

Chapter 123. 직렬 입력 병렬 출력 레지스터 - 74595

ATmega328의 경우 14개의 디지털 입출력 핀과 6개의 아날로그 입력 핀을 사용할 수 있으며 아날로그 입력 핀의 경우 디지털 입출력 핀으로도 사용할 수 있으므로 최대 20개의 디지털 입출력을 사용할 수 있다. 20개의 디지털 입출력 핀이 많은 것으로 보일 수도 있겠지만 1자리 7세그먼트 표시장치를 사용하기 위해서만 8개의 디지털 입출력 핀이 필요한 등 몇 개의 주변장치만을 연결하여도 20개의 디지털 입출력 핀은 곧 모자라게 되며, 특히 많은 개수의 LED를 사용하는 경우 디지털 입출력 핀의 부족은 흔한 현상이다. 더 많은 수의 디지털 입출력 핀이 필요하다면 다른 마이크로컨트롤러를 사용할 수도 있지만, 가격과 크기 등을 고려할 때 74595와 같은 디지털 출력 확장 칩을 사용하는 것을 고려해볼 수 있다. 74595 칩은 직렬 입력을 병렬 출력으로 변환해주는 레지스터로 3개의 디지털 입출력 핀을 통해 최소 8개의 입출력 핀을 사용할 수 있도록 해준다. 이 장에서는 74595 칩을 사용하여 적은 수의 디지털 입출력 핀으로 많은 개수의 LED를 제어하는 방법에 대해 알아본다.

1. 74595

74595는 16핀의 칩으로 하나의 데이터 입력 핀과 두 개의 제어 핀, 총 세 개의 디지털 출력 핀을 사용하여 8개의 디지털 출력을 사용할 수 있도록 해준다.

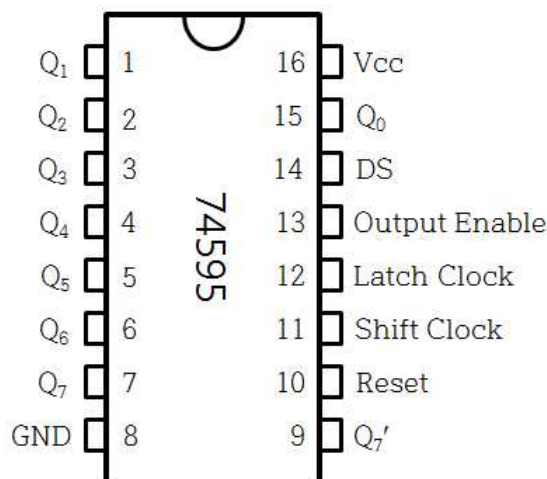


그림 1. 74595 핀 배치도

74595 칩은 직렬로 1비트씩 8번에 걸쳐 8비트의 데이터를 입력 받고, 이를 병렬로 출력하는 (Serial In Parallel Out) 역할을 한다. 그림 1에서 14번 핀 DS가 직렬 입력에 해당하며, Q_0 에서 Q_7 는 데이터를 병렬로 출력하는 핀이다. 74595는 11번과 12번 핀에 두 개의 CLOCK을 필요로 한다. 11번 핀의 Shift Clock은 송신단에서 전송하는 데이터를 수신하여 74595 내부에서 1비트씩 이동시키기 위해 사용된다. 반면 12번 Latch Clock은 74595 칩에서 병렬로 데이터가 출력되는 시점을 결정하기 위해 사용되는 클록이다. Shift Clock이 8번 발생하면 8비트의 데이터가 전송 완료되어 8비트 래치에 저장되고, 이 때 Latch Clock을 발생시키면 현재 래치에 저장된 데이터가 Q_0 에서 Q_7 핀을 통해 출력된다. Output Enable은 LOW 입력이 가해질 때 출력단의 래치(latch)를 활성화 시켜 출력이 가능하도록 하는 역할을 한다. 따라서 Output Enable을 GND(LOW 상태)에 연결하고 출력 시점은 Latch Clock으로 조절함으로써 병렬 데이터 출력이 가능하다. 물론 Latch Clock을 Shift Clock과 동일하게 설정하고, 8비트 데이터가 전송된 이후 Output Enable을 HIGH에서 LOW로 설정하여 병렬 데이터 출력이 가능하도록 할 수도 있다. 이 책에서는 첫 번째 방법을 사용한다. 그림 2는 74595의 동작을 설명한 것으로 데이터와 이동 클록을 8번 가한 후 마지막에 래치 클록을 가함으로써 출력이 발생하는 순서를 나타낸다. 단, Output Enable은 LOW에 연결된 것으로 가정한다.

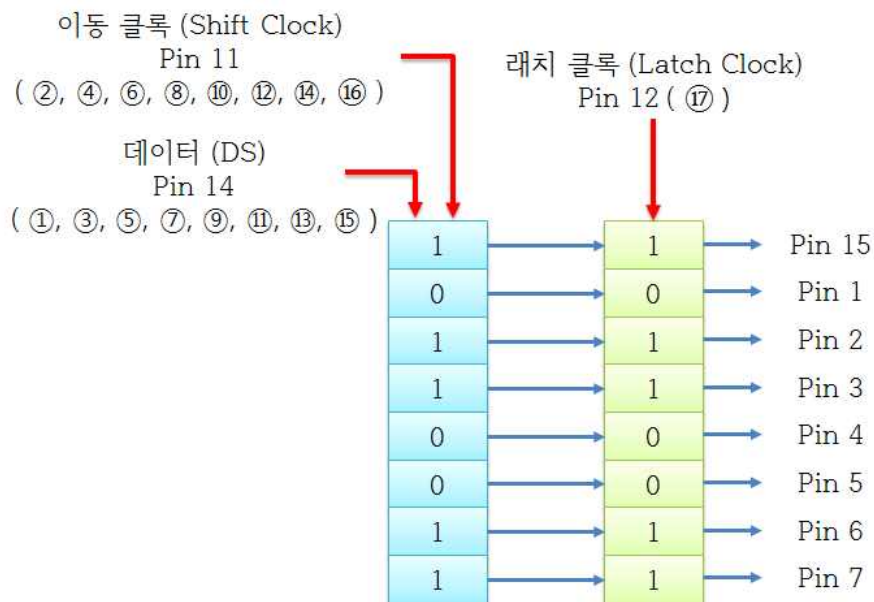


그림 2. 74595의 동작 순서

10번 Reset 핀은 비동기 초기화 핀으로 VCC에 연결시키면 아무런 동작도 수행하지 않는다. 9번 Q_7' 는 쉬프트 레지스터의 값을 직렬로 출력하는 핀으로 74595를 여러 개 연결시킬 수

있도록 해준다. 9번 Q_7' 핀을 다른 74595의 DS 핀으로 연결하면 3개의 디지털 입출력 핀만으로 16개 또는 그 이상의 데이터를 출력하도록 구성할 수 있다. 그림 3은 '0000 0000'으로 초기화된 쉬프트 레지스터에 8비트의 데이터(11001101, MSB 우선 전송)가 8번의 Shift Clock을 통해 이동하여 저장되는 과정을 보여준다. 이후 Latch Clock을 통해 8비트 데이터를 병렬로 출력할 수 있다.

클록	Q_0 LSB	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7 MSB
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0
4	0	0	1	1	0	0	0	0
5	1	0	0	1	1	0	0	0
6	1	1	0	0	1	1	0	0
7	0	1	1	0	0	1	1	0
8	1	0	1	1	0	0	1	1

그림 3. 74595 쉬프트 레지스터에 데이터가 저장되는 과정

74595를 ATmega328과 그림 4와 같이 연결해 보자. 데이터 입력을 위한 DS 핀은 17번 핀에 (PB3, 아두이노 디지털 11번 핀), 데이터 이동을 위한 Shift Clock 핀은 18번 핀에 (PB4, 아두이노 디지털 12번 핀), 래치 출력을 위한 Latch Clock 핀은 19번 핀에 (PB5, 아두이노 디지털 13번 핀) 연결한다. Output Enable은 GND에 연결하여 출력이 활성화되도록 설정한 후 Latch Clock 핀의 입력에 의해 출력이 조절되도록 한다.

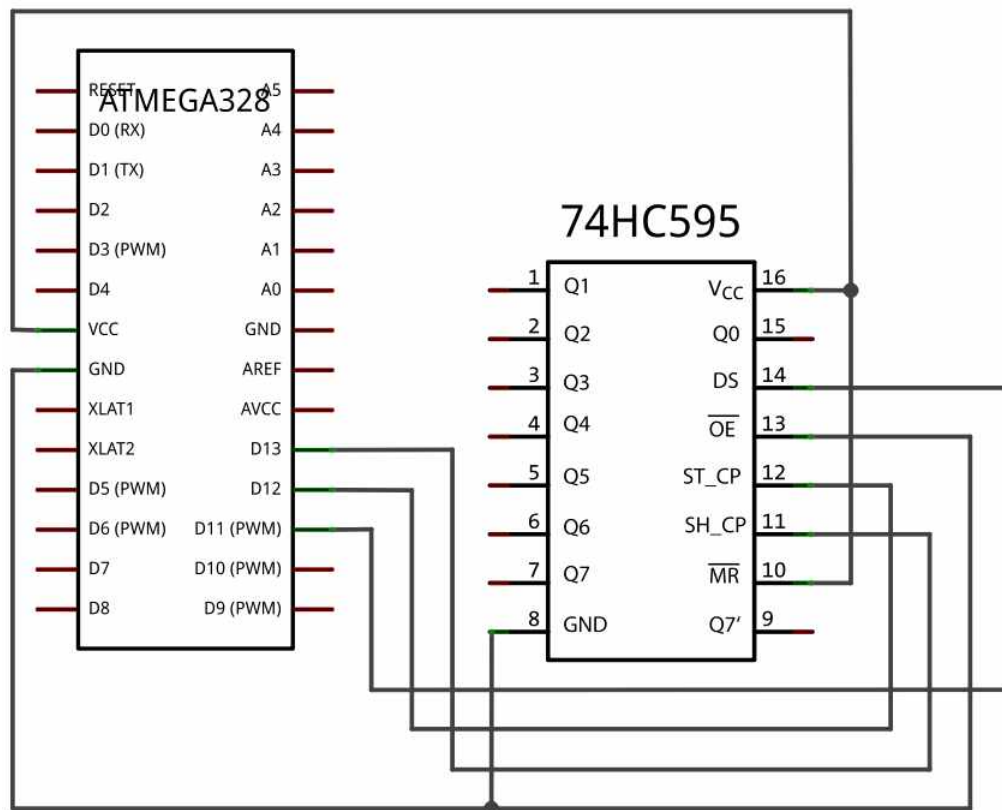


그림 4. ATmega328과 74595 연결 회로도

74595의 8비트 병렬 출력에는 8개의 LED를 연결한다.

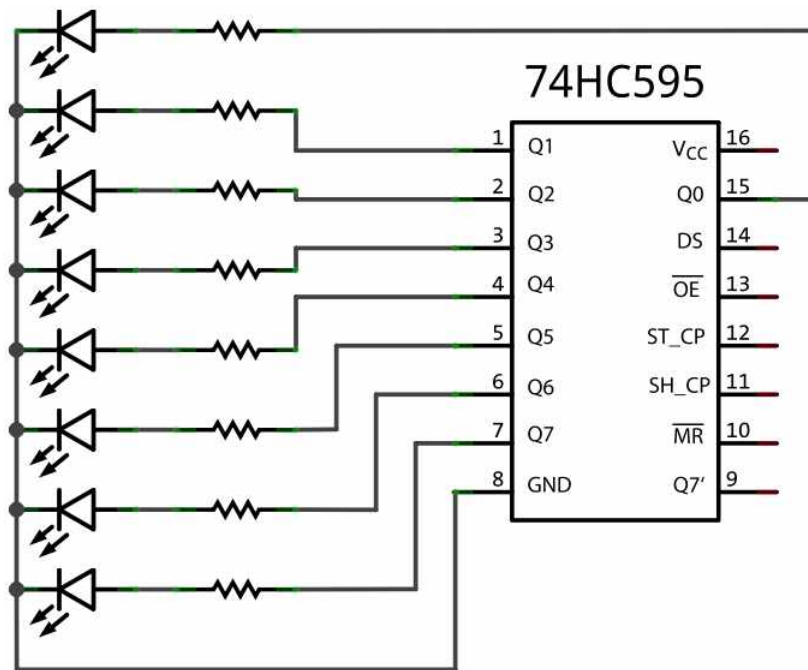


그림 5. 74595와 8개 LED 연결 회로도

코드 1은 74595에 연결된 8개 LED를 번갈아 켜는 코드의 예로 ShiftClock 함수와 LatchClock 함수는 해당 핀의 상태를 HIGH와 LOW로 전환함으로써 이동이 발생하도록 해주는 역할을 한다. 실제 데이터를 전달하는 함수는 ByteDataWrite 함수로 1 비트를 전송한 이후에는 ShiftClock 함수를 호출하여 74595 내 레지스터에서 이동이 발생하도록 하고, 8 비트를 전송한 이후에는 LatchClock 함수를 호출하여 데이터가 출력되도록 한다. ByteDataWrite 함수의 동작은 그림 2에 표시한 순서를 참고한다.

코드 1. 74595 칩으로 8개 LED 제어

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

void Init_74595(void)
{
    // data pin          : 아두이노 11, PB3
    // latch clock pin    : 아두이노 12, PB4
    // shift clock pin    : 아두이노 13, PB5
```

```
    DDRB |= 0b00111000;                // 제어핀 3개를 출력으로 설정
}

void ShiftClock(void)
{
    PORTB |= 0b00100000;                // 이동 클록을 HIGH로
    PORTB &= 0b11011111;                // 이동 클록을 LOW로
}

void LatchClock(void)
{
    PORTB |= 0b00010000;                // 래치 클록을 HIGH로
    PORTB &= 0b11101111;                // 래치 클록을 LOW로
}

void ByteDataWrite(uint8_t data)        // 1 바이트 데이터 출력
{
    for(uint8_t i = 0; i < 8; i++){
        if(data & 0b10000000)            // MSB부터 1비트 출력
            PORTB |= 0b00001000;
        else
            PORTB &= 0b11110111;

        ShiftClock();                    // 1비트 출력 후 비트 이동
        data = data << 1;                // 다음 출력할 비트를 MSB로 이동
    }

    LatchClock();                        // 1바이트 전달 후 실제 출력 발생
}

int main(void)
{
    uint8_t index = 0;

    Init_74595();                        // 74595 초기화
```

```

while(1)
{
    uint8_t pattern = 1 << index;          // 출력 패턴 결정
    index = (index + 1) % 8;                // 출력 패턴에서의 위치 결정

    ByteDataWrite(pattern);                // 바이트 데이터 출력

    _delay_ms(200);
}
}

```

2. 아두이노에서 74595 사용

아두이노에서 74595를 사용하는 방법 역시 다르지 않다. 다만 아두이노에서는 1 비트 데이터를 전송하고 이동 클록을 전송하는 과정을 위해 shiftOut 함수를 제공하고 있다. shiftOut 함수는 데이터 출력을 위한 핀과 데이터 이동을 위한 클록 출력 핀을 함께 지정하며, 바이트 단위 데이터를 MSB부터 또는 LSB부터 출력할 수 있다.

```

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t value)
매개변수
    dataPin : 비트 출력이 일어날 핀
    clockPin : 한 비트 출력 후 데이터 출력을 알려주기 위해 펄스가 출력되는 핀
    bitOrder : 비트 출력 순서, MSBFIRST 또는 LSBFIRST
    value : 출력될 데이터 (byte)
반환값 : 없음

```

코드 2는 shiftOut 함수를 이용하여 코드 1과 동일한 동작을 하도록 작성된 코드의 예이다.

코드 2. 74595 칩으로 8개 LED 제어 - 아두이노

```
int dataPin = 11;
```

```
int latchClockPin = 12;
int shiftClockPin = 13;

int index = 0;

void setup() {
  pinMode(dataPin, OUTPUT);
  pinMode(shiftClockPin, OUTPUT);
  pinMode(latchClockPin, OUTPUT);
}

void loop() {
  byte pattern = 1 << index;
  index = (index + 1) % 8;

  shiftOut(dataPin, shiftClockPin, MSBFIRST, pattern);    // 데이터 출력

  digitalWrite(latchClockPin, HIGH);                      // 래치 출력
  digitalWrite(latchClockPin, LOW);

  delay(200);
}
```