در بخش قبل با نحوه طراحی کارتهای کامپیوتری آشنا شدیم. امروزه بسیاری از وسایل ورودی/خروجی مانند کارت صوتی و گرافیکی (حتی مودم و کارت شبکه) روی بورد اصلی کامپیوتر جاسازی شده‌اند[۱] و شکافهای بورد نیز برای اضافه کردن کارتهای دیگر تعبیه شده‌اند. اما رفته‌رفته پورتهای خارجی کامپیوتر، جای کارتها را برای اتصال ورودی/خروجی به کامپیوتر می‌گیرند. در بازار امروز تجهیزات بسیاری از جمله کارت حافظه، چاپگر، CD-ROM خارجی[۲]، اسکنر، کارت TV، دیسک سخت و ... از طریق پورت USB به کامپیوتر متصل می‌شوند. مزیت مهم این اتصال این است که نیازی به باز کردن درب جعبه کامپیوتر نیست و به راحتی می‌توان تجهیزات ورودی/خروجی را به کامپیوتر متصل و نیز قطع نمود. به علاوه اگر در طراحی کارت برای انجام یک عملیات ورودی/خروجی اشتباهی رخ دهد، با قرار دادن کارت در شکاف بورد اصلی کمترین حاصل روشن نشدن کامپیوتر و بدترین نتیجه سوختن بورد اصلی کامپیوتر خواهد بود.

در این بخش دو پورت موازی[۳] و سریال[۴] کامپیوتر را معرفی و نحوه اتصال تجهیزات به آنها را بررسی خواهیم کرد. چاپگرهای قدیمی‌تر (و حتی بعضی انواع جدید) به پورت موازی کامپیوتر متصل می‌شوند. به همین لحاظ پورت موازی به پورت چاپگر نیز معروف است. پورت سریال نیز قبلاً برای اتصال اسکنر و مودمهای خارجی[۵] مورد استفاده قرار می‌گرفت. با مطالبی که در این بخش می‌آموزید خواهید توانست به راحتی دستگاههایی که خودتان می‌سازید (مانند دستگاههای جمع‌آوری اطلاعات یا روباتها) را با این دو پورت متداول کامپیوتر کنترل کنید. به علاوه از این دو پورت برای ارتباط اطلاعاتی بین دو کامپیوتر نیز استفاده می‌شود.

پیش از بررسی جزیی ویژگیهای این دو پورت، بیان مطالبی راجع به ارتباطات کامپیوتری و طبقه‌بندی آنها ضروری به نظر می‌رسد. این مطالب در مورد ارتباط بین هر دو سیستم مبتنی بر پردازنده نیز صدق می‌کند.

دسته‌بندی ارتباطات کامپیوتری

برای بررسی و مقایسه ویژگیهای انواع ارتباطات کامپیوتری، به دسته‌بندی آنها از دیدگاههای مختلف می‌پردازیم:

- از دید حجم سیمهای رابط
- از دید شیوه کنترل جریان داده
- از دید جهت انتقال اطلاعات

---

[1] به اصطلاح on-board هستند.

[2] ***External***
[3] ***Parallel***
[4] ***Serial***
[5] ***External***

ارتباطات کامپیوتری از دید حجم سیمهای رابط

از این دیدگاه، ارتباطات کامپیوتری به دو دسته ارتباط موازی و ارتباط سریال تقسیم می‌شوند. فرض کنید می‌خواهیم یک داده ۸ بیتی را از یک سیستم به سیستم دیگری انتقال دهیم. هر بیت، یک ولتاژ دیجیتال است که باید روی سیم انتقال داده شود. یک راه برای انتقال این داده ۸ بیتی این است که ۸ سیم از سمت فرستنده به گیرنده برقرار شود و هر بیت داده روی یک سیم منتقل شود[1]. شکل زیر این نوع ارتباط را نشان می‌دهد:

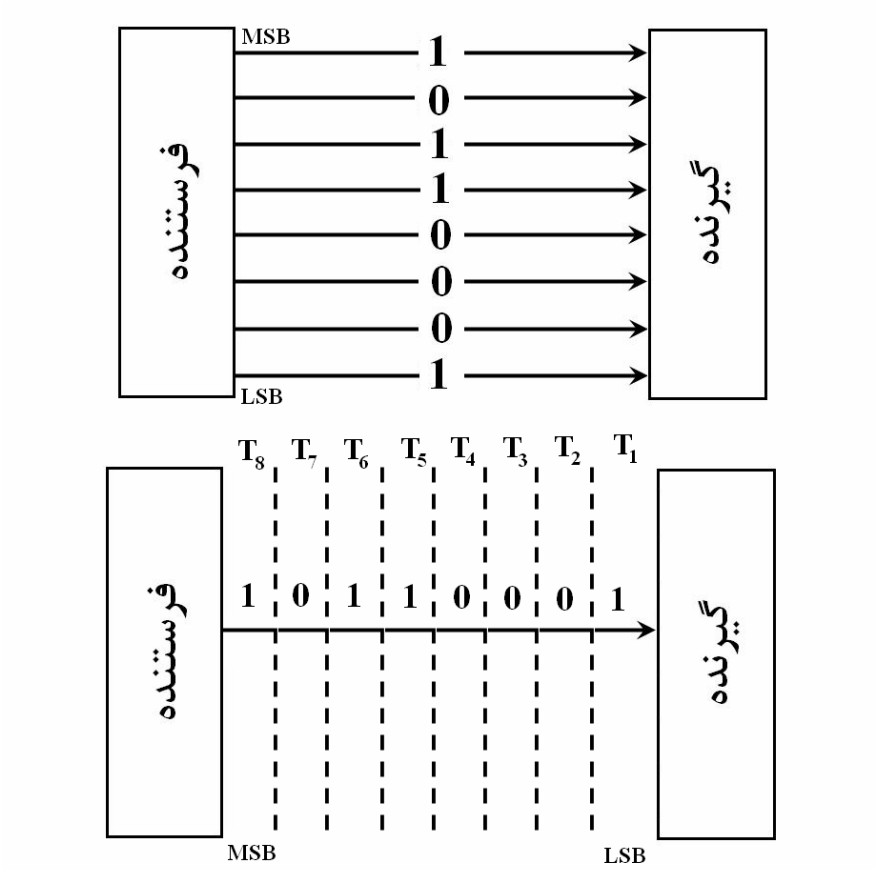
چون هر ۸ بیت موازی با هم از فرستنده ارسال و در گیرنده دریافت می‌شوند، این نوع ارتباط به **ارتباط موازی** معروف است. هنگامی که از یکی از پورتهای ورودی/خروجی یک میکروکنترلر اطلاعاتی را ارسال یا دریافت می‌کنید و یا هنگام استفاده از پورت چاپگر کامپیوتر، از ارتباط موازی استفاده می‌کنید.

راه دیگر برای برقراری ارتباط بین فرستنده و گیرنده این است که به جای استفاده از ۸ سیم و ارسال هر بیت داده روی یک سیم، تنها از یک سیم بین فرستنده و گیرنده برای انتقال داده‌ها استفاده کنیم. واضح است یک سیم در هر لحظه تنها می‌تواند یک ولتاژ (یک بیت داده) را روی خود نگه دارد؛ بنابراین فرستنده باید ۸ بیت داده را یکی یکی و به صورت متوالی (سریال) روی سیم انتقال داده‌ها قرار دهد و گیرنده نیز بیتها را یکی یکی از رسانه انتقال داده‌ها بخواند و بایت انتقالی را بسازد. این ارتباط به **سریال** معروف است. پورت COM و پورت USB کامپیوتر و نیز شبکه‌های کامپیوتری از این نوع ارتباط استفاده می‌کنند. اکثر میکروکنترلرها امکانات ارتباط سریال دارند.

برای مثال فرض کنید می‌خواهیم بایت 10110001 را از فرستنده به گیرنده انتقال دهیم. شکل زیر این ارسال را در دو حالت موازی و سریال نشان می‌دهد.

---

[1] البته واضح است که دست‌کم یک سیم زمین به عنوان ولتاژ مرجع مورد نیاز است. به طوری که خواهیم دید، در پورت موازی کامپیوتر به ازای هر سیم داده یک سیم زمین وجود دارد.

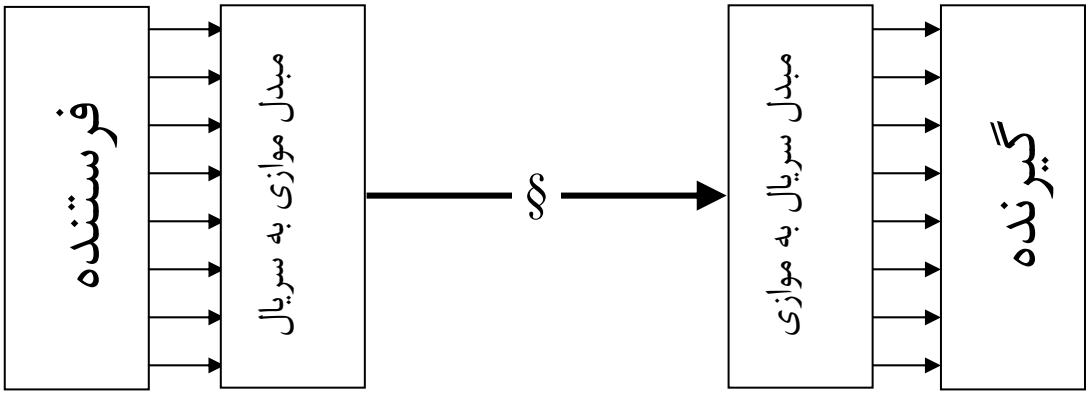


همانطور که می‌بینید در ارتباط سریال ابتدا کم‌ارزش‌ترین بیت ارسال می‌شود. هر بیت مدت زمانی روی خط می‌ماند تا گیرنده فرصت خواندن آن را داشته باشد. معمولاً زمانهای T_1 تا T_8 با هم مساوی هستند.

با مشخصاتی که گفته شد، مقایسه‌ای بین ارتباط موازی و سریال مفید است:

- حجم سیمهای ارتباطی در ارتباط موازی بیشتر از ارتباط سریال است. این موضوع از طرفی باعث بالا رفتن هزینه سیم‌کشی می‌شود و از طرف دیگر باعث می‌شود سیمهای ارتباطی روی یکدیگر ایجاد نویز کنند. به همین لحاظ، فاصله دو سیستم با ارتباط موازی حداکثر می‌تواند تا ۵ متر باشد؛ اما طول ارتباط سریال می‌تواند در محیط‌های کم‌نویز تا ۱۵ متر باشد.
- چون در ارتباط موازی هر ۸ بیت داده به صورت موازی ارسال می‌شوند، سرعت آن بیشتر از ارتباط سریال است که در آن، بیتها یکی یکی فرستاده می‌شوند. سرعت ارتباط موازی می‌تواند تا ۱۰۰ کیلوبایت در ثانیه نیز باشد، در حالی که سرعت ارتباط سریال حداکثر ۲۵۶۰۰۰ بیت در ثانیه (۳۲ کیلوبایت در ثانیه) است.
- وقتی در ارتباط سریال سیم رابط قطع شود، کل ارتباط قطع می‌شود؛ اما در ارتباط موازی با قطع یک سیم، فقط یکی از بیتها مخدوش می‌شود.

- اگر در طول ارتباط، نویز شدید و کوتاه‌مدتی رخ دهد، بیتهای ارسالی سریال را خراب می‌کند. اما در ارتباط موازی چون هر ۸ بیت اطلاعات مدت زمانی روی خطوط ارتباطی می‌مانند، امکان خراب نشدن اطلاعات بیشتر است.
- ارتباط موازی ساده است؛ چون به ازای هر بیت داده یک سیم وجود دارد. اما در ارتباط سریال چون تنها یک سیم برای انتقال داده موجود است، در فرستنده باید یک «مبدل موازی به سریال» و در گیرنده یک «مبدل سریال به موازی» باید وجود داشته باشد تا هماهنگی ارسال و دریافت اطلاعات را انجام دهند:



وظیفه مبدل موازی به سریال، دریافت اطلاعات به صورت موازی از فرستنده و گذاشتن (به اصطلاح «سوار کردن») اطلاعات روی خط انتقال است. مبدل سریال به موازی، اطلاعات را به اصطلاح از روی خط «پیاده کرده» و به صورت موازی در اختیار گیرنده قرار می‌دهد. سیستمهای آماده‌ای وجود دارند که عمل سوار کردن (مدولاسیون-MOdulation) و پیاده کردن (دمدولاسیون-DEModulation) اطلاعات را انجام می‌دهند که به اصطلاح مودم (MODEM) نامیده می‌شوند[1]. حتماً از مودم خط تلفن[2] یا مودم ADSL برای اتصال به اینترنت استفاده کرده‌اید. این کارتها، رده خاصی ار مودمها هستند که از خط تلفن برای انتقال اطلاعات به صورت سریال بین کامپیوتر شما و شرکت سرویس‌دهنده اینترنت[3] استفاده می‌کنند. در این مثال فاصله بین فرستنده و گیرنده دو سر ارتباط می‌تواند میلیونها کیلومتر باشد.

بنا به مقایسه‌ای که انجام شد، برای برقراری ارتباط بین دو سیستم می‌توان از هرکدام از ارتباطات موازی یا سریال استفاده کرد.

---

[1] این موضوع یک دیدگاه کلی است. از دید مخابراتی، اگر بیتها به صورت دو ولتاژ مختلف روی رسانه انتقال قرار داده شوند، به مدولاسیون نیازی نیست. بلکه در حالتی که سیگنالهای دیجیتال باید به شکل دیگری از سیگنال (معمولاً آنالوگ) تبدیل شوند تا قابل حمل روی رسانه باشند (مثلاً وقتی مودم کامپیوتر «صفر» و «یک»ها را به سیگنالهای قابل حمل به کمک حامل‌های روی خط تلفن تبدیل می‌کند)، به مدولاسیون و در مقصد به دمدولاسیون نیاز است.

[2] ***Dial-up Modem***

[3] ***Internet Service Proider: ISP***

دسته‌بندی ارتباطات کامپیوتری از دید شیوه کنترل جریان داده[1]

فرض کنید سیستم فرستنده می‌خواهد ۱۰۰۰ قلم اطلاعات را برای سیستم گیرنده ارسال کند. باید مکانیسمی بین فرستنده و گیرنده پیش‌بینی شود تا جریان داده بین آنها کنترل‌شده باشد. به بیان دیگر، فرستنده باید پس از ارسال هر قلم اطلاعات، مطمئن شود که گیرنده آن را دریافت و ذخیره (و در بعضی مواقع پردازش) کرده و بعد قلم بعدی اطلاعات را ارسال کند. به اصطلاح فرستنده نباید گیرنده را در اطلاعات غرق[2] کند.

برای کنترل جریان داده دو روش **تأخیر**[3] و **دست‌دهی**[4] وجود دارد. برای بررسی این دو روش بیان یک مثال مفید است.

فرض کنید شما می‌خواهید برای دوستتان اسم ده نفر را بخوانید تا او بنویسد. طبیعی است که نمی‌توان اسم ده نفر را پشت سرهم بخوانید؛ وگرنه دوستتان نمی‌تواند آنها را بنویسد. یک راه ساده این است که بعد از خواندن هر اسم، مدت زمان ثابتی (مثلاً ۳۰ ثانیه) صبر کنید و سپس اسم بعدی را بخوانید. این روش ساده است، اما ممکن است منجر به ایجاد مشکل شود؛ مثلاً سرعت نوشتن دوستتان زیاد باشد و زمانی که بین خواندن دو اسم صرف می‌کنید، به هدر رود. از طرف دیگر، ممکن است در حین نوشتن مشکلی برای دوستتان پیش بیاید (مثلاً نوک مدادش بشکند!) و نتواند در مدت زمان ثابتی که بین خواندن دو اسم صبر می‌کنید، اسم قبلی را بنویسد. به صورت خلاصه اشکال روش ایجاد تأخیر ثابت بین ارسال دو قلم اطلاعات این است که ممکن است این زمان زیاد باشد که باعث اتلاف وقت شود یا کم باشد که گیرنده نتواند اطلاعات را به درستی ثبت کند.

روش دیگر این است که یک اسم را بخوانید، صبر کنید تا دوستتان به شما بگوید که اسم قبلی را نوشته و سپس اسم بعدی را بخوانید. زحمت این روش کمی بیشتر از روش قبل است؛ اما در عوض زمان به هدر نمی‌رود و در صورت بروز مشکل در ثبت اطلاعات، فرستنده قبل از تصدیق دریافت اطلاعات قبلی توسط گیرنده اطلاعات بعدی را ارسال نخواهد کرد. به این روش به اصطلاح دست‌دهی[5] گفته می‌شود.

به صورت خلاصه برای کنترل جریان داده دو روش وجود دارد:

- **روش تأخیر:** در این روش، فرستنده پس از ارسال هر قلم اطلاعات، مدت زمان ثابتی (مثلاً T) تأخیر ایجاد کرده و سپس داده بعدی را ارسال می‌کند. گیرنده نیز در فواصل

---

[1] ***Flow Control***
[2] ***Overwhelm***
[3] ***Delay***
[4] ***Hand-Shaking***
[5] اصطلاح دست‌دهی معادل فارسی Hand Shaking است که در زبان انگلیسی به معنای دست دادن (و به نوعی توافق و تفاهم کردن) می‌باشد.

زمانی T خطوط ارتباطی را خوانده و اطلاعات را ثبت می‌کند. این روش ساده است؛ اما ثابت بودن زمان تأخیر ایجاد مشکل می‌کند. ممکن است این زمان زیاد باشد که باعث اتلاف وقت شود یا کم باشد که گیرنده نتواند اطلاعات را به درستی ثبت کند.

- **روش دست‌دهی:** در این روش، فرستنده پس از ارسال هر قلم اطلاعات، صبر می‌کند تا گیرنده دریافت آن را تصدیق کند و سپس داده بعدی را می‌فرستد. در این روش علاوه بر داده‌ها، سیگنالهای کنترلی هم باید رد و بدل شوند و به همین خاطر ماهیتی پیچیده‌تر از روش قبل دارد؛ اما در عوض زمان به هدر نمی‌رود و در صورت بروز مشکل در ثبت اطلاعات، فرستنده قبل از تصدیق دریافت اطلاعات قبلی توسط گیرنده اطلاعات بعدی را ارسال نخواهد کرد.

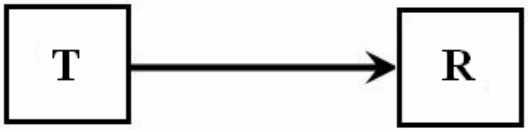
مراحل ارسال هر قلم اطلاعات به این روش توسط فرستنده عبارتند از: ۱) ارسال داده ۲) آگاه کردن گیرنده از ارسال داده ۳) انتظار برای تصدیق دریافت داده توسط گیرنده

توجه کنید دسته‌بندی ارتباطات از دید شیوه کنترل جریان داده، ارتباطی با دسته‌بندی از دید حجم سیمهای ارتباطی ندارد؛ یک ارتباط موازی (یا سریال) می‌تواند از هرکدام از دو شیوه تأخیر یا دست‌دهی برای کنترل جریان داده استفاده کند.

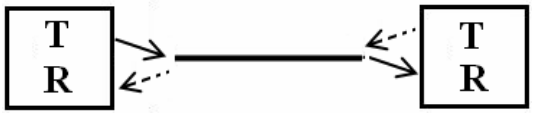
دسته‌بندی ارتباطات کامپیوتری از دید جهت انتقال داده‌ها

از این دیدگاه، سه نوع ارتباط وجود دارد:

- **ارتباط یک‌طرفه**[1]: در این نوع ارتباط یک فرستنده[2] و یک گیرنده[3] داریم و جهت انتقال داده‌ها همواره از فرستنده به گیرنده است.



- **ارتباط نیمه دوطرفه**[4]: در این نوع ارتباط هرکدام از طرفین ارتباط می‌توانند فرستنده یا گیرنده باشند؛ اما چون تنها یک کانال داده بین آنها وجود دارد، در هر لحظه یکی از دو طرف فرستنده و دیگری گیرنده است. به بیان دیگر در هر لحظه یکی از طرفین فرستنده و دیگری گیرنده است.



[1] *Simplex*
[2] *Transmitter*
[3] *Receiver*
[4] *Half-Duplex*

- **ارتباط کاملاً دوطرفه**[1]: در این نوع ارتباط نیز هرکدام از طرفین ارتباط می‌توانند فرستنده یا گیرنده باشند. اما دو کانال مجزا بین فرستنده و گیرنده وجود دارد و هر دو طرف می‌توانند همزمان هم فرستنده و هم گیرنده باشند.

مانند قبل، دسته‌بندی ارتباطات از این دیدگاه، ارتباطی با دسته‌بندی‌های قبلی ندارد. به عنوان مثال یک ارتباط یک‌طرفه می‌تواند برای کنترل جریان داده از روش تأخیری یا دست‌دهی استفاده کند. توجه کنید تصدیق دریافت داده قبلی توسط گیرنده که در روش دست‌دهی استفاده می‌شود، سیگنالی از سمت گیرنده به فرستنده است؛ اما منظور از ارتباط یکطرفه، انتقال یک‌طرفه داده‌هاست و سیگنالهای کنترلی می‌تواند از سمت گیرنده به سمت فرستنده باشد.

تشخیص خطا[2] و تصحیح خطا[3]

در حین یک ارتباط ممکن است به علت وجود نویز، قطع و وصل رسانه انتقال داده و ... خطایی در ارتباط رخ دهد و اطلاعاتی که فرستنده ارسال می‌کند به درستی در گیرنده دریافت نشود. به همین لحاظ معمولاً در ارتباطات کامپیوتری مکانیسمی برای تشخیص و در صورت امکان تصحیح خطا در نظر گرفته می‌شود.

روش اساسی برای تشخیص خطا این است که در فرستنده، محاسباتی روی داده‌هایی که قرار است ارسال شود انجام شده و نتیجه این محاسبات به عنوان اطلاعات کشف خطا به همراه داده‌ها ارسال می‌شود. در گیرنده همین محاسبات روی داده‌های دریافتی انجام شده و نتیجه آن با نتیجه‌ای که از فرستنده دریافت شده (اطلاعات کشف خطا) مقایسه می‌شود. اگر دو نتیجه یکسان نباشند، نشان‌دهنده این است که در طول ارتباط خطایی رخ داده است. اما نکته مهم این است که اگر نتیجه‌ها یکسان باشند نمی‌توان با قاطعیت گفت خطایی رخ نداده است!

به عنوان مثال فرض کنید قرار است ۵ عدد از فرستنده به گیرنده ارسال شود. فرستنده برای کشف خطا، اعداد فوق را با هم جمع کرده و رقم یکان مجموع را به عنوان اطلاعات کشف خطا ارسال می‌کند. در گیرنده نیز اعداد دریافتی با هم جمع می‌شوند و رقم یکان مجموع با اطلاعات کشف خطا مقایسه می‌شود تا وقوع خطا بررسی شود. بسته اطلاعات زیر را در نظر بگیرید:

---

[1] ***Full-Duplex***
[2] ***Error Detection***
[3] ***Error Correction***

داده‌های ارسالی					اطلاعات کشف خطا
۱۰	۲۴	۳۳	۱۸	۶	۱

فرستنده داده‌های ارسالی را با هم جمع می‌کند (۹۱ = ۱۰ + ۲۴ + ۳۳ + ۱۸ + ۶) و رقم یکان آن را به عنوان اطلاعات کشف خطا به همراه داده‌ها ارسال می‌کند.
در گیرنده داده‌های دریافتی با هم جمع می‌شوند و اگر رقم یکان مجموع «۱» شود، گیرنده به این نتیجه می‌رسد که داده‌ها به درستی دریافت شده‌اند.
حال فرض کنید خطایی رخ دهد و به جای عدد ۳۳ عدد ۳۵ دریافت شود. گیرنده اعداد فوق را با هم جمع می‌کند (۹۳ = ۱۰ + ۲۴ + ۳۵ + ۱۸ + ۶) و با مقایسه رقم یکان مجموع «۳» با اطلاعات کشف خطا (۱) متوجه وقوع خطا می‌شود.
اما آیا ممکن است خطایی رخ دهد و گیرنده متوجه وقوع خطا نشود؟! فرض کنید خطایی رخ دهد و به جای عدد ۳۳ عدد ۳۵ و به جای عدد ۶ عدد ۴ دریافت شود. گیرنده اعداد فوق را با هم جمع می‌کند (۹۱ = ۱۰ + ۲۴ + ۳۵ + ۱۸ + ۴) و با مقایسه رقم یکان مجموع «۱» با اطلاعات کشف خطا (۱) به این نتیجه نادرست می‌رسد که خطایی رخ نداده است! در حالی که نه تنها خطا رخ داده، بلکه دو قلم اطلاعات را نیز خراب کرده است.

روشهای معمول کشف خطا در ارتباطات کامپیوتری از همین شیوه پیروی می‌کنند. در روش ارسال بیت توازن[1]، بر مبنای تعداد «یک»ها در یک بسته داده عمل می‌شود. روش بیت توازن که از یک بیت به عنوان اطلاعات کشف خطا استفاده می‌کند، به دو صورت توازن زوج[2] و توازن فرد[3] عمل می‌کند. در روش توازن زوج، فرستنده تعداد «یک»های بسته داده را می‌شمارد و در صورتی که تعداد آنها زوج باشد بیت توازن را «صفر» و اگر فرد باشد بیت توازن را «یک» می‌کند. به بیان دیگر مجموع تعداد «یک»های بسته داده و بیت توازن باید زوج باشد. در روش توازن فرد مجموع تعداد «یک»های بسته داده و بیت توازن باید فرد باشد. مثالهای زیر بسته‌های ۸ بیتی به همراه بیت توازن زوج را نشان می‌دهند.

بسته داده								بیت توازن زوج
۰	۱	۱	۰	۰	۰	۱	۱	۰

بسته داده								بیت توازن زوج
۱	۱	۰	۰	۱	۱	۰	۱	۱

[1] ***Parity***
[2] ***Even Parity***
[3] ***Odd Parity***

اگر در جریان انتقال داده‌ها خطایی رخ دهد و یک بیت «یک» به «صفر» یا یک بیت «صفر» به «یک» تبدیل شود، گیرنده با ارزیابی بیت توازن متوجه وقوع این خطا می‌شود. مثلاً فرض کنید بسته داده دومی به صورت زیر دریافت شود:

بسته داده								بیت توازن زوج
۱	۱	۱	۰	①	۱	۱	۰	۱

گیرنده با شمارش تعداد «یک»های بسته داده و بیت توازن به عدد ۷ می‌رسد و چون از قبل مکانیسم توازن زوج بین فرستنده و گیرنده توافق شده و ۷ یک عدد فرد است، گیرنده متوجه وقوع خطا می‌شود.

واضح است که اگر تعداد بیتهای خطا زوج باشد (دو بیت یا چهار بیت یا شش بیت داده دچار خطا شود)، با مکانیسم توازن (چه توازن زوج و چه توازن فرد) نمی‌توان به آن پی برد. چون تعداد «یک»ها تغییری نمی‌کند.

در شبکه‌های TCP/IP از مکانیسمی به نام CRC[1] برای کشف خطا استفاده می‌شود. در این روش، فرستنده اطلاعات بسته داده که شامل «صفر» و «یک» است را بر یک چندجمله‌ای ویژه تقسیم می‌کند و باقیمانده آن (موسوم به CRC) را به همراه بسته داده می‌فرستد. گیرنده بسته دریافتی را بر همان چندجمله‌ای تقسیم می‌کند و باقیمانده را با CRC دریافتی مقایسه می‌کند. تحقیقات فراوانی برای انتخاب چندجمله‌ای مزبور انجام شده تا امکان وقوع خطا و کشف نشدن آن توسط مکانیسم CRC (مانند آنچه در مورد بیت توازن دیدیم) را به حداقل برساند.

حال فرض کنید وقوع خطا کشف شده است. اکنون چه باید کرد؟!

می‌توان با تغییراتی در روش کشف خطا، آن را به روش تصحیح خطا تبدیل کرد؛ به گونه‌ای که گیرنده بتواند به کمک این اطلاعات محل وقوع بیت خطا را تشخیص دهد. تعداد بیتهایی که فرستنده برای تصحیح خطا باید به همراه بسته داده ارسال کند، از تعداد بیتهای کشف خطا بیشتر است و قسمت زیادی از پهنای باند را هدر می‌دهد؛ به همین لحاظ معمولاً از روش تصحیح خطا استفاده نمی‌شود. شیوه معمول این است که گیرنده با کشف خطا، درخواست ارسال مجدد بسته[2] را می‌نماید که از دید آماری نسبت به روش تصحیح خطا، پهنای باند کمتری صرف می‌کند.

---

[1] ***Cyclic Redundancy Check***

[2] ***Retransmission***

پروتکل[1]

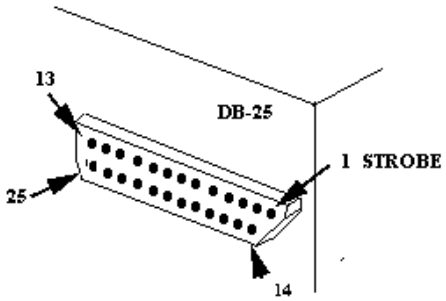
به مجموعه قراردادهای مابین فرستنده و گیرنده برای یک ارتباط صحیح، پروتکل گفته می‌شود. در پروتکل باید موارد زیر مشخص شود:

- نحوه آغاز و پایان ارتباط
- سرعت ارتباط
- شیوه کنترل جریان داده
- نوع ارتباط از دید حجم سیمهای ارتباطی (موازی یا سریال)
- جهت انتقال داده‌ها
- نحوه تشخیص و تصحیح خطا
- ...

پورت موازی کامپیوتر

اگر به پشت بورد اصلی کامپیوتر خود نگاه کنید، یک پورت با ۲۵ پایه مانند شکل مقابل می‌بینید که به پورت موازی، پورت چاپگر، پورت LPT[2] یا در اصطلاح فنی الکترونیک به پورت DB25 معروف است.

در این بخش به بررسی سیگنالهای پورت موازی کامپیوتر می‌پردازیم. با شناخت این سیگنالها دو مطلب را خواهیم آموخت؛ اول چگونگی کنترل یک چاپگر بدون نیاز به کامپیوتر و دوم نحوه کنترل یک دستگاه جانبی به کمک پورت موازی کامپیوتر.



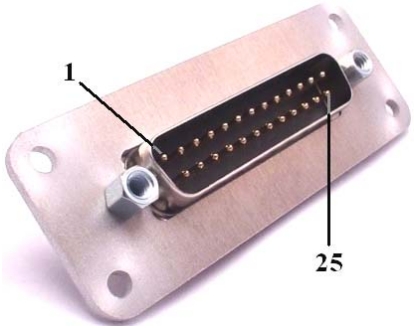
پیش از توضیح عملکرد پایه‌ها، توضیحی راجع به نحوه شماره‌گذاری پایه‌ها لازم است. پورتی که پشت کامپیوتر مشاهده می‌کنید، دارای ۲۵ **حفره** است که سیگنالها را در اختیار کاربر قرار می‌دهد[3]. شماره‌گذاری این حفره‌ها به صورت مقابل است:

معمولاً برای استفاده از این پورت، قطعه‌ای به کار می‌رود که به جای حفره دارای **میله** است و میله‌ها درون حفره‌ها قرار می‌گیرند[4]. این قطعه نیز به نام DB25 معروف است. شکل بعدی را ببینید. سر کابل چاپگر که درون پورت موازی قرار می‌گیرد نیز چنین شکلی دارد.

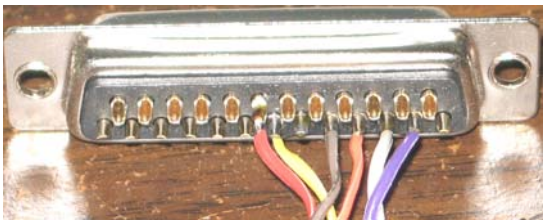
---

[1] ***Protocol***
[2] ***Line Printer***

[3] این نوع پورت معمولاً به Socket یا Female معروف است.
[4] این نوع پورت معمولاً به Plug یا Male معروف است.



دقت کنید که در پورت اصلی کامپیوتر، پایه بالا سمت راست و در قطعه روبرو که درون پورت قرار می‌گیرد، پایه بالا سمت چپ شماره یک است. چون وقتی قطعه بالا درون پورت قرار می‌گیرد، سمت چپ آن با سمت راست پورت مقابل هم قرار می‌گیرد. پشت این قطعه در محل هر میله، جایی برای لحیم کردن سیم تعبیه شده تا به راحتی بتوانید از سیگنالهای پورت موازی استفاده کنید. شکل زیر را ببینید:



جدول زیر سیگنالهای پورت موازی را نشان می‌دهد:

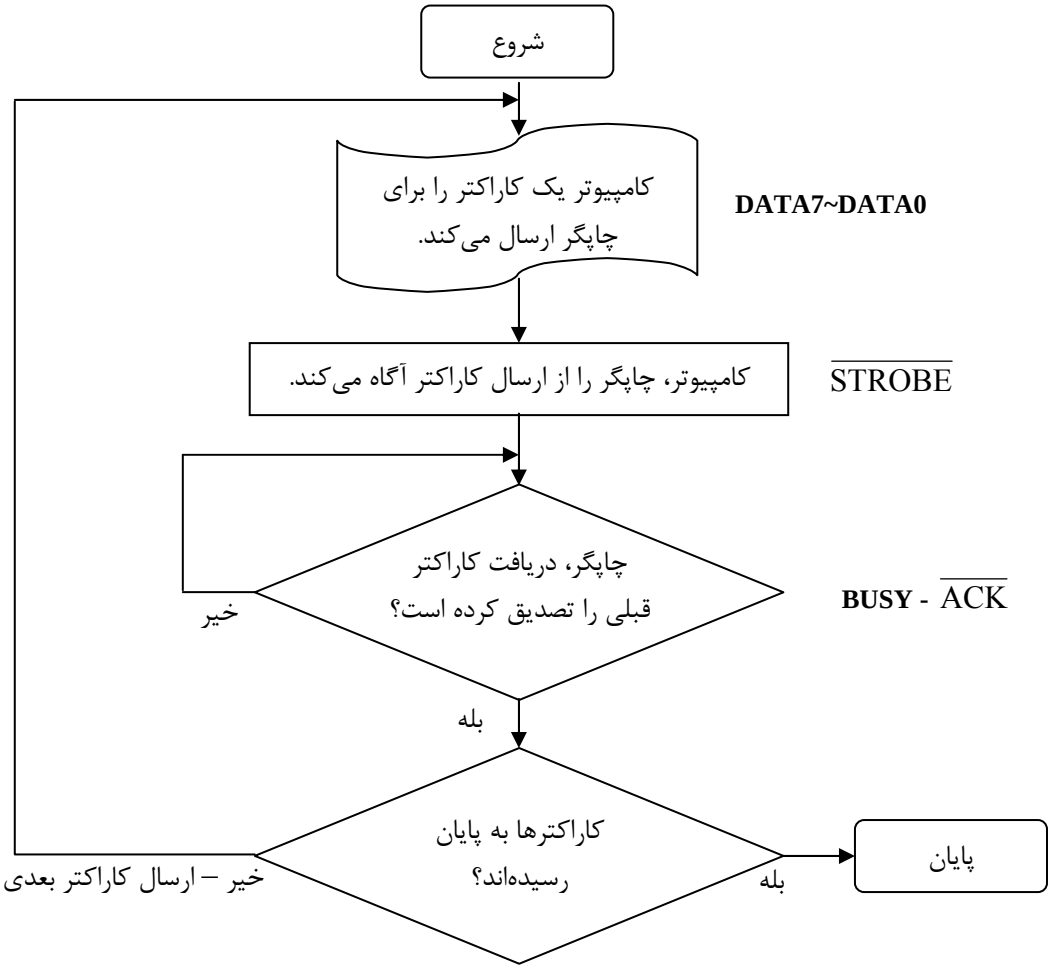
شماره پایه	عملکرد	جهت سیگنال (از دید چاپگر)	نوع سیگنال
۱	**-STROBE**	ورودی	کنترل
۲	**+DATA BIT 0**	ورودی/ خروجی	داده
۳	**+DATA BIT 1**	ورودی/ خروجی	داده
۴	**+DATA BIT 2**	ورودی/ خروجی	داده
۵	**+DATA BIT 3**	ورودی/ خروجی	داده
۶	**+DATA BIT 4**	ورودی/ خروجی	داده
۷	**+DATA BIT 5**	ورودی/ خروجی	داده
۸	**+DATA BIT 6**	ورودی/ خروجی	داده
۹	**+DATA BIT 7**	ورودی/ خروجی	داده
۱۰	**-ACK**	خروجی	وضعیت
۱۱	**+BUSY**	خروجی	وضعیت
۱۲	**+PE**	خروجی	وضعیت
۱۳	**+SLCT OUT**	خروجی	وضعیت
۱۴	**-AUTO FEED**	ورودی	کنترل
۱۵	**-ERROR**	خروجی	وضعیت
۱۶	**-INIT**	ورودی	کنترل
۱۷	**-SLCT IN**	ورودی	کنترل
۱۸-۲۵	**GND**	-	داده

چون پورت موازی از ابتدا برای اتصال چاپگر به کامپیوتر استاندارد شد، سیگنالهای این پورت نیز با نامهایی که در ارتباط کامپیوتر و چاپگر به کار می‌روند نامگذاری شده‌اند.

سه نوع سیگنال در پورت موازی کامپیوتر وجود دارد:

- **سیگنالهای داده**[1] که برای تبادل اطلاعات استفاده می‌شوند و در پورتهای موازی جدید[2] می‌توانند هم ورودی و هم خروجی باشند.
- **سیگنالهای وضعیت**[3] که برای اطلاع دادن وضعیت چاپگر به کامپیوتر به کار می‌رود. همانطور که در جدول می‌بینید، جهت تمام سیگنالهای وضعیت، خروجی از چاپگر و ورودی به کامپیوتر است.
- **سیگنالهای کنترل**[4] که برای کنترل چاپگر توسط کامپیوتر به کار می‌رود. همانطور که در جدول می‌بینید، جهت تمام سیگنالهای کنترل، ورودی به چاپگر و خروجی از کامپیوتر است.

هنگامی که کامپیوتر به کمک یک چاپگر می‌خواهد یک جمله را چاپ کند، باید کاراکترهای آن جمله را یکی‌یکی و با کنترل جریان داده برای چاپگر ارسال کند تا چاپگر فرصت چاپ هر کاراکتر را داشته باشد. در ارتباط کامپیوتر باچاپگر، از روش دست‌دهی برای کنترل جریان داده استفاده می‌کند. برای ارسال داده‌ها به روش دست‌دهی گامهای زیر باید برداشته شوند:



[1] ***Data***

[2] با استانداردهای EPP و ECP که بعداً خواهیم دید.

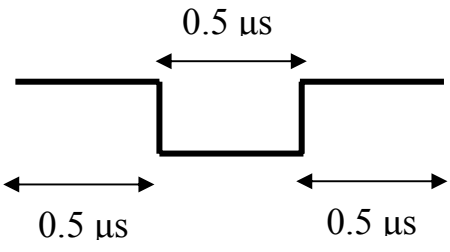
[3] ***Status***

[4] ***Control***

برای شرح سیگنالهای پورت موازی، ابتدا به سیگنالهایی می پردازیم که در این رابطه دست‌دهی شرکت دارند.

سیگنالهای داده: هشت پایه (از شماره ۲ تا شماره ۹) برای ارسال کد اسکی کاراکترها به چاپگر به کار می‌روند. پایه‌های ۱۸ تا ۲۵ نیز هشت پایه زمین هستند. هرچند استفاده از یکی از این پایه‌های زمین نیز به عنوان ولتاژ مرجع کافی است؛ اما اگر از سیمهای داده و زمین در کابل انتقال داده به صورت یکی در میان استفاده شود، در کاهش نویز تأثیر به سزایی دارد و به این صورت می‌توان فاصله کامپیوتر و چاپگر را افزایش داد. به علاوه استفاده از یک سیم زمین به ازای هر سیم داده، نویز ناشی از تداخل جریانهای برگشتی را کاهش می‌دهد.

سیگنال $\overline{\text{STROBE}}$: یک سیگنال کنترلی فعال پایین و جهت آن از کامپیوتر به چاپگر می‌باشد. هنگامی که کامپیوتر کد اسکی کاراکتری که باید چاپ شود را روی خطوط داده قرار می‌دهد، یک پالس H-L-H[1] روی این پایه ایجاد می‌کند تا چاپگر را از ارسال کاراکتر آگاه کند. کد اسکی کاراکتر باید دست‌کم ۰/۵ میکروثانیه قبل و ۰/۵ میکروثانیه بعد از فعال شدن STROBE، روی پایه‌های داده پورت موازی قرار داشته باشد. شکل بالا را ببینید.



سیگنال BUSY: یک سیگنال وضعیت فعال بالا و جهت آن از چاپگر به سمت کامپیوتر می‌باشد. هنگامی که چاپگر از طریق سیگنال STROBE متوجه می‌شود کامپیوتر برایش کاراکتری ارسال کرده، سیگنال BUSY را فعال می‌کند تا به کامپیوتر نشان دهد که مشغول چاپ کاراکتر است. کامپیوتر مرتباً این سیگنال را کنترل می‌کند و مادامی که فعال است، کاراکتر بعدی را ارسال نمی‌کند.

سیگنال $\overline{\text{ACK}}$[2]: یک سیگنال وضعیت فعال پایین و جهت آن از چاپگر به سمت کامپیوتر می‌باشد. هنگامی که چاپگر به چاپ یک کاراکتر پایان می‌دهد، یک پالس H-L-H (که دست‌کم باید ۰/۵ میکروثانیه در حالت فعال باقی بماند) روی این پایه برای کامپیوتر ارسال می‌کند. کامپیوتر با دریافت این پالس متوجه می‌شود چاپ کاراکتر قبلی به پایان رسیده و می‌تواند کاراکتر بعدی را برای چاپگر ارسال کند. معمولاً از این سیگنال در ارتباط با پایه وقفه دستگاه درخواست‌کننده چاپ (مثلاً کامپیوتر) استفاده می‌شود.

به عنوان مثال روند چاپ رشته ALI در کامپیوتر را بررسی می‌کنیم. برای این کار کامپیوتر:

---

[1] ***High to Low to High***

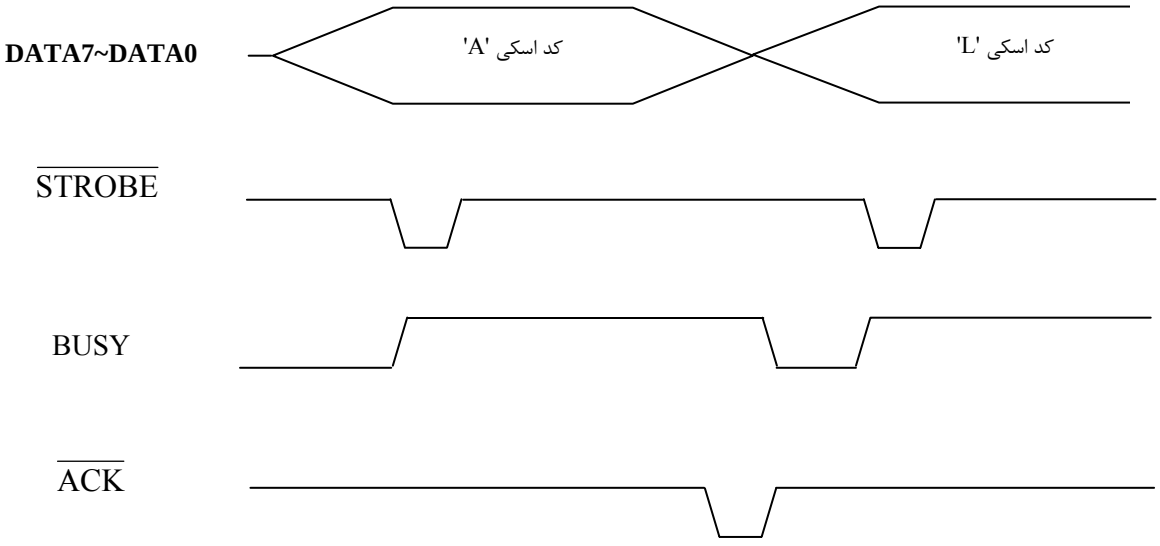
[2] مخفف Acknowledgment و به معنای تصدیق است.

- کد اسکی کاراکتر A (01000001) را روی خطوط داده پورت موازی قرار می‌دهد.
- یک پالس روی پایه STROBE ایجاد می‌کند.
- منتظر می‌ماند تا چاپگر سیگنال BUSY را غیرفعال کند و یک پالس روی سیگنال ACK برایش ارسال کند.
- کد اسکی کاراکتر L را روی خطوط داده پورت موازی قرار می‌دهد.
- یک پالس روی پایه STROBE ایجاد می‌کند.
- منتظر می‌ماند تا چاپگر سیگنال BUSY را غیرفعال کند و یک پالس روی سیگنال ACK برایش ارسال کند.
- کد اسکی کاراکتر I را روی خطوط داده پورت موازی قرار می‌دهد.
- یک پالس روی پایه STROBE ایجاد می‌کند.
- منتظر می‌ماند تا چاپگر سیگنال BUSY را غیرفعال کند و یک پالس روی سیگنال ACK برایش ارسال کند.

در سمت چاپگر چه روی می‌دهد؟ چاپگر:

- منتظر می‌ماند تا پالس STROBE را از کامپیوتر دریافت کند.
- با دریافت سیگنال STROBE، سیگنال BUSY را فعال می‌کند.
- اطلاعات مربوط به کد اسکی ارسالی توسط کامپیوتر را از روی خطوط داده می‌خواند و مشغول به چاپ آن می‌شود.
- پس از پایان چاپ، سیگنال BUSY را غیرفعال کرده و یک پالس روی پایه ACK برای کامپیوتر می‌فرستد.
- منتظر می‌ماند تا پالس STROBE را از کامپیوتر دریافت کند.
- ...

شکل زیر بخشی از نمودار زمانبندی چاپ رشته ALI را نشان می‌دهد:



وظیفه عملیاتی سیگنالهای BUSY و ACK مشابه یکدیگر است و در بعضی سیستم‌ها از اولی و در بعضی از دومی استفاده می‌شود. در زیر به معرفی سیگنالهای دیگر پورت موازی کامپیوتر می پردازیم:

سیگنال PE[1]: یک سیگنال وضعیت فعال بالا و جهت آن از چاپگر به کامپیوتر است. هنگامی که چاپگر با اتمام کاغذ مواجه شود این سیگنال را فعال می‌کند تا مراتب را به کامپیوتر اطلاع دهد.

سیگنال $\overline{\text{SLCT IN}}$: یک سیگنال کنترلی فعال پایین و جهت آن از کامپیوتر به چاپگر می‌باشد. در رابطه کامپیوتر با چاپگر این سیگنال به نشانه انتخاب شدن چاپگر باید فعال باشد.

سیگنال SLCT OUT: یک سیگنال وضعیت فعال بالا و جهت آن از چاپگر به کامپیوتر است. هنگامی که کامپیوتر برای انتخاب چاپگر سیگنال SLCT IN را فعال می‌کند، چنانچه چاپگر مشکلی برای این ارتباط نداشته باشد، در پاسخ سیگنال SLCT OUT را فعال می‌نماید.

سیگنال $\overline{\text{AUTO FEED}}$: یک سیگنال کنترلی فعال پایین و جهت آن از کامپیوتر به چاپگر می‌باشد. اگر کامپیوتر این سیگنال را فعال کرده باشد، به چاپگر فرمان می‌دهد پس از پایان چاپ یک خط (دریافت کاراکتر ۱۳ یا CR[2]) به صورت خودکار به خط بعد برود.

سیگنال $\overline{\text{ERROR}}$: یک سیگنال وضعیت فعال پایین و جهت آن از چاپگر به کامپیوتر است. اگر در حین چاپ خطایی (غیر از اتمام کاغذ) رخ دهد، چاپگر به کمک این سیگنال کامپیوتر را از وقوع این خطا مطلع می‌کند.

سیگنال $\overline{\text{INIT}}$: یک سیگنال کنترلی فعال پایین و جهت آن از کامپیوتر به چاپگر می‌باشد. در ابتدای ارتباط، کامپیوتر با ایجاد یک پالس روی این پایه (که دست‌کم باید ۵۰ میلی‌ثانیه فعال بماند)، چاپگر را Reset می‌کند. با این کار، چاپگر مجدداً مقداردهی درونی اولیه‌اش را انجام می‌دهد (مثلاً حافظه بافر داخلی‌اش را پاک می‌کند).

حال که نحوه تعامل بین کامپیوتر و چاپگر را شناختیم، به دو موضوع مهم میپردازیم. اول اینکه چگونه می‌توان بدون کمک کامپیوتر (مثلاً با یک میکروکنترلر) از چاپگر استفاده کرد و دوم اینکه چگونه به کمک پورت موازی کامپیوتر و با الهام از نحوه ارتباط بین کامپیوتر و چاپگر، می‌توان یک دستگاه جانبی (مثلاً یک نمایشگر LCD) را کنترل کرد.

---

[1] ***Paper End***
[2] ***Carriage Return***

استفاده از چاپگر بدون نیاز به کامپیوتر

در بخش قبل آموختیم که کامپیوتر چگونه برای چاپ یک رشته با چاپگر ارتباط برقرار می‌کند. حال فرض کنید می‌خواهید اطلاعات جمع‌آوری شده توسط یک دستگاه را چاپ کنید؛ مثلاً یک دستگاه کارت‌خوان که پس از خواندن بارکد روی کارت یک کارمند، باید برای او ژتون غذا چاپ کند، دستگاهی که باید دمای اتاق را هر ۱۰ ثانیه یک بار چاپ کند و .... آیا برای چاپ این اطلاعات، باید حتماً آنها را به کامپیوتر فرستاد و به کمک کامپیوتر این اطلاعات را چاپ نمود؟

مطالبی که در بخش قبل آموختیم به ما می‌گویند در این موارد نیازی به استفاده از کامپیوتر نیست. کافی است یک میکروکنترلر را طوری برنامه‌ریزی کنید که برای چاپ یک رشته اطلاعات، دقیقاً رفتاری مشابه کامپیوتر با چاپگر داشته باشد. به این صورت بدون اینکه نیازی به کامپیوتر باشد می‌توانید اطلاعات مورد نظر خود را به کمک چاپگر چاپ کنید.

به عنوان مثال، رشته ALI را به کمک میکروکنترلر ۸۰۵۱ چاپ می‌کنیم. برای این کار اتصالات مقابل را بین میکروکنترلر و چاپگر برقرار کنید. توجه کنید که پورت P0 میکروکنترلر برای تبادل داده با چاپگر، بیتهای پورت P1 برای ارسال سیگنالهای کنترلی و بیتهای پورت P2 برای دریافت سیگنالهای وضعیت چاپگر مورد استفاده قرار گرفته‌اند. دقت کنید سر کابل چاپگر چون باید داخل پورت موازی کامپیوتر قرار گیرد دارای ۲۵ میله است که شماره‌گذاری پایه‌های آن معکوس آینه‌ای پورت پشت کامپیوتر است (یعنی جای چپ و راست عوض می‌شود). اما چون شما می‌خواهید میکروکنترلر خود را به این کابل متصل کنید، باید از یک قطعه DB25 دارای حفره (مانند شکل مقابل) استفاده کنید. شماره‌گذاری پایه‌های این قطعه (که در

شماره پایه	عملکرد	پورت میکروکنترلر ۸۰۵۱
۱	-STROBE	P1.0
۲	+DATA BIT 0	P0.0
۳	+DATA BIT 1	P0.1
۴	+DATA BIT 2	P0.2
۵	+DATA BIT 3	P0.3
۶	+DATA BIT 4	P0.4
۷	+DATA BIT 5	P0.5
۸	+DATA BIT 6	P0.6
۹	+DATA BIT 7	P0.7
۱۰	-ACK	P2.0
۱۱	+BUSY	P2.1
۱۲	+PE	P2.2
۱۳	+SLCT OUT	P2.3
۱۴	-AUTO FEED	P1.1
۱۵	-ERROR	P2.4
۱۶	-INIT	P1.2
۱۷	-SLCT IN	P1.3
۱۸-۲۵	GND	زمین میکروکنترلر

جدول بالا آمده است) مانند پورت پشت کامپیوتر است.

برنامه زیر برای میکروکنترلر ۸۰۵۱ نوشته شده تا با فرض وجود اتصالات بالا، رشته ALI را به کمک چاپگر چاپ کند.

```
ORG      0
MOV      P2,#255    ; Define P2 as input (for reading status signals)
CLR      P1.3       ; Activating –SLCT IN
JNB      P2.3,$     ; Wait until +SLCT OUT is actiated by printer
CLR      P1.2       ; Activating -INIT
ACALL    DELAY      ; Wait
SETB     P1.2       ; Deactivating –INIT to make a pulse to reset the printer
JNB      P2.3,$     ; Wait until +SLCT OUT is actiated by printer
JB       P2.1,$     ; Wait until +BUSY is deactiated by printer

MOV      P0,#'A'    ; Put the ASCII code of 'A' on DATA lines
CLR      P1.0       ; Activating –STROBE
ACALL    DELAY      ; Wait
SETB     P1.0       ; Deactivating –STROBE to make a pulse
JB       P2.1,$     ; Wait until +BUSY is deactiated by print

MOV      P0,#'L'    ; Put the ASCII code of 'L' on DATA lines
CLR      P1.0       ; Activating –STROBE
ACALL    DELAY      ; Wait
SETB     P1.0       ; Deactivating –STROBE to make a pulse
JB       P2.1,$     ; Wait until +BUSY is deactiated by print

MOV      P0,#'I'    ; Put the ASCII code of 'I' on DATA lines
CLR      P1.0       ; Activating –STROBE
ACALL    DELAY      ; Wait
SETB     P1.0       ; Deactivating –STROBE to make a pulse
JB       P2.1,$     ; Wait until +BUSY is deactiated by print

JMP      $

DALEY:
MOV    R1,#50
AGAIN:
       NOP
DJNZ    R1, AGAIN
RET

END
```

توجه کنید در این برنامه تنها از سیگنال BUSY برای کنترل وضعیت چاپگر استفاده شده است. می‌توانید از سیگنال ACK همزمان با BUSY استفاده کنید.

پرسش) با استفاده از ACK به جای BUSY و اتصال آن به پایه وقفه میکروکنترلر می‌توانید برنامه حرفه‌ای‌تری بنویسید. برنامه بالا را با این فرض اصلاح کنید.

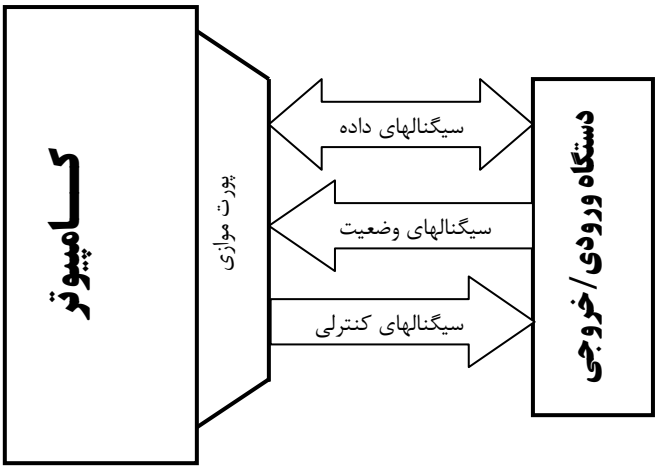
پرسش) در برنامه بالا فرض کرده‌ایم فرآیند چاپ بدون هیچ مشکلی انجام شود. برای حرفه‌ای‌تر شدن برنامه می‌توانید هنگامی که منتظر غیرفعال شدن سیگنال BUSY هستید، سیگنالهای PE و ERROR را نیز چک کنید و در صورت فعال شدن آنها واکنش مناسب را نشان دهید. به علاوه می‌توانید زمان انتظار برای غیرفعال شدن BUSY را نیز اندازه بگیرید و اگر بیش از حد عادی به طول انجامید پیغام خطا صادر کنید. برنامه میکروکنترلر را طوری اصلاح کنید که این موارد را نیز به انجام برساند.

دقت کنید مطالب گفته شده تنها در مورد چاپگرهای سوزنی (ماتریسی) صادق است. نمونه‌ای از کاربرد این نوع چاپگر پرصدا در بانکها برای چاپ اطلاعات روی قبوض آب و برق و ... است. اگر یک چاپگر سوزنی را به صورت گفته شده به میکروکنترلر ۸۰۵۱ متصل کنید و برنامه بالا را روی میکروکنترلر اجرا کنید، رشته ALI به چاپ می‌رسد. چاپگرهای جوهرافشان و لیزری دارای پردازنده‌های خاصی داخل خود هستند و برای تبادل اطلاعات با آنها باید از زبانهای برنامه‌نویسی مخصوص استفاده کرد.

کنترل دستگاههای جانبی به کمک پورت موازی کامپیوتر

در بخش قبل آموختیم چگونه کامپیوتر با چاپگر رفتار می‌کند. آیا می‌توان دستگاهی دیگر غیر از چاپگر (مثلاً یک تابلو تبلیغات، یک روبات یا ...) را با پورت موازی کامپیوتر کنترل کرد؟ خوشبختانه پاسخ مثبت است. به کمک برنامه‌نویسی پورت موازی و با طراحی مناسب سخت‌افزار می‌توان به سادگی دستگاههای جانبی را به کمک این پورت کنترل کرد. جزییات این کار، موضوع این بخش است.

دیدیم که در پورت موازی کامپیوتر سه دسته سیگنال داده، وضعیت و کنترل وجود دارند. برای طراحی دستگاهی که به پورت موازی کامپیوتر متصل شود، از سیگنالهای داده برای ارتباط داده‌ای، از سیگنالهای کنترلی برای کنترل دستگاه و از سیگنالهای وضعیت برای بررسی وضعیت دستگاه استفاده می‌شود. شکل زیر این ارتباط را نشان می‌دهد.



متناظر با این سه دسته سیگنال، هر پورت موازی در کامپیوتر دارای سه ثبات داده[1] و وضعیت[2] و کنترل[3] است که بیتهای این ثباتها مرتبط با سیگنالهای متناظر در پورت موازی کامپیوتر هستند. از طریق برنامه‌ریزی این سه ثبات می‌توان پورت موازی کامپیوتر را برنامه‌نویسی کرد.

جدول زیر بیتهای این سه ثبات را نشان می‌دهد:

ثبات کنترلی			ثبات وضعیت			ثبات داده		
عملکرد	Low	High	عملکرد	Low	High	عملکرد	Low	High
Not Used	-	-	**Busy**	**Busy**	**Not Busy**	Data7	0	1
Not Used	-	-	Acknowledge	Ack	Nack	Data6	0	1
Direction	Output	Input	Paper Status	Paper	No Paper	Data5	0	1
Interrupt Control	Interrupt Disabled	Interrupt Enabled	Selection Status	Not Selected	Selected	Data4	0	1
Select	**Not Selected**	**Selected**	Error Status	Error	No Error	Data3	0	1
Initialize	Enable	Disable	IRQ	True	False	Data2	0	1
Auto Feed	**Disable**	**Enable**	Not Used	-	-	Data1	0	1
Strobe	**Disable**	**Enable**	Not Used	-	-	Data0	0	1

همانطور که در جدول بالا می‌بینید:

- سیگنالهای داده پورت موازی به بیتهای ثبات داده پورت موازی مربوط هستند؛ یعنی هرچه در این ثبات توسط کامپیوتر نوشته شود از طریق سیگنالهای داده پورت موازی به صورت سیگنالهای دیجیتال الکترونیکی قابل دسترسی است. به همین صورت اگر به شیوه‌ای که بعداً می‌بینیم، از پورت موازی به عنوان ورودی استفاده شود، برای خواندن عدد دیجیتالی که به سیگنالهای داده پورت موازی متصل شده کافی است ثبات داده پورت موازی را بخوانیم.
- بیتهای ثبات وضعیت با سیگنالهای وضعیت پورت موازی متناظرند. در واقع کامپیوتر برای خواندن سیگنالهای وضعیت، ثبات وضعیت را می‌خواند. مثلاً اگر بیت شماره ۳ ثبات وضعیت صفر باشد، یعنی سیگنال $\overline{\text{ERROR}}$ پورت موازی توسط دستگاه

[1] ***Data Register***
[2] ***Status Register***
[3] ***Control Register***

خارجی (چاپگر یا دستگاه دیگر) فعال (صفر) شده است. اما سیگنال BUSY (که با حروف پررنگ‌تر مشخص شده است) داستانی متفاوت دارد؛ حالت بیت BUSY در ثبات وضعیت، عکس حالت سیگنال BUSY در پورت موازی است؛ به بیان دیگر دستگاه خارجی برای غیرفعال کردن سیگنال فعال بالای BUSY باید آن را «صفر» کند؛ در این حال بیت ویژه BUSY در ثبات وضعیت «یک» خواهد بود. بنابراین کامپیوتر مرتباً ثبات وضعیت را می‌خواند و تا زمانی که **بیت** BUSY «صفر» است (**سیگنال** BUSY «یک» است) منتظر می‌ماند.

- برای تغییر سیگنالهای کنترلی پورت موازی باید از ثبات کنترلی استفاده کنیم. به عنوان مثال برای فعال کردن سیگنال INIT کافی است بیت شماره ۲ این ثبات را صفر کنیم. سه بیت این ثبات که با حروف پررنگ‌تر مشخص شده‌اند، وضعیتی مانند سیگنال BUSY دارند و حالت بیتی آنها عکس حالت سیگنال معادل آنهاست. مثلاً برای فعال کردن سیگنال $\overline{\text{STROBE}}$ باید در بیت معادل آن در ثبات کنترلی «یک» بنویسیم؛ با این کار سیگنال فوق در پورت موازی «صفر» می‌شود.

بیت شماره ۵ ثبات کنترلی اهمیتی خاص دارد. اگر بخواهید از پورت موازی کامپیوتر که به صورت پیش‌فرض در حالت خروجی قرار دارد به عنوان ورودی استفاده کنید، باید این بیت را در ثبات کنترلی «یک» کنید.

بنابراین برای کنترل هر دستگاهی که به پورت موازی متصل است، باید با سیگنالهای این پورت به کمک ثباتهای ذکر شده کار کنیم. اما چگونه می‌توان به این ثباتها دسترسی داشت؟

آدرس ثباتهای پورت موازی

هر پورت موازی در کامپیوتر دارای یک آدرس پایه در فضای آدرس‌دهی ورودی/خروجی است که سه ثبات ذکر شده با شروع از این آدرس قرار می‌گیرند. اگر این آدرس را Base بنامیم، آدرس ثبات داده پورت موازی برابر Base، آدرس ثبات وضعیت پورت موازی برابر Base+1 و آدرس ثبات کنترل پورت موازی برابر Base+2 خواهد بود. مثلاً اگر آدرس پورت موازی کامپیوتر برابر 0378h[1] باشد (که در اکثر کامپیوترهای IBM چنین است)، آنگاه آدرس ثبات داده پورت موازی برابر 0378h، آدرس ثبات وضعیت پورت موازی برابر 0379h و آدرس ثبات کنترل پورت موازی برابر 037Ah خواهد بود. به بیان دیگر برای تبادل داده با پورت موازی با ثباتی به آدرس 0378h، برای خواندن سیگنالهای وضعیت پورت موازی با ثباتی به آدرس

[1] به یاد دارید که آدرسهای ورودی/خروجی در کامپیوتر ۱۶ بیتی هستند.

0379h و برای ایجاد تغییر در سیگنالهای کنترلی پورت موازی با ثباتی به آدرس 037Ah سروکار داریم.

در کامپیوترهای امروزی اغلب یک پورت موازی وجود دارد که آدرس مبنای آن معمولاً 0378h است. در کامپیوترهای قدیمی‌تر به کمک کارت I/O امکان اضافه کردن پورتهای موازی بیشتری به کامپیوتر نیز وجود داشت. در صورت وجود دو پورت موازی، آدرس LPT1 و LPT2 به ترتیب برابر 0378h و 0278h یا 03BCh و 0378h بود. در انتهای این بخش چگونگی به دست آوردن آدرسهای مزبور در ویندوز را خواهیم دید.

به کمک BIOS نیز می‌توان آدرس مبنای ثباتهای ورودی/خروجی پورت موازی را دانست. در واقع این آدرسها از آفست 0008 به بعد سگمنت 0040 حافظه ذخیره می‌شود. جدول مقابل را ببینید.

پورت LPT	آدرس پایه ورودی/خروجی ثباتهای پورت
LPT1	0040:0008 – 0040:0009
LPT2	0040:000A – 0040:000B
LPT3	0040:000C – 0040:000D
LPT4	0040:000E – 0040:000F

مثلاً دستور *D 0040:0008 L8* در نرم‌افزار Debug، آدرس پایه پورتهای موازی نصب شده در کامپیوتر را نشان می‌دهد. قبلاً با نرم‌افزار Debug و فرمانهای آن آشنا شدیم.

به کمک برنامه‌نویسی C نیز می‌توان آدرسهای ذخیره شده در این بخش حافظه را مشاهده کرد. برنامه زیر، آدرس پورت LPT1 و در صورت عدم نصب پورت مزبور، پیغامی را نمایش می‌دهد.

```
#include    <STDIO.H>
#include    <DOS.H>

void   main(){
      unsigned int far *fptr;
      fptr = (unsigned int far *) 0x00000408;
      if (*fptr > 0)
            printf("LPT1 I/O base address = %X",*fptr);
      else
            printf("LPT1 not found");
}
```

برنامه‌نویسی پورت موازی

برای این کار باید از ثباتهای داده و وضعیت و کنترل پورت موازی استفاده کنیم. سه تابع برای این کار پیشنهاد می‌کنیم:

ارسال داده‌ها به پورت موازی

```
void  SendData(int  PortID, unsigned char  Data){
      outportb(PortID, Data);
}
```

قبلاً با توابع outportb و outp برای ارسال یک بایت به پورتی با آدرس مشخص آشنا شدیم[1]. برای استفاده از این توابع، شامل کردن فایلهای سرآیند مناسب را فراموش نکنید. مثلاً برای ارسال عدد 55h به پورت موازی از دستور ;(SendData(0x0378,0x55 استفاده می‌کنیم. این دستور عدد 55h (01010101) را در ثبات داده پورت موازی می‌نویسد. با این کار عدد فوق روی سیگنالهای داده این پورت نمایش داده خواهد شد.

بررسی وضعیت یک بیت ثبات وضعیت

```
int    Bit_Status(int   PortID, int BitNumber){
       unsigned char status, temp;
       status = inportb(PortID);
       temp = 1;
       temp = temp << BitNumber;
       if ((status & temp) > 0)
            return 1;
      else
         return 0;
}
```

قبلاً با توابع inportb و inp برای دریافت یک بایت از پورتی با آدرس مشخص آشنا شدیم[2]. برای استفاده از این توابع، شامل کردن فایلهای سرآیند مناسب را فراموش نکنید. تابع Bit_Status مقدار بیت شماره BitNumber ثباتی به آدرس PortID را برمی‌گرداند. به عنوان مثال، دستور ;(printf("%d", Bit_Status(0x0379, 5)) مقدار بیت شماره ۵ ثبات وضعیت (بیت Paper Status) را نمایش می‌دهد. این کار چگونه انجام می‌شود؟

ابتدا مقدار پورت به کمک دستور inportb خوانده و در متغیر status ذخیره می‌شود. برای بررسی مقدار بیت شماره ۵ این متغیر، باید کاری کرد که مقدار تمام بیتهای آن غیر از بیت شماره ۵ صفر شود. دستور زیر این کار را انجام می‌دهد:

```
          D7  D6  D5  D4  D3  D2  D1  D0
AND
           0   0   1   0   0   0   0   0
         ---------------------------------------------
           0   0  D5   0   0   0   0   0
```

[1] در فصل قبل نحوه استفاده از دستور OUT در زبان اسمبلی برای ارسال داده‌ها به یک پورت خروجی را مشاهده کردید.

[2] در فصل قبل نحوه استفاده از دستور IN در زبان اسمبلی برای دریافت داده‌ها از یک پورت ورودی را مشاهده کردید.

حال اگر مقدار حاصل AND برابر صفر باشد نشان‌دهنده این است که D5 نیز صفر بوده است. اگر حاصل مقداری غیر از صفر داشته باشد (۳۲) یعنی بیت D5 یک بوده است.
در تابع بالا متغیر temp با مقدار اولیه 1 تعریف شده است. برای بررسی بیت شماره BitNumber، مقدار temp به تعداد BitNumber به سمت چپ شیفت داده شده و حاصل AND آن با متغیر status بررسی می‌شود.

مثال) قطعه برنامه زیر، اعداد صفر تا ۲۵۵ را برای دستگاهی که به پورت موازی کامپیوتر متصل است، ارسال می‌کند. بعد از ارسال هر عدد، کامپیوتر باید منتظر بماند تا دستگاه مقابل سیگنال Busy را غیرفعال کند:

```
void main(void){
	for (unsigned char num = 0;num <= 255;num++){
		SendData(0x378,num);
		while (!Bit_Status(0x379,7));
	}
}
```

تغییر مقدار یک بیت ثبات کنترلی

```
void  Change_Bit(int  PortID, int BitNumber, int value){
	unsigned char In, temp, Out;
	In = inportb(PortID);

	temp = 1;
	temp = temp << BitNumber;

	if  (value == 1)
		Out = In | temp;

	else  // value = 0
		Out = In & (~temp);

	outportb(PortID, Out);
}
```

تابع فوق مقدار بیت شماره BitNumber ثباتی به آدرس PortID را بر اساس پارامتر value تغییر می‌دهد. مثلاً اجرای تابع ;Change_Bit(0x37A,5,1) باعث می‌شود بیت شماره ۵ ثبات کنترلی پورت موازی (بیت Direction) یک شود. همانطور که قبلاً دیدیم، این کار باعث می‌شود پورت موازی در حالت ورودی عمل کند. برای صفر کردن این بیت می‌توان از دستور ;Change_Bit(0x37A, 5,0) استفاده کرد.

همانطور که در تابع فوق دیده می‌شود برای تغییر یک بیت از یک ثبات، ابتدا مقدار فعلی آن ثبات را می‌خوانیم، مقدار بیت مورد نظر را تغییر می‌دهیم و مجدداً مقدار جدید را در ثبات می‌نویسیم. با این کار فقط مقدار بیت مورد نظر ما تغییر می‌کند.

پرسش) با توجه به توضیحات تابع Bit_Status، راجع به نحوه عملکرد تابع Change_Bit توضیح دهید.

اتصال دستگاههای خروجی به پورت موازی

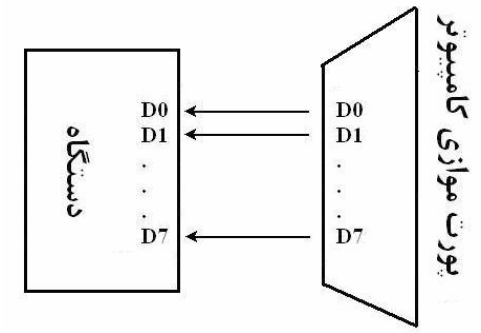
اکنون آماده هستیم نحوه اتصال دستگاههای خارجی (غیر از چاپگر) به پورت موازی و کنترل آنها را شرح دهیم. در حال حاضر فقط در مورد ارسال داده‌ها به دستگاه خارجی از طریق پورت موازی صحبت می‌کنیم و در ادامه به نحوه دریافت داده‌ها از این پورت نیز خواهیم پرداخت.

پیشتر دیدیم که برای ارسال داده‌های متوالی از فرستنده به گیرنده، به روشهای کنترل جریان داده نیازمندیم تا اطمینان حاصل شود حجم انتقال داده‌ها از فرستنده به گیرنده کنترل شده است و گیرنده فرصت پردازش آنها را دارد.

اکنون می‌خواهیم دستگاهی طراحی کنیم که از طریق پورت موازی یک بلوک اطلاعات (مثلاً ۱۰۰ بایت داده) برای آن ارسال شود. به عنوان مثال دستگاهی طراحی می‌کنیم که بایتهای داده را از پورت موازی کامپیوتر دریافت و در حافظه داخلی‌اش ذخیره کند. بدیهی است دستگاه برای ذخیره هر بایت نیاز به زمان دارد و کامپیوتر نباید بدون وقفه داده‌ها را برای آن ارسال کند.

بنابراین باید در برنامه‌ای که در کامپیوتر برای ارسال این داده‌ها به پورت موازی اجرا می‌شود، مکانیسمی برای کنترل جریان داده پیش‌بینی شود. قبلاً دو مکانیسم تأخیر و دست‌دهی را در مبحث کنترل جریان داده بررسی کردیم. نحوه طراحی دستگاه فوق با دو مکانیسم فوق را بررسی می‌کنیم.

کنترل جریان داده به شیوه تأخیر



در این روش، در برنامه کامپیوتری بین هر ارسال داده به پورت موازی مقداری تأخیر ایجاد می‌کنیم تا دستگاهی که به این پورت متصل است، فرصت کافی برای دریافت و ذخیره این اطلاعات را داشته باشد. برنامه مقابل اعداد یک تا ۱۰۰ را با کنترل جریان داده به روش تأخیر برای پورت موازی ارسال می‌کند.

```
#include    <STDIO.H>
#include    <STDLIB.H>
#include    <DOS.H>

void   SendData(int PortID, unsigned char Data){
     outportb(PortID, Data);
}

void   main(){

     SendData(0x0378, 0x55); // Start of Data Transfer
     delay(1000);

     for (unsigned char i = 1; i < 101; i++){
          SendData(0x0378, i);
          delay(1000);
     }

     SendData(0x0378, 0xAA); // End of Data Transfer
}
```

حلقه اصلی برنامه، یک بایت را به پورت موازی ارسال می‌کند و سپس یک ثانیه تأخیر ایجاد می‌کند تا دستگاهی که به این پورت متصل است فرصت دریافت و ذخیره این بایت را داشته باشد. سپس بایت بعدی را ارسال می‌کند.

چرا در ابتدای ارتباط عدد 55h (01010101) و در انتها عدد AAh (10101010) برای دستگاه ارسال شده است؟ چون اطلاعات به روش تأخیر برای پورت ارسال می‌شود، دستگاه باید در فواصل زمانی مشخص (که برابر با تأخیر تعبیه شده در برنامه کامپیوتری است)، سیگنالهای داده پورت موازی را برای دریافت اطلاعات ارسالی بخواند. نکته مهم این است که در برنامه‌ریزی دستگاه، باید نحوه آغاز و پایان ارتباط به نحوی مشخص شود تا دستگاه بداند از چه زمانی دریافت اطلاعات را آغاز کند. در این ارتباط ما از این قرارداد استفاده کرده‌ایم که کامپیوتر در ابتدای ارتباط عدد 55h (01010101) و در انتها عدد AAh (10101010) را ارسال می‌کند. دستگاه نیز باید از همین قرارداد پیروی کند. توجه کنید اعداد فوق (55h و AAh) نشانه‌اند و نباید جزء داده‌های ارسالی باشند.

شبه‌کد برنامه دستگاه را ببینید.

```
Wait to receive 55h;// Start of communication
Delay (1500);       // Wait untill the first data is ready

Repeat {
     Receive new data from Parallel Port;

     If data = AAh then
          exit the loop; // End of communication
     else {
          Store Data;
          Delay (1000);
     }
}
```

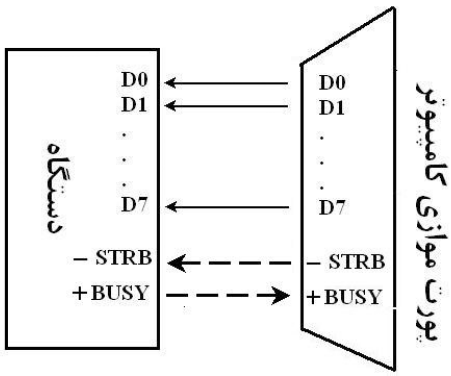
پرسش) چرا بعد از دریافت عدد نشانه 55h به میزان ۱۵۰۰ میلی‌ثانیه و در دفعات بعدی به میزان ۱۰۰۰ میلی‌ثانیه تأخیر ایجاد می‌شود؟

روش فوق همانطور که قبلاً دیدیم ساده، اما دارای مشکلاتی است. ممکن است دستگاه نیازی به یک ثانیه زمان برای ذخیره داده نداشته باشد که در این حالت زمان به هدر خواهد رفت. از طرف دیگر ممکن است به دلیل بروز مشکلی، دستگاه نتواند طی یک ثانیه بایت دریافتی را ذخیره کند؛ در این حال کامپیوتر بدون توجه به این مشکل بایت بعدی را ارسال خواهد کرد.

کنترل جریان داده به شیوه دست‌دهی

راه فنی‌تر و البته پیچیده‌تر کنترل جریان داده استفاده از سیگنالهای دست‌دهی است. در این روش دو سیگنال دست‌دهی به سیگنالهای داده اضافه می‌شوند.

برنامه کامپیوتری باید:



- داده را ارسال کند.
- یک پالس روی سیگنال STROBE ارسال کند تا دستگاهی را که به پورت موازی متصل است، از ارسال داده مطلع کند.
- منتظر بماند تا سیگنال BUSY از سوی دستگاه مقابل غیرفعال شود.
- داده بعدی را ارسال کند.
- .....

اکنون می‌خواهیم برنامه قبلی را با فرض اتصال سیگنالها به صورت شکل بالا بنویسیم. برنامه صفحه بعد، اعداد یک تا ۱۰۰ را با کنترل جریان داده به روش دست‌دهی برای پورت موازی ارسال می‌کند.

دستگاه متصل به پورت موازی باید طوری برنامه‌ریزی شده باشد که در ابتدا سیگنال BUSY را غیرفعال کند و سپس:

- منتظر دریافت پالس روی سیگنال STROBE بماند.
- با دریافت این پالس، سیگنال BUSY را فعال کند.
- سیگنالهای داده را بخواند و پردازش اطلاعات را آغاز کند.

- پس از پایان پردازش، سیگنال BUSY را غیر فعال کند.
- منتظر دریافت سیگنال STROBE بعدی بماند.
- ...

در رابطه دست‌دهی، به جای سیگنال BUSY می‌توان از سیگنال ACK هم استفاده کرد. شبه‌کد برنامه دستگاه را در زیر می‌بینید:

```
Inactivate BUSY signal

Reapeat{
      Wait for a pulse on STROBE signal
      Activate BUSY signal
      Read DATA signals and process the data unit
      Inactivate BUSY signal
}
```

```
#include <STDIO.H>
#include <STDLIB.H>
#include <DOS.H>

void SendData(int   PortID, unsigned char   Data){
     outportb(PortID, Data);
}

int  Bit_Status(int  PortID, int BitNumber){
     unsigned char status, temp;
     status = inportb(PortID);
     temp = 1;
     temp = temp << BitNumber;
     if ((status & temp) > 1)
        return 1;
     else
         return 0;
}

void Change_Bit(int  PortID, int BitNumber, int value){
     unsigned char In, temp, Out;
     In = inportb(PortID);

     temp = 1;
     temp = temp << BitNumber;

     if   (value == 1)
          Out = In | temp;

     else  // value = 0
          Out = In & (~temp);

     outportb(PortID, Out);
}

void main(){

      unsigned char status;

      Change_Bit(0x037A, 0, 0);   // Set STROBE bit to 0 (disabled)

      for (unsigned char i = 1 ; i < 101 ; i++){

           // Send data
          SendData(0x0378, i);

          // Send a pulse on STROBE pin
          Change_Bit(0x037A, 0, 1); // Set STROBE bit to 1 (Enabled)
          delay(1);
          Change_Bit(0x037A, 0, 0);// Set STROBE bit to 0 (disabled)

          //  wait while BUSY bit = 0 (active)
          while (Bit_Status(0x0379, 7) == 0);
     }
}
```

توجه کنید چون آغاز ارتباط با ارسال پالس روی STROBE مشخص می‌شود، مانند برنامه قبل نیازی نیست به کمک ارسال بایتهای خاص، آغاز ارتباط را معلوم کنیم. در صورت لزوم می‌توانید مانند قبل روشی برای پایان ارتباط به کار ببندید.

آیا می‌توان از سیگنالهای دیگر پورت موازی نیز در این ارتباط استفاده کرد؟

پرسش) برنامه کامپیوتری را طوری اصلاح کنید که در ابتدای کار یک پالس روی پایه INIT ارسال کند تا دستگاه مقابل reset شود. سیگنال INIT پورت موازی باید به پایه reset تراشه‌های دستگاه متصل شود.

پرسش) برنامه کامپیوتری را طوری اصلاح کنید که در حین انتظار برای غیرفعال شدن سیگنال BUSY، سیگنال ERROR را هم چک کند و در صورت فعال شدن آن، با صادر کردن پیغام مناسب از برنامه خارج شود. سیگنال ERROR پورت موازی را در دستگاه به یکی از پایه‌های تراشه کنترل‌کننده دستگاه متصل می‌کنیم. اصلاحات لازم را در برنامه تراشه فوق نیز طوری انجام دهید که در صورت بروز خطا در سیستم (مثلاً به طول انجامیدن بیش از حد ذخیره یک بایت) پایه فوق را فعال کند.

توجه کنید که سیگنالهای کنترلی پورت موازی برای استفاده در رابطه با دستگاههای دیگر باید با مقاومت 4.7 KΩ بالا کشیده[1] شوند.

استفاده از پورت موازی به عنوان ورودی

در کامپیوترهای ابتدایی IBM، از پورت موازی تنها به هدف ارسال اطلاعات به چاپگر و به صورت خروجی استفاده می‌شد. این استاندارد به گونه‌ای که به زودی خواهیم دید SPP نامیده می‌شد. در این استاندارد خواندن اطلاعات از پورت موازی (استفاده از پورت موازی به عنوان پورت ورودی) مجاز نیست و باعث آسیب رسیدن به بورد اصلی کامپیوتر می‌شود.
در استانداردهای بعدی که EPP و ECP نامیده می‌شوند، خواندن از پورت موازی نیز تعریف شده است. برای این کار باید مراحل زیر انجام شوند:

- بیت Direction ثبات کنترلی باید «یک» شود.
- پینهای داده را با مقاومت 10 KΩ بالا بکشید.
- بین سیگنالهای پورت موازی و دستگاه خارجی از بافر استفاده کنید تا چنانچه دستگاه دچار مشکلی شد و جریان زیادی کشید، به پورت موازی و بورد اصلی کامپیوتر شما آسیب نرساند.

[1] ***Pull-up***

- در بعضی پورتهای موازی، بهتر است قبل از هر عملیات خواندن پورت، یک بایت 11111111 (FFh) برای این پورت ارسال کنید.

مثال کاربردی – قفل سخت‌افزاری

تاکنون راههای زیادی برای نصب قفل روی نرم‌افزارها و جلوگیری از کپی غیرمجاز آنها ارائه شده است که به جرأت می‌توان گفت اکثر آنها با ناکامی مواجه شده‌اند. این قفلهای نرم‌افزاری با وجود ملاحظات امنیتی بسیار تاکنون نتوانسته‌اند از دسترسی‌های غیرمجاز به نرم‌افزارها جلوگیری کنند.

یک راه مطمئن (و البته هزینه) برای جلوگیری از کپی غیرمجاز نرم‌افزارها استفاده از قفلهای سخت‌افزاری است. در این روش، تولیدکننده نرم‌افزار به همراه لوح فشرده حاوی نرم‌افزار، یک دستگاه کوچک به نام «قفل سخت‌افزاری» نیز ارائه می‌کند که در یکی از پورتهای کامپیوتر نصب می‌شود. نرم‌افزار فوق را فقط وقتی می‌توان مورد استفاده قرار داد که قفل مزبور داخل پورت کامپیوتر نصب شده باشد. با این روش، دیگر نمی‌توان کپی‌های غیرمجاز دیگری از نرم‌افزار فوق تهیه کرد.

کاربرد دیگر قفلهای سخت‌افزاری در سیستم‌های امنیتی است. فرض کنید پایگاه اطلاعاتی مهمی در یک سازمان وجود دارد که تنها مدیران ارشد آن سازمان باید به آن دسترسی داشته باشند. می‌توان به تمام اشخاص مجاز یک قفل سخت‌افزاری داد و نرم‌افزار را به گونه‌ای طراحی کرد که استفاده از آن منوط به نصب قفل فوق در پورت کامپیوتر باشد. به بیان دیگر، تنها هنگامی می‌توان به این پایگاه اطلاعاتی دسترسی داشت که قفل در پورت نصب شده باشد. برای بالاتر بردن امنیت سیستم، معمولاً از روشهای نرم‌افزاری در کنار قفل سخت‌افزاری استفاده می‌شود.

قفل سخت‌افزاری چگونه کار می‌کند؟

روش معمول در طراحی این قفلها این است که ارتباطی بین قفل و نرم‌افزار کامپیوتری قرارداد می‌شود که در صورت برقراری آن ارتباط دسترسی به نرم‌افزار انجام خواهد شد.

طراحی قفل سخت‌افزاری با پورت موازی کامپیوتر

مثلاً فرض کنید نرم‌افزار کامپیوتری را به گونه‌ای طراحی می‌کنیم که در ابتدای کار، پنج عدد تصادفی تولید و برای پورت موازی ارسال کند و منتظر بماند تا عددی از این پورت دریافت کند. سپس عدد دریافتی را با مجموع پنج عددی که ارسال کرده مقایسه می‌کند و در صورت تساوی، اجازه ادامه کار را صادر می‌کند.

قفل سخت‌افزاری که به پورت موازی کامپیوتر متصل است باید پنج عدد از این پورت دریافت و با هم جمع کند و نتیجه را برای پورت موازی ارسال کند.
با ترکیب این قفل و نرم‌افزار کامپیوتری، یک روش امنیتی برای دسترسی به سیستم ایجاد می‌شود. اگر قفل در پورت موازی باشد، این ارتباط به درستی صورت می‌گیرد؛ وگرنه عددی که نرم‌افزار کامپیوتری از پورت دریافت می‌کند برابر مجموع پنج عددی که ارسال کرده نیست و مجوز ادامه کار با نرم‌افزار صادر نمی‌شود.
برنامه صفحه بعد نمونه‌ای از استفاده از این روش را نشان می‌دهد.

توجه داشته باشید همانطور که قبلاً گفته شد، این برنامه صرفاً تحت DOS یا ویندوز ۹۸ اجرا می‌شود. برای استفاده از این برنامه تحت ویندوزهای NT به بعد، باید از روشهای خاص دسترسی به پورتها استفاده کنید.

شبه‌کد برنامه قفل سخت‌افزاری به صورت زیر است:

```
Wait to receive 55h;// Start of communication
Delay (300);       // Wait untill the first data is ready

unsigned char sum = 0;

Repeat for 5 times{
      Receive new data from Parallel Port;
      Add new data to sum;

      Delay (200);
}

Send sum to parallel port;
```

پرسش) سیستم بالا به کمک روش کنترل جریان داده تأخیر طراحی شده است. برنامه‌ها را طوری اصلاح کنید که از روش دست‌دهی برای کنترل جریان داده استفاده کنند.

استفاده از پورت موازی به شیوه وقفه

بیت شماره ۴ ثبات کنترلی Interrupt Control نام دارد. اگر این بیت را در برنامه کامپیوتری خود «یک» کنید، وقفه پورت موازی فعال می‌شود. در این حالت، هنگامی که دستگاه خارجی یک لبه بالارونده روی پین ACK پورت موازی ایجاد کند، وقفه‌ای رخ می‌دهد، اجرای برنامه فعلی قطع و زیربرنامه ISR مربوط به آن وقفه اجرا می‌شود و سپس اجرای برنامه قبلی دنبال می‌شود.

```
#include    <STDIO.H>
#include    <STDLIB.H>
#include    <DOS.H>

void SendData(int PortID, unsigned char Data){
     outportb(PortID, Data);
}

void Change_Bit(int  PortID, int BitNumber, int value){
     unsigned char In, temp, Out;
     In = inportb(PortID);

     temp = 1;
     temp = temp << BitNumber;

     if   (value == 1)
          Out = In | temp;

     else  // value = 0
          Out = In & (~temp);

     outportb(PortID, Out);
}

void Check_Lock(int PortID){
     SendData(PortID, 0x55); // Start of Data Transfer
     delay(200);

     unsigned char Data[5], total = 0, sum;

     // Generate random numbers
     randomize();
     for (int i = 0; i < 5; i++){
          Data[i] = random(10);
          total = total + Data[i];
     }

     // Send random numbers to parallel port
     for (i = 0; i < 5; i++){
          SendData(PortID, Data[i]);
          delay(200);
     }
     Change_Bit(PortID+2, 5, 1); // Parallel Port as input
     delay(1000); // wait to receive the result from port
     sum = inport(PortID);
     Change_Bit(PortID+2, 5, 0); // Parallel Port as output

     if (total != sum){
          printf("Lock is not installed in parallel port");
          exit(0);
     }
     else
          return;
}

void   main(){

     Check_Lock(0x0378);

     // Continue to program
     …
}
```

این روش برای جلوگیری از هدر رفتن وقت پردازنده برای کنترل رخ دادن یک اتفاق بسیار مفید است. به بیان دیگر، وقتی پردازنده باید در صورت روی دادن یک اتفاق خاص به اجرای زیربرنامه‌ای بپردازد، نیازی نیست مرتباً روی دادن آن اتفاق را کنترل کند. در روش وقفه به هنگام رخ دادن آن اتفاق، به پردازنده اطلاع داده می‌شود و پردازنده عملیات خود را رها کرده و زیربرنامه مورد نظر را اجرا می‌کند.

در پورت موازی معمولاً از IRQ5 یا IRQ7 استفاده می‌شود (هرچند احتمال استفاده از شماره‌های دیگر نیز وجود دارد). برای استفاده از پورت موازی به شیوه وقفه، منبع وقفه‌دهنده (که در اینجا یک دستگاه خارجی است) باید سیگنالی در حالت خروجی داشته باشد که در هنگام نیاز به جلب توجه پردازنده (مثلاً وقتی ارتفاع ماده مذاب در یک دیگ از حد مجاز زیاد می‌شود و باید به سرعت برنامه‌ای اجرا شود که ارتفاع را کم کند)، یک لبه بالارونده روی آن ایجاد کند. با اتصال این سیگنال به پایه ACK پورت موازی، در صورت وقوع لبه بالارونده وقفه (شماره ۵ یا ۷ یا شماره‌های دیگر) در کامپیوتر رخ داده و زیربرنامه ISR مربوط به آن وقفه اجرا می‌شود. تنها کار باقی مانده این است که زیربرنامه ISR فوق را به صورت مطلوبتان تغییر دهید.

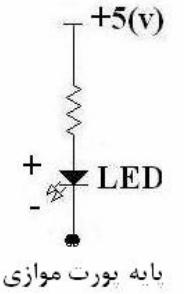
تغییر ISR یک وقفه در کامپیوتر چندان پیچیده نیست و می‌توان آن را به کمک برنامه‌های اسمبلی یا سطح بالا (مانند C) انجام داد. در کتابهای زبان اسمبلی در بخش برنامه‌های مقیم در حافظه[1] می‌توانید شیوه این کار را فرا بگیرید. در منابع انتهای این فصل نیز برنامه‌های نمونه آورده شده است.

نکاتی در مورد استفاده از پورت موازی برای کنترل دستگاهها

- برد مطمئن ارسال داده‌ها به کمک روش موازی حدود ۳ متر است. در فواصل بیش از ۳ متر سیمهای کابل انتقال موازی روی یکدیگر ایجاد نویز کرده و باعث اختلال در ارسال صحیح داده‌ها می‌شوند. برای به کارگیری ارتباط موازی در فواصل زیاد باید اطلاعات در فواصل ۳ متری بافر شده و مجدداً ارسال شوند.
- برای کنترل جریان داده‌ها، استفاده از روش دست‌دهی راهکار حرفه‌ای‌تری نسبت به روش تأخیر است؛ اما این روش دو سیم به تعداد سیمهای زیاد ارتباط موازی اضافه کرده و هزینه ارتباط را بالاتر می‌برد.
- مقاومت خروجی پایه‌های پورت موازی برابر ۴۷۰ اهم و حداکثر جریان هر پایه ۱۰/۶ میلی‌آمپر است که در طراحی بخش واسط بین دستگاه و پورت موازی باید مورد توجه

[1] ***Terminate & Stay Resident - TSR***

قرار گیرد. اگر بیش از مقدار مجاز از یکی از پایه‌های پورت موازی جریان کشیده شود، احتمال آسیب رسیدن به پورت زیاد است. حتی توصیه می‌شود برای اتصال LED به پایه‌های پورت موازی، LED را به صورت مقابل در گرایش معکوس ببندید تا جریان زیادی از پورت نکشد. بنابراین هرگز از پایه‌های پورت موازی به صورت مستقیم برای روشن کردن عناصری مانند رله، SCR، Triac، Tristor، سوییچهای الکترونیکی[1] و ... استفاده نکنید و حتماً واسطهایی مانند بافر، ترانزیستور، اوپتوکوپلر[2] و ... را به کار ببرید. این کار به ویژه در مورد مدارات قدرت که با جریانهای زیاد سروکار دارند برای پیشگیری از سوختن بورد اصلی کامپیوتر الزامی است.



- در برنامه‌هایی که دیدیم، از سیگنالهای وضعیت و کنترل پورت موازی برای عملیات دست‌دهی استفاده شد. طبیعی است که می‌توانید از سیگنالهای وضعیت برای دریافت و از سیگنالهای کنترلی برای ارسال داده‌ها نیز استفاده کنید.
- همانطور که قبلاً گفته شد، چون پورت موازی از ابتدا برای اتصال چاپگر به کامپیوتر استاندارد شده است، سیگنالهای این پورت نیز با نامهایی که در ارتباط کامپیوتر و چاپگر به کار می‌روند نامگذاری شده‌اند. اما در طراحی دستگاههای خارجی که باید به پورت موازی کامپیوتر متصل شوند، نیازی نیست از این سیگنالها با همان اسامی و معانی استفاده کنید. مثلاً سیگنال PE که در رابطه کامپیوتر با چاپگر برای اعلان خطای اتمام کاغذ توسط چاپگر به کار می‌رود، می‌تواند در رابطه کامپیوتر با دستگاه خارجی دیگر برای اعلام وضعیت یا خطایی مربوط به همان رابطه مورد استفاده قرار گیرد.

استانداردهای پورت موازی

تاکنون استانداردهای مختلفی از پورت موازی معرفی شده است که در زیر به معرفی مختصر آنها می‌پردازیم:

- **استاندارد SPP[3] (ATI-Type یا ISA-Compatible)**: در سال ۱۹۸۱ و در اولین کامپیوترهای IBM به عنوان رابط چاپگر استانداردی به نام چاپگر سنترونیکس[4] یا

[1] این سوییچها به عنوان جایگزینی برای رله‌های مکانیکی به کار می‌روند که از جمله آنها می‌توان به تراشه 4066B که با تکنولوژی CMOS ساخته شده یا CD74HC22106 اشاره کرد.

[2] Optocoupler قطعه‌ای است که ورودی و خروجی آن به کمک نور با یکدیگر در ارتباط هستند و از دید الکتریکی کاملاً مجزا می‌باشند. از جمله آنها می‌توان به CNY17 اشاره کرد.

[3] ***Standard Parallel Port***

[4] ***Centronics***

EPSON FX-100 معرفی شد. مدارات داخلی پورت استاندارد SPP فقط برای خروج داده طراحی شده و استفاده از آن به عنوان ورودی باعث آسیب رساندن به آن می‌شود.

- **استاندارد PS/2:** در سال ۱۹۸۷ معرفی شد و برخلاف پورت استاندارد SPP به عنوان ورودی نیز قابل استفاده است. این کار با «یک» کردن بیت Direction ثبات کنترلی پورت موازی انجام می‌شود.

در دو استاندارد ذکر شده، هر عملیات ورودی/خروجی در ۴ سیکل ساعت گذرگاه ISA انجام می‌شود.

- **استاندارد EPP**[1]: این استاندارد در ۱۹۹۶ تحت عنوان IEEE-1284 ارایه شد و توسط شرکتهای Intel و Xircom و Zenith توسعه یافت. در این استاندارد، عملیات ورودی/خروجی در یک سیکل ساعت گذرگاه ISA انجام می‌شود که باعث سرعت بالاتر در تبادل داده (از 500 KBps تا 2 MBps) و نیز سرعت سوییچ بیشتر می‌شود. یکی از دلایل این سرعت بالا، تولید سیگنالهای کنترلی دست‌دهی به کمک مدارات سخت‌افزاری در پورت LPT است. در استانداردهای قبلی تولید این سیگنالها از وظایف نرم‌افزار کامپیوتر بود. در این استاندارد، تعداد ثباتهای پورت موازی به ۸ عدد افزایش یافته است.
- **استاندارد ECP**[2]: این استاندارد تمام قابلیت‌های استاندارد EPP علاوه بر قابلیت‌های DMA و فشرده‌سازی تا حد $\frac{1}{64}$ را داراست و برای تبادل بلوکهای بزرگ داده کارایی خوبی دارد.

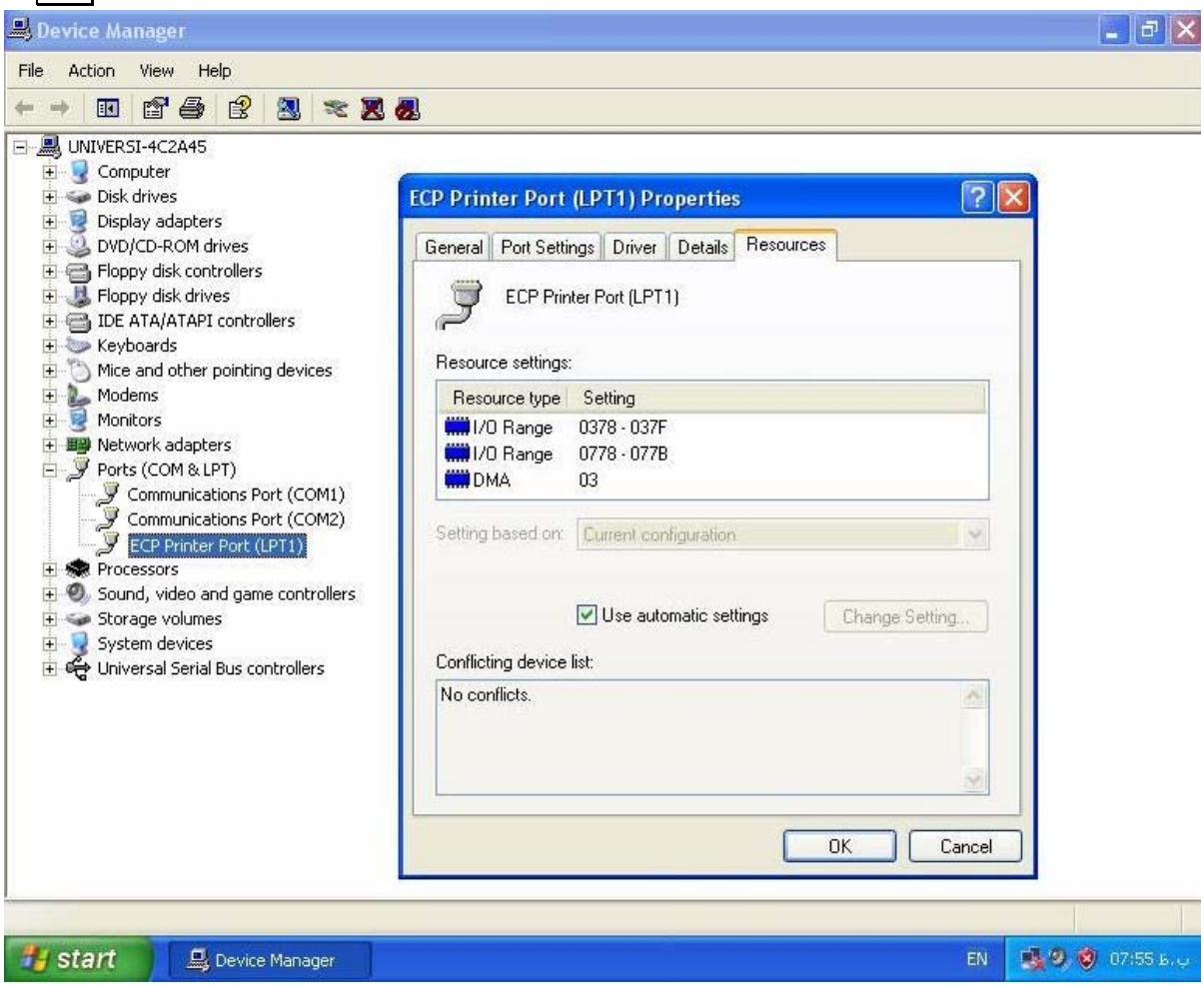
امروزه اکثر کامپیوترها از پورتهای موازی دارای استاندارد ECP استفاده می‌کنند. در ویندوز می‌توانید استاندارد پورت موازی کامپیوتر خود و نیز آدرس‌های تخصیص داده شده به ثباتهای ویژه پورت را ببینید. برای این کار:

- از Control Panel گزینه System را انتخاب کنید.
- به سربرگ Hardware بروید و گزینه Device Manager را انتخاب کنید.

حال در قسمت Ports می‌توانید مشخصات پورتهای کامپیوتر خود را ببینید. شکل زیر، گزینه Ports مربوط به یک کامپیوتر نوعی را نشان می‌دهد.

---

[1] ***Enhanced Parallel Port***
[2] ***Extended Compatibilities Port***



همانطور که در شکل بالا می‌بینید، پورت موازی این کامپیوتر از استاندارد ECP پیروی می‌کند. با کلیک راست روی نام پورت موازی و انتخاب گزینه Properties، کادر سمت راست ظاهر می‌شود که ویژگیهای پورت موازی را نشان می‌دهد. با انتخاب سربرگ Resources، منابع اختصاص داده شده به پورت موازی را می‌بینید (شکل بالا). به این پورت بازه آدرسهای ورودی/خروجی 0378 h – 037F h (مجموعاً ۸ ثبات) و نیز کانال شماره ۳ DMA اختصاص داده شده است.

توجه کنید آنچه در این فصل برای ارتباط با پورت موازی گفته شد، ساده‌ترین استاندارد این ارتباط را نشان می‌دهد که سازگار با تمام کامپیوترهاست. ارتباط با پورتهایی با استاندارد EPP و ECP به کمک ثباتهای اضافی و ویژگیهای پیشرفته‌تر این پورتها، شامل عملیاتی نسبتاً متفاوت است که طی آن عملکرد پینهای پورت موازی نیز تغییر می‌یابند. ارتباط با استاندارد ECP از استاندارد EPP پیچیده‌تر است؛ چرا که در آن لازم است وسیله‌ای که به پورت موازی متصل می‌شود نیز برنامه‌ریزی مفصل و هوشمندی داشته باشد. در حالی که در استاندارد EPP کنترل ارتباط کاملاً در دست کامپیوتر است. برای طولانی نشدن بحث، این شیوه ارتباطی را در اینجا بررسی نمی‌کنیم. اگر به مطالب

بیشتری در این رابطه علاقه دارید، می‌توانید جزییات بیشتر را در منابع معرفی شده در انتهای فصل ببینید.

پروژه‌های کاربردی با پورت موازی کامپیوتر

۱) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید یک رقص نور روی هشت عدد LED که به پورت موازی متصل است ایجاد کند. برای این کار LEDها باید هر یک ثانیه یک بار به صورت یکی‌درمیان روشن شوند.

۲) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که محتویات ۸ عدد کلید دو حالته که به پورت موازی کامپیوتر متصل است را بخواند و روی نمایشگر کامپیوتر نشان دهد.

۳) تعداد LEDهای سیستم ۱ را با استفاده از سیگنالهای کنترلی به عنوان سیگنالهای داده خروجی، به ۱۲ عدد LED افزایش و اصلاحات لازم را در سخت‌افزار و نرم‌افزار انجام دهید.

۴) تعداد کلیدهای سیستم ۲ را با استفاده از سیگنالهای وضعیت به عنوان سیگنالهای داده ورودی، به ۱۳ عدد کلید افزایش و اصلاحات لازم را در سخت‌افزار و نرم‌افزار انجام دهید.

۵) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که محتویات هشت کلید دو حالته را از طریق پورت موازی بخواند، در برنامه کامپیوتری آن را منفی کند و حاصل را روی هشت LED متصل به پورت موازی نمایش دهد.

راهنمایی: چون پورت موازی تنها هشت پایه داده دارد که هم برای خواندن محتویات کلیدها و هم برای نمایش نتیجه روی LEDها به کار می‌رود، باید از تکنیکهایی که در فصل اول برای استفاده مشترک از گذرگاه داده آموختید استفاده کنید؛ یعنی تراشه نگهدار (Latch) را برای اتصال خروجی و تراشه بافر سه‌حالته را برای اتصال ورودی به کار ببرید. از سیگنالهای کنترلی برای فعال کردن این تراشه‌ها در زمان مناسب استفاده کنید.

۶) LCD یک نمایشگر ساده است که در دستگاههای بسیاری برای نمایش اطلاعات کاراکتری و رقمی مورد استفاده قرار می‌گیرد. نمونه‌ای از LCD را در شکل مقابل می‌بینید.

برای ارسال اطلاعات متوالی به LCD، باید از روشهای کنترل جریان داده بهره ببرید که در LCD هر دو روش قابل استفاده است. با مطالعه ساختار LCD، سیستمی شامل نرم‌افزار و سخت‌افزار طراحی کنید که با اجرای برنامه‌ای روی کامپیوتر، نامتان روی LCD متصل به پورت موازی نمایش داده شود. این سیستم را در عمل می‌توانید به راحتی بسازید.

۷) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که یک LED را با فرکانس ۱ هرتز روشن و خاموش کند.

۸) برنامه قبلی را به گونه‌ای تغییر دهید که فرکانس مورد نظر کاربر را در ابتدای برنامه از وی سؤال کند. بازه این فرکانس باید از ۱ هرتز تا ۱۰ کیلوهرتز متغیر باشد. توجه کنید در فرکانسهای بالاتر از ۵۰ هرتز، چشم متوجه خاموش و روشن شدن LED نمی‌شود و همیشه آن را روشن می‌بیند. برای دیدن این فرکانسهای بالا می‌توانید از اسیلوسکوپ استفاده کنید.

۹) سیستم ۷ را به گونه‌ای تغییر دهید که روی هشت پایه داده پورت موازی، هشت موج مربعی با فرکانسهای مختلف تولید کند. برای این کار باید از کوچکترین دوره تناوب به عنوان پایه دوره‌های تناوب دیگر استفاده کنید.

۱۰) سیستمی طراحی کنید که به کمک مبدل دیجیتال به آنالوگ (DAC) یک موج سینوسی با فرکانس دلخواه را روی یکی از پایه‌های داده پورت موازی ایجاد کند.

۱۱) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که هر یک ثانیه یک بار دمای اتاق را از یک سنسور دما خوانده و به کمک مبدل آنالوگ به دیجیتال (ADC) مقدار آن را به یک عدد دیجیتال تبدیل کند و روی نمایشگر کامپیوتر نشان دهد.

۱۲) سیستمی طراحی کنید که بدون نیاز به کامپیوتر، هر یک ثانیه یک بار دمای اتاق را به کمک چاپگر چاپ کند.

۱۳) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که به عنوان یک کارت اسیلوسکوپ عمل کند. سخت‌افزار این سیستم باید یک سیگنال آنالوگ را به کمک ADC به عدد دیجیتال تبدیل کرده و به پورت موازی کامپیوتر ارسال کند. برنامه کامپیوتری نیز باید منحنی تغییرات آن سیگنال را در نمایشگر کامپیوتر نشان دهد.

۱۴) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که فرکانس یک موج مربعی که به کمک تراشه ۵۵۵ تولید و به پورت موازی کامپیوتر وارد می‌شود را روی نمایشگر کامپیوتر نشان دهد.

۱۵) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که جهت و سرعت چرخش یک موتور پله‌ای را به کمک برنامه کامپیوتری کنترل کند. با این سیستم می‌توانید روباتی بسازید که به کمک پورت موازی کامپیوتر کنترل شود.

۱۶) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که اعداد صفر تا ۹ را با فواصل زمانی یک ثانیه روی یک نمایشگر هفت‌قسمتی[1] نمایش دهد. از تراشه مبدل کد BCD به کد نمایشگر هفت قسمتی استفاده نکنید. برنامه خود را طوری بنویسید که نوع نمایشگر هفت‌قسمتی (آند مشترک یا کاتد مشترک) را در ابتدای برنامه از کاربر سؤال کند.

۱۷) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که اعداد 0.00 تا 9.99 را با فواصل زمانی ۵۰۰ میلی‌ثانیه روی سه عدد نمایشگر هفت‌قسمتی نمایش دهد. چون پورت موازی تنها دارای ۸ بیت داده است که باید به هر سه نمایشگر متصل شود، در هر لحظه باید یکی از آنها انتخاب شود. برای انتخاب نمایشگر مورد نظر از سیگنالهای کنترلی پورت موازی استفاده کنید.

۱۸) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که یک ولتاژ آنالوگ بین صفر تا ۵ ولت که به کمک یک پتانسیومتر تولید می‌شود را از طریق پورت موازی خوانده و مقدار این ولتاژ را با دقت دو رقم اعشار روی سه عدد نمایشگر هفت‌قسمتی که به پورت موازی متصلند، نمایش دهد.

۱۹) سیستمی شامل سخت‌افزار و نرم‌افزار طراحی کنید که مقدار یک ولتاژ آنالوگ بین صفر تا ۵ ولت که به کمک یک پتانسیومتر تولید می‌شود را از طریق پورت موازی کنترل کند. چنانچه این ولتاژ از ۳ ولت بیشتر شد (به کمک تراشه‌های مقایسه‌کننده مانند LM311 می‌توانید این موضوع را نشان دهید)، باید از طریق وقفه به کامپیوتر اطلاع داده شده و پیغام خاصی روی نمایشگر کامپیوتر نشان داده شود.

۲۰) فرض کنید می‌خواهید یک فایل متنی را از کامپیوتر ۱ به کامپیوتر ۲ ارسال کنید. روشهایی مانند استفاده از شبکه‌های کامپیوتری، اینترنت و یا حافظه‌های قابل حمل (فلاپی، CD، فلاش و ...) اولین راههایی است که به ذهن می‌رسد.
یک راه ساده و کم‌خرج، اتصال مستقیم پورتهای دو کامپیوتر به یکدیگر و انتقال اطلاعات از این راه است. این شیوه به اصطلاح مودم پوچ[2] نامیده می‌شود؛ چون کامپیوترها تصور می‌کنند از طریق مودم به یکدیگر متصل هستند، در حالی که عملاً مودمی وجود ندارد و کامپیوترها فقط از طریق یک کابل ساده به یکدیگر متصل هستند. یکی از راههای ایجاد مودم پوچ، اتصال پورتهای موازی دو کامپیوتر به یکدیگر است.

---
[1] ***Seven Segment***
[2] ***Null Modem***

سیستمی شامل نرم‌افزار کامپیوتر فرستنده و کامپیوتر گیرنده و سخت‌افزار طراحی کنید که با اتصال پورتهای موازی دو کامپیوتر و اجرای برنامه‌ها روی دو کامپیوتر، ارسال فایل از فرستنده به گیرنده میسر شود.

منابع

[۱] "اصول کامل راه‌اندازی و کنترل دستگاههای جانبی توسط کامپیوتر"، محسن شکیبافر، انتشارات نص، ۱۳۸۴.

[۲] "گذرگاهها و درگاههای کامپیوترهای شخصی"، عبدالمجید منصوریان‌فر و اصغر کریمی، انتشارات دانش‌پژوهان برین - ارکان، ۱۳۸۳

[۳] "دیجیتال پایه و طراحی مدارهای واسط"، شیرزاد شهریاری، انتشارات پرتونگار، ۱۳۸۵.