【 기술 노트 11 】

8비트 마이크로프로세서에서 16비트 데이터의 입출력

8비트 마이크로프로세서에 16비트의 병렬 입출력장치를 접속하기 위한 16비트 병렬 입출력 인터페이스 회로를 설계할 경우에는 매우 세심한 주의가 필요하다. 왜냐하면, CPU는 한 번에 8비트씩의 데이터를 입출력하므로 16비트 입출력장치와 데이터 전송에서는 반드시 2차례의 입출력 동작을 수행해야 하며, 이들 입출력 동작 사이에 시차(time gap)가 발생하기 때문이다.

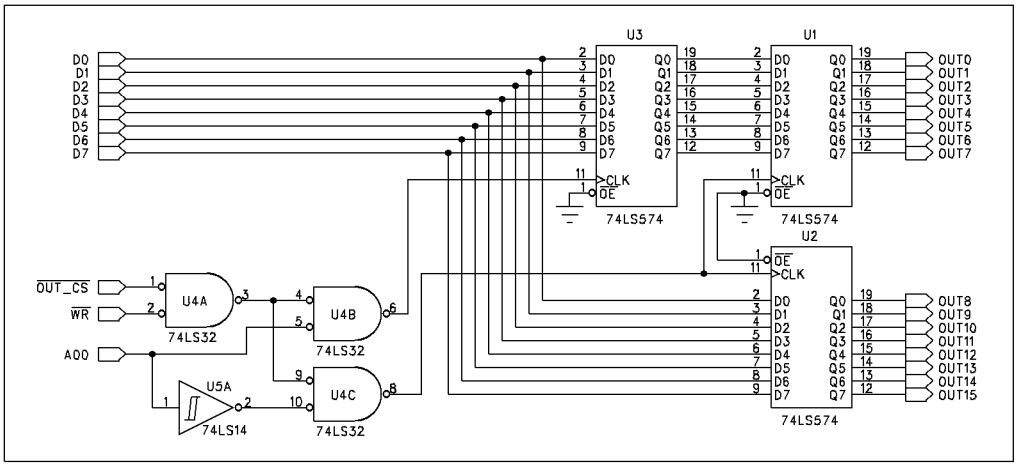
이러한 것은 10비트, 12비트, 16비트의 A/D 또는 D/A 컨버터나 카운터에서 모두 마찬가지이다. D/A 컨버터와 같은 병렬 출력장치에서는 이와 같은 2번의 8비트 데이터 출력 사이의 시차 때문에 하위 바이트는 새로 갱신된 데이터가 적용되고 상위 바이트는 갱신되기 전의 데이터가 적용됨으로써 아날로그로 변환된 출력신호에 글리치(glitch)가 발생하여 심각한 문제를 야기할 수도 있다. 또한, 입력장치의 경우에는 2회의 8비트 데이터 입력 사이의 시차 때문에 하위 바이트는 갱신되기 전의 데이터가 적용되고 상위 바이트는 새로 갱신된 데이터가 적용됨으로써 입력된 디지털 데이터가 실제와는 큰 오차를 가질 수 있다.

1. 16비트 병렬 출력 인터페이스

8비트 CPU에서 16비트의 병렬 출력장치를 접속하기 위한 16비트 병렬 출력 인터페이스 회로에서는 2회의 8비트 출력 동작 사이의 시차 때문에 글리치가 발생할 수 있으므로, 실제로 출력장치로 출력되는 데이터가 동시에 16비트로 출력되도록 하는 것이 중요하다.

예를 들어 지금 16비트의 D/A 컨버터를 위하여 하위 바이트에 래치1을 접속하고 상위 바이트에 래치2를 접속하였으며, 8비트 마이크로프로세서는 하위 8비트 데이터를 먼저 출력하고 상위 8비트 데이터를 나중에 출력한다고 가정하자. 지금 만약 D/A 컨버터로 삼각파를 발생시키기 위하여 현재 00FFH가 출력되어 있는 상태라면 래치1에는 FFH, 래치2에는 00H가 저장되어 있다. 이때 이보다 1증가된 0100H를 출력하기 위하여 먼저 하위 바이트인 00H를 래치1에 출력하게 되면 이것이 래치2에 이미 저장되어 있던 00H와 함께 D/A 컨버터에 일시적으로 0000H를 공급하게 된다. 이것은 00FFH나 0100H와는 매우 다른 출력값이다. 이러한 바람직하지 않은 현상은 다음 명령으로 상위 바이트인 01H를 래치2에 출력하여 전체적으로 0100H를 공급할 때까지 지속된다.

이렇게 바람직하지 않은 과도적 현상은 D/A 컨버터에 글리치라고 불리는 펄스 출력으로 나타난다. 마이크로프로세서의 처리속도가 매우 빠르다면 이러한 과도 시간도 짧아지므로 D/A 컨버터의 다음단에 LFP(Low-Pass Filter) 정도를 달아서 해결할 수 있지만, 마이크로프로세서의 속도가 느려지면 글리치 펄스의 폭도 그만큼 길어지므로 이를 근본적으로 없애는 회로를 사용해야만 한다.



<그림 1> 16비트 병렬출력 회로

이를 위해서는 <그림 1>의 회로에서 보인 것처럼 하위 바이트의 데이터는 첫번째 단계에서 먼저 래치에 출력하여 저장해 두고, 두번째 단계에서 상위 바이트의 데이터가 출력될 때 하위 바이트의 데이터와 상위 바이트의 데이터가 동시에 출력장치로 전송되도록 하면 된다. 즉, 제1단계에서는 하위의 짝수번지(A00 = 0)에 대한 8비트 출력명령으로 CPU의 데이터 버스 D0~D7을 통하여 하위 바이트를 출력한다. 그러면 이 데이터는 8비트 플립플롭인 U3에 저장되어 있다가 나중에 U1의 입력이 된다. 제2단계에서는 상위의 홀수번지(A00 = 1)에 대한 8비트 출력명령으로 역시 데이터 버스 D0~D7을 통하여 상위 바이트를 출력한다. 그러면 이 상위 바이트의 데이터가 U2에 저장되어 출력장치로 전송됨과 동시에 이미 U3에 저장되어 있던 하위 바이트의 데이터가 U1에 저장됨으로써 최종 출력단에는 동시에 상위 및 하위 바이트의 16비트 데이터 XD0~XD15가 출력된다.

16비트뿐만 아니라 10비트, 12비트 등의 D/A 컨버터를 접속할 경우에도 이와 마찬가지로 사용한다.

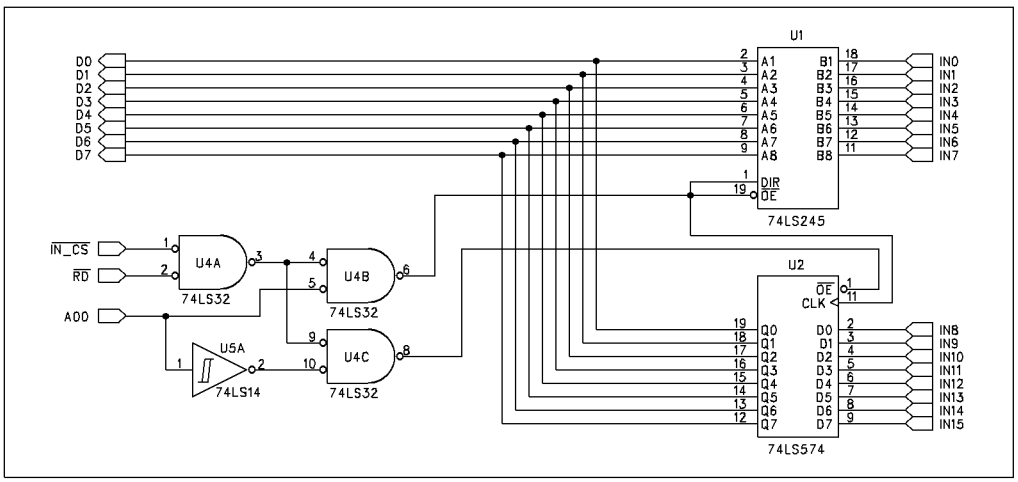
2. 16비트 병렬 입력 인터페이스

8비트 CPU에서 16비트의 병렬 입력장치를 접속하기 위한 16비트 병렬 입력 인터페이스 회로에서는 2회의 8비트 입력 동작 사이의 시차 때문에 입력 데이터가 변할 수 있으므로, 실제로 입력장치에서 입력하는 데이터를 동시에 순간적으로 16비트로 저장시켜 놓고 입력하는 것이 중요하다.

예를 들어 지금 16비트의 카운터를 읽어들이기 위하여 하위 바이트에 트리 스테이트 버퍼1을 접속하고 상위 바이트에 버퍼2를 접속하였으며, 8비트 마이크로프로세서는 하위 8비트 데이터를 먼저 입력하고 상위 8비트 데이터를 나중에 입력한다고 가정하자. 지금 만약 카운터의 값이 00FFH인 상태라면 버퍼1을 통하여 FFH, 버퍼2를 통하여 00H가 읽혀져야

한다. 그러나, 여기서 마이크로프로세서가 하위 바이트인 FFH를 읽어들이는 동안 카운터에 새로운 펄스가 입력되어 카운터값이 1증가한다면 2차로 버퍼2에서 상위 바이트가 01H로 읽혀지므로 마이크로프로세서는 01FFH를 읽어들인 것으로 처리된다. 이것은 카운터가 변하기 전의 00FFH나 변한 후의 0100H와는 매우 다른 입력값이다.

이러한 바람직하지 않은 현상은 입력신호가 변할 수 있는 값이라면 아무리 빠른 마이크로프로세서를 사용하더라도 피할 수가 없으므로 이를 근본적으로 없애는 회로를 사용해야만 한다.



<그림 2> 16비트 병렬입력 회로

이를 위해서는 <그림 2>의 회로에서 보인 것처럼 하위 바이트의 데이터를 읽는 순간에 전체 16비트의 데이터를 저장시키면서 1단계로 하위 바이트의 데이터를 먼저 입력하고, 2단계에서는 이미 1단계에서 저장시켜 놓았던 상위 바이트의 데이터를 입력하면 된다. 즉, 제1단계에서는 하위의 짝수번지(A00 = 0)에 대한 8비트 입력명령으로 CPU의 데이터 버스 D0~D7을 통하여 하위 바이트를 입력한다. 그러면 이 하위 바이트의 데이터는 8비트 3상태 버퍼인 U1을 통하여 읽혀짐과 동시에 8비트 플립플롭인 U2에 상위 바이트가 저장된다. 제2단계에서는 상위의 홀수번지(A00 = 1)에 대한 8비트 입력명령으로 역시 데이터 버스 D0~D7을 통하여 상위 바이트를 입력한다. 이 상위 바이트의 데이터는 제1단계에서 하위 바이트를 읽어들이는 순간에 함께 저장되어 있던 것이므로 그 동안에 입력장치로부터 전송되는 데이터가 변화하였더라도 아무 영향이 없이 읽혀진다.

이러한 입력회로는 빠르게 변화하는 10비트, 12비트 또는 16비트의 카운터 등을 읽어들일 때 매우 유용하다.

<<추가>> 8비트 마이크로콘트롤러 AVR의 내부에서 사용하는 16비트 I/O 레지스터를 액

세스할 때는 이러한 글리치 문제를 근원적으로 해결하기 위하여 8비트의 임시 레지스터가 사용된다. 이 임시 레지스터는 출력 동작의 경우 <그림 1>의 U3과 같은 기능을 수행하고, 입력 동작의 경우 <그림 2>의 U2와 같은 기능을 수행한다. 따라서, AVR은 이런 문제를 해결하는 기능을 MCU가 기본적으로 가지고 있기 때문에 아무 문제없이 16비트 I/O 레지스터를 액세스할 수 있다.

AVR에서 이러한 기능을 수행하는 8비트 임시 레지스터는 상위 바이트 쪽에 붙어 있기 때문에 16비트 I/O 레지스터에 출력 동작을 수행할 때는 항상 먼저 상위 바이트를 출력하고 나중에 하위 바이트를 출력해야 하며, 16비트 I/O 레지스터에서 입력 동작을 수행할 때는 항상 먼저 하위 바이트를 입력하고 나중에 상위 바이트를 입력해야 한다.