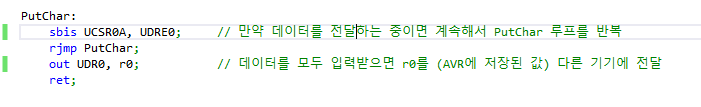
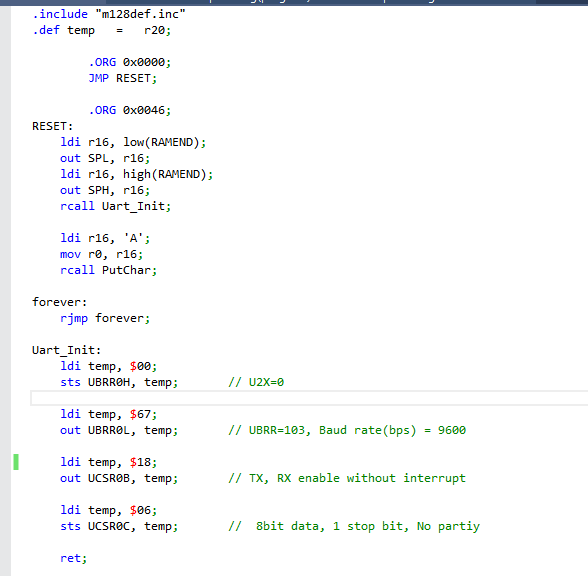
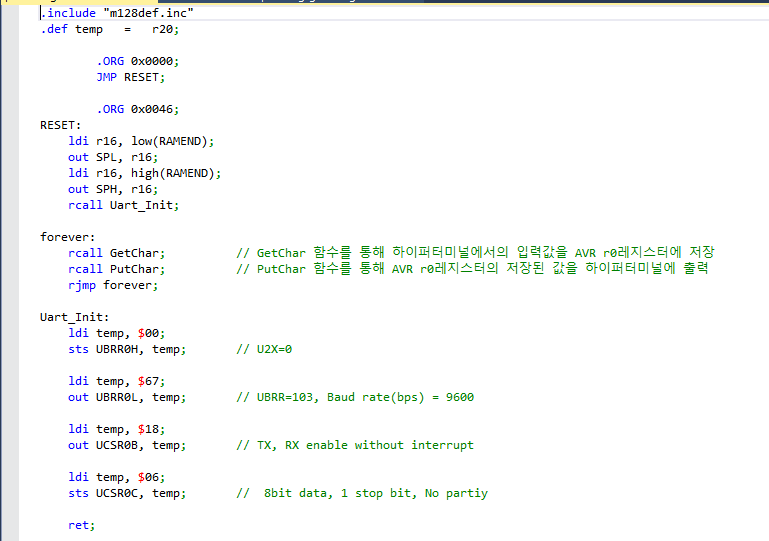
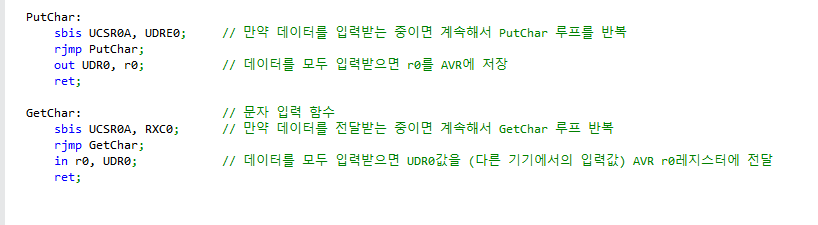
Lab 8-1



UART 직렬통신을 통해 AVR에 문자 ‘A’를 출력하는 프로그램을 저장하고 모드를 바꾸어 다른 기기에 AVR에 저장되어 있는 프로그램을 통해 ‘A’를 출력시킨다.

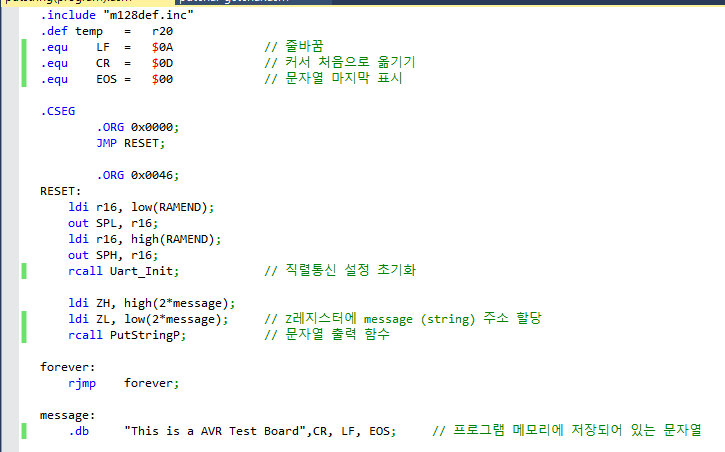
Lab 8-2

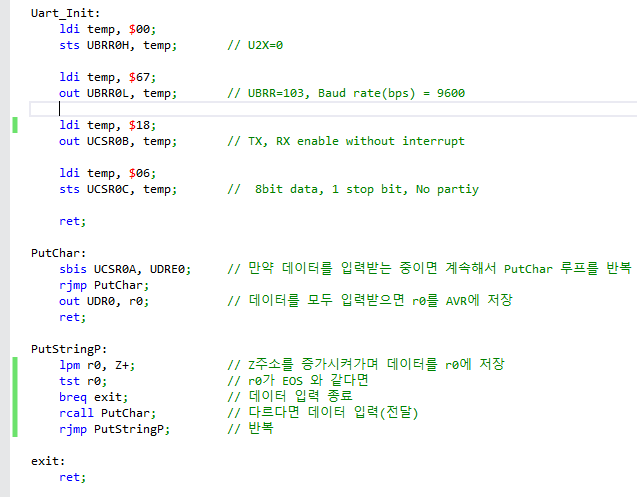




하이퍼터미널로부터 입력받은 아스키값을 AVR에 저장하고 그 값을 다시 하이퍼터미널로 출력하는 예제이다. 사용자는 하이퍼터미널에 키보드 입력값을 눈으로 바로 확인할 수 있는데 이는 눈으로 확인하는 키보드 값은 AVR이 다시 putchar함수를 통해 하이퍼 터미널로 보내주는 값이다.

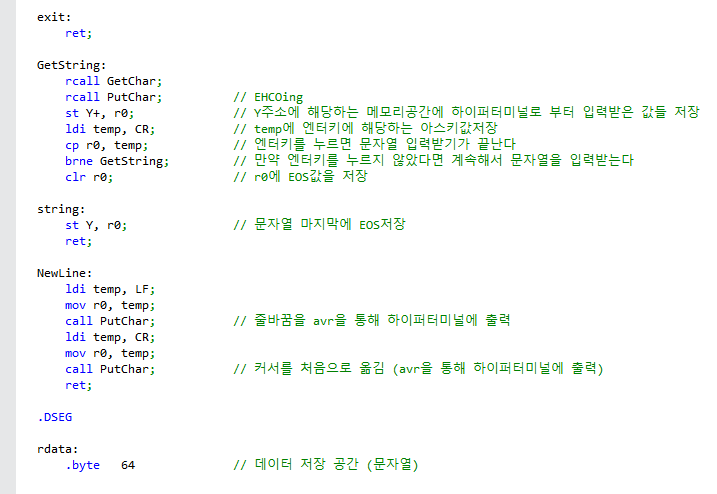
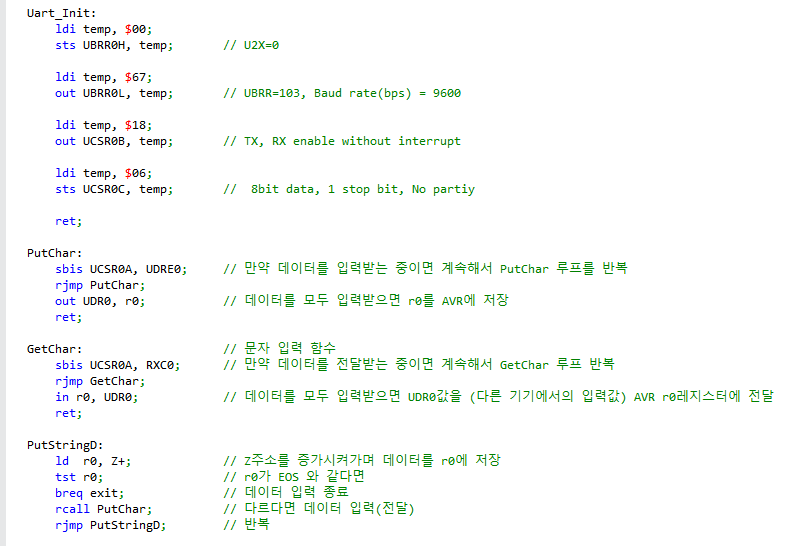
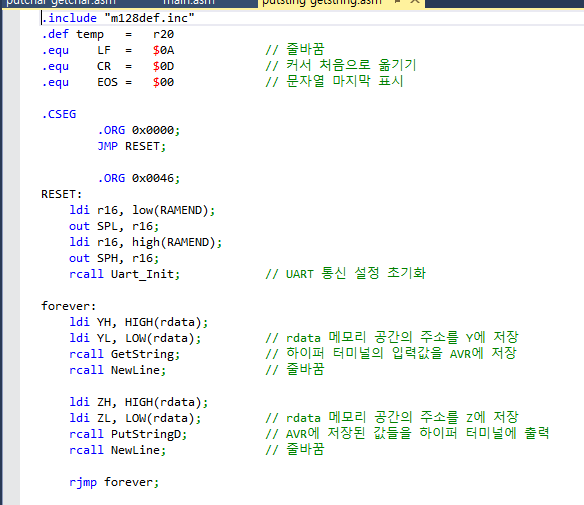
lab 8-3





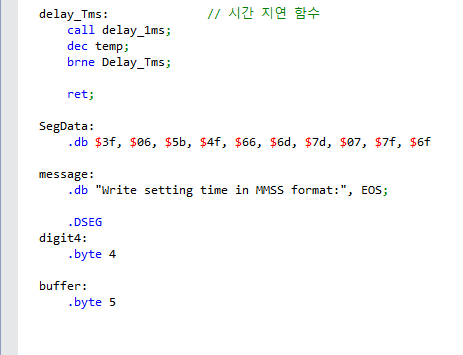
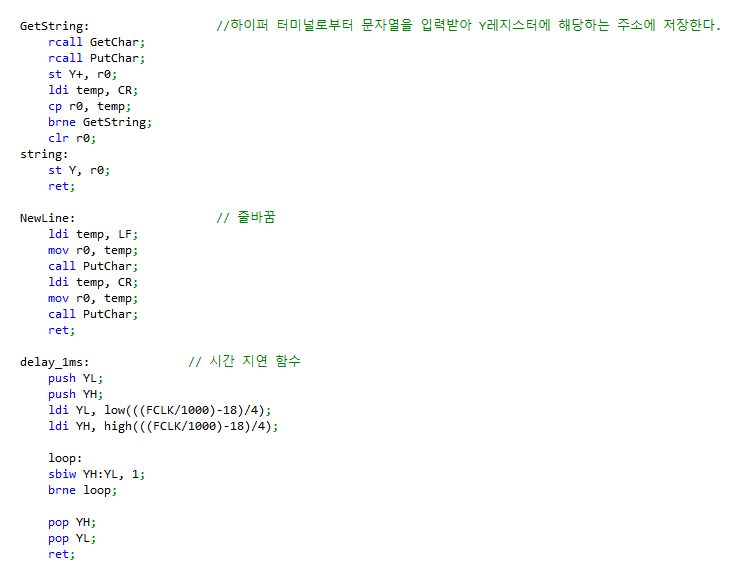
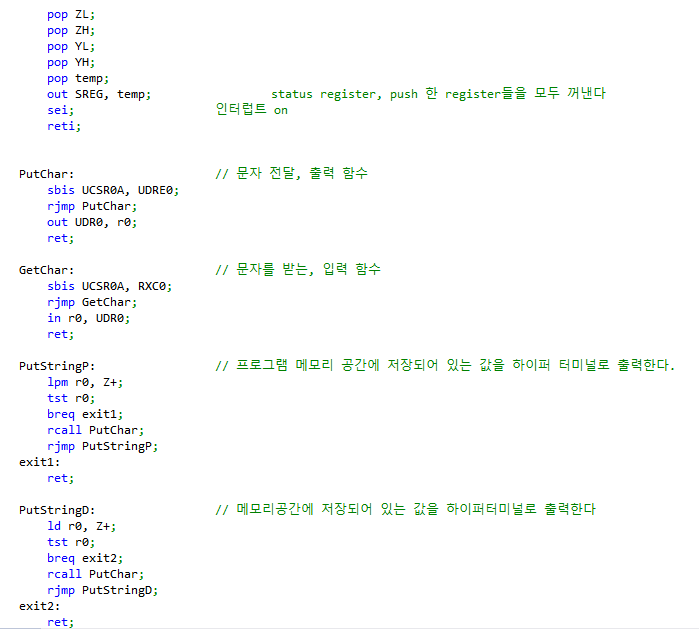
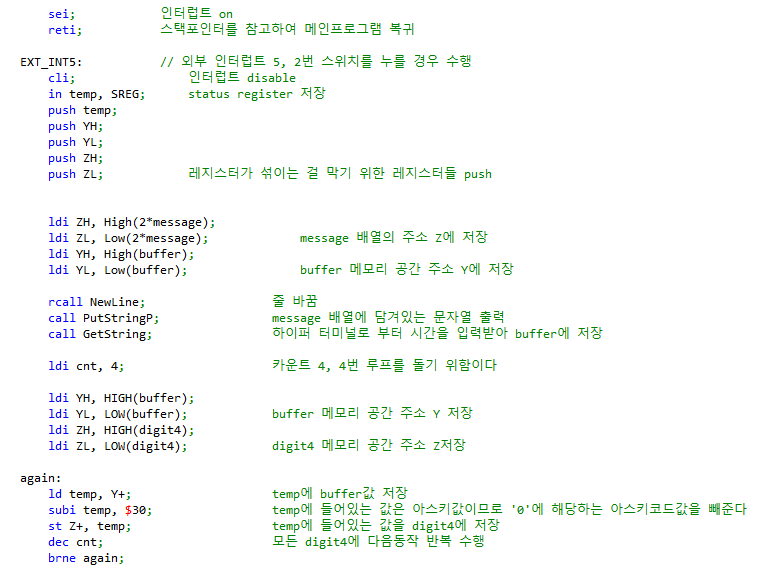
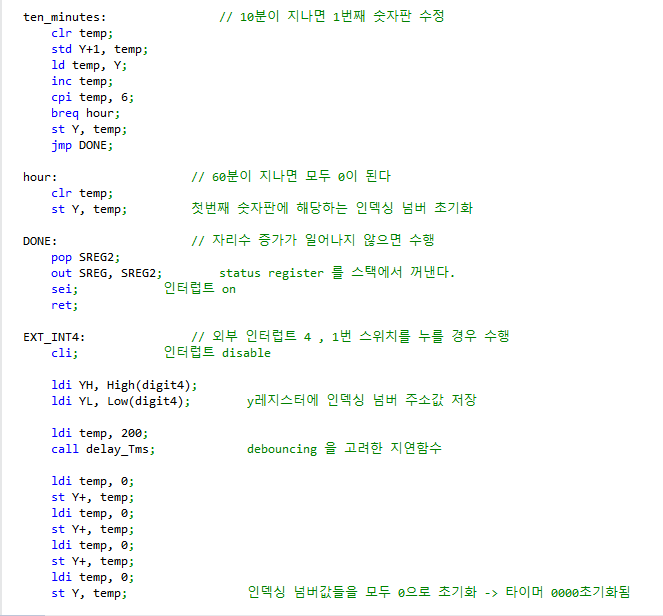
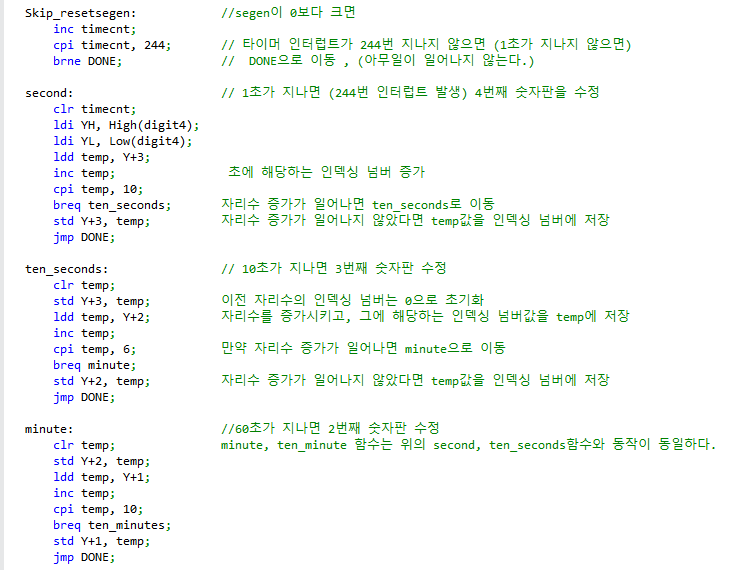
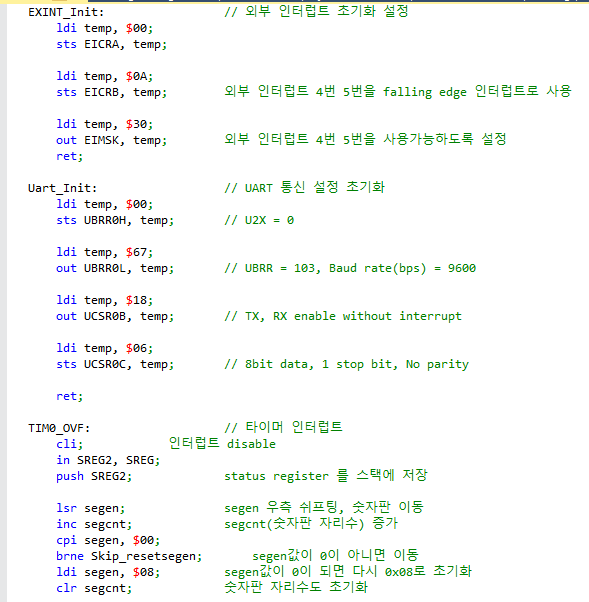
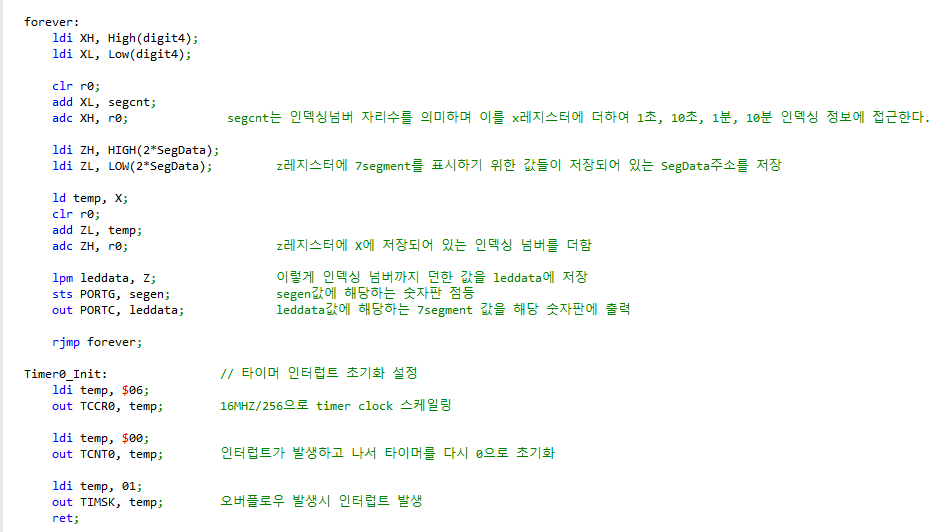
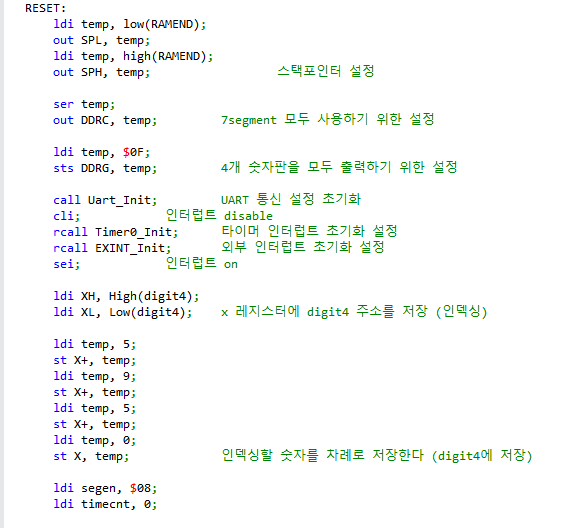
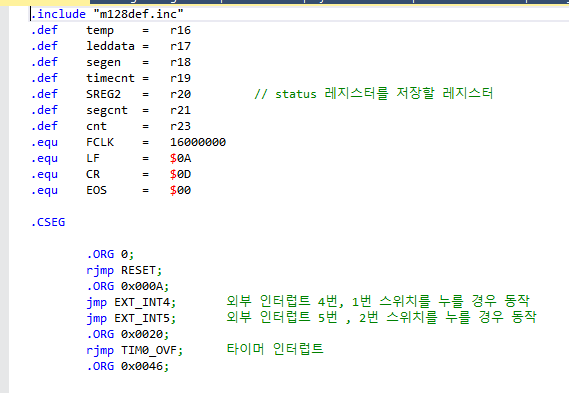
AVR의 프로그램 메모리에 저장되어 있는 문자열의 주소를 Z레지스터에 저장, Z레지스터를 이용하여 문자열의 끝(EOS)를 만나기 전까지 문자데이터를 putchar함수를 반복하여 AVR데이터를 하이퍼터미널로 보낸다.

lab 8-4



lab 8-2에서와 같이 키보드로 입력한 값을 눈으로 확인할 수 있는 예제이다. 다만 차이점은 8-2에서는 character 단위로 통신을 주고받았다면 위 예제는 string 단위로 통신을 한다. 즉 데이터 메모리 공간에 하이퍼 터미널로부터 입력받은 문자열을 저장하고, 그 값을 다시 AVR에서 하이퍼터미널로 송신하여 사용자가 확인할 수 있다. 이때 하이퍼 터미널에서의 입력과정에서 역시 사용자가 눈으로 확인할 수 있도록 하였는데 이는 character 단위의 Echoing이다. 만약 사용자가 엔터키를 입력하게 되면 문자열의 입력이 끝난 것으로 간주하여 그 값을 AVR에 저장하고 , 다시 AVR로부터 수신 받는다.

lab 8-5



이전 실습에서 수행하였던 타이머 인터럽트 실습에서 외부 인터럽트 5번에 시간 일시정지대신

UART통신을 통해 원하는 시간값을 입력받고 해당 시간값으로 타이머를 변경시키는 실습이다.

위 실습은 지금 껏 배워왔던 내용을 모두 활용하여 수행할 수 있으며 자세한 코드의 설명은 주석처리 하였다.

고찰)

이번 실습을 통해 UART통신에 대해 배웠다. 하이퍼 터미널을 이용하여 AVR과 직렬 통신을 진행하는 과정은 낯설지만 흥미로웠다. 그 전 실습까지는 AVR은 다운받은 프로그램을 수행하는 것 밖에 못하는 줄 알았는데, 모드를 바꿔 다운받은 프로그램을 통해 AVR이 통신까지 할 수 있다는 사실은 놀라웠다. UART 역시 관련된 다양한 레지스터가 있지만 레지스터들의 bit 설정에 따른 동작과정을 이해할 수 있다면 크게 어렵지 않은 정도였다. lab 8-5의 경우엔 지금껏 배웠던 타이머 인터럽트, 외부 인터럽트, UART통신을 모두 집대성한 실습으로 여태까지 배워왔던 것들을 한번 더 정리하는 시간을 가졌다. UART통신역시 레지스터의 기능과 동작만 이해하면 코딩 자체는 크게 어렵지 않은 것 같다.