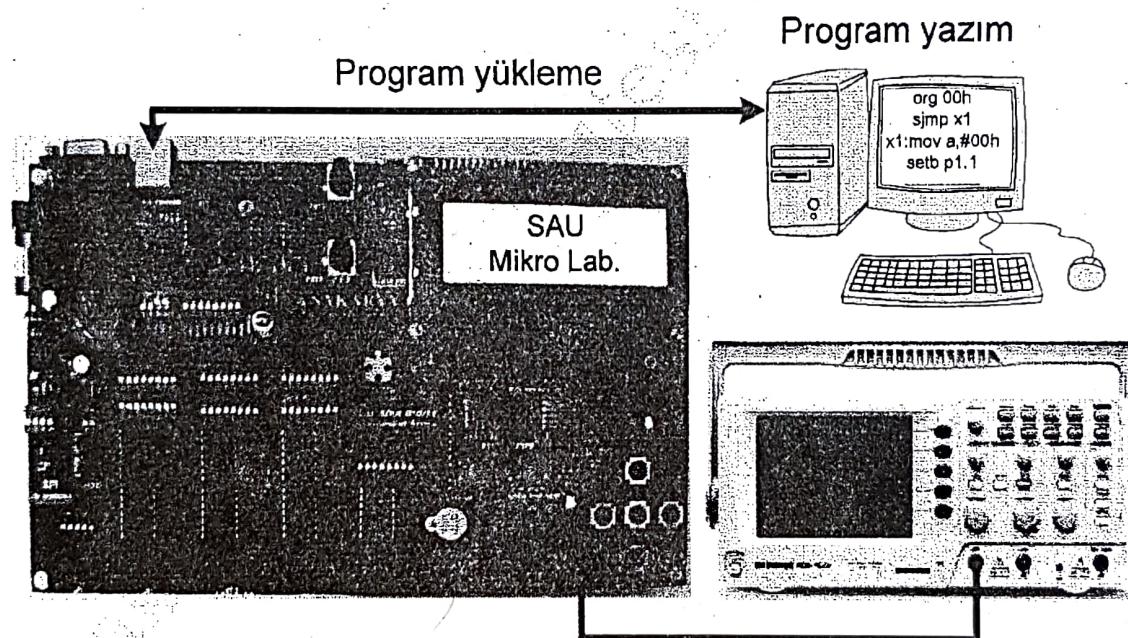


SAKARYA ÜNİVERSİTESİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

MİKROİŞLEMCİLER VE SAYISAL İŞARET İŞLEME
LABORATUVARI- EEM 473

MİKROİŞLEMCİLER
LABORATUVARI DENEY FÖYÜ



2018

DENEY TAKVİMİ – 2018

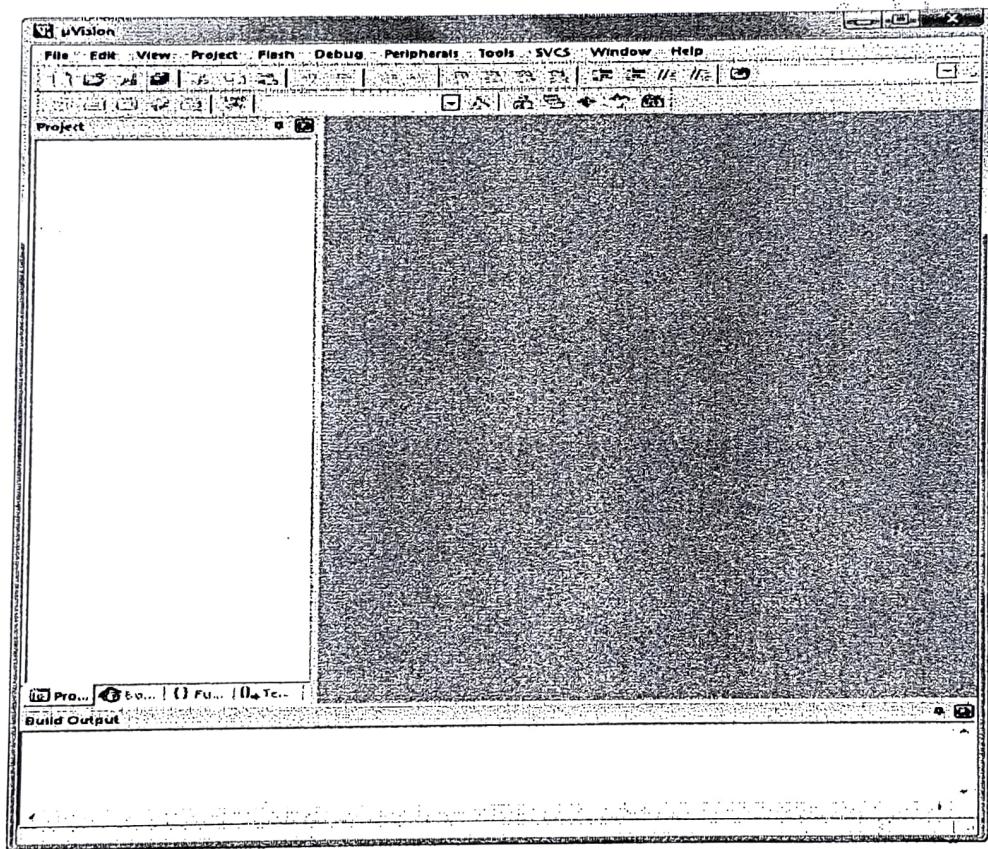
Deney No	Deney Adı	Tarih
-	TANITIM, BİLGİLENDİRME VE GRUP OLUŞTURMA	28.09.2018
DENEY 1	LCD EKRAN SÜRÜLMESİ	05.10.2018
DENEY 2	STEP MOTOR SÜRÜLMESİ	12.10.2018
DENEY 3	ANALOG SAYISAL ÇEVİRİCİLER (ADC)	19.10.2018
DENEY 4	SAYISAL ANALOG ÇEVİRİCİ (DAC) İLE ÇİFT TON ÇOK FREKANSLI (DTMF) İŞARET ÜRETİLMESİ	26.10.2018
DENEY 5	BİR TRANSFER FONKSİYONUNUN AYRIKLAŞTIRILARAK ADUC841 İLE GERÇEKLEŞTİRİLMESİ VE PWM MODÜLÜNÜN KULLANILMASI	02.11.2018

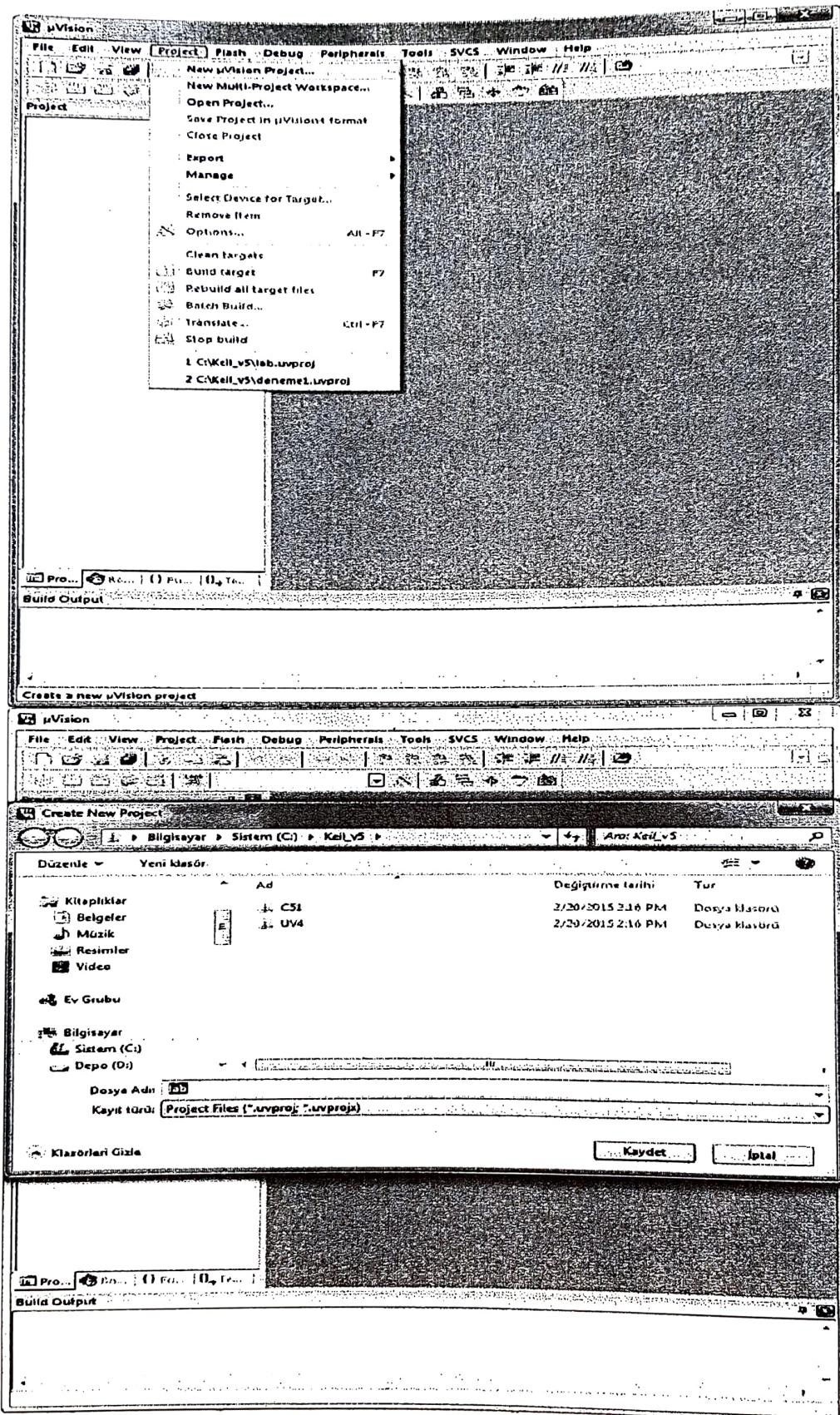
DENEYLERDE DİKKAT EDİLMESİ GEREKEN HUSULAR:

1. Deneye **10 dk'dan** fazla geç kalan öğrenci deneye alınmaz.
2. Deneyler **M4 binası zemin katta** Mikroişlemciler Lab.'da yapılacaktır.
3. Öğrenciler ilgili deneyin **ön çalışmasını** deney sırasında gerçekleştirmek zorundadırlar.
4. Her öğrenci **kendi deney foyünü** yanında getirmelidir.

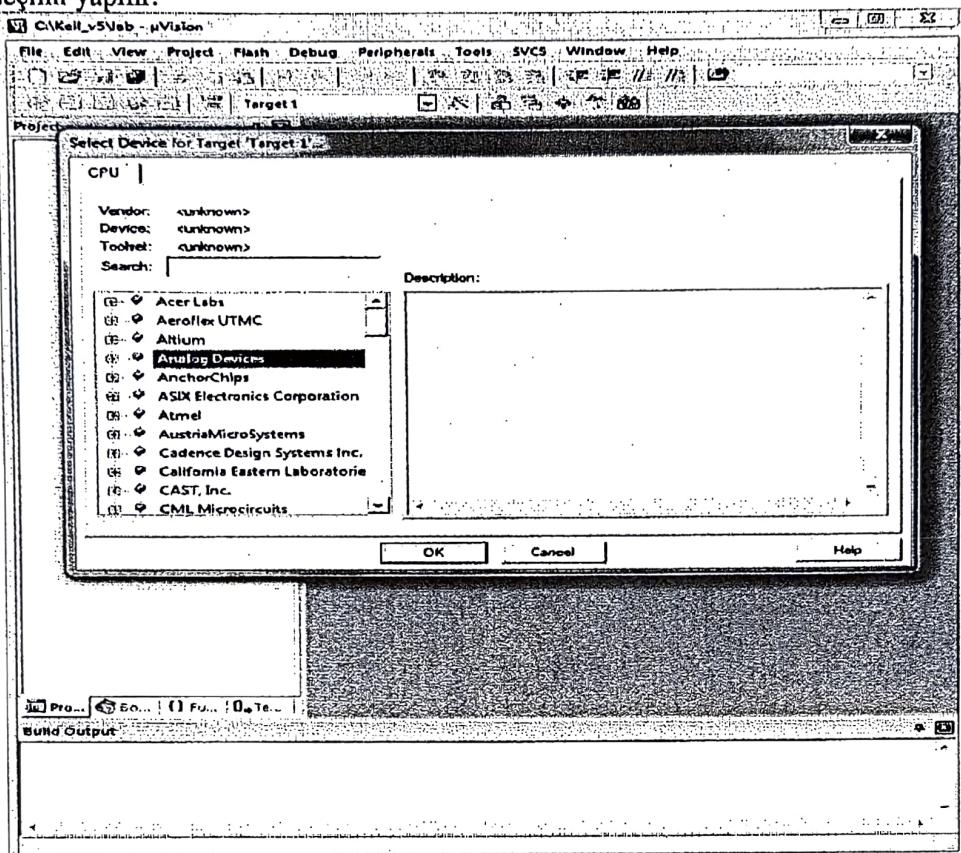
KEİL UVİSİON 5 KULLANILARAK ADUC841 MİKROKONTROLÖRÜNE PROGRAM YÜKLENMESİ

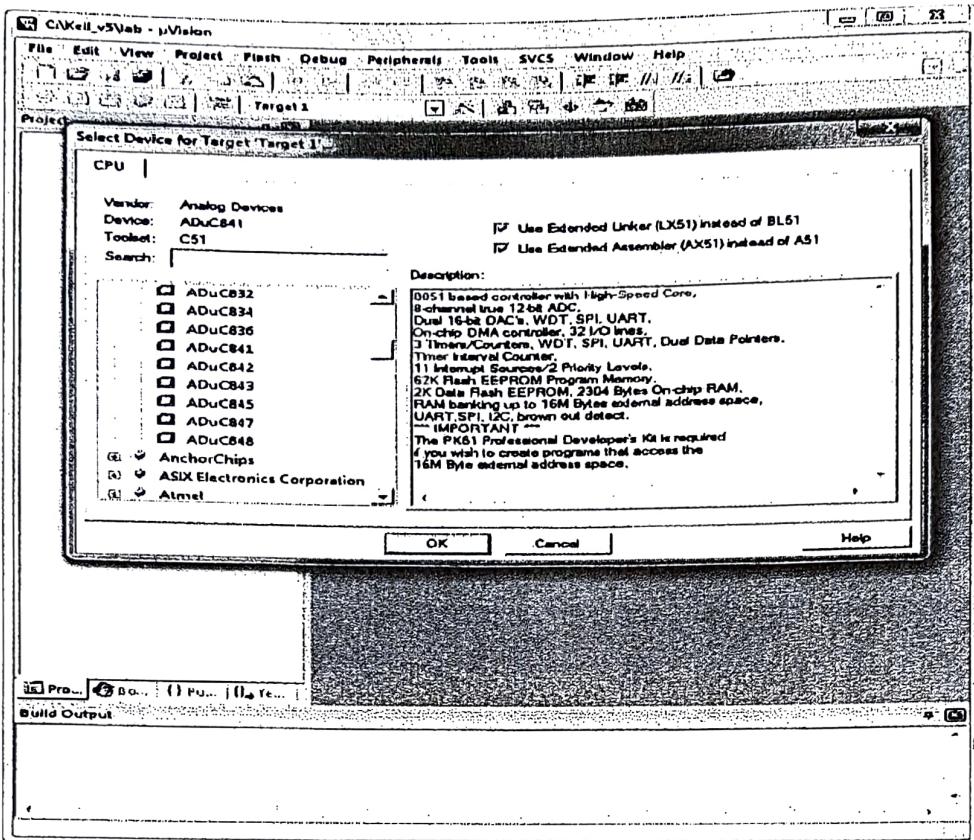
Keil uvision 5 programı ilk açıldığında aşağıdaki gibi bir ekran görüntüsü elde edilir. Yeni proje oluşturulması ve diğer işlemler için yapılması gereken işlemler görsel olarak sırasıyla aşağıdaki gibi verilmiştir.



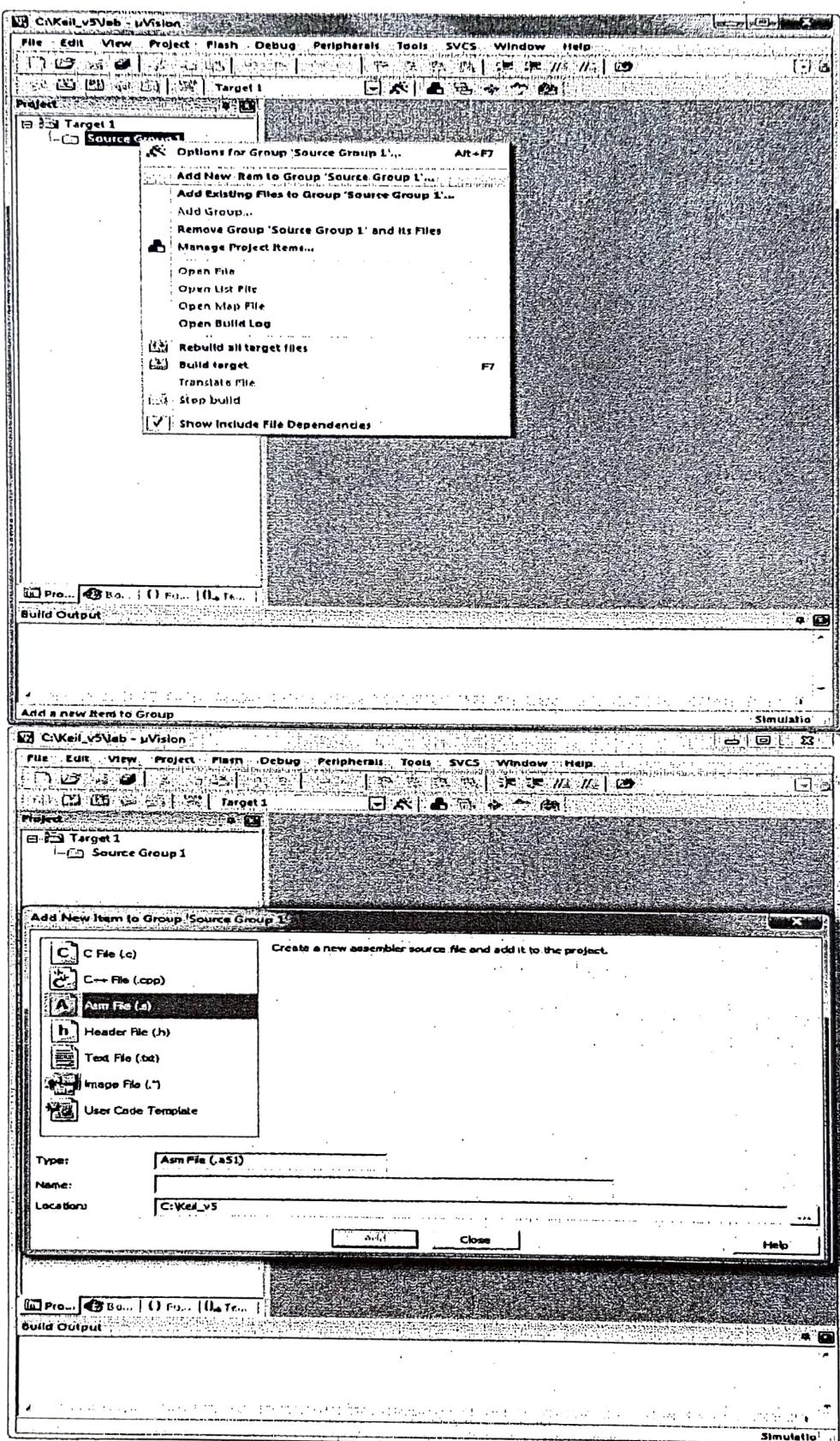


Bu işlemden sonra aşağıdaki gibi bir pencere gelir ve buradan Analog Devices-Aduc841 seçimi yapılır.

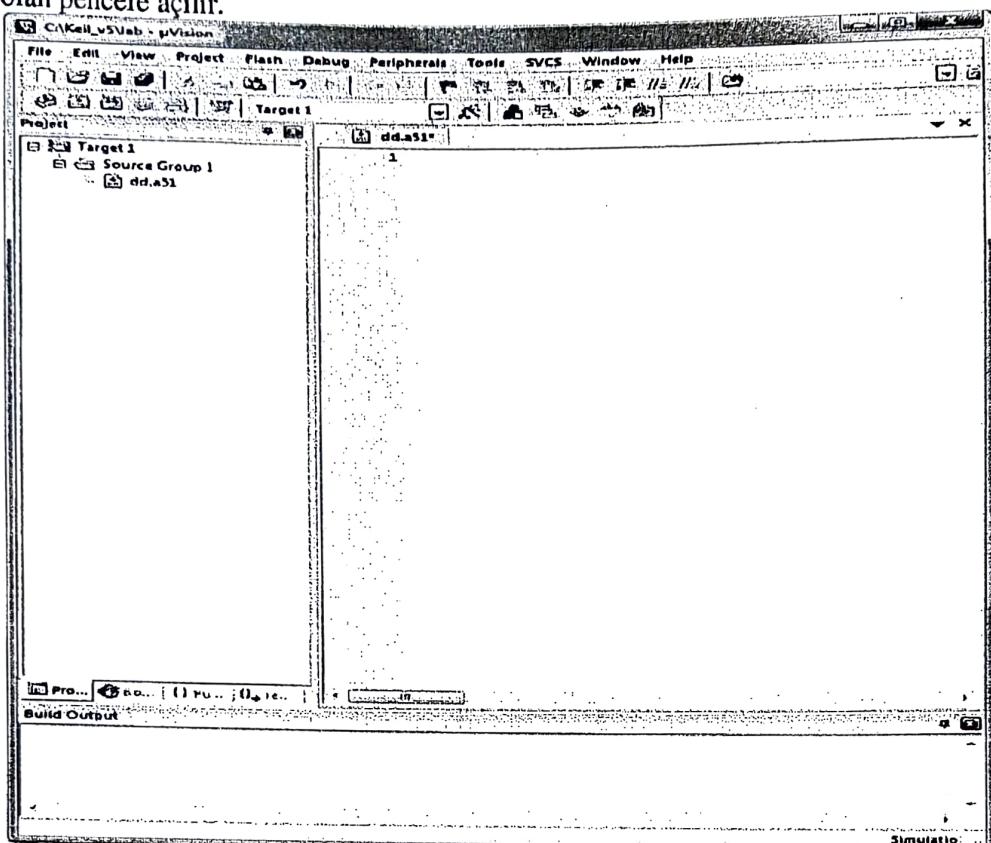




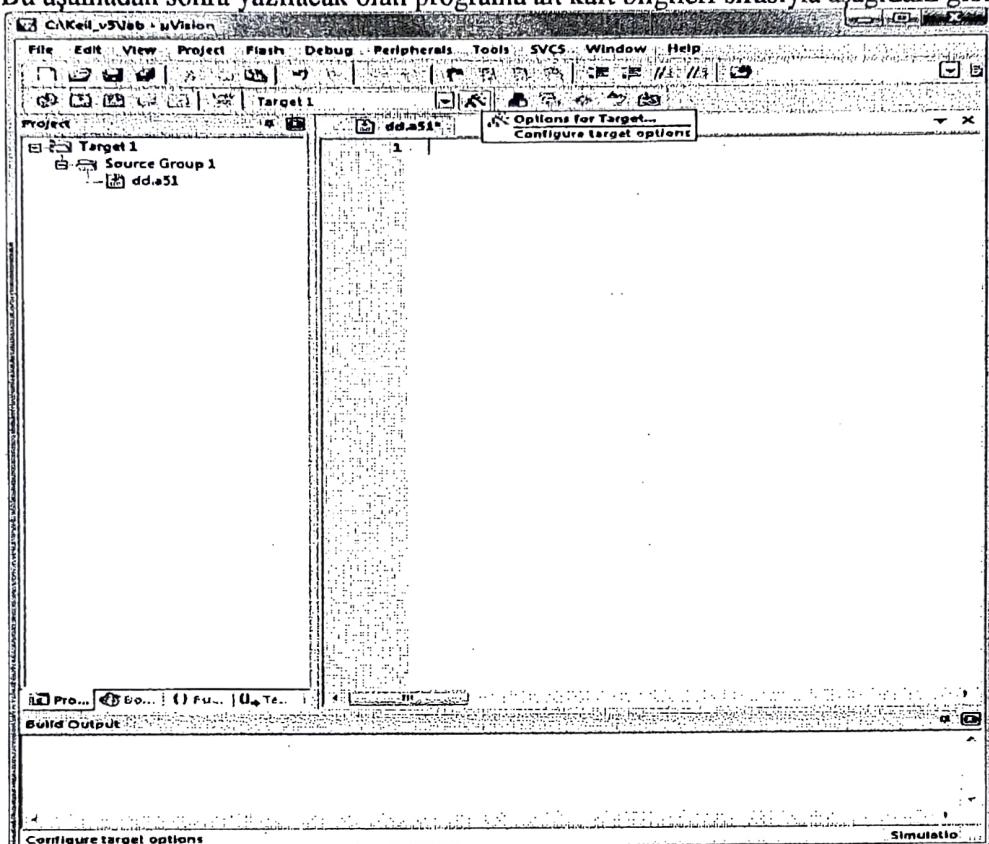
Bu seçimlerin yapılması ile birlikte yukarıdaki görüntüde sağ üst bölümdeki işaretlemeler unutulmamalıdır. "OK" tıklanıldığında "Project" kısmında "Target" oluşturulmuş olur.



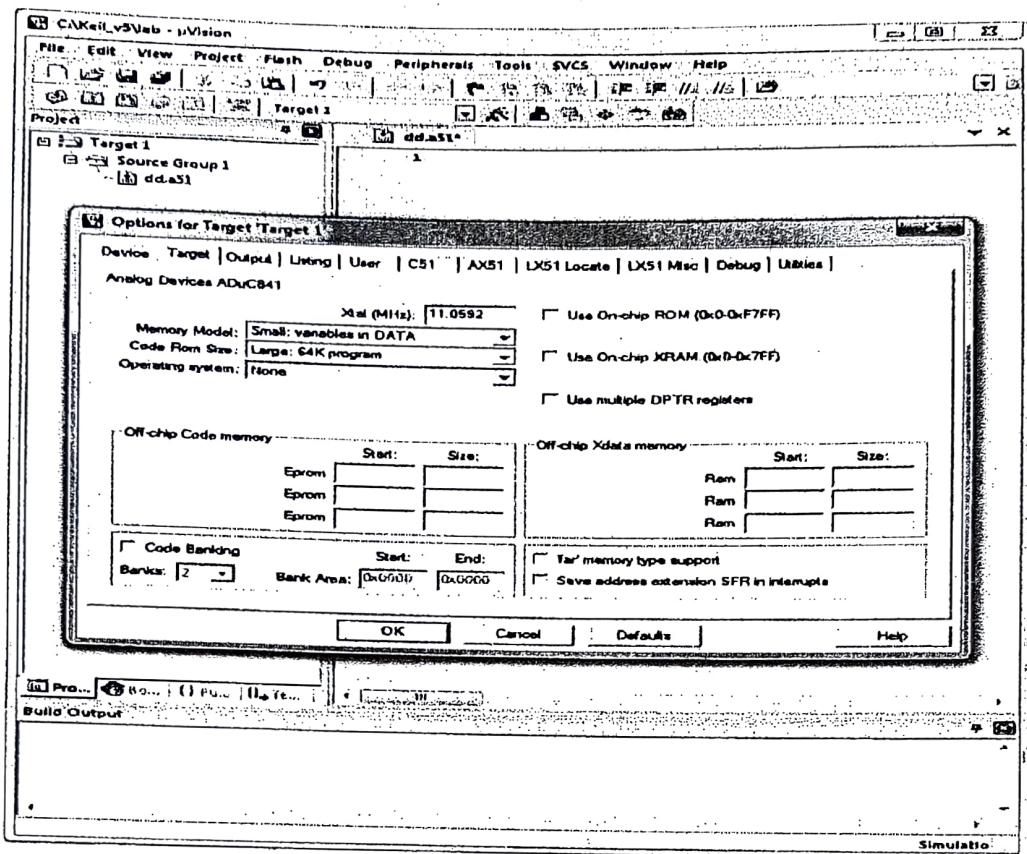
Asm file seçilir ve "Name" kısmında dosya isimlendirilir. "add" tıklanarak program yazılacak olan pencere açılır.



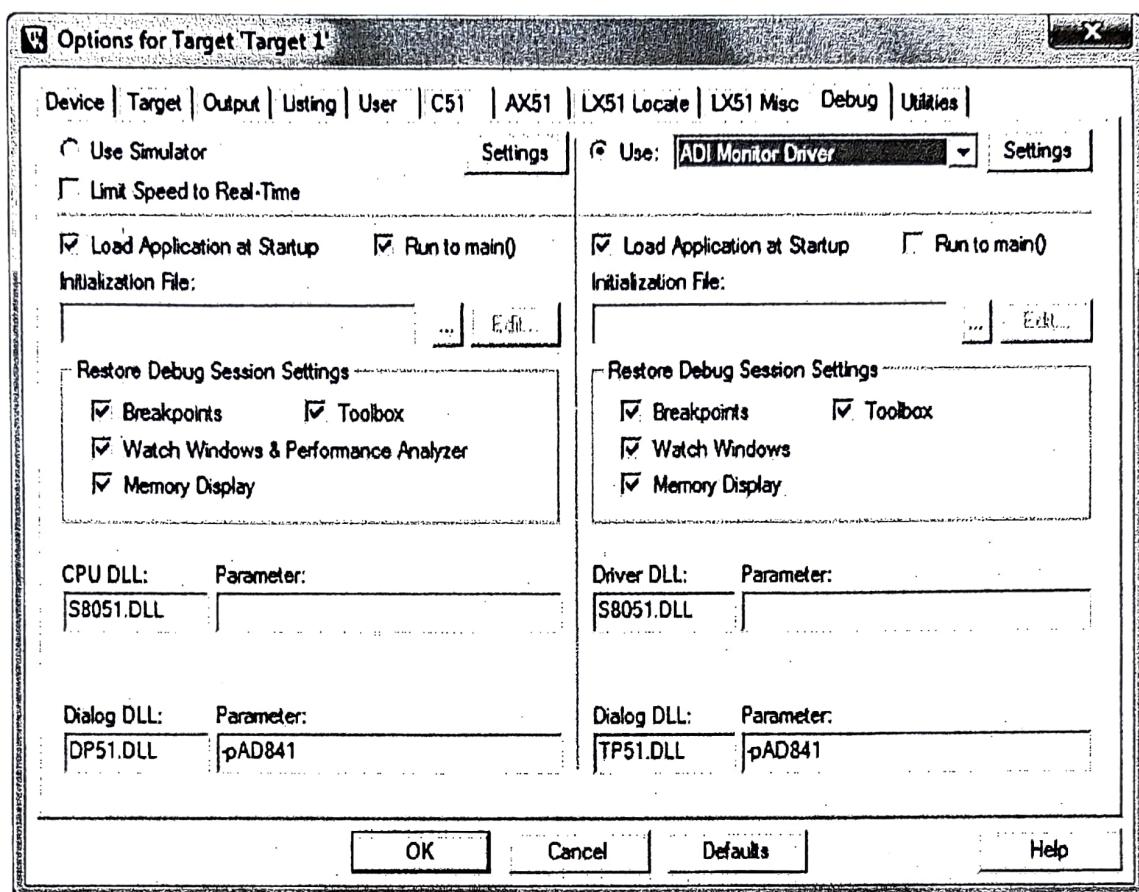
Bu aşamadan sonra yazılacak olan programa ait kart bilgileri sırasıyla aşağıdaki gibi ayarlanır.



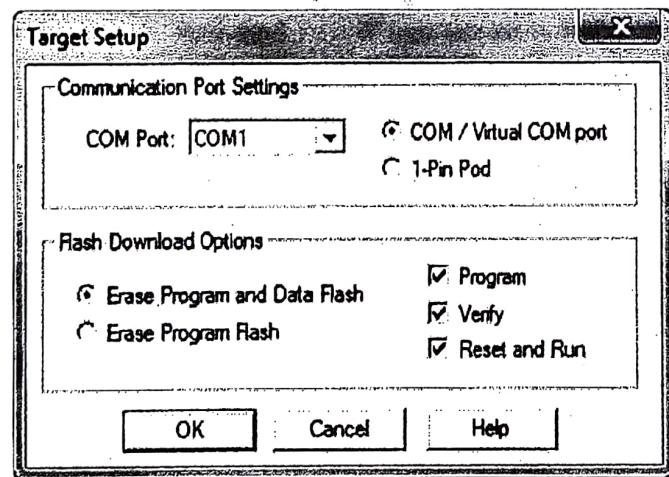
Default değeri 12 MHz olan kismı kart üzerindeki kristal frekansı olan 11.0592 MHz olarak ayarlanır.



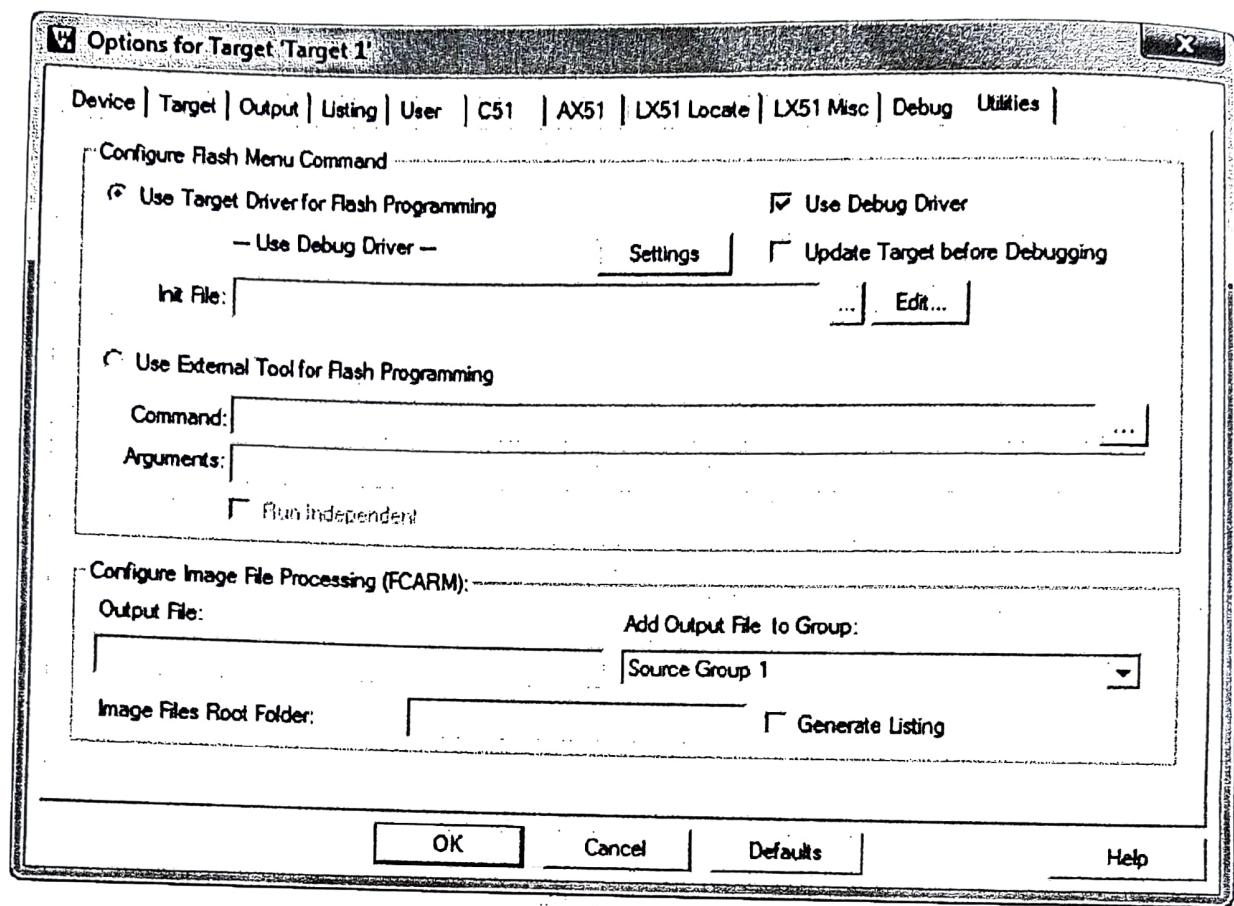
Aşağıda "Simulator" ve gerçek uygulamalar için opsiyonlar mevcuttur. Bu kısımda aşağıdaki gibi seçim yapılip gerçek uygulama yapılacağı belirtilir.

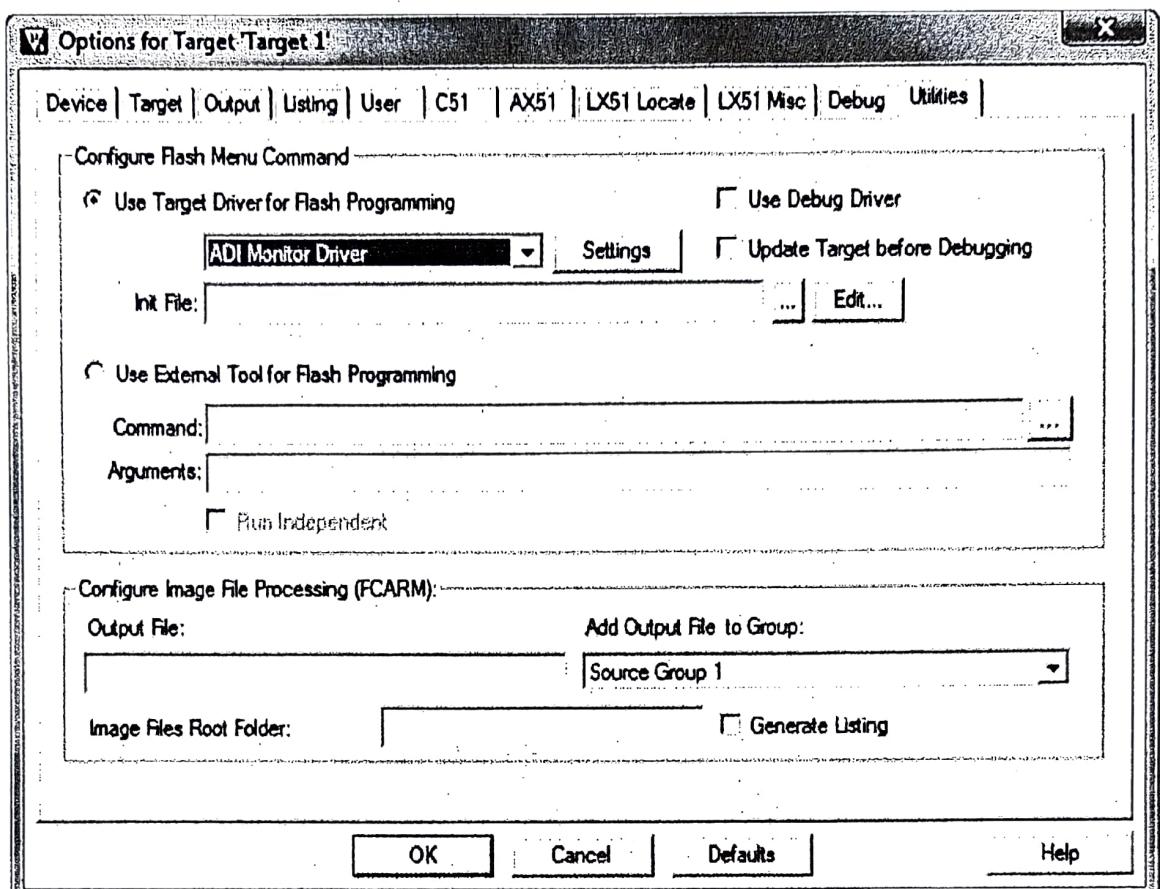


Buradaki "COM" ayarı Bilgisayarım-Özellikler-Aygıt yönetici-Bağlantı noktaları kısmından USB Serial Port'un bağlı olduğu COM adresine göre seçilir.

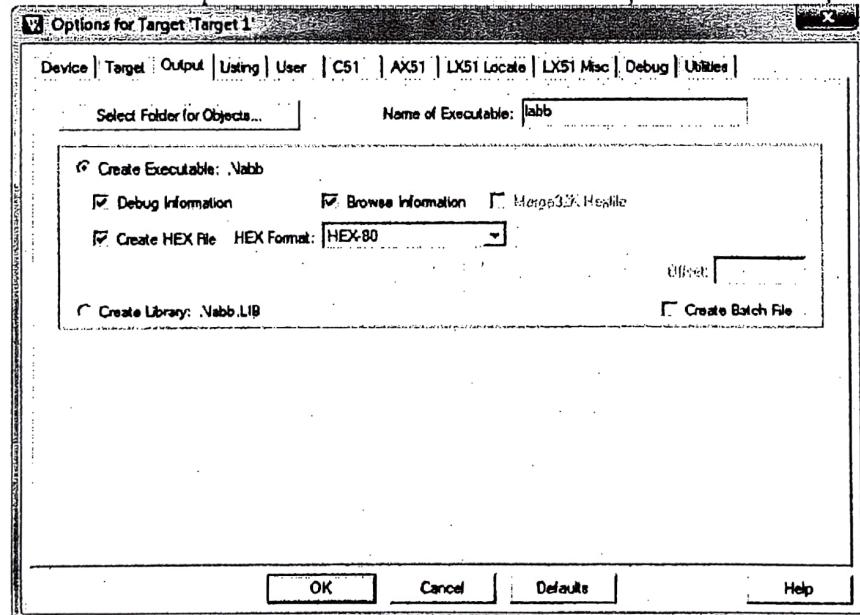


Aşağıdaki iki şekilde ise seçim olarak iki opsiyon mevcuttur. İki opsiyonda yapılabilir.





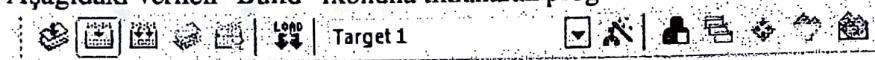
Daha sonra "Output" ikonuna tıklanarak HEX file oluşturulması aktifleştirilir.



Bu ayarlardan sonra "OK" tuşuna basılarak program penceresine gereken program yazılır. Aşağıda basit bir örnek program yazılmıştır. Bu programın en üst satırında yer alan kod "Header File" tanımlaması için kullanılır. Her program başında yer olması önerilir. Aksi halde program kullanılan bazı SFR etiketlerini, komutları vs. tanımlayamayabilir.

```
1 #include    <aduc841.h>
2 org 00h
3 sjmp basla
4 basla:
5 mov R1,$00h
6 clr P2.3
7 mov R0,$255d
8 xx: mov R1,$255d
9 x: djnz R0,x
10 mov R0,$255d
11 djnz R1,x
12 cpl P0.3
13 sjmp xx
D 14 end
```

Aşağıdaki verilen “Build” ikonuna tıklanarak program derlenir.



Pencerenin alt kısmında yer alan “Build Output” penceresinde aşağıdaki gibi hata ve uyarı mesajları yer alır. Bu ekranda hata mesajı tıklanıldığında program üzerinde hatalı olan satır ok işaretleri ile belirtilir.

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The Project window on the left shows Target1 with a Source Group containing aaa.asm. The main code editor window displays the following assembly code:

```
1 #include <aduc841.h>
2 org 00h
3 sjmp basla
basla:
4 mov p, #00h
5 clr p2.3
6 mov r0, #255d
7 xx: mov r1, #255d
8 x: djnz r0, x
9 mov r0, #255d
10 djnz r1, x
11 cpl p0.3
12 sjmp xx
13
14 end
```

The Build Output window at the bottom shows the assembly process and an error message:

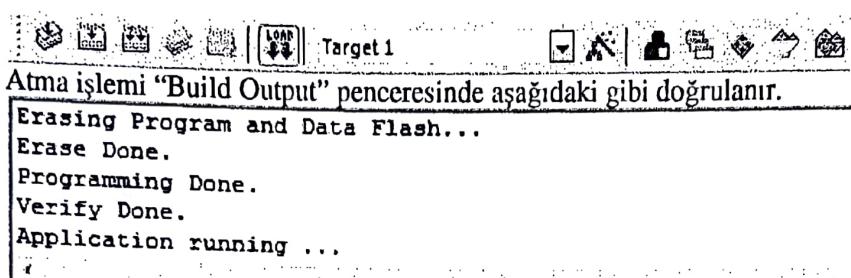
```
Build target 'Target 1'
assembling aaa.asm...
aaa.asm(5): error A48: DATA-ADDRESS EXPECTED
Target not created.
```

Hata düzeltilerek tekrar derleme işlemi yapılır.

The screenshot shows the Keil uVision IDE interface after the error was fixed. The assembly code remains the same as in the previous screenshot. The Build Output window now shows a successful build process:

```
assembling aaa.asm...
linking...
Program Size: data=8.0 xdata=0 const=0 code=21
Creating hex file from "labb"...
"labb" - 0 Error(s), 0 Warning(s).
```

Programın derlenmesi ile oluşturulan **hex** dosyasının karta yüklenmesi için kart üzerinde **PSEN** butonuna basılı tutarak **RST butonuna** basılır. Sonra sırasıyla **RST ve PSEN** butonu bırakılarak karta reset atılır. Aşağıdaki “Load” ikonu tıklanarak karta derleyicide yazılmış olan program yüklenir.



Böylelikle karta yüklenmesi gereken adımlar tamamlanmış ve örnek program karta yüklenmiştir.

Ayrıca kartın tanıtıması ve programın yüklenmesi ile ilgili aşağıdaki youtube linkleri izlenebilir.

<https://www.youtube.com/watch?v=7LzId7M7v6w>

<https://www.youtube.com/watch?v=TWAE4C1BCks>

Deneysel 1: LCD Ekranın Sürülmesi ve FLASH/EE Hafızasının Kullanılması

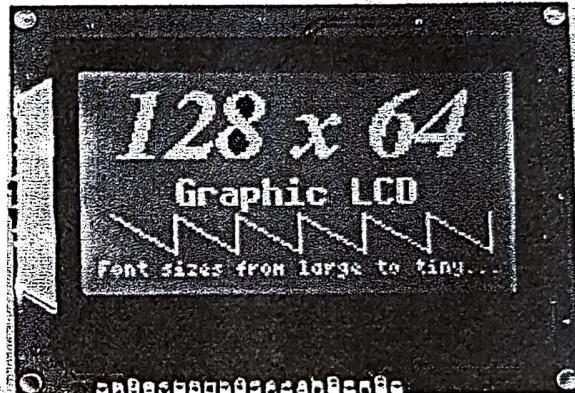
1. Deneysel Amacı

- a. Kullanıcı ve mikrokontrolör arasında bir arayüz elamanı olan LCD'lerin sürülmeleri
- b. ADUC841 mikrokontrolör mimarisinde bulunan 4 kByte lik Flash/EE veri hafızanın kullanılması
- c. Veri tabanı (Database, DB) kullanılması
- d. Timer0 kesmesi kullanılması

2. LCD (Liquid Crystal Display) Ekranın Tanıtılması



Şekil 1: Deneye kullanılacak olan LCD 2x20



Şekil 2: Grafik LCD

LCD bir görüntüleme teknolojisidir. Bu teknolojiyi kullanan cihazlar ise LCD gösterge olarak adlandırılmışlardır. 7-Segment (7 Parçalı Gösterge) göstergelerin fazla akım çekmesi ve kullanım zorluğu nedeniyle, son yıllarda LCD göstergelerin kullanımı popüler hale gelmiştir.

Bu deneyde kullanacağımız LCD (ve çoğu LCD), aslında sadece bir göstergeden ibaret değildir. Bünyesinde,

1. Kendisine ait bir mikroişlemcisi,
2. RAM ve ROM'LARI,
3. Giriş çıkışları,

olan bir cihazdır. Yani bir nevi bütünlesik mikrodenetleyici+LCD cihazdır.

LCD'ler 1 satırda 4 satırda kadar, 16 karakterden 80 karaktere kadar ve 5X7, 5X10 nokta font gibi değişik ölçütlerde üretilip satılmaktadır. Bazılarda ise tüm ekran tek bir karakter gibi yapılandırılmıştır, bu türlerine grafik LCD gösterge adı verilmektedir. LCD gösterge ile iletişim, TTL standardında 4 veya 8 veri hattı ile yapılır. 4 bit iletişim G/C hatlarının başka işler(görevler, amaçlar) için kullanımını kolaylaştırırken, iletişim süresini iki kat uzatmaktadır.

2.2. LCD Hafıza Haritası (Memory Map)

LCD göstergeler üzerinde kullanılan denetleyiciler, Hitachi firması tarafından üretilen HD44780U mikrodenetleyicisidir ve bu mikrodenetleyici standart bir hale gelmiştir.

LCD'lerin mimarisinde 3 adet hafıza yapısı bulunmaktadır. Bunlar DDRAM, CGROM ve CGRAM'dır.

DDRAM: LCD göstergeler, 40 karaktere ve 4 satırda kadar değişik seçenekler sunar. LCD göstergeler 80 adet karaktere kadar kodu saklayabilmek için dâhili bir RAM bulundururlar. Bu RAM'a Gösterge Veri RAM'i(Display Data RAM-DDRAM) denir. Örneğin bir satırda 16 karakteri olan iki satırlık bir göstergeyi,

SAU / Muh.Fak./ EEM/ LCD Ekranın Sürülmesi

her birinde 40 karakteri olan 2 sanal satır olarak düşünülebiliriz. 40 karakterlik bir satır bulunmaktadır ancak biz onu 16 karakterlik bir pencere ile görebilinmektedir. Sanal satırındaki diğer karakterleri görebilmek için gösterge kaydırılmalıdır. Örneğin, bu göstergenin birinci satırına aşağıdaki 40 karakterli diziyi yazalım.

9876543210QWERTYUIOPLKJHGFDASZXCVBNMsedn

Göstergede sadece ilk 16 tanesi gözükecektir.

9876543210QWERTY

Bu gösterimde, aşağıda bahsedileceği gibi Entry (Giriş) moduna bağlı olarak ekran kayabilir veya kaymayabilir. Burada ekran kaydırılmamıştır. "U" karakterini görüntülemek istediğimizde, aşağıdaki gibi ekran bir karakter sağa kayacak ve "9" karakteri gizlenecektir.

876543210QWERTYU

CGROM: LCD göstergede önceden programlanmış veya kullanıcı tarafından tanımlanan karakterleri tanımlanan gösterebilmektedir. LCD kontrolörü, 192 adet karakter içeren bir Karakter Üretici ROM'a (Character Generator ROM – CGROM) sahiptir. Karakterler belirli kodlarla seçilirler. Bu karakterlerden 96 tanesi ASCII karakter (ASCII kodları seçilir), 64 adeti japon karakterleri (Kana Alfabesi) ve 32 adeti Yunan harfleri gibi özel karakterlerdir.

CGRAM: LCD kontrolörü aynı zamanda, Karakter Üretici RAM (Character Generator RAM – CGRAM) olarak adlandırılan ve kullanıcı tarafından tanımlanabilen 8 karakter içerebilen bir hafızaya sahiptir. Bu karakter kullanılmadan önce tanımlanmalı, CGRAM'a yüklenmeli ve gösterilmek için gereken yerlerde çağrılmalıdır.

ASCII KOD TABLOSU: CGROM hafızasında dâhilî olarak bulunur. Latin alfabesi üzerine kurulu 7 bitlik bir karakter setidir. ASCII'de 33 tane basılmayan (ekranda görülmeyen) kontrol karakteri ve 95 tane basılan (ekranda görülen) karakter bulunur. Kontrol karakterleri metnin akışını kontrol eden, ekranda çıkmayan karakterlerdir. Basılan karakterler ise ekranda görünen, okuduğumuz metni oluşturan karakterlerdir.

Tablo 1: ASCII Kod Tablosu

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	!	"	#	%	%	*	*	()	*	+	,	-	-	/	
3	0	1	2	3	4	5	6	7	8	9	:	:	<	=	>	?
4	8	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
7	p	q	r	s	t	u	v	w	x	y	z	{	}	-	-	

Tablo 1'den bir karakterin ASCII kodunu bulmak için önce karakterin bulunduğu satır numarası sonradan sütün numarasına bakılır. Aşağıda örnek olarak çeşitli karakterlerin ASCII kodları verilmiştir.

Karakter	ASCII Kodu
A	41'H
a	61'H
5	35'H
=	3D'H

LCD Ekranın Adreslenmesi

(LCD Göstergeye bu dokümdan bundan sonra kısaca LCD denilecektir.)

SAU / Muh.Fak./ EEM/ LCD Ekrannın Sürülmesi

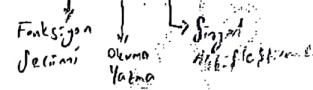
LCD'lerde aşağıda gösterildiği gibi her karakterin ayrı bir adresi vardır. Aşağıdaki resimlerde 2x16 ve 4x20 karakter boyutundaki LCD'lerin her karakter için belirlenen adresleri verilmiştir.

2x16	80	81	•	•	•	•	8E	8F	4x20	80	81	•	•	•	•	•	92	93
	C0	C1	•	•	•	•	CE	CF		C0	C1	•	•	•	•	•	D2	D3

4x20

94	95	•	•	•	•	•	96	97	A6	A7
D4	D5	•	•	•	•	•	E6	E7		

Değişik firmalar tarafından üretilen LCD'lerde, LCD ekrannın sürülebilir devresini de içerir. Dolayısıyla ilave devre kurmadan doğrudan mikrokontrolör ile LCD modüller sürülebilir. LCD'ler farklı firmalar tarafından üretilmelerine rağmen erişim protokollerinin çoğunluğu aynıdır. LCD'lerin 3 kontrol hattı (RS, R/W, E) ve 8 veri hattı (DB0...DB7) vardır.



2.2. LCD Bağlantı Uçları

Şekil 2'de görüldüğü gibi günümüzde üretilen LCD panelerinin çoğunda tek sıra halinde 16 pin bulunur. Bazı LCD'lerde kontrol için kullanılan 14 pin 2 adet 7'li sıra halinde de bulunabilir. LCD bağlantı uçları Tablo 2'de verilmiştir. Bağlantı uçlarını besleme, denetim ve veri olmak üzere üç grupta inceleyebiliriz.

Tablo 2: Pinlerin Görevleri

Pinlerin Görevleri		
No	Sembol	Fonksiyon
1	Vss	GND
2	Vdd	+5V
3	Vee	LCD Sürmek için
4	RS	Fonksiyon Seçimi
5	R/W	Okuma/Yazma
6	E	Sinyal Aktifles.
7	DB0	Komut veya Veri Yolu
8	DB1	Komut veya Veri Yolu
9	DB2	Komut veya Veri Yolu
10	DB3	Komut veya Veri Yolu
11	DB4	Veri Yolu
12	DB5	Veri Yolu
13	DB6	Veri Yolu
14	DB7	Veri Yolu
15	LEDA	İşik +5v
16	LEDA	İşik 0V

2.1.1. Besleme Gerilimleri

HD44780U standardında besleme ile ilgili üç uç yer almaktadır. Bunlar Vcc, Vee (V0), Vss (GND)'dır. Vss ve Vcc standart TTL gerilimi 0 ve 5 Volt'tur. Vee ise ekranın parlaklığını belirleyen bir gerilimidir ve değeri en çok Vcc'dir.

2.1.2. Kontrol ve Veri Pinleri

LCD'yi kontrol etmek amacıyla üç uç yer almaktadır. Bunlar RS, R/W, E uçlarıdır. Ayrıca 8 tane de veri ucu bulunmaktadır. Bunlar;

Tablo 3: Kontrol Uçları

Sembol	Görevi
RS	LCD'ye komut mu yoksa veri mi gönderileceğini belirler. LCD ekrana veri aktarılacaksa RS= 1, komut gönderilecekse RS= 0
R/W	LCD ekranı silme, kursör on/off, kursör başa dön, yazma başlangıç adresinin belirtilmesi gibi işlemler komut olarak adlandırılır. LCD'lerde kullanılan komutlar ve ilgili komutlar için pin değerleri aşağıda tablo halinde verilmiştir. LCD ekranına yazılan (örneğin "SAU" "12+3=15", vb.) değerler ise veri olarak adlandırılır. Lcd den okuma mu yoksa lcd ye yazma yapılacağını belirler. Lojik R/W=1 seviyesi LCD'lerden okuma, lojik R/W=0 ise LCD'ye yazma işlemini gösterir. Deneylerde LCD'den okuma işlemi yapılmayacağı için bu pin Şekil 1 de gösterildiği gibi

E	donanımsal olarak GND pinine bağlanarak Lojik 0 seviyesinde tutulmuştur. Enable (Aktifleştirme) ucu LCD ve pinler arasındaki gerçek veri alışverişini sağlayan bacaktır. Bu girişi mikrodenetleyiciye program aracılığıyla tanıttıktan sonra mikrodenetleyici kendisi veri gönderileceği zaman bu bacağa enable (aktifleştir) darbesi gönderir. Yani bu uca 0-1-0 darbesi üretilir.
DB0-DB7	Data hattı olan bu pinler doğrudan mikrodenetleyicinin bir portuna bağlanır. Veri 4 ya da 8 bitlik veri yolu ile gönderilebilir.

2.2. LCD Komut Tipleri ve Zaman Çizelgesi

Bir LCD işlemi, ya kontrol ya da veri işlemidir. Kontrol işlemleri, ya LCD'ye gönderilen komutlardır, ya da LCD'den okunan bir hafıza adresidir (kaydedici-register). RS (Register Select) ucu lojik 0'a çekilirse yapılacak işlem, kontrol işlemidir. Bütün komutlar ve sinyaller, harici bir mikrodenetleyici, bilgisayar vb. tarafından üretilir. Komutlar LCD'ye 8-bit olarak yazılır. Bunun için R/W lojik 0'da tutulmalıdır. Bu uç lojik 1'e çekilirse, LCD'den veri okunabilir. Komut çeşitleri ve ayar bitleri Tablo 4'te verilmiştir.

Tablo 4: LCD Komut Gönderimi

KOMUT	KOD										İŞLEM SÜRESİ
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Ekranı Sil	0	0	0	0	0	0	0	0	0	1	1,64 ms
Kursör basa don	0	0	0	0	0	0	0	0	1	*	1,64 ms
Giriş kipini seç	0	0	0	0	0	0	0	1	I/D	S	40μs
Ekran aç/kapa	0	0	0	0	0	0	1	D	C	B	40μs
Kursör kaydır	0	0	0	0	0	1	S/C	R/L	*	*	40μs
Fonksiyon seç	0	0	0	0	1	DL	N	F	*	*	40μs
Meşgul bayrağını oku	0	1	BF								0μs
Veri yaz	1	0									40μs
Veri oku	1	1									40μs

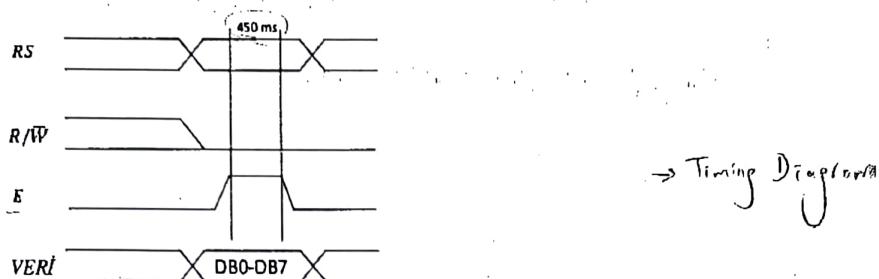
Komut seti; (* : Aldığı değer (1-0) önezsiz, DDRAM: Ekran Veri Belleği) Not: Tabloda belirtildiği gibi LCD modülle yazma işlemi minimum 40us sürmektedir. Dolayısı ile ard arda yapılan yazma işlemlerinde bir önceki verinin yazılabilmesi için en az belirtilen süre kadar beklenmelidir.

Tablo 5: Kontrol bitlerinin görevleri

Kod	AÇIKLAMA
I/D	0 = Her yazma işleminden sonra kursör pozisyonunu azalt 1 = Her yazma işleminden sonra kursör pozisyonunu artır
S	0 = Ekran kaydırma modu kapalı 1 = Ekran açık
D	0 = Ekran kapalı 1 = Kursör açık
C	0 = Kursör kapalı 1 = Kursör blink açık
B	0 = Kursör taşınması gereklidir. Manuel. 1 = Kursör kaydır
S/C	0 = Kursör taşınması gereklidir. Manuel. 1 = Sağa kaydır
R/L	0 = Sola kaydır 1 = Veri hattı 4 bit
DL	0 = Veri hattı 4 bit 1 = Veri hattı 8 bit
N	0 = 1 satır 1 = 2 satır
F	0 = 5x7 pixel 1 = 5x10 pixel
BF	0 = Komut kabul edebilir 1 = LCD Mesgul

2.2.1. Komut/Veri Gönderim Zaman Çizelgesi

Tablo 3'teki LCD kontrol pinlerinin zamanlama çizelgesi aşağıda verilmiştir.



Şekil 3: LCD'ye veri yazmak için kontrol pinlerinin zamanlanması

2.3. LCD'nin Başlangıç Ayarlarının Yapılması

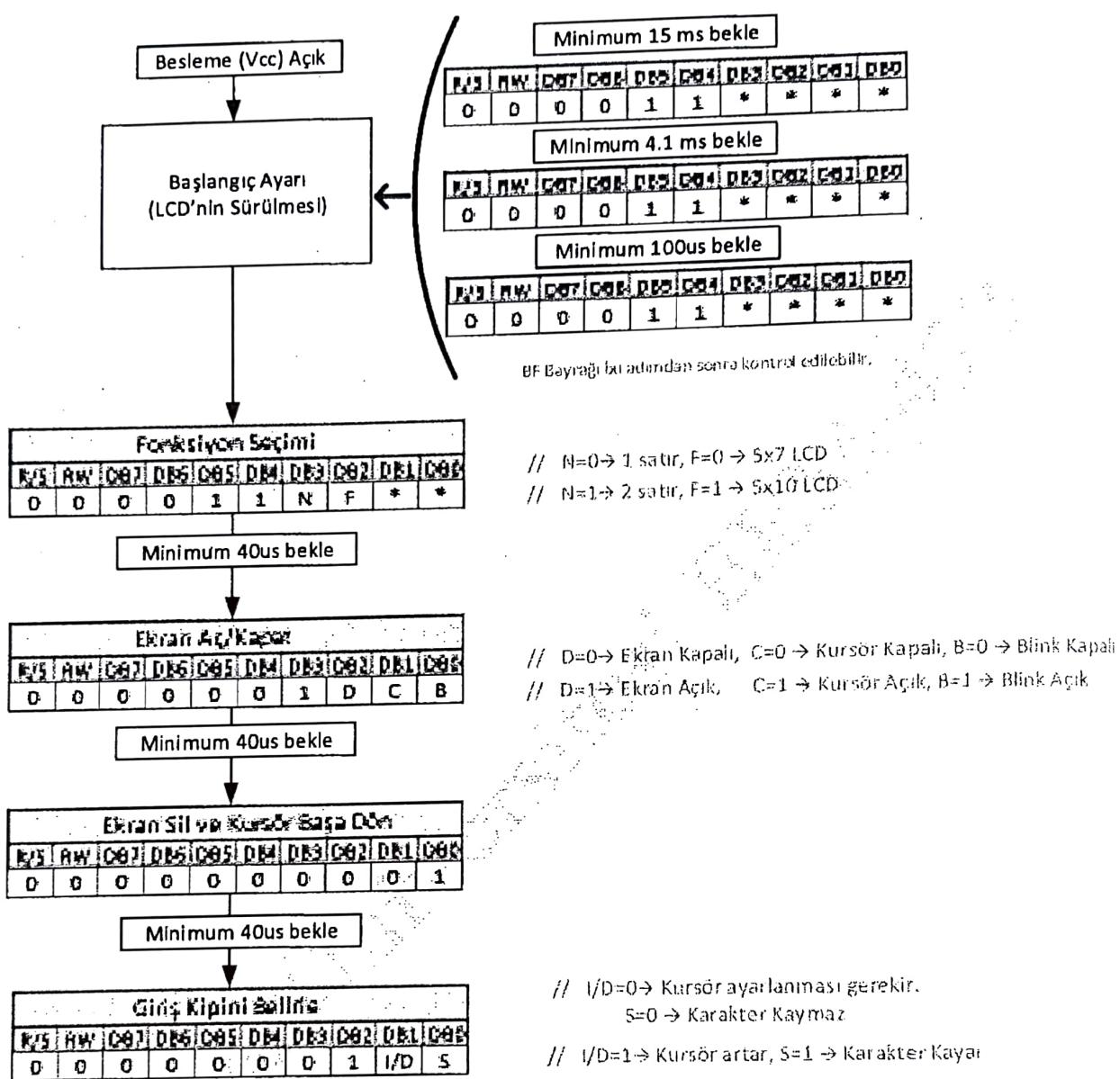
LCD'ye ilk enerji verilmesinden ardından, hazırlanan program ile arka arkaya 3 defa 30h komutu LCD'ye gönderilir. Burada bu komutun düşük dörtlüsü (son dört biti) ihmali edilir; DB7 ve DB6=0, DB5 ve DB4=1 olarak atanır.

Yukarıdaki başlangıç ayarının ardından LCD için fonksiyon belirleme işlemi Tablo 4 ve Tablo 5 ile gerçekleştirilebilir yani veri yolunun büyüklüğü (4 bit veya 8 bit), göstergedeki satır sayısı (1-2-3-4) ve font büyüklüğü (5x7 veya 5x10 gibi) belirlenir. Ardından sırası önemlilik üzere giriş kipi, ekranın, kursörün, blink'in (imlecin) açık/kapalı ayarları yapılır. Giriş kipi; her karakter okuma veya yazma işlemini takiben imlecin veya göstergenin yerini belirler.

LCD ilk başlangıç ayarı aşağıdaki şekilde verilmiştir.

RS → 1 : Veri
 0 : Komut
 R/W → 1 : Oku
 0 : Yaz
 E → 1 : Etme
 0 : Pasa

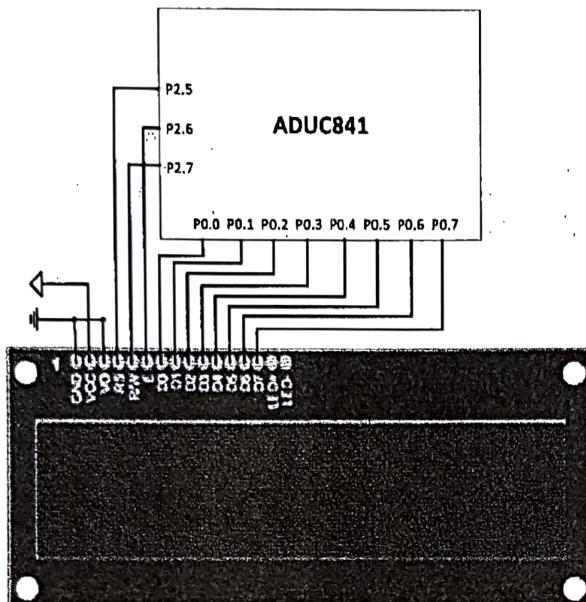
SAU / Muh.Fak. / EEM / LCD Ekranın Sürülmesi



Şekil 4: LCD Başlangıç Ayarı

2.4. Deney Setindeki Aduc841 İle Lcd Bağlantı Şeması Ve Pinleri

ADUC841 deney setindeki mikrodenetleyici ile LCD arasında aşağıdaki şekilde bağlantılar gerçekleştirilmiştir.



3. FLASH VERİ HAFİZASI

ADUC841 mikrokontrolöründe bulunan 4 kByte Flash/EE veri hafızası her biri 4 Byte lik alana sahip 1024 sayfadan oluşur ($4 \times 1024 = 4096$ Byte). Flash veri hafızası için 1 tane kontrol (ECON), 2 tane adres (EADR_H/EADR_L) 4 tane veri tutmak (EDATA1-4) amacıyla kullanılan 7 adet SFR vardır.

Sayfa adreslenmesi (seçilmesi) EADR_H/EADR_L saklayıcıları kullanılarak gerçekleştirilir. 1024 adet sayfa için $0 \leq \text{EADR}_H/L < \text{H}'0400$ olmalıdır. EADR_H/L saklayıcıları Byte adresleme içinde kullanılır. Bu durumda $0 \leq \text{EADR}_H/L \leq \text{H}'0FFF$ olur.

EDATA1-4 saklayıcıları okuma işlemi esnasında seçili olan sayfaya ait verileri tutar. Seçili sayfaya yazma işlemi de bu saklayıcılar üzerinden yapılır.

3FFH	BYTE 1 (0FFCH)	BYTE 2 (0FFDH)	BYTE 3 (0FFEH)	BYTE 4 (0FFFH)
3FEH	BYTE 1 (0FF8H)	BYTE 2 (0FF9H)	BYTE 3 (0FFAH)	BYTE 4 (0FFBH)
03H				
02H				
01H				
00H				

Veri hafızasında y: BYTE ADDRESSES ARE GIVEN IN BRACKETS

FR ile belirlenir.

Figure 41. Flash/EE Data Memory Control and Configuration

SAU / Muh.Fak./ EEM/ LCD Ekranın Sürülmesi

ECON saklayıcısının aldığı değerlere göre yapılan işlemler şu şekildedir;

ECON = 01'H: READ = EADRHL kullanılarak seçilen sayfadaki veriler EDATA1-4 saklayıcılarına aktarılır.
ECON = 02'H: WRITE = EDATA1-4 saklayıcılarındaki veriler seçilen sayfaya yazılır. (Not: Yazma işleminden önce ilgili sayfanın silinmesi gereklidir.)

ECON = 04'H: VERIFY = Adreslenen sayfadaki değerlerin EDATA1-4 saklayıcılarındaki değerlerle aynı olup olmadığını kontrol eder. Eğer aynı iseler ECON SFR sinin değeri 0 olur, herhangi biri farklı ise ECON SFR si 0 dan dan farklı bir değer alır.

ECON = 05'H: ERASE PAGE = Adreslenen sayfayı siler, (temizler).

ECON = 06'H: ERASE ALL = 4 kByte lik veri hafızasının tamamını siler.

ECON = 81'H: READ BYTE = EADRHL ile adreslenen Byte EDATA1 saklayıcısına aktarılır.

ECON = 82'H: WRITE BYTE = EDATA1 saklayıcısında tutulan veri EADRHL ile adreslenen Byte' a yazılır.

ECON = 0F'H: EXULOAD = ECON SFR sinin Flash/EE veri hafızasının kontrolünde kullanılmasını sağlar.

ECON = F0'H: ULOAD = ECON SFR sinin Flash/EE program hafızasının kontrolünde kullanılmasını sağlar.

Flash/EE veri hafızasına yazma ve hafızadan okuma işlemleri şüreleri aşağıda tabloda verilmiştir.

READPAGE (4 byte)	22 makine çevrimi
WRITEPAGE (4 byte)	380 μ s
VERIFYPAGE (4 byte)	22 makine çevrimi
ERASEPAGE (4 bytes)	2 ms
ERASEALL (4 kByte)	2 ms
READBYTE (1 byte)	9 makine çevrimi
WRITEBYTE (1 byte)	200 μ s

Örnek 1: Veri hafızasının H'203 sayfasına sayfanın 1. Byte ' tundan başlayarak sırasıyla H'65, H'2B, H'3E, H'7F sabit değerlerini yazmak için yapılan操作ları sırasıyla şu şekildedir;

- 1- Sayfa seçimi EADRHL kullanılarak gerçekleştirilecektir.
- 2- ECON saklayıcıları kullanılarak (ECON =05'H) seçilen sayfa temizlenir (silenir).
- 3- Silme işleminden sonra ilgili saklayıcılara yazma değerleri aktarılır.
- 4- ECON saklayıcısına yazma komutu (ECON =02'H) yüklenir.
- 5- Yazma işleminin doğruluğu kontrol edilir. (zururi değil, isteğe bağlı)

MOV EADRHL, #02H ; sayfa seçimi
MOV EADRL, #03H

ERROR:

MOV ECON,#05H ; seçili olan sayfa silindi

MOV EDATA1, #H'65 ; birinci byte'a istenen değer yazıldı
MOV EDATA2, #H'2B ; ikinci byte
MOV EDATA3, #H'3E ; Üçüncü byte
MOV EDATA4, #H'7F ; dördüncü byte

MOV ECON,#02H ; EDATA1-EDATA4 saklayıcılarındaki veriler seçilen sayfaya aktarıldı.
MOV ECON,#04H ; yazma işleminde hata var mı?
MOV A,ECON
JNZ ERROR ; hata varsa yazma işlemini tekrarla

Veri hafızadan yapılacak bir okuma işlemi için, EADRHL' saklayıcıları kullanılarak sayfa (veya byte) seçimi yapılır. Sayfa seçildikten sonra ECON SFR sine okuma komutu yazıldığında seçili sayfadaki (Byte'taki) değerler otomatik olarak EDATA1-4 (EDATA1) saklayıcılarına aktarılır. İstenen değerler bu saklayıcılardan, EDATA1-4, okunur.

Örnek 2: Veri hafızanın H'153 sayfasındaki 1. ve 2. Byte 'i okuyup bu iki değeri çarpan program parçası şu şekildedir;

SAU / Muh.Fak./ EEM/ LCD Ekranın Sürülmesi

- 1- EADRHL saklayıcıları kullanılarak sayfa seçimi yapılır.
- 2- Sayfa seçildikten sonra ECON saklayıcı ile okuma işlemi yapılmayı belirtir. ECON' okuma komutu yazıldığında seçili sayfadaki değerler otomatik olarak EDATA1-4 saklayıcılarına aktarılır.
- 3- İstenen değerler bu saklayıcılarından, EDATA1-2, okunur.
- 4- Çarpma işlemi "mul AB" şeklinde olduğundan okunan değerler bu saklayıcılara aktarılır.

MOV EADRHL, #01
MOV EADRL, #53H

; sayfa seçimi

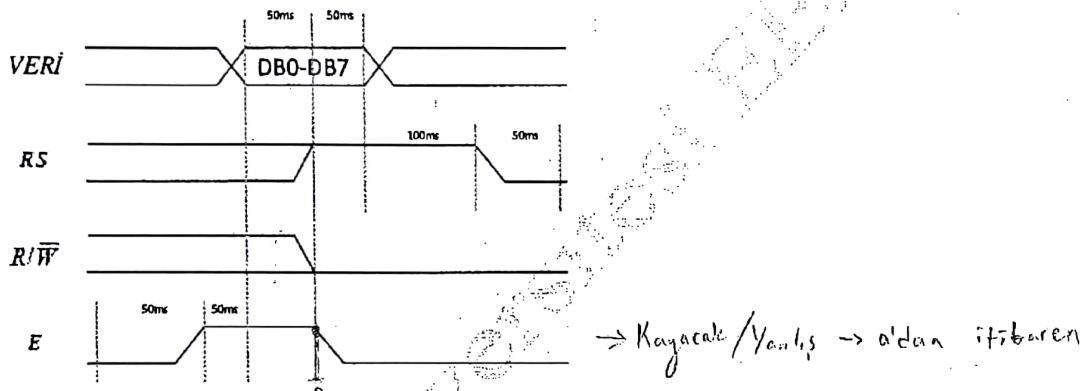
MOV ECON, #01 ; seçili olan sayfadan okuma işlemi seçildi

MOV A, EDATA1 ; H'53 sayfasındaki 1. Byte "A" ya

MOV B, EDATA2 ; 2. Byte "B" ye aktarıldı.

MUL AB ; okunan iki değer birbiri ile çarpıldı..

ÖN CALISMA:



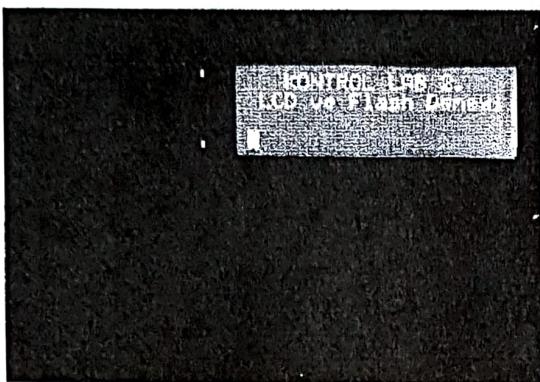
Sekil 5: Veri yazma zamanlama diyagramı

Yukarıdaki şekilde, bir cihaza ait kontrol pinlerinin veri yazma zamanlama diyagramı verilmiştir.

Şekil 5'i kullanarak sadece kontrol uçları için (3 pin için) asm kodunu yazınız. Yazılan asm kodunun kendini tekrarlanması istenmektedir. İstenilen aduc841 port ve pinleri kullanılabilir. Bekleme süreleri timer ya da alt program ile yapabilirsiniz.

UYGULAMA 1:

Deneyde aşağıdaki şekilde gözüktüğü gibi LCD ekranın ilk satırına 4. kolandan itibaren "KONTROL LAB. 2" ikinci satırına ise 2. kolandan itibaren "LCD ve Flash Deneyi" yazdırılacaktır.

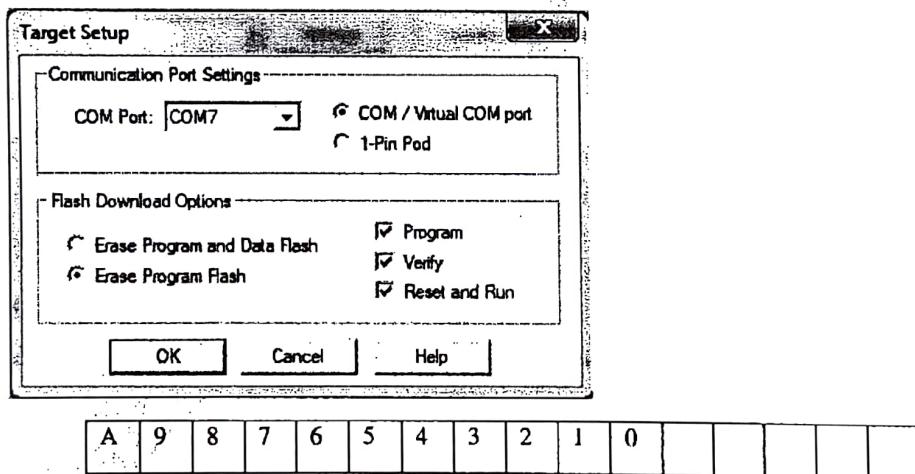


UYGULAMA 2:

Veri hafızasının 3.sayfasına 0A'H sabiti yazılacaktır.

UYGULAMA 3:

Bir önceki deneyde veri hafızasının 3.sayfasına yazılan sayı (0Ah) yazıldığı adresden okunup LCD ekranın 1.satır 16.sütunundan başlanarak, her 500ms de değeri bir azaltılarak (0 olana kadar) bir önceki karakter alanına yazılacaktır. Derlenen kodun hex dosyası Aduc841'e gönderilirken aşağıda verilen şekilde mikroişlemciye gönderilmesi gerekmektedir. (Data Flash'in programlanması gerekmektedir.) Aksi takdirde bir önceki deneyde flash'a yazılan veri silinir. uVision ayarı aşağıdaki şekilde verilmiştir.



(Not: Uygulama 2 de veri hafızaya yazma işlemi yapıldıktan sonra işlemcinin beslemesi kapatılıp tekrar açılır. Ardından uygulama 3 gerçekleştirilir. Uygulama 3'te veri hafızaya yazma işlemi yapmadan uygulama 2 de veri hafızaya yazılan veri okunup yukarıda belirtildiği gibi LCD ekrana aktarılacaktır.)

Uygulamalara Ait Kodlar

Uygulama 1:

```
#include<aduc841.h>
```

```
LRS EQU P2.5
LEE EQU P2.6
LCD EQU P0
RW EQU P2.7
```

```
ORG 00H
sjmp BASLA
```

BASLA:

```
clr RW ; LCD'den okuma modunu iptal et.
mov r0, #07FH ; Temizleme
temiz: mov @r0, #00H ; Temizleme
        djnz r0, temiz ; Temizleme

lcall lcd_ayar ; LCD'nin Başlangıç Ayarları Yapılıyor.
        mov dptr, #Data1
```

; _____ 1. Satır _____

```
clr LRS ;komut girişi
mov a, #83H ;birinci satır
lcall yaz ;baslangic adresi
setb LRS ;veri girişi
mov r0, #00H
str1: mov a, r0
        movc a, @a+dptr
        cjne a, #'0', go1
        sjmp str2
go1: lcall yaz
        inc r0
        sjmp str1
```

; _____ 2. Satır _____

```
str2: clr LRS ;komut girişi
        mov a,#0c1H ;ikinci satır
        lcall yaz ;baslangic adresi
        setb LRS ;veri girişi
        mov dptr, #Data2
        mov r0, #00H
go:   mov a, r0
        movc a, @a+dptr
        cjne a, #'0', go2
        sjmp dur
go2: lcall yaz
        inc r0
        sjmp go
```

DUR: SJMP DUR

; _____ LCD Ayarı _____

lcd_ayar:

```
clr    LEE
clr    LRS ;komut girişi
```

;

; Minimum 15 ms bekleme

```
lcall  gecik
lcall  gecik
lcall  gecik
```

;

; LCD'nin Sürülmesi için Gerekli Kod Parçası

```
; RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
; 0 0 0 0 1 1 * * * * => 30H
```

```
mov   a,    #30H
lcall yaz
lcall gecik
mov   a,    #30H
lcall yaz
lcall gecik
mov   a,    #30H
lcall yaz
```

;

; LCD Ayarları

```
; RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
; 0 0 0 0 1 1 N F * *
; 0 0 0 0 1 1 I I I I => 3CH => N=1 icin 2 satır, F=1 icin 5x10 LCD
mov   a,    #3cH ;2 satır, 5x10 pixel
lcall yaz
```

```
; RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
; 0 0 0 0 0 0 1 D C B
; 0 0 0 0 0 0 1 1 1 1 => OFH => D=1 Ekran Açık, C=1 Kursör Açık, B=1 Blink Açık.
mov   a,    #0fH ;Ekran, Kursör ve Blink açık
lcall yaz
mov   a,    #01H ;Ekranı sil, kursör başa dön.
lcall yaz
```

```
; RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
; 0 0 0 0 0 0 0 1 VD S
; 0 0 0 0 0 0 0 1 1 0 => O6H => VD=1 Kursör Arttır, S=0 Ekran Kaydırma Kapalı
mov   a,    #06H ;Giriş Modu => Kursör pozisyonunu artır
lcall yaz
ret
```

_____LCD Ayarı Bitti_____

```
yaz: setb LEE
lcall gecik
mov LCD,a ; LCD = PortO (P0)
clr LEE
ret
```

```
gecik: mov r3, #4fh
w2:   mov r4, #0ffh
```

```
w1: djnz r4, w1  
djnz r3, w2  
ret
```

```
;
```



```
Data1: DB 'MIKRO LAB.0'  
Data2: DB 'SAU EEM BOLUMU0'  
end
```

Uygulama 2:

```
#include<aduc841.h>  
WRITE EQU 02h ; 'write page'  
VERIFY EQU 04h ; 'verify page'  
ERASE EQU 05h ; 'erase page'  
;  
ORG 0000h  
JMP MAIN  
MAIN:  
    MOV EADRH,#0 ;3.sayfa seçildi  
    MOV EADRL,#3  
    MOV ECON,#ERASE ;yazma yapmadan önce ilgili byte silinmeli  
    MOV EDATA1,#0AH ;  
    MOV ECON,#WRITE ;3. sayfadaki 1. Byte alanına H'0F yaz  
    MOV ECON,#VERIFY ;yazma işlemini doğrula..  
    MOV A,ECON  
    JNZ MAIN  
END
```

Uygulama 3:

```
#include<aduc841.h>  
LRS EQU P2.5  
LEE EQU P2.6  
LCD EQU P0  
RW EQU P2.7  
  
ORG 00H  
sjmp BASLA  
ORG 0BH  
LJMP TIMER0  
BASLA:  
    clr RW  
    mov r0, #07fH  
temiz: mov @r0, #00H  
    djnz r0, temiz
```

SAU / Muh.Fak./ EEM/ LCD Ekranın Sürülmesi

```
lcall lcd_ayar
mov dptr, #Sayi

;-----[MOV EADRH, #0h ;veri hafızanın
;-----[MOV EADRL, #3h ;3.sayfası seçildi
;-----[MOV ECON, #01 ;3.sayfadaki ilk Byte'in içeriği okundu
;-----[MOV R1, EDATA1 ;ve R1 saklayıcısına aktarıldı..]

;-----[clr LRS ;LCD için komut girişi
;-----[mov a, #80H ;başlangıç adresi
;-----[lcall yaz
;-----[setb LRS ;LCD için veri girişi

;-----[MOV TMOD, #081H
;-----[MOV TH0, #00 ;Timer0 ayarlandı
;-----[MOV TL0, #00
;-----[MOV R0, #84d ;500ms=5.9ms* 84
;-----[SETB EA
;-----[SETB ET0
;-----[SETB TR0 ;Timer0 start

;-----[MOV A, R1 ;Veri hafızadan okunan değer R1 de tutuluyordu..
;-----[MOVC A, @A+DPTR ;ilk değeri
;-----[LCALL YAZ ;başlangıç adresine yaz..

DUR: CJNE R0, #0, DUR ;500ms doldumu?
      MOV R0, #84d ;doldu..
      DEC R1 ;değeri 1 azalt
      MOV A, R1
      MOVC A, @A+DPTR ;LCD ya yaz..
      LCALL YAZ
      CJNE R1, #0, DUR

STOP: SJMP STOP

;-----[TIMER0: dec R0
;-----[reti

;-----[LCD Ayarı
lcd_ayar:
;-----[clr LEE
;-----[clr LRS ;komut girişi
;-----[; Minimum 15 ms bekleme
;-----[lcall gecik
;-----[lcall gecik
;-----[lcall gecik
;-----[;
;-----[;
;-----[; LCD'nin Sürülmesi için Gerekli Kod Parçasığı
;-----[RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
;-----[0 0 0 0 1 1 * * * * => 30H
```

SAU / Muh.Fak. / EEM / LCD Ekranın Sürülmesi

```
mov a, #30H
lcall yaz
lcall gecik
mov a, #30H
lcall yaz
lcall gecik
mov a, #30H
lcall yaz
-----
;-----  
; LCD Ayarları

;RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
;0 0 0 0 1 1 N F * *
;0 0 0 0 1 1 1 1 * * => 3CH => N=1 için 2 satır, F=1 için 5x10 LCD
mov a, #3cH ;2 satır, 5x10 pixel
lcall yaz

;RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
;0 0 0 0 0 0 1 D C B
;0 0 0 0 0 0 1 1 1 1 => OFH => D=1 Ekran Açık, C=1 Kursör Açık, B=1 Blink Açık.
mov a, #0fH ;Ekran, Kursör ve Blink açık
lcall yaz
mov a, #01H ;Ekranı sil, kursör başa dön.
lcall yaz
ret
```

LCD Ayarı Bitti

```
yaz: setb LEE
lcall gecik
mov LCD,a
clr LEE
ret
;
gecik: mov r3, #4fh
w2: mov r4, #0ffh
w1: djnz r4, w1
djnz r3, w2
ret
;
Sayi: DB '0123456789ABCDEF'
```

end

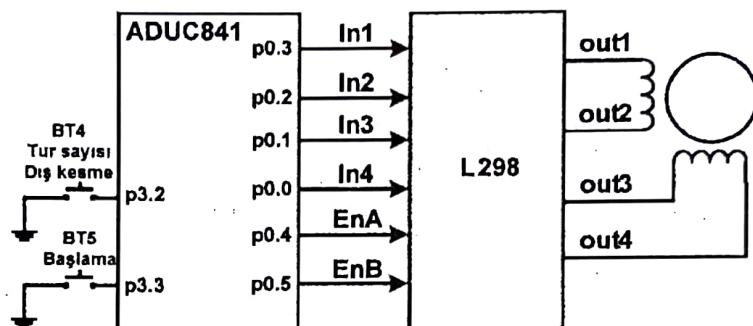
Deney 2: Step Motor Sürülmesi

Amaç: Aduc841 mikrokontrolör ile;

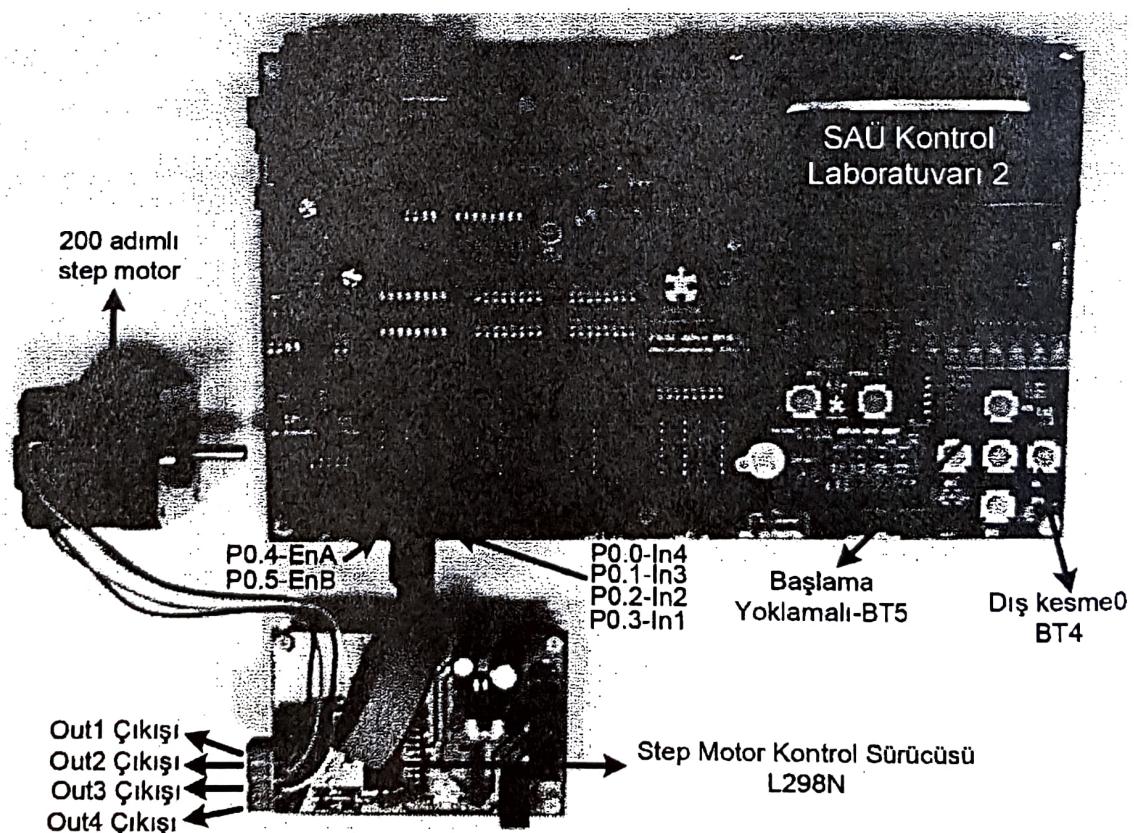
- 1- Dış Kesme0 ve Timer0 (yoklamalı) kullanımı
- 2- Seçilen port pinlerinden zamanlama diyagramına göre darbelerin üretilmesi
- 3- H-Köprü sürücü entegresi ile step (adım) motorun sürülmESİ

Deney düzeneğine ait devre şeması Şekil 1'de verilmiştir. Step motoru ve sürücüsü (L298) için EK-1'e bakınız.

*Interrupt
Kod hafıza
RAM
Mimari
Soruya*



Şekil 1: Step motor deneyi devre şeması



Şekil 2: Step motor deneyi devre şeması

Ön Çalışma: Bu kısım ile ilgili yazılım kodları deney esnasında yazılacaktır (yazılım kodları verilmeyecektir). Kullanılacaklar:

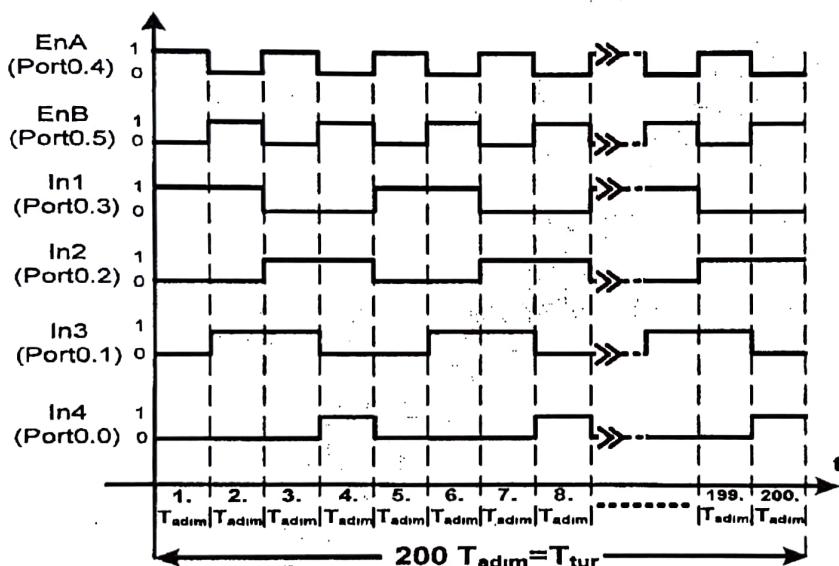
- 1-) Timer0 yoklamalı 1s
- 2-) Led3-P0.2
- 3-) Dış kesme0 (INT0)-BT4
- 4-) Yoklamalı-BT5

İstenen:

1. LED3 başlangıçte sönük
2. BT4 ile dış kesme0 (INT0) kullanılarak LED3 yanıp sönme sayısını girilecektir.
3. Her BT5'e (yoklamalı) basıldığında LED3 1s aralıklarla Adım-2'de girilen adet kadar yanıp sönecektir.

Deney:

- 1- Önce, BT4 ile dış kesme0 (INT0) kullanılarak tur sayısını girilmek zorundadır.
- 2- Her BT5'e (yoklamalı) basıldığında aşağıdaki zamanlama diyagramına uygun olarak adım-1'de girilen adet kadar step motor tur atıp duracaktır. Motorun tekrar harekete geçmesi için yeni tur bilgisi girilmek zorundadır.



- 3- Adım-2'de verilen tur sayısını step motorun **60 devir/dakika** hızla dönmesi istenmektedir. Bunun için; yukarıdaki şekilde görülen $T_{adım}$ süresi hesabı aşağıda verilmiştir.

Step motorun 1 tur dönmesi için geçen süre “ T_{tur} ” olarak tanımlanır. Step motorun **60 devir/dakika** dönmesi için, $T_{tur}=1 \text{ sn}$ olarak hesaplanır. Şekil 3 den $T_{adım}$ süresi, $T_{adım} = 1\text{sn}/200 = 5 \text{ msn}$ olarak hesaplanır. 5 msn süresi **Timer0** aracılığı ile **yoklamalı** olarak üretilecektir.

- 4- Zamanlama diyagramına bakıldığından üretilen işaretler için bir periyodun dört $T_{adım}$ 'dan oluşanluğu görülebilir. Deney esnasında, step motorun bir tur yapması için 50 kere periyot tekrarı yapılması gerekmektedir.

Yazılım adımları aşağıdaki gibi verilmiştir:

- 1- INT0 harici kesme0 ve kesme ile ilgili diğer ayarlar yapılacak.
- 2- T_{adim} için Timer0 16-bit mod zamanlayıcı ayarı yapılacak ve zaman aşımı yoklamalı olarak algılanacaktır.
- 3- BT4 ile tur bilgisi alınacaktır.
- 4- BT5'e basılıp bırakıldığından,
 - i- Yukarıda verilmiş olan zamanlama diyagramına uygun olarak 4 adet T_{adim} üretilicektir.
 - ii- Step motorun bir tur atabilmesi için i. adımı 50 defa tekrar edecektir.
 - iii- Adım-3'te girilmiş olan tur sayısı kadar i. ve ii. adımları tekrar edecektir.
- 5- 3. adım tamamlandığında motor durdurulacak. Yani Port0'a ait tüm pinler Lojik "0" yapılacaktır.
- 6- Program 3. adıma döndürülecek.

UYARI: Step motorun zarar görmesini engellemek amacıyla kullanılmadığı zaman diliminde tüm girişleri Lojik "0" olarak ayarlanmalıdır. Aksi takdirde motorun bazı sargıları enerjili kalabileceğinden ısınmasına ve zarar görmesine neden olabilir.

Örnek Program kodları:

```
#include <aduc841.h>
;program etiketleri tanımlanıyor
ENA      EQU  P0.4 ; Sürücü A köprüsü enerjilendirme girişi.
ENB      EQU  P0.5 ; Sürücü B köprüsü enerjilendirme girişi.
IN1      EQU  P0.3 ; Sürücü IN1girişi.
IN2      EQU  P0.2 ; Sürücü IN2girişi.
IN3      EQU  P0.1 ; Sürücü IN3girişi.
IN4      EQU  P0.0 ; Sürücü IN4girişi.
BT5      EQU  P3.3 ; Program başlatma butonu.
TUR      EQU  00H ; Kullanıcıdan alınan tur bilgisi.
ADIM     EQU  01H ; Adım kontrol etiketi.

ORG 0000h
JMP Basla
ORG 0003H          ; Dış kesme INT0-P3.2
JMP EXT_0
Basla:
    MOV R0,#7FH
ZERO:
    MOV @R0,#0
    DJNZ R0,ZERO   ; Son 3 satır ile Hafıza Temizleme işlemi
    MOV p0,#00h     ; Sürücü giriş enerjileri kesildi...
    MOV TMOD,#81H ; Timer0 16-Bit Zamanlayıcı
    MOV dptr,#10225D ; 5ms için başlangıç değeri
    MOV tl0,dpl
    MOV th0,dph
    SETB EA         ; Kesmeler aktif
```

SETB EX0 ; Dış kesme0 aktif

X1: JB BT5, X1	; BT5'e basıldı mı?
X2: JNB BT5, X2	; BT5 bırakıldı mı?
MOV A,tur	; En az 1 tur girilmeli...
JZ X1	; Tur sayısı 0 ise tekrar kesme ve BT 5'e basılması bekleniyor.
MOV ADIM,#50	; $50*4=200$ adımlı step motor
SETB TR0	; Timer0 başla komutu

DEVAM: ; Step motor sargıları sırasıyla enerjilendirilerek dönmesi sağlanıyor.

SETB ENA	; Sürücü A köprüsü enerjilendirildi.
CLR ENB	; Sürücü B köprüsü enerjisi kesildi.
SETB IN1	; Sürücü IN1 girişi enerjilendirildi.
CLR IN2	; Sürücü IN2 girişi enerjisi kesildi.
ACALL BEKLE	; T_{adim} süresi kadar bekle...
SETB ENB	; Sürücü B köprüsü enerjilendirildi.
CLR ENA	; Sürücü A köprüsü enerjisi kesildi.
SETB IN3	; Sürücü IN3 girişi enerjilendirildi.
CLR IN4	; Sürücü IN4 girişi enerjisi kesildi.
ACALL BEKLE	; T_{adim} süresi kadar bekle...
SETB ENA	; Sürücü A köprüsü enerjilendirildi.
CLR ENB	; Sürücü B köprüsü enerjisi kesildi.
SETB IN2	; Sürücü IN2 girişi enerjilendirildi.
CLR IN1	; Sürücü IN1 girişi enerjisi kesildi.
ACALL BEKLE	; T_{adim} süresi kadar bekle...
SETB ENB	; Sürücü B köprüsü enerjilendirildi.
CLR ENA	; Sürücü A köprüsü enerjisi kesildi.
SETB IN4	; Sürücü IN4 girişi enerjilendirildi.
CLR IN3	; Sürücü IN3 girişi enerjisi kesildi.
ACALL BEKLE	; T_{adim} süresi kadar bekle...

DJNZ ADIM,DEVAM ; $50*4=200$ adım tamamlandı mı?

1 TUR DONDU...

MOV ADIM,#50d ; Milin 1 tur dönmesi için

DJNZ TUR,DEVAM ; Tur sayısı tamamlandı mı?

MOV p0,#00h ; Tur sayısı tamamlandı. Sürücü giriş enerjileri kesildi...

SJMP X1

BEKLE:

JNB TF0,BEKLE ; Taşma oldu mu?

CLR TF0

MOV tl0,dpl ; 5ms başlangıç değerlerini yeniden yükle.

MOV th0,dph

RET

EXT_0:

ACALL GECIKME

MOV TUR,#3 ; Kullanıcı tur bilgisini girer.

RETI

GECIKME:

; Bas-bırak butonun bozuculardan etkilenmemesi için bir gecikme konulur.

; İç içe döngü ile sağlanmıştır.

MOV R4,#10d

SAU / Muh.Fak./ EEM/ Step Motor Sürillmesi

```
MOV R5,#0FFh  
MOV R6,#0FFh  
x:  
DJNZ R5,x  
MOV R5,#0FFh  
DJNZ R6,x  
MOV R6,#0FFh  
DJNZ R4,x  
RET  
=====  
END
```

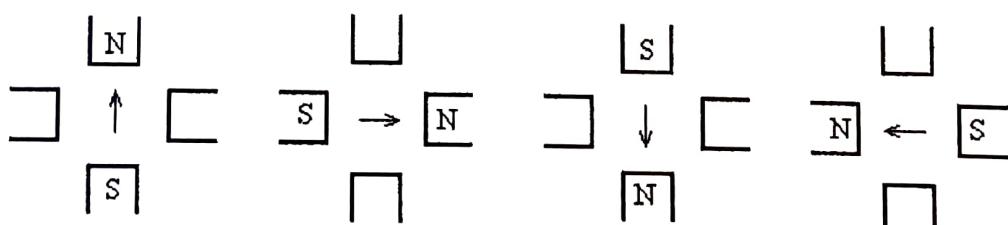
EK-1

Step (Adım) Motor

Step motorlar DC motorların aksine sargılarından akım akınca serbestçe dönmeye başlamazlar. Step motorları döndürebilmek için **stator sargılarına** (bobinlerine) belirli bir sıra dahilinde akım darbeleri vermek gerekir. Bu darbelerin sırası ve veriliş sıklığı step motorun dönme yönünü ve dönme hızını belirler. Her bir darbe geldiğinde step motor sabit bir açı miktarı kadar hareket eder. Bu açı adım açısı olarak bilinir ve motorun imalatı esnasında belirlenir.

Bant sürücülerı, imalat tezgâhları, yazıcılar, teyp sürücülerı, tıbbi cihazlar, makine tezgâhları, dikiş makineleri, taksimetreler, kart okuyucular vb. yerlerde kullanılırlar. Genel olarak adım motorlar; her türlü denetlenmiş hareket ve pozisyon gerekliliği olan yerlerde, dijital bilgileri mekanik harekete çeviren bir eleman (transduser) olarak görev yaparlar.

Step motorlar manyetik alanların karşılıklı etkileşimi (itmeye-çekme) prensibiyle çalışırlar. Sürücü durumındaki manyetik alan stratejik olarak yerleştirilmiş bobin gruplarının enerjilendirilip ardından enerjinin kesilmesi yoluyla döner. Böylece Şekil 2 de gösterildiği gibi dönen bir manyetik alan oluşturulur. Dönen bu manyetik alan step motorun sabit mıknatışlı rotorunda beraber çekerek döndürür ve hareket oluşur. Eğer bobinleri enerjilerken belirli bir sıraya uyulursa rotor istenen yönde döner, aksi takdirde (bobinlerin rastgele enerjilendirilmesi durumunda) vizildayan ve yerinde duran bir rotor mili elde ederiz. Aşağıdaki şekilde adım açısı 90° olan step motorun bir tur dönüsü verilmiştir.

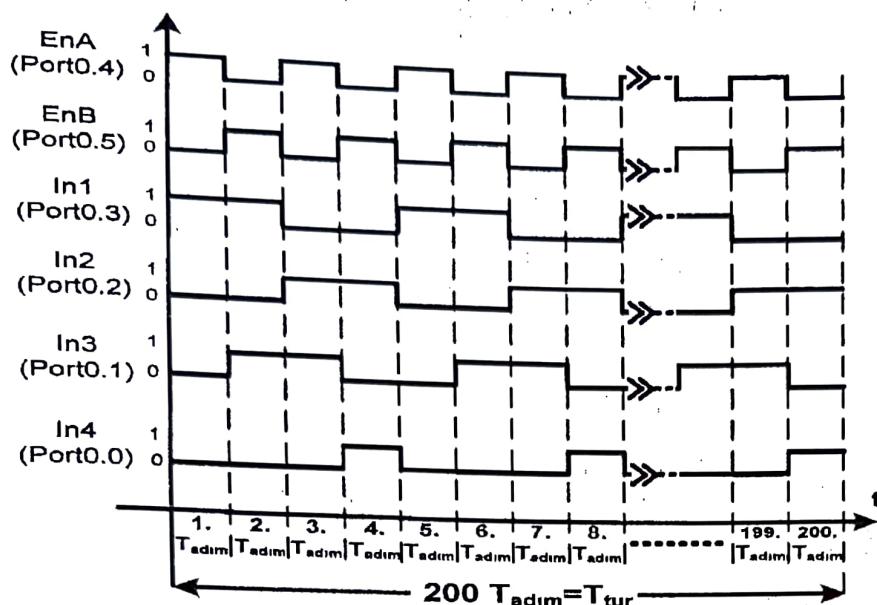


Şekil 2: Adım açısı 90° olan step motorun 1 tur dönüsü

Step motorlar bir turdaki adım sayısı ile anılırlar. Örnek olarak 48 adımlık bir step motorun **bir tam tur dönebilmesi** için **48 adım** atması gereklidir. Bu durumda adım açısı $360/48 = 15^\circ$ olur. Bu değer, step motorun hassasiyetinin bir göstergesidir. Bir turdaki adım sayısı yükseldikçe step motorun hassasiyeti ve dolayısı ile maliyeti artar. Deneylerde kullanılacak olan step motorlar 200 adımlıdır. Şekil 3 de deneylerde kullanılacak olan 200 adımlı step motorun, Şekil 1 de verilen bağlantıya uygun olarak saat yönünde 1 tur dönüsü için L298N sürücü entegresinin girişlerine uygulanması gereken kontrol sinyallerini gösteren zaman diyagramı verilmiştir. Zamanlama diyagramından görüleceği üzere 4 adımda

bir kendini tekrarlayan bir enerjilendirme sırasıyla 200 adım tamamlandığında motorun 1 tur yapması sağlanır.

$$360^\circ / 200 = 1,8^\circ$$



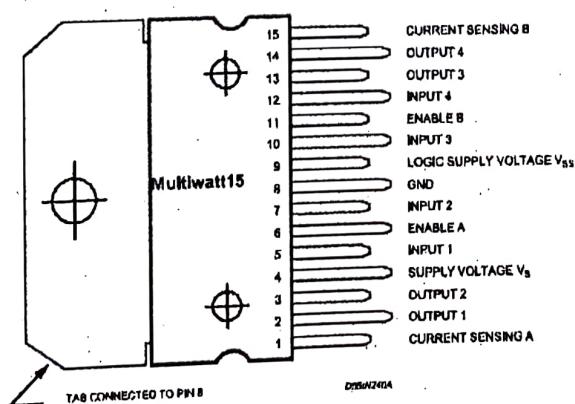
Şekil 3: 200Adımlı step motorun zamanlama diyagramı

EnA, EnB, In1, In2, In3 ve In4 Şekil 1'deki devre şemasında gösterildiği gibi step motor sürücü entegresinin (L298N) giriş pinleridir.

Şekil 3 deki T_{tur} süresi motorun 1 tur olması esnasında geçen zamandır. Motorun tur sayısı (d/dk) T_{tur} süresine, T_{tur} süresi de Şekil 3 te görüldüğü gibi EnA, EnB, In1, In2, In3 ve In4 girişlerine uygulanan T_{adim} süresine bağlı olarak değişir.

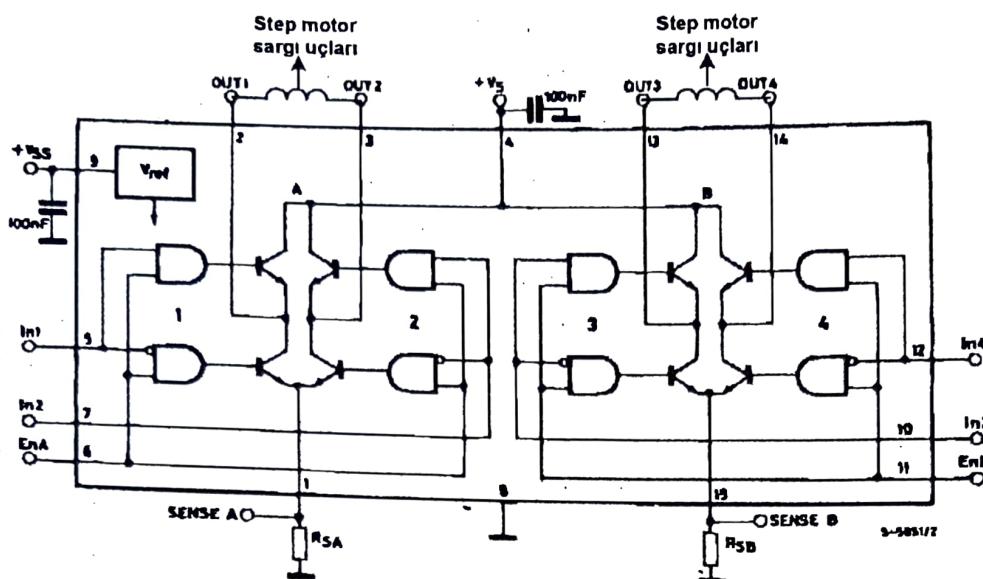
L298N

Şekil 1 de görüldüğü gibi step motor ile mikrokontrolör arasında sürücü bir devrenin olması gereklidir. Bu sürücü devre yapılan işe göre çok çeşitli olabilir. Deneyde step motor sürülmesi için L298N entegresi kullanılmıştır. L298N entegresinin pin bağlantıları ve blok diyagramı aşağıda verilmiştir.



Otsu's Algorithm

SAU / Muh.Fak./ EEM/ Step Motor Sürülmlesi



Şekil 4: L298N entegresine ait pin bağlantıları ve blok divagramı

Blok diyagram etiketleri	İşlevleri
Sense A; Sense B	A ve B noktalarından akan yük akımını ölçme pinleri.
Out 1;Out 2	A köprüsü çıkışlarıdır. Pin 1'den izlenen ve bu iki pin arasına bağlı olan yüke akım akar.
V _s	Kaynak gerilimidir. Endüktif olmayan 100 nF kapasitör bu pin ve ground arasına bağlanmalıdır.
Input 1; Input 2	A köprüsü girişleridir.
Enable A; Enable B	A ve B köprülerini aktif etme girişleridir.
GND	Toprak noktasıdır.
V _{ss}	Lojik blokları kaynak gerilimidir. 100 nF'luk bir kapasitör bu pin ile toprak arasında bağlanmalıdır.
Input 3; Input 4	B köprüsü girişleridir.
Out 3;Out 4	B köprüsü çıkışlarıdır. Pin 15'den izlenen ve bu iki pin arasına bağlı olan yüke akım akar.
N.C.	Bağlı değil (Not Connected).

```

mov    r@#0H      } Just ram
mov    @r@, #Ver
mov    80H, #Ver  } SFR

```

$$\frac{5000000}{90,42} = 55309 \quad \text{SMS}$$

63535
55109
10226 dan
| kohlepulat
Herk C
G.D.R.

$$12 \text{ bit} \Rightarrow 2^{12} = 4096$$

SAU / Muh.Fak./ EEM/ Analog-Sayısal Çeviriciler $V_{ref} = 5V$ ise.

$$\frac{V_{ref}}{4096} \approx 1,2mV$$

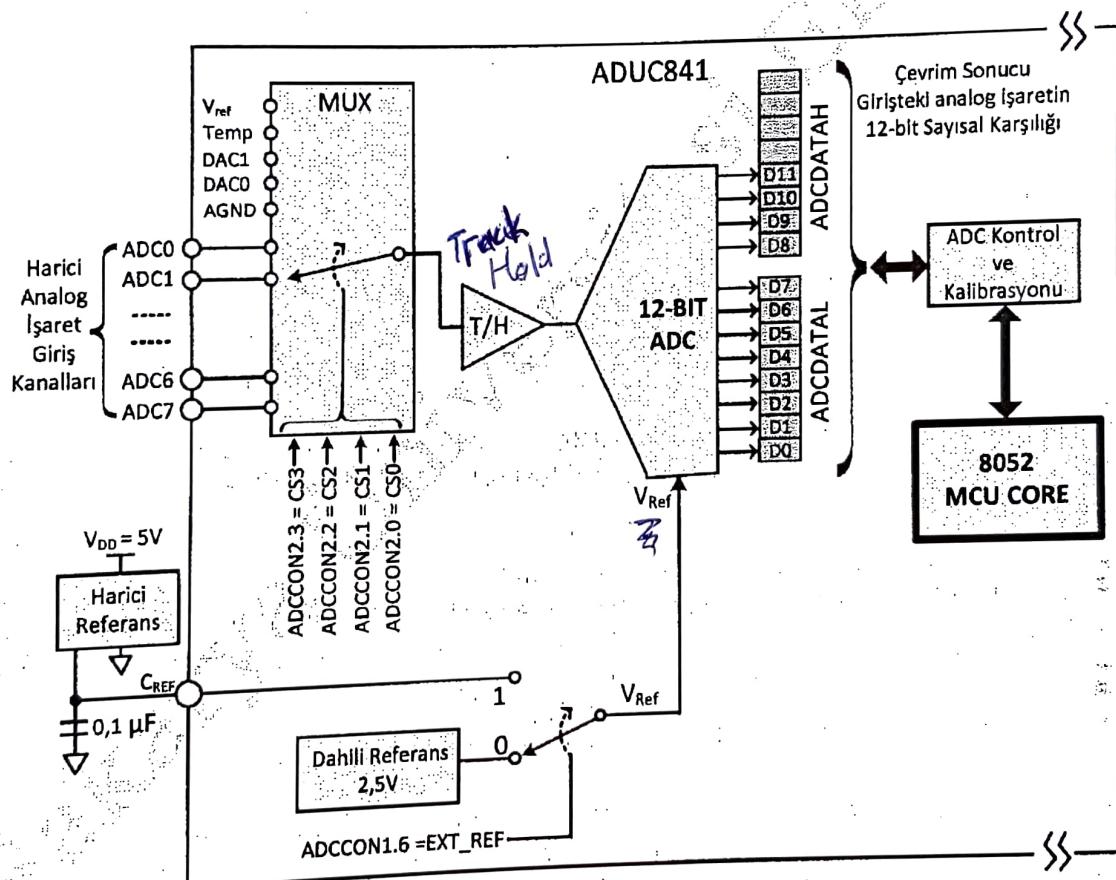
Deney 3: Analog-Sayısal Çeviriciler (ADC)

Amaç: ADUC841 mikrokontrolörü mimarisinde bulunan ADC (Analog to Digital Converter=Analog Sayısal Çevirici) nin farklı çevrim modlarında kullanabilme becerisinin kazandırılması.

ADC

Fiziksnel sistemlerin giriş-çıkış işaretleri (sıcaklık, basınç, hız, konum, akım, gerilim, vb.) çoğunlukla sürekli zaman analog işaretlerdir. Sayısal işlemciler (mikrokontrolör, DSP, PLC, PC vb.) ise sadece sayısal (ikilik tabanda kodlanmış - 10110110...) işaretler üzerinde işlem yapabilirler. Bu nedenle bir analog işaretin sayısal işlemci tarafından ölçme, kontrol, vb. amaçla değerlendirilebilmesi için ilk olarak sayısal işarete çevrilmesi gerekmektedir. Analog-Sayısal çeviriciler (ADC) genliği sınırlanmış bir analog işaretin ikilik tabanda kodlanmış sayısal işaretin çeviren sistemlerdir.

Aduc841 mikrokontrolörü mimarisinde, 12-bit, 8 adet harici analog giriş kanalına sahip, maksimum çevrim hızı 420 kSPS (420.000 çevrim/saniye) olan tek-kutuplu (unipolar) bir ADC birimi mevcuttur. Aduc841 mimarisinde bulunan ADC'nin basitleştirilmiş blok diyagramı aşağıda verilmiş ve ilgili blokların işlevi kısaca açıklanmıştır.



Şekil 1: Aduc841 mimarisinde bulunan ADC için basitleştirilmiş blok diyagramı

MUX (Çoğullayıcı=Multiplexer): Şekil 1'de verilen blok diyagramında gösterildiği gibi mimaride sadece bir adet ADC birimi bulunmaktadır. Fakat harici (Adc0...Adc7) ve dahili (DAC0, DAC1, vd.) olmak üzere toplam 13 adet analog giriş kanalı bulunmaktadır. ADC herhangi bir anda bu giriş kanallarından sadece bir tanesini sayısal dönüştürebilir. Dolayısıyla ADC çevrimini başlatmadan önce analog giriş kanalının seçilmesi gerekmektedir. MUX yapısı aşağıda açıklanan seçme bitleri (ADCCON2 saklayıcısındaki CS0..CS3 bitleri) aracılığı ile analog giriş kanalının seçimini sağlar.

Unipolar $\rightarrow O \leftrightarrow V_{ref}$
 Bipolar $\rightarrow -V_{ref} \leftrightarrow O \leftrightarrow V_{ref}$

T/H (TRACK and HOLD= İzleyici ve tutucu): MUX aracılığı ile seçilen analog işareti ADC çevrime başlayana kadar izler. ADC çevrime başladığında T/H girişindeki analog işaretin o anki değerini tutar ve çevrim bitene kadar bu değerin sabit kalmasını sağlar. Aksi halde, yani T/H kullanılmadığı durumda, ADC çevrimi devam ederken girişteki analog işaret değişebileceğinden çevrim sonucu hatalı olacaktır.

ADC'lerin analog işaretini sayısal dönüştürebilmeleri için referans gerilime, Vref, gereksinimleri vardır. Referans gerilim değeri ADC'nin doğru olarak sayısal dönüştürebileceği maksimum gerilim değerini belirler. Aduc841 mimarisinde Vref değeri ADC'nin kendi içinden (dahili=internal) sağlanabileceği gibi harici (external) olarak bağlanabilir. Çevrim öncesinde ADC'nin hangi referans kaynağını kullanacağı belirtilmelidir. Harici referans kaynağı kullanılmak istenmesi durumunda işlemcinin ilgili pinine (C_{REF}) harici bir referans kaynağı bağlanmalıdır. Deney için kullanılan ADUC841 geliştirme kartlarında harici gerilim referans kaynağı $V_{ref}=V_{DD}=5V$ kullanılmaktadır. ADC' nin çalışma ayarları (analog giriş kanalının seçilmesi, referans kaynağının seçilmesi, çevrimin başlatılması, vb.) ADCCON1, ADCCON2 ve ADCCON3 kontrol SFR'leri kullanılarak yapılır.

ADCCON1 Saklayıcısı

Çevrim ve data yakalama sürelerinin ayarlandığı, çevrime başlama modunun seçildiği ve ADC' nin beslemesinin açılıp-kapatıldığı kontrol SFR'sidir. ADCCON1 SFR'si bit adreslenemez.

ADCCON1:	MDI	EXT_REF	CK1	CK0	AQ1	AQ0	T2C	EXC
MSB								LSB

Bit	İsim	Açıklama															
7	MDI	Setlendiğinde, "1", ADC'nin güçlü açılır (Power On) ve ADC analog-sayısal çevrim yapabilir duruma gelir. Temizlendiğinde, "0", ise ADC'nin güçlü kapatılır (Power Off).															
6	EXT_REF	"1", ADC için <u>harici referans</u> kaynağı seçilir. Bu durumda işlemcinin ilgili pinine harici bir referans kaynağı bağlanmalıdır. Deney için kullanılan ADUC841 geliştirme kartlarında harici gerilim referans kaynağı $V_{ref}=V_{DD}=5V$ kullanılmaktadır. "0", ADC için <u>dahili referans</u> ($V_{ref}=2.5V$) kaynağı seçilir															
5	CK1	ADC' nin çalışma frekansı f_{ADC} mikrokontrolöre bağlı olan harici kristal frekansının (f_{osc}) CK0 ve CK1 bitleri ile belirlenen MCLK katsayısına bölünmesi ile elde edilir. MCLK katsayısi $400kHz < f_{ADC} < 8.38MHz$ şartı sağlanacak şekilde aşağıdaki tabloya göre belirlenir. Not: Deneylerde kullanılacak olan geliştirme kartlarında bulunan harici kristal frekansı $f_{osc}=11.0592 MHz$ dir.															
4	CK0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>CK1</th> <th>CK0</th> <th>MCLK</th> </tr> <tr> <td>0</td> <td>0</td> <td>32</td> </tr> <tr> <td>0</td> <td>1</td> <td>4</td> </tr> <tr> <td>1</td> <td>0</td> <td>8</td> </tr> <tr> <td>1</td> <td>1</td> <td>2</td> </tr> </table> $[f_{ADC} = f_{osc} / MCLK]$	CK1	CK0	MCLK	0	0	32	0	1	4	1	0	8	1	1	2
CK1	CK0	MCLK															
0	0	32															
0	1	4															
1	0	8															
1	1	2															
3	AQ1	AQ1 ve AQ0 bitleri ADC mimarisinde bulunan <u>izleyici ve tutucunun</u> (T/H) giriş işaretini yakalaması için kullanılacağı sürenin (t_{TH}) kaç adet ADC periyodundan oluşacağını aşağıdaki tabloya göre belirler. Üretici firma bu sürenin 3 veya 4 ADC periyodundan oluşmasını önermektedir.															
2	AQ0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>AQ1</th> <th>AQ0</th> <th>k_{TH}</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>2</td> </tr> <tr> <td>1</td> <td>0</td> <td>3</td> </tr> <tr> <td>1</td> <td>1</td> <td>4</td> </tr> </table> $t_{TH} = k_{TH} * \text{ADC periyodu}$ $\text{ADC periyodu} = 1/f_{ADC}$	AQ1	AQ0	k_{TH}	0	0	1	0	1	2	1	0	3	1	1	4
AQ1	AQ0	k_{TH}															
0	0	1															
0	1	2															
1	0	3															
1	1	4															
1	T2C	"1" yapıldığında ADC' nin çevrime başlaması (Timer/Counter-T/C) T/C-2'nin taşıma biti (TF2) ile kontrol edilir. T/C-2 taşılığında taşıma bayrağı TF2 donanım tarafından otomatik olarak setlenir ve ADC çevrime başlar. Çevrim bittiğinde TF2 yazılımla temizlenmelidir. T/C-2 için detaylı bilgi ileriki kısımda verilmiştir.															
0	EXC	"1" yapıldığında ADC' nin çevrime başlaması P3.5 (CONVST) pininden gelen düşük-aktif işaret ile kontrol edilir. P3.5 pininden gelen işaretin ADC çevrimini başlatabilmesi için minimum 100ns Lojik-0 mertebesinde kalmalıdır.															

ADCCON2 Saklavacısı

Analog giriş kanalının ve çevrim modunun seçildiği ADCCON2 SFR'si bit adreslenebilir.

ADCCON2:

ADC1	DMA	CCONV	SCONV	CS3	CS2	CS1	CS0
MSB							LSB

SOC → Start Of Conversion

Bit	İsim	Açıklama																														
7	ADCI	ADC kesme bitidir. Çevrim tamamlandığında donanım tarafından otomatik olarak setlenir. Program ADC kesme vektör adresine (ORG 0033H) dallandığında yine donanım tarafından temizlenir. Kesme alt programı kullanılmadığı durumlarda ise yazılım tarafından temizlenmelidir.																														
6	DMA	DMA (Direct Memory Access = Doğrudan Hafızaya Erişim) ADC nin bu özelliği deneyerde kullanılmayacağından DMA = 0 yapılacaktır.																														
5	CCONV	"1" yapıldığı anda ADC sürekli çevrim modunda çalışmaya başlar. Bir çevrim bittiğinde otomatik olarak diğer çevrim başlar. (CCONV = Continuous Conversion = Sürekli Çevrim Modu)																														
4	SCONV	"1" yapıldığı anda ADC tek bir çevrim yapar. Çevrim tamamlandığında donanım tarafından SCONV = 0 yapılır. Yeni bir çevrim için yazılımla tekrar SCONV = 1 yapılmalıdır. (SCONV = Single Conversion = Tek Çevrim Modu)																														
3	CS3	CS0,CS1,CS2,CS3 bitleri aşağıdaki tabloya göre analog giriş kanalını belirlemek için kullanılır.																														
2	CS2	<table border="1"> <thead> <tr> <th>CS3</th><th>CS2</th><th>CS1</th><th>CS0</th><th>Analog Giriş Kanalı</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>Adc0</td></tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td>Adc1</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>0</td><td>Adc2</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>1</td><td>Adc3</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>Adc4</td></tr> </tbody> </table>	CS3	CS2	CS1	CS0	Analog Giriş Kanalı	0	0	0	0	Adc0	0	0	0	1	Adc1	0	0	1	0	Adc2	0	0	1	1	Adc3	0	1	0	0	Adc4
CS3	CS2	CS1	CS0	Analog Giriş Kanalı																												
0	0	0	0	Adc0																												
0	0	0	1	Adc1																												
0	0	1	0	Adc2																												
0	0	1	1	Adc3																												
0	1	0	0	Adc4																												
1	CS1	<table border="1"> <thead> <tr> <th>CS3</th><th>CS2</th><th>CS1</th><th>CS0</th><th>Analog Giriş Kanalı</th></tr> </thead> <tbody> <tr> <td>0</td><td>1</td><td>0</td><td>1</td><td>Adc5</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td>Adc6</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>1</td><td>Adc7</td></tr> </tbody> </table>	CS3	CS2	CS1	CS0	Analog Giriş Kanalı	0	1	0	1	Adc5	0	1	1	0	Adc6	0	1	1	1	Adc7										
CS3	CS2	CS1	CS0	Analog Giriş Kanalı																												
0	1	0	1	Adc5																												
0	1	1	0	Adc6																												
0	1	1	1	Adc7																												
0	CS0	<table border="1"> <thead> <tr> <th>CS3</th><th>CS2</th><th>CS1</th><th>CS0</th><th>İşlemci dahili sıcaklık sensörü</th></tr> </thead> <tbody> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>DAC0</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>0</td><td>DAC1</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>1</td><td>AGND</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>0</td><td>Vref</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>1</td><td>DMA durdur</td></tr> </tbody> </table>	CS3	CS2	CS1	CS0	İşlemci dahili sıcaklık sensörü	1	0	0	0	DAC0	1	0	1	0	DAC1	1	0	1	1	AGND	1	1	0	0	Vref	1	1	1	1	DMA durdur
CS3	CS2	CS1	CS0	İşlemci dahili sıcaklık sensörü																												
1	0	0	0	DAC0																												
1	0	1	0	DAC1																												
1	0	1	1	AGND																												
1	1	0	0	Vref																												
1	1	1	1	DMA durdur																												

Yukarıda verilen açıklamalarda belirtildiği gibi Aduc841 mimarisinde bulunan ADC için analog-sayısal çevrimi 4 farklı yöntemle başlatılabilir.

- i) Tek çevrim modu (SCONV = ADCCON2.4)
 - ii) Sürekli çevrim modu (CCONV = ADCCON2.4)
 - iii) Z/S-2 taşıma bayrağı (TF2)
 - iv) P3.5(CONVST) pininden gelen harici işaret

Gerçekleştirilmek istenen uygulamaya bağlı olarak bu yöntemlerden herhangi biri ADC çevrimini başlatmak için kullanılabilir.

ADCCON3 Saklavıçısı

ADC'nin kalibrasyon ayarlarının yapıldığı ADCCON3 SFR'si bit adreslenemez.

ADCCON3:

BUSY	RSVD	AVGS1	AVGS0	RSVD	RSVD	TYPICAL	SCAL
				MSB	LSB		

Bit No	İsim	Açıklama
7	BUSY	BUSY bitinin içeriği okunabilir fakat yazılımla değeri değiştirilemez (Read Only). BUSY biti analog-sayısal çevrimi veya ADC nin kalibrasyonu devam ederken donanım tarafından setlenir (BUSY=1). Çevrim veya kalibrasyon bittiğinde yine donanım tarafından temizlenir.
6	RSVD	Bu bit donanım gereği daima 0 olmalıdır
5	AVGS1	*
4	AVGS0	*
3	RSVD	Bu bit donanım gereği daima 0 olmalıdır
2	RSVD	*
1	TYPICAL	*
0	SCAL	*

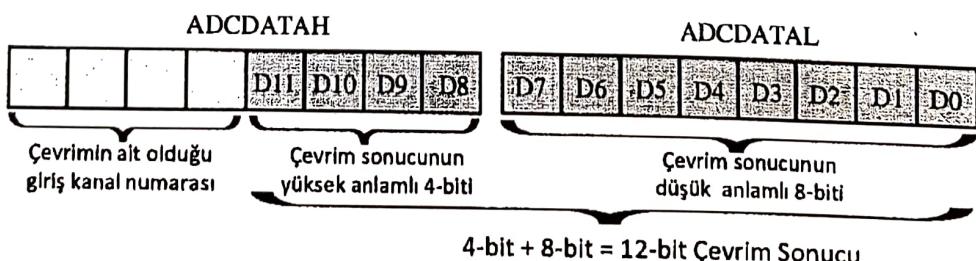
* Bu bitler ADC'nin kalibrasyonu ile ilgiliidir. Deneylerde kullanılmayacağı için burada açıklama verilmemiştir. Detaylı bilgi için Aduc841 kataloğu incelenebilir.

Yukarıda yöntemlerden herhangibiri ile başlatılan ADC çevrimi tamamlandığında ADC girişindeki analog işaretin 12-bit'lik sayısal karşılığı **donanım tarafından** otomatik olarak ADCDATAH ve ADCDATAL veri SFR'lerine yüklenir. Çevrim sonucu yazılımla bu SFR'lerden okunabilir. ADC çevriminin tamamlandığı 2 farklı şekilde tespit edilebilir.

- i) **Kesme:** ADC kesmesi (EADC=1) ve tüm kesmeleri yetkilendirme biti (EA=1) aktif yapılır. Bu durumda ADC çevrimi tamamlandığı anda ADC kesme biti (ADCI = ADCCON2.7) donanım tarafından setlenir ve programın akışı ADC kesme vektör adresine (ORG 0033H) dallanır. *Deney 2'yi inceleyiniz.*

ii) **Yoklama (Polling):** Kesme kullanılmadığı durumda ADCI biti çevrim başlatılmadan önce yazılımla temizlenir ve donanım tarafından ADCI bitinin setlenmesi beklenir. *Deney 1'i inceleyiniz.*

Şekil 2'de gösterildiği gibi ADCDATAH SFR'sinin yüksek değerli 4-bit'lik kısmı çevrim sonucunun ait olduğu analog giriş kanalını belirtir. Bu nedenle ADC çevrim sonucu aritmetik işlemlerde kullanılacak ise ADCDATAH'dan okunan değerin yüksek 4-bit'i yazılımla temizlenmelidir.



Şekil 2 : ADC 'nin çevrim sonucu donanımı tarafından doğrudan ADCDATA1/H saklayıcılarına aktarılacak.

Çevrim Süresi

A/D çevriminin başladığı an ile çevrim sonucunun ADC'nin sayısal çıkışlarında (ADCDATAH ve ADCDATA1) hazır hale geldiği an arasındaki süre ADC'nin çevrim süresini verir. Çevrim süresi ADC'lerin seçiminde dikkat edilmesi gereken önemli kriterlerden biridir. ADUC841 mikrokontrolöründe bulunan 12-bitlik ADC'nin çevrim süresi, 16 ADC periyodu + T/H için seçilen ADC periyodu (1...4) ifadesi ile hesaplanır. (**1 ADC periyodu = $1/f_{ADC}$ dir.**) ADC'nin çalışma frekansı (f_{ADC}) yukarıda belirtildiği gibi ADDCON1 deki CK0-CK1 bitleri ile ayarlanır. Dolayısıyla ADUC841 mimarisinde bulunan ADC'nin çevrim süresi ADC için seçilen çalışma frekansına bağlı olarak değişmektedir.

Örnek: Kristal frekansı $f_{osc}=11.0592$ olan geliştirme kartlarında ADC için $MCLK = 2$ ($CK0 = 1$ ve $CK1=1$) ve T/H için 4 ADC periyodu ($AQ0=1$ ve $AQ1=1$) seçilmesi durumunda 1 ADC çevrim süresinin değeri aşağıdaki gibi hesaplanır.

$$f_{ADC} = \frac{f_{osc}}{MCLK} = \frac{f_{osc}}{2} = \frac{11.0592MHz}{2} = 5.5296MHz$$

$$\text{ADC periyodu} = \frac{1}{f_{ADC}} = \frac{1}{5.5296 \times 10^6} = 0.1808 \times 10^{-6} s \text{ olur.}$$

Bu durumda ADC çevrim süresi = 16 ADC periyodu + T/H için seçilen ADC periyodu
 $= (16+4)$ ADC periyodu
 $= 20 \times 0.1808 \times 10^{-6} s \approx 3.617 \times 10^{-6} s = 3.617 \mu s$ olarak hesaplanır.

ADC Gerilim Çözünürlüğü, Q

ADC gerilim çözünürlüğü (Q) ADC'nin sayısal çıkış değerinin değişmesine neden olacak giriş işaretindeki en küçük gerilim değişimini tanımlar. Başka bir ifade ile ADC'nin algılayabileceği en küçük gerilim değeri olarak tanımlanabilir. ADC gerilim çözünürlüğü aşağıdaki gibi hesaplanır.

$$Q = \frac{V_{FSR}}{2^N}, \quad V_{FSR} = \text{ADC ölçme aralığı (range)} = V_{max} - V_{min}, \quad N = \text{ADC bit sayısı}$$

Aduc841 mimarisinde bulunan ADC tek kutuplu olduğundan $V_{min} = 0V$ 'dur. V_{max} değeri dahili referans seçilmesi durumunda $V_{max} = V_{ref} = 2.5V$, harici referans seçilmesi durumunda ise $V_{max} = V_{ref} = 5.0V$ olacaktır. Bu durumda deneyde kullanılacak olan 12-bit ADC için Q değeri;

$$\text{dahili referans } (V_{ref} = 2.5V) \text{ seçilmesi durumunda; } Q = \frac{2.5 - 0}{2^{12}} = 0.61mV$$

$$\text{harici referans } (V_{ref} = 5.0V) \text{ seçilmesi durumunda; } Q = \frac{5.0 - 0}{2^{12}} = 1.22mV$$

olarak hesaplanır.

successive approximation ADC → otomasyon
 flash ADC → $N, NSN \rightarrow$ gorsnto işlene

T2CON Saklayıcısı

Sayısal işlemci tabanlı ölçme ve/veya kontrol işlemleri çoğunlukla belirli bir T değeri ile periyodik olarak gerçekleştirilir. T değeri ölçülecek analog işaretin veya kontrol edilecek fiziksel sistemin dinamiğine bağlı olarak değişir.

Yukarıda belirtildiği gibi Aduc841 mimarisinde bulunan ADC Zamanlayıcı/Sayıçı-2 (Z/S-2)'nin taşıma biti TF2 ile de çevrime başlatılabilmiştir. ADC'nin TF2 biti ile çevrime başlatılabilmesi analog işaretin sabit bir T periyodu ile örneklenmesini kolaylaştırır. Tasarımcı tarafından belirlenen T periyodu Z/S-2'nin TH2/TL2 saklayıcılarına gerekli değerler yazılarak üretilir.

Örnek: Analog işaret T=5ms periyodu ile örneklenmek istensin.

Geliştirme kartları için kristal frekansı $f_{osc} = 11.0592MHz$ 'dır.

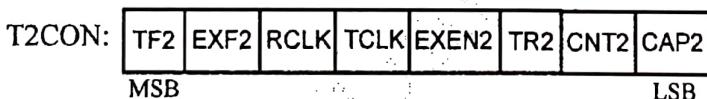
Bu durumda Aduc841 için 1 makine çevrimi $= \frac{1}{f_{osc}} = \frac{1}{11.0592MHz} \approx 90.42 \times 10^{-9} s$ olur.

5 ms'lik süre için $\frac{5 \times 10^{-3} s}{90.42 \times 10^{-9} s} = 55297.5 \rightarrow 55298$ makine çevriminin geçmesi gereklidir.

16-bit Zamanlayıcı2' nin maksimum sayma değeri = FFFFh = 65535'dir.

Bu durumda Zamanlayıcı2 için sayma başlangıç değeri $65535 - 55298 = 10237d = 27FDh$ olarak hesaplanır. Bu sayının yüksek değerli kısmı (27h) TH2 ye düşük değerli kısmı (FDh) ise TL2 ye yüklenmesi gereken değerlerdir. Böylece Zamanlayıcı2 27FDh değerinden başlayarak yukarı yönde sayacak ve FFFFh değerine ulaşana kadar toplam 55298 makine çevrimi olmuş olacaktır. Böylece T=5ms süresi üretilmiş olacaktır.

Z/S-2'nin çalışma ayarlarının yapıldığı T2CON SFR'si bit adreslenebilirdir. T2CON saklayıcısındaki bitlerin kullanımı şu şekildedir;



Bit No	İsim	Açıklama
7	TF2	Z/S-2 taşıma bayrağı; Z/S-2 taşılığında donanım turafından setlenir (TF2=1). Kesme altprogramı kullanılsa dahi yazılım tarafından temizlenmesi gereklidir.
6	EXF2	EXEN2=1 iken yakalama (Capture) veya yeniden yükleme (Reload) olaylarından biri gerçekleştiğinde T2EX (P1.1)'den gelen işaretin düşen kenarında (↓) donanım turafından setlenir. Yazılımla temizlenir.
5	RCLK	RCLK (Receive Clock Enable Bit) Yazılım tarafından setlendiğinde seri portun Mode1 ve Mode3 çalışmalarında seri kanaldan veri almak için gerekli saat darbeleri Z/S-2'nin taşıma biti turafından sağlanır. Temizlendiğinde bu işlem için Z/S-1'in taşıma biti kullanılır.
4	TCLK	TCLK (Transmit Clock Enable) Seri kanaldan veri gönderme için RCLK ile aynı şekilde kullanılır.
3	EXEN2	EXEN2 Z/S-2 seri kanal için saat üretici olarak kullanılmadığı durumlarda, yazılım tarafından EXEN=1 yapılarak T2EX (P1.1) pininden gelen işaretin düşen kenarında (↓) yakalama veya yeniden yükleme olayının gerçekleşmesi sağlanır.
2	TR2	TR2 (Z/S-2 Start/Stop) Setlendiğinde Z/S-2 çalışmaya başlar, temizlendiğinde Z/S-2 durur.
1	CNT2	CNT2 setlendiğinde T2 pininden gelen pozitif darbeleri saymak için <u>Sayıçı</u> olarak ayarlanır. Temizlendiğinde ise <u>Zamanlayıcı</u> olarak kullanılır.
0	CAP2	CAP2 Setlendiğinde (eğer aynı anda EXEN2=1 ise) T2EX (P1.1)'den gelen işaretin düşen kenarında (↓) yakalama işlemi olur.

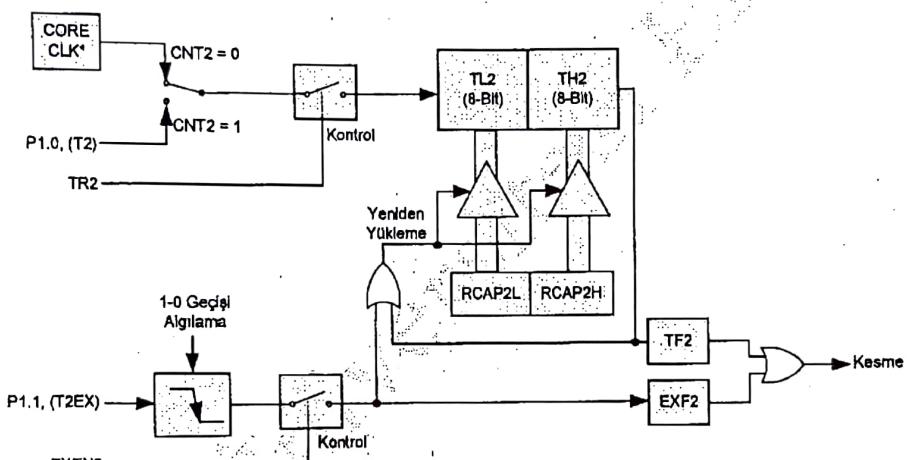
Z/S-2 'nin T2CON saklayıcısı tarafından belirlenen 3 çalışma modu aşağıda verilmiştir.

RCLK (veya) TCLK	CAP2	TR2	Mode
0	0	1	16-bit Otomatik yeniden yüklemeli mod
0	1	1	16-bit Yakalama modu
1	X	1	Seri Harleme için BaudRate Üreteci
X	X	0	OFF

Z/S-2 'nin kendisine ait 4 adet 8-bitlik veri hafızası vardır (TH2-TL2, RCAP2H-RCAP2L). TH2 ve TL2, Z/S-2 'nın çalışma esnasındaki sayma değerini tutar. Dolayısıyla herhangi bir anda Z/S-2 'nın değeri bu saklayıcılarından okunabilir.

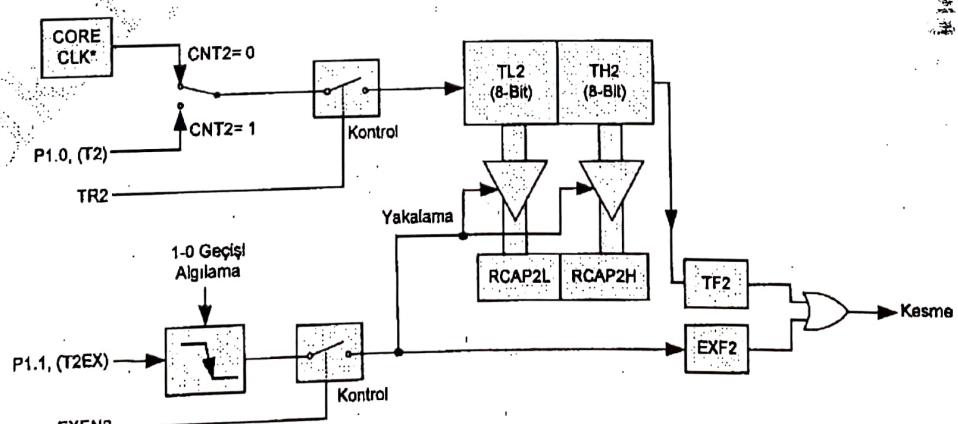
RCAP2H ve RCAP2L saklayıcıları ise otomatik yükleme (autoreload) modunda sırasıyla TH2 ve TL2'ye yüklenecek değerleri tutar. Şekil 3'te gösterildiği Z/S-2'de taşıma gerçekleştirliğinde taşıma bayrağı TF2 donanım tarafından setlenir ve RCAP2H - RCAP2L saklayıcılarında tutulan değerler donanım tarafından TH2 – TL2 saklayıcılarına yüklenir.

EXEN2=1 yapılması durumunda otomatik yükleme işlemi P1.1 pininden gelen işaretin düşen kenarında gerçekleştirilir.



Şekil 3. Zamanlayıcı/Sayıci-2 için 16-Bit “yeniden yükleme modu” blok diyagramı

Şekil 4'te gösterilen yakalama (capture) modunda ise, P1.1 pininden gelen işaretin düşen kenarında yakalama komutu geldiği anda TH2 ve TL2'in değerleri bu saklayıcılara aktarılır. Aşağıdaki her iki çalışma durumu gösterilmiştir.



Şekil 4. Zamanlayıcı/Sayıci-2 için 16-Bit “yakalama modu” blok diyagramı

Ön Çalışma

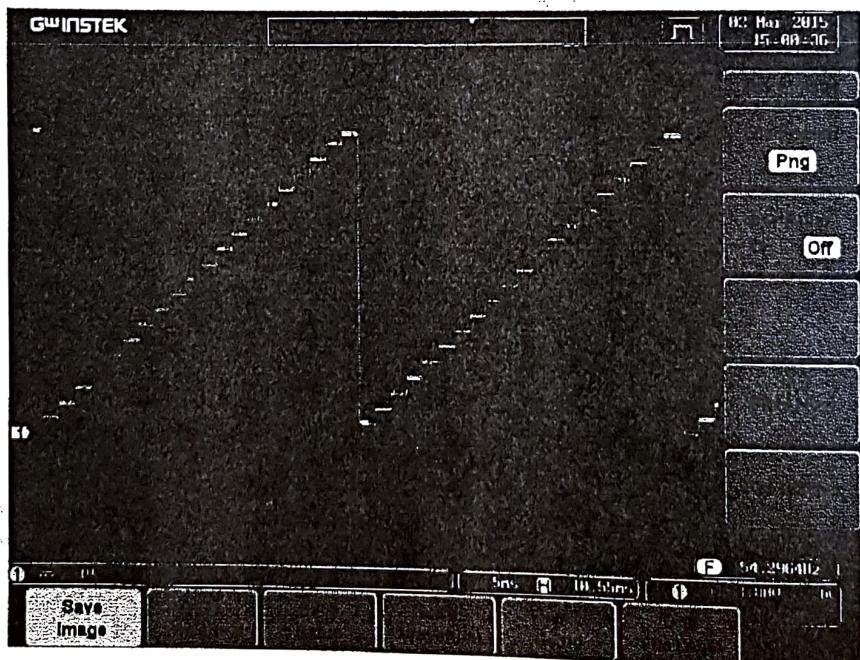
1. $T = 1ms$ periyodu ile $u[k] = u[k-1]+200$ (başlangıçta $u[0]=0$) işlemi gerçekleştirilip sonuç DAC0'a yazılması istenmektedir.

Bu amaçla;

- i) VeriH (RAM'ın 41h adresi, yüksek anlamlı byte) ve VeriL (RAM'ın 40h adresi, düşük anlamlı byte) olarak iki değişken tanımlayıp başlangıç değerlerini 0 olarak giriniz.
- ii) $T=1ms$ periyodu Zamanlayıcı-2 kesme yapısı kullanarak üretilecektir. Yukarıda verilen örneği inceleyerek $T=1ms$ için gerekli başlangıç değerini hesaplayıp TH2-TL2 ve RCAP2H- RCAP2L SFR'lerine yükleyiniz. Timer kesme ayarlarını yapınız.
- iii) DAC0 (12-bit, harici referans) ayarlarını yapınız.
- iv) T2CON SFR'si ile Zamanlayıcı/Sayıci-2 16-bit otomatik yeniden yüklemeli zamanlayıcı olarak çalıştırıp programı sonsuz döngü içerisinde beklemeye alınız.
- v) Zamanlayıcı-2 kesme hizmet alt programında VeriH-VeriL değişkenlerinde tutulan değeri sabit 200d ile toplayıp (add/addc komutlarını kullanarak) sonuçları hem DAC0'a hemde bir sonraki döngü için VeriH-VeriL değişkenlerine yazınız.

Bu uygulamada DAC0 çıkışında oluşması beklenen işaret Şekil 5'te gösterilmiştir.

2. Yukarıda istenen işlemi $T=500ms$ periyodu ile gerçekleştiriniz. Bu uygulama için T periyodu Zamanlayıcı-2 yoklamalı kullanarak üretilecektir.



Şekil 5. Ön çalışmaya ait DAC0 çıkışında gözlemlenmesi beklenen işaret

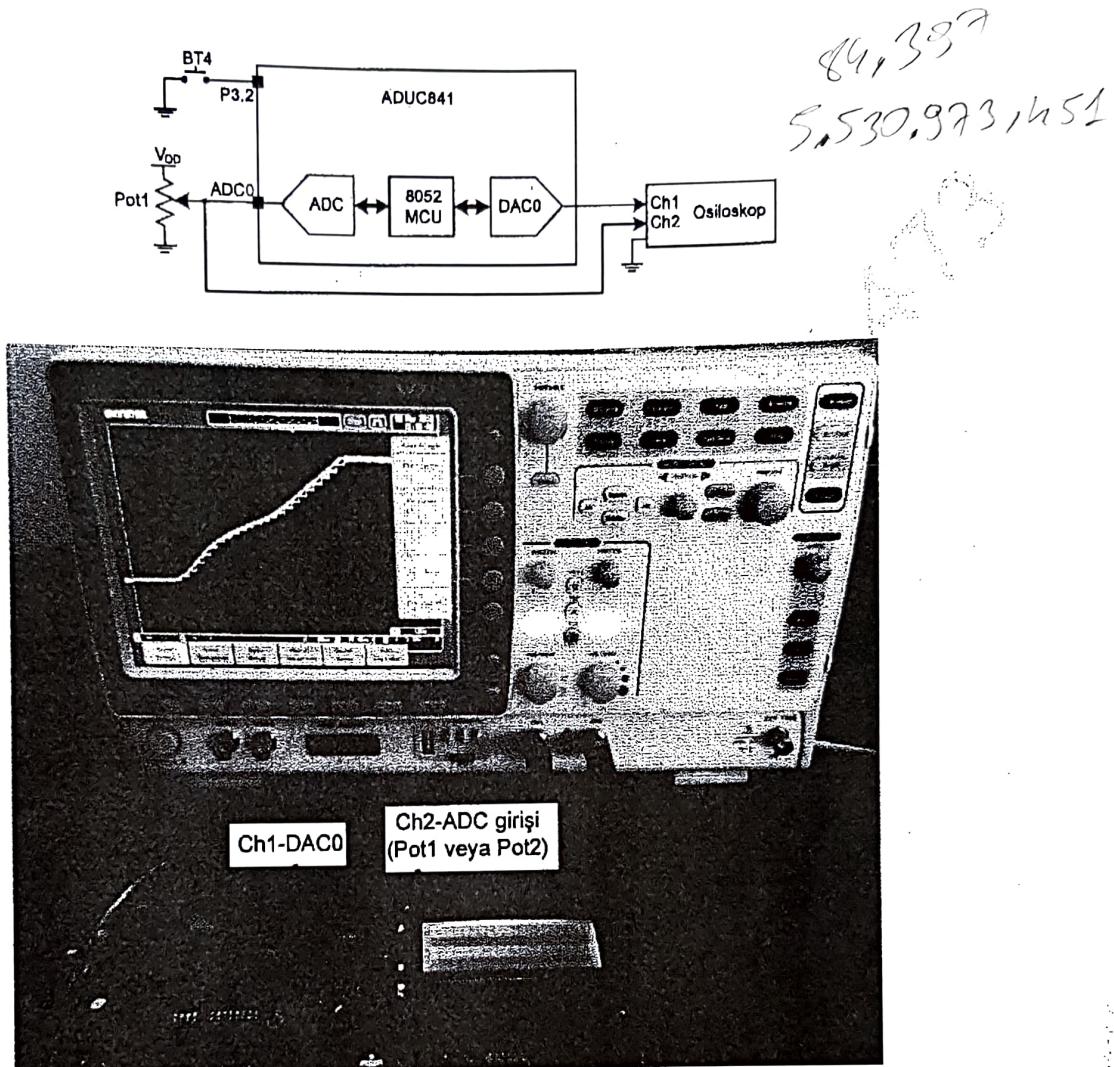
Deney 1.

Deneyde kullanılacak olan donanıma ait basitleştirilmiş şematik gösterim ve gerçek zaman görüntüsü aşağıda verilmiştir, Şekil 6.

Deneyde kullanılacak olan geliştirme kartı üzerindeki BT4 (P3.2/int0) butonuna her basılıp bırakıldığından (yoklama ile algılanacak) ADC0 kanalına bağlı olan potansiyometreden (Pot1) gelen analog işaret sayısal çevirilerek doğrudan DAC0'a yazılacaktır. Analog-sayısal çevrimin tamamlandığı ADCI (ADCCON2.7) biti yoklanarak belirlenecektir.

Bu uygulamada ADC çalışma ayarlarını:

tek çevrim modu, MCLK=4 ($f_{ADC}=11.0592\text{MHz}/4 = 2.765\text{MHz}$), T/H için 4 ADC periyodu ve dahili referans kaynağı şeklinde olması istenmektedir. DAC0, 12-bit, dahili referans modunda kullanılacaktır.



Şekil 6. Deney düzeneğinin basitleştirilmiş blok diyagramı ve gerçek zaman görüntüsü

Deneyin gerçekleştirilmesi:

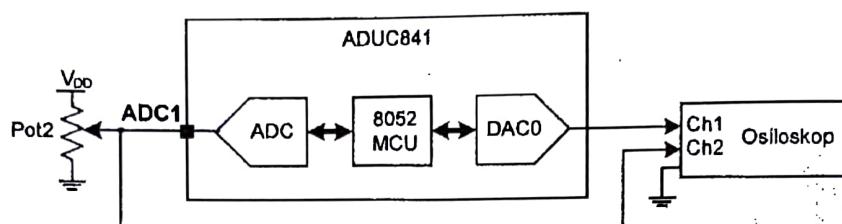
- 1- ADC için belirtilen çalışma ayarları ADCCON1 SFR'sine gerekli değerler yüklenerek yapılır.
- 2- Analog giriş kanalının (ADC0) seçimi ADCCON2de bulunan CS0...CS3 bitleri yardımıyla yapılır.
- 3- DAC0 ayarları DACCON SFR'sine gerekli değerler yüklenerek yapılır.
- 4- P3.2'ye bağlı butondan negatif darbe (1-0-1, basılıp-bırakılması) üretilmesi beklenir.
- 5- SCONV=1 yapılarak ADC çevrimi başlatılır.
- 6- ADCI biti yoklanarak çevrimin tamamlanması beklenir.
- 7- ADC çevrimi tamamlandığında çevrim sonuçları ADCDATAH/L saklayıcılarından DAC0H/L saklayıcılarına yüklenir.

ADC0 girişine bağlı olan Pot1'i minimum-maksimum arasında değiştirerek DAC0 çıkışını ADC0 girişine bağlı olan Pot1'i minimum-maksimum arasında değiştirerek DAC0 çıkışını gözlemleyiniz. Potun her durumu için DAC0 çıkış ile ADC giriş aynı değerdemidir? Neden?

Deney 2.

Bu deneyde Pot2'den gelen ve ADC'nin ADC1 kanalına bağlı olan analog işaret $T=5$ ms periyodu ile örneklenecek doğrudan DAC0'a yazılıacaktır. Analog-sayısal çevrimin bittiği ADC'ye ait kesme ile belirlenecektir.

Bu uygulama için **ADC çalışma avarları**; MCLK=2 ($f_{ADC}=11.0592\text{MHz}/2 = 5.5296\text{MHz}$), T/H için 4 ADC periyodu, harici referans kaynağı ve ADC nin çevrime başlaması Zamanlayıcı/Sayıci2 taşıma biti TF2 ile sağlanacaktır. DAC0, 12-bit, harici refarans modunda kullanılacaktır.

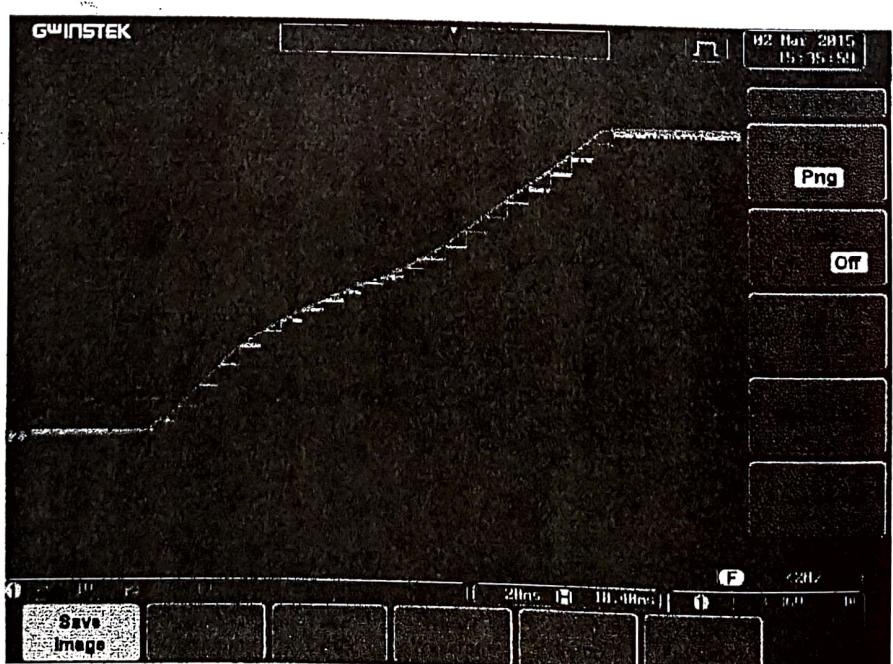


Deneyin gerçekleştirilmesi;

- 1- ADC için belirtilen çalışma ayarları ADCCON1 SFR'sine gerekli değerler yüklenerek yapılır.
- 2- Analog giriş kanalının (ADC1) seçimi ADCCON2de bulunan CS0...CS3 bitleri yardımıyla yapılır.
- 3- DAC0 ayarları DACCON SFR'sine gerekli değerler yüklenerek yapılır.
- 4- T2CON saklayıcısından Zamanlayıcı/Sayıci2 16 bit otomatik-yeniden-yükleme zamanlayıcı olarak ayarlanır.
- 5- $T=5\text{ms}$ periyodu için TH2-TL2 ve RCAP2H-RCAP2L saklayıcılarına yüklenmesi gereken değer $10237d = 27FDh$ olarak yukarıda hesaplanmıştır. Bu değer saklayıcılara yüklenir.
- 6- İlgili kesme bitleri setlenir (EA ve EADC) ve Zamanlayıcı-2 çalıştırılır.
- 7- Program sonsuz döngüye sokulup ADC kesmesinin oluşması beklenir.
- 8- ADC kesme hizmet alt programında çevrim sonuçları ADCDATAH/L saklayıcılarından DAC0H/L saklayıcılarına yüklenir.

ADC1 girişine bağlı olan Pot2'yi 0V-5V arasında istenen hızda değiştirerek DAC0 çıkışını gözlemleyiniz. Potun her durumu için DAC0 çıkışı ile ADC girişi aynı değerdemidir?

Aşağıda Pot2'nin 0V'dan 5V'a hızlıca çevrildiği bir uygulamaya ait osiloskop görüntüsü verilmiştir.



SAU / Muh.Fak./ EEM/ Analog-Sayısal Çeviriciler

Şekil 7. Ch1 – DAC0 çıkışı, Ch2 – ADC girişi,

Hatırlatma:

Aduc841 için kesme yetkilendirme SFR'si (IE), kesme vektör adresleri ve doğal öncelik sırası aşağıda verilmiştir.

IE:	EA	EADC	ET2	ES	ET1	EX1	ET0	EX0
MSB					LSB			

Bit	İsim	Açıklama
7	EA	EA=1 tüm kesmeler aktif, EA=0 ise tüm kesmeler pasif.
6	EADC	EADC = 1/0 ADC kesmesi aktif/pasif.
5	ET2	ET2 = 1/0 zamanlayıcı/sayıçı-2 kesmesi aktif/pasif.
4	ES	ES = 1/0 seri veri alma/gönderme kesmesi aktif/pasif.
3	ET1	ET1 = 1/0 zamanlayıcı/sayıçı-1 kesmesi aktif/pasif.
2	EX1	EX1 = 1/0 harici kesme1 aktif/pasif.
1	ET0	ET0 = 1/0 zamanlayıcı/sayıçı-0 kesmesi aktif/pasif.
0	EX0	EX = 1/0 harici kesme 0 aktif/pasif.

Kesme Vektör Adresleri:

Kesme Kaynağı	Vektör Adresi
Harici Kesme 0	0003H
Zamanlayıcı/Sayıçı 0	000BH
Harici Kesme 1	0013H
Zamanlayıcı/Sayıçı 1	001BH
Seri Kanal (RI+TI)	0023H
Zamanlayıcı/Sayıçı 2	002BH
ADC	0033H
ISPI / I2CI	003BH
PSMI	0043H
TII	0053H
WDS	005BH

Kesme Öncelik Sırası:

Kesme Kaynağı	Öncelik Sırası
PSMI	1 (En yüksek)
WDS	2
Harici Kesme 0	2
ADC	3
Zamanlayıcı/Sayıçı 0	4
Harici Kesme 1	5
Zamanlayıcı/Sayıçı 1	6
ISPI / I2CI	7
Seri Kanal (RI+TI)	8
Zamanlayıcı/Sayıçı 2	9
TII	11(En düşük)

Deney 1 için örnek kod aşağıda verilmiştir.

```
#include      <aduc841.h>

BT4          EQU    P3.2H

        ORG      0000h
        SJMP    BASLA

BASLA:
        MOV     ADCCON1,#10011100B ; ADC Power-On, Dahili referans, MCLK=4,
                                ; T/H için 4 darbe
        MOV     ADCCON2,# 0H       ; ADC0 kanalı seçildi...
        MOV     DACCON, 01001101B ; DAC0 12-bit, dahili referans(2.5V) olarak ayarlandı.

B1:   JB      BT4, B1      ; Buton için negatif-darbe (1-0-1) yoklaması yapılıyor..
        ACALL  BEKLE
B0:   JNB    BT4, B0

        CLR     ADCI      ; ADC kesme bayrağını temizle..
        SETB    SCONV    ; ADC çevrimi başlaşın (tek çevrim!!)

DUR:  JNB    ADCI, DUR ; ADC çevriminin tamamlanmasını bekle..

        MOV     DAC0H,ADCDATAH ; çevrim sonuçları ADCDATAH/L okunarak
        MOV     DAC0L,ADCDATAL ; doğrudan DAC0H/L'a yazılıyor..
        SJMP    B1           ; bir sonraki örnek için tekrar butona basılmasını bekle..

BEKLE:
        MOV     41H,#0FFH ; iç içe döngü yapısı ile gecikme oluşturuluyor...
X0    MOV     40H,#0FFH
X1:  DJNZ   40H, X1
        DJNZ   41H, X0
        RET

END
```

Not: ADC çevrimi tamamlandığında donanım SCONV bitini "0"'a çekmektedir. Bu durumda

DUR: JB SCONV, DUR
komutu ilede beklenebilir.

Deney 2 için örnek kod aşağıda verilmiştir.

#include <aduc841.h>

ORG 0000h
SJMP BASLA

ORG 0033H ;ADC Kesme adresi
SJMP OKU_YAZ

BASLA:

MOV ADCCON1, #11111110B ; ADC Power-On, Harici referans, MCLK=2,
; T/H için 4 darbe, TF2 ile çevrim başlat

MOV ADCCON2, #01H ; ADC1 kanalı seçildi

MOV DACCON, #01101101B; DAC0 12-bit, harici referans(5V) olarak ayarlandı..

MOV T2CON, #0H ; 16-bit otomatik-yeniden-yüklemeli zamanlayıcı

MOV RCAP2L,#0FDH ; T=5ms için hesaplanan değerler ilgili

MOV RCAP2H,#027H ; SFR'lere yükleniyor..

MOV TL2,#0FDH

MOV TH2,#027H

SETB EADC ; ADC kesmesi aktif

SETB EA ; tüm kesmeler aktif

SETB TR2 ; Timer2 başlat..

DUR: JMP DUR ; Timer2 taşarak TF2=1 olacak ve ADC çevrimi başlayacak..

OKU_YAZ: ; ADC çevrimi bitti !!

CLR TF2 ; Z/S-2 taşıma bayrağını temizle..

MOV DAC0H,ADCDATAH ;çevrim sonuçları ADCDATAH/L okunarak

MOV DAC0L,ADCDATAL ;doğrudan DACOH/L'a yazılıyor..

RETI

END

Deney 4 : Sayısal Analog Çevirici (DAC) ile Çift Ton Çok Frekanslı (DTMF) İ işaret Üretilmesi

Amaç:

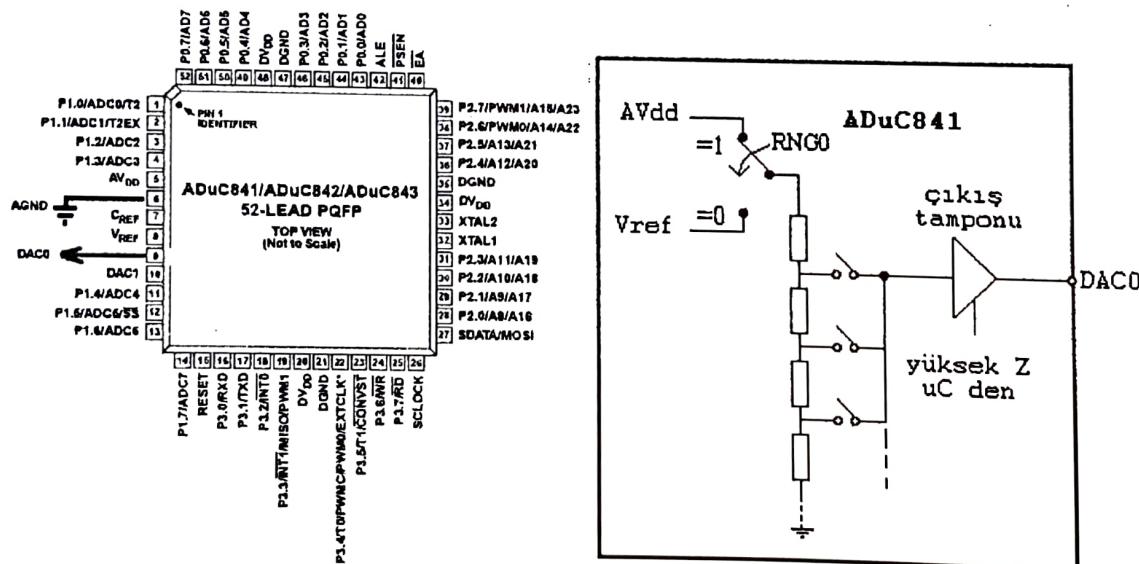
Aduc841 mikrokontrolörün mimarisinde bulunan DAC (Digital to Analog Converter, Sayısal Analog Çevirici) modülünü kullanabilme becerisini kazandırılması.

DAC

Sayısal tabanlı işlemcilerin (μ C, DSP, PLC, PC vb.) giriş ve çıkış işaretleri sayısalıdır (1010101...). Diğer taraftan fiziksel sistemlerin giriş-çıkış işaretleri ise çoğunlukla sürekli zaman analog işaretlerdir. Bu nedenle fiziksel işaretlerin-sistemlerin ölçme, kontrol vb. işlemlerinin sayısal işlemciler ile gerçekleştirilebilmesi için *Analog-Sayısal* ve *Sayısal-Analog* dönüştürücülere ihtiyaç vardır. Bu deneyde, sayısal işlemcinin ürettiği *ayrık-zaman sayısal işaret* fiziksel sistemlere uygulanmak üzere *sürekli-zaman analog* işaretе dönüştüren DAC'lar incelenecaktır.

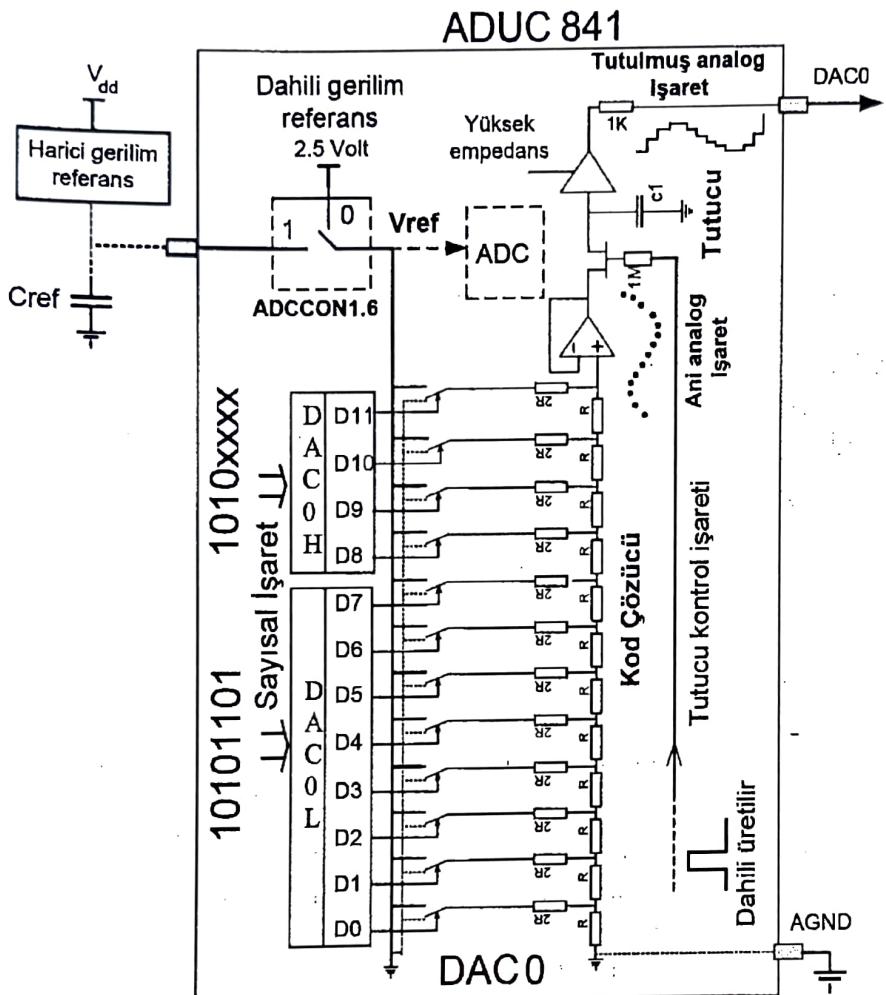
ADUC841 mimarisinde iki adet, 12-Bit gerilim çıkışlı direnç-dizisi (resistor string) türünde DAC (DAC0 ve DAC1) modülü bulunmaktadır. Kullanıcı tercihine göre, DAC0 ve DAC1'in çıkış gerilim aralıkları 0V–Vref (=2.5V) veya 0V–AVDD olarak ayrı ayrı veya her iki DAC içinde aynı gerilim aralıkları seçilebilir.

Aşağıda sırası ile 52-pin PQFP kılıf ADUC841 mikrokontrolör ve basitleştirilmiş dahili DAC verilmiştir.



Şekil 1 ADuC841 mikrokontrolör ve mimarisinde bulunan DAC modülünün basitleştirilmiş yapısı.

ADUC841 mimarisinde bulunan DAC birimleri, lojik '1' ve '0' lardan oluşan 12-bit uzunluğundaki o andaki sayısal işaretin anı analog işaretin *kod çözülcü* ve bu anı analog işaretin bir sonraki anı analog işaret gelinceye kadar sabit tutan *tutucu* olmak üzere iki temel bloktan oluşan donanımsal bir birimdir. ADUC841 mimarisinde bulunan DAC0 ve DAC1 donanımsal olarak özdeşdir. Aşağıda DAC0 fonksiyonel özellikler için basitleştirilmiş blok olarak verilmiştir.



Sekil 2 DAC0 basitleştirilmiş blok şeması

Vref: DAC çıkış gerilim aralığının "0" sıfır volt'tan "2.5" volt'a kadar olması için dahili gerilim referans **Vref**'in seçilmesi durumunda DAC'ların doğru çalışabilmesi için ADC biriminin enerjilendirilmesi **MECBURIDİR** (Power-On). Çünkü dahili $V_{ref} = 2.5$ volt gerilim referans üreticisi yukarıda verilen şekilde görüldüğü gibi ADC ve DAC tarafından ortak olarak kullanılmaktadır.

Eğer dahili gerilim referans $V_{ref}=2.5$ volt kullanılacak ise, aşağıda verilmiş olan ADCCON1 kaydedicisinde ilgili bitler kullanıcı tarafından yazılım ile,

ADCCON1: Bit 7 Bit 0

MDI	EXT_REF	CK1	CK0	AQ1	AQ0	T2C	EXC
-----	---------	-----	-----	-----	-----	-----	-----

ADCCON1.7 (MDI)=1 ve ADCCON1.6 (EXT_REF)=0 yapılmalıdır, (*İşlemci donanımsal olarak reset edildiğinde veya ilk enerjilendirildiğinde ADCCON1.7=0 ve ADCCON1.6=0* dır.).

ADUC841 mimarisinde yer alan DAC0 ve DAC1'in kullanıcı isteğine göre koşullandırılmaları DACCON (Special Function Register (SFR)) özel fonksiyon kaydedicisi ile yapılmaktadır.

DACCON Saklayıcısı

ADUC841 mimarisinde bulunan her iki DAC modülünün (DAC0 ve DAC1) kullanıcı isteğine göre koşullandırımları bit adreslenemez **DACCON** özel fonksiyon kaydedicisi ile yapılmaktadır. Aşağıda tablo halinde her bir bite ait açıklamalar verilmiştir.

DACCON:	MODE	RNG1	RNG0	CLR1	CLR0	SYNC	PD1	PD0
---------	------	------	------	------	------	------	-----	-----

Bit	İsim	Açıklama
7	MODE	"1", Her iki DAC 8-bit mod ve sayısal veriler DACxL yazılır. "0" ise her iki DAC 12-bit mod çalışır, veriler sırası ile DACxH ve DACxL yazılır.
6	RNG1	"1", DAC1 için çıkış gerilim aralığı 0V–AVDD, "0" ise 0V–Vref (2.5V) olarak seçilir.
5	RNG0	"1", DAC0 için çıkış gerilim aralığı 0V–AVDD, "0" ise 0V–Vref (2.5V) olarak seçilir. .
4	CLR1	"1", DAC1 çıkışı DAC1H/L kaydedicilerinde tutulan sayısal veriye karşılık gelen analog değer üretilir. "0" ise DAC1 çıkışı 0V'a çekilir.
3	CLR0	"1", DAC0 çıkışı DAC0H/L kaydedicilerinde tutulan sayısal veriye karşılık gelen analog değer üretilir. "0" ise DAC0 çıkışı 0V'a çekilir.
2	SYNC	"1", DACXL (X=0,1) kaydedicisine veri yazıldığı anda DACX çıkışı güncellenir. "0", ise, her iki DAC senkron çalışır. Bu modda, öncelikle DACXL/H kaydedicilerine veriler yazılır; daha sonra da SYNC biti yazılır ile "1" yapılır ve her iki DAC çıkışı aynı anda güncellenir, çıkış üretir.
1	PD1	"1" DAC1 enerjilendirilir (Power-On). "0" DAC1 enerjisi kesilir (tasarruf için).
0	PD0	"1" DAC0 enerjilendirilir (Power-On). "0" DAC1 enerjisi kesilir (tasarruf için).

Analog işarete dönüştürülecek sayısal işaret, kullanılacak olan DAC1 ve/veya DAC0 için sırası ile DAC1H/L ve/veya DAC0 için ise DAC0H/L özel fonksiyon kaydedicilerine sağa yanaşık olarak yazılır.

DACXH:	x	x	x	x	D11	D10	D9	D8
--------	---	---	---	---	-----	-----	----	----

DACXL:	D7	D6	D5	D4	D3	D2	D1	D0
--------	----	----	----	----	----	----	----	----

NOT: 12-bit sayısal verinin önce yüksek anlamlı 8-bit'i DACXH' e yazılır. Sonra düşük anlamlı 8-bit DACXL' a yazılır. DACXL'a sayısal veri yazıldıktan sonra donanımsal olarak çıkış güncellenir/çıkış gerilimi üretilir (SYNC=0 için).

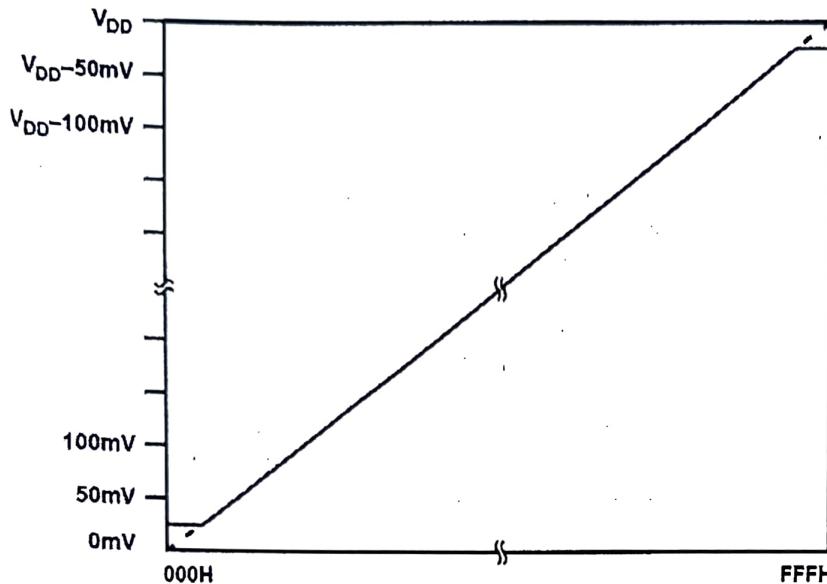
8-bit veri yoluna ve RAM/Kaydedici alınana sahip olan ADUC841 sayısal işlemcisinde, 12-bitlik bir sayı 8-bit veri paketleri halinde ancak iki kerede bir yerden diğerine taşınabilir.

12-bitlik bir veri Yüksek anlamlı 8-bit ve Düşük anlamlı 8-bit olmak üzere toplam 16-bit veri alan ile temsil edilebilir.

Düşük Anlamalı 8-Bit:	D7	D6	D5	D4	D3	D2	D1	D0
-----------------------	----	----	----	----	----	----	----	----

Yüksek Anlamalı 8-Bit:	x	x	x	x	D11	D10	D9	D8
------------------------	---	---	---	---	-----	-----	----	----

DAC KAZANCI: ADUC841 mimarisinde bulunan 12-bit DAC0 ve DAC1'e ait kazanç eğrisi aşağıda verilmiştir. Deney için kullanılan ADUC841 geliştirme kartlarında harici gerilim referansı $V_{ref} = V_{DD} = 5 \text{ Volt}$ kullanılmaktadır.



Şekil 3 DAC0 ve DAC1 giriş-çıkış karakteristiği

Yukarıda verilmiş olan ideal karakteristik göz önünde bulundurulur ise DAC kazancı K_{DAC} olmak üzere

harici gerilim referansı $V_{ref} = V_{DD} = 5 \text{ Volt}$ seçilmiş olsun. Bunu anlamı, DAC0 veya DAC1 çıkışından $V_{çıkış} = 0 - 5 \text{ Volt}$ arası gerilim üretilebilir.

DAC0 veya DAC1 ile üretilebilecek en küçük gerilim veya çözünürlük

$$K_{DAC} = \frac{V_{DD} = V_{ref}}{0x0FFF} = \frac{5}{4095} \rightarrow K_{DAC} = 1.221 \text{ mV} \quad \text{olarak hesap edilir.}$$

$V_{istenen}$ istenilen çıkış gerilimini üretmek için DAC'a yüklenmesi gereken sayı

$$N = \frac{V_{istenen}}{K_{DAC}} \quad \text{hesap edilir.}$$

ÖRNEK: DAC0 çıkışında 2.5 Volt işaret üretmek için DAC0H ve DAC0L yüklenmesi gereken sayısal değerleri onaltılık tabanda (HEX) hesap ediniz.

$$N = \frac{V_{istenen}}{K_{DAC}} = \frac{2500}{1.221} = 2047.5 \quad \text{yaklaşık olarak } N = 2048 \text{ Alınabilir.}$$

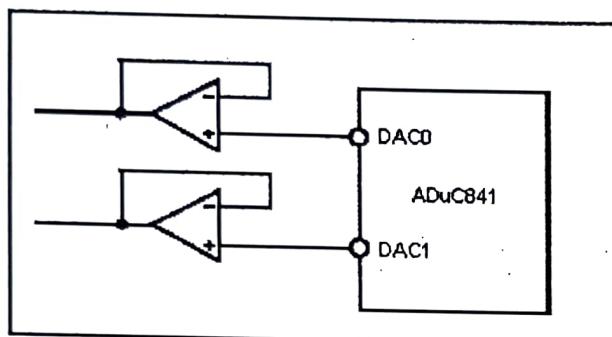
$$\text{Onaltılık tabanda (Hex)} \quad N = 2048 = 0x0800$$

Sırası ile DCA0'a ait özel fonksiyon kaydedicilerine

DAC0H=0x08 ve DAC0L=0x00 Hex sayıları yüklenmelidir.

DAC0 ve DAC1 çıkış işaretleri Şekil 1'de gösterildiği gibi Aduc841 mimarisinde bulunan dahili tamponlar (buffer) üzerinden çıkışa iletilir. Şekil 3'de verilmiş olan DAC0 ve DAC1 giriş-çıkış karakteristiğinden görüleceği üzere, çıkış tamponunun maksimum ve minimuma yakın değerleri için doğrusallık bozulmaktadır.

Bu nedenle hassas tasarımlarda veya isteğe bağlı olarak, CFG841 kaydedicisinde bulunan DBUF biti lojik "1" yapılarak dahili tamponlar devre dışı bırakılır. Bu durumda DAC çıkışlarına Şekil 4'de gösterildiği gibi harici tamponlar bağlanmalıdır.

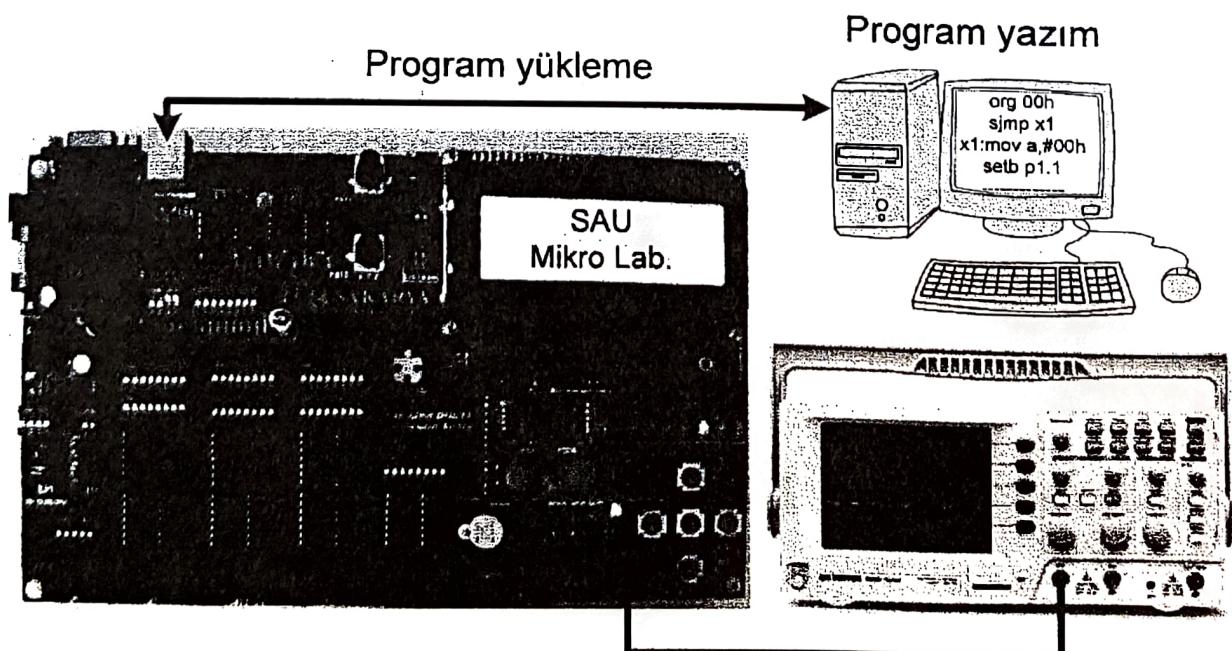


Şekil 4 DAC çıkışlarına bağlanmış harici tamponlar (harici gerilim izleyiciler).

1. Deney

Bu deneyin amacı AduC841 mikrokontrolör mimarisi ve adresleme yöntemlerini kullanarak modeli bilinen işaretler için sinyal üretici yapmaktadır.

Endüstride ve haberleşme cihazlarında yaygın olarak kullanılan DTMF (Dual Tone Multi Frequency, Çift Tonlu Çoklu Frekanslı) işaretleri üretilecektir. DAC0 çıkışında üretilen DTMF işaretleri Şekil 5'de gösterildiği gibi bir osiloskop yardımcı ile gözlenecek/incelenecektir.



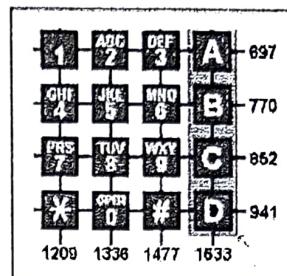
Şekil 5: Deney düzeneği

1.1 DTMF İşaretleri

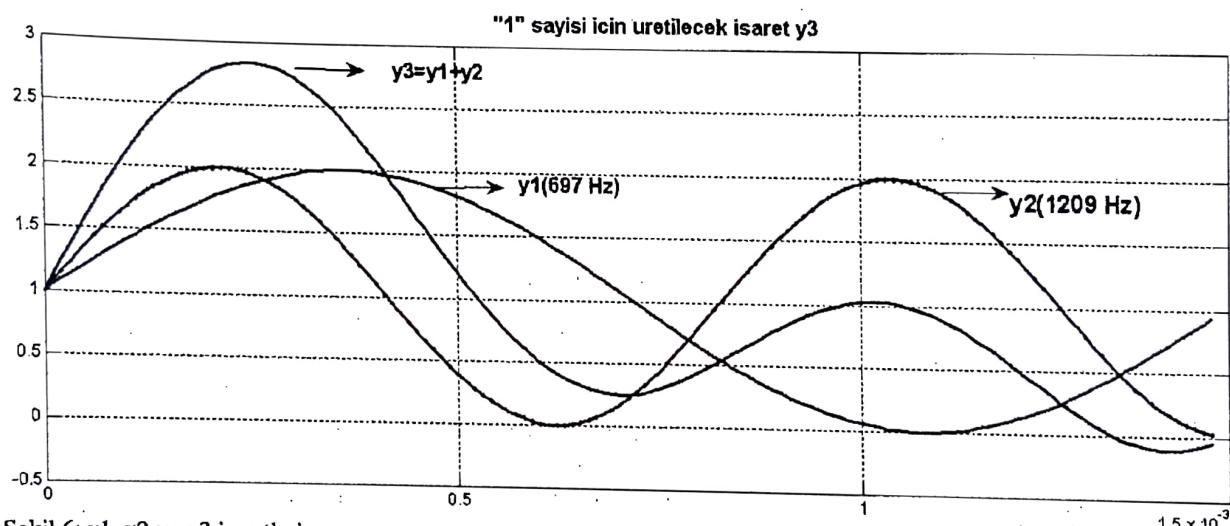
Kısaca ifade etmek gerekir ise DTMF farklı frekans değerine sahip iki sinüs işaretinin toplanması ile elde edilen, Bell telefon laboratuvarlarında geliştirilmiş bir kodlama sistemidir. Tablo 2'de bir telefon tuş takımında bulunan her bir tuş için DTMF işaretini oluşturan sinüs işaretinin frekans değeri belirtilmiştir. Aşağıda tuş takımına ait sayı ve karşılık gelen DTMF frekans bileşenleri verilmektedir.

Tablo 2: TDMF kodları.

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D



Ömek olarak, "1" sayısına karşılık gelen DTMF işaretini üretmek için, $y_1 = \sin(2\pi \cdot 697 \cdot t)$ ve $y_2 = \sin(2\pi \cdot 1209 \cdot t)$ işaretlerinin toplamından oluşan $y_3 = y_1 + y_2$ işaretini DAC0 kullanılarak üretir.



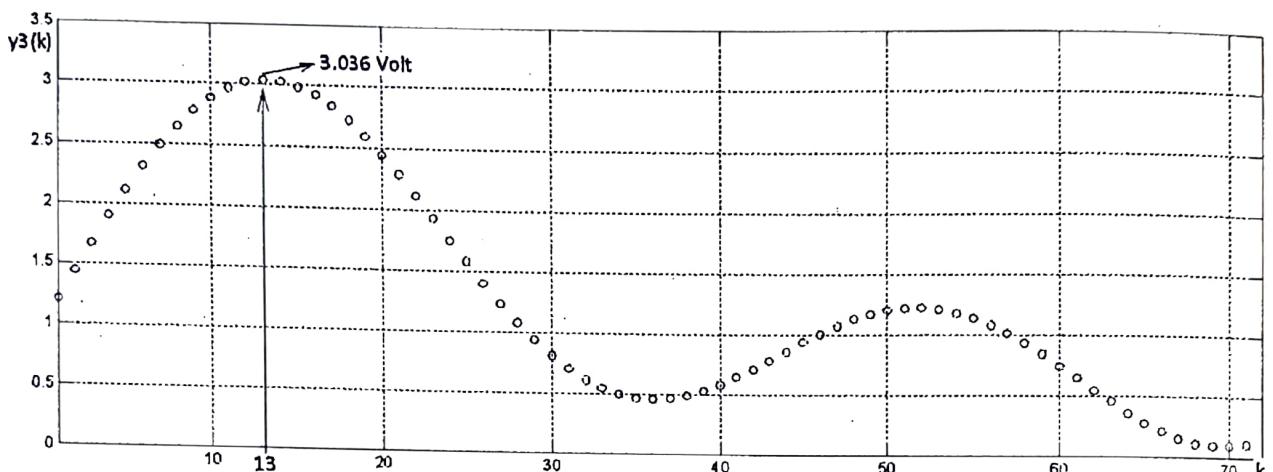
Şekil 6: y_1 , y_2 ve y_3 işaretleri

y_3 işaretinin DAC0 ile sürekli zaman işaret olarak üretilebilmesi için y_3 işaretini önce ADUC841 ile yazılım ile sayısal olarak üretilmelidir. $y_1 = \sin(2\pi \cdot 697 \cdot t)$ ve $y_2 = \sin(2\pi \cdot 1209 \cdot t)$ işaretlerinin T örneklemme zamanı üzere, $t = kT$, $k = 0, 1, 2, 3, \dots, N$ anları için trigonometrik hesapların yapılmış ve toplanması ciddi hesap yükü getirmektedir.

Bunun yerine, Tablo 2 de verilmiş olan işaretler belirgin ve sabit genlik/frekans olduğundan y_1 ve y_2 işaretlerinin toplamından oluşan y_3 işaretin çevrimdışı olarak DAC0 giriş-cıkış karakteristiği kullanılarak hesaplanır ve her bir k. anı değere karşılık gelen onaltılık tabandaki kodlardan oluşan tablo hazırlanır.

1.2 Tablo oluşturulması

y3 işaretinin tablo oluşturulması: y3 işaretinin şekil 7 de verilmiş olduğu gibi $T=20\mu s$ periyodu ile örneklenir.



Şekil 7: $T=20 \mu s$ ile örneklenmiş y3 işaretti

DAC0 kazancı kullanılarak örneklenmiş olan her bir anı değer için onaltılık tabanda (Hex) kod üretilir.

Aşağıda, sadece örneklemme anı $k=13$ 'e karşılık 3.036 Volt noktası için kod üretilecek olup aynı işlemlerin tüm $N=71$ adet örnek için tekrarlanması gerekmektedir. Bunun için bir matematik yardımcı programdan faydalananabilir ve basit bir yazılım ile tablo 3 te verildiği gibi oluşturulabilir.

Ömek: $V_{ref}=5$ Volt için DAC0 kazancı $K_{DAC} = 1.221 mV$ idi.

$$N = \frac{V_{istenen}}{K_{DAC}} = \frac{3036}{1.221} = 2486.48 \quad \text{yaklaşık olarak } N = 2486 \quad \text{alınır.}$$

$$N = 2486 = 0x09B6 \quad (\text{on altılık taban})$$

Açıklama: ADUC841 de DAC0 veya DAC1 kullanılarak 3.036 volt üretilmek isteniyor ise, $V_{ref}=5$ Volt için, DAC0 veya DAC1'e 2486 sayısı yüklenmelidir.

Şekil 7 de verilmiş olan y3 işaretini kullanılarak deney için üretilmiş olan anı değerler ve Hex kodlar tablo halinde aşağıda oluşturulmuştur.

Tablo 3 T=20us ile örneklenmiş y3(k)'ya ait Hex Kodlar

k	y3(k)	Hex Kod	k	y3(k)	Hex Kod
1	1.200000	03d6	36	0.453491	0173
2	1.438819	049a	37	0.461538	017a
3	1.673481	055b	38	0.486420	018e
4	1.899914	0614	39	0.526219	01af
5	2.114216	06c4	40	0.578687	01da
6	2.312731	0766	41	0.641305	020d
7	2.492131	07f9	42	0.711351	0247
8	2.649474	087a	43	0.785976	0284
9	2.782269	08e7	44	0.862274	02c2
10	2.888523	093e	45	0.937364	0300
11	2.966778	097e	46	1.008458	033a
12	3.016137	09a6	47	1.072940	036f
13	3.036278	09b6	48	1.128430	039c
14	3.027458	09af	49	1.172846	03c1
15	2.990495	0991	50	1.204459	03da
16	2.926751	095d	51	1.221938	03e9
17	2.838092	0914	52	1.224386	03eb
18	2.726847	08b9	53	1.211361	03e0
19	2.595747	084e	54	1.182894	03c9
20	2.447862	07d5	55	1.139489	03a5
21	2.286534	0751	56	1.082109	0376
22	2.115295	06c4	57	1.012159	033d
23	1.937789	0633	58	0.931449	02fb
24	1.757693	05a0	59	0.842154	02b2
25	1.578631	050d	60	0.746760	0264
26	1.404099	047e	61	0.648002	0213
27	1.237385	03f5	62	0.548799	01c1
28	1.081501	0376	63	0.452183	0172
29	0.939121	0301	64	0.361219	0128
30	0.812526	0299	65	0.278931	0e4
31	0.703555	0240	66	0.208227	0ab
32	0.613578	01f7	67	0.151819	07c
33	0.543468	01bd	68	0.112157	05c
34	0.493592	0194	69	0.091361	04b
35	0.463810	017c	70	0.091165	04b
			71	0.112866	05c

Tablo 3 te y3 işaretin verilmiş olan veriler kullanılarak "DTMF_1" isimli veri tabanı oluşturulmuş ve "9" sayısı için üretilmiş DTMF_9 veri tabanı ile beraber aşağıda verilmiştir. y3 işaretin verilen bu veri tabanları deney esnasında ilgili yazılıma doğrudan kopyala yapıştır yapılabilir. Bu veriler T=20us de bir DAC0 yazılacaktır. Program ilk yüklendiğinde/resetlendiğinde DTMF_1 işaretini üretilecek. bt2 butonuna basılıncaya DTMF_9, bt2 butonuna tekrar basılır ise DTMF_1 işaretini üretilecek olup bu işlem periyodik olarak devam edecektir. Her bt2 butonuna basılıncaya işaret diğeri ile değişecektir.

NOT: ADUC841 ADC ve DAC üniteleri unipolar dır, tek kutupludur. Yani sıfır "0" volt ile +Vref arasındaki işaretleri çevirebilirler. Negatif işaretler için dönüşüm yapamazlar. y1 ve y2 işaretleri -1 volt ve +1 volt arasında sintez olarak değişmektedir. Bu yüzden y3 işaretin 1.2 Volt ile toplanmış ve böylece y3 işaretin pozitif bölgeye kaydırılmıştır.

Db 0x8E

DTMF_1: ;"1" sayısı için veri tabanı

Dw 0x3d6, 0x49a, 0x55a, 0x614, 0x6c3, 0x766, 0x7f9, 0x879, 0x8e6, 0x93d, 0x97d, 0x9a6, 0x9b6, 0x9af
Dw 0x991, 0x95d, 0x914, 0x8b9, 0x84d, 0x7d4, 0x750, 0x6c4, 0x633, 0x59f, 0x50c, 0x47d, 0x3f5, 0x375
Dw 0x301, 0x299, 0x240, 0x1f6, 0x1bd, 0x194, 0x17b, 0x173, 0x17a, 0x18e, 0x1ae, 0x1d9, 0x20d, 0x246
Dw 0x283, 0x2c2, 0x2ff, 0x339, 0x36e, 0x39c, 0x3c0, 0x3da, 0x3e8, 0x3ea, 0x3e0, 0x3c8, 0x3a5, 0x376
Dw 0x33c, 0x2fa, 0x2b1, 0x263, 0x212, 0x1c1, 0x172, 0x127, 0xe4, 0xaa, 0x7c, 0x5b, 0x4a, 0x4a
Dw 0x3d6

DTMF_9_veri: ;"9" sayısı için toplam veri adedi 106=0x74

Db 0x74

DTMF_9: ;"9" sayısı için veri tabanı

Dw 0x3d6, 0x4c5, 0x5ad, 0x68a, 0x754, 0x808, 0x8a0, 0x919, 0x971, 0x9a6, 0x9b7, 0x9a5, 0x971, 0x91e
Dw 0x8ae, 0x827, 0x78d, 0x6e5, 0x634, 0x57f, 0x4cd, 0x422, 0x384, 0x2f5, 0x27a, 0x215, 0x1c7, 0x193
Dw 0x177, 0x172, 0x184, 0x1a8, 0x1dc, 0x21d, 0x265, 0x2b1, 0x2fd, 0x343, 0x381, 0x3b3, 0x3d6, 0x3e8
Dw 0x3e8, 0x3d5, 0x3af, 0x377, 0x331, 0x2dd, 0x281, 0x21f, 0x1bc, 0x15c, 0x105, 0xba, 0x7f, 0x59
Dw 0x3d6, 0x4c5

Deney1 ön çalışma: Bu kısım ile ilgili yazılım kodları deney esnasında yazılmacaktır
(yazılım kodları verilmeyecektir).

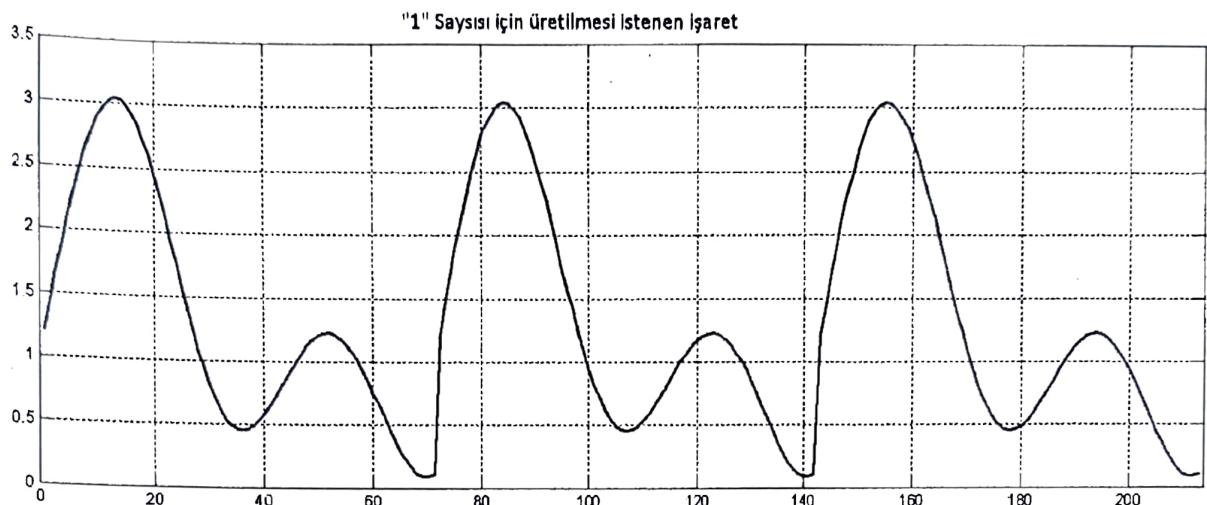
- 1- DAC0 Vref harici ve 12-bit mod seçilir.
- 2- Timer0 20us ve kesmeli olarak ayarlanır.
- 3- sin_Y3HB değişkeni 0x24 adresine ataması yapılır, Yüksek anlamlı 8-bit.
- 4- sin_Y3LB değişkeni 0x25 adresine ataması yapılır. Düşük anlamlı 8-bit.
- 5- sin_Y3LB ve sin_Y3HB adresleri 16-sayıci olarak kullanılacaktır. Her Timer0 kesmesinde, Kesme Hizmet Programında bu 16-bit sayıci "1" artırılacaktır. 16-bit sayıci değerleri sırası ile DAC0H ve DAC0L 'a yazılacaktır.
- 6- DAC0 ile üretilen işaret gözlenecek ve frekansı ölçülecektir.
- 7- Üretilen işaretin frekansını iki kat artırmak için ne yapılma lıdır ? Yazılımda gerekli değişikliği yaparak frekansı ölçünüz.

Deney2: DTMF işaretlerinin üretilmesi (Yazılım kodları çalışma amaçlı olarak aşağıda verilmiştir.)

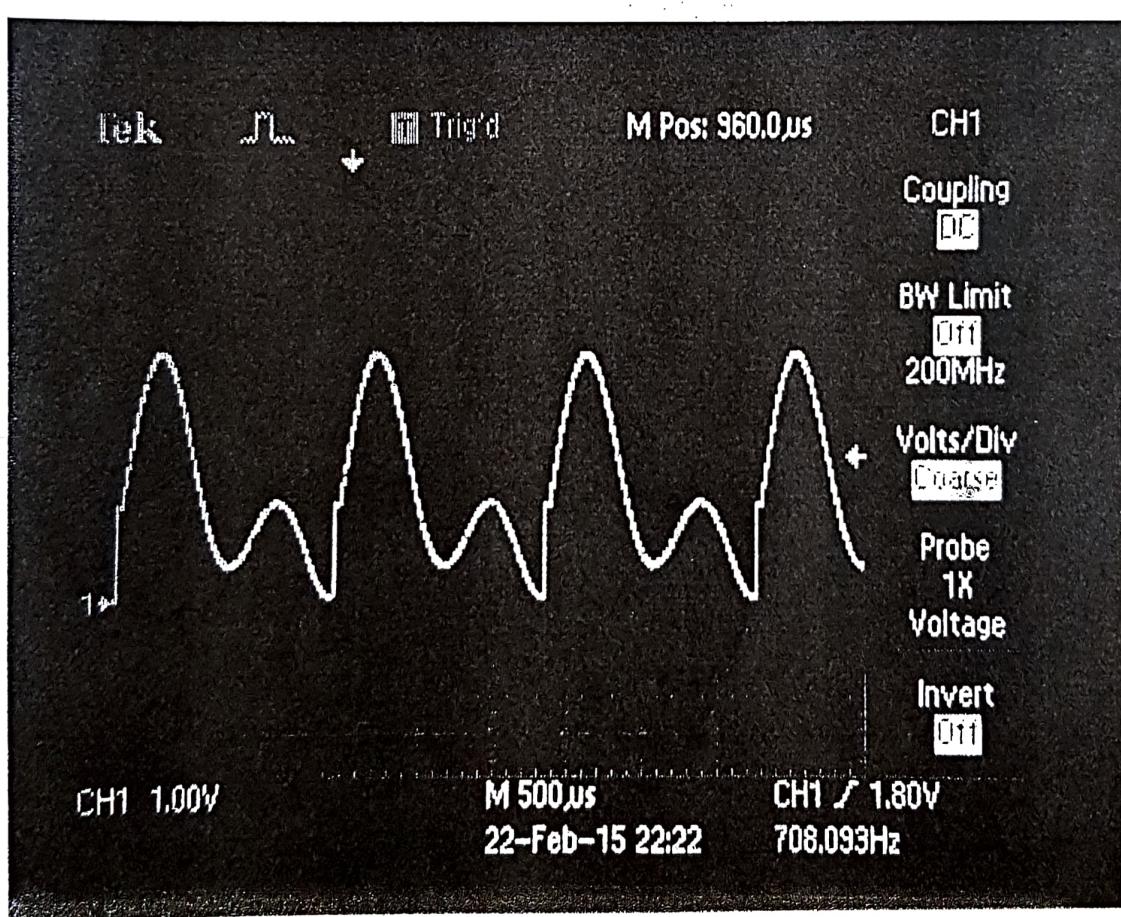
Program basamakları:

- 1- Değişkenler tanımlanır ve RAM adresleri belirlenir.
- 2- DAC0 Vref harici ve 12-bit mod seçilir.
- 3-Timer0 20us ve kesmeli olarak ayarlanır.
- 4- "1" ve "9" sayılarının veri uzunlukları kod hafızadan okunur.
- 5- Timer0 kesmesi yetkilendirilir ve çalıştırılır.
- 6- bt2 tuşuna basılıp basılmadığı yoklaması negatif darbe algılama mantığı ile yapılır.
- 7- bt2 buton yoklama esnasında oluşabilecek transient işaretler için "bekleme" yapılmaktadır. Bekleme gecikmesi 21*(255*20us) süresi kadardır.
- 8- Timer0 her 20us de bir kesme üretir. Kesme hizmet programında buton sayacı=0 ise "1" sayısına buton sayacı=1 ise "9" sayısına karşılık gelen kodlar ilgili veri tabanından okunmaktadır.
- 9- Veri tabanından önce Yüksek anlamlı 8-bit sonra Düşük anlamlı 8-bit okunmaktadır.
- 10-Bu veriler sırası ile DAC0H ve DAC0L yazılır.
- 11-Bu işlemler periyodik olarak devam eder.

Deney2 sonucuna ait osiloskop görüntüsü ve yazılamaya ait kodlar aşağıda verilmiştir.



Aşağıda ise ADUC841 ile DAC0 kullanılarak "1" sayısına karşılık üretilen gerçek zaman y3 işaretine ait osiloskop görüntüsü verilmiştir.



ADUC841 ile DAC0 kullanılarak "1" sayısına karşılık üretilen gerçek zaman y3 işaretini Deney2 ile İlgili Kodlar aşağıda verilmiştir.

;----22/OCAK/2015

#include <aduc841.h>

;--- Yazılımda kullanılan RAM adres ve tanımlamaları

```
DTMF1      equ 20H ;"1" sayısı için Offset adres COD hizizada okumak için
DTMF9      equ 21h ;"9" sayısı için Offset adres COD hizizada okumak için

DTMF_1veri   equ 22H ;"1" sayısı için veri taban uzunluğu, Kod hizizada okunacak.
DTMF_9veri   equ 23H ;"9" sayısı için veri taban uzunluğu, Kod hizizada okunacak.

sin_Y3HB    equ 24h ;Cikisa (DAC0) yazılacak verinin Yuksek anlamlı 8-bit
sin_Y3LB    equ 25h ;Cikisa (DAC0) yazılacak verinin Düsük anlamlı 8-bit

buton_say   equ 26h
buton       equ p2.1 ;bt2 butonu secildi ve buton olarak isimlendirildi.

gecikme     equ 27h
gecikme2    equ 28h
```

=====

CSEG

```
ORG 00h
SJMP MAIN
```

```
ORG 0BH
LJMP TIMER0_KESME
```

=====

MAIN:

```
MOV SP,#070H
```

;--- DAC0 koşullandırılıyor -----

```
MOV DACCON,#0111111B ; Vref=5Volt harici
MOV DAC0H,#00H
MOV DAC0L,#00H

MOV TMOD,#00000010B ; timer0 otomatik yüklemeli
MOV TL0,#024H
MOV TH0,#024H ; 20usn

SETB IE.7
SETB IE.1 ;timer0 kesme aktif

mov DTMF1,#00h ;"1" sayısı offset adres sıfırlandı.
mov DTMF9,#00h ;"9" sayısı offset adres sıfırlandı.

mov buton_say,#00h ;isaretler arası değişim için kullanılacak
```

setb buton

;---- "1" sayisi icin tablo uzunlugu Kod hafizadan "DTMF_1_veri" adresinden okunuyor.

```
mov    dptr,#DTMF_1_veri      ;"1" sayisi icin KOD hafizada veri adedi
mov    a,#00h                  ;"1" sayisi icin KOD hafizada veri adedi
movc   a,@a+DPTR
mov    DTMF_1veri,a          ;"1" sayisi icin tablo uzulugu okundu.
```

;---- "9" sayisi icin tablo uzunlugu Kod hafizadan "DTMF_9_veri" adresinden okunuyor.

```
mov    dptr,#DTMF_9_veri      ;"9" sayisi icin KOD hafizada baslangic adres
mov    a,#00h                  ;"9" sayisi icin KOD hafizada offset adres.
movc   a,@a+DPTR
mov    DTMF_9veri,a          ;"9" sayisi icin tablo uzulugu okundu.
```

SETB TCON.4 ; timer0 20us kesme uretebilmesi icin calistirildi.

bekle_on: ;Buton icin -- --- negatif pulse denetimi yapiliyor.

```
jb     p2.1,bekle_on
acall bekle
```

bekle_off:

```
jnb    buton,bekle_on
acall bekle
inc    buton_say           ; gecici parazitleri olemek icin "bekle" yazılımı ile gecikme üretilir.
mov    a,buton_say
cjne   a,#02h,bekle_on
mov    buton_say,#00h
sjmp   bekle_on
```

bekle: ;0.1071 saniyelik bekleme olusuyor. 21*(255*20us)= 0.1071 saniye
 mov gecikme,#01h ;her 20us lik kesmede gecikme +1 oluyor. 255 kere saydimi 00' a dusur .
 mov gecikme2,#015h ;Bu olay 15 kere tekrar ediliyor.

zmn_as:

```
mov    a.gecikme
cjne   a,#00h,zmn_as
mov    gecikme,#01h
djnz   gecikme2,zmn_as
ret
```

TIMER0_KESME:

```
mov    a,buton_say
cjne   a,#00h,DTMF9_
acall  al_DTMF_1
sjmp   ada_devam
```

DTMF9_: Acal l al_DTMF_9

ada_devam: inc gecikme

x1:

```
mov    dac0H,sin_Y3HB      ; DAC0H Yuksek anlamlı 8-bit yaz
mov    dac0L,sin_Y3LB      ; DAC0L Dusuk anlamlı 8-bit yaz
```

RETI

al_DTMF_1:

```

    mov  dptr,#DTMF_1      ;"1" sayisi icin KOD hafizada veri tabaninda ilk veri baslangic adresi
    mov  a,DTMF1           ;"1" sayisi icin KOD hafizada baslangictan sonra kacinci verinin okunac
    movc a,@a+DPTR
    mov  sin_Y3HB,a        ; verini yüksek anlamlı 8-bitini oku
    inc  DTMF1
    mov  a,DTMF1
    movc a,@a+dptr
    mov  sin_Y3LB,a        ; verini düşük anlamlı 8-bitini oku

    inc  DTMF1
    mov  a,DTMF1
    clr  c
    subb a,DTMF_1veri     ;"1" sayisi icin tablo sonuna gelin dimi ?
    jc   ad_DTMF1         ; Hayir ise ad_DTMF1' e git ve devam et
    mov  DTMF1,#00h         ; Tablo basindan basla.

```

ad_DTMF1:

RET

al_DTMF_9:

```

    mov  dptr,#DTMF_9      ;"9" sayisi icin KOD hafizada veri tabaninda ilk veri baslangic adresi
    mov  a,DTMF9            ;"9" sayisi icin KOD hafizada baslangictan sonra kacinci verinin ounacagi,
    movc a,@a+DPTR
    mov  sin_Y3HB,a        ; verini yüksek anlamlı 8-bitini oku
    inc  DTMF9
    mov  a,DTMF9
    movc a,@a+dptr
    mov  sin_Y3LB,a        ; verini düşük anlamlı 8-bitini oku

    inc  DTMF9
    mov  a,DTMF9
    clr  c
    subb a,DTMF_9veri      ; "9" sayisi icin tablo sonuna gelin dimi ?
    jc   ad_DTMF9          ; Hayir ise ad_DTMF9' e git ve devam et
    mov  DTMF9,#00h          ; Tablo basindan basla.

```

ad_DTMF9:

RET

;örnekleme araligi 20 mikrosaniye

DTMF_1_veri: ;"1" sayisi için, toplam veri adedi 142(0x8E) dir.
Db 0x8E

DTMF_1: ;"1" sayisi için veri tabani

Dw 0x3d6, 0x49a, 0x55a, 0x614, 0x6c3, 0x766, 0x7f9, 0x879, 0x8e6, 0x93d, 0x97d, 0x9a6, 0x9b6, 0x9af
Dw 0x991, 0x95d, 0x914, 0x8b9, 0x84d, 0x7d4, 0x750, 0x6c4, 0x633, 0x59f, 0x50c, 0x47d, 0x3f5, 0x375
Dw 0x301, 0x299, 0x240, 0x1f6, 0x1bd, 0x194, 0x17b, 0x173, 0x17a, 0x18e, 0x1ae, 0x1d9, 0x20d, 0x246
Dw 0x283, 0x2c2, 0x2ff, 0x339, 0x36e, 0x39c, 0x3c0, 0x3da, 0x3e8, 0x3ea, 0x3e0, 0x3c8, 0x3a5, 0x376
Dw 0x33c, 0x2fa, 0x2b1, 0x263, 0x212, 0x1c1, 0x172, 0x127, 0xe4, 0xaa, 0x7c, 0x5b, 0x4a, 0x4a
Dw 0x3d6

SAU / Müh.Fak./ EEM/ Sayısal Analog Dönüşürtücü

DTMF_9_veri: ;"9" sayısı için toplam veri adedi 106 (0x74 Hex) dir.
Db 0x74

DTMF_9: ;"9" sayısı için veri tabanı
Dw 0x3d6, 0x4c5, 0x5ad, 0x68a, 0x754, 0x808, 0x8a0, 0x919, 0x971, 0x9a6, 0x9b7, 0x9a5, 0x971, 0x91e
Dw 0x8ae, 0x827, 0x78d, 0x6e5, 0x634, 0x5f7, 0x4cd, 0x422, 0x384, 0x2f5, 0x27a, 0x215, 0x1c7, 0x193
Dw 0x177, 0x172, 0x184, 0x1a8, 0x1dc, 0x21d, 0x265, 0x2b1, 0x2fd, 0x343, 0x381, 0x3b3, 0x3d6, 0x3e8
Dw 0x3e8, 0x3d5, 0x3af, 0x377, 0x331, 0x2dd, 0x281, 0x21f, 0x1bc, 0x15c, 0x105, 0xba, 0x7f, 0x59
Dw 0x3d6, 0x4c5

END

```
db 48h,8Bh,0B6h,95h,0CEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h
db 0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh
db 0ACh,3hD,75h,0Dh,0c2h,70h,15h,0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh
db 0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h
db 0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h
db 2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h,0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h
db 9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h
db 0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h
db 0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h,0BFh,2Eh,6Bh,42h,90h,51h,8Ah
db 48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h
db 0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh
db 0ACh,3hD,75h,0Dh,0c2h,70h,15h,0BFh,2Eh,6Bh,42h,90h,51h,8Ah,48h,8Bh,0B6h,9Eh
db 0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h,15h,0BFh,2Eh,6Bh,42h,90h,51h,8Ah
db 48h,8Bh,0B6h,9Eh,0C5h,0BEh,65h,97h,0C8h,0B2h,2Fh,0ACh,3hD,75h,0Dh,0c2h,70h
```

Deney 5: Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

Amaç

Bu deneyin amacı; Sürekli zaman bir fonksiyonun/transfer fonksiyonun verilen örnekleme zamanı T'ye göre ayrıklaştırılması, matematiksel işlemlerin makine dilinde mikrokontrolör ile yazılımsal olarak gerçekleştirilmesi ile fonksiyona/transfer fonksiyonuna ait çıkış işaretinin sayısal olarak elde edilmesi ve Aduc841 mimarisinde bulunan **PWM** (Pulse Width Modulation, darbe genişlik modülasyonu) modülünü kullanma becerisinin kazandırılmasıdır.

Sürekli Zaman İşaretin Ayrık Zamanda Elde Edilmesi

Sürekli zamanda verilen bir fonksiyonun mikrokontrolör tabanlı koşturulabilmesi için öncelikle bir örnekleme zamanı seçilerek fonksiyonun ayrık zaman transfer fonksiyonun elde edilmesi gerekir. Bu deneye gerçekleştirilecek olan sinüs fonksiyonu " $f(t) = \sin 2\pi ft$ " ait Denklem (1)'de verilmiştir. $f(t) = \sin bt$ 'ye ait Laplace dönüşümü denklem (2)'de ve "T" örnekleme zamanı olacak şekilde ayrık zaman transfer fonksiyonu Denklem (3)'te sırasıyla elde edilmiştir. (Not: İlgili dönüşümler ile ilgili Otomatik Kontrol Notları Syf. 40'ta verilen Tablo'dan incelenebilinir.) Aynık zaman transfer fonksiyonu elde edilen sistemin doğrudan programlama yöntemi ile durum diyagramı elde edilerek, sembolik dilde programlanacaktır. Böylece sürekli zamanda verilen sinüs fonksiyonu sayısal işlemci tabanlı programlanabilir duruma getirilecektir.

$$f(t) = \sin 2\pi ft = \sin \omega t = \sin bt \quad (1)$$

$$L\{f(t)\} = F(s) = \frac{b}{s^2 + b^2} \quad (2)$$

$$Z(F(s))|_r = F(z) = \frac{z \sin(bT)}{z^2 - 2z \cos(bT) + 1} \quad T = \text{örneklemme zamanı} \quad (3)$$

Denklem (3)'te verilen fonksiyon virgülü sayılarla işlem içerebilir. Ondalıklı sayılar, sabit noktalı veya kayan noktalı sayılar (floating point) olmak üzere iki farklı yöntem ile temsil edilebilirler. Bazı mikrokontrolörler kayan noktalı sayılar ile doğrudan işlem yapabilen mimari içermektedirler. Ancak Aduc841 mimarisinde sadece isaretsiz 8-bit çarpma, bölme, toplama ve çıkartma işlemleri için donanım yapısı mevcuttur. Yani Aduc841 ile sadece isaretsiz 8-bit sayılar için çarpma, bölme, toplama ve çıkartma matematiksel işlemleri donanımsal olarak yapılabılır ve ilgili komutlar mevcuttur. Diğer tüm Tam ve Küsurath sayılar için matematiksel işlemler programcı tarafından yazılımsal olarak gerçekleştirilir. Bundan dolayı, deney esnasında küsuratlı sayıların çarpımı için aşağıda anlatılacak olan **Q15 yöntemi** kullanılacaktır. Bölüm 3'te Q15 yöntemi açıklanmıştır.

Q15 Yöntemi

Q15 yöntemi, $0 < F < 1$ arasındaki bir F sayısını (Hex) onaltılık sayı tabanında Q15 formatında temsil edilmesini sağlayan bir yöntemdir. Hex tabanında temsil edilen 16-bit F sayısı artık, ADUC841 ile yazılımsal olarak başka bir 16-bit sayı ile çarpılabilir.

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayırılaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

Bu yöntemde, ondalıklı sayı ile tam sayının çarpım sonucu en yakın tam sayıya yuvarlanır bundan dolayı bu yöntem çok hassas gerektiren işlemler için kullanılmamalıdır.

Q15 yöntemi aşağıda verilen örnekte anlatılmıştır.

Burada, tam sayı Xd ile çarpılacak küsuratlı sayı Yd ve $0 < Yd < 1$ dir.

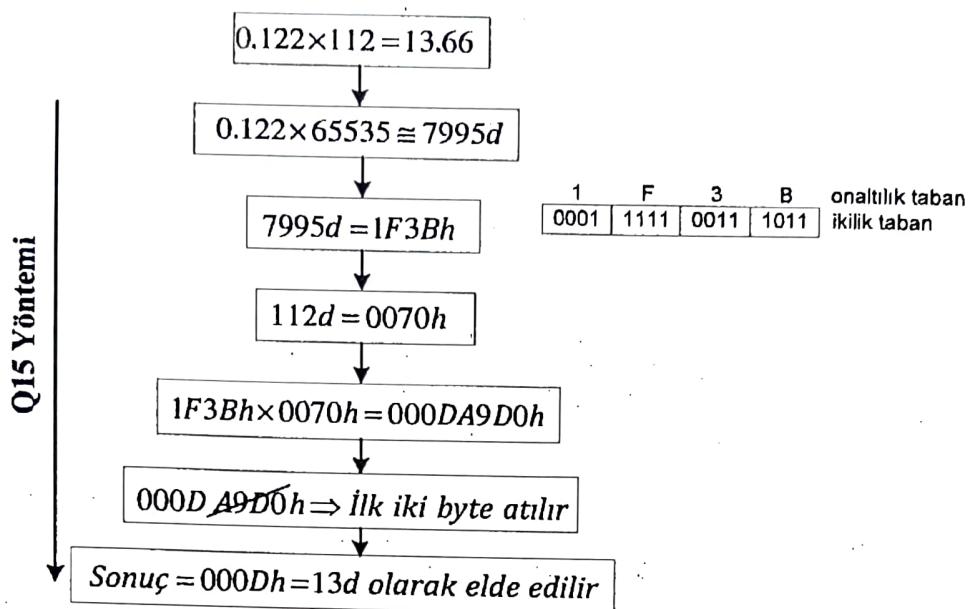
Burada, tam sayı (Xd) ile çarpılacak ondalık sayı (Yd ; $0 < Yd < 1$) ilk olarak 65535 ile çarpılıp ($Yd * 65535 = Zd$) 16'lık (heksadesimal) tabana ($Zd \rightarrow Wh$) dönüştürülür. 16'lık (heksadesimal) tabana dönüştürülen sayı (Wh) ile tam sayının 16'lık tabandaki karşılığı ($Xd=Qh$) çarpılırak ($Wh * Qh = 32\text{ bit}$) 32 bitlik sonuç elde edillir. Elde edilen son değerden *ilk 2 byte atılır*. Geri kalan 2 byte bize virgülü sayı ile çarpmaya veya bölme yaptığımız sayı arasındaki işlem sonucunu tamsayı değeri olarak verir. Aşağıda Şekil 1'de Q15 yöntemi ile ilgili örnek ve örneğe ait akış diyagramı Şekil 1'de verilmiştir.

Örnek: $0.122d \times 112d = 13.66d$ işlemini hex tabanında gerçekleştiriniz.

Çözüm:

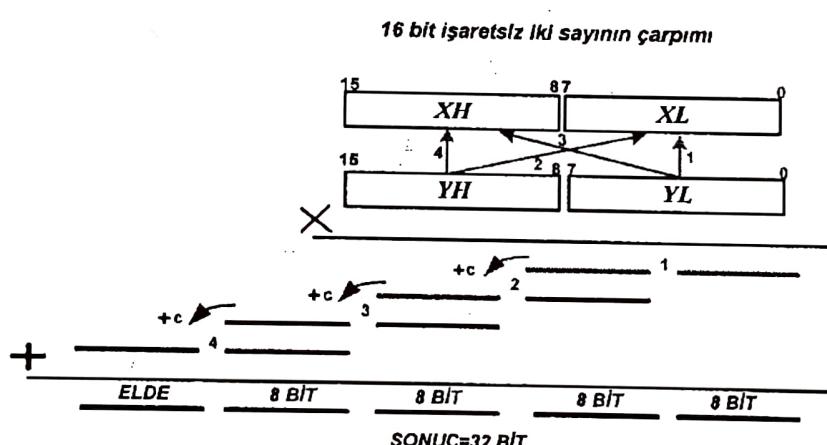
- i. 10 tabanında ondalıklı sayı ile 65535 sayısı çarpılır,
 $0.122d \times 65535d = 7995.27d \approx 7995d$
 - ii. Elde edilen sonuç 16 tabanına çevrilir, elde edilen bu sayı 0.125d sayısını temsil eder,
 $7995d = 1F3Bh$
 - iii. 10 tabanındaki çarpılacak olan tam sayı 16 tabanına çevrilir,
 $112d = 0070 h$
 - iv. (Adım (ii) ve adım (iii)'te elde edilen 16 tabanındaki değerler çarpılır,
 $1F3Bh \times 0070h = 000DA9D0h$
 - v. Adım (iv)'te elde edilen 16 tabanındaki sayının en anlamlı 2 byte'sı Q15 yönteminin sonucu olarak elde edilir,
 $000Dh = 13d$
- Görüldüğü gibi ondalık değerdeki çarpım sonucu $13.66d$ değeri; Q15 formatında yaklaşık $13d$ olarak elde edilmiştir.

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayırılaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünlün Kullanılması



Şekil 1. Q15 Yöntemine ait akış diyagramı

16 bit işaretetsiz iki sayının çarpımı 32 bittir, çarpımın nasıl yapıldığı Şekil 2'de gösterilmiştir.

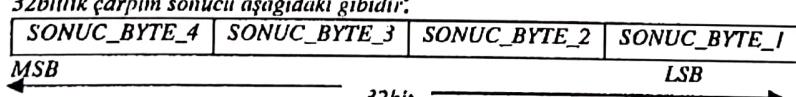


Şekil 2. 16 bit işaretetsiz iki sayının çarpımı

Şekil 2'de temsilen gösterilen 16-Bit x 16-Bit işaretetsiz iki adet sayının çarpımını yapan alt program aşağıda verilmiştir.

Bu programın çıktısında eğer Q15 yönteminin sonucu elde edilecek ise program sonucunda elde edilen sonuç byte'larının (SONUÇ_BYTE_4/3/2/1) **en anlamlı 2 byte** yani (SONUÇ_BYTE_4/3) alınır. Bu alt program ön çalışma ve deney için doğrudan kullanılacaktır.

32bitlik çarpım sonucu aşağıdaki gibidir:



CARP_SAYI2xSAYI1:

; 16-Bit x 16-Bit işaretsiz sayı çarpım alt programı
;çarpım sonucu 32 bittir
;**BANK 0** seçilidir.
; giriş: r1, r0 = çarpan X (SAYI2_H,SAYI2_L)
; r3, r2 = çarpan Y (SAYI1_H,SAYI1_L)
; çarpım: r3, r2, r1, r0 = X x Y= SAYI2xSAYI1=SONUÇ_BYTE_4/3/2/1

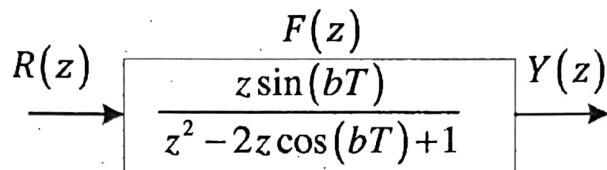
```
push B
push dpl
mov a, r0
mov b, r2
mul ab      ; XL x YL (Bkz. Şekil 2 → 1 nolu çarpım)
push acc    ; Sonuc'un low byte kısmı stack'e atılır
push b      ; Sonuc'un high byte kısmı stack'e atılır
mov a, r0
mov b, r3
mul ab      ; XL x YH çarpımı (Bkz. Şekil 2 → 2 nolu çarpım)
pop 00H
add a, r0
mov r0, a
clr a
addc a, b
mov dpl, a
mov a, r2
mov b, r1
mul ab      ; XH x YL çarpımı (Bkz. Şekil 2 → 3 nolu çarpım)
add a, r0
mov r0, a
mov a, dpl
addc a, b
mov dpl, a
clr a
addc a, #00H
push acc    ; eldeyi stack'te sakla
mov a, r3
mov b, r1
mul ab      ; XH x YH çarpımı (Bkz. Şekil 2 → 4 nolu çarpım)
add a, dpl
mov r2, a
pop acc     ; elde'yi al
addc a, b
mov r3, a
mov r1,00H
pop 00H      ; Sonuc'un low byte'ını cek
pop dpl
pop B
```

RET

Doğrudan Programlama Yöntemi ile Fonksiyonun Gerçekleştirilmesi

Bölüm 2'de denklem (3)'te sinüs fonksiyonu $f(t) = \sin(bt)$ 'ye ait elde edilmiş olan $F(z)$ ayrık zaman transfer fonksiyonu örneklemme zamanı $T = 0.1\text{ms}$ periyodu ile koşturulacaktır.

Çıkış işaretini $Y(z)$ ve giriş işaretini $R(z)$ olmak üzere,



$$F(z) = \frac{Y(z)}{R(z)} = \frac{z \sin(bT)}{z^2 - 2z \cos(bT) + 1} = \frac{zA}{z^2 - 2zC + 1} \quad (4)$$

$$F(z) = \frac{Y(z)}{R(z)} = \frac{zA}{z^2 - 2zC + 1} \cdot \frac{z^{-2}}{z^{-2}} \cdot \frac{X(z)}{X(z)} = \frac{AX(z)z^{-1}}{X(z) - 2CX(z)z^{-1} + X(z)z^{-2}} \quad (5)$$

$$Y(z) = AX(z)z^{-1} \quad (6)$$

$$R(z) = X(z) - 2CX(z)z^{-1} + X(z)z^{-2} \quad (7)$$

$$X(z) = R(z) + 2CX(z)z^{-1} - X(z)z^{-2} \quad (8)$$

elde edilir. Denklem (6) ve Denklem (8) kullanılarak ayrık sinüs fonksiyonuna ait doğrudan programlama yöntemi işaret akış diyagramı aşağıda Şekil 3'te gösterilmiştir.

Değişken tanımlamaları:

$$GIRIS = R(z)$$

$$CIKIS = Y(z)$$

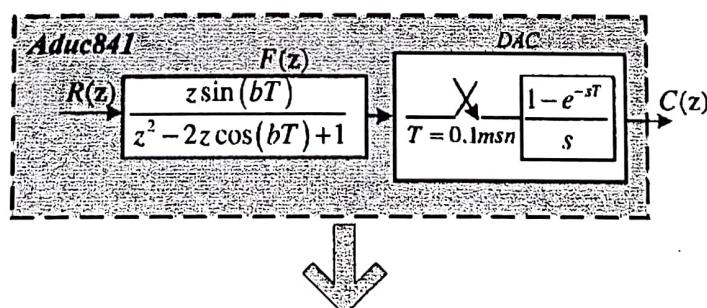
$$A = \sin(bT); b = 2 * pi * f$$

$$C = \cos(bT);$$

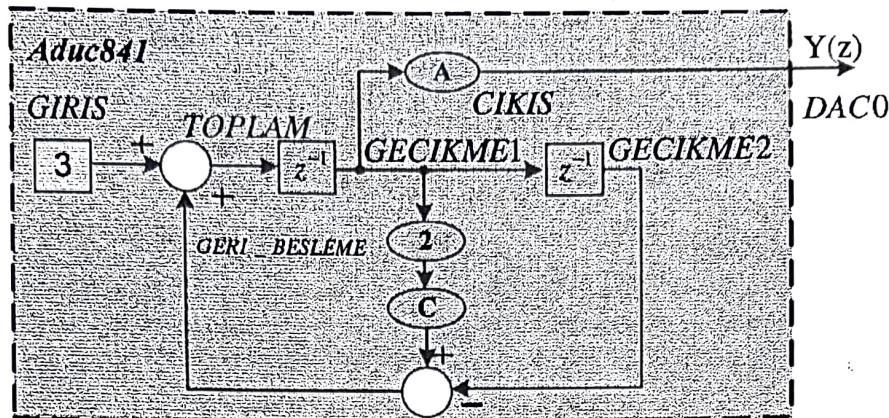
$$GECIKME 1 = X(z)z^{-1}$$

$$GECIKME 2 = X(z)z^{-2}$$

$$GERI_BESLEME = 2cX(z)z^{-1} - X(z)z^{-2}; \text{ olmak üzere,}$$



SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması



Şekil 3. Doğrudan programlama yöntemi ile gerçekleştirilmiş ayrık sinüs fonksiyonuna ait işaret akış diyagramı

Üretilen Sinüs işaretinin f=100Hz olacaktır.

$$A = \sin(bT) = \sin(2 * \pi * 100 * 0.001) = 0.0628$$

$$A = 0.0628 \times 65535 = 4115d = 1013h$$

$$C = \cos(bT) = \cos(2 * \pi * 100 * 0.001) = 0.998$$

$$C = 0.998 \times 65535 = 65403d = FF7Bh$$

$$\begin{aligned} z &= 0,0628 \\ &\frac{z^2 - 20,998z + 1}{z^2} \end{aligned}$$

Bilgisayar ortamında sembolik dil kullanılarak yazılmış ayrık sinüs fonksiyonuna ait komut satırları aşağıda verilmiştir.

```

10      GIRIS=3
20      CIKIS=0
30      GERI_BESLEME =0
40      GECIKME1=0
50      GECIKME2 =0 ; GIRIS ,GERI_BESLEME,GECIKME1,GECIKME2 sabitlerine ilk
                     değerler atanır
60      A=1013h
70      C=FF7Bh ; A ve C katsayı değerleri yukarıda hesaplanan değerlere göre
                     atandı.
80      CIKIS = GECIKME1*A
90      GERI_BESLEME = GECIKME1*2*C- GECIKME2
100     TOPLAM= GIRIS + GERI_BESLEME
    
```

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrılaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

110 GECIKME2= GECIKME1

120 GECIKME1=TOPLAM

130 GO TO 60

DACCON Saklayıcısı

ADUC841 mimarisinde bulunan her iki DAC modülünün (DAC0 ve DAC1) kullanıcı isteğine göre koşullandırılmaları **bit adreslenemez** **DACCON** özel fonksiyon kaydedicisi ile yapılmaktadır. Aşağıda tablo halinde her bir bite ait açıklamalar verilmiştir.

DACCON:	MODE	RNG1	RNG0	CLR1	CLR0	SYNC	PD1	PD0
---------	------	------	------	------	------	------	-----	-----

Bit	İsim	Açıklama
7	MODE	“1”, Her iki DAC 8-bit mod ve sayısal veriler DACxL yazılır. “0” ise her iki DAC 12-bit mod çalışır, veriler sırası ile DACxH ve DACxL yazılır.
6	RNG1	“1”, DAC1 için çıkış gerilim aralığı 0V–AVDD, “0” ise 0V–Vref (2.5V) olarak seçilir.
5	RNG0	“1”, DAC0 için çıkış gerilim aralığı 0V–AVDD, “0” ise 0V–Vref (2.5V) olarak seçilir.
4	CLR1	“1”, DAC1 çıkışı DAC1H/L kaydedicilerinde tutulan sayısal veriye karşılık gelen analog değer üretilir. “0” ise DAC1 çıkışı 0V'a çekilir.
3	CLR0	“1”, DAC0 çıkışı DAC0H/L kaydedicilerinde tutulan sayısal veriye karşılık gelen analog değer üretilir. “0” ise DAC0 çıkışı 0V'a çekilir.
2	SYNC	“1”, DACXL (X=0,1) kaydedicisine veri yazıldığı anda DACX çıkışı güncellenir. “0”, ise, her iki DAC senkron çalışır. Bu modda, öncelikle DACXL/H kaydedicilerine veriler yazılır; daha sonra da SYNC biti yazılım ile “1” yapılır ve her iki DAC çıkışı aynı anda güncellenir, çıkış üretir.
1	PD1	“1” DAC1 enerjilendirilir (Power-On). “0” DAC1 enerjisi kesilir (tasarruf için).
0	PD0	“1” DAC0 enerjilendirilir (Power-On). “0” DAC0 enerjisi kesilir (tasarruf için).

Analog işarete dönüştürülecek sayısal işaret, kullanılacak olan DAC1 ve/veya DAC0 için sırası ile DAC1H/L ve/veya DAC0 için ise DAC0H/L özel fonksiyon kaydedicilerine sağa yanaşık olarak yazılır.

DACXH:	x	x	x	x	D11	D10	D9	D8
--------	---	---	---	---	-----	-----	----	----

DACXL:	D7	D6	D5	D4	D3	D2	D1	D0
--------	----	----	----	----	----	----	----	----

NOT: 12-bit sayısal verinin önce yüksek anlamlı 8-bit'i DACXH' e yazılır. Sonra düşük anlamlı 8-bit DACXL' a yazılır. DACXL'a sayısal veri yazıldıktan sonra donanımsal olarak çıkış güncellenir/çıkış gerilimi üretilir (SYNC=0 için).

8-bit veri yoluna ve RAM/Kaydedici alınana sahip olan ADUC841 sayısal işlemcisinde, 12-bitlik bir sayı 8-bit veri paketleri halinde ancak iki kerede bir yerden diğerine taşınabilir. 12-bitlik bir veri Yüksek anlamlı 8-bit ve Düşük anlamlı 8-bit olmak üzere toplam 16-bit veri alan ile temsil edilebilir.

Düşük Anlamalı 8-Bit:	D7	D6	D5	D4	D3	D2	D1	D0
--------------------------	----	----	----	----	----	----	----	----

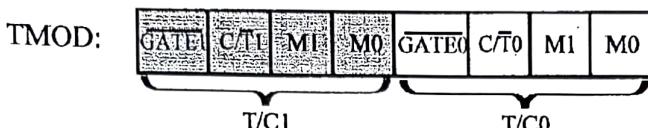
SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrılaştırularak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

Yüksek
Anlamalı 8-Bit:

x	x	x	x	D11	D10	D9	D8
---	---	---	---	-----	-----	----	----

TMOD Saklayıcısı

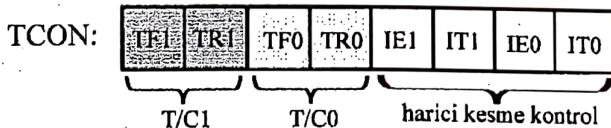
Zamanlayıcı/Sayıcıların çalışma ayarlarının yapıldığı kontrol bitlerini içerir. Aşağıda gösterildiği gibi düşük 4-bit'i T/C-0 için yüksek 4-bit'i ise T/C-1 için kullanılan kontrol bitleridir. TMOD SFR'si **bit adreslenemez**.



Bit	İsim	Açıklama
7	<u>GATE1</u>	T/C-1 için kapı kontrolü. <u>GATE1=0</u> iken, TR1 =1 yapılması ile T/C-1 aktif olur. <u>GATE1=1</u> yapılması durumunda, T/C-1 'in aktif olabilmesi için TR1=1 yapılmalı ve P3.3(INT1) pini lojik '1' seviyesinde olmalıdır.
6	<u>C/T1</u>	T/C-1 için giriş kaynağını (zamanlayıcı/sayıçı ?) seçme biti. <u>C/T1 =0</u> Zamanlayıcı, <u>C/T1 =1</u> Sayıcı
5	M1	T/C-1 için verilen tabloya uygun olarak çalışma modunu seçme bitleridir.
4	M0	M1 M0 Mod Açıklama 0 0 0 13-bit zamanlayıcı/sayıçı modu. 0 1 1 16-bit zamanlayıcı/sayıçı modu. 1 0 2 8-bit otomatik yüklemeli zamanlayıcı/sayıçı modu. 1 1 3 Bağımsız çalışma modu.
3	<u>GATE0</u>	T/C-0 için kapı kontrolü. T/C-0 için <u>GATE1</u> ile aynı işlev sahiptir. T/C-0 için harici start/stop işaretini P3.2(INT0) pininden gelmektedir.
2	<u>C/T0</u>	T/C-0 için giriş kaynağını (zamanlayıcı/sayıçı ?) seçme biti. <u>C/T0 =0</u> Zamanlayıcı, <u>C/T0 =1</u> Sayıcı
1	M1	T/C-0 için üstte verilen tabloya uygun olarak çalışma modunu seçme bitleridir.
0	M0	

TCON Saklayıcısı

TCON SFR'sinin yüksek 4-bit'i zamanlayıcı/sayıclarla, düşük 4-bit'i ise harici kesme kaynakları ile ilgilidir. TCON saklayıcısı **bit adreslenebilir**.



Bit	İsim	Açıklama
7	TF1	T/C-1 taşıma bayrağı. T/C-1 taşıması durumunda donanım tarafından otomatik olarak setlenir. Kesme yapısı kullanılması durumunda program ilgili kesme hizmet altprogramına (Interrupt Service Routine, ISR) dallandığında yine donanım tarafından temizlenir. Kesme yapısı kullanılmadığında ise taşıma sonrasında yazılımla temizlenmelidir.

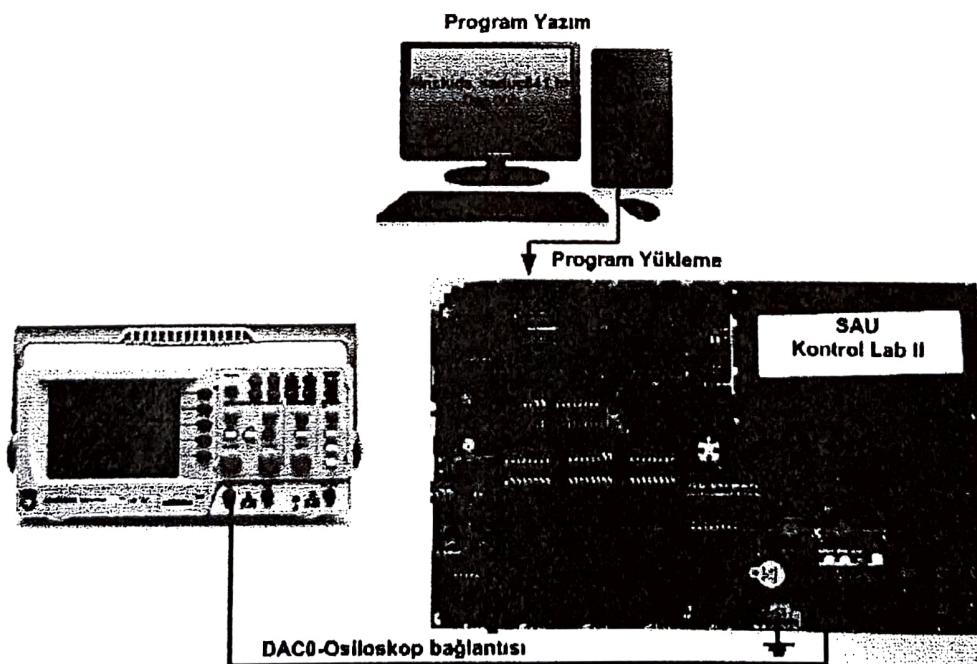
SAU / Muh.Fak./EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak AduC841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

6	TR1	T/C-1 çalışma biti. TR1 = 0 durumunda T/C-1 pasiftir. TR1 = 1 durumunda ise T/C-1' in aktif olması GATE1 bitinin veya P3.3(INT1) pininin durumuna bağlıdır. Yukarıda verilen blok diyagramlarını ve açıklamalarını inceleyiniz.
5	TF0	T/C-0 taşıma bayrağı. T/C-0 için TF1 ile aynı işlev sahiptir.
4	TR0	T/C-0 çalışma biti. T/C-0 için TR1 ile aynı işlev sahiptir.
3	IE1	Harici kesme1 oluşturgunda donanım tarafından setlenir. İlgili ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
2	IT1	Harici kesme1 (INT1) için kesme algılama kontrol bitidir. Harici kesme1 girişi P3.3 pinidir. IT1=1 yapıldığında P3.3 pininden gelen işaretin düşen kenarında (1-0 geçişinde) kesme oluşur. IT1=0 yapıldığında P3.3 pininden gelen işaretin Lojik 0 olması durumunda kesme üretilir. Bu durumda giriş işaretin Lojik 0 olduğu sürece sürekli kesme üretilir.
1	IE0	Harici kesme0 oluşturgunda donanım tarafından setlenir. ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
0	IT0	Harici kesme0 (INT0) için kesme algılama kontrol bitidir. İşlevi IT1 ile aynıdır. Harici kesme0 girişi P3.2 pinidir.

T/C-0 ve T/C-1 4 farklı çalışma modu aşağıda detaylı olarak incelenmiştir. Çalışma modlarına ait şekiller T/C-1 için verilmiştir.

DENEY 1:

Bu deneyin amacı AduC841 mikrokontrolör mimarisini ve adresleme yöntemlerini kullanarak bir transfer fonksiyonun mikrokontrolör tabanlı gerçekleştirilecektir. Deney sonucunda $T=0.1$ msn periyotlarla $f(t)=\sin bt$ fonksiyonu değerleri Q15 yöntemi ile birlikte hesaplanarak DAC0 çıkışından üretilecektir.

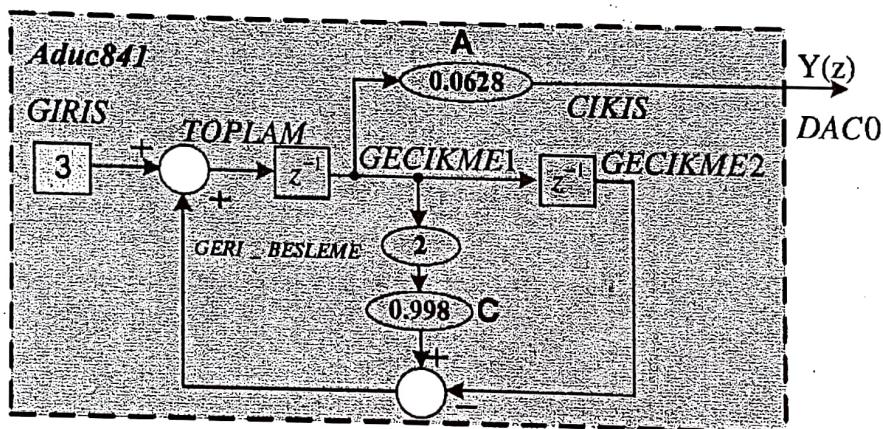


Şekil 4. Deney Düzeneği

- Deney 1 ön çalışma: Bu kısım ile ilgili yazılım kodları deney esnasında yazılacaktır
(Sadece işaretsiz 2 adet 16 bit sayının çarpma algoritması alt program olarak kullanılacaktır).
 - SAYI1=2500d ; SAYI2=0.685d (9)

- 1- DAC0 Vref harici ve 12-bit mod seçilir.
 - 2- Denklem (9)'da verilen SAYI1 ve SAYI2'nin çarpımı CARP_SAYI2xSAYI1 altprogramı kullanılarak Q15 yöntemine göre hesaplanır.
 - 3- Hesaplanan sonuçlar (SONUC_BYTEx_4/3) sırasıyla DAC0H ve DAC0L 'a yazılacaktır.
 - 4- uVision simülatörü kullanılarak r0,r1,r2,r3 (SONUC_BYTEx_1/2/3/4) kaydedicilerine hesaplanan sonuç değerleri gözlenecektir.
 - 5- DAC0 ile üretilen işaret osiloskop ile ($2.09V \pm 5mV$) gözlenecektir.

Deney2: Sinüs işaretinin üretilmesi (Yazılım kodları çalışma amaçlı olarak aşağıda verilmiştir.)



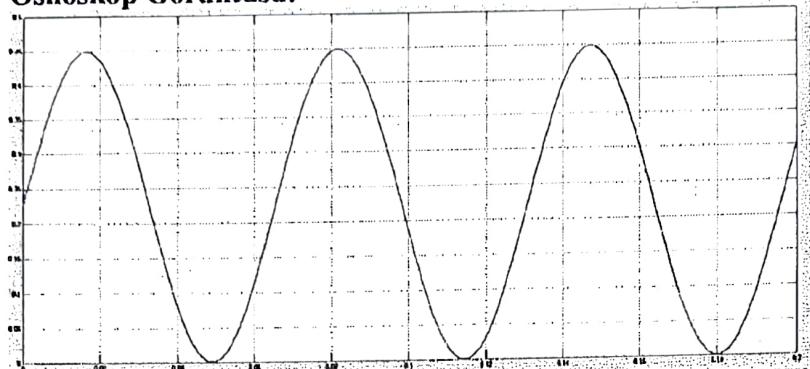
Program basamakları:

- 1- Değişkenler tanımlanır ve RAM adresleri belirlenir.
 - 2- DAC0 Vref harici ve 12-bit mod seçilir.
 - 3- Timer0 0.1ms ve kesmeli olarak ayarlanır.
 - 4- Q15 yöntemi kullanılarak Şekil 2'de işaret akış diyagramı ve sembolik dilde programı verilen transfer fonksiyon gerçekleştirilir.
 - 5- Bu veriler sırası ile DAC0H ve DAC0L yazılır.
 - 6- Bu işlemler periyodik olarak devam eder.

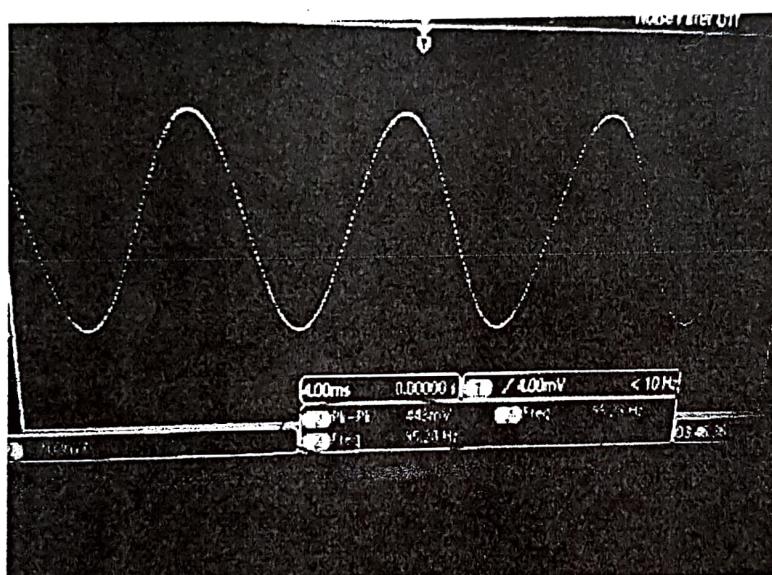
Deney2 sonucuna ait osiloskop görüntüsü ve yazılıma ait kodlar aşağıda verilmiştir.

*SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile
Geçerleştirilmesi ve PWM Modülünün Kullanılması*

Osiloskop Görüntüsü:



Şekil 5. 100 Hz Sinüs işaretine ait bilgisayar simülasyonu görüntüsü



Şekil 6. Aduc841'de üretilen 100 Hz Sinüs işaretinin DAC0 çıkışına ait osiloskop görüntüsü

Yazılım:

#include<Aduc841.h>

GIRIS	EQU 20H
CIKIS_L	EQU 21H
CIKIS_H	EQU 22H
GERI_BESLEME_L	EQU 23H
GERI_BESLEME_H	EQU 24H
GECIKME1_L	EQU 25H
GECIKME1_H	EQU 26H
GECIKME2_L	EQU 27H
GECIKME2_H	EQU 28H
TOPLAM_L	EQU 29H
TOPLAM_H	EQU 2AH

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayırılaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

```
KATSAYI_A_L    EQU 2CH
KATSAYI_A_H    EQU 2DH
KATSAYI_C_L    EQU 2EH
KATSAYI_C_H    EQU 2FH ; 16 bit X 16 bit İşgertsiz sayı çarpımı
```

CSEG

ORG 00h
SJMP MAIN

ORG 0BH
SJMP TIMER0 KESME

MAIN.

MIN.
:DAG avar

```
MOV DACCON ,#0111111B ; 12bit, Vref=5V, asenkron çalışma, DAC0 power-on
MOV DAC0H ,#00H
MOV DAC0L ,#00H
```

;Timer0 ayar

```
MOV TMOD    ,#00000001B ;mod1,dahili start/stop,dahili saat  
MOV TL0     ,#0ADH  
MOV TH0     ,#0FBH ;0.1msn,bknz. NotI
```

;timer0 kesme ayarı

SETB EA
SETB ET0

;değişken ilk değerler

```
MOV SP      ,#040H    ;serbest olarak seçildi
MOV GECIKME1_L ,#00H
MOV GECIKME1_H ,#00H
MOV GECIKME2_H ,#00H
MOV GECIKME2_L ,#00H
MOV GIRIS ,#02H
```

```

MOV KATSAYI_A_H    ,#10H      ;A=SIN(2*PI*100*0.0001)=0.0628
MOV KATSAYI_A_L    ,#13H      ;A=0.06286*65535=4115d=1013h Q15
MOV KATSAYI_C_H    ,#0FFH      ;C=COS(2*PI*100*0.0001)=0.998

```

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayriklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

MOV KATSAYI_C_L ,#07CH ;C=0.9999*65535=65529d=FFF8h Q15
SETB TR0

bekle:

sjmp bekle ;sonsuz döngü,timer0 kesmesini bekle.

=====;
;NOT:

;tl0 ve th0 100us için değer girişi dptr kullanılarak şu şekilde gerçekleştirilebilir:

;0.1ms=100,000ns ve bir çevrim 90.4ns ise

;timer0 rru sayısını gereken $100,000/90.4 = 1106d$ dir (osc=11.0593MHz)

;timer0 ra yüklenmesi gereken $65535d - 1106d = 64429$.

; mov dptr,#64429d

; mov th0,dph

; mov tl0,dpl

TIMER0_KESME:

MOV DAC0H ,CIKIS_H
MOV DAC0L ,CIKIS_L

MOV TL0,#0ADH

MOV TH0,#0FBH

ACALL HESAP_CIKIS ; Çıkış hesaplanır
ACALL HESAP_GERIBESLEME ; Geri besleme değeri hesaplanır
ACALL HESAP_TOPLAM ; Geri besleme ile giriş değeri toplanır
ACALL HESAP_GECIKME2 ; Gecikme1 değişkeni Gecikme2 değişkenine atanır
ACALL HESAP_GECIKME1 ; Gecikme değişkeni Toplam değişkenine atanır

RETI

HESAP_CIKIS:

MOV KATSAYI_A_H ,#00H ;
MOV KATSAYI_A_L ,#03H ;

MOV SAY2_L ,GECIKME1_L
MOV SAY2_H ,GECIKME1_H
MOV SAY1_L ,KATSAYI_A_L
MOV SAY1_H ,KATSAYI_A_H
ACALL CARP_SAYI2xSAYI1 ; A*GECIKME1

MOV CIKIS_L ,SONUC_BYTE_1
MOV CIKIS_H ,SONUC_BYTE_2

RET

=====;
HESAP_GERIBESLEME:

MOV SAY2_L ,GECIKME1_L
MOV SAY2_H ,GECIKME1_H
MOV SAY1_L ,#02H

*SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile
Geçerleştirilmesi ve PWM Modülüün Kullanılması*

```
MOV      SAY1_H      ,#00H
ACALL    CARP_SAYI2xSAYII ;GECIKME1*2
MOV      SAY2_L      ,SONUC_BYTE_1
MOV      SAY2_H      ,SONUC_BYTE_2
MOV      SAY1_L      ,KATSAYI_C_L
MOV      SAY1_H      ,KATSAYI_C_H
ACALL    CARP_SAYI2xSAYII
MOV      SAY2_L      ,SONUC_BYTE_3
MOV      SAY2_H      ,SONUC_BYTE_4 ;ilk iki byte ihmali(Q15)
MOV      SAY1_L      ,GECIKME2_L
MOV      SAY1_H      ,GECIKME2_H
ACALL    FARK_SAYI2_SAYII ;GECIKME1*2*C - GECIKME2
MOV      GERI_BESLEME_L ,SONUC_BYTE_1
MOV      GERI_BESLEME_H ,SONUC_BYTE_2
RET
```

HESAP_TOPLAM:

```
MOV      ACC      ,GIRIS
ADD      A      ,GERI_BESLEME_L
MOV      TOPLAM_L ,A
MOV      ACC      ,#00H
ADDC    A      ,GERI_BESLEME_H
MOV      TOPLAM_H ,A
RET
```

FARK_SAYI2_SAYII:

```
MOV      ACC ,SAY2_L
CLR      C
SUBB   A ,SAY1_L

MOV      SONUC_BYTE_1 ,A
MOV      A ,SAY2_H
SUBB   A ,SAY1_H

MOV      SONUC_BYTE_2 ,A
RET
```

HESAP_GECIKME2:

```
MOV      GECIKME2_L ,GECIKME1_L
MOV      GECIKME2_H ,GECIKME1_H
```

*SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayriklaştırularak Aduc841 ile
Geçerleştirilmesi ve PWM Modüllünün Kullanılması*

RET

=====;
HESAP_GECIKME1:

```
MOV GECIKME1_L ,TOPLAM_L  
MOV GECIKME1_H ,TOPLAM_H  
RET
```

=====;
CARP_SAYI2xSAYI1:

```
; 16-Bit x 16-Bit işaretsiz sayı çarpım alt programı  
; çarpım sonucu 32 bitir  
; giriş: r1, r0 = çarlanan X (say2_H,say2_L)  
; r3, r2 = çarpan Y (say1_H,say1_L)  
; çarpum: r3, r2, r1, r0 = X x Y = SAY2xSAYI=SONUÇ_BYTE_4/3/2/1  
push B  
push dpl  
mov a, r0  
mov b, r2  
mul ab ; multiply XL x YL  
push acc ; stack result low byte  
push b ; stack result high byte  
mov a, r0  
mov b, r3  
mul ab ; multiply XL x YH  
pop 00H  
add a, r0  
mov r0, a  
clr a  
addc a, b  
mov dpl, a  
mov a, r2  
mov b, r1  
mul ab ; multiply XH x YL  
add a, r0  
mov r0, a  
mov a, dpl  
addc a, b  
mov dpl, a  
clr a  
addc a, #00H  
push acc ; save intermediate carry  
mov a, r3  
mov b, r1  
mul ab ; multiply XH x YH  
add a, dpl  
mov r2, a  
pop acc ; retrieve carry  
addc a, b  
mov r3, a
```

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

```

mov r1,00H
pop 00H      ; retrieve result low byte
pop dpl
pop B

ret
=====
END

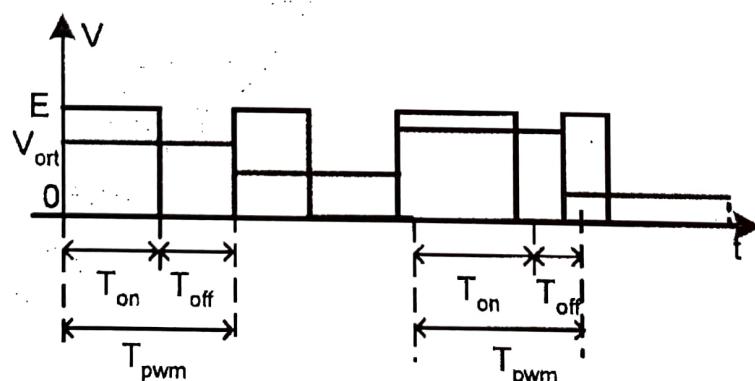
```

Aduc841'de PWM Modülünün kullanılması

Pulse-width modulation (PWM, Darbe genişlik modülasyonu), sabit T_{pwm} periyotlu işaretin ortalama değerini ayarlamak için işaretin iletim süresi T_{on} genişliğinin değiştirilmesi tekniği temeline dayanmaktadır.

PWM elektrik ve elektronikte birçok alanda, farklı amaçlar için kullanılmaktadır. Telekomünikasyon, güç, voltaj düzenleyiciler, ses üreteçleri veya yükselteçler gibi çeşitli uygulama alanları ve farklı uygulamaları bulunmaktadır.

PWM, üretilen sabit T_{pwm} periyotlu işaretin bir periyodu içerisinde iletim süresi T_{on} değiştirilerek Şekil 1'de gösterildiği gibi gerçekleştirilebilir. Şekil 1, bir periyot içerisinde meydana gelecek T_{on} değişimi, sabit T_{pwm} periyotlu işaretin ortalama (V_{ort}) değerini değiştirir. PWM işaretinin periyodu (T_{pwm}), T_{on} ve T_{off} sürelerinin toplamıdır. T_{on} ve T_{pwm} sürelerine bağlı olarak DC işaretin E 'nin ortalama değeri, aşağıda verilen denklem (10) ve (11) kullanılarak hesaplanır.



Şekil 7 : E, Vort gerilimleri ve T_{on} , T_{off} , T_{pwm} zamanları

$$T_{pwm} = T_{on} + T_{off} \quad (10)$$

$$V_{ort} = E \frac{T_{on}}{T_{pwm}}$$

(11)

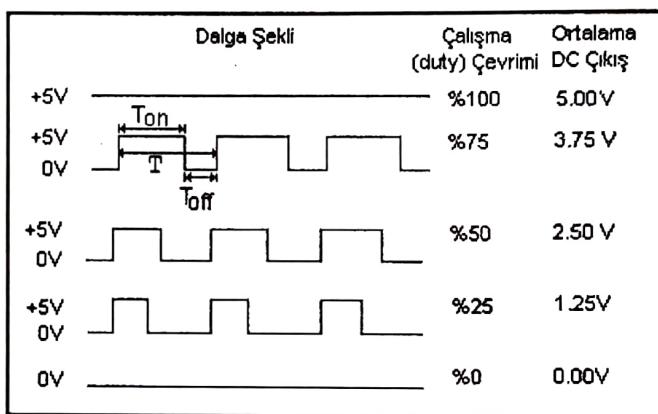
SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

Doluluk Oranı- "Duty" (D) T_{on} süresinin T_{pwm} süresine oranı olarak tanımlanır. Duty (D) değeri yüzde doluluk olarak da ifade edilmektedir. Yüzde doluluk durumunda %Duty değeri denklem (13)'te verilmiştir. Deneyde %D kullanılmaktadır.

$$D = \frac{T_{on}}{T_{pwm}} \quad (12)$$

$$(\%)D = \frac{T_{on}}{T_{pwm}} * 100 \quad (13)$$

Örnek: Şekil 2'de çeşitli % 100, %75, %50, %25 ve %0 duty değerleri için sabit T_{pwm} periyotlu işaretin zamana göre ani değişimi ve sırasıyla ortalama gerilim değerleri 5V, 3.75V, 2.5V, 1.25V ve 0V olarak verilmiştir.

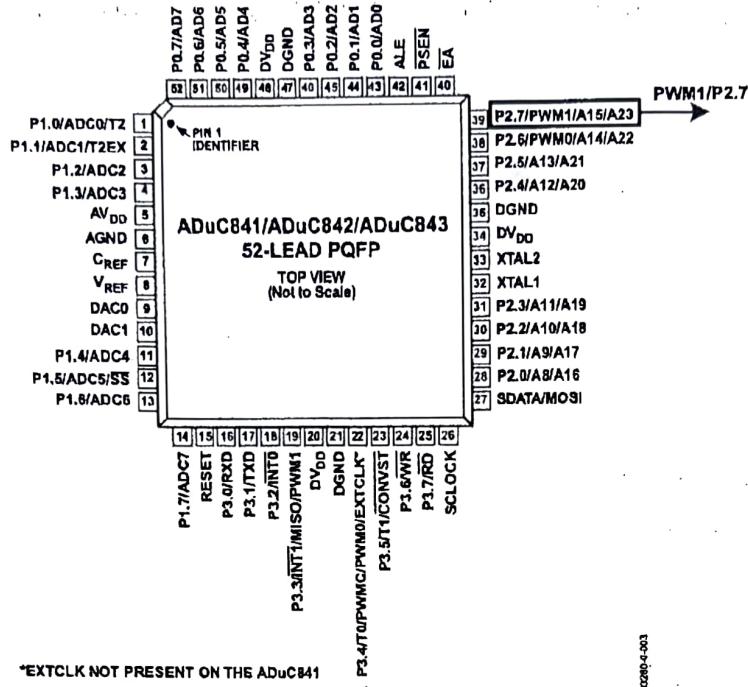


Şekil 8. %Duty değerleri için işaretin ani değişimi ve ortalama gerilim değerleri

Aduc841 Mimarısında PWM Modülü

ADUC 841 mikrokontrolör mimarısında bir adet PWM modülü bulunmaktadır, bu modül PWM0 ve PWM1 olmak üzere iki farklı PWM çıkışlarını kontrol etmektedir. PWM0 ve PWM1 çıkışları şekilde PWM işaretleri, P2.6 ve P2.7 numaralı portlar üzerinden dış dünyaya verilir.

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayriklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

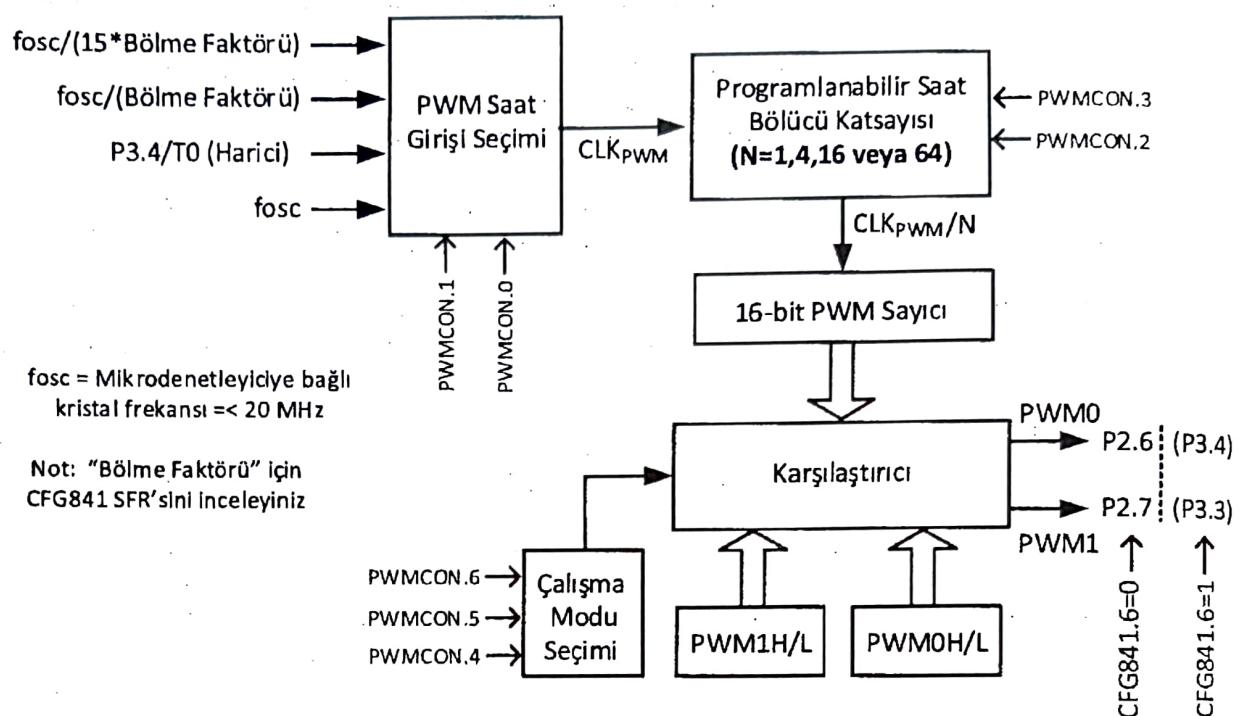


Şekil 9. Aduc841 mikrokontrolörlü yonga çip üzerinde PWM çıkışları

PWM frekansı ve çalışma modu iki adet byte adreslenebilir SFR (**PWMCON** ve **CFG841**) ile belirlenmektedir. PWM frekansı ve duty değeri **PWM0H/L** ve **PWM1H/L** kaydedicileri yardımı ile belirlenmektedir. PWM modülüne ait blok diyagram aşağıda verilmiştir.

ADUC841 mimarisinde çözünürlük ve giriş kaynak saati yazılımla ayarlanabilen, 6 farklı çalışma moduna sahip PWM birimi bulunur. PWM için basitleştirilmiş blok diyagramı aşağıda verilmiştir.

SAU / Muh.Fak./EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması



Şekil 10. ADuC841're ait PWM modülü ve Özel fonksiyon kaydedicileri

Şekilde gösterildiği gibi PWM için 4 farklı saat girişi kullanılabilir. PWMCON SFR'si aracılığı ile seçilen saat girişine ait saat darbeleri programlanabilir (1,4,16,64) bölücten geçirilerek 16-bit uzunluğundaki PWM sayıcısı için giriş kaynağını oluşturur.

PWM biriminin çalışma ayarları (çalışma modu, giriş saati, bölücü katsayı) PWMCON SFR'si aracılığı ile yapılır. PWM işaretinin duty cycle değeri ise PWM0H/L , PWM1H/L veri SFR'lerine, seçilen saat girişine ve programlanabilir bölücü katsayısına bağlıdır.

PWMCON SFR'sine veri yazıldığı anda PWM sayıcısının değeri resetlenir (PWM sayıcısı=0). PWM sayıcısı girişindeki kaynak saat darbeleri ile 0'dan itibaren yukarı yönde sayar. PWM mimarisinde bulunan karşılaştırıcıyı anlık sayma değerini PWM0H/L ve PWM1H/L SFR'lerine yazılımla atanan değerler ile karşılaştırarak PWM çıkışlarının Lojik seviyesini belirler. PWM sayacının anlık sayma değeri çalışma modu için belirlenen üst değere ulaştıktan sonra tekrar sıfır düşer.

PWM çıkış pinleri P2.6 ve P2.7 ($\text{CFG841.6} = \text{PWPO} = 0$ için) veya P3.4 ve P3.3 ($\text{CFG841.6} = \text{PWPO} = 1$ için) olarak ADUC841 konfigürasyon saklayıcısı (CFG841) aracılığı ile belirlenir.

NOT: PWM'in 16-bit'lik çalışma modlarında (Mod 1,3,4 ve 6) ilk önce PWM0L ve PWM1L SFR'lerine ardından PWM0H ve PWM1H SFR'lerine değer yazılmalıdır. PWM0H/L ve PWM1H/L SFR'lerine yazılan yeni değer devam etmekte olan PWM periyodu tamamlandıktan sonra dikkate alınır.

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

PWMCON Saklayıcısı

PWM biriminin çalışma ayarlarının yapıldığı PWMCON SFR'si bit adreslenemez.

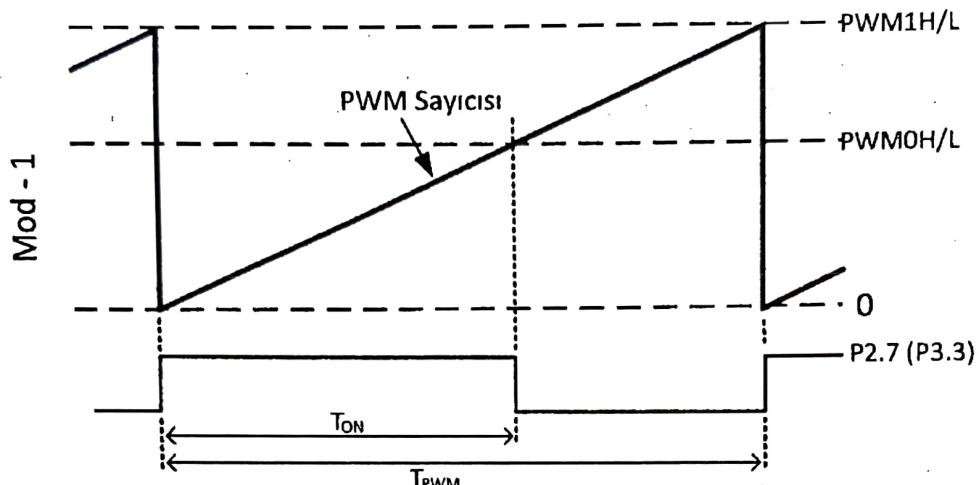
PWMCON:

SNGL	MD2	MD1	MD0	CDIV1	CDIV0	CSEL1	CSEL0
------	-----	-----	-----	-------	-------	-------	-------

Bit	İsim	Açıklama																																				
7	SNGL	Setlendiğinde tek çıkışlı PWM işaretü üretilir. P2.6 (veya P3.4) genel amaçlı kullanıma bırakılır. Çift çıkışlı çalışma modları için SNGL = 0 yapılmalıdır.																																				
6	MD2	MD0, MD1 ve MD2 bitleri aşağıdaki tabloya göre PWM çalışma modunu belirler.																																				
5	MD1	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>MD2</th><th>MD1</th><th>MD0</th><th>Çalışma Modu</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>Mod 0 : PWM Kapalı</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>Mod 1: P2.7 (veya P3.3) pininden tek çıkışlı PWM</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>Mod 2: Çift çıkışlı (twin) 8-bit PWM</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>Mod 3: Çift çıkışlı (twin) 16-bit PWM</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>Mod 4: Çift NRZ 16-bit Σ/Δ DAC</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>Mod 5: Çift çıkışlı (dual) 8-bit PWM</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>Mod 6: Çift RZ 16-bit Σ/Δ DAC</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>Rezerve</td></tr> </tbody> </table>	MD2	MD1	MD0	Çalışma Modu	0	0	0	Mod 0 : PWM Kapalı	0	0	1	Mod 1: P2.7 (veya P3.3) pininden tek çıkışlı PWM	0	1	0	Mod 2: Çift çıkışlı (twin) 8-bit PWM	0	1	1	Mod 3: Çift çıkışlı (twin) 16-bit PWM	1	0	0	Mod 4: Çift NRZ 16-bit Σ/Δ DAC	1	0	1	Mod 5: Çift çıkışlı (dual) 8-bit PWM	1	1	0	Mod 6: Çift RZ 16-bit Σ/Δ DAC	1	1	1	Rezerve
MD2	MD1	MD0	Çalışma Modu																																			
0	0	0	Mod 0 : PWM Kapalı																																			
0	0	1	Mod 1: P2.7 (veya P3.3) pininden tek çıkışlı PWM																																			
0	1	0	Mod 2: Çift çıkışlı (twin) 8-bit PWM																																			
0	1	1	Mod 3: Çift çıkışlı (twin) 16-bit PWM																																			
1	0	0	Mod 4: Çift NRZ 16-bit Σ/Δ DAC																																			
1	0	1	Mod 5: Çift çıkışlı (dual) 8-bit PWM																																			
1	1	0	Mod 6: Çift RZ 16-bit Σ/Δ DAC																																			
1	1	1	Rezerve																																			
4	MD0																																					
3	CDIV1	CDIV1 ve CDIV0 bitleri aşağıdaki tabloya göre PWM Saat darbeleri böltücü katsayısını belirler. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>CDIV1</th><th>CDIV0</th><th>Böltücü Katsayı, N</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>4</td></tr> <tr><td>1</td><td>0</td><td>16</td></tr> <tr><td>1</td><td>1</td><td>64</td></tr> </tbody> </table>	CDIV1	CDIV0	Böltücü Katsayı, N	0	0	1	0	1	4	1	0	16	1	1	64																					
CDIV1	CDIV0	Böltücü Katsayı, N																																				
0	0	1																																				
0	1	4																																				
1	0	16																																				
1	1	64																																				
2	CDIV0																																					
1	CSEL1	CSEL1 ve CSEL0 bitleri aracılığıyla aşağıdaki tabloya göre PWM Saat kaynağı seçilir. Not: Bölme Faktörü CFG841 SFR'sinden belirlenir.																																				
0	CSEL0 <i>Clock Chip Select</i>	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>CSEL</th><th>CSEL</th><th>PWM giriş saati, CLK_{PWM}</th></tr> </thead> <tbody> <tr><td>1</td><td>0</td><td>f_{osc}/(15*Bölme Faktörü)</td></tr> <tr><td>0</td><td>1</td><td>f_{osc}/Bölme Faktörü</td></tr> <tr><td>1</td><td>0</td><td>Harici giriş, P3.4</td></tr> <tr><td>1</td><td>1</td><td>f_{osc}</td></tr> </tbody> </table>	CSEL	CSEL	PWM giriş saati, CLK _{PWM}	1	0	f _{osc} /(15*Bölme Faktörü)	0	1	f _{osc} /Bölme Faktörü	1	0	Harici giriş, P3.4	1	1	f _{osc}																					
CSEL	CSEL	PWM giriş saati, CLK _{PWM}																																				
1	0	f _{osc} /(15*Bölme Faktörü)																																				
0	1	f _{osc} /Bölme Faktörü																																				
1	0	Harici giriş, P3.4																																				
1	1	f _{osc}																																				

PWM Çalışma Modları

Mod 1: Bu çalışma modunda hem darbe genişlik süresi (Ton) hem de PWM çözünürlüğü yazılımla ayarlanabilir. Maksimum PWM çözünürlüğü 16-bit'dir.



Şekil 61. PWM mod-1 çalışma

Şekilde gösterildiği gibi bu modda PWM işaretinin periyodu (T_{PWM}) $PWM1H/L$, T_{ON} süresi ise $PWM0H/L$ SFR'leri ile ayarlanır.

Yukarıda belirtildiği gibi PWMCON SFR'sine veri yazıldığı anda PWM sayıcısının değeri sıfırlanır. PWM sayacı=0 olduğunda PWM çıkışı donanım tarafından Lojik 1 seviyesine çekilir. PWM sayıcısının değeri Şekil 61'de gösterildiği gibi kaynak saat darbeleri ile artar. PWM sayacı=PWM0H/L olduğu anda PWM çıkışı donanım tarafından Lojik 0 seviyesine çekilir. PWM sayacının değeri artmaya devam eder ve PWM sayacı=PWM1H/L olduktan sonra gelen ilk saat darbesi ile PWM sayacı=0'a düşer ve PWM çıkışı donanım tarafından tekrar Lojik 1 seviyesine çekilir.

Mod 1'de PWM1H/L 'a yüklenen değerin küçülmesi çıkış işaretinin frekansını artırırken PWM çözünürlüğü azalır. Örneğin PWM1H/L SFR'lerine 65535 yazıldığında 16-bit çözünürlüğünde PWM elde edilirken maksimum PWM frekansı 169 Hz (11.0592MHz/65536) olur. 12-bit çözünürlüğünde PWM (PWM1H/L = 4095) ise maksimum PWM frekansı 2703 Hz (11.0592 Mhz/4096) olur.

Mod 1'de PWM işaretini P2.7 (CFG841.6 = PWPO = 0) veya P3.3 (CFG841.6 = PWPO = 1) pininden dış dünyaya aktarılır.

CFG841 Kaydedicisi

bit 7	EXSP	PWPO	DBUF	EPM2	EPM1	EPM0	MSPI	XRAMEN	bit 0
-------	------	-------------	------	------	------	------	------	--------	-------

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayırılaştırularak Adıuc841 ile Gerçekleştirilmesi ve PWM Modüllerinin Kullanılması

Bu kaydedicinin yalnızca 2, 3, 4 ve 6 numaralı bitleri PWM kontrolü amacıyla PWM frekansının belirlenmesinde ve PWM işaretlerinin çıkış portlarının seçiminde kullanılır. Bu dört bit ile bölen katsayı ve PWM çıkış pinleri belirlenir. Bölen katsayı, PWM frekansının oluşturulmasında bir sonraki bölümde (PWMCN ayarlarında) $katsayı_{cfg841}$ olarak kendini gösterecektir. CFG841 kaydedicisinde bulunan 4 bitin fonksiyonu tablo 1 de verilmiştir.

Tablo 1 : CFG841 kaydedicisine ait PWM sürecinden sorumlu bit ve fonksiyonları.

Bit No	İsim	Açıklama
6	PWPO	PWM çıkış pini belirleme
		= '1' ise PWM çıkışları p3.4, p3.3 (PWM0,PWM1) = '0' ise PWM çıkışları p2.6, p2.7 (PWM0,PWM1)
4	EPM 2	Katsayı belirleyici 2
3	EPM 1	Katsayı belirleyici 1
2	EPM 0	Katsayı belirleyici 0
		EPM2 EPM1 EPM0 katsayı_{cfg841}
		0 0 0 32
		0 0 1 64
		0 1 0 128
		0 1 1 256
		1 0 0 512
		1 0 1 1024

- ✓ Mod 1'de PWM işaret P2.7 (CFG841.6 = PWPO = 0) veya P3.3 (CFG841.6 = PWPO = 1) pininden dış dünyaya aktarılır.

PWM frekansının belirlenebilmesi içi aşağıdaki formül kullanılacaktır.

$$f_{pwm} = \frac{1}{\frac{1}{f_{saat}} * \frac{1}{SGB} * katsayı_1} \quad (5)$$

Yukarıdaki denklemde kullanılan $katsayı_1$ değişkeni, PWM1H ve PWM1L'a yüklenmesi gereken 16 bitlik sayı değerini temsil eder.

PWM frekansı belirlendikten sonra PWM işaretinin Ton süresi hesaplanması gereklidir. Ton süresi aşağıdaki gibi hesaplanır.

$$T_{on} = \frac{1}{f_{saat}} * \frac{1}{SGB} * katsayı_0 \quad (8)$$

Yukarıdaki denklemde kullanılan $katsayı_0$ değişkeni, PWM0H ve PWM0L'a yüklenmesi gereken 16 bitlik sayı değerini temsil eder.

Burada dikkat edilmesi gereken,

$$T_{on} < T_{pwm} \quad (9)$$

$$f_{pwm} = 200 \text{ Hz}$$

$$T_{pwm} = \frac{1}{f_{pwm}} = 0,005$$

$$f_{osc} = 90 \text{ ns}$$

$$\text{Pwm seysi} = \frac{5000000}{90} = 55555$$

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

şartının sağlanmasıdır. Eşitsizliğin sağlanmadığı durumlarda şekil 3'den de görüleceği üzere PWM işaret çıkışı sürekli lojik '1' de kalacaktır.

Deney3:

Bu deneyde gerilimin ortalama değeri PWM tekniği ile değiştirilerek, PWM modülasyonu gözlemlenecektir. Aduc841 mikrokontrolörün içerisinde bulunan PWM modülü kullanılarak, %25, %50 ve %75 duty değerlerine dayalı 200Hz'lik PWM darbeleri üretilecektir. Üretilen bu PWM darbeleri p2.7 üzerinden uygulanarak, farklı duty değerleri için PWM işaretin gözlemlenecektir.

Üretilen PWM işaretin ve PWM modülüne ait özellikler aşağıda verilmiştir.

1. PWM0-P2.7 çıkışı kullanılacaktır.
2. PWM modu = 1
3. saat giriş bölücüsü (SGB) = 1
4. $f_{saat} = f_{osc}$
5. $f_{pwm} = 200Hz$

Katsayı_0 ve Katsayı_1 değerlerinin hesaplanması:

$f_{pwm} = 200Hz$; $f_{saat} = 11.0592MHz$ %25 duty için **PWM0H/L** ve **PWM1H/L** değerlerinin hesaplanması;

Katsayı_1 Hesabı:

PWM Mod-1 kullanılacağı için f_{pwm} değerini **PWM1H/L** kaydedicilerine yüklenecek değer belirler. Aşağıda hesabı yapılan ve yazılım sırasında tanımlanacak olan 16 bitlik **katsayı_1** değişkenin karşılığı **PWM1H/L**'a yüklenmelidir.

$$f_{pwm} = \frac{1}{\frac{1}{f_{saat}} * \frac{1}{SGB} * \text{katsayı}_1}$$

$$200Hz = \frac{1}{\frac{1}{11.0592 \cdot 10^6 Hz} * \frac{1}{1} * \text{katsayı}_1} \Rightarrow \text{katsayı}_1 = 55296d = 0D800h$$

PWM1H = 0D8h
PWM1L = 00h

Katsayı_0 Hesabı:

%25 doluluk oranı için $T_{pwm} = \frac{1}{200Hz} = 0.005sn \Rightarrow \%D = \%25 \Rightarrow T_{on} = 0.00125sn = 12.5msn$ olarak elde edilir.

$$T_{on} = \frac{1}{f_{saat}} * \frac{1}{SGB} * \text{katsayı}_0$$

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması

$$12.5 \cdot 10^{-3} msn = \frac{1}{11.0592 \cdot 10^6 Hz} * \frac{1}{1} * \text{katsayı}_0$$

$$\text{katsayı}_0 = 13824d = 3600h$$

$$\boxed{\text{PWM0H} = 36h}$$

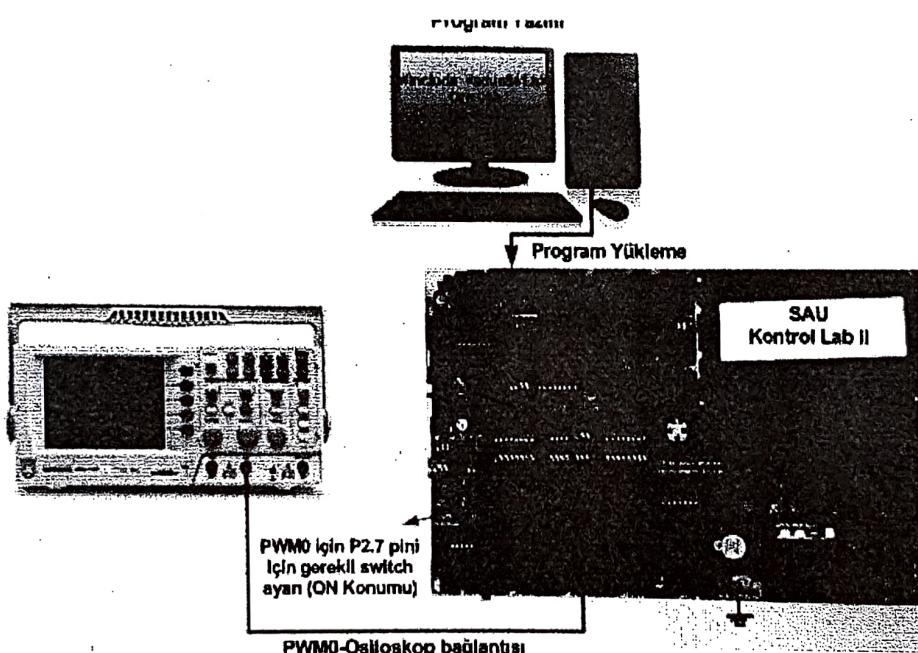
$$\boxed{\text{PWM0L} = 00h}$$

Deney Sonrası çalışma

%50 ve %75 doluluk oranlarına sahip PWM işaretlerini üretiniz

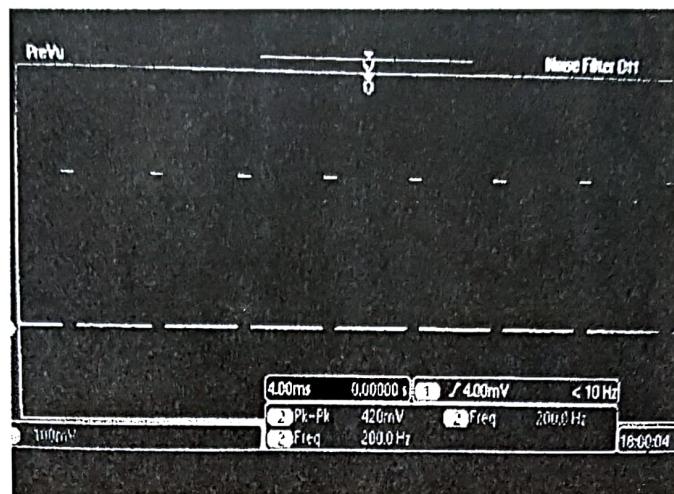
Deney Düzeneği

NOT: Deney düzeneğinde PWM çıkışının osiloskop bağlantısını P2.7 üzerinden alınız. P2.7 portunun ilgili anahtarnın ON konumunda bulunmasına dikkat ediniz.



Osiloskop Görüntüsü:

SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Aduc841 ile Gerçekleştirilmesi ve PWM Modülünün Kullanılması



Yazılım:

```
#include <aduc841.h>
```

```
=====
;Deney'e ait asagida verilen yazılım da %25 doluluk oranına sahip 200 Hz frekansında PWM işaretini
;üretilecektir.
=====
```

```
katsayi_0L equ 20h ; PWM0L değerine yüklenmesi gereken değer
katsayi_0H equ 21h ; PWM0H değerine yüklenmesi gereken değer
```

```
katsayi_1L equ 22h ; PWM1L değerine yüklenmesi gereken değer
katsayi_1H equ 23h ; PWM1H değerine yüklenmesi gereken değer
```

CSEG

```
ORG 00h
SJMP MAIN
```

MAIN:

```
MOV SP ,#070H
MOV PWMCON,#00010011B; PWM mode-1, Çarpan_fosc=1, pwm saat girişi fosc
MOV katsayi_0L, # 00H
MOV katsayi_0H, # 36H ; PWM0H/L kaydedicisine yüklenmesi gereken değer Ton süre ayarı %25 doluluk
;orası için ilgili hesaplar yukarıda yapılmıştır
MOV katsayi_1L, #00H
MOV katsayi_1H, #0D8H ; PWM1H/L kaydedicisine yüklenmesi gereken değer 200Hz pwm frekans ayarı,
;ilgili hesaplar yukarıda yapılmıştır
```

```
acall pwm_cikis_ver
```

*SAU / Muh.Fak./ EEM/ Bir Transfer Fonksiyonunun Ayrıklaştırılarak Ad uc841 ile
Geçekleştirilmesi ve PWM Modülünün Kullanılması*

bekle:

sjmp bekle ;sonsuz döngü

pwm_cikis_ver:

mov cfg841, #00H ;CFG841 üzerinden pwm çıkış p2.7 ayarı

mov pwm0L, katsayi_0L

mov pwm1L, katsayi_1L

mov pwm0H, katsayi_0H

mov pwm1H, katsayi_1H

RET

END

KOMUT TABLOSU İLE İLGİLİ AÇIKLAMALAR:

1) Etkilenen Bayraklar (E.B.) : Bazı komutlar koşturulurken PSW saklayacısındaki bayrakların değeri değişebilir. Herbir komut için etkilenen bayraklar belirtilmiştir.

2) Byte: Mikrodenetleyicinde kullanılan komutların herbirini farklı uzunluğa sahiptir. Herbir komut için uzunluk (Byte) değeri tabloda belirtilmiştir.

Program Counter (PC) 'in değeri her komuttan sonra koşturulan komutun "Byte" değerini kadar donanım tarafından otomatik olarak arttırılır.

Örneğin; "MOV A, R0" komutu 1-Byte'lık bir komuttur. Bu komut koşturulduktan sonra PC değeri 1 artırılır.

Yani; $PC \leftarrow PC + 1$ olur.

Benzer şekilde "MOV 40h, #35d" komutu 3-Byte'lık bir komuttur. Dolayısı ile bu komut icra edildikten sonra PC değeri; $PC \leftarrow PC + 3$ şeklinde donanım tarafından otomatik olarak güncellenir.

3) Makine Çevrimi (M.Ç.) : Bir komutun koşturulması için kaç makine çevrimi geçtiğini dolayısı ile komutun içası için harcanan süreyi gösterir. Standart 8051'de (12 MHz kristal) bir makine çevrimi 12 osilatör periyodundan oluşur. Dolayısıyla standart 8051'de 1 Makina Çevrimi = $1\mu s$ 'dir.

Aduc841'de ise 1 Makina Çevrimi = 1 osilatör periyodu şeklindedir. Dolayısı ile 12 MHz Kristal ile çalıştırılan Aduc841 için 1 Makina Çevrimi = $1/12 = 0.0833\mu s$ 'dir.

4) Komutlarda; Rn ifadesi R0,R1, R2,R3, R4,R5, R6 ve R7 saklayıcılarını,

Ri ifadesi ise R0 ve R1 saklayıcılarının temsil etmektedir.

direct : Dahili RAM hafızanın doğrudan adreslenebilir alanlarının tümünü ifade etmektedir. Standart 8051'de alt 128-Byte hem doğrudan hem de dolaylı adreslenebilir. SFR bölgesinin ise sadece doğrudan adreslenebilir olduğu unutulmamalıdır.

Komut	Komut	Kullanımı	Açıklama	Sembolik Gösterim	E.B.	Byte	M.C.
MOV	A, Rn	Rn saklayıcısındaki değeri akümülatöre yükle		(A) ← [Rn]	-	1	1
MOV	A, direct	Adresteki değeri akümülatöre yükle		(A) ← (direct)	-	2	1
MOV	A, #data	Sabit değeri akümülatöre yükle		(A) ← #data	-	2	1
MOV	Rn, A	Akümülatörü Rn saklayıcısına yükle		(Rn) ← (A)	-	1	1
MOV	Rn, direct	Adresteki değeri Rn saklayıcısına yükle		(Rn) ← (direct)	-	2	2
MOV	Rn, #data	Sabit değeri Rn saklayıcısına yükle		(Rn) ← #data	-	2	1
MOV	direct ,A	Akümülatördeki değeri adres'e yükle		(direct) ← (A)	-	2	1
MOV	direct, Rn	Rn saklayıcısındaki değeri adres'e yükle		(direct) ← (Rn)	-	2	2
MOV	direct1,direct2	Adres 2'deki değeri adres 1'e yükle		(direct1) ← (direct2)	-	2	1
MOV	direct, #data	Sabit değeri direct adres'e yükle		(direct) ← #data	-	2	2
MOV	A, @Ri	Ri'nin gösterdiği adresteki değeri akümülatöre yükle		(A) ← ((Ri))	-	3	2
MOV	direct, @Ri	Ri'nin gösterdiği adresteki değeri direct adres'e yükle		(direct) ← ((Ri))	-	2	2
MOV	@Ri, A	Akümülatörü Ri'nin gösterdiği adres'e yükle		((Ri)) ← (A)	-	1	1
MOV	@Ri, direct	Adresteki değeri Ri'nin gösterdiği adres'e yükle		((Ri)) ← (direct)	-	2	2
MOV	@Ri, #data	Sabit değeri Ri'nin gösterdiği adres'e yükle		((Ri)) ← #data	-	2	1
MOV	DPTR,#data 16	16 bitlik sabit değeri DPTR ye yükle		DPH-DPL ← #data15-1 - #data7-0	-	3	2
MOV	C, bit	Bit değerini elde bayrağına yükle		(C) ← (bit)	C	2	1
MOV	bit, C	Elde bayrağındaki değeri bite yükle		(bit) ← (C)	C	2	1
MOV	A, @A+DPTR	Program hafızanın A+DPTR ile gösterilen adresindeki değeri akümülatöre yükle		(A) ← ((A) + (DPTR))	-	1	2
MOVC	A, @A+PC	Program hafızanın A+PC ile gösterilen adresindeki değeri akümülatöre yükle		(A) ← ((A) + (PC)) ·(PC) ← (PC) + 1	-	1	2
MOVX	A, @Ri	Ri saklayıcısının gösterdiği harici RAM adresindeki değeri akümülatöre yükle		(A) ← ((Ri))	-	1	2
MOVX	@Ri, A	Akümülatördeki değeri Ri'nin gösterdiği harici RAM adresine yükle		((Ri)) → (A)	-	1	2
MOVX	A, @DPTR	DPTR'nin gösterdiği harici RAM (16 bitlik) adresindeki değeri ACC'ye yükle		(A) ← ((DPTR))	-	1	2
MOVX	@DPTR, A	Akümülatördeki değeri DPTR'nin gösterdiği harici RAM adresine yükle		((DPTR)) ← (A)	-	1	2
PUSH	Direct	Direct adrestekideğiri yığının (SP) gösterdiği adres'e yükle		1. (SP) ← (SP)+1 2. ((SP)) ← (direct)	-	2	2
POP	Direct	Yığının gösterdiği adreseki bilgiyi direct adres'e yükle		1.(direct) ← ((SP)) 2.(SP)← (SP)-1	-	2	2
SWAP	SWAP A	Akümülatörün ilk nibbles'i ile ikinci nibblesini yer değiştir	(No operation)	(A ₃₋₀) ↔ (A ₇₋₄)	-	1	1
NOP	NOP	(No operation) Herhangi bir işlem yapılmaz, program akışı 1 makine çevrimi kadar geciktirilmiş olur.		(PC) ← (PC) + 1	-	1	1

Komut	Komut Kullanımı	Açıklama	Sembolik Gösterim	E.B.	Byte	M.C.
XCH	A, Rn	Rn ve akümülatörün içeriklerini değiştirir	(A) \leftrightarrow (Rn)	-	1	1
XCH	A, direct	Adresteği değer ve akümülatörün içeriğini değiştirir	(A) \leftrightarrow (direct)	-	2	1
XCH	A, @Ri	Ri'nin gösterdiği adres ve akümülatörün içeriklerini değiştirir	(A) \leftrightarrow ((RI))	-	1	1
XCHD	A, @Ri	Ri'nin gösterdiği adres ve akümülatörün içeriklerinin ilk dört bitini değiştirir	(A) \leftarrow ((RI _{3..0}))	-	1	1
ANL	A, Rn	Rn ile akümülatörü "VE" işlemine tabi tut	(A) \leftarrow (A) \wedge (Rn)	-	1	1
ANL	A, direct	Adresteği değer ile akümülatörü "VE" işlemine tabi tut	(A) \leftarrow (A) \wedge (direct)	-	2	1
ANL	A, @Ri	Ri'nin gösterdiği adresteki değer ile akümülatörü "VE" işlemine tabi tut	(A) \leftarrow (A) \wedge ((RI))	-	1	1
ANL	A, #data	Sabit değer ile akümülatörü "VE" işlemine tabi tut	(A) \leftarrow (A) \wedge #data	-	2	1
ANL	direct, A	Adresteği değer ile akümülatörü "VE" işlemine tabi tut	(direct) \leftarrow (direct) \wedge (A)	-	2	1
ANL	direct, #data	Sabit değer ile adresteği değeri "VE" işlemine tabi tut	(direct) \leftarrow (direct) \wedge #data	-	3	2
ANL	C, bit	Elde ile biti "VE" işlemine tabi tut	(C) \leftarrow (C) \wedge (bit)	-	2	2
ANL	C, /bit	Elde ile bitin tersini "VE" işlemine tabi tut	(C) \leftarrow (C) \wedge / (bit)	-	2	2
ORL	A, Rn	Rn ile akümülatörü "VEYA" işlemine tabi tut	(A) \leftarrow (A) V (Rn)	-	1	1
ORL	A, direct	Adresteği değer ile akümülatörü "VEYA" işlemine tabi tut	(A) \leftarrow (A) V (direct)	-	2	1
ORL	A, @Ri	Ri'nin gösterdiği adresteki değer ile akümülatörü "VEYA" işlemine tabi tut	(A) \leftarrow (A) V ((RI))	-	1	1
ORL	A, #data	Sabit değer ile akümülatörü "VEYA" işlemine tabi tut	(A) \leftarrow (A) V #data	-	2	1
ORL	direct, A	Adresteği değer ile akümülatörü "VEYA" işlemine tabi tut	(direct) \leftarrow (direct) V (A)	-	2	1
ORL	direct, #data	Sabit değer ile adresteği değeri "VEYA" işlemine tabi tut	(direct) \leftarrow (direct) V #data	-	3	2
ORL	C, bit	Elde ile biti "VEYA" işlemine tabi tut	(C) \leftarrow (C) V (bit)	-	2	2
ORL	C, /bit	Elde ile bitin tersini "VEYA" işlemine tabi tut	(C) \leftarrow (C) V / (bit)	-	2	2
XRL	A, Rn	Rn ile akümülatörü "ÖZEL VEYA" işlemine tabi tut	(A) \leftarrow (A) $\vee\neg$ (Rn)	-	1	1
XRL	A, direct	Adresteği değer ile akümülatörü "ÖZEL VEYA" işlemine tabi tut	(A) \leftarrow (A) $\vee\neg$ (direct)	-	2	1
XRL	A, @Ri	Ri'nin gösterdiği adresteki değer ile akümülatörü "ÖZEL VEYA" işlemine tabi tut	(A) \leftarrow (A) $\vee\neg$ ((RI))	-	1	1
XRL	A, #data	Sabit değer ile akümülatörü "ÖZEL VEYA" işlemine tabi tut	(A) \leftarrow (A) $\vee\neg$ #data	-	2	1
XRL	direct, A	Adresteği değer ile akümülatörü "ÖZEL VEYA" işlemine tabi tut	(direct) \leftarrow (direct) $\vee\neg$ (A)	-	2	1
XRL	direct, #data	Sabit değer ile adresteği değeri "ÖZEL VEYA" işlemine tabi tut	(direct) \leftarrow (direct) $\vee\neg$ #data	-	3	2
CLR	A	Akümülatörü temizle	(A) \leftarrow 0	-	1	1
CLR	C	Elde bitini temizle (sıfırla)	(C) \leftarrow 0	C	1	1
CLR	bit	Biti temizle	(bit) \leftarrow 0	-	2	1
CPL	A	Akümülatörü tersle	(A) \leftarrow / (A)	-	1	1
CPL	C	Elde bitini tersle	(C) \leftarrow / (C)	C	1	1
CPL	bit	Biti tersle	(bit) \leftarrow / (bit)	-	2	1

Komut	Kullanımı	Açıklama	Sembolik Gösterimi	E.B.	Byte	WC
SETB	SETB C SETB bit	Elde bitini birle (C=1) Bit değerini birle	(C) $\leftarrow 1$ (bit) $\leftarrow 1$	C	1	1
RL	A	Akümülatörü 1 bit sola döndür	$(A_{n+1}) \leftarrow (A_n)$, $n = 0 - 6$ $(A_0) \leftarrow (A_n)$	-	2	1
RLC	A	Akümülatörü elde üzerinden 1 bit sola döndür	$(A_{n+1}) \leftarrow (A_n)$, $n = 0 - 6$ $(A_0) \leftarrow (C)$ (C) $\leftarrow (A_7)$	C	1	1
RR	RR	Akümülatörü 1 bit sağa döndür	$(A_n) \leftarrow (A_{n+1})$, $n = 0 - 6$ $(A_7) \leftarrow (A_0)$	-	1	1
RRC	A	Akümülatörü elde üzerinden 1 bit sağa döndür	$(A_n) \leftarrow (A_{n+1})$, $n = 0 - 6$ $(A_7) \leftarrow (C)$ (C) $\leftarrow (A_0)$	C	1	1
ACALL	addr11	Adres11 etiketli alt programı çağır	(PC) $\leftarrow (PC) + 2$ (SP) $\leftarrow (SP) + 1$ (SP) $\leftarrow (PC_{7-0})$	2	2	2
ACALL	LCALL		(SP) $\leftarrow (SP) + 1$ (SP) $\leftarrow (PC_{15-8})$ (PC ₁₀₋₀) \leftarrow adres sayfası	-		
LCALL	addr16	Adres 16 etiketli alt programı çağır	(PC) $\leftarrow (PC) + 3$ (SP) $\leftarrow (SP) + 1$ ((SP)) $\leftarrow (PC_{7-0})$ (SP) $\leftarrow (SP) + 1$ ((SP)) $\leftarrow (PC_{15-8})$ (PC ₁₀₋₀) \leftarrow adres sayfası	-	3	2
RET		Alt programdan çıkış, ana programda kaldığıн yere dön	(PC ₁₅₋₈) $\leftarrow ((SP))$ (SP) $\leftarrow (SP) - 1$ (PC ₇₋₀) $\leftarrow ((SP))$ (SP) $\leftarrow (SP) - 1$	-	1	2
RETI		Kesme alt programından çıkış, ana programda kaldığın yere dön	((PC ₁₅₋₈) $\leftarrow ((SP))$ (SP) $\leftarrow (SP) - 1$ (PC ₇₋₀) $\leftarrow ((SP))$ (SP) $\leftarrow (SP) - 1$	-	1	2

Komut	Komut Kullanımı	Açıklama	Sembolik Gösterim	E.B.	Byte	M.C.
KOSULLUZ DALANNA KOMUTLARI	SIMP <adres>	Kısa dallanma (adrese dallan) “+128 byte,-128 byte”lik alanda dallanma yapılabılır	(PC) \leftarrow (PC) + 2 (PC) \leftarrow (PC) + <adres>	-	2	2
	AJMP <adres>	11 bitlik adres alanı içerisinde ($\pm 2KB$) dallanma sağlar	(PC) \leftarrow (PC) + 2 (PC _{16b}) \leftarrow <adres>	-	2	2
	JMP <adres>	16 bitlik adres alanı (program hafızanın tamamı) içerisinde dallanma sağlar	(PC) \leftarrow <adres> ₁₅₋₀	-	3	2
	JMP @A+DPTR	A+DPTR’ının gösterdiği adreste dallan	(PC) \leftarrow (A) + (DPTR)	-	1	2
	JC <adres>	Eğer C=1 ise adreste dallan	(PC) \leftarrow (PC) + 2 IF (C) = 1 THEN (PC) \leftarrow (PC) + <adres>	-	2	2
	JC <adres>	Eğer C=0 ise adreste dallan	(PC) \leftarrow (PC) + 2 IF (C) = 0 THEN (PC) \leftarrow (PC) + <adres>	-	2	2
	JNC <adres>	Eğer C=0 ise adreste dallan	(PC) \leftarrow (PC) + 2 IF (C) = 1 THEN (PC) \leftarrow (PC) + <adres>	-	2	2
	JB bit, <adres>	Eğer bit=1 ise adreste dallan	(PC) \leftarrow (PC) + 3 IF (bit) = 1 THEN (PC) \leftarrow (PC) + <adres>	-	3	2
	JNB bit, <adres>	Eğer bit=0 ise adreste dallan	(PC) \leftarrow (PC) + 3 IF (bit) = 0 THEN (PC) \leftarrow (PC) + <adres>	-	3	2
	JBC bit, <adres>	Eğer bit=1 ise adreste dallan sonra biti sıfırla (bit=0)	(PC) \leftarrow (PC) + 3 IF (bit) = 1 THEN (bit) \leftarrow 0 (PC) \leftarrow (PC) + <adres>	-	3	2
JZ <adres>	Eğer akümülatör sıfır (A=0) ise adreste dallan	(PC) \leftarrow (PC) + 2 IF A = 0 THEN (PC) \leftarrow (PC) + <adres>	-	2	2	
JNZ <adres>	Eğer A=0 değil ise adreste dallan	(PC) \leftarrow (PC) + 2 IF A \neq 0 THEN (PC) \leftarrow (PC) + <adres>	-	2	2	

KOSULLU DALANNA KOMUTLARI

Kodu	Komut Kullanımı	Açıklama	Sembolik Gösterim	E.B.	Byte	M.C.
DJNZ	Rn, <adres>	Rn'ı bir azalt ve Rn sıfır değil ise adres'e döndür	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF ((Rn) > 0 V (Rn) < 0) THEN (PC) ← (PC) + <adres>	-	2	2
DJNZ	direct, <adres>	Direct adresin değerini bir azalt, sıfır değil ise belirtilen adres'e döndür	(PC) ← (PC) + 2 (direct) ← (direct) - 1 IF ((direct1) > 0 V (direct1) < 0) THEN (PC) ← (PC) + <adres>	-	3	2
CINE	A, direct, <adres>	Akümülatör ile direct adres'deki değeri karşılaştır, eşit değilse belirtilen adres'e döndür	(PC) ← (PC) + 3 IF (A) < (direct) THEN (PC) ← (PC) + <adres> IF (A) < (direct) THEN (C) ← 1 ELSE (C) ← 0	C	3	2
CINE	A, #data, <adres>	Akümülatör ve sabit değerleri karşılaştır, eşit değil ise direct adres'e döndür	(PC) ← (PC) + 3 IF (A) < > data THEN (PC) ← (PC) + <adres> IF (A) < data THEN (C) ← 1 ELSE (C) ← 0	C	3	2
CINE	Rn, #data, <adres>	Rn ile değerleri karşılaştır, eşit değil ise direct adres'e döndür	(PC) ← (PC) + 3 IF (Rn) < > data THEN (PC) ← (PC) + <adres> IF (Rn) < data THEN (C) ← 1 ELSE (C) ← 0	C	3	2
CINE	@Ri, #data, <adres>	Ri'nin gösterdiği adressteki değer ile sabit değeri karşılaştır, eşit değil ise direct adres'e döndür	(PC) ← (PC) + 3 IF ((Ri)) < > data THEN (PC) ← (PC) + <adres> IF ((Ri)) < data THEN (C) ← 1 ELSE (C) ← 0	C	3	2

Komut	Komut Kullanımı	Açıklama	Sembolik Gösterim	E.B.	Byte	M.C.
INC A		Akümülatörün değerini 1 arttır	(A) \leftarrow (A) + 1	-	1	1
INC Rn		Saklayıcının değerini 1 arttır	(Rn) \leftarrow (Rn) + 1	-	1	1
INC direct		Adresteki değeri 1 arttır	(direct) \leftarrow (direct) + 1	-	2	1
INC @Ri		Ri saklayıcısının gösterdiği adresdeki değeri 1 artır	((Ri)) \leftarrow ((Ri)) + 1	-	1	1
INC DPTR		DPTR saklayıcısının değerini 1 artır	(DPTR) \leftarrow (DPTR) + 1	-	1	2
DEC A		Akümülatörün değerini 1 azalt	(A) \leftarrow (A) - 1	-	1	1
DEC Rn		Saklayıcının değerini 1 azalt	(Rn) \leftarrow (Rn) - 1	-	1	1
DEC direct		Adresteki değeri 1 azalt	(direct) \leftarrow (direct) - 1	-	2	1
DEC @Ri		Ri saklayıcısının gösterdiği adresdeki değeri 1 azalt	((Ri)) \leftarrow ((Ri)) - 1	-	1	1
ADD A,Rn		Rn saklayıcı değerini akümülatöre ekle	(A) \leftarrow (A) + (Rn)	C,OV,AC	1	1
ADD A,direct		Adresteki değeri akümülatöre ekle	(A) \leftarrow (A) + (direct)	C,OV,AC	2	1
ADD A,@Ri		Saklayıcının gösterdiği adresdeki değeri akümülatöre ekle	(A) \leftarrow (A) + ((Ri))	C,OV,AC	1	1
ADD A,#data		Sabit değeri akümülatöre ekle	(A) \leftarrow (A) + #data	C,OV,AC	2	1
ADDC A,Rn		Rn saklayıcı değerini ve elde bitini (C) akümülatöre ekle	(A) \leftarrow (A) + (C) + (Rn)	C,OV,AC	1	1
ADDC A,direct		Adresteki değeri ve elde bitini (C) akümülatöre ekle	(A) \leftarrow (A) + (C) + (direct)	C,OV,AC	2	1
ADDC A,@Ri		Saklayıcının gösterdiği adresdeki değeri ve elde bitini (C) akümülatöre ekle	(A) \leftarrow (A) + (C) + ((Ri))	C,OV,AC	1	1
ADDC A,#data		Sabit değeri ve elde bitini (C) akümülatöre ekle	(A) \leftarrow (A) + (C) + #data	C,OV,AC	2	1
SUBB A, Rn		Akümülatörden saklayıcının değerini ve borcu (C) çıkart	(A) \leftarrow (A) - (C) - (Rn)	C,OV,AC	1	1
SUBB A, direct		Akümülatörden adresdeki değeri ve borcu (C) çıkart	(A) \leftarrow (A) - (C) - (direct)	C,OV,AC	2	1
SUBB A, @Ri		Akümülatörden saklayıcının gösterdiği adresdeki değeri ve borcu (C) çıkart	(A) \leftarrow (A) - (C) - (Ri)	C,OV,AC	1	1
SUBB A, #data		Akümülatörden sabit değeri ve borcu (C) çıkart	(A) \leftarrow (A) - (C) - (#data)	C,OV,AC	2	1
MUL AB		Akümülatördeki değeri ile B saklayıcısındaki değeri çarp. Çarpım sonucunun yüksek anlamlı byte'ını B saklayıcısına, düşük anlamlı byte'ni ise akümülatöre yaz.	(A) ₇₋₀ \leftarrow (A) x (B) (B) ₁₅₋₈	C,OV	1	4
DIV AB		Akümülatördeki değeri, B saklayıcısındaki değere bölg. İşlem sonucunda bölüm değerini akümülatöre, kalan değerini B saklayıcısına yükle.	(A) ₁₅₋₈ \leftarrow (A)/(B) (B) ₇₋₀	C,OV	1	4
DA DA A		Akümülatörü onluk tabana ayarla	IF [[(A ₃₋₀) > 9] [(AC) = 1]] THEN(A ₃₋₀) \leftarrow (A ₃₋₀) + 6 AND IF [[(A ₇₋₄) > 9] [(C) = 1]] THEN(A ₇₋₄) \leftarrow (A ₇₋₄) + 6	C	1	1

Oku-Değiştir-Yaz (Read-Modify-Write) Komutları

8051 komut kümesinde I/O portlarından (P0,P1,P2,P3) veri okuyan komutlardan bazıları (kullanımlarına bağlı olarak) porta ait tutucuya (latch) bazaları ise portun fizikselliğindeki değeri okur. Tutucuya okuyan komutlar ve tutucunun okunduğu örneklere aşağıda verilmiştir.

Komut	Açıklama
ANL	Lojik "VE" işlemi ANL P1, A
ORL	Lojik "VEYA" işlemi ORL P2, #55h
XRL	Lojik "ÖZEL-VEYA" işlemi XRL P3, #0AAh
JBC	Eğer bit=1 ise adresе dالan sonra biti sıfırla JBC P2.0, devam
CPL	Biti tersle CPL P3.2
INC	INC P2
DEC	DEC P3
DINZ	DINZ P2, git
MOV Px,Y, C	MOV P2.3, C
CLR Px,Y	CLR P3.6
SETB Px,Y	SETB P0.3

*Bu komutların içrasında ilk önce portun tutucuları (8-Bit'in tamamı) okunur, ilgili bitin değeri değiştirilir ve ardından oluşan yeni Byte değeri tutuculara yazılır.

Tabloda verilen komutların kullanımında eğer **hedef operand** herhangi bir port veya portun koşturusmasında ilk önce tutucu değeri okunur, okunan veri üzerinde işlem yapılır ve oluşan yeni değer tekrar tutucuya yazılır. Bu işlemlerin hepsi donanım tarafından otomatik olarak yerine getirilir. Eğer port veya portun bir biti **kaynak operand** ise bu durumda komutun koşturusmasında portun fizikselliğindeki değeri okunur.

Örneğin ANL komutunun "**ANL P1, A**" şeklindeki kullanımında P1 portu **hedef operand'dır**. Bu durumda komut koşturusurken donanım P1 portuna ait tutucuya okur, akümülatördeki değer ile lojik ve işleminin tatabi tutar ve ardından lojik ve işleminin sonucunu P1 tutucularına yazar.

Fakat, ANL komutunun "**ANL A, P1**" şeklindeki kullanımında ise P1 portu **kaynak operand'dır**. Bu durumda komut koşturusurken donanım P1 portuna fizikselliğindeki değeri ile lojik ve işleme tatabi tutar ve ardından lojik ve işleminin sonucu akümülatöre yazılır.

Deney foyü ile gördüğünüz hataları sayfa
numarası ve kısa bir açıklama ile
aozdemir@sakarya.edu.tr veya
barakli@sakarya.edu.tr adresine mail atmanızı rica
ederiz.



L

Flash

4 Byte. Also ; 1024 say for

$$4^{*}1024 = 4\text{KByte}$$

Nov EARTH, #02D

Nov EADBZL, #03D

3 Sayfa sección

HATA:

MOV ECON, #05 ; segs sysd strw

MOV EDATA1, #H'65

EDATA 2, #H'2B } Python def file
MOV

MOV EDATA3, #H'3E

MOV EDATA4, #H'7F

Nov ECON, #102 → Beyerle Seifert d'and.

Mon May ECON, #04 → Brenda has to sign?

116X A, Econ
TWE

JNZ ~~ECHO HATA~~

NOV EARTH, HOLD

MOV EAXL, #53D

MOV ECON, #01 → se col se fa unde

MOY A, EDATA1

MOY B, E DATA 2

ML AB

H	P	S	G	P	C	G1	Pz
31			1	2	3		
32	4	5	6	7	8	9	10
33	11	12	13	14	15	16	17
34	18	19	20	21	22	23	24
35	25	26	27	28	29	30	31

AUGUSTOS 2014

2



SMS

$$\frac{5000\ 000}{39,42} = 55309$$

$$65535 - 55323 = 1022$$

setb	en	A
dr	en	B
setb	m	1
cler	m	2

—	Acall	bchle
	setb	enB
	cler	enA
	sets	M3
	cler	M4
—	acall	bchle
	Sebb	enA
	cler	enB
	setb	M2
	cler	M1
—	acall	bchle
	sets	enB
	cler	enA
	sets	M4
	cler	M3
—	acall	bchle

Belle	JNB	JFΦ, Belle
Cler	TFΦ	
Mov	ILΦ, dPL	
Max	THΦ, sph	
Ret		

AUGUST | SUNDAY

41

AUGUST | SATURDAY
AGUSTOS | CUMARTESİ

91

35	25	26	27	28	29	30	31
34	18	19	20	21	22	23	24
33	11	12	13	14	15	16	17
32	4	5	6	7	8	9	10
31							

AdusTos 2014



$a = \text{Sbt}$
 $b = \text{Sbt}$
 $d \rightarrow a$

$2^2 \times 2^{(2-1)}$
 $\times 2^{(2-1)}$

GPIOPinType GPIOInput (GPIO_PORTF_Base, GPIO_PIN_4)

GPIO Pad Config Set (GPIO_PORTF_Base, GPIO_Pin_4, ...)

//pm4 kann Pull UP für aktifieren

- 40023000 → Base Address
- Hierpm verringt Pin 4 an address vor
- 400253FC → data → istm vermisst

$$4 \times 2^b \quad b = \text{bit (signen)}.$$

$$\begin{array}{l} 0 \rightarrow 4 \\ 1 \rightarrow 8 \\ 2 \rightarrow 10 \\ 3 \rightarrow 20 \end{array}$$

(40025000)

Base address + flagbündigen Sabr 1

+ " Sabr 2
+ " a 3

$$0x40004000 + 4 = 1$$

$$0x40004000 + 8 = 2$$

$$0x40004000 + 20 = 5$$

76543210
00000000

101

Opn 21d

$$0x40004000 + 4 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + \dots + 4 \times 2^{20}$$

$$0x40004000 + 3FC$$

$$0x400043FC \quad \text{Tom pm4}$$

NISAN 2014	P2	C5	G5	P5	S5	H5
14	1	2	3	4	5	6
15	7	8	9	10	11	12
16	14	15	16	17	18	19
17	21	22	23	24	25	26
18	28	29	30			
19						
20						
21						
22						
23						
24						
25						
26						
27						

APRIL THURSDAY
NISAN PERŞEMBE

100
E0

HWRREG(0x0025400)-0x02; Work
Whl(1)



6x2¹; 1.pm

1

HWRREG(0x0025000+8)=2; Msadegh
SysCtlDelay(20000000);

HWRREG(0x0025000+8)=0;
SysCtlDelay(20000000);

HWRREG() == GPIOPINWrite

Int osd(Int x, Int y)

Int r;

Int sonuc=1;

for (r=1; i<y+1; r++)

sonuc=sonuc*x;

}

3

```
#include <"inc/hw_gpio.h"\n#include "
```

F1WREG(GPIO_PORTF_BASE + GPIO_O_Lock
= GPIO_LockKey;

NISAN ÇARSIMBA APRIL WENDESDAY

02

H	P	S	G	P	C	Ct	Pz
14	1	2	3	4	5	6	
15	7	8	9	10	11	12	13
16	14	15	16	17	18	19	20
17	21	22	23	24	25	26	27
18	28	29	30				

NISAN 2014

```

#include <aduc841.h>           #include <ADUC841.H>          mov r0,a
VLOW EQU 40H                   org 00h                         mov a,dpl
VHIGH EQU 41H                  sjmp basla                 addc a,b
ORG 0000H                      basla:                     mov dpl,a
SJMP AYARLAR                   mov b, #00h                clr a
ORG 002BH                      clr a                     addc a, #00h
SJMP T2KESME                    clr c                     push acc
AYARLAR:                        mov daccon, #00101101b    mov a,r3
MOV DACCON,#00101101B           //12bit,;,dac0harici,;,dac0cikisidac0h/l,dacilay    mov b,r1
MOV T2CON,#00000000B            1Xsayi2=1712.5          mul ab
MOV DPTR,#54473D ;1 MS         //sayi1=2500d,sayi2=0.685d=>sayi    add a,dpl
MOV TL2,#DPL                   //sayi2=0.685X65535=44891.475    mov r2,a
MOV TH2,#DPH                   //sayi1=2500d=09C4h//r0,r1 olsun    pop acc
MOV RCAP2L,#DPL                //sayi2=44891=AF5Bh//r2,r3 olsun    addc a,b
MOV RCAP2H,#DPH                mov r0, #0c4h             mov r3,a
                                         mov r1, #09h             mov r1, 00h
                                         mov r2, #5bh             pop 00h
                                         mov r3, #0afh            pop dpl
                                         push b                 pop b
                                         push dpl               mov dac0h, r3
SETB EADC                      mov a,r0                 mov dac0l, r2
SETB EA                         mov b, r2               end
SETB ET2                        mul ab
SETB TR2                        push acc
SONSUZ:                         push b                 PWM #include<Aduc841.h>
SJMP SONSUZ                     push b                 KATSAYI_0L EQU 20H
T2KESME:                        mov a, r0               KATSAYI_0H EQU 21H
CLR TF2                         mov b,r3               KATSAYI_1L EQU 22H
CLR C                           mul ab
MOV A,43H                        pop 00h               KATSAYI_1H EQU 23H
ADD A,VLOW                       add a,r0               CSEG
MOV VLOW,A                        mov r0,a
MOV A,44H                        clr a
ADDC A,VHIGH                      addc a,b
MOV VHIGH,A                      mov dpl,a
MOV DAC0H,VHIGH                  mov a, r2
MOV DAC0L,VLOW                   mov b, r1
RETI                            mul ab
END                             add a,r0
                                mov KATSAYI_1L, #00H
                                mov KATSAYI_1H, #0D8H
                                ACALL PWM_CIKIS_VER

```

ASSDF:JNB P3.2,ASSDF	mov tl0, #20H	BASLA:		
SERS:JB P3.2,SERS	mov th0, #0ffH	CLR	P2.3	
MOV PWM0L,#00H	setb et0	CLR	LED3	
MOV PWM1L, KATSAYI_1L	setb ea	MOV	R0,#07FH	
MOV PWM0H,#06CH	setb tr0	TMZ:MOV @R0,#00H		
MOV PWM1H, KATSAYI_1H	MOV DPTR,#DTMF_1_VERI	DJNZ	R0,TMZ	
ASSDF1:JNB P3.2,ASSDF1	MOV A,#00H	SETB	I TO	
SERS2:JB P3.2,SERS2	MOVC A,@A+DPTR	SETB	EXO	
MOV PWM0L,#00H	MOV DTMF_1VERI,A	SETB	EA	
MOV PWM1L, KATSAYI_1L	setb tr0	MOV	TMOD,#01H ;TIMERO 16	
MOV PWM0H,#0A2H	DUR:sjmp DUR	DONGU:		
MOV PWM1H, KATSAYI_1H	A1_DTMF_1:	MOV	R0,#169D ;1SN	
bekle:	MOV DPTR,#DTMF_1	SETB	EA	
SJMP bekle	MOV A,DTMF1	JB	BTNS,\$	
PWM_CIKIS_VER:	MOVC A,@A+DPTR	D:	JNB	BTNS,\$
MOV CFG841, #00H	MOV SIN_Y3HB,A	JZ	D	
MOV PWM0L, KATSAYI_0L	INC DTMF1	MOV	R1,A	
MOV PWM1L, KATSAYI_1L	MOV A,DTMF1	MOV	A,#00H	
MOV PWM0H, KATSAYI_0H	MOVC A,@A+DPTR	CLR	EA	
MOV PWM1H, KATSAYI_1H	MOV SIN_Y3LB,A	X:	SETB	LED3
RET	INC DTMF1	ACALL	BEKLE	
END	MOV A,DTMF1	CLR	LED3	
#include <aduc841.h>	CLR C	ACALL	BEKLE	
DTMF1 EQU 20H	SUBB A,DTMF_1VERI	DJNZ	R1,X	
DTMF_1VERI EQU 22H	JC AD_DTMF1	MOV	R1,#00H	
SIN_Y3HB EQU 24H	MOV DTMF1,#00H	SJMP	DONGU	
SIN_Y3LB EQU 25H	AD_DTMF1:	BEKLE:		
org 00h	mov tl0, #20H	MOV	R0,#169D	
sjmp basla	mov th0, #0ffH	SETB	TRO	
org 000bh	RETI	Y:	JNB TFO,\$	
sjmp A1_DTMF_1	#include "aduc841.h"	CLR	TFO	
basla:	LED3 EQU P0.2	MOV	TLO,#00H	
mov A, #00h	BTN4 EQU P3.2	MOV	TH0,#00H	
mov DACCON, #00101101b //dac 12 bit, harici ref,dac0 aktif, veri Li ye yazildiginda güncellenir	BTNS EQU P3.3	DJNZ	R0,Y	
mov tmod, #00000001 //t0 aktif, mod1	ORG 00H	CLR	TRO	
//65314d=ff22h yuklendi //1000000usn=11059200 kre dalgaysa 20usn=221 ve 65535-221=65314	SJMP BASLA	RET	HARICI:	
	ORG 03H	ACALL	GECIKME ;BTNS ARK	
	SJMP HARICI	JNB	BTN4,\$	

INC A

RETI

GECIKME:

```
SETB PSW.3 ;BANK 1  
MOV R4,#10D  
MOV R5,#0FFH  
MOV R6,#0FFH
```

W:

```
DJNZ R5,W  
MOV R5,#0FFH  
DJNZ R6,W  
MOV R6,#0FFH  
DJNZ R4,W  
RET
```

END

V

```
WRITE EQU 02H  
04H VERIFY EQU  
05H WRITE EQU
```

```
ORG 000H  
SJMP MAIN
```

MAIN:

```
MOV EADRH,#0  
MOV EADRL,#3  
MOV ECON,#ERASE  
MOV EDATA1,#0AH  
MOV ECON,#WRITE
```

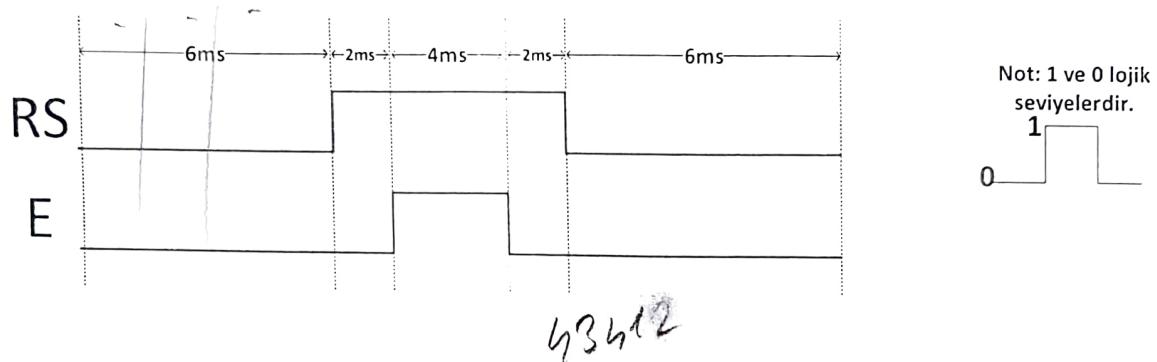
```
MOV ECON,#VERIFY
```

```
MOV A,ECON  
JNZ MAIN  
END
```

Mikroişlemciler ve Sayısal İşaret İşleme Laboratuvarı Ara Sınav

22/11/2018

S.1) Aşağıda, bir cihaza ait kontrol pinlerinin veri yazma zaman diyagramı verilmiştir. P0.1 pini RS ve P0.2 pini E olmak üzere asm kodunu yazınız. Verilen çalışma/gecikme süreleri timer0 yoklama yöntemi ile yapılmalıdır.



S.2) Aduc841 mikrokontrolörün mimarisinde bulunan PWM modülü kullanılarak **%40** doluluğa sahip **500 Hz**'lik PWM darbeleri PWM0-P2.7 çıkışları ile üretilicektir. Gerekli PWM çalışma modunu belirtiniz ve asm kodunu yazınız. (saat giriş bölücüsü (SGB) = 1, $f_{saat} = f_{osc}$ seçiniz.)

$$8, 10^{-4}$$

52428

S.3) ADUC841 temelli mikrokontrolör analog – sayısal dönüştürücüsünün **ADC0** kanalına bağlı olan analog işaret **T=4 ms** periyodu ile örneklenenek ve okunan değer doğrudan **DAC0**'dan çıkış olarak verilecektir. Gerekli asm kodunu yazınız.

ADC çalışma ayarları; harici referans kaynağı, MCLK=4, T/H için 4 ADC periyodu, DAC0, 12-bit, harici refarans modunda kullanılacaktır. (Mikrokontrolör frekansı: 11.0592 MHz)(timer yoklamalı yada kesme yapısı ile kullanılabilir.)

4212370

Süre **60** dakika. **Başarılar...**

Prof.Dr.Ayhan Özdemir

Dr. Öğr. Üyesi Burhan Baraklı

Dr. Öğr. Üyesi Şuayb Çağrı Yener

5 mil sonne
lcall gerakme → lösre ff olrsa 5,9 msn

gerakme:

mov R3, #4FH
w2: mov Rh, #FFH
w1: djnz Rh, w1
djnz R3, w2
ret