به نام خدا

تمرین سری دوم میکرو

سید محمد مظفری 9423106

LCD دارای 16 پایه است که کاربرد هر کدام از آن ها به قرار زیر می باشد:

* VCC که ولتاژ +5 را برامان تامین خواهد کرد.
* VSS که زمین LCD را تعیین میکند.
* VEE که میزان contrast صفحه ی LCD را تعیین میکند.
* RS انتخاب کننده­ی رجیستر است. (Register select) دو رجیستر مهم در LCD وجود دارد که پین RS این دو رجیستر را انتخاب خواهد کرد به این صورت که اگر RS = 0 آنگاه رجیستر instruction command code انتخاب خواهد شد که به کاربر اجازه می­دهد دستورهایی از قبیل clear display, cursor at home و غیره را به LCD بدهد. اگر RS = 1 باشد آنگاه data register انتخاب می­شود که به کاربر اجازه می­دهد data را برای نمایش بروی LCD به آن ارسال کند.
* R/W(read/write) به کاربر اجازه­ی خوادن اطلاعات از LCD و نوشتن اطلاعات بر روی LCD را می­دهد به این صورت که R/W = 1 برای خواندن و R/W = 0 برای نوشتن.
* E(enable) پین enable بدین منظور به­کار می­رود که data های قرار گرفته بر روی data pin ها را latch کند (یا به عبارتی دیگر نگه دارد). برای این کار هنگامی که اطلاعاتی بر روی پین های ورودی اطلاعات LCD قرار گرفت، یک پالس پایین رونده (از سطح بالا به پایین) باید به پین enable اعمال کنیم تا LCD اطلاعات را Latch کند. طول این پالس باید حداقل 450nS باشد.
* D0-D7 پین­های data برای LCD هستند که برای فرستادن اطلاعات به LCD و خواندن محتوای رجیسترهای داخلی آن به کار می­روند. برای نمایش دادن کلمات و اعداد بر روی LCD کدهای ASCII مربوط به حروف A-Z و a-z و اعداد 0-9 را در حالتی که پین RS = 1 است (یعنی data register انتخاب شده است) به این پین­ها ارسال می­کنیم.

می­توان در خروجی میکرو از ساختار یک ترانزیستور که ورودی LCD از درین آن گرفته می­شود استفاده کرد در این صورت اگر فرض کنیم خروجی میکرو در حدود 2 ولت در سطح بالا و نزدیک به 0 در سطح پایین باشد و همچنین اگر درین ترانزیستور قرار گرفته در خروجی میکرو را به 5 ولت وصل کنیم، آنگاه هنگامی که خروجی میکرو high است، ترانزیستور به اشباع رفته و ورودی LCD صفر خواهد شد و نیز هنگامی که خروجی میکرو low است ورودی LCD به 5 ولت رسیده و high خواهد شد. در واقع با این روش ولتاژ کم خروجی میکرو را به یک ولتاژ با سطح بالاتر تبدیل کرده­ایم.



در مورد ورودی توابع که به صورت پویینتر داده شده بود، علت آن است که هنگامی که یک struct تعریف می­کنیم و می­خواهیم آن را به عنوان ورودی به توابع بدهیم، اگر به صورت void این ورودی­ها را داده و از متغیرهایی که به صورت struct تعریف کرده استفاده کنیم، آنگاه تمام این متغیرها یکبار کپی شده و سپس مورد استفاده قرار می­گیرد در صورتی که این کار اتلاف حافظه و source است. ولی اگر ورودی توابع را به صورت pointer بدهیم آنگاه خود متغیرها به توابع رفته و مورد استفاده قرار می­گیرند.