

**GEÇİŞ ZAMANI ESASLI
ULTRASONİK AKIŞ ÖLÇER**

Emre ERTAŞ

**YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ**

**GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

OCAK 2007

ANKARA

Emre ERTAŞ tarafından hazırlanan GEÇİŞ ZAMANI ESASLI ULTRASONİK AKIŞ ÖLÇER adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. İres İSKENDER
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliğiyle Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan : Yrd. Doç. Dr. Mehmet Timur AYDEMİR

Üye : Yrd. Doç. Dr. İres İSKENDER

Üye : Yrd. Doç. Dr. Hasan Şakir BİLGE

Üye : _____

Üye : _____

Tarih : 15/01/2007

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Emre ERTAŞ

**GEÇİŞ ZAMANI ESASLI
ULTRASONİK AKIŞ ÖLÇER
(Yüksek Lisans Tezi)**

Emre ERTAŞ

**GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

Ocak 2007

ÖZET

Bu çalışmada, kalori-metrik akış ölçerlerde kullanımı giderek yaygınlaşan ultrason ile adlandırılan yüksek frekanslı ses dalgaları teknolojisi, alışılmış akış ölçerler üzerinde denenmiştir. Akış ölçümü için geçiş zamanı yöntemi esas alınmıştır. Çalışmada yüksek doğrulukta zaman ölçümü için sayısal zaman çevirici yongası ve PIC18F4520 mikro-denetleyici birlikte kullanılmıştır. Sistemde ayrıca, 4 MHz frekanslı iki ultrasonik transdüser kullanılmıştır. Basit yapısı nedeniyle çapraz biçim tüp tercih edilmiştir. Çalışmada söz konusu akış ölçerin maruz kaldığı etkilere ve bu etkilerden dolayı oluşan ölçüm doğruluğu hatalarının deneysel sonuçlarına yer verilmiştir.

Bilim Kodu	: 905.1.035
Anahtar Sözcükler	: Akış ölçer, ses ötesi, ölçüm doğruluğu, kalibrasyon
Sayfa Adedi	: 145
Tez Yöneticisi	: Yrd.Doç.Dr. İres İSKENDER

**THE TRANSIT TIME BASED
ULTRASONIC FLOW METER**

(M.Sc. Thesis)

Emre ERTAŞ

**GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY**

January 2007

ABSTRACT

In this study, the high frequency sound wave technology which is known as ultrasound and is becoming widely used in calorie-metric flow meters was experimented on the ordinary flow meters. The transit time method has been considered as the basis of the flow measure. Time to digital converter and the PIC18F4520 as a microcontroller were used to achieve a high accuracy time measurement in this study. In addition, two 4 MHz ultrasound transducers were used in the system. A diagonal type tube was used due to its simple construction. The effects that the flow meter is exposed to and the experimental results accuracy errors obtained from these effects are studied in the study.

Science Code	: 905.1.035
Key Words	: Flow meter, ultrasound, measurement accuracy, calibration
Page Number	: 145
Adviser	: Assist.Prof.Dr. İres İskender

TEŞEKKÜR

Çalışmalarım boyunca yardım ve katkılarıyla beni yönlendiren değerli hocam ve danışmanım Yrd. Doç. Dr. İres İSKENDER'e, değerli hocalarım ve çalışma arkadaşlarıma teşekkürlerimi sunmayı bir borç bilirim.

Ayrıca maddi ve manevi desteklerinden ötürü sevgili annem Aysel ERTAŞ ve babam Ömer ERTAŞ'a çok teşekkürler.

İÇİNDEKİLER

Sayfa

ÖZET	iv
ABSTRACT	v
TEŞEKKÜR	vi
İÇİNDEKİLER	vii
ÇİZELGELERİN LİSTESİ	x
ŞEKİLLERİN LİSTESİ	xi
RESİMLERİN LİSTESİ	xiv
SİMGELER VE KISALTMALAR	xv
1. GİRİŞ	1
2. AKIŞKAN ÖZELLİKLERİ VE TEMEL AKIŞ ÖLÇÜM PRENSİPLERİ	4
2.1. Akış Oranı	4
2.1.1. Noktasal hız	4
2.1.2. Ortalama akış hızı	5
2.1.3. Akış oranı	5
2.2. Akış Ölçümüne Tarihsel Bakış	6
2.3. Çeşitli Akış Ölçer Yapıları	8
2.3.1. Basınç farkı esaslı akış ölçerler	8
2.3.2. Diğer akış ölçüm prensipleri	11
3. SES VE ULTRASES	14
3.1. Ses Dalgalarının Fiziği	14
3.2. Ses Dalgalarının Özellikleri	15
3.3. Ultrasonik Transdüserler	17

Sayfa

3.3.1. Piezoelektrik olay	18
3.4. Ultrasonik Dalgaların Su İçerisinde Yayılma Hızları	18
4. ULTRASONİK AKIŞ ÖLÇME	20
4.1. Ultrasonik Akış Ölçümüne Tarihsel Bakış.....	20
4.2. Ultrasonik Akış Ölçme Yöntemleri.....	21
4.2.1. Doppler akış ölçer	21
4.2.2. Çapraz ilişkili akış ölçer	24
4.2.3. Geçiş zamanı esaslı ultrasonik akış ölçer	26
5. TASARLANAN AKIŞ ÖLÇERİN BİLEŞENLERİ.....	40
5.1. Mikrodenetleyici	40
5.2. Ultrasonik Transdüserler	41
5.3. Sayısal Zaman Çevirici	42
5.3.1. Okunabilir ve yazılabilir kayıtçılar	46
5.3.2. Zaman ölçüm akışı	52
5.3.3. Darbe üretici ve sıcaklık ölçüm birimi	53
6. TASARIM HAKKINDA DETAYLI BİLGİLER.....	58
6.1. TDC GP2 Konfigürasyonu.....	58
6.1.1. Ölçüm parametrelerinin konfigürasyonu	58
6.1.2. Donanıma göre uyarlanan parametreler	59
6.1.3. Diğer parameteler	61
6.2. Donanım	61
6.3. Gömülü Yazılım	73
7. DENEYSEL ÇALIŞMALAR	75

Sayfa

8. TARTIŞMA VE SONUÇLAR	88
KAYNAKLAR	91
EKLER.....	94
EK-1 main.c fonksiyonları	95
EK-2 main.h tanımlamaları	105
EK-3 gp2.c fonksiyonları	108
EK-4 gp2.h tanımlamaları	113
EK-5 kalibrasyon.c fonksiyonları	116
EK-6 kalibrasyon.h tanımlamaları	123
EK-7 eeprom.c fonksiyonları	125
EK-8 rtc.c fonksiyonları	128
EK-9 _delays.c fonksiyonları	131
EK-10 _delays.h tanımlamaları	132
EK-11 ext_E2_driver.c fonksiyonları	133
EK-12 ext_E2_driver.h tanımlamaları	137
EK-13 lcd.c fonksiyonları	138
EK-14 lcd.h tanımlamaları	140
EK-15 lcd_drv.c fonksiyonları	141
EK-16 global.h tanımlamaları	144
ÖZGEÇMİŞ	145

ÇİZELGELERİN LİSTESİ

Çizelge	Sayfa
Çizelge 2.1. Farklı ölçüm esaslı akış ölçerlerin prensiplerinin ve endüstride ilk kullanılış tarihleri	7
Çizelge 3.1 Bazı materyaller içerisinde sesin yayılım hızı	15
Çizelge 4.1. Suyun kinematik ve dinamik viskozitelerinin sıcaklıkla değişimi	33
Çizelge 4.2. Örnek kalibrasyon tablosu	37
Çizelge 5.1. Ölçümde kullanılan darbelerin zaman gereksinimleri	45
Çizelge 5.2. TDC GP2 için operasyon kodları.....	45
Çizelge 5.3. Yazılabilir Kayıtcı0.....	47
Çizelge 5.4. Yazılabilir Kayıtcı1	48
Çizelge 5.5. Yazılabilir Kayıtcı2.....	49
Çizelge 5.6. Yazılabilir Kayıtcı3.....	49
Çizelge 5.7. Yazılabilir Kayıtcı4.....	50
Çizelge 5.8. Yazılabilir Kayıtcı5.....	50
Çizelge 5.9. Okunabilir Kayıtcılar	51
Çizelge 7.1. Test edilen akış ölçerin kalibrasyon tablosu	77
Çizelge 7.2. Doğruluk testi sonuçları(20°C)	78
Çizelge 7.3. Akış ölçerin 200 mm önünde akış bozucu koyularak Q3’de yapılan doğruluk testleri (filtresiz).....	80
Çizelge 7.4. Akış ölçerin 200 mm önünde akış bozucu koyularak Q3’de yapılan doğruluk testleri (filtreli).....	81
Çizelge 7.5. Doğruluk testi sonuçları(35 °C)	82
Çizelge 7.6. Doğruluk testi sonuçları(5 °C)	83

ŞEKİLLERİN LİSTESİ

Şekil	Sayfa
Şekil 1.1. Farklı ölçüm esaslı akış ölçerlerin 2001 ve 2005 yılları arasındaki tercih yüzdeleri arasındaki değişim.....	1
Şekil 2.1. Orifis akış ölçer.....	9
Şekil 2.2. Venturi akış ölçer.....	9
Şekil 2.3. Dirsek tipi akış ölçer.....	10
Şekil 2.4. Pitot tüpü.....	11
Şekil 2.5. Coriolis akış ölçer.....	13
Şekil 3.1. Ses dalgasının genliği ve periyodu.....	15
Şekil 3.2. İki farklı ortam yüzeyinde kırılma ve yansıma.....	16
Şekil 3.3. Su içerisinde ses-ötesi dalga'nın yayılma hızının sıcaklığa bağlı değişimi.....	19
Şekil 4.1. Doppler kayma prensibi.....	21
Şekil 4.2 Ultrasonik doppler akış ölçer.....	22
Şekil 4.3. Ultrasonik çapraz ilişkili akış ölçer.....	25
Şekil 4.4. Alternatif ultrasonik çapraz ilişkili akış ölçer.....	26
Şekil 4.5. Örnek bir boylamsal biçimli tüp yapısı.....	27
Şekil 4.6. Örnek bir çapraz biçimli tüp yapısı.....	27
Şekil 4.7. Ultrasonik akış ölçer şeması.....	29
Şekil 4.8. Tipik ultrasonik dalga darbesi.....	29
Şekil 4.9. Boylamsal tüp içerisinde hız profilinin simulasyonu.....	34
Şekil 4.10. Çapraz biçim tüp içerisinde hız profilinin simulasyonu.....	34
Şekil 4.11. İyileştirilmiş çapraz biçimli tüpten 18 °C derecede elde edilmiş kalibrasyon eğrisi.....	35

Şekil	Sayfa
Şekil 4.12. Geliştirme yapılmamış bir çapraz biçimli tüpten 20 °C derecede elde edilmiş kalibrasyon eğrisi	35
Şekil 4.13. İyileştirilmiş çapraz biçim tüpün hata dağılımı	36
Şekil 4.14. İyileştirilme yapılmamış çapraz biçimli tüpün hata dağılımı	36
Şekil 4.15. Kalibrasyon eğrisi	37
Şekil 5.1. Kullanılan 4MHz transdüserin yankı tepkisi	41
Şekil 5.2. Zaman ölçüm prensibi.....	43
Şekil 5.3. Kullanılan ölçüm darbeleri	44
Şekil 5.4. TDC GP2 zaman ölçümüm akış diyagramı	52
Şekil 5.5. TDC GP2 darbe üretici için yedi darbeli örnek darbe dizisi	54
Şekil 5.6. TDC GP2 darbe üretici blok diyagramı	54
Şekil 5.7. TDC GP2 darbe üretici frekans çıkışı	55
Şekil 5.8. Sıcaklık sensörleri bağlantı şekli	55
Şekil 5.9. TDC GP2 sıcaklık ölçümü zamanlamaları	56
Şekil 6.1. Gerilim seviyeleri ve pil gerilimi ölçme devresi.....	62
Şekil 6.2. TDG GP2 ve osilatörler	64
Şekil 6.3. Transdüser tetikleme düzeneği	66
Şekil 6.4. Transdüser tetikleme sinyalleri.....	67
Şekil 6.5. STOP darbesini oluşturan karşılaştırmacı birim.....	67
Şekil 6.6. START ve STOP sinyalleri	68
Şekil 6.7. STOP sinyallerinin ayrıntılı görünüşü	68
Şekil 6.8. Mikrokontrolör devresi	70
Şekil 6.9. I ² C eeprom devresi.....	71

Şekil	Sayfa
Şekil 6.10. M-BUS ara yüzü köle yonga devresi	71
Şekil 6.11. LCD sürücü ve cam lcd için bağlantı biçimi	72
Şekil 6.12. Gömülü yazılım akış diyagramı.....	74
Şekil 7.1. Test edilen akış ölçerin kalibrasyon eğrisi.....	77
Şekil 7.2. Doğruluk testi dağılımları(20°C)	78
Şekil 7.3. Doğruluk testi hata dağılımları(35 °C).....	82
Şekil 7.4. Doğruluk testi hata dağılımları(5 °C).....	83
Şekil 7.5. Hesaplanan akış hızının standart sapması.....	84
Şekil 7.6. Hata dağılımı.....	84
Şekil 7.7. Basınç kaybı değişimi	85
Şekil 7.8. Etrafa çınlama döngüsü	86

RESİMLERİN LİSTESİ

Resim	Sayfa
Resim 5.1. Kullanılan transdüserlerin dış görünüşü	41
Resim 7.1. Filtreler.....	75
Resim 7.2. Kullanılan akış bozucular	80
Resim 7.3. Akış ölçerin dış görünüşü	85

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklama
a	Tüp kesiti
c	Ses hızı
d	Çap
e	Elastiklik katsayısı
f	Frekans
g	Kütle ağırlığı, kg/m^3
h_w	İki nokta arasındaki basınç farkı
l	Sensör mesafesi, mm
n	Dinamik viskozite
p	Periyot
q	Hacimsel akış oranı
q_m	Kütleli akış oranı
t	Zaman
t_1	Akış yönünde iletim zamanı
t_2	Akış yönünün tersi yönde iletim zamanı
u	Ortalama akış hızı
v	Noktasal akış hızı
z	Akustik empedans
λ	Dalga boyu
ρ	Yoğunluk, g/cm^3
Kısaltmalar	Açıklama
ALU	Aritmetik Mantık Birimi

Kısaltmalar**Açıklama****GZE**

Geçiş Zamanı Esaslı

K

Kalibrasyon Faktörü

DAB

Düşük Anlamlı Bit

SHG

Ses Hızının Gecikmesi

SKG

Sıfır Kalibrasyon Gecikmesi

TDC

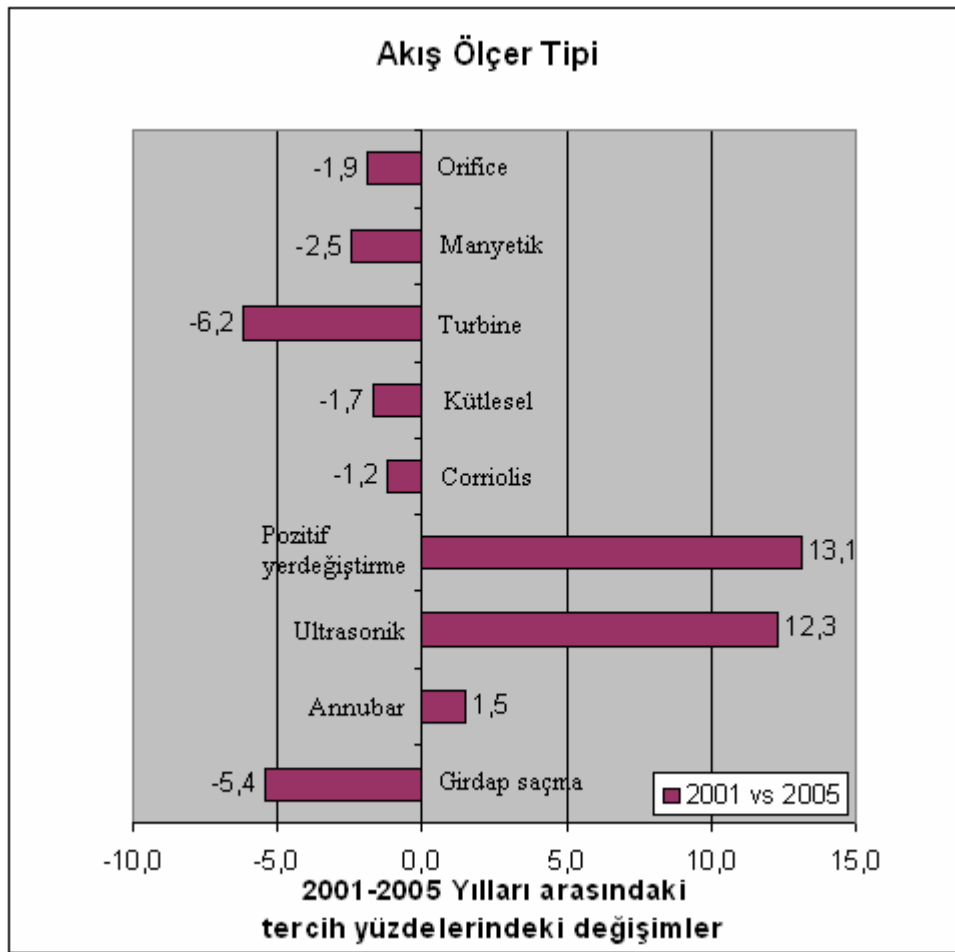
Sayısal Zaman Çevirici

YAB

Yüksek Anlamlı Bit

1. GİRİŞ

Günümüzde akış ölçerler endüstrinin birçok alanında farklı amaçlara hizmet etmektedir. Basit mekanik yapıları, düşük basınç kayıpları, kolay kurulum, düşük maliyet, içerisinde hareketli mekanik aksam bulundurmamaları, yaşlanma ve yıpranma etkilerinin az olması, yüksek doğruluk, ultrasonik akış ölçerlerin avantajlarıdır. Kurulum etkilerine ve diğer hata kaynaklarına karşı duyarlı olmaları dezavantajlarıdır. Ultrasonik akış ölçerlerin bu özelliklerinden dolayı son zamanlarda kullanımı giderek artmaktadır ve gelişimi devam etmektedir. Şekil 1.1, 2001 ve 2005 yılları arasında farklı akış ölçer tiplerinin tercih yüzdelerindeki değişimleri göstermektedir [1].



Şekil 1.1. Farklı ölçüm esaslı akış ölçerlerin 2001 ve 2005 yılları arasındaki tercih yüzdeleri arasındaki değişim

Akış ölçme alanındaki yenilikler ile birlikte farklı ölçüm prensipleriyle üretilen akış ölçerler % 0.5 doğruluk hassasiyetine erişebilmektedirler. Düşük maliyet, üretim kolaylığı, uzun ömürlülük, yıpranma etkilerinin az olması, kolay bakım, tekrarlanabilirlik, düşük basınç kaybı, çevresel etmenlere ve akışkanın hız profiline karşı duyarlılığın az olması akış ölçerde aranan diğer özelliklerdir.

Bu çalışmada kalori-metrik akış ölçerlerde kullanımı giderek yaygınlaşan ultrason ile adlandırılan yüksek frekanslı ses dalgaları teknolojisi, ISO4064-1 standardına uygun ev tipi su sayacı olarak denenmiştir. Akış ölçümü için, transdüserler üzerinden gönderilen ultrasonik ışın demetlerinin iletim sürelerindeki değişimler esas alınmıştır (geçiş zamanı yöntemi). Ülkemizde üretilen ev tipi su sayaçlarının tamamı mekanik yapıdır. Bu tip mekanik akış ölçerlerde sıvı akışkan bir türbin, rotor yada herhangi bir pervane düzeneği üzerinden akar ve bu akış oranıyla orantılı bir dönme hareketi oluştur. Rotor hızı akış oranı ile birlikte doğrusal olarak artar. Böylece rotor turları ile akış oranı ölçülmektedir. Sudaki tortular bu sayaçlar için tehlikelidir. Maliyeti yüksektir. Bunlara rağmen geniş bir ölçüm aralığında iyi doğrulukta çalışabilirler. Özellikle düşük akış oranları için akışkanın hız profilinden, pervanelerdeki ve gövdedeki diğer sürtünmelerden etkilenmektedirler. En büyük problemi akışkan çok hızlı olduğunda yada akışkanın birden hatta girdiğinde paletlerde aşınmalara yol açabilmesidir. Ultrasonik akış ölçer içerisinde hareket eden bir mekanik aksam bulundurmayaacağı için yıpranma ve yaşlanma etkileri mekanik sayaçlara göre çok daha az olacak ve sudaki tortu ve kirlilikten çok daha az etkilenecektir.

Son yıllarda elektronikteki gelişmeler ile birlikte elektronik malzemeler oldukça düşük maliyetler ile temin edilebilmektedir. Bu nedenden ötürü tamamen elektronik tasarım olan bir ultrasonik akış ölçeri, mekanik yapıdaki rakiplerine nazaran oldukça düşük maliyetlerde üretmek mümkündür. Kullanılan yüksek empedanslı piezoelektrik transdüserler ve düşük güçlü diğer yongalar arasında aşırı akım tüketecek bir materyal olmayacaktır. Bu nedenle pil ile çalışacak bir uygulama için ultrasonik akış ölçerler çok uygundur. Tasarımda kullanılan elektronik malzemeler özellikle pilli uygulamalara uygun çalışma gerilimlerine sahip ve düşük güç tüketimli

bileşenler arasından seçilmiştir. Böylelikle tasarlanan akış ölçerin 14000mAh lityum piller ile 10 yıldan daha fazla çalışabilmesi sağlanmıştır.

Çalışmada yüksek doğrulukta zaman ölçümü için sayısal zaman çevirici yongası ve PIC18F4520 mikrodenetleyici birlikte kullanılmıştır. Sistemde ayrıca, 4MHz frekanslı iki ultrasonik transdüser kullanılmıştır. Basit mekanik yapısı nedeniyle çapraz biçim tüp tercih edilmiştir. Kullanılan bu materyaller ile teorik olarak $11,2 \times 10^{-4}$ l/s akış oranı ölçüm çözünürlüğü elde edilmiştir. Bu çözünürlük bir çok uygulama için yeterlidir. Yinede ölçüm çözünürlüğünü arttırmak için etrafa çınlama döngüsü Bölüm 7’de kısa bir şekilde tanıtılmıştır.

Bir sonraki bölümünde tezin geri kalan bölümleri için akış oranı kavramının kısa olarak tanıtılması gerekli görülmüştür. Ayrıca bu bölümde akış ölçümünün tarihsel perspektifi ve çeşitli akış ölçüm prensipleri üzerinde durulmuştur. Üçüncü bölümde ses dalgalarının özelliklerine ve ultrasonik ses dalgalarına değinilmiştir. Bölüm 4’te ultrasonik ile akış ölçme metotları detaylı olarak incelenmiştir. Bölüm 5’de tasarımı yapılan akış ölçeri oluşturan temel bileşenler tanıtılmış ve tasarım altıncı bölümde detaylı olarak açıklanmıştır. Söz konusu akış ölçerin maruz kaldığı etkilere ve bu etkilerden dolayı oluşan ölçüm doğruluğu hatalarının deneysel sonuçlarına Bölüm 7’de yer verilmiştir. Son olarak sekizinci bölümde elde edilen bulgu ve sonuçlar tartışılmıştır.

2. AKIŞKAN ÖZELLİKLERİ VE TEMEL AKIŞ ÖLÇÜM PRENSİPLERİ

Akış, bir akışkanın bir hat boyunca sürekli olarak hareketidir. Akışkanlar maddenin üç hali şeklinde (Katı, Sıvı, Gaz) olabilir. Suyun nehirde akışı, petrol veya gazın bir boru hattından akışı, havanın nefes olarak ciğerlere alınıp verilmesi, helezonlarla tahılların taşınması akış sistemlerine birer örnektir.

Akış ölçümü için bilinmesi gereken önemli fiziksel büyüklükler; akışkanın birim zamanda akan hacimsel miktarı (Hacimsel debi yada hacimsel akış oranı), birim zamanda akan kütleli miktarı (Kütleli debi (kg/sn)) ve birim zamanda kat etmiş olduğu yol (Hız (m/sn))dur.

Direkt hacimsel veya kütleli ölçüm, ölçüm işlemini sonlandırır. Bazı ölçümlerde hacimsel veya kütleli ölçümlerin zorluğundan dolayı veya alternatif bir ölçüm sistemi olarak hız ölçümü kullanılır. Hız ölçümünün yapıldığı yerin kesiti bilineceği için çeşitli matematiksel işlemlerle hacimsel veya kütleli akış oranı bulunabilir. Ultrasonik akış ölçerler de bu şekilde hız ölçümünü kullanır.

2.1. Akış Oranı

Genel olarak akışı tanımlayan iki farklı bildirim mevcuttur:

1. ortalama akış hızı
2. hacimsel akış oranı

Fakat bu tanımlamalardan önce noktasal hız tanımı açıklanmalıdır.

2.1.1. Noktasal hız

Noktasal hız akışta ki en küçük akışkan elementin hızıdır. Noktasal hız ölçümü anemometri olarak adlandırılır ve pitot tüpü, lazer-doppler hız ölçer ve sıcak tel gibi yöntemlerle ölçülür. “m” noktasında, “t” zamanındaki noktasal hız $v(m,t)$ şeklinde gösterilir [2].

2.1.2. Ortalama akış hızı

Ortalama akış hızı “t” zamanında bir tüpün “a” kesitindeki noktasal hızların ortalaması şeklinde tanımlanır ve aşağıdaki gibi ifade edilir.

$$u(t) = \frac{1}{a} \int_{m \in a} v(m, t).ds(m) \quad (2.1)$$

Eşitlikte $u(t)$ ortalama akış hızı olarak adlandırılır iken, $ds(m)$ ise “a” kesiti içerisinde ki akışın “m” noktasındaki en küçük elementin yüzeyidir. Bu bilgi bu çalışmada önem taşımaktadır çünkü, bir ultrasonik akış ölçer tüp içerisindeki ortalama akış hızını öncelikle hesaplayarak hacimsel akış oranı bilgisine ulaşır.

2.1.3. Akış oranı

Hacimsel akış oranı q , genel olarak akış oranı şeklinde anılır. “a” kesitindeki hacimsel akış oranı basit olarak aşağıdaki eşitlikle ifade edilir.

$$q(t) = u(t).a \quad (2.2)$$

Akış oranı ifadesi ortalama akış hızı ifadesinden daha sezgiseldir. Direkt olarak tüm tüp içerisindeki akış ile ilgili bir bilgi taşır. Sıvı akışkanların sıkıştırılmaz özelliğine dayanarak, boru içerisindeki akışkan kesiti ve dolayısıyla akışkanın ortalama hızı ne kadar değişirse değişsin, boru içerisine birim zamanda giren akışkan miktarı aynı zaman diliminde boruyu terk edecektir. Genel olarak m^3/h , l/h , l/s gibi birimlerle ifade edilir.

Eğer akışkanın yoğunluğu biliniyorsa kütleli akış oranı $q_m(t)$, yoğunluk ρ ve $q(t)$ ’nin çarpımıyla bulunabilir. Genel olarak kg/h , kg/s gibi birimlerle ifade edilir.

$$q_m(t) = q(t).\rho \quad (2.3)$$

2.2. Akış Ölçümüne Tarihsel Bakış

Gazların ve sıvıların davranışlarını anlamamız eski yunan bilimci Archimede'nin çalışmalarına dayanmaktadır. 1687 yılında, İngiliz matematikçi Sir Isaac Newton genel yer çekimi kuramını keşfetmiştir. Bu gün kullanılan açılı momentum esaslı akış ölçerler Newton'un ikinci kuramına dayanmaktadır.

İsviçreli fizikçi Danielle Bernoulli Hidrodinamik adlı eserini yayınlamasıyla 1783 yılında akışkanları anlama konusunda bir dönüm noktası gerçekleşmiştir. Eserinde sıvı akışkanlar için enerji korunumu kavramını tanıtmıştır. Bernoulli akışkan hızındaki artışın beraberinde kinetik enerjisi artarken statik enerjisinin azaldığını belirtmiştir. Bir akış sınırlayıcı akışkan hızında artışa neden olurken statik basınç da ise düşüşe neden olur. Bu çalışmalar basınç farkı esaslı akış ölçerler için bir temel teşkil eder [3].

Manyetik akış ölçerlerin prensibi 1830'lu yıllarda İngiliz bilimci Michael Faraday manyetik indüksiyonla ilgili çalışmalarına dayanmaktadır.

1842'de Avusturyalı fizikçi Cristian Doppler ses kaynağı ile alıcı arasındaki mesafenin değişimiyle alınan sesin frekansındaki değişimi gözlemlemiştir. Çalışmalarından yaklaşık bir yüz yıl sonra ilk ultrasonik doppler akış ölçer markette yerini almıştır.

Fransız sivil mühendis Gaspard Coriolis 1843 yılında rüzgar, okyanus akıntıları ve hatta havadaki top mermileri dünyanın dönüşünden ve şeklinden dolayı yana sürüklendiğini fark etmiştir. Sensörler ve elektronikteki ağır gelişmeler nedeniyle ilk coriolis akış ölçer 1970'lerde üretilmiştir.

Amerikan-Macar havacılık mühendisi Theodore Von Karhan Transilvanya'da yaşadığı yıllarda sabit kayaların akan su üzerinde girdaplar oluşturduğu ve bu girdapların aralarındaki mesafenin sabit olduğunu gözetlemiştir. Daha sonraki hayatında, rüzgarda bir bayrak dalgalanması sırasında bu dalganın dalga boyunun

rüzgar hızından bağımsız yalnızca bayrak direğinin çapına bağımlı olduğunu gözlemlemiştir. Bu teori girdap saçmalı akış ölçerlerin temelini oluşturmuştur.

Tarihsel olarak, akış ölçümü ilk olarak 1500'lü yıllarda Mısırda ilk olarak gerçekleştirilmiştir [4]. Fakat ilk akış ölçerler 1700'lü yılların sonlarında geliştirildiği bilinmektedir. Bu ilk akış ölçerler türbin tipli ve pozitif yer değiştirme esaslı akış ölçerler gibi mekanik yapıda olmuşlardır. Çizelge 2.1 farklı tipteki akış ölçerlerin bulunuş yıllarını özetlemektedir. Çizelgede görüldüğü gibi geçiş zamanı esaslı (GZE) ultrasonik tekniği yeni bir teknolojidir ve gelişimi devam etmektedir.

Çizelge 2.1. Farklı ölçüm esaslı akış ölçerlerin prensiplerinin ve endüstride ilk kullanılış tarihleri

Akış ölçer tipi	Temel Prensip	İlk Patent	İlk Endüstride Kullanımı
Türbin	Archimede (1500)	1790	1938
Pozitif Yer Değiştirme	1800	1815	1820
Basınç Farkı	Torrecelli Bernoulli Newton Venturi (1808)	1896	1909
Elektromanyetik	Faraday (1832)	1917	1952
GZE Ultrasonik	Galileo	1931	1980
Coriolis	Coriolis	1953	1980
Girdap Saçma	Da Vinci (1513) Strouhal (1878) Von Karman (1912)	1954	1958

2.3. Çeşitli Akış Ölçer Yapıları

Bu bölümde endüstride sıklıkla kullanılan akış ölçerler özetlenmiştir. Akış ölçme alanındaki yenilikler ile birlikte farklı ölçüm prensipleriyle üretilen akış ölçerler ideale yakın koşullarda %0.5 doğruluğa erişebilmektedirler ve ölçme bildikleri akış oranı aralıkları 1:100 oranlarına kadar çıkabilmektedir [5].

2.3.1. Basınç farkı esaslı akış ölçerler

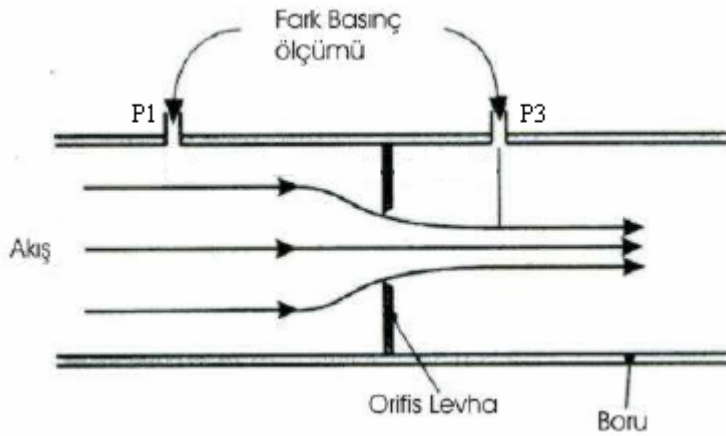
Bu tip akış ölçerler Bernoulli'nin enerji korunumu ilkesine dayanır. Basınç farkı esaslı akış ölçerler için akış oranı Eş. 2.4'deki gibi yazılabilir. Gazlar için yoğunluk, basınç ölçüm noktalarında farklı olacağı için harici bir düzeltme faktörüne daha ihtiyaç duyar.

$$q = K \sqrt{\frac{h_w}{\rho}} \quad (2.4)$$

Kalibrasyon faktörü (K) akış ölçere ait düzeltme katsayısıdır , h_w ölçülen basınç farkıdır. Orifice, venturi, dirsek tipi akış ölçerler ve pitot yapıları bu gruba girmektedir.

Orifis

Bir dar ağızlı plaka akış sınırlayıcı olarak bir tüp içerisine yerleştirilir. Tipik olarak dar ağızlı plaka akış tüpü ile eşmerkezlidir. Şekil 2.1 örnek bir orifis akış ölçeri göstermektedir. Daralan ağızdan dolayı akışkanın hızı bu noktada artış gösterir ve basınçta bir düşme yaratır. P1 ve P3 noktalarından basınç farkının ölçülmesi ile akış oranı hesaplanabilir. Geniş kullanım alanına sahiptirler fakat kirli suda tıkanma ve yüksek basınç kaybı oluşturmalar. Ayrıca dar ağızlı plaka yarattığı türbülanslar nedeniyle farklı basınç farkı ölçümlerine neden olabilmektedir. Düşük maliyetlidir.

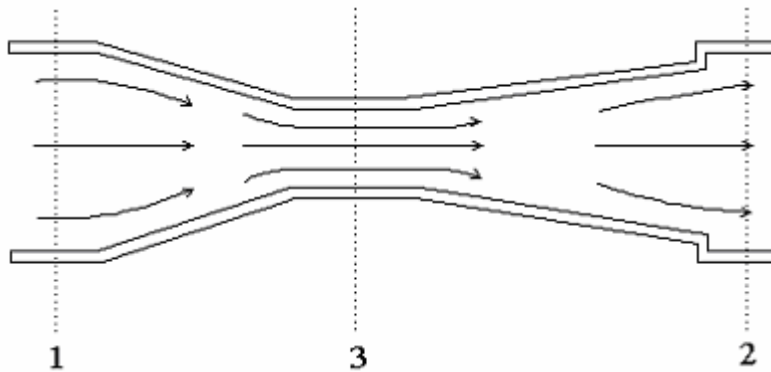


Şekil 2.1. Orifis akış ölçer

Delik çapının, tüp çapına oranı beta oranı olarak tanımlanır. Bir çok uygulamada bu oran 0.2 ile 0.75 arasındadır. Yüksek beta oranı aynı debide düşük beta oranında daha düşük diferansiyel basıncını oluşturur. 0.75'den büyük beta oranları doğruluğun çok önemli olmadığı yerlerde kullanılır [3].

Venturi

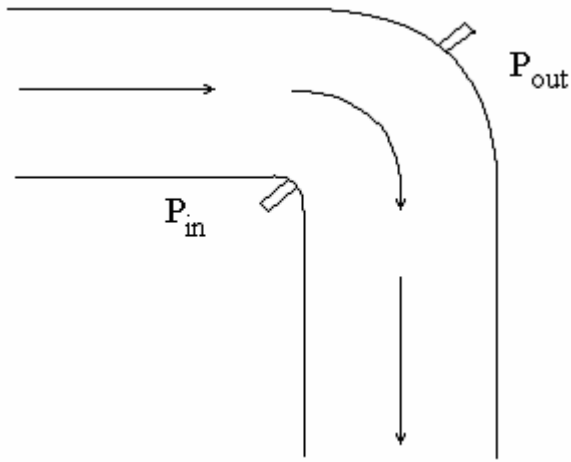
Şekil 2.2 venturi akış ölçeri göstermektedir. Venturi tüp içersindeki kesitlerdeki değişme bir basınç farkı doğurur ve bu basınç farkından akış oranı bulunabilir. Dar ağızlı plaka kullanımından daha pahalıdır buna rağmen basınç düşüşü daha kontrollüdür. Tıkanma olasılığı daha düşüktür.



Şekil 2.2. Venturi akış ölçer

Dirsek tipi akış ölçer

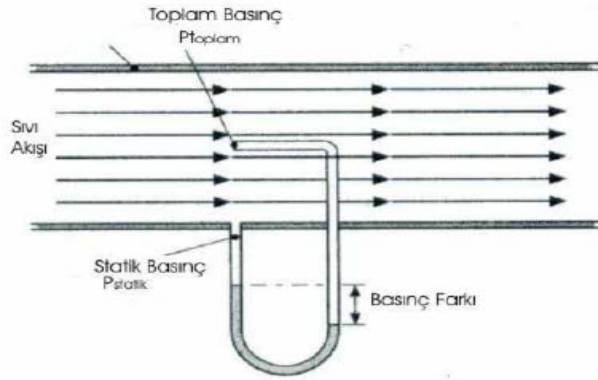
Akış dönüşleri yada bir dirsek yardımıyla bir basınç farkı oluşturulması esasına dayanır. Şekil 2.3 bu yapıdaki bir ölçüm mekanizmasını göstermektedir. Basınç farkının kaynağı burada merkez kaç kuvvetidir. Maliyeti çok düşüktür fakat bu yapıdaki akış ölçerlerin doğruluğu çok zayıf, tekrarlanabilirlikleri ise iyidir. Başlıca dezavantajları özellikle gazlar için çok düşük diferansiyel basınç oluşturmalarıdır.



Şekil 2.3. Dirsek tipi akış ölçer

Pitot tüpü

Şekil 2.4’de görünen pitot tüpü statik ve dinamik basınçları tüp içerisindeki bir noktadan ölçerler. Bu basınçlar arasındaki farktan akış oranı hesaplanır. Annubar adlı yapı bu pitot tüplerinin farklı sayıda, akışkanın hız profiline göre, tüp içerisine monte edilmiş halidir ve bu şekilde toplam akış birden fazla ölçümle hesaplanır. Çok düşük basınç kayıpları vardır, sadece temiz sularda kullanılabilir, maliyeti diğer diferansiyel basınç esaslı akış ölçerlere göre fazladır ve uygulaması zordur.



Şekil 2.4. Pitot tüpü

2.3.2. Diğer akış ölçüm prensipleri

Türbin akış ölçer

Bu tip akış ölçerlerde sıvı akışkan bir türbin, rotor yada herhangi bir pervane düzeneği üzerinden akar ve bu akış oranıyla orantılı bir dönme hareketi oluştur. Rotor hızı akış oranı ile birlikte doğrusal olarak artar. Böylece rotor turları ile akış oranı ölçülebilir. Kirli sularda kullanılamaz. Maliyeti yüksektir. Bunlara rağmen geniş bir ölçüm aralığında iyi doğrulukta çalışabilirler. Geniş ölçüm aralığına sahiptirler ve bu aralıkta rotor hızı ve akış hızı doğrusaldır. Özellikle düşük akış oranları için akışkanın hız profilinden, pervanelerdeki ve gövdedeki diğer sürtünmelerden etkilenmektedirler. En büyük problemi akışkan çok hızlı olduğunda yada akışkanın birden hatta girdiğinde paletlerde aşınmalara yol açabilmektedirler. Bu durum kalibrasyon kaymasına yol açabilmektedir. Sıvı ve gaz uygulamalarında tercih edilmektedirler, fakat yüksek yoğunluk farkından dolayı iki ayrı tasarım gerektirir [3].

Pozitif yer değiştirme

Bu tip akış ölçerlerde akışkan bağımsız hacimsel odacıklara ayrılmıştır ve akış ölçer içinden geçen bu elementlerin sayısı direkt olarak ölçülür. Geniş bir ölçüm aralığında yüksek doğruluk sağlarlar. Bununla birlikte yüksek basınç kayıpları vardır ve sıvı

içerisindeki tortular bu tür akış ölçerler için tehlikelidir. Ev tipi su, düşük basınçlı doğal gaz gibi uygulamalarda yaygın olarak kullanılır. Son yıllarda doğruluğun gerekli olduğu karışım uygulamalarında da sıklıkla kullanılmaktadır. Zaman esaslı olmadıkları için akış oranı hesaplamada nadiren kullanılır.

Girdap saçma esaslı akış ölçerler

Akış alanı içerisine yerleştirilen yapılar ile akışla beraber girdaplar oluşturulur. Bu girdapların frekansı akış oranı ile doğru orantılı olarak değişir. Üretilen girdaplar termistör vasıtası ile hissedilebilir. Tüp içerisinde ki hız ve girdap frekansı arasındaki ilişki doğrusal ve akış yoğunluğundan bağımsızdır. Çok geniş bir akış aralığında çalışırlar. Düzeltme katsayısı gazlarda ve sıvılarda aynıdır. Farklı tasarıma gerek yoktur bu özellik bu tip sayaçları çok popüler yapmıştır. Buhar, doğal gaz ve sıvılar için kullanılmaktadır. Kirlilikten etkilenmezler. Türbin tipli akış ölçerlerde olan yüksek hız sorunu yoktur. Doğruluk oranları %0.5 ile %1 arasındadır.

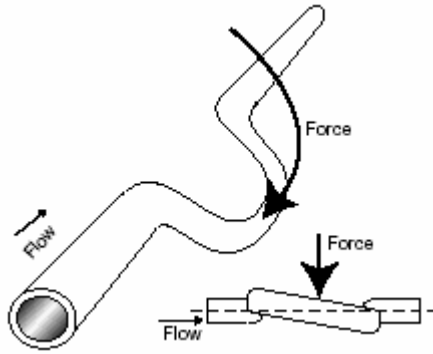
Manyetik akış ölçer

Manyetik indüksiyon esaslı manyetik akış ölçerler tüp alanı içerisinde akış engeli bulundurmeyen akış ölçerlerdir. İletken olmayan akış duvarı üzerine iki elektrot karşılıklı olarak akış yönüne ve manyetik alana dik olarak yerleştirilir. Ölçülen milivolt mertebesindeki sinyal akış oranıyla oransaldır ve akış hız profiline duyarlılığı azdır. Bu nedenle manyetik akış ölçerler iletken sıvılar için düzenli ve türbülanslı akışlar için idealdir. Basınç farkıyla çalışan akış ölçerlere zıt olarak akış hesabına akışkanın yoğunluğu girmez. Manyetik akış ölçerler çoğunlukla geniş bir ölçüm aralığında doğrusaldır ve doğrulukları %0.5 ile %1 arasındadır. Düşük basınç kaybına sahiptirler.

Coriolis akış ölçer

Bu tip akış ölçer akışı saptamak için akışın geçtiği tüp üzerindeki deformasyondan faydalanır. Şekil 2.5 coriolis akış ölçerin mantığını açıklamaktadır. Eğer bir akış

yoksa toka şeklindeki boru düz kalır ve tüp eksenini ile aynı düzlemedir. Eğer akış varsa bu boru coriolis etkisinden dolayı burkulacaktır. Bu burkulma akış oranıyla değişecektir ve bir osilasyon şeklinde olacaktır. Tüp deformasyonunu ölçmek zor fakat mümkündür. Bu tip bir akış ölçer ile yoğunluk ölçmekte mümkündür [6]. Sıcaklığın boru sertliğini ve boyutlarını etkilemesi durumunda sıcaklığın ölçülmesi doğruluk için önemlidir.



Şekil 2.5. Coriolis akış ölçer

Termal akış ölçer

Bu tip akış ölçerler akış oranını akışkanı ısıtarak ve bu ısıtılan akışkanın sıcaklığındaki değişimi kullanarak ölçerler. Akışkan sıcaklığı ısıtıcının önünden ve arkasından ayrı ayrı ölçülür. Doğruluğu arttırmak için iki ayrı ısıtma birimi kullanılabilir ve sıcaklık eş zamanlı ölçülür. Akış varsa ikinci birim daha çok sıcaklık ölçecektir. Doğrulukları uygulamaya göre %1 in çevresindedir. Genel olarak gazlarda kullanılır.

3. SES VE ULTRASES

3.1. Ses Dalgalarının Fiziği

Ses, ortamda basınç değişimiyle oluşan ve mekaniksel bir hareketle yayılan bir dalgadır [7]. Ses dalgaları bir frekansa sahiptir. İnsan kulağı 20Hz ile 20KHz arasındaki ses dalgalarını duyabilir. İnsan kulağının duyma sınırının üzerindeki akustik dalgalara ultrasonik dalgalar denir. Düşük frekanslı sesler ile aynı özelliklere sahiptirler. Tıp elektroniğinde, ölçme ve savunma sanayi gibi alanlarda genellikle transdüserler tarafından piezoelektrik olay ile üretilen 1MHz-30MHz frekansa sahip dalgalar kullanılır.

Megahertz mertebesindeki dalgalar radyo frekans dalgaları olarak adlandırılırlar fakat radyo frekans dalgaları ile ultrasonik dalgalar arsında bazı farklar vardır. Aralarındaki en önemli fark radyo frekans dalgaları elektro manyetik yapıda iken ultrasonik dalgaların akustik yapıda olmasıdır. Ses dalgaları salına bilen bir ortama ihtiyaç duyar bu nedenle vakumlu bir ortam üzerinden gönderimi yapılamaz [8]. Bu sistemi kullanan canlılar arasında yunus balıkları ve yarasalar en çok bilinenlerdir. Radar ve sonar cihazları bu canlılar örnek alınarak yapılmışlardır.

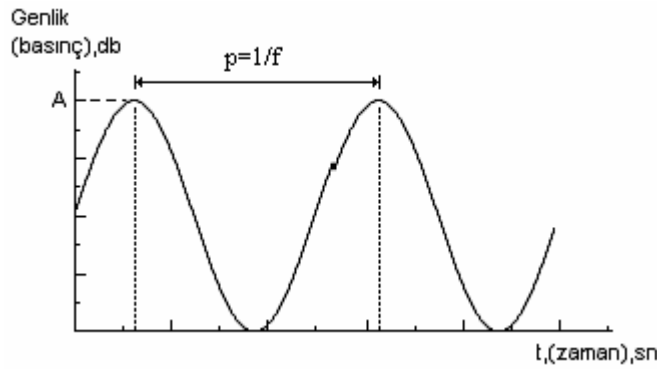
Ses gönderildiği zaman bu dalganın parametreleri hız, dalga boyu ve frekansıdır. Frekans ve dalga şekli, sese neden olan vibrasyonun dalga şekli ve frekansı ile belirlenir. Fakat dalga boyu ve hız ortama bağlıdır. Ses hızı ortamın yoğunluğuna ve elastiklik katsayısına bağlıdır ve katılarda çok hızlı sıvılarda daha yavaş gazlarda en yavaştır. Çizelge 3.1 bazı materyaller içerisindeki ses hızını vermektedir.

Çizelge 3.1 Bazı materyaller içerisinde sesin yayılım hızı

Ortam	Ses hızı (m/s)	Ortam	Ses hızı (m/s)
Hava	343	Deniz suyu	1510
Cam	4500	Hidrojen	1284
Helyum	965	Nitrojen	334

3.2. Ses Dalgalarının Özellikleri

Ses dalgalarının tekrarlanan bir seri basınç dalgalarından meydana geldiği ve yayıldığı ortamdaki molekülleri titreştirerek ilerlediği bilinmektedir. Ses dalgaları frekanslarına göre infra ses, işitilebilir ses ve ultrason olarak sınıflandırılırlar. İnfra ses 20Hz den küçük ses dalgaları 20Hz ile 20KHz arasındaki sesler işitilebilir ses 20KHz ile 30MHz arasındaki sesler ultrason olarak sınıflandırılabilir. Bütün ses ve elektromanyetik dalgalar Eş. 3.1 ve Eş. 3.2'ye uygun olarak yayılırlar [9].



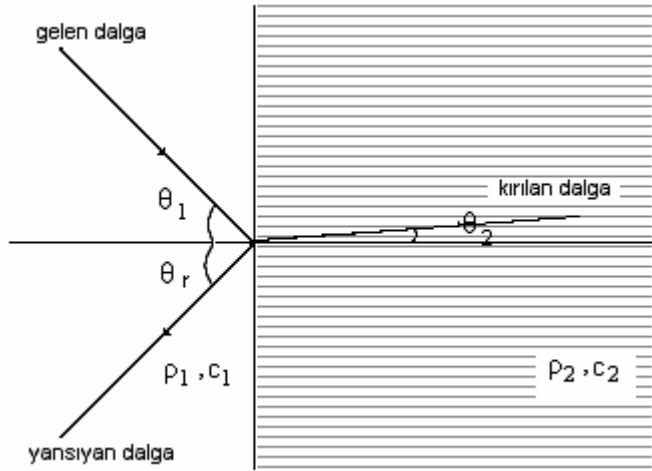
Şekil 3.1. Ses dalgasının genliği ve periyodu

$$c = f \cdot \lambda \quad (3.1)$$

$$p = \frac{1}{f} \quad (3.2)$$

Burada c sesin ortamdaki hızı, f frekans, λ dalga boyu, p periyottur. Optikteki ışığın kırılması ve yansıması akustikte de geçerlidir. Borular üzerine monte edilen (clamp-

on) ultrasonik akış ölçerler başta olmak üzere bir çok ultrasonik akış ölçer bu prensiplerden yararlanmaktadır.



Şekil 3.2. İki farklı ortam yüzeyinde kırılma ve yansıma

Yoğunlukları farklı bir ortam ile karşılaşan ultrasonik dalgalar kırılarak ikinci ortama geçmeleri, optikteki Snell yasasına uygun olarak gerçekleşir [9]. Şekil 3.2’de görülen θ_1 gelme açısı, θ_r yansıma açısı, θ_2 kırılma açısıdır, c_1 birinci ortamdaki hız c_2 ikinci ortamdaki hız, ρ_1 birinci ortam yoğunluğu, ρ_2 ikinci orta yoğunluğudur. Gelme açısı yansıma açısına eşittir.

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2} \quad (3.3)$$

Eş. 3.3’de görülebileceği gibi $c_1 < c_2$ ise $\theta_1 < \theta_2$ (az yoğun ortandan çok yoğun ortama geçiş) aynı şekilde $c_1 > c_2$ ise $\theta_1 > \theta_2$ ’dır.

Sesin ortamda yayılmasını etkileyen diğer bir parametre ortamın akustik empedansıdır. Akustik empedans ortamın ses dalgalarının iletimine karşı gösterdiği dirençtir. Birimi rayl’dır (MKS sistemine göre $1 \text{ rayl} = 1 \text{ kg/m}^2 \cdot \text{sn}$) . Eş. 3.4 ile ifade edilir.

$$z = g.c \quad (3.4)$$

Eşitlikte g , kg/m^3 cinsinden kütle ağırlığı iken c , m/s cinsinden ortamdaki ses hızıdır.

3.3. Ultrasonik Transdüserler

Hoparlörler ve mikrofonlar benzer çalışma prensiplerine sahip olsalar da ses dalgası alma ve yaratma arasındaki farklar bunları farklı amaçlarda kullanmayı mümkün kılmaz. Bir hoparlör bir mikrofon gibi kullanılabilir fakat duyarsız ve düşük kaliteli bir mikrofon olur. Bir mikrofon kulaklık olarak kullanılabilir fakat elverişli bir hoparlör olamaz çünkü üzerinde yüksek güç tutacak şekilde tasarlanmamıştır.

Ultrason için kullanılan transdüserler terslenebilir bir şekilde çalışabilmektedirler. Ultrasonik dalga yayınlamak ve almak için kullanılan ultrasonik transdüserler hem alıcı hem de gönderici olarak iki yönde de çalışabilir. Hava yada herhangi bir gaz üzerinden ultrasonik dalga gönderebilmek için transdüserler diyaframlarla ve onların çeperleri ile kullanılır.

Önemli ultrasonik transdüserler piezoelektrik yada manyetostriktiftir. Manyetostriktif manyetik materyallerin ölçütlerinin değişmesidir. Birçok nikel alaşımı manyetostriktiftir ve düşük frekanslı ultrasonik transdüserlerde sıklıkla kullanılır (30KHz-100KHz).

Manyetostriktif transdüserler manyetostriktif çekirdek sargısı içerir. Elektriksel dalga formu sargıya uygulanır. Sargı indüktansı yüksek olduğu için düşük ultrasonik frekanslarında çalışabilir. Yeterince yüksek bir uyarım akımı manyetostriktif malzemeyi vibrasyona sokacaktır. Bu vibrasyonda ortama ultrason olarak iletilir [8].

Piezoelektrik transdüserler daha geniş bir kullanım alanına sahip olsa da güç çıkışları manyetostriktif transdüserlerden daha düşüktür. Transdüserlerde kullanılan kristaller baryum titanat yada quartz kristalleridir. Bu kristaller verilen yönde maksimum

salınım yada duyarlılık sağlaması için hazırlanırlar. Kristaller elektriksel kontak sağlamak için zıt yüzleri metalle kaplanır. Empedansları yüksektir ve sinyal seviyeleri alıcı iken mili volt mertebesinde, gönderici iken birkaç volt mertebesinde.

3.3.1. Piezoelektrik olay

Sıkıştırıldığında, büküldüğünde yada biçimi bozulduğunda elektriksel şarj olan kristallere piezoelektrik kristal denir. Bu kristaller maruz kaldığı titreşimlerle senkron bir elektriksel titreşim oluşturur. Piezoelektrik kristallere elektriksel bir titreşim uygulanırsa kristalin hacmi değişir tepki olarak mekanik bir vibrasyon sergilerler ve çevresinde bir basınç farkı oluştururlar ki buda ultrason kaynağıdır [10]. Bu titreşimler kararlı ve değişmezdir. Baryum titanat, Kurşun titanat ve kurşun zirkonat piezoelektrik etki gösteren seramik materyallerdir.

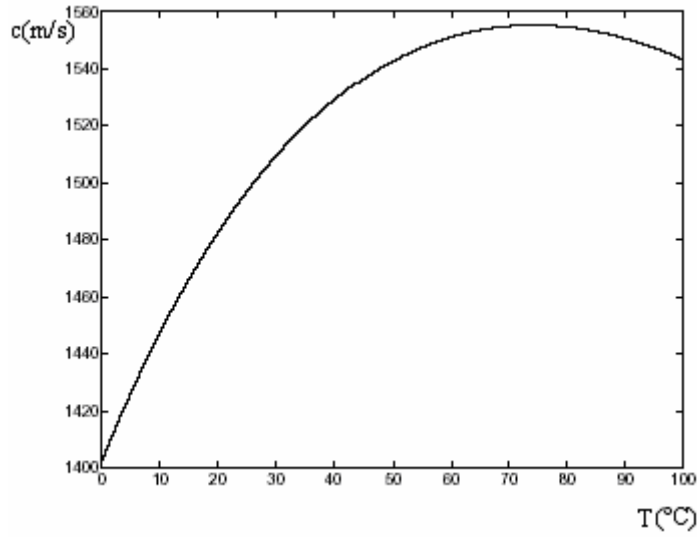
3.4. Ultrasonik Dalgaların Su İçerisinde Yayılma Hızları

Sesin ortamdaki yayılma hızı ortamın yoğunluğuna ve elastikiyetliğine bağlıdır. Eğer yayıldığı ortam su ise su sıcaklığı, sudaki tuz oranı iletim hızına etkiyen faktörlerdir. Ayrıca su basıncının iletim hızına küçükte olsa bir etkisi vardır. Eş. 3.5 su içerisindeki ultrasonik dalganın teorik olarak iletim hızını vermektedir [10].

$$c = \frac{1}{\sqrt{(\rho \cdot e)}} \quad (3.5)$$

Burada e ortamın elastiklik katsayısıdır.

Şekil 3.3. su içerisinde ultrasonik dalganın yayılma hızının sıcaklığa bağlı değişimini göstermektedir. Bu değişimin deneysel verilerle elde edilmiş polinomik biçimi Eş. 3.6'da verilmektedir.



Şekil 3.3. Su içerisinde ses-ötesi dalgaının yayılma hızının sıcaklığa bağlı değişimi

$$c = 1.4024 \cdot 10^3 + 5.0384 \cdot T - 5.8115 \cdot 10^{-2} \cdot T^2 + 3.3456 \cdot 10^{-4} \cdot T^3 - 1.4815 \cdot 10^{-6} \cdot T^4 + 3.1608 \cdot 10^{-9} \cdot T^5 \quad (3.6)$$

Burada “c” m/s cinsinden su içerisinde ses-ötesi dalga hızı, “T” °C cinsinden sıcaklıktır. Ultrasonik dalgaların sudaki maksimum hızı: 74 °C’de 1555 m/s’dir [10].

4. ULTRASONİK AKIŞ ÖLÇME

4.1. Ultrasonik Akış Ölçümüne Tarihsel Bakış

1940'lı yıllarda sonar, United States Navy tarafından geliştirilmiştir. Sonarlar deniz altı ve gemileri gözlemlemek için kullanılmıştır. Askeri sonar araştırmalarının sonunda bir akış alanı içerisindeki ortalama akışın bulunabileceği keşfedilmiştir. İlk akustik akış ölçerler 1940 ve 1950'lerin başında geliştirilmiştir. [11, 12]. Fakat bu uygulamalar endüstriyel olmamıştır. İlk doppler akış ölçer ise ilk kez 1954 de Kalmus tarafından geliştirilmiştir [9].

Çalışmalar daha büyük hacimlerdeki suyun kanallardan ve barajlardan daha doğru ölçümünü gerçekleştirmeye yönelmiştir. Department of Water Resources, U.S. Geological Survey, ve U.S. Army Corps of Engineers, Raytheon Manufacturing Company ile birlikte akustik akış ölçer üretmek için 1957 de bir araya gelmişlerdir. O zamanlarda tasarlanan akustik akış ölçerlerin genel tasarımları ve çalışmaları şimdiki tasarımlara benzerdi, ses bir akış alanı üzerinden gönderilir ve iletim zamanı ölçülürdü. 1960'ların ortasında Westinghouse Electric Corporation kanalların içine ve borular içerisine yerleştirmek amacıyla California Aqueduct ile birlikte akustik akış ölçer tasarlamaya başlamıştır.

Westinghouse'un akış ölçer tasarımına katkısı kristal gömülü transdüserin daha önceleri kullanılan akustik hidrofonların yerini alması olmuştur. Bu kristaller düşük frekanslı ses dalgalarına göre daha yüksek frekans yada ultrasonik dalgalar yayımlayabiliyor böylelikle sistem doğruluğunu arttırıyordu.

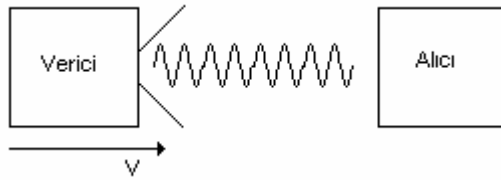
1960'larda ve 1970'lerde özellikle açık kanal ve büyük hacimli akışları ölçmek için çalışmalar devam etmiştir . Bu akış ölçerler 1950'lerdeki tekniklerle geliştirilmiştir.

4.2. Ultrasonik Akış Ölçme Yöntemleri

Ultrasonik akış ölçerlerde akış ölçümü için birkaç yöntem uygulanabilir. Doppler tekniği, geçiş zamanı tekniği, çapraz bağıntı, darbe tekrarlanma tekniği, faz kayması tekniği bu yöntemler arasındadır. Bu yöntemler arasında temel olarak geçiş zamanı, Doppler ve çapraz bağıntı yöntemi yaygın olarak kullanılmaktadır [13, 14]. Doppler ve çapraz bağıntı metodları benzer biçimdedir. Tipik olarak geçiş zamanı yönteminin Doppler yönteminden daha hatasız olduğu bilinmektedir. Endüstride geçiş zamanı tekniği daha baskın olarak kullanılmaktadır ve düşük maliyetli ve eklentiye ihtiyaç duymayan (partikül hava kabarcığı) tekniktir [2]. Bu çalışma geçiş zamanı yöntemi esas alınarak gerçekleştirilmiştir.

Ultrasonik akış ölçerlerin boyutları için bir sınır yoktur. Kurulum etkilerine karşı duyarlıdırlar. Çünkü akış tüm tüp üzerinden değil sadece transdüserler arasındaki kesit üzerinden ölçülür. Bozulan akış profiline karşı duyarlılık tüp ölçütleri arttıkça artar [15]. Ultrasonik akış ölçerler arasında dikkate değer farklar genellikle kullanılan ses yolu sayısıdır. Birden fazla ses yolu kullanan akış ölçerler kullandığı ses yollarından aldığı bilgilerin ortalamasını kullanırlar. Bu durum akış profiline olan bağımlılığı azaltmaktadır.

4.2.1. Doppler akış ölçer



Şekil 4.1. Doppler kayma prensibi

Doppler etkisi, bir ses vericisi ile alıcısı arasında Şekil 4.1'deki gibi göreceli bir hareket olduğunda meydana gelir. Eğer gönderilen frekans f_t , sesin hızı c ve bağlı hareketin hızı v_r ise bu durumda gözlemlenen f_r frekansı Eş. 4.1'den bulunabilir [16].

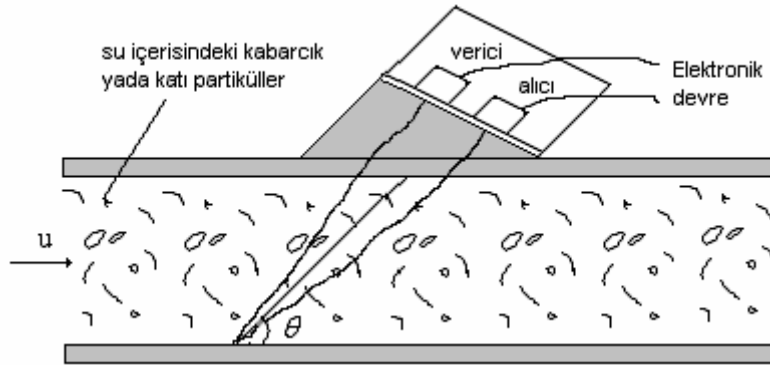
$$f_r = f_t \frac{(v_r + c)}{v_r} \quad (4.1)$$

Eğer alıcı ile verici birbirlerinden v_r göreceli hızı ile uzaklaşmakta ise Eş. 4.2'deki gibi yazılabilir.

$$f_r = f_t \frac{(v_r - c)}{v_r} \quad (4.2)$$

Bu etki genel olarak bir araba motorunun geçerken tonunda meydana gelen değişikliklerle algılanabilir. Eş. 4.1 ve Eş. 4.2 hem sabit bir verici hareketli alıcı hem de sabit alıcı hareketli verici durumunda uygulanabilir.

Doppler kayması elektromanyetik ışımda da meydana gelmektedir. Mikrodalga doppler etkisi radar hız tuzakları hızı ölçmekte ve görünür ışığın doppler kayması da diğer galaksilerin uzaklaşma hızını ölçmede kullanılır [16].



Şekil 4.2 Ultrasonik doppler akış ölçer

Bir doppler akış ölçer Şekil 4.2'de görüldüğü gibi hareket halindeki akışkana bir ultrasonik ses dalgası gönderir. Bu ses dalgasının küçük bir bölümü katı cisimler, buhar, hava kabarcıkları veya akıntı ve girdaplarından dolayı farklı yönlerde saçılırlar. Bu ses dalgaları akışkandan geçmekte iken, eğer akış var ise alınan ekoların frekanslarında, partikül yada hava baloncuklarının bağıl hızından dolayı kayma olacaktır. Burada alınan frekans Eş. 4.3'den bulunabilir.

$$f_r = f_t \frac{(c + u \cdot \cos \theta)}{(c - u \cdot \cos \theta)} \quad (4.3)$$

Burada u akışkanın ortalama hızını ve θ ise akışkanın akışına odaklanan ultrasonik demetinin açısını göstermektedir. Eğer frekans değişikliği Δf ile gösterilecek olursa $(f_r - f_t)$ ve $u \ll c$ iken Eş. 4.3, Eş. 4.4'deki gibi ifade edilir.

$$\Delta f = \frac{2 \cdot f_t \cdot u \cdot \cos \theta}{c} \quad (4.4)$$

Doppler ve çapraz ilişkili akış ölçerler hem darbeli dalga hem de sürekli dalga prensipleriyle çalışabilmektedirler. Sürekli dalga sistemde sabit frekanslı ultrason dalgası bir kristalden sürekli olarak gönderilir, komşu kristalden ise herhangi bir derinlikteki hareketli ortamdan yansiyarak gelen ve frekansı Δf kadar sapmış olan dalga alınır. Gönderme ve alma sürekli olduğu için çok yüksek akış hızları belirlenebilir. Darbeli sistemde ise ultrasonik darbe dizisi gönderilir ve belli bir gecikme ile yansıyan sinyaller alınır. Bu sinyaller alınır alınmaz diğer bir darbe dizisi gönderilir.

Sürekli dalga sisteminde gönderilen ve alınan frekanslar birbirlerine eklenir. Bu işlem $\Delta f / 2$ frekanslı bir sinüzoid sinyal oluşturmak üzere, bir alçak geçiren filtreye sokularak ortama akış hızı ile orantılı bir DC gerilim yada akıma çevrilir. Darbeli doppler metodunda doppler kaymasını hesaplamak daha zordur fakat bu tip akış ölçerler tek bir transdüserle yapılabilmektedir [17].

Doppler akış ölçerler her türlü akışkanlarda kullanılabilir. Paslandırıcı sıvılar veya yoğun tortu çamurları gibi zor sıvılarla kullanılmaya son derece uygundur. Bu akış ölçerler ayrıca düzenli ve çalkantılı akışlarda da çalışabilmektedir. Ancak basit akış ölçme uygulamaları için aşırı karmaşık ve pahalı bir aygıttır [16].

Doppler akış ölçerler sudaki küçük partiküllere ve kirliliğe güvenirler fakat bir çok temiz su uygulamasında kirli su uygulamalarında olduğu kadar iyi sonuçlar vermektedirler [3].

Bu metodun dezavantajı geçiş zamanı tekniğinde de olduğu gibi akışı ince bir hat üzerinden ölçülmesidir bu nedenle akış profili bilinmelidir. Ayrıca hız tüm kesit boyunca aynı olmayacağından doppler akış ölçerler lokal hız değişimlerinden de etkilenirler.

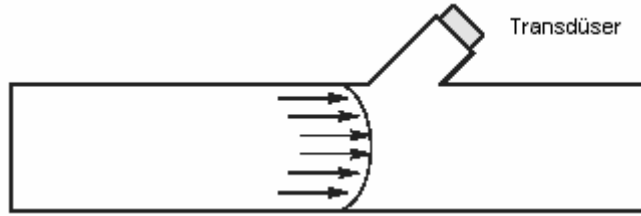
Bununla birlikte, hava baloncuklarının yada partiküllerin sensörün yüzeyi üzerine yığılması nedeniyle, alınan ekonun genliğinde sıvının bilinen akustik empedansının neden olduğundan daha fazla azalma olacaktır.

4.2.2. Çapraz ilişkili akış ölçer

Tortulu sıvıların ölçülmesinde bir çok zorluk vardır. Bunlar yoğunluktaki değişimler, sensörlerin tıkanması gibi hususlardır. Ayrıca akış içerisinde bulunan tortular, ses dalgasında saçılmalara neden olacaktır. Bu durumda ultrasonik ile zaman gecikmesini hesaplamakta sorunlar olacaktır. Bu nedenle geçiş zamanı esaslı akış ölçerlerin içerisinde katı partiküller yada hava kabarcıkları bulunduran sıvılarda kullanımı performansında düşüşe neden olacaktır yinede bu teknikler bazı tortulu su uygulamalarında kullanılmaktadır [17].

İşte çapraz ilişkili akış ölçerler, akış ölçülmesinde sıvı içinde asılı bulunan katı parçacıkların özelliklerini kullanmak üzere tasarlanmıştır. Şekil 4.3’de tortulu sıvı ortalama u hızı ile bir boru içinden soldan sağa doğru akmaktadır. Birbirinden l uzaklığında konumlandırılmış iki adet ultrasonik ışın demeti boru boyunca geçirilir. Burada sürekli yada darbeli dalga prensibi kullanılabilir. Çalkantılı akışta yada tortulu sıvılarda, katı madde parçacıklarının tortulu akışkan içindeki rasgele dağılımından dolayı sesin fazı, frekansı ve genliğinde rasgele değişiklikler meydana gelecektir. Bu durum alınan sinyaller de rasgele karakteristikte olacaktır. Eğer burada

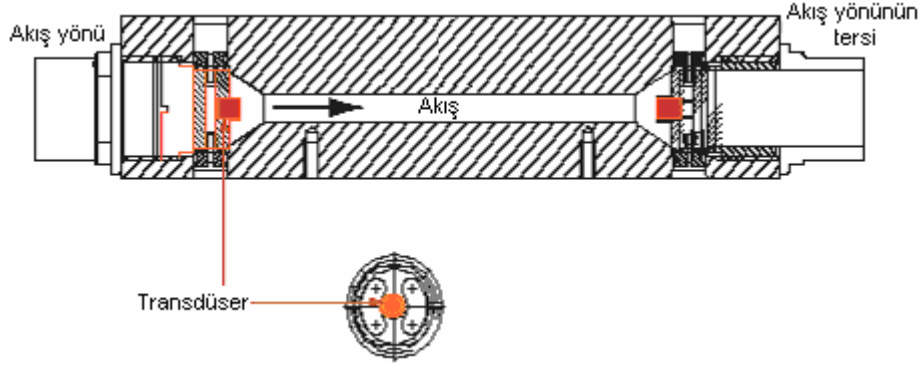
zaman aralığı boyunca devam edecektir ve akışkanın yerdeğiřtirmesi bu süre ile bağıntılı olacaktır.



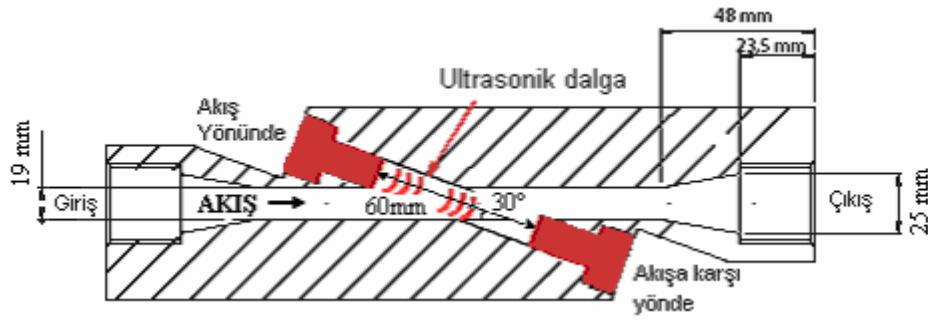
Şekil 4.4. Alternatif ultrasonik çapraz ilişkili akış ölçer

4.2.3. Geçiş zamanı esaslı ultrasonik akış ölçer

Bu metodun patenti ilk kez Rutten tarafından 1931’de alınmıştır [15]. Şekil 4.5 ve Şekil 4.6 ultrasonik akış ölçerlerin iki uygulamasını göstermektedir. Boylamsal veya eksensel konfigürasyonda ses yolu akış düzlemine paraleldir. Çapraz biçimli konfigürasyonda ultrason yolu ile akış düzlem arasında bir açı mevcuttur. Eksensel biçim 25mm den küçük çaplarda sıklıkla kullanılmaktadır. Çapraz biçimli konfigürasyon birkaç milimetreden 10 metreye kadar olan geniş çaplarda kullanılmaktadır. Çapraz biçimli uygulamalar eksensel uygulamalardan daha az endüstriyeldir. Akış hızını daha iyi tanımlamak için, herhangi bir ultrasonik akış ölçer bir transdüser çiftinden daha fazla transdüserle sahip olabilirler ve reflektör kullanabilirler. Transdüserler genel olarak 1MHz den daha yüksek frekanslı ve piezo tabanlı olarak seçilirler. Doğrulukları genellikle %2 veya daha iyidir. Yinede bu tasarıma göre değişir. Bütün akış ölçerlerde olduğu gibi bu doğruluk akış oranına göre değişmektedir [2].



Şekil 4.5. Örnek bir boylamsal biçimli tüp yapısı



Şekil 4.6. Örnek bir çapraz biçimli tüp yapısı

Burada bir yüksek frekanslı basınç dalgası tüp içine yayımlanır. Dalga'nın karşı transdüsera ulaşması için gerekli zaman sesin ortam içindeki hızına ve ortamın hızına bağlıdır. Akış oranı bilgisi ölçülen zamandan elde edilir.

Bazı kaynaklarda geçiş zamanı ultrasonik akış ölçerler, uçuş zamanı akış ölçerler şeklinde anılmaktadır.

Genel olarak ses dalgasının sudaki partiküller tarafından kesilmediği veya dalga şiddetinin zayıflamadığı temiz sularda kullanılırlar. Yinede bazı üreticiler kirli sularda daha başarılı olduklarını iddia etmektedirler. Özel detaylı laboratuvar kalibrasyon bilgileri üreticiler ve bağımsız laboratuvarlar tarafından geniş çapta paylaşılmamıştır [3].

Çalışmada basit yapısı nedeniyle tek transdüser çiftine sahip çapraz biçimli akış tüpü tercih edilmiştir. Tüm deneysel ve teorik çalışmalar bu akış ölçer üzerinde uygulanmıştır.

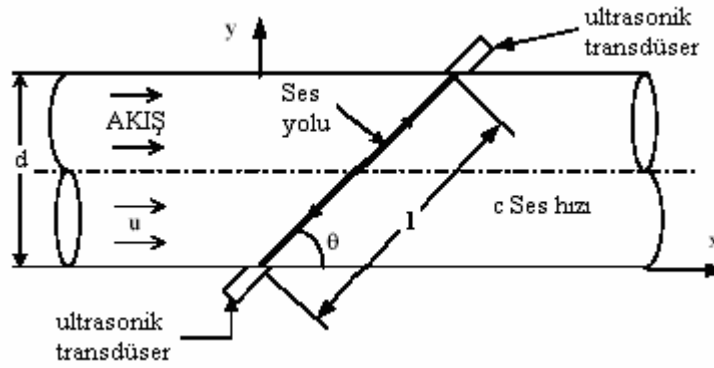
Temel geçiş zamanı prensibi

Bölüm 3.1’de ayrıntılı olarak değinildiği üzere ses dalgasının yayılması ortamdaki titreşim hareketi şeklindedir. Atomlar ve moleküller normal pozisyonlarını ses dalgasıyla birlikte değiştirirler. Bunun anlamı eğer ortam hareket ediyor ise ses de onunla birlikte hareket edecektir. Böylece, akış yönünde hareket eden ses akışın tersi yönde hareket eden sestten daha yavaş olacaktır. Bu ultrasonik geçiş zamanı tekniğinin temelidir [17].

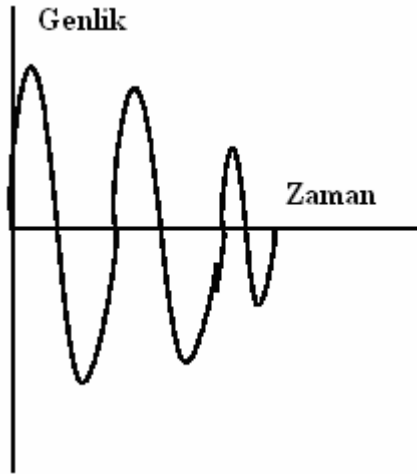
Akış yönünde gönderilen ses darbelerinin iletim süresi “Akış yönünde geçiş zamanı”, akış yönünün tersi yönde gönderilen ses darbelerinin iletim süresi ise “Ters akış yönünde geçiş zamanı” olarak adlandırılır. “Akış yönünde geçiş zamanı” ile “Ters akış yönünde geçiş zamanı” arasındaki fark akışkanın debisinin bir göstergesidir. Eğer akışkanın debisi artarsa, geçiş zamanları arasındaki fark artar, akışkanın debisi azalır ise aradaki fark azalır.

Ölçüm, karşılıklı monte edilmiş transdüserler üzerinden sırası ile ses darbelerin gönderilmesiyle yapılır. Her transdüser hem alıcı hem de verici olarak çalışabilir. Bir transdüser alıcı olarak beklerken diğer transdüser bir ses darbesi gönderecektir. Akustik dalgaının bu şekilde ulaşım süresi kaydedilir. Transdüserlerin rolleri değiştirilir ve olay tekrarlanır böylelikle ters yönde zaman ölçülür. Eğer akış durağan ise ölçülen iki zaman aynı olacak ve sensör mesafesinin ortamdaki ses hızı c ’ye oranına eşit olacaktır. Eğer akış var ise akış yönünde ölçülen süre akışın tersi yönde ölçülenden daha kısa olacaktır. Bu şekilde ortalama akış hızı Eş. 4.12 ile hesaplanabilir. Ortalama hız, hız ölçümü yapılan bölgenin kesiti ile çarpılarak akış oranı bulunabilir.

Şekil 4.7 tipik ultrasonik akış ölçer şemasını gösteriyor. İki ultrasonik transdüser akış eksenine verilen θ açısıyla birbirlerine bakacak şekilde akış kanalına yada boruya yerleştirilir. Sensörler arasındaki düz çizgi akış yolu olarak adlandırılır. θ açısı boşluğa yada doğruluk gereksinimlerine göre 30^0 ve 65^0 arasında seçilir. Gönderilen ses paketi Şekil 4.8’ye benzerdir [18].



Şekil 4.7. Ultrasonik akış ölçer şeması



Şekil 4.8. Tipik ultrasonik dalga darbesi

Eğer ses akış yönünde gönderiliyor ise, ortamdaki ses hızı c , ortalama akış hızı u , mesafe l , akış eksenine ile ses yolu arasındaki açı θ iken iletim zamanı t_1 Eş. 4.6’dan bulunabilir [17].

$$t_1 = \frac{l}{c + u \cdot \cos \theta} \quad (4.6)$$

aynı şekilde akış yönünün tersi yönde yol alan sesin ulaşım süresi t_2 Eş. 3.7'den bulunabilir [17].

$$t_2 = \frac{l}{c - u \cdot \cos \theta} \quad (4.7)$$

Eş. 4.6 ve Eş. 4.7'den ;

$$c = \frac{l - u \cdot t_1 \cdot \cos \theta}{t_1} = \frac{l + u \cdot t_2 \cdot \cos \theta}{t_2} \quad (4.8)$$

$$l \cdot t_2 - u \cdot t_1 \cdot t_2 \cdot \cos \theta = l \cdot t_1 + u \cdot t_2 \cdot t_1 \cdot \cos \theta \quad (4.9)$$

$$l(t_2 - t_1) = 2 \cdot u \cdot t_2 \cdot t_1 \cdot \cos \theta \quad (4.10)$$

$$u = \frac{l(t_2 - t_1)}{2 \cdot \cos \theta \cdot t_2 \cdot t_1} \quad (4.11)$$

$$u = \frac{l}{2 \cdot \cos \theta} \left(\frac{1}{t_1} - \frac{1}{t_2} \right) \quad (4.12)$$

Düzeltilme faktörleri

Tasarlanan akış ölçer de elektronik aksamdaki gecikmeler, karakteristik farklılıklar ve akışkan sıcaklığı etkilerinden kurtulmak için iki adet düzeltme faktörü kullanılmıştır. Birincisi sayaç üzerinde kullanılan transdüserler ve elektronik bileşenlerin karakteristik farklılıklarından dolayı sıfır akış anında ölçülen t_1 ve t_2 zamanları arasında bir fark olması muhtemeldir. Bu fark sıfır akış anında akış ölçerin bir akış değeri ölçmemesi için dengelenmelidir. Akış yok iken ölçülen zamanlar arasındaki fark sıfır kalibrasyon gecikmesi (SKG) olarak adlandırılır.

$$SKG = t_1^0 - t_2^0 \quad (4.13)$$

Burada SKG sıfır kalibrasyon gecikmesi, t_1^0 sıfır akış anında ölçülen t_1 değeri, t_2^0 sıfır akış anında ölçülen t_2 değeridir.

t_1 ve t_2 zamanlarının aritmetik ortalaması akışkanın debisinden bağımsızdır. Akış oranının değişimi zamanlardan birini artırma eğiliminde iken diğerini azaltma eğiliminde olacağından, akış oranı ne kadar değişirse değişsin bu aritmetik ortalama büyük ölçüde sabit kalmaktadır. Buna karşılık aritmetik ortalama t_1 ve t_2 'yi aynı yönde değiştiren etmenlere karşı duyarlıdır. Bu özellikten faydalanarak ikinci düzeltme faktörü, ses hızının gecikmesi (SHG) tanımlanmıştır. Elektronik bileşenlerde zaman içinde oluşacak karakteristik değişimlerini ve akışkan sıcaklığının değişiminden oluşacak hataları önlemek için kullanılmaktadır. Bu parametre, ölçüm esnasında t_1 ve t_2 zamanlarının aritmetik ortalamasının, kalibrasyon esnasında alınan aritmetik ortalamaya farkıdır.

$$A_{cal} = \frac{t_{1cal} + t_{2cal}}{2} \quad (4.14)$$

$$SHG = \frac{t_1 + t_2}{2} - A_{cal} \quad (4.15)$$

Burada A_{cal} kalibrasyon esnasında zamanların aritmetik ortalaması, SHG ses hızının gecikmesidir. Eş. 4.12'ye düzeltme faktörleri uygulandığında Eş. 4.16 düzeltme faktörleri uygulanmış olarak elde edilir.

$$u = \frac{l}{2 \cdot \cos \theta} \left(\frac{1}{t_1 - SHG - SKG} - \frac{1}{t_2 - SHG} \right) \quad (4.16)$$

Eş. 1.2'den ortalama akış hızı ile kesitin çarpılması ile akış oranı bilgisi elde edilir.

$$q = a \frac{l}{2.Cos\theta} \left(\frac{1}{t_1 - SHG - SKG} - \frac{1}{t_2 - SHG} \right) \quad (4.17)$$

Kalibrasyon

Eş. 4.17'deki gibi hesaplanan akış oranı bilgisi akış ölçer akış yolu uzunluğu, kurulum etkileri, akış profili ve bazı özel detayların etkilerini telafi için kalibre edilir. Bu faktör genelde akış tüpüne, kurulum etkilerine ve akışkanın hız profiline bağlı olarak reynolds sayılarının yada ölçülen debinin bir fonksiyonudur. Tasarımda ölçülen akış oranı bilgilerine göre kalibrasyon gerçekleştirilmiştir. Şekil 4.11'de çizilen eğride K faktörü ölçülen akış oranlarının bir fonksiyonu olarak gösterilmiştir. K faktörü Eş. 4.18'den bulunabilmektedir.

$$K = \frac{q_{gerçek}}{q_{ölçülen}} \quad (4.18)$$

Reynolds sayısı akışın ne kadar çalkantılı olduğunu yada ne kadar düzenli olduğunu gösterir. Şekil 4.9 ve Şekil 4.10'da görüldüğü üzere düzenli akışlarda (düşük reynolds sayılarında) hız profili paraboliktir. Akışkan partiküllerinin hızı akış ölçerin merkez ekseninde, tüp duvarının yanındaki partiküllerin hızından daha büyüktür. Akış çalkantılı olduğunda akış profili düzdür ve bütün akış ekseninde eşittir. Reynolds sayıları Re sembolü ile gösterilir, bir oran bilgisi olduğu için birimi yoktur.

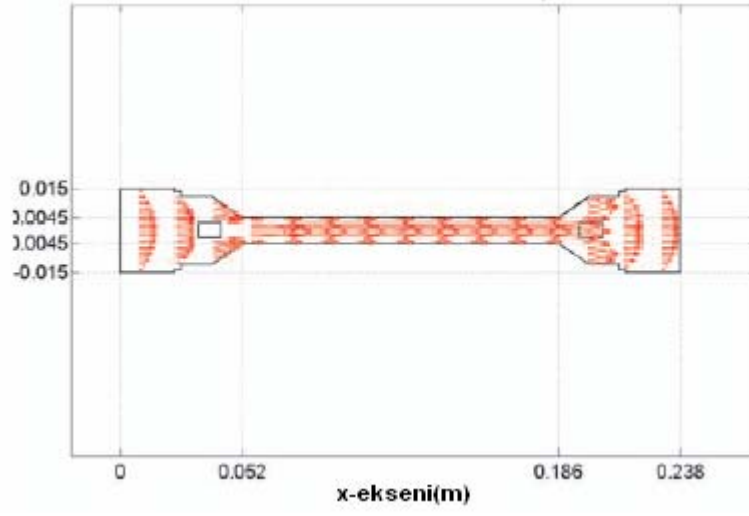
$$Re = \frac{u.d.\rho}{n} \quad (4.19)$$

Burada Re reynolds sayısı, d akışkan tüpün çapı, u ortalama akış hızı, ρ yoğunluk, n ise akışkanın viskozitesidir. Birçok kaynakta n/ρ oranı kinematik viskozite olarak bu hesaplamada kullanılır. Çizelge 4.1 suyun kinematik ve dinamik viskozitelerinin sıcaklıkla değişimini göstermektedir.

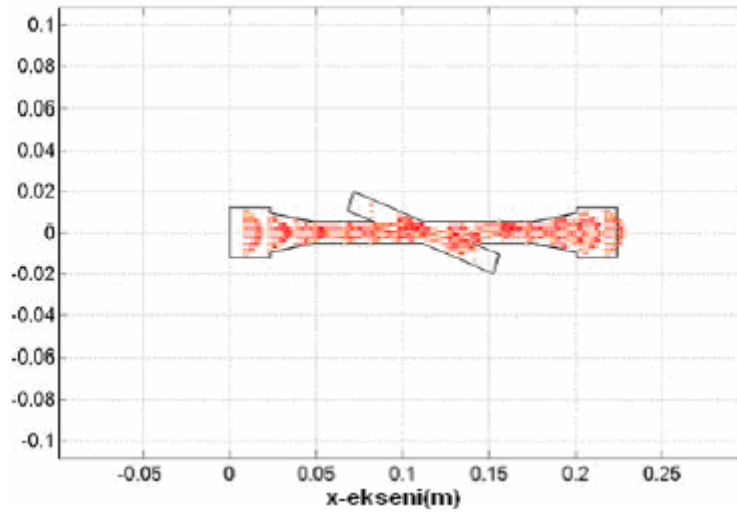
Çizelge 4.1. Suyun kinematik ve dinamik viskozitelerinin sıcaklıkla değişimi

Sıcaklık	Dinamik viskozite	Kinematik viskozite
-T-	- n -	- n/ρ -
(°C)	$10^{-3} (N.s/m^2)$	$10^{-6} (m^2/s)$
0	1.787	1.787
5	1.519	1.519
10	1.307	1.307
20	1.002	1.004
30	0.798	0.801
40	0.653	0.658
50	0.547	0.553
60	0.467	0.475
70	0.404	0.413
80	0.355	0.365
90	0.315	0.326
100	0.282	0.294

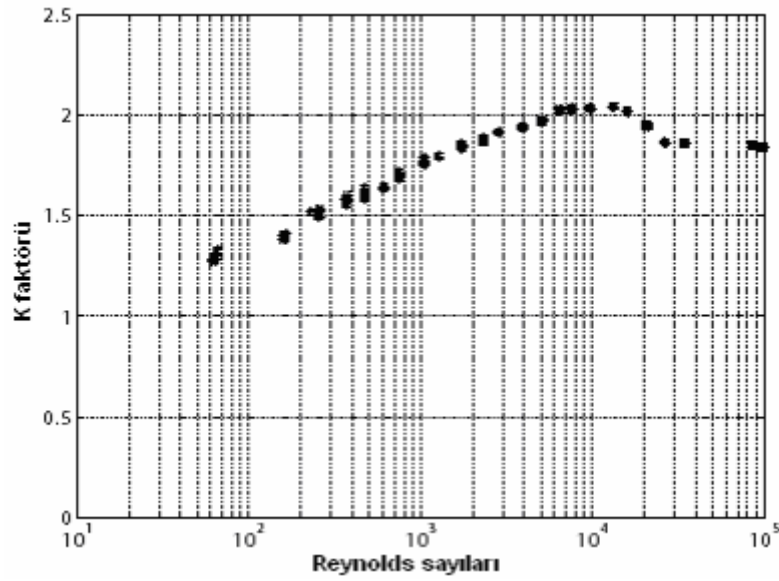
Akış ölçer türbülanslı akışlar ve düzenli akışlar arasında bir geçişe uğrar. Burada kalibrasyon eğrisi bir kambur oluşturur (Şekil 4.11, Şekil 4.12). Şekil 4.11 ve Şekil 4.12 iki çapraz biçimli tüpe ait reynolds sayılarının bir fonksiyonu şeklinde oluşturulmuş kalibrasyon eğrilerini göstermektedir [2]. İyileştirilmiş tasarımda kalibrasyon eğrisi daha az kambur oluşturmuştur. Çap daraltmak geçiş bölgesini daha da düzleştirecektir [2]. Bu kambur mümkün olduğunca önlenmelidir, çünkü bu bölgede ölçülen akış oranı hatalı olacak ayrıca seçilen yanlış K faktörü bu hatayı büyütecektir. Şekil 4.12 ve Şekil 4.13’de görüldüğü gibi kamburu azaltmak performansı daha iyi yapacaktır. Akış çalkantılı olduğunda akış profili düzdür ve bütün akış ekseninde eşittir. Böylece K faktörleri çalkantılı akışta neredeyse aynı olacaklardır [2].



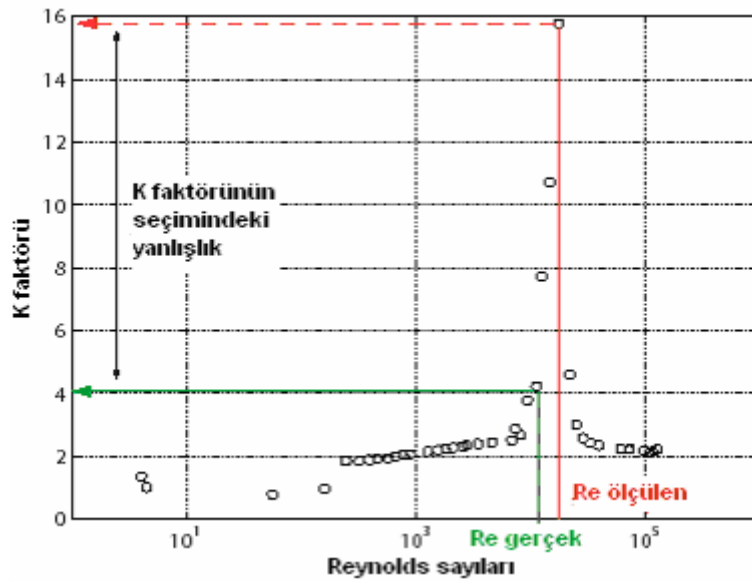
Şekil 4.9. Boylamsal tüp içerisinde hız profilinin simülasyonu (Akış ölçer girişinde akış bozucu yok)



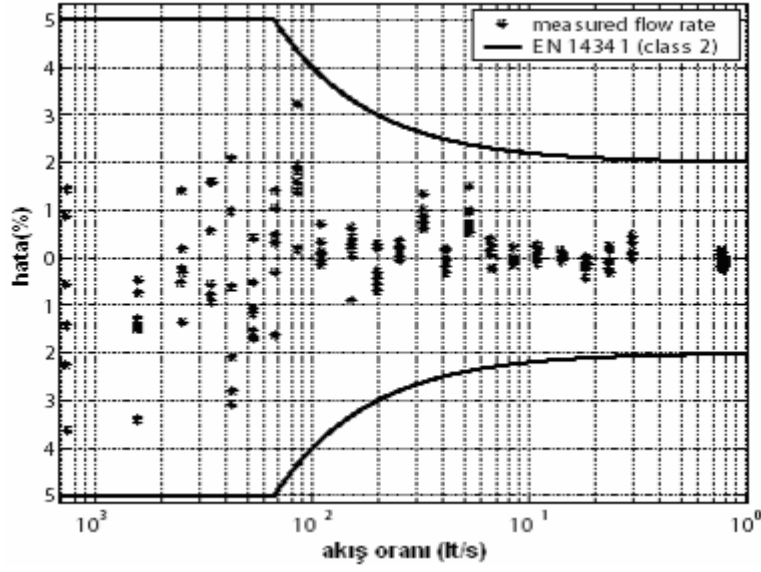
Şekil 4.10. Çapraz biçim tüp içerisinde hız profilinin simülasyonu (Akış ölçer girişinde akış bozucu yok)



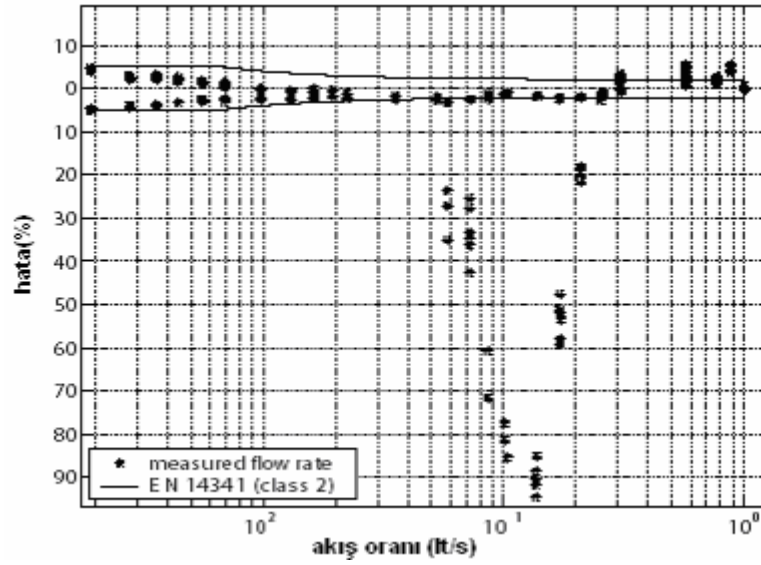
Şekil 4.11. İyileştirilmiş çapraz biçimli tüpten 18 °C derecede elde edilmiş kalibrasyon eğrisi



Şekil 4.12. İyileştirme yapılmamış bir çapraz biçimli tüpten 20 °C derecede elde edilmiş kalibrasyon eğrisi



Şekil 4.13. İyileştirilmiş çapraz biçim tüpün hata dağılımı



Şekil 4.14. İyileştirilme yapılmamış çapraz biçimli tüpün hata dağılımı

Son olarak kalibre edilmiş akış oranı bilgisi Eş. 4.20 aracılığıyla bulunabilir.

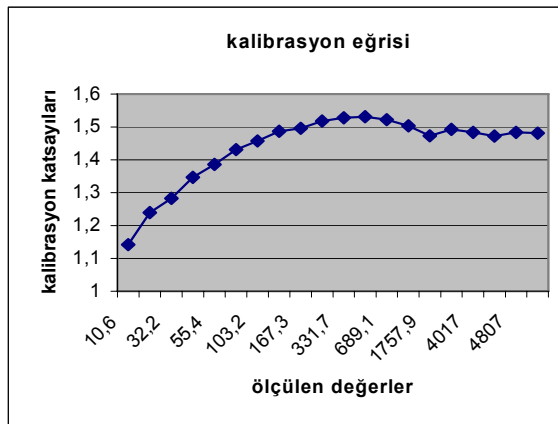
$$q = K.a \frac{l}{2.Cos\theta} \left(\frac{1}{t_1 - SHG - SKG} - \frac{1}{t_2 - SHG} \right) \quad (4.20)$$

Gömülü sistemde kalibrasyon eğrisinin oluşturulması

Akış ölçerin kalibrasyonu için Çizelge 4.2’de verilen yirmi ayrı ölçüm sonuçları kullanılmıştır. Akış ölçerin kalibrasyonsuz ölçülen debi bilgisine karşılık bu noktaların gerçek debi değerleri ve K kalibrasyon katsayısı verilmiştir.

Çizelge 4.2. Örnek kalibrasyon tablosu

GERÇEK(l/h)	ÖLÇÜLEN(l/h)	K
12,1	10,6	1,1415094
30	24,2	1,2396694
41,3	32,2	1,2826087
50,1	37,2	1,3467742
76,8	55,4	1,3862816
100,2	70	1,4314286
150,4	103,2	1,4573643
208,2	140	1,4871429
250,3	167,3	1,4961148
363,3	239,3	1,518178
506,8	331,7	1,5278866
727,3	475	1,5311579
1048,7	689,1	1,5218401
1597	1062,4	1,5032003
2590	1757,9	1,4733489
4384	2937,9	1,4922223
5961	4017	1,4839432
6870	4667	1,4720377
7130	4807	1,4832536
8200	5535	1,4814815



Şekil 4.15. Kalibrasyon eğrisi

Şekil 4.15 ölçülen kalibrasyonsuz akış oranı değerlerine karşılık kalibrasyon katsayılarının (K faktörleri) değişimini gösteriyor. Bu egride K faktörleri ölçülen debi değerlerinin bir fonksiyonudur. Bu fonksiyon tanımlanan her iki kalibrasyon faktörü arasındaki değişim lineer kabul edilerek kurulmuştur. Aşağıdaki denklemler bunu açıklamaktadır.

Eş. 4.21 X/Y düzleminde iki noktadan geçen ve y noktalarını x eksenine ait noktalara göre tanımlayan doğru denklemdir..

$$y = m(x - x_0) + y_0 \quad (4.21)$$

$$m = \frac{y_1 - y_0}{x_1 - x_0} \quad (4.22)$$

Burada m doğru eğimidir.

Çizelge 4.2'deki kalibrasyon tablosunun ilk iki noktası için kalibrasyon eğrisi aşağıdaki gibidir.

$$K_0 = \frac{q_{ölçülen0}}{q_{gerçek0}} = \frac{12,1}{10,6} = 1,141509 \quad (4.23)$$

$$K_1 = \frac{q_{ölçülen1}}{q_{gerçek1}} = \frac{30}{24,2} = 1,239669 \quad (4.24)$$

$d_{ölçülen0}$ ve $d_{ölçülen1}$ aralığı için katsayıyı tanımlayan doğru denklemi aşağıda verilen yöntemle elde edilir.

$$K = \frac{K_1 - K_0}{q_{ölçülen1} - q_{ölçülen0}} (q_{ölçülen} - q_{ölçülen0}) + K_0 \quad (4.25)$$

$$K = \frac{1,239669 - 1,141509}{24,2 - 10,6} (q_{ölçülen} - 10,6) + 1,141509 \quad (4.26)$$

$$K = 0,00722 \times q_{ölçülen} + 1,065 \quad (4.27)$$

Diğer ölçüm noktaları için de kalibrasyon eğrisi aynı şekilde hesaplanmış 20 doğrusal denklem elde edilmektedir.

Örnek olarak ölçülen değer 15,2 lt/h için tablodan dönen katsayı ve kalibreli akış oranı;

$$K = 0,00722 \times 15,2 + 1,065 = 1,174744$$

$$q_{gerçek} = K \times q_{ölçülen} = 1,174744 \times 15,2$$

$$q_{gerçek} = 17,85 \text{ lt / h}$$

5. TASARLANAN AKIŞ ÖLÇERİN BİLEŞENLERİ

Bu çalışmada, 3,6V D boyutlu lityum pil ile akış ölçerin pil ömrünün 10 yıl olması hedeflenmiştir. Bu nedenle devre üzerinde kullanılacak yongalar özellikle düşük güçlü ve çalışma gerilim aralıkları 2,7V-3,6V arasında seçilmişlerdir. Devre 3,6V'luk tek bir bataryadan besleneceğinden 5V gerilim gerektiren kısımlar için içerisinde sürücü transistör ihtiva eden ve düşük güç tüketimli torex 9111c dc/dc çevirici kullanıldı.

Düşük güç tüketimleri nedeniyle cam sıvı kristal display (LCD) ve Holtek HT16220 LCD sürücüsü kullanılmıştır.

Elektronik bir akış ölçerin önemli bir özelliği çok değişkenli olması ve bu değişkenlerin elektronik ortamda rahatlıkla okunulabilmesidir. M-bus (meter bus) arabirimi otomatik okuma için kullanılmıştır [19]. Tüketim bilgileri, geçmişe ait okuma bilgileri, akış ölçeri tanımlayan bilgiler, kalibrasyon tablosu ..vs gibi bilgilerin kalıcı olarak tutulması için 24LC128 seri eeprom kullanılmıştır.

Zaman ölçümü için ayrı tasarımlarda D-flow UfO A2 ve TDC GP2 kullanılmış ve bu çalışmada TDC GP2 üzerinde durulmuştur.

5.1. Mikrodenetleyici

Hedeflenen pil ömrü için nanowatt teknoloji adı altında üretilen PIC18f4520 kullanılmıştır. Bu Mikrodenetleyici 40 MHz'e kadar çalışma frekansına çıkabilmektedir. Mikrodenetleyici bu çalışma frekansında saniyede 10 milyon talimatı gerçekleştirebilmektedir. PIC18f4520 2V-5V arasında çalışmaya gerilimine sahiptir. 8X8 donanım çarpma modülü içermektedir. 32K program hafızasına ve 1536 byte rasgele erişimli hafızaya sahiptir. 36 genel amaçlı giriş/çıkış birimine sahiptir. "Zamanlayıcı1" birimine harici düşük güçlü 32,768 KHz kristal bağlanarak gerçek zamanlı saat olarak çalıştırılabilmektedir.

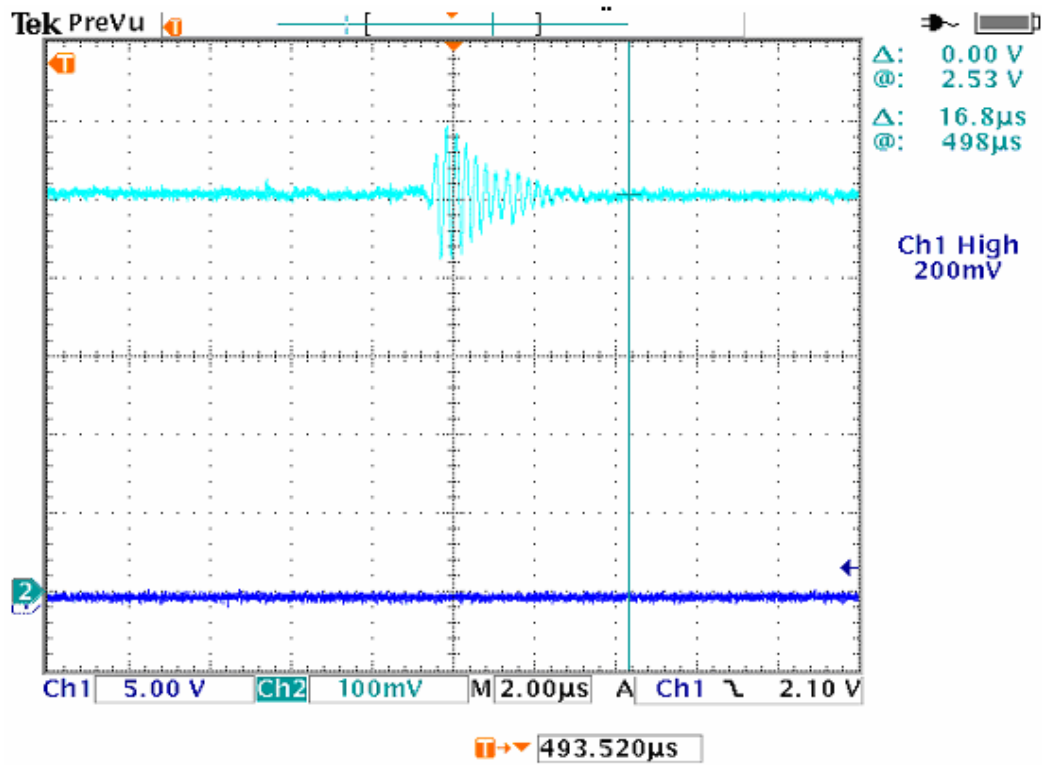
İki ayrı güç yönetim moduna sahiptir. Idle durumunda CPU kapalıdır, çevre birimler açık, çekilen akım tipik olarak 5.8 uA'ın altındadır. Uyuma durumunda ise CPU ve çevre birimler kapalıdır, çekilen akım tipik olarak 0.1 uA'ın altındadır. Ayrıca 2.1 uA akım tüketimi olan bekçi köpeği zamanlayıcıya sahiptir

5.2. Ultrasonik Transdüserler

Sistemde piezo tabanlı 4 MHz'lik transdüserler kullanılmıştır. Yüksek frekansta çalışması zaman çözünürlüğünü arttırmaktadır. Dış kaplaması termoplastik malzemeden yapılmıştır. Bu sayede her türlü çevre şartlarında çalışma performansını korumaktadır.



Resim 5.1. Kullanılan transdüserlerin dış görünüşü



Şekil 5.1. Kullanılan 4MHz transdüserin yankı tepkisi

5.3. Sayısal Zaman Çevirici

Sayısal zaman çeviriciler (TDC) nükleer araştırmalardan lazer uygulamalarına kadar bir çok alanda kesin zamanı yüksek çözünürlükte ölçmek için kullanılırlar. Bu çalışma kapsamında tasarlanan düşük güçlü ultrasonik akış ölçer prototipinde, ultrasonik akış ölçerler ve kalorimetreler için uygun olan genel amaçlı dijital zaman çevirici TDC GP2 denenmiş ve uygulama sonuçları gözlemlenmiştir.

TDC GP2 düşük akım tüketimi, darbe üretici, durmayı aktif etme, inen yada çıkan kenar tetikleme seçeneği, zaman aralıklarını maskeleye, saat kalibrasyonu gibi özellikleri ile esnek bir tasarım olanağı sunar. Ultrasonik akış ölçer ve kalori metre uygulamaları için çok uygundur.

TDC GP2 zaman aralığı ölçümünden başka ayrıca PT500/PT1000 gibi sıcaklık sensörlerini kullanarak 16bit gibi yüksek çözünürlükte, bir yada iki sensör üzerinden sıcaklık ölçebilmektedir. Giriş/çıkış birimleri 1.8V'dan 5V'a kadar, çekirdek gerilimi 1.8Vdan 3.6V'a kadar çalışabilmektedir. Bir adet RSTN (reset) girişi, bir adet INTN (kesme) çıkışı ve dört kablolu SPI ara yüzü ile toplam altı giriş/çıkış ile harici bir mikrokontrolör yardımıyla kontrol edilebilmektedir.

Birinci ölçüm aralığı ve ikinci ölçüm aralığı olmak üzere iki adet ölçüm aralığına sahiptir. Birinci ölçüm aralığı 0 ile 1.8 us aralığını, ikinci ölçüm aralığı ise 500ns ile 4 mili saniye ye kadar olan zaman aralıklarını ölçmektedir. Dolayısıyla ultrasonik akış ölçer uygulaması için uygun olan ikinci ölçüm aralığıdır.

İkinci ölçüm aralığında 50ps'ye kadar zaman ölçüm çözünürlüğü elde edilebilir. İki durdurma kanalından yalnız STOP1 kanalı kullanılabilir. Çoklu darbe ölçüm özelliği ile üç darbeye kadar darbe zaman aralıklarını ölçmeye izin verir. 10ns çözünürlükte zaman maskeleye kullanılabilir.

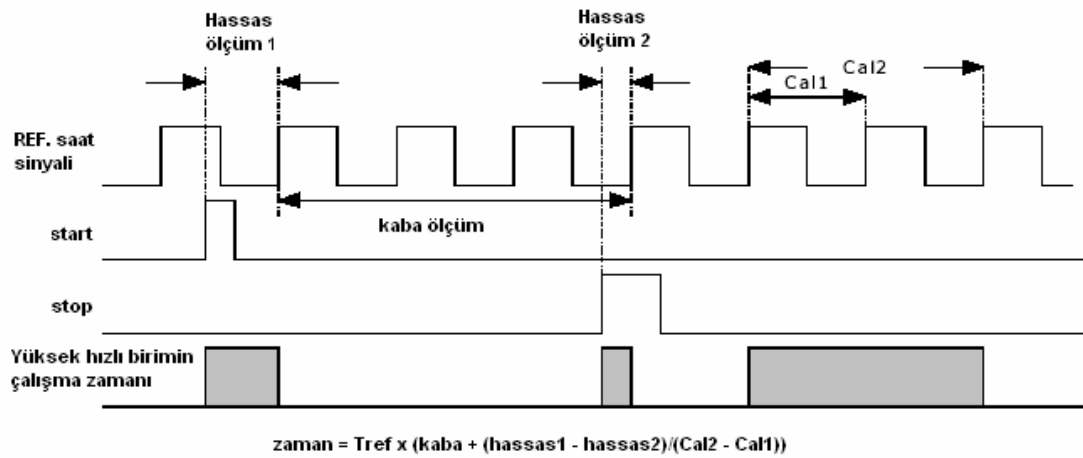
İkinci ölçüm aralığında TDC GP2 sadece kalibreli bilgiyi destekler. Zaman ölçümünün sonuncu referans saatlerin toplanmasıyla elde edilir. Kalibreli ölçüm

bilgisi 32bit sabit noktalı yapıda, okunabilir 32bitlik sonuç kayıtlarında depolanır. 32bitlik ölçüm bilgisinin yüksek anlamlı 16biti tamsayı, düşük anlamlı 16 biti ise kesirli yapıdadır. Her hangi bir ölçüm sonucu yalnızca tek bir sonuç kayıtlarında konumlanır. SPI ara yüzü ile bu bilginin okunması esnasında seri veri aktarımı yüksek anlamlı bittten (YAB) başlar ve en düşük anlamlı bit (DAB) ile sonlanır. Ölçüm zamanı Eş. 5.1 ile hesaplanır.

$$t = RESX.T_{ref}.2^{ClkHSDiv} \quad (5.1)$$

Burada RESX okunulan sonuç kayıtlarında içindeki bilgi, T_{ref} yüksek frekanslı saatin periyodu iken ClkHSDiv ise yüksek frekanslı saatin ön bölücü değeri olup 1, 2 yada 4 değerini alabilir.

Sayısal zaman çeviriciler sinyallerin kapılar üzerinden iç iletim süresindeki gecikmeleri kullanarak zaman aralıklarını yüksek kesinlikte ölçerler. İkinci ölçüm aralığında maksimum ölçüm aralığı bir ön bölücü kullanılarak arttırılabilir. Bu ölçüm aralığında yüksek hızlı birim tüm zaman aralığı boyunca çalışmaz. Bunun yerine sadece START ve STOP anlarında referans saatin yükselen kenarına kadar olan zamanı ölçerler. Bu nedenle ön bölücü kullanmak çözünürlüğü etkilemez. Şekil 5.2 ölçüm prensibini göstermektedir.



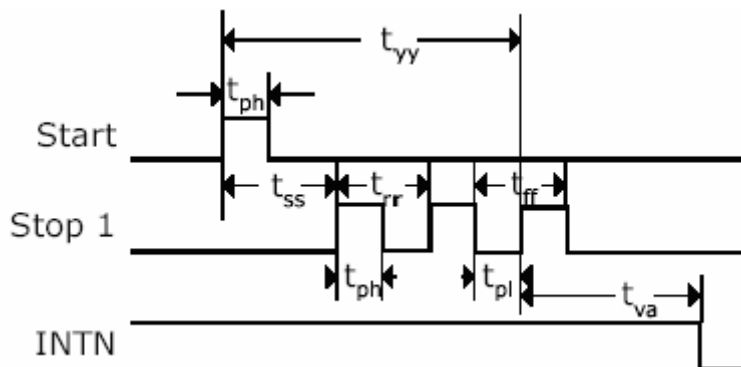
Şekil 5.2. Zaman ölçüm prensibi

Ölçüm hassas ve kaba ölçüm sonuçlarının toplamıdır. Hassas ölçüm kapı gecikmelerinden yararlanılarak elde edilir. Kapı gecikmeleri sıcaklığa ve gerilime büyük ölçüde bağımlıdır. Bu nedenle ikinci ölçüm aralığında kalibrasyon kullanmak zorunludur. Kalibrasyon boyunca TDC GP2, periyodu kesin olarak bilinen referans sinyalinin bir ve iki periyodunu hassas birim aracılığı ile ölçerek Cal1 ve Cal2 bilgilerini elde eder. Referans saat sinyalinin darbeleri kaba ölçüm olarak toplanır. Ölçüm aralığı kaba ölçümün boyutu ile sınırlıdır. Kaba ölçüm 14bittir. Bu nedenle 4Mhz yüksek frekanslı saat kullanıldığında ölçülebilecek en büyük zaman dilimi Eş. 5.2 yardımıyla bulunur.

$$250 \times 2^{14} = 4ms \quad (5.2)$$

“Hassas ölçüm 1” zamanı start darbesi ile referans saat sinyalinin ilk çıkan kenarı arasındaki süre, “Hassas ölçüm 2” zamanı STOP darbesi ile STOP darbesinden sonra gelen referans saat sinyalinin ilk çıkan kenarı arasındaki süredir. Bu süreler hassas birim ile ölçülür.

Ölçümde kullanılan darbeler ve zamanlama bilgileri Şekil 5.3 ve Çizelge 5.1’de verilmektedir. Ölçüm START darbesi ile başlar ve STOP1 kanalına gelen darbeler ile sonlanır.



Şekil 5.3. Kullanılan ölçüm darbeleri

Çizelge 5.1. Ölçümde kullanılan darbelerin zaman gereksinimleri

	Zaman [koşul]	Tanım
t_{ph}	2.5ns [minimum]	Minimum darbe genişliği
t_{pl}	2.5ns [minimum]	Minimum darbe genişliği
t_{ss}	$2 \cdot T_{ref}$ $12 \cdot T_{ref}$	START'tan STOP'a, Dis_PhaseNoise=1 START'tan STOP'a, Dis_PhaseNoise=0
t_{rr}	$2 \cdot T_{ref}$	Çıkan kenardan çıkan kenara
t_{ff}	$2 \cdot T_{ref}$	İnen kenardan inen kenara
t_{va}	4.6us [maksimum]	Aritmetik mantık birimi (ALU) data geçerli
t_{yy}	4ms [maksimum]	Maksimum ölçüm aralığı

Harici mikrokontrolör ile TDC GP2 arasındaki haberleşme ve kontrol SPI ara yüzü üzerinden, toplam sekiz adet operasyon kodu ile yapılır. Çizelge 5.2 operasyon kodlarını tanıtmaktadır.

Çizelge 5.2. TDC GP2 için operasyon kodları

8bit operasyon kodları								
YAB					DAB			Tanım
1	0	0	0	0	ADR2	ADR1	ADR0	Adrese yaz
1	0	1	1	0	ADR2	ADR1	ADR0	Adresten oku
0	1	1	1	0	0	0	0	Hazırla
0	1	0	1	0	0	0	0	Reset
0	0	0	0	0	0	0	1	Başlat
0	0	0	0	0	0	1	0	Sıcaklık Başlat
0	0	0	0	0	0	1	1	Rez. kalibrasyonu başlat
0	0	0	0	0	1	0	0	TDC kalibrasyonu başlat

5.3.1. Okunabilir ve yazılabilir kayıtçılar

Ölçüme başlamadan önce TDC GP2'nin çeşitli konfigürasyonlarının yapılması gerekir. Bu konfigürasyonlar için her biri 24bit olan altı adet yazılabilir kayıtçısı vardır. GP2 yi ve SPI ara yüzünü denetlemek için bir adet 8bit test kayıtçısı, ölçüm sonuçlarının saklandığı ve her biri 32bit olan dört adet sonuç kayıtçısı, TDC-GP2'nin durumu hakkında bilgiler içeren bir adet 16bit durum kayıtçısı olmak üzere toplam altı adet okunabilir kayıtçısı bulunmaktadır. Çizelge 5.3, Çizelge 5.4, Çizelge 5.5, Çizelge 5.6, Çizelge 5.7, Çizelge 5.8 yazılabilir kayıtçıları, Çizelge 5.9 okunulabilir kayıtçıları göstermekte ve özetlemektedir.

Çizelge 5.3. Yazılabilir Kayıtçı0

Bit	Kayıtçı0	Tanım
23	FIRE#	Darbe üretici tarafından üretilecek darbe sayıları
22		
21		
20		
19	DIV_FIRE	Darbe üretici saat sinyali için ön bölücü derleri.
18		
17		
16		
15	CALRES#	Seramik rezonatör kalibrasyonu için kullanılacak düşük frekanslı saat sinyalinin çevrim sayısı
14		
13	CLKHSDiv	Referans saat sinyali için ön bölücü değerleri.
12		
11	START_ClkHS	Yüksek hızlı saat sinyali seçenek bitleri.
10		
9	PORT#	Sıcaklık ölçümü için kullanılacak sensör sayısı seçme biti
8	Tcycle	Sıcaklık ölçümü için çevrim süresi seçme biti
7	FAKE#	Sıcaklık ölçümünün başında yapılacak sahte ölçüm sayısı
6	SelClkT	Sıcaklık ölçümü için referans saat sinyali seçimi
5	Calibrate	ALU'da kalibrasyonlu ölçümü açma kapama biti, Eğer bit değeri 1 yapılırsa, Cal1 ve Cal2, ALU tarafından kullanılır.
4	DisAutoCal	Ölçümden sonra otomatik kalibrasyonu açma biti, Eğer bit temizlenir ise ölçüm sonunda Cal1 ve Cal2 güncellenir.
3	MRange2	Ölçüm aralığı seçme biti
2	NEG_STOP2	STOP2 çıkan yada inen kenar tetikleme seçeneği
1	NEG_STOP1	STOP1 çıkan yada inen kenar tetikleme seçeneği
0	NEG_START	START çıkan yada inen kenar tetikleme seçeneği

Çizelge 5.4. Yazılabilir Kayıtçı1

Bit	Kayıtçı1	Tanım
23	HIT2	ALU için ölçüm operatörlerini seçme biti.
22		
21		
20		
19	HIT1	ALU için ölçüm operatörlerini seçme biti.
18		
17		
16		
15	EN_FAST_INIT	Birinci ölçüm aralığı için hızlı hazırlama modu açma kapama biti. Fast initialization: Ölçüm aralığı 1 de tdc-gp2 hızlı hazırlama olanağını sunar. Register1 bit15 EN_FAST_INIT=1 interrupt flağı otomatik olarak TDC'yi hazırlar. Böylelikle TDC zaten data okunurken tdc bir sonraki ölçüme hazırlanır. Bu mod sadece çok hızlı uygulamalar içindir.
14	s.c	Üreticiye ait özel bit.
13	HITIN2	STOP2 kanalında beklenen darbe sayısı
12		
11		
10	HITIN1	STOP1 kanalında beklenen darbe sayısı
9		
8		
7	nc	
...		
0		

Çizelge 5.5. Yazılabilir Kayıtçı2

Bit	Kayıtçı2	Tanım
23	EN_INT	Kesme kaynakları açma ve kapama bitleri.
22		
21		
20	RFEDGE2	STOP2 için kenar duyarlılığı seçme bitleri
19	RFEDGE1	STOP1 için kenar duyarlılığı seçme bitleri
18 ... 0	DELVAL1	START ile brinci STOP arasında maskelenmiş zaman aralığı değeri. Bu aralık değeri boyunca gelen birinci STOP darbeleri göz ardı edilir.

Çizelge 5.6. Yazılabilir Kayıtçı3

Bit	Kayıtçı3	Tanım
23	s.c.	Üreticiye ait özel bit
22	s.c.	Üreticiye ait özel bit
21	EN_ERR_VAL	Bit değeri 1 yapılırsa zaman aşımı gerçekleştiğinde ilgili sonuç kayıtçısının 0xffffffff değeri yazar
20 19	SEL_TIMO_MR2	Zaman aşımı süresini belirleme bitleri
18 ... 0	DELVAL2	START ile ikinci STOP arasında maskelenmiş zaman aralığı değeri. Bu aralık değeri boyunca gelen ikinci STOP darbeleri göz ardı edilir.

Çizelge 5.7. Yazılabilir Kayıtçı4

Bit	Kayıtçı4	Tanım
23	s.c.	Üreticiye ait özel bit
22	s.c.	Üreticiye ait özel bit
21	s.c.	Üreticiye ait özel bit
20	s.c.	Üreticiye ait özel bit
19	s.c.	Üreticiye ait özel bit
18 ... 0	DELVAL3	START ile üçüncü STOP arasında maskelenmiş zaman aralığı değeri. Bu aralık değeri boyunca gelen üçüncü STOP darbeleri göz ardı edilir.

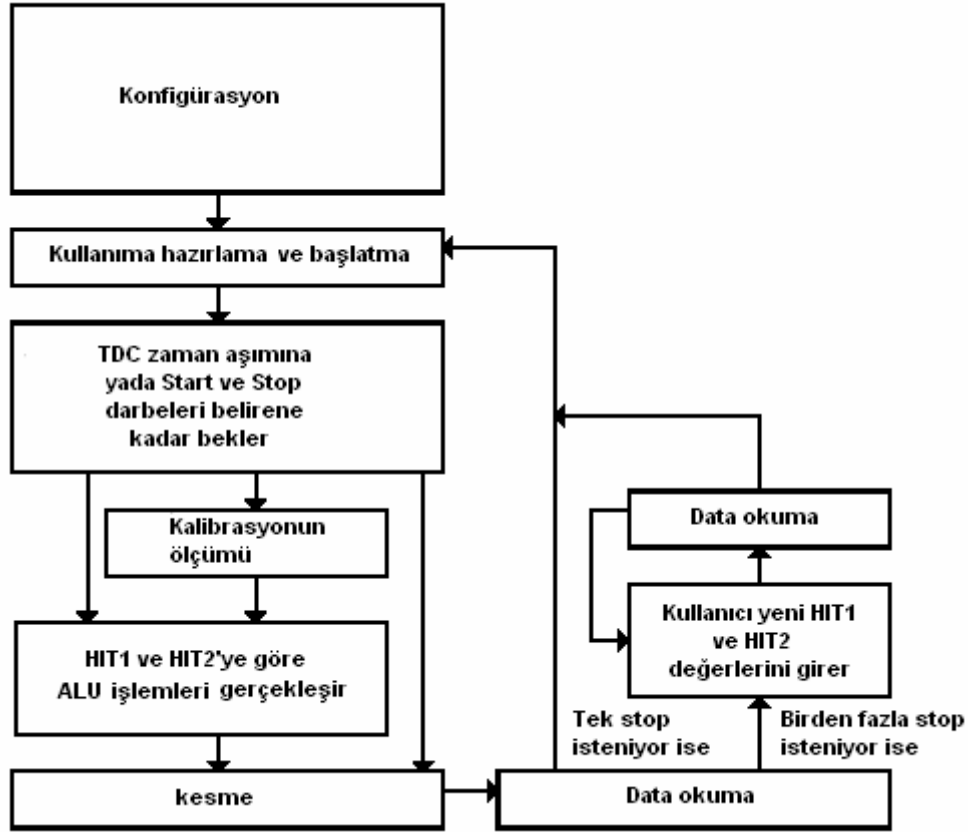
Çizelge 5.8. Yazılabilir Kayıtçı5

Bit	Kayıtçı 5	Tanım
23	CONF_FIRE	FIRE1 ve FIRE2 çıkışlarını aktif/pasif yapma bitleri
22		
21		
20	EN_STARTNOISE	Start darbesine rasgele kaymalar ekleme birimini aktif yapmak için bir bit
19	DIS_PHASENOISE	1 yapılması zorunludur Kalibrasyon cloğunu fire pulse generatörden ayırır.
18	REPEAT_FIRE	Etrafa çınlama döngüsü için darbe tekrar sayısı
17		
16		
15 ... 0	PHASE_FIRE	FIRE1 yada FIRE2 çıkışından yayımlanacak darbelerin her birinin faz bilgisi.

Çizelge 5.9. Okunabilir Kayıtlar

isim	Adres	Bit	tanım	
Sonuç1	0	32	Ölçüm sonucu 1 2 ¹⁵ 2 ⁰ ,2 ⁻¹ 2 ⁻¹⁵	
Sonuç2	1	32	Ölçüm sonucu 2 2 ¹⁵ 2 ⁰ ,2 ⁻¹ 2 ⁻¹⁵	
Sonuç3	2	32	Ölçüm sonucu 3 2 ¹⁵ 2 ⁰ ,2 ⁻¹ 2 ⁻¹⁵	
Sonuç4	3	32	Ölçüm sonucu 4 2 ¹⁵ 2 ⁰ ,2 ⁻¹ 2 ⁻¹⁵	
Durum	4	16	bitler	tanım
			0...2	sonuç göstericisi, bir sonraki boş sonuç kayıtcısını gösterir
			3...5	STOP1 kanalında beliren darbelerin sayısı
			6...8	STOP2 kanalında beliren darbelerin sayısı
			9	TDC zaman aşımının belirdiğini gösterir (1 olmuşsa)
			10	İkinci ölçüm aralığında 14 bitlik zaman aşımı ön sayıcısının taşması ile değeri 1 olur.
			11	1 değerini alması, sıcaklık sensörü açık devre olduğunu gösterir
			12	1 değerini alması, sıcaklık sensörü kısa devre olduğunu gösterir
			13.15	kullanılmıyor
test	5	8	Kayıtcı1'in yüksek anlamlı byte değerinin içerir.	

5.3.2. Zaman ölçüm akışı



Şekil 5.4. TDC GP2 zaman ölçümüm akış diyagramı

Her hangi bir anda zaman ölçümünden önce TDC GP2 ölçüm öncesinde konfigürasyonu uygulamaya uygun olarak yapılmış olmalıdır. Kullanılacak ölçüm aralığı, darbe sayıları, giriş uçlarının tetikleme seviyeleri gibi ayarlar kullanıcı tarafından belirlenip TDC GP2'nin kayıtçılarında ilgili bitlere girilmelidir. Bu işlem “Adrese yaz” komutu ile gerçekleştirilir. Kullanıcı “10000” bilgisine 3bit kayıtçı adres bilgisini ekleyerek “Adrese yaz” kodunu oluşturur. “Adrese yaz” operasyon kodunun gönderilmesinden sonra TDC GP2 24bit seri bilgi bekler ve gelen bu 24bit bilgiyi ilgili adresteki kayıtçısına yerleştirir. Yapılan konfigürasyonun ardından “Hazırla” ve “Başlat” operasyon kodlarının gönderilmesiyle ölçüm işlemi başlatılır. TDC GP2 bu komutlar ile birlikte seçilen konfigürasyona göre FIRE1 ve FIRE2 çıkışlarından darbeler yayınlayacak ve ölçümün START ve STOP darbelerini bekleyecektir. Eğer beklenen darbe belirlenen aralık içerisinde alınamaz ise bir

zaman aşımı gerçekleşecek, bu bir kesme yaratacak ve durum kayıtcısı aracılığıyla harici mikrokontrolöre bildirilecektir. İkinci ölçüm aralığında kalibrasyonlu ölçüm kullanmak zorunludur. START ve STOP darbelerinin ardından kalibrasyonda elde edilen Cal1 ve Cal2, ölçüm anında elde edilen hassas ve kaba ölçümler ALU tarafından işleme sokulur. Zaman ölçümü 16bit tam sayı ve 16bit kesirli sayı içerecek şekilde sonuç kayıtcısına depolanır ve TDC GP2 bir kesme oluşturur. ALU'nun bu işlemi kaynak gerilimine göre 3.3V da 1.8us, 2.5V da 2.5us, 2.0V da 4.6us zaman alır. Bir ölçüm sonunda sonuç göstericisinin değeri 1 artar ve yeni boş sonuç kayıtcısını gösterir. Sonuç göstericisinin gerçek pozisyonu “Durum“ kayıtcısında görülebilir.

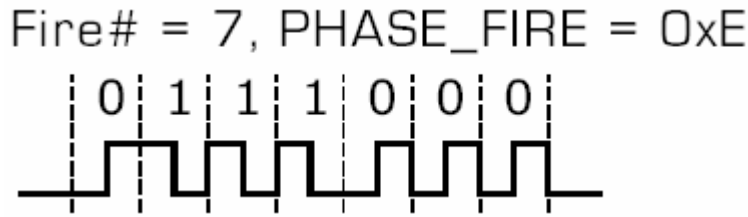
Kullanıcı sonuç bilgisini okumak için TDC GP2'ye “Adresten oku” komutu gönderir. Bu komuttan sonra TDC GP2'nin göndereceği seri bilgiler okunulacak adresteki kayıtcının içeriğidir. Eğer birden fazla STOP darbesinin süresi ölçülecek ise ALU işlem parametreleri HIT1 ve HIT2 güncellenerek bir sonraki STOP darbesinin ölçülmesi aynı şekilde tekrarlanır. İkinci ölçüm aralığında maksimum üç STOP darbesinin ölçümü mümkündür. Dolayısıyla üçüncü stop darbesi ile START sinyali arasındaki zamanın ölçülmesi ile ölçüm işlemi sonlanır. Ölçüm tekrar başlatılmadan önce TDC GP2 “Hazırla” komutu ile ölçüme hazır hale getirilmelidir.

5.3.3. Darbe üretici ve sıcaklık ölçüm birimi

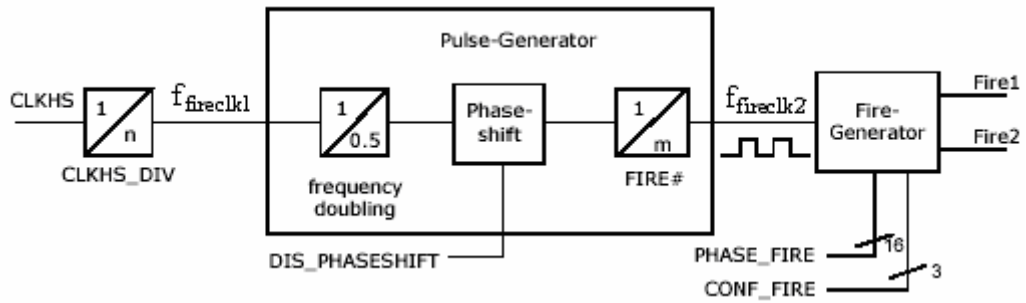
Darbe üretici frekans, faz ve darbe sayısı programlama yeteneğine sahiptir. Yüksek hızlı osilatör frekansı seçilen ClkHSdiv faktörü ile bölünür ve temel frekans olarak kullanılır. Şekil 5.6 darbe üreticinin iç diyagramını göstermektedir. Bu frekans bağımsız olarak DIV_FIRE tarafından 2 den 15 e kadar bir sayı ile bölünebilir. FIRE# bitlerine yazılan değer ile darbe üretici üzerinden 1 den 15'e kadar darbe yaratmak mümkündür. Her bir darbenin fazı PHASE_FIRE bitleri ile belirlenebilir. PHASE_FIRE bitlerinde ilgili bit 0 yapılarak darbenin fazı alçaktan yükseğe, 1 yapılarak yüksekte şeklinde ayarlanabilir. Darbe dizisi DAB dan başlar YAB ile sonlanır. Şekil 5.5 yedi darbe yayınlayacak bir uygulama örneğini sunmaktadır.

Darbe üretici “Başlat” operasyon kodu gönderilerek aktif hale getirilir. Darbe üretici FIRE1 ve FIRE2 olmak üzere iki çıkışa sahiptir. Her iki çıkış 5V 48 mA sağlar. Her iki çıkış paralellenerek sürücü çıkışı 96mA e çıkartılabilir.

Darbe üretici etrafa çınlama döngüsü benzeri kullanım için çok kez darbeleri yaratıp göndermeye izin verir. Bu özelliği kullanmak için alınan darbe dizisi TDC GP2’nin FIRE_LN girişine beslenir. Bu herhangi gecikme olmaksızın direkt olarak çıkış tamponuna gönderilir. Tekrarlanacak darbe sayısı REPEAT_FIRE bitleri ile belirlenir. Yalnızca darbelerin FIRE# tarafından belirlenen darbeler tekrarlanır. Eğer 5 us boyunca darbe alınmazsa TDC darbe dizinin sonu olduğunu anlar.



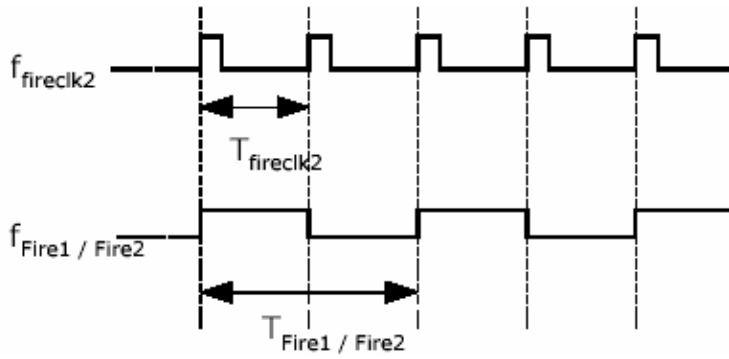
Şekil 5.5. TDC GP2 darbe üretici için yedi darbeli örnek darbe dizisi



Şekil 5.6. TDC GP2 darbe üretici blok diyagramı

$f_{fireclk2}$ FIRE1/FIRE2 için referans sinyaldir ve ana saat sinyali $f_{fireclk1}$ ile arasındaki ilişki Eş. 5.3’deki gibidir. Bu darbeler FIRE1/FIRE2 tampon çıkışları tarafından gönderilir. CONF_FIRE ile FIRE1 ve FIRE2 çıkışlarının konfigürasyonu yapılır.

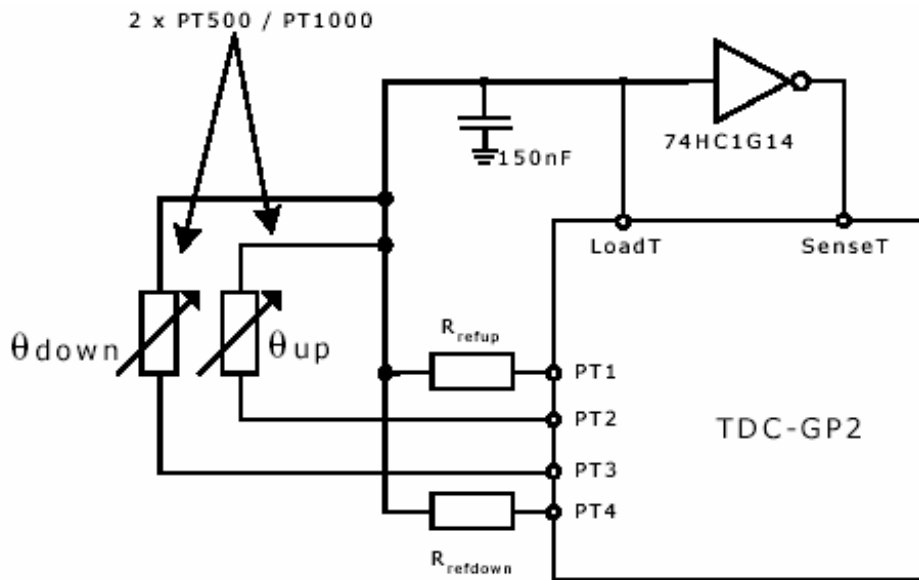
$$f_{fireclk2} = f_{fireclk1} \times \frac{1}{DIV_FIRE + 1} \quad (5.3)$$



Şekil 5.7. TDC GP2 darbe üretici frekans çıkışı

Şekil 5.7’de görüldüğü gibi, FIRE1/FIRE2 çıkış sinyalinin bir alçak evresi bir de yüksek evresi için en az iki periyot $f_{fireclk2}$ kullanılır.

Özellikle kalorimetre uygulamaları için TDC GP2 picostrain tabanlı sıcaklık ölçüm birimine sahiptir. Çok yüksek çözünürlük ve düşük akım tüketimine sahip bu birimde ölçüm kapasitörün boşalma zamanının ölçümüne dayanır. Bu yüzden bir kapasitör sırayla bir sensör direncinden ve referans direncinden boşaltılır. Şekil 5.8 sıcaklık sensörlerinin bağlantı biçimini göstermektedir.

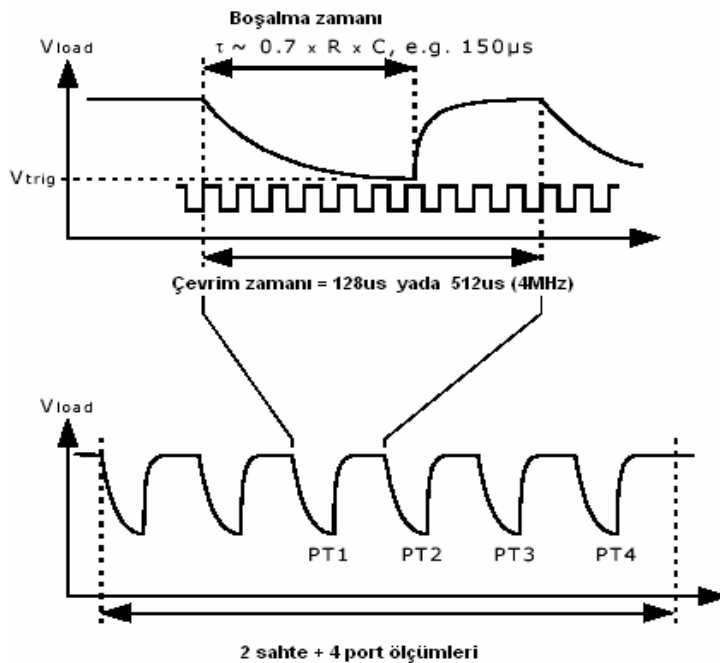


Şekil 5.8. Sıcaklık sensörleri bağlantı şekli

Sıcaklık birimi belirtilen fonksiyonları ile dört ölçüm portuna sahiptir. Bu portlar PT1 referans direnç düşük sıcaklık, PT2 sensör direnci düşük sıcaklık, PT3 sensör direnci yüksek sıcaklık, PT4 referans direnç yüksek sıcaklık portlarıdır.

Sıcaklık sensörleri minimum 500R rezistansa sahip olmalı. TDC-GP2 RC bağlantısının boşalma zamanını ölçmesi sırasında kesinlik yaklaşık 0.004 °C dir. Mikrokontrolörün gönderdiği “Sıcaklık Başlat” operasyon kodu ile tetiklenir. TDC GP2 dört ölçümü kontrol eder. Dört ölçümün sonunda işlem tamamlanır ve kesme oluşur. Dört zaman bilgisi sonuç kayıtçılarında tutulur.

Tcycle sıcaklık ölçümü için çevrim süresini belirler iken PORT# kullanılacak port sayısını dolayısıyla kullanılacak sensör sayısını belirler. PORT# değeri sıfır yapılır ise 2 port dolayısıyla tek sensör, bir yapılır ise 4 port dolayısıyla iki sensör kullanılır. FAKE# sıcaklık ölçümünün başlangıcında sahte ölçüm sayısını belirler. Yük kapasitörünün mekanik etkilerinin üstesinden gelmek için bu zorunludur. Bütün sıcaklık ölçümü $2 \times (\text{PORT\#}) + (\text{FAKE\#})$ çevrim sürer. Şekil 5.9 iki sahte ve 4 port kullanan ölçümün zamanlamasını göstermektedir.



Şekil 5.9. TDC GP2 sıcaklık ölçümü zamanlamaları

Sonuç2/Sonuç1 ve Sonuç3/Sonuç4 oranlarından mikrokontrolör, Rsensör/Rref oranını hesaplayabilir. Kullanımda olan özel sensörler için bir çevrim tablosu aracılığıyla sıcaklık ölçülebilir.

6. TASARIM HAKKINDA DETAYLI BİLGİLER

6.1. TDC GP2 Konfigürasyonu

6.1.1. Ölçüm parametrelerinin konfigürasyonu

Tasarımda kullanılacak zaman aralığı için ikinci ölçüm aralığı seçileceğinden Kayıtçı0(3) Mrange2 bitinin değeri 1 yapılmıştır. İkinci ölçüm aralığı için yüksek hızlı saate ihtiyaç duyarız. Söz konusu tasarım düşük güç uygulaması olduğundan yüksek frekanslı saat sinyali ölçümler arasında durdurulabilir. Bu durumda 32.768KHz osilatör sürekli açık kalacak ve zaman kontrolü için kullanılacaktır. Kayıtçı0(6) SelClkT bit değeri 1 yapılarak ölçümde kullanılacak referans saat sinyali yüksek frekanslı saat olarak seçilmiştir. Tasarımda fiyat ve düşük akım avantajlarından dolayı, yüksek frekanslı osilatör kaynağı olarak 4MHz seramik rezonatör kullanılmıştır. Seramik rezonatörler kapatılıp tekrar açıldıklarında ortalama olarak 200us sonra durağan hale gelirler. Kayıtçı0(10&11) START_CLKHS bit alanı değerlerine 2 değeri yazılarak yüksek frekanslı saat sinyalinin ölçüm sonlarında otomatik olarak kapanması sağlanmıştır. Bu seçeneğin kullanılmasıyla TDC GP2'ye gönderilen “Hazırla” komutundan sonra, TDC GP2 saat sinyalinin 640us kadar durağan duruma gelmesini bekleyecektir. Bu saat sinyalinde ön bölücü kullanmamak için Kayıtçı0(12&13) ClkHSDiv değerine 0 yazılmıştır. Böylelikle minimum ve maksimum ölçüm aralıkları Eş. 6.1 ve Eş. 6.2 yardımıyla bulunur.

$$t_{\min} = 2 \times T_{\text{ref}} \times 2^{\text{ClkHSDiv}} = 2 \times 250 = 500 \text{ ns} \quad (6.1)$$

$$t_{\min} = 2^{14} \times T_{\text{ref}} \times 2^{\text{ClkHSDiv}} = 2 \times 250 = 4096 \text{ us} \quad (6.2)$$

Kalibrasyon esnasında zaman aşımı olmaması için $2 \times T_{\text{ref}} \times 2^{\text{ClkHSDiv}} < 1.8 \text{ us}$ olmalıdır. Kayıtçı3(19&20) SEL_TIMO_MR2 değeri 3 yapılmış ölçüm esnasında oluşabilecek zaman aşımı için değer 4ms olarak belirlenmiştir.

Kayıtçı1(8..10) HITIN1 değeri 2 yapılarak, bir adet START ve bir adet STOP olmak üzere TDC GP2'nin beklemesi gereken darbelerin sayısı iki olarak belirlenmiştir. STOP2 kanalı kullanılmayacağından Kayıtçı1(11..13) HITIN2 değerine 0 yüklenmiştir.

Hassas zaman ölçüm birimi üç STOP darbesine kadar izin verir iken ALU birimi sadece tek bir STOP darbesinin süresini hesaplayabilir. Bu nedenle ALU'nun o an hangi parametreleri hesaplayacağı HIT1 ve HIT2 ye girilmelidir. Kayıtçı1(16..19) HIT1 değeri 1 yapılarak ilk parametre START, Kayıtçı1(20..23) HIT2 değeri 2 yapılarak ikinci parametre olarak ilk STOP darbesi seçilmiştir. Böylelikle TDC GP2 START darbesi ile STOP1 girişine gelen ilk STOP darbesi arasındaki zaman farkını ölçecektir.

Kayıtçı0(5) Calibrate değeri 1 yapılarak, ALU'nun zaman ölçme işlemlerinde Cal1 ve Cal2'yi kullanması sağlanmıştır. Kayıtçı0(4) DisAutoCal değeri 0 yapılarak Cal1 ve Cal2'nin her ölçüm sonunda güncellenmesi sağlanmıştır. Cal1 ve Cal2 yi güncellenmenin diğer bir yolu "TDC kalibrasyonu başlat" operasyon kodunu kullanmaktır.

Zaman aşımı gerçekleşip gerçekleşmediğini anlamak için "Durum" kayıtçısını kontrol etmek yerine, Kayıtçı3(21) EN_ERR_VAL değeri 1 yapılarak zaman aşımı olduğunda sonuç kayıtçısına 0xffffffff değerinin yazılması sağlanmıştır.

6.1.2. Donanıma göre uyarlanan parametreler

Kayıtçı2(19) RFEDGE1 değerine 0 ve Kayıtçı_2(20) RFEDGE2 değerine 0 yazılarak STOP1 ve STOP2 kanallarının düşen kenara yada inen kenara duyarlı olması sağlanmıştır. Kayıtçı0(0) NEG_START, Kayıtçı0(1) NEG_STOP1 ve Kayıtçı0(2) NEG_STOP2 değerlerine 0 yazılarak, girişlerdeki tersleyiciler pasifleştirilmiştir. Dolayısıyla START, STOP1 ve STOP2 girişleri yükselen kenar tetiklemeli olarak çalışmaları sağlanmıştır. Darbe üreticinin tek darbe yaratması için Kayıtçı0(20..23) FIRE# değerine 1 yazılmıştır. Akış yönünde ölçüm anında Kayıtçı5(21..23) FIRE

CONF_FIRE değerine 1 yazılarak FIRE2 çıkışı aktif, akışın tersi yönde zaman ölçümü anında CONF_FIRE değerine 2 yazılarak FIRE1 çıkışı aktif olması sağlanmıştır. Kayıtçı0(16..19) DIV_FIRE değerine 1 yazılarak darbe üretici için ön bölücü 2 yapılmıştır. Darbe üretici tarafından yayımlanacak her darbenin fazı alçaktan yükseğe geçiş yaratacak şekilde olması için Kayıtçı5(0..15) PHASE_FIRE değerine 0 yazılmıştır. Darbe tekrar özelliği kullanılmayacağından Kayıtçı5(16..18) REPEAT_FIRE değerine 0 yazılmıştır.

TDC GP2 STOP1 üzerindeki her 3 darbe için 10ns çözünürlükte zaman maskeleme kurabilir. Bu zaman maskeleri “dahili durdurma açık” birimini aktif eder. Dahili durdurma açık birimi harici durdurma açık girişlerine bir mantıksal VE kapı ile bağlıdır. Dahili açık birimini kullanabilmek için harici açık girişleri yüksek seviyede olmalıdır. Zaman maskeleme değeri t_{Delay} Eş. 6.3 ile hesaplanabilir.

$$t_{Delay} = DELVALX.T_{ref} \quad (6.3)$$

DELVALX değerleri 14bit tamsayı, 5bit kesirli sayı şeklindedir. Akış tüpümüzde sensörler arasındaki mesafe 60mm olduğundan ortalama iletim süresi 41us kadardır. Kayıtçı2(0..19) DELVAL1 değerine 4096 yazılmıştır. DELVALX değerinin yüksek anlamlı 14 biti tam sayı, düşük anlamlı 5 biti kesirli olduğundan DELVALX değeri 128,0 olacaktır. Eş. 6.3’den t_{Delay} değeri 32us olacak ve TDC GP2 bu aralıkta gelecek herhangi bir STOP darbesini göz ardı edecektir.

ALU kesmesini ve zaman aşımı kesmesini aktif etmek için Kayıtçı2(21..23) EN_INT değerine 5 yazılmıştır.

Seramik rezonatörlerin sıcaklık teloransları %0.3 ile %0.5 arasındadır. Bu nedenle seramik rezonatörün kullanılması durumunda seramik rezonatör 32.768KHz kristal aracılığıyla kalibre edilmelidir. TDC GP2 bu olanağı sunmaktadır.

Bu kalibrasyon kesin doğrulukta olan 32.768KHz'nin ölçümlerine dayanır. TDC-GP2 START ve STOP darbelerini 32.768KHz aracılığı ile yaratır ve zaman aralığını ölçer. Ölçüm sonucu sonuç kayıtcısında tutulur ve bir kesme oluşur. Seramik rezonatörün frekans hatası mikrokontroller aracılığıyla hesaplanabilir. Kayıtcı0(14..15) CALRES# değerine 3 yazılmıştır. Böylelikle rezonatör kalibrasyonu için TDC GP2 32.768KHz'nin on altı çevrimini kullanacaktır. Kalibrasyon “Rez.kalibrasyonu.başlat” komutu ile yapılır. Ölçümün tamamlanmasıyla birlikte mikrokontrolör sonucu okur ve teorik ölçüm ile karşılaştırır ve bir düzeltme faktörü bulur. Daha sonra ölçülen ultrason iletim zamanlarında bu düzeltme faktörü uygulanarak rezonatörden kaynaklanan hata girilecektir.

6.1.3. Diğer parameteler

Tasarımda sıcaklık ölçümü kullanılmamıştır. Yinede Kayıtcı0(7) FAKE# değerine 0 yazılarak ölçüm öncesinde iki sahte ölçüm yap seçeneği seilmiştir. Kayıtcı0(9) PORT# değerine 0 yazılarak iki sensör kullanımı seçilmiştir.

Kayıtcı1(15) EN_FAST_INIT değeri, ikinci ölçüm aralığında kullanılmadığı için sıfırlanmıştır.

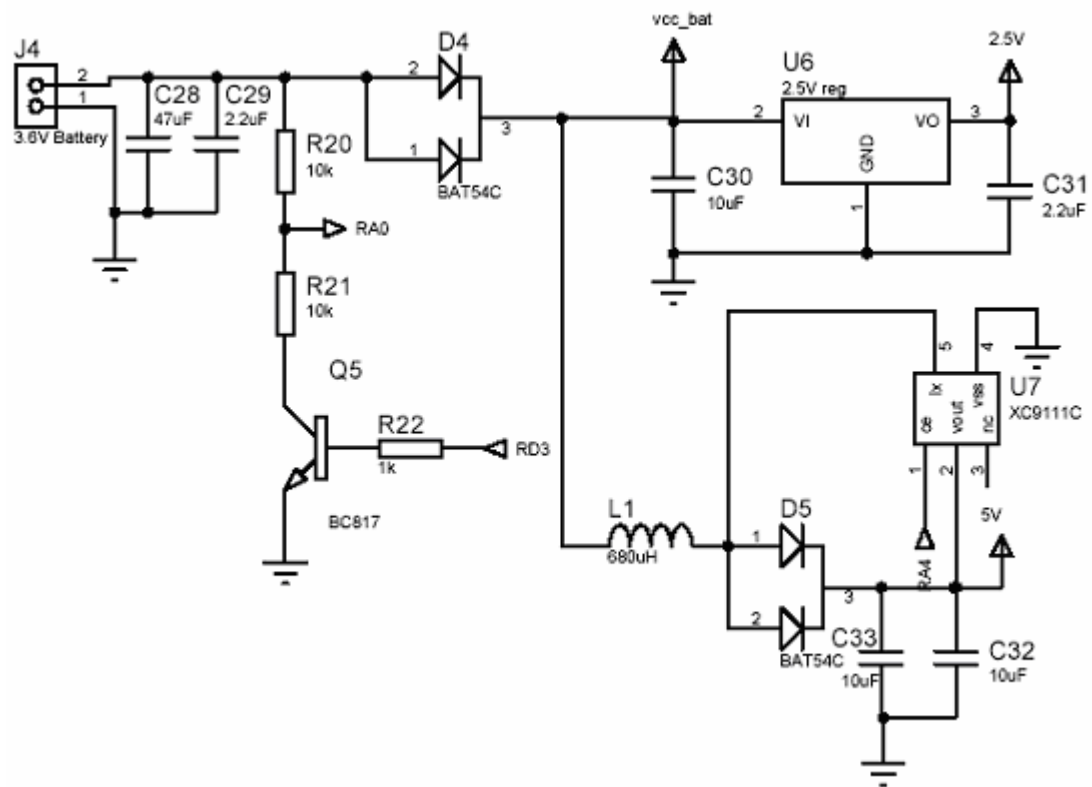
Tek STOP darbesi kullanıldığı için Kayıtcı3(0..19) DELVAL2 ve Kayıtcı4(0..19) DELVAL3 değerleri sıfırlanmıştır.

Kayıtcı5(19) DIS_PHASENOISE zorunlu olarak 1 değeri verilmiş ve Kayıtcı5(20) EN_STARTNOISE değerine 0 yazılmıştır.

6.2. Donanım

TDC GP2 tamamen dijital devre olmasına rağmen bazı analog etmenler ölçüme etki eder. TDC GP2 dahili analog iletim zamanı gecikmesi kullandığından sıcaklıktan ve güç kaynağından etkilenir. Bir çok uygulamada anahtarlamalı mod güç kaynaklarının

direkt besleme gerilimlerine uygulanması tavsiye edilmez. Bunların yerine lineer regülatörler kullanılmalıdır. Bu tip uygulamalarda NPO yada COG tipinde kapasitörler ve yüzey montaj komponentler kullanılması önerilmektedir. Devre tasarımında besleme hattının iyi tasarlanması iyi ölçüm için gereklidir. Şekil 6.1 kullanılan gerilim seviyeleri ve pil gerilimi ölçme devresini göstermektedir.



Şekil 6.1. Gerilim seviyeleri ve pıl gerilimi ölçme devresini

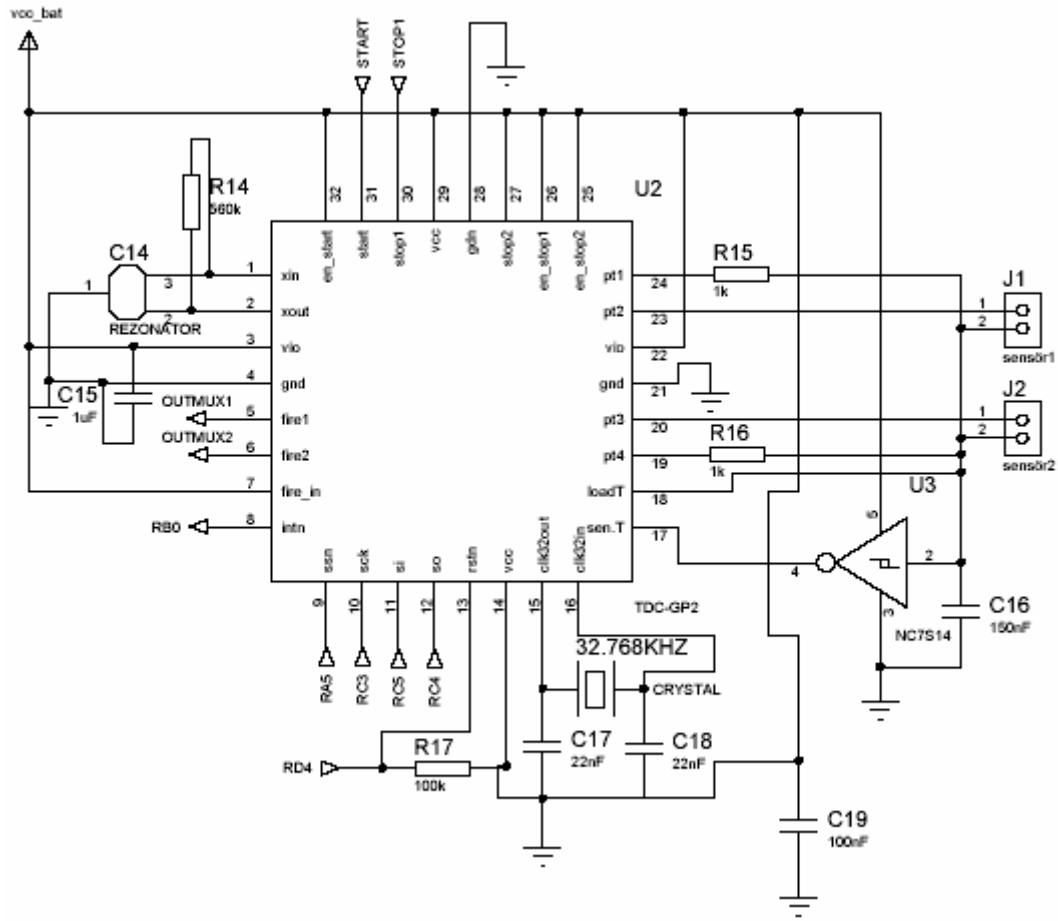
Uygulamaya göre TDC GP2 iki farklı saat sinyal kaynağı kullanabilir. Bunlardan birincisi frekansı 2MHz ile 8MHz arasında olan yüksek hızlı osilatördür. Yüksek hızlı osilatör ölçüm referansı ve hassas birim kalibrasyonu için kullanılır. İkinci sinyal kaynağı 32.768KHz quartz osilatörüdür ve dahili zamanlama birimleri için kullanılır.

Yüksek frekanslı osilatör olarak quartz kristali yerine seramik rezonatör kullanmak fiyat, durulma zamanı ve düşük akım tüketimi gibi avantajlar sunar. Fakat seramik

rezonatörler zayıf bir sıcaklık karakteristiklerine sahiptirler. Bu dezavantajın kullanılan 32.768KHz quartz osilatörü yardımıyla üstesinden gelinebilir.

32.768KHz bir quartz kristalinin akım tüketimi 3.6Vda 4.5uA kadar, durulma süreleri ise 2sn civarındadır. Yüksek frekanslı osilatörler için akım tüketimi 270uA kadar çıkmaktadır. Seramik osilatörün kısa durulma zamanı akım tüketimini biraz daha azaltabilir. Bu şekilde on yıllık bir uygulamada bir kaç yüz mili amper saat tasarruf sağlayabilir.

Quartz osilatörleri aracılığıyla üretilen saat sinyalleri kapatılıp açılması durumunda rezonatörlere nazaran daha yüksek bir zamanda kararlı hale gelir. Bu süre quartz osilatörleri için 3ms'ye kadar ulaşabilir iken, rezonatörlerde bu süre 200us civarındadır. TDC GP2 640us ve 1280us'lık durulma gecikmelerini desteklemektedir. Bu durumun üstesinden gelebilmek için mikrokontrolör START_CLKHS bitleri yardımıyla osilatörü dolaylı yoldan açıp yada kapatabilir. Dolayısıyla durulma zamanı kullanıcı tarafından ayarlanabilir. Bir diğer yol ise 32.768KHz yerine daha düşük referans saatler kullanmaktır. Örneğin 4KHz'lik bir saat sinyali kullanmak 10ms'ye kadar gecikme zamanı sağlayacaktır. Fakat bu durumlar ortalama akım tüketimine kötü yönde etki edecektir. Şekil 6.2 TDC GP2 yongasını, kullanılan osilatörleri göstermektedir.

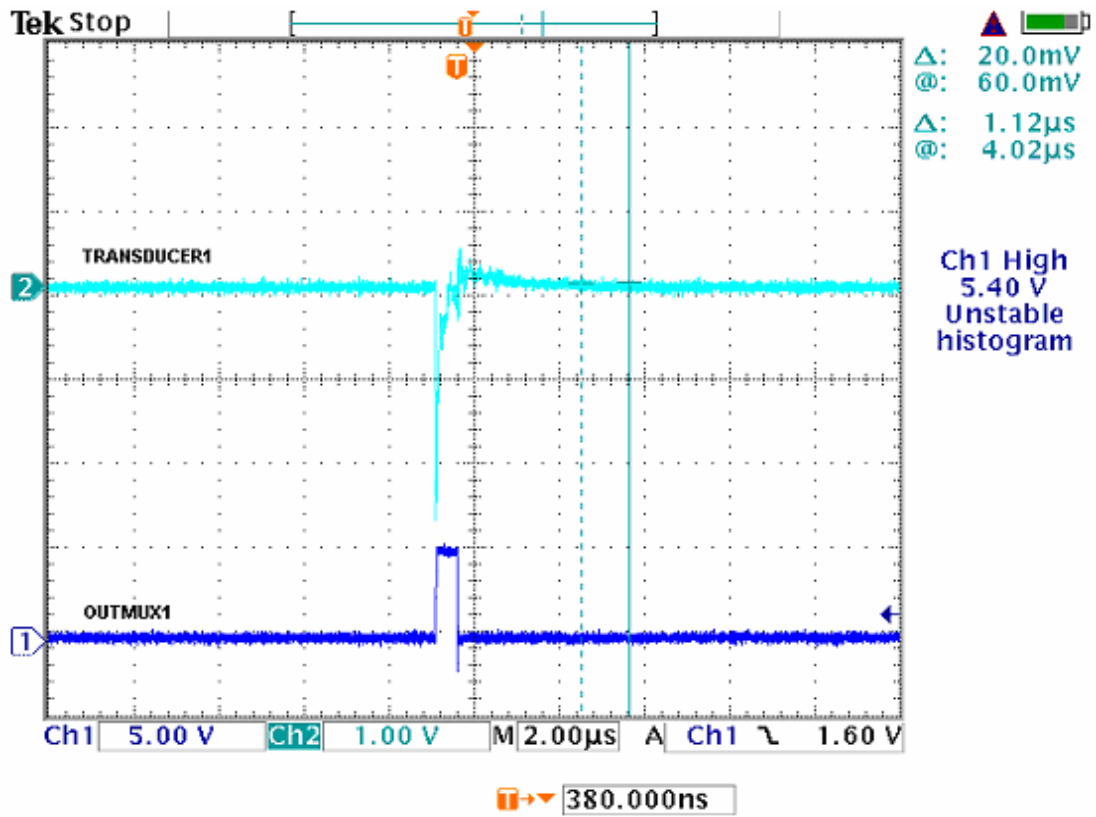


Şekil 6.2. TDC GP2 ve osilatörler

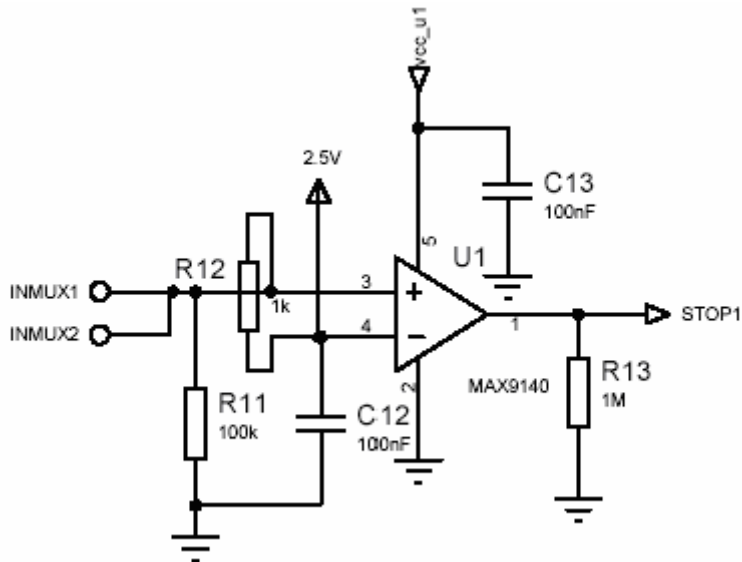
32.768KHz saatin ppm (her milyon darbeye darbe sayısı) hataları çok küçüktür fakat darbeleri 3-5ns'lik bir salınıma sahiptir. Rezonatör kalibrasyonunda bu salınım hatası rezonatör kalibrasyon bilgisine ve dolayısıyla ölçülen zamanlara yansır. Fakat aynı kalibrasyon değeri ile t_1 ve t_2 zamanları hesaplandığında Eş. 4.12'de bunun etkisi akışa ihmal edilebilir miktarda yansıyacaktır. Harici bir mikrokontrolör aracılığıyla 32.768KHz'lik osilatör yerine düşük frekanslı kare dalga CLK32Out bacağına sağlanabilir. Bu akım tüketimini 1.2uA kadar azaltacaktır. 32.768KHz osilatör kullanılmayacak ise clk32in girişi GND'ye bağlanmalıdır.

Şekil 6.3 transdüser tetikleme düzeneklerini, Şekil 6.4 ise transdüser tetikleme sinyallerini göstermektedir. Mikrokontrolör ölçüm yönünü STATUS1, STATUS2 sinyalleri ve TDC GP2'nin FIRE1/FIRE2 konfigürasyonu ile ayarlar. Akış yönünde

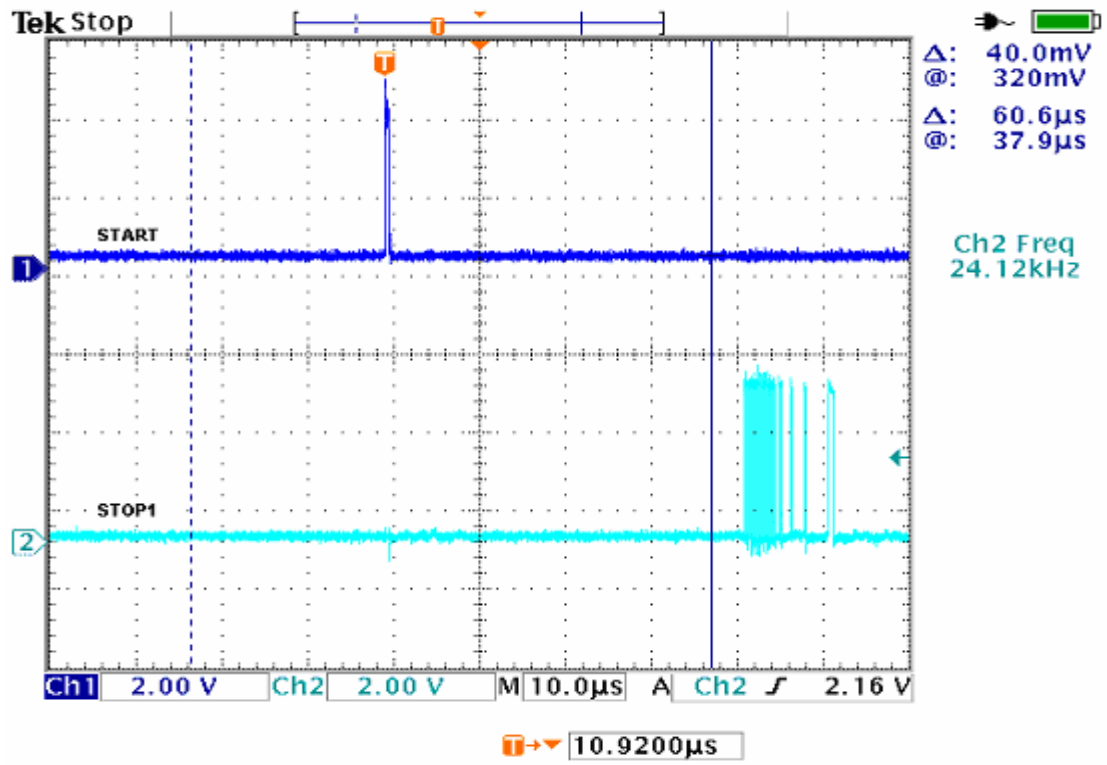
iletim süresini ölçmek için FIRE2 çıkışı yüksek empedans, FIRE1 çıkışı aktif yapılır. STATUS2 yüksek seviyede, STATUS1 ise düşük seviyede tutulur. Darbe üreticinin yarattığı tek darbe transdüseri tetikleyecektir aynı zamanda bir diyot üzerinden START darbesini oluşturacaktır. Alıcı transdüser ultrason yankılarını INMUX hattına iletir. Alınan ses ekoları bir karşılaştırıcı aracılığıyla dijital kare dalgalara çevrilir. Karşılaştırıcı çıkışı STOP1 kanalını beslemektedir. Şekil 6.5 karşılaştırıcı devresini, Şekil 6.6 START ve STOP darbelerini göstermektedir. Ayrıca şekil 6.7 STOP1 darbelerini ayrıntılı göstermektedir. Karşılaştırıcı çıkışında oluşan darbe dizisi ultrasonik transdüserle ile aynı frekansda 4MHz'dir. Yankı sinyalinin sonunda sistemin geçici tepkisi nedeniyle ile artık sinyallerden oluşan darbelerde ayrıca görünmektedir. 4MHz kare dalganın ilk darbesi ile ölçüm sonlanır. Akışın tersi yönde zaman ölçümü için STATUS1/STATUS2 ve FIRE1/FIRE2 durumları terslenir ve benzer aşamalar ile ölçüm gerçekleşir.



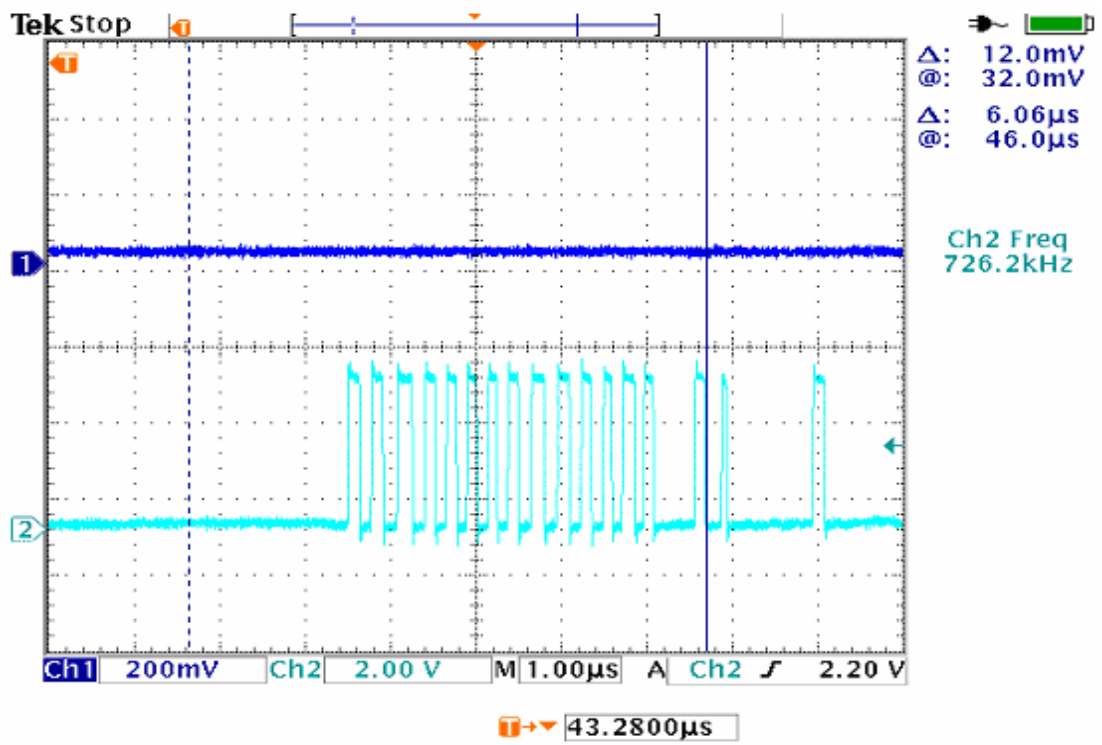
Şekil 6.4. Transdüser tetikleme sinyalleri



Şekil 6.5. STOP darbesini oluşturan karşılaştırıcı birim

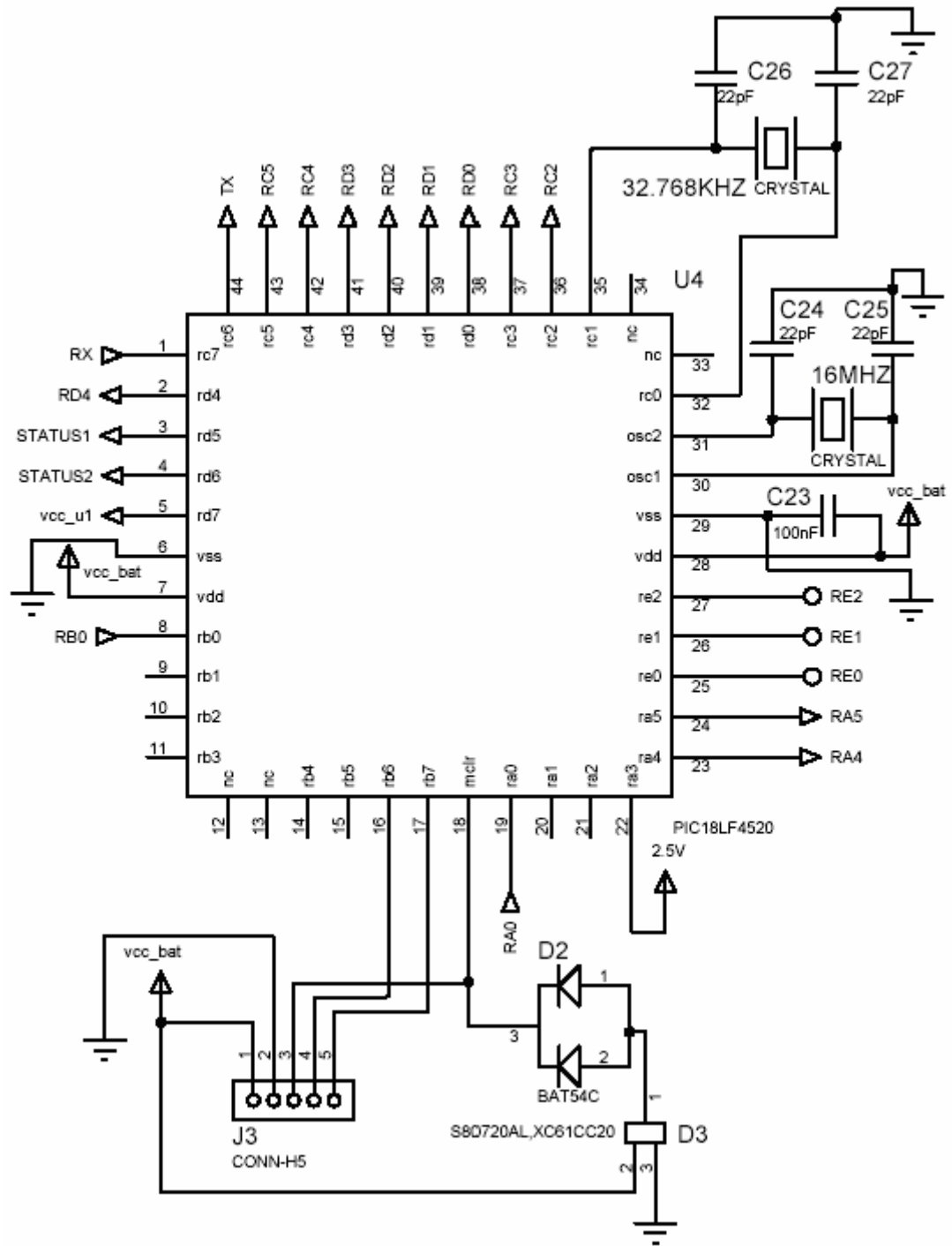


Şekil 6.6. START ve STOP sinyalleri

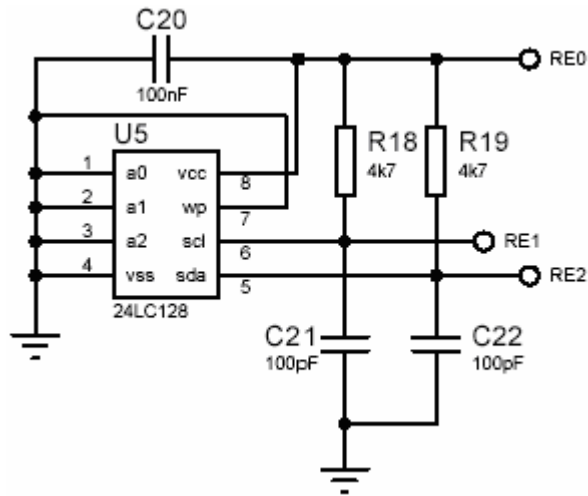


Şekil 6.7. STOP sinyallerinin ayrıntılı görünüşü

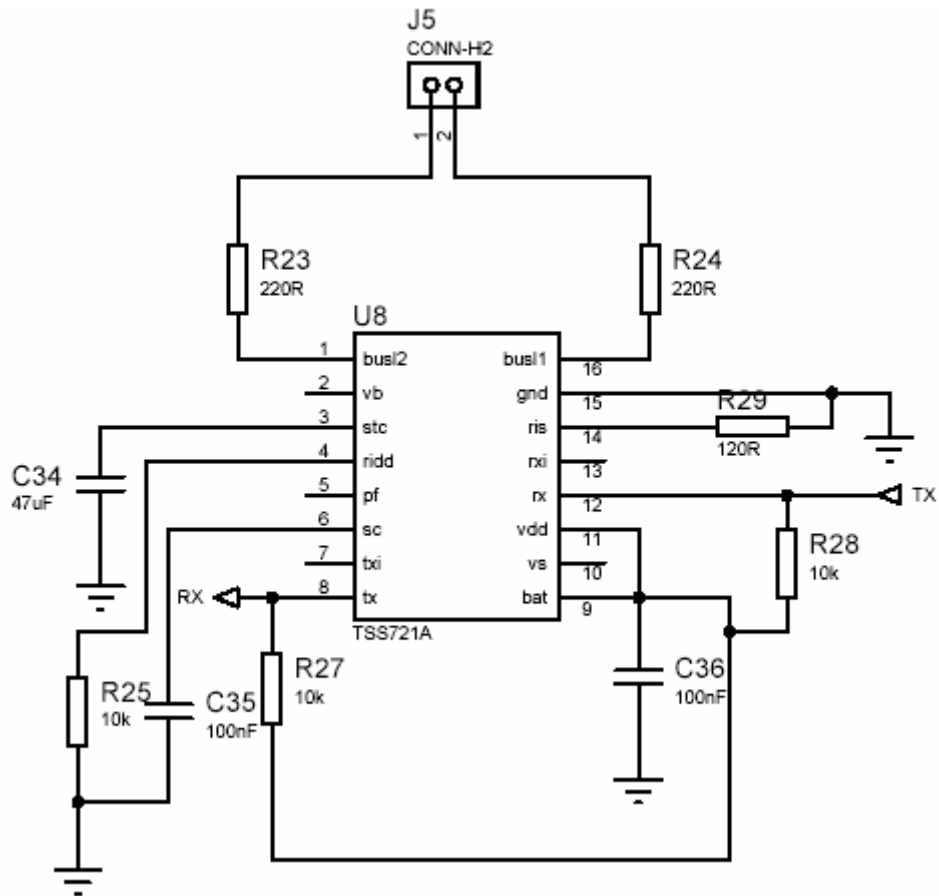
Şekil 6.8 mikrokontrolörün bağlantılarını göstermektedir. Çalışma frekansı için 16MHz kristal kullanılmıştır. Ayrıca “Zamanlayıcı1” osilatörüne 32.768KHz kristal ayrıca bağlanarak gerçek zamanlı saat olarak kullanılmıştır. Tasarımı yapılan akış ölçer için bu yapı hem saat/tarih kavramları için hem de kesin doğrulukta saniyede bir kez akış oranının örneklenmesi için kullanılmıştır. 18F4520 dahili “brown out reset” birimine sahiptir. Bu birim yaklaşık olarak 20uA akım tükettiği için “X80720AL” yada “XC61CC20” gerilim detektörlerinden biri kullanılmıştır. Gerilim detektörlerinin amacı herhangi bir gerilim düşmesi durumunda mikrokontrolörü yeniden başlatmaktır. Şekil 6.9, Şekil 6.10, Şekil 6.11, sırası ile 24lc128 I²C eeprom, tss721a M-bus köle yonga ve HT16220 LCD sürücü bağlantılarını göstermektedir. M-bus arabirimi özellikle düşük güçlü sayaç uygulamaları için ideal iki kablolu iletişim ara yüzüdür. Yönetici ve köle olarak iki ana birime sahiptir. Köle, besleme enerjisini yönetici tasarımdan sağlar. Bu nedenle devrede kullanılan pile her hangi bir yük teşkil etmez.



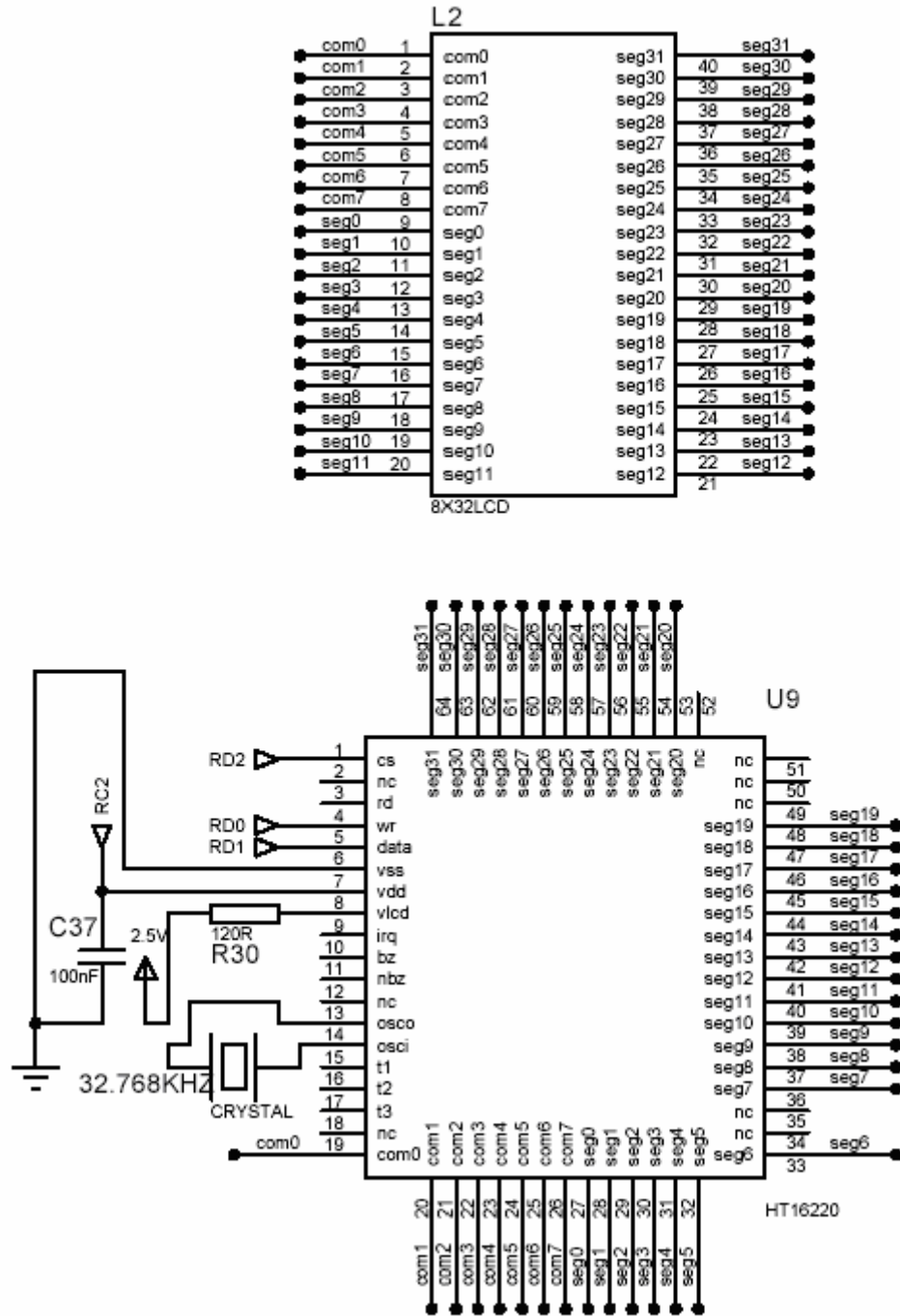
Şekil 6.8. Mikrokontrolör devresi



Şekil 6.9. I²C eeprom devresi



Şekil 6.10. M-BUS ara yüzü köle yonga devresi



Şekil 6.11. LCD sürücü ve cam lcd için bağlantı biçimi

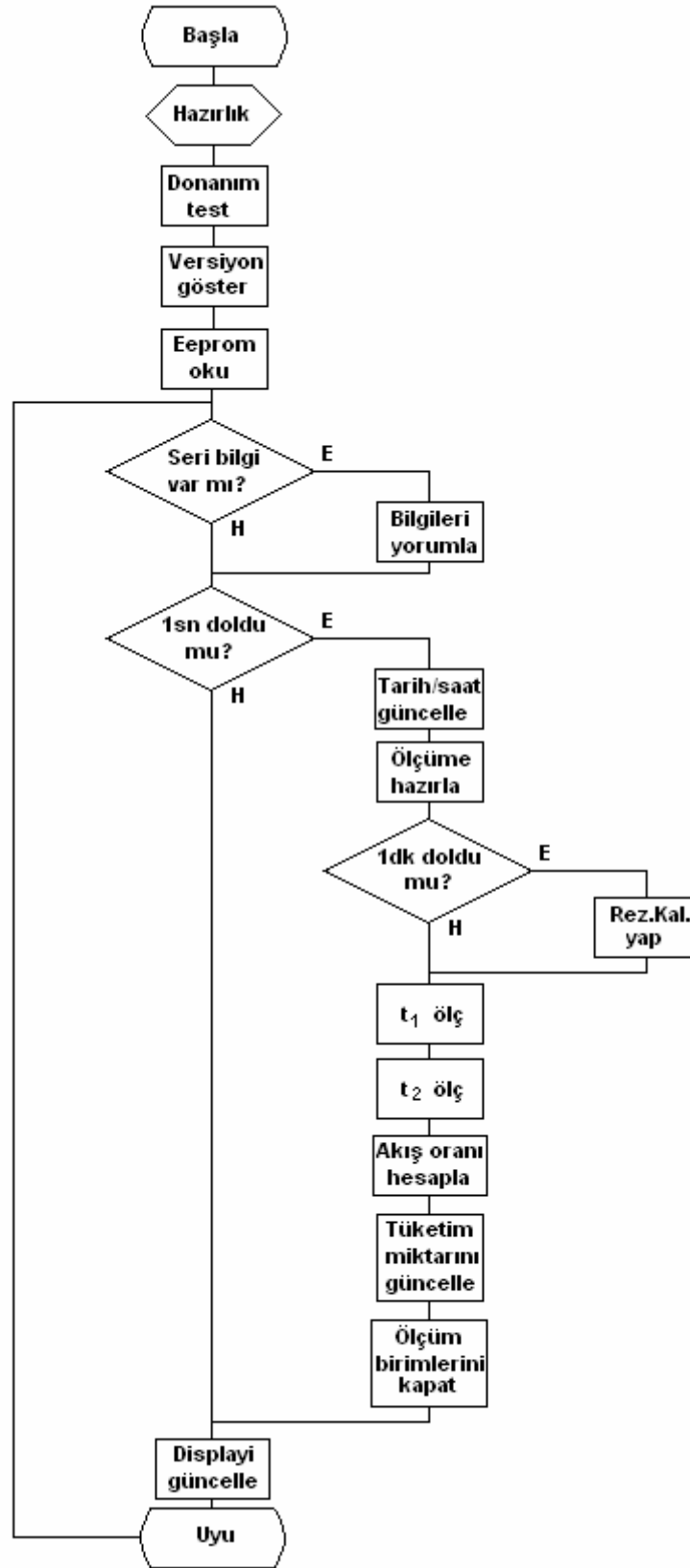
6.3. Gömülü Yazılım

Şekil 6.12 gömülü yazılımın genel akışını göstermektedir. Hazırlık aşamasına değişkenlere ilk değerler verilmesi, portların ayarlanması, kesmelerin zamanlayıcıların ve diğer çevre birimlerin kurulması ile başlanmaktadır. Mikro işlemci hazırlığında sonra LCD ve TDC GP2 ayarları ayrıca gerçekleştirilmektedir.

Donanım test işlemlerinde TDC GP2, 24LC128 test edilir iken birde pil ölçümü gerçekleştirilmektedir. TDC GP2 test kayıtcısı aracılığı ile test edilir. 24LC128 belli bir adresine yazma ve o adresi tekrar okuma işlemleri yapılarak test edilmektedir. Pil ölçümü saatte bir kez tekrarlanmaktadır. RD3 bacağı yüksek seviyeye çekilip RA0'dan analog okuma yapılarak pil ölçümü gerçekleştirilmiştir. Analog ölçüm için referans gerilimler 2.5V ve GND uçları seçilmiştir. Bu nedenle RA3 girişi 2.5V ile beslenmektedir.

LCD üzerinde kısa bir süre yazılım versiyonunun gösterilmesinin ardından eepromdan okuma işlemleri gerçekleştirilir. Bu esnada kalibrasyon bilgileri, akış ölçerin seri numarası, üretim tarihi gibi akış ölçere ait bazı veriler, tüketim bilgileri ve tarih/saat bilgileri gibi bilgiler okunulur.

Mikrokontrolör akım tasarrufu için zamanının çoğunu uyuma pozisyonunda geçirmektedir. Uyuma pozisyonundan “Zamanlayıcı1” kesmesi ile yada M-bus sorgusunda oluşacak USART kesmesi ile mikrokontrolör koşma durumuna geçer. Bu durumda bir M-bus sorgu yada komutu var ise buna cevap verilir. Eğer uyanma “Zamanlayıcı1” ile gerçekleşmiş ise bu durumda t_1 ve t_2 zamanları ölçülerek akış oranı hesaplanır. Ölçüm sonunda bulunan akış oranının birimi lt/h'dir. LCD üzerinde 10ml hassasiyetinde tüketim gösterilebilmektedir. Bu nedenle bu oran 36'ya bölünerek 10ml çözünürlüğünde tüketim bilgisi elde edilir ve bu bilgi toplam tüketime eklenir.



Şekil 6.12. Gömülü yazılım akış diyagramı

7. DENEYSEL ÇALIŞMALAR

Sıvı akışkanların doğruluk testleri sekiz farklı akış oranında yapılmaktadır. Akış oranları yapılan testler boyunca sabittir. Test boyunca akış ölçer içerisinden geçen akışkan bir kazanda toplanır ve kazanda biriken akışkanın kütlesi referans alınarak akışkanın yoğunluğu yardımıyla akış ölçerden geçen toplam akışkan hacmi bulunur. Bulunan toplam hacim akış ölçerin ilk ve son endekslerinin arasındaki fark ile karşılaştırılarak bu debide akış ölçerin yaptığı hata bulunur. Çalışmalarda su yoğunluğundaki değişimler çok düşük olduğundan sabit 1 gr/cm^3 alınmıştır. Çizelge 7.1 ve Şekil 7.1'deki kalibrasyon bilgilerine sahip sayacın kalibrasyon şartlarındaki doğruluk deneyi sonuçlarını Çizelge 7.2 ve Şekil 7.2'de gösterilmektedir. Kalibrasyon ve testler akış ölçer ağzında 1,50mm delik çapına sahip filtre (akış düzeltici amaçlı), akış ölçer önünde ve arkasında 200 mm düz boru olan bağlantı ile 20 °C referans su sıcaklığında yapılmıştır. Resim 7.1 filtreleri göstermektedir.



Resim 7.1. Filtreler

60 mm yatay eksen ses üreteç mesafesi, 19 mm iç çapı ve giriş çapı 25 mm sahip bir çapraz biçimli tüp modeline sahip akış ölçer ISO4064-1 standardına uyum ev tipi su sayacı olarak geliştirilmiştir [20]. Ses yolunun akış eksenine olan açısı 30°'dir. Bu standarda göre maksimum debi (Q_4) ve hassasiyet sınıfı (R) üretici tarafından

belirlenir. Tasarımda Q_4 7000lt/h ve R ise 160 olarak hedeflenmiştir. Hedeflenen Q_1 , Q_2 , Q_3 değerleri aşağıdaki eşitliklerden bulunmaktadır.

$$Q_3 = Q_4 / 1.25 = 5600 \text{ lt / h} \quad (7.1)$$

$$Q_1 = Q_3 / R = Q_3 / 160 = 35 \text{ lt / h} \quad (7.2)$$

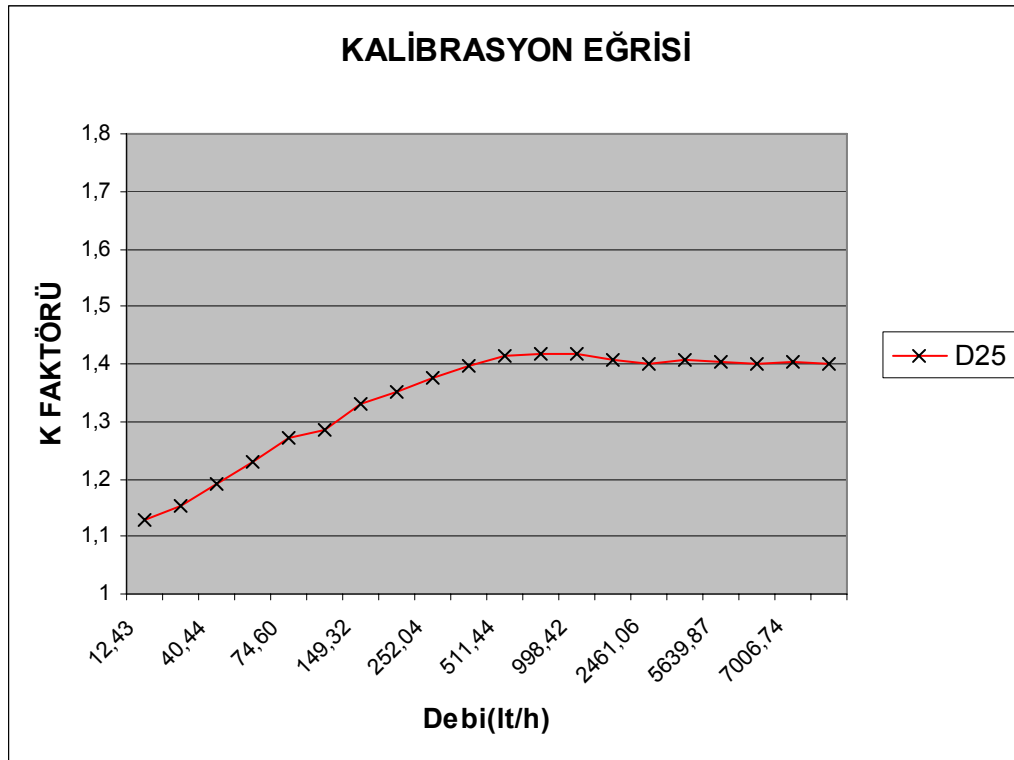
$$Q_2 = Q_1 * 1.6 = 56 \text{ lt / h} \quad (7.3)$$

Maksimum izin verilebilir hatalar Q_1 - Q_2 debileri arasında $\pm\%5$, Q_2 - Q_4 arasında $\pm\%2$ 'dir. Doğruluk testleri Q_1 'de, Q_1 ve Q_2 arasında bir noktada, Q_2 'de, Q_2 ile $Q_4/2$ arasında iki noktada, $Q_4/2$ 'de, Q_3 'de ve Q_4 'de yapılmaktadır.

Yapılan normal şartlardaki testlerde hata kabul edilebilir doğruluk sınırları içerisindedir. Bununla birlikte farklı su sıcaklıklarının etkisini ve akışkan hız profiline olan bağımlılığı gözlemlemek için sıcak/soğuk su testleri ve akış bozucu testleri ayrıca uygulanmış ve sonuçlar gözlemlenmiştir. Çizelge 7.3, Çizelge 7.4, Çizelge 7.5, Çizelge 7.6, Şekil 7.3, Şekil 7.4 bu deneyleri özetlemektedir.

Çizelge 7.1. Test edilen akış ölçerin kalibrasyon tablosu

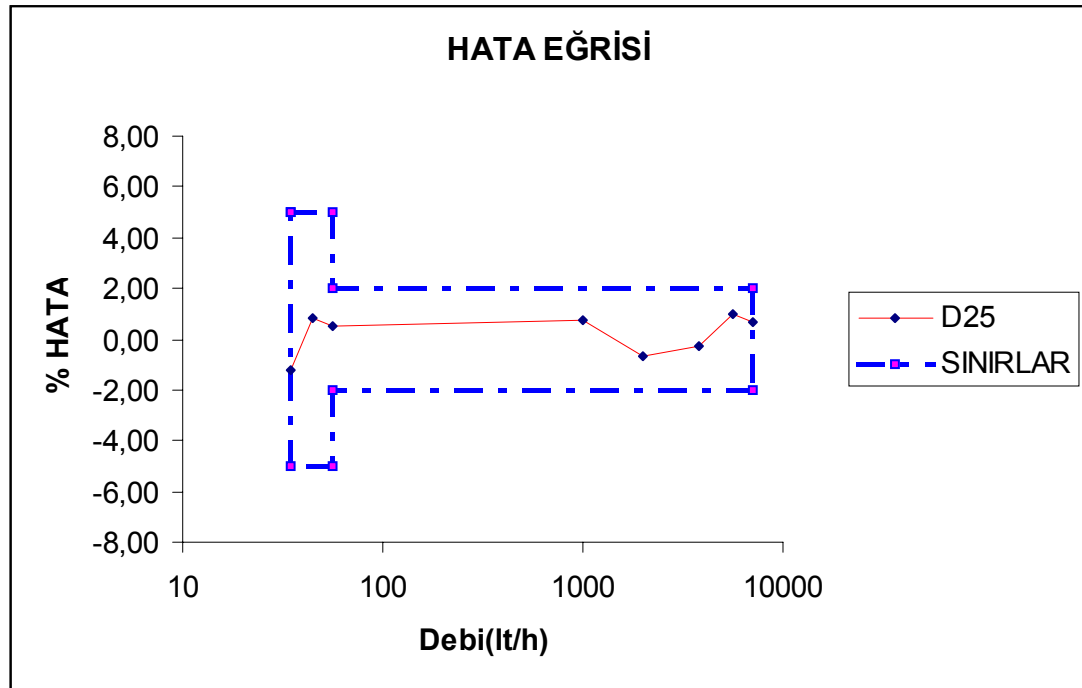
GERÇEK(l/h)	ÖLÇÜLEN(l/h)	K
12,43	11	1,13
29,52	25,6	1,153
40,44	33,9	1,193
49,32	40,1	1,23
74,60	58,6	1,273
101,44	79	1,284
149,32	112,1	1,332
199,80	148	1,35
252,04	183,3	1,375
351,07	251,3	1,397
511,44	361,7	1,414
702,41	495	1,419
998,42	705,1	1,416
1495,86	1062,4	1,408
2461,06	1757,9	1,4
4133,63	2937,9	1,407
5639,87	4017	1,404
6538,47	4667	1,401
7006,74	4987	1,405
7894,64	5635	1,401



Şekil 7.1. Test edilen akış ölçerin kalibrasyon eğrisi

Çizelge 7.2. Doğruluk testi sonuçları(20°C)

Gerçek Akış oranı $Q_{(i)}$ lt/h	Çalışma basıncı P_w bar	Çalışma sıcaklığı T_w °C	İlk okuma $V_i (i)$ lt	Son okuma $V_i (f)$ lt	Ölçülen miktar V_i lt	Gerçek miktar V_a lt	Akış ölçer hatası E_m %	İzin verilen en büyük hata %
35	5	19,8	5436,96	5440,44	3,48	3,522	-1,21	± 5
45	5	19,8	5440,44	5445,01	4,57	4,532	0,83	± 5
56	5	20,3	5445,01	5450,46	5,45	5,424	0,48	± 2
1000	5	20,1	5450,46	5554,12	103,66	102,872	0,76	± 2
2000	5	20,5	5554,12	5753,38	199,26	200,636	-0,69	± 2
3800	5	21	5753,38	6003,4	250,02	250,762	-0,30	± 2
5600	5	21	6003,4	6256,65	253,25	250,832	0,95	± 2
7000	5	21	6256,65	6509,22	252,57	250,796	0,70	± 2



Şekil 7.2. Doğruluk testi dağılımları(20°C)

Akış ölçerler verilen uygulamaya ve bütçeye göre en iyi performansı almak için seçilirler. Bu performans kullanımda bazı çevresel faktörler ve kurulum etkileri ile azalabilir. Sıcaklık ve nem gibi çevresel faktörler, elektronikteki EMC bozulmaları, transdüserlerin tıkanması, akışkanın özelliklerindeki değişimler (sıcaklık yoğunluk

basınç viskozite vs...), hava baloncukları ve hız profilindeki değişimler ultrasonik akış ölçerlerde hata ya sebep olan kaynaklardır.

Birçok akış ölçer her hangi bir alana monte edildiğinde, kalibre edildiği ve hatalarının telafi edildiği alandaki doğruluklarından farklı davranabilirler . Bu durum ses-ötesi akış ölçerler için de geçerlidir. Çünkü ses-ötesi akış ölçerler kurulum etkilerine karşı çok duyarlıdırlar.

Herhangi bir sayacın performansları kalibre edildiği yere benzeyen bir çevre içerisinde en iyidir. Kurulum etkileri akış ölçümünde çok bilinen bir kavramdır. Statik ve dinamik kurulum etkileri üzerine bir çok çalışmalar yapılmıştır. Kurulum etkilerinden dolayı oluşan hatalar akış ölçer tipine göre önemliliği değişir örneğin pozitif yer değiştirme esaslı akış ölçerler akış hız profilinden etkilenmez iken ultrasonik akış ölçerler çok duyarlıdırlar. Bu rağmen pozitif yer değiştirme esaslı akış ölçerlerin bazı dezavantajları vardır. Çıkışlarında vurutulu akış oluştururlar, dar bir akış oranı aralığında çalışırlar, pahalıdır, kolay arızalana bilir ve basınç kayıpları yüksektir [5].

Kurulum etkileri statik ve ya dinamik olabilir [21, 5, 15]. Akış alanındaki hızlı değişimlerden doğan etkiler veya akıştaki vurutular dinamik kurulum etkileri olarak anılır. Bu etkiler pompa yada kompresörlerden oluşmaktadır. Zamanla değişmeyen kurulum etkileri statik kurulum etkileri olarak anılır. Akış ölçerin monte edildiği borulardaki çap daralmaları, bükümler, boru eklemleri, akış dönüşleri (dirsekler), yarı açık valfler ve özellikle kalorimetrelerde kullanılan sıcaklık sensörleri akış profilini etkileyen statik etmenlerden bazılarıdır [22, 5, 15]. Bu etkileri azaltmak için akış-ölçerler genel olarak uzun düz bir borunun ortasına monte edilir veya akış düzenleyici mekanik aksamlar kullanılır. Ses-ötesi akış ölçerlerde alternatif olarak birden çok ses-ötesi transdüser çifti kullanımı (multiple paths) akışkanın hız profilinden bağımsız ölçüm yapma olanağı sunar [2, 23]. Bu yöntem ilk kez 1964 de Knapp tarafından ortaya atılmıştır.

Çoğu kez ölçümdeki hatanın kaynağının kurulum etkilerinden mi, yoksa akışkanın parametrelerindeki değişimlerden dolayı mı kaynaklandığını saptamak zordur. Ultrasonik akış ölçerlerin hata kaynakları arasında en baskın olanlar akış profilindeki düzensizlik, akışkan özelliklerindeki değişim ve transdüserlerdeki tıkanmalardır. Bir çok akış ölçer akışkanın hız profiline ve akışkan özelliklerindeki değişimlere bağlı olarak hata yaparlar.

Sistemde akış profilini bozmak için resim 7.2’de görülen akış bozucu elemanlar akış ölçer önünde 200mm mesafeye koyularak akış bozulma testleri filtreli ve filtresiz olarak gözlemlenmiştir.



Resim 7.2. Kullanılan akış bozucular

Çizelge 7.3. Akış ölçerin 200 mm önünde akış bozucu koyularak Q3’de yapılan doğruluk testleri (filtresiz)

Gerçek Akış oranı $Q_{(i)}$ lt/h	Çalışma basıncı P_w bar	Çalışma sıcaklığı T_w $^{\circ}\text{C}$	İlk okuma $V_i (i)$ lt	Son okuma $V_i (f)$ lt	Ölçülen miktar V_i lt	Gerçek miktar V_a lt	Akış ölçer hatası E_m %	İzin verilen en büyük hata %
5600	5	21	6412,24	6679,95	267,71	250,672	6,80	± 2
5600	5	21	6679,95	6946,48	266,53	250,714	6,31	± 2

Çizelge 7.4. Akış ölçerin 200 mm önünde akış bozucu koyularak Q3’de yapılan doğruluk testleri (filtreli)

Gerçek Akış oranı $Q_{(i)}$ lt/h	Çalışma basıncı P_w bar	Çalışma sıcaklığı T_w °C	İlk okuma $V_i (i)$ lt	Son okuma $V_i (f)$ lt	Ölçülen miktar V_i lt	Gerçek miktar V_a lt	Akış ölçer hatası E_m %	İzin verilen en büyük hata %
5600	5	21,2	6947,53	7199,23	251,7	250,702	0,4	± 2
5600	5	21,2	7199,23	7451,22	251,99	250,688	0,52	± 2

Su sıcaklığındaki değişimin yoğunluk ve viskozitedeki değişimin yanında ses hızında da bir değişime yol açtığını Bölüm 3.3.2’de gördük. Sıcaklığın büyük aralıklarda değişimi ses hızını büyük ölçüde etkilemektedir. Gazlarda ve sıvılarda ses hızı sıcaklığın bir fonksiyonudur. Bu nedenle sıcaklık değişimi geçiş zamanında bir değişime neden olur. Eş. 4.17’de bu etkiler minimize edilmiştir. Bu durumda ses hızı t_1 ve t_2 ölçümleri arasında sabit olduğundan eğer örnekleme frekansı yeteri kadar yüksek ise akış ölçümü hatası üzerinde etkiler önemsizdir. Eğer örnekleme frekansı ses hızının sabit olmasını garanti edecek kadar yüksek değil ise farklı algoritmalar ile bu etkiler telafi edilmelidir.

Bunun yanında sıcaklıktaki değişim büyük miktarlarda ise tüp kesitinde genişmeden dolayı kesitte bir değişiklik de meydana getirecektir. Sıcaklıktaki değişimi ayrıca hız profiline de etki etmektedir. Buda kalibrasyon eğrisinde bir değişime yol açar. Düzenli akışlar için hız profili paraboliktir ve hız profili basınç değişimi, dinamik viskozite ve akışkanın geçtiği kesite bağlı olarak değişir.

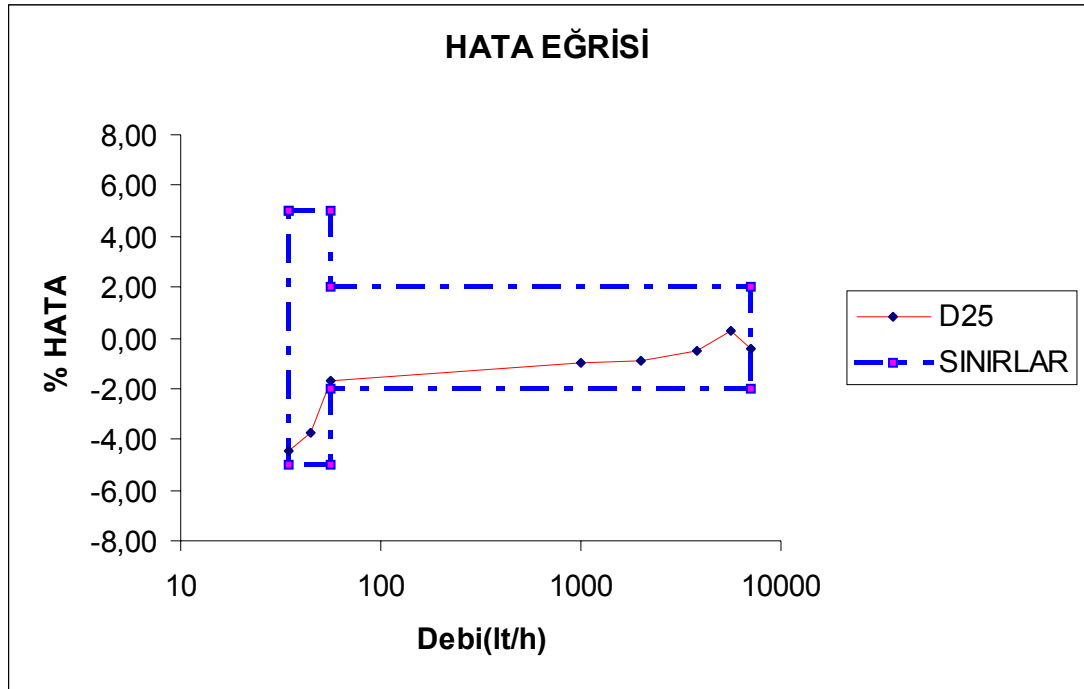
Dinamik viskozite ve tüp yarı çapı sıcaklığın bir fonksiyonudur. Bu nedenle farklı sıcaklıklarda ve basınçlarda hız profili de değişmektedir. Farklı sıcaklıklarda ayrı kalibrasyon eğrileri kullanarak ve bu sıcaklıklarda doğru kalibrasyon eğrisi seçilerek performans artırılabilir. Ayrıca kesit alanı tüp çapının karesinin bir fonksiyonu olduğundan, tüp çapındaki genişleme akış hızından akış oranına geçişte de bir hataya neden olur.

Sıcaklık değişimlerinden ve hız değişimlerinden dolayı akustik ışın yolları farklı yönlerde bükülmeler yaratacak ve faz ve genliklerde değişimler olacaktır. “Ray

Tracing” algoritmaları bir çok araştırmacı tarafından sıcaklık değişiminden dolayı oluşan ölçüm belirsizliğinin simülasyonu için kullanılmıştır.

Çizelge 7.5. Doğruluk testi sonuçları(35 °C)

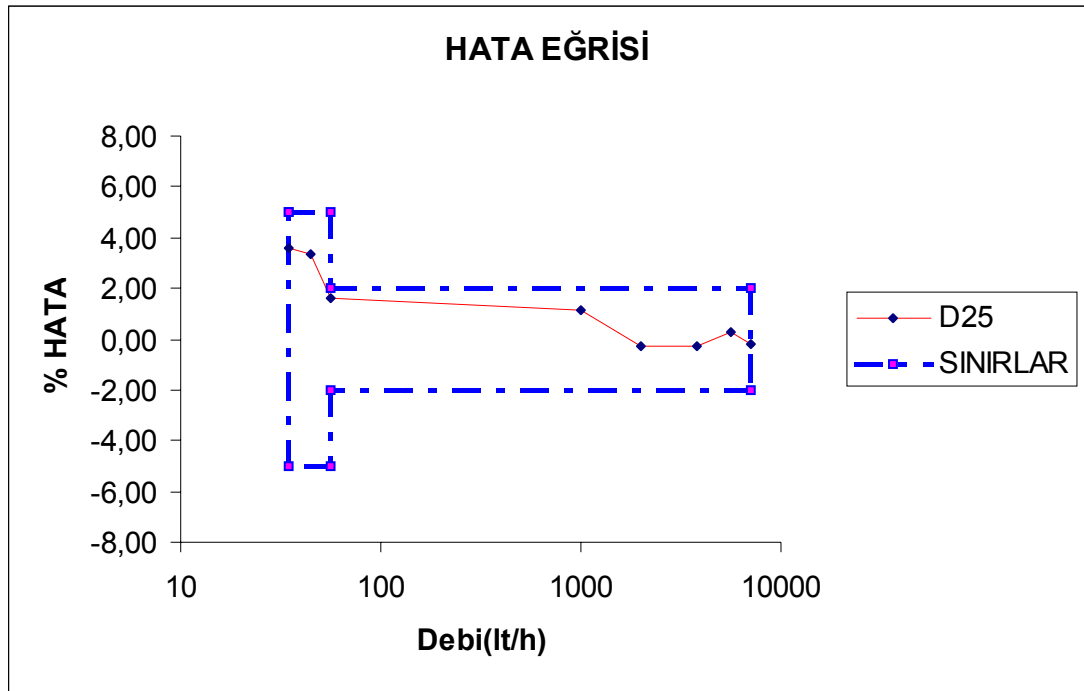
Gerçek Akış oranı $Q_{(i)}$ lt/h	Çalışma basıncı P_w bar	Çalışma sıcaklığı T_w °C	İlk okuma $V_i (i)$ lt	Son okuma $V_i (f)$ lt	Ölçülen miktar V_i lt	Gerçek miktar V_a lt	Akış ölçer hatası E_m %	İzin verilen en büyük hata (%)
35	5	34,9	9031,45	9034,81	3,36	3,51	-4,46	± 5
45	5	34,9	9034,81	9039,19	4,38	4,544	-3,74	± 5
56	5	34,9	9039,19	9044,65	5,46	5,554	-1,72	± 2
1000	5	35	9044,65	9144,65	100,00	101,024	-1,02	± 2
2000	5	35	9144,65	9343,39	198,74	200,586	-0,93	± 2
3800	5	34,9	9343,39	9592,77	249,38	250,744	-0,55	± 2
5600	5	35	9592,77	9844,19	251,42	250,736	0,27	± 2
7000	5	35,1	9844,19	10093,85	249,66	250,818	-0,46	± 2



Şekil 7.3. Doğruluk testi hata dağılımları(35 °C)

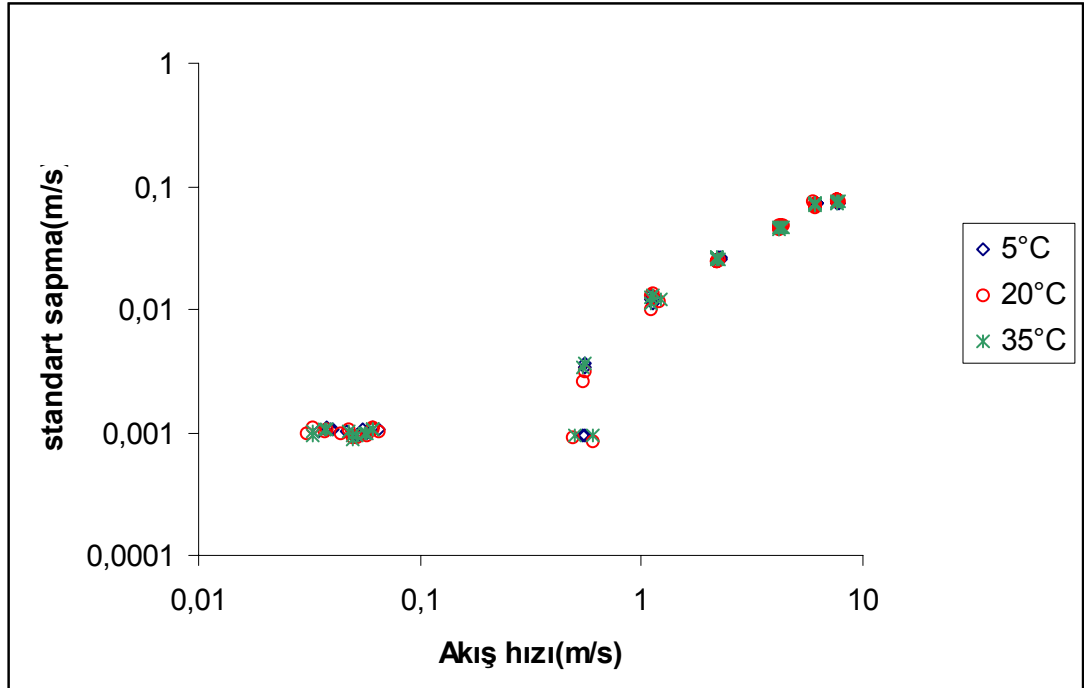
Çizelge 7.6. Doğruluk testi sonuçları(5 °C)

Gerçek Akış oranı $Q_{(i)}$ lt/h	Çalışma basıncı P_w bar	Çalışma sıcaklığı T_w °C	İlk okuma $V_i (i)$ lt	Son okuma $V_i (f)$ lt	Ölçülen miktar V_i lt	Gerçek miktar V_a lt	Akış ölçer hatası E_m %	İzin verilen en büyük hata (%)
35	5	34,9	10100,12	10103,78	3,66	3,53	3,55	± 5
45	5	34,9	10103,78	10108,52	4,74	4,58	3,38	± 5
56	5	34,9	10108,52	10114,13	5,61	5,52	1,60	± 2
1000	5	35	10114,13	10215,87	101,74	100,614	1,11	± 2
2000	5	35	10215,87	10415,99	200,12	200,732	-0,31	± 2
3800	5	34,9	10415,99	10665,99	250,00	250,728	-0,29	± 2
5600	5	35	10665,99	10917,51	251,52	250,78	0,29	± 2
7000	5	35,1	10917,51	11167,79	250,28	250,744	-0,19	± 2

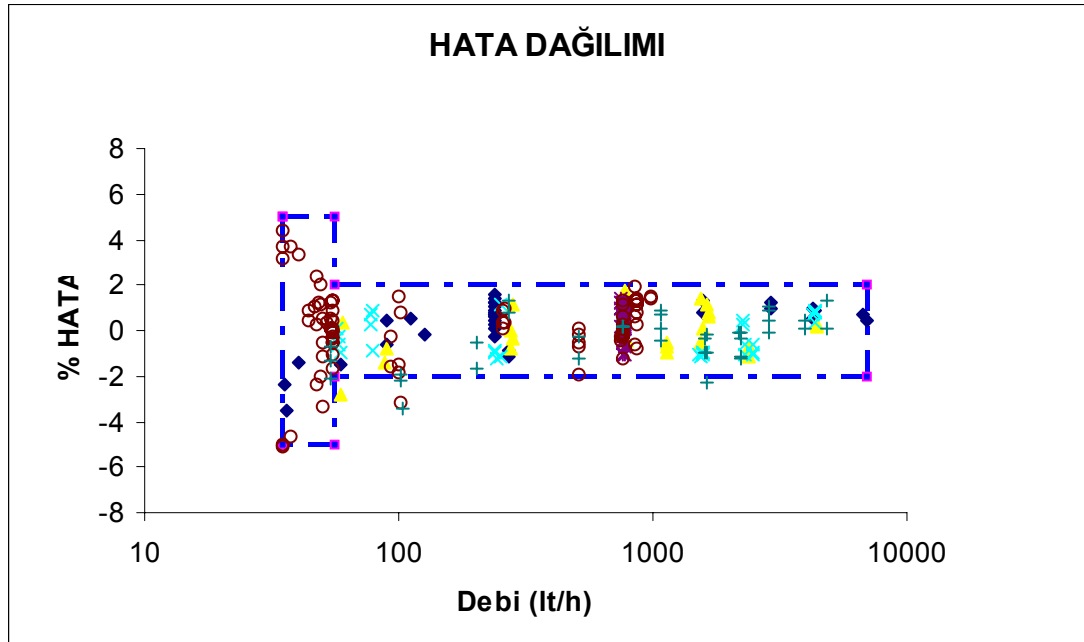


Şekil 7.4. Doğruluk testi hata dağılımları(5 °C)

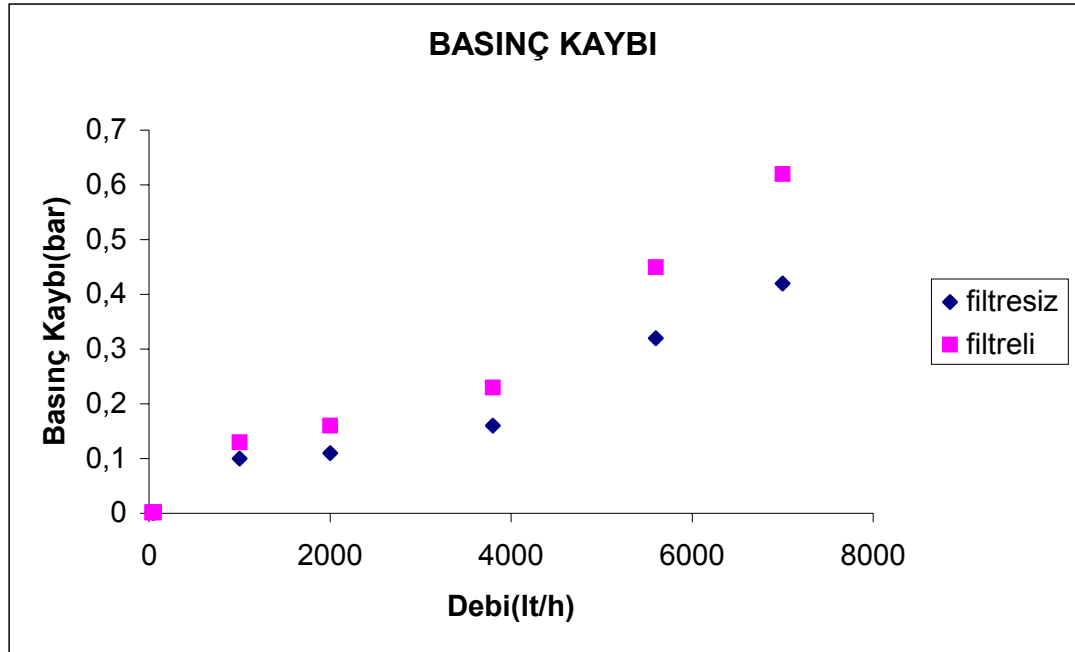
Ölçülen akış hızının standart sapması akışkanın hızındaki artış ile artmaktadır. Şekil 7.5 hesaplanan akış hızının standart sapmasının akış hızına göre değişimini göstermektedir. Şekil 7.6 farklı sıcaklıklarda gün aşırı yapılan doğruluk testlerinin dağılımını vermektedir. Akış oranına göre basınç kayıplarının değişimi akış ölçerin ağız girişinde filtre var iken ve yok iken Şekil 7.7’de görünmektedir. Resim 7.3 tasarımı yapılan akış ölçerin dış görünüşüdür.



Şekil 7.5. Hesaplanan akış hızının standart sapması



Şekil 7.6. Hata dağılımı



Şekil 7.7. Basınç kaybı değişimi



Resim 7.3. Akış ölçerin dış görünüşü

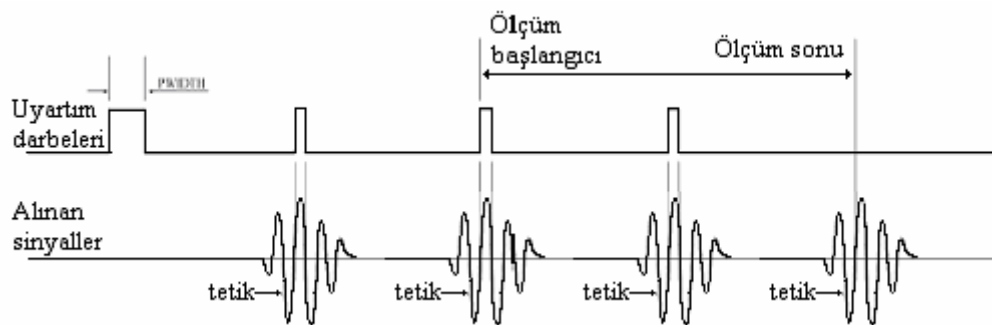
Ortalama akış hızı ses hızından çok daha düşük olduğu için t_1 ve t_2 sıfır akış geçiş zamanına oldukça yakın olacaktır (t_0).

$$t_0 = \frac{l}{c} \quad (7.4)$$

Bu nedenle transdüser mesafesi 60 mm ve 20 °C referans su sıcaklığında ses hızı Eş. 3.6'dan 1482m/s olduğundan iletim zamanı 41 us kadar olacaktır. t_0 'dan sıfır olmayan akıştaki geçiş zamanlarını ayırt etmeyi mümkün yapmak için geçiş zamanını çözünürlüğü Δt , t_0 dan çok daha küçük olmalıdır. Genel olarak Δt 100ps'dir. Bizim uygulamamızda bu zaman çözünürlüğü 50ps dir. Bu şekilde hız çözünürlüğü zaman çözünürlüğünün bir fonksiyonunu olarak Eş. 7.5 yardımıyla bulunur.

$$\Delta u \cong \frac{l \cdot \Delta t}{\cos \theta \cdot t_0^2} \quad (7.5)$$

Bu nedenle tipik hız çözünürlüğü 0,004 m/s akış oranı çözünürlüğü $11,2 \times 10^{-4}$ l/s'dir. Etrafa çınlama döngüsü kullanılarak bu çözünürlük arttırılabilir [24, 15]. Bu alınan darbenin hemen akabinde yani bir darbe yayınlamaktır böylelikle birden fazla sesin iletim süresinin ortalaması bulunur. Şekil 7.8 etrafa çınlama döngüsünü özetlemektedir.



Şekil 7.8. Etrafa çınlama döngüsü

Etrafa cınlama döngüsünde göndericiden alıcıya N adet darbenin gönderilmesi durumunda bu darbelerin toplam sürelerini t_{1N} yada t_{2N} 'i bulur. Bu durumda t_1 ve t_2 zamanları Eş. 7.6 ve Eş. 7.7 yardımıyla bulunur.

$$t_1 = \frac{t_{1N}}{N} \quad (7.6)$$

$$t_2 = \frac{t_{2N}}{N} \quad (7.7)$$

Bu metot ile tek darbenin zaman çözünürlüğü hala aynıdır fakat hız ve akış oranı çözünürlüğü N kat artmaktadır. Toplam atılan darbe sayılarının ortalamasının alınması ile elde edilen zaman çözünürlüğü $50\text{ps}/N$ olur. Bu nedenle $N=10$ alınarak akış oranı çözünürlüğü $11,2 \times 10^{-5} \text{l/s}$ olacaktır. N değerinin arttırılması ile bu çözünürlüğü istenen seviyeye çıkartmak mümkündür.

8. TARTIŞMA VE SONUÇLAR

Bu çalışma son zamanlarda kullanımı giderek yaygınlaşan geçiş zamanı esaslı bir ultrasonik akış ölçer tasarımı sunulmuş, kalibrasyon ve test aşamalarından bahsedilmiştir.

Akış hızı ses hızından oldukça küçük olduğu için geçiş zamanı esaslı ultrasonik akış ölçerlerde akış hızının sesin iletim süresine olan etkisi çok azdır. Özellikle düşük akış oranlarında “akış yönünde geçiş zamanı” ile “ters akış yönünde geçiş zamanı” arasındaki fark oldukça küçüktür. Bu nedenle geçiş zamanlarının hassas bir şekilde ve yüksek çözünürlükte ölçülmesi özellikle düşük akış oranları için çok önemlidir.

Kararlı ölçüm için gömülü elektronik her türlü elektromanyetik bozulmalara karşı korunmalıdır. Ortam sıcaklığın değişimi osilatörler ve devre elemanlarında da karakteristik değişimlerine neden olabilir. Bu nedenle devrede kullanılacak elemanlarının sıcaklık karakteristikleri çok iyi olmalıdır yada tasarım bu etkileri telafi edebilmelidir. Ayrıca ortamdaki nemin elektronik kısma teması engellenmelidir.

Çeşitli nedenlerden dolayı akışkan sıcaklığındaki değişimler sayaç doğruluğuna etki etmektedir. Su sıcaklığındaki değişimin yoğunluk ve viskozitedeki değişimin yanında ses hızında da bir değişime yol açmaktadır. Sıcaklığın büyük aralıklarda değişimi ses hızını büyük ölçüde etkilemektedir. Gazlarda ve sıvılarda ses hızı sıcaklığın bir fonksiyonudur. Bu nedenle sıcaklık değişimi geçiş zamanında bir değişime neden olur. Eş. 4.17’de bu etkiler minimize edilmiştir. Bu durumda ses hızı t_1 ve t_2 ölçümleri arasında sabit olduğundan eğer örnekleme frekansı (t_1 örnekleme zamanı ve t_2 örnekleme zamanı arasındaki zaman aralığı) yeteri kadar yüksek ise akış ölçümü hatası üzerinde etkiler önemsizdir. Eğer örnekleme frekansı ses hızının sabit olmasını garanti edecek kadar yüksek değil ise farklı algoritmalar ile bu etkiler telafi edilmelidir.

Bunun yanında sıcaklıktaki değişim büyük miktarlarda ise tüp kesitinde genleşmeden dolayı kesitte bir değişiklik de meydana getirecektir. Tüp çapındaki genleşme akış hızından akış oranına geçişte de bir hataya neden olur.

Sıcaklıktaki değişimi ayrıca hız profiline de etki etmektedir. Düzenli akışlar için hız profili paraboliktir ve hız profili basınç değişimi, dinamik viskozite ve akışkanın geçtiği kesite bağlı olarak değişir. Dinamik viskozite ve tüp yarı çapı sıcaklığın bir fonksiyonudur. Bu nedenle farklı sıcaklıklarda ve basınçlarda hız profili de değişmektedir. Bu nedenle yapılan testlerde düzenli akışlar için, sıcaklık artışı doğruluk eğrisini aşağı yönde kaydırır iken sıcaklıktaki düşüş doğruluk eğrisini yukarı yönde kaydırıldığı gözlemlenmiştir.

Yapılan çalışmada 5°C-35°C aralığında sayaç doğruluğunun kabul edilebilir sınırlar içerisinde kaldığı gözlemlenmiştir. Eğer daha büyük sıcaklık aralıklarında çalışılması gerekiyor ise sıcaklığının ölçümü daha güvenilir akış ölçümü sağlayacaktır.

Ultrasonik akış ölçerler akış profilindeki değişimlere karşı çok duyarlıdır. Bu değişimin kaynağı girişindeki çap daralmaları, borulardaki bükümler, boru eklemleri ve özellikle kalorimetrelerde kullanılan sıcaklık sensörler, pompa vuruntusu gibi kurulum etkileri yada farklı bir akış bozucu etkisi olabilir. Bu etmenler kurulum etkileri olarak adlandırılır ve akış ölçümünde çok bilinen kavramlardır. Kurulum etkileri yapılan tasarımda karşılaşılan en baskın hata kaynağı olarak gözlemlenmiştir. Uzun düz bir borunun ortasına montaj yapılması yada akış düzenleyici mekanik aksamlar kullanılması bu etkilere karşılık genel olarak kabul görmüş yöntemlerdir. Bu nedenle uygulamada akış düzenleyici filtreler kullanılmıştır. Kurulum etkisi olarak sayacın 200mm önüne yerleştirilen akış bozucu ile testler yapılmış ve test sonuçları akış düzenleyicili ve akış düzenleyicisiz olarak gözlemlenmiştir. Deney sonuçlarında akış düzenleyici filtre kullanımının akış bozucudan kaynaklanan hatanın üstesinden geldiği görülmüştür. Ultrasonik akış ölçerlerde alternatif olarak birden çok ultrasonik transdüser çifti kullanımı (multiple paths) akışkanın hız profilinden ve dolayısıyla kurulum etkilerinden bağımsız ölçüm yapma olanağı sunar. Fakat bu yöntem hem maliyeti arttırmakta hem de iç çapı 100 mm den büyük

akış ölçerler tarafından kullanılmaktadır. Alternatif bir yol olarak Carlender doktora tezinde ultrasonik akış ölçümü için kurulum etkilerini ve bu etkileri otomatik tanımlamayı açıklamıştır ve ölçüm gürültüsünü analiz ederek kurulum etkilerini saptamanın mümkün olduğunu göstermiştir [15, 25]. Ölçümdeki gürültü seviyesini saptamak için ölçülen standart sapmayı geçerli bir referans seviye ile karşılaştırarak türbülanslı ölçmekte kullanılabilir. Böylece bir tanımlama prosedürü akış ölçerde tanımlanabilir. Carlender bu metodu kendinden tanımlama olarak adlandırmıştır. Akış düzenleyici filtre kullanımı kurulum etkilerini büyük ölçüde azalttığından dolayı tasarımda bu tür algoritmalar kullanılmamıştır. Buna benzer olarak Berrebi vuruntulu akışların saptanmasının mümkün olduğunu göstermiştir [2, 15]. Yapılan testlerin tamamında akış kaynağı bir pompa düzeneği kullanılmıştır. Dolayısıyla testlerin tamamı vuruntulu akış altında gerçekleştirilmiş. Akış oranından saniyede 1 kez alınan örnek ile sayacın doğruluk sınırları içerisinde kaldığı görülmüştür. Yapılan çalışmada basınç değişiminin doğruluk üzerinde etkisi gözlemlenememiştir.

Yapılan çalışmada teorik olarak $11,2 \times 10^{-4}$ l/s akış oranı ölçüm çözünürlüğü elde edilmiştir. Bu çözünürlük bir çok uygulama için yeterlidir.

KAYNAKLAR

1. Johnson, D., "Measuring Process Fluid Flow: an Exacting Task", *Control Engineering*, 52 (6): 59-65(2005)
2. Berrebi, J., "Self-Diagnostic Techniques and Their Applications to Error Reduction for Ultrasonic Flow Measurement", Doktora Tezi, *Luleå University of Technology Department of Computer Science and Electrical Engineering, Division of EISLAB*, Luleå, 5-6, 6-7, 7, 13-15, 13-15, 13-15, 31, 27-55 (2004)
3. Miller, R. W., "Flow Measurement Engineering Handbook 2nd ed.", *Mc Graw-hill publishing company*, Newyork, 6.1-6.4, 6.5, 6.16-6.17, 6.21-6.23, 6.22 (1989)
4. Wilson, M. P., "Historical Perspective", Flow Measurement 2nd ed., D. W. Spitzer, *Research Triangle Park*, North Carolina, 5-27 (2001)
5. C. Carlander, "Installation effect and Self-Diagnostics for Ultrasonic Flow Measurement", Doktora Tezi, *Luleå University of Technology Department of Computer Science and Electrical Engineering, Division of EISLAB*, Luleå, 1-11, 13-46, 13-46, 13-46 (2001)
6. Deventer, J. V., "Material Investigations and Simulation Tools Towards a Design Strategy for an Ultrasonic Densitometer", Doktora Tezi, *Luleå University of Technology Department of Computer Science and Electrical Engineering, Division of EISLAB*, Luleå, 25-32 (2001)
7. internet : Rochester Institute of Technology "Research : Biomedical Imaging Laboratory/Ultrasound" <http://www.cis.rit.edu/research/ultrasound/ultraintro.htm> (2006)
8. Sinclair, I. R., "Sensors and Transducers 3rd ed", *Oxford [England]*, Newnes, 116-119, 119-141 (2001)
9. Karagöz, İ., Eroğul, O., "Tıbbi Görüntüleme Sistemleri", *Haberal Eğitim Vakfı*, Ankara, 226-228, 229, 250 (1998)
10. internet : Department of Physics of University of Kuopio "Laboratory Works in Physics 1, part 1 : Ultrasound Echo Imaging" <http://venda.uku.fi/opiskelu/kurssit/FLT1/tyoohjeet/uswork.pdf> (2002)
11. Weber, J. F., "Ultrasonic Beam Propagation in Turbulent Flow", Doktora Tezi, *Faculty of the Worcester Polytechnic Institute*, Worcester, 8-32 (2003)

12. Moore, P. I., Piomelli, U., Johnson, A. N., Espina, P. I., "Simulations of Ultrasonic Transit Time in a Fully Developed Turbulent Flow using a Ray-Tracing Method", *20th North Sea Flow Measurement Workshop*, St Andrews, 1-8 (2002)
13. Asher, R. C., "Ultrasonic Sensors", *Institute of Physics Publishing*, Bristol, 227 (1997)
14. Godley, A., "Flow measurement in partially filled closed conduits", *Flow Measurement and Instrumentation*, 13 (5): 197–201 (2002)
15. Sarker, S. D., "Detection of Flow Pulsation by Micro-controller Equipped Ultrasonic Flow Meter", Yüksek Lisans Tezi, *Luleå University of Technology Department of Computer Science and Electrical Engineering, Division of EISLAB*, Luleå, 7-9, 7, 9, 10, 7, 10, 11-12 (2006)
16. Parr, E. A., "Endüstriyel Kontrol El Kitabı Cilt 1: Transdüserler 3rd ed.", *Milli Eğitim Bakanlığı*, İstanbul, 153-154, 154, 155-156, 157-160 (2002)
17. Carlson, J., "Ultrasonic Characterization of Materials and Multiphase Flows", Doktora Tezi, *Luleå University of Technology Department of Computer Science and Electrical Engineering, Division of EISLAB*, Luleå, 7-9, 7, 6-7, 6, 7 (2002)
18. Brassier, P., Hosten, B., Vulovic, F., "High-Frequency Transducers and Correlation Method to Enhance Ultrasonic Gas Flowmetering", *Flow Measurement and Instrumentation*, 12 (3): 201-211 (2001)
19. Türk Standartları Enstitüsü (TSE). "Isı Ölçerler - Bölüm 3: Veri Değişimi ve Ara Yüzler": TS EN 1434-3 (2003)
20. International Organization for Standardization (ISO). "Measurement of water flow in fully charged closed conduits – Meters for cold potable water and hot water Part 1: Specifications": ISO 4064-1 (2005)
21. Berrebi, J., Martinson, P. E., Willatzen, M., Delsing, J., "Ultrasonic flow Metering errors due to pulsating flow", *Flow Measurement and Instrumentation*, 15 (3): 179–185 (2004)
22. Ruppel, C., Perters, F., "Effects of upstream installations on the reading", *Flow Measurement and Instrumentation*, 15 (3): 167–177 (2004)
23. Moore, P. I., Brown, G. J., Stimpson, B. P., "Ultrasonic transit-time flowmeters modelled with theoretical velocity profiles: methodology", *Measurement Science and Technology*, 11(12): 1802-1811 (2000)

24. J. Delsing, "The zero-flow performance of a sing-around ultrasonic flowmeter", *Flow Measurement and Instrumentation*, 2 (4): 205-208 (1991)
25. Carlender, C., Delsing J., "Installation effect on an ultrasonic flow meter with implications for self diagnostics, *Flow Measurement and Instrumentation*, 11 (2):109-122 (2000)

EKLER

EK-1. main.c fonksiyonları

```

#include <p18f4520.h>
#include "global.h"
#include "main.h"

void init_pic(void)
{
    OSCCON = 0x70;
    OSCTUNE = 0x80;
    RCONbits.IPEN = 1; //öncelikli int açık

    INTCON = 0x40;
    INTCON2 = 0x80;
    INTCON3 = 0x00;

    PIR1=0x00;
    PIR2=0x00;

    PIE1=0x00;
    PIE2=0x00;

    IPR1=0x00;
    IPR2=0x00;

    ADCON0 = 0x00;
    ADCON1 = 0x1E;
    ADCON2 = 0x88;

    // PORT init.
    PORTA = 0b00101111;
    PORTB = 0b11110111;
    PORTC = 0b01100000;
    PORTD = 0b00010000;
    PORTE = 0b00000000;

    TRISA = 0b00001111;
    TRISB = 0b11111111;
    TRISC = 0b10010001;
    TRISD = 0x00000000;
    TRISE = 0b00000000;

    //-----spi init-----
    init_acam();
    //-----int0 h.piriority (acam isr)-----
    INTCONbits.INT0IF = 0;
    INTCONbits.INT0IE = 1;
    INTCON2bits.INTEDG0 = 0; //falling edge

    //-----ACAM time out-----
    PIR2bits.TMR3IF = 0;
    PIE2bits.TMR3IE = 1;
    IPR2bits.TMR3IP = 0;
    T3CON = 0x00; //1ms
    TMR3H = 0xf0;
    TMR3L = 0x5f;

    //-----uart-----
    PIR1bits.RCIF = 0;
    init_uart (); //2400 baudrate
    PIE1bits.RCIE = 1; //receiver interrupt on
    IPR1bits.RCIP = 1;
    //-----RTC-----
    init_rtc();
}

void init_uart(void)
{
    BAUDCONbits.BRG16=0; //8 bit SPRG

```

EK-1. (Devam) main.c fonksiyonları

```

TRISCBits.TRISC7=1;           //UART RX
TRISCBits.TRISC6=1;           //UART TX
TXSTABits.TX9=1; //9 bit Transmission
RCSTABits.RX9=1;           //9 bits receive
TXSTABits.SYNC=0;           //Asynchronous mode
TXSTABits.BRGH=0;           //low speed
SPBRG=103;
RCSTABits.SPEN=1;           //Serial Port Enabled
TXSTABits.TXEN=1;           //Transmit Enabled
RCSTABits.CREN=1;           //Receive Enabled
}

void init_rtc(void)
{
    PIR1bits.TMR1IF = 0;
    IPR1bits.TMR1IP=1;
    PIE1bits.TMR1IE = 1; // timer1 int enable...
    TMR1H = 0x80; // RTC timer...
    TMR1L = 0x00; // 1 saniye...

    T1CON = 0x0f; // two 8 bit 0x0f
                // Timer1 osc enable
                // don't synchronize ext clk input
                // Ext clk from pin RC0
                // Timer1 ON...
}

void delay_msx_in_idle (unsigned int sure)//wait in idle mode until irq ufo
{
    T3CONbits.TMR3ON = 1;
    timer3 = 0;
    while (!irq_acam)
    {
        if (T3CONbits.TMR3ON)
        {
            OSCCONbits.IDLEN=1;
            PIR2bits.TMR3IF = 0;
            _asm sleep_endasm
            _asm nop_endasm
        }
        if ((timer3)>=sure)
        {
            break;
        }
    }
    OSCCONbits.IDLEN=0;
    T3CONbits.TMR3ON = 0;
    TMR3H = 0xf0;
    TMR3L = 0x5f;
    PIR2bits.TMR3IF = 0;
}

void wakeup (void)
{
    ANALOG_VCC_ON();
    CHARGE_PUMP_ENABLE();
}

void sleep (void)
{
    ANALOG_VCC_OFF();
    UNSELECT_TRANSDUCE();
    CHARGE_PUMP_DISABLE();
    PORTE=0;//eeprom disable

    if (byte_received==TRUE)
        OSCCONbits.IDLEN=1; //main clock running but cpu off
    else {

```

EK-1. (Devam) main.c fonksiyonları

```

    OSCCONbits.IDLEN=0;//main clock disable
    BAUDCONbits.WUE=1;//auto wake up on received firs byte by uart
}
_asm sleep_endasm
_asm nop_endasm
}

void update_display(void)
{
    if (avg_flow<0)
        display_signals_2.j=1;
    else
        display_signals_2.j=0;
    display_long((long)total,(long)total);
}

void power_fail (void)//battery state control
{
    unsigned int ad=0,adh=0;
    BATTERY_MEAS_EN();
    ADCON0bits.ADON=1;
    ADCON0bits.GO=1;
    while(ADCON0bits.GO);
    ad=ADRESL;
    adh=ADRESH;
    ad+=(adh<<8);

    if (ad>600)
        display_signals_2.battery_low=0;
    else
        display_signals_2.battery_low=1;

    BATTERY_MEAS_DIS();
    ADCON0bits.ADON=0;
}

void calc_ins_flow(void)
{
    shg = ((transit_time_up + transit_time_down) / 2.0) - shg_cal;
    velocity= L_div_2 * ( 1 / (transit_time_down - shg - skg) - 1 / (transit_time_up - shg));
    flow=velocity*CROSS_SECTION;
}

void calculate_measurement(void)
{
    unsigned char t;
    if (flow<0.0)//reverse flow
    {
        negative_flag=0x55;
        flow=-1.0*flow;
    }else
        negative_flag=0;

    if (current_state != STATE_CALIBRATION )
    {
        look_up_value=look_up(flow);
        flow=look_up_value*flow;
    }

    if (flow>MAX_FLOW_RATE && current_state!=STATE_CALIBRATION)
    {
        flow=MAX_FLOW_RATE;
    }

    if (negative_flag) flow=-1.0*flow;

    for (t=0;t<9;++t)
        flows[t]=flows[t+1];
}

```

EK-1. (Devam) main.c fonksiyonları

```

flows[9]=flow;
total_flow=0;
for (t=0;t<10;++t)
    total_flow+=flows[t];
avg_flow=total_flow/10.0;

if (avg_flow>2.0 || avg_flow<-2.0)
{
    display_signals_2.no_flow=0;

    if (current_state!=STATE_CALIBRATION) current_state = STATE_MEASURE;
    double_total=double_total+avg_flow/36.0;
    if (double_total>1.0)
    {
        unsigned long long_deger=0;
        long_deger=(unsigned long)double_total;
        double_total=double_total-(double)long_deger;
        totall+=long_deger;

        if (totall>99)
        {
            unsigned char degerl=0;
            unsigned long degerh=0;
            degerl=totall % 100;
            degerh=(totall-degerl)/100;
            totall=totall-degerh*100;
            total=total+degerh;//
            if (total>999999999) total=0;
        }
    }
    else if (double_total<-1.0)//revers flow existing
    {
        signed long long_deger=0;
        long_deger=(signed long)double_total;
        double_total=double_total-(double)long_deger;
        totall+=long_deger;
        if (totall<0)
        {
            unsigned char degerl=0;
            unsigned long degerh=0;
            totall=~totall+1;
            degerl=totall % 100;
            degerh=(totall-degerl)/100;
            totall=totall-degerh*100;
            total=total-degerh-1;
            totall=-1*totall;
            totall+=100;
            if (total<=0)
            {
                total=0;
                totall=0;
            }
        }
    }
}
} else //akış yoksa
{
    display_signals_2.no_flow=1;
    if (current_state == STATE_MEASURE ) current_state = STATE_EEPROM;
}

}

void hardware_test(void)
{
    long data;

    data=0x5555;
    ext_write_sequential(638,4,&data);
    data=0x0000;

```

EK-1. (Devam) main.c fonksiyonları

```

    ext_read_sequential(638,4,&data);
    if (data!=0x5555)
    {
        display_signals_1.h_up=1;
    }

    if (!test_gp2())
        display_signals_1.W=1;

    power_fail ();
}

void Send_abyte_to_Uart (unsigned char byte) // for even Prity
{
    unsigned char even,k;
    even=0;
    for(k=0;k<8;k++)
    {
        if( (0x01 & (byte>>k)) )
            even++;
    }
    if( (even % 2) == 1 )
        TXSTAbits.TX9D=1;
    else
        TXSTAbits.TX9D=0;
    RCSTAbits.CREN=0;          //Receive Enabled
    TXREG=byte;
    while(!TXSTAbits.TRMT);
    RCSTAbits.CREN=1;          //Receive Enabled
}

void send_ack(void)
{
    Send_abyte_to_Uart ('O');
    delay10msx(5);//-----
    Send_abyte_to_Uart ('K');
}

void main(void)
{
    unsigned char tick;

    delay10msx(50);
    for (tick=0;tick<10;tick++)
        flows[tick]=0;

    RTC_flags.f = 0;

    init_pic();
    init_lcd();
    Clear_Lcd ();

    reset_acam();
    display_signals_1.byte=0xc0;
    display_signals_2.byte=0x00;

    hardware_test();

    Display_Versiyon();
    delay10msx(50);

    INTCONbits.GIE = 1;
    delay10msx(10);

    read_values_from_eeprom();

    current_state = STATE_IDLE;
    byte_received=FALSE;
}

```

EK-1. (Devam) main.c fonksiyonları

```

hfc_fail=0x55;
rtc_ok=0;
irq_acam=0;
tick=0;

while(1)
{
    if(rtc_ok && current_state != STATE_QUERY)
    {
        display_signals_2.rf=1;
        display_signals_2.error=0;
        wakeup();

        if(++tick>30 || hfc_fail)
        {
            irq_acam=0;
            start_hfclk_cal();
            delay_msx_in_idle(20);
            if(irq_acam)
                calc_acam_hfconst ();
            else
            {
                hfc_fail=0x55;
                display_signals_2.error=1;
            }
            tick=0;
        }
        if (!hfc_fail)
        {
            inst_avg_flow=0.0;
            avg_count=0;
            measure_count=0;
            while (measure_count<MEASUREMENT_COUNT_PER_SEC)
            {
                measure_count++;
                irq_acam=0;
                start_cycle(UP_STREAM);
                delay_msx_in_idle (20);
                if(irq_acam)
                {
                    transit_time_up=calc_acam_transmit_time();
                    if ((transit_time_up<TIME_LIMIT_UPPER) && (transit_time_up>TIME_LIMIT_LOWER))
                    {
                        aborted = 0x00;
                    }else
                    {
                        aborted = 0x55;
                        display_signals_2.rf=0;
                    }
                }
            }
            else
            {
                aborted = 0x55;
                display_signals_2.error=1;
            }
            if (!aborted)
            {
                irq_acam=0;
                start_cycle(DOWN_STREAM);
                delay_msx_in_idle (20);
                if(irq_acam)
                {
                    transit_time_down=calc_acam_transmit_time();
                    if ((transit_time_down<TIME_LIMIT_UPPER) && (transit_time_down>TIME_LIMIT_LOWER))
                    {
                        aborted = 0x00;
                    }else
                    {

```

EK-1. (Devam) main.c fonksiyonları

```

        aborted = 0x55;
        display_signals_2.rf=0;
    }
}
else
{
    aborted = 0x55;
    display_signals_2.error=1;
}
}

if (!aborted)
    if (((transit_time_up-transit_time_down)>-600.0) && ((transit_time_up-transit_time_down)<600.0))
    {
        aborted = 0x00;
    } else {
        aborted=0x55;
        display_signals_2.rf=0;
    }

if (!aborted)
{
    calc_ins_flow();
    if ((flow<9000.0) && (flow>-9000.0))
    {
        ++avg_count;//correct measurement count
        inst_avg_flow+=flow;//for average flow
    }
    else
        flow=0.0;
}
} //while
ANALOG_VCC_OFF();
if (avg_count==MEASUREMENT_COUNT_PER_SEC)
{
    flow=(double)inst_avg_flow/(double)avg_count;
    calculate_measurement();
    if (current_state==STATE_CALIBRATION && calibration_mode==CALIBRATION_MODE_3)
    {
        static unsigned char cal_count=0;
        static double skg_temporary=0.0,shg_cal_temporary=0.0;
        ++cal_count;
        skg_temporary+=(transit_time_up - transit_time_down);
        shg_cal_temporary+=(transit_time_up + transit_time_down) / 2.0;
        if(cal_count>=5)
        {
            skg=skg_temporary/(double)cal_count;
            shg_cal=shg_cal_temporary/(double)cal_count;
            cal_count=0;
            skg_temporary=0.0;
            shg_cal_temporary=0.0;
            finalize_calibration();
        }
    }
} else
    display_signals_2.no_flow=1;
} //hfc_fail

update_display();

if (current_state == STATE_EEPROM)
{
    save_values_to_eeprom ();
    current_state = STATE_IDLE;
}

}

irq_acam = 0;
rtc_ok=0;

```

EK-1. (Devam) main.c fonksiyonları

```

    if (byte_received==TRUE)
    {
        if (current_state==STATE_CALIBRATION )
        {
            if (calibration_mode == CALIBRATION_MODE_2)
            {
                if(++state_error_count>1200)
                {
                    state_error_count=0;
                    finalize_calibration();
                }
            }
            else if (calibration_mode != CALIBRATION_MODE_3)
            {
                if(++state_error_count>2)
                {
                    state_error_count=0;
                    finalize_calibration();
                }
            }
        }
        else {
            if(++state_error_count>2)
            {
                state_error_count=0;
                byte_received=FALSE;
            }
        }
    }
} //rtc ok

if(RCSTAbits.OERR)
{
    RCSTAbits.CREN=0;
    RCSTAbits.CREN=1;
    temp=RCREG;
}

if (aborted || hfc_fail)
{
    reset_acam();
    hfc_fail=0x55;
    irq_acam = 0;
}
sleep();
}

//LOW priority int vector
/*extern*/
void low_isr(void);

#pragma code low_vector = 0x18
void interrupt_at_low_vector(void)
{
    _asm goto low_isr_endasm
}
#pragma code

#pragma interruptlow low_isr
void low_isr(void)
{
    INTCONbits.PEIE = 0;
    if (PIR2bits.TMR3IF)
    {
        TMR3H = 0xf0;
        TMR3L = 0x5f;
        ++timer3;
        PIR2bits.TMR3IF = 0;
    }
}

```

EK-1. (Devam) main.c fonksiyonları

```

    INTCONbits.PEIE = 1;
}

// High priority interrupt vector
void high_isr (void);

#pragma code high_vector = 0x08
void interrupt_at_high_vector (void)
{
    _asm
        goto high_isr //jump to interrupt routine
    _endasm
}

// High priority interrupt routine
#pragma code

#pragma interrupt high_isr

void high_isr ()
{
    INTCONbits.GIE = 0;
    if (INTCONbits.INT0IF)
    {
        isr_acam();
        irq_acam=55;
        INTCONbits.INT0IF = 0;
    }else
    if(PIR1bits.TMR1IF)
    {
        TMR1H = 0x80;// 1 saniyede 1 ölçüm yap RTC timer...
        TMR1L = 0x00;
        PIR1bits.TMR1IF = 0;
        rtc_isr();
        rtc_task();
        rtc_ok=1;
    }else
    if(PIR1bits.RCIF)
    {
        unsigned char i;
        if (BAUDCONbits.WUE)
        {
            RCSTAbits.CREN=0;    //Receive Enabled
            delay1msx(3);
            delay10usx(25);
            RCSTAbits.CREN=1;    //Receive Enabled
        }
        byte_received=TRUE;

        state_error_count=0;
        PIE1bits.RCIE=0;
        for (i=0;i<10;++i)
            in_data[i]=in_data[i+1];
        in_data[10]=RCREG;

        if (current_state==STATE_CALIBRATION)
            apply_cal();
        else//cal mod değilse
        if (in_data[7]!='E' && in_data[10]!='>' && (in_data[8]!='M' || in_data[8]==sayac_adres))
        {
            current_state=STATE_CALIBRATION;
            display_signals_1.I_of_k=1;
            display_signals_1.c_of_k=1;
            chksum_calculated=0;
            if (in_data[8]==sayac_adres)
            {
                calibration_type=ADRESS;
                send_ack();
            }
        }
    }
}

```

EK-1. (Devam) main.c fonksiyonları

```
    }
    else
        calibration_type=GENERAL;
    select_kalibration_mode();

}

if(RCSTAbits.OERR)//overrun hatası
{
    RCSTAbits.CREN=0;
    RCSTAbits.CREN=1;
    temp=RCREG;
}
PIE1bits.RCIE=1;

}
INTCONbits.GIE = 1;
} //high int.
```

EK-2. main.h tanımlamaları

```
#define BATTERY_MEAS_EN() PORTDbits.RD3=1
#define BATTERY_MEAS_DIS() PORTDbits.RD3=0

#define ANALOG_VCC_ON() PORTDbits.RD7=1
#define ANALOG_VCC_OFF() PORTDbits.RD7=0

#define UNSELECT_TRANSDUCERS() PORTDbits.RD5=0; PORTDbits.RD6=0
#define CHARGE_PUMP_ENABLE() PORTAbits.RA4=1
#define CHARGE_PUMP_DISABLE() PORTAbits.RA4=0
#define UP_STREAM 1
#define DOWN_STREAM 2

//uart*****
unsigned char sayac_adres;
volatile unsigned char in_data[11];
volatile unsigned char temp;
unsigned int chksum_calculated,chksum_received;
//*****

//kalibrasyon*****
extern unsigned char calibration_point_count;

volatile double transit_time_down,transit_time_up;
volatile double inst_avg_flow=0,avg_flow=0,total_flow=0;
volatile double flows[10];
volatile double look_up_value;

volatile signed long total=0,totall=0;
unsigned char negative_flag,measure_count=0,avg_count=0;
unsigned int state_error_count=0;

volatile double double_total=0;
volatile double flow=0;//instant
volatile double skg=0,shg=0,shg_cal=0,velocity=0;

unsigned char rtc_ok=0;

volatile unsigned int timer3 = 0;//acam isr için timeout

//lcd
union
{
    {
        unsigned char byte;
        struct
        {
            unsigned char I_of_k:1;
            unsigned char c_of_k:1;
            unsigned char slash:1;
            unsigned char W:1;
            unsigned char h_down:1;
            unsigned char h_up:1;
            unsigned char _3_of_m3:1;
            unsigned char m_of_:1;
        };
    }display_signals_1;

    union
    {
        {
            unsigned char byte;
            struct
            {
                unsigned char y_of_M:1;
                unsigned char penalty:1;
                unsigned char error:1;
                unsigned char rf:1;
                unsigned char no_flow:1;
                unsigned char battery_low:1;
                unsigned char s:1;
            };
        };
    };
};
```

EK-2. (Devam) main.h tanımlamaları

```

    unsigned char j:1;
};
}display_signals_2;

#define MEASUREMENT_COUNT_PER_SEC 1

#define US25

#ifndef US25
#define TIME_LIMIT_LOWER 39000.0
#define TIME_LIMIT_UPPER 49000.0
#define CROSS_SECTION 1020.7035 //10(dm/s)*3600(dm/h)*pi*r*r/10000(dm.dm)
#define L_div_2 (34640000.0) // 60/cos30*2 (nm)
#define Q4 7000.0
#define MAX_FLOW_RATE 9000.0
#endif

#if defined(US15) && defined(US20)
#error Birden fazla sayac tipi tanımladiniz.
#endif

#if defined(US15) && defined(US25)
#error Birden fazla sayac tipi tanımladiniz.
#endif

#if defined(US20) && defined(US25)
#error Birden fazla sayac tipi tanımladiniz.
#endif

#if !defined(US15) && !defined(US20) && !defined(US25)
#error Sayac tipi tanımlamadiniz.
#endif

enum { STATE_IDLE = 0, STATE_MEASURE, STATE_EEPROM, STATE_QUERY, STATE_CALIBRATION }
current_state;
extern enum { CALIBRATION_MODE_1=0,CALIBRATION_MODE_2,CALIBRATION_MODE_3,DOWNLOAD,READ;
calibration_mode;
extern enum { ADRESS=0,GENERAL } calibration_type;
enum { FALSE=0,TRUE=1 } byte_received;

#pragma udata access my_access

near unsigned char irq_acam;
near unsigned char aborted;
near volatile double hfConst;
near unsigned char hfc_fail;

extern near struct _tarih
{
    unsigned char gun;
    unsigned char ay;
    unsigned char yıl;
    unsigned char saat;
    unsigned char dakika;
    unsigned char saniye;
    unsigned char h_gun; // haftanın gunu ( 0 pazar )...
}dt;

extern near union
{
    unsigned char f;
    struct
    {
        unsigned char saniye:1;
        unsigned char dakika:1;
        unsigned char saat:1;
        unsigned char gun:1;
        unsigned char ay:1;
    }

```

EK-2. (Devam) main.h tanımlamaları

```

    unsigned char yıl:1;
    unsigned char yk_degis:1; // yaz kis saat degisikligi...
    unsigned char :1;
};
}RTC_flags;

//fonksiyon bildirimleri---

void init_pic(void);

void delay_msx_in_idle (unsigned int sure);
void wakeup (void);
void sleep (void);
void update_display(void);
void power_fail (void);
void calc_ins_flow(void);
void calculate_measurement(void);
void hardware_test(void);
void Send_abyte_to_Uart (unsigned char byte);
void send_ack(void);
void init_uart(void);
void init_rtc(void);

extern void init_acam (void);
extern BOOL test_gp2(void);
extern void isr_acam (void);
extern void start_cycle(unsigned char direction);
extern double calc_acam_transmit_time (void);
extern void reset_acam(void);
extern void start_hfclk_cal(void);
extern void calc_acam_hfconst (void);
extern void sleep_acam (void);
extern void wakeup_acam (void);

extern void init_lcd(void);
extern void Clear_Lcd(void);
extern void Sleep_LCD(void);
extern void wakeup_LCD(void);
extern void display_measure(long tf,long lm,long mk);
extern void Display_Versiyon(void);
extern void display_long(long it,long lg);

extern void read_values_from_eeprom(void);
extern void ext_write_byte(unsigned int adr,unsigned char data);
extern unsigned char ext_read_byte(unsigned int adr);
extern void ext_read_sequential(unsigned int adr, unsigned char n,near unsigned char *pbuff);
extern void ext_write_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);

extern void rtc_isr(void);
extern void rtc_task(void);

//kalibrasyon*****
extern double look_up (double calibration_index);
extern void finalize_calibration(void);
extern void select_kalibration_mode(void);
extern void apply_cal(void);
//*****
extern void save_values_to_eeprom(void);

```

EK-3. gp2.c fonksiyonları

```
#include <p18f4520.h>
#include "global.h"
#include "gp2.h"

static void init_acam_ports (void)
{
    TRIS_MEAS_VCC_OUTPUT();
    ACAM_CHIP_UNSELECT();
    TRIS_SSN_OUTPUT();

    TRIS_RESET_ACAM_OUTPUT();
    ACAM_RESET_PIN=1;

    TRIS_TR1_CONTROL_OUTPUT();
    TRIS_TR2_CONTROL_OUTPUT();
    UNSELECT_TRANSDUCERS();

    TRIS_SSN_OUTPUT();
    TRIS_SPI_SCK_OUTPUT();
    TRIS_SPI_SI_INPUT();
    TRIS_SPI_SO_OUTPUT();

    PIE1bits.SSPIE = 0;
    IPR1bits.SSPIP = 0;
    PIR1bits.SSPIF = 0;

    SSPCON1 = 0x20;
    SSPSTATbits.CKE = 0;
    SSPSTATbits.SMP = 0;
}

static void reset_acam_config(void)
{
    WriteAcamReg(0, &Reg0.Reg);
    WriteAcamReg(1, &Reg1.Reg);
    WriteAcamReg(2, &Reg2.Reg);
    WriteAcamReg(3, &Reg3.Reg);
    WriteAcamReg(4, &Reg4.Reg);
    WriteAcamReg(5, &Reg5.Reg);
}

static void config_acam(void)
{
    Reg0.NEG_START=0;
    Reg0.NEG_STOP1=0;
    Reg0.NEG_STOP2=0;
    Reg0.MRange2=1;           //mrange2 selected
    Reg0.DisAutoCal=0;        //update cal1, cal2 end of meas
    Reg0.Calibrate=1;         //use cal1 cal2
    Reg0.SelClkT=1;           //use 128*hsclk refference signal for temp. meas.
    Reg0.FAKE=0;              //2 fake meas
    Reg0.TCycle=1;            //300us cycle time for temp meas
    Reg0.PORT=0;              //use 2 ports of temp ports
    Reg0.START_CLKHS=2;       //measurement is started with 640 us delay
    Reg0.ClkHSDiv=0;          //set clkhs predivider 1
    Reg0.CALRES=3;            //use 16 periods for ceramic osc. cal.
    Reg0.DIV_FIRE=1;          //predivider 2
    Reg0.FIRE=1;              //1 pulse generate

    Reg1.nc_0=0;
    Reg1.HITIN1=2;            //wait for 2 pulses on stop1 channel
    Reg1.HITIN2=0;            //wait for 0 pulses on stop2 channel
    Reg1.sc_0=1;
    Reg1.EN_FAST_INIT=0;
    Reg1.HIT1=1;              //mrange2:start ALU process parameters
    Reg1.HIT2=2;              //mrange2:1.stop1 ALU process parameters

    Reg2.DELVAL1_0=0x00;
```

EK-3. (Devam) gp2.c fonksiyonları

```

Reg2.DELVAL1_1=0x10;
Reg2.DELVAL1_2=0x00;
Reg2.RFEDGE1=0;           //edge sensitivity falling or rising edge for stop1 channel
Reg2.RFEDGE2=0;           //edge sensitivity falling or rising edge for stop2 channel
Reg2.EN_INT=5;             //Alu and time out interrupt enable

Reg3.DELVAL2_0=0;
Reg3.DELVAL2_1=0;
Reg3.DELVAL2_2=0;
Reg3.SEL_TIM0_MR2=3;       //pre divider for time out 4ms
Reg3.EN_ERR_VAL=1;         //write 0xffffffff into result reg if timeout occurred
Reg3.nc_0=0;

Reg4.DELVAL3_0=0;
Reg4.DELVAL3_1=0;
Reg4.DELVAL3_2=0;
Reg4.nc_0=0;
Reg4.sc_0=1;
Reg4.nc_1=0;

Reg5.PHASE_FIRE_0=0;       //no pulse inverting in sequence
Reg5.PHASE_FIRE_1=0;       //no pulse inverting in sequence
Reg5.REPEAT_FIRE=0;        //no signal repetitions for quasy sing around
Reg5.DIS_PHASENOISE=1;     //enable phase shifting
Reg5.EN_STARTNOISE=0;      //noise unit disable
Reg5.CONF_FIRE=0;          //fire2 and fire 1 enable

reset_acam_config();
}

void init_acam (void)
{
    init_acam_ports ();
    config_acam();
}

static unsigned long ReadAcamResultReg(unsigned char Address)
{
    char i;
    union
    {
        unsigned long _result;
        unsigned char bytes[4];
    } result;
    // Select the chip and send the Read Control Register opcode/address
    ACAM_CHIP_SELECT();

    SSPBUF = Address | READ_ADR;
    while(!PIR1bits.SSPIF); // Wait until the opcode/address is transmitted
    PIR1bits.SSPIF = 0;
    for (i=3;i>=0;--i)
    {
        SSPBUF=0;
        while (!PIR1bits.SSPIF);
        result.bytes[i]=SSPBUF;
        PIR1bits.SSPIF = 0;
    }
    PIR1bits.SSPIF = 0;
    ACAM_CHIP_UNSELECT();
    return result._result;
}

static unsigned char ReadAcamReg1(void) //for test communication
{
    unsigned char reg1;
    // Select the chip and send the Read Control Register opcode/address
    ACAM_CHIP_SELECT();

```

EK-3. (Devam) gp2.c fonksiyonları

```

SSPBUF = 0x05 | READ_ADR;
while(!PIR1bits.SSPIF); // Wait until the opcode/address is transmitted
PIR1bits.SSPIF = 0;
SSPBUF=0;
while (!PIR1bits.SSPIF);
reg1=SSPBUF;
PIR1bits.SSPIF = 0;
ACAM_CHIP_UNSELECT();

return reg1;
}

static unsigned int ReadAcamStatusReg(void)
{
    char i;
    union
    {
        unsigned int _status;
        unsigned char bytes[2];
    } status;
    // Select the chip and send the Read Control Register opcode/address
    ACAM_CHIP_SELECT();

    SSPBUF = 0x04 | READ_ADR;
    while(!PIR1bits.SSPIF); // Wait until the opcode/address is transmitted
    PIR1bits.SSPIF = 0;
    for (i=1;i>=0;--i)
    {
        SSPBUF=0;
        while (!PIR1bits.SSPIF);
        status.bytes[i]=SSPBUF;
        PIR1bits.SSPIF = 0;
    }
    PIR1bits.SSPIF = 0;
    ACAM_CHIP_UNSELECT();
    return status._status;
}

static void WriteAcamReg(unsigned char Address, unsigned char *Data)
{
    char i;
    ACAM_CHIP_SELECT();
    SSPBUF = Address | WRITE_ADR; // Send the opcode and address.
    while(!PIR1bits.SSPIF); // Wait until opcode/address is transmitted.
    PIR1bits.SSPIF = 0;
    for (i=2;i>=0;--i)
    {
        SSPBUF = Data[i]; // Send the byte to be written.
        while(!PIR1bits.SSPIF); // Wait until register is written.
        PIR1bits.SSPIF = 0;
    }
    ACAM_CHIP_UNSELECT();
}

BOOL test_gp2(void)
{
    unsigned char reg1;
    BOOL gp2_OK;

    reg1=ReadAcamReg1();

    gp2_OK=(reg1==0x21) ? TRUE:FALSE;

    return gp2_OK;
}

```

EK-3. (Devam) gp2.c fonksiyonları

```

static void send_command(unsigned char command)
{
    ACAM_CHIP_SELECT();

    SSPBUF = command;
    while(!PIR1bits.SSPIF);
    PIR1bits.SSPIF = 0;

    ACAM_CHIP_UNSELECT();
}

void start_cycle(unsigned char direction)
{
    if (direction==1)
    {
        SELECT_T1();
    }
    else
    {
        SELECT_T2();
    }

    Reg5.CONF_FIRE=direction;
    WriteAcamReg(5, &Reg5.Reg);
    send_command(INIT);
    send_command(START_CYCLE);
}

void isr_acam (void)
{
    UNSELECT_TRANSDUCERS();
    acam_status.stat=ReadAcamStatusReg();
    result=ReadAcamResultReg(0);
}

double calc_acam_transmit_time (void)
{
    double transit_time;
    if (acam_status.TIME_OUT || acam_status.TIME_OUT_PREC)
        aborted=0x55;
    transit_time=((double)result)*250.0;
    transit_time/=65536.0;
    transit_time*=hfConst;
    return transit_time;
}

void start_hfclk_cal(void)
{
    send_command(INIT);
    send_command(START_CAL_RESONATOR);
}

void calc_acam_hfconst (void)
{
    if (acam_status.TIME_OUT || acam_status.TIME_OUT_PREC)
        hfc_fail=0x55;
    else
        hfc_fail=0x00;
    hfConst=((double)result)*0.25; //0.25 us refclk per
    hfConst/=65536.0;
    hfConst/=488.281; //16 periods of 32,768 khz
}

void reset_acam(void)
{
    ACAM_RESET_PIN=0;
    delay10us();
}

```

EK-3. (Devam) gp2.c fonksiyonları

```
ACAM_RESET_PIN=1;  
send_command(POWER_ON_RESET);  
config_acam();  
}
```

```
void sleep_acam (void)  
{  
    Reg0.START_CLKHS=0;  
    WriteAcamReg(0, &Reg0.Reg);  
}
```

```
void wakeup_acam (void)  
{  
    Reg0.START_CLKHS=1;  
    WriteAcamReg(0, &Reg0.Reg);  
}
```

EK-4. gp2.h tanımlamaları

```
#define ACAM_CHIP_SELECT() PORTAbits.RA5=0
#define ACAM_CHIP_UNSELECT() PORTAbits.RA5=1
#define TRIS_SSN_OUTPUT() TRISAbits.TRISA5=0

#define TRIS_RESET_ACAM_OUTPUT() TRISDbits.TRISD4=0
#define ACAM_RESET_PIN PORTDbits.RD4

#define TRIS_SPI_SCK_OUTPUT() TRISCbits.TRISC3=0
#define TRIS_SPI_SI_INPUT() TRISCbits.TRISC4=1
#define TRIS_SPI_SO_OUTPUT() TRISCbits.TRISC5=0

#define TRIS_MEAS_VCC_OUTPUT() TRISDbits.TRISD7=0
#define MEAS_VCC_EN() PORTDbits.RD7=1
#define MEAS_VCC_DIS() PORTDbits.RD7=0

#define TRIS_TR1_CONTROL_OUTPUT() TRISDbits.TRISD5=0
#define TRIS_TR2_CONTROL_OUTPUT() TRISDbits.TRISD6=0
#define SELECT_T1() PORTDbits.RD5=1;PORTDbits.RD6=0
#define SELECT_T2() PORTDbits.RD5=0;PORTDbits.RD6=1
#define UNSELECT_TRANSDUCERS() PORTDbits.RD5=0; PORTDbits.RD6=0

//-----OPCODES-----
#define WRITE_ADR 0x80
#define READ_ADR 0xB0

#define INIT 0x70
#define POWER_ON_RESET 0x50

#define START_CYCLE 0x01
#define START_TEMP 0x02
#define START_CAL_RESONATOR 0x03
#define START_CAL_TDC 0x04

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char NEG_START:1;
        unsigned char NEG_STOP1:1;
        unsigned char NEG_STOP2:1;
        unsigned char MRange2:1;
        unsigned char DisAutoCal:1;
        unsigned char Calibrate:1;
        unsigned char SelClkT:1;
        unsigned char FAKE:1;
        unsigned char TCycle:1;
        unsigned char PORT:1;
        unsigned char START_CLKHS:2;
        unsigned char ClkHSDiv:2;
        unsigned char CALRES:2;
        unsigned char DIV_FIRE:4;
        unsigned char FIRE:4;
    };
}Reg0;

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char nc_0:8;
        unsigned char HITIN1:3;
        unsigned char HITIN2:3;
        unsigned char sc_0:1;
        unsigned char EN_FAST_INIT:1;
        unsigned char HIT1:4;
    };
}
```

EK-4. (Devam) gp2.h tanımlamaları

```

    unsigned char HIT2:4;
};
}Reg1;

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char DELVAL1_0:8;
        unsigned char DELVAL1_1:8;
        unsigned char DELVAL1_2:3;
        unsigned char RFEDGE1:1;
        unsigned char RFEDGE2:1;
        unsigned char EN_INT:3;
    };
}Reg2;

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char DELVAL2_0:8;
        unsigned char DELVAL2_1:8;
        unsigned char DELVAL2_2:3;
        unsigned char SEL_TIM0_MR2:2;
        unsigned char EN_ERR_VAL:1;
        unsigned char nc_0:2;
    };
}Reg3;

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char DELVAL3_0:8;
        unsigned char DELVAL3_1:8;
        unsigned char DELVAL3_2:3;
        unsigned char nc_0:2;
        unsigned char sc_0:1;
        unsigned char nc_1:2;
    };
}Reg4;

union
{
    unsigned short long Reg;
    struct
    {
        unsigned char PHASE_FIRE_0:8;
        unsigned char PHASE_FIRE_1:8;
        unsigned char REPEAT_FIRE:3;
        unsigned char DIS_PHASENOISE:1;
        unsigned char EN_STARTNOISE:1;
        unsigned char CONF_FIRE:3;
    };
}Reg5;

typedef union
{
    unsigned int stat;
    struct
    {
        unsigned char POINTER_RES:3;
        unsigned char HITS_OF_CH1:3;
        unsigned char HITS_OF_CH2_0:2;
    };
};

```

EK-4. (Devam) gp2.h tanımlamaları

```

    unsigned char HITS_OF_CH2_1:1;
    unsigned char TIME_OUT:1;
    unsigned char TIME_OUT_PREC:1;
    unsigned char OPEN_ERR:1;
    unsigned char SHORT_ERR:1;
    unsigned char nc_0:3;
};
}STATUS;

STATUS acam_status;
static unsigned long result;

#pragma udata access my_access

extern near unsigned char aborted;
extern near unsigned char hfc_fail;
extern near volatile double hfConst;
#pragma udata

static void init_acam_ports (void);
static void config_acam(void);
static void reset_acam_config(void);
static void WriteAcamReg(unsigned char Address, unsigned char *Data);
static unsigned long ReadAcamResultReg(unsigned char Address);
static unsigned char ReadAcamReg1(void);
static unsigned int ReadAcamStatusReg(void);
static void send_command(unsigned char command);
void start_cycle(unsigned char direction);
void init_acam (void);
double calc_acam_transmit_time (void);
void calc_acam_hfconst (void);

```

EK-5. kalibrasyon.c fonksiyonları

```
#include <p18f4520.h>
#include "kalibrasyon.h"
#include "global.h"

double look_up (double calibration_index)
{
    static unsigned char i;
    double _look_up;
    if (calibration_index>=Calibration_parameters[0].measured_val &&
        calibration_index<=Calibration_parameters[1].measured_val)
        i=0;
    else

    if (calibration_index<=Calibration_parameters[2].measured_val)
        i=1;
    else

    if (calibration_index<=Calibration_parameters[3].measured_val)
        i=2;
    else

    if (calibration_index<=Calibration_parameters[4].measured_val)
        i=3;
    else

    if (calibration_index<=Calibration_parameters[5].measured_val)
        i=4;
    else

    if (calibration_index<=Calibration_parameters[6].measured_val)
        i=5;
    else

    if (calibration_index<=Calibration_parameters[7].measured_val)
        i=6;
    else

    if (calibration_index<=Calibration_parameters[8].measured_val)
        i=7;
    else

    if (calibration_index<=Calibration_parameters[9].measured_val)
        i=8;
    else

    if (calibration_index<=Calibration_parameters[10].measured_val)
        i=9;
    else

    if (calibration_index<=Calibration_parameters[11].measured_val)
        i=10;
    else

    if (calibration_index<=Calibration_parameters[12].measured_val)
        i=11;
    else

    if (calibration_index<=Calibration_parameters[13].measured_val)
        i=12;
    else

    if (calibration_index<=Calibration_parameters[14].measured_val)
        i=13;
    else

    if (calibration_index<=Calibration_parameters[15].measured_val)
        i=14;
    else
```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

if (calibration_index<=Calibration_parameters[16].measured_val)
i=15;
else

if (calibration_index<=Calibration_parameters[17].measured_val)
i=16;
else

if (calibration_index<=Calibration_parameters[18].measured_val)
i=17;
else

if (calibration_index<=Calibration_parameters[19].measured_val)
i=18;
else

if (calibration_index>Calibration_parameters[19].measured_val)
i=19;
else return 0;
_look_up=Calibration_coefficient[i].coefficient*calibration_index+Calibration_coefficient[i].offset;
return _look_up;
}

void calculate_coefficients(void)
{
    unsigned char i,data;
    long actual_val,long_data;
    double double_actual_val[2];

    Calibration_parameters[0].lsb1=0;
    Calibration_parameters[0].lsb2=0;
    Calibration_parameters[0].msb1=0;
    Calibration_parameters[0].msb2=0;
    Calibration_parameters[0].measured_val=0.0;
    actual_val=0;
    double_actual_val[0]=0.0;

    for (i=1;i<=calibration_point_count;++i)
    {
        long_data=Calibration_parameters[i].lsb1;
        actual_val=long_data;
        long_data=Calibration_parameters[i].lsb2;
        actual_val=actual_val+(long_data<<8);
        long_data=Calibration_parameters[i].msb1;
        actual_val=actual_val+(long_data<<16);
        long_data=Calibration_parameters[i].msb2;
        actual_val=actual_val+(long_data<<24);
        double_actual_val[1]=((double)actual_val)/100.0;

        if (i==1)
        {
            long long_zero_min_coefx100;
            double zero_min_coef;
            long_zero_min_coefx100=double_actual_val[1]/Calibration_parameters[i].measured_val;
            zero_min_coef=((double)long_zero_min_coefx100)/2.0;

            Calibration_coefficient[i-1].coefficient=((double_actual_val[1]/Calibration_parameters[i].measured_val)-
            zero_min_coef)/(Calibration_parameters[i].measured_val-Calibration_parameters[i-1].measured_val);
        }else
        {
            Calibration_coefficient[i-1].coefficient=((double_actual_val[1]/Calibration_parameters[i].measured_val)-
            (double_actual_val[0]/Calibration_parameters[i-1].measured_val))/(Calibration_parameters[i].measured_val-
            Calibration_parameters[i-1].measured_val);
        }
        Calibration_coefficient[i-1].offset=((double_actual_val[1]/Calibration_parameters[i].measured_val)-
        Calibration_coefficient[i-1].coefficient*Calibration_parameters[i].measured_val;

```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

    double_actual_val[0]=double_actual_val[1];
}

}

void finalize_calibration(void)
{
    if (calibration_mode==CALIBRATION_MODE_1)
    {
        unsigned int i;
        if (start_total_cons_from_zero)
        {
            total=0;
            total=0;
            start_total_cons_from_zero=0;
            avg_flow=0;
        }
        save_values_to_eeprom();
    }
    else if (calibration_mode==CALIBRATION_MODE_3 )
    {
        static long long_deger=0;
        long_deger = (long)(skg*100.0);
        ext_write_sequential(0,4,&long_deger);
        long_deger = (long)(shg_cal*100.0);
        ext_write_sequential(26,4,&long_deger);
        long_deger=0;
    }
    else if (calibration_mode==CALIBRATION_MODE_2 || calibration_mode==DOWNLOAD)
    {
        unsigned char data;
        long long_avg;
        unsigned char i;

        ext_write_byte(30,calibration_point_count);
        for (i=1;i<=calibration_point_count;++i)
        {
            ext_write_byte((31+(i-1)*8),Calibration_parameters[i].lsb1);
            ext_write_byte((32+(i-1)*8),Calibration_parameters[i].lsb2);
            ext_write_byte((33+(i-1)*8),Calibration_parameters[i].msb1);
            ext_write_byte((34+(i-1)*8),Calibration_parameters[i].msb2);

            long_avg =(long)Calibration_parameters[i].measured_val*100.0;

            ext_write_sequential((35+(i-1)*8), 4,&long_avg);

        }
        calculate_coefficients();
    }
    current_state=STATE_IDLE;
    received_byte_count=0;
    display_signals_1.I_of_k=0;
    display_signals_1.c_of_k=0;
    byte_received=FALSE;
}

void send_calibration_table(void)
{
    unsigned char i,j,data,loop;
    long long_data;
    int chksum;

    for (i=1;i<21;++i)
    {
        loop=0;
        while(!loop)
        {
            ClrWdt();

```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

    delay10msx(25);
    chksum=0;
    Send_abyte_to_Uart (Calibration_parameters[i].lsb1);

    delay10msx(5);//-----
    chksum+=Calibration_parameters[i].lsb1;

    Send_abyte_to_Uart (Calibration_parameters[i].lsb2);

    delay10msx(5);//-----
    chksum+=Calibration_parameters[i].lsb2;

    Send_abyte_to_Uart (Calibration_parameters[i].msb1);

    delay10msx(5);//-----
    chksum+=Calibration_parameters[i].msb1;

    Send_abyte_to_Uart (Calibration_parameters[i].msb2);

    delay10msx(5);//-----
    chksum+=Calibration_parameters[i].msb2;

    long_data=(long)Calibration_parameters[i].measured_val*100.0;

    for (j=0;j<4;++j)
    {
        data=(long_data>>(8*j));
        Send_abyte_to_Uart (data);
        delay10msx(5);//-----
        chksum+=data;
    }
    for (j=0;j<2;++j)
    {
        data=(chksum>>(8*j));
        Send_abyte_to_Uart (data);
        delay10msx(5);//-----
    }
    delay10msx(100);

    if (PIR1bits.RCIF)
    {
        data=RCREG;
        if (data!=0)
        {
            loop=255;
        }
        else
        {
            loop=0;
        }
    }
}
} //for
}

void select_kalibration_mode(void)
{
    if (in_data[9]==1)
    {
        calibration_mode=CALIBRATION_MODE_1;
        received_byte_count=0;
    } //date time setting
    else if (in_data[9]==2)
    {
        calibration_mode=CALIBRATION_MODE_2;
        calibration_point_count=0;
    } //calibration
}

```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

else if (in_data[9]==3)
{
    calibration_mode=CALIBRATION_MODE_3;
    received_byte_count=0;
} //skg.shgcal
else if (in_data[9]==4)
{
    calibration_mode=DOWNLOAD;
    received_byte_count=0;
    calibration_point_count=0;
} else if (in_data[9]==5)
{
    calibration_mode=READ;
    send_calibration_table();
    finalize_calibration();
}

}

void receive_calibration_table(void)//receive existing calibration table from pc
{
    unsigned char i;
    unsigned char data;
    signed long long_data;

    chksum_calculated=0;
    for (i=0;i<9;++i)
        chksum_calculated+=in_data[i];
    chksum_received=in_data[9];
    chksum_received+=((unsigned int)in_data[10]<<8);
    if (calibration_type==ADDRESS )
    {
        if (chksum_calculated==chksum_received && sayac_adres==in_data[0])
        {
            ++calibration_point_count;
            Calibration_parameters[calibration_point_count].lsb1=in_data[1];
            Calibration_parameters[calibration_point_count].lsb2=in_data[2];
            Calibration_parameters[calibration_point_count].msb1=in_data[3];
            Calibration_parameters[calibration_point_count].msb2=in_data[4];

            long_data=0;
            for (i=0;i<4;++i)
            {
                data=in_data[i+5];
                long_data+=((long)data<<(8*i));
            }
            Calibration_parameters[calibration_point_count].measured_val=long_data/100.0;
            send_ack();
        }
    }
    else //adres
    {
        if (chksum_calculated==chksum_received)
        {
            ++calibration_point_count;
            Calibration_parameters[calibration_point_count].lsb1=in_data[1];
            Calibration_parameters[calibration_point_count].lsb2=in_data[2];
            Calibration_parameters[calibration_point_count].msb1=in_data[3];
            Calibration_parameters[calibration_point_count].msb2=in_data[4];

            long_data=0;
            for (i=0;i<4;++i)
            {
                data=in_data[i+5];
                long_data+=((long)data<<(8*i));
            }
            Calibration_parameters[calibration_point_count].measured_val=long_data/100.0;
        }
    }
}

```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

}

void apply_cal(void)
{
    unsigned char i;

    if (in_data[6]=='I' && in_data[7]=='R' && in_data[8]=='E' && in_data[9]=='I')
    {
        if (calibration_type==ADRESS)
        {
            if (calibration_mode==CALIBRATION_MODE_1)
            {
                if (sayac_adres==in_data[10] && chksum_calculated==chksum_received)
                {
                    send_ack();
                    finalize_calibration();
                }
            } else if (sayac_adres==in_data[10] && (calibration_mode==CALIBRATION_MODE_2 ||
calibration_mode==DOWNLOAD))
            {
                send_ack();
                finalize_calibration();
            }
        }
    } else
    {
        finalize_calibration();
    }

    } else //gelen sonlanma değil
    {

        if (calibration_mode==DOWNLOAD)
            receive_calibration_table();
        else
            if (calibration_mode==CALIBRATION_MODE_1)
            {

                ++received_byte_count;
                if (calibration_type==ADRESS && received_byte_count<15)
                    chksum_calculated+=in_data[10];
                switch (received_byte_count)
                {
                    case 1:dt.dakika=in_data[10]; break;
                    case 2:dt.saat=in_data[10]; break;
                    case 3:dt.gun=in_data[10]; break;
                    case 4:dt.ay=in_data[10]; break;
                    case 5:dt.yil=in_data[10]; break;
                    case 6:dt.h_gun=in_data[10]; break;
                    case 7:Production_date.day=in_data[10]; break;
                    case 8:Production_date.month=in_data[10]; break;
                    case 9:Production_date.year=in_data[10]; break;
                    case 10:Calibration_date.day=in_data[10]; break;
                    case 11:Calibration_date.month=in_data[10]; break;
                    case 12:Calibration_date.year=in_data[10]; break;
                    case 13:sayac_adres=in_data[10]; break;
                    case 14:
                        if (in_data[10]==0)
                            start_total_cons_from_zero=0x55; break;
                    case 15: chksum_received=in_data[10]; break;
                    case 16: chksum_received+=((unsigned int)in_data[10]<<8);

                }

            } else if (calibration_mode==CALIBRATION_MODE_2 )
            {
                chksum_calculated=0;
                for (i=4;i<9;++i)
                    chksum_calculated+=in_data[i];
            }
    }
}

```

EK-5. (Devam) kalibrasyon.c fonksiyonları

```

    checksum_received=in_data[9];
    checksum_received+=((unsigned int)in_data[10]<<8);

    if (calibration_type==ADRESS)
    {
        if (checksum_calculated==checksum_received && sayac_adres==in_data[4])
        {
            unsigned char data;
            unsigned int chk;
            signed long long_data;

            long_data=(long)avg_flow*100.0;
            chk=0;

            for (i=0;i<4;++i)
            {
                data=(long_data>>(i*8));
                chk+=data;
                Send_abyte_to_Uart (data);
                delay10msx(5);
            }
            data=chk;
            Send_abyte_to_Uart (data);
            delay10msx(5);
            data=chk>>8;
            Send_abyte_to_Uart (data);
            delay10us();

            delay10msx(100);
            if (PIR1bits.RCIF)
            {
                data=RCREG;
                if (data!=0)
                {
                    ++calibration_point_count;
                    Calibration_parameters[(calibration_point_count)].lsb1=in_data[5];

                    Calibration_parameters[(calibration_point_count)].lsb2=in_data[6];
                    Calibration_parameters[(calibration_point_count)].msb1=in_data[7];
                    Calibration_parameters[(calibration_point_count)].msb2=in_data[8];
                    Calibration_parameters[(calibration_point_count)].measured_val=avg_flow;
                }
            }
        }
    }
    }else //
    {
        if (checksum_calculated==checksum_received)
        {
            ++calibration_point_count;
            Calibration_parameters[(calibration_point_count)].lsb1=in_data[5];

            Calibration_parameters[(calibration_point_count)].lsb2=in_data[6];
            Calibration_parameters[(calibration_point_count)].msb1=in_data[7];
            Calibration_parameters[(calibration_point_count)].msb2=in_data[8];
            Calibration_parameters[(calibration_point_count)].measured_val=avg_flow;
        }
    }
}

} //sonlanma değil
}

```

EK-6. kalibrasyon.h tanımlamaları

```
#pragma udata xdata=0x300
struct calibration_coef
{
    double coefficient,offset;
}Calibration_coefficient[20];

#pragma udata
#pragma udata mydata=0x400
struct calibration_param
{
    unsigned char lsb1,lsb2,msb1,msb2;
    double measured_val;
}Calibration_parameters[21];
unsigned char received_byte_count=0,calibration_point_count=0,start_total_cons_from_zero=0;
#pragma udata

enum { CALIBRATION_MODE_1=0,CALIBRATION_MODE_2,CALIBRATION_MODE_3, DOWNLOAD,READ}
calibration_mode;
extern enum { STATE_IDLE = 0, STATE_MEASURE,STATE_EEPROM,STATE_QUERY,STATE_CALIBRATION }
current_state;
enum { ADDRESS=0,GENERAL } calibration_type;
extern enum {FALSE=0,TRUE=1} byte_received;

extern volatile signed long total,totall;
extern volatile double skg,shg_cal;
extern volatile double avg_flow;
extern unsigned int chksum_calculated,chksum_received;

extern unsigned char sayac_adres;
extern volatile unsigned char in_data[11];//uart buffer

double look_up (double calibration_index);
void calculate_coefficients(void);
void finalize_calibration(void);
void send_calibration_table(void);
void select_kalibration_mode(void);
void receive_calibration_table(void);
void apply_cal(void);

extern void save_values_to_eeprom(void);
extern void display_long(long it,long lg);

extern void ext_read_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);
extern void ext_write_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);
extern void ext_write_byte(unsigned int adr,unsigned char data);

extern void Send_abyte_to_Uart (unsigned char byte);
extern void send_ack(void);
extern union
{
    unsigned char byte;
    struct
    {
        unsigned char I_of_k:1;
        unsigned char c_of_k:1;
        unsigned char slash:1;
        unsigned char W:1;
        unsigned char h_down:1;
        unsigned char h_up:1;
        unsigned char _3_of_m3:1;
        unsigned char m_of_:1;
    };
}display_signals_1;

extern union
{
    unsigned char byte;
    struct
```

EK-6. (Devam) kalibrasyon.h tanımlamaları

```

{
    unsigned char y_of_M:1;
    unsigned char penalty:1;
    unsigned char error:1;
    unsigned char rf:1;
    unsigned char no_flow:1;
    unsigned char battery_low:1;
    unsigned char s:1;
    unsigned char j:1;
};
}display_signals_2;

#pragma udata access my_access

near struct date
{
    unsigned char day;
    unsigned char month;
    unsigned char year;
}Production_date,Calibration_date;

extern near struct _tarih
{
    unsigned char gun;
    unsigned char ay;
    unsigned char yil;
    unsigned char saat;
    unsigned char dakika;
    unsigned char saniye;
    unsigned char h_gun; // haftanın gunu ( 0 pazar )...
}dt;
```

EK-7. eeprom.c fonksiyonları

```
#include <p18f4520.h>
#include "global.h"

void read_date_time_from_eeprom (void);
void write_date_time_to_eeprom (void);

extern void calculate_coefficients(void);
extern void ext_write_byte(unsigned int adr,unsigned char data);
extern unsigned char ext_read_byte(unsigned int adr);
extern void ext_read_sequential(unsigned int adr, unsigned char n,near unsigned char *pbuff);
extern void ext_write_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);

extern enum { STATE_IDLE = 0, STATE_MEASURE,STATE_EEPROM,STATE_QUERY,STATE_CALIBRATION }
current_state;
extern volatile double skg,shg_cal;
extern volatile signed long total,totall;
extern unsigned char calibration_point_count;

extern unsigned char sayac_adres;
extern unsigned long ID;

extern struct calibration_param
{
    unsigned char lsb1,lsb2,msb1,msb2;
    double measured_val;
}Calibration_parameters[21];

#pragma udata access my_access

extern near struct _tarih
{
    unsigned char gun;
    unsigned char ay;
    unsigned char yil;
    unsigned char saat;
    unsigned char dakika;
    unsigned char saniye;
    ed char h_gun; // haftanın gunu ( 0 pazar )...
}dt;

extern near struct date
{
    unsigned char day;
    unsigned char month;
    unsigned char year;
}Production_date,Calibration_date;

#pragma udata

unsigned char read_eeprom (unsigned char address)
{
    EEADR = address;
    EECON1bits.EEPGD = 0;
    EECON1bits.CFGS = 0;
    EECON1bits.RD = 1;
    while (EECON1bits.RD);
    return EEDATA;
}

void write_eeprom (unsigned char value,unsigned char address)
{
    EEADR = address;
    EEDATA = value;
    EECON1bits.EEPGD = 0;
    EECON1bits.CFGS = 0;
    EECON1bits.WREN = 1;

    INTCONbits.GIE=0;
```

EK-7. (Devam) eeprom.c fonksiyonları

```

EECON2=0x55;
EECON2=0xaa;
EECON1bits.WR = 1;
while (EECON1bits.WR);
delay10us();
INTCONbits.GIE = 1;

EECON1bits.WREN = 0;
}

void read_values_from_eeprom()
{
    unsigned char i;
    long long_avg,long_data;

    ext_read_sequential(0, 4,&long_data);
    skg= ((double)long_data)/100.0;

    sayac_adres=ext_read_byte(5);
    if(sayac_adres==0xFF)
        sayac_adres=1;

    read_date_time_from_eeprom ();

    Production_date.day = ext_read_byte(12);
    Production_date.month = ext_read_byte(13);
    Production_date.year= ext_read_byte(14);

    Calibration_date.day = ext_read_byte(15);
    Calibration_date.month = ext_read_byte(16);
    Calibration_date.year= ext_read_byte(17);

    ext_read_sequential(18, 4,&total);

    ext_read_sequential(22, 4,&total);

    ext_read_sequential(26, 4,& long_data);
    shg_cal=((double)long_data)/100.0;

    calibration_point_count=ext_read_byte(30);
    if (calibration_point_count!=0 && calibration_point_count!=0xff)
    {
        for (i=1;i<=calibration_point_count;++i)
        {

            Calibration_parameters[i].lsb1=ext_read_byte(31+(i-1)*8);
            Calibration_parameters[i].lsb2=ext_read_byte(32+(i-1)*8);
            Calibration_parameters[i].msb1=ext_read_byte(33+(i-1)*8);
            Calibration_parameters[i].msb2=ext_read_byte(34+(i-1)*8);

            ext_read_sequential((35+(i-1)*8), 4,&long_avg);
            Calibration_parameters[i].measured_val = (double)(long_avg/100.0);
        }
        calculate_coefficients();
    }
}

void save_values_to_eeprom()
{
    ext_write_sequential(18,4,&total);
    ext_write_sequential(22,4,&total);
    ext_write_sequential(26,4,&shg_cal);

    if (current_state==STATE_CALIBRATION)
    {
        ext_write_byte(5,sayac_adres);
        write_date_time_to_eeprom ();
    }
}

```

EK-7. (Devam) eeprom.c fonksiyonları

```
}

void write_date_time_to_eeprom ()
{
    ext_write_byte(6,dt.dakika);
    ext_write_byte(7,dt.saat);
    ext_write_byte(8,dt.gun);
    ext_write_byte(9,dt.ay);
    ext_write_byte(10,dt.yil);
    ext_write_byte(11,dt.h_gun);
    ext_write_byte(12,Production_date.day);
    ext_write_byte(13,Production_date.month);
    ext_write_byte(14,Production_date.year);
    ext_write_byte(15,Calibration_date.day);
    ext_write_byte(16,Calibration_date.month);
    ext_write_byte(17,Calibration_date.year);
}

void read_date_time_from_eeprom ()
{
    dt.dakika = ext_read_byte(6);
    dt.saat = ext_read_byte(7);
    dt.gun = ext_read_byte(8);
    dt.ay = ext_read_byte(9);
    dt.yil = ext_read_byte(10);
    dt.h_gun = ext_read_byte(11);
}
```

EK-8. rtc.c fonksiyonları

```
#include <p18f4520.h>

extern void power_fail(void);
extern void ext_write_byte(unsigned int adr,unsigned char data);
extern unsigned char ext_read_byte(unsigned int adr);
extern void ext_write_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);
extern volatile signed long total,totall;

#pragma udata access my_access

near struct _tarih
{
    unsigned char gun;
    unsigned char ay;
    unsigned char yil;
    unsigned char saat;
    unsigned char dakika;
    unsigned char saniye;
    unsigned char h_gun; // haftanın gunu ( 0 pazar )...
}dt; // date_time...

extern near struct date
{
    unsigned char day;
    unsigned char month;
    unsigned char year;
}Production_date,Calibration_date;

near union
{
    unsigned char f;
    struct
    {
        unsigned char saniye:1;
        unsigned char dakika:1;
        unsigned char saat:1;
        unsigned char gun:1;
        unsigned char ay:1;
        unsigned char yil:1;
        unsigned char yk_degis:1; // yaz kis saat degisikligi...
        unsigned char :1;
    };
}RTC_flags;

void rtc_isr(void)
{
    dt.saniye ++;

    RTC_flags.saniye = 1;

    if(dt.saniye > 59)
    {
        dt.saniye = dt.saniye - 60;
        dt.dakika++;
        RTC_flags.dakika = 1;

        if(dt.dakika > 59)
        {
            dt.dakika = 0;
            dt.saat++;
            RTC_flags.saat = 1;
            if(dt.saat > 23)
            {
                dt.saat = 0;
                dt.gun++;
                RTC_flags.gun = 1;
            }
        }
    }
}
```

EK-8. (Devam) rtc.c fonksiyonları

```

if(++dt.h_gun>7) dt.h_gun = 1;
if(dt.gun > 28)
{
    switch(dt.ay)
    {
        case 2:
            if(dt.yil & 0x03)
            {
                dt.gun = 1;
                dt.ay++;
                RTC_flags.ay = 1;
            }
            else // 4 un kati ise..
            {
                if(dt.gun > 29)
                {
                    dt.gun = 1;
                    dt.ay++;
                    RTC_flags.ay = 1;
                }
            }
            break;

        case 4:case 6:case 9:case 11:
            if(dt.gun > 30)
            {
                dt.gun = 1;
                dt.ay++;
                RTC_flags.ay = 1;
            }
            break;

        case 12:
            if(dt.gun > 31)
            {
                dt.gun = 1;
                dt.ay = 1;
                dt.yil++;
                RTC_flags.ay = 1;
                RTC_flags.yil = 1;
            }
            break;

        default: // 31 gun olan aylar...
            if(dt.gun > 31)
            {
                dt.gun = 1;
                dt.ay++;
                RTC_flags.ay = 1;
            }
    }

    if(dt.ay > 12) // guvenlik amacli...
        dt.ay = 1;
    } // ay...
    } // gun...
} // dakika...
} // saniye...
}

/*static void second_task(void)
{
    // display_saat();
}*/

static void minute_task(void)
{
    power_fail();
}

```

EK-8. (Devam) rtc.c fonksiyonları

```

    RTC_flags.dakika = 0;
}

static void hour_task(void)
{
    ext_write_sequential(25,4,&total);
    ext_write_sequential(29,4,&total);
    ext_write_byte(9,dt.dakika);
    ext_write_byte(10,dt.saat);
    RTC_flags.saat = 0;
}

```

```

static void day_task(void)
{
    ext_write_byte(11,dt.gun);
    ext_write_byte(14,dt.h_gun);
    RTC_flags.gun = 0;
}

```

```

static void mounth_task(void)
{
    ext_write_byte(12,dt.ay);
    RTC_flags.ay = 0;
}

```

```

static void year_task(void)
{
    ext_write_byte(13,dt.yil);
    RTC_flags.yil = 0;
}

```

```

void rtc_task(void)
{
    /* if(RTC_flags.saniye)
        second_task();
    */
    if(RTC_flags.dakika)
        minute_task();

    if(RTC_flags.saat)
        hour_task();

    if(RTC_flags.gun)
        day_task();

    if(RTC_flags.ay)
        mounth_task();

    if(RTC_flags.yil)
        year_task();

    RTC_flags.f &= 0xC0;
}

```

EK-9. _delays.c fonksiyonları

```
#include "_delays.h"

void delay10us(void)
{
    Delay10TCYx(1);
}

void delay10usx(unsigned char t)
{
    Delay10TCYx(t);
}

void delay1ms(void)
{
    Delay1KTCYx(1);
}

void delay1msx(unsigned char t)
{
    Delay1KTCYx(t);
}

void delay10ms(void)
{
    Delay10KTCYx(1);
}

void delay10msx(unsigned char t)
{
    Delay10KTCYx(t);
}
```

EK-10. _delays.h tanımlamaları

```
#ifndef MHZ_4

#include <delays.h>

#define delay10us()   Delay10TCYx(1)
#define delay10usx(t) Delay10TCYx(t)
#define delay1ms()    Delay1KTCYx(1)
#define delay1msx(t)  Delay1KTCYx(t)
#define delay10ms()   Delay10KTCYx(1)
#define delay10msx(t) Delay10KTCYx(t)

void delay10msx_critical(unsigned char t);

// #define delay10msx(t) Delay10KTCYx(t)
#endif

#ifndef MHZ_4
// #else // calisma frekansi 4mhz den farkli ise...

#include <delays.h>

void delay10us(void);
void delay10usx(unsigned char t);
void delay1ms(void);
void delay1msx(unsigned char t);
void delay10ms(void);
void delay10msx(unsigned char t);
void delay10msx_critical(unsigned char t);

#endif
```

EK-11. ext_E2_driver.c fonksiyonları

```

#include <p18f4520.h>
#include "ext_E2_driver.h"
#include "global.h"

static void i2c_start(void)
{
    E2_DATA_OUTPUT;

    E2_DATA_HIGH;

    E2_CLK_HIGH;
    delay10usx(2);

    E2_DATA_LOW;
    delay10usx(2);

    E2_CLK_LOW;
    delay10usx(2);
}

static void i2c_stop(void)
{
    E2_DATA_OUTPUT;

    E2_DATA_LOW;

    E2_CLK_HIGH;
    delay10usx(2);

    E2_DATA_HIGH;
    delay10usx(2);

    E2_CLK_LOW;
    delay10usx(2);

    E2_DATA_LOW;
}

static void send_ack(void)
{
    E2_DATA_OUTPUT;
    E2_DATA_LOW;

    E2_CLK_HIGH;
    delay10usx(2);

    E2_CLK_LOW;
    delay10usx(2);

    E2_DATA_INPUT;
}

static unsigned char i2c_rx(void)
{
    unsigned char result,i;

    E2_DATA_INPUT;

    result = 0;
    for(i = 0; i < 8; i++)
    {
        result <<= 1;
        E2_CLK_HIGH;
        delay10usx(2);
        result |= E2_DATA_PIN;
        E2_CLK_LOW;
        delay10usx(2);
    }
}

```

EK-11. (Devam) ext_E2_driver.c fonksiyonları

```

    }

    return result;
}

static unsigned char i2c_tx(unsigned char data)
{
    unsigned char ack,i;

    E2_DATA_OUTPUT;

    for(i = 0; i < 8; i++)
    {
        if(data & 0x80)
            E2_DATA_HIGH;
        else
            E2_DATA_LOW;

        data <<= 1;

        E2_CLK_HIGH;
        delay10usx(2);

        E2_CLK_LOW;
        delay10usx(2);
    }

    E2_DATA_INPUT;
    delay10usx(2);
    E2_CLK_HIGH;
    delay10usx(1);

    ack = 0;
    if(E2_DATA_PIN)
        ack = 1;

    delay10usx(1);
    E2_CLK_LOW;

    return ack;
}

unsigned char ext_read_byte(unsigned int adr)
{
    unsigned char ack,result;

    E2_VCC_ON;
    delay10msx(1);

    ack = 0;
    i2c_start();
    ack |= i2c_tx(0xA0);
    ack |= i2c_tx((adr & 0xFF00)>>8);
    ack |= i2c_tx(adr & 0x00FF);
    i2c_start();
    ack |= i2c_tx(0xA1);
    result = i2c_rx();

    E2_DATA_OUTPUT; // NAK...
    E2_DATA_HIGH;

    E2_CLK_HIGH;
    delay10usx(2);

    E2_CLK_LOW;
    delay10usx(2);

    i2c_stop();

```

EK-11. (Devam) ext_E2_driver.c fonksiyonları

```

    E2_VCC_OFF;

    return result;
}

void ext_read_sequential(unsigned int adr, unsigned char n, unsigned char *pbuff)
{
    unsigned char ack, i;

    E2_VCC_ON;
    delay10ms(1);

    ack = 0;
    i2c_start();
    ack |= i2c_tx(0xA0);
    ack |= i2c_tx((adr & 0xFF00) >> 8);
    ack |= i2c_tx(adr & 0x00FF);
    i2c_start();
    ack |= i2c_tx(0xA1);

    if(ack != 0)
    {
        E2_VCC_OFF;
        return ;
    }

    for(i = 0 ; i < n-1; i++)
    {
        pbuff[i] = i2c_rx();
        send_ack();
    }

    pbuff[i] = i2c_rx();

    E2_DATA_OUTPUT; // NAK...
    E2_DATA_HIGH;

    E2_CLK_HIGH;
    delay10us(2);

    E2_CLK_LOW;
    delay10us(2);

    i2c_stop();

    E2_VCC_OFF;
}

void ext_write_byte(unsigned int adr, unsigned char data)
{
    unsigned char ack;

    E2_VCC_ON;
    delay10ms(1);

    ack = 0;
    i2c_start();
    ack |= i2c_tx(0xA0);
    ack |= i2c_tx((adr & 0xFF00) >> 8);
    ack |= i2c_tx(adr & 0x00FF);
    ack |= i2c_tx(data);

    i2c_stop();

    delay10ms(); // write cycle time...

    E2_VCC_OFF;
}

```

EK-11. (Devam) ext_E2_driver.c fonksiyonları

```

}

void ext_write_sequential(unsigned int adr,unsigned char n,unsigned char *pbuff)
{
    unsigned char ack,i;

    E2_VCC_ON;
    delay10ms(1);
    i = 0;
again:
    ack = 0;
    i2c_start();
    ack |= i2c_tx(0xA0);
    ack |= i2c_tx((adr & 0xFF00)>>8);
    ack |= i2c_tx(adr & 0x00FF);

    for( ; i < n; i++)
    {
        if ((adr+i)!=0 && (adr+i)%64==0)
        {
            adr+=i;
            i2c_stop();
            delay10ms();    // write cycle time...
            goto again;
        }
        ack |= i2c_tx(pbuff[i]);
    }

    i2c_stop();

    delay10ms();    // write cycle time...

    E2_VCC_OFF;
}

```

EK-12. ext_E2_driver.h tanımlamaları

```
#define E2_DATA_PIN                PORTEbits.RE2
#define E2_DATA_OUTPUT            TRISEbits.TRISE2 = 0
#define E2_DATA_INPUT            TRISEbits.TRISE2 = 1
#define E2_DATA_HIGH              PORTEbits.RE2 = 1
#define E2_DATA_LOW              PORTEbits.RE2 = 0
#define E2_CLK_HIGH              PORTEbits.RE1 = 1
#define E2_CLK_LOW               PORTEbits.RE1 = 0
#define E2_VCC_OFF               PORTEbits.RE0 = 0
#define E2_VCC_ON                PORTEbits.RE0 = 1

unsigned char ext_read_byte(unsigned int adr);
void ext_read_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);
void ext_write_byte(unsigned int adr,unsigned char data);
void ext_write_sequential(unsigned int adr,unsigned char n,near unsigned char *pbuff);

void ext_write_sil(unsigned int adr,unsigned char n,unsigned char *pbuff);
```

EK-13. lcd.c fonksiyonları

```

#include "lcd.h"

static char long_to_ch(long numh,long num)
{
    char rt;
    unsigned char i;

    rt = 1;
    if(numh < 0)
    {
        numh = ~numh; // num *= -1;
        numh++;
        rt = -1;
    }

    if (num<0)
    {
        num = ~num +1; // num *= -1;
    }

    buffer[0]=num%10;
    buffer[1]=(num-buffer[0])/10;

    for(i = 2; i < 11; i++)
    {
        buffer[i] = numh % 10;
        numh-=buffer[i];
        numh /= 10;
    }

    return (rt * i);
}

static void clr_buffer(void)
{
    unsigned char i;

    for(i = 0; i < 11; i++)
        buffer[i] = 0x0a;
}

void display_long(long it,long lg)
{
    char i;

    clr_buffer();
    i = long_to_ch(it,lg);

    Display_String();
}

void display_measure(long tf,long lm,long mk)
{
    char i;
    clr_buffer();
    if(tf<0) tf=~tf+1;
    if(mk<0) mk=~mk+1;
    if(lm<0) lm=~lm+1;
    for(i = 0; i < 5; i++)
    {
        buffer[i] = mk % 10;
        mk-=buffer[i];
        mk /= 10;
    }
    buffer[5]=lm%10;
    buffer[6]=(lm-buffer[5])/10;
    for(i = 7; i <= 10; i++)

```

EK-13. (Devam) lcd.c fonksiyonları

```
{
    buffer[i] = tf % 10;
    tf/=buffer[i];
    tf /= 10;
}
Display_String();
}
```

EK-14. lcd.h tanımlamaları

```
void display_long(long it,long lg);  
extern void Display_String(void);  
extern unsigned char buffer[11];
```

EK-15. lcd_drv.c fonksiyonları

```
#include <p18f4520.h>
#include "global.h"

#define Lcd_Enable          PORTDbits.RD2=0
#define Lcd_Disable        PORTDbits.RD2=1

#define Lcd_Data_High      PORTDbits.RD1=1
#define Lcd_Data_Low       PORTDbits.RD1=0

#define Lcd_Wr_Low         PORTDbits.RD0=0
#define Lcd_Wr_High        PORTDbits.RD0=1

#define Lcd_Vcc_On         PORTCbits.RC2=1
#define Lcd_Vcc_Off        PORTCbits.RC2=0

unsigned char index,index1;
unsigned int int_index;

unsigned char buffer[11];
rom unsigned char tablo[12]={0xee,0x48,0xd6,0xda,0x78,0xba,0xbe,0xc8,0xfe,0xfa,0x00,0xff};

extern union
{
    unsigned char byte;
    struct
    {
        unsigned char l_of_k:1;
        unsigned char c_of_k:1;
        unsigned char slash:1;
        unsigned char W:1;
        unsigned char h_down:1;
        unsigned char h_up:1;
        unsigned char _3_of_m3:1;
        unsigned char m_of_:1;
    };
}display_signals_1;

extern union
{
    unsigned char byte;
    struct
    {
        unsigned char y_of_M:1;
        unsigned char penalty:1;
        unsigned char error:1;
        unsigned char rf:1;
        unsigned char no_flow:1;
        unsigned char battery_low:1;
        unsigned char s:1;
        unsigned char j:1;
    };
}display_signals_2;

void Lcd_Command(unsigned char s,unsigned char f);

void Sleep_LCD(void)
{
    Lcd_Enable;
    Lcd_Data_High;
    Lcd_Command(0x02,9); // com(0x02);
    Lcd_Command(0x00,9); // com(0x00);
    Lcd_Disable;
}

void wakeup_LCD(void)
{

```

EK-15. (Devam) lcd_drv.c fonksiyonları

```

    Lcd_Enable;
    Lcd_Data_High;
    Lcd_Command(0x01,9); // com(0x02);
    Lcd_Command(0x03,9); // com(0x00);
    Lcd_Disable;
}

void Lcd_WR(unsigned char s,unsigned char n)
{
    for (index=0;index<n;index++)
    {
        Lcd_Wr_Low;
        Delay10TCYx(1);
        if( s & 0x80 ) Lcd_Data_High;
        else Lcd_Data_Low;
        s<<=1;
        Lcd_Wr_High;
        Delay10TCYx(1);
    }
}

void Lcd_Command(unsigned char s,unsigned char fg)
{
    if(fg==6){Lcd_Enable; Lcd_WR(0xa0,3);} // aktarma=0xA0; //DatB
    else if(fg==9){Lcd_Enable; Lcd_WR(0x80,3);} // aktarma=0x80; //Com
    Lcd_WR(s,fg);
    if(fg==9) Lcd_Disable;
}

void Display_Segment(void)
{
    Lcd_Command(0x00,6);
    for(index1=11;index1>=1;index1--) // segmentler
    {
        if (index1==6) buffer[index1-1]=0x01; // nokta
        Lcd_Command(buffer[index1-1],8); //*****
    }
    Lcd_Command(display_signals_1.byte,8);
    Lcd_Command(display_signals_2.byte,8);

    Lcd_Disable;
}

void Display_String(void)
{
    unsigned char i;
    for (i=0;i<11;++i)
        buffer[i]=tablo[buffer[i]];
    Display_Segment();
}

void Display_Char(unsigned char seg,unsigned char r7)
{
    Lcd_Command(seg,6);
    Lcd_Command(r7,8);
    Lcd_Disable;
}

void Clear_Lcd(void)
{
    unsigned char index1;
    Lcd_Command(0x00,6);
    for(index1=0;index1<32;index1++) Lcd_Command(0,8);
    Lcd_Disable;
}

```

EK-15. (Devam) lcd_drv.c fonksiyonları

```

void init_lcd(void)
{
    Lcd_Vcc_On;
    delay10msx(5);
    Lcd_Wr_Low;
    Delay10TCYx(8);
    Lcd_Disable;
    Delay10TCYx(8);
    Lcd_Wr_High;
    Delay10TCYx(8);
    Lcd_Enable;
    Delay10TCYx(8);
    Lcd_Command(0x01,9); // com(0x01); //sys en
    delay10msx(1);
    Lcd_Command(0xE3,9); // com(0xE3); ///normal mode
    Lcd_Command(0x03,9); // com(0x03); ///lcd on
    delay10msx(1);
}

void Display_Versiyon(void)
{
    unsigned char i;
    for (i=0;i<11;++i)
        buffer[i]=0x00;
    buffer[3]=0x48;           //"I"
    buffer[4]=0xee;          //"O"
    buffer[7]=0x26;          //"L"
    buffer[8]=0x6E;          //"U"
    Display_Segment();
}

```

EK-16. global.h tanımlamaları

```
#include <delays.h>
extern void delay10us(void);
extern void delay10usx(unsigned char t);
extern void delay1ms(void);
extern void delay1msx(unsigned char t);
extern void delay10ms(void);
extern void delay10msx(unsigned char t);
extern void delay10msx_critical(unsigned char t);

typedef enum _BOOL_ {FALSE=0,TRUE} BOOL;
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ERTAŞ, Emre
 Uyuğu : T.C.
 Doğum tarihi ve yeri : 04.05.1981 Ankara
 Medeni hali : Bekar
 Telefon : 0 (312) 322 02 05
 Faks :
 e-mail : ertasemre@yahoo.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniv./ Elk-Elktro Müh.	2006
Lisans	Kırıkkale Üniv./Elk-Elktro Müh.	2003
Lise	Kanuni Lisesi	1998

İş Deneyimi

Yıl	Yer	Görev
2003-2004	Orsis A.Ş	Tasarım mühendisi
2004-2005	Randa Ltd, Şti.	Tasarım mühendisi
2005-2006	Elektromed Ltd. Şti.	Tasarım mühendisi

Yabancı Dil

İngilizce

Hobiler

Futbol, Yüzme, Bilgisayar teknolojileri, Basketbol

