KOMUT SATIRI	AÇIKLAMA
ORG 00H	bu satırdan sonraki kodlar 00H adresinden başlasın
SJMP MAIN	MAIN etiketine atla
ORG 03H	bu satırdan sonraki kodlar 03H adresinden başlasın bu adres INT0(Harici Kesme 0) kesmesinin çalıştığı adrestir
LJMP SENSOR	SENSOR etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) INT0 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 1BH	bu satırdan sonraki kodlar 1BH adresinden başlasın bu adres T1(Zamanlayıcı 1) kesmesinin çalıştığı adrestir
LJMP TMR	TMR etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) T1 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 30H	bu satırdan sonraki kodlar 30H adresinden başlasın ana programımın bulunduğu adrestir
MAIN: CLR P1.0	başlangıçta iticinin durduğu biliniyormuş bu nedenle iticinin bağlı olduğu P1.0 pinini lojik 0 yapıyorum
SETB P1.1	başlangıçta motorun çalıştığı biliniyormuş bu nedenle motorun bağlı olduğu P1.1 pinini lojik 1 yapıyorum
MOV IE, #89H	IE kaydedicisine 89H bilgisini yaz burada 89H(1000 1001B) yazılmasının nedenine bakarsak 8 bitin yüksek kısmındaki ilk bit EA bitidir ve kesme kullanılacaksa 1 yapılmalıdır 8 bitin yüksek kısmında bizi ilgilendiren başka bit bulunmadığı için kalanına 0 yazıyoruz 8 bitin düşük kısmındaki ilk bit ET1 bitidir T1 kesmesini kullanmamız gerektiği için 1 yazıyoruz son bit EX0 bitidir INT0 kesmesini kullanmamız gerektiği için 1 yazıyoruz 8 bitin düşük kısmında bizi ilgilendiren başka bit bulunmadığı için kalanına 0 yazıyoruz
MOV TMOD, #18H	TMOD kaydedicisine 18H bilgisini yaz burada 18H(0001 1000B) yazılmasının nedenine bakarsak 8 bitin yüksek kısmı T1 ve düşük kısmı T0 içindir her biri için kullanılan 4 biti kısaca şu şekilde açıklayabilirim GATE C/T M1 M0 GATE: zamanlayıcı/sayıcının aktifliğini kontrol eder zamanlayıcı/sayıcının aktif olabilmesi için değeri 0 olmalıdır C/T: zamanlayıcı/sayıcının zamanlayıcı mı sayıcı mı olarak kullanılacağını kontrol eder 0 değeri verilirse zamanlayıcı 1 değeri verilirse sayıcı olur M1 M0: zamanlayıcı/sayıcının modunu belirtir açıklamaya yüksek kısımdan başlarsam T1 zamanlayıcı/sayıcısını kullanacağım için GATE biti 0 olmalıdır T1 zamanlayıcı/sayıcısını zamanlayıcı olarak kullanacağım için C/T biti 0 olmalıdır T1 zamanlayıcı/sayıcısını 60ms(60 000µs) sayma amacıyla kullanacağım için sadece MOD 1'i kullanabilirim bu nedenle M1 M0 bitleri 01 olmalıdır düşük kısmı açıklarsam T0 zamanlayıcı/sayıcısını kullanmadığım için düşük kısma verilen değer çok önemli değil ancak sistemde T0'ın çalışmadığından emin olmak için GATE bitine 1 vermeyi seçtim böylece T0'ı pasif hale getirdim
SJMP $	sonsuz döngü

SENSOR: SETB P1.0	sensör tetiklendiğinde iticinin çalıştığı biliniyormuş bu nedenle iticinin bağlı olduğu P1.0 pinini lojik 1 yapıyorum
CLR P1.1	sensör tetiklendiğinde motorun durduğu biliniyormuş bu nedenle motorun bağlı olduğu P1.1 pinini lojik 0 yapıyorum
MOV TL1, #LOW(5536)	T1 zamanlayıcısının düşük kısmına 5536 sayısının düşük kısmını yaz
MOV TH1, #HIGH(5536)	T1 zamanlayıcısının yüksek kısmına 5536 sayısının yüksek kısmını yaz
SETB TR1	T1 zamanlayıcısını başlat
RETI	kesmeden dön
TMR: CLR P1.0	T1 tetiklendiğinde iticinin durduğu biliniyormuş bu nedenle iticinin bağlı olduğu P1.0 pinini lojik 0 yapıyorum
SETB P1.1	T1 tetiklendiğinde motorun çalıştığı biliniyormuş bu nedenle motorun bağlı olduğu P1.1 pinini lojik 1 yapıyorum
CLR TR1	T1 zamanlayıcısını durdur
RETI	kesmeden dön
END	kodu bitir

Kod içerisinde bulunan
CLR P1.0 SETB P1.1 satırları yerine MOV P1, #02H
ve
SETB P1.0 CLR P1.1 satırları yerine MOV P1, #01H
satırı yazılabilir. Daha kısadır ancak kafa karıştırmaya daha uygundur.
Bu nedenle kod içerisinde gösterdiğim gibi yapılmasını öneririm.
T1 zamanlayıcısının değerlerini yazdırırken kullanılan 5536 değeri MOD 1 ve 60 ms den gelir.
Yani 60 ms (60 000µs) sayabilmek için T1'in değeri maksimumdan 60000 kadar geriye gitmelidir.
MOD 1'in maksimum değeri olan 65536(16 bit - 2^{16}) sayısından 60000 geri gidersek 5536 kalır. Bu değer yerine geriye gitmek negatiflik olacağı için direk –60 000 yazabilirdik.
Son olarak hocaların yapmış olduğu çözüm ile aradaki önemli bir farkı açıklamak istiyorum. Yani TCON kullanımını açıklıyorum. TCON kaydedicisi yüksek 4 bitinde T0 ve T1 zamanlayıcı/sayıcıları ile ilgili kontrolü için kullanılır. Buradaki TF1 T1 zamanlayıcı/sayıcısının taşma biti, TR1 T1 zamanlayıcı/sayıcısının başlatma/durdurma biti, TF0 T0 zamanlayıcı/sayıcısının taşma biti ve TR0 T0 zamanlayıcı/sayıcısının başlatma/durdurma bitidir. Düşük 4 bitinin T0 veya T1 ile ilgisi yoktur. Harici kesmelerle ilgilidir. Buradaki IT0 değeri 0 yapıldığında INT0 ucu lojik 0 seviyesinde, 1 yapıldığında INT0 ucu düşen kenarda kesme üretir, IE0 INT0 kesmesi oluştuğunda sistem tarafından 1, kesmeden dönüş komutu olan RETI çalıştırıldığında sistem tarafından 0 değeri verilir, IT1 değeri 0 yapıldığında INT1 ucu lojik 0 seviyesinde, 1 yapıldığında INT1 ucu düşen kenarda kesme üretir, IE1 INT1 kesmesi oluştuğunda sistem tarafından 1, kesmeden dönüş komutu olan RETI çalıştırıldığında sistem tarafından 0 değeri verilir. Hocaların çözümde vermiş olduğu MOV TCON, #41H satırı T1 zamanlayıcısını çalıştırır ve INT0 kesmesinin düşen kenarda tetiklenmesini sağlarken MOV TCON, #0H satırı T1 zamanlayıcısını durdurur ve INT0 kesmesinin lojik 0'da tetiklenmesini sağlar. Burada INT0 kesmesinin tetiklenme şekli sabit olduğundan IT0 biti MAIN içerisinde lojik 1 yapılması daha mantıklıdır diye düşünüyorum. TCON ile ilgili genel olarak bilgi az olduğu için kodun içinde kullanmak yerine sadece bu tarz bir açıklama yapmayı uygun buldum. Herkes kendisine doğru gelen haliyle çözüm yapsın diye kafa karıştırmak istemedim.