۱

فصل اول

درگاه موازي كامپيوتر

۱-۱) تاریخچه درگاه موازی

وقتی IBM در سال ۱۹۸۱ ، PC را معرفی کرد، پورت پارالل بعنوان جایگزینی برای پورت سریال، به جهت سرویس دهی و راه اندازی پرینترهای Dot Matrix با بازده بالا در نظر گرفته شد.

پورت موازی این خاصیت را داشت که در هر لحظه هشت بیت داده را منتقل کند. این درحالیست که پورت سریال فقط می توانست یک بیت داده را در هر لحظه منتقل کند. همراه با رشد تکنولوژی، نیاز به اتصالات خروجی قویتر و بزرگتر افزایش یافت، لذا پورت موازی با این هدف که شما می توانستید وسایل جنبی با بازده بالاتری را به آن متصل کنید، بوجود آمد. این وسایل جنبی هم اکنون شامل محدودهٔ وسیعی از پرینترهای اشتراکی، دیسک درایوهای پرتابل و Tape Backup گرفته تا آداپتورهای شبکه های محلی (LAN) و CD-ROM Player ها می شود .

مشکلاتی که توسعه دهندگان و خریداران این وسایل جنبی با آن روبرو بوده اند، به سه دسته تقسیم می شد. اول اینکه بازده PC بصورت هیجان آوری زیاد شده بود، در حالیکه تغییری در ساختمان پورت پارالل احساس نمی شد، چون حداکثر قدرت انتقال توسط این ساختمان حدود ۱۵۰ کیلو بایت بر ثانیه بود، که این واقعا نیاز به یک نرم افزار قوی و قدرتمند داشت. دوم، آنکه هیچ استانداردی برای واسط های الکتریکی وجود نداشت، که این موجب مشکلات فراوانی در ضمانت عملکرد سیستم در محدوده های مختلف می شد. و سرانجام اینکه نقص استانداردهای طراحی، استفاده از کابلهایی با طول بیش از شش پا را اجازه نمیداد.

در سال ۱۹۹۱ دیداری توسط سازندگان پرینتر برای شروع بحث و مناظره روی گسترش یک استاندارد جدید، برای کنترل هوشمند پرینترها از طریق شبکه برگزار شد. این سازندگان که شامل Lexmark, IBM, Texas instuments و بقیه می شد، پیمان بین المللی پرینت شبکه ای (Network Printing Alliance) را بوجود آوردند.

NPA مجموعه ای از پارامترهایی را توصیف می کند که وقتی بر روی پرینتر و میزبان پیاده سازی شود، کنترل کامل کاربردها (Applications) وکارها (Jobs) را ممکن میسازد.

وقتی که این کار در حال پیشروی و رشد بود، معلوم شد که پیاده سازی کامل این استاندارد، به یک ارتباط دو طرفه در PC نیاز خواهد داشت. و به نظر می رسید که پورت پارالل معمولی PC قادر نبود تا نیازهای مورد نظر این استاندارد را پشتیبانی کند.

NPA یک پیشنهاد به IEEE ارائه کرد که برای رسیدن به یک ارتباط دو طرفه سریع بر روی PC کمیته ای تشکیل دهد. لازم بود که این کمیته در نظر داشته باشد که استاندارد جدید باید کاملا سازگار با پورت پارالل اصلی (Original) و وسایل جنبی آن باشد. و در عین حال نرخ داده را تا بیشتر از یک مگابایت در ثانیه افزایش دهد. این کمیته استاندارد IEEE 1284 را بوجود آورد.

استاندارد IEEE 1284 یا Standard Signaling Method for Bi-directional” “ Parallel Pripheral Interface for Personal Computers برای آخرین ویرایش در مارس ۱۹۹٤ تصویب شد.

۱-۲) آشنایی با درگاه موازی

درگاه موازی یا همان Parallel Port یکی از پورتهای کامپیوترهاست که اطلاعات از طریق آن خوانده و به کامپیوتر منتقل می شود، و یا بر روی آن نوشته می شود.

درکل IBM سه نوع آداپتور که شامل پورت موازی پرینتر هستند، برای میکرو کامپیوترهای PC/ XT/ AT تدارک دیده است. بسته به آنکه کدامیک نصب شده باشند، هر پورت قابل دستیابی دارای یکی از سه آدرس 3BC, 378, 278 (همگی بصورت HEX) خواهد بود. اکثر PC ها با یک پورت موازی و آنهم با آدرس 378 HEX تولید شده اند.

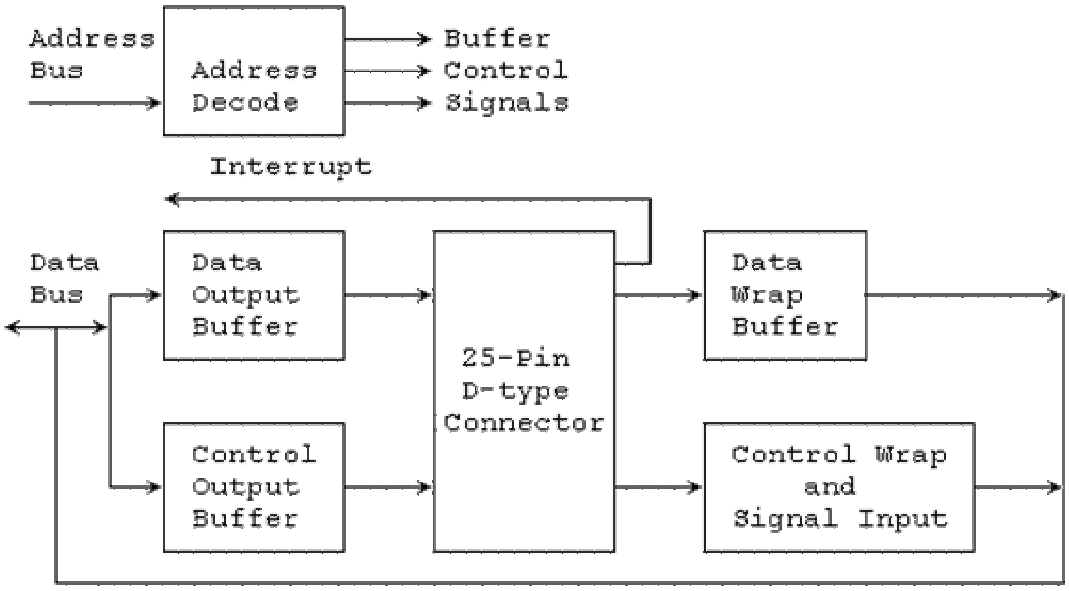
پورت موازی PC بطور اخص برای اتصال پرینترها بوسیلهٔ یک واسط (Interface) طراحی شده اند. اما می توان از آن بعنوان یک پورت ورودی/ خروجی عمومی برای هر وسیله یا هر کاربرد دیگری که با قابلیتهای ورودی و خروجی آن سازگار باشد، استفاده کرد.این پورت دارای ۱۲ بافر TTL است که قابل نوشتن و خواندن تحت برنامهٔ کنترلی و با استفاده از دستورالعملهای ورود و خروج هستند.

آداپتور کامپیوتر همچنین دارای پنج ورودی مجزا است که ممکن است توسط دستورالعملهای ورودی پروسسور خوانده شوند. در مجموع یکی از ورودی ها می تواند برای تولید وقفهٔ پروسسور استفاده شود. این وقفه میتواند، تحت برنامهٔ کنترل فعال یا غیر فعال شود. همچنین توسط یک خروجی می توان وسیلهٔ متصل شده به پورت را همزمان با وقفهٔ روشن شدن (Reset from the Power-on Circuite) راه اندازی کرد.

سیگنالهای خروجی توسط یک متصل کنندهٔ ۲۵ پین از نوع D ، (D-Type) که در پشت آداپتور قرار دارد در دسترس هستند. وقتی که این پورت برای استفاده از پرینتر در نظر گرفته می شود، اطلاعات و دستورات بصورت هشت بیتی منتقل می شوند، و پایهٔ STROBE نیز فعال است. در این حالت ممکن است برنامه پینهای ورودی را جهت اطلاع از وضعیت پرینتر بخواند، و سپس کاراکتر بعدی را بفرستد که این عمل با استفاده از خط "Not Busy" صورت می گیرد.

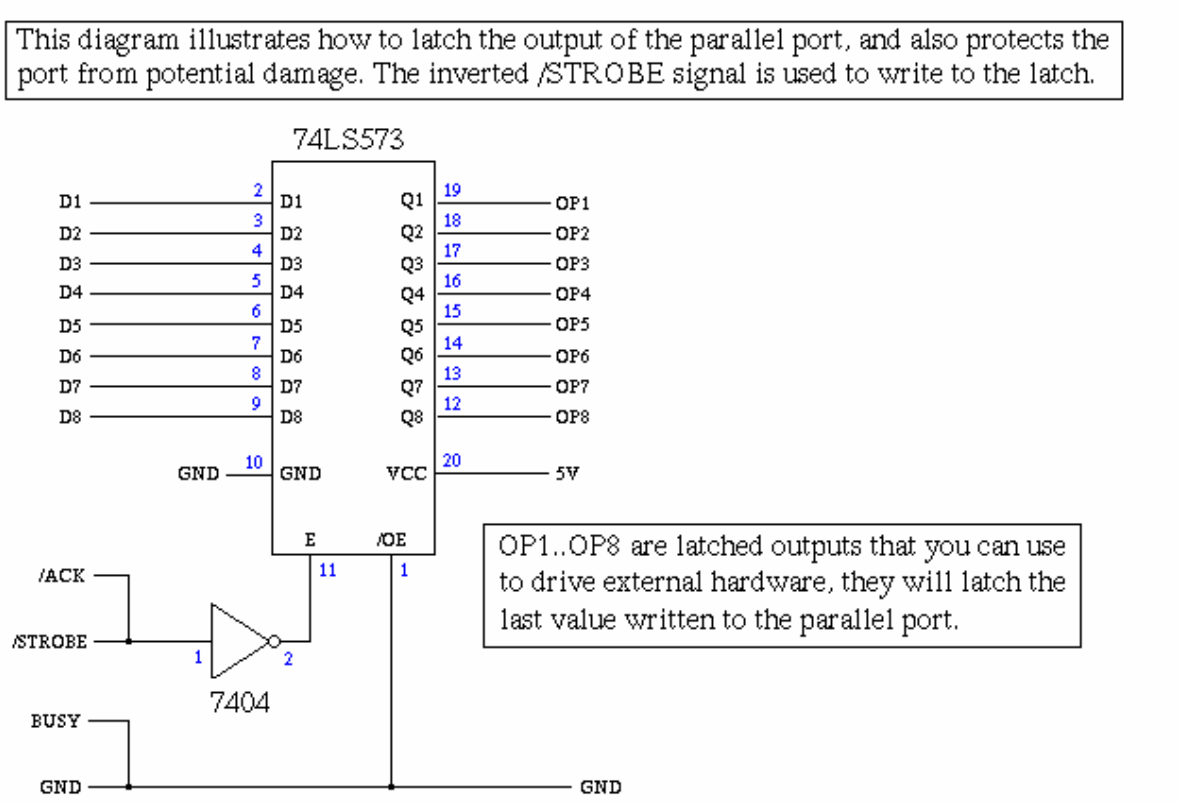
همچنین ممکن است اطلاعات بر روی مدار واسط نوشته ویا از روی آن خوانده شود.این کار اجازه می دهد که وسیلهٔ متصل شده و پورت موازی از هم ایزوله یا مجزا گردند، و آسیبی به آنها وارد نشود.

درشکل زیر(۱-۱) بلاک دیاگرام آداپتور موازی پرینتر (IBM PC/AT) را مشاهده میکنید.

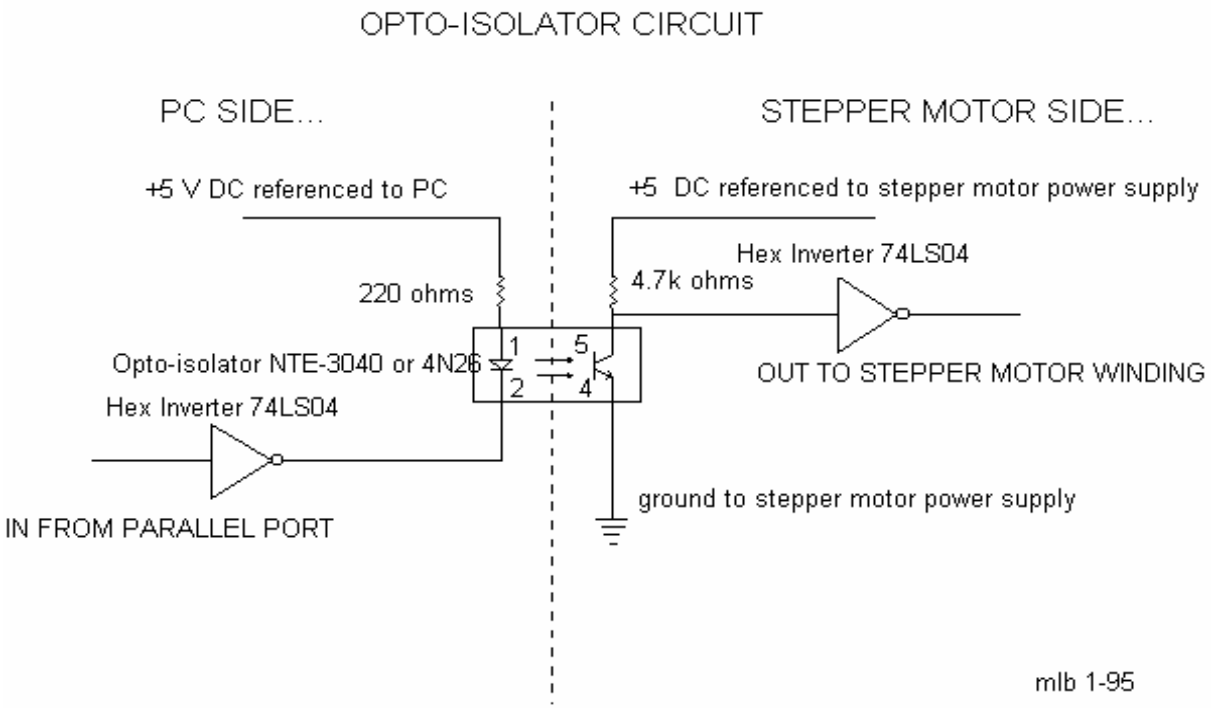


شکل ۱-۱Parallel Printer Adapter (IBM PC/AT) Block Diagram

برای ایزولاسیون پورت و حفاظت از آن راه های مختلفی وجود دارد، که در شکل های بعدی نمونه هایی از آنها را ملاحظه خواهید کرد. در شکل ۱-۲ صفحهٔ بعد مدار ایزوله کنندهٔ پورت موازی را که بوسیله IC 74LS573 صورت گرفته می بینید.



همچنین در شکل ۱-۳ زیر مدار ایزوله کنندهٔ نوری را مشاهده می کنید.



همانطور که گفته شد، پارالل پورت مهمترین پورت مورد استفاده برای پروژه های دارای مدار واسط است. این پورت امکان استفاده از ۹ بیت ورودی یا ۱۲ بیت خروجی را در هر زمان مهیا می سازد. بنا بر این در کوچکتر کردن مدارهای خارجی در بسیاری از پروژه ها به ما کمک میکند. این پورت از ٤ خط کنترل، ٥ خط وضعیت و ٨ خط داده تشکیل شده است.

پورتهای پارالل جدید تحت استاندارد IEEE 1284 ، ویرایش نخست سال ۱۹۹٤ هستند. این استاندارد ٥ مود عملیاتی را که به شرح زیر هستند، تعریف می کند:

1. Compatibility Mode.
2. Nibble Mode. (Protocol not Described in this Document)
3. Byte Mode. (Protocol not Described in this Document)
4. EPP Mode (Enhanced Parallel Port).
5. ECP Mode (Extended Capabilities Mode).

هدف این بود که، درایورها و وسایل جانبی، طوری طراحی شوند که با همدیگر سازگار باشند و همچنین با پورت پارالل استاندارد (SPP) نیز سازگاری داشته باشند. مود های Compatibility,Nibble و Byte فقط از سخت افزارهای استاندارد موجود بر پورت پارالل استاندارد استفاده می کنند، در حالیکه مودهای EPP و ECP نیاز به سخت افزار مضاعفی که بتواند با سرعت بیشتری عمل کند نیازمندند، درحالیکه همچنان با پورت پارالل استاندارد نیز سازگاری دارند. Compatibility Mode یا "Centronics Mode" فقط می تواند داده را بصورت یکطرفه با سرعتی معادل 50 Kbyte/s (حداکثر 150 kb/s) منتقل کند.

در هر حال برای دریافت داده شما مجبورید از یکی از مود های "نیبل" یا "بایت" استفاده کنید. Nibble Mode می تواند یک نیبل (٤ بیت) را از وسیلهٔ جانبی به کامپیوتر منتقل کند. Byte Mode از خصوصیت دوطرفه (که روی بعضی از کارتها وجود دارد) برای ورود یک بایت (٨ بیت) استفاده می کند.

پورتهای پارالل توسعه یافته و بهبود یافته (Enhanced and Extended)، سخت افزار مضاعفی را برای ایجاد مدیریت hand shaking استفاده می کنند. برای فرستادن یک

بایت به یک پرینتر یا هر وسیلهٔ دیگر، با استفاده از Compatibility Mode نرم افزار باید مراحل زیر را انجام دهد.

۱- نوشتن یک بایت بر روی پورت داده.

۲- بررسی اینکه آیا پرینتر در وضعیت مشغول است یا نه. اگر پرینتر مشغول باشد هیچ داده ای را نمی پذیرد و داده نوشته شده از دست می رود.

۳- قرار دادن پایه Strobe (پین ۱) در حالت Low . این به پرینتر می فهماند که دادهٔ معتبر روی خطوط داده (پینهای ۲ تا ۹) قرار دارد.

٤- برگرداندن Strobe به حالت high پس از حدود ۵ میکروثانیه از زمانیکه Strobe به حالت Low رفته بود. (برگشت به قدم سوم).

این عملیات سرعت پورت را محدود می کند. پورتهای ECP & EPP تقریبا به همین شکل عمل می کنند، در حالیکه از hand shaking استفاده می کنند، که این موجب افزایش سرعت می شود. این پورتها می توانند حدود ۱ تا ۲ مگابایت بر ثانیه منتقل کنند. همچنین پورت ECP توانایی استفاده از کانالهای DMA (دسترسی مستقیم به حافظه) را دارد، لذا داده می تواند بصورت گردشی شیفت پیدا کند در حالیکه از دستورالعمل I/O استفاده نمی شود.

۱-۳) پینها وثبات های پورت پارالل

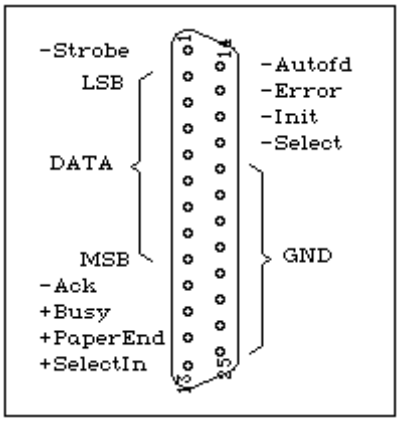
پورت پارالل دارای ۲۵ پایه می باشد که بترتیب شماره در دیاگرام زیر آمده اند.

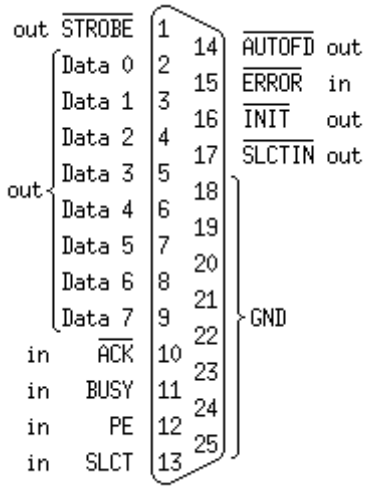
Παραλλελ Πορτ Πινουτ Διαγραμ

Pin	Signal
1	-Strobe
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	-Acknowledge
11	Busy
12	Paper Empty
13	+Select
14	-Auto Feed
15	-Error
16	-Init
17	-Slctin
18	Ground
19	Ground
20	Ground
21	Ground
22	Ground
23	Ground
24	Ground
25	Ground

در شکل ۱-٤ صفحة بعد نمايي از کانکتور ۲۵ پين مخصوص پورت پارالل را مشاهده ميکنيد

که اسامی پین ها را در محل واقعی آنها بیان می کند.





شکل ۱-٤ - کانکتور ماده(Female) ۲۵ پایه پورت پارالل

در جداول و نماهاي زير، وضعيت ثباتها و اتصالات پينهاي پورت موازي رامشاهده مي کنيد.

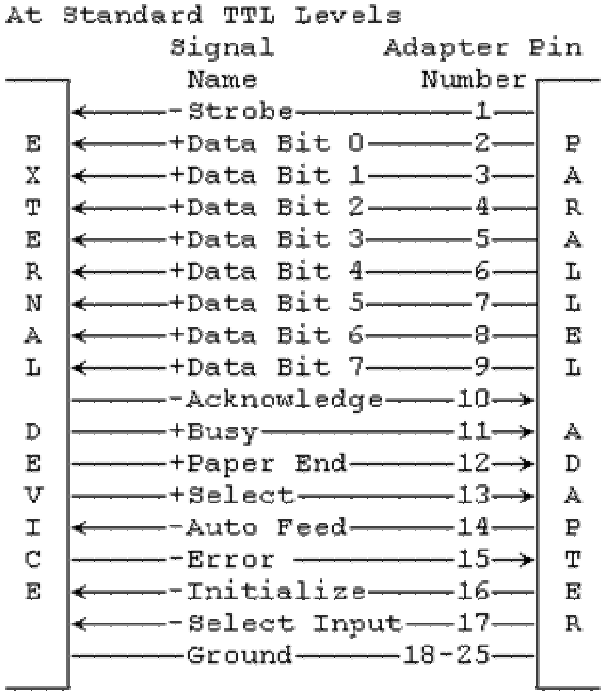
Registers & Pinouts IBM-PC Parallel Printer Port

Registers (- unavailable)

	7	6	5	4	3	2	1	0	I/O Port
DATA	x	x	x	x	x	x	x	x	Base = 278/378/3BC Hex
STATUS	$\overline{x}$	x	x	x	x	-	-	-	Base+1
CONTROL	-	-	-	-	$\overline{x}$	x	$\overline{x}$	$\overline{x}$	Base+2

Note: S7, C0, C1 & C3 are inverted

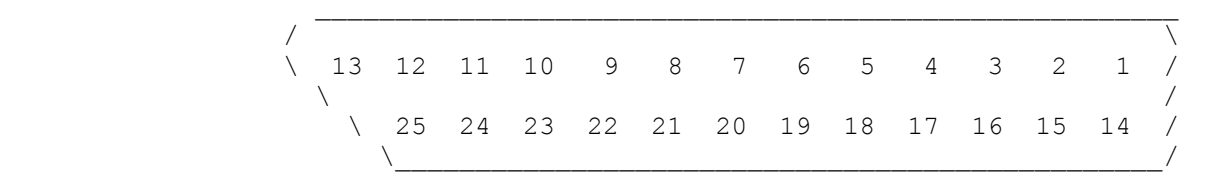
i.e. Parallel Port pin 11 High will set S7 = 0
C0 = 1 will cause Parallel Port pin 1 to go Low, etc



Signal Name	Register Bit	DB-25 Pin	I/O Direction
-Strobe	¬C0	1	Output
+Data Bit 0	D0	2	Output
+Data Bit 1	D1	3	Output
+Data Bit 2	D2	4	Output
+Data Bit 3	D3	5	Output
+Data Bit 4	D4	6	Output
+Data Bit 5	D5	7	Output
+Data Bit 6	D6	8	Output
+Data Bit 7	D7	9	Output
-Acknowledge	S6	10	Input
+Busy	¬S7	11	Input
+Paper End	S5	12	Input
+Select In	S4	13	Input
-Auto Feed	¬C1	14	Output
-Error	S3	15	Input
-Initialize	C2	16	Output
-Select	¬C3	17	Output
Ground	-	18-25	-

(Note again that the S7, C0, C1 & C3 signals are inverted)

IBM-PC Parallel Printer Port Female DB-25 Socket external Pin layout



So it's also the Pin layout on the solder side of the Male DB-25 Cable Connector that plugs into it

در جدول بعد پینهای خروجی (Pinouts) کانکتورهایD-Type 25 pin و Centronics 36 pinرا مشاهده می کنید. همانطور که می دانید، کانکتور D-Type

25 pin متداولترین کانکتور پورت پارالل کامپیوتر است، در حالیکه کانکتور Centronics معمولا روی پرینترها دیده می شود. استاندارد IEEE 1284 در مجموع ۳ نوع مختلف از کانکتور را برای پورت پارالل بوجود آورد. اولین آن 1284 Type A است، که همان D-Type 25 pin می باشد، که در پشت کامپیوترها دیده می شود. دومین نوع آن 1284 Type B یا 36 pin Centronics Connector می باشد، که در اغلب پرینترها موجود است. نوع سوم آن IEEE 1284 Type C است که یک 36 Conductor Connector شبیه به Centronics است، در حالیکه کمی کوچکتر می باشد. این کانکتور دارای ویژگیهای نصب آسان، لچ(Latch) بهتر و خصوصیات الکتریکی مناسبتر می باشد. همچنین دارای ۲ پین اضافی برای سیگنالهایی است که می توانند اتصال وسیلهٔ جانبی را، چک کنند. 1284 Type C برای طراحی های جدید در نظر گرفته شده است، لذا می توان انتظار داشت، که در آینده کانکتورهای جدیدتر دیگری هم عرضه شوند.

به جدول صفحهٔ بعد که خصوصیات سخت افزاری پورت و کانکتورها را مورد بررسی قرار داده است، توجه کنید. در این جدول از حرف "n" در جلوی نام بعضی از سیگنالها استفاده شده، که برای توجه دادن به اینکه آن سیگنال Active Low می باشد، بکار رفته است. بطور مثال nError ، اگر در پرینتر خطایی رخ دهد، این خط Low خواهد شد. این خط بصورت نرمال high می باشد، که نمایانگر عملکرد صحیح پرینتر است. " hardware inverted" نیز به این معنی است که سیگنال توسط سخت افزار کارت پارالل معکوس شده است. یک نمونهٔ آن سیگنال Busy است. اگر 5v+ (منطق1) به این پین نسبت داده شود، و ثبات وضعیت خوانده شود، مقدار 0 در بیت ۷ از ثبات داده، مشاهده خواهد شد.

همانطور که گفته شد، خروجی پارالل پورت بصورت نرمال در سطح منطقی TTL است. غالب پورتهای پارالل می توانند حدود 12 mA بکشند یا بدهند(Sink and Source) . به هر حال بعضی از آنها در Data Sheet خود مقادیر زیر را هم دارند:

Sink 16 mA/Source 4 mA
Sink 12 mA/Source 20mA
Sink/Source 4 mA
Sink/Source 12mA

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out / Paper-End	In	Status	
13	13	Select	In	Status	
14	14	nAutø Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer / nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

Assignments of the D-Type 25 pin Parallel Port Connector.

جدول ۱-۱

۱-٤) شرح پینهای درگاه موازی

در این بخش به شرح وظیفهٔ تک تک پینهای پورت پارالل پرداخته شده است.

1.STROBE signal

سیگنال STROBE/ با پالس زیر (low pulse) نشانگر دادهٔ معتبر روی D1..D8 است، که شما می توانید از این سیگنال برای فرستادن داده به یک Latch یا Register و یا برای تولید یک وقفه، روی بعضی سخت افزارهای خارجی استفاده کنید.

ممکن است سیگنال STROBE/ خیلی صاف نباشد، لذا پالس خروجی یک سیگنال با طول کوتاه و یک سیگنال با طول بلند تا حدی متفاوت خواهد بود که شما می توانید برای تصحیح آن از یک اشمیت تریگر 74LS14 یا مشابه استفاده کنید، تا یک خروجی پالس مربعی خوب بدست آورید.

2-9. D1..D8

اینها پایه های خروجی داده هستند که از نوع TTL میباشند. در بعضی از ماشینها (بطور مثال Indy) این پایه ها دو طرفه هستند، وقتی که قرار است اطلاعات خوانده شود، این پایه ها امپدانس بالا (High Impedance) میشوند. (به قسمت PR/SC توجه کنید).

بعضی از ماشینها این پینها را با هم Latch (چفت و بست) می کنند و بعضی دیگر نه. مطمئن ترین روش اینکار این است که خودتان آنها را Latch کنید، اینکار موجب می شود که سخت افزار شما روی هر ماشینی کار کند. (برای اطلاعات بیشتر می توانید به برنامه Portdemo.c در بخش ضمیمه الف مراجعه کنید.)

/ACKNOWLEDGE

وقتی یک وسیله خارجی اطلاعاتی از منبع خود دریافت کند (مثلا بعد از پالس STROBE/) باید یک پالس ACKNOWLEDGE/ برای مشخص کردن اینکه انتقال داده با موفقیت انجام شده بفرستد. برای یک خروجی ساده شما می توانید پایه های STROBE/ و ACKNOWLEDGE/ را بهم ببندید (متصل کنید) که در

اینصورت خود واسط (Interface) به خودش ACKNOWLEDGE میدهد. این به این معناست که شما می توانید بر روی پورت، با سریعترین سرعت ممکن بنویسید.

BUSY

وسیلهٔ خروجی می تواند پایهٔ BUSY را بصورت سیگنال بالا (High Signal) برای اینکه مشخص کند در حالت اشغال است قرار دهد و بفهماند که قادر به ارتباط (Communicate) نیست. معمولا پرینترها از این پایه برای اعلام اینکه وظیفهٔ وقتگیری در حال اجرا است استفاده می کنند. (از جمله حرکت هد پرینتر). اگر نمی خواهید از این پایه استفاده کنید می توانید آنرا به یکی از پایه های زمین (Ground) متصل کنید.

EOP

پرینتر از این پین در حالت High برای اعلام تمام شدن کاغذ استفاده می کند. ولی شما می توانید از این ورودی برای اهداف دیگری استفاده کنید.

ON LINE

پرینتر از این پایه در حالت High برای اعلام اینکه روی خط (انتخاب شده Selected) است، استفاده می کند.

PR/SC

این سیگنال برای اعلام اینکه پورت قصد خواندن و یا نوشتن دارد استفاده می شود. این پین در حالت معمولی بصورت High (Printer Mode) می باشد. اما در زمانیکه از پورت بصورت دو طرفه (Bidirectional) استفاده می شود، در صورتیکه دستوری برای خواندن داده شود، این پین به حالت Low می رود (Scanner Mode).

FAULT

پرینتر از این پین در حالت Low برای اعلام اینکه خطایی اتفاق افتاده استفاده می کند. مثل EOP ،توجه داشته باشید که این پین مقدار عکس را بر می گرداند. یعنی اگر این پین در حالت High باشد، بیت PLPFAULT مربوطه مقدار صفر را بر میگرداند و بر عکس.

RESET

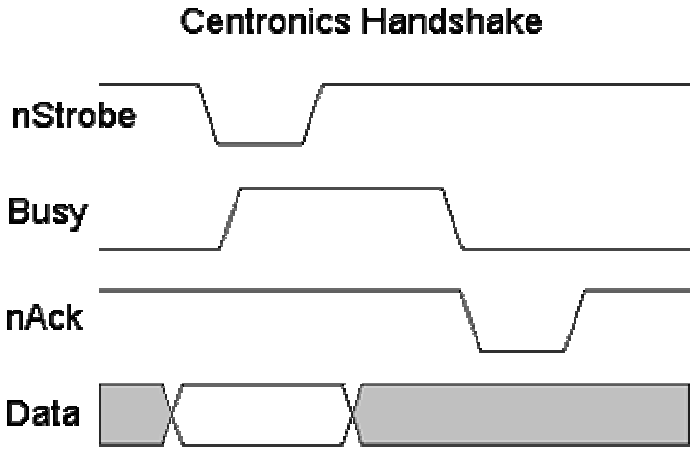
میزبان (ماشین) با قرار دادن این پایه در حالت Low وسیلهٔ خارجی را دوباره تنظیم (RESET) می کند. شما می توانید در هر زمان که بخواهید Reset ایجاد کنید. (اینکار توسط صدا زدن PLPIOCRESET انجام می شود).

EOI

پرینتر از این پایه در حالت High برای اعلام اینکه جوهر تمام کرده ، استفاده می کند. شما می توانید از این پایه برای اهداف دلخواه خود استفاده کنید.

۱-۵) استاندارد *Centronics*

Centronicsیک استاندارد قدیمی برای انتقال داده از یک میز بان به یک پرینتر می باشد. اکثر پرینترها از این hand shake استفاده می کنند. این hand shake بصورت نرمال با استفاده از یک پورت پارالل استاندارد، تحت یک نرم افزار کنترلی قابل پیاده سازی است. در زیر یک دیاگرام ساده از پروتکل Centronics آورده شده است.



داده در ابتدا روی پینهای ۲ تا ۹ پورت پارالل قرار داده می شود، سپس میزبان چک می کند، که آیا پرینتر مشغول است یا نه. بطوریکه خط Busy باید در حالت Low باشد. برنامه، Strobe را برای حداقل ۱ میکروثانیه فعال می کند، و سپس آنرا غیر فعال می کند. داده توسط پرینتر/ وسیلهٔ جنبی در زمان لبهٔ بالا روندهٔ Strobe خوانده می شود. و در این زمان پرینتر از طریق خط Busy می فهماند که مشغول پردازش داده است. وقتی که پرینتر داده را پذیرفت، از طریق یک پالس منفی در حدود ۵ میکروثانیه، روی خط nAck اعلان پذیرش میکند. در بعضی از مواقع میزبان برای صرفه جویی در وقت از کنترل خط nAck صرف نظر می کند. اخیرا شما در پورتهای ECP یک مود سریع Contronics می بینید، که به سخت افزار اجازه می دهد، همهٔ hand shakingها را خودش برای شما انجام دهد.فقط برنامه نویس باید داده را روی پورت I/O بنویسد، خود سخت افزار کنترل خواهد کرد که پرینتر مشغول است یا نه، و Strobe تولید خواهد کرد. این مود معمولا nAck را چک نمی کند.

۱-٦) آدرسهای پورت (Port Addresses)

پورت پارالل دارای ۳ پایه آدرس متداول می باشد، که در جدول زیر آمده است. پایه آدرس 3BCh در واقع برای معرفی پورتهای پارالل، واقع بر ویدئو کارتهای قدیمی بوجود آمد. این آدرس بعدها، وقتی که پورت پارالل از روی ویدئوکارتها حذف گردید، منسوخ گشت. و امروزه از آن بعنوان انتخابی دیگر برای پورت پارالل، روی مادربوردها استفاده می شود، که از طریق BIOS قابل تنظیم و تغییر است.

LPT1 بطور نرمال به پایه آدرس 378h اطلاق می گردد، در حالیکه LPT2 به 278h نسبت داده می شود. 278h & 378h همیشه برای استفادهٔ پورت پارالل در نظر گرفته شد. حرف "h" به این معنی است، که آدرس در مبنای ١٦ (hexadecimal) است.

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated on to Video Cards - Doesn't support ECP addresses
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

جدول ۱-۲) آدرس پورتها Port Addresses

وقتی که کامپیوتر در ابتدا روشن می شود BIOS (Basic Input Output System) تعداد پورتهای موجود را مشخص می کند، و وسایل جانبی را با نامهای LPT1, LPT2, LPT3 را به آنها منتسب می کند. BIOS ابتدا، آدرس 3BCh را کنترل می کند، اگر پورت پاراللی یافت شد آنرا LPT1 فرض می کند. سپس در مکان 378h جستجو می کند، اگر کارت پاراللی در آن یافت شد، به آن برچسب وسیلهٔ آزاد بعدی را لقب خواهد داد. یعنی اگر در مکان 3BCh کارتی یافت نشد، نام آن LPT1 خواهد بود، ولی اگر در 3BCh کارت پاراللی وجود داشت، نام مکان جدید LPT2 خواهد بود.آخرین پورت مورد جستجو 278h خواهد بود که، روتینی مشابه دو پورت قبل را طی خواهد کرد. چیزی که مسئله را کمی پیچیده می کند، این است که بعضی از

سازندگان کارتهای پورت پارالل جامپرهایی(jumper) را در نظر گرفته اند که، به شما اجازه می دهد، تا پورت را روی LPT1, LPT2, LPT3 تنظیم کنید. حال کدام آدرس LPT1 است؟ در اکثر کارتها، 378h ، LPT1 و 278h ،LPT2 است. اما بعضی دیگر 3BCh را بعنوان LPT1 و 278h را بعنوان LPT2 در نظر می گیرند.
وسایل جانبی ملقب به LPT3, LPT2, LPT1 نباید موجب نگرانی کسانی گردد، که می خواهند وسیله ای را توسط مدار واسط به کامپیوتر خود متصل کنند. اغلب اوقات آدرس پایه را به LPT1 نسبت می دهند. و شما می توانید برای پیدا کردن آدرس LPT1 به Lookup Table مهیا شده توسط BIOS مراجعه کنید. وقتی BIOS آدرسی را به وسیلهٔ متصل شدهٔ شما نسبت میدهد، آن آدرس را در مکان خاصی از حافظه، که ما می توانیم به آن دسترسی داشته باشیم قرار می دهد. به جدول زیر توجه کنید:

Start Address	Function
0000:0408	LPT1's Base Address
0000:040A	LPT2's Base Address
0000:040C	LPT3's Base Address
0000:040E	LPT4's Base Address (Note 1)

LPT Addresses in the BIOS Data Area جدول ۱-۳)

جدول فوق آدرس پورت پرینترهایی را نمایش می دهد که ما می توانیم، آنها را در محیط دادهٔ BIOS پیدا کنیم. هر آدرس شامل دو بایت است. نمونه برنامهٔ نوشته شده به زبان C که در زیر ملاحظه می کنید، نشان می دهد که چگونه می توان مکان اختصاص داده شده به آدرس پورتهای پرینتر را مشخص نمود.

```
include <stdio.h>#
#include <dos.h>

void main(void)
{
unsigned int far *ptraddr;  /* Pointer to location of Port
Addresses */
unsigned int address;       /* Address of Port*/
int a;

ptraddr=(unsigned int far *)0x00000408;
```

```
for (a = 0; a < 3; a++)
{
address = *ptraddr;
if (address == 0)

printf("No port found for LPT%d \n",a+1);
else
printf("Address assigned to LPT%d is %Xh\n",a+1,address);

                *ptraddr++;
                }
                }
```

۱-۷) ثباتهای نرم افزار در پورت پارالل استاندارد

Software Registers - Standard Parallel Port (SPP)

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7
			Bit 6	Data 6
			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

جدول ۱-٤) Data Port

Note 1 : If the Port is Bi-Directional then Read and Write Operations can be performed on the Data Register.

آدرس پایه معمولا، پورت داده یا ثبات داده نامیده می شود، و عموما برای خروج داده روی خطوط دادهٔ پورت پارالل (پینهای ۲ تا ۹) استفاده می شود. این ثبات بطور نرمال تنها یک پورت فقط نوشتنی است. اگر شما از پورت بخوانید، مجبورید آخرین بایت فرستاده شده را دریافت کنید، درحالیکه اگر پورت شما دوطرفه باشد، شما می توانید روی این آدرس، داده دریافت کنید.(برای اطلاعات بیشتر به بخش پورتهای دوطرفه رجوع کنید).

Offset	Name	Read/Write	Bit No.	Properties
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

جدول ۱-۵) Status Port

پورت وضعیت (base address + 1) یک پورت فقط خواندنی است. هر داده ای که روی این پورت نوشته شود، در نظر گرفته نخواهد شد. پورت وضعیت از ۵ خط ورودی (پینهای ۱۰، ۱۱، ۱۲، ۱۳ و۱۵) و یک ثبات وضعیت *IRQ* و دو بیت رزرو ساخته شده است. توجه داشته باشید که بیت ۷ (BUSY) یک ورودی Active Low است. برای مثال اگر بیت هفت نمایش دهندهٔ منطق صفر باشد، بدین معنی است که ولتاژ +5v روی پین ۱۱ قرار دارد. بطور مشابه اگر بیت ۲ (nIRQ) نمایانگر"1" باشد، آنگاه وقفه ای رخ نداده است.

Offset	Name	Read/Write	Bit No.	Properties
Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

جدول ۱-٦) Control Port

پورت کنترل (base address + 2) بصورت یک پورت فقط نوشتنی در نظر گرفته شده است. وقتی که یک پرینتر به پورت پارالل متصل است، چهار کنترل مورد استفاده

واقع شده است، که شامل Strobe, Auto Linefeed, Initialize و Select Printer می باشد که همگی بجز Initialize بصورت معکوس هستند.

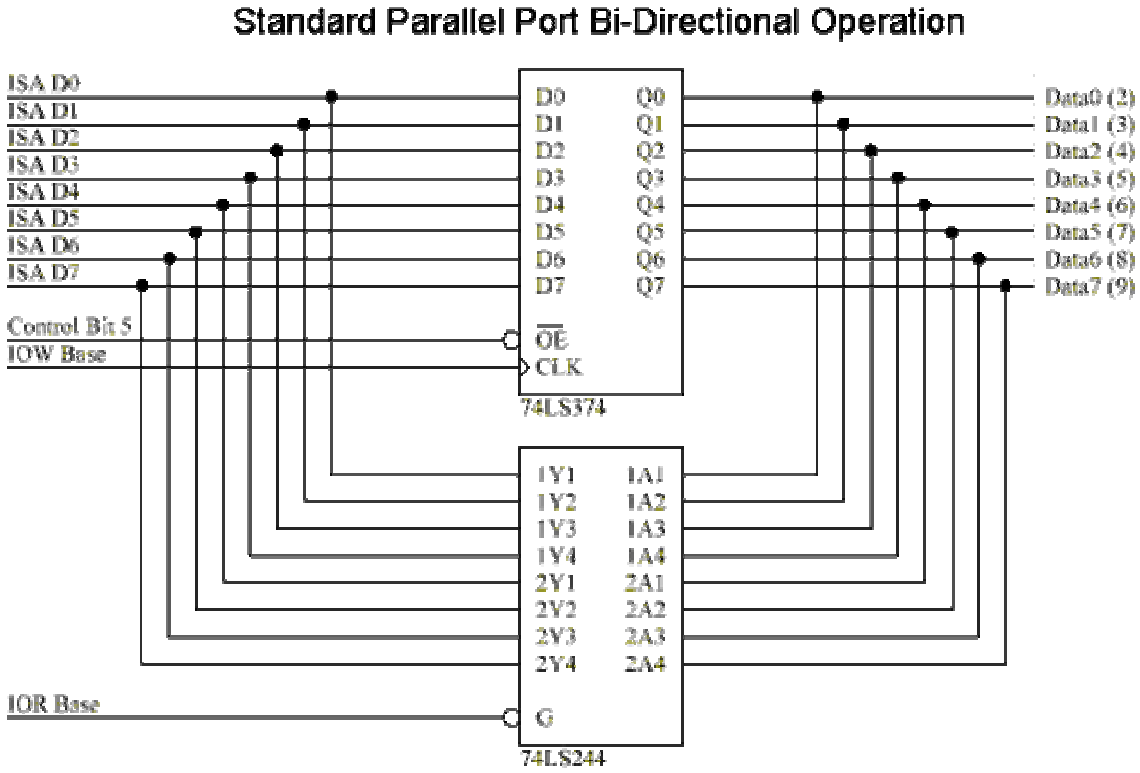
پرینتر نه سیگنالی برای مقدار دهی اولیه (Initialize) کامپیوتر می فرستد، و نه به کامپیوتر می گوید که از auto linefeed استفاده کند. به هر حال این چهار خروجی بعنوان ورودی هم قابل استفاده هستند.وقتی کامپیوتر یک پین را در حالت high (+5v) قرار می دهد، و وسیلهٔ جانبی شما می خواهد آنرا به low تغییر دهد، شما باید بطور موثری پورت را آزاد کنید تا از بروز conflict روی آن جلوگیری شود. به همین دلیل این خطوط، بصورت خروجی open collector هستند. این بدان معنی است که دارای دو حالت هستند. یک حالت low (0v) و یک حالت امپدانس بالا (مدار باز).

بطور معمول کارت پرینتر، مقاومتهای داخلی از نوع pull-up خواهد داشت، اما همیشه اینطور نیست. بعضی ها فقط دارای خروجی کلکتور باز هستند، در حالیکه ممکن است بقیه دارای خروجی قطب مضاعف نرمال باشند.در هر حال برای اینکه شما بتوانید وسیله ای بسازید، که روی اکثر پورتها دارای عملکرد صحیح باشد، می توانید از یک مقاومت خارجی استفاده کنید. از یک مقاومت خارجی 4.7k می توان برای کشیدن پین به high استفاده کرد. وقتی پین پورت پارالل در حالت high (+5v) باشد، وسیلهٔ خارجی می تواند پین را به low بکشد، و پورت کنترل مقدار مخالفی را نشان دهد. از این طریق از چهار پین پورت کنترل می توان بصورت دو طرفه برای انتقال داده استفاده کرد.به هر حال پورت کنترل باید با مقدار xxx01100 تنظیم شود، تا قادر باشد داده را بخواند، که در اینصورت همه پینها +5v هستند، و لذا شما می توانید آنرا بسمت زمین (GND) بکشید (منطق صفر کنید).

بیتهای ٤و٥ کنترلهای داخلی هستند. بیت ٤، IRQ را ممکن می سازد(به بخش IRQ پورت پارالل مراجعه کنید) و بیت ٥ اجازه استفادهٔ دوطرفه از پورت را می دهد، یعنی شما می توانید ٨ بیت را با استفاده از (DATA 0-7) وارد کنید. این مود فقط برای کارتهایی که آنرا پشتیبانی می کنند ممکن است. بیتهای ٦و٧ رزرو هستند، هرگونه نوشتاری بر روی این دو بیت نادیده گرفته می شود.

۱–۸) پورتهای دو طرفه (*Bi-directional Ports*)

دیاگرام شماتیک زیر نمایی ساده از رجیستر دادهٔ پورت پارالل را نشان میدهد. پیاده سازی کارتهای پورت پارالل اصلی بصورت منطق 74LS است. امروزه همهٔ اینها بصورت مجتمع در یک ASIC قرار دارند، ولی تئوری عملکرد آنها هنوز یکسان است.



پورتهای غیر دوطرفه که با خروجی 74LS374 ساخته شده اند، بصورت همیشه خروجی استفاده می شوند. وقتی شما رجیستر دادهٔ پورت پارالل را می خوانید، داده ها از طریق 74LS374 که به پینهای داده متصل هستند وارد می شوند. پورتهای دوطرفه از بیت کنترل ۵ که به پایهٔ Output Enable آی سی 374 متصل است، استفاده می کنند، لذا راه اندازهای خروجی قابلیت خاموش شدن دارند. از این طریق شما می توانید داده های موجود بر پینهای دادهٔ پورت پارالل را بدون اینکه تضادی (conflict) ایجاد شود بخوانید.

بیت ۵ کنترل عملکرد دوطرفهٔ پورت پارالل را فعال یا غیر فعال می کند، که فقط روی پورتهای دوطرفه ممکن است.وقتی که این پین '1' می شود، پینهای ۲ تا ۹ به حالت

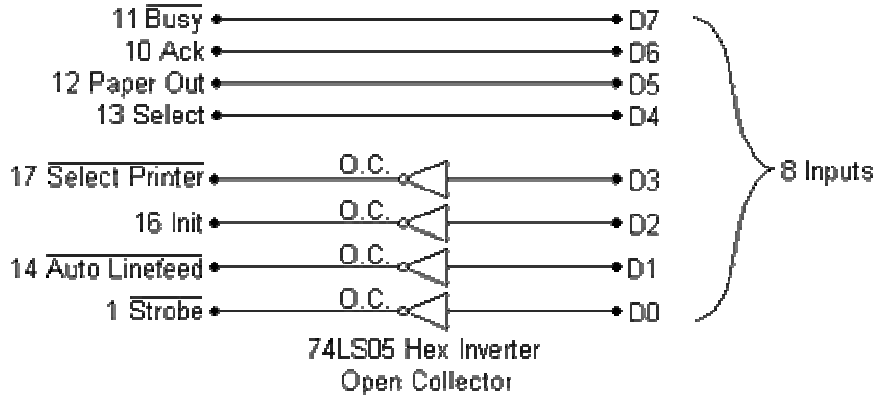
امپدانس بالا می روند. در این حالت شما می توانید یکبار بر روی این خطوط داده وارد کنید، و آنرا از پورت داده (آدرس پایه) بازیابی کنید. هر داده ای که روی پورت داده نوشته شود ذخیره خواهد شد، ولی روی پینهای داده قابل دسترسی نخواهد بود. برای خاموش کردن مود دوطرفه، بیت ۵ پورت کنترل را '0' کنید.

به هر حال همهٔ پورتها یکسان نیستند. در برخی از پورتها نیاز است که بیت ٦ پورت کنترل برای فعال کردن مود دوطرفه، وبیت ۵ برای غیر فعال کردن مود دوطرفه set گردد. سازندگان مختلف، پورتهای دوطرفه خود را بصورتهای مختلفی پیاده سازی کرده اند. اگر شما قصد دارید تا از پورت خود برای ورود داده استفاده کنید، ابتدا آنرا با یک پروب منطقی یا مولتی متر کنترل کنید، تا از قرار داشتن آن در مود دوطرفه مطمئن شوید.

۱–۹) استفاده از پورت پارالل در ورود ۸ بیت

Using The Parallel Port to Input 8 Bits.

اگر پورت پارالل شما مود دوطرفه(bi-directional) را پشتیبانی نمی کند، مایوس نشوید. شما می توانید حداکثر ۹ بیت را در هر نوبت وارد کنید. برای اینکار شما می توانید از ۵ خط ورودی پورت وضعیت، و٤ خط خط ورودی(open collector)پورت



کنترل استفاده کنید.

ورودی های پورت پارالل بصورت زیر انتخاب می شوند تا کار ما را آسانتر کنند. Busy بعنوان MSB (بیت ۷) در نظر گرفته می شود، سپس Select, Paper Out, Ack بقیهٔ سازندگان چهار بیت پر ارزش خواهند بود. مابقی ورودی ها بصورت سخت افزاری معکوس هستند، یعنی +5v نمایانگر صفر بودن بیت معادل در رجیستر، و GND معرف '1' است.

پورت کنترل برای خواندن چهار بیت کم ارزش استفاده می شود. بطوریکه قبلاً گفته شد، خروجی های پورت کنترل بصورت کلکتورباز هستند، یعنی دارای دو حالت امپدانس بالا و زمین هستند. اگر ما ورودیها را مستقیماً به پورت متصل کنیم (برای مثال یک ADC0804 با خروجی پل مضاعف) ، وقتی که پورت بخواهد به حالت low برود، و خروجی ها بصورت high باشند، یک conflict رخ خواهد داد. لذا ما از معکوس کنندهٔ کلکتور باز استفاده می کنیم، هر چند که این کار همیشه ضروری نیست. اگر ما سوئیچ های تک قطبی را از طریق یک مقاومت pull-up به پورت متصل کنیم، دیگر نیازی به این محافظت نخواهد بود. همچنین اگر نرم افزار شما پورت کنترل را با xxxx0100

مقدار دهی بکند، همهٔ پینهای پورت کنترل high خواهند بود، که باز هم نیاز به آن نخواهید داشت.

مشکل دیگری که باید از آن اطلاع داشت، مقاومتهای pull-up پورت کنترل است. معمولاً مقدار این مقاومتها 4.7k است. در این صورت برای اینکه وسیلهٔ جانبی شما آن خط را به حالت low ببرد، باید جریانی معادل 1mA بکشد، که بعضی از وسایل کم قدرت را با مشکل مواجه می کند. حال اگر این مقاومتها 1k باشد چه می شود؟ بله چنین کارتهایی هم وجود دارد، و وسیلهٔ جانبی شما باید جریانی معادل ۵ میلی آمپر بکشد. اینها دلایل بیشتری برای استفاده از معکوس کنندهٔ کلکتورباز هستند. علاوه بر معکوس کنندهٔ کلکتورباز می توان از بافرهای کنندهٔ کلکتورباز هم استفاده کرد. امکان دیگر استفاده از ترانزیستورها است.

ورودیD3 از طریق معکوس کننده بهSelect printerمتصل است، Select printer بیت ۳ پورت کنترل است. D0, D1, D2 به Strobe, Auto Linefeed, Init متصل هستند، که چهار بیت کم ارزش را تشکیل می دهند. حال می توانیم از طریق نرم افزار بقیه کارها را دنبال کنیم. در ابتدا باید در پورت کنترل مقدار xxxx0100 را قرار داد. این عمل موجب high شدن همهٔ خطوط پورت کنترل می شود، لذا آنها می توانند برای ورود داده به low تغییر داده شوند.

```
outportb(CONTROL, inportb(CONTROL) &
                  0xF0 | 0x04);
```

حال که این کار انجام شد، می توان چهار بیت پر ارزش را خواند، که همان چهار بیت پرارزش پورت وضعیت خواهند بود. چون ما فقط دنبال چهار بیت پرارزش هستیم، پاسخ را با 0xF0 ، AND می کنیم، که در اینصورت چهار بیت کم ارزش پاک خواهد شد.

```
a = (inportb(STATUS) & 0xF0); /* Read
MSnibble */
```

اکنون می توانیم نیبل کم ارزش را بخوانیم، که همان چهار بیت کم ارزش پورت کنترل خواهد بود. در این نوبت ما دنبال نیبل پرارزش نیستیم، لذا پاسخ را با 0x0F ، AND می کنیم، تا نیبل پرارزش پاک شود.اولین کار پس از آن ادغام دو بایت حاصل است. اینکار با OR کردن دوبایت امکانپذیر است، که یک بایت را برای ما بوجود خواهد آورد. اما هنوز کار تمام نشده، چون بیتهای ۲و۷ معکوس هستند. برای رفع این مشکل بایت را با 0x84 ،XOR می کنیم، که آندو بیت را toggle می کند.

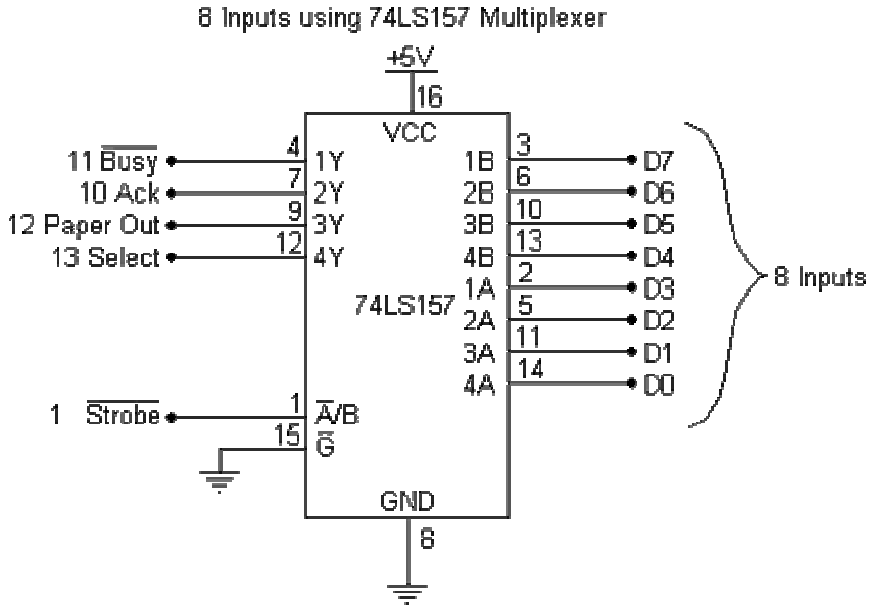
```
a = a |(inportb(CONTROL) & 0x0F); /* Read
LSnibble */
a = a ^ 0x84; /* Toggle Bit 2 & 7 */
```

توجه: بعضی از پورتهای کنترل کلکتورباز نیستند، اما دارای خروجی پل مضاعف هستند، این موردی است که برای پورتهای EPP, ECP صدق می کند.

بصورت نرمال وقتی شما یک پورت را در مود ECPیا EPP قرار میدهید، پورت کنترل فقط بصورت پل مضاعف خواهد بود. اکنون اگر شما وسیلهٔ خود را به پورت پارالل متصل کنید، چه اتفاقی خواهد افتاد؟ برای ایجاد حالت پرتابلی (portability) توصیه می کنیم از مدار بعدی برای خواندن یک نیبل در هر نوبت استفاده

۱-۱۰) مود چهار بیتی (*Nibble Mode*)

مود نیبل یکی از راههای مناسب برای خواندن ۸ بیت داده، بدون قرار دادن پورت در حالت معکوس ویا استفاده از خطوط داده است. در مود نیبل از یک مالتی پلکسر ۲x٤ استفاده شده است، که دارای دو ورودی چهارتایی است، و می تواند در هرنوبت یک نیبل را بخواند و در نوبت دوم به نیبل دوم سوئیچ کند. با استفاده از نرم افزار می توان دو نیبل را به یک بایت تبدیل کرد. تنها عیب این روش سرعت کم آن است. چون نیاز به چند دستورالعمل برای خواندن یک بایت دارد، و نیاز به یک آی سی خارجی نیز هست.



عملکرد مالتی پلکسر ۲x٤ ، 74LS157 بسیار ساده است. این آی سی بسادگی بعنوان چهار کلید عمل میکند. وقتی که ورودی A/B در حالت Low قرار دارد، ورودی های A انتخاب شده اند، یعنی 1A از طریق 1Y ، 2A از طریق 2Y و... منتقل می شوند. وقتی A/B در حالت high قرار دارد، ورودیهای B از طریق خروجی های Y که به پورت وضعیت پارالل پورت متصل هستند، منتقل می شوند.

برای استفاده از مدار فوق باید ابتدا مالتی پلکسر را به یکی از حالات A یا B قرار دهیم. می خواهیم ابتدا نـیبل کم ارزش را بخوانیم، پس باید A/B را در حالت Low

قرار دهیم. Strobe بصورت سخت افزاری معکوس شده است، لذا ما باید بیت صفر پورت کنترل را set کنیم، تا در پین ۱ حالت Low داشته باشیم.

```
outportb(CONTROL, inportb(CONTROL) | 0x01);/* Select low
                                              Nibble(A)*/
```

ابتدا که نیبل پایین انتخاب شد، می توان آنرا از پورت وضعیت خواند. به خاطر داشته باشید خط Busy معکوس شده است، هر چند که اکنون با آن کاری نداریم. ما تنها می خواهیم که MS nibble را داشته باشیم، لذا پاسخ را با 0xF0 ، AND می کنیم تا LS nibble پاک شود.

```
a = (inportb(STATUS) & 0xF0); /* Read Low
                                 Nibble */
```

اکنون نوبت آن است که این نیبل را شیفت دهیم تا در قسمت LS nibble متغییر a قرار گیرد.

```
a = a >> 4; /* Shift Right 4 Bits
               */
```

تا اینجا نیمی از راه طی شده و اکنون زمان گرفتن MS nibble است، لذا باید مالتی پلکسر را برای انتخاب ورودی B تغییر وضعیت دهیم. سپس می توانیم MS nibble را بخوانیم و دو نیبل را برای ساختن یک بایت با هم ادغام کنیم.

```
outportb(CONTROL, inportb(CONTROL) & 0xFE);/* Select High
                                              Nibble(B)*/
        a = a |(inportb(STATUS) & 0xF0); /* Read High Nibble */
                                            byte = byte ^ 0x88;
```

آخرین خط برنامه دو بیت معکوس را که توسط خط Busy خوانده می شود Toggle می کند. اگر پاسخ اشتباهی دریافت کردید، شاید لازم باشد که مقداری وقفه (delay) به پروسسورتان بدهید.

۱-۱۱) بکارگیری IRQ پورت پارالل

Using the Parallel Port's IRQ

درخواست وقفه(interrupt request) پارالل پورت برای چاپ، تحت DOS و Windows استفاده نمی شود. نسخه های قدیمی OS-2 از آنها استفاده می کردند. استفاده از وقفه ها برای مانیتور کردن وسایل جانبی خوب است، مثل اخطار گرم شدن وسیله وغیره، ولی شما نمی دانید که وقفه در چه زمانی فعال خواهد شد. اگر یک درخواست وقفه داشته باشیم، بهتر از آن است که بخواهیم توسط نرم افزار، پورت را از نظر اینکه چیزی تغییر کرده یا نه کنترل کنیم. این عمل وقتیکه شما بخواهید از کامپیوترتان برای عملیات چند وظیفه ای (multi tasking) بهره بگیرید، نمود بیشتری پیدا می کند.

درخواست وقفهٔ پارالل پورت بصورت نرمال IRQ5 یا IRQ7 است، ولی اگر اینها تحت استفاده باشند، ممکن است مقادیر دیگری باشد.حتی ممکن است وقتی که کارت پورت فقط برای پرینت کردن مورد استفاده قرار می گیرد، وقفه ها را کاملاً غیر فعال کرد. IRQ از طریق خط ACK فعال می شود. وقتی که فعال باشد، وقفه از طریق انتقال(transition) لبهٔ بالا رونده nACK اطلاع داده می شود. به هر حال ممکن است بعضی از کارتها از لبهٔ پایین رونده برای اعلان وقفه استفاده کنند.

کد برنامهٔ زیر یک تست کنندهٔ پلاریته وقفه استاین کد تعیین می کند که وقفهٔ پورت پارالل شما دارای چه پلاریته ای است، همچنین نحوهٔ استفاده از وقفه پورت پارالل را هم نشان میدهد. این وقفه مشخص می کند که وقفه در لبهٔ بالا رونده خط nACK رخ می دهد و یا در لبهٔ پایین روندهٔ آن. برای استـفاده از برنامه به سادگی می توانید یکی از خطوط داده(پینهای ۲ تا ۹) را به پین ACK (پین ۱۰) با سیم متصل کنید. آسانترین کار این است که یک اتصال از DATA 7(پین ۹) به ACK(پین ۱۰) روی کانکتور DB25 برقرار کنید.

۱-۱۲) مودهای پارالل پورت در BIOS

Parallel Port Modes in BIOS

امروزه پورتهای پارالل بصورت مالتی مود هستند. آنها از طریق BIOS به چندین مود قابل تنظیم هستند. شاخصترین مودها به شرح زیر هستند:

Printer Mode (Sometimes called Default or Normal Modes)
Standard & Bi-directional (SPP) Mode
EPP1.7 and SPP Mode
EPP1.9 and SPP Mode
ECP Mode
ECP and EPP1.7 Mode
ECP and EPP1.9 Mode

Printer Mode مبنایی ترین مود است که فقط یک مود پارالل پورت استاندارد است، و دارای ویژگی دو طرفه نیست، لذا بیت ۵ پورت کنترل آن پاسخ نخواهد داد. ***SPP Mode*** یک مود دوطرفه است. با استفاده از این مود بیت ۵ پورت کنترل مسیر را معکوس می کند، لذا شما می توانید یک مقدار را روی خطوط داده بخوانید. ***EPP1.7 and SPP Mode*** ترکیبی است از مودهای SPP و EPP 1.7 (Enhanced parallel port). در این مود عملیاتی، شما به رجیسترهای SPP (Data, Status, Control) و رجسیترهای EPP دسترسی خواهید داشت. در این مود شما قادر خواهید بود که مسیررا با استفاده از بیت ۵ پورت کنترل معکوس کنید. شاید نسخهٔ 1.7 از EPP شامل بیت Time-Out نباشد. مود EPP 1.9 و SPP شبیه مود قبلی است، که تفاوتش استفاده از نسخهٔ EPP 1.9 است. در این مود شما به بیت EPP Timeout دسترسی خواهید داشت.

ECP(Extended Capabilities Port) به شما یک پورت با قابلیت توسعه یافته می دهد. این مود توسط set کردن کنترل رجیستر توسعه یافته (ECR) قابل وصول است. در این مود از طریق BIOS ، EPP Mode (100) مقدور نخواهد بود. مود EPP 1.7 و ECP و مود EPP 1.9 و ECP هم به شما پورت با قابلیت توسعه یافته را خواهد داد، که شبیه مود قبلی خواهد بود. به هر حال مود EPP در ECR مودECP قابل وصول نخواهد بود. اگر شما در مود ECP و1.7 EPP باشید، یک پورت EPP 1.7 و اگر در مود ECP و EPP 1.9 باشید یک پورت EPP 1.9 خواهید داشت.

مودهای فوق از طریق BIOS قابل انتخاب هستند. شما می توانید بوسیلهٔ نرم افزار خود آنها را دوباره تنظیم (reconfigure) کنید، ولی اینکار توصیه نمی شود. این ثباتهای نرم افزاری عمدتاً در آدرسهای 0x2FA, 0x3F0 و 0x3F1 و غیره یافت می شوند، که عمدتاً فقط از طریق BIOS قابل دستیابی هستند. برای این ثباتها هیچ مجموعه استانداردی وجود ندارد، لذا اگر بخواهید از این ثباتها استفاده کنید، نرم افزار شما خیلی قابلیت حمل (portability) نخواهد داشت. با سیستمهای عامل چندوظیفه ای امروزی عقیدهٔ جالبی نیست که آنها را تغییر دهید، در حالیکه برای استفادهٔ شما مناسب هستند.

گزینش بهتر اینست که مود EPP 1.7, ECP یا مود EPP 1.9, ECP را از BIOS انتخاب کنید و سپس از رجیستر کنترل توسعه یافته ECP برای انتخاب مود پارالل پورت خود بهره ببرید. ولی در نهایت بهترین انتخاب مود EPP 1.9,ECP می باشد.

۱-۱۳) مودهای پورت پارالل وکنترل رجیستر توسعه یافتهٔ ECP

Parallel Port Modes and the ECP's Extended Control Register

همانطوریکه کمی قبل بحث کردیم، بهترین تنظیم پورت پارالل EPP 1.9, ECP و استفاده از کنترل رجیستر توسعه یافتهٔ ECP برای انتخاب مودهای مختلف عملیاتی است. ثباتهای ECP تحت استاندارد:

Microsoft's Extended Capabilities Port Protocol and ISA Interface Standard

قرار دارند. وقتی در مود ECP قرار داریم یک مجموعه از ثباتها در آدرس پایه + 0x400h قابل دستیابی هستند. در اینجا فقط قصد داریم کنترل رجیستر توسعه یافته (ECR) را که در آدرس پایه + 0x402h قرار دارد بررسی کنیم. باید دانست که ثباتهای ECP برای پورتهایی با آدرس پایهٔ 0x3BCh قابل دستیابی نیستند.

Bit		Function
7:5		*Selects Current Mode of Operation*
	000	Standard Mode
	001	Byte Mode
	010	Parallel Port FIFO Mode
	011	ECP FIFO Mode
	100	EPP Mode
	101	Reserved
	110	FIFO Test Mode
	111	Configuration Mode
4		ECP Interrupt Bit
3		DMA Enable Bit
2		ECP Service Bit
1		FIFO Full
0		FIFO Empty

جدول ۱-۷) : Extended Control Register (ECR)

جدول فوق مر بوط به ECR است. ما فقط مایلیم که سه بیت پرارزش (MSB) از کنترل رجیستر توسعه یافته که مودهای عملیاتی را انتخاب می کنند، بررسی کنیم. لذا ۷ مود عملیاتی قابل بررسی هستند. ولی همهٔ پورتها همهٔ مودها را پشتیبانی نمی کنند. مود EPP یک نمونه است که روی همهٔ پورتها قابل وصول نیست.

Modes of Operation

Standard Mode Selecting this mode will cause the ECP port to behave as a Standard Parallel Port, without bi-directional functionality.

Byte Mode / PS/2 Mode Behaves as a SPP in bi-directional mode. Bit 5 will place the port in reverse mode.

Parallel Port FIFO Mode In this mode, any data written to the Data FIFO will be sent to the peripheral using the SPP Handshake. The hardware will generate the handshaking required. Useful with non-ECP devices such as Printers. You can have some of the features of ECP like FIFO buffers and hardware generation of handshaking but with the existing SPP handshake instead of the ECP Handshake.

ECP FIFO Mode	Standard mode for ECP use. This mode uses the ECP Handshake described in Interfacing the Extended Capabilities Port. - *When in ECP Mode though BIOS, and the ECR register is set to ECP FIFO Mode (011), the SPP registers may disappear.*
EPP Mode*/Reserved*	This will enable EPP Mode, if available. Under BIOS, if ECP mode is set then it's more than likely, this mode is not an option. However if BIOS is set to ECP and EPP1.x Mode, then EPP 1.x will be enabled. - *Under Microsoft's **Extended Capabilities Port Protocol and ISA Interface Standard** this mode is Vendor Specified.*
Reserved	Currently Reserved. - *Under Microsoft's **Extended Capabilities Port Protocol and ISA Interface Standard** this mode is Vendor Specified.*
FIFO Test Mode	While in this mode, any data written to the Test FIFO Register will be placed into the FIFO and any data read from the Test FIFO register will be read from the FIFO buffer. The FIFO Full/Empty Status Bits will reflect their true value, thus FIFO depth, among other things can be determined in this mode.
Configuration Mode	In this mode, the two configuration registers, cnfgA & cnfgB become available at their designated register addresses.

اگر شما در مود ECP تحت BIOS هستید، یا کارت شما توسط jumper روی ECP قرار دارد، خوب است که پورت ECP خود را قبل از استفاده یکبار مقدار دهی اولیه کنید. اگر شما از SPP استفاده می کنید پس قبل از هر چیز پورت را در مود استاندارد تنظیم کنید، و فرض نکنید که خود پورت در حالت استاندارد SPP قرار دارد. تحت بعضی از مودها ممکن است ثباتهای SPP ناپدید شوند و یا درست کار نکنند. اگر شما از SPP استفاده می کنید پس ECR را در مود استاندارد قرار دهید. این خطایی است که اکثراً مرتکب آن می شوند.

۲

فصل دوم

موتورهای پله ای

۲ –۱) آشنایی با موتور پله ای (STEPPER MOTOR)

موتور پله ای نوعی موتور سنکرون است وبگونه ای طراحی شده است که حرکات معین و محدود زاویه ای در ازای دریافت پالسهای خاصی، روی روتورش ایجاد می کند. انواع مختلف این نوع موتورها می توانند در ازای دریافت پالسهای معیین در سیم پیچهای خود، حرکت زاویه ای باندازهٔ ۹/. °، ۱/۸°، ۲°، ۲/۵°، ۵°، ۷/۵°، ۱۵°، ٤٥° و ۹۰ درجه ای داشته باشند. این موتورها که بصورت دیجیتالی عمل می کنند، در صنعت و علوم گوناگون که نیاز به اعمال حرکات مکانیکی کاملاً دقیق و کنترل شده می باشد، کاربرد فراوانی دارند. با توجه به ساختمان داخلی و نحوهٔ طراحی اینگونه موتورها، می توان حرکات دقیق و قرار گرفتن در موقعیتهای از قبل تعیین شده ای را توسط مدارات منطقی، میکروپروسسوری و کامپیوتری در سیستمهای مکانیکی ایجاد نمود.

بطور مثال میتوان از موارد استفادهٔ این موتورها در زمینه های زیر نام برد:

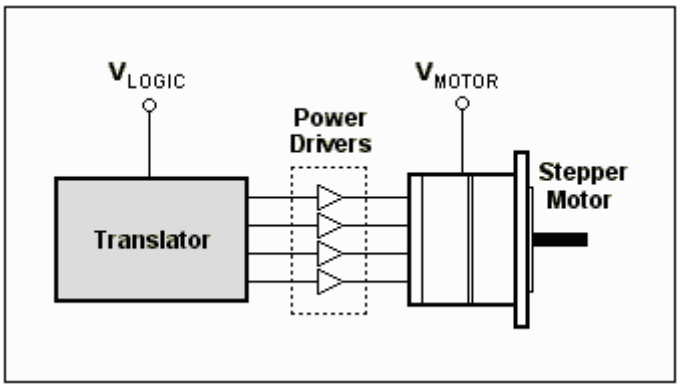
- دستگاههای مربوط به کامپیوتر : حرکت هد در دیسک سخت و فلاپی و پرینترهای مختلف، CD ROM و ...
- اتوماسیون صنعتی : پلاترهای X,Y ، ماشینهای CNC، پردازشهای لیزری و ...
- کارخانجات تولید قطعات نیمه هادی : پردازش و تولید ویفرها، انتقال و برش ویفرها، اتصالات IC و...
- تجهیزات پزشکی : پمپهای خون ، سانتریفوژها، اسپکتیو گرافها و ...
- تجهیزات اداری : فتوکپی ها ، فکسها و ...
- تجهیزات اندازه گیری : دستگاههای CMM ، اندازه گیری های میکرونی و ...

موتور پله ای بصورت دیجیتالی کنترل، و با حرکات منقطع راه اندازی می گردد و بخاطر ماهیت دیجیتالی اش می تواند براحتی توسط سیستمهای میکروپروسسوری و PLC تحت کنترل قرار گیرد. کنترل سرعت و موقعیت موتور پله ای می تواند بدون استفاده از مدار فیدبک (بازخور) و بصورت حلقه باز (open loop) و با حداقل خطای مکان و سرعت انجام گیرد. همچنین طراحی و ساخت مدارات کنترل و راه انداز این موتورها

ساده بوده و می توان بسادگی تغییرات سرعت ، جهت و کنترل موقعیت را توسط مدارات دیجیتالی پیاده سازی نمود.
موارد پیش گفته همگی از مزایای بسیار خوب موتورهای پله ای می باشد، در عین حال برای انگونه موتورها می توان معایبی را نیز برشمرد که از آن جمله محدود بودن سرعت در موتورهای پله ای است. سرعت این موتورها از فرکانس مخصوصی که توسط سازندهٔ موتور ارائه میگردد بیشتر نخواهد شد، و در صورت اعمال پالسهای سریع خطای حرکتی بوجود خواهد آمد، و نیز این موتورها بدلیل حرکتهای منقطع خود دارای لرزش می باشند که در برخی از سیستمهای مکانیکی بسیار حساس ایجاد مشکل می کند.

موتورهای پله ای بکلی با موتورهای DC معمولی متفاوتند. قبل از همه اینکه آنها خودشان به تنهایی با اتصال به منبع انرژی عمل نمی کنند. ثانیاً آنها دقیقاً کاری را انجام می دهند که از اسمشان برمی آید، یعنی در هر نوبت یک حرکت (step) انجام میدهند. همچنین در سرعت و گشتاور نیز با موتور DC متفاوتند. موتورهای DC در دورهای پایین قادر به ایجاد گشتاور زیادی نیستند، ولی بعکس موتورهای پله ای از این لحاظ توانایی زیادی دارند، و علاوه بر این دارای خاصیت گشتاور نگهدارنده (holding torgue) نیز می باشند. این ویژگی به موتور پله ای این امکان را می دهد که در زمانیکه بی حرکت است، موقعیت فیزیکی محور خود را حفظ کند، و در جاهاییکه حرکت برو بایست مورد نظر است و به محور موتور نیروی مخالف اعمال می شود بسیار مورد نیاز است و به مثابه یک ترمز عمل می کند.محور این موتورها بوسیلهٔ انرژی دار شدن سیم پیچهای موتور خود با ترتیب صحیح، شروع به دوران می کند و چنانچه این ترتیب عکس شود ، جهت حرکت نیز عکس خواهد شد. به مدار منطقی سازندهٔ این ترتیبها translator گفته می شود.
در صفحه بعد شماتیک اتصال translator و درایور (مدار راه انداز) و موتور پله ای را مشاهده می نمایید.

A basic example of the "translator + driver" type of configuration. Notice the separate voltages for logic and for the stepper motor. Usually the motor will require a different voltage than the logic portion of the system. Typically logic voltage is +5 Vdc and the stepper motor voltage can range from +5 Vdc up to about +48 Vdc. The driver is also an "open collector" driver, wherein it takes its outputs to GND to activate the motor's windings. Most semiconductor circuits are more capable of sinking(providing a GND or negative voltage) than sourcing (outputting a positive voltage).



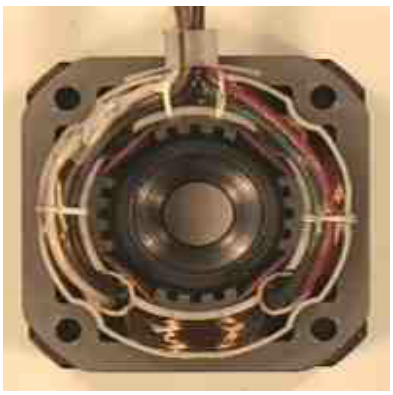
شکل ۲-۱ شماتیک اتصال یک موتور پله ای و درایور

۲-۲) ساختمان داخلي موتور پله اي

۲-۲-۱ روتور : قسمت متحرک و مرکزی موتور می باشد که غالباً بصورت مغناطیس دائم با دیواره ای بصورت دندانه ای است، که هر دندانه بعنوان یک قطب مغناطیسی عمل

مینماید.

۲-۲-۲ استاتور : قسمت ثابت موتور می باشد که دیوارهٔ استاتور هم بصورت دندانه ای بوده و این دندانه ها در داخل موتور، مقابل دندانه های روتور قرار میگیرند و با ایجاد قطبهای مغناطیسی همنام یا غیر همنام با روتور، باعث حرکت در روتور می شود.



۲-۲-۳ فازها : سیم پیچهای نصب شده بر روی استاتور می باشند که توالی پالسهای اعمال شده در آنها با نظم خاص موجب حرکت در روتور می گردد.

۲-۲-٤ دندانه ها : بر روی محیط روتور و استاتور دندانه هایی تعبیه گردیده که با فاصلهٔ بسیار کم مابین خود و استفاده از زاویه دندانه ها و تعدادشان، عامل تعیین کننده مقدار زاویه حرکت روتور با یک پالس می باشند.

ساختمان موتورهای پله ای مختلف از نظر وضعیت دندانه ها به سه دسته مختلف تقسیم می شود.

الف – تعداد دندانه های روتور و استاتور برابر می باشند، که این نوع از ساختمان در موتور با روتور چند طبقه بکار میرود و در آن نیروی وارد بر روتور بطور همزمان به تمام دندانه ها وارد می شود.

ب – تعداد دندانه های روتور و استاتور با هم برابر نمی باشند، که این نوع از ساختمان در موتورهای با روتور یک طبقه، با زاویه پله بزرگ بکار می رود که در آن نیروی وارد بر روتور بطور همزمان بر تمام دندانه ها وارد نمی شود.

ج – دندانه های استاتور بصورت گروه های قطبی قرار می گیرند ولی دندانه های روتور بطور یکنواخت روی محیط روتور قرار دارند که این نوع از ساختمان معمولاً در موتور پله ای یک طبقه با زاویه پله کوچک بکار می رود.

بطور کلی اساس کار دندانه ها و عملکرد آنها در هر سه نوع فوق تقریباً یکسان می باشد. امروزه موتورهای پله ای کاربردهای بیشماری پیدا کرده اند، به همین منظور سازندگان، انواع مختلفی چه از نظر شکل ، یا ابعاد ، قدرت، و پارامترهای دیگر در اختیار مصرف کنندگان قرار داده اند. بعنوان مثال موتورهای پله ای خطی جهت حرکات رفت و برگشت ساخته شده اند. همچنین موتورهای پله ای با روتور ثابت نیز ساخته شده است که با توجه به موارد کاربردشان مورد استفاده قرار می گیرند. و یا موتور پله ای با قدرت بالا و یا ابعاد بسیار کوچک جایگاه ویژه ای به این پدیده در صنعت امروز جهان داده است.

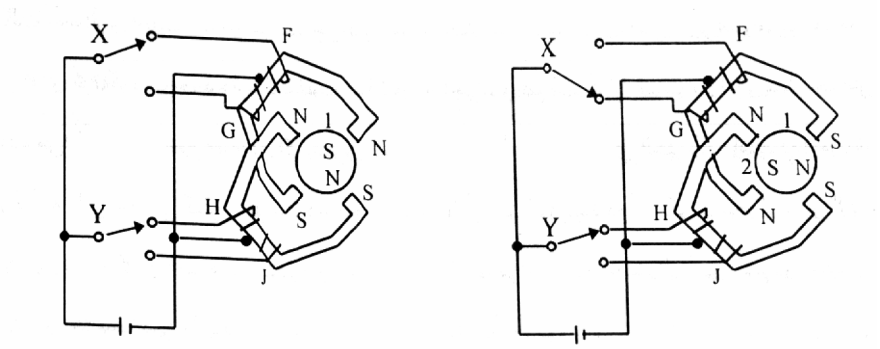
۲-۳) طبقه بندی موتورهای پله ای

الف) موتورهای پله ای نوع آهنربای دائمی :

(Permanent Magnet Stepper Motors)

شکل (۲-۲) نمای ساده ای از این نوع موتورها را نشان می دهد. روتور اینگونه موتورها از آهنربای طبیعی و معمولاً با چند زوج قطب می باشد. در اینجا موتور نشان داده شده از یک زوج قطب می باشد.

استاتور این موتور دارای چهار سری سیم پیچ می باشد که مطابق شکل زیر هر دو سیم پیچ روبروی یک زوج قطب استاتور پیچیده شده است. جهت پیچش هر یک از سیم پیچها روی زوج قطب بگونه ای است که عبور جریان الکتریکی از هر یک از آنها پلاریتهٔ زوج قطب استاتور را معکوس می کند. برای بررسی طرز کار این نوع موتورها فرض کنید سوئیچهای X,Y در وضعیت نشان داده شده در شکل (۲-۲-الف) می باشند. در این حالت سیم پیچهای F,H تغذیه می شوند و پلاریته قطبهای استاتور مطابق شکل خواهد بود. با توجه به تقارن موجود، حالت مینیمم انرژی روتور ایجاب می کند که در وضعیتی مطابق شکل قرار گیرد.



(ب) (الف)

شکل ۲-۲) موتور پله ای با آهنربای طبیعی

یعنی قطب جنوب در وضعیت ۱ باشد و هرگونه تغییر بار خارجی برای جابجایی روتور با کوپل مغناطیسی ایجاد شده خنثی می گردد. اکنون اگر وضعیت سوئیچ X تغییر کند سیم پیچ G تغذیه شده و پلاریته قطبهای استاتور مطابق شکل (۲-۲- ب) خواهد بود. این امر سبب می شود که قطب جنوب روتور به وضعیت ۲ رانده شده و در آن وضعیت، پایدار گردد.

در این صورت روتور باندازهٔ ۹۰° در خلاف جهت عقربه های ساعت چرخیده است. همینطور اگر وضعیت سوئیچ های X,Y را به ترتیب مطابق شکل (۲-۲- ج و د) قرار دهیم، قطب جنوب روتور در موقعیتهای ۳ و ٤ قرار خواهد گرفت.

(د) (ج)

ادامه شکل ۲-۲)

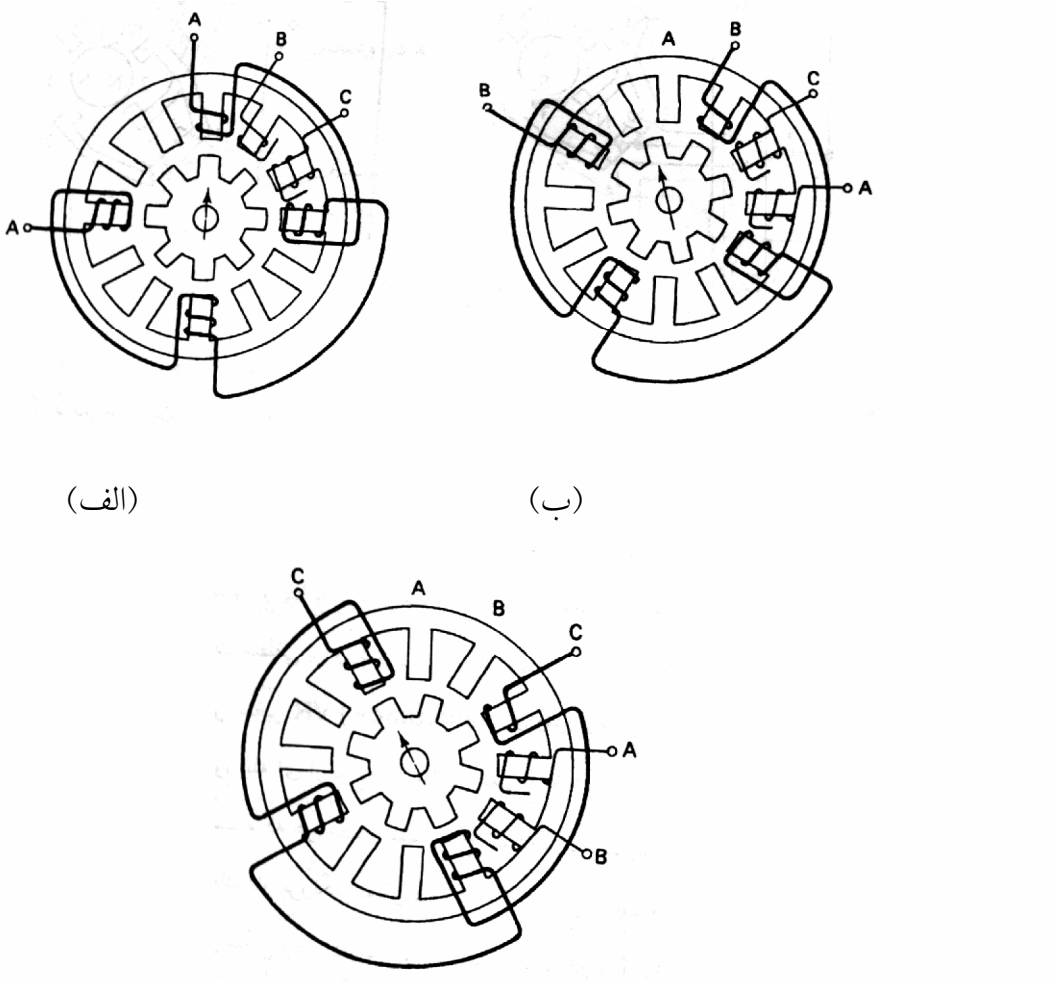
اگر در هر یک از شکلهای قبل، تغییر سوئیچ ها را بصورت عکس انجام دهیم، موتور در جهت عکس خواهد چرخید. مثلاً اگر در شکل (۲-۲- الف) بجای عوض کردن وضعیت سوئیچ X وضعیت سوئیچ Y را عوض کنیم روتور در جهت عقربه های ساعت چرخیده و به وضعیت شکل (۲-۲- د) در می آید.

ترتیب تعویض سوئیچ های X,Y و اطلاع از موقعیت سوئیچ ها در هر لحظه برای حرکت درست و پیش بینی شده موتور حائز اهمیت است. مثلاً فرض کنید در حالتیکه روتور مطابق شکل (۲-۲- الف) می باشد، سوئیچ های X,Y بطور همزمان تغییر وضعیت دهند، در این حالت ممکن است روتور اصلاً حرکتی نکند، و یا با گردش در جهت عقربه های ساعت به وضعیت ۳ برود و یا با گردش در خلاف جهت عقربه های ساعت به وضعیت ۳ برود. بنابراین استفاده از یک سری مدارات منطقی برای کنترل ترتیب و جهت حرکت موتور ضروری می باشد.

همچنین قابل ذکر است که یکی از ویژگی های خاص موتور های مغناطیس دائم این است که روتور هنگام قطع فازها نیز تحت تاثیر گشتاوری واقع شده و در یک وضعیت مشخص قرار می گیرد. این خصوصیت را مکانیزم نگهدار (Detent Mechanism) و گشتاور ایجاد شده در این حالت را گشتاور نگهدار (Detent Torque) می گویند.

ب) موتور پله ای نوع رلوکتانس متغییر (Variable - Reluctance) : VR

شکل (۲-۳) شمای یک موتور پله ای از نوع رلوکتانس متغییر (VR) را نشان می دهد. استاتور این موتور از دوازده سیم پیچ تشکیل شده که هر چهار سیم پیچ تشکیل یک گـروه (فاز) را می دهند. در این شکل سیم پیچهای مربوط به یک فاز پررنگتر نشان داده شده است. روتور از یک مادهٔ قابل مغناطیس دندانه دندانه تشکیل گردیده است. در این شکل روتور دارای هشت دندانه است.



شکل (۲-۳) موتور پله ای با رلوکتانس متغییر

در قسمت الف شکل فوق فاز A **تغذیه** شده و در نتیجه دندانه های روتور بگونه ای قرار می گیرند که حداقل رلوکتانس مغناطیسی بین روتور و استاتور بوجود آید.

اکنون اگر مطابق قسمت ب از شکل (۲-۳) تغذیه فاز A قطع و تغذیه فاز B برقرار گردد، رعایت اصل فوق باعث می گردد که روتور به اندازهٔ ۱۵ درجه چرخیده و دندانه

ها در وضعیتی که نشان داده شده اند قرار گیرند. به همین ترتیب تغذیهٔ فاز C باعث می گردد که روتور ۱۵ درجهٔ دیگر در خلاف جهت عقربه های ساعت بچرخد.
در مرحلهٔ فوق اگر بجای فاز C مجدداً فاز A تغذیه شود، روتور ۱۵ درجه در خلاف جهت عقربه های ساعت می چرخد و مجدداً به حالت شکل (۲-۳- الف) برمیگردد. ملاحظه می نمائید که در این نوع موتورها هم، به مدارات منطقی جهت رعایت ترتیب تغذیه فازها و پیش بینی جهت چرخش موتور نیاز می باشد.
موتورهای (VR) با زاویه پله کوچکتر نسبت به موتورهای (PM) قابل ساخت هستند، و بدین لحاظ در کاربردهایی که نیاز به گامهای چرخشی ظریف تری می باشد مورد استفاده قرار می گیرند. اما کوپل نگهدارنده این موتور در یک وضعیت دلخواه کمتر از نوع PM است.

مزایای موتورهای رلوکتانس متغییر (VR) **:**

- گشتاور زیاد
- سرعت پله زیاد
- پاسخ سریع
- آزاد بودن روتور در زمان قطع تحریک
- ساختمان ساده و عمر زیاد
- قابلیت حرکت در هر دو جهت

معایب موتورهای رلوکتانس متغییر (VR):

- گشتاور نگهدار صفر
- پاسخ نوسانی

موتورهای پله ای دارای مزایا ومعایب مخصوص به خود هستند. از معایب عمده آنها می توان قدرت و راندمان کم و قیمت زیاد در مقایسه با موتورهای معمولی را نام برد. اما علیرغم این معایب ماهیت دیجیتالی و مزایای دیگر آنها باعث گردیده تا مورد استفادهٔ روز افزون قرار گیرند. از مزایای این موتورها، موارد زیر قابل توجه اند:

- این موتورها خودبخود دارای یک فیدبک وضعیت می باشند. بدین معنی که هرگاه فـرمانی برای چـرخش به یک وضـعیت مشـخص داده شود، با توجه به اصول کار موتور پله ای، روتور تنها می تواند در یک موقعیت که همان وضعیت مطلوب باشد متوقف گردد.

- مزیت دیگر کمی خطای مطلق وضعیت می باشد. بدین معنی که هرگاه هدف، چرخش از یک وضعیت فعلی به یک وضعیت مطلوب باشد، و این کار با طی چندین پله مقدور شود، خطای وضعیت احتمالی، خطای ناشی از آخرین پله خواهد بود، یعنی اینکه روتور در آخرین پله با چه دقتی در موقعیت گسستهٔ مطلوب قرار می گیرد. به بیان دیگر در اینگونه موتورها خطای وضعیت در هر لحظه ناشی از خطای وضعیت قطبهای روتور نسبت به قطبهای استاتور در آن لحظه می باشد. بدین ترتیب خطای موجود صرفاً ناشی از خطای ساختمان مکانیکی موتور می باشد و با دقت در ساخت مکانیکی موتور می توان تا حد ممکن آنرا کمتر نمود.

- در بسیاری از مواقع لازم است تا یک جسم (بار) با منحنی خاصی سرعت گرفته و با دقت زیاد در یک سرعت تثبیت گردد، و سپس در صورت نیاز با شتاب کنترل شده ای مجدداً متوقف گردد. از نمونه های این نیاز می توان به واحد دیسک درایو کامپیوتر و دستگاه های ویدئو اشاره کرد. در چنین کاربرهایی استفاده از موتورهای معمولی اگر هم مقدور باشد باصرفه نخواهد بود، زیرا نیاز به مدارات جانبی بسیار پیچیده و سنسورهای سرعت و شتاب با مدارهای مربوطه دارد.

مزایای قبلی موتورهای پله ای و ماهیت دیجیتالی آنها استفاده از اینگونه موتورها را در کاربردهای فوق عملی و باصرفه نموده است.

ج) موتور هیبرید Hybrid Stepper Motor :

نوع دیگر موتور پله ای که روتورش شامل مغناطیس دائم می باشد، موتور هـیبرید است. مکانیزم عملکرد این موتورها بر اساس ترکیبی از دو نوع VR, PM می باشد.

هسته استاتور شبیه موتور رلوکتانس متغییر است، اما نحوهٔ اتصال سیم پیچها در این موتورها تفاوت دارد. در موتور رلوکتانس متغییر روی هر قطب فقط یک بوبین قرار می گیرد. بوبین فوق یکی از دو بوبین متعلق به یک فاز موتور است. اما در موتور هیبرید هر قطب تنها به یک فاز تعلق ندارد. اینگونه سیم بندی را که روی یک قطب دو بوبین مجزا قرار می گیرد Bipolar نامند. پلاریته حاصل از هر دو سیم پیچ مخالف یکدیگرند. در این نوع موتور روتور شامل دو قسمت است. هسته استوانه ای شکل که یکپارچه و مغناطیسی است، و پوشش دندانه ای شکل که استوانهٔ مغناطیسی را در بر دارد.

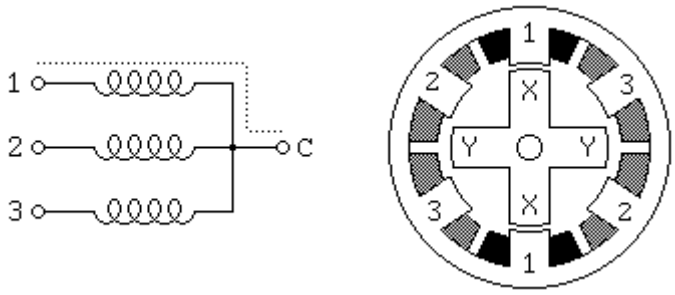
۲-٤) انواع موتورهای پله ای و چگونگی عملکرد آنها
Stepping Motor Types

همانطور که گفته شد، موتورهای پله ای به دودستهٔ ermanent Magnet و Variable Reluctance (همچنین Hybrid که از نقطه نظر کنترل با انواع PM تفاوتی نمی کند) تقسیم می شوند. چنانچه این موتورها دارای برچسب نباشند، می توانید بطور کلی آنها را وقتی که به نیرو متصل نیستند به این شکل جدا کنید که موتورهای مغناطیس دائم در هنگام چرخیدن روتورشان بوسیلهٔ دست، بصورت دندانه ای و پله ای می چرخند، درصورتیکه موتورهای VR تقریباً بطور آزاد می چرخند. همچنین شما می توانید بوسیلهٔ یک اهم متر این دو دسته را از هم جدا کنید. موتورهای VR دارای ۳ (بعضی اوقات٤) سیم پیچ و یک سیم برگشتی مشترک می باشند. درحالیکه موتورهای PM معمولاً دارای ۲ سیم پیچ کاملاً جدا هستند.

موتورهای پله ای در اشکال بسیار متفاوتی وجود دارند. موتورهای خشن و بزرگ نوعاً به ازاء هر پله نود درجه می چرخند، درحالیکه موتورهای پله ای حساستر، معمولاً قادرند ۱/۸ درجه یا حتی ۷۲/. درجه به ازاء هر پله بچرخند. با یک کنترلر مناسب اکثر

موتورهای PM و Hybrid می توانند حرکات نیم پله (نصف یک پله) و با برخی کنترلرهای دیگر پله های کوچکتر از این یا micro step را ایجاد کنند.
برای هر دو نوع موتور PM, VR اگر فقط یکی از سیم پیچهای موتور را انرژی دهیم، روتور (بدون بار) در یک زاویهٔ ثابت قفل خواهد شد، و آنگاه آن زاویه را تا زمانیکه گشتاوری بیشتر از گشتاور نگهدار به آن وارد نشود حفظ خواهد کرد.

- موتورهای با مقاومت مغناطیسی متغییر Variable Reluctance Motors



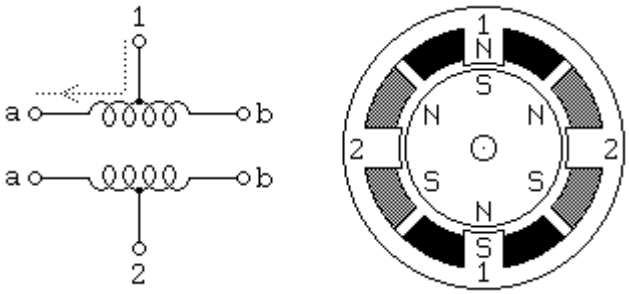
شکل (۲-٤)

اگـر موتور شمـا دارای سه سیم پیچ باشـد، نـوعاً بصورت دیاگرام نمایش داده شده در شکل (۲-٤) همراه با یک خروجی مشترک برای همهٔ سیم پیچها است، که بیشتر شبیه به یک موتور رلوکتانس متغییر است. در زمان استفاده، سیم مشترک معمولاً به پایهٔ مثبت منبع متصل و سر سیم پیچها به ترتیب انرژی دار خواهند شد.
در شکل (۲-٤) فوق یک موتور VR با زاویهٔ پله ۳۰ درجه را مشاهده می کنید. روتور در این موتور دارای چهار دندانه، و استاتور دارای شش پل است، با سیم پیچهایی که هر کدام دور دو پل مخالف پیچیده شده است. با انرژی دار شدن سیم پیچ شمارهٔ ۱ دندانهٔ روتور که با X مشخص شده است، به سمت پلهایی که با این سیم پیچ پیچیده شده اند کشیده می شود. اگر جریان سیم پیچ شمارهٔ ۱ قطع و جریان سیم پیچ شماره ۲ وصل گردد، روتور به مقدار ۳۰ درجه در جهت عقربه های ساعت، به قدری که پلهای مشخص شده با علامت Y روبروی پلهای مشخص شده با شمارهٔ ۲ قرار گیرند، خواهد چرخید.
برای چرخیدن مداوم این موتور فقط باید سه سیم پیچ را به ترتیب انرژی دار کرد. ترتیب کنترلی زیر، موتور فوق را در جهت عقربه های ساعت، و به اندازهٔ ۲٤ دندانه یا دو دور خواهد چرخاند.

Winding 1 10010010010010010010010010
Winding 2 01001001001001001001001001
Winding 3 00100100100100100100100100
time --->

همچنین موتورهای پله ای VR با ٤ و ٥ سیم پیچ نیز وجود دارند که، اصول راه اندازی آنها شبیه به انواع سه سیم پیچ است. موتور تشریح شده در شکل (٢-٤) از تعداد کمی دندانه در روتور و پل در استاتور استفاده کرده است، و پله هایی با ٣٠ درجه ایجاد می کند. استفاده از پلها و دندانه های بیشتر ایجاد پله های کوچکتر را ممکن می سازد.

- موتورهای تک قطبی Unipolar Motors



شکل (٢-٥)

موتورهای تک قطبی، موتورهای پله ای مغناطیس دائم و هیبرید با سیم بندی ٥ سر یا ٦ سر معمولاً بصورت شماتیک شکل (٢-٥) سیم بندی شده اند، که دارای یک سر وسط برای هر سیم پیچ می باشند. در زمان استفاده، سرهای وسط سیم پیچها معمولاً به پایهٔ مثبت منبع و دو سر انتهایی هر سیم پیچ بصورت متناوب به پایهٔ زمین منبع وصل می شود تا جهت میدان ایجاد شده توسط سیم پیچ را معکوس کند.

شکل (٢-٥) موتور مغناطیس دائم و یا هیبرید را نشان می دهدکه می تواند به ازاء هر پله ٣٠ درجه بچرخد (تفاوت این دو دسته موتور در بخش(٢-٣) توضیح داده شده است). سیم پیچ شمارهٔ ١ موتور، مابین دو پل بالا و پائین استاتور تقسیم شده است، در حالیکه سیم پیچ شمارهٔ ٢ ، بین دو پل سمت چپ و راست استاتور توزیع شده است. روتور آن از یک مغناطیس دائم با ٦ پل، سه قطب جنوب و سه قطب شمال، که در پیرامون آن چیده شده ساخته شده است.

برای ایجاد دقت زاویه ای بیشتر، روتور می باید از پلهای بیشتری ساخته شود. موتور پله ای نشان داده شده در تصویر فوق که در هر پله ۳۰ درجه می چرخد، یکی از رایجترین طرحهای موتور پله ای مغناطیس دائم می باشد، گرچه انواع ۱۵ و ۷/۵ درجه ای آن هم بطور وسیع یافت می شود. موتورهای PM با دقت زاویه ای ۱/۸° ساخته شده اند، و موتورهای هیبرید رایج هم با دقت ۳/٦ و ۱/۸ درجه و همچنین ۷۲°/. قابل تهیه هستند.

همانطوریکه در شکل نشان داده شده، عبور جریان از سر وسط سیم پیچ ۱ به ترمینال a موجب ایجاد قطب شمال در پل بالای استاتور خواهد شد درحالیکه پل پائین استاتور قطب جنوب شده است، که این حالت موجب جذب شدن روتور به مکان نشان داده شده در شکل می گردد. چنانچه نیروی سیم پیچ ۱ قطع و سیم پیچ ۲ متصل گردد، روتور ۳۰ درجه (یک پله) خواهد چرخید. برای چرخش مداوم روتور، ما فقط باید نیرو را بین دو سیم پیچ به ترتیب تقسیم کنیم. اگر ۱ را منطق مثبت فرض کنیم، که نشانگر عبور جریان از یک سیم پیچ باشد، دو ترتیب کنترل زیر موجب چرخش موتور به اندازهٔ ۲٤ پله در جهت عقربه های ساعت، یا دو دور کامل خواهد شد.

Winding 1a 10001000100010001000100010001
Winding 1b 00100010001000100010001000100
Winding 2a 01000100010001000100010001000
Winding 2b 00010001000100010001000100010
time --->

Winding 1a 11001100110011001100110011001
Winding 1b 00110011001100110011001100110
Winding 2a 01100110011001100110011001100
Winding 2b 10011001100110011001100110011
time --->

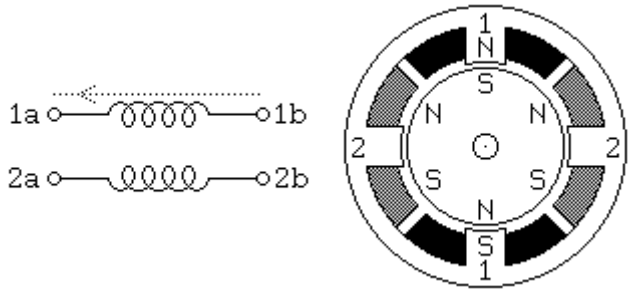
توجه داشته باشید که دو نیمهٔ سیم پیچ، هیچگاه بطور همزمان فعال نخواهد شد. هر دو ترتیب نمایش دادهٔ فوق در هر نوبت مغناطیس دائم را یک پله خواهد چرخاند. ترتیب بالایی فقط یک سیم پیچ را بطوریکه در شکل بالا نشان داده شده فعال می کند، بدین معنی که کمترین انرژی را مصرف می کند. و ترتیب زیرین، دو سیم پیچ را در هر نوبت فعال خواهد کرد، و بطور کلی، گشتاوری معادل ۱/٤ برابر بزرگتر از ترتیب بالایی، با استفاده از دو سیم پیچ ایجاد خواهد کرد.

مکانهای پله تولید شده توسط دو ترتیب فوق یکسان نخواهد بود، بعنوان یک نتیجه می توان گفت که ترکیب دو ترتیب بالا نیم پله را ایجاد می نماید، که با متوقف کردن روتور،

بصورت یک در میان در مکانهای قابل دستیابی توسط یکی از دو ترتیب بالا ممکن خواهد شد.

```
Winding 1a 11000001110000011100000111
Winding 1b 00011100000111000001110000
Winding 2a 01110000011100000111000001
Winding 2b 00000111000001110000011100
time --->
```

- موتورهای دو قطبی Bipolar Motors



شکل (۲-٦)

موتورهای دو قطبی مغناطیس دائم و موتورهای هیبرید در واقع با مکانیزمی مشابه آنچه که در موتورهای تک قطبی استفاده شده ساخته می شوند، اما دو سیم پیچ آنها بصورت ساده تری سیم بندی شده اند، که بدون سر وسط می باشند. به این معنی که خود موتور ساده تر شده است ولی مدار درایور آن جهت تعویض هر جفت پل موتور، کمی پیچیده تر شده است. شکل (۲-٦) نشان می دهد که یک چنین موتوری چگونه سیم بندی شده است. بطوریکه می بینید درواقع بسیار شبیه به شکل (۲-۵) می باشد. مدارات درایور چنین موتوری نیاز به یک مدار کنترل H-bridge برای هر سیم پیچ دارد. که این موضوع در مبحث مدارات کنترل موتورهای پله ای، بیشتر توضیح داده خواهد شد. خلاصه اینکه یک H-bridge اجازه میدهد که پلاریته جریان هر یک از سرهای سیم پیچ ها، جداگانه تعیین شود. ترتیب کنترل برای چنین موتوری در زیر نشان داده شده است. استفاده از + و – نمایانگر پلاریـته ترمینالهای هر سیم پیچ می باشد.

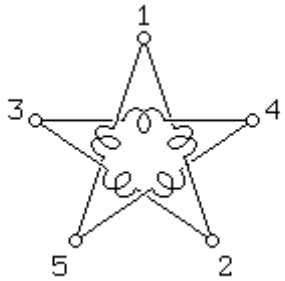
```
Terminal 1a +---+---+---+---  ++--++--++--++--
Terminal 1b --+---+---+---+-  --++--++--++--++
Terminal 2a -+---+---+---+--  -++--++--++--++-
Terminal 2b ---+---+---+---+  +--++--++--++--+
time --->
```

توجه داشته باشید که این ترتیبات به شکلی انتزاعی عیناً منطبق با آنهایی است که برای یک موتور مغناطیس دائم تک قطبی بکار برده می شود. همچنین توجه داشته باشید که بسیاری از تراشه های درایور full H-bridge ، یک ورودی برای فعال نمودن خروجی و یکی دیگر برای کنترل جهت دارند. چنین تراشه ای برای اینکه ترتیبات فوق را بوجود بیاورد بصورت زیر عمل می کند.

```
Enable    1 1010101010101010  1111111111111111
Direction 1 1x0x1x0x1x0x1x0x  1100110011001100
Enable    2 0101010101010101  1111111111111111
Direction 2 x1x0x1x0x1x0x1x0  0110011001100110
time --->
```

برای تمیز دادن یک موتور دو قطبی از سایر موتورهای چهار سیم، مقاومت بین ترمینالهای مختلف را بسنجید. مهم است بدانید که موتور پله ای مغناطیس دائم، با چهار سیم پیچ مجزا نیز وجود دارد، و از دو مجموعهٔ دوتایی تشکیل شده است. اگر دو سیم پیچ در هر مجموعه بصورت سری متصل باشند، می توان از آن موتور در ولتاژهای بالا استفاده کرد. اگر به شکل موازی بسته شده باشند، می توان از آنها در ولتاژهای پائین استفاده کرد. و اگر آنها بصورت سری و دارای سر وسط باشند، می توانند بعنوان موتورهای ولتاژ پائین تک قطبی مورد استفاده قرار گیرند.

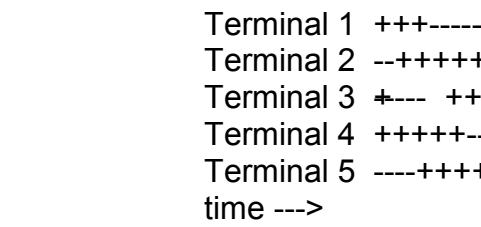
– موتورهای چند فاز Multiphase Motors



شکل ۲–۷)

رده ای از موتورهای پله ای مغناطیس دائم که کمتر رایج است به شکلی است که همهٔ سیم پیچهای آن، بصورت چرخه ای، سری می باشند، که سر وسطی مابین هر جفت سیم پیچ از حلقه وجود دارد. رایج ترین طراحی این دسته از موتورها، انواع ۳ و ۵ فاز آن می باشد. کنترل آن نیازمند نصف H-bridge برای هر ترمینال موتور می باشد. این موتورها گشتاور بیشتری ایجاد می کنند، چون همه، یا همه بجز یکی از سیم پیچها،

در همهٔ نوبتها فعال هستند. بعضی از موتورهای ۵ فاز دقت بالایی به اندازهٔ ۷۲/. درجه (۵۰۰ پله به ازاء یک دور کامل) ایجاد می کنند. با یک موتور ۵ فاز، ده پله در هر چرخهٔ تکراری به صورت زیر می تواند ایجاد شود.



در اینجا، به همان صورتی که در مورد دو قطبی ها دیدیم، هر ترمینال نشان داده شده، در سیستم نیروی موتور به پایهٔ + یا – متصل خواهد شد. توجه داشته باشید که در هر پله فقط یک ترمینال پلاریته خود را عوض می کند. این تعویض، نیرو را از یک سیم پیچ متصل به ترمینال قطع می کند (چون دو ترمینال دو سر این سیم پیچ، دارای یک پلاریته خواهد شد) و نیرو به سیم پیچی که در مرحلهٔ قبل بیکار بود منتقل خواهد شد. موتور تشریح شده در شکل (۲–۷) با ترتیب داده شده، دو دوران کامل خواهد داشت.

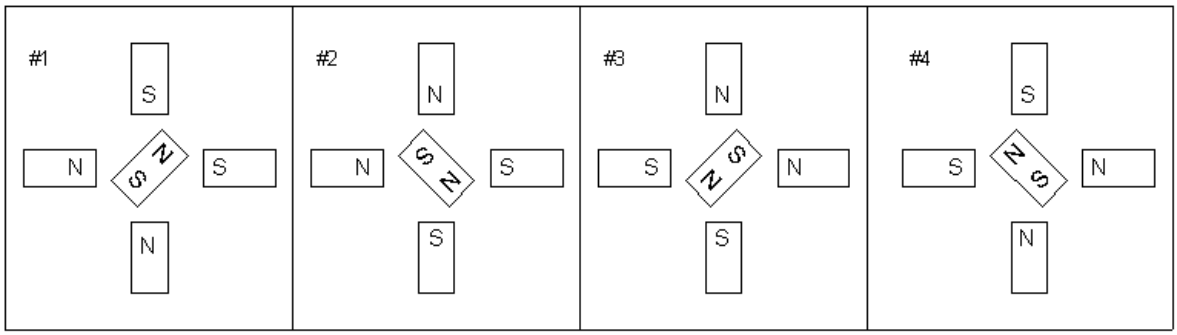
برای متمایز کردن موتورهای ۵ فاز از بقیه موتورهای ۵ سیم، توجه کنید به اینکه اگر مقاومت بین دو ترمینال متوالی یک موتور ۵ فاز، R باشد، مقاومت بین ترمینالهای غیر متوالی 1.5R خواهد بود.

توجه کنید که بعضی از موتورهای ۵ فاز دارای ۵ سیم پیچ جداگانه هستند، که کلاً ده سر خواهد داشت. که اینها بصورت ستاره ای، مانند شکل بالا، قابل اتصال هستند، و با مدار درایور 5 half-bridge ، و یا هر سیم پیچ با یک full-bridge اختصاصی راه اندازی می شوند. با توجه به اینکه از نظر تئوری درایور half-bridge کندتر است، استفاده از یک تراشهٔ full-bridge احتمالاً دارای برتری خواهد بود.

۲–۵) ترتیب فازهای موتور پله ای
Stepper Motor Phase Sequencing

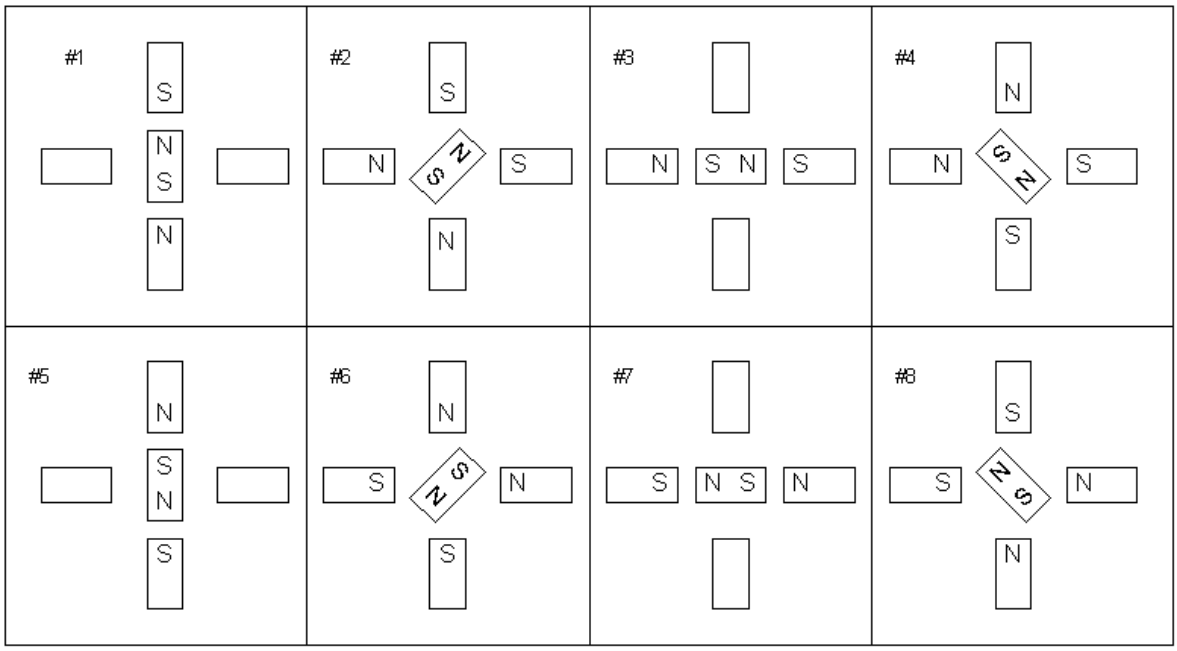
شکل های بعدی چگونگی حرکت پله ای full step و half step را در موتور پله ای یک قطبی و دو قطبی را نشان میدهد.

Bipolar Full Step :



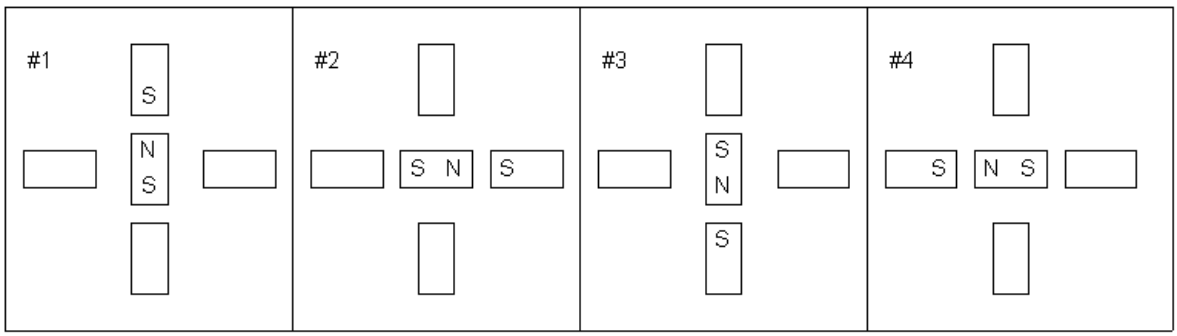
شکل (۲-۸)

Bipolar Half Step :



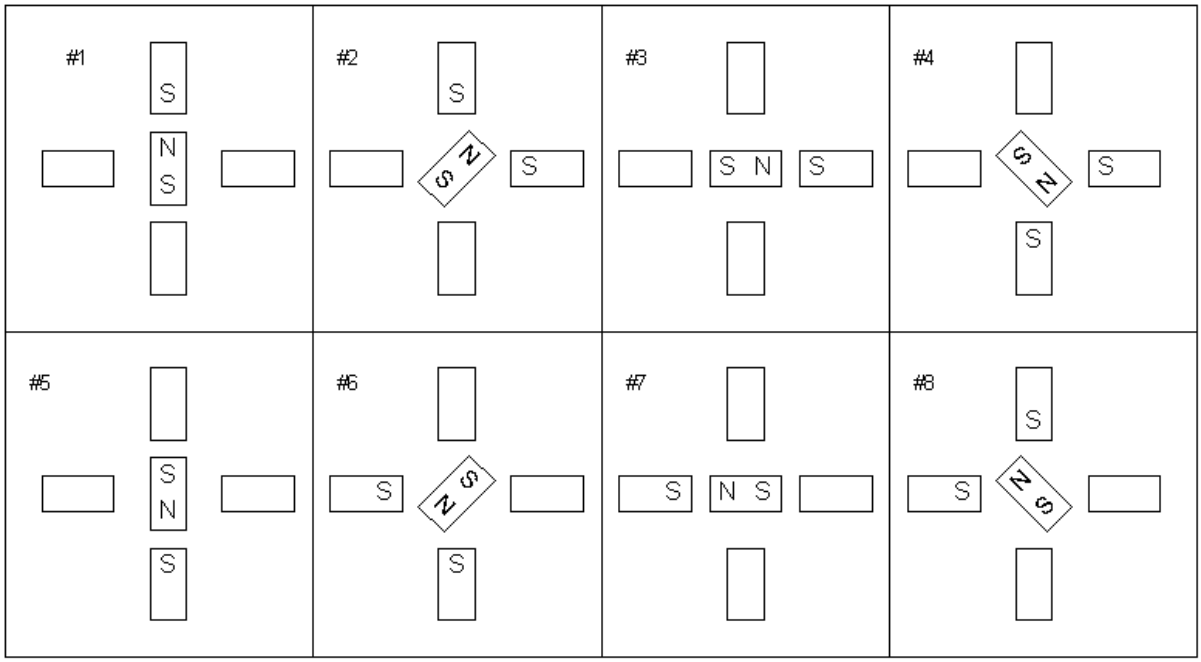
شکل (۲-۹)

Unipolar Full Step :



شكل (٢-١٠)

Unipolar Half Step :



شكل (٢-١١)

Stepping Sequences for a Four-Phase Unipolar Permanent Magnet Stepper Motor

ترتیب پله ها در یک موتور چهارفاز تک قطبی با هستهٔ آهنربای دائمی

این موتورها چهار سیم پیچ دارند که وقتی در جهت صحیح انرژی داده شوند، موجب حرکت آهنربای دائمی که بر محور آنها قرار دارد، خواهند شد. دو ترتیب پلهٔ مبنا وجود دارد و بعد از پله ٤ مجدداً حرکت از پلهٔ ١ تکرار می شود. معکوس کردن روند تغذیه سیم پیچ ها، موجب عکس شدن جهت چرخش خواهد شد.

a. Single-Coil Excitation - Each successive coil is energised in turn.

Step	Coil 4	Coil3	Coil2	Coil1	
a.1	on	off	off	off	
a.2	off	on	off	off	
a.3	off	off	on	off	
a.4	off	off	off	on	

ترتیب صفحهٔ قبل از حداقل انرژی استفاده می کرد، لذا گشتاور کمتری تولید می کرد. ترتیب زیر که دو سیم پیچ را در هر نوبت تغذیه می کند، انرژی بیشتری مصرف می کند و گشتاور بیشتری نیز ایجاد می کند.

b. Two-Coil Excitation - Each successive pair of adjacent coils is energised in turn.

Step	Coil 4	Coil3	Coil2	Coil1	
b.1	on	on	off	off	
b.2	off	on	on	off	
b.3	off	off	on	on	
b.4	on	off	off	on	

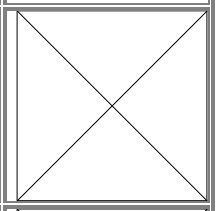
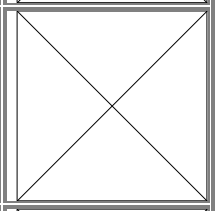
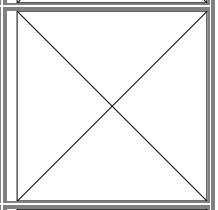
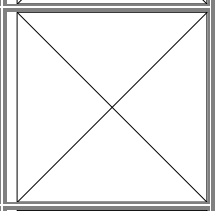
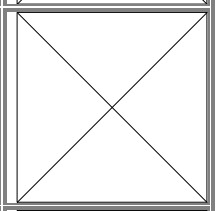
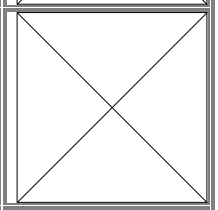
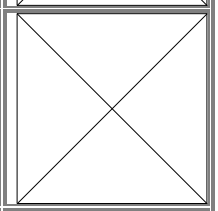
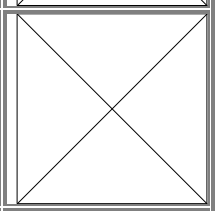
توجه :

نوع تحریک سیم پیچ ٤ همیشه برعکس سیم پیچ ٢ است.

نوع تحریک سیم پیچ ٣ همیشه برعکس سیم پیچ ١ است.

بنابراین با استفاده از یک مدار کنترل مناسب می توانید، برای تولید ترتیب حرکت موتور، از دو خط داده استفاده کنید.

ادغام دو ترتیب گذشته، موجب تولید حرکات نیم پله خواهد شد.

Step	Coil 4	Coil3	Coil2	Coil1	
a.1	on	off	off	off	
b.1	on	on	off	off	
a.2	off	on	off	off	
b.2	off	on	on	off	
a.3	off	off	on	off	
b.3	off	off	on	on	
a.4	off	off	off	on	
b.4	on	off	off	on	

۲-٦) پارامترها و اصطلاحات موتور پله ای

- رابطهٔ بین تعداد دندانه، تعداد فاز و تعداد پله در یک دور

رابطهٔ عمومی بین تعداد فاز (m) ، تعداد دندانهٔ روتور (Nr) و تعداد پله در یک دور چرخش (S)، بصورت زیر می باشد.

$$S= mNr$$

این رابطه برای تحریک یک فاز و دوفاز صحیح است، اما برای حالتی که تحریک بصورت نیم پله باشد، رابطه زیر مصداق خواهد داشت.

$$S= 2mNr$$

فرض کنید سیم پیچهای یک موتور m فازه به ترتیب و بطور متوالی تحریک شود. یعنی با ترتیب خاصی شروع به تحریک فازهای موتور کنیم. هرگاه این عمل بطور متوالی صورت گیرد تا دوباره به فاز اول برسیم، در اینصورت تعداد m پالس ایجاد شده است.در یک سیکل کامل تحریک روتور به اندازهٔ یک دندانه دوران خواهد کرد. بنابراین برای چرخش موتور به اندازه عرض یک دندانه و شیار روتور، تعداد m پالس ایجاد می شود، پس برای یک دور چرخش تعداد mNr پالس نیاز می باشد. این مطلب برای موتورهای چند طبقه نیز صادق است.

ـ زاویه پله

روتور یک موتور پله ای به ازاء هر پالس ورودی به اندازهٔ مشخص دوران می کند. همانطور که قبلاً اشاره شد مقدار این دوران را زاویهٔ پله می گوئیم که برحسب درجه می باشد. رابطهٔ بین زاویهٔ پله (Os) و تعداد پله (S) بصورت زیر می باشد.

$$S= 360 / Os$$

موتورهای بکار رفته در پرینترها و ماشینهای تایپ، معمولاً دارای ۹٦، ۱۲۸ یا ۱۳۲ پله در یک دور می باشند. تعداد پله در یک موتور چهارفاز معمولی ۲۰۰ است، و موتورهای دیگری با تعداد پله ۵۰۰ یا ۱۰۰۰ طراحی و ساخته شده اند.

ـ تاثیر افزایش دقت در موقعیت روتور

دقت در موقعیت مکانی روتور یک فاکتور مهم برای تعیین کیفیت موتور پله ای می باشد. این موتورها طوری طراحی می شوند که با هر پالس به مقدار زاویهٔ از پیش تعیین شده، چرخیده و بعد از رسیدن به موقعیت صحیح توقف کنند. میزان دقت در تعیین مکان محور به تکنولوژی ساخت و ساختمان روتور و استاتور بستگی دارد. به همین خاطر این اجزاء با دقت زیادی ساخته می شوند. در ضمن ساختمان این موتور طوری طراحی می شود که در شروع حرکت به همراه بار، گشتاور زیادی را تولید کند. بنابراین تعیین موقعیت صحیح روتور تنها به خصوصیات ماشین و مدار راه انداز بستگی دارد، و پارامترهای دیگر روی آن اثری ندارند.

ـ ماکزیمم گشتاور استاتیکی

الف) گشتاور نگهدارنده: ماکزیمم گشتاور استاتیکی که به محور یک موتور تحریک شده اعمال می شود تا در یک موقعیت مشخص توقف کند.

ب) گشتاور نگهدار : ماکزیمم گشتاور استاتیکی که به محور یک موتور تحریک نشده اعمال می شود تا در یک موقعیت مشخص توقف کند.

عموماً هر چه گشتاور نگهدارنده بیشتر باشد، موقعیت مکانی روتور دارای خطای کمتری خواهد بود. گشتاور نگهدار مربوط به مغناطیس دائم موجود در ساختمان موتورها می باشد، و در موتورهایی که از مغناطیس دائم استفاده نشده این گشتاور وجود ندارد.

ـ مکانهای توقف روتور

الف) محل توقف روتور با موقعیت تعادل : موقعیت مکانی یک موتور تحریک شده را در حالت بی باری گویند.

ب) موقعیت نگهدار : موقعیتی است که یک موتور تحریک نشده به علت وجود مغناطیس دائم در حالت بی باری پیدا می کند. موقعیت نگهدار و تعادل اغلب یکسان نیستند.

ـ میزان دقت در تعیین مکان روتور

الف) خطا در موقعیت پله : بیشترین خطای موجود در مقدار زاویه نسبت به زاویهٔ مطلوب، زمانیکه روتور از یک موقعیت مکانی به موقعیت بعد حرکت می کند.

ب) دقت در تعیین مکان روتور : مکانهایی که روتور باید در آنجا توقف کند بصورت ضریبی از میزان زاویهٔ پله مشخص می گردند. در عمل روتور بطور دقیق در این مکانها قرار نمی گیرد، و نسبت به آنها مقداری انحراف خواهد داشت، این خطا ممکن است در جهت مثبت یا منفی صورت گیرد. اختلاف بین حداکثر و حداقل خطا، تعیین کنندهٔ میزان دقت در تعیین مکان روتور می باشد. برای توضیح بیشتر جداول (۲–الف) و (۲– ب) مربوط به یک موتور رلوکتانس متغییر با زاویهٔ پله ۱۵° می باشد را در نظر می گیریم.

جدول (۲– الف) مربوط به میزان دقت در تعیین مکان و جدول (۲– ب) مربوط به میزان خطا در موقعیت مکانی پله می باشد. همانطور که مشاهده می شود در جدول اول، میزان خطا در یک دور برای مکانهای مختلف از ۰/۰۳°– تا حداکثر ۰/۰۸° تغییر می کند، که با توجه به تعریف میزان دقت مکان برابر ۰/۱۱° =۰/۰۳° + ۰/۰۸° خواهد بود. این خطا معمولاً بصورت ۰/۰۵۵°– نشان داده می شود، که مشخص کنندهٔ محدودهٔ خطا از ۰/۰۵۵°– تا ۰/۰۵۵°+ می باشد.

جدول (۲– الف): داده ها براساس اندازه گیری دقت مکان موتور

Number of steps	Theoretical angle	Measured angle	Error
0	0	0	0
1	15.00	15.06	+ 0.06
2	30.00	29.97	- 0.03
3	45.00	45.07	+ 0.07
4	60.00	60.00	0
5	75.00	75.06	+ 0.06
6	90.00	89.97	- 0.03
7	105.00	105.07	+ 0.07
8	120.00	120.01	+ 0.01
9	135.00	135.05	+ 0.05
10	150.00	149.97	- 0.03
11	165.00	165.07	+ 0.07
12	180.00	180.01	+ 0.01
13	196.00	195.05	+ 0.05
14	210.00	209.97	- 0.03
15	225.00	225.08	+ 0.08
16	240.00	240.00	0
17	255.00	255.05	+ 0.05
18	270.00	269.97	- 0.03
19	285.00	285.08	+ 0.08
20	300.00	300.00	0
21	315.00	315.05	+ 0.05
22	330.00	329.97	- 0.03
23	345.00	345.07	+ 0.07
24	360.00	360.00	0

جدول (۲– ب) : داده ها براساس میزان خطای موقعیت مکانی پله

Number of steps	Measured step angle	Error
0	0	0
1	15.06	+ 0.06
2	14.941	- 0.09
3	15.10	+ 0.10
4	14.93	- 0.07
5	15.06	+ 0.06
6	14.91	- 0.09
7	15.10	+ 0.10
8	14.49	- 0.06
9	15.04	+ 0.04
10	14.92	- 0.08
11	15.10	+ 0.10
12	14.94	- 0.06
13	15.04	+ 0.04
14	14.92	- 0.08
15	15.11	+ 0.11
16	14.92	- 0.08
17	15.05	+ 0.05
18	14.92	- 0.08
19	15.11	+ 0.11
20	14.92	- 0.08
21	15.05	+ 0.05
22	14.92	- 0.08
23	15.11	+ 0.11
24	14.93	- 0.07

خطا در موقعیت پله برای این موتور برابر $^{\circ}$۰/۱۱ می باشد. باید دقت داشت که در این مثال بصورت اتفاقی دو خطای فوق با یکدیگر برابر شده اند، ولی در عمل معمولاً میزان دقت در تعیین موقعیت، بیشتر است.

- نسبت گشتاور به اینرسی

موتور پله ای باید با تندترین سرعت ممکن به پالسهای ورودی پاسخ دهد. در این موتور نه تنها یک شروع حرکت سریع، بلکه یک توقف سریع نیز لازم است. اگر در حالتی که موتور با یک سرعت ثابت دوران می کند، وقفه ای در پالسهای فرمان ایجاد شود، روتور با توجه به آخرین پالس ورودی در یک موقعیت معین می ایستد. مطلب فوق نشان می دهد که نسبت گشتاور به اینرسی در موتور پله ای بیشتر از موتورهای الکتریکی دیگر است.

- سرعت و فرکانس پالس ورودی Pulse Per Second

سرعت دوران موتور پله ای بر حسب تعداد پله های موتور در یک ثانیه مشخص می گردد. هر موتور پله ای سرعت و فرکانس پالس ورودی محدودی دارد، که این موضوع با افت گشتاور (torque) متناسب است. یعنی افزایش PPS باعث کم شدن خروجی می گردد.

$$n= 60f / S$$

-n سرعت چرخش (RPM)

-f میزان پله که از فرکانس پالس ورودی تبعیت می کند

-s تعداد پله در یک دور کامل روتور

ـ در انواع موتورهای پله ای، اکثراً یک روتور با پوشش فلزی، نسبت به تغییر میدان مغناطیسی ایجاد شده در الکترو مگنت اساتور، توسط مدار کنترل واکنش نشان می دهد. این مدارات کنترل باید پالس مورد نیاز فازهای استاتور را با ترتیب صحیح برای صحت

راه اندازی، و قرار دادن آن در موقعیت مناسب تا زمانیکه مدار کنترل فرمان بدهد، تولید کنند. پارامترهای زیر مربوط به موتوری می شوند که می تواند نیازهای خواسته شده را برآورده کند (که بعضی از آنها قبلاً توضیح داده شده است).

- dynamic torque—the torque developed by the motor when in motion; dependent on the current in stator electromagnets;
- phase inductance—the electrical parameter that limits phase-current risetime and, hence, dynamic torque;
- holding torque—the torque a motor can develop to prevent a static load from pulling the system out of step;
- torque stiffness—the motor's ability to resist angular displacements within a step; and
- rotor inertia—the mechanical parameter that limits motor acceleration and deceleration.

اغلب مشکلات مربوط به موتورهای پله ای مربوط به وقتی است که یک مشتری نزد فروشندهٔ موتور پله ای می رود و با اطلاعات نارسا یا غلط مواجه می شود. برای مثال ممکن است یک طراح، به یک گشتاور نگهدار خاصی نیاز داشته باشد، که آنچه تهیه می کند، شاید فاقد گشتاور دینامیکی مطلوب باشد. وقتی این موتور برای هدف درنظر گرفته شده خوب جواب ندهد، طراح، موتوری با گشتاور نگهدار(holding torque) بالاتر را درنظر می گیرد.

آن موتور هم به خوبی عمل نخواهد کرد، چون گشتاور نگهدار بزرگتر، الزاماً به معنی گشتاور دینامیکی بزرگتر نیست. در مجموع باید گفت که یک موتور با گشتاور نگهدار بالاتر، همیشه سکون بیشتری برای روتور ایجاد می کند، که مانع شتاب گیری میشود و بازده را کاهش می دهد. با وجود بسیاری از متغییرها، شما می توانید از راهبردهای سریعی برای تعیین اینکه چه موتورهایی میتوانند در زمینهٔ کاربرد مورد نیاز شما به خوبی جواب دهند، بهره ببرید.

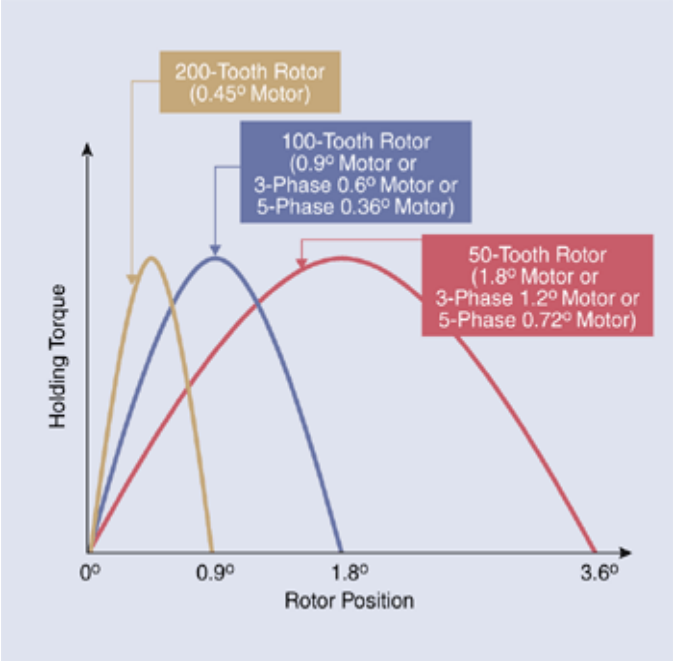
صحت پله ها به ساختمان موتور و به اندازهٔ گشتاور موتور بستگی دارد. رابطه بین گشتاور نگهدار، T ، و تغییر مکان زاویه ای ، θ ، تقریباً برابر است با :

$$T = T_0 \bullet \sin(N\,\theta)$$

که N تعداد دندانه های روتور و T_0 ماکزیمم گشتاور استاتیکی است. اندازهٔ گشتاور برابر است با :

$$dT/d\theta = N \bullet T_0 \bullet \cos(N\theta)$$

که dT تغییر گشتاور و dθ تغییر زاویه ای است. این تساوی ها نشان می دهد که افزایش N یا T_0 تنها راه های بهبود اندازه گشتاور هستند. ایجاد پله های الکتریکی اضافی (micro stepping) بین پله های مکانیکی ابتدایی (متناظر با دندانه های روتور) یاافزایش تعداد فاز (برای مثال بکارگیری ۳ فاز یا ۵ فاز) اندازه گشتاور را بهبود نخواهد داد. به شکل (۲-۱۲) توجه کنید.

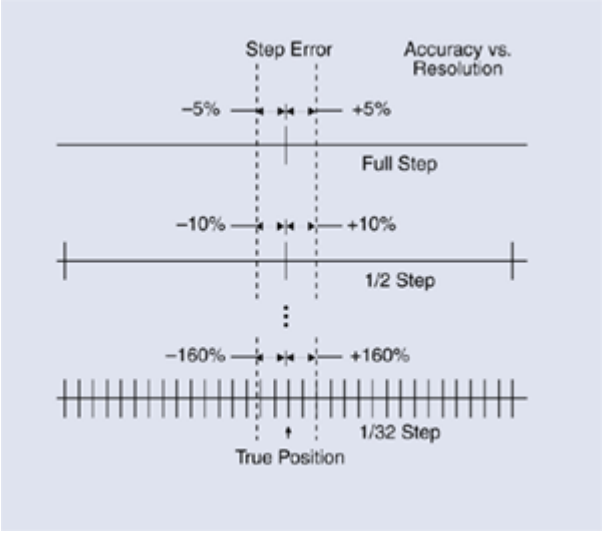


شکل (۲-۱۲)

Torque stiffness depends on a stepper motor's number of rotor teeth and on its maximum holding torque.

پارامترهای طراحی مکانیکی، روشهای تولید، و تولرانس ها در عملکرد صحیح یک موتور پله ای دخالت دارند. درستی پله، بطور مثال، به درستی محل قرارگیری دندانه های استاتور، یکسان بودن توزیع دندانه های روتور و یکسانی فواصل هوایی مابین دندانه های روتور و استاتور وابسته است. سازندگان می توانند به دقت پله ±5% دست یابند. ±5% خطا در یک موتور با زاویه پله ۱/۸°، ٦٥,٤ arc minutes است.

این خطای مطلق در زمان تولید micro stepping نیز باقی می ماند. Micro stepping تعداد پله ها را افزایش می دهد نه دقت آنها را. لذا با micro stepping یک خطای مطلق ±5%، تبـدیل خواهـد شد به ±10% در نیم پله، یا خطای ±160% برای ۳۲ پله. به شکل (۲-۱۳) توجه کنید. در سطوح خطای پله بالای ±100% شما نمی توانید تضمین کنید که موتور به ازاء هر پالسی که دریافت می کند یک پله حرکت خواهد کرد.



شکل (۲-۱۳)

Microstepping will improve stepper-motor resolution but not accuracy. A ±5% absolute step error becomes a ±160% error at 32nd-step microstepping.

برای کسب اطلاعات بیشتر در خصوص ارتباط لحظهٔ خیزجریان (Current Risetime) و گشتاور، همچنین بعضی از پارامترهای مفید دیگر به بخش ضمیمه ب (مربوط به موتورهای پله ای) مراجعه کنید.

۲-۷) مدارات کنترل موتور پله ای

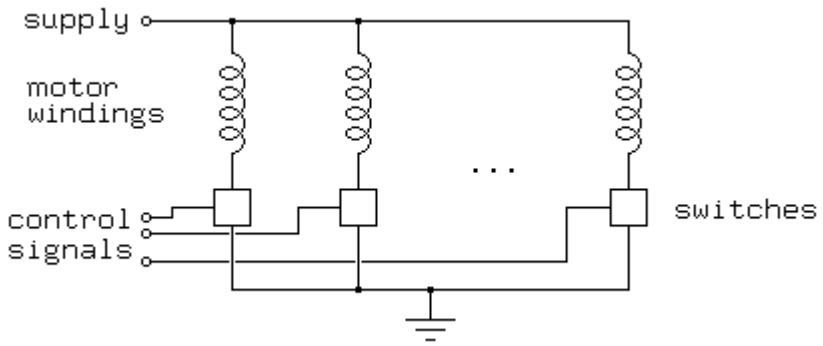
Βασιχ Στεππινγ Μοτορ Χοντρολ Χιρχυιτσ

این بخش از نوشتار، پیرامون مدارات راه انداز موتورهای پله ای بحث خواهد کرد. این مدارت روی موضوع قطع و وصل جریان در هر سیم پیچ موتور و کنترل جهت آنها متمرکز هستند. مدارات بحث شده در این بخش مستقیماً به موتور و منبع تغذیه متصل می شوند، و این مدارات توسط یک سیستم دیجیتال که وظیفه اش روشن و خاموش کردن سوئیچ هاست، کنترل می شوند.

این بخش تمامی انواع موتورها را پوشش می دهد، از مدارات ابتدائی برای کنترل یک موتور رلوکتانس متغییر گرفته، تا مدارات H-bridge برای کنترل یک موتور دو قطبی مغناطیس دائم. هر دسته از مدارات راه انداز با مثالهای کاربردی توضیح داده شده است. ولی این مثالها جهت آموزش و درک این مبحث است و نه برابر با نمونه های تجاری که در بازار یافت می شوند. این مبحث فقط مدارات کنترل مقدماتی مربوط به هر دسته را پوشش می دهد.

- موتورهای رلوکتانس متغییر ***Variable Reluctance Motors***

کنترلرهای مرسوم مربوط به موتورهای رلوکتانس متغییر به خروجی های نشان داده شده در شکل زیر وابسته هستند.

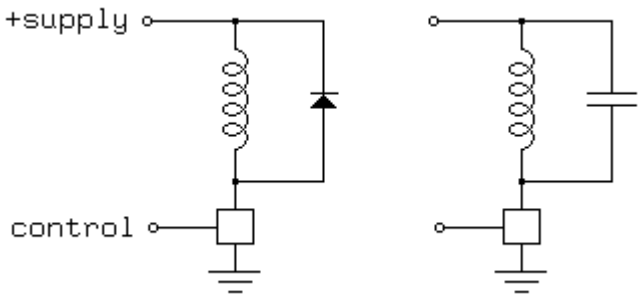


شکل (۲-۷-۱)

در شکل فوق جعبه ها نمایانگر وجود سوئیچ هاست. یک قسمت کنترل که نشان داده نشده، عهده دار تولید سیگنالهای کنترل، برای باز کردن و بستن سوئیچها در زمان مقتضی است، تا به موتور فرمان چرخش را بدهد. در بسیاری از موارد، بخش کنترل یک کامپیوتر یا کنترلر واسط برنامه پذیر می باشد که بوسیلهٔ یک نرم افزار مستقیماً خروجی های مورد نیاز برای کنترل سوئیچها را تولید می کند.

سیم پیچ های موتور، اعم از سولئونویدها و مشابه آنها دارای بار القایی هستند. بطوریکه جریان عبوری از سیم پیچ موتور، بدون آنکه ولتاژ بالای آن مهار شود نمی تواند بصورت آنی قطع و وصل شود. وقتی یک سوئیچ کنترل سیم پیچ بسته باشد، به جریان اجازهٔ حرکت می دهد، که نتیجه اش بالا رفتن آرام سطح جریان می باشد. و وقتی سوئیچ کنترل سیم پیچ موتور باز شود، نتیجه اش ایجاد یک ولتاژ بالا است که می تواند موجب بروز خسارت جدی به سوئیچ شود، مگر اینکه آن ولتاژ بصورت مقتضی مهار گردد.

دو روش اساسی برای مقابله با این ولتاژ بالا وجود دارد، یکی ایجاد پل در سیم پیچ موتور با یک دیود، و دیگری ایجاد یک پل در سیم پیچ موتور با یک خازن می باشد. شکل زیر هر دو طریق را نشان می دهد.



شکل (۲-۷-۲)

دیود نشان داده شده در شکل فوق باید کل جریان را از طریق سیم پیچ موتور هدایت کند. این عمل فقط در زمان خاموش شدن سوئیچ، تا زمانیکه جریان تنزل کرده و قطع شود، صورت می گیرد. اگر از دیود نسبتاً کندی مثل 1N400X های معمولی به همراه سوئیچ های سریع استفاده شود، ممکن است لازم باشد که یک خازن بصورت موازی با دیود متصل گردد.

دیود نشان داده شده در شکل فوق پیچیدگی بیشتری در مشکلات طراحی بوجود می آورد. وقتی سوئیچ بسته می شود، خازن از طریق سوئیچ به زمین متصل و دشارژ خواهد

شد، و سوئیچ باید توانایی هدایت این ولتاژ مربوط به دشارژ جریان را داشته باشد. یک مقاومت سری شده با خازن یا سری شده با منبع تغذیه این جریان را محدود خواهد کرد. وقتی سوئیچ باز شود، انرژی ذخیره شده در سیم پیچ موتور، خازن را به مقداری بالا تر از ولتاژ منبع تغذیه شارژ خواهد کرد، و سوئیچ باید توانایی تحمل این ولتاژ را داشته باشد.

برای محاسبهٔ اندازهٔ خازن، ما دو فرمول مربوط به انرژی ذخیره شده در مدار تشدید کننده را مساوی قرار می دهیم.

$$P = C V^2 / 2$$
$$P = L I^2 / 2$$

Where:

P -- stored energy, in watt seconds or coulomb volts
C -- capacity, in farads
V -- voltage across capacitor
L -- inductance of motor winding, in henrys
I -- current through motor winding

محاسبهٔ حداقل اندازهٔ خازن برای محافظت از ایجاد ولتاژ بالا روی سوئیچ، نسبتاً آسان است:

$$C > L I^2 / (V_b - V_s)^2$$

Where:

V_b -- the breakdown voltage of the switch
V_s -- the supply voltage

موتورهای رلوکتانس متغییر، ضریب القایی متفاوتی دارند که به زاویهٔ محور آنها بستگی دارد. اگر سیستم کنترل، موتور را در فرکانس هایی نزدیک به فرکانس تشدید راه اندازی کند، جریان موتور به واسطه سیم پیچ موتور، و گشتاور ایجاد شده توسط موتور، با گشتاور یکنواخت (steady state) در یک ولتاژ عملیاتی اسمی، کاملاً متفاوت خواهد بود. فرکانس تشدید به صورت زیر است.

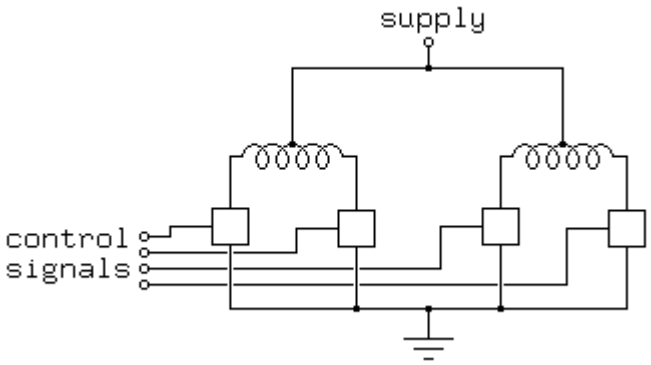
$$f = 1 / (2\pi (L C)^{0.5})$$

از طرف دیگر فرکانس تشدید الکتریکي براي يك موتور رلوکتانس متغییر، وابسته به زاوية محور مي باشد! وقتي يك موتور رلوکتانس متغییر، در حال کار با پالس تحریکي نزديك به فرکانس تشديد مي باشد، جريان نوساني در سيم پيچ موتور، ميداني مغناطيسي ايجاد مي کند که مي تواند به شدت گشتاور قابل وصول را کاهش دهد.

- موتورهای مغناطیس دائم تک قطبی و موتورهای هیبرید

Unipolar Permanent Magnet and Hybrid Motors

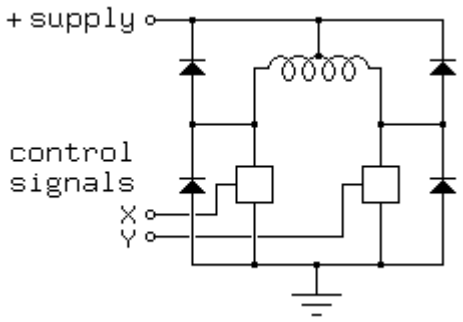
نوعاً کنترلرهای موتورهای مغناطیس دائم تک قطبی وابسته به خروجی های نشان داده شده در شکل (۲-۷-۳) هستند.



شکل (۲-۷-۳)

در شکل (۲-۷-۳) مانند شکل (۲-۷-۱) جعبه هایی که نمایانگر سوئیچ ها هستند، بکار برده شده اند. و یک بخش کنترل که نشان داده نشده، برای تولید سیگنالهای کنترل باز و بسته کردن سوئیچ ها درنظر گرفته شده است، که در زما مقتضی عمل می کند تا موتور به چرخش درآید. معمولاً یک کامپیوتر یا کنترلـر واسط برنامه پذیـر بوسیلهٔ یک نرم افزار مستقیماً خروجی های مورد نیاز برای کنترل سوئیچها را تولید می کند.

نظر به اینکه در مدار راه انداز موتور رلوکتانس متغییر، ما با القاء بازگشتی زمان خاموش شدن سوئیچ مواجه بودیم، مجدداً باید القاء بازگشتی را توسط دیود هایی از بین ببریم، ولی این دفعه ٤ دیود بطوریکه در شکل (۲-۷-٤) نشان داده شده مورد نیاز است.



شکل (۲-۷-٤)

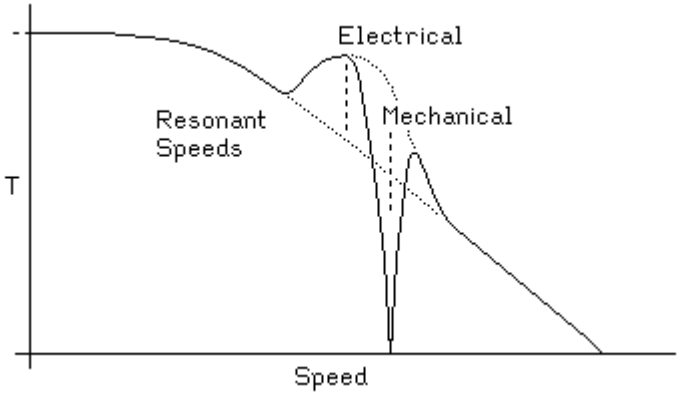
در اینجا دیودهای بیشتری مورد نیاز است، چون سیم پیچ موتور از دو ایناکتور مجزا تشکیل نشده، بلکه از یک ایناکتور مرکزی با سر وسط که دارای ولتاژ ثابتی در سر وسط

می باشد تشکیل شده است. که این، اثری شبیه به یک خود مبدل (auto transformer) دارد! وقتی یک سر انتهایی سیم پیچ پایین کشیده (pulled down) شود، سر دیگر بالا کشیده (fly up) خواهد شد، و برعکس. وقتی که یک سوئیچ باز می شود، القای بازگشتی، آن سر موتور را به سمت منبع مثبت خواهد برد، که با یک دیود بسته شده است. و سر مخالفش حرکتی رو به پایین (fly downward) خواهد داشت، و اگر نشود که در آن زمان ولتاژ منبع، شناور شود، به سطحی پایین تر از ground می رود، به عکس ولتاژی که روی دیود آن سر قرار دارد. بعضی از سوئیچ ها در برابر چنین بازگشتهایی محافظت شده اند ولی برخی دیگر از این بابت شدیداً آسیب پذیر هستند. همچنین شاید یک خازن هم بتواند ولتاژ بازگشتی را محدود کند، بطوریکه در شکل (۲-۷-۵) زیر نشان داده شده است :

شکل (۲-۷-۵)

قواعد تعیین اندازهٔ خازن در شکل (۲-۷-۵) مشابه قوانین تعیین اندازه برای شکل (۲-۷-۲) هستند. ولی اثر تشدید کاملاً متفاوت است! با یک موتور مغناطیس دائم، اگر خازن در فرکانس تشدید یا نزدیک به آن بکار برده شود، گشتاور دو مرتبه شبیه به گشتاور در سرعت پائین

افزایش خواهد یافت. گشتاور حاصل مطابق با منحنی نشان داده شده در شکل (۲-۷-٦) خواهد بود :

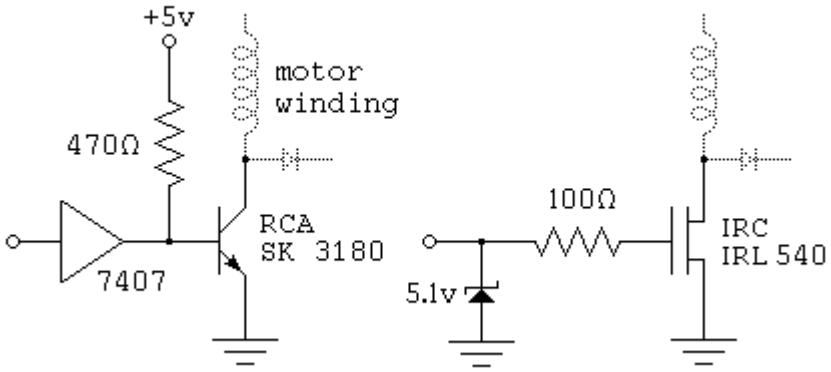


شکل (۲-۷-٦)

شکل (۲-۷-٦) یک نقطه اوج (پیک) را در فرکانس تشدید الکتریکی، و یک گودی را در فرکانس تشدید مکانیکی نشان میدهد. فرکانس تشدید مکانیکی وابسته است به گشتاور، لذا اگر فرکانس تشدید مکانیکی در جایی نزدیک به فرکانس تشدید قرار گیرد، توسط تشدید الکتریکی تغییر مکان (shift) خواهد یافت. از این گذشته پهنای تشدید مکانیکی وابسته است به شیب موضعی منحنی سرعت مخالف گشتاور. اگر گشتاور همراه سرعت افت کند، تشدید مکانیکی تیزتر خواهد بود، درحالیکه اگر گشتاور همراه سرعت افزایش یابد، دارای پهنای بیشتری خواهدبود یا بصورت دوتکه روی چند فر کانس تشدید تبدیل خواهد شد.

- راه اندازهای تک قطبی و رلوکتانس متغییر کاربردی
Practical Unipolar and Variable Reluctance Drivers

در مدارات فوق جزئیات سوئیچ های لازم، تعمداً آورده نشده است. در اینجا هر تکنولوژی سوئیچینگ از toggle switches گرفته تا power MOSFETS عمل خواهد کرد. شکل (۲-۷-۷) شامل بعضی از پیشنهادات برای پیاده سازی هر سوئیچ به همراه سیم پیچ موتور و دیود محافظ آن می باشد.



شکل (۲-۷-۷)

هر یک از سوئیچ های نمایش داده شده در شکل فوق با یک ورودی TTL سازگار است. منبع ۵ ولت استفاده شده برای عملیات منطقی، شامل درایور کلکتور باز 7407 بکار برده شده، باید به شکل درستی تهیه گردد. ولتاژ موتور معمولاً بین ۵ ولت تا ۲٤ ولت می باشد، و نیاز به تنظیم خاصی ندارد. توجه داشتن به این نکته اهمیت دارد که این مدارات سوئیچ قدرت، مختص راه اندازی سولئونویدها، موتورهای DC و بقیه بارهای القایی (inductive loads) مانند موتورهای پله ای هستند.

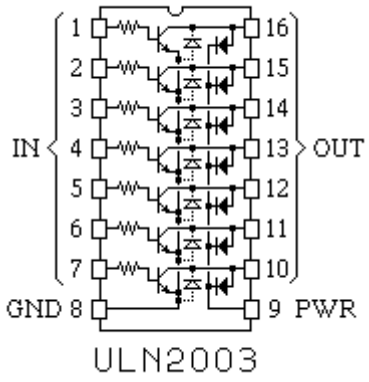
ترانزیستور SK3180 نشان داده شده در شکل (۲-۷-۷) یک دارلینگتون قدرت با گین جریان ۱۰۰۰ می باشد. بنابراین ده میلی آمپر از طریق مقاومت ٤۷۰ اهمی کافی است تا به ترانزیستور اجازه دهد که جریان آمپر پایینی را از طریق سیم پیچ موتور هدایت کند. بافر 7407 که برای راه اندازی دارلینگتون استفاده شده است، می تواند با هر تراشهٔ کلکتور باز ولتاژ بالای دیگری که بتواند جریان سینک حداقل ده میلی آمپری داشته باشد جایگزین گردد.

در صورتیکه ترانزیستور تحمل نکند، درایور کلکتور باز ولتاژ بالا، برای محافظت مابقی مدارات منطقی از منبع تغذیهٔ موتور عمل خواهد کرد.

IRC IRL540 نشان داده شده در شکل (۲-۷-۷) یک ترانزیستور قدرت میدانی است، که می تواند جریانی معادل حداکثر ۲۰ آمپر را تحمل کند و در ۱۰۰ ولت break down (قطع) می گردد. در نتیجه این تراشه می تواند القای بازگشتی را بدون دیود محافظ، در صورتیکه به یک هیت سینک متصل بوده باشد، جذب کند. این ترانزیستور زمان سوئیچینگ بسیار سریعی دارد، لذا دیود های محافظ باید متناظراً سریع باشند، یا با خازنهای کوچکی بسته شده باشند. در زمانیکه ترانزیستور قطع (fail) می کند، دیود زنر و مقاومت ۱۰۰ اهم، مدار TTL را محافظت می کنند. همچنین مقاومت ۱۰۰ اهمی برای کند کردن زمان سوئیچینگ ترانزیستور عمل می کند.

برای کاربردهایی که سیم پیچ موتور زیر ۵۰۰ میلی آمپر می کشد، خانوادهٔ ULN200X که بصورت آرایه ای دارلینگتونی می باشد و ساخت Allergo Microsystem می باشد، یا DS200X ساخت National Semiconductor ویا آرایهٔ دارلینگتونی MC1413 ساخت Motorolla سیم پیچ های چند گانهٔ موتور یا بقیهٔ بارهای القایی را مستقیماً بوسیلهٔ ورودی منطقی راه اندازی خواهد کرد.

شکل (۲-۷-۸) خروجی پینهای ULN2003 که یکی از تراشه هایی است که به وفور یافت می شود، را نشان می دهد. که یک آرایه دارای هفت ترانزیستور دارلینگتون با دو ورودی سازگار با TTL است.



شکل (۲-۷-۸)

مقاومت پایه روی هر ترانزیستور دارلینگتون، منطبق با خروجی TTL استاندارد دوقطبی می باشد. هر دارلینگتون NPN به امیتر خودش که به پایهٔ ۸ متصل است بسته شده، که بعنوان پین ground استفاده می شود. هر ترانزیستور این پکیج توسط دو دیود محافظت می شود. یکی که امیتر را به کلکتور متصل (shorting) می کند، که موجب محافظت در برابر ولتاژهای معکوس وارد شده به ترانزیستور می گردد، و دیگری که کلکتور را به پین ۹ متصل می کند. اگر پین ۹ به منبع مثبت موتور بسته شود، این دیود از ترانزیستور در برابر ضربهٔ القائی محافظت می کند.

تراشهٔ ULN2803 اساساً شبیه به تراشهٔ ULN2003 تشریح شده در بالا می باشد، به غیر از آنکه یک پکیج ۱۸ پین می باشد و شامل ۸ دارلینگتون است که راه اندازی یک جفت موتور مغناطیس دائم دوقطبی یا رلوکتانس متغییر را توسط یک تراشه ممکن می سازد.

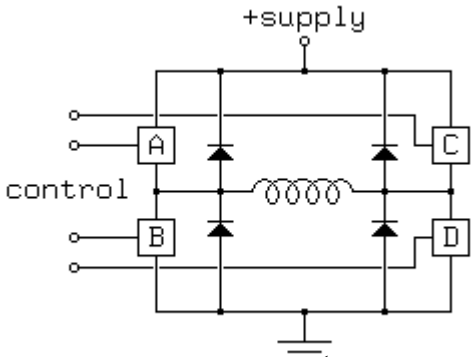
برای موتورهایی که هر سیم پیچشان کمتر از ٦۰۰ میلی آمپر می کشد، درایور چهار قدرتهٔ UDN2547B ساخت Allergo Microsystem ، چهار سیم پیچ موتورهای تک قطبی متعارف را راه اندازی می کند. برای موتورهایی که هر سیم پیچشان کمتر از ۳۰۰ میلی آمپر می کشد، درایور دو قدرتهٔ SN7541, 7542, 7543

انتخابهای مناسبی هستند. هر دو جایگزین فوق دارای یک قسمت منطق (logic) به همراه درایور قدرت می باشند.

- موتورهای دوقطبی و H-bridge

Bipolar Motors and H-Bridges

موضوع برای موتورهای پله ای مغناطیس دائم دوقطبی دارای پیچیدگی بیشتری است، چونکه اینها روی سیم پیچشان دارای سر وسط نیستند. بنابراین برای معکوس کردن جهت میدان تولید شده توسط یک سیم پیچ موتور، ما مجبور به معکوس کردن جهت جریان موجود در سیم پیچ هستیم. می توانیم از یک سوئیچ دوپل دو کنتاکت الکترومکانیکی برای اینکار استفاده کنیم. *معادل الکترونیکی یک چنین سوئیچی* ***H-bridge*** *نامیده می شود*. و در شکل (۲-۷-۹) بصورت اجمالی نشان داده شده است :



شکل (۲-۷-۹)

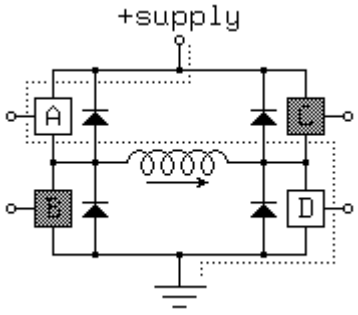
بطوریکه در مدار درایور تک قطبی که قبلاً بحث شده، سوئیچهای بکار برده شده در H-bridge باید دربرابر ولتاژ بازگشتی، که در زمان خاموش شدن یک سیم پیچ موتور بوجود می آید محافظت شود. که بوسیلهٔ دیودهایی که در شکل (۲-۷-۹) نشان داده شده صورت می گیرد.

مهم است که بدانیم H-bridge نه تنها برای کنترل موتورهای پله ای دو قطبی، بلکه برای کنترل موتورهای DC ، سولئونویدهای کششی- فشاری (push-pull soleonoids) (که در میدان ایجاد شده توسط مغناطیس دائمی قرار دارند) و بسیار کاربردهای دیگر قابل استفاده است.

بوسیلهٔ چهار سوئیچ، H-bridge می تواند ۱۶ مود عملیاتی امکانپذیر را ایجاد کند. مودهای عملیاتی مورد نیاز به شرح زیرند:

Forward mode, switches A and D closed.
Reverse mode, switches B and C closed.

اینها مودهای عملیاتی مرسوم هستند، که اجازه می دهند جریان منبع از طریق سیم پیچ به سمت ground به گردش درآید. شکل (۲-۷-۱۰) Forward mode را شرح می دهد.

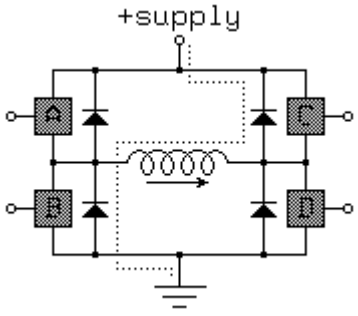


شکل (۲-۷-۱۰)

Fast decay mode or *coasting mode*, all switches open:

هرگونه گردش جریان از طریق سیم پیچ موتور، مخالف ولتاژ کل منبع خواهد بود، بعلاوه دو دیود موجب افت سریع جریان خواهد شد، این مود تا حدی بصورت غیر پویا موجب بروز حالتی شبیه به ترمز روی روتور موتور خواهد شد

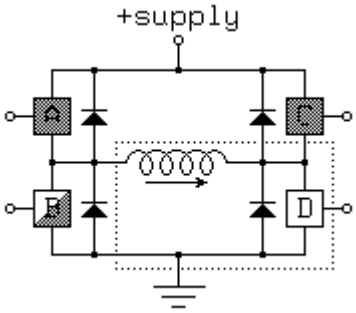
، لذا اگر همهٔ سیم پیچ ها در این مود به انرژی متصل باشند، روتور آزاد خواهد شد. شکل (۲-۷-۱۱) نشان دهندهٔ گردش جریان بلافاصله بعد از سوئیچینگ از مود Forward running به مود Fast decacy می باشد.



شکل (۲-۷-۱۱)

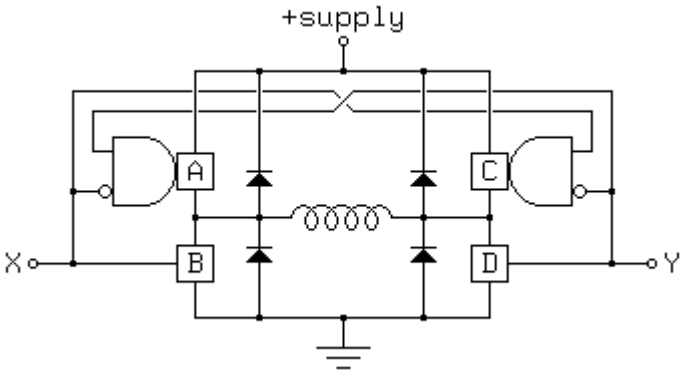
Slow decay modes or *dynamic braking modes*.

در این مودها جریان می تواند دوباره در داخل سیم پیچ ها با حداقل مقاومت به گردش دربیاید. در نتیجه، اگر جریان در یک سیم پیچ موتور در حال چرخش باشد، در حالیکه یکی از این مودها شروع به استفاده گردد، جریان به آرامی افت خواهد کرد. و اگر روتور در حال چرخش باشد، جریانی ایجاد خواهد کرد که موجب ایجاد ترمز روی موتور خواهد گشت. شکل (۲-۷-۱۲) یکی از مودهای افت آرام (slow decacy mode) را تشریح می کند که در آن سوئیچ D بسته است. اگر سیم پیچ موتور اخیراً در مود جلو رونده (Forward runnig mode) قرار داشته باشد، حالت سوئیچ می تواند باز یا بسته باشد.



شکل (۲-۷-۱۲)

اغلب H-bridge ها بگونه ای طراحی شده اند که از بروز اتصال کوتاه جلوگیری شود. در شکل (۲-۷-۱۳) سعی شده بهترین ترتیب قرار گیری اجزاء نمایش داده شود:



شکل (۲-۷-۱۳)

در اینجا مودهای عملیاتی زیر قابل دسترسی هستند:

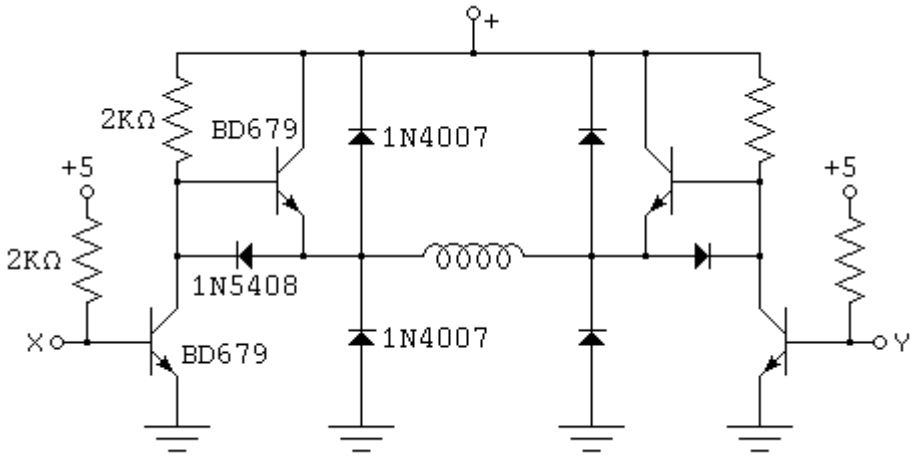
XY	ABCD	Mode
00	0000	fast decay
01	1001	**forward**
10	0110	**reverse**
11	0101	slow decay

مزیت ترتیب (arrangement) فوق این است که همهٔ مودهاي عملياتي حفظ شده اند، درحالیکه با کمترین تعداد بیت رمز (encode) شده اند. مهمترین مساله این است که در موقع استفاده از میکروکنترلر و سیستمهاي کامپیوتري براي راه اندازي H-bridge ، باید به محدودیت تعداد بیتهاي خروجي قابل استفاده در این سیستمها توجه داشت. متاسفانه تعداد معدودي از تراشه هاي مجتمع H-bridge موجود در بازار، از چنین طرح هاي ساده اي برخوردارند.

– مدارات راه انداز دوقطبي کاربردي

Practical Bipolar Drive Circuits

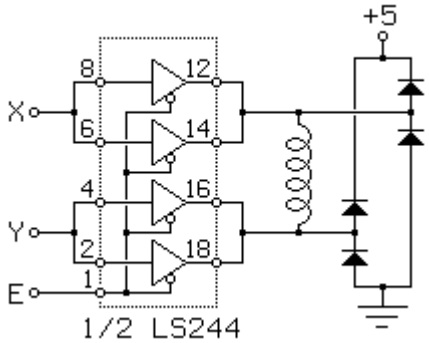
شماره هایی از درایورهای مجتمع H-bridge ، (integrated H-bridge drivers) در بازار وجود دارد، ولی هنوز هم می توانند در جدا کردن اجزاء برای درک چگونگی عملکرد یک H-bridge مورد استفاده واقع شوند. بعضی افراد چنین طرحی را برای یک H-bridege مطرح کرده اند:



شکل (۲-۷-۱٤)

ورودی های X,Y این مدار می تواند توسط خروجی های TTL کلکتور باز، به همان شکلی که در مدارات راه انداز تک قطبی بر پایهٔ دارلینگتون در شکل (۲-۷-۷) دیده شد راه اندازی شود. چنانچه یکی از X, Y بدرستی در حالت high و دیگری در حالت low باشد، به سیم پیچهای موتور انرژی داده خواهد شد. اگر هر دو low باشند، ترانزیستورهای pull-down خاموش خواهد شد. اگر هر دو high باشند، ترانزیستورهای pull-up خاموش خواهد شد. در نتیجه، این مدار ساده موتور را به مود ترمز دینامیکی (dynamic braking mode) در دو حالت 11, 00 خواهد برد و اجازهٔ سریدن (coasting mode) را نخواهد داد. شکل (۲-۷-۱٤) عیناً از دو قسمت مشابه تشکیل شده است، که هر کدام می توانند یک درایور pull–push نامیده شوند. اصطلاح half H-bridge در بعضی از مواقع برای این مدارات بکار می رود.

این نکته مهم است که بدانیم یک half H-bridge مداری کاملاً مشابه با مدار راه انداز خروجی دارد، که در منطق TTL استفاده می شود. در واقع راه انداز خط سه وضعیتهٔ TTL ، مثل 74LS125A و 74LS244 می تواند بعنوان یک half H-bridge برای بارهای کم مورد استفاده قرار گیرد. بطوری که در شکل زیر توضیح داده شده:



شکل (۲-۷-۱۵)

این مدار برای راه اندازی موتورهایی با سیم پیچ هایی با مقاومت حدود ۵۰ اهم در ولتاژ حداکثر ٤/۵ ولت با استفاده از منبع ۵ ولتی کاربرد دارد. هر بافر سه وضعیته در LS244 می تواند جریان سینکی معادل دو برابر جریان source اش داشته باشد. و مقاومت داخلی بافر برای زمانیکه جریان source بصورت زوج بین دو درایور که بصورت موازی در حال کارند کفایت می کند. این درایور موتور، همهٔ حالتهای قابل استفاده

قابل تولید توسط درایور شکل (۲-۷-۱۳) را بوجـود می آورد، ولـی این حـالات بصورت موثر رمزگذاری(encode) نشده اند.

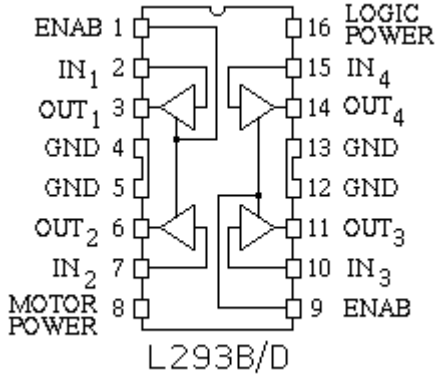
XYE	Mode
--1	fast decay
000	slower decay
010	**forward**
100	**reverse**
110	slow decay

دومین مود ترمز دینامیکی (dynamic braking mode) ، XYE=110 ، دارای عمل ترمز ضعیفتری نسبت به اولی است، چون در حقیقت درایور LS244 جریان سینک بیشتری نسبت به جریان سورس اش دارد.

یکی از مشکلات تراشه های موتورهای پله ای موجود در بازار این است که زمان عرضهٔ آنها به بازار بسیـار کوتاه است. برای مثال تـراشه های dual H-bridge سـری lpxMxx (IP1M10 تا IP3M12) دارای استقبال زیادی بود، ولی متاسفانه معلوم شد که seagate آنها را فقط برای یافتن موقعیت هد در دیسک درایوهای خود ساخته است. و یا درایور dual H-bridge مدل TA7279 ساخت توشیبا یکی دیگر از انتخابهای بسیار عالی برای موتورهای زیر یک آمپر بود، ولی دوباره مشخص شد که آنهم برای استفادهٔ داخلی آن شرکت ساخته شده است.

L293 dual H-bridge ساخت SGS-Thompson (و بقیه) رقیبی برای تراشه های فوق است، ولی برخلاف آنها دارای دیود محافظ نیست. تراشهٔ L293D ، که بعداً شرح داده خواهد شد، از نظر پایه ها (پین ها) شبیه L293 است، ولی دارای دیود محافظ هست. اگر مدلهای قدیمی L293 را مورد استفاده قرار دهیم، باید هر سیم پیچ موتور را به یک پل یکسوساز (مثل1N4001) ببندیم. استفاده از دیودهای خارجی این امکان را می دهد که یک مقاومت سری در مسیر چرخش جریان قرار بگیرد و جریان موجود در سیم پیچ موتور را در هنگام خاموش شدن سریعتر میرا کند، که برای بعضی

کاربردها مطلوب خواهد بود. خانوادهٔ L293 انتخابهای بسیار خوبی برای راه اندازی موتورهای دوقطبی حداکثر با یک آمپر جریان به ازاء هر سیم پیچ می باشند که حداکثر با ۳۶ ولت کار می کنند. شکل (۲-۷-۱۶) خروجی پین مدلهای مرسوم تراشه های L293B و L293D را نشان می دهد.



شکل (۲-۷-۱۶)

این تراشه می تواند بعنوان چهار half H-bridge مستقل دیده شود، که بصورت جفت فعال می شوند، یا دو full H-bridge که پایه های ٤، ٥، ۱۲ و ۱۳ آن برای هدایت گرما به یک بورد PC یا به یک هیت سینک خارجی طراحی شده است.

L298 dual H-bridge ساخت SGS-Thompson (و بقیه) کاملاً شبیه بالاست، ولی قادر به بکارگیری بیش از دو آمپر جریان به ازاء هر کانال است، و بصورت یک مولفهٔ قدرت بسته بندی (پکیج) شده است. بعنوان مثال با LS244 ، مشکلی نیست اگر دوH-bridge در پکیج L298 را به یک H-bridge ، چهار آمپر متصل کنیم (دفترچهٔ مشخصات این تراشه چگونگی این عمل را مشخص کرده است). یک اخطار در مورد L298 وجود دارد. این تراشه بسیار سریع سوئیچ می کند، آنقدر سریع که دیودهای محافظ (1N400x و مشابه) عمل نمی کنند. در عوض از یک دیود مثل BYV27 می توان استفاده کرد. LMD18200 H-bridge ساخت National Semiconductor یک مثال خوب دیگر است و دارای دیودهای محافظ مجتمع شده می باشد و تا سه آمپر را پشتیبانی میکند.

در حالیکه H-bridgeهای مجتمع برای جریان و ولتاژهای خیلی بالا مناسب نیستند، اجزاء طراحی شده مناسبی در بازار یافت می شود که می تواند ساختمان H-bridge ها

را بوسیلهٔ سوئیچ های مجزا ساده کند. بطور مثال، International Rectifier یک خط فروش درایور half H-bridge دارد. دو تا از این تراشه ها بعلاوهٔ چهار ترانزیستور سوئیچینگ MOSFET برای ساخت یک H-bridge کافی است. IR2101 ، IR2102 و IR2103 درایورهای H-bridge مبنا هستند. هر یک از این تراشه ها دارای دو ورودی منطقی برای کنترل مسقیم دو ترانزیستور سوئیچینگ روی یک پایه از یک H-bridge می باشـند. IR2104 و IR2111 دارای خـروجی های مشابهی برای کنترل سوئیچینگ یک H-bridge هستند. همچنین آنها دارای ورودیهای منطقی ای هستند که در بعضی کاربردها، می تواند نیاز به منطق خارجی را کاهش دهد. بصورت اخص 2104 شامل یک enable input می باشد که چهار تا 2104 بعلاوهٔ هشت ترانزیستور سوئیچینگ بدون نیاز به منطق اضافی دیگر می تواند بجای یک L293 استفاده شود. تعدادی از سازندگان، تراشه های complex H-bridge می سازند که دارای مدارات محدود کنندهٔ جریان می باشد. همچنین خوب است بدانیم که تعدادی درایور 3-phase bridge در بازار وجود دارد که برای راه اندازی موتورهای پله ای مغناطیس دائم سه فاز دارای پیکر بندی دلتا یاY می باشد. تعداد کمی از چنین موتورها وجود دارد که، درایورهایش به همراه آنها گسترش نیافت. با این وجود مدلهای P TA288 ، GL7438 ، TA8400 و TA8405 طرحهای خوبی هستند، و دو تا از چنین تراشه هایی به همراه شش half bridge ، به خوبی برای کنترل یک موتور پله ای پنج سیم ، با ده پله به ازاء یک دور کامل قابل استفاده است.

۲–۸) نرم افزار کنترل موتور پله ای

Στεππερ Μοτορ Χοντρολ Σοφτωαρε

چنانچه یکی از کنترلرهای بخش (۲–۷) را بسازیم، بوسیلهٔ کدهایی که در زیر مشاهده خواهید کرد می توانیم موتورهایی را به کنترلرها متصل کرده و آنها را به چرخش درآوریم. در اینجا برای اینکه خیلی سخت گرفته نشود از زبان پاسکال که عمومیت زیادی دارد استفاده شده است. این کد فقط یک موتور را پشتیبانی می کند و این کار را با استفاده از بیتهای کم ارزش پورت پارالل انجام خواهد داد. درحالیکه می توان از بیتهای پر ارزش برای راه اندازی موتور دیگر یا کاربردهای همزمان دیگری استفاده کرد. حتی می توان از آن برای کنترل مالتی پلکسر که موتورهای زیادی به آن متصل است استفاده کرد و در هر نوبت موتور خاصی را راه اندازی کرد. خلاصه اینکه این بخش بیشتر جنبهٔ آشنایی با چگونگی فرمان دادن به موتور از طریق پورت پارالل و مدار کنترل را مد نظر قرار داده است.

این قطعه برنامه برای یک موتور رلوکتانس متغییر با سه سیم پیچ استفاده می شود:

```
      const maxstep = 2;
            steps = 3;
      var   steptab: array [0..maxstep] of integer;
            step: integer;
            motor: file of integer; { this is the I/O port for the motor
}
      begin
            step := 0;
            steptab[0] = 1; { binary 001 }
            steptab[1] = 2; { binary 010 }
            steptab[2] = 4; { binary 100 }
            write( motor, steptab[step] );
```

این قطعه برنامه برای یک موتور مغناطیس دائم تک قطبی با سیم پیچ دارای سر وسط و یا دوقطبی با مدار درایور **H-bridge** *استفاده می شود:*

```
      const maxstep = 3;
            steps = 4;
      var   steptab: array [0..maxstep] of integer;
            step: integer;
            motor: file of integer; { this is the I/O port for the motor
}
      begin
            step := 0;
```

```
        steptab[0] = 1; { binary 0001 }
        steptab[1] = 4; { binary 0100 }
        steptab[2] = 2; { binary 0010 }
        steptab[3] = 8; { binary 1000 }
        write( motor, steptab[step] );
```

این قطعه برنامه برای کنترل نیم پله یک موتور پله ای مغناطیس دائم استفاده می شود:

```
    const maxstep = 7;
          steps = 8;
    var   steptab: array [0..maxstep] of integer;
          step: integer;
          motor: file of integer; { this is the I/O port for the motor
}
    begin
          step := 0;
          steptab[0] = 1;  { binary 0001 }
          steptab[1] = 5;  { binary 0101 }
          steptab[2] = 4;  { binary 0100 }
          steptab[3] = 6;  { binary 0110 }
          steptab[4] = 2;  { binary 0010 }
          steptab[5] = 10; { binary 1010 }
          steptab[6] = 8;  { binary 1000 }
          steptab[7] = 9;  { binary 1001 }
          write( motor, steptab[step] );
```

این قطعه برنامه برای کنترل یک موتور پنج فاز به همراه یک **H-bridge** *روی هر یک از پنج سر موتور استفاده می شود:*

```
    const maxstep = 9;
          steps = 10;
    var   steptab: array [0..maxstep] of integer;
          step: integer;
          motor: file of integer; { this is the I/O port for the motor
}
    begin
          step := 0;
          steptab[0] = 13;  { binary 01101 }
          steptab[1] = 9;   { binary 01001 }
          steptab[2] = 11;  { binary 01011 }
          steptab[3] = 10;  { binary 01010 }
          steptab[4] = 26;  { binary 11010 }
          steptab[5] = 18;  { binary 10010 }
          steptab[6] = 22;  { binary 10110 }
          steptab[7] = 20;  { binary 10100 }
          steptab[8] = 21;  { binary 10101 }
          steptab[9] = 5;   { binary 00101 }
          write( motor, steptab[step] );
```

باقیماندهٔ کدهای زیر برای همه یکسان است و به نوع موتور بستگی ندارد. زیر برنامه زیر، موتور را در هر یک از جهات قابل حرکت، حرکت خواهد داد، که پارامتر جهت باید

۱+ یا ۱- باشد تا جهت حرکت را مشخص کند.

```
procedure onestep( direction: integer );
begin
    step := step + direction;
    if step > maxstep then step := 0
    else if step < 0 then step := maxstep;
    write( motor, steptab[step] );
end;
```

کنترل نرم افزاری موتور پله ای، یک وظیفهٔ زمان واقعی (real time task) است، و شما حداقل یک بیت فیدبک برای کنترل آنچه که اتفاق افتاده نیاز دارید. بصورت نوعی، این بیتی خواهد بود که نشانگر حرکت دنده ها (یا هر چیز دیگر که نشان دهد موتور در حال چرخش است) می باشد، که وقفه ای را بوجود می آورد و بوسیلهٔ یک شعاع نوری یا بسته شدن یک میکروسوئیچ بوجود می آید. برای اجتناب از بروز مشکل در خواندن مکان از طریق این سنسور، مجبوریم که یک نقطه را بعنوان نقطهٔ صفر برای حرکت در نظر بگیریم. بخصوص در استفاده از سوئیچ ها یا جائیکه دنده هایی مابین موتور و صفحهٔ دوار دیگری درگیر باشند، معمولاً بازگشت به مکان صفر به درستی اتفاق نمی افتد. با فرض اینکه می توانیم یک بیت بازگشتی را بخوانیم و یک وقفهٔ تایمر زمانی برنامه پذیر روی سیستم داریم، ساختن یک روتین سرویس وقفه تایمر، که موتور را راه بیاندازد به شکل زیر امکانپذیر است:

```
const maxpos = 11111; { maxpos + 1 is calls to onestep per rev }
    var position: integer; { current position of motor }
        destination: integer; { desired position of motor }
        direction: integer; { direction motor should rotate }
        last: integer; { previous value from position sensor }
        sensor: file of integer; { parallel input port }
    begin
        read( sensor, last );
        position := 1;
        setdest( 0, 1 ); { force turntable to spin on power-up until
                           it finds it's home position }

    procedure timer; { interval timer interrupt service routine }
    var sense: integer;
    begin
        read( sensor, sense );
        if (direction = 1) and (last = 0) and (sense = 1)
          then position = 0;
        last := sense;

        if position <> destination then begin
            onestep( direction );
            position := position + direction;
            if position > maxpos then position := 0
            else if position < 0 then position := maxpos;
        end;
```

```
    if position <> destination
      then settimer( interval_until_next_step );
end;
```

زیربرنامهٔ زیر مکان مقصد صفحهٔ گردنده و مسیر چرخش را تنظیم می کند، سپس تایمر زمانی را برای دادن یک وقفهٔ فوری تنظیم می کند، و اجازه می دهد که روتین تایمر، چرخش صفحهٔ دوار را به آخر برساند، در حالیکه برنامهٔ کاربردی (application program) می تواند هر کار دیگری که می خواهد را انجام دهد.

```
procedure setdest( dst,dir: integer );
begin
    destination := dst;
    direction := dir;
    if position <> destination
      then settimer( min_interval ); { force a timer interrupt }
end;
```

اگر بخواهیم چند موتور پله ای را کنترل کنیم کارمان خیلی ساده خواهد شد اگر یک تایمر زمانی (interval timer) ، به همراه یک پورت پارالل به ازاء هر موتور پله ای داشته باشیم. اگر سخت افزارمان فقط یک تایمر داشته باشد، می توانیم با شبیه سازی آنرا به چند تایمر بدل کنیم، که این البته کار زیادی برای نوشتن یک برنامهٔ اجرایی زمان واقعی خواهد برد.

نتیجه نهایی اینکه: اگر بخواهیم یک موتور را خیلی سریع حرکت دهیم، آن موتور دچار خطا خواهد شد و نرم افزار، موقعیت واقعی محور موتور را از دست خواهد داد. موتورها، همانگونه که گفته شد دارای یک نرخ پله ماکزیمم هستند که توانایی موتور در ایجاد حداکثر پله به ازاء یک ثانیه را معین می سازد، که ممکن است نتوان موتور را بدون آنکه از ابتدا آنرا با حرکت آرام چرخانده باشیم، از حالت سکون با آن تعداد پله در ثانیه به چرخش درآورد. لذا شتابدهی به موتورهای پله ای تا حدی محدود است.

در کدهای بالا interval_unit_next_step بعنوان یک ثابت نشان داده شده است. اگر شما بخواهید در مواردی که بارهای سنگین به موتور اعمال می شود موتور را از حالت سکون به سرعت حرکت دهید، یا از زمان حرکت کوتاه استفاده کنید، می توانید آنرا متغییر فرض کنید، تا فرجهٔ شروع و ایست حرکت موتور را طولانی تر کنید که اینکار برای مراقبت از شتاب گیری و کاهش شتاب سریع موتور صورت می گیرد.

۲-۹) آشنایی با چند موتور پله ای قابل دسترس در بازار

مثال -۱

The STH-39D137-04 75ohm stepper motor, from a PC clone half-height5.25 " floppy disk drive, has Brown, Black, Orange, Yellow and Red wires.

Table of resistances between pairs of wires

Brown				
Black	149.6			
Orange	148.9	148.9		
Yellow	148.4	148.6	148.3	
Red	74.9	75.0	74.6	74.1
	Brown	Black	Orange	Yellow

So it's clear that Red is the Common Power wire

Supplying power (up to 12v for a disk drive stepper motor) to the Red wire, grounding the Yellow wire and assuming it's connected to Coil 4 -

- Grounding the Orange wire has no effect - Coil 2
- Grounding the Black wire produces a small clockwise rotation - Coil 3
- Grounding the Brown wire produces a small anticlockwise rotation - Coil 1

So the connections are as follows

Red	Common Power
Brown	Coil 1
Black	Coil 3
Orange	Coil 2
Yellow	Coil 4

It is the same for the following stepper motors out of half-height5.25 " floppy disk drives -

Model Name	Motor Manufacturer	Drive Manufacturer
MSJE200A73	Mitsubishi/Sankyo	Mitsubishi
KP39HM4	Japan Servo Motors	Yaskawa Electric/IBM

مثال – ۲

The TEAC 14769070-10 stepper motor, from a PC clone half-height 5.25" floppy disk drive, has six wires - two Brown, Yellow, Red, Blue and White.

Table of resistances between pairs of wires

Brown1					
Brown2	7.0				
Yellow	81.2	74.4			
Red	74.1	80.9	155.2		
Blue	81.3	74.5	148.6	155.2	
White	74.3	81.1	155.3	148.1	155.3
	Brown1	Brown2	Yellow	Red	Blue

So it's clear that the two Brown wires are Common Power, both connected to all four coils.

Supplying power (up to 12v for a disk drive stepper motor) to the Brown wires, grounding the Yellow wire and assuming it's connected to Coil 4 -

- Grounding the Red wire produces a small anticlockwise rotation - Coil 1
- Grounding the Blue wire has no effect - Coil 2
- Grounding the White wire produces a small clockwise rotation - Coil 3

So the connections are as follows

Browns	Common Power
Red	Coil 1
White	Coil 3
Blue	Coil 2
Yellow	Coil 4

مثال – ۳

The stepper motor out of an Apple II 5.25" floppy disk drive has 6 wires - Brown, Black and Red in one group and Orange, Yellow and Red in another.

Table of resistances between pairs of wires

Brown					
Black	73.2				
Red1	36.6	36.8			
Orange	*	*	*		
Yellow	*	*	*	73.1	
Red2	*	*	*	36.4	37.0
Brown	Black	Red1	Orange	Yellow	Red2

* No Reading, so not connected

So it's clear that one Red wire provides power to the Brown and Black coils and the other Red Wire provides power to the Orange and Yellow coils. Common sense would suggest this, from the way the wires are grouped, and the readings in the table confirm it.

Supplying power (up to 12v for a disk drive stepper motor) to the Red wires, grounding the Yellow wire and assuming it's connected to Coil 4 -

- Grounding the Orange wire has no effect - Coil 2
- Grounding the Black wire produces a small clockwise rotation - Coil 3
- Grounding the Brown wire produces a small anticlockwise rotation - Coil 1

So the wiring scheme is as follows

Reds	Common Power
Brown	Coil 1
Black	Coil 3
Orange	Coil 2
Yellow	Coil 4

which *happens* to be the same as in Example 1

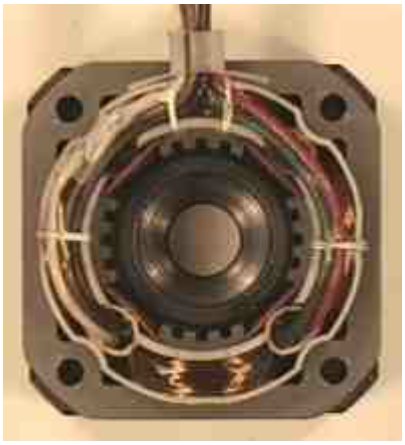
مثال – ٤

KP4M4-001 Stepper Motor

+12v dc, four-phase, unipolar, permanent magnet, 3.6°per step

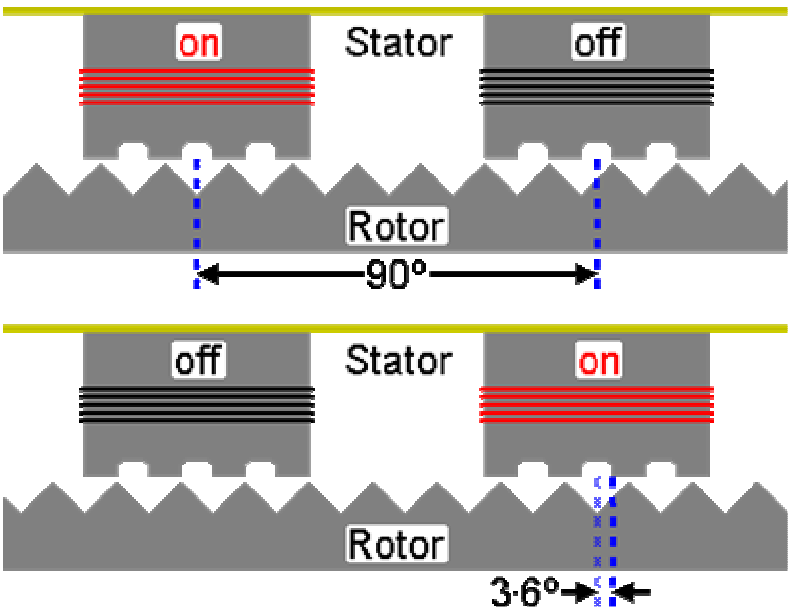
در موتور پله ای KP4M4-001 ، مغناطیس دائم بصورت شمال – جنوب در امتداد محور روتور قرار دارد، و بصورت دو قسمتی پوشش داده شده است که هر کدام از قسمتها دارای ۲۵ دندانه در لبهٔ خود هستند. دندانه ها در قسمت جنوب و شمال بعلت وجود فضا (gap) بین آنها دارای دو فاز جداگانه هستند.

این به آن معنی است که در زمانهایی که دندانه های قسمت شمال در حال جذب شدن توسط پلهای استاتور هستند، دندانه های جنوب درحال دفع آن پلها هستند.



با داشـتن ۲۵ دنـدانه در محـیط روتور و چهار سیـم پیچ در محیـط استـاتور، موتور پله ای KP4M4-001 در هر دور کامل، ۱۰۰ پله تولید می کند.

فاصلهٔ بین دندانه ها ۱٤/٤° = ۲۵ / ۳٦۰° است. وقتی دندانه های روتور با دندانه های پلی که روی استاتور درحال القاء کردن است همتراز می شوند، به اندازهٔ یک چهارم زاویهٔ دندانه ها، با پل بعدی استاتور ناهمـتراز هستند. بنابراین وقتی سیم پیچ بعدی انرژی داده شود، روتور به انـدازهٔ یک چهارم ۱٤/٤° گـردش خواهد کرد که پله ای به اندازهٔ ۳/٦° را تولید می نماید. تصویر زیر به درک آنچه گفته شد کمک می کند.



بصورت عمومی، برای موتورهایی از این نوع داریم :

تعداد دندانه های روتور × تعداد فاز (سیم پیچ) = تعداد پله به ازاء یک دور کامل

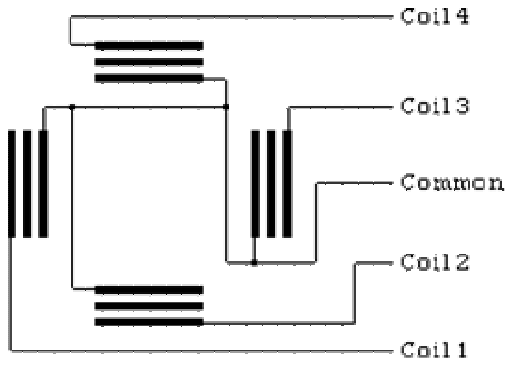
(در بسیاری از انواع موتورهای پله ای، تعداد فاز با تعداد سیم پیچ برابر نیست.)

۲-۹-۱) شناسایی بعضی از موتورهای پله ای از روی تعداد و رنگ سیم

* یک موتور پله ای با ۵ سیم، تقریباً همیشه یک تک قطبی چهار فاز است.
* یک موتور پله ای با ٦ سیم، احتمالاً یک تک قطبی چهار فاز است ولی دو سر مشترک دارد که هر دو می تواند دارای رنگ یکسان باشد.
* یک موتور پله ای با فقط ٤ سیم، به احتمال زیاد یک دو قطبی است.

دو مرحله برای جدا سازی موتورهای تک قطبی در حالت ۵ و یا ٦ سیم باید طی شود.

۱- جداسازی سر(های) مشترک با چک کردن مقاومت هر جفت سیم، بوسیلهٔ یک اهم متر. یک سر مشترک باید وجود داشته باشد که فقط دارای نصف مقاومت مابین همهٔ سرها می باشد.

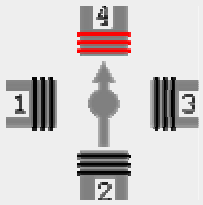
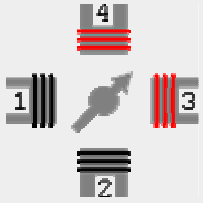
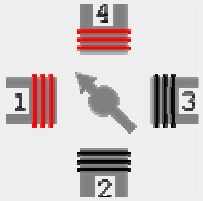
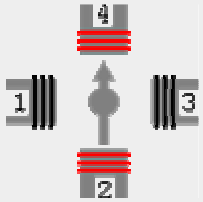


1.

این به آن دلیل است که سر مشترک فقط دارای یک سیم پیچ بین خود و سیمهای دیگر است، در حالیکه بقیهٔ سیمها دارای ۲ سیم پیچ مابین همدیگر هستند. از اینرو نصف مقاومت بقیه را دارد.

۲ – جداسازی از طریق شناسایی سیم پیچ ها بوسیلهٔ دادن ولتاژ به سر(های) مشترک و نگاهداشتن یکی دیگر از سیم ها در حالت زمین درحالیکه به نوبت سرهای دیگر را به زمین متصل کرده و نتیجه را ملاحظه می کنیم.

1.

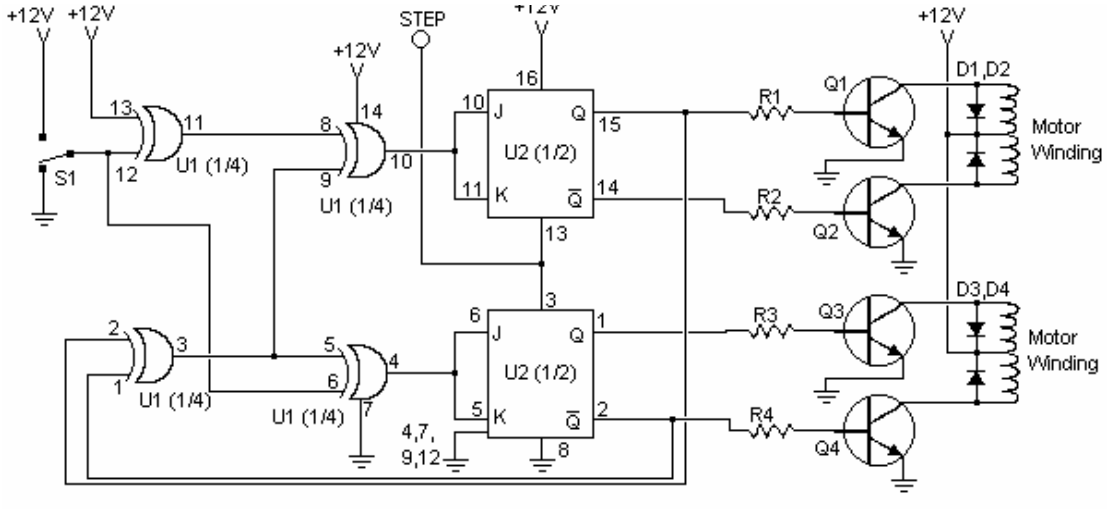
Select one wire and ground it Assume it's connected to coil 4	
Keeping it grounded, ground each of the other three wires one by one	
Grounding one wire should make the rotor turn a little clockwise That'll be the wire connected to Coil 3	
Grounding one wire should make the rotor turn a little anticlockwise That'll be the wire connected to Coil 1	
Grounding one wire should do nothing That'll be the wire connected to Coil 2	

توجه : تعداد سیم پیچ های فوق کاملاً اختیاری است.

تنها چیزی که باقی می ماند، اعمال ولتاژ با ترتیب مناسب به موتور شناسایی شده است.

۲-۱۰) بررسی بعضی از مدارات کنترل و درایور موتورهای پله ای رایج

ابتدا یک شماتیک از کنترلر بسیار ساده و ارزان موتور پله ای را بررسی می کنیم:



شکل (۲-۱۰-۱)

لسیت قطعات :

Part	Total Qty.	Description	Substitutions
R1, R2 ,R3, R4	4	1K 1/4W Resistor	
D1, D2, D3, D4	4	1N4002 Silicon Diode	
Q1, Q2, Q3, Q4	4	TIP31 NPN Transistor (See Notes)	TIP41, 2N3055
U1	1	4070 CMOS XOR Integrated Circuit	
U2	1	4027 CMOS Flip-Flop	
S1	1	SPDT Switch	
MISC	1	Case, Board, Wire, Stepper Motor	

توجه :

۱- باید بجای Q1-Q4 از ترانزیستورهای قدرت (مثل2N3055 و مشابه) استفاده شود.

۲- هر بار که به خط STEP پالس وارد شود، موتور یک پله حرکت خواهد کرد.

۳- S1 جهت چرخش موتور را تغییر می دهد.

در مدار فوق از دوJK flip-flop استفاده شده است که جدول مشخصهٔ آن به صورت زیر است :

J	K	Q(t=1)
0	0	0
0	1	0
1	0	1
1	1	Q(t)*

خروجی این حالت به خروجی قبلی وابسته است ،در واقع Q(t) قبلی اگر مثلا یک باشد Q(t) جدید صفر می شود و بالعکس.

همچنین جدول درستی گیت های XOR بکار برده شده نیز به صورت زیر است :

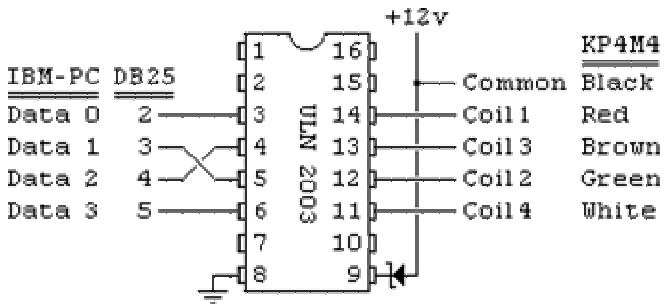
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

با داشتن این دو جدول به راحتی می توان مدار فوق تحلیل کرد.

-نمونه دیگر از مدارات کنترل که با ULN2003 ساخته شده به شکل زیر است.

الف: استفاده از چهار خط داده برای کنترل یک موتور:

ULN2003/MC1413 هفت بیتی ۵۰ ولت ۵۰۰ میلی آمپری هستند که دارای ورودی TTL از نوع درایور دارلینگتون می باشند. مدار زیر برای کنترل یک موتور تک قطبی چهارفاز مثل KP4M4-001 در نظر گرفته شده است.



شکل (۲-۱۰-۳)

توصیه می شود که از یک دیود زنر ۱۲ ولت بین منبع تغذیه و V_{DD} تراشه (پین ۹) استفاده شود، تا اثر بازگشتی حاصل از میدان مغناطیسی لحظهٔ خاموش شدن سیم پیچ های موتور خنثی گردد.

در زیر تعـدادی نمونه برنامه به زبان پاسـکال که می تواننـد بوسیـلهٔ مدار شکل فوق، موتور پله ای چهـارفاز (مثل KP4M4-001) را در هر دو جهت بچـرخانند، آورده شـده است.

الف –۱) برنامه تحریک یک سیم پیچ :

```
program first1; { Single-Coil Excitation }

uses crt;

const OUTREG    = $378; { or $278 or $3BC }

var i: integer;

procedure delay; { 33MHz 486 }
var i: integer;
begin
    for i := 1 to 10000 do { nothing }
end;

procedure delay2;
var i: longint;
begin
    for i := 1 to 100000 do { nothing }
end;
begin { main }
    ClrScr;

    writeln;
    writeln('***** Direct control *****');
    writeln;

    port[OUTREG] := $00; { all off }

    write('All OFF to start with  ');
    readln;

    for i := 1 to 25 do { Clockwise }
    begin
        port[OUTREG] := $08; delay;
        port[OUTREG] := $04; delay;
        port[OUTREG] := $02; delay;
        port[OUTREG] := $01; delay
    end;

    for i := 1 to 25 do { Anti-Clockwise }
    begin
        port[OUTREG] := $01; delay;
        port[OUTREG] := $02; delay;
        port[OUTREG] := $04; delay;
        port[OUTREG] := $08; delay
    end;
```

```
    port[OUTREG] := $00; { all off }

    write('That''s all, folks !    ');
    readln
end.
```

الف – ۲) برنامهٔ تحریک دو سیم پیچ :

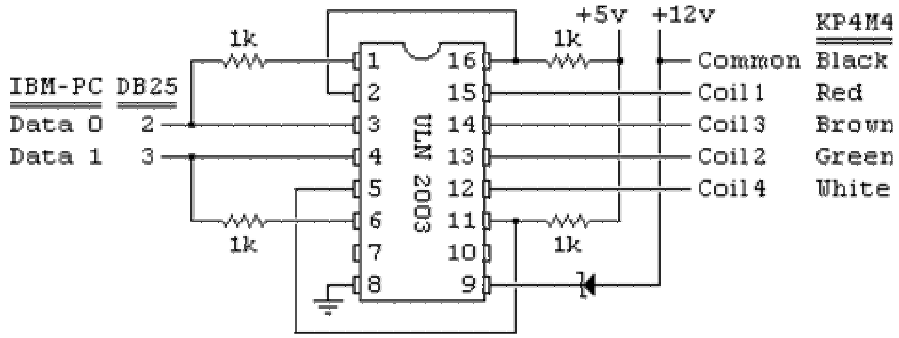
```
program second2; { Two-Coil Excitation }

uses crt;

const OUTREG    = $378;

var i: integer;

procedure delay;
var i: integer;
begin
    for i := 1 to 10000 do { nothing }
end;

begin { main }
    ClrScr;

    writeln;
    writeln('***** Direct control *****');
    writeln;

    port[OUTREG] := $00; { all off }

    write('All OFF to start with  ');
    readln;

    for i := 1 to 25 do { clockwise }
    begin
        port[OUTREG] := $0C; delay;
        port[OUTREG] := $06; delay;
        port[OUTREG] := $03; delay;
        port[OUTREG] := $09; delay
    end;

    for i := 1 to 25 do { anti-clockwise }
    begin
        port[OUTREG] := $09; delay;
        port[OUTREG] := $03; delay;
        port[OUTREG] := $06; delay;
        port[OUTREG] := $0C; delay
    end;

    port[OUTREG] := $00; { all off }

    write('That''s all, folks !    ');
    readln
end.
```

الف – ۳) برنامهٔ تحریک مختلط با کنترل نیم پله :

```
program third3; {interleaved Single-and Two-Coil Excitation- Half-
Stepping}
```

```
uses crt;

const OUTREG    = $378;

var i: integer;

procedure delay;
var i: integer;
begin
    for i := 1 to 10000 do { nothing }
end;

begin { main }
    ClrScr;

    writeln;
    writeln('***** Direct control *****');
    writeln;

    port[OUTREG] := $00; { all off }

    write('All OFF to start with  ');
    readln;

    for i := 1 to 25 do { Clockwise }
    begin
        port[OUTREG] := $08; delay;
        port[OUTREG] := $0C; delay;
        port[OUTREG] := $04; delay;
        port[OUTREG] := $06; delay;
        port[OUTREG] := $02; delay;
        port[OUTREG] := $03; delay;
        port[OUTREG] := $01; delay;
        port[OUTREG] := $09; delay
    end;

    for i := 1 to 25 do { Anti-Clockwise }
    begin
        port[OUTREG] := $09; delay;
        port[OUTREG] := $01; delay;
        port[OUTREG] := $03; delay;
        port[OUTREG] := $02; delay;
        port[OUTREG] := $06; delay;
        port[OUTREG] := $04; delay;
        port[OUTREG] := $0C; delay;
        port[OUTREG] := $08; delay
    end;

    port[OUTREG] := $00; { all off }

    write('That''s all, folks !    ');
    readln
end.
```

ب: استفاده از دو خط داده : با افزودن چند مقاومت، می توانید چنین موتورهایی (تک قطبی چهارفاز) را با استفاده از تنها دو خط data پورت پارالل کنترل کنید. این مدار از آن اصل که در تحریک دوفاز (Two-coil Excitation) ، در هر لحظه دو سیم پیچ دارای تحریک مخالف دو سیم پیچ دیگر هستند، استفاده می کند.



شکل (۲-۱۰-٤)

ایرادی که این مدار دارد این است که نمی توان از آن برای تولید نیم پله استفاده کرد.

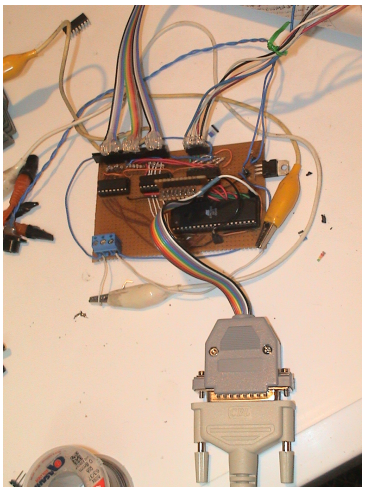
۳

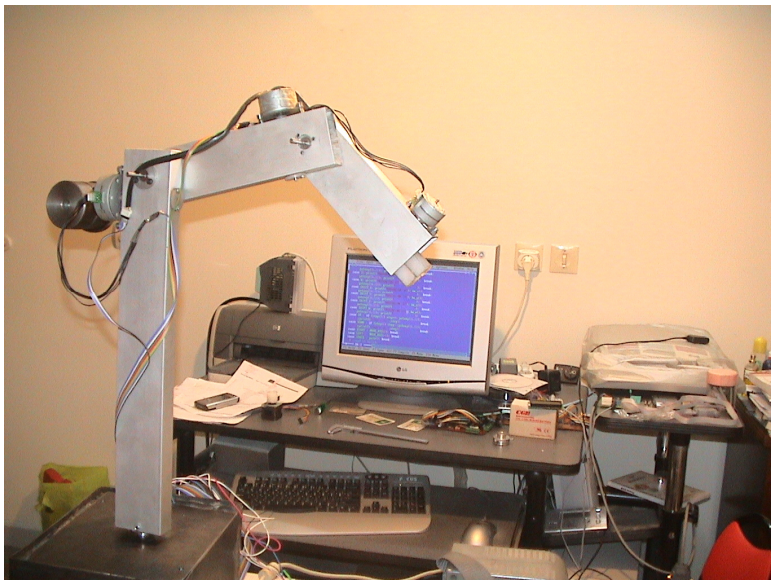
فصل سوم

نرم افزار و سخت افزار پروژه

پس از مطالعهٔ نسبتاً کامل پورت پارالل و موتورهای پله ای در دو فصل گذشته، اینک نوبت به طراحی و برنامه نویسی موضوع پروژه رسیده است. در این بخش با توجه به مطالب گفته شده در فصول قبل و با بررسی نوع نیازمان که بکارگیری چهار موتور پله ای از طریق پورت پارالل در کنترل بازوی ربات می باشد، بصورت زیر عمل می کنیم.

بازو روی یک پایه (بدنه)سنگین تر قرار می گیرد تا بتواند وزن linkهای بعدی وموتور های بعدی را تحمل کند. موتور اول که بزرگتر از همه است در داخل بدنه کار گذاشته می شود که دارای ۳۶۰ درجه زاویهٔ چرخش است وموتور های بعدی حداقل ۱۸۰ درجه زاویهٔ چرخش را دارند. (البته بنا به محدودیت ایجاد شده در انتقال کابلهای موتورها و محدودیتهای مکانیکی قادر نیستیم بصورت آزاد از این محورها استفاده کنیم). در شکل زیر تصویر نمونه ربات ساخته شده آزمایشی نشان داده شده است :





^ تصویر درایور چهار استپرموتور با استفاده از **AVR ATMEGA32** و ۳ عدد **ULN2003A**

موتورهای پله ای در نظر گرفته شده برای این منظور، از نوع چهار فاز تک قطبی می باشند، که در بسیاری از دیسک درایوهای قدیمی یافت می شوند. البته نمونهٔ ارزان قیمت و فراوان آن که در فروشگاه های داخل کـشور نیز یافت می شود مـدل KP4M4-001

ساخت Tandon themselves and Japan Servo Motors می باشد، در شکل زیر تصویر این موتور نشان داده شده است.

شکل (۳-۱) موتور پله ای KP4M4

شرح مشخصات این موتور در بخش (۲-۹) آمده است. از موتورهاي قابل استفاده ديگري كه داراي مشخصات فني يكسان با موتور مورد نظر ما باشد، مي توان مدلهاي زير را برشمرد: Minebea Co., Ltd - type 17PS-C007-04

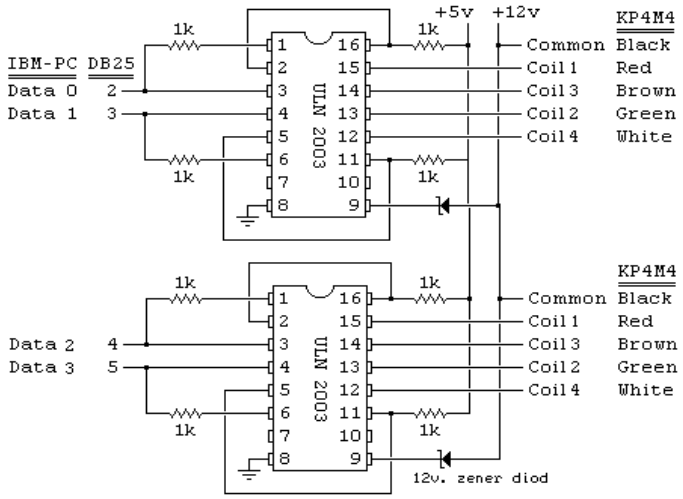
- Shinano Kenshi Co., Ltd - type STH-42G100

که اغلب دارای مشخصات زیر می باشند.

+12v dc, four-phase, unipolar, 3.6° per step

در ساخت نمونه آزمایشی از موتورهای مشابه برگرفته از پرینتر **HP660C** استفاده شده است.

در ساخت درایورهای مد نظر ما، از ULN2003 استفاده می شود که به تعداد لازم در کنارهم قرار می گیرند. در شکـل(۳-۲) نحوهٔ ساخت این درایورتنها برای دو موتور نشان داده شده است که قابل تعمیم به چهار موتور می باشد که با توجه به استفاده از میکرو Atmega32 جهت ارتباط واسطه ای با PC جهت کنترل موتورها بصورت تمام پله(یا نیم پله در صورت نیاز) پینهای ULN2003 بعنوان زوج دارلینگتونهای مورد نیاز جهت راه اندازی موتورها استفاده شده است .



شکل (۳-۲) درایور کنترل دوموتور پله ای تک قطبی چهارفاز KP4M4-001

البته می توان برای حفاظت بیشتر پورت پارالل کامپیوتر، در برابر خسارات احتمالی ناشی از اشتباهات یا ولتاژ برگشتی بالا، از مدارات ایزوله کننده که بهترین آنها opti-isolator است، استفاده کرد و سپس خروجی ایزولاتور را به مدار درایور متصل نمود.(در فصل اول بصورت کامل شرح داده شده است).

لازم به ذکر است که به منظور کاهش هزینه، درساخت درایور ربات از میکروکنترلر AVR استفاده شده است .

اکنون باید با توجه به نیازمان در پروژه، منبع تغذیه مناسبی تهیه کنیم. البته چنانچه مصرف انرژی مدارات و سیستم ساخته شده زیاد نباشد و وسایل جانبی زیادی به کامپیوترمان متصل نباشد، می توانیم با رعایت توان تولیدی پاور pc، از آن بعنوان منبع تغذیه استفاده کنیم.

بعد از تست صحت عملکرد مدارات کنترل و موتورهای پله ای که توسط دادن پالسهای دستی قابل انجام است (در صورت تمایل) نوبت به نوشتن برنامهٔ نرم افزاری پروژه می رسد. برای نوشتن برنامهٔ کنترل از زبان c استفاده می کنیم. البته در ضمیمهٔ ج نکات خوبی در مورد بکارگیری پورت پارالل در بعضی از زبانهای دیگر برنامه نویسی رایج آورده شده است که قابل استفاده خواهد بود.

در این برنامه برای حرکت به هر جهت، یک تابع خاص با نامهای متناسب نوشته شده است. یعنی تابع fw_variable( ) برای حرکت رو به جلو ، تابع bw_variable ( ) برای حرکت رو به عقب در نظر گرفته شده است.

توابع محاسباتی و کنترلی دیگری نیز نوشته شده است که بصورت اجمال شرح داده می شوند.

- تابع gotoxy( ) یکی از مفید ترین توابعی است که جهت قرار دادن مکان نما در مختصات مشخصی از یک پنجرهٔ فعال به کاربرده می شود . که در آنxوy مختصات نقطه ای هستند که مکان نما درآن قرار می گیرد.تابع فوق در پروندهٔconio.h تعریف می شود.
- تابع path( ) کنترل سرعت موتور ها را بعهده می گیرد.
- تابع Getkey$( ) بافر صفحه کـلید را می خواند (کد کلید فشرده شده را می خواند).
- تابع clrscr( ) پنجرهٔ فعال صفحه نمایش کامپیوتر راپاک می کند .

توابعی که جهت ورودی خروجی در نظر گرفته شده انددر پروندهٔ stdio.h قرار دارند.

توابعی همچون exit( ) , abs( ) , abort( ) و... در پروندهٔstdlib.h قرار دارند.

در این قسمت به منظور آشنایی بیشتر با میکروکنترلر AVR وطریقهٔ ساخت درایور ربات ونحوهٔ برنامه نویسی آن مطالبی آورده شده است .چون در ساخت درایور از میکروکنترلر Atmega 32 استفاده شده است توضیحات به این میکرو معطوف شده است .

۳-۱ معرفی میکروکنترلر AVR

به طور کلی میکروکنترلر ها برای کاربرد هایی که در آنها مشکل قیمت و حجم سیستم وجود دارد ،مناسب هستند .میکروکنترلر ها به طور گستردهای در تولید سیستم های تک منظوره به کار می روند .منظور از سیستم تک منظوره سیستمی است که از میکرو کنترلر فقط برای یک کار استفاده می کند ،مانند :پردازندهٔ درون موس که تنها به منظور یافتن مکان اشاره گر موس وارسال آن به PC برنامه ریزی شده است . چند نمونه از وسایلی که در ساخت آنها از میکروکنترلرها استفاده می شود عبارتند از : کنترل از راه دور تلویزیون،تلفن ،دوربین فیلم برداری ،فاکس ، چاپگر ، دستگاه فتوکپی ، سیستم های حفاظتی ، دزدگیر و سیستم های کنترل صنعتی .

میکروکنترلر های AVR از یک ساختار RISC بهبود یافته استفاده می کنند .دستور ها در این میکروکنترلر ها به گونه اتی طراحی شده است که حجم برنامه کوچک شود ،حتی اگر به زبان اسمبلی نوشته شده باشد . ساختارRISC با دستور های فراوان در این میکرو کنترلرها باعث کم شدن حجم برنامه و بالا رفتن سرعت می شود .ساختار بهینهٔ I/O در این میکروها باعث کاهش نیاز به افزودن اجزای خارجی می شود . میکروکنترلر های AVR دارای اسیلاتور داخلی ،تایمر ، UART ، SPI درون تراشه هستند .این میکروکنترلر ها دارای مقاومت PULL UP داخلی هستند همچنین دارای مبدل A/D ، مقایسه کنندهٔ آنالوگ ، تایمر WATCHDOG و قابلیت مدولاسیون عرض پالس هستند واز تکنولوژی حافظهٔ non-volatile وhigh density برخوردار هستند . حافظه های flash و EEPROM در این میکرو ها ،قابلیت برنامه ریزی شدن در داخل مدار را دارند (ISP) . حافظهٔ FLASH به دو روش در داخل مدار برنامه ریزی می گردد . یکی توسط پروگرمر حافظهٔ non- volatile و از طریق رابطهٔ سریال SPI و دیگری به وسیلهٔ اجرای boot program که بر روی تراشه موجود است . boot program می تواند از هر مدار واسطی برای ریختن برنامه بر روی حافظهٔ flash استفاده کند.

از نرم افزار های BASCOM (BASIC)، CODE VISION (C) می توان برای ارتباط کاربر و میکروکنترلر از طریق PC استفاده کرد که قابلیت شبیه سازی عملکرد میکرو وبرنامه ریزی روی میکرو را دارد .

۳-۲ خصوصیات ATMEGA 32

میکرو کنترلر atmega 32 کارایی بالا و توان مصرفی کمی دارد ،دارای ۱۳۱ دستورالعمل با کارایی بالا است که اکثراً تنها در یک کلاک سیکل اجرا می شوند و دارای ۳۲ رجیستر ۸ بیتی هستند .

۳۲ کیلو بایت حافظهٔ flash داخلی قابل برنامه ریزی دارد که پایداری آن تا ۱۰۰۰۰ بار قابلیت نوشتن و پاک کردن دارد و 2k بایت حافظهٔ داخلی sram دارد و 1024 بایت حافظهٔ eeprom داخلی قابل برنامه ریزی دارد که پایداری آن تا 10000 بار قابلیت نوشتن و پاک کردن دارد .و قابلیت قفل کردن برنامهٔ حافظهٔ flash وجود دارد .

قابل برنامه ریزی برنامهٔ flash ، eeprom ، fuse bits ، lock bits از طریق ارتباط JTAG وجود دارد .

دارای دو تایمر –کانتر (timer/counter) 8 بیتی با prescaler مجزا و دارای مد compare است .همچنین دارای یک تایمر –کانتر 16 بیتی با prescaler مجزا و دارای مد های compareوcopture است .

4 کانال pwm دارد و 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی دارد و دارای دو کانال تفاضلی با کنترل گین 200x, 10x, 1x است و یک مقایسه کنندهٔ آنالوگ داخلی دارد .

دارای RTC (real-time clock)با اسیلاتور مجزا است .

Watchdog قابل برنامه ریزی با اسیلاتور داخلی در میکرو کنترلر atmega 32 وجود دارد .

ارتباط سریال SPI برای برنامه ریزی داخل مدار و همچنین قابلیت ارتباط سریال SPI به صورت master یا slave و همچنین قابلیت ارتباط با پروتکل سریال دو سیمه (two wire) و همچنین دارای USART سریال قابل برنامه ریزی است .

این میکرو کنترلر دارای شش حالت sleep (ADC NOISE REDUCTION , EXTENDED STANDBY , STANDBY , POWER SAVE , IDLE , POWER DOWN) می باشد .

دارای منابع وقفه (INTERRUPT) داخلی و خارجی است و دارای اسیلاتور RC داخلی کالیبره شده است .

برای ATMEGA 32L ولتاژهای عملیاتی از 2.7 v تا 5.5 v است و فرکانس کاری از 0mhz تا 8mhz و برای ATMEGA32 ولتاژ کاری از 4.5 V تا 5.5 Vاست و فرکانس کاری از 0MHZ تا 16MHZ است . 32 خط ورودی /خروجی (I/O) قابل برنامه ریزی دارد و 40 پایه است .

۳-۳ معرفی مختصر کامپایلر BASCOM

کامپایلر bascom تمام میکرو های AVR را حمایت کرده و از زبان BASIC برای برنامه نویسی AVR ها استفاده می کند . یکی از قابلیت های بسیار ارزندهٔ محیط

BASCOM داشتن تحلیل گر یا به عبارتی SIMULATOR داخلی است که برای یادگیری برنامه نویسی AVR بسیار کار آمد است .
ورودی سیگنال آنالوگ ADC و مقایسه کنند آنالوگ ، ایجاد پالس بر روی پایه ای خاص ، صفحه کلید 4×4 ، LCD ،ایجاد تمام وقفه ها بصورت اختیاری ،نوشتن و خواندن حافظهٔ EEPROM و SRAM ، رؤیت تمام رجیستر ها و متغیر های محلی و سراسری برنامه ،اجرای برنامه به صورت خط به خط ، رؤیت صفر یا یک بودن تمام پایه های میکرو توسط LED ، تغییر منطق پایهٔ دلخواه و بسیاری امکانات دیگر توسط محیط تحلیل گر (SIMULATOR) و از همه مهمتر برنامه نویسی ساده باعث شده است که این کامپایلر در کنار دیگر کامپایلر های معروف مورد تایید واستفادهٔ برنامه نویسان قرار گیرد .
پس از اجرای برنامهٔ bascom پنجرهٔ محیط برنامه نویسی ظاهر می شود .که دارای منوهای file ، منوی program ، منوی tools و منوی options است که بعد نوشتن برنامه با استفاده از program compile برنامهٔ نوشته شده را در صورت نداشتن error کامپایل می کنیم اما اگر برنامهٔ ما error داشته باشد ابتدا باید آن خطا ها را رفع کنیم و بعد از کامپایل شدن با استفاده از کلید simulator وارد محیط شبیه سازی bascom می شویم که با فشار دادن دکمهٔ RUN شبیه سازی آغاز می شود .

۳-٤ استفاده از ATMEGA 32 به عنوان درایور یک STEPPERMOTOR

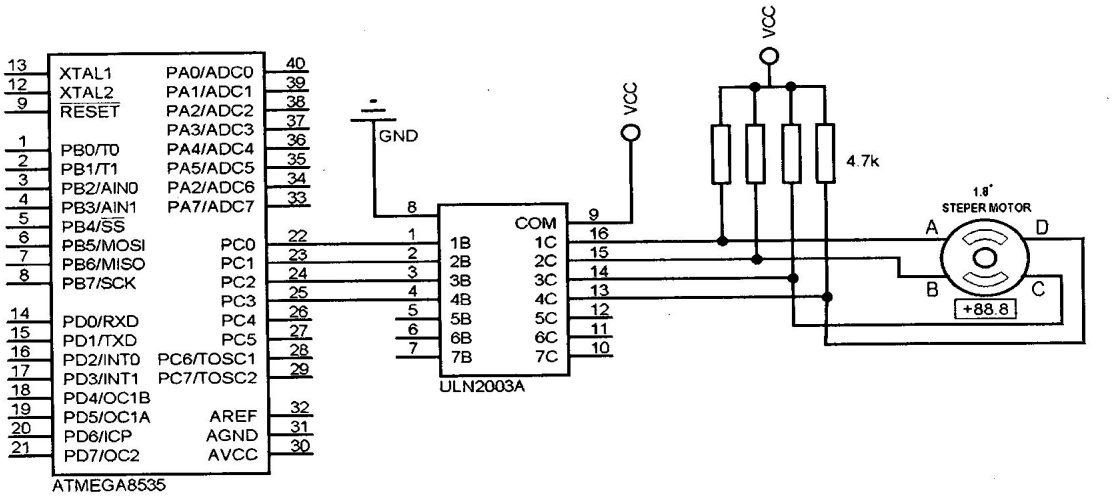
برای راه اندازی موتور پله ای می توان از راه انداز ULN2003 استفاده کرد ،این درایور 16 پایه دارد که دارای 7 پایه راه اندار کلکتور باز است و بیشترین جریان خروجی آن برای هر پایه 500mA می باشد ، اگر از موتور پله ای چهار سیم استفاده کنیم با هر ULN2003 یک موتور پله ای می تواند درایو شود ، در نتیجه با دو تا ULN2003 می توان سه موتور پله ای را درایو کرد که با اعمال رشته هایی از تغذیه یا پالس به هر سیم پیچ استاتور ، روتور خواهد چرخید .

خلاف جهت ساعت	سیم پیچ D	سیم پیچ C	سیم پیچ B	سیم پیچ A	درجهت ساعت
↑	0	0	0	1	
	0	0	1	0	
	0	1	0	0	↓
	1	0	0	0	

جدول رشتهٔ چهار پله (پلهٔ کامل)

تحریک نیم پله موجب می شود که دقت موتور دو برابر شود در این حالت با توجه به موتور استفاده شده نیاز به 400 پالس می باشد .

با استفاده از برنامهٔ زیر یک موتور پله ای با دقت 1.8 درجه را بطور پیوسته یک بار پله کامل و بار دیگر به صورت نیم پله به اندازهٔ 360 درجه می چرخوانیم .



شکل مدار بسته شده برای برنامه موتور پله ای

شکل(۳-۳)

برنامهٔ زیر به زبان bascom (basic) نوشته شده است .

```
$regfile = "m32def.dat"
$crystal = 8000000
Config Portc = Output
Dim A As Byte , B As Byte , E As Byte , F As Byte
Do
```

```
For A = 1 To 50
       E = 128
    For B = 1 To 4
       Rotate E , Left
          Portc = E
          Waitms 20
    Next B
Next A
Wait 1
For A = 1 To 50
       F = 129
       E = 128
     For B = 1 To 4
          Rotate E , Left
          Portc = E
          Waitms 20
          Rotate F , Left
          If F = 24 Then F = 9
          Portc = F
          Waitms 20
     Next B
Next A
Loop
End
'end program
```

۳-۵ استفاده از میکروکنترلر ATMEGA32 به عنوان درایور چهار محور ربات (چهار موتور)

```
$regfile = "m32def.dat"
$crystal = 8000000
'$baud = 19600
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Config Porta = Input
Config Portb = Output
Config Portc = Output
Config Pind.7 = Output                                          'for PC
acknowledge
' sub functions decleration-----------------------------------------
Declare Sub Movex
Declare Sub Movey
Declare Sub Movez
Declare Sub Movew
'''''''decleration variables ---------------------------------
------
Dim Portxy As Byte
Dim Portzw As Byte
Portxy = 0
Portzw = 0

Dim Dirx As Bit , Diry As Bit , Dirz As Bit , Dirw As Bit
Dim Px As Bit , Py As Bit , Pz As Bit , Pw As Bit
Dim Plx As Bit , Ply As Bit , Plz As Bit , Plw As Bit
Dim Rotationx As Byte , Rotationy As Byte , Rotationz As Byte ,
Rotationw As Byte
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''
' initializing last pulses of X, Y , Z , W
Set Plx : Set Ply : Set Plz : Set Plw
'initializing the direction of 4 axes to rotation to left
Reset Dirx : Reset Diry : Reset Dirz : Reset Dirw
' initializing the buffers of state of each axes befor writing
them on the related port
Rotationx = 1
Rotationy = 16
Rotationz = 1
Rotationw = 16
Portb = 17
Portc = 17
' M      A        I        N
Do
'Reading the status of port A and assigning it to the variables

Dirx = Pina.1
Diry = Pina.3
Dirz = Pina.5
Dirw = Pina.7
Px = Pina.0
Py = Pina.2
Pz = Pina.4
Pw = Pina.6

If Px = 0 And Plx = 1 Then Call Movex
Plx = Px

If Py = 0 And Ply = 1 Then Call Movey
Ply = Py

If Pz = 0 And Plz = 1 Then Call Movez
Plz = Pz

If Pw = 0 And Plw = 1 Then Call Movew
Plw = Pw

'acknowledge for parallel port of PC for declaring that the
command has done
Portd.7 = 1
Waitms 1
Portd.7 = 0
Loop
End                                                          'end
progrm



Sub Movex
If Dirx = 0 Then Shift Rotationx , Left , 1
If Dirx = 1 Then Shift Rotationx , Right , 1

If Rotationx = 0 Then Rotationx = 8
If Rotationx = 16 Then Rotationx = 1

Rotationx = Rotationx And &B00001111
```

```
Portxy = Portxy And &B11110000
Portxy = Portxy Or Rotationx

Portb = Portxy
End Sub Movex

Sub Movey
If Diry = 0 Then Shift Rotationy , Left , 1
If Diry = 1 Then Shift Rotationy , Right , 1
If Rotationy = 8 Then Rotationy = 128
If Rotationy = 0 Then Rotationy = 16
Rotationy = Rotationy And &B11110000

Portxy = Portxy And &B00001111
Portxy = Portxy Or Rotationy

Portb = Portxy
End Sub Movey

Sub Movez
If Dirz = 0 Then Shift Rotationz , Left , 1
 If Dirz = 1 Then Shift Rotationz , Right , 1
If Rotationz = 0 Then Rotationz = 8
If Rotationz = 16 Then Rotationz = 1
Rotationz = Rotationz And &B00001111

Portzw = Portzw And &B11110000
Portzw = Portzw Or Rotationz

Portc = Portzw
End Sub Movez

Sub Movew
If Dirw = 0 Then Shift Rotationw , Left , 1
If Dirw = 1 Then Shift Rotationw , Right , 1
If Rotationw = 8 Then Rotationw = 128
If Rotationw = 0 Then Rotationw = 16
Rotationw = Rotationw And &B11110000

Portzw = Portzw And &B00001111
Portzw = Portzw Or Rotationw
Portc = Portzw
End Sub Movew
```

۳-٦ استفاده از کامپایلر C++ در برنامه نویسی پورت پارالل

برنامه زیر چگونگی نوشتن اطلاعات بر روی پورت پارالل را نشان می دهد:

```
#include <stdio.h>
#include <sys/plp.h>
#include <fcntl.h>
#include <unistd.h>


/*
 * Name    : portdemo.c
```

```
 * Machines: IRIS 4D/30, 4D/35, 4D/RPC, 4D/RPC-50, Indy, Indigo2 and
 *           Challenge/Onyx. (only tested on an Indy so far).
 * Purpose : Demonstrate how to write to the parallel port.
 * Author  : James Ward
 * Contact : j.w.ward@dcs.hull.ac.uk
 *           http://www.enc.hull.ac.uk/~cssjww
 * Date    : 29/05/96 - Created.
 * Changes : 30/05/96 - Added more info to note 1, describing (roughly)
what
 *                      happens when you write to the rewired port.
(JWW)
 *
 * Distribute freely, use at your own risk.
 * This setup has worked for several months on our Indy without any
problems.
 *
 */


/* Notes:
 *      1. You can turn the parallel port into a simple eight-bit
latched
 *         output port (ie. eight TTL lines that you can control from
software)
 *         by simply wiring /STROBE (pin 1) to /ACKNOWLEDGE (pin 10),
and
 *         linking BUSY (pin 11) to GROUND (pins 19..25). Values
written to the
 *         port will then appear on D1..D8 (pins 2..9) and will remain
there
 *         until you write to the port again. Very useful for hardware
projects!
 *         See the plp man page for a full connector pinout.
 *
 *         How this works:
 *              1. Indy wants to write data to the port, so it looks for
the
 *                 BUSY signal. We have tied this low (ie. to ground) so
that
 *                 the Indy never gets a BUSY signal and doesn't wait
for one.
 *              2. The Indy places the data on pins D1..D8, then pulses
/STROBE
 *                 low (it is normally high), to indicate that valid
data is
 *                 available on D1..D8.
 *              3. Normally, the Indy would expect a /ACKNOWLEDGE signal
but,
 *                 since we have tied /ACKNOWLEDGE to /STROBE, the
strobe signal
 *                 automatically acknowledges the transfer - we can
write to the
 *                 port freely, without any delays.
 *
 *
 *      2. There are ioctls that allow you to alter read/write timeout
and to
 *         change the length of the strobe pulses that the port
generates.
 *         If you are using a simple setup like that described above,
you
 *         probably don't need to use the ioctls.
 *         More details are available on the plp man page, and I have
examples
```

```
 *         of using these controls if you get stuck.
 *
 *      3. This program writes the values 0..9 to the parallel port,
with a
 *         short delay after each write. Here is what you should see
(providing
 *         you have some hardware plugged in to the port, or the port
is
 *         connected as described in note 1.

            > cc -o portdemo portdemo.c                          ....
compile this file
            > portdemo                                    .... run it!
            Success: opened parallel port for writing
            Writing 0 ... Done.
            Writing 1 ... Done.
            Writing 2 ... Done.
            Writing 3 ... Done.
            Writing 4 ... Done.
            Writing 5 ... Done.
            Writing 6 ... Done.
            Writing 7 ... Done.
            Writing 8 ... Done.
            Writing 9 ... Done.
            >
 */


int theport;    /* File descriptor returned for printer device */


/* This function opens the parallel port for writing, it is a
prerequisite for
 * using the port. Returns 1 if the port was opened successfully, or 0
if the
 * operation failed.
 */

int openDevice (void) {
    theport = open ("/dev/plp", O_WRONLY);  /* Open printer port for
writes    */
    return ((theport >= 0) ? 1:0);        /* Return 1 (success) or 0
(fail) */
}


/* This function closes the parallel port when you have finished using
it */

void closeDevice (void) {
    close (theport);
}


/* Asserts a reset signal on the parallel port. ie. the reset line is
normally
 * high. When you call this function, the RESET signal will pulse low
briefly.
 * This could be useful for resetting external hardware, or it could
even be
 * used as some sort of handshake.
 */

int resetDevice (void) {
```

```
    return (ioctl (theport, PLPIOCRESET, 0) == 0) ? 1:0; /* 1=success,
0=fail */
}



/* This writes a single byte to the parallel port, returning one for
success,
 * or 0 for an error condition.
 */

int writeByte (char xbyte) {
    return (write (theport, &xbyte, 1) == 1) ? 1:0;
}


/* Main program... */

void main (void) {
    int n;

    if (openDevice()) {           /* Open the parallel port */
        printf("Success: opened parallel port for writing\n");

        for (n=0;n
```

۳-۷ برنامه کنترل ربات نوشته شده تحت کامپایلر C++

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#define DATA 0x3bc
#define X 11640
#define Y 5497
#define Z 11386
#define W 4471
#define SHIFT_X 11608
#define SHIFT_Y 5465
#define SHIFT_Z 11354
#define SHIFT_W 4439
#define  Esc 283
#define UP 18432
#define DOWN 20480
#define RIGHT 19712
#define LEFT 19200
#define SPACE 14624
int step=37;
int data= 0x00;/*note global */
int x=1;
int a;

int get_key();
void bw_x(void);
void fw_x(void);
void bw_y(void);
void fw_y(void);
void bw_z(void);
void fw_z(void);
void bw_w(void);
```

```
void fw_w(void);
void MOVE_XYZ(int direction);
void path(void);

void main(void)
{
 textcolor(10);
 textbackground(9);
 clrscr();
 printf("    THIS PROGRAM IS WRITEN BY Fatemeh Shafinia Gat-Abi");
 printf("\n\n\n                    SIMPLE    ROBOT     CONTROLER
");
 printf("\n         Press X or Y or Z or W with/without SHIFT to play
the motors");
 printf("\n   Press arrow key UP or DOWN to set speed of motors
");
 printf("\n   Press arrow key RIGHT or LEFT to control motors by
step_count");
 printf("\n   Press SPACE to Enter path ");


 while(a!=-1)
  {
 a=get_key();
// int *ptr;
// ptr=&a;
 gotoxy(36,12);
    switch(a)
  {
  case Esc:  exit(0); break;
  case X: printf("MOVING ON  X"); fw_x();
      gotoxy(36,12); printf("            ");   break;
  case Y: printf("MOVING ON  Y"); fw_y();
      gotoxy(36,12); printf("            ");  break;
  case Z: printf("MOVING ON  Z"); fw_z();
      gotoxy(36,12); printf("            ");  break;
  case W: printf("MOVING ON  W"); fw_w();
      gotoxy(36,12); printf("            ");  break;
  case SHIFT_X: printf("MOVING ON -X"); bw_x();
      gotoxy(36,12); printf("            ");  break;
  case SHIFT_Y: printf("MOVING ON -Y"); bw_y();
      gotoxy(36,12); printf("            ");  break;
  case SHIFT_Z: printf("MOVING ON -Z"); bw_z();
      gotoxy(36,12); printf("            ");  break;
  case SHIFT_W: printf("MOVING ON -W"); bw_w();
      gotoxy(36,12); printf("            ");  break;
  case UP : if (step<37) step++; gotoxy(36,12);
       cprintf("step * %d ",step);            break;
  case DOWN : if (step>1) step--;gotoxy(36,12);
       cprintf("step * %d ",step);            break;
  case RIGHT : MOVE_XYZ(1); break;
  case LEFT :  MOVE_XYZ(-1); break;
  case SPACE : path(); break;
   } //end of switch
  }// end of while
} //end of program

int get_key()
  {
  union REGS r;
  r.h.ah=0;
 return int86(0x16,&r,&r);
 }
```

```
void fw_x(void)// forward of motor X
{
   for (int i=0; i<step; i++)
     {

    outportb (DATA,0x01); delay(x);
    outportb (DATA,0x00); delay(x);
    }
}
void bw_x(void)// backward of motor X
{
     for (int i=0; i<step; i++)
     {

      outportb (DATA,0x03); delay(x);
      outportb (DATA,0x02); delay(x);
      }
 }

void fw_y(void)// forward of motor Y
{
   for (int i=0; i<step; i++)
     {

    outportb (DATA,0x04); delay(x);
    outportb (DATA,0x00); delay(x);
    }
}
void bw_y(void)// backward of motor Y
{
     for (int i=0; i<step; i++)
     {

      outportb (DATA,0x0c); delay(x);
      outportb (DATA,0x08); delay(x);
      }
 }

 void fw_z(void)// forward of motor Z
{
   for (int i=0; i<step; i++)
     {

    outportb (DATA,0x10); delay(x);
    outportb (DATA,0x00); delay(x);
    }
}
void bw_z(void)// backward of motor Z
{
     for (int i=0; i<step; i++)
     {

      outportb (DATA,0x30); delay(x);
      outportb (DATA,0x20); delay(x);
      }
 }

  void fw_w(void)// forward of motor w
{
   for (int i=0; i<step; i++)
     {

    outportb (DATA,0x40); delay(x);
    outportb (DATA,0x00); delay(x);
```

```
    }
}
void bw_w(void)// backward of motor w
{
     for (int i=0; i<step; i++)
     {

      outportb (DATA,0xc0); delay(x);
      outportb (DATA,0x80); delay(x);
      }
 }


 void MOVE_XYZ(int direction)
  {
  int step_temp=step;//to memorize of the original step count
  int x_temp=x;//to memorize of the original delay value
  x=1;
  char motor;
  gotoxy(16,12);
  cprintf("Press X ,Y ,Z or W to select the motor:\>  ");
  motor = getch();
  gotoxy(16,12);
  cprintf("  How many step do you want? :\>           ");
  scanf("%d",&step);
  gotoxy(16,12);
  cprintf("
");
  if (direction==1)
    {
    if (motor =='X'||motor=='x') fw_x();
    if (motor =='Y'||motor=='y') fw_y();
    if (motor =='Z'||motor=='z') fw_z();
    if (motor =='W'||motor=='w') fw_w();
    }
  else
    {
    if (motor =='X'||motor=='x') bw_x();
    if (motor =='Y'||motor=='y') bw_y();
    if (motor =='Z'||motor=='z') bw_z();
    if (motor =='W'||motor=='w') bw_w();
    }
 step=step_temp;// to remember of original step
 x=x_temp; //  to remember of original x

  }// end of sub MOVE_XYZ

void path(void)
{
 int step_array[50]={0};
 char xyz_array[50]={' '};
 int i=0;
 int step_temp=step;//to memorize of the original step count
 int x_temp=x;//to memorize of the original delay value
 x=1;// global delay

 do{
 gotoxy(20,16);
 printf("Enter motor name (X, Y or Z)  :\>");
 xyz_array[i]=getch();
 gotoxy(1,16);
 printf("Enter number of steps like 320 or -410 or ENTER 0 to end of
path :\>");
 scanf("%d",&step_array[i++]);
```

```
 gotoxy(1,16);
 printf("
");
 }
 while(step_array[i-1]!=0);

 for (int n=0; n<i; n++)
   {
   if (xyz_array[n]=='x'|| xyz_array[n]=='X')//detecting motor
       if (step_array[n]>0)
       {step=step_array[n];
      fw_x();
       }
      else if(step_array[n]<0)
            {step=step_array[n];
             step= step* -1;
             bw_x();
             }
    if(xyz_array[n]=='y'|| xyz_array[n]=='Y')//detecting motor
       if (step_array[n]>0)
       {step=step_array[n];
      fw_y();
       }
      else if(step_array[n]<0)
            {step=step_array[n];
             step= step* -1;
             bw_y();
             }
    if(xyz_array[n]=='z'|| xyz_array[n]=='Z')//detecting motor
       if (step_array[n]>0)
       {step=step_array[n];
      fw_z();
       }
      else if(step_array[n]<0)
            {step=step_array[n];
             step= step* -1;
             bw_z();
          }
      }
step=step_temp;// to remember of original step
x=x_temp; //  to remember of original x
} //end of sub path


 /*
void xyz(int X,int Y,int Z,int SX,int SY,int SZ)
{
    for (int i=0; i<400; i++)
     {

    outportb (DATA,0x01); delay(x);
    outportb (DATA,0x00); delay(x);
    }
}
 */
```

بعد از تست عملکرد صحیح برنامه با موتورها (کل سیستم)، نوبت به سرهم کردن اجزاء مي رسد.

مراحل انجام این پروژه و امتحان عملکرد آن در همینجا پایان می یابد، و از آنچه که ساخته شد می توان بعنوان یک نمونهٔ آزمایشگاهی یا کاربردی استفاده کرد. همچنین از

تجربیات این پروژه می توان در ساختن ابزارهای آزمایشگاهی و کاربردی دیگر بهره برد. کاربرد چنین سیستمی، استفاده از آن در یک وسیلهٔ رباتیک و یا دستگاه های صنعتی مانند انواع cnc ها می باشد

لازم به ذکر است، در این پروژه موتورها بصورت open loop بکار برده شده اند، و چنانچه در پروژه های حساستر نیاز به اطمینان از از انجام دستورات داده شده احساس گردد، می توان موتورها را بصورت close loop متصل کرد، و با استفاده از میکروسوئیچ یا انکودر محور، موقعیت واقعی محورهای موتور را از طریق پایه های status پورت پارالل که در فصل اول بطور مفصل شرح داده شده، پردازش کرد و عکس العمل مناسب را اعمال نمود.

پایان

ضمیمه الف
(مربوط به پورت پارالل)

در این قسمت بعضی از نکات پیرامون پورت پارالل، برگرفته از انتشارات موسسه علمی IEEE آورده شده است :

Introduction to the IEEE 1284-1994 Standard

This section is implemented as a multilevel document. This page serves as an executive summary of the 1284 standard. By clicking on the various highlighted points, you may explore each concept in greater detail.

The recently released standard, *"IEEE Std.1284-1994 Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers"*, is for the parallel port what the Pentium processor is to the 286. The standard provides for high speed bi-directional communication between the PC and an external peripheral that can communicate 50 to 100 times faster than the original parallel port. It can do this and still be fully backward compatible with all existing parallel port peripherals and printers.

The 1284 standard defines 5 modes of data transfer. Each mode provides a method of transferring data in either the forward direction (PC to peripheral), reverse direction (peripheral to PC) or bi-directional data transfer (half duplex). The defined modes are:

- **Forward direction only**

  Compatibility Mode
  "Centronics" or standard mode

- **Reverse direction only**

  Nibble Mode
  4 bits at a time using status lines for data.
  Hewlett Packard Bi-tronics
  Byte Mode
  8 bits at a time using data lines, sometimes referred to as a "bi-directional" port.

- **Bi-directional**

  EPP
  Enhanced Parallel Port- used primarily by non-printer peripherals, CD ROM, tape, hard drive, network adapters, etc....
  ECP
  Extended Capability Port- used primarily by new generation of printers and scanners

All parallel ports can implement a bi-directional link by using the Compatible and Nibble modes for data transfer. Byte mode can be utilized by about 25% of the installed base of parallel ports. All three of these modes utilize software only to transfer the data. The driver has to write the data, check the handshake lines (i.e.: BUSY), assert the appropriate control signals (i.e.: STROBE) and then go on to the next byte. This is very software intensive and limits the effective data transfer rate to 50 to 100 Kbytes per second.

In addition to the previous 3 modes, EPP and ECP are being implemented on the latest I/O controllers by most of the Super I/O chip manufacturers. These modes use hardware to assist in the data transfer. For example, in EPP mode, a byte of data can be transferred to the peripheral by a simple OUT instruction. The I/O controller handles all the handshaking and data transfer to the peripheral.

Overall, the 1284 standard provides the following:

1. 5 modes of operation for data transfer
2. A method for the host and peripheral to determine the supported modes and to negotiate to the requested mode.
3. Defines the physical interface
   - Cables
   - Connectors
4. Defines the electrical interface
   - Drivers/Receivers
   - Termination
   - Impedance

In summary, the 1284 parallel port provides an easy to use, high performance interface for portable products and printers.

IEEE 1284 Connectors

The 1284 standard goes beyond describing new data transfer modes and actually defines the mechanical interface and the electrical properties of a compliant parallel port. Many of the problems associated with parallel port-attached devices arise from the fact that there has been no standard for the electrical interface for the parallel port. The DB25 female connector has become standard for the PC or host connector, but there have been many different implementations of the drivers, resistors, capacitors, etc, for electrical the interface.

The 1284 committee felt that it was paramount to define what these properties should be in order to meet the following objectives:

1. Ensure electrical and mechanical compatibility among all 1284 compliant devices.
2. Ensure that 1284 interfaces would operate with existing parallel port peripherals and adapters.
3. Ensure operation and data integrity at the highest data rates
4. Extend operation to 10M (30')

To meet these objectives, the standard defines the connectors, electrical interface, and cable requirements.

1284 Connectors

The standard identifies three types of connectors for a 1284 interface.

1284 Type A
25 pin DB25
1284 Type B
36 Conductor, .085 centerline Champ connector with bale locks
1284 Type C
36 conductor, .050 centerline mini connector with clip latches

Figure 1 shows what these connectors look like.

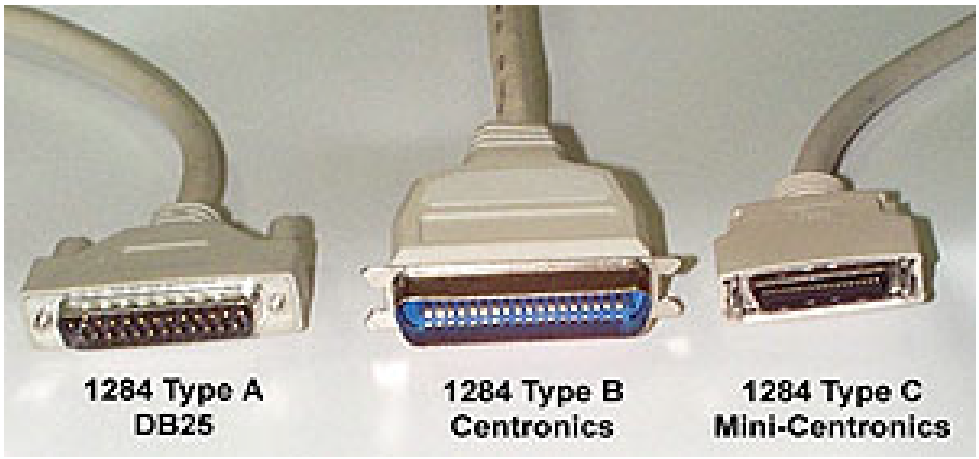


Figure 1 -- 1284 Interface I/O Connectors

The type C connector is the one recommended for new designs. This connector offers a smaller footprint than the previous connectors, has a simple-to-use clip latch for cable retention, and provides for the easiest cable assembly with the optimal electrical properties. In addition, a cable assembly built with this connector provides for two more signals. These signals are Peripheral Logic High and Host Logic High. These signals may be used to determine if the device at the other end of the cable is powered on. This enables some degree of intelligent power management for 1284 interfaces.

IEEE 1284 Electrical Interface

The original parallel port did not have a defined electrical specification that identified the driver, receiver, termination and capacitance requirements in order to guarantee any compatibility between devices. Host adapters and peripherals were built with any number of pull-up values on the control lines, open collector or totem pole drivers for the data and control lines, and most offensive of all, up to 10,000pF capacitors on the data and strobe lines. This type of design

variation makes it impossible to create a new interface protocol without explicitly defining the required electrical parameters with which to guarantee operation.

The 1284 standard defines two levels of interface compatibility, Level I and Level II. The Level I interface is defined for products that are not going to operate at the high speed advanced modes, but need to take advantage of the reverse channel capabilities of the standard. The Level II interface is for devices that will operate in the advanced modes, with long cables, and at the higher data rates. This discussion will deal primarily with Level II interfaces. Please refer to the standard for the full requirements for either a Level I or Level II interface.

The requirements for the Level II drivers and receivers are defined at the connector interface. The driver requirements are:

1. The open circuit high-level output voltage shall not exceed +5.5V.
2. The open circuit low-level output voltage shall be no less than -0.5V.
3. The DC steady state, high-level output voltage shall be at least +2.4V at a source current of 14mA.
4. The DC steady state, low-level output voltage shall not exceed +0.4V at a sink current of 14mA.
5. The driver output impedance (Ro), measured at the connector, shall be 50 +/- 5 ohms at 1/2 the actual driver Voh minus Vol voltage.
6. The driver slew rate shall be 0.05-0.40 V/nS

Like the driver requirements, the receiver requirements are defined at the connector interface. The receiver requirements are:

1. The receiver shall withstand peak input voltage transients between -2.0V and +7.0V without damage or improper operation.
2. The receiver high-level input threshold shall not exceed 2.0V
3. The receiver low-level input threshold shall be at least 0.8V.
4. The receiver shall provide at least 0.2V input hysteresis, but not more than 1.2V.
5. The receiver high-level sink current shall not exceed 20uA at +2.0V.
6. The receiver low-level input source current shall not exceed 20uA at +0.8V.
7. Circuit and stray capacitance shall not exceed 50pF.

Figure 1 shows the recommend termination for a driver/receiver pair. Ro represents the output impedance at the connector. It is intended that this impedance match the cable impedance so as to minimize the noise caused by mismatched impedances. Depending upon the type of driver used, a series resistor, Rs may be required to obtain the correct impedance.

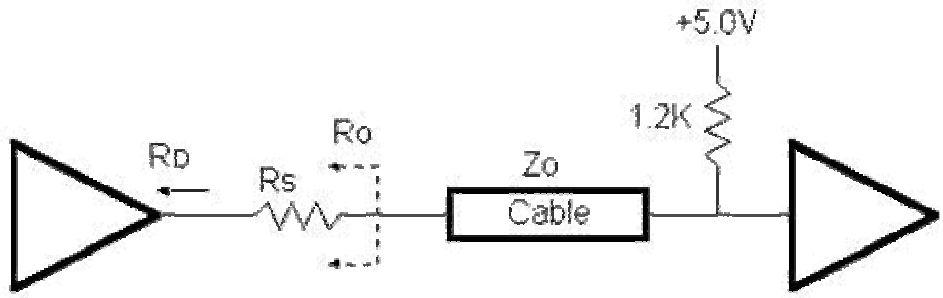


Figure 1 -- Level II Driver/Receiver Pair Termination Example

Figure 2 shows the recommended termination for a Level II transceiver pair, such as the data lines.

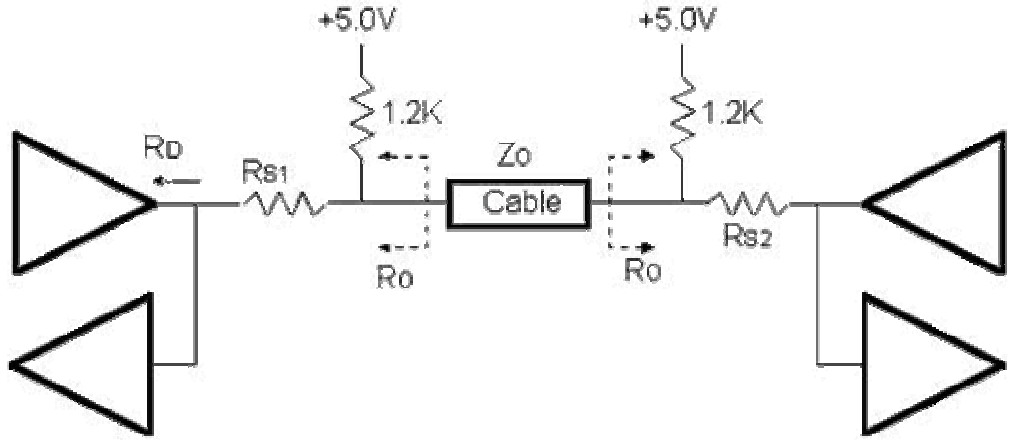


Figure 2 -- Level II Transceiver Termination Example

There are products being introduced by companies such as Texas Instruments and National that provide integrated solutions for a 1284 Level II interface. These include active drivers and receivers as well as resister sip networks.

NOTE: When ECP was first introduced Microsoft made a recommendation for an electrical and termination requirement that was not consistent with the 1284 specification. This included an AC terminator for each of the lines. This suggestion has since been retracted and the current recommendation is to use the interface defined in the IEEE 1284 specification.

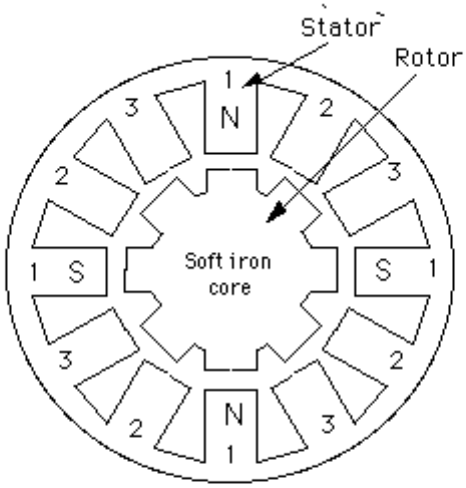
در این قسمت برنامه ای آورده شده که می تواند یک موتور پله ای را در یک جهت معین بچرخاند:

```
' Program STEP.TIG
'
' Illustrates how to turn a stepping motor in one direction.  Uses an
' array of patterns and advances the index through the array.

' Copyright, Krystal Donald, Morgan State University, Oct 21, '97



   ARRAY STEP_PATTERN (8) OF BYTE   'DECLARATION OF A GLOBAL ARRAY
TASK MAIN                           'STARTS MAIN
   DIR_PORT 8,0                     'SETS PORT 8 TO OUTPUTS
   BYTE STEP_INDEX                  'DECLARE BARIBLE STEP_INDEX

   BYTE A
   BYTE B
   STEP_INDEX =0B                   'THE FOLOWNG LINES ARE VALUES FOR
   STEP_PATTERN (0)=  01B           'EACH SPACE OF THE ARRAY
  STEP_PATTERN
   STEP_PATTERN (1)= 11B
   STEP_PATTERN (2)= 10B
   STEP_PATTERN (3)= 110B
   STEP_PATTERN (4)= 100B
   STEP_PATTERN (5)= 1100B
   STEP_PATTERN (6)= 1000B
   STEP_PATTERN (7)= 1001B

   B=1                             'B IS EQUAL TO ONE
   WHILE B = 1                     'RUN FOREVER
   A=STEP_PATTERN(STEP_INDEX)      'A=STEP_PATTERN(POSITION)
   LL_IPORT_OUT 8,A                'OUTPORT VALUE OF A
   STEP_INDEX = STEP_INDEX +1      'INCREMENT POSITION
   IF STEP_INDEX = 8D THEN       'IF IN LAST POSITION THEN
      STEP_INDEX = 0B               'START OVER
   ENDIF                           'END IF STATEMENT
   WAIT_DURATION  50               'DELAY
   ENDWHILE                        'END WHILE LOOP
END                                 'END PROGRAM
```

ضمیمه ب
(مربوط به موتورهای پله ای)

در این قسمت دیاگرام یک موتور پله ای سه فاز و بعضی مطالب مربوطه آورده شده است:

Figure 1. Pole layout for 3-phase stepper motor



Step	Orange	Red	Brown
1	GND		
2		GND	
3			GND

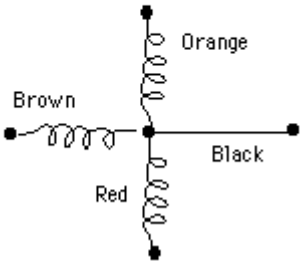
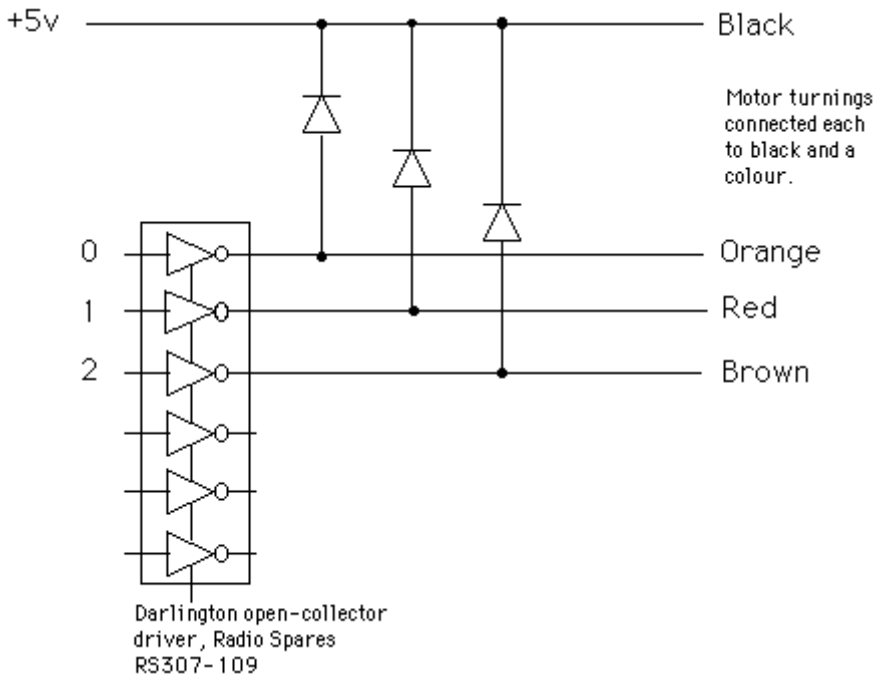
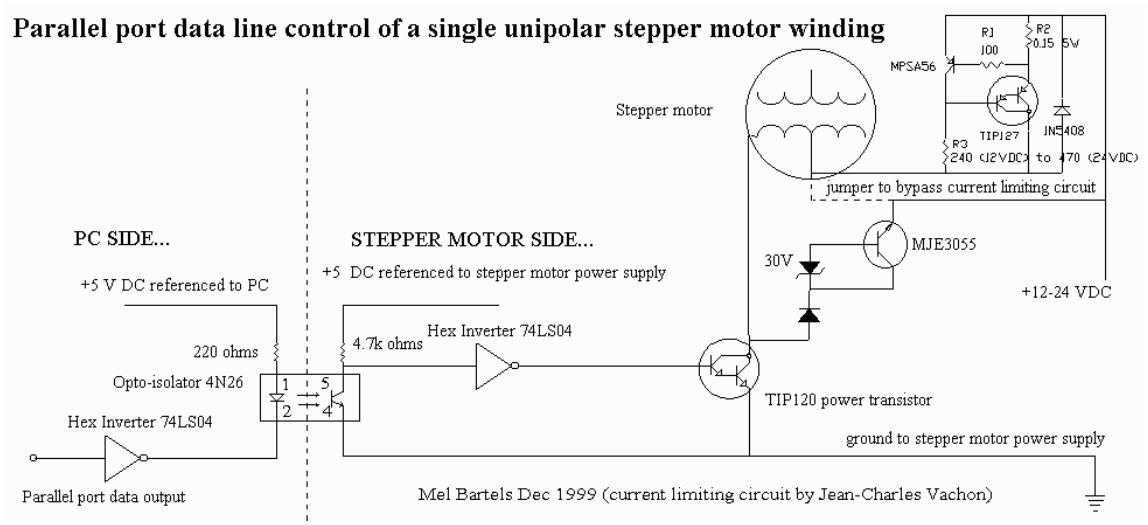
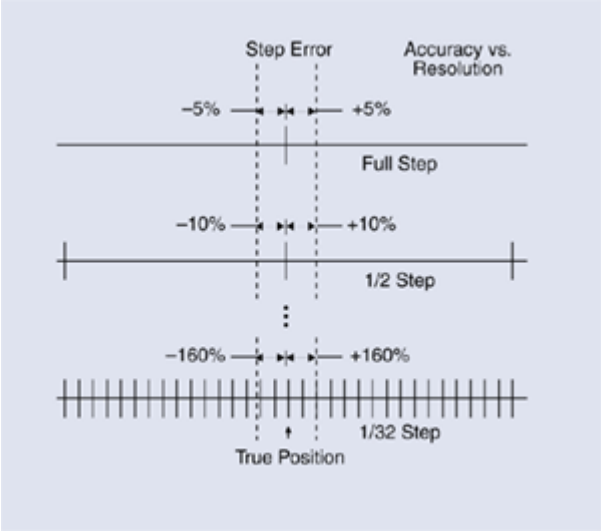


Figure 2. Connections for a three phase motor



در شکل زیر تصویر مداری را می بینید که، با استفاده از روش حفاظت نوری (opti-isolation) ، پورت پارالل را به ترانزیستور قدرتی که یک سیم پیچ موتور پله ای تک قطبی را راه اندازی می کند، متصل کرده است. برای استفاده از این روش باید به تعداد سیم پیچ های موتور، از این مدار ساخته و استفاده شود.





شکل (۲-۱۳)

Microstepping will improve stepper-motor resolution but not accuracy. A ±5% absolute step error becomes a ±160% error at 32nd-step microstepping.

برای کسب اطلاعات بیشتر در خصوص ارتباط لحظهٔ خیزجریان (Current Risetime) و گشتاور، همچنین بعضی از پارامترهای مفید دیگر به بخش ضمیمه ب (مربوط به موتورهای پله ای) مراجعه کنید.

Current Risetime and Torque

The major contributor to a stepper motor's loss of running torque at high speeds is the fact that pulse switching times can become shorter than motor phase-current risetimes; that is, the pulse ends before phase current rises to a level that provides sufficient torque. A simple RL series circuit's mesh equations illustrate the relationships involved, where t is current risetime:

$$V = IR + L(dI/dt)$$

$$I(t) = \frac{V}{R} \bullet \left(1 - e^{(-Rt/L)}\right)$$

To reduce t, the power-supply voltage, V, must be as high as possible and the motor phase inductance, L, and resistance, R, must be as low as possible. **Figure 3a**, where $\tau = L/R$ is the time constant, shows current vs. time for the series RL circuit. Note that at time τ current has risen to $1\text{-}e^{-1}$ times its final level, I_O, or $0.63I_O$, but only to $1\text{-}e^{-0.5}$ times I_O, or $0.39I_O$, at time $\tau/2$. Because torque is approximately proportional to current, the torque available at speed $1/t_1$(Fig. 3b), where $t_1=\tau$, is 62% (or 100% x $0.63I_O/0.39I_O$) greater than at the higher speed $1/t_2$ (Fig. 3c),

$$t = -\frac{L}{R} \bullet \ln\left(1 - \frac{R \bullet I(t)}{V}\right)$$

where $t_2 = \tau/2$. Torque is optimized if phase current is applied for at least 2.3τ, at which time the current has risen to 90% of I_O.

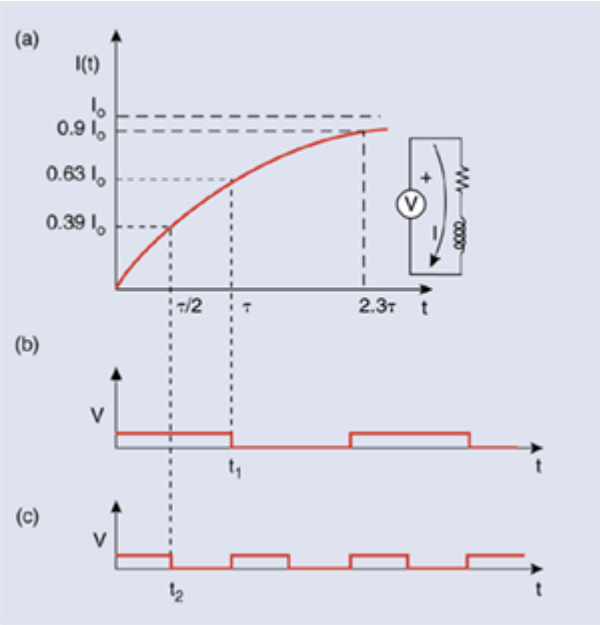


Figure 3. (a) Motor phase inductance, L, and resistance, R, limit phase-current risetime. For example, application of phase voltage for $\tau/2$, τ, and 2.3τ, respectively, results in peak currents reaching 39%, 63%, and 90% of final value I_o. Because torque is approximately proportional to current, motors that operate at (b) lower speeds, which permit longer applications of phase voltage, deliver more torque than those (c) operating at higher speeds.

Holding vs. Dynamic Torque

Assume that full current rise is obtained during phase switching. You can calculate the maximum dynamic torque of a 2-phase bipolar motor as follows:

$$\overline{T_{MAX}} = T_0 \int_{\pi/2}^{\pi} T(\theta)\,d\theta/(\pi/2)$$

$$= T_0 \left(\int_{\pi/2}^{3\pi/4} \sin(\theta)\,d\theta + \int_{3\pi/4}^{\pi} -\cos(\theta)\,d\theta \right) \Big/ \frac{\pi}{2}$$

$$= T_0 \left(\frac{-\cos(3\pi/4) + \cos(\pi/2) - \sin(\pi) + \sin(3\pi/4)}{\pi/2} \right)$$

$$= 0.900\,T_0$$

where T_O is the maximum holding torque (**Fig. 4a**).

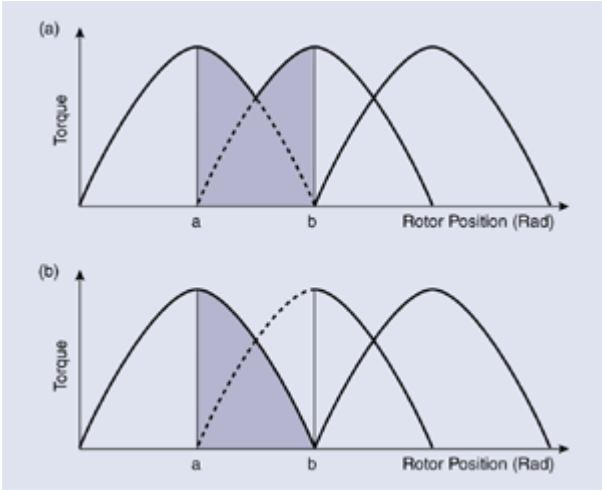


Figure 4. The shaded portions of these curves represent (a) the maximum and (b) the minimum available dynamic torque, averaged between points a and b.

If you need to approach this ideal level, which requires perfect switching time relative to rotor position, you should employ closed-loop control. Open-loop control for the 2-phase bipolar motor will give you at least this minimum dynamic torque:

$$\overline{T_{MIN}} = T_0 \int_{\pi/2}^{\pi} T(\theta)d\theta/(\pi/2)$$

$$= T_0 \int_{\pi/2}^{\pi} \sin(\theta)d\theta/(\pi/2)$$

$$= T_0\left(\frac{-\cos(\pi)-\cos(\pi/2)}{\pi/2}\right) = 0.637T_0$$

or 63.7% of T_O (Fig. 4b). Typical dynamic torque is (90% + 63.7%)/2 or 76.8%.

Given a certain power input, you can't change a specific motor's power output, $\int T(\omega)d\omega$ (where $T(\omega)$ is the torque at angular velocity ω). If your test instructions call for more dynamic torque than $T(\omega)$ at angular velocityω, then you are once again dealing with a flawed design, not bad components. But if you know your application's speed, accuracy, resolution, and torque requirements, you can work with your firm's designers to determine whether a specific stepper motor delivers sufficient torque at the desired operating speed within your specific mechanical constraints. T&MW

ضمیمه ج
(مربوط به سخت افزارونرم افزار پروژه)

در این قسمت نحوهٔ دستیابی به پورت ها در چند زبان برنامه نویسی رایج نشان داده شده است :

I/O Port Access in Microsoft Visual C++

Microsoft Visual C/C++ provides access to the I/O ports on the 80x86 CPU via the predefined functions **_inp** / **_inpw** and **_outp** / **_outpw**.

```
       int _inp(unsigned portid); /* returns a byte read from the I/O
port portid */

 unsigned _inpw(unsigned portid); /* returns a word read from the I/O
port portid */

      int _outp(unsigned portid,  /* writes the byte value to the I/O
port portid */
                int value);       /* returns the data actually written
*/

unsigned _outpw(unsigned portid,  /* writes the word value to the I/O
port portid */
                unsigned value);  /* returns the data actually written
*/
```

portid can be any unsigned integer in the range 0-65535

```
#include <conio.h> /* required only for function declarations */

#define Data    0x378
#define Status  0x379
#define Control 0x37a

int Bits,      /* 0 <= Bits <= 255 */
    Dummy;

Dummy = _outp(Data,Bits); /* output data */

Bits = _inp(Status);      /*  input data */
```

I/O Port Access in Turbo C, Borland C/C++

Turbo C and Borland C/C++ provide access to the I/O ports on the 80x86 CPU via the predefined functions **inportb** / **inport** and **outportb** / **outport**.

```
int inportb(int portid); /* returns a byte read from the I/O port
portid */

 int inport(int portid); /* returns a word read from the I/O port
portid */

void outportb(int portid, unsigned char value);
                        /* writes the byte value to the I/O port
portid */
```

```
 void outport(int portid, int value);
                        /* writes the word value to the I/O port
portid */
```

```
#include <stdio.h>
#include <dos.h>

#define Data    0x378
#define Status  0x379
#define Control 0x37a

unsigned char Bits;

outportb(Data,Bits);   /* output data */

Bits = inportb(Status); /* input data */
```

I/O Port Access in Watcom C

Watcom C provides access to the I/O ports on the 80x86 CPU via the predefined functions **inp** / **inpw** and **outp** / **outpw**.

```
  unsigned int inp(int portid);   /* returns a byte read from the I/O
port portid */

 unsigned int inpw(int portid);   /* returns a word read from the I/O
port portid */

 unsigned int outp(int portid,    /* writes the byte value to the I/O
port portid */
                   int value);    /* returns the data actually written
*/

unsigned int outpw(int portid,    /* writes the word value to the I/O
port portid */
              unsigned int value); /* returns the data actually written
*/
```

portid can be any unsigned integer in the range 0-65535

```
#include <conio.h>

#define Data    0x378
#define Status  0x379
#define Control 0x37a

int Bits,     /* 0 <= Bits <= 255 */
    Dummy;

Dummy = outp(Data,Bits); /* output data */

Bits = inp(Status);      /*  input data */
```

I/O Port Access in Turbo Pascal

Turbo Pascal provides access to the I/O ports on the 80x86 CPU via two predefined arrays, **Port** and **PortW**

```
var Port:  array[0..65535] of byte;

    PortW: array[0..65534] of word;
```
The indexed elements of each array match the port at the corresponding I/O address. Assigning a value to an element of the Port or PortW arrays causes that value to be written out to the corresponding port. When an element of the Port or PortW arrays is referenced in an expression, the value is read in from the corresponding port.

```
Const Data    = $378;
      Status  = Data + 1;
      Control = Data + 2;

var Bits: Byte;

Port[Data] := Bits;  { output data }

Bits := Port[Status]; { input data }
```

I/O Port Access in QBasic

QBasic provides access to the I/O ports on the 80x86 CPU via the **INP** function and the **OUT** statement.

```
INP(portid) ' returns a byte read from the I/O port portid

OUT portid, value ' writes the byte value to the I/O port portid
```
portid can be any unsigned integer in the range 0-65535. *value* is in the range 0-255.

```
pdata   = &H378
status  = &H379
control = &H37A

OUT pdata, bits    ' output data

bits = INP(status) '  input data
```

Accessing I/O Ports with Debug

```
C:\>debug /?
Runs Debug, a program testing and editing tool.

Debug [[drive:][path]filename [testfile-parameters]]

  [drive:][path]filename  Specifies the file you want to test.
  testfile-parameters     Specifies command-line information required
by
                          the file you want to test.

After Debug starts, type ? to display a list of debugging commands.
```

Debug provides input and output commands -

```
i port_address

o port_address byte_value
```

```
C:\>debug
-o 378 5a
-i 379
7F
-q

C:\>
```