrap Supply Max Voltage to 95V_{DC}
- Drives 1000pF Load at 1MHz in Free Air at 50°C with Rise and Fall Times of Typically 10ns
- User-Programmable Dead Time
- On-Chip Charge-Pump and Bootstrap Upper Bias Supplies
- DIS (Disable) Overrides Input Control
- Input Logic Thresholds Compatible with 5V to 15V Logic Levels
- Very Low Power Consumption
- Undervoltage Protection

Applications

- Medium/Large Voice Coil Motors
- Full Bridge Power Supplies
- Class D Audio Power Amplifiers
- High Performance Motor Controls
- Noise Cancellation Systems
- Battery Powered Vehicles
- Peripherals
- U.P.S.

Description

The HIP4081A is a high frequency, medium voltage Full Bridge N-Channel FET driver IC, available in 20 lead plastic SOIC and DIP packages. The HIP4081A can drive every possible switch combination except those which would cause a shoot-through condition. The HIP4081A can switch at frequencies up to 1MHz and is well suited to driving Voice Coil Motors, high-frequency Class D audio amplifiers, and power supplies.

For example, the HIP4081A can drive medium voltage brush motors, and two HIP4081As can be used to drive high performance stepper motors, since the short minimum "on-time" can provide fine micro-stepping capability.

Short propagation delays of approximately 55ns maximizes control loop crossover frequencies and dead-times which can be adjusted to near zero to minimize distortion, resulting in rapid, precise control of the driven load.

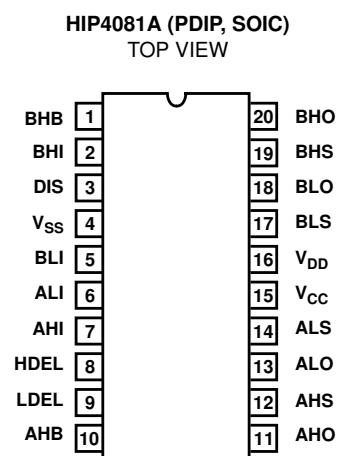
A similar part, the HIP4080A, includes an on-chip input comparator to create a PWM signal from an external triangle wave and to facilitate "hysteresis mode" switching.

The Application Note for the HIP4081A is the AN9405.

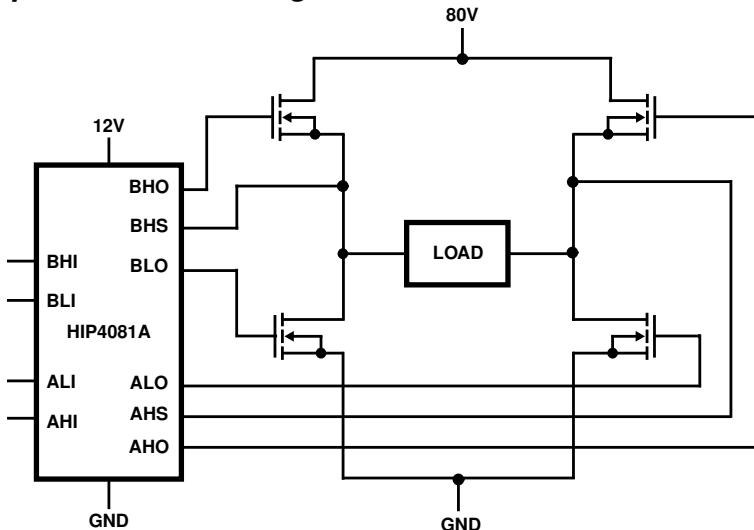
Ordering Information

PART NUMBER	TEMP RANGE (°C)	PACKAGE	PKG. NO.
HIP4081AIP	-40 to 85	20 Ld PDIP	E20.3
HIP4081AIB	-40 to 85	20 Ld SOIC (W)	M20.3

Pinout

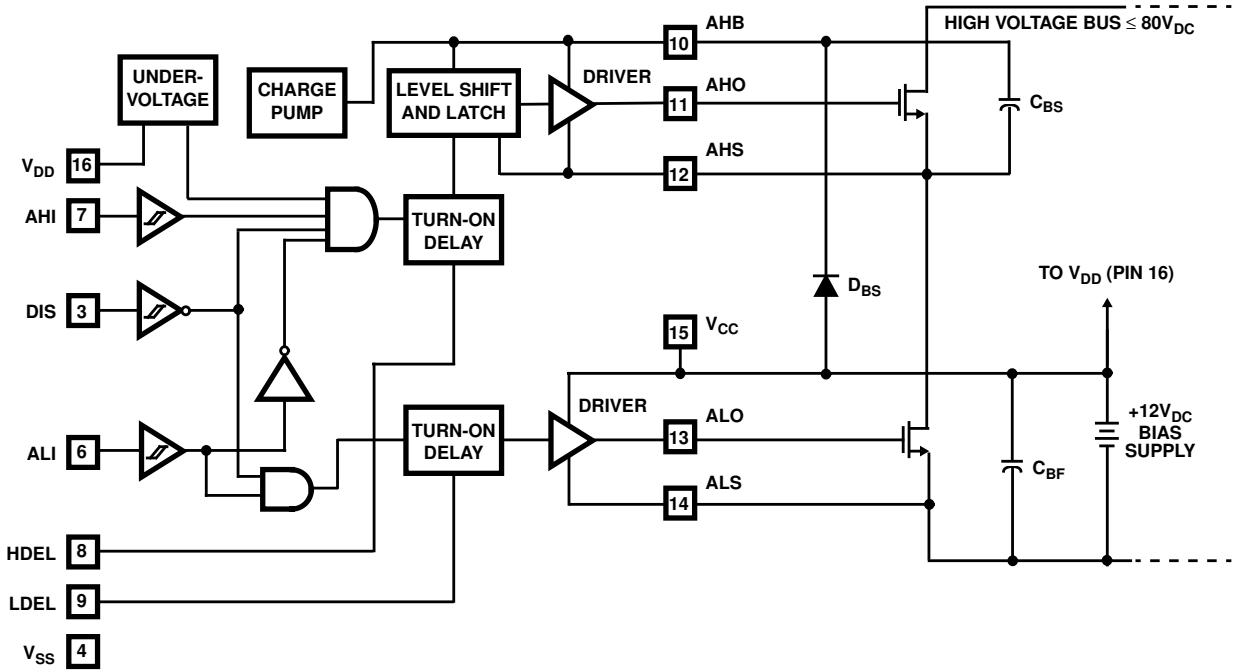


Application Block Diagram

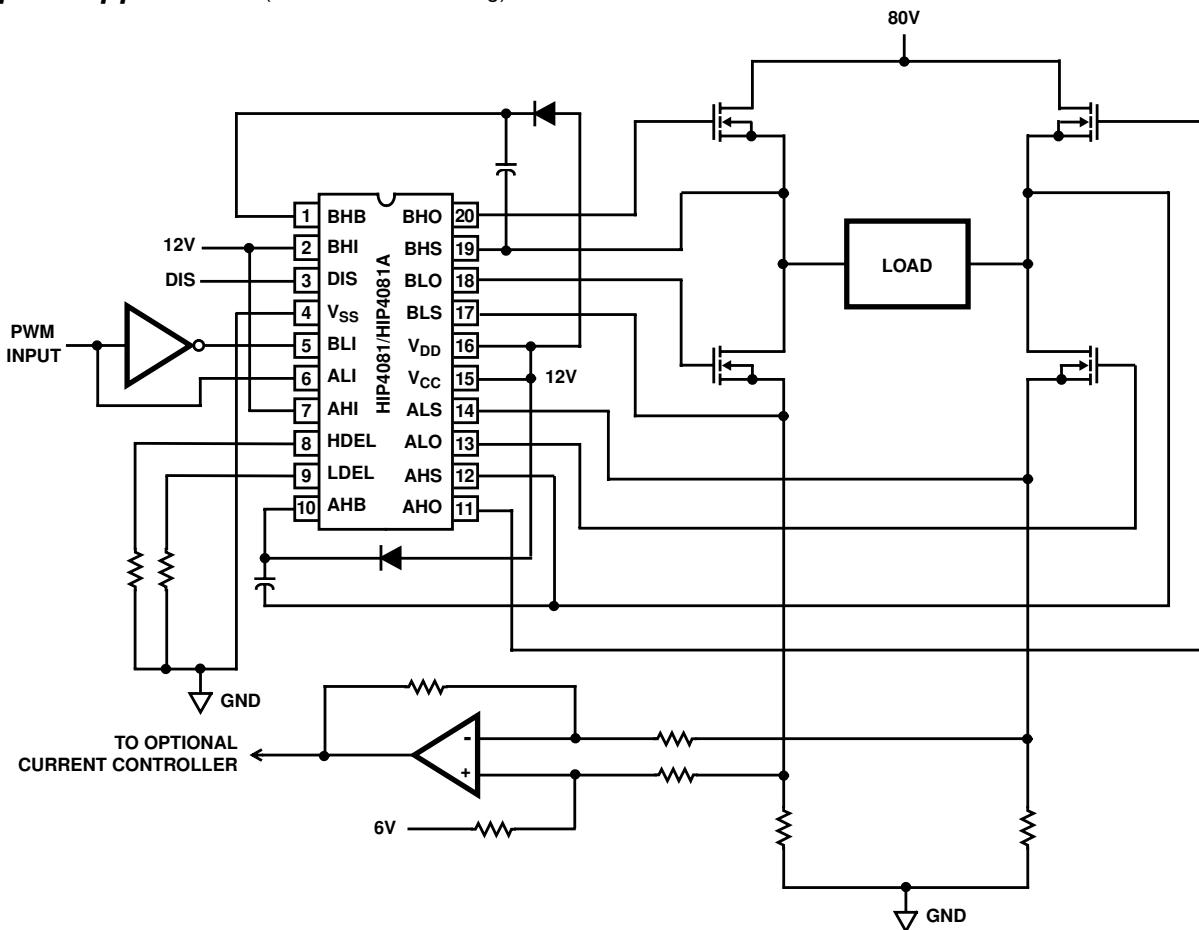


HIP4081A

Functional Block Diagram (1/2 HIP4081A)



Typical Application (PWM Mode Switching)



HIP4081A

Absolute Maximum Ratings

Supply Voltage, V_{DD} and V_{CC}	-0.3V to 16V
Logic I/O Voltages	-0.3V to V_{DD} +0.3V
Voltage on AHS, BHS	-6.0V (Transient) to 80V (25°C to 125°C)
Voltage on ALS, BLS	-6.0V (Transient) to 70V (-55°C to 125°C)
Voltage on ALS, BLS	-2.0V (Transient) to +2.0V (Transient)
Voltage on AHB, BHB	V_{AHS}, BHS -0.3V to V_{AHS}, BHS + V_{DD}
Voltage on ALO, BLO	V_{ALS}, BLS -0.3V to V_{CC} +0.3V
Voltage on AHO, BHO	V_{AHS}, BHS -0.3V to V_{AHO}, BHO +0.3V
Input Current, HDEL and LDEL	-5mA to 0mA
Phase Slew Rate	20V/ns

NOTE: All Voltages relative to V_{SS} , unless otherwise specified.

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

Supply Voltage, V_{DD} and V_{CC}	+9.5V to +15V	Input Current, HDEL and LDEL	-500μA to -50μA
Voltage on ALS, BLS	-1.0V to +1.0V	Operating Ambient Temperature Range	-40°C to 85°C
Voltage on AHB, BHB	V_{AHS}, BHS +5V to V_{AHS}, BHS +15V		

Electrical Specifications $V_{DD} = V_{CC} = V_{AHS} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 100K$ and $T_A = 25^\circ C$, Unless Otherwise Specified

PARAMETER	SYMBOL	TEST CONDITIONS	$T_J = 25^\circ C$			$T_{JS} = -40^\circ C$ TO $125^\circ C$		UNITS
			MIN	TYP	MAX	MIN	MAX	
SUPPLY CURRENTS AND CHARGE PUMPS								
V_{DD} Quiescent Current	I_{DD}	All inputs = 0V	8.5	10.5	14.5	7.5	14.5	mA
V_{DD} Operating Current	I_{DDO}	Outputs switching $f = 500kHz$	9.5	12.5	15.5	8.5	15.5	mA
V_{CC} Quiescent Current	I_{CC}	All Inputs = 0V, $I_{ALO} = I_{BLO} = 0$	-	0.1	10	-	20	μA
V_{CC} Operating Current	I_{CCO}	$f = 500kHz$, No Load	1	1.25	2.0	0.8	3	mA
AHB, BHB Quiescent Current - Qpump Output Current	I_{AHB}, I_{BHB}	All Inputs = 0V, $I_{AHO} = I_{BHO} = 0$ $V_{DD} = V_{CC} = V_{AHS} = V_{BHB} = 10V$	-50	-30	-11	-60	-10	μA
AHB, BHB Operating Current	I_{AHBO}, I_{BHBO}	$f = 500kHz$, No Load	0.6	1.2	1.5	0.5	1.9	mA
AHS, BHS, AHB, BHB Leakage Current	I_{HLK}	$V_{BHS} = V_{AHS} = 80V$, $V_{AHS} = V_{BHB} = 93V$	-	0.02	1.0	-	10	μA
AHB-AHS, BHB-BHS Qpump Output Voltage	$V_{AHS}-V_{AHS}$ $V_{BHB}-V_{BHS}$	$I_{AHB} = I_{AHS} = 0$, No Load	11.5	12.6	14.0	10.5	14.5	V
INPUT PINS: ALI, BLI, AHI, BHI, AND DIS								
Low Level Input Voltage	V_{IL}	Full Operating Conditions	-	-	1.0	-	0.8	V
High Level Input Voltage	V_{IH}	Full Operating Conditions	2.5	-	-	2.7	-	V
Input Voltage Hysteresis			-	35	-	-	-	mV
Low Level Input Current	I_{IL}	$V_{IN} = 0V$, Full Operating Conditions	-130	-100	-75	-135	-65	μA
High Level Input Current	I_{IH}	$V_{IN} = 5V$, Full Operating Conditions	-1	-	+1	-10	+10	μA
TURN-ON DELAY PINS: LDEL AND HDEL								
LDEL, HDEL Voltage	V_{HDEL}, V_{LDEL}	$I_{HDEL} = I_{LDEL} = -100\mu A$	4.9	5.1	5.3	4.8	5.4	V
GATE DRIVER OUTPUT PINS: ALO, BLO, AHO, AND BHO								
Low Level Output Voltage	V_{OL}	$I_{OUT} = 100mA$	0.7	0.85	1.0	0.5	1.1	V
High Level Output Voltage	$V_{CC}-V_{OH}$	$I_{OUT} = -100mA$	0.8	0.95	1.1	0.5	1.2	V
Peak Pullup Current	I_{O+}	$V_{OUT} = 0V$	1.7	2.6	3.8	1.4	4.1	A
Peak Pulldown Current	I_{O-}	$V_{OUT} = 12V$	1.7	2.4	3.3	1.3	3.6	A

HIP4081A

Electrical Specifications $V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 100K$ and $T_A = 25^\circ C$, Unless Otherwise Specified (**Continued**)

PARAMETER	SYMBOL	TEST CONDITIONS	$T_J = 25^\circ C$			$T_{JS} = -40^\circ C$ TO $125^\circ C$		UNITS
			MIN	TYP	MAX	MIN	MAX	
Undervoltage, Rising Threshold	UV+		8.1	8.8	9.4	8.0	9.5	V
Undervoltage, Falling Threshold	UV-		7.6	8.3	8.9	7.5	9.0	V
Undervoltage, Hysteresis	HYS		0.25	0.4	0.65	0.2	0.7	V

Switching Specifications $V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 10K$, $C_L = 1000pF$.

PARAMETER	SYMBOL	TEST CONDITIONS	$T_J = 25^\circ C$			$T_{JS} = -40^\circ C$ TO $125^\circ C$		UNITS
			MIN	TYP	MAX	MIN	MAX	
Lower Turn-off Propagation Delay (ALI-ALO, BLI-BLO)	T_{LPHL}		-	30	60	-	80	ns
Upper Turn-off Propagation Delay (AHI-AHO, BHI-BHO)	T_{HPHL}		-	35	70	-	90	ns
Lower Turn-on Propagation Delay (ALI-ALO, BLI-BLO)	T_{LPLH}	$R_{HDEL} = R_{LDEL} = 10K$	-	45	70	-	90	ns
Upper Turn-on Propagation Delay (AHI-AHO, BHI-BHO)	T_{HPLH}	$R_{HDEL} = R_{LDEL} = 10K$	-	60	90	-	110	ns
Rise Time	T_R		-	10	25	-	35	ns
Fall Time	T_F		-	10	25	-	35	ns
Turn-on Input Pulse Width	$T_{PWON-ON}$	$R_{HDEL} = R_{LDEL} = 10K$	50	-	-	50	-	ns
Turn-off Input Pulse Width	$T_{PWON-OFF}$	$R_{HDEL} = R_{LDEL} = 10K$	40	-	-	40	-	ns
Turn-on Output Pulse Width	$T_{PWOUT-ON}$	$R_{HDEL} = R_{LDEL} = 10K$	40	-	-	40	-	ns
Turn-off Output Pulse Width	$T_{PWOUT-OFF}$	$R_{HDEL} = R_{LDEL} = 10K$	30	-	-	30	-	ns
Disable Turn-off Propagation Delay (DIS - Lower Outputs)	T_{DISLOW}		-	45	75	-	95	ns
Disable Turn-off Propagation Delay (DIS - Upper Outputs)	$T_{DISHIGH}$		-	55	85	-	105	ns
Disable to Lower Turn-on Propagation Delay (DIS - ALO and BLO)	T_{DLPLH}		-	40	70	-	90	ns
Refresh Pulse Width (ALO and BLO)	T_{REF-PW}		240	410	550	200	600	ns
Disable to Upper Enable (DIS - AHO and BHO)	T_{UEN}		-	450	620	-	690	ns

TRUTH TABLE

INPUT				OUTPUT		
ALI, BLI	AHI, BHI	U/V	DIS	ALO, BLO	AHO, BHO	
X	X	X	1	0	0	
1	X	0	0	1	0	
0	1	0	0	0	1	
0	0	0	0	0	0	
X	X	1	X	0	0	

NOTE: X signifies that input can be either a "1" or "0".

HIP4081A

Pin Descriptions

PIN NUMBER	SYMBOL	DESCRIPTION
1	BHB	B High-side Bootstrap supply. External bootstrap diode and capacitor are required. Connect cathode of bootstrap diode and positive side of bootstrap capacitor to this pin. Internal charge pump supplies 30µA out of this pin to maintain bootstrap supply. Internal circuitry clamps the bootstrap supply to approximately 12.8V.
2	BHI	B High-side Input. Logic level input that controls BHO driver (Pin 20). BLI (Pin 5) high level input overrides BHI high level input to prevent half-bridge shoot-through, see Truth Table. DIS (Pin 3) high level input overrides BHI high level input. The pin can be driven by signal levels of 0V to 15V (no greater than V_{DD}). An internal 100µA pull-up to V_{DD} will hold BHI high, so no connection is required if high-side and low-side outputs are to be controlled by the low-side input.
3	DIS	DISable input. Logic level input that when taken high sets all four outputs low. DIS high overrides all other inputs. When DIS is taken low the outputs are controlled by the other inputs. The pin can be driven by signal levels of 0V to 15V (no greater than V_{DD}). An internal 100µA pull-up to V_{DD} will hold DIS high if this pin is not driven.
4	V_{SS}	Chip negative supply, generally will be ground.
5	BLI	B Low-side Input. Logic level input that controls BLO driver (Pin 18). If BHI (Pin 2) is driven high or not connected externally then BLI controls both BLO and BHO drivers, with dead time set by delay currents at HDEL and LDEL (Pin 8 and 9). DIS (Pin 3) high level input overrides BLI high level input. The pin can be driven by signal levels of 0V to 15V (no greater than V_{DD}). An internal 100µA pull-up to V_{DD} will hold BLI high if this pin is not driven.
6	ALI	A Low-side Input. Logic level input that controls ALO driver (Pin 13). If AHI (Pin 7) is driven high or not connected externally then ALI controls both ALO and AHO drivers, with dead time set by delay currents at HDEL and LDEL (Pin 8 and 9). DIS (Pin 3) high level input overrides ALI high level input. The pin can be driven by signal levels of 0V to 15V (no greater than V_{DD}). An internal 100µA pull-up to V_{DD} will hold ALI high if this pin is not driven.
7	AHI	A High-side Input. Logic level input that controls AHO driver (Pin 11). ALI (Pin 6) high level input overrides AHI high level input to prevent half-bridge shoot-through, see Truth Table. DIS (Pin 3) high level input overrides AHI high level input. The pin can be driven by signal levels of 0V to 15V (no greater than V_{DD}). An internal 100µA pull-up to V_{DD} will hold AHI high, so no connection is required if high-side and low-side outputs are to be controlled by the low-side input.
8	HDEL	High-side turn-on DELay. Connect resistor from this pin to V_{SS} to set timing current that defines the turn-on delay of both high-side drivers. The low-side drivers turn-off with no adjustable delay, so the HDEL resistor guarantees no shoot-through by delaying the turn-on of the high-side drivers. HDEL reference voltage is approximately 5.1V.
9	LDEL	Low-side turn-on DELay. Connect resistor from this pin to V_{SS} to set timing current that defines the turn-on delay of both low-side drivers. The high-side drivers turn-off with no adjustable delay, so the LDEL resistor guarantees no shoot-through by delaying the turn-on of the low-side drivers. LDEL reference voltage is approximately 5.1V.
10	AHB	A High-side Bootstrap supply. External bootstrap diode and capacitor are required. Connect cathode of bootstrap diode and positive side of bootstrap capacitor to this pin. Internal charge pump supplies 30µA out of this pin to maintain bootstrap supply. Internal circuitry clamps the bootstrap supply to approximately 12.8V.
11	AHO	A High-side Output. Connect to gate of A High-side power MOSFET.
12	AHS	A High-side Source connection. Connect to source of A High-side power MOSFET. Connect negative side of bootstrap capacitor to this pin.
13	ALO	A Low-side Output. Connect to gate of A Low-side power MOSFET.
14	ALS	A Low-side Source connection. Connect to source of A Low-side power MOSFET.
15	V_{CC}	Positive supply to gate drivers. Must be same potential as V_{DD} (Pin 16). Connect to anodes of two bootstrap diodes.
16	V_{DD}	Positive supply to lower gate drivers. Must be same potential as V_{CC} (Pin 15). De-couple this pin to V_{SS} (Pin 4).
17	BLS	B Low-side Source connection. Connect to source of B Low-side power MOSFET.
18	BLO	B Low-side Output. Connect to gate of B Low-side power MOSFET.
19	BHS	B High-side Source connection. Connect to source of B High-side power MOSFET. Connect negative side of bootstrap capacitor to this pin.
20	BHO	B High-side Output. Connect to gate of B High-side power MOSFET.

Timing Diagrams

X = A OR B, A AND B HALVES OF BRIDGE CONTROLLER ARE INDEPENDENT

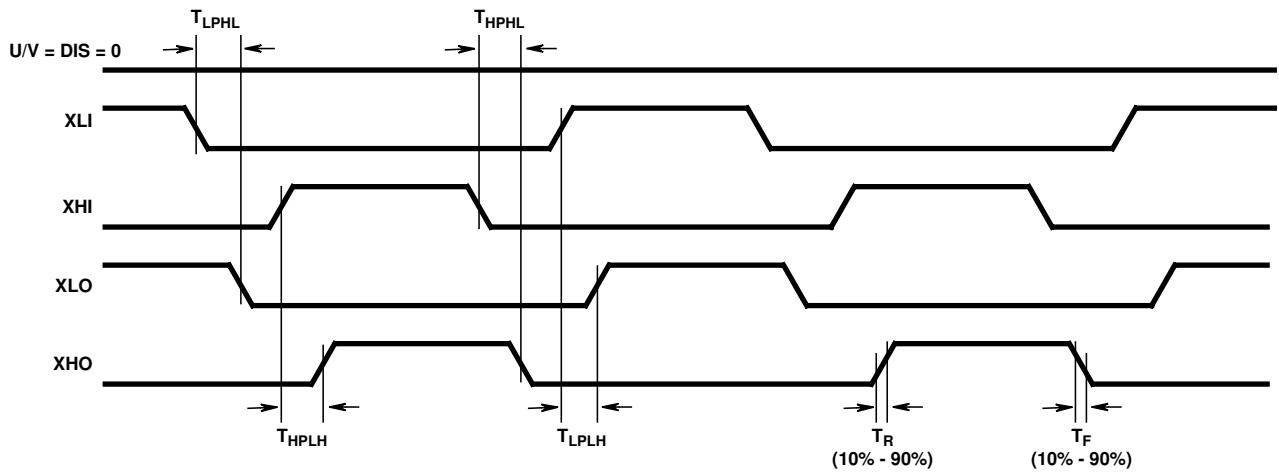


FIGURE 1. INDEPENDENT MODE

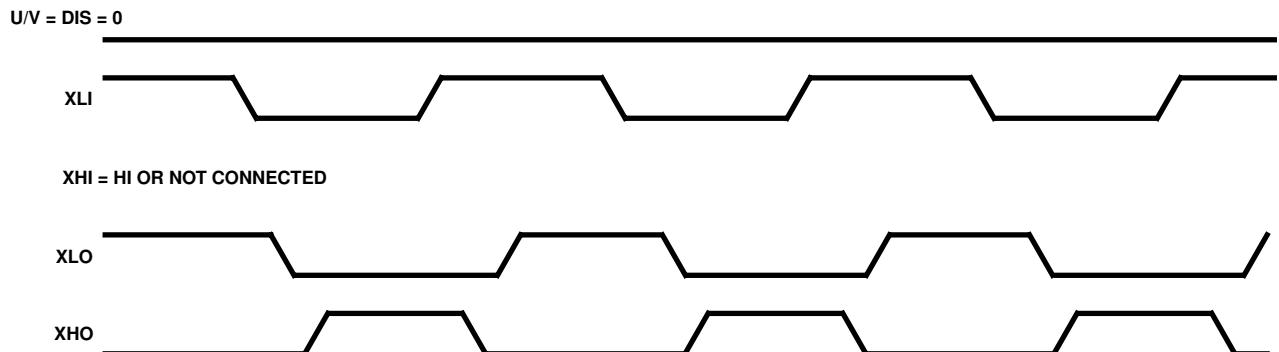


FIGURE 2. BISTATE MODE

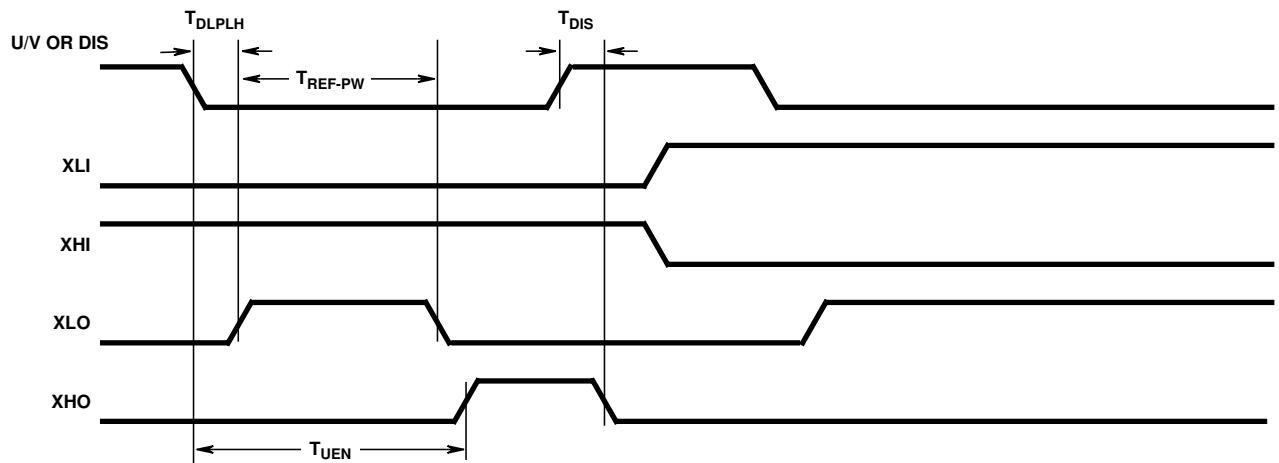


FIGURE 3. DISABLE FUNCTION

HIP4081A

Typical Performance Curves $V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 100K$ and $T_A = 25^{\circ}\text{C}$, Unless Otherwise Specified

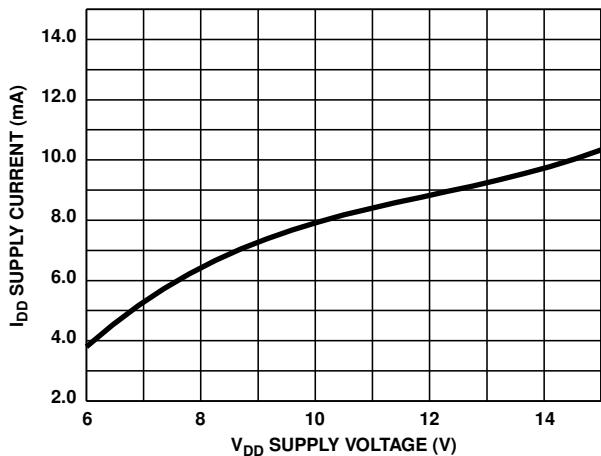


FIGURE 4. QUIESCENT I_{DD} SUPPLY CURRENT vs V_{DD} SUPPLY VOLTAGE

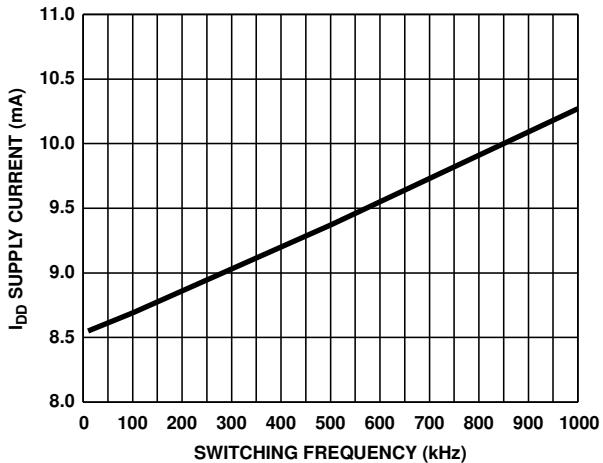


FIGURE 5. I_{DDO} , NO-LOAD I_{DD} SUPPLY CURRENT vs FREQUENCY (kHz)

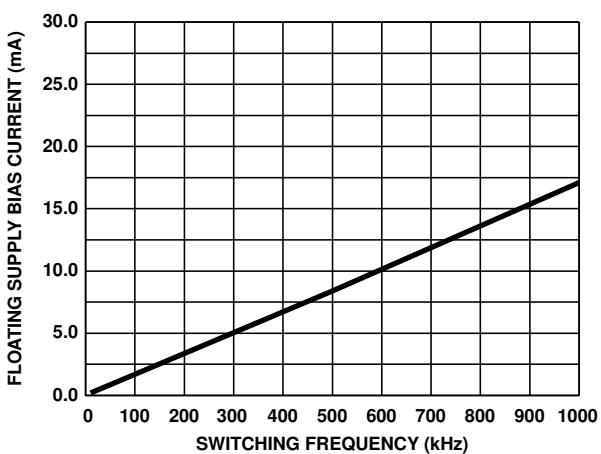


FIGURE 6. SIDE A, B FLOATING SUPPLY BIAS CURRENT vs FREQUENCY (LOAD = 1000pF)

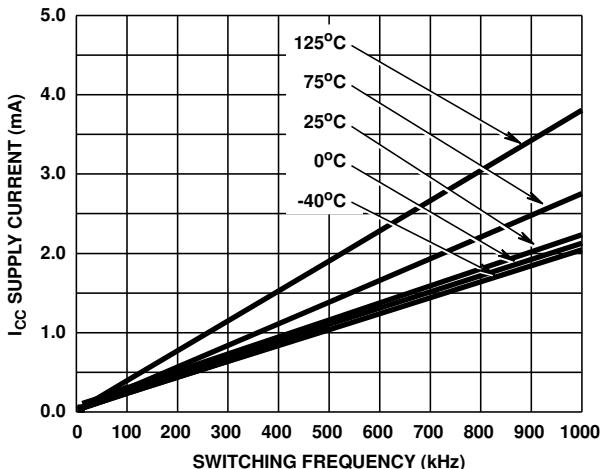


FIGURE 7. I_{CCO} , NO-LOAD I_{CC} SUPPLY CURRENT vs FREQUENCY (kHz) TEMPERATURE

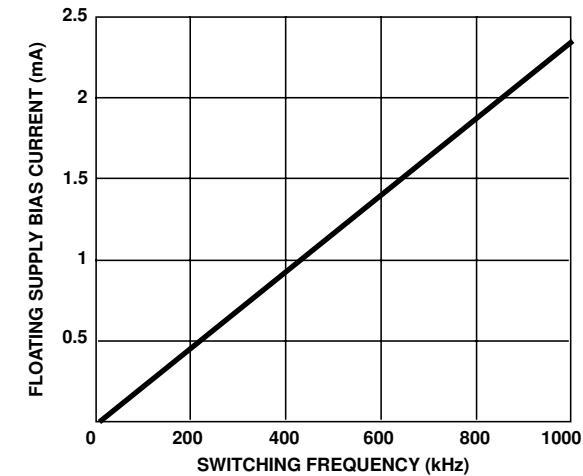


FIGURE 8. I_{AHB}, I_{BHB} , NO-LOAD FLOATING SUPPLY BIAS CURRENT vs FREQUENCY

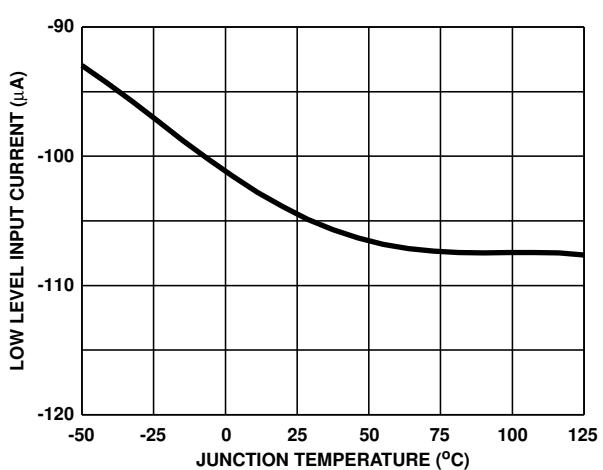


FIGURE 9. ALI, BLI, AHI, BHI LOW LEVEL INPUT CURRENT I_{IL} vs TEMPERATURE

Typical Performance Curves $V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 10K$ and $T_A = 25^{\circ}\text{C}$, Unless Otherwise Specified

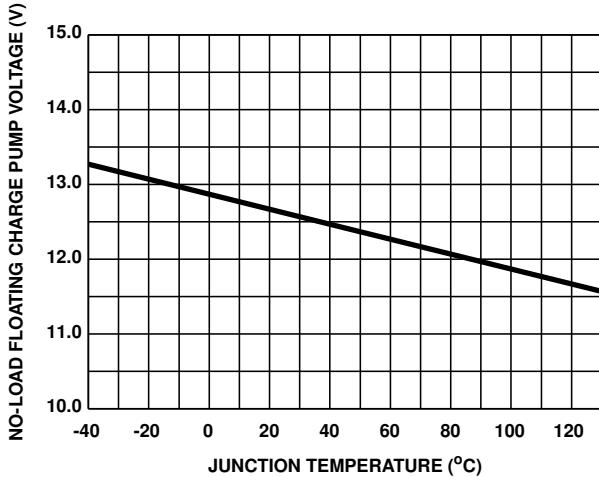


FIGURE 10. AHB - AHS, BHB - BHS NO-LOAD CHARGE PUMP VOLTAGE vs TEMPERATURE

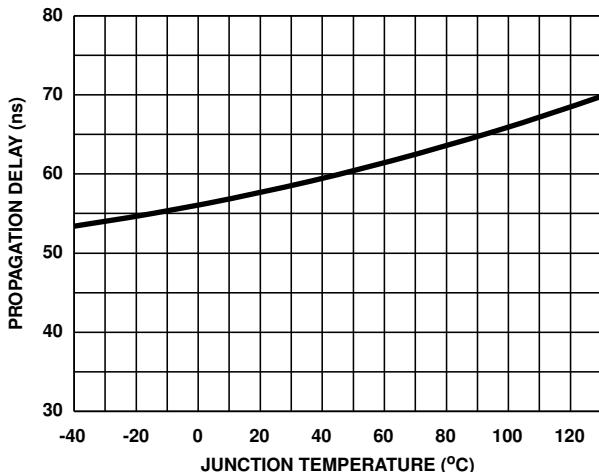


FIGURE 11. UPPER DISABLE TURN-OFF PROPAGATION DELAY $T_{DISHIGH}$ vs TEMPERATURE

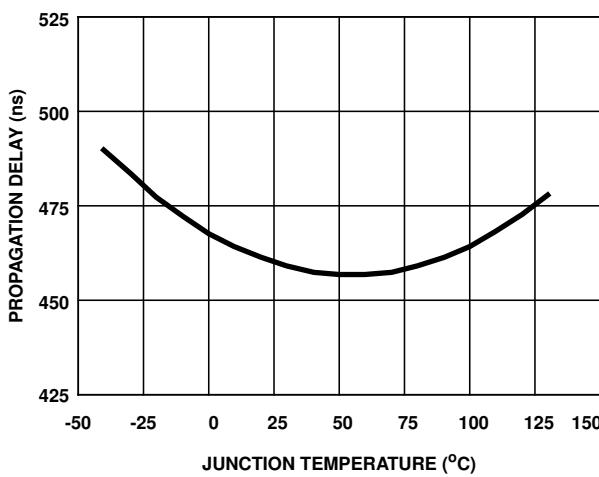


FIGURE 12. DISABLE TO UPPER ENABLE, T_{UEN} , PROPAGATION DELAY vs TEMPERATURE

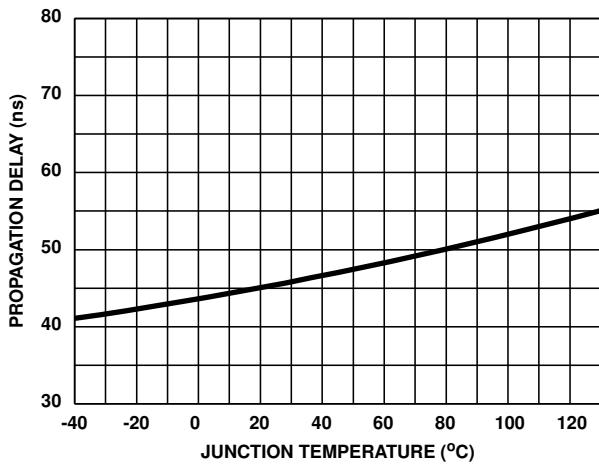


FIGURE 13. LOWER DISABLE TURN-OFF PROPAGATION DELAY T_{DISLOW} vs TEMPERATURE

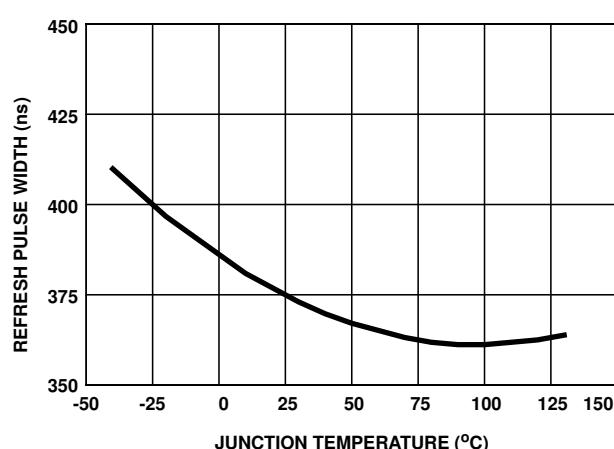


FIGURE 14. T_{REF-PW} REFRESH PULSE WIDTH vs TEMPERATURE

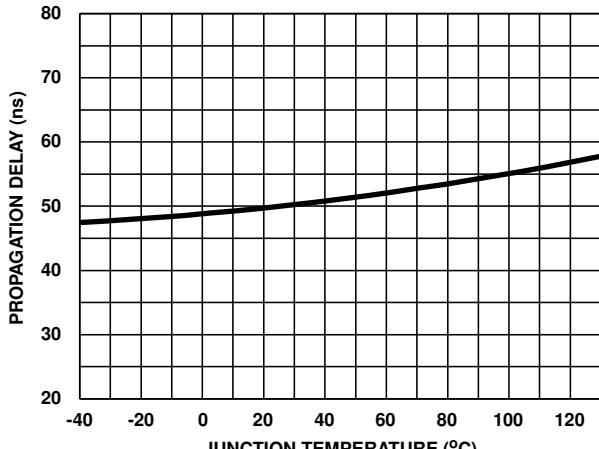


FIGURE 15. DISABLE TO LOWER ENABLE T_{DLPLH} PROPAGATION DELAY vs TEMPERATURE

Typical Performance Curves

$V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 10K$ and $T_A = 25^{\circ}\text{C}$, Unless Otherwise Specified (Continued)

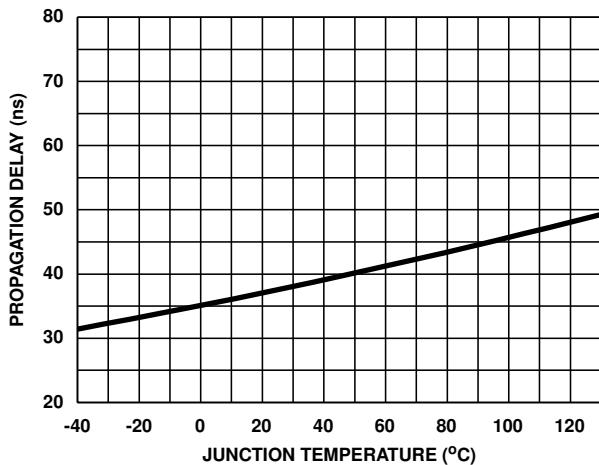


FIGURE 16. UPPER TURN-OFF PROPAGATION DELAY T_{HPLH} VS TEMPERATURE

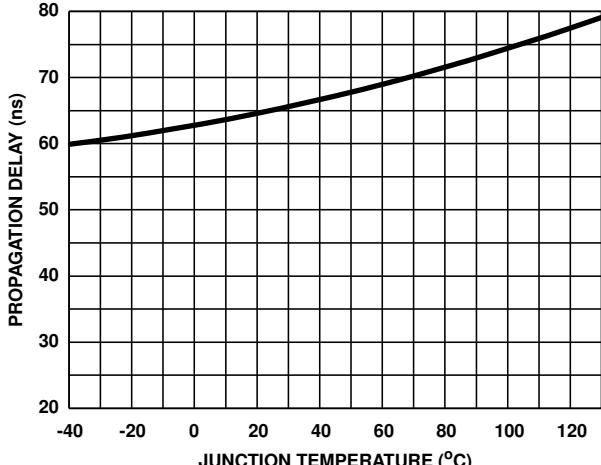


FIGURE 17. UPPER TURN-ON PROPAGATION DELAY T_{HPLH} VS TEMPERATURE

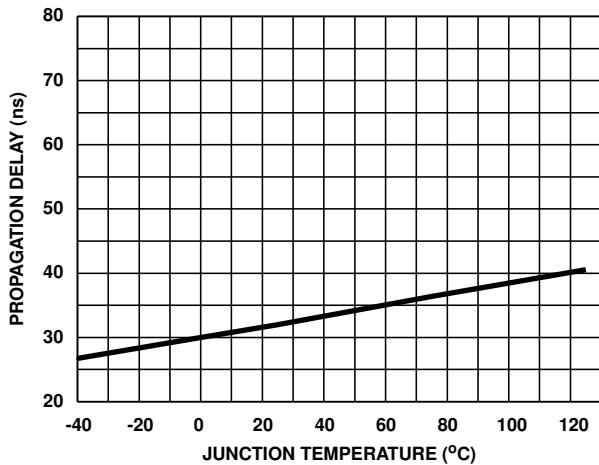


FIGURE 18. LOWER TURN-OFF PROPAGATION DELAY T_{LPHL} VS TEMPERATURE

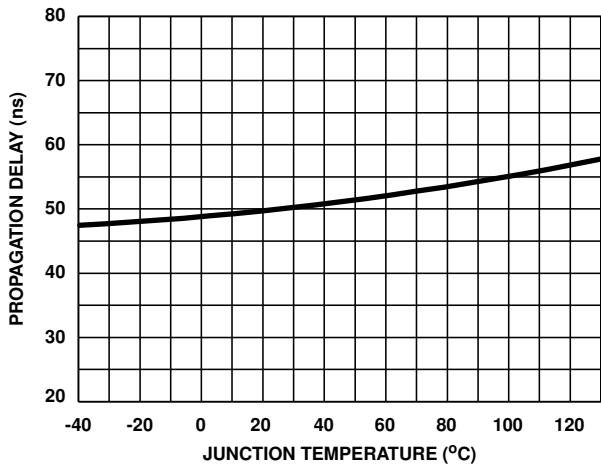


FIGURE 19. LOWER TURN-ON PROPAGATION DELAY T_{LPLH} VS TEMPERATURE

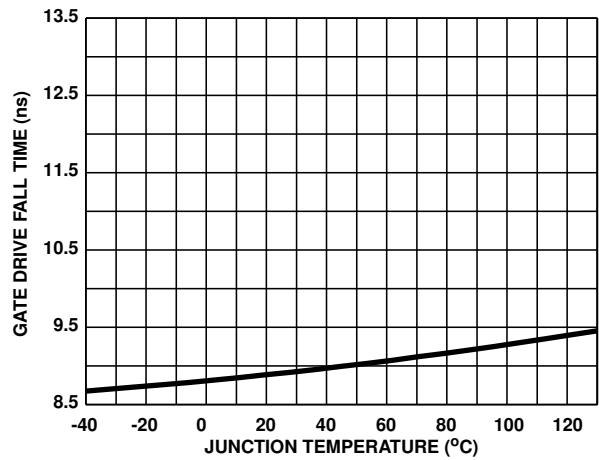


FIGURE 20. GATE DRIVE FALL TIME T_F VS TEMPERATURE

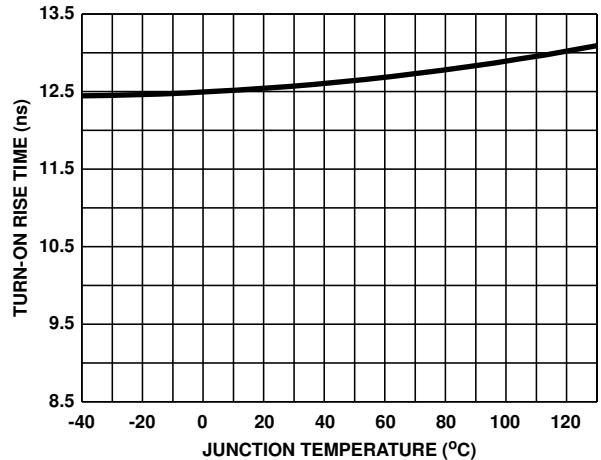


FIGURE 21. GATE DRIVE RISE TIME T_R VS TEMPERATURE

HIP4081A

Typical Performance Curves $V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 100K$ and $T_A = 25^{\circ}\text{C}$, Unless Otherwise Specified

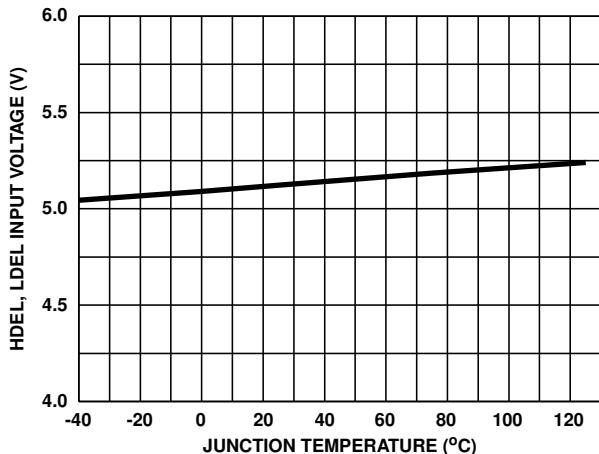


FIGURE 22. V_{LDEL} , V_{HDEL} VOLTAGE vs TEMPERATURE

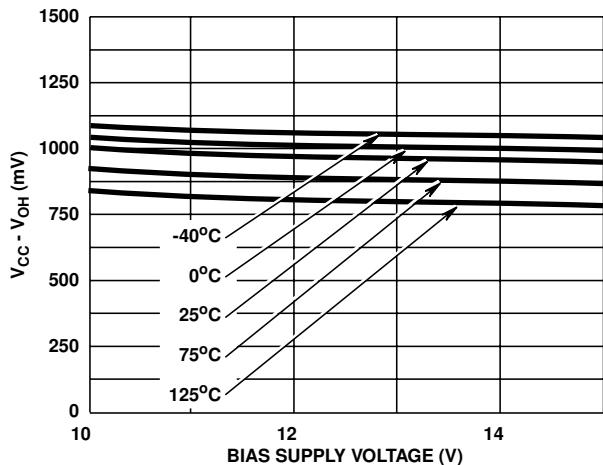


FIGURE 23. HIGH LEVEL OUTPUT VOLTAGE $V_{CC} - V_{OH}$ vs BIAS SUPPLY AND TEMPERATURE AT 100mA

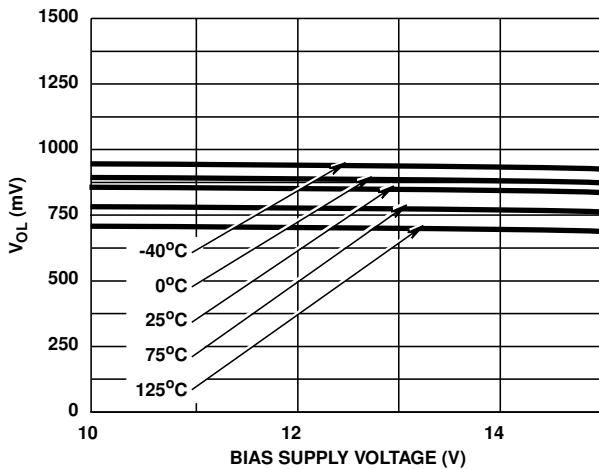


FIGURE 24. LOW LEVEL OUTPUT VOLTAGE V_{OL} vs BIAS SUPPLY AND TEMPERATURE AT 100mA

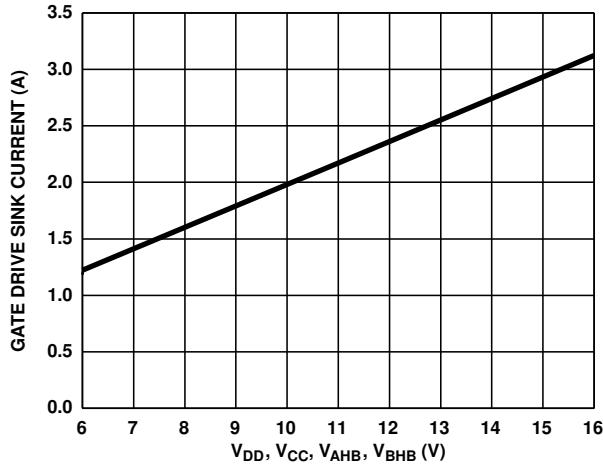


FIGURE 25. PEAK PULLDOWN CURRENT I_O vs BIAS SUPPLY VOLTAGE

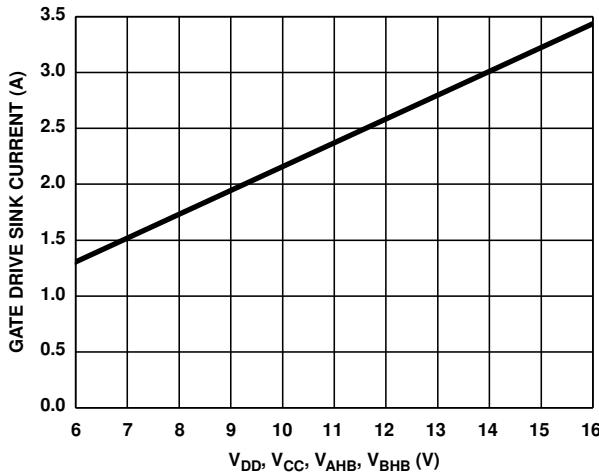


FIGURE 26. PEAK PULLUP CURRENT I_{O+} vs BIAS SUPPLY VOLTAGE

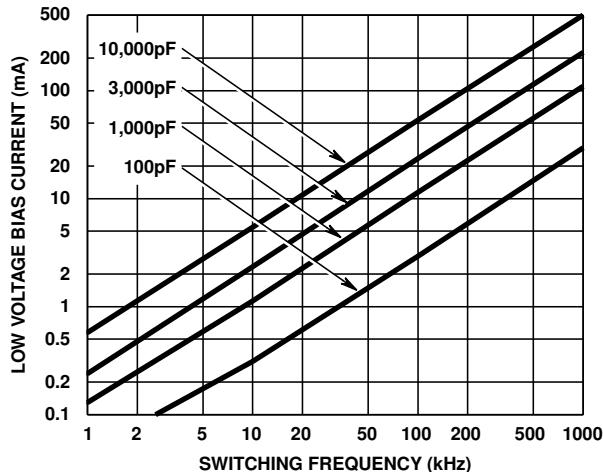


FIGURE 27. LOW VOLTAGE BIAS CURRENT I_{DD} (LESS QUIESCENT COMPONENT) vs FREQUENCY AND GATE LOAD CAPACITANCE

HIP4081A

Typical Performance Curves

$V_{DD} = V_{CC} = V_{AHB} = V_{BHB} = 12V$, $V_{SS} = V_{ALS} = V_{BLS} = V_{AHS} = V_{BHS} = 0V$, $R_{HDEL} = R_{LDEL} = 100K$ and $T_A = 25^{\circ}\text{C}$, Unless Otherwise Specified (Continued)

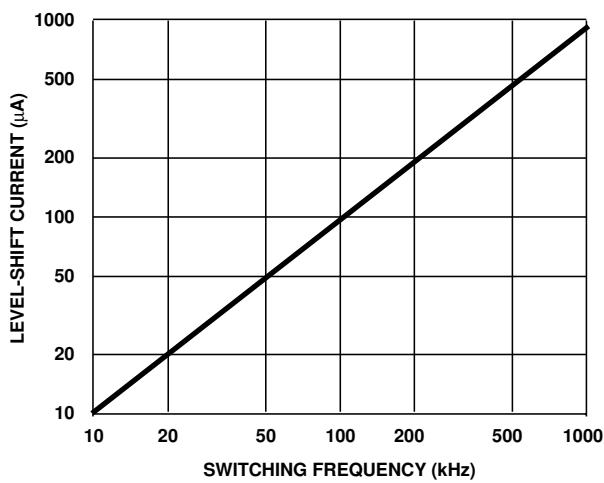


FIGURE 28. HIGH VOLTAGE LEVEL-SHIFT CURRENT vs FREQUENCY AND BUS VOLTAGE

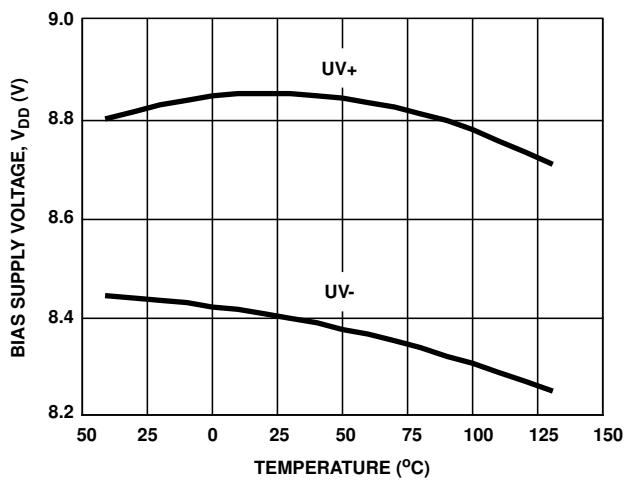


FIGURE 29. UNDERVOLTAGE LOCKOUT vs TEMPERATURE

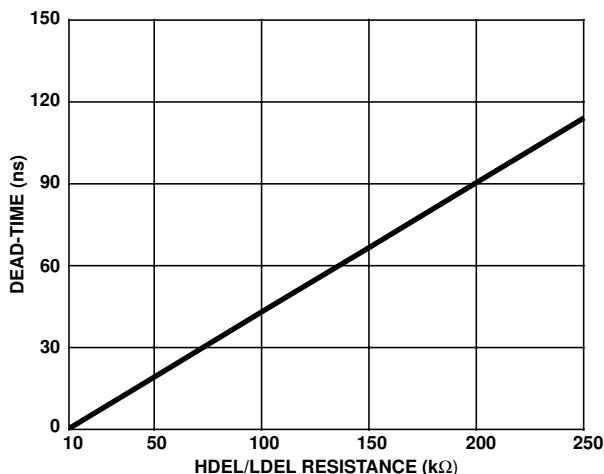


FIGURE 30. MINIMUM DEAD-TIME vs DEL RESISTANCE

HIP4081A

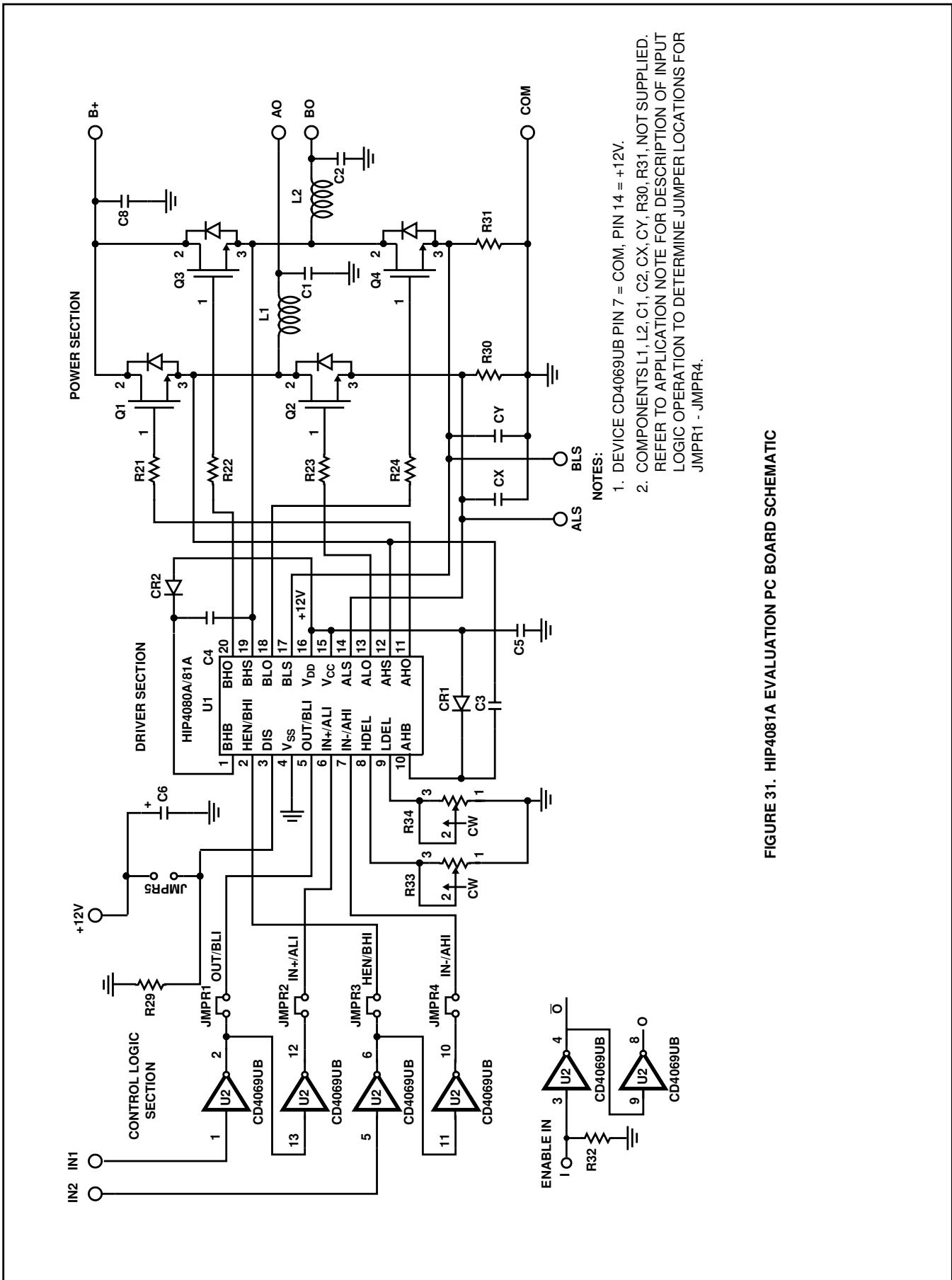


FIGURE 31. HIP4081A EVALUATION PC BOARD SCHEMATIC

HIP4081A

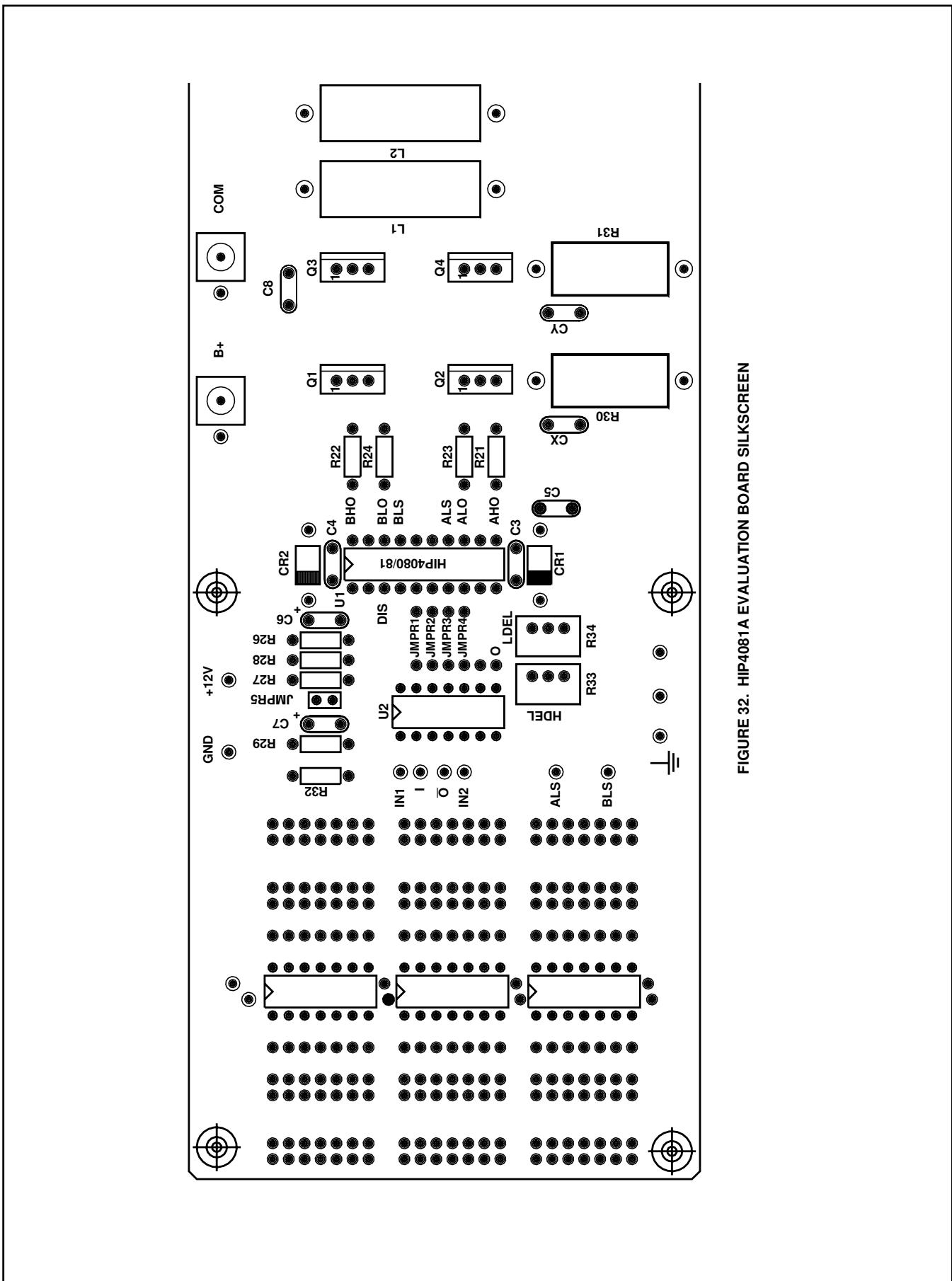
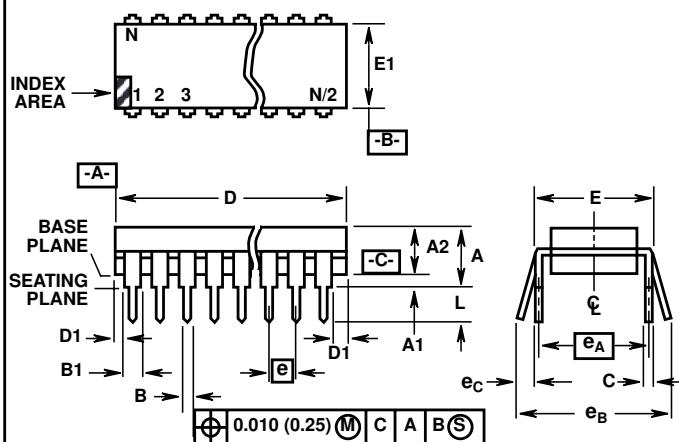


FIGURE 32. HIP4081A EVALUATION BOARD SILKSCREEN

Dual-In-Line Plastic Packages (PDIP)



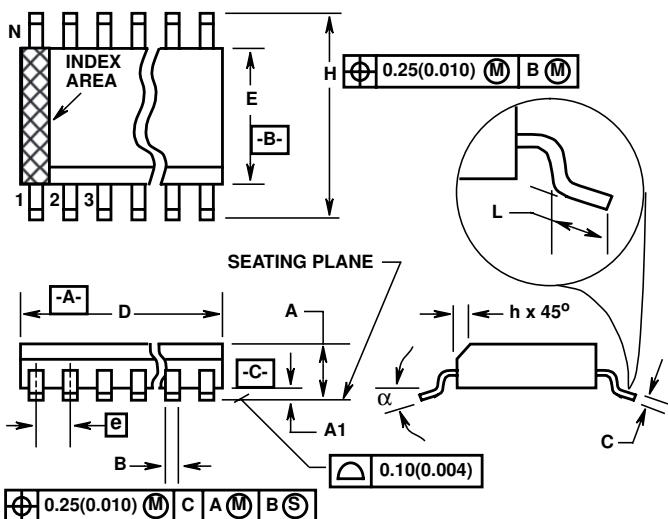
NOTES:

1. Controlling Dimensions: INCH. In case of conflict between English and Metric dimensions, the inch dimensions control.
2. Dimensioning and tolerancing per ANSI Y14.5M-1982.
3. Symbols are defined in the "MO Series Symbol List" in Section 2.2 of Publication No. 95.
4. Dimensions A, A1 and L are measured with the package seated in JEDEC seating plane gauge GS-3.
5. D, D1, and E1 dimensions do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.010 inch (0.25mm).
6. E and e_A are measured with the leads constrained to be perpendicular to datum $-C-$.
7. e_B and e_C are measured at the lead tips with the leads unconstrained. e_C must be zero or greater.
8. B1 maximum dimensions do not include dambar protrusions. Dambar protrusions shall not exceed 0.010 inch (0.25mm).
9. N is the maximum number of terminal positions.
10. Corner leads (1, N, N/2 and N/2 + 1) for E8.3, E16.3, E18.3, E28.3, E42.6 will have a B1 dimension of 0.030 - 0.045 inch (0.76 - 1.14mm).

**E20.3 (JEDEC MS-001-AD ISSUE D)
20 LEAD DUAL-IN-LINE PLASTIC PACKAGE**

SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A	-	0.210	-	5.33	4
A1	0.015	-	0.39	-	4
A2	0.115	0.195	2.93	4.95	-
B	0.014	0.022	0.356	0.558	-
B1	0.045	0.070	1.55	1.77	8
C	0.008	0.014	0.204	0.355	-
D	0.980	1.060	24.89	26.9	5
D1	0.005	-	0.13	-	5
E	0.300	0.325	7.62	8.25	6
E1	0.240	0.280	6.10	7.11	5
e	0.100 BSC		2.54 BSC		-
eA	0.300 BSC		7.62 BSC		6
eB	-	0.430	-	10.92	7
L	0.115	0.150	2.93	3.81	4
N	20		20		9

Rev. 0 12/93

Small Outline Plastic Packages (SOIC)

NOTES:

1. Symbols are defined in the "MO Series Symbol List" in Section 2.2 of Publication Number 95.
2. Dimensioning and tolerancing per ANSI Y14.5M-1982.
3. Dimension "D" does not include mold flash, protrusions or gate burrs. Mold flash, protrusion and gate burrs shall not exceed 0.15mm (0.006 inch) per side.
4. Dimension "E" does not include interlead flash or protrusions. Interlead flash and protrusions shall not exceed 0.25mm (0.010 inch) per side.
5. The chamfer on the body is optional. If it is not present, a visual index feature must be located within the crosshatched area.
6. "L" is the length of terminal for soldering to a substrate.
7. "N" is the number of terminal positions.
8. Terminal numbers are shown for reference only.
9. The lead width "B", as measured 0.36mm (0.014 inch) or greater above the seating plane, shall not exceed a maximum value of 0.61mm (0.024 inch).
10. Controlling dimension: MILLIMETER. Converted inch dimensions are not necessarily exact.

**M20.3 (JEDEC MS-013-AC ISSUE C)
20 LEAD WIDE BODY SMALL OUTLINE PLASTIC PACKAGE**

SYMBOL	INCHES		MILLIMETERS		NOTES
	MIN	MAX	MIN	MAX	
A	0.0926	0.1043	2.35	2.65	-
A1	0.0040	0.0118	0.10	0.30	-
B	0.013	0.0200	0.33	0.51	9
C	0.0091	0.0125	0.23	0.32	-
D	0.4961	0.5118	12.60	13.00	3
E	0.2914	0.2992	7.40	7.60	4
e	0.050 BSC		1.27 BSC		-
H	0.394	0.419	10.00	10.65	-
h	0.010	0.029	0.25	0.75	5
L	0.016	0.050	0.40	1.27	6
N	20		20		7
α	0°	8°	0°	8°	-

Rev. 0 12/93

All Intersil semiconductor products are manufactured, assembled and tested under ISO9000 quality systems certification.

Intersil products are sold by description only. Intersil Corporation reserves the right to make changes in circuit design and/or specifications at any time without notice. Accordingly, the reader is cautioned to verify that data sheets are current before placing orders. Information furnished by Intersil is believed to be accurate and reliable. However, no responsibility is assumed by Intersil or its subsidiaries for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Intersil or its subsidiaries.

For information regarding Intersil Corporation and its products, see web site <http://www.intersil.com>

Sales Office Headquarters**NORTH AMERICA**

Intersil Corporation
P. O. Box 883, Mail Stop 53-204
Melbourne, FL 32902
TEL: (407) 724-7000
FAX: (407) 724-7240

EUROPE

Intersil SA
Mercure Center
100, Rue de la Fusée
1130 Brussels, Belgium
TEL: (32) 2.724.2111
FAX: (32) 2.724.22.05

ASIA

Intersil (Taiwan) Ltd.
Taiwan Limited
7F-6, No. 101 Fu Hsing North Road
Taipei, Taiwan
Republic of China
TEL: (886) 2 2716 9310
FAX: (886) 2 2715 3029

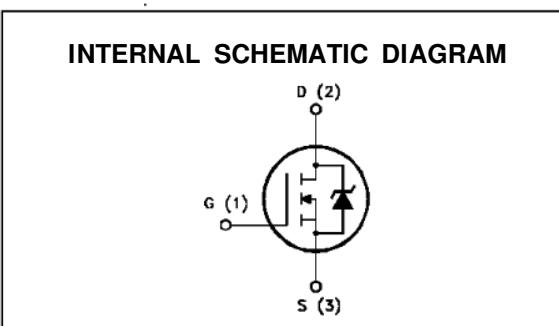
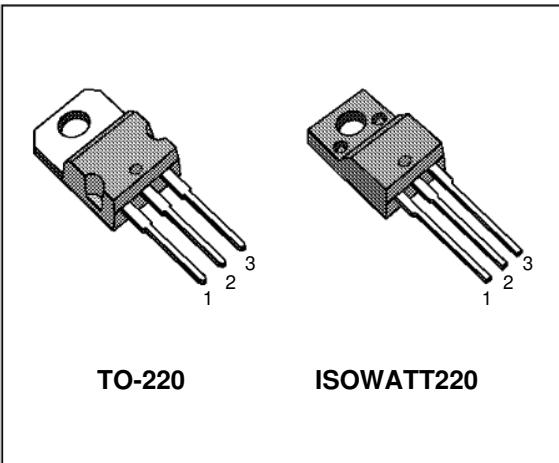
N - CHANNEL ENHANCEMENT MODE
 POWER MOS TRANSISTORS

TYPE	V _{DSS}	R _{DS(on)}	I _D
IRF520	100 V	< 0.27 Ω	10 A
IRF520FI	100 V	< 0.27 Ω	7 A

- TYPICAL R_{DS(on)} = 0.23 Ω
- AVALANCHE RUGGED TECHNOLOGY
- 100% AVALANCHE TESTED
- REPETITIVE AVALANCHE DATA AT 100°C
- LOW GATE CHARGE
- HIGH CURRENT CAPABILITY
- 175°C OPERATING TEMPERATURE

APPLICATIONS

- HIGH CURRENT, HIGH SPEED SWITCHING
- SOLENOID AND RELAY DRIVERS
- REGULATORS
- DC-DC & DC-AC CONVERTERS
- MOTOR CONTROL, AUDIO AMPLIFIERS
- AUTOMOTIVE ENVIRONMENT (INJECTION, ABS, AIR-BAG, LAMPDRIVERS, Etc.)


ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value		Unit
		IRF520	IRF520FI	
V _{DS}	Drain-source Voltage (V _{GS} = 0)	100		V
V _{DGR}	Drain-gate Voltage (R _{GS} = 20 kΩ)	100		V
V _{GS}	Gate-source Voltage	± 20		V
I _D	Drain Current (cont.) at T _c = 25 °C	10	7	A
I _D	Drain Current (cont.) at T _c = 100 °C	7	5	A
I _{DM(•)}	Drain Current (pulsed)	40	40	A
P _{tot}	Total Dissipation at T _c = 25 °C	70	35	W
	Derating Factor	0.47	0.23	W/°C
V _{ISO}	Insulation Withstand Voltage (DC)	—	2000	V
T _{stg}	Storage Temperature	-65 to 175		°C
T _j	Max. Operating Junction Temperature	175		°C

(•) Pulse width limited by safe operating area

IRF520/FI

THERMAL DATA

			TO-220	ISOWATT220	
R _{thj-case}	Thermal Resistance Junction-case	Max	2.14	4.29	°C/W
R _{thc-amb}	Thermal Resistance Junction-ambient	Max	62.5		°C/W
R _{thc-s}	Thermal Resistance Case-sink	Typ	0.5		°C/W
T _I	Maximum Lead Temperature For Soldering Purpose		300		°C

AVALANCHE CHARACTERISTICS

Symbol	Parameter	Max Value	Unit
I _{AR}	Avalanche Current, Repetitive or Not-Repetitive (pulse width limited by T _j max, δ < 1%)	10	A
E _{AS}	Single Pulse Avalanche Energy (starting T _j = 25 °C, I _D = I _{AR} , V _{DD} = 25 V)	36	mJ
E _{AR}	Repetitive Avalanche Energy (pulse width limited by T _j max, δ < 1%)	9	mJ
I _{AR}	Avalanche Current, Repetitive or Not-Repetitive (T _c = 100 °C, pulse width limited by T _j max, δ < 1%)	7	A

ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

OFF

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V _{(BR)DSS}	Drain-source Breakdown Voltage	I _D = 250 μA V _{GS} = 0	100			V
I _{DSS}	Zero Gate Voltage Drain Current (V _{GS} = 0)	V _{DS} = Max Rating V _{DS} = Max Rating × 0.8 T _c = 125 °C			250 1000	μA μA
I _{GSS}	Gate-body Leakage Current (V _{DS} = 0)	V _{GS} = ± 20 V			± 100	nA

ON (*)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V _{GS(th)}	Gate Threshold Voltage	V _{DS} = V _{GS} I _D = 250 μA	2	2.9	4	V
R _{D(on)}	Static Drain-source On Resistance	V _{GS} = 10 V I _D = 5 A		0.23	0.27	Ω
I _{D(on)}	On State Drain Current	V _{DS} > I _{D(on)} × R _{D(on)max} V _{GS} = 10 V	10			A

DYNAMIC

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
g _{fs} (*)	Forward Transconductance	V _{DS} > I _{D(on)} × R _{D(on)max} I _D = 5 A	2.7	4.5		S
C _{iss} C _{oss} C _{rss}	Input Capacitance Output Capacitance Reverse Transfer Capacitance	V _{DS} = 25 V f = 1 MHz V _{GS} = 0		330 90 25	450 120 40	pF pF pF

ELECTRICAL CHARACTERISTICS (continued)

SWITCHING RESISTIVE LOAD

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$t_{d(on)}$ t_r	Turn-on Time Rise Time	$V_{DD} = 50 \text{ V}$ $I_D = 5 \text{ A}$ $R_{GS} = 4.7 \Omega$ $V_{GS} = 10 \text{ V}$		10 50	15 75	ns ns
$t_{d(off)}$ t_f	Turn-off Delay Time Fall Time	(see test circuit)		25 20	40 30	ns ns
Q_g Q_{gs} Q_{gd}	Total Gate Charge Gate-Source Charge Gate-Drain Charge	$I_D = 10 \text{ A}$ $V_{GS} = 10 \text{ V}$ $V_{DD} = \text{Max Rating} \times 0.8$ (see test circuit)		15 7 4	25	nC nC nC

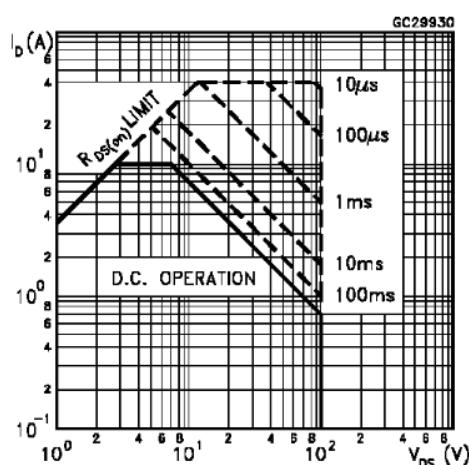
SOURCE DRAIN DIODE

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{SD} $I_{SDM}(\bullet)$	Source-drain Current Source-drain Current (pulsed)				10 40	A A
$V_{SD} (*)$	Forward On Voltage	$I_{SD} = 10 \text{ A}$ $V_{GS} = 0$			1.6	V
t_{rr} Q_{rr}	Reverse Recovery Time Reverse Recovery Charge	$I_{SD} = 10 \text{ A}$ $di/dt = 100 \text{ A}/\mu\text{s}$ $V_{DD} = 20 \text{ V}$ $T_j = 150^\circ\text{C}$		80 0.22		ns μC

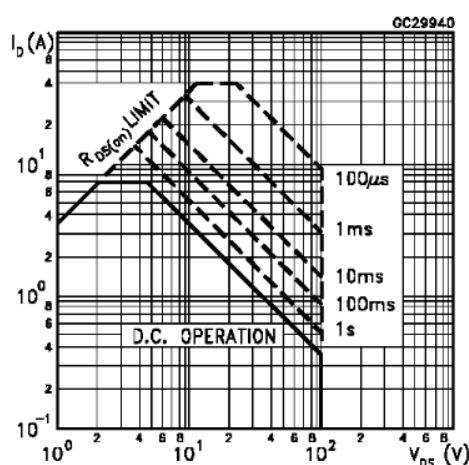
(*) Pulsed: Pulse duration = 300 μs , duty cycle 1.5 %

(•) Pulse width limited by safe operating area

Safe Operating Area for TO-220

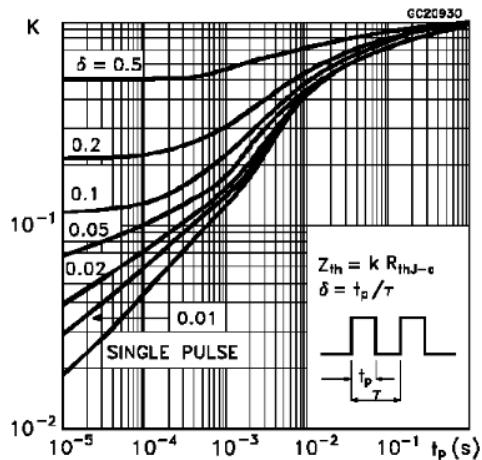


Safe Operating Area for ISOWATT220

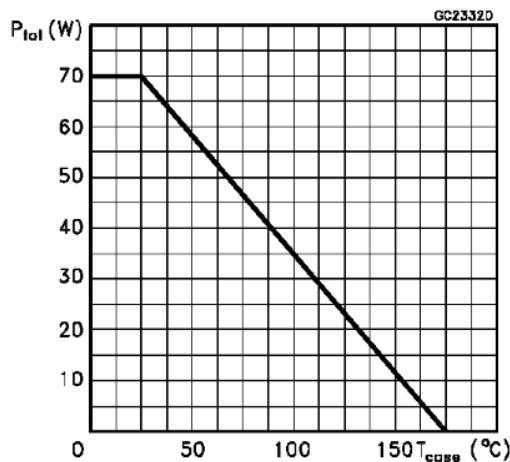


IRF520/FI

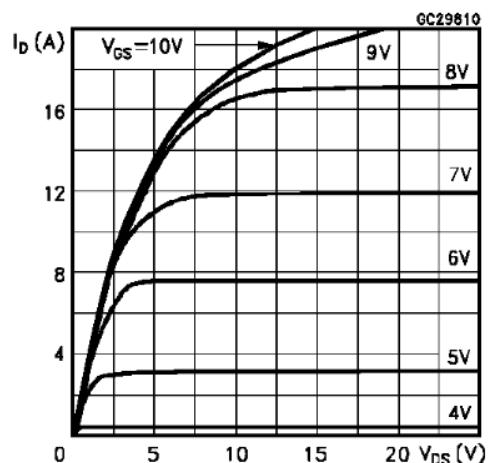
Thermal Impedance for TO-220



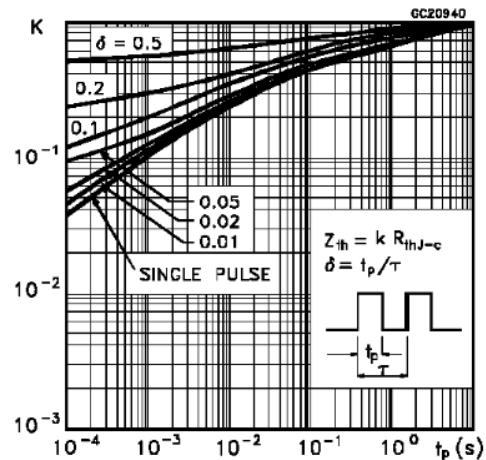
Derating Curve for TO-220



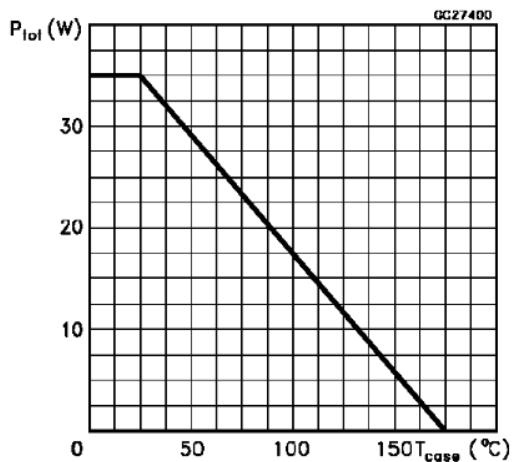
Output Characteristics



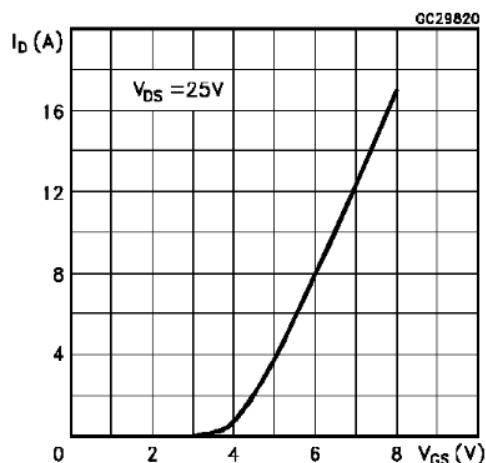
Thermal Impedance for ISOWATT220



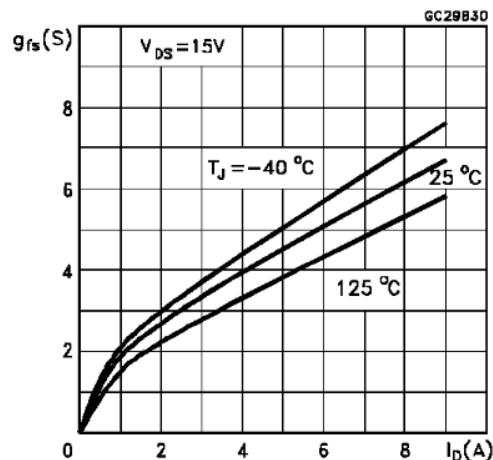
Derating Curve for ISOWATT220



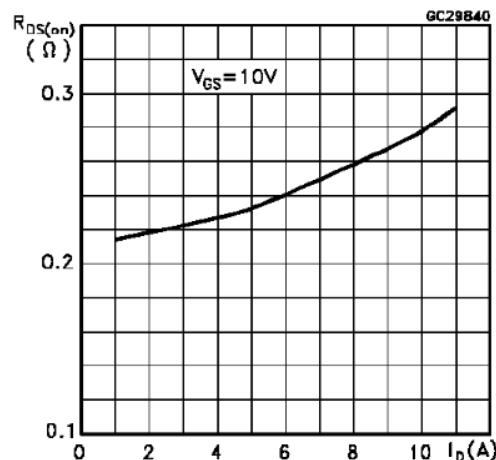
Transfer Characteristics



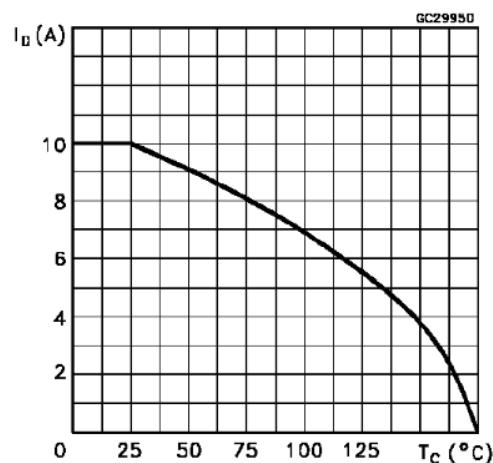
Transconductance



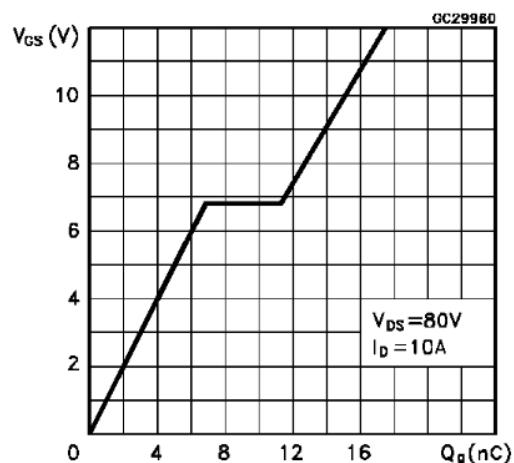
Static Drain-source On Resistance



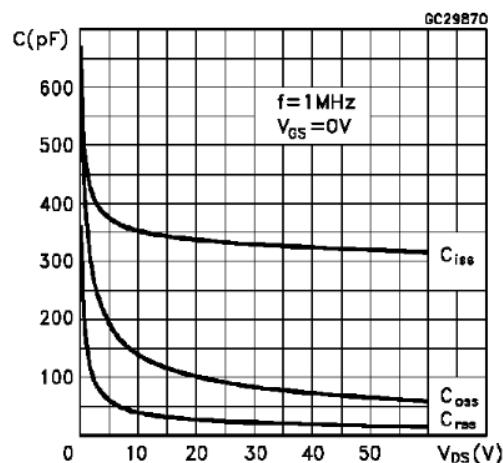
Maximum Drain Current vs Temperature



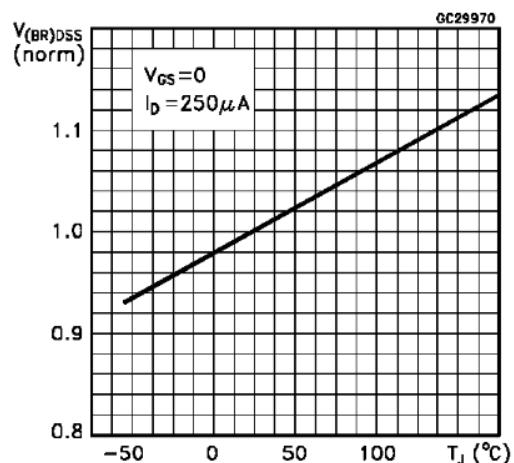
Gate Charge vs Gate-source Voltage



Capacitance Variations

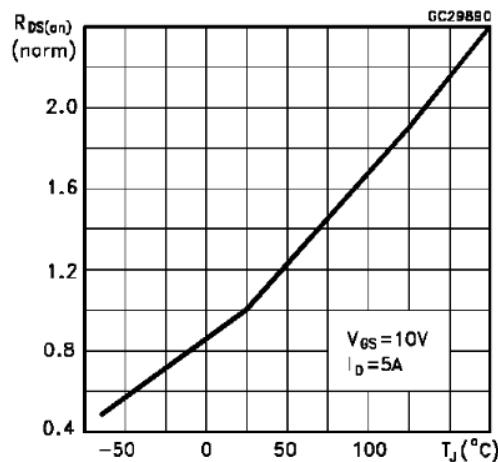


Normalized Breakdown Voltage vs Temperature

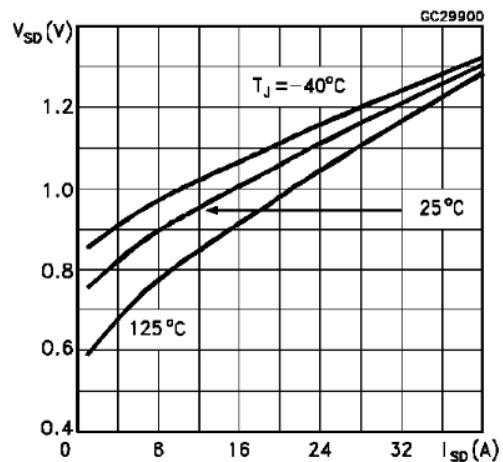


IRF520/FI

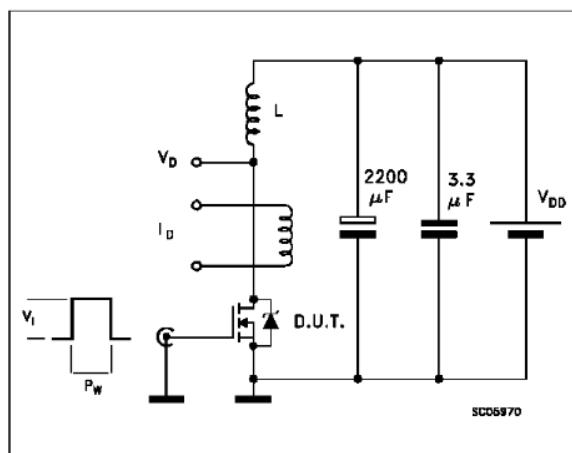
Normalized On Resistance vs Temperature



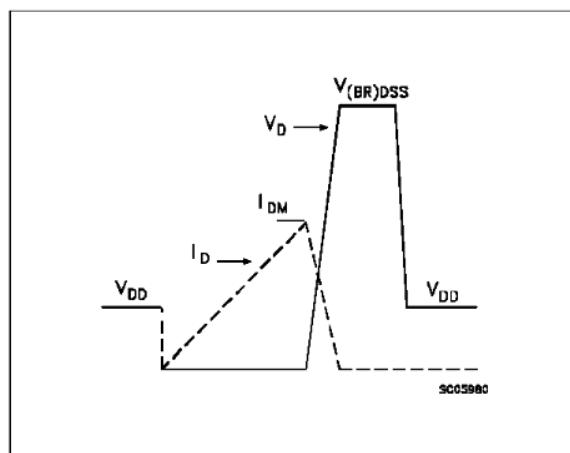
Source-drain Diode Forward Characteristics



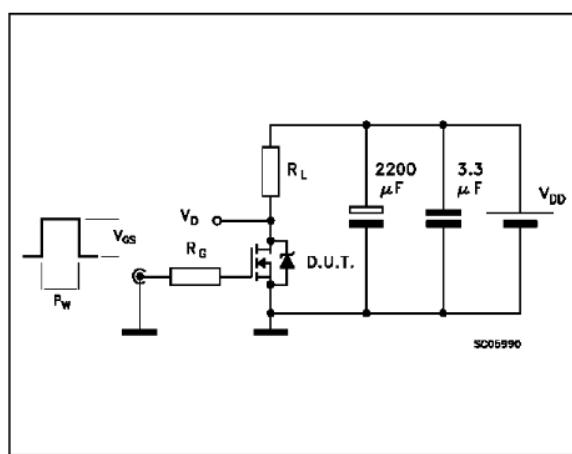
Unclamped Inductive Load Test Circuit



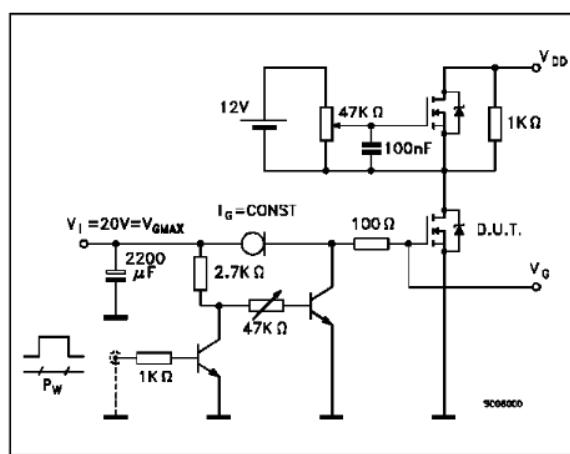
Unclamped Inductive Waveforms



Switching Time Test Circuit

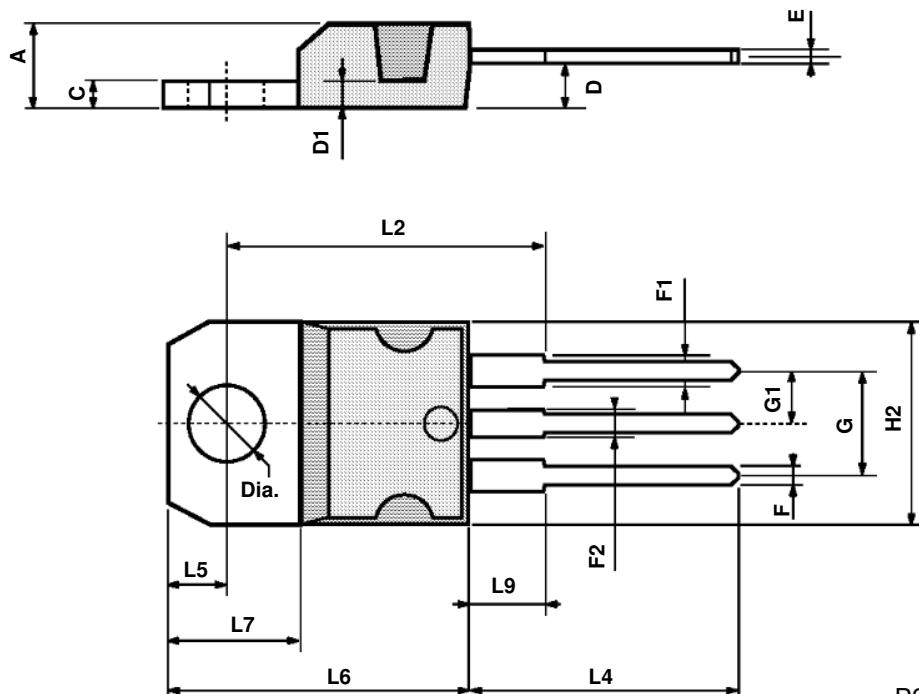


Gate Charge Test Circuit



TO-220 MECHANICAL DATA

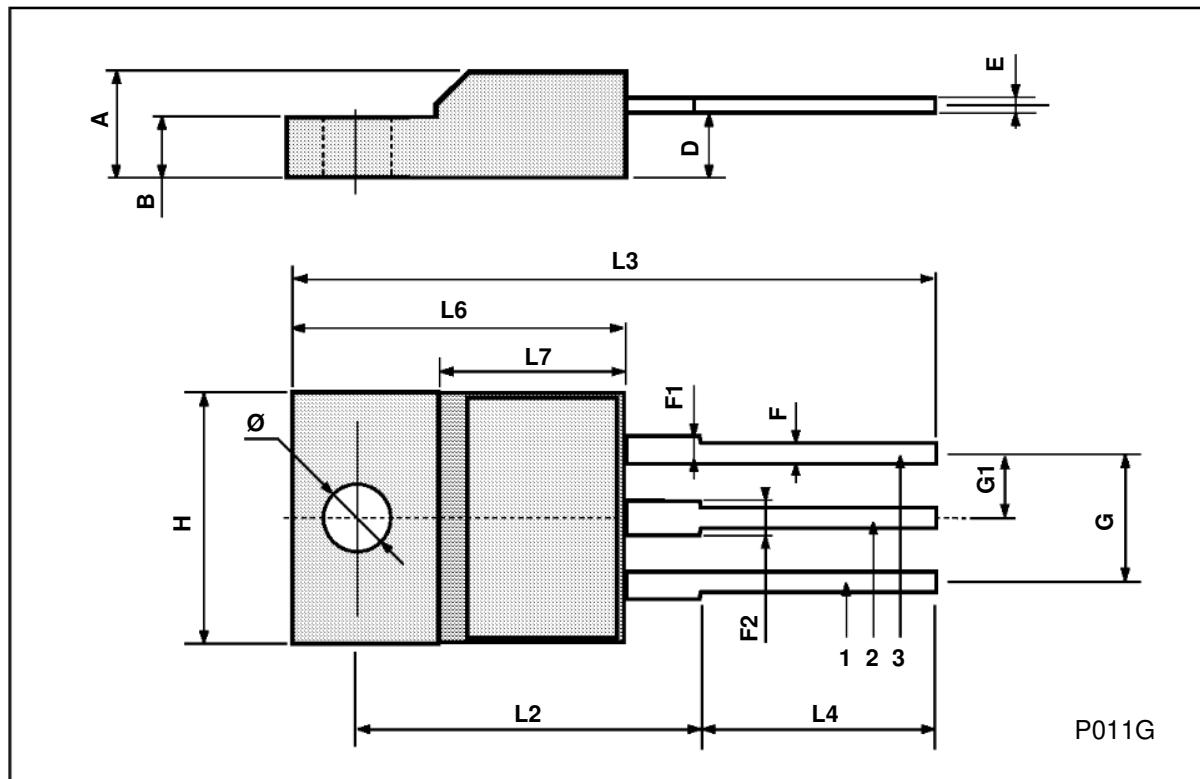
DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.40		4.60	0.173		0.181
C	1.23		1.32	0.048		0.051
D	2.40		2.72	0.094		0.107
D1		1.27			0.050	
E	0.49		0.70	0.019		0.027
F	0.61		0.88	0.024		0.034
F1	1.14		1.70	0.044		0.067
F2	1.14		1.70	0.044		0.067
G	4.95		5.15	0.194		0.203
G1	2.4		2.7	0.094		0.106
H2	10.0		10.40	0.393		0.409
L2		16.4			0.645	
L4	13.0		14.0	0.511		0.551
L5	2.65		2.95	0.104		0.116
L6	15.25		15.75	0.600		0.620
L7	6.2		6.6	0.244		0.260
L9	3.5		3.93	0.137		0.154
DIA.	3.75		3.85	0.147		0.151



P011C

ISOWATT220 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.4		4.6	0.173		0.181
B	2.5		2.7	0.098		0.106
D	2.5		2.75	0.098		0.108
E	0.4		0.7	0.015		0.027
F	0.75		1	0.030		0.039
F1	1.15		1.7	0.045		0.067
F2	1.15		1.7	0.045		0.067
G	4.95		5.2	0.195		0.204
G1	2.4		2.7	0.094		0.106
H	10		10.4	0.393		0.409
L2		16			0.630	
L3	28.6		30.6	1.126		1.204
L4	9.8		10.6	0.385		0.417
L6	15.9		16.4	0.626		0.645
L7	9		9.3	0.354		0.366
Ø	3		3.2	0.118		0.126



Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of SGS-THOMSON Microelectronics.

© 1994 SGS-THOMSON Microelectronics - All Rights Reserved

SGS-THOMSON Microelectronics GROUP OF COMPANIES

Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands -
Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A

Low power dual voltage comparator**LM193/A/293/A/393/A/2903****DESCRIPTION**

The LM193 series consists of two independent precision voltage comparators with an offset voltage specification as low as 2.0mV max. for two comparators which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. These comparators also have a unique characteristic in that the input common-mode voltage range includes ground, even though operated from a single power supply voltage.

The LM193 series was designed to directly interface with TTL and CMOS. When operated from both plus and minus power supplies, the LM193 series will directly interface with MOS logic where their low power drain is a distinct advantage over standard comparators.

FEATURES

- Wide single supply voltage range 2.0VDC to 36VDC or dual supplies ± 1.0 VDC, to ± 18 VDC
- Very low supply current drain (0.8mA) independent of supply voltage (2.0mW/comparator at 5.0VDC)
- Low input biasing current 25nA
- Low input offset current ± 5 nA and offset voltage ± 2 mV
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Low output 250mV at 4mA saturation voltage
- Output voltage compatible with TTL, DTL, ECL, MOS and CMOS logic systems

APPLICATIONS

- A/D converters
- Wide range VCO
- MOS clock generator
- High voltage logic gate
- Multivibrators

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
8-Pin Ceramic Dual In-Line Package (Cerdip)	-55°C to +125°C	LM193FE	0580A
8-Pin Ceramic Dual In-Line Package (Cerdip)	-25°C to +85°C	LM293FE	0580A
8-Pin Plastic Dual In-Line Package (DIP)	-25°C to +85°C	LM293N	SOT97-1
8-Pin Plastic Small Outline (SO) Package	-25°C to +85°C	LM293D	SOT96-1
8-Pin Plastic Dual In-Line Package (DIP)	-25°C to +85°C	LM293AN	SOT97-1
8-Pin Ceramic Dual In-Line Package (Cerdip)	0 to +70°C	LM393AFE	0580A
8-Pin Ceramic Dual In-Line Package (Cerdip)	0 to +70°C	LM393FE	0580A
8-Pin Plastic Small Outline (SO) Package	0 to +70°C	LM393D	SOT96-1
8-Pin Plastic Dual In-Line Package (DIP)	0 to +70°C	LM393N	SOT97-1
8-Pin Plastic Dual In-Line Package (DIP)	0 to +70°C	LM393AN	SOT97-1
8-Pin Plastic Dual In-Line Package (DIP)	-40°C to +125°C	LM2903N	SOT97-1
8-Pin Plastic Dual In-Line Package (DIP)	-40°C to +125°C	LM2903D	SOT97-1

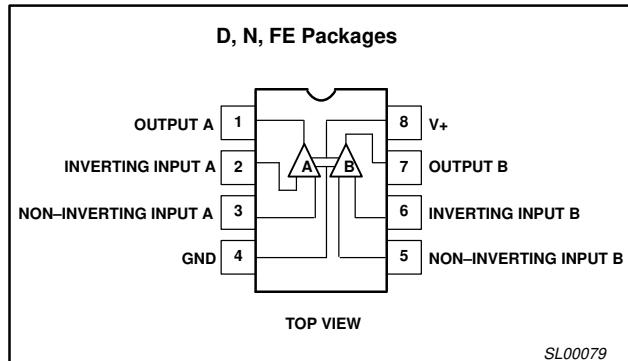
PIN CONFIGURATION

Figure 1. Pin Configuration

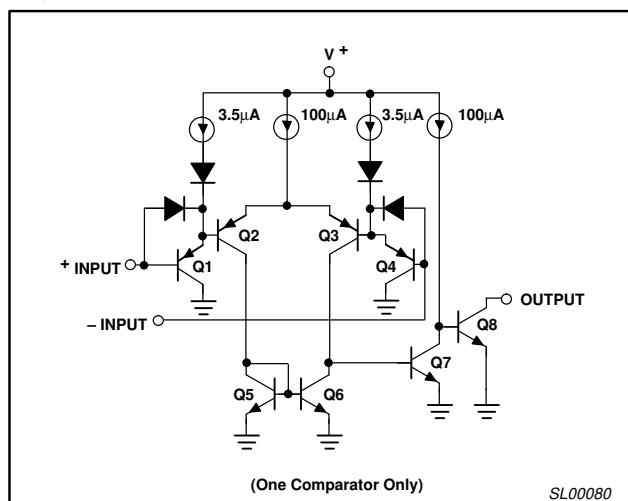
EQUIVALENT CIRCUIT

Figure 2. Equivalent Circuit

Low power dual voltage comparator

LM193/A/293/A/393/A/2903

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_{CC}	Supply voltage	36 or ± 18	V_{DC}
	Differential input voltage	36	V_{DC}
V_{IN}	Input voltage	-0.3 to +36	V_{DC}
P_D	Maximum power dissipation, $T_A=25^\circ\text{C}$ (still-air) ¹		
	F package	780	mW
	N package	1160	mW
	D package	780	mW
	Output short-circuit to ground ²	Continuous	
I_{IN}	Input current ($V_{IN} < -0.3V_{DC}$) ³	50	mA
T_A	Operating temperature range LM193/193A	-55 to +125	°C
	LM293/293A	-25 to +85	°C
	LM393/393A	0 to +70	°C
	LM2903	-40 to +125	°C
T_{STG}	Storage temperature range	-65 to +150	°C
T_{SOLD}	Lead soldering temperature (10sec max)	300	°C

NOTES:

1. Derate above 25°C , at the following rates:
F package at $6.2\text{mW/}^\circ\text{C}$
N package at $9.3\text{mW/}^\circ\text{C}$
D package at $6.2\text{mW/}^\circ\text{C}$
2. Short circuits from the output to V_+ can cause excessive heating and eventual destruction. The maximum output current is approximately 20mA independent of the magnitude of V_+ .
3. This input current will only exist when the voltage at any of the input leads is driven negative. It is due to the collector-base junction of the input PNP transistors becoming forward biased and thereby acting as input diode clamps. In addition to this diode action, there is also lateral NPN parasitic transistor action on the IC chip. This transistor action can cause the output voltages of the comparators to go to the V_+ voltage level (or to ground for a large overdrive) for the time duration that an input is driven negative. This is not destructive and normal output states will re-establish when the input voltage, which was negative, again returns to a value greater than $-0.3V_{DC}$.

Low power dual voltage comparator

LM193/A/293/A/393/A/2903

DC AND AC ELECTRICAL CHARACTERISTICS

V₊=5VDC, LM193/193A: -55°C T_A ≤ +125°C, unless otherwise specified. LM293/293A: -25°C T_A ≤ +85°C, unless otherwise specified.LM393/393A: 0°C T_A ≤ +70°C, unless otherwise specified. LM2903: -40°C T_A ≤ +125°C, unless otherwise specified.

SYMBOL	PARAMETER	TEST CONDITIONS	LM193A			LM293A/393A			LM2903			UNIT
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V _{OS}	Input offset voltage ²	T _A =25°C Over temp.		±1.0	±2.0 ±4.0		±1.0	±2.0 ±4.0		±2.0 ±9	±7.0 ±15	mV mV
V _{CM}	Input common-mode voltage range ^{3, 6}	T _A =25°C Over temp.	0 0	V ₊ -1.5 V ₊ -2.0	0 0	V ₊ -1.5 V ₊ -2.0	0 0	V ₊ -1.5 V ₊ -2.0	0 0	V ₊ -1.5 V ₊ -2.0	V V	
V _{IDR}	Differential input voltage ¹	Keep all V _{INs} ≥ 0V _{DC} (or V ₋ if need)		V ₊			V ₊			V ₊	V	
I _{BIAS}	Input bias current ⁴	I _{IN(+)} or I _{IN(-)} with output in linear range T _A =25°C Over temp.		25	100 300		25	250 400		25 200	250 500	nA nA
I _{OS}	Input offset current	I _{IN(+)} -I _{IN(-)} T _A =25°C Over temp.		±3.0	±25 ±100		±5.0	±50 ±150		±5 ±50	±50 ±200	nA nA
I _{OL}	Output sink current	V _{IN(-)} ≥1V _{DC} , V _{IN(+)} =0, V ₀ ≤1.5V _{DC} T _A =25°C	6.0	16		6.0	16		6.0	16		mA
	Output leakage current	V ₀ =5V _{DC} , T _A =25°C V _{IN(+)} ≥1V _{DC} , V _{IN(-)} =0 V ₀ =30V _{DC} Over temp.		0.1	1.0		0.1	1.0		0.1	1.0	nA μA
I _{CC}	Supply current	R _L =∞ on both comparators, T _A =25°C		0.8	1		0.8	1		0.8	1	mA
		R _L =∞ on both comparators, V ₊ =30V		1	2.5		1	2.5		1	2.5	mA
A _V	Voltage gain	R _L ≥15kΩ, V ₊ =15V _{DC} , T _A =25°C	50	200		50	200		25	100		V/mV
V _{OL}	Saturation voltage	V _{IN(-)} ≥1V _{DC} , V _{IN(+)} =0, I _{SINK} ≤4mA T _A =25°C Over temp.		250	400 700		250	400 700		400	400 700	mV mV
t _{LSR}	Large-signal response time	V _{IN} =TTL logic swing, V _{REF} =1.4V _{DC} V _{RL} =5V _{DC} , R _L =5.1kΩ, T _A =25°C		300			300			300		ns
t _R	Response time ⁵	V _{RL} =5V _{DC} , R _L =5.1kΩ T _A =25°C		1.3			1.3			1.3		μs

Low power dual voltage comparator

LM193/A/293/A/393/A/2903

DC ELECTRICAL CHARACTERISTICS (Continued)V₊=5VDC, LM193/193A: -55°C T_A ≤ +125°C, unless otherwise specified. LM293/293A: -25°C T_A ≤ +85°C, unless otherwise specified.LM393/393A: 0°C T_A ≤ +70°C, unless otherwise specified. LM2903: -40°C T_A ≤ +125°C, unless otherwise specified.

SYMBOL	PARAMETER	TEST CONDITIONS	LM193			LM293/393			UNIT
			Min	Typ	Max	Min	Typ	Max	
V _{OS}	Input offset voltage ²	T _A =25°C Over temp.		±2.0	±5.0 ±9.0		±2.0	±5.0 ±9.0	mV mV
V _{CM}	Input common-mode voltage range ^{3, 6}	T _A =25°C Over temp.	0 0		V ₊ -1.5 V ₊ -2.0	0 0		V ₊ -1.5 V ₊ -2.0	V V
V _{IDR}	Differential input voltage ¹	Keep all V _{INs} ≥ 0V _{DC} (or V-if need)			V ₊			V ₊	V
I _{BIAS}	Input bias current ⁴	I _{IN(+)} or I _{IN(-)} with output in linear range T _A =25°C Over temp.		25	100 300		25	250 400	nA nA
I _{OS}	Input offset current	I _{IN(+)} -I _{IN(-)} T _A =25°C Over temp.		±3.0	±25 ±100		±5.0	±50 ±150	nA nA
I _{OL}	Output sink current	V _{IN(-)} ≥1V _{DC} , V _{IN(+)} =0, V _O ≤1.5V _{DC} T _A =25°C	6.0	16		6.0	16		mA
	Output leakage current	V _{IN(+)} ≥1V _{DC} , V _{IN(-)} =0, V _O =5V _{DC} T _A =25°C V _O =30V _{DC} over temp.		0.1	1.0		0.1	1.0	nA μA
I _{CC}	Supply current	R _L =∞ on both comparators, T _A =25°C		0.8	1		0.8	1	mA
		R _L =∞ on both comparators, V ₊ =30V			2.5			2.5	mA
A _V	Voltage gain	R _L ≥15kΩ, V ₊ =15V _{DC}	50	200		50	200		V/mV
V _{OL}	Saturation voltage	V _{IN(-)} ≥1V _{DC} , V _{IN(+)} =0, I _{SINK} ≤4mA T _A =25°C Over temp.		250	400 700		250	400 700	mV mV
t _{LSR}	Large signal response time	V _{IN} =TTL logic swing, V _{REF} =1.4V _{DC} , V _{RL} =5V _{DC} R _L =5.1kΩ, T _A =25°C		300			300		ns
t _R	Response time ⁵	V _{RL} =5V _{DC} , R _L =5.1kΩ T _A =25°C		1.3			1.3		μs

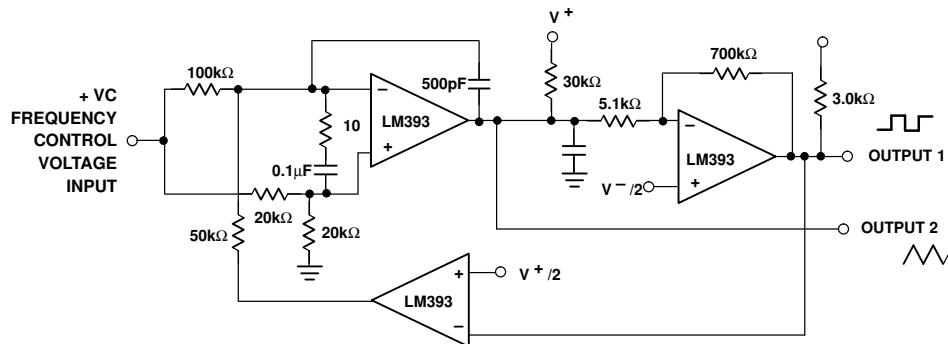
NOTES:

- Positive excursions of input voltage may exceed the power supply level by 17V. As long as the other voltage remains within the common-mode range, the comparator will provide a proper output state. The low input voltage state must not be less than -0.3V_{DC} (V_{DC} below the magnitude of the negative power supply, if used).
- At output switch point, V_O ≈ 1.4V_{DC}, R_S=0Ω with V₊ from 5V_{DC} to 30V_{DC} and over the full input common-mode range (0V_{DC} to V₊-1.5V_{DC}).
- The input common-mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is V₊-1.5V, but either or both inputs can go to 30V_{DC} without damage.
- The direction of the input current is out of the IC due to the PNP input stage. This current is essentially constant, independent of the state of the output so no loading change exists on the reference or input lines.
- The response time specified is for a 100mV input step with a 5mV overdrive.
- For input signals that exceed V_{CC}, only the over-driven comparator is affected. With a 5V supply, V_{IN} should be limited to 25V maximum, and a limiting resistor should be used on all inputs that might exceed the positive supply.

Low power dual voltage comparator

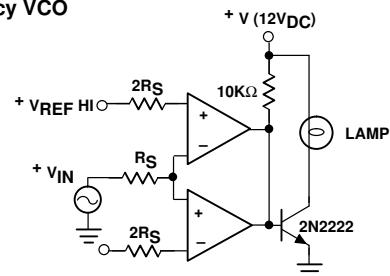
LM193/A/293/A/393/A/2903

EQUIVALENT CIRCUIT

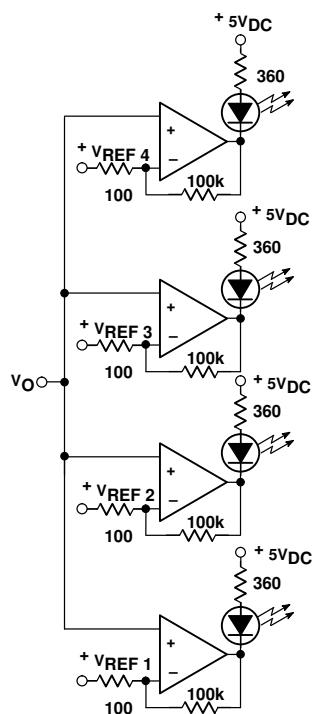


NOTES:
 $V_+ = 30\text{VDC}$
 $+250\text{mVDC} \leq V_C = 50\text{VDC}$
 $700\text{Hz} \leq f_O = 100\text{kHz}$

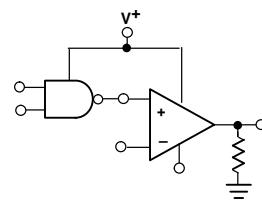
Two-Decade High-Frequency VCO



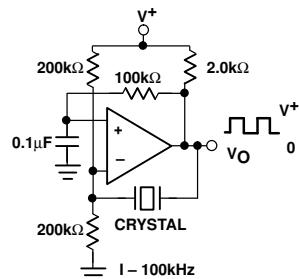
Limit Comparator



Visible Voltage Indicator



TTL-to-MOS Logic Converter



Crystal-Controlled Oscillator

NOTE:
Input of unused comparators should be grounded.

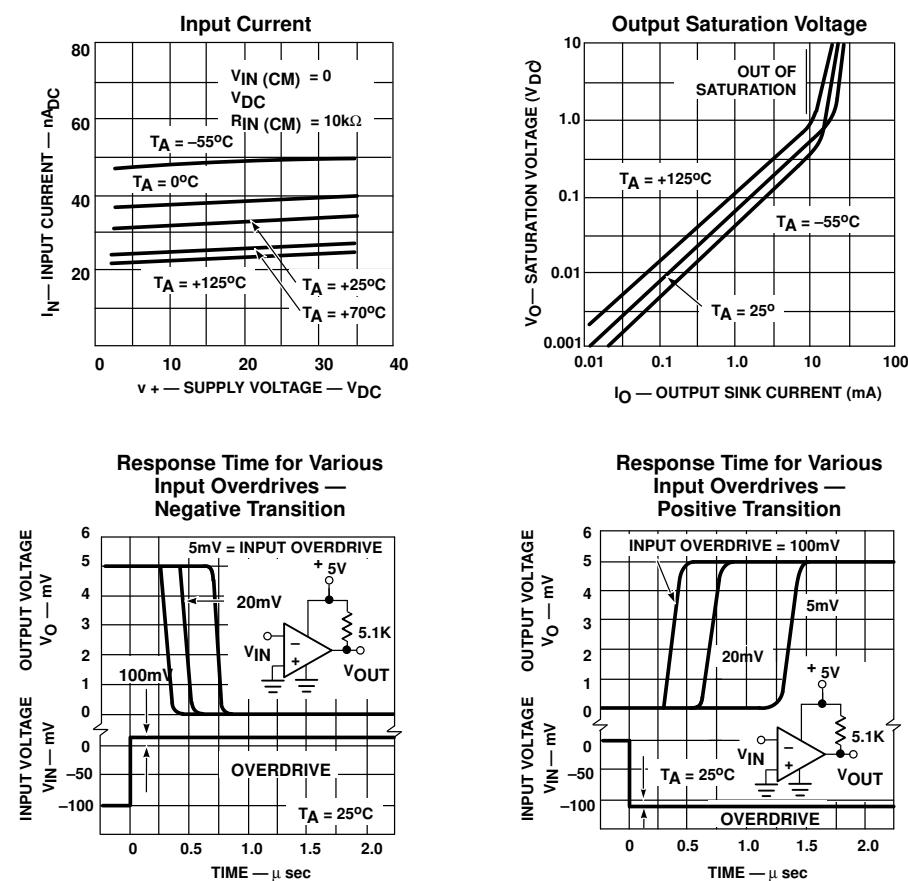
SL00081

Figure 3. Equivalent Circuit

Low power dual voltage comparator

LM193/A/293/A/393/A/2903

TYPICAL PERFORMANCE CHARACTERISTICS



SL00082

Figure 4. Typical Performance Characteristics



High CMR, High Speed TTL Compatible Optocouplers

Technical Data

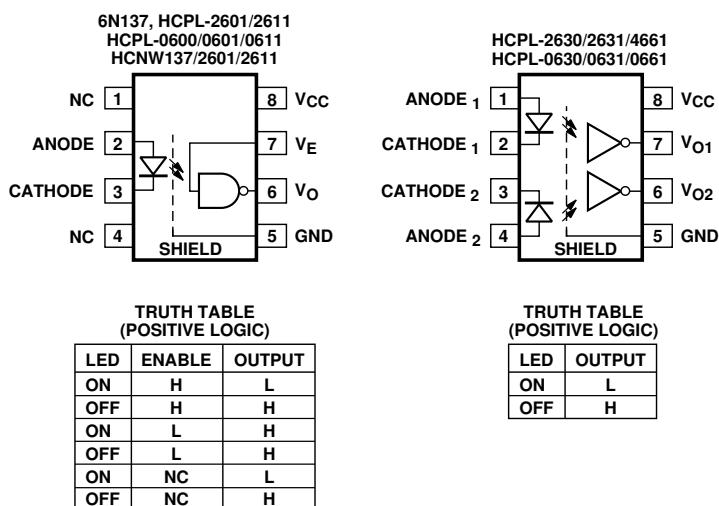
Features

- 5 kV/ μ s Minimum Common Mode Rejection (CMR) at $V_{CM} = 50$ V for HCPL-X601/X631, HCNW2601 and 10 kV/ μ s Minimum CMR at $V_{CM} = 1000$ V for HCPL-X611/X661, HCNW2611
- High Speed: 10 MBd Typical
- LSTTL/TTL Compatible
- Low Input Current Capability: 5 mA
- Guaranteed ac and dc Performance over Temperature: -40°C to +85°C
- Available in 8-Pin DIP, SOIC-8, Widebody Packages
- Stroable Output (Single Channel Products Only)
- Safety Approval
UL Recognized - 2500 V rms for 1 minute and 5000 V rms* for 1 minute per UL1577
CSA Approved
VDE 0884 Approved with $V_{IORM} = 630$ V peak for HCPL-2611 Option 060 and $V_{IORM} = 1414$ V peak for HCNW137/26X1
BSI Certified (HCNW137/26X1 Only)
- MIL-STD-1772 Version Available (HCPL-56XX/66XX)

Applications

- Isolated Line Receiver
- Computer-Peripheral Interfaces
- Microprocessor System Interfaces
- Digital Isolation for A/D, D/A Conversion
- Switching Power Supply
- Instrument Input/Output Isolation
- Ground Loop Elimination
- Pulse Transformer Replacement

Functional Diagram



*5000 V rms/1 Minute rating is for HCNW137/26X1 and Option 020 (6N137, HCPL-2601/11/30/31, HCPL-4661) products only.

A 0.1 μ F bypass capacitor must be connected between pins 5 and 8.

CAUTION: It is advised that normal static precautions be taken in handling and assembly of this component to prevent damage and/or degradation which may be induced by ESD.

an open collector Schottky-clamped transistor. The internal shield provides a guaranteed common mode transient immunity specification of 5,000 V/ μ s for the HCPL-X601/X631 and HCNW2601, and 10,000 V/ μ s for the HCPL-X611/X661 and HCNW2611.

This unique design provides maximum ac and dc circuit isolation while achieving TTL compatibility. The optocoupler ac and dc operational parameters are guaranteed from -40°C to +85°C allowing troublefree system performance.

The 6N137, HCPL-26XX, HCPL-06XX, HCPL-4661, HCNW137, and HCNW26X1 are suitable for high speed logic interfacing, input/output buffering, as line receivers in environments that conventional line receivers cannot tolerate and are recommended for use in extremely high ground or induced noise environments.

Selection Guide

Minimum CMR		Input On-Current (mA)	Output Enable	8-Pin DIP (300 Mil)		Small-Outline SO-8		Widebody (400 Mil)	Hermetic
dV/dt (V/ μ s)	V _{CM} (V)			Single Channel Package	Dual Channel Package	Single Channel Package	Dual Channel Package	Single Channel Package	Single and Dual Channel Packages
NA	NA	5	YES	6N137		HCPL-0600		HCNW137	
			NO		HCPL-2630		HCPL-0630		
			YES	HCPL-2601		HCPL-0601		HCNW2601	
			NO		HCPL-2631		HCPL-0631		
			YES	HCPL-2611		HCPL-0611		HCNW2611	
			NO		HCPL-4661		HCPL-0661		
			YES	HCPL-2602 ^[1]					
			YES	HCPL-2612 ^[1]					
1,000	50	3	YES	HCPL-261A ^[1]		HCPL-061A ^[1]			
			NO		HCPL-263A ^[1]		HCPL-063A ^[1]		
			YES	HCPL-261N ^[1]		HCPL-061N ^[1]			
			NO		HCPL-263N ^[1]		HCPL-063N ^[1]		
1,000	50	12.5	[3]						HCPL-193X ^[1] HCPL-56XX ^[1] HCPL-66XX ^[1]

Notes:

1. Technical data are on separate HP publications.
2. 15 kV/ μ s with V_{CM} = 1 kV can be achieved using HP application circuit.
3. Enable is available for single channel products only, except for HCPL-193X devices.

Ordering Information

Specify Part Number followed by Option Number (if desired).

Example:

HCPL-2611#XXX

- 020 = 5000 V rms/1 minute UL Rating Option*
 - 060 = VDE 0884 V_{IORM} = 630 Vpeak Option**
 - 300 = Gull Wing Surface Mount Option†
 - 500 = Tape and Reel Packaging Option

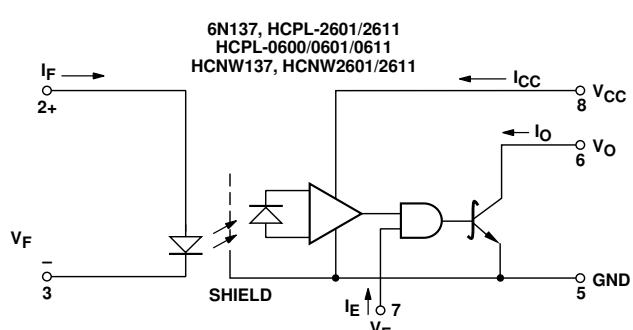
Option data sheets available. Contact Hewlett-Packard sales representative or authorized distributor for information.

*For 6N137, HCPL-2601/11/30/31 and HCPL-4661 (8-pin DIP products) only.

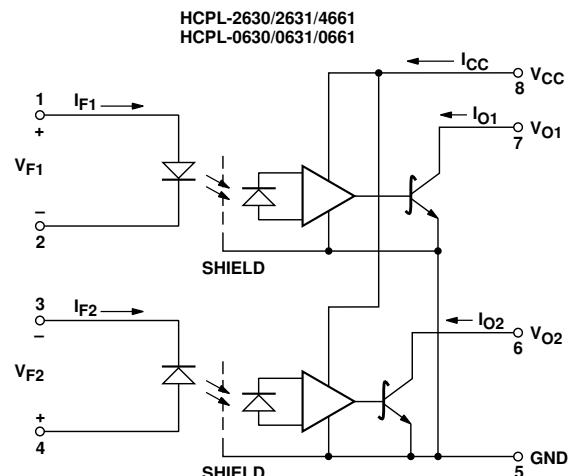
**For HCPL-2611 only. Combination of Option 020 and Option 060 is not available.

[†]Gull wing surface mount option applies to through hole parts only.

Schematic

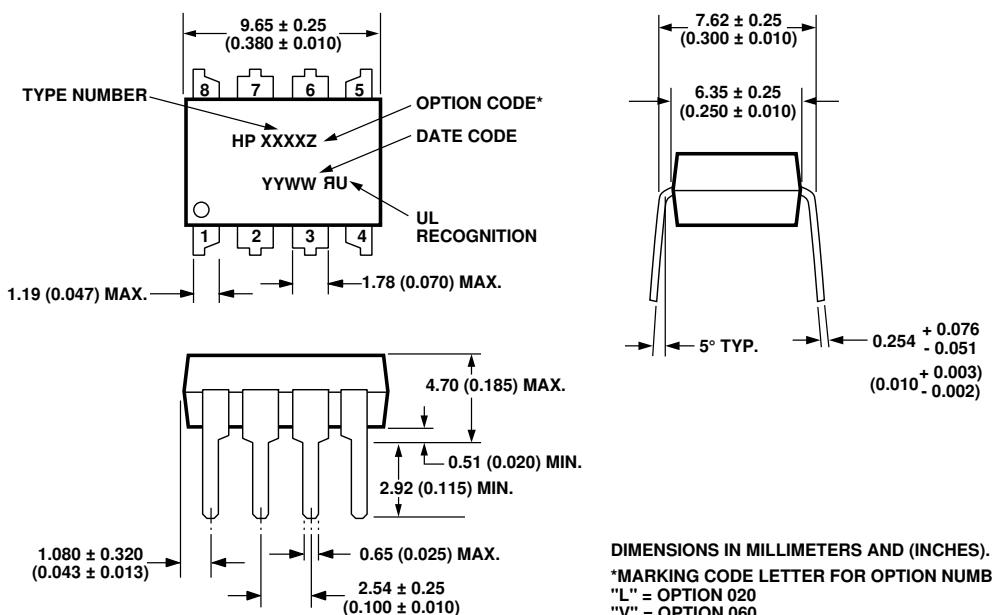


USE OF A 0.1 μ F BYPASS CAPACITOR CONNECTED BETWEEN PINS 5 AND 8 IS RECOMMENDED (SEE NOTE 5).



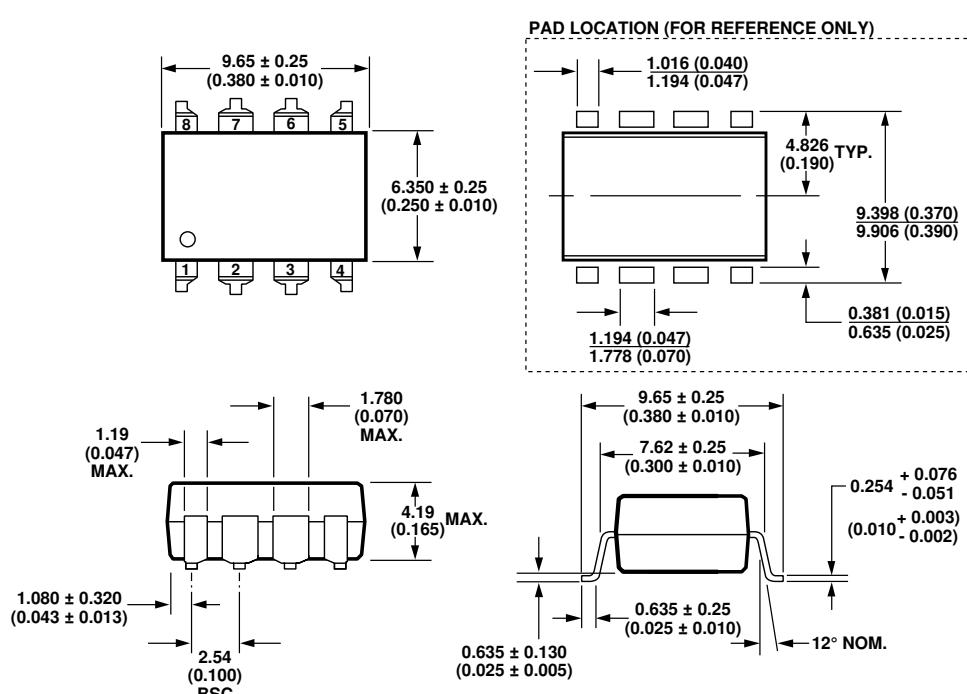
Package Outline Drawings

8-pin DIP Package** (6N137, HCPL-2601/11/30/31, HCPL-4661)

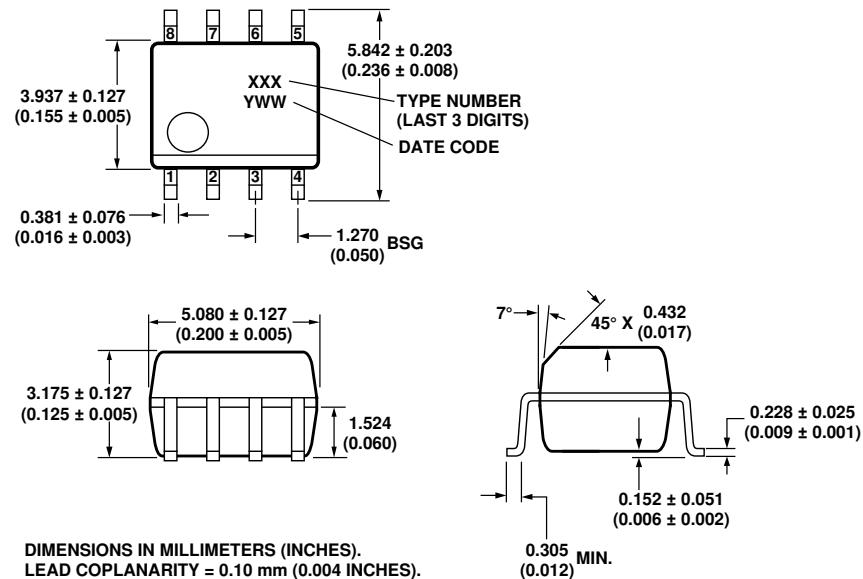


**JEDEC Registered Data (for 6N137 only).

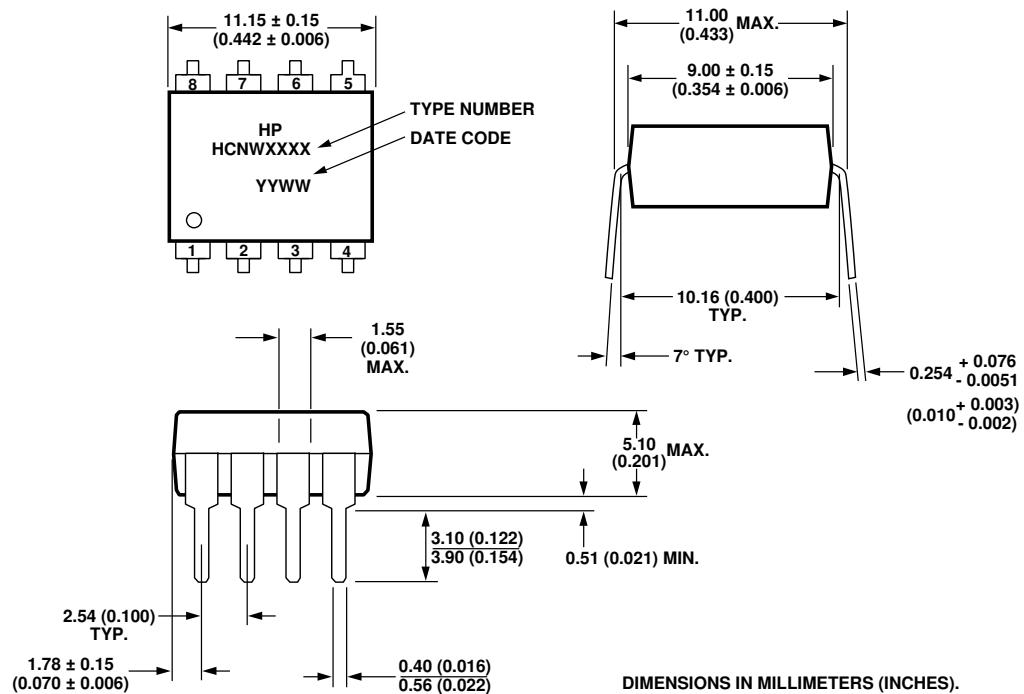
8-pin DIP Package with Gull Wing Surface Mount Option 300 (6N137, HCPL-2601/11/30/31, HCPL-4661)



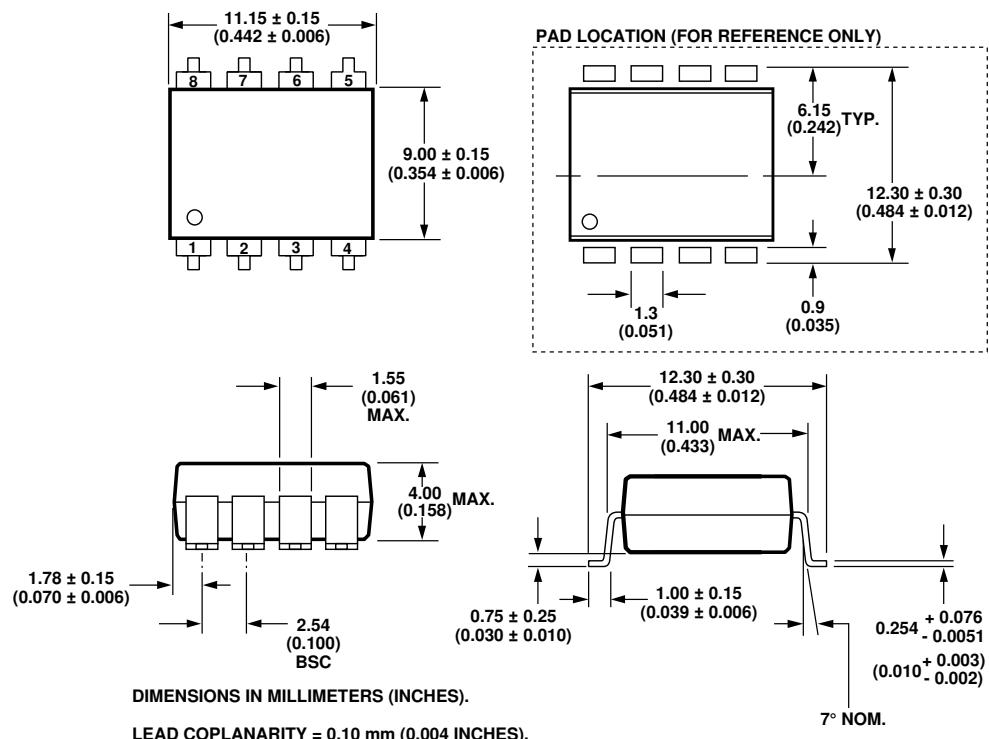
Small-Outline SO-8 Package (HCPL-0600/01/11/30/31/61)



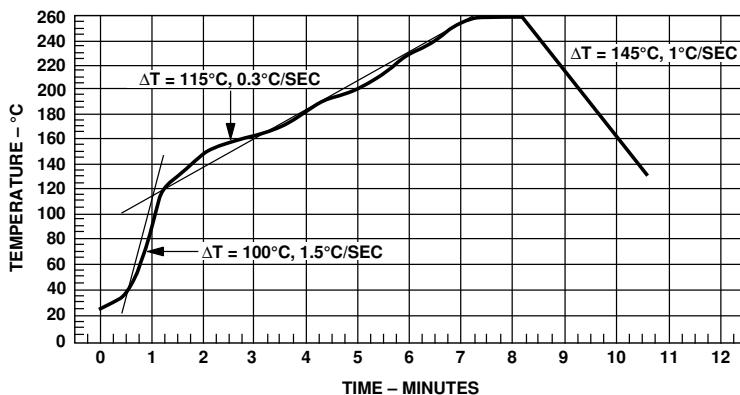
8-Pin Widebody DIP Package (HCNW137, HCNW2601/11)



**8-Pin Widebody DIP Package with Gull Wing Surface Mount Option 300
(HCNW137, HCNW2601/11)**



Solder Reflow Temperature Profile (HCPL-06XX and Gull Wing Surface Mount Option 300 Parts)



Note: Use of nonchlorine activated fluxes is highly recommended.

Regulatory Information

The 6N137, HCPL-26XX/06XX/46XX, and HCNW137/26XX have been approved by the following organizations:

UL

Recognized under UL 1577, Component Recognition Program, File E55361.

CSA

Approved under CSA Component Acceptance Notice #5, File CA 88324.

VDE

Approved according to VDE 0884/06.92. (HCPL-2611 Option 060 and HCNW137/26X1 only)

BSI

Certification according to BS415:1994 (BS EN60065:1994), BS7002:1992 (BS EN60950:1992) and EN41003:1993 for Class II applications. (HCNW137/26X1 only)

Insulation and Safety Related Specifications

Parameter	Symbol	8-pin DIP (300 Mil) Value	SO-8 Value	Widebody (400 Mil) Value	Units	Conditions
Minimum External Air Gap (External Clearance)	L(101)	7.1	4.9	9.6	mm	Measured from input terminals to output terminals, shortest distance through air.
Minimum External Tracking (External Creepage)	L(102)	7.4	4.8	10.0	mm	Measured from input terminals to output terminals, shortest distance path along body.
Minimum Internal Plastic Gap (Internal Clearance)		0.08	0.08	1.0	mm	Through insulation distance, conductor to conductor, usually the direct distance between the photoemitter and photodetector inside the optocoupler cavity.
Minimum Internal Tracking (Internal Creepage)		NA	NA	4.0	mm	Measured from input terminals to output terminals, along internal cavity.
Tracking Resistance (Comparative Tracking Index)	CTI	200	200	200	Volts	DIN IEC 112/VDE 0303 Part 1
Isolation Group		IIIa	IIIa	IIIa		Material Group (DIN VDE 0110, 1/89, Table 1)

Option 300 - surface mount classification is Class A in accordance with CECC 00802.

VDE 0884 Insulation Related Characteristics (HCPL-2611 Option 060 Only)

Description	Symbol	Characteristic	Units
Installation classification per DIN VDE 0110/1.89, Table 1 for rated mains voltage ≤ 300 V rms for rated mains voltage ≤ 450 V rms		I-IV	
		I-III	
Climatic Classification		55/85/21	
Pollution Degree (DIN VDE 0110/1.89)		2	
Maximum Working Insulation Voltage	V_{IORM}	630	V _{peak}
Input to Output Test Voltage, Method b*	V_{PR}	1181	V _{peak}
$V_{IORM} \times 1.875 = V_{PR}$, 100% Production Test with $t_m = 1$ sec, Partial Discharge < 5 pC			
Input to Output Test Voltage, Method a*	V_{PR}	945	V _{peak}
$V_{IORM} \times 1.5 = V_{PR}$, Type and sample test, $t_m = 60$ sec, Partial Discharge < 5 pC			
Highest Allowable Ovvoltage*	V_{IOTM}	6000	V _{peak}
(Transient Overvoltage, $t_{ini} = 10$ sec)			
Safety Limiting Values (Maximum values allowed in the event of a failure, also see Figure 16, Thermal Derating curve.)			
Case Temperature	T_S	175	°C
Input Current	$I_{S,INPUT}$	230	mA
Output Power	$P_{S,OUTPUT}$	600	mW
Insulation Resistance at T_S , $V_{IO} = 500$ V	R_S	$\geq 10^9$	Ω

*Refer to the front of the optocoupler section of the current catalog, under Product Safety Regulations section (VDE 0884), for a detailed description.

Note: Isolation characteristics are guaranteed only within the safety maximum ratings which must be ensured by protective circuits in application.

VDE 0884 Insulation Related Characteristics (HCNW137/2601/2611 Only)

Description	Symbol	Characteristic	Units
Installation classification per DIN VDE 0110/1.89, Table 1 for rated mains voltage ≤ 600 V rms for rated mains voltage ≤ 1000 V rms		I-IV	
		I-III	
Climatic Classification (DIN IEC 68 part 1)		55/100/21	
Pollution Degree (DIN VDE 0110/1.89)		2	
Maximum Working Insulation Voltage	V_{IORM}	1414	V _{peak}
Input to Output Test Voltage, Method b*	V_{PR}	2651	V _{peak}
$V_{IORM} \times 1.875 = V_{PR}$, 100% Production Test with $t_m = 1$ sec, Partial Discharge < 5 pC			
Input to Output Test Voltage, Method a*	V_{PR}	2121	V _{peak}
$V_{IORM} \times 1.5 = V_{PR}$, Type and sample test, $t_m = 60$ sec, Partial Discharge < 5 pC			
Highest Allowable Ovvoltage*	V_{IOTM}	8000	V _{peak}
(Transient Overvoltage, $t_{ini} = 10$ sec)			
Safety Limiting Values (Maximum values allowed in the event of a failure, also see Figure 16, Thermal Derating curve.)			
Case Temperature	T_S	150	°C
Input Current	$I_{S,INPUT}$	400	mA
Output Power	$P_{S,OUTPUT}$	700	mW
Insulation Resistance at T_S , $V_{IO} = 500$ V	R_S	$\geq 10^9$	Ω

*Refer to the front of the optocoupler section of the current catalog, under Product Safety Regulations section (VDE 0884), for a detailed description.

Note: Isolation characteristics are guaranteed only within the safety maximum ratings which must be ensured by protective circuits in application.

Absolute Maximum Ratings* (No Derating Required up to 85°C)

Parameter	Symbol	Package**	Min.	Max.	Units	Note	
Storage Temperature	T _S		-55	125	°C		
Operating Temperature†	T _A		-40	85	°C		
Average Forward Input Current	I _F	Single 8-Pin DIP		20	mA	2	
		Single SO-8 Widebody		15		1, 3	
Reverse Input Voltage	V _R	8-Pin DIP, SO-8		5	V	1	
		Widebody		3			
Input Power Dissipation	P _I	Widebody		40	mW		
Supply Voltage (1 Minute Maximum)	V _{CC}			7	V		
Enable Input Voltage (Not to Exceed V _{CC} by more than 500 mV)	V _E	Single 8-Pin DIP Single SO-8 Widebody		V _{CC} + 0.5	V		
Enable Input Current	I _E			5	mA		
Output Collector Current	I _O			50	mA	1	
Output Collector Voltage (Selection for Higher Output Voltages up to 20 V is Available.)	V _O			7	V	1	
Output Collector Power Dissipation	P _O	Single 8-Pin DIP Single SO-8 Widebody		85	mW		
		Dual 8-Pin DIP Dual SO-8		60		1, 4	
Lead Solder Temperature (Through Hole Parts Only)	T _L S	8-Pin DIP	260°C for 10 sec., 1.6 mm below seating plane				
		Widebody	260°C for 10 sec., up to seating plane				
Solder Reflow Temperature Profile (Surface Mount Parts Only)		SO-8 and Option 300	See Package Outline Drawings section				

*JEDEC Registered Data (for 6N137 only).

Ratings apply to all devices except otherwise noted in the **Package column.

†0°C to 70°C on JEDEC Registration.

Recommended Operating Conditions

Parameter	Symbol	Min.	Max.	Units
Input Current, Low Level	I _{FL} *	0	250	µA
Input Current, High Level ^[1]	I _{FH} **	5	15	mA
Power Supply Voltage	V _{CC}	4.5	5.5	V
Low Level Enable Voltage†	V _{EL}	0	0.8	V
High Level Enable Voltage†	V _{EH}	2.0	V _{CC}	V
Operating Temperature	T _A	-40	85	°C
Fan Out (at R _L = 1 kΩ) ^[1]	N		5	TTL Loads
Output Pull-up Resistor	R _L	330	4 k	Ω

*The off condition can also be guaranteed by ensuring that V_{FL} ≤ 0.8 volts.

**The initial switching threshold is 5 mA or less. It is recommended that 6.3 mA to 10 mA be used for best performance and to permit at least a 20% LED degradation guardband.

†For single channel products only.

Electrical Specifications

Over recommended temperature ($T_A = -40^\circ\text{C}$ to $+85^\circ\text{C}$) unless otherwise specified. All Typicals at $V_{CC} = 5\text{ V}$, $T_A = 25^\circ\text{C}$. All enable test conditions apply to single channel products only. See note 5.

Parameter	Sym.	Package	Min.	Typ.	Max.	Units	Test Conditions		Fig.	Note	
High Level Output Current	I_{OH}^*	All		5.5	100	μA	$V_{CC} = 5.5\text{ V}$, $V_E = 2.0\text{ V}$, $V_O = 5.5\text{ V}$, $I_F = 250\text{ }\mu\text{A}$		1	1, 6, 19	
Input Threshold Current	I_{TH}	Single Channel Widebody		2.0	5.0	mA	$V_{CC} = 5.5\text{ V}$, $V_E = 2.0\text{ V}$, $V_O = 0.6\text{ V}$, I_{OL} (Sinking) = 13 mA		2, 3	19	
		Dual Channel		2.5							
Low Level Output Voltage	V_{OL}^*	8-Pin DIP SO-8		0.35	0.6	V	$V_{CC} = 5.5\text{ V}$, $V_E = 2.0\text{ V}$, $I_F = 5\text{ mA}$, I_{OL} (Sinking) = 13 mA		2, 3, 4, 5	1, 19	
		Widebody		0.4							
High Level Supply Current	I_{CCH}	Single Channel		7.0	10.0*	mA	$V_E = 0.5\text{ V}$	$V_{CC} = 5.5\text{ V}$, $I_F = 0\text{ mA}$	7		
				6.5			$V_E = V_{CC}$				
		Dual Channel		10	15		Both Channels				
Low Level Supply Current	I_{CCL}	Single Channel		9.0	13.0*	mA	$V_E = 0.5\text{ V}$	$V_{CC} = 5.5\text{ V}$, $I_F = 10\text{ mA}$	8		
				8.5			$V_E = V_{CC}$				
		Dual Channel		13	21		Both Channels				
High Level Enable Current	I_{EH}	Single Channel		-0.7	-1.6	mA	$V_{CC} = 5.5\text{ V}$, $V_E = 2.0\text{ V}$				
Low Level Enable Current	I_{EL}^*			-0.9	-1.6	mA	$V_{CC} = 5.5\text{ V}$, $V_E = 0.5\text{ V}$			9	
High Level Enable Voltage	V_{EH}		2.0			V				19	
Low Level Enable Voltage	V_{EL}				0.8	V					
Input Forward Voltage	V_F	8-Pin DIP SO-8	1.4	1.5	1.75*	V	$T_A = 25^\circ\text{C}$	$I_F = 10\text{ mA}$	6, 7	1	
			1.3		1.80		$T_A = 25^\circ\text{C}$				
		Widebody	1.25	1.64	1.85						
			1.2		2.05						
Input Reverse Breakdown Voltage	BV_R^*	8-Pin DIP SO-8	5			V	$I_R = 10\text{ }\mu\text{A}$		7	1	
		Widebody	3				$I_R = 100\text{ }\mu\text{A}$, $T_A = 25^\circ\text{C}$				
Input Diode Temperature Coefficient	$\Delta V_F / \Delta T_A$	8-Pin DIP SO-8		-1.6		mV/°C	$I_F = 10\text{ mA}$		7	1	
		Widebody		-1.9							
Input Capacitance	C_{IN}	8-Pin DIP SO-8		60		pF	$f = 1\text{ MHz}$, $V_F = 0\text{ V}$		1	1	
		Widebody		70							

*JEDEC registered data for the 6N137. The JEDEC Registration specifies 0°C to $+70^\circ\text{C}$. HP specifies -40°C to $+85^\circ\text{C}$.

Switching Specifications (AC)

Over Recommended Temperature ($T_A = -40^\circ\text{C}$ to $+85^\circ\text{C}$), $V_{CC} = 5 \text{ V}$, $I_F = 7.5 \text{ mA}$ unless otherwise specified.
All Typicals at $T_A = 25^\circ\text{C}$, $V_{CC} = 5 \text{ V}$.

Parameter	Sym.	Package**	Min.	Typ.	Max.	Units	Test Conditions	Fig.	Note
Propagation Delay Time to High Output Level	t_{PLH}		20	48	75*	ns	$T_A = 25^\circ\text{C}$ $R_L = 350 \Omega$ $C_L = 15 \text{ pF}$	8, 9, 10	1, 10, 19
					100				
Propagation Delay Time to Low Output Level	t_{PHL}		25	50	75*	ns	$T_A = 25^\circ\text{C}$		1, 11, 19
					100				
Pulse Width Distortion	$ t_{PHL} - t_{PLH} $	8-Pin DIP SO-8		3.5	35	ns	$R_L = 350 \Omega$, $C_L = 15 \text{ pF}$, $V_{EL} = 0 \text{ V}$, $V_{EH} = 3 \text{ V}$	8, 9, 10, 11	13, 19
		Widebody			40				
Propagation Delay Skew	t_{PSK}				40	ns			12, 13, 19
Output Rise Time (10-90%)	t_r			24		ns		12	1, 19
Output Fall Time (90-10%)	t_f			10		ns		12	1, 19
Propagation Delay Time of Enable from V_{EH} to V_{EL}	t_{ELH}	Single Channel		30		ns		13, 14	14
Propagation Delay Time of Enable from V_{EL} to V_{EH}	t_{EHL}	Single Channel		20		ns			15

*JEDDEC registered data for the 6N137.

Ratings apply to all devices except otherwise noted in the **Package column.

Parameter	Sym.	Device	Min.	Typ.	Units	Test Conditions		Fig.	Note
Logic High Common Mode Transient Immunity	$ CM_H $	6N137 HCPL-2630 HCPL-0600/0630 HCNW137		10,000	V/ μs	$ V_{CM} = 10 \text{ V}$	$V_{CC} = 5 \text{ V}$, $I_F = 0 \text{ mA}$, $V_{O(\text{MIN})} = 2 \text{ V}$, $R_L = 350 \Omega$, $T_A = 25^\circ\text{C}$	15	1, 16, 18, 19
		HCPL-2601/2631 HCPL-0601/0631 HCNW2601				$ V_{CM} = 50 \text{ V}$			
		HCPL-2611/4661 HCPL-0611/0661 HCNW2611				$ V_{CM} = 1 \text{ kV}$			
Logic Low Common Mode Transient Immunity	$ CM_L $	6N137 HCPL-2630 HCPL-0600/0630 HCNW137		10,000	V/ μs	$ V_{CM} = 10 \text{ V}$	$V_{CC} = 5 \text{ V}$, $I_F = 7.5 \text{ mA}$, $V_{O(\text{MAX})} = 0.8 \text{ V}$, $R_L = 350 \Omega$, $T_A = 25^\circ\text{C}$	15	1, 17, 18, 19
		HCPL-2601/2631 HCPL-0601/0631 HCNW2601				$ V_{CM} = 50 \text{ V}$			
		HCPL-2611/4661 HCPL-0611/0661 HCNW2611				$ V_{CM} = 1 \text{ kV}$			

Package Characteristics

All Typicals at $T_A = 25^\circ\text{C}$.

Parameter	Sym.	Package	Min.	Typ.	Max.	Units	Test Conditions	Fig.	Note
Input-Output Insulation	I_{I-O}^*	Single 8-Pin DIP Single SO-8			1	μA	45% RH, $t = 5 \text{ s}$, $V_{I-O} = 3 \text{ kV dc}$, $T_A = 25^\circ\text{C}$		20, 21
Input-Output Momentary Withstand Voltage**	V_{ISO}	8-Pin DIP, SO-8	2500			V rms	RH $\leq 50\%$, $t = 1 \text{ min}$, $T_A = 25^\circ\text{C}$		20, 21
		Widebody	5000						20, 22
		OPT 020†	5000						
Input-Output Resistance	R_{I-O}	8-Pin DIP, SO-8		10^{12}		Ω	$V_{I-O} = 500 \text{ V dc}$ $T_A = 25^\circ\text{C}$ $T_A = 100^\circ\text{C}$		1, 20, 23
		Widebody	10^{12}	10^{13}					
			10^{11}						
Input-Output Capacitance	C_{I-O}	8-Pin DIP, SO-8		0.6		pF	$f = 1 \text{ MHz}$, $T_A = 25^\circ\text{C}$		1, 20, 23
		Widebody		0.5	0.6				
Input-Input Insulation Leakage Current	I_{I-I}	Dual Channel		0.005		μA	RH $\leq 45\%$, $t = 5 \text{ s}$, $V_{I-I} = 500 \text{ V}$		24
Resistance (Input-Input)	R_{I-I}	Dual Channel		10^{11}		Ω			24
Capacitance (Input-Input)	C_{I-I}	Dual 8-Pin DIP		0.03		pF	$f = 1 \text{ MHz}$		24
		Dual SO-8		0.25					

*JEDEC registered data for the 6N137. The JEDEC Registration specifies 0°C to 70°C . HP specifies -40°C to 85°C .

**The Input-Output Momentary Withstand Voltage is a dielectric voltage rating that should not be interpreted as an input-output continuous voltage rating. For the continuous voltage rating refer to the VDE 0884 Insulation Characteristics Table (if applicable), your equipment level safety specification or HP Application Note 1074 entitled "Optocoupler Input-Output Endurance Voltage."

†For 6N137, HCPL-2601/2611/2630/2631/4661 only.

Notes:

1. Each channel.
2. Peaking circuits may produce transient input currents up to 50 mA, 50 ns maximum pulse width, provided average current does not exceed 20 mA.
3. Peaking circuits may produce transient input currents up to 50 mA, 50 ns maximum pulse width, provided average current does not exceed 15 mA.
4. Derate linearly above 80°C free-air temperature at a rate of $2.7 \text{ mW}/^\circ\text{C}$ for the SOIC-8 package.
5. Bypassing of the power supply line is required, with a $0.1 \mu\text{F}$ ceramic disc capacitor adjacent to each optocoupler as illustrated in Figure 17. Total lead length between both ends of the capacitor and the isolator pins should not exceed 20 mm.
6. The JEDEC registration for the 6N137 specifies a maximum I_{OH} of $250 \mu\text{A}$. HP guarantees a maximum I_{OH} of $100 \mu\text{A}$.
7. The JEDEC registration for the 6N137 specifies a maximum I_{CCH} of 15 mA . HP guarantees a maximum I_{CCH} of 10 mA .
8. The JEDEC registration for the 6N137 specifies a maximum I_{CCL} of 18 mA . HP guarantees a maximum I_{CCL} of 13 mA .
9. The JEDEC registration for the 6N137 specifies a maximum I_{EL} of -2.0 mA . HP guarantees a maximum I_{EL} of -1.6 mA .
10. The t_{PLH} propagation delay is measured from the 3.75 mA point on the falling edge of the input pulse to the 1.5 V point on the rising edge of the output pulse.
11. The t_{PHL} propagation delay is measured from the 3.75 mA point on the rising edge of the input pulse to the 1.5 V point on the falling edge of the output pulse.
12. t_{PSK} is equal to the worst case difference in t_{PHL} and/or t_{PLH} that will be seen between units at any given temperature and specified test conditions.
13. See application section titled "Propagation Delay, Pulse-Width Distortion and Propagation Delay Skew" for more information.
14. The t_{EELH} enable propagation delay is measured from the 1.5 V point on the falling edge of the enable input pulse to the 1.5 V point on the rising edge of the output pulse.
15. The t_{EELH} enable propagation delay is measured from the 1.5 V point on the rising edge of the enable input pulse to the 1.5 V point on the falling edge of the output pulse.
16. CM_H is the maximum tolerable rate of rise of the common mode voltage to assure that the output will remain in a high logic state (i.e., $V_O > 2.0 \text{ V}$).
17. CM_L is the maximum tolerable rate of fall of the common mode voltage to assure that the output will remain in a low logic state (i.e., $V_O < 0.8 \text{ V}$).
18. For sinusoidal voltages, $(|dV_{CM}| / dt)_{max} = \pi f_{CM} V_{CM(p-p)}$.

19. No external pull up is required for a high logic state on the enable input. If the V_E pin is not used, tying V_E to V_{CC} will result in improved CMR performance. For single channel products only.
20. Device considered a two-terminal device: pins 1, 2, 3, and 4 shorted together, and pins 5, 6, 7, and 8 shorted together.
21. In accordance with UL1577, each optocoupler is proof tested by applying an insulation test voltage ≥ 3000 V rms for one second (leakage detection current limit, $I_{L0} \leq 5 \mu\text{A}$). This test is performed before the 100% production test for partial discharge (Method b) shown in the VDE 0884 Insulation Characteristics Table, if applicable.
22. In accordance with UL 1577, each optocoupler is proof tested by applying an insulation test voltage ≥ 6000 V rms for one second (leakage detection current limit, $I_{L0} \leq 5 \mu\text{A}$). This test is performed before the 100% production test for partial discharge (Method b) shown in the VDE 0884 Insulation Characteristics Table, if applicable.
23. Measured between the LED anode and cathode shorted together and pins 5 through 8 shorted together. For dual channel products only.
24. Measured between pins 1 and 2 shorted together, and pins 3 and 4 shorted together. For dual channel products only.

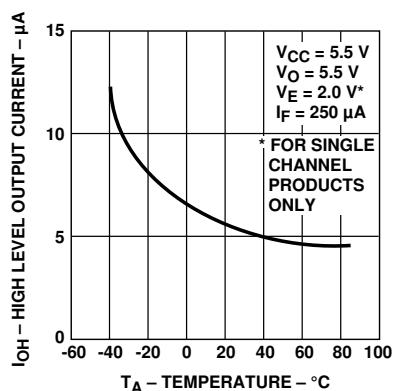


Figure 1. Typical High Level Output Current vs. Temperature.

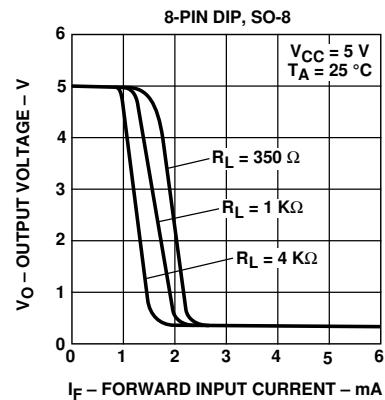


Figure 2. Typical Output Voltage vs. Forward Input Current.

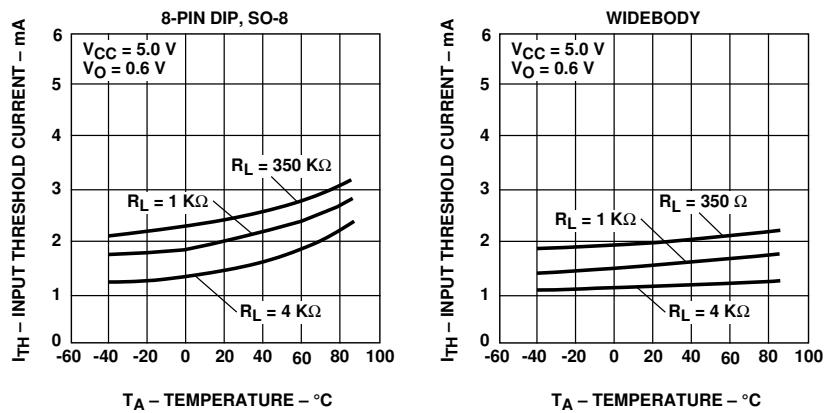


Figure 3. Typical Input Threshold Current vs. Temperature.

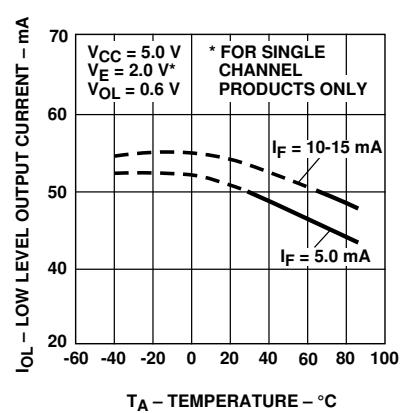
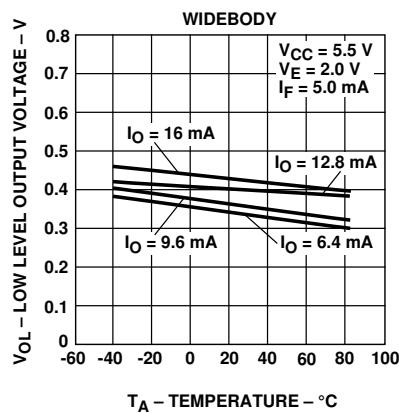
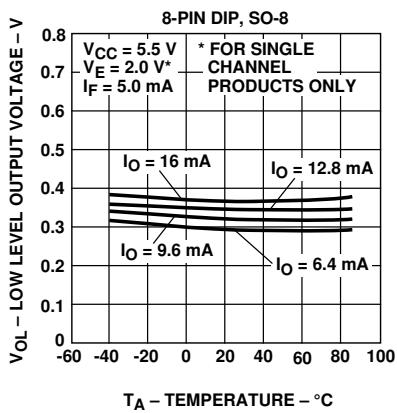


Figure 4. Typical Low Level Output Voltage vs. Temperature.

Figure 5. Typical Low Level Output Current vs. Temperature.

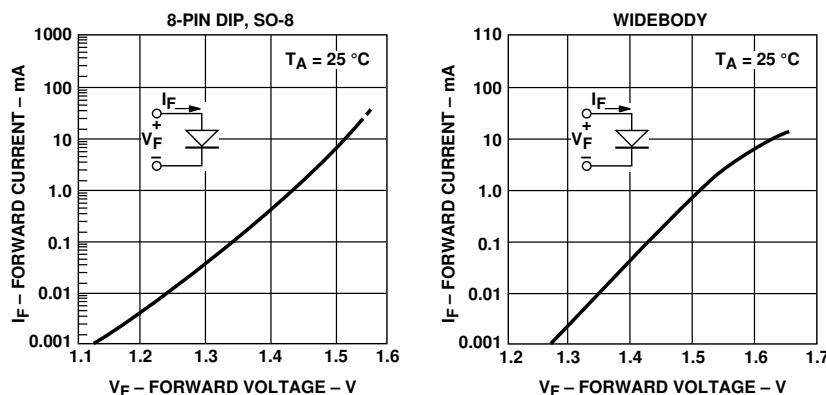


Figure 6. Typical Input Diode Forward Characteristic.

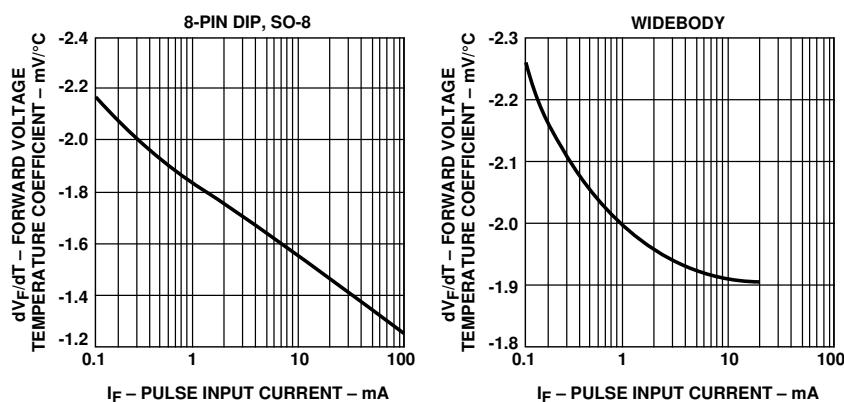


Figure 7. Typical Temperature Coefficient of Forward Voltage vs. Input Current.

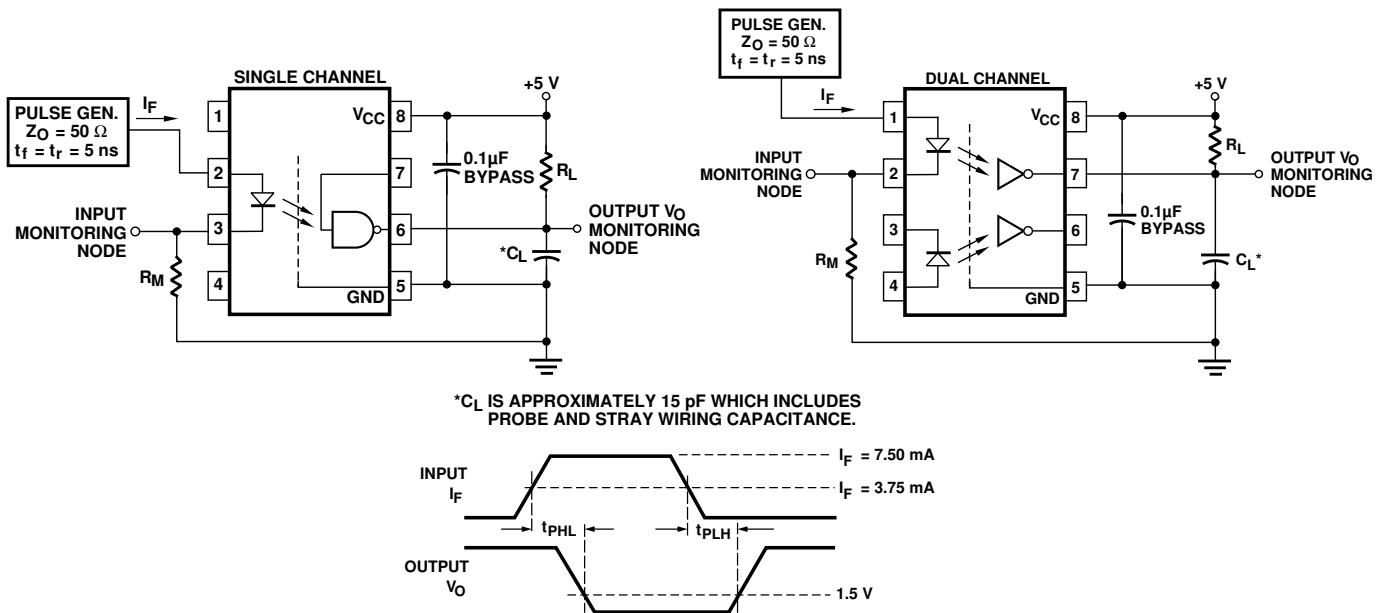


Figure 8. Test Circuit for t_{PHL} and t_{PLH} .

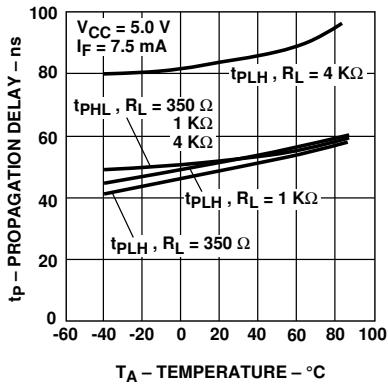


Figure 9. Typical Propagation Delay vs. Temperature.

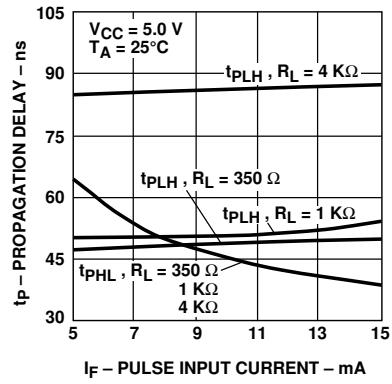


Figure 10. Typical Propagation Delay vs. Pulse Input Current.

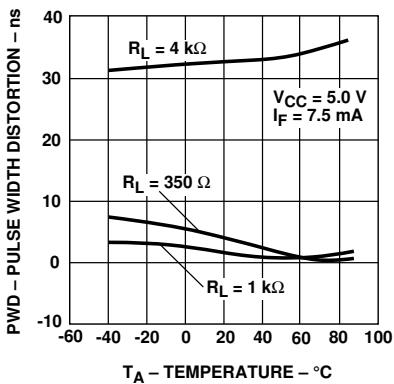


Figure 11. Typical Pulse Width Distortion vs. Temperature.

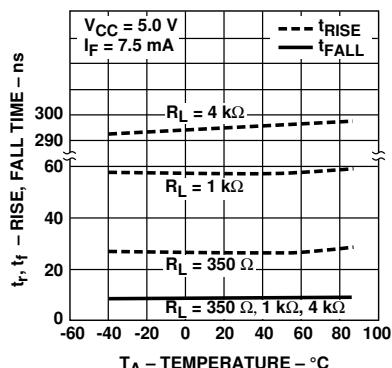


Figure 12. Typical Rise and Fall Time vs. Temperature.

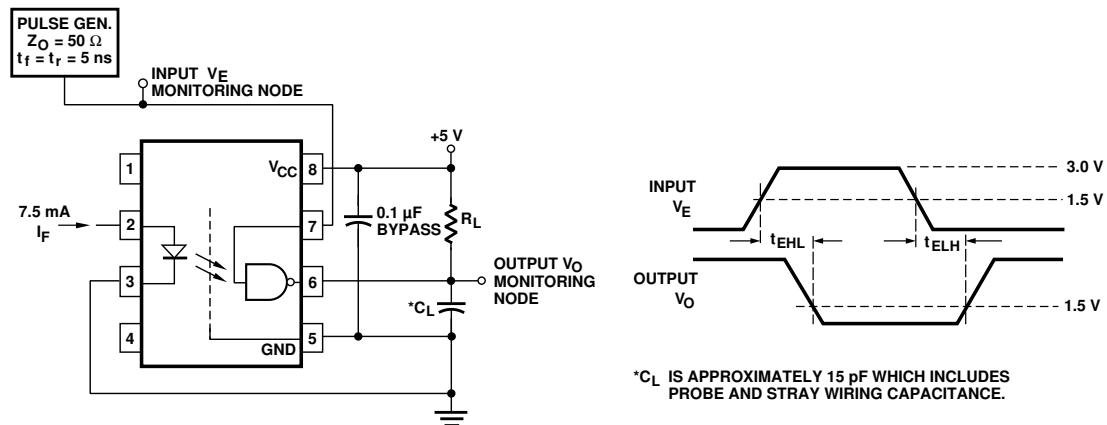


Figure 13. Test Circuit for t_{EHL} and t_{ELH} .

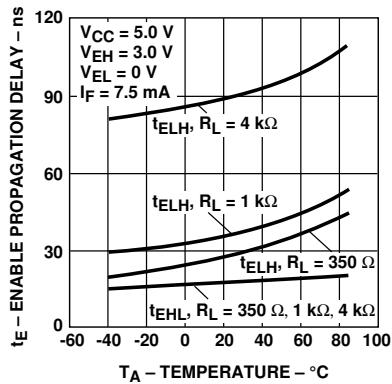


Figure 14. Typical Enable Propagation Delay vs. Temperature.

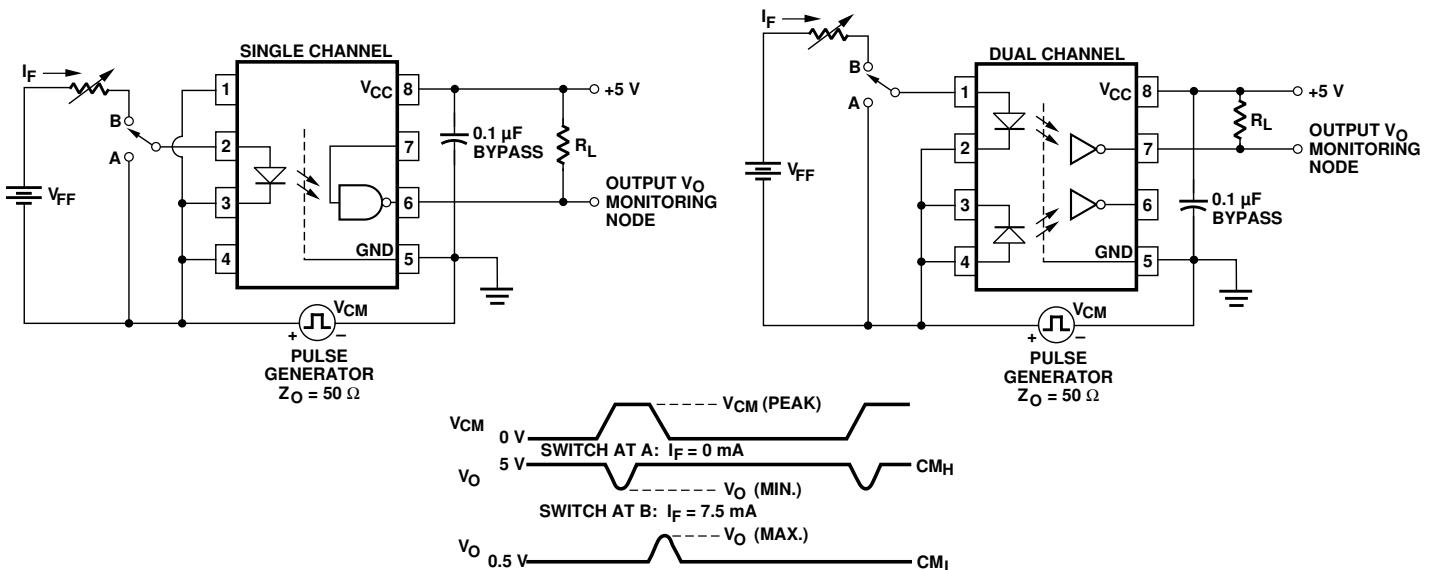


Figure 15. Test Circuit for Common Mode Transient Immunity and Typical Waveforms.

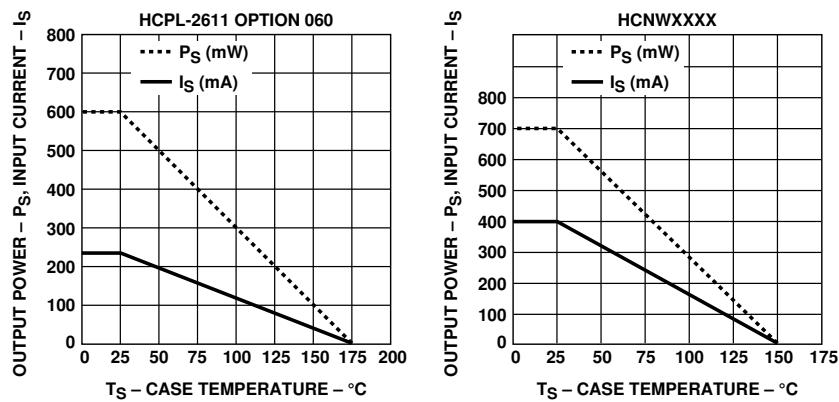


Figure 16. Thermal Derating Curve, Dependence of Safety Limiting Value with Case Temperature per VDE 0884.

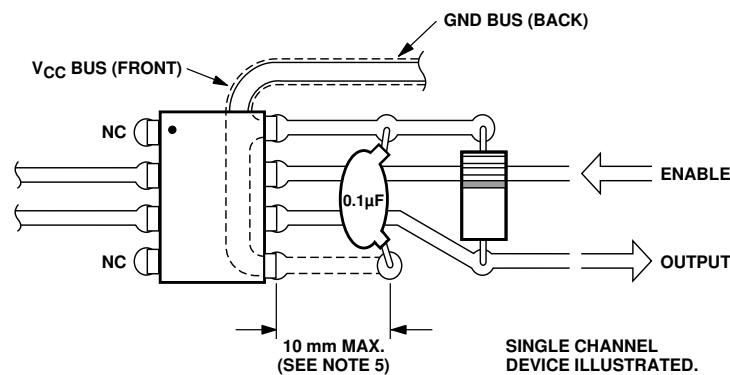
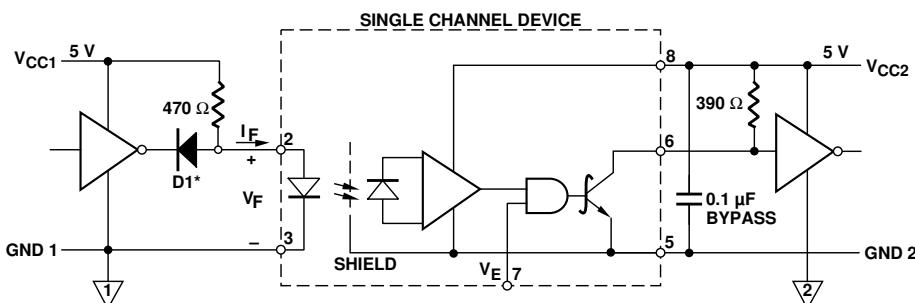


Figure 17. Recommended Printed Circuit Board Layout.



*DIODE D1 (1N916 OR EQUIVALENT) IS NOT REQUIRED FOR UNITS WITH OPEN COLLECTOR OUTPUT.

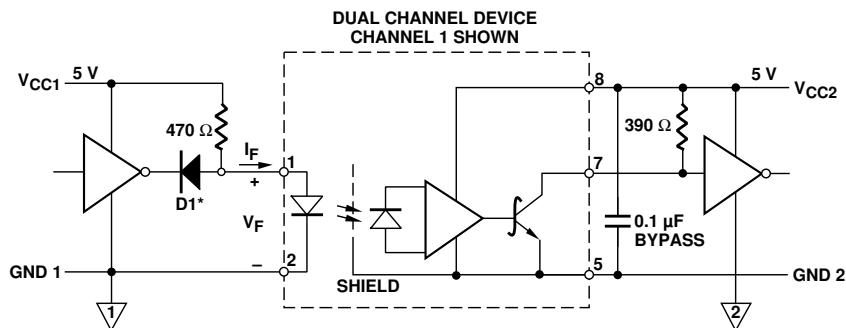


Figure 18. Recommended TTL/LSTTL to TTL/LSTTL Interface Circuit.

Propagation Delay, Pulse-Width Distortion and Propagation Delay Skew

Propagation delay is a figure of merit which describes how quickly a logic signal propagates through a system. The propagation delay from low to high (t_{PLH}) is the amount of time required for an input signal to propagate to the output, causing the output to change from low to high.

Similarly, the propagation delay from high to low (t_{PHL}) is the amount of time required for the input signal to propagate to the output causing the output to change from high to low (see Figure 8).

Pulse-width distortion (PWD) results when t_{PLH} and t_{PHL} differ in value. PWD is defined as the difference between t_{PLH} and t_{PHL} and often determines the maximum data rate capability of a transmission system. PWD can be expressed in percent by dividing the PWD (in ns) by the minimum pulse width (in ns) being transmitted. Typically, PWD on the order of 20-30% of the minimum pulse width is tolerable; the exact figure depends on the particular application (RS232, RS422, T-1, etc.).

Propagation delay skew, t_{PSK} , is an important parameter to consider in parallel data applica-

tions where synchronization of signals on parallel data lines is a concern. If the parallel data is being sent through a group of optocouplers, differences in propagation delays will cause the data to arrive at the outputs of the optocouplers at different times. If this difference in propagation delays is large enough, it will determine the maximum rate at which parallel data can be sent through the optocouplers.

Propagation delay skew is defined as the difference between the minimum and maximum propagation delays, either t_{PLH} or t_{PHL} , for any given group of optocouplers which are operating under the same conditions (i.e., the same drive current, supply voltage, output load, and operating temperature). As illustrated in Figure 19, if the inputs of a group of optocouplers are switched either ON or OFF at the same time, t_{PSK} is the difference between the shortest propagation delay, either t_{PLH} or t_{PHL} , and the longest propagation delay, either t_{PLH} or t_{PHL} .

As mentioned earlier, t_{PSK} can determine the maximum parallel data transmission rate. Figure 20 is the timing diagram of a typical parallel data application with both the clock and the data lines being sent through optocouplers. The figure shows data and clock

signals at the inputs and outputs of the optocouplers. To obtain the maximum data transmission rate, both edges of the clock signal are being used to clock the data; if only one edge were used, the clock signal would need to be twice as fast.

Propagation delay skew represents the uncertainty of where an edge might be after being sent through an optocoupler. Figure 20 shows that there will be uncertainty in both the data and the clock lines. It is important that these two areas of uncertainty not overlap, otherwise the clock signal might arrive before all of the data outputs have settled, or some of the data outputs may start to change before the clock signal has arrived. From these considerations, the absolute minimum pulse width that can be sent through optocouplers in a parallel application is twice t_{PSK} . A cautious design should use a slightly longer pulse width to ensure that any additional uncertainty in the rest of the circuit does not cause a problem.

The t_{PSK} specified optocouplers offer the advantages of guaranteed specifications for propagation delays, pulsedwidth distortion and propagation delay skew over the recommended temperature, input current, and power supply ranges.

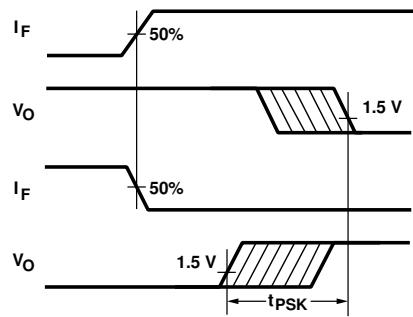


Figure 19. Illustration of Propagation Delay Skew - t_{PSK} .

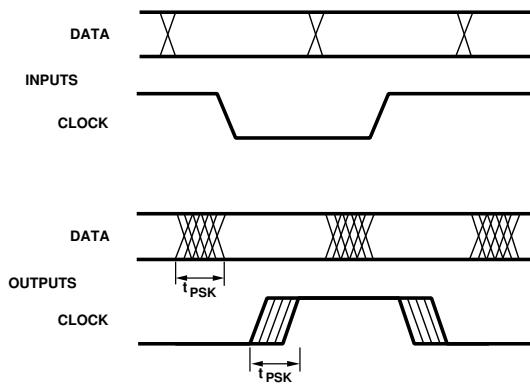


Figure 20. Parallel Data Transmission Example.

TOSHIBA PHOTOCOUPLER GaAs IRED & PHOTO-TRANSISTOR

TLP521-1, TLP521-2, TLP521-4

PROGRAMMABLE CONTROLLERS

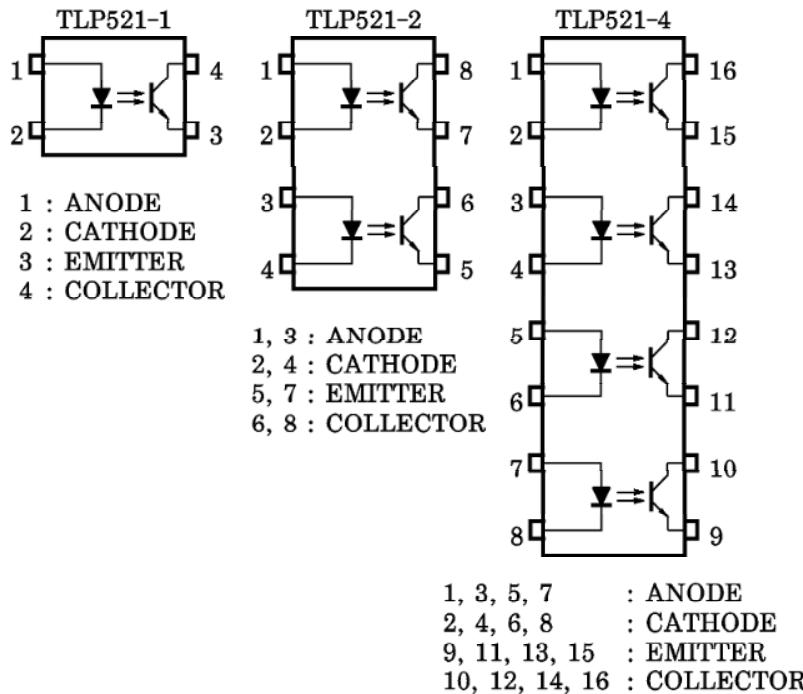
AC / DC-INPUT MODULE

SOLID STATE RELAY

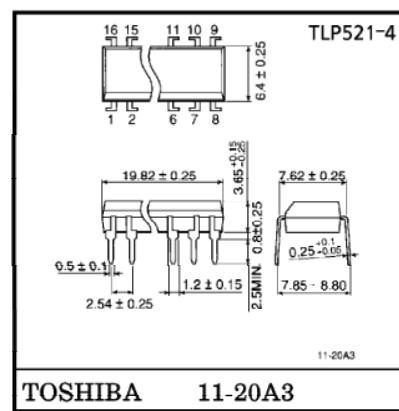
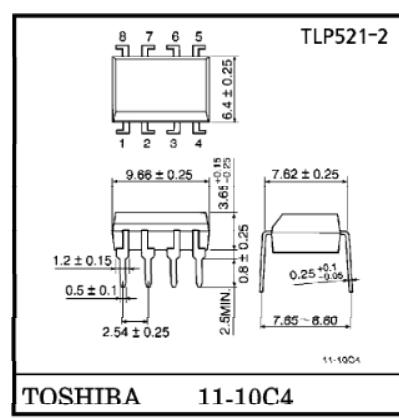
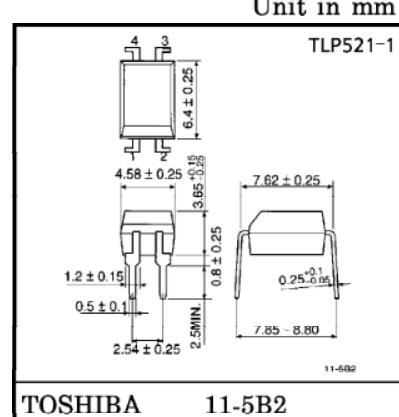
The TOSHIBA TLP521-1, -2 and -4 consist of a photo-transistor optically coupled to a gallium arsenide infrared emitting diode. The TLP521-2 offers two isolated channels in an eight lead plastic DIP package, while the TLP521-4 provides four isolated channels in a sixteen lead plastic DIP package.

- Collector-Emitter Voltage : 55V (Min.)
- Current Transfer Ratio : 50% (Min.)
- Rank GB : 100% (Min.)
- Isolation Voltage : 2500Vrms (Min.)
- UL Recognized
 - made in Japan : UL1577, File No. E67349
 - made in Thailand : UL1577, File No. E152349

PIN CONFIGURATIONS (TOP VIEW)



● TOSHIBA is continually working to improve the quality and the reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to observe standards of safety, and to avoid situations in which a malfunction or failure of a TOSHIBA product could cause loss of human life, bodily injury or damage to property. In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent products specifications. Also, please keep in mind the precautions and conditions set forth in the TOSHIBA Semiconductor Reliability Handbook.



MAXIMUM RATINGS ($T_a = 25^\circ\text{C}$)

	CHARACTERISTIC	SYMBOL	RATING		UNIT
			TLP521-1	TLP521-2 TLP521-4	
LED	Forward Current	I_F	70	50	mA
	Forward Current Derating	$\Delta I_F / ^\circ\text{C}$	-0.93 ($T_a \geq 50^\circ\text{C}$)	-0.5 ($T_a \geq 25^\circ\text{C}$)	mA / $^\circ\text{C}$
	Pulse Forward Current	I_{FP}	1 (100 μ pulse, 100pps)		A
	Reverse Voltage	V_R		5	V
	Junction Temperature	T_j		125	$^\circ\text{C}$
DETECTOR	Collector-Emitter Voltage	V_{CEO}		55	V
	Emitter-Collector Voltage	V_{ECO}		7	V
	Collector Current	I_C		50	mA
	Collector Power Dissipation (1 Circuit)	P_C	150	100	mW
	Collector Power Dissipation Derating (1 Circuit, $T_a \geq 25^\circ\text{C}$)	$\Delta P_C / ^\circ\text{C}$	-1.5	-1.0	mW / $^\circ\text{C}$
	Junction Temperature	T_j		125	$^\circ\text{C}$
	Storage Temperature Range	T_{stg}		-55~125	$^\circ\text{C}$
Operating Temperature Range		T_{opr}		-55~100	$^\circ\text{C}$
Lead Soldering Temperature		T_{sol}		260 (10 sec.)	$^\circ\text{C}$
Total Package Power Dissipation		P_T	250	150	mW
Total Package Power Dissipation Derating ($T_a \geq 25^\circ\text{C}$)		$\Delta P_T / ^\circ\text{C}$	-2.5	-1.5	mW / $^\circ\text{C}$
Isolation Voltage		BVS	2500 (AC, 1 min., R.H. $\leq 60\%$) (Note 1)		Vrms

(Note 1) Device considered a two terminal device : LED side pins shorted together and DETECTOR side pins shorted together.

RECOMMENDED OPERATING CONDITIONS

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply Voltage	V_{CC}	—	5	24	V
Forward Current	I_F	—	16	20	mA
Collector Current	I_C	—	1	10	mA
Operating Temperature	T_{opr}	-25	—	85	$^\circ\text{C}$

961001EBC2'

- Gallium arsenide (GaAs) is a substance used in the products described in this document. GaAs dust and fumes are toxic. Do not break, cut or pulverize the product, or use chemicals to dissolve them. When disposing of the products, follow the appropriate regulations. Do not dispose of the products with other industrial waste or with domestic garbage.
- The products described in this document are subject to foreign exchange and foreign trade control laws.
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA CORPORATION for any infringements of intellectual property or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any intellectual property or other rights of TOSHIBA CORPORATION or others.
- The information contained herein is subject to change without notice.

TYPE	CLASSIFICATION (*1)	CURRENT TRANSFER RATIO (%) (I_C / I_F)		MARKING OF CLASSIFICATION	
		$I_F = 5\text{mA}$, $V_{CE} = 5\text{V}$, $T_a = 25^\circ\text{C}$			
		MIN.	MAX.		
TLP521	A	50	600	BLANK, Y, Y■, G, G■, B, B■, GB	
	Rank Y	50	150	Y, Y■	
	Rank GR	100	300	G, G■	
	Rank BL	200	600	B, B■	
	Rank GB	100	600	G, G■, B, B■, GB	
TLP521-2	A	50	600	BLANK, GR, BL, GB	
TLP521-4	Rank GB	100	600	GR, BL, GB	

*1 : Ex. Rank GB : TLP521-1 (GB)

(Note) Application type name for certification test, please use standard product type name,
i.e.

TLP521-1 (GB) : TLP521-1, TLP521-2 (GB) : TLP521-2

INDIVIDUAL ELECTRICAL CHARACTERISTICS ($T_a = 25^\circ C$)

CHARACTERISTIC		SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
LED	Forward Voltage	V_F	$I_F = 10\text{mA}$	1.0	1.15	1.3	V
	Reverse Current	I_R	$V_R = 5\text{V}$	—	—	10	μA
	Capacitance	C_T	$V = 0, f = 1\text{MHz}$	—	30	—	pF
DETECTOR	Collector-Emitter Breakdown Voltage	$V_{(\text{BR})\text{CEO}}$	$I_C = 0.5\text{mA}$	55	—	—	V
	Emitter-Collector Breakdown Voltage	$V_{(\text{BR})\text{ECO}}$	$I_E = 0.1\text{mA}$	7	—	—	V
	Collector Dark Current	I_{CEO}	$V_{CE} = 24\text{V}$	—	10	100	nA
			$V_{CE} = 24\text{V}, T_a = 85^\circ C$	—	2	50	μA
	Capacitance (Collector to Emitter)	C_{CE}	$V = 0, f = 1\text{MHz}$	—	10	—	pF

COUPLED ELECTRICAL CHARACTERISTICS ($T_a = 25^\circ C$)

CHARACTERISTIC	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Current Transfer Ratio	I_C / I_F	$I_F = 5\text{mA}, V_{CE} = 5\text{V}$ Rank GB	50	—	600	%
			100	—	600	
Saturated CTR	$I_C / I_F(\text{sat})$	$I_F = 1\text{mA}, V_{CE} = 0.4\text{V}$ Rank GB	—	60	—	%
			30	—	—	
Collector-Emitter Saturation Voltage	$V_{CE(\text{sat})}$	$I_C = 2.4\text{mA}, I_F = 8\text{mA}$	—	—	0.4	V
		$I_C = 0.2\text{mA}, I_F = 1\text{mA}$	—	0.2	—	
		Rank GB	—	—	0.4	

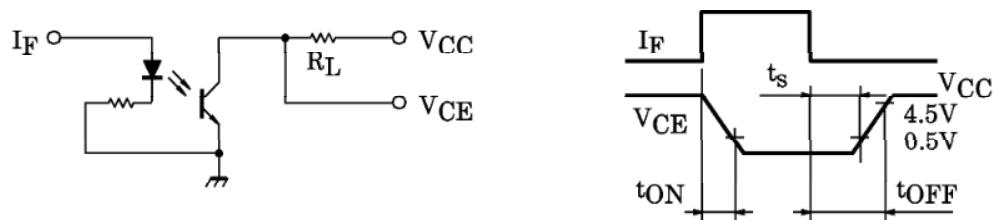
ISOLATION CHARACTERISTICS ($T_a = 25^\circ C$)

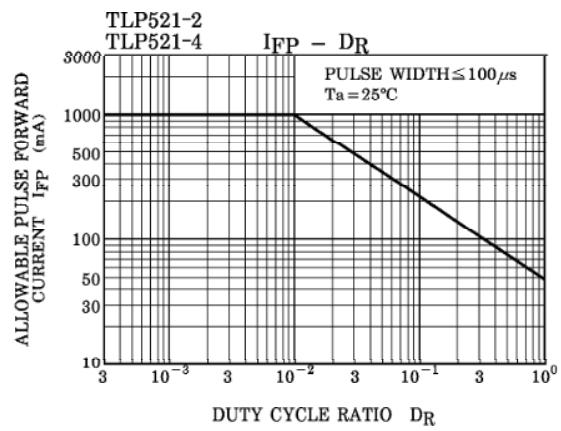
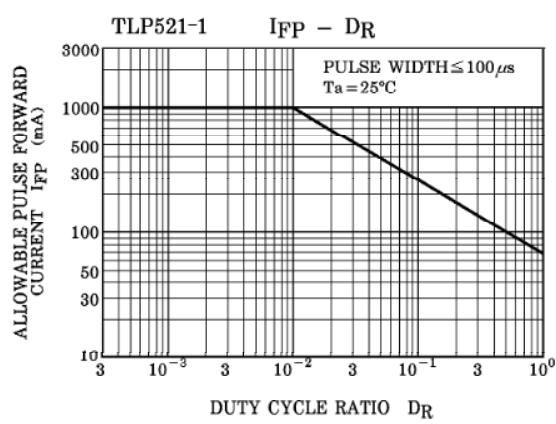
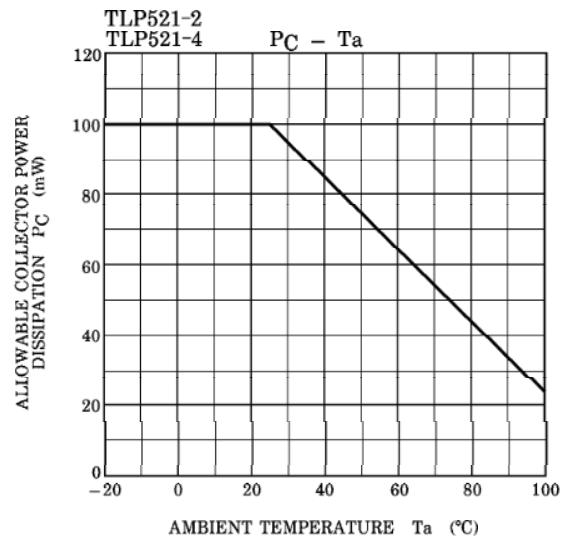
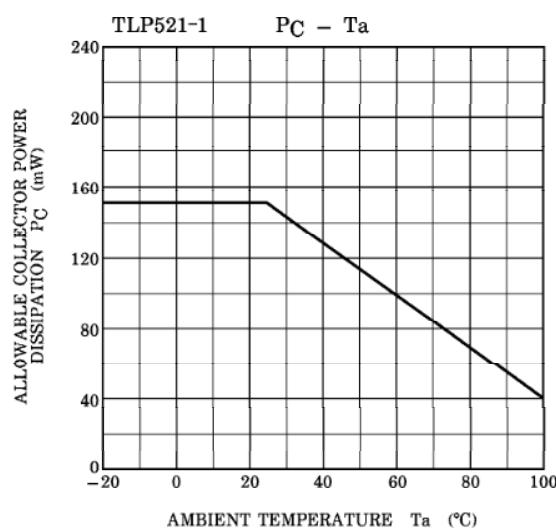
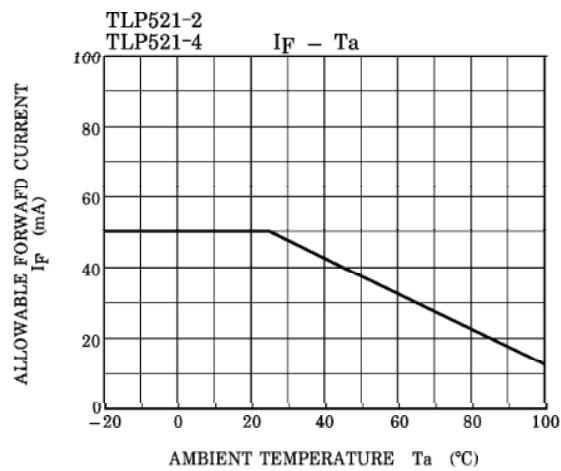
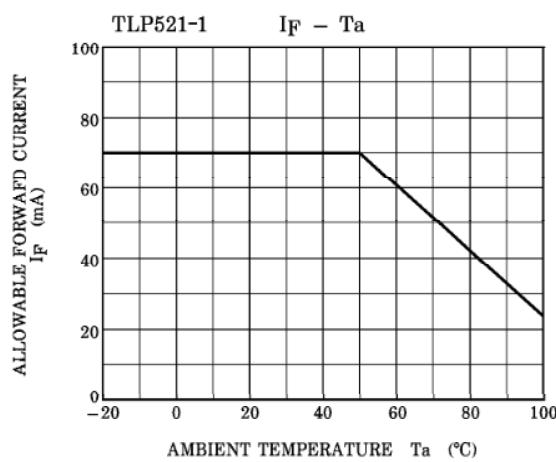
CHARACTERISTIC	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Capacitance (Input to Output)	C_S	$V_S = 0, f = 1\text{MHz}$	—	0.8	—	pF
Isolation Resistance	R_S	$V_S = 500\text{V}, \text{R.H.} \leq 60\%$	—	10^{11}	—	Ω
Isolation Voltage	BVS	AC, 1 minute	2500	—	—	Vrms
		AC, 1 second, in oil	—	5000	—	
		DC, 1 minute, in oil	—	5000	—	Vdc

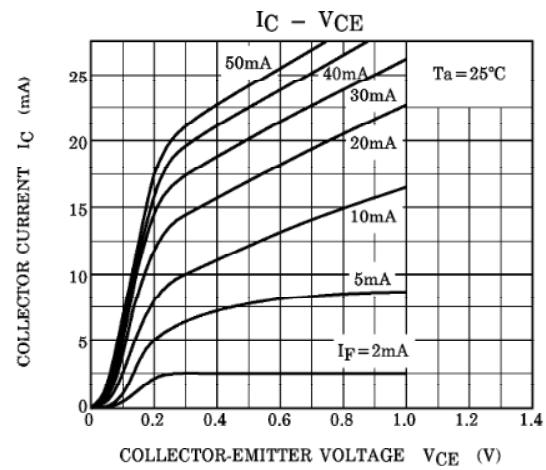
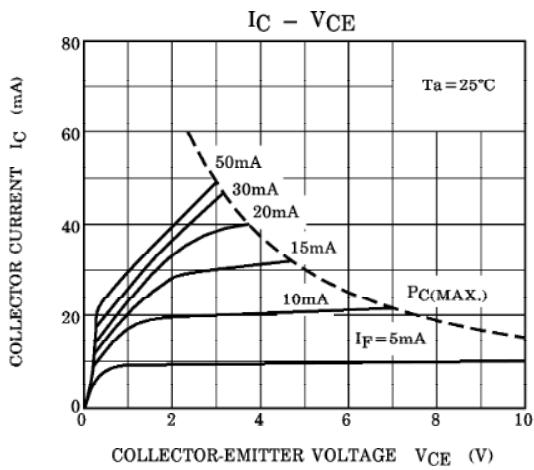
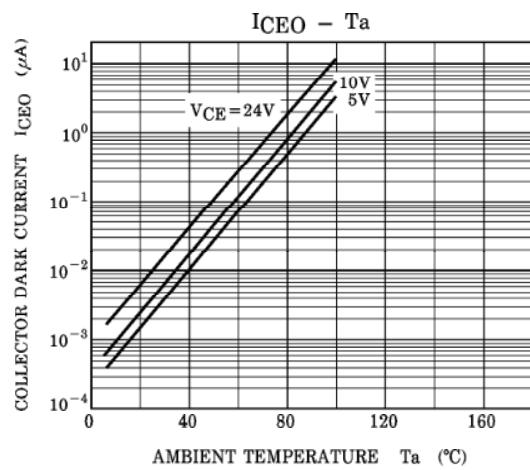
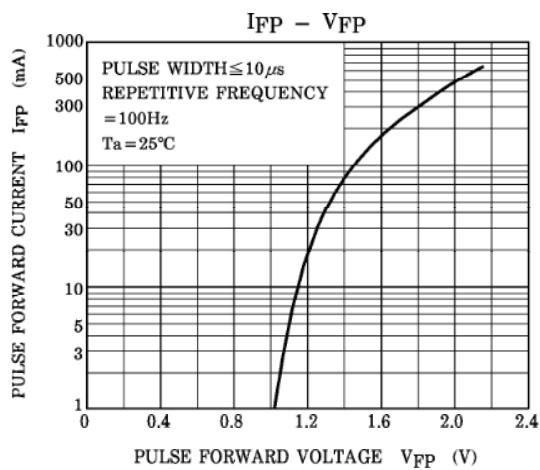
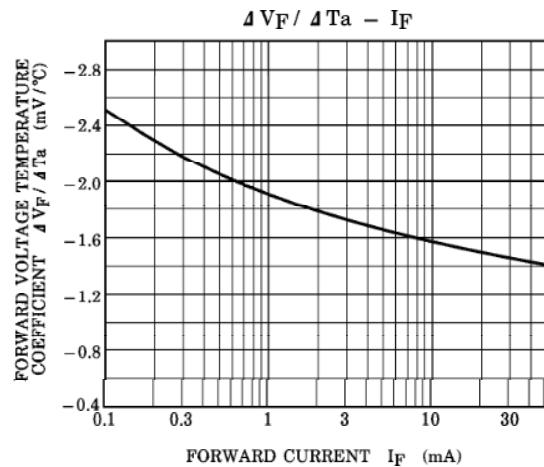
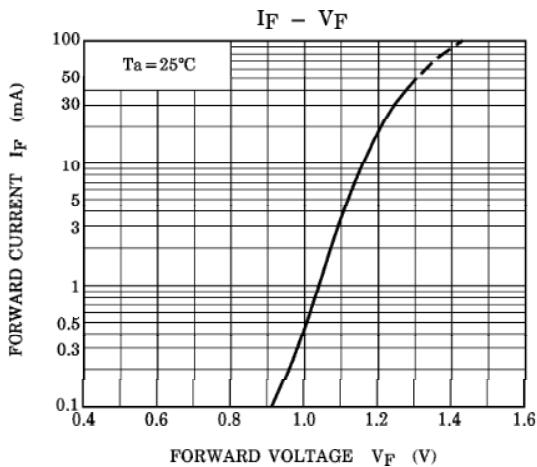
SWITCHING CHARACTERISTICS ($T_a = 25^\circ C$)

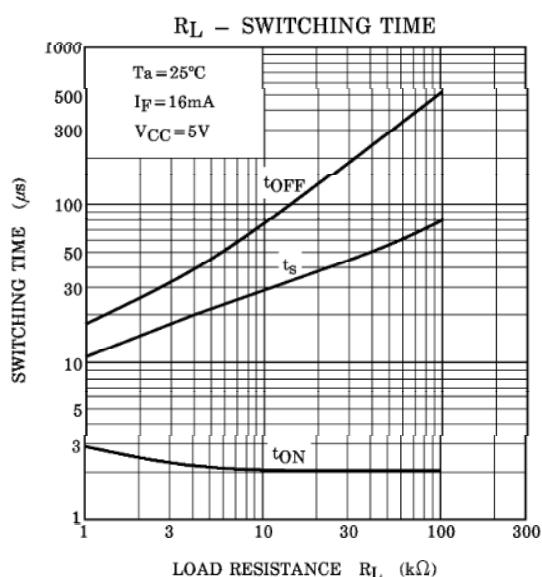
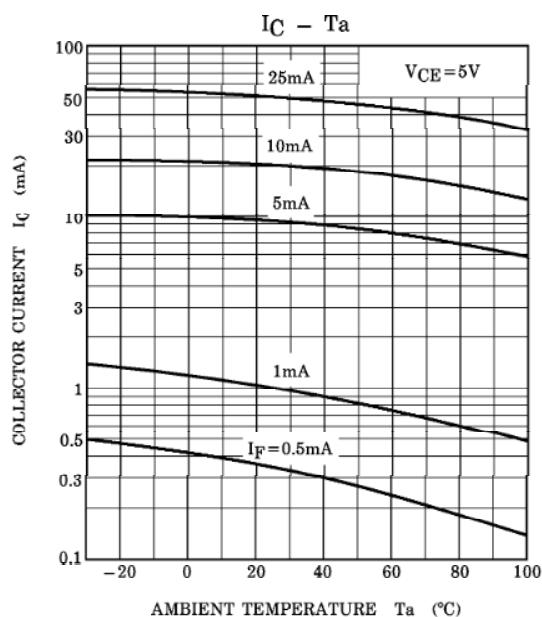
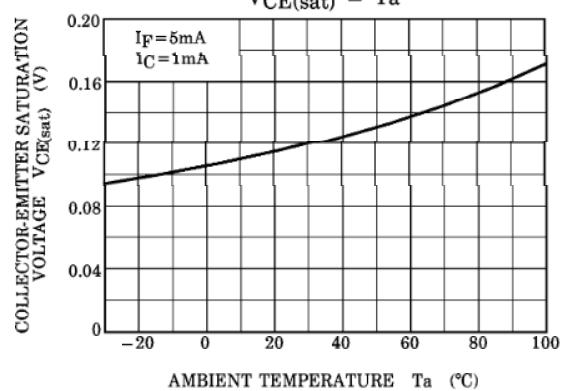
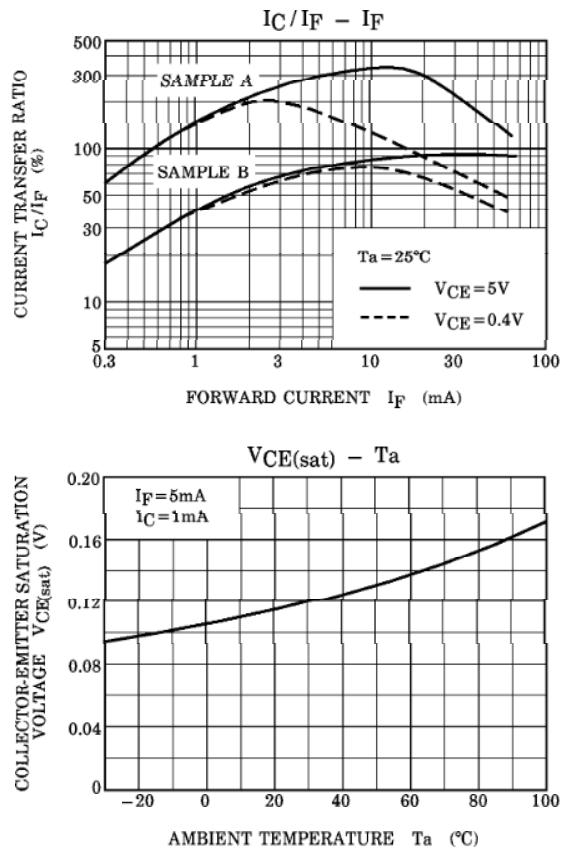
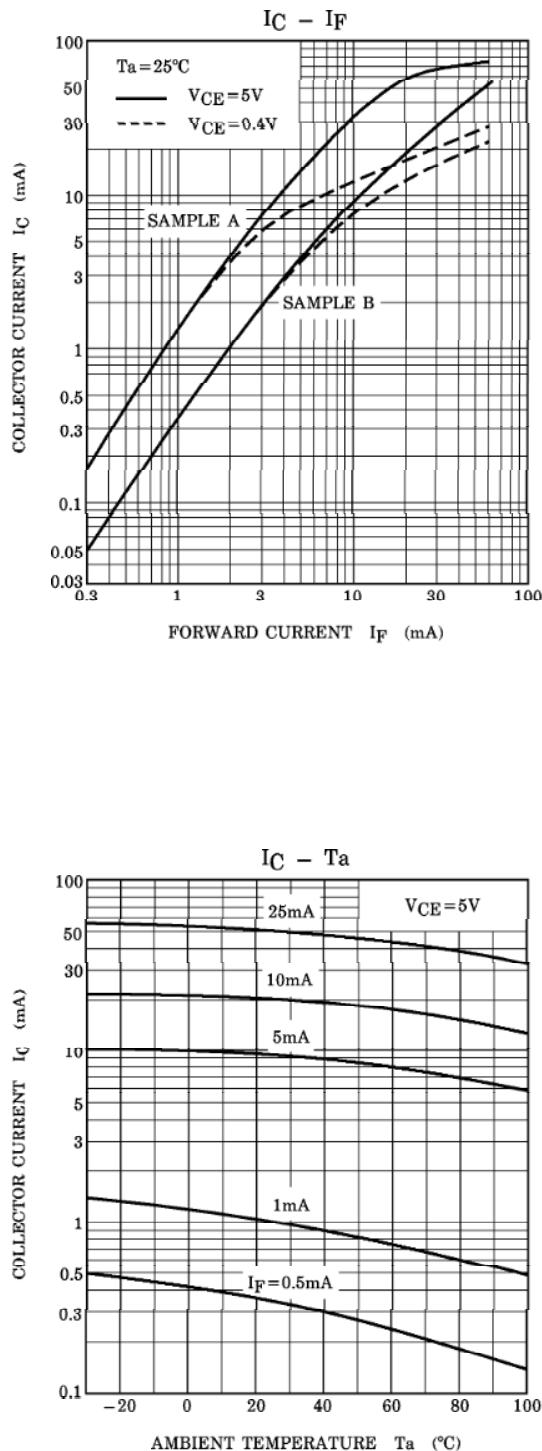
CHARACTERISTIC	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Rise Time	t_r	$V_{CC} = 10V$ $I_C = 2mA$ $R_L = 100\Omega$	—	2	—	μs
Fall Time	t_f		—	3	—	
Turn-on Time	t_{on}		—	3	—	
Turn-off Time	t_{off}		—	3	—	
Turn-on Time	t_{ON}	$R_L = 1.9k\Omega$ (Fig.1) $V_{CC} = 5V$, $I_F = 16mA$	—	2	—	μs
Storage Time	t_s		—	15	—	
Turn-off Time	t_{OFF}		—	25	—	

Fig.1 SWITCHING TIME TEST CIRCUIT









Low saturation voltage type 3-pin regulator

BAOOT / FP series

The BAOOT/ FP series are fixed positive output low drop-out type, 3-pin voltage regulators with positive output.. These regulators are used to provide a stabilized output voltage from a fluctuating DC input voltage. There are 10 fixed output voltages, as follows:3V, 3.3V, 5V, 6V*, 7V, 8V, 9V, 10V, 12V and 15V. The maximum current capacity is 1A for each of the above voltages. (Items marked with an asterisk are under development.)

●Applications

Constant voltage power supply

●Features

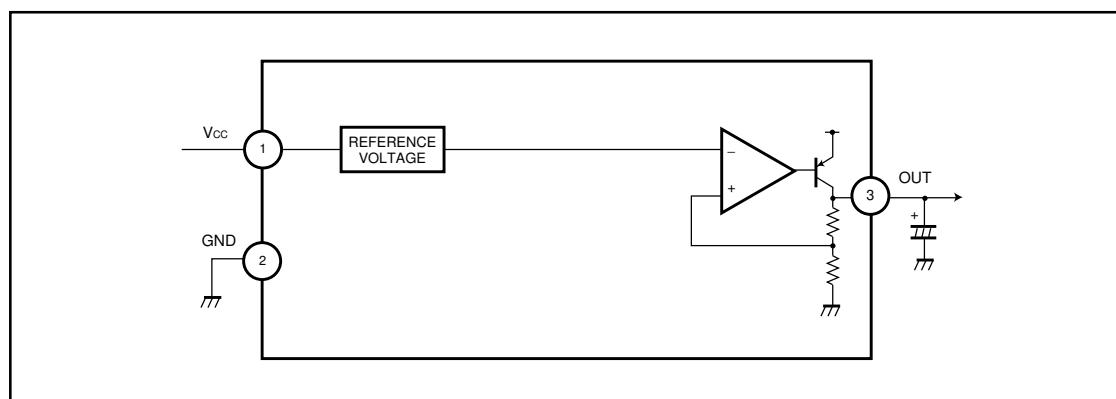
- 1) Built-in overvoltage protection circuit, overcurrent protection circuit and thermal shutdown circuit.
- 2) TO220FP and TO252-3 packages are available to cover a wide range of applications.
- 3) Compatible with the BA17800 series.
- 4) Richly diverse lineup.
- 5) Low minimum I / O voltage differential.

●Product codes

Output voltage (V)	Product No.	Output voltage (V)	Product No.
3.0	BA03T / FP	8.0	BA08T / FP
3.3	BA033T / FP	9.0	BA09T / FP
5.0	BA05T / FP	10.0	BA10T / FP
6.0	BA06T * / FP *	12.0	BA12T / FP
7.0	BA07T / FP	15.0	BA15T / FP

* : Under development.

●Block diagram



Regulator ICs

●Absolute maximum ratings (Ta = 25°C)

Parameter	Symbol	Limits	Unit
Power supply voltage	V _{CC}	35	V
Power dissipation	Pd	2000 ^{*1}	mW
TO252 - 3	Pd	1000 ^{*2}	
Operating temperature	T _{OPR}	-40~85	°C
Storage temperature	T _{STG}	-55~150	°C
Peak applied voltage	V _{SURGE}	50 ^{*3}	V

^{*1} Reduced by 16mW for each increase in Ta of 1°C over 25°C^{*2} Reduced by 8mW for each increase in Ta of 1°C over 25°C^{*3} Voltage application time : 200 msec. or less

●Recommended operating conditions

BA03T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	4	-	25	V
Output current	I _O	-	-	1	A

BA033T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	4.3	-	25	V
Output current	I _O	-	-	1	A

BA05T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	6	-	25	V
Output current	I _O	-	-	1	A

BA06T / FP (under development)

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	7	-	25	V
Output current	I _O	-	-	1	A

BA07T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	8	-	25	V
Output current	I _O	-	-	1	A

BA08T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	9	-	25	V
Output current	I _O	-	-	1	A

BA09T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	10	-	25	V
Output current	I _O	-	-	1	A

BA10T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	11	-	25	V
Output current	I _O	-	-	1	A

BA12T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	13	-	25	V
Output current	I _O	-	-	1	A

BA15T / FP

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input voltage	V _{IN}	16	-	25	V
Output current	I _O	-	-	1	A

BAOOT / FP series

Regulator ICs

●Electrical characteristics

BA03T / FP (unless otherwise noted, Ta = 25°C, Vcc = 8V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{O1}	2.85	3.0	3.15	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 4→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _O	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA033T / FP (unless otherwise noted, Ta = 25°C, Vcc = 8V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{O1}	3.13	3.3	3.47	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 4.3→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _O	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA05T / FP (unless otherwise noted, Ta = 25°C, Vcc = 10V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{O1}	4.75	5.0	5.25	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 6→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 4.75V	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BAOOT / FP series

Regulator ICs

BA06T / FP (unless otherwise noted, Ta = 25°C, Vcc = 11V, Io = 500mA) (under development)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{O1}	5.7	6.0	6.3	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 7 → 25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA → 1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA07T / FP (unless otherwise noted, Ta = 25°C, Vcc = 12V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{O1}	6.65	7.0	7.35	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 8 → 25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA → 1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _O	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA08T / FP (unless otherwise noted, Ta = 25°C, Vcc = 13V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement Circuit
Output voltage	V _{O1}	7.6	8.0	8.4	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 9 → 25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA → 1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _O	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BAOOT / FP series

Regulator ICs

BA09T / FP (unless otherwise noted, Ta = 25°C, Vcc = 14V, Io = 500mA) (under development)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{o1}	8.45	9.0	9.45	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 10→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _o	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA10T / FP (unless otherwise noted, Ta = 25°C, Vcc = 15V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{o1}	9.5	10	10.5	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 11→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _o	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BA12T / FP (unless otherwise noted, Ta = 25°C, Vcc = 17V, Io = 500mA)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V _{o1}	11.4	12	12.6	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	V _{IN} = 13→25V	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	e _{IN} = 1V _{rms} , f = 120Hz, Io = 100mA	Fig.2
Load regulation	Reg.L	-	50	150	mV	Io = 5mA→1A	Fig.1
Temperature coefficient of output voltage	T _{cvo}	-	±0.02	-	% / °C	Io = 5mA, T _j = 0~125°C	Fig.1
Dropout voltage	V _d	-	0.3	0.5	V	V _{CC} = 0.95V _o	Fig.3
Bias current	I _b	-	2.5	5.0	mA	Io = 0mA	Fig.4
Peak output current	I _{o-P}	1.0	1.5	-	A	T _j = 25°C	Fig.1
Output short-circuit current	I _{os}	-	0.4	-	A	V _{CC} = 25V	Fig.5

BAOOT / FP series

Regulator ICs

BA15T / FP (unless otherwise noted, $T_a = 25^\circ\text{C}$, $V_{cc} = 20\text{V}$, $I_o = 500\text{mA}$)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions	Measurement circuit
Output voltage	V_{o1}	14.25	15	15.75	V	-	Fig.1
Input stability	Reg.I	-	20	100	mV	$V_{IN} = 6 \rightarrow 25\text{V}$	Fig.1
Ripple rejection ratio	R.R.	45	55	-	dB	$e_{IN} = 1\text{V}_{rms}$, $f = 120\text{Hz}$, $I_o = 100\text{mA}$	Fig.2
Load regulation	Reg.L	-	90	200	mV	$I_o = 5\text{mA} \rightarrow 1\text{A}$	Fig.1
Temperature coefficient of output voltage	T_{cvo}	-	± 0.02	-	% / $^\circ\text{C}$	$I_o = 5\text{mA}$, $T_j = 0 \sim 125^\circ\text{C}$	Fig.1
Dropout voltage	V_d	-	0.3	0.5	V	$V_{cc} = 0.95V_o$	Fig.3
Bias current	I_b	-	2.5	5.0	mA	$I_o = 0\text{mA}$	Fig.4
Peak output current	I_{o-P}	1.0	1.5	-	A	$T_j = 25^\circ\text{C}$	Fig.1
Output short-circuit current	I_{os}	-	0.4	-	A	$V_{cc} = 30\text{V}$	Fig.5

Regulator ICs

● Measurement circuits

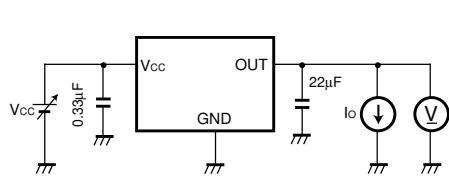
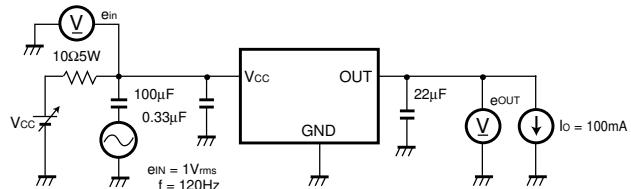


Fig. 1 Measurement circuit for output voltage, input stability, load regulation, temperature coefficient of output voltage



Ripple rejection ratio R.R. = 20 log ($\frac{I_{OINITIAL}}{I_{OLoad}}$)

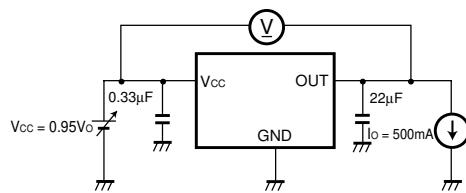


Fig. 3 Measurement circuit for minimum I/O voltage differential

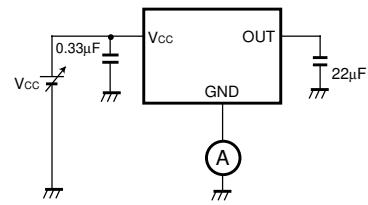


Fig. 4 Measurement circuit for bias current

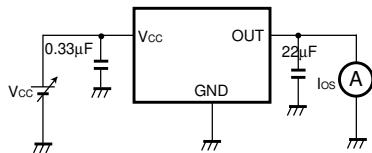


Fig. 5 Measurement circuit for output short-circuit current

Regulator ICs

●Operation notes

(1) Operating power supply voltage

When operating within the normal voltage range and within the ambient operating temperature range, most circuit functions are guaranteed.

The rated values cannot be guaranteed for the electrical characteristics, but there are no sudden changes of the characteristics within these ranges.

(2) Power dissipation

Heat attenuation characteristics are noted on a separate page and can be used as a guide in judging power dissipation.

If these ICs are used in such a way that the allowable power dissipation level is exceeded, an increase in the chip temperature could cause a reduction in the current capability or could otherwise adversely affect the performance of the IC. Make sure a sufficient margin is allowed so that the allowable power dissipation value is not exceeded.

(3) Output oscillation prevention and bypass capacitor

Be sure to connect a capacitor between the output pin and GND to prevent oscillation. Since fluctuations in the valve of the capacitor due to temperature changes may cause oscillations, a tantalum electrolytic capacitor with a small internal series resistance (ESR) is recommended. A $22\mu F$ capacitor is recommended; however, be aware that if an extremely large capacitance is used ($1000\mu F$ or greater), then oscillations may occur at low frequencies. Therefore, be sure to perform the appropriate verifications before selecting the capacitor.

Also, we recommend connecting a $0.33\mu F$ bypass capacitor as close as possible between the input pin and GND.

(4) Overcurrent protection circuit

An overcurrent protection circuit is built into the outputs, to prevent destruction of the IC in the even the load is

shorted.

This protection circuit limits the current in the shape of a '7'. This circuit is designed with a high margin, so that that current is restricted and latching is prevented, even if a high-capacitance capacitor causes a large amount of current to temporary flow through the IC.

However, these protection circuits are only good for preventing damage from sudden accidents and should not be used for continuous protection (for instance, clamping at an output of $1V_F$ or greater; below $1V_F$, the short mode circuit operates). Note that the capacitor has negative temperature characteristics, and the design should take this into consideration.

(5) Thermal overload circuit

A built-in thermal overload circuit prevents damage from overheating. When the thermal circuit is activated, the outputs are turned OFF. When the temperature drops back to a constant level, the circuit is restored.

(6) Internal circuits could be damaged if there are modes in which the electric potential of the application's input (V_{cc}) and GND are the opposite of the electric potential normally used by each of the outputs. Use of a diode or other such bypass path is recommended.

(7) Although the manufacture of this product includes rigorous quality assurance procedures, the product may be damaged if absolute maximum ratings for voltage or operating temperature are exceeded. If damage has occurred, special modes (such as short circuit mode or open circuit mode) cannot be specified. If it is possible that such special modes may be needed, please consider using a fuse or some other mechanical safety measure.

(8) When used within a strong magnetic field, be aware that the possibility of malfunction exists.

BAOOT / FP series

Regulator ICs

● Electrical characteristic curves

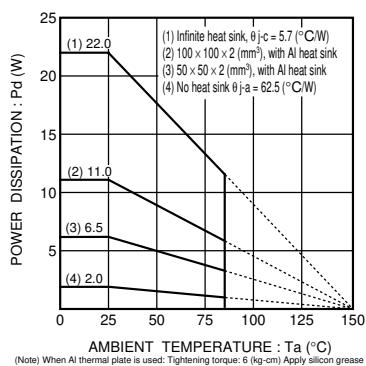


Fig.6 Ta - power dissipation characteristics (TO220FP)

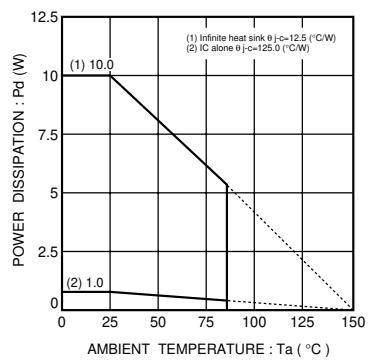


Fig. 7 Ta - power dissipation characteristics
(TO 252-3)

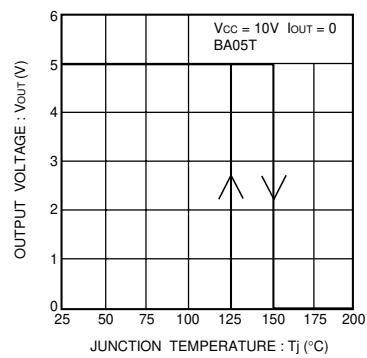


Fig. 8 Thermal cutoff circuit characteristics

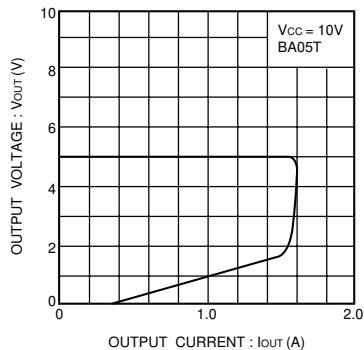


Fig. 9 Current limit characteristics

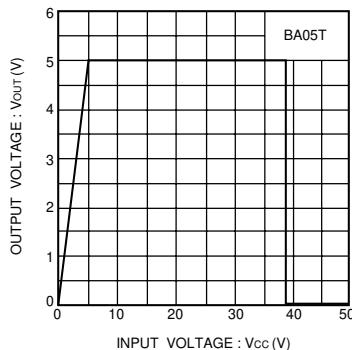
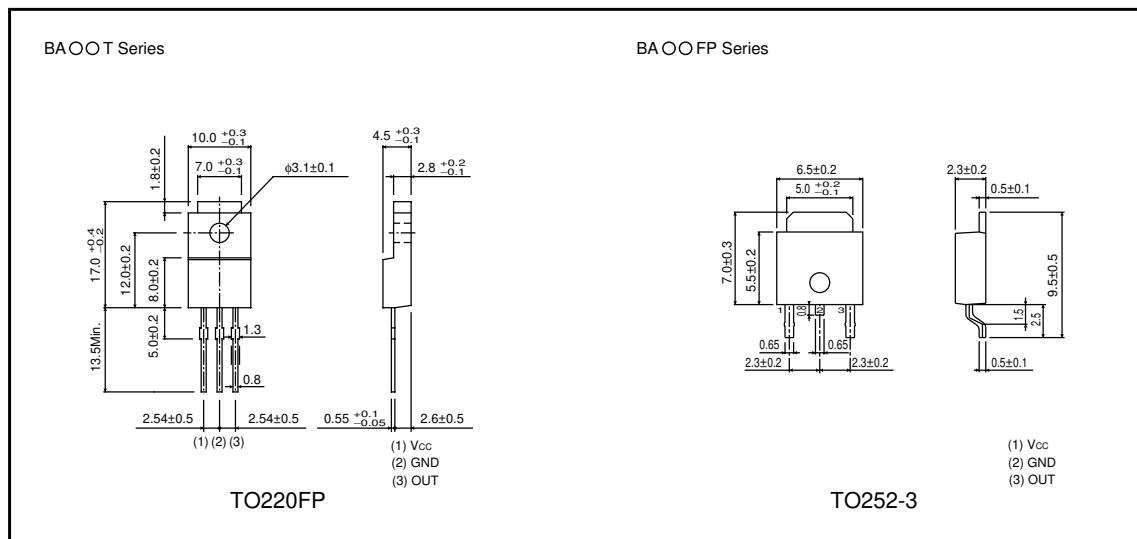


Fig. 10 Over voltage protection characteristics

● **External dimensions** (Units : mm)

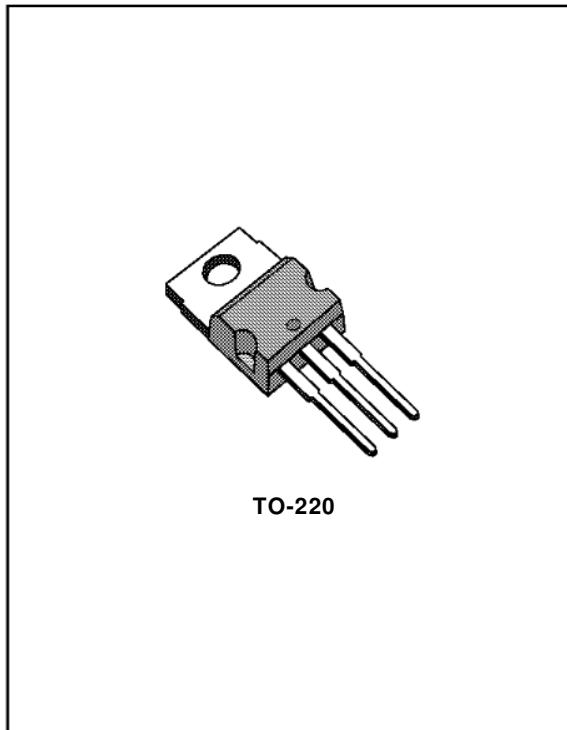


PRECISION 1A REGULATORS

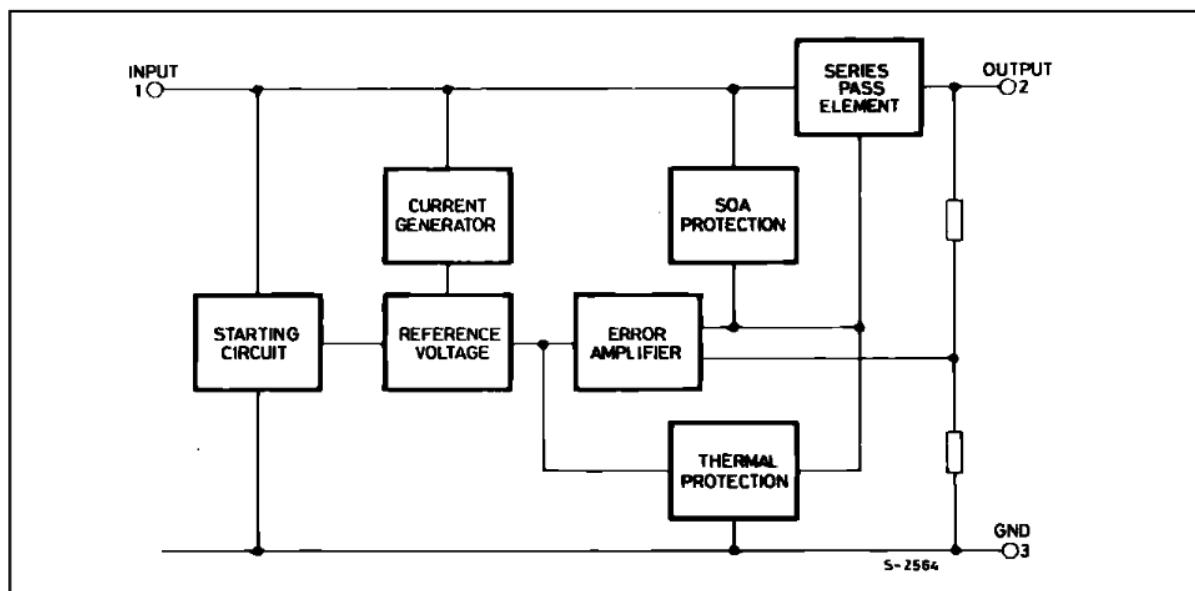
- OUTPUT CURRENT IN EXCESS OF 1A
- OUTPUT VOLTAGES OF 5; 6; 8; 9; 12; 15; 18; 24V
- THERMAL OVERLOAD PROTECTION
- SHORT CIRCUIT PROTECTION
- OUTPUT TRANSISTOR SOA PROTECTION
- 2% OUTPUT VOLTAGE TOLERANCE
- GUARANTEED IN EXTENDED TEMPERATURE RANGES

DESCRIPTION

The L7800A series of three-terminal positive regulators is available in TO-220 and TO-3 packages and with several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



BLOCK DIAGRAM



L7800AB/AC SERIES

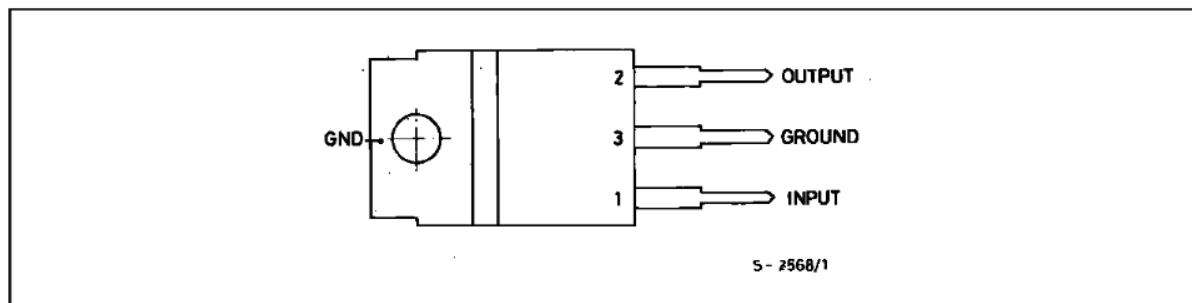
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_i	DC Input Voltage (for $V_o = 5$ to 18V) (for $V_o = 24V$)	35 40	V
I_o	Output Current	Internally limited	
P_{tot}	Power Dissipation	Internally limited	
T_j	Operating Junction Temperature for L7800AC for L7800AB	0 to 125 - 40 to 125	°C
T_{stg}	Storage Temperature	- 65 to + 150	°C

THERMAL DATA

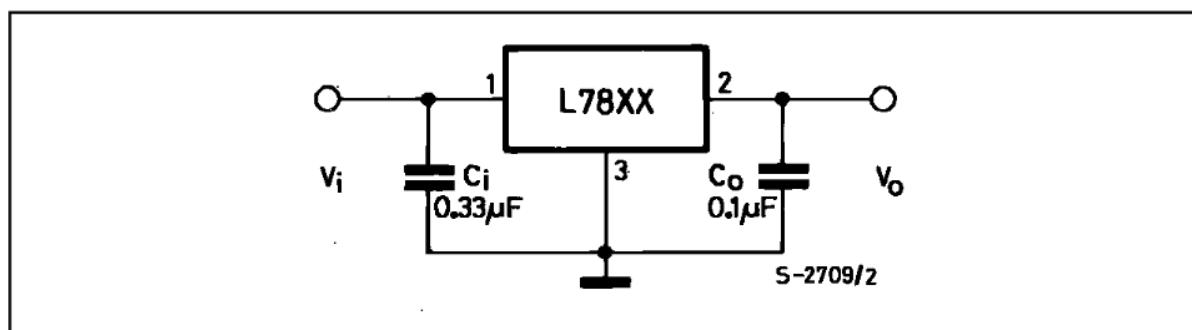
Symbol	Parameter	Value	Unit
$R_{thj-case}$	Thermal Resistance Junction-case	3	°C/W
$R_{thj-amb}$	Thermal Resistance Junction-ambient	50	°C/W

CONNECTION DIAGRAM AND ORDERING NUMBERS (top view)

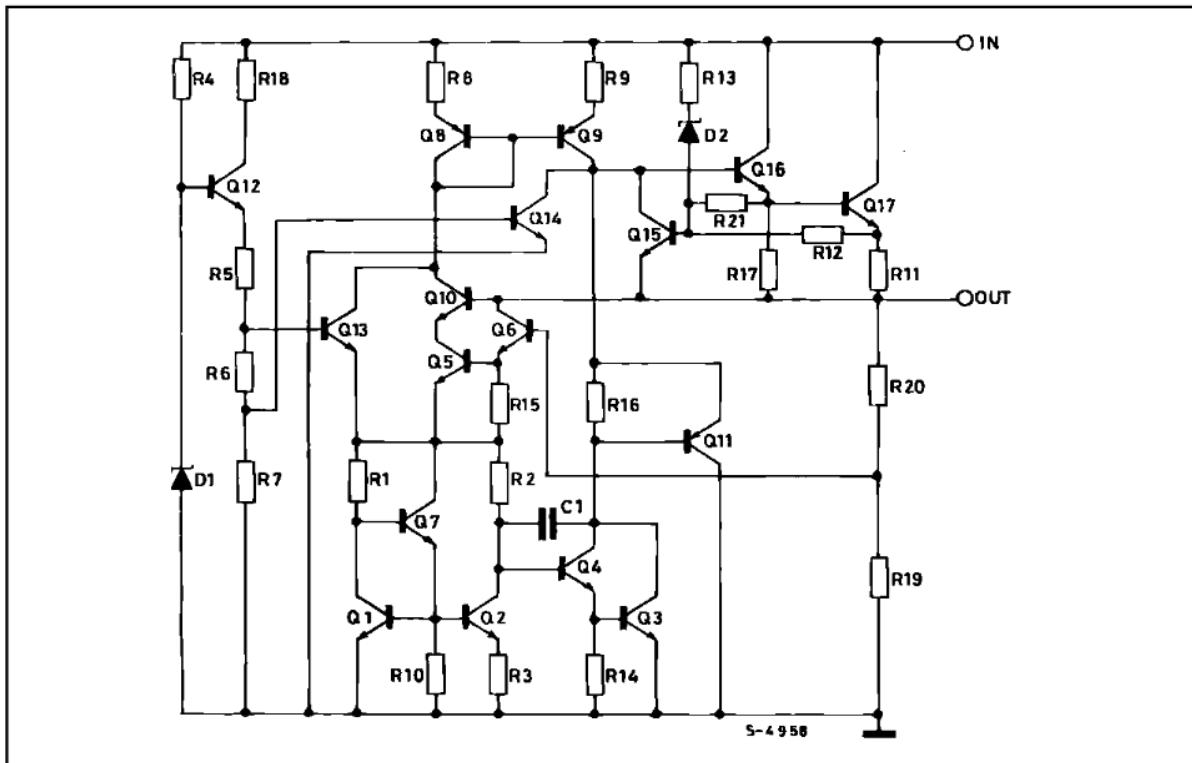


TO-220 ($T_j = -40$ to 125 °C)	TO-220 ($T_j = 0$ to 125 °C)	Output Voltage
L7805ABV	L7805ACV	5V
L7806ABV	L7806ACV	6V
L7808ABV	L7808ACV	8V
L7809ABV	L7809ACV	9V
L7812ABV	L7812ACV	12V
L7815ABV	L7815ACV	15V
L7818ABV	L7818ACV	18V
L7824ABV	L7824ACV	24V

TYPICAL APPLICATION



SCHEMATIC DIAGRAM



TEST CIRCUITS

Figure 1 : DC Parameters.

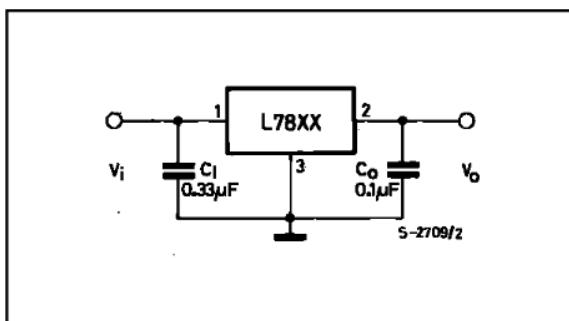


Figure 2 : Load Regulation.

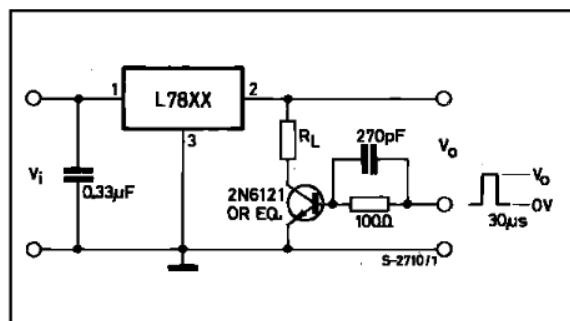
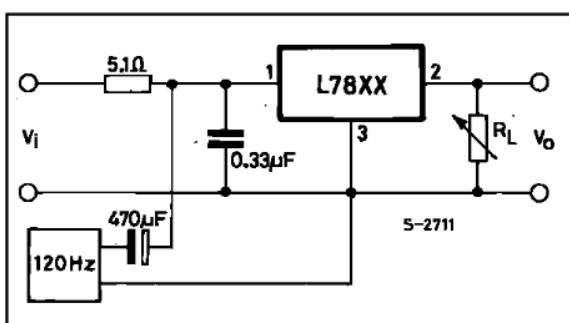


Figure 3 : Ripple Rejection.



L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7805A ($V_i = 10V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7805AC), $T_j = -40$ to $125^\circ C$ (L7805AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	4.9	5	5.1	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 7.5 \text{ to } 20 \text{ V}$	4.8	5	5.2	V
ΔV_o^*	Line Regulation	$V_i = 7.5 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 8 \text{ to } 12 \text{ V}$ $V_i = 8 \text{ to } 12 \text{ V}$ $T_j = 25^\circ C$ $V_i = 7.3 \text{ to } 20 \text{ V}$ $T_j = 25^\circ C$		7 10 2 7	50 5 25 50	mV mV mV mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 8	100 100 50	mV mV mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.3	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 8 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 7.5 \text{ to } 20 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA mA mA
SVR	Supply Voltage Rejection	$V_i = 8 \text{ to } 18 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		68		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		17		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-1.1		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7806A ($V_i = 11V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7806AC), $T_j = -40$ to $125^\circ C$ (L7806AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	5.88	6	6.12	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 8.6 \text{ to } 21 \text{ V}$	5.76	6	6.24	V
ΔV_o^*	Line Regulation	$V_i = 8.6 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 9 \text{ to } 13 \text{ V}$ $V_i = 9 \text{ to } 13 \text{ V}$ $T_j = 25^\circ C$ $V_i = 8.3 \text{ to } 21 \text{ V}$ $T_j = 25^\circ C$		9 11 3 9	60 60 30 60	mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.3	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 9 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 8.6 \text{ to } 21 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA
SVR	Supply Voltage Rejection	$V_i = 9 \text{ to } 19 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		65		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		17		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-0.8		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7808A ($V_i = 14V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7808AC), $T_j = -40$ to $125^\circ C$ (L7808AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	7.84	8	8.16	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 10.6 \text{ to } 23 \text{ V}$	7.7	8	8.3	V
ΔV_o^*	Line Regulation	$V_i = 10.6 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 11 \text{ to } 17 \text{ V}$ $V_i = 11 \text{ to } 17 \text{ V}$ $T_j = 25^\circ C$ $V_i = 10.4 \text{ to } 23 \text{ V}$ $T_j = 25^\circ C$		12 15 5 12	80 80 40 80	mV mV mV mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV mV mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.3	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 11 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 10.6 \text{ to } 23 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA mA mA
SVR	Supply Voltage Rejection	$V_i = 11.5 \text{ to } 21.5 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		62		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		18		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-0.8		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7809A ($V_i = 15V$, $I_o = 1 A$, $T_j = 0$ to $125^{\circ}C$ (L7809AC), $T_j = -40$ to $125^{\circ}C$ (L7809AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^{\circ}C$	8.82	9	9.18	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 10.6 \text{ to } 23 \text{ V}$	8.65	9	9.35	V
ΔV_o^*	Line Regulation	$V_i = 10.6 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 11 \text{ to } 17 \text{ V}$ $V_i = 11 \text{ to } 17 \text{ V}$ $T_j = 25^{\circ}C$ $V_i = 10.4 \text{ to } 23 \text{ V}$ $T_j = 25^{\circ}C$		12 15 5 12	90 90 45 90	mV mV mV mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^{\circ}C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV mV mV
I_d	Quiescent Current	$T_j = 25^{\circ}C$		4.3	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 11 \text{ to } 25 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 10.6 \text{ to } 23 \text{ V}$ $T_j = 25^{\circ}C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA mA mA
SVR	Supply Voltage Rejection	$V_i = 11.5 \text{ to } 21.5 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		61		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^{\circ}C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^{\circ}C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		18		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^{\circ}C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^{\circ}C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-0.8		$\text{mV}/^{\circ}\text{C}$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7812A ($V_i = 19V$, $I_o = 1 A$, $T_j = 0$ to $125^{\circ}C$ (L7812AC), $T_j = -40$ to $125^{\circ}C$ (L7812AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^{\circ}C$	11.75	12	12.25	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 14.8 \text{ to } 27 \text{ V}$	11.5	12	12.5	V
ΔV_o^*	Line Regulation	$V_i = 14.8 \text{ to } 30 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 16 \text{ to } 22 \text{ V}$ $V_i = 16 \text{ to } 22 \text{ V}$ $T_j = 25^{\circ}C$ $V_i = 14.5 \text{ to } 27 \text{ V}$ $T_j = 25^{\circ}C$		13 16 6 13	120 120 60 120	mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^{\circ}C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV
I_d	Quiescent Current	$T_j = 25^{\circ}C$		4.4	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 15 \text{ to } 30 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 14.8 \text{ to } 27 \text{ V}$ $T_j = 25^{\circ}C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA
SVR	Supply Voltage Rejection	$V_i = 15 \text{ to } 25 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		60		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^{\circ}C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^{\circ}C$		10		$\mu\text{V}/\text{V}_o$
R_o	Output Resistance	$f = 1\text{KHz}$		18		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^{\circ}C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^{\circ}C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-1		$\text{mV}/^{\circ}\text{C}$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7815A ($V_i = 23V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7815AC), $T_j = -40$ to $125^\circ C$ (L7815AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	14.7	15	15.3	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 17.9 \text{ to } 30 \text{ V}$	14.4	15	15.6	V
ΔV_o^*	Line Regulation	$V_i = 17.9 \text{ to } 30 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 20 \text{ to } 26 \text{ V}$ $V_i = 20 \text{ to } 26 \text{ V}$ $T_j = 25^\circ C$ $V_i = 17.5 \text{ to } 30 \text{ V}$ $T_j = 25^\circ C$		13 16 6 13	150 150 75 150	mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.4	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 17.5 \text{ to } 30 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 17.5 \text{ to } 30 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA
SVR	Supply Voltage Rejection	$V_i = 18.5 \text{ to } 28.5 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		58		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		19		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-1		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7818A ($V_i = 27V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7818AC), $T_j = -40$ to $125^\circ C$ (L7818AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	17.64	18	18.36	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 21 \text{ to } 33 \text{ V}$	17.3	18	18.7	V
ΔV_o^*	Line Regulation	$V_i = 21 \text{ to } 33 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 24 \text{ to } 30 \text{ V}$ $V_i = 24 \text{ to } 30 \text{ V}$ $T_j = 25^\circ C$ $V_i = 20.6 \text{ to } 33 \text{ V}$ $T_j = 25^\circ C$		25 28 10 5	180 180 90 180	mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.5	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 21 \text{ to } 33 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 21 \text{ to } 33 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA
SVR	Supply Voltage Rejection	$V_i = 22 \text{ to } 32 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		57		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		19		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-1		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

ELECTRICAL CHARACTERISTICS FOR L7824A ($V_i = 33V$, $I_o = 1 A$, $T_j = 0$ to $125^\circ C$ (L7824AC), $T_j = -40$ to $125^\circ C$ (L7824AB) unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	$T_j = 25^\circ C$	23.5	24	24.5	V
V_o	Output Voltage	$I_o = 5 \text{ mA to } 1 \text{ A}$ $P_o \leq 15 \text{ W}$ $V_i = 27.3 \text{ to } 38 \text{ V}$	23	24	25	V
ΔV_o^*	Line Regulation	$V_i = 27 \text{ to } 38 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 30 \text{ to } 36 \text{ V}$ $V_i = 30 \text{ to } 36 \text{ V}$ $T_j = 25^\circ C$ $V_i = 26.7 \text{ to } 38 \text{ V}$ $T_j = 25^\circ C$		31 35 14 31	240 240 120 240	mV
ΔV_o^*	Load Regulation	$I_o = 5 \text{ mA to } 1 \text{ A}$ $I_o = 5 \text{ mA to } 1.5 \text{ A}$ $T_j = 25^\circ C$ $I_o = 250 \text{ to } 750 \text{ mA}$		25 30 10	100 100 50	mV
I_d	Quiescent Current	$T_j = 25^\circ C$		4.6	6 6	mA
ΔI_d	Quiescent Current Change	$V_i = 27.3 \text{ to } 38 \text{ V}$ $I_o = 500 \text{ mA}$ $V_i = 27.3 \text{ to } 38 \text{ V}$ $T_j = 25^\circ C$ $I_o = 5 \text{ mA to } 1 \text{ A}$			0.8 0.8 0.5	mA
SVR	Supply Voltage Rejection	$V_i = 28 \text{ to } 38 \text{ V}$ $f = 120 \text{ Hz}$ $I_o = 500 \text{ mA}$		54		dB
V_d	Dropout Voltage	$I_o = 1 \text{ A}$ $T_j = 25^\circ C$		2		V
e_N	Output Noise Voltage	$B = 10\text{Hz to } 100\text{KHz}$ $T_j = 25^\circ C$		10		$\mu\text{V}/V_o$
R_o	Output Resistance	$f = 1\text{KHz}$		20		$\text{m}\Omega$
I_{sc}	Short Circuit Current	$V_i = 35 \text{ V}$ $T_{amb} = 25^\circ C$		0.2		A
I_{scp}	Short Circuit Peak Current	$T_j = 25^\circ C$		2.2		A
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift			-1.5		$\text{mV}/^\circ C$

* Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

L7800AB/AC SERIES

APPLICATIONS INFORMATION

DESIGN CONSIDERATIONS

The L7800A Series of fixed voltage regulators are designed with Thermal Overload Protection that shuts down the circuit when subjected to an excessive power overload condition, Internal Short-circuit Protection that limits the maximum current the circuit will pass, and Output Transistor Safe-Area Compensation that reduces the output short-circuit current as the voltage across the pass transistor is increased.

In many low current applications, compensation capacitors are not required. However, it is recommended that the regulator input be bypassed with a

capacitor if the regulator is connected to the power supply filter with long wire lengths, or if the output load capacitance is large. An input bypass capacitor should be selected to provide good high-frequency characteristics to insure stable operation under all load conditions. A $0.33\mu F$ or larger tantalum, mylar, or other capacitor having low internal impedance at high frequencies should be chosen. The bypass capacitor should be mounted with the shortest possible leads directly across the regulators input terminals. Normally good construction techniques should be used to minimize ground loops and lead resistance drops since the regulator has no external sense lead.

Figure 4 : Current Regulator.

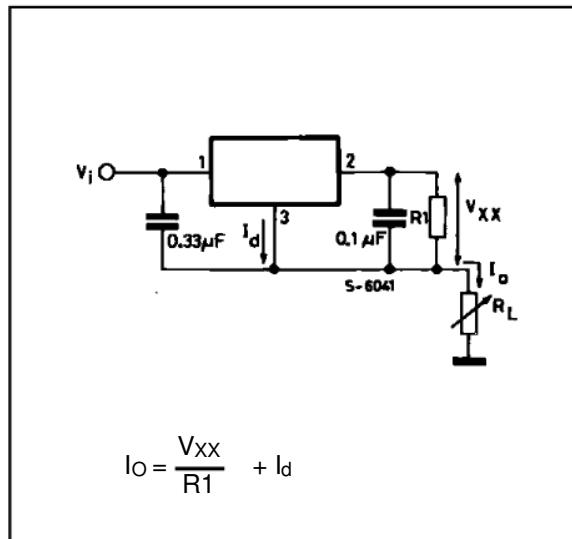


Figure 6 : Current Boost Regulator.

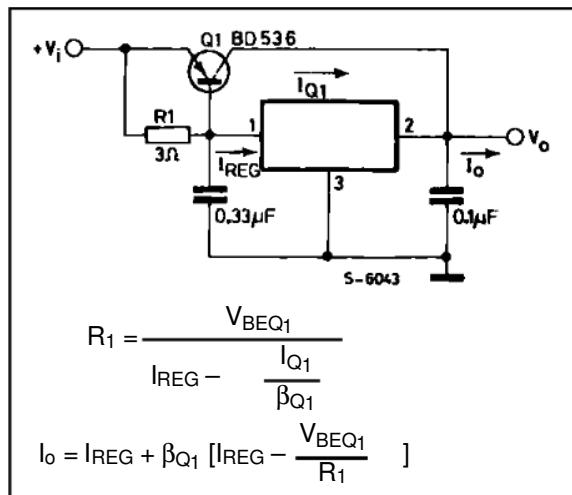
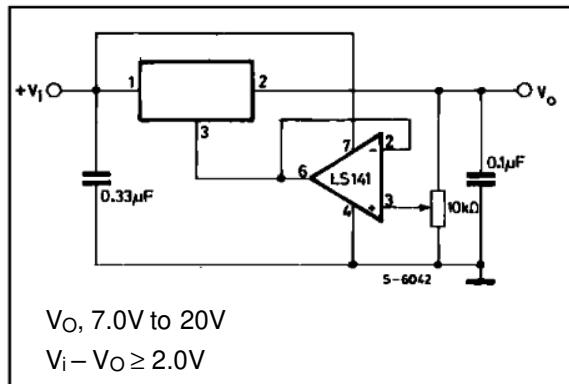
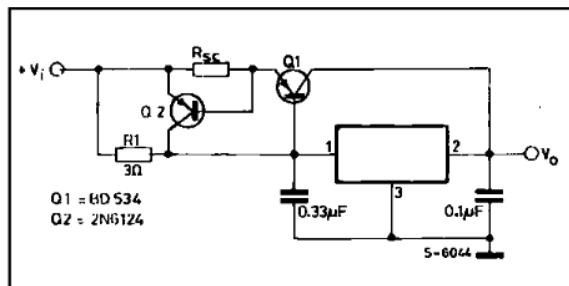


Figure 5 : Adjustable Output Regulator.



The addition of an operational amplifier allows adjustment to higher or intermediate values while retaining regulation characteristics. The minimum voltage obtainable with this arrangement is 2.0V greater than the regulator voltage.

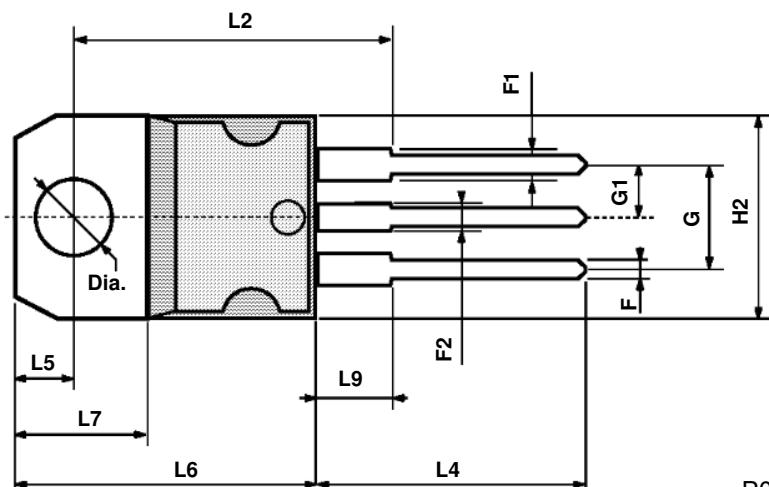
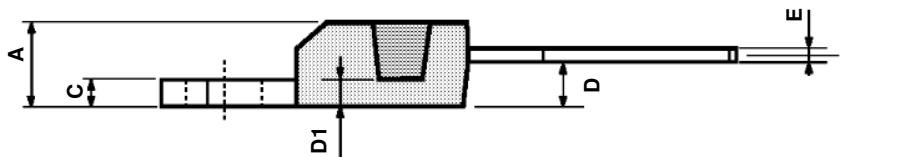
Figure 7 : Short-circuit Protection.



The circuit of figure 6 can be modified to provide supply protection against short circuit by adding a short-circuit sense resistor, R_{sc} , and an additional PNP transistor. The current sensing PNP must be able to handle the short-circuit current of the three-terminal regulator. Therefore, a four-ampere plastic power transistor is specified.

TO-220 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.40		4.60	0.173		0.181
C	1.23		1.32	0.048		0.051
D	2.40		2.72	0.094		0.107
D1		1.27			0.050	
E	0.49		0.70	0.019		0.027
F	0.61		0.88	0.024		0.034
F1	1.14		1.70	0.044		0.067
F2	1.14		1.70	0.044		0.067
G	4.95		5.15	0.194		0.203
G1	2.4		2.7	0.094		0.106
H2	10.0		10.40	0.393		0.409
L2		16.4			0.645	
L4	13.0		14.0	0.511		0.551
L5	2.65		2.95	0.104		0.116
L6	15.2		15.9	0.598		0.625
L7	6.2		6.6	0.244		0.260
L9	3.5		4.2	0.137		0.165
DIA.	3.75		3.85	0.147		0.151



P011C

L7800AB/AC SERIES

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of SGS-THOMSON Microelectronics.

© 1994 SGS-THOMSON Microelectronics - All Rights Reserved

SGS-THOMSON Microelectronics GROUP OF COMPANIES

Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands -
Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A

Appendice A6

La RoboCup e la squadra ART

1. La RoboCup



La Robot World Cup Iniziative, *Robocup*, è un tentativo di promuovere la ricerca in intelligenza artificiale, la robotica ed altri settori correlati ponendosi un obiettivo ambizioso: entro il 2050, una squadra di robot umanoidi, completamente automatici, sarà in grado di vincere un incontro di calcio contro la squadra campione del mondo, secondo le regole approvate dalla FIFA (l'organizzazione che governa il calcio mondiale) [Asada e Kitano, 1999].

L'idea della RoboCup è nata nel 1993 da un gruppo di ricercatori giapponesi che hanno ritenuto utile iniziare la ricerca con robot che giocano tra di loro. Così, per facilitare lo scambio di conoscenze e la valutazione dei progressi scientifici, si è tenuta nel 1997 a Nagoya, in Giappone, la prima Robot World Cup Soccer Games a cui è seguita l'edizione del 1998 a Parigi, quella del 1999 a Stoccolma e quella del 2000 a Melbourne.

Attualmente la RoboCup prevede diversi tipi di competizioni: vi sono sia tornei per robot reali, sia una gara per simulatori, che sfrutta la simulazione software di robot virtuali. Le gare per robot reali sono a loro volta suddivise in diversi campionati:

- *Categoria piccoli robot.* A questa categoria partecipano squadre composte da cinque robot, ciascuno dei quali non può eccedere i 15 centimetri di diametro. Il campo è grande come un tavolo da ping pong, e il pallone è costituito da una pallina da golf. Si usa una telecamera disposta sul soffitto

per guardare il campo dall'alto, il che permette al controllo software di gestire la posizione del robot. Questa categoria è stata scelta come terreno di prova per la ricerca nel campo dei sistemi autonomi e semiautonomi in ambienti dove sono dislocati diversi sensori.

- *Categoria robot di media taglia.* Sono in gioco quattro robot per squadra ciascuno dei quali ha un diametro inferiore ai 50 centimetri. Il campo è largo circa 5 metri e lungo 9 e il pallone ha dimensioni normali. Tutti i sensori devono trovarsi a bordo del robot. Questa regola è stata posta per incentivare la ricerca su sistemi autonomi, in grado di riconoscere gli oggetti e agire in modo indipendente. Ciò potrebbe contribuire alla ricerca sui robot dislocati in luoghi che gli operatori non possono controllare a distanza.
- *Categoria robot con gambe.* A questa categoria appartengono robot a quattro zampe somiglianti a cani. I robot hanno 16 gradi di libertà e tutti i sistemi di rilevamento e di calcolo sono a bordo. Ciascuna squadra è composta da tre robot.
- *Categoria robot umanoidi.* I programmi prevedono l'introduzione di un torneo per robot umanoidi a partire dal 2002. Questi robot dovranno muoversi su due gambe. È la sfida più importante della RoboCup, quella destinata a condurre all'obiettivo finale dell'iniziativa.

2. Squadra ART



ART-Azzurra Robot Team è la nazionale italiana di robot calciatori che partecipa alle competizioni a carattere scientifico organizzate dalla RoboCup. ART è stata realizzata nell'ambito del progetto RoboCup Italia da 6 gruppi di ricerca, impegnati su diversi aspetti della realizzazione di robot mobili autonomi, e dal Consorzio Padova Ricerche che ha messo a disposizione risorse e realizzato il campo di gioco presso la propria sede di Padova.

La prima partecipazione alla manifestazione RoboCup è del Luglio 1998 a Parigi che è stata la seconda edizione della manifestazione, la prima con una grande partecipazione da parte di università di tutto il mondo. In quest'occasione la squadra ART si è qualificata per i quarti di finale, su 16 squadre iniziali.

Nell'edizione 1999, tenutasi a Stoccolma, tra 21 squadre partecipanti, il team ART si è classificato al secondo posto, perdendo la gara finale contro la squadra iraniana che era dotata di un'ottima base meccanica.

ART Azzurra Robot Team e' formata da robot calciatori con diverse caratteristiche hardware e software, ma completamente autonomi, cioè con tutti i sensori e le capacità di elaborazione a bordo. Più precisamente vengono utilizzate due piattaforme hardware:

- **BaseART**, sulla quale sono stati sviluppati
 - i giocatori **RonalTino** e **TotTino**, dall'Università di Roma,
 - i giocatori Bart e Homer, dall'Università di Padova,
 - il giocatore Relè, dall'Università di Genova,
 - il portiere TinoZoff, dall'Università di Parma;
- **Mo²Ro**, una base per robot mobili per scopi generici, sviluppata dal Politecnico di Milano

3 Architettura di base ART

L'architettura hardware del robot calciatore e' formata dai seguenti componenti:

1. *piattaforma mobile*: il robot si basa sulla piattaforma **PIONEER** nella sua versione con motori a 12V predisposta per la RoboCup, dotata di sonar;
2. *computer di bordo*: e' basato su una architettura convenzionale, con componenti di normale reperibilità: mother board Asustek P5A, processore AMD K6 2 3Dnow a 450MHz, 64 Mbyte di RAM, HD 2GB;
3. *alimentazione*: alimentatore switching e una batteria a 12V;
4. *visione*: scheda di acquisizione immagini basata sul chip BT848 e telecamera a colori CCD 1/4" 380 linee, standard PAL, sens. 2 lux, F1.2;
5. *collegamento in rete*: collegamento wireless a 2Mbit (Lucent Wavelan).

L'architettura software e' formata dai seguenti componenti:

1. *sistema operativo* Linux;
2. *drivers*: driver per la scheda wavelan e per il frame grabber BT848;
3. *librerie di sistema*: per il supporto dell'interfaccia grafica e per l'acquisizione delle immagini;
4. *programmi di utilità e di debug*.

L'architettura di base del giocatore ART nasce con l'obiettivo di realizzare un robot specializzato per competere in manifestazioni calcistiche, secondo le specifiche regolamentari di *RoboCup*, totalmente autonomo ed in grado di coordinare il proprio comportamento con i compagni di squadra.

Pertanto sono stati aggiunti alla baseART alcuni dispositivi che gli consentono di calciare la palla e di acquisire informazioni sull'ambiente di gioco e sono stati progettati, implementati e sperimentati alcuni componenti software che ne hanno permesso il funzionamento in squadra.

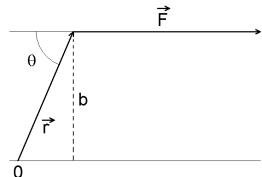


RonalTino

Appendice A7

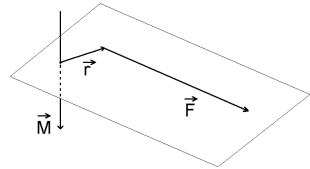
RICHIAMI DI FISICA

Momento di una forza (rispetto al punto 0)



$$\vec{M} = \vec{r} \times \vec{F}$$

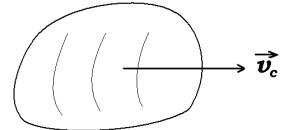
$$|\vec{M}| = r F \sin\theta = bF = C$$



Dinamica dei corpi rigidi: moto traslatorio

Sia $\vec{F}_i^{(e)}$ l'*i-esima* forza esterna applicata al corpo rigido ed m la massa del corpo. Si ha:

$$\sum_i \vec{F}_i^{(e)} = m \frac{d\vec{v}_c}{dt} \quad \sum_i \vec{F}_i^{(e)} = m \vec{a}_c$$

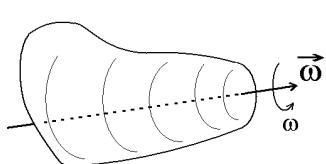


con \vec{v}_c velocità del *centro di massa* e \vec{a}_c accelerazione del *centro di massa*.

Un attrito viscoso è caratterizzato dall'espressione $\vec{F}_a = -k \vec{v}_c$.

Dinamica dei corpi rigidi: moto rotatorio

Sia $\vec{M}_i^{(e)}$ l'*i-esimo* momento esterno applicato al corpo rigido ed I il momento d'inerzia del corpo. Si ha:



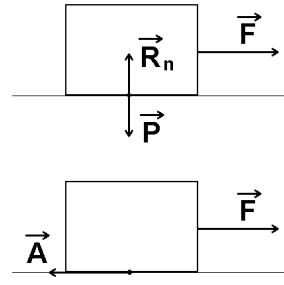
$$\sum_i \vec{M}_i^{(e)} = I \frac{d\vec{\omega}}{dt} \quad \text{con } \vec{\omega} \text{ velocità angolare}$$

$$I = \frac{1}{2} m R^2 \quad \text{per un disco di raggio } R \text{ e massa } m$$

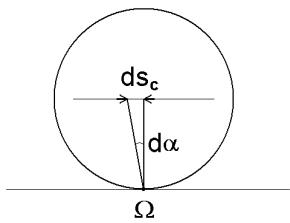
Un attrito viscoso è caratterizzato dall'espressione $\vec{M}_a = -k \vec{\omega}$

Attrito secco

Si consideri un corpo in quiete soggetto ad una forza F (sia P la forza peso e R_n la reazione normale al vincolo). In presenza di attrito, il corpo resta fermo finché: $F \leq \mu_s R_n$ ove μ_s è il coefficiente di *attrito statico* (o di *primo distacco*). Nel vincolo nasce quindi una forza di attrito A , in grado di equilibrare la F finché questa non supera il valore $\mu_s R_n$. Una volta che il corpo si è mosso, per mantenerlo in moto uniforme occorre fornire una forza più piccola della precedente, pari ad $F = \mu_d R_n$ ove μ_d è il coefficiente di *attrito dinamico* (o di *attrito radente*). Risulta sempre $\mu_d \leq \mu_s$



Moto di rotolamento



Si consideri un cilindro omogeneo di raggio r e massa m che avanza rotolando su di un piano orizzontale. La generatrice di contatto fra il cilindro e il piano di appoggio ha velocità nulla: si ha quindi una rotazione istantanea attorno al punto Ω (asse istantaneo di rotazione). Una rotazione infinitesima, causa una traslazione lineare dell'asse del cilindro pari a:

$$ds_c = r \cdot d\alpha$$

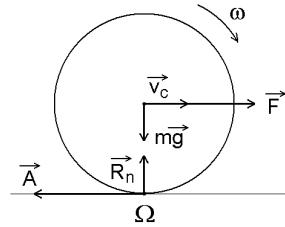
(α piccolo: $\sin(\alpha) \approx \alpha$ e $\cos(\alpha) \approx 1$). La velocità del centro di massa è dunque:

$$v_c = \frac{ds_c}{dt} = r \frac{d\alpha}{dt} = \omega \cdot r$$

ove ω , velocità angolare istantanea attorno alla generatrice di contatto, coincide con la velocità angolare di rotazione del cilindro. Infine, essendo la generatrice in quiete, su di essa agisce l'attrito statico.

Rotolamento causato da forza applicata

Si ha moto di rotolamento grazie alla forza di attrito, diretta sempre in verso contrario al moto. Il sistema è descritto da tre equazioni cardinali (assi x, y per le forze, asse ω per i momenti). L'unico momento agente è quello determinato dalla forza di attrito, di intensità pari a $r \cdot A$. Si ricava per l'accelerazione del centro massa:



$$a_c = \frac{2}{3} \frac{F}{m}$$

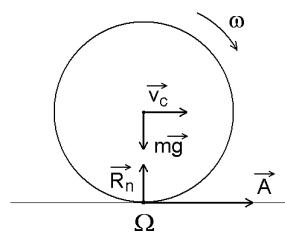
Affinché il moto sia di puro rotolamento (cioè senza strisciamento) deve essere:

$$A = F - m a_c \leq \mu_s mg \quad \Rightarrow \quad F \leq 3\mu_s mg$$

Rotolamento causato da coppia motrice

Si consideri una situazione di quiete, con il cilindro fermo. Applicando un momento M all'asse di rotazione (momento non nullo ed entrante nel foglio) si provoca una variazione della velocità angolare ω .

Affinché ciò sia possibile deve nascere una forza esterna (nell'ipotesi di rotolamento puro), poiché se così non fosse si avrebbe $v_c = \text{cost}$ ma ciò sarebbe in contrasto con il fatto che ω è variabile. Tale forza esterna è necessariamente l'attrito (essendo l'unica forza esterna presente), che diventa la forza motrice ed è sempre diretta nel verso del moto. Anche qui il sistema è descritto da tre equazioni cardinali (assi x, y per le forze, asse ω per i



momenti). I momenti agenti sono la coppia motrice M e quella resistente determinata dalla forza di attrito e di intensità pari a $-r \cdot A$. Si ricava per l'accelerazione del centro massa:

$$a_c = \frac{2}{3} \frac{M}{m \cdot r}$$

Affinché il moto sia di puro rotolamento deve essere:

$$A = ma_c \leq \mu_s mg \quad \Rightarrow \quad M \leq \frac{3}{2} \mu_s mgr$$

Bibliografia

Testi

[Borenstein *et al.*, 1996] J. Borenstein, H. R. Everett e L. Feng, “*Navigating mobile robots*”. A K Peters Wellesley, Massachusetts, 1996

[Grasso *et al.*, 1982/a] N. Grasso, A. Liberatore, G. Ricci, “*Radioelettronica, Volume 1 – Bassa Frequenza*”, Edizioni Cremonese, 1982

[Grasso *et al.*, 1982/b] N. Grasso, A. Liberatore, G. Ricci, “*Radioelettronica, Volume 2 – Alta Frequenza*”, Edizioni Cremonese, 1982

[Isidori, 1992] A. Isidori, “*Sistemi di Controllo, Volume Primo – Seconda Edizione*”, Edizioni Siderea, 1992

[Millman e Halkias, 1978] J. Millman e C.C. Halkias, “*Microelettronica*”, Boringhieri Editore, Torino, 1978

[Oppenheim *et al.*, 1998] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, “*Discrete-Time Signal Processing, 2nd Edition*”, Prentice Hall, 1998

[Sciavicco e Siciliano, 1995] L. Sciavicco e B. Siciliano, “*Robotica Industriale*”. McGraw-Hill, Milano, 1995

[Sette, 1991] D. Sette, “*Lezioni di fisica*”. Editoriale Veschi, Milano, 1991

[Wall *et al.*, 2000] Kurt Wall, Mark Watson e Mark Whitis, “*Programmare in Linux Tutto&Oltre*”, Apogeo srl, Milano, 2000

Pubblicazioni e siti Web

[3W, 1] “*efg's Color Reference Library -- Color Science / Color Theory*”

<http://www.efg2.com/Lab/Library/Color/Science.htm>

[3W, 2] “*Pittman Servo Motor Application Notes*”

<http://www.pittmannet.com>

[3W, 3] “*Filterless Class D simplifies audio amplifier design*”

<http://www.edtn.com/story/tech/OEG20001016S0061-T.html>

[3W, 4] “*H-Bridge Circuit*”

<http://www.ece.cmu.edu/~ece778/lecture-notes/Motor-drive-lecture/sld005.htm>

[3W, 5] Pittman, “*Series GM9000 DC GearMotors*”

<http://www.pittmannet.com>

[3W, 6] “*The RTLinux Manifesto*”

<http://www.rtlinux.org/documents/papers/rtmanifesto/rtlmanifesto.html>

[3W, 7] “*Testing RT-Linux*”

<http://www.fisica.unlp.edu.ar/rt/>

[3W, 8] “*The CUPL Environment*”

<http://www.ee.washington.edu/class/371/doc/cupl.html>

[Agent, 1991] Agent, A., 1991, “*The Advantages of Absolute Encoders for Motion Control*”. Sensors, April, pp. 19-24

[Asada e Kitano, 1999] M. Asada e H. Kitano, “*RoboCup: e ora in campo gli automi*”, Le Scienze, agosto 1999

[Avolio, 1993] Avolio, G., 1993, “*Principles of Rotary Optical Encoders*” Sensors, April, pp. 10-18

[Barabanov e Yodaiken, 1997] Michael Barabanov e Victor Yodaiken, “*Realtime Linux*”, Linux journal, Febbraio 1997

[Bollella, 1997] Greg Bollella, “*Slotted Priorities: Supporting Real-Time Computing Within General-Purpose Operating Systems*”, PhD thesis, University of North Carolina, 1997

[Borenstein e Feng, 1995] Borenstein, J. and Feng, L., 1995, “*Measurement and Correction of Systematic Odometry Errors in Mobile Robots*”. Accepted for publication as a regular paper in the IEEE, Transactions on Robotics and Automation, Apr

[Cox, 1991] Cox, I.J., 1991, “*Blanche - An Experiment in Guidance and Navigation of an Autonomous Mobile Robot*”. IEEE Transactions Robotics and Automation, 7(3), pp. 193-204

[De Carli, 1996] A. De Carli, “*Controllo in presenza di non-linearietà*”. Facoltà di Ingegneria, Università di Roma “La Sapienza”, a.a. 1996/97

[De Carli, 1998/a] A. De Carli, “*Scelta e progettazione di sistemi avanzati di movimentazione*”. Facoltà di Ingegneria, Università di Roma “La Sapienza”, a.a. 1998/99

[De Carli, 1998/b] A. De Carli, “*Controllo Intelligente*”. Facoltà di Ingegneria, Università di Roma “La Sapienza”, a.a. 1998/99

[De Carli, 1998/c] A. De Carli, “*Regolatori PID*”. Facoltà di Ingegneria, Università di Roma “La Sapienza”, a.a. 1998/99

[De Luca, 1995] A. De Luca, “*Controllori Convenzionali per un Singolo Asse di Robot*”. Facoltà di Ingegneria, Università di Roma “La Sapienza”, a.a. 1995/96

[De Luca, 2000] A. De Luca, “*Kinematics and Motion Generation for Wheeled Mobile Robots*”. International School RoboCup 2000 Camp, Padova, February 28 – March 4, 2000

[Dunlap e Shufeldt, 1972] Dunlap, G.D. and Shufeldt, H.H., 1972, “*Dutton’s Navigation and Piloting*”, Naval Institute Press, pp. 557-579

[ELCIS, 1991] ELCIS, “*Encoder rotativi incrementali ed assoluti*”. Collegno (TO), 1991 - <http://www.elcis.com/index.html>

[INTEL, 1994] INTEL, “*82C54 Chmos Programmable Interval Timer*”, DATASHEET October 1994.

[INTEL 1995] INTEL, “*82C55A Chmos Programmable Peripheral Interface*”, DATASHEET October 1995.

[INTERSIL, 1996] Intersil, “*HIP4081A*”, DATASHEET November 1996

[INTERSIL, 1997] Intersil, “*HIP4081A - AN9405.3*”, APPLICATION NOTE September 1997

[Jones, 1995] Douglas W. Jones “*Control of Stepping Motors, a tutorial*”, University of Iowa, Department of Computer Science.
<http://www.cs.uiowa.edu/~jones/step/index.html>

[Klarer, 1988] Klarer, P.R., 1988, “*Simple 2-D Navigation for Wheeled Vehicles*” Sandia Report SAND88 - 0540, Sandia National Laboratories, Albuquerque, NM, April

[LATTICE, 1998] Lattice Semiconductor Corporation, “*GAL 16V8*”, DATASHEET October 1998

[Leonard e Durrant-Whyte, 1991] Leonard, J. and Durrant-Whyte, H. F., 1991, “*Mobile Robot Localization by Tracking Geometric Beacons.*” IEEE Transactions on Robotics and Automation, Vol. 7, No. 3, pp. 376-382

[Marcellini, 1985] L. Marcellini, “*Mosfet di potenza*”, Selezione di Elettronica e Microcomputer, Aprile 1985, Edizioni JCE

[MICROCHIP, 1993] Microchip, “*Servo Control of a DC-Brush Motor*”, APPLICATION NOTE 1993 - <http://www.microchip.com>

[MICROCHIP, 1999] Microchip, “*Brush-DC Servomotor Implementation using PIC17C756A*”, APPLICATION NOTE 1999

<http://www.microchip.com>

[MOTOROLA, 1996] Motorola Semiconductor, “*16-Bit DSP Servo Control With the MC68HC16ZI*”, APPLICATION NOTE 1996

[NATIONAL, 1997] National Semiconductor “*Technical literature database*”, CDROM 1997

[Nickson, 1985] Nickson, P., 1985, “*Solid-State Tachometry*”. Sensors, April, pp. 23-26

[Pisano, 2001] Alessandro Pisano “*I Servomotori Elettrici in Corrente Continua: Modello Matematico e Tecniche di Controllo*” – Facoltà di Ingegneria, Università di Cagliari, a.a. 2000/01
<http://www.diee.unica.it/~pisano/MotoreDC.PDF>

[Pomerantz, 1999] “*Linux Kernel Module Programming Guide*”
<http://www.linux.org/docs/ldp/projectguide.html>

[SGS, 1995/a] SGS-Thomson, “*How to drive DC Motors with Smart Power ICs*”, APPLICATION NOTE 1995

[SGS, 1995/b] SGS-Thomson, “*Driving DC Motors*”, APPLICATION NOTE 1995

[SGS, 1995/c] SGS-Thomson, “*Load Current Sensing in Switchmode Bridge Motor Driving Circuits*”, APPLICATION NOTE 1995

[SGS, 1997] SGS-Thomson, “*Data on disc*”, CDROM 1997

[Shanley e Anderson, 1995] Tom Shanley and Don Anderson, "ISA System Architecture, 3rd Edition", Mindshare Inc 1995

[Stankovic e Ramamritham, 1988] John A. Stankovic e Krithi Ramamritham, “*Hard Real-Time Systems*”, volume 819 of IEEE Tutorials, IEEE 1988

[Tonielli, 2000] A. Tonielli, “*Compatibilità Elettromagnetica*”. Facoltà di Ingegneria, Università di Bologna, a.a. 1999/2000

[Venturini, 1985] M. Venturini, “*L’azionamento Brushless*”, Selezione di Elettronica e Microcomputer, Dicembre 1985, Edizioni JCE